数据仓库服务

开发指南

文档版本 01

发布日期 2025-12-12

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2025。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 使用前必读 1
2 DWS 数据迁移 4
2.1 迁移数据到 DWS 4
2.2 数据类型映射关系 6
2.2.1 DWS 外表与 HIVE/SPARK、ORC 及 PARQUET 数据类型的映射关系 6
2.2.2 MySQL 迁移到 DWS 的数据类型映射 10
2.2.3 Oracle 迁移到 DWS 的数据类型映射 12
2.2.4 Teradata 迁移到 DWS 的数据类型映射 13
2.3 导入数据 14
2.3.1 从 OBS 并行导入数据 14
2.3.1.1 关于 OBS 并行导入 14
2.3.1.2 从 OBS 导入 CSV、TXT 数据 19
2.3.1.2.1 创建访问密钥（AK 和 SK） 19
2.3.1.2.2 上传数据到 OBS 20
2.3.1.2.3 创建 OBS 外表 22
2.3.1.2.4 执行导入数据 25
2.3.1.2.5 处理导入错误 26
2.3.1.3 从 OBS 导入 ORC、CARBONDATA、PARQUET 和 JSON 数据 29
2.3.1.3.1 OBS 上的数据准备 29
2.3.1.3.2 创建外部服务器 30
2.3.1.3.3 创建外表 34
2.3.1.3.4 通过外表查询 OBS 上的数据 36
2.3.1.3.5 清除资源 37
2.3.1.3.6 支持的数据类型 39
2.3.2 使用 GDS 从远端服务器导入数据 42
2.3.2.1 关于 GDS 并行导入 42
2.3.2.2 准备源数据 46
2.3.2.3 安装配置和启动 GDS 46
2.3.2.4 创建 GDS 外表 50
2.3.2.5 执行导入数据 53
2.3.2.6 处理导入错误 54
2.3.2.7 停止 GDS 57
2.3.2.8 GDS 导入示例 58

 版权所有 © 华为云计算技术有限公司

2.3.3 从 MRS 导入数据到集群 64
2.3.3.1 从 MRS 导入数据概述 64
2.3.3.2 MRS 集群上的数据准备 64
2.3.3.3 手动创建外部服务器 68
2.3.3.4 创建外表 71
2.3.3.5 执行数据导入 75
2.3.3.6 清除资源 76
2.3.4 基于 GDS 的跨集群互联互通 78
2.3.5 使用 gsql 元命令\COPY 导入数据 81
2.3.6 使用 COPY FROM STDIN 导入数据 84
2.3.6.1 关于 COPY FROM STDIN 导入数据 84
2.3.6.2 CopyManager 类简介 85
2.4 整库迁移 86
2.4.1 使用 CDM 迁移数据到 DWS 86
2.4.2 使用 DSC 工具迁移 SQL 脚本 86
2.5 元数据迁移 87
2.5.1 使用 gs_dump 和 gs_dumpall 命令导出元数据 87
2.5.1.1 概述 87
2.5.1.2 导出单个数据库 89
2.5.1.2.1 导出数据库 89
2.5.1.2.2 导出模式 92
2.5.1.2.3 导出表 94
2.5.1.3 导出所有数据库 97
2.5.1.3.1 导出所有数据库 97
2.5.1.3.2 导出全局对象 99
2.5.1.4 无权限角色导出数据 101
2.5.2 使用 gs_restore 导入数据 104
2.6 导出数据 108
2.6.1 导出数据到 OBS 108
2.6.1.1 关于 OBS 并行导出 108
2.6.1.2 导出 CSV、TXT 数据到 OBS 113
2.6.1.2.1 规划导出数据 113
2.6.1.2.2 创建 OBS 外表 114
2.6.1.2.3 执行导出 117
2.6.1.2.4 示例 117
2.6.1.3 导出 ORC 和 PARQUET 数据到 OBS 121
2.6.1.3.1 规划导出数据 121
2.6.1.3.2 创建外部服务器 122
2.6.1.3.3 创建外表 123
2.6.1.3.4 执行导出 125
2.6.2 导出 ORC 和 PARQUET 数据到 MRS 125
2.6.2.1 导出 ORC 和 PARQUET 数据概述 125

 版权所有 © 华为云计算技术有限公司

2.6.2.2 规划导出数据 126
2.6.2.3 创建外部服务器 127
2.6.2.4 创建外表 131
2.6.2.5 执行导出 132
2.6.3 使用 GDS 导出数据到远端服务器 133
2.6.3.1 关于 GDS 并行导出 133
2.6.3.2 规划导出数据 136
2.6.3.3 安装配置和启动 GDS 136
2.6.3.4 创建 GDS 外表 139
2.6.3.5 执行导出数据 140
2.6.3.6 停止 GDS 141
2.6.3.7 GDS 导出示例 141
2.7 其他操作 146
2.7.1 GDS 管道文件常见问题 146
2.7.2 查看数据倾斜状态 147
2.7.3 分析表 150
3 DWS 开发设计建议 152
3.1 DWS 总体开发设计规范 152
3.2 DWS 连接管理规范 155
3.3 DWS 对象设计规范 156
3.3.1 DATABASE 对象设计规范 156
3.3.2 USER 对象设计规范 157
3.3.3 SCHEMA 对象设计规范 158
3.3.4 TABLESPACE 对象设计规范 158
3.3.5 TABLE 对象设计规范（重点） 158
3.3.6 INDEX 对象设计规范（重点） 162
3.3.7 VIEW 对象设计规范 162
3.4 DWS SQL 开发规范 162
3.4.1 DDL 操作规范 163
3.4.2 INSERT 操作规范 163
3.4.3 UPDATE&DELETE 操作规范 164
3.4.4 SELECT 操作规范 164
3.5 DWS 外表功能开发规范 167
3.6 DWS 存储过程开发规范 168
3.7 DWS 各对象设计详细规则 169
3.7.1 DWS 数据库对象命名规则 169
3.7.2 DWS 数据库对象设计规则 170
3.7.2.1 DWS Database 和 Schema 设计规则 170
3.7.2.2 DWS 表设计规则 171
3.7.2.3 DWS 字段设计规则 173
3.7.2.4 DWS 约束设计规则 174
3.7.2.5 DWS 视图和关联表设计规则 175

 版权所有 © 华为云计算技术有限公司

3.7.3 DWS JDBC 配置规则 175
3.7.4 DWS SQL 编写规则 176
3.7.5 自定义 DWS 外部函数(pgSQL/Java)使用规则 179
3.7.6 DWS PL/pgSQL 使用规则 180
4 创建和管理 DWS 数据库对象 183
4.1 创建和管理 DWS 数据库 183
4.2 创建和管理 DWS Schema 184
4.3 创建和管理 DWS 表 187
4.4 选择 DWS 表存储模型 191
4.5 创建和管理 DWS 分区表 194
4.6 创建和管理 DWS 索引 199
4.7 创建和使用 DWS 序列 202
4.8 创建和管理 DWS 视图 204
4.9 创建和管理 DWS 定时任务 205
4.10 查看 DWS 系统表 207
5 Oracle、Teradata 和 MySQL 语法兼容性差异 210
6 DWS 数据库安全管理 232
6.1 DWS 用户及权限管理 232
6.1.1 DWS 数据库用户类型 232
6.1.2 DWS 数据库用户管理 233
6.1.3 自定义 DWS 密码策略 234
6.1.4 DWS 数据库权限管理 239
6.1.5 DWS 三权分立 242
6.2 DWS 敏感数据管理 244
6.2.1 DWS 行级访问控制 244
6.2.2 DWS 数据脱敏 245
6.2.3 DWS 字符串加解密 249
6.2.4 使用 pgcrypto 加密 DWS 数据 251
7 查询 DWS 数据 261
7.1 单表查询 261
7.2 多表连接查询 262
7.3 子查询表达式 267
7.4 WITH 表达式 271
7.5 UNION 操作符的使用 277
7.6 SQL on Hudi 280
7.6.1 Hudi 简介 280
7.6.2 使用 Hudi 前准备 281
7.6.3 Hudi 用户接口 282
7.6.4 创建 Hudi 数据（外表） 283
7.6.5 Hudi 任务同步 284
7.6.6 Hudi 外表查询 286

 版权所有 © 华为云计算技术有限公司

7.6.7 支持访问 MRS 上的 Hudi 表 287
8 DWS 排序规则 289
9 DWS 物化视图 292
9.1 物化视图概述 292
9.2 异步物化视图 295
9.2.1 物化视图查询重写 295
9.2.1.1 概述 295
9.2.1.2 文本匹配查询重写 295
9.2.2 分区物化视图 297
9.2.2.1 分区物化视图概述 297
9.2.2.2 创建分区物化视图 297
9.2.2.3 刷新分区物化视图 300
9.2.3 嵌套物化视图 301
9.2.3.1 嵌套物化视图概述 301
9.2.3.2 创建嵌套物化视图 301
9.2.3.3 刷新嵌套物化视图 303
10 DWS 用户自定义函数 306
10.1 DWS PL/Java 语言函数 306
10.2 DWS PL/pgSQL 语言函数 316
11 DWS 存储过程 317
11.1 DWS 存储过程简介 317
11.2 DWS 存储过程数据类型转换 317
11.3 DWS 存储过程数组和 record 318
11.3.1 数组 318
11.3.2 record 324
11.4 DWS 存储过程声明语法 327
11.5 DWS 存储过程基本语句 329
11.6 DWS 存储过程动态语句 332
11.6.1 执行动态查询语句 332
11.6.2 执行动态非查询语句 334
11.6.3 动态调用存储过程 335
11.6.4 动态调用匿名块 337
11.7 DWS 存储过程控制语句 338
11.7.1 返回语句 338
11.7.2 条件语句 340
11.7.3 循环语句 342
11.7.4 分支语句 345
11.7.5 空语句 347
11.7.6 错误捕获语句 347
11.7.7 GOTO 语句 349
11.8 DWS 存储过程其他语句 350

 版权所有 © 华为云计算技术有限公司

11.9 DWS 存储过程游标 351
11.9.1 游标概述 351
11.9.2 显式游标 351
11.9.3 隐式游标 355
11.9.4 游标循环 356
11.10 DWS 存储过程高级包 357
11.10.1 DBMS_LOB 357
11.10.2 DBMS_RANDOM 365
11.10.3 DBMS_OUTPUT 367
11.10.4 UTL_RAW 368
11.10.5 DBMS_JOB 370
11.10.6 DBMS_SQL 378
11.11 DWS 存储过程调试 387
12 使用 PostGIS Extension 391
12.1 PostGIS 概述 391
12.2 PostGIS 使用 392
12.3 PostGIS 支持和限制 392
12.4 OPEN SOURCE SOFTWARE NOTICE (For PostGIS) 405
13 使用 JDBC 或 ODBC 进行 DWS 二次开发 453
13.1 开发前准备 453
13.2 基于 JDBC 开发 453
13.2.1 JDBC 开发流程 453
13.2.2 JDBC 包与驱动类 454
13.2.3 加载驱动 455
13.2.4 连接数据库 455
13.2.5 执行 SQL 语句 458
13.2.6 处理结果集 460
13.2.7 常用 JDBC 开发示例 462
13.2.8 应用端加工 RoaringBitmap 结果集并入库 DWS 开发示例 472
13.2.9 JDBC 接口参考 474
13.3 基于 ODBC 开发 487
13.3.1 ODBC 包及依赖的库和头文件 489
13.3.2 Linux 下配置数据源 489
13.3.3 Windows 下配置数据源 495
13.3.4 ODBC 开发示例 499
13.3.5 ODBC 接口参考 504
14 DWS 冷热数据管理 522
15 DWS 资源监控 526
15.1 用户资源监控 526
15.2 资源池资源监控 528
15.3 内存资源监控 530

 版权所有 © 华为云计算技术有限公司

15.4 实例资源监控 530
15.5 实时 TopSQL 532
15.6 历史 TopSQL 536
15.7 TopSQL 查询示例 539
16 DWS 性能调优 542
16.1 性能调优概述 542
16.2 性能诊断 544
16.2.1 集群性能分析 544
16.2.2 慢 SQL 分析 544
16.2.2.1 查询最耗性能的 SQL 544
16.2.2.2 分析作业是否被阻塞 545
16.2.3 SQL 诊断 546
16.2.4 表诊断 547
16.3 系统调优 548
16.3.1 数据库系统参数调优 548
16.3.2 SMP 并行执行 554
16.3.3 配置 LLVM 557
16.4 SQL 调优 559
16.4.1 SQL 查询执行流程 559
16.4.2 SQL 执行计划 561
16.4.3 执行计划算子 572
16.4.4 SQL 调优流程 576
16.4.5 更新统计信息 576
16.4.6 审视和修改表定义 584
16.4.7 SQL 调优进阶 584
16.4.7.1 SQL 自诊断 584
16.4.7.2 语句下推调优 588
16.4.7.3 子查询调优 594
16.4.7.4 统计信息调优 602
16.4.7.5 算子级调优 606
16.4.7.6 数据倾斜调优 608
16.4.7.7 磁盘缓存主动预热调优 615
16.4.7.8 SQL 语句改写规则 615
16.4.8 优化器参数调整 617
16.4.9 使用 Plan Hint 进行调优 618
16.4.9.1 Plan Hint 调优概述 618
16.4.9.2 Join 顺序的 Hint 620
16.4.9.3 Join 方式的 Hint 623
16.4.9.4 行数的 Hint 624
16.4.9.5 Stream 方式的 Hint 625
16.4.9.6 Scan 方式的 Hint 628
16.4.9.7 子链接块名的 hint 629

 版权所有 © 华为云计算技术有限公司

16.4.9.8 运行倾斜的 hint 630
16.4.9.9 指定子查询不提升的 hint 634
16.4.9.10 Drive 方式的 Hint 636
16.4.9.11 字典编码的 hint 637
16.4.9.12 配置参数的 hint 638
16.4.9.13 Hint 的错误、冲突及告警 641
16.4.9.14 Plan Hint 实际调优案例 643
16.4.10 例行维护表 648
16.4.11 例行重建索引 649
16.4.12 SQL 语句出错自动重试 650
16.4.13 query_band 负载识别 653
16.5 SQL 调优案例 657
16.5.1 案例：选择合适的分布列 657
16.5.2 案例：建立合适的索引 658
16.5.3 案例：增加 JOIN 列非空条件 659
16.5.4 案例：典型不下推语句整改案例集 660
16.5.4.1 案例：使排序下推 660
16.5.4.2 案例：With-Recursive contains only values rte is not shippable 662
16.5.4.3 案例：With-Recursive contains system table is not shippable 665
16.5.4.4 案例：SubPlan exec on CN can't be shipped 666
16.5.4.5 案例：Type of Record in TargetList can not be shipped 668
16.5.4.6 案例：Type of Record in non-real table can not be shipped 669
16.5.4.7 案例：main plan exec on CN and SubPlan exec on DN can't be shipped 669
16.5.4.8 案例：With-Recursive under multi-nodegroup scenario is not shippable 670
16.5.4.9 案例：With-Recursive contains conflict distribution in none-recursive(xxx) recursive(xxx) 674
16.5.4.10 案例：Table in TargetList can not be shipped 676
16.5.4.11 案例："Row () Expression" can not be shipped 676
16.5.4.12 案例：Merge Into can't be shipped in the sence temporarily 678
16.5.4.13 案例：Type of Record in dual that is not a real table can not be shipped 679
16.5.5 案例：设置 cost_param 对查询性能优化 680
16.5.6 案例：调整局部聚簇键 683
16.5.7 案例：调整中间表存储方式 685
16.5.8 案例：改建分区表 686
16.5.9 案例：调整 GUC 参数 best_agg_plan 687
16.5.10 案例：改写 SQL 排除剪枝干扰 689
16.5.11 案例：改写 SQL 消除 in-clause 691
16.5.12 案例：使用 partial cluster key 692
16.5.13 案例：NOT IN 转 NOT EXISTS 694
17 实时数仓 696
17.1 实时数仓简介 696
17.2 支持与限制 699
17.3 实时数仓语法 700

 版权所有 © 华为云计算技术有限公司

17.3.1 CREATE TABLE 700
17.3.2 INSERT 707
17.3.3 DELETE 708
17.3.4 UPDATE 709
17.3.5 UPSERT 710
17.3.6 MERGE INTO 712
17.3.7 SELECT 713
17.3.8 ALTER TABLE 715
17.4 实时数仓函数 716
17.5 实时数仓 Binlog 723
17.5.1 订阅实时数仓 Binlog 723
17.5.2 Flink 实时消费 Binlog 727
18 DWS 系统表和系统视图 734
18.1 系统表和系统视图概述 734
18.2 系统表 734
18.2.1 GS_BLOCKLIST_QUERY 734
18.2.2 GS_BLOCKLIST_SQL 735
18.2.3 GS_OBSSCANINFO 736
18.2.4 GS_RESPOOL_RESOURCE_HISTORY 737
18.2.5 GS_WLM_INSTANCE_HISTORY 739
18.2.6 GS_WLM_OPERATOR_INFO 740
18.2.7 GS_WLM_SESSION_INFO 742
18.2.8 GS_WLM_USER_RESOURCE_HISTORY 748
18.2.9 PG_AGGREGATE 749
18.2.10 PG_AM 750
18.2.11 PG_AMOP 751
18.2.12 PG_AMPROC 752
18.2.13 PG_ATTRDEF 753
18.2.14 PG_ATTRIBUTE 753
18.2.15 PG_AUTHID 755
18.2.16 PG_AUTH_HISTORY 757
18.2.17 PG_AUTH_MEMBERS 757
18.2.18 PG_BLOCKLISTS 758
18.2.19 PG_CAST 759
18.2.20 PG_CLASS 759
18.2.21 PG_COLLATION 764
18.2.22 PG_CONSTRAINT 764
18.2.23 PG_CONVERSION 766
18.2.24 PG_DATABASE 767
18.2.25 PG_DB_ROLE_SETTING 768
18.2.26 PG_DEFAULT_ACL 768
18.2.27 PG_DEPEND 769

 版权所有 © 华为云计算技术有限公司

18.2.28 PG_DESCRIPTION 770
18.2.29 PG_ENUM 771
18.2.30 PG_EXCEPT_RULE 771
18.2.31 PG_EXTERNAL_NAMESPACE 772
18.2.32 PG_EXTENSION 772
18.2.33 PG_EXTENSION_DATA_SOURCE 773
18.2.34 PG_FINE_DR_INFO 773
18.2.35 PG_FOREIGN_DATA_WRAPPER 774
18.2.36 PG_FOREIGN_SERVER 775
18.2.37 PG_FOREIGN_TABLE 775
18.2.38 PG_INDEX 776
18.2.39 PG_INHERITS 777
18.2.40 PG_JOB_INFO 778
18.2.41 PG_JOBS 778
18.2.42 PG_LANGUAGE 779
18.2.43 PG_LARGEOBJECT 780
18.2.44 PG_LARGEOBJECT_METADATA 781
18.2.45 PG_MATVIEW 781
18.2.46 PG_MATVIEW_DEPEND 782
18.2.47 PG_MATVIEW_PARTITION 782
18.2.48 PG_NAMESPACE 783
18.2.49 PG_OBJECT 783
18.2.50 PG_OBSSCANINFO 784
18.2.51 PG_OPCLASS 785
18.2.52 PG_OPERATOR 785
18.2.53 PG_OPFAMILY 786
18.2.54 PG_PARTITION 787
18.2.55 PG_PLTEMPLATE 789
18.2.56 PG_PROC 790
18.2.57 PG_PUBLICATION 792
18.2.58 PG_PUBLICATION_NAMESPACE 793
18.2.59 PG_PUBLICATION_REL 794
18.2.60 PG_RANGE 794
18.2.61 PG_REDACTION_COLUMN 795
18.2.62 PG_REDACTION_POLICY 796
18.2.63 PG_RELFILENODE_SIZE 796
18.2.64 PG_RLSPOLICY 797
18.2.65 PG_RESOURCE_POOL 798
18.2.66 PG_REWRITE 798
18.2.67 PG_SECLABEL 799
18.2.68 PG_SHDEPEND 800
18.2.69 PG_SHDESCRIPTION 800

 版权所有 © 华为云计算技术有限公司

18.2.70 PG_SHSECLABEL 801
18.2.71 PG_STATISTIC 801
18.2.72 PG_STATISTIC_DATA 802
18.2.73 PG_STATISTIC_EXT 803
18.2.74 PG_STATISTIC_INFO 804
18.2.75 PG_STAT_OBJECT 805
18.2.76 PG_SUBSCRIPTION 808
18.2.77 PG_SYNONYM 809
18.2.78 PG_TABLESPACE 810
18.2.79 PG_TRIGGER 810
18.2.80 PG_TS_CONFIG 811
18.2.81 PG_TS_CONFIG_MAP 811
18.2.82 PG_TS_DICT 812
18.2.83 PG_TS_PARSER 812
18.2.84 PG_TS_TEMPLATE 813
18.2.85 PG_TYPE 813
18.2.86 PG_USER_MAPPING 816
18.2.87 PG_USER_STATUS 816
18.2.88 PG_WORKLOAD_ACTION 817
18.2.89 PGXC_CLASS 817
18.2.90 PGXC_GROUP 818
18.2.91 PGXC_NODE 819
18.2.92 PLAN_TABLE_DATA 821
18.2.93 SNAPSHOT 822
18.2.94 TABLES_SNAP_TIMESTAMP 822
18.2.95 性能视图快照系统表 823
18.3 系统视图 824
18.3.1 ALL_ALL_TABLES 824
18.3.2 ALL_CONSTRAINTS 824
18.3.3 ALL_CONS_COLUMNS 825
18.3.4 ALL_COL_COMMENTS 825
18.3.5 ALL_DEPENDENCIES 825
18.3.6 ALL_IND_COLUMNS 826
18.3.7 ALL_IND_EXPRESSIONS 826
18.3.8 ALL_INDEXES 827
18.3.9 ALL_OBJECTS 827
18.3.10 ALL_PROCEDURES 828
18.3.11 ALL_SEQUENCES 828
18.3.12 ALL_SOURCE 829
18.3.13 ALL_SYNONYMS 829
18.3.14 ALL_TAB_COLUMNS 829
18.3.15 ALL_TAB_COMMENTS 830

 版权所有 © 华为云计算技术有限公司

18.3.16 ALL_TABLES 830
18.3.17 ALL_USERS 831
18.3.18 ALL_VIEWS 831
18.3.19 DBA_DATA_FILES 832
18.3.20 DBA_USERS 832
18.3.21 DBA_COL_COMMENTS 832
18.3.22 DBA_CONSTRAINTS 832
18.3.23 DBA_CONS_COLUMNS 833
18.3.24 DBA_IND_COLUMNS 833
18.3.25 DBA_IND_EXPRESSIONS 834
18.3.26 DBA_IND_PARTITIONS 834
18.3.27 DBA_INDEXES 835
18.3.28 DBA_OBJECTS 835
18.3.29 DBA_PART_INDEXES 836
18.3.30 DBA_PART_TABLES 837
18.3.31 DBA_PROCEDURES 837
18.3.32 DBA_SEQUENCES 837
18.3.33 DBA_SOURCE 838
18.3.34 DBA_SYNONYMS 838
18.3.35 DBA_TAB_COLUMNS 838
18.3.36 DBA_TAB_COMMENTS 839
18.3.37 DBA_TAB_PARTITIONS 840
18.3.38 DBA_TABLES 841
18.3.39 DBA_TABLESPACES 841
18.3.40 DBA_TRIGGERS 842
18.3.41 DBA_VIEWS 842
18.3.42 DUAL 842
18.3.43 GET_ALL_TSC_INFO 842
18.3.44 GET_TSC_INFO 843
18.3.45 GLOBAL_COLUMN_TABLE_IO_STAT 843
18.3.46 GLOBAL_REDO_STAT 844
18.3.47 GLOBAL_REL_IOSTAT 845
18.3.48 GLOBAL_ROW_TABLE_IO_STAT 845
18.3.49 GLOBAL_STAT_DATABASE 846
18.3.50 GLOBAL_TABLE_CHANGE_STAT 847
18.3.51 GLOBAL_TABLE_STAT 848
18.3.52 GLOBAL_WORKLOAD_SQL_COUNT 849
18.3.53 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME 849
18.3.54 GLOBAL_WORKLOAD_TRANSACTION 850
18.3.55 GS_ALL_CONTROL_GROUP_INFO 851
18.3.56 GS_BLOCKLIST_QUERY 851
18.3.57 GS_BLOCKLIST_SQL 852

 版权所有 © 华为云计算技术有限公司

18.3.58 GS_CLUSTER_RESOURCE_INFO 852
18.3.59 GS_COLUMN_TABLE_IO_STAT 853
18.3.60 GS_OBS_READ_TRAFFIC 853
18.3.61 GS_OBS_WRITE_TRAFFIC 854
18.3.62 GS_INSTR_UNIQUE_SQL 855
18.3.63 GS_NODE_STAT_RESET_TIME 858
18.3.64 GS_OBS_LATENCY 858
18.3.65 GS_QUERY_MONITOR 859
18.3.66 GS_QUERY_RESOURCE_INFO 860
18.3.67 GS_REL_IOSTAT 861
18.3.68 GS_RESPOOL_RUNTIME_INFO 862
18.3.69 GS_RESPOOL_RESOURCE_INFO 862
18.3.70 GS_RESPOOL_MONITOR 864
18.3.71 GS_ROW_TABLE_IO_STAT 866
18.3.72 GS_SESSION_CPU_STATISTICS 866
18.3.73 GS_SESSION_MEMORY_STATISTICS 867
18.3.74 GS_SQL_COUNT 868
18.3.75 GS_STAT_DB_CU 869
18.3.76 GS_STAT_SESSION_CU 870
18.3.77 GS_TABLE_CHANGE_STAT 870
18.3.78 GS_TABLE_STAT 871
18.3.79 GS_TOTAL_NODEGROUP_MEMORY_DETAIL 872
18.3.80 GS_USER_MONITOR 872
18.3.81 GS_USER_TRANSACTION 874
18.3.82 GS_VIEW_DEPENDENCY 874
18.3.83 GS_VIEW_DEPENDENCY_PATH 875
18.3.84 GS_VIEW_INVALID 875
18.3.85 GS_WAIT_EVENTS 875
18.3.86 GS_WLM_OPERATOR_INFO 877
18.3.87 GS_WLM_OPERATOR_HISTORY 878
18.3.88 GS_WLM_OPERATOR_STATISTICS 880
18.3.89 GS_WLM_SESSION_INFO 882
18.3.90 GS_WLM_SESSION_HISTORY 888
18.3.91 GS_WLM_SESSION_STATISTICS 894
18.3.92 GS_WLM_SQL_ALLOW 898
18.3.93 GS_WORKLOAD_SQL_COUNT 899
18.3.94 GS_WORKLOAD_SQL_ELAPSE_TIME 899
18.3.95 GS_WORKLOAD_TRANSACTION 900
18.3.96 MPP_TABLES 900
18.3.97 PG_AVAILABLE_EXTENSION_VERSIONS 901
18.3.98 PG_AVAILABLE_EXTENSIONS 901
18.3.99 PG_BULKLOAD_STATISTICS 902

 版权所有 © 华为云计算技术有限公司

18.3.100 PG_COMM_CLIENT_INFO 902
18.3.101 PG_COMM_DELAY 903
18.3.102 PG_COMM_STATUS 903
18.3.103 PG_COMM_RECV_STREAM 904
18.3.104 PG_COMM_SEND_STREAM 905
18.3.105 PG_COMM_QUERY_SPEED 906
18.3.106 PG_CONTROL_GROUP_CONFIG 907
18.3.107 PG_CURSORS 907
18.3.108 PG_EXT_STATS 908
18.3.109 PG_GET_INVALID_BACKENDS 909
18.3.110 PG_GET_SENDERS_CATCHUP_TIME 909
18.3.111 PG_GLOBAL_TEMP_ATTACHED_PIDS 910
18.3.112 PG_GROUP 910
18.3.113 PG_INDEXES 911
18.3.114 PG_JOB 912
18.3.115 PG_JOB_PROC 913
18.3.116 PG_JOB_SINGLE 913
18.3.117 PG_LIFECYCLE_DATA_DISTRIBUTE 915
18.3.118 PG_LOCKS 915
18.3.119 PG_LWLOCKS 917
18.3.120 PG_NODE_ENV 918
18.3.121 PG_OS_THREADS 918
18.3.122 PG_POOLER_STATUS 918
18.3.123 PG_PREPARED_STATEMENTS 919
18.3.124 PG_PREPARED_XACTS 920
18.3.125 PG_PUBLICATION_TABLES 920
18.3.126 PG_QUERYBAND_ACTION 921
18.3.127 PG_REPLICATION_SLOTS 921
18.3.128 PG_ROLES 922
18.3.129 PG_RULES 923
18.3.130 PG_RUNNING_XACTS 923
18.3.131 PG_SECLABELS 924
18.3.132 PG_SEQUENCES 925
18.3.133 PG_SESSION_WLMSTAT 925
18.3.134 PG_SESSION_IOSTAT 927
18.3.135 PG_SETTINGS 927
18.3.136 PG_SHADOW 928
18.3.137 PG_SHARED_MEMORY_DETAIL 929
18.3.138 PG_STATS 930
18.3.139 PG_STAT_ACTIVITY 932
18.3.140 PG_STAT_ALL_INDEXES 934
18.3.141 PG_STAT_ALL_TABLES 934

 版权所有 © 华为云计算技术有限公司

18.3.142 PG_STAT_BAD_BLOCK 936
18.3.143 PG_STAT_BGWRITER 936
18.3.144 PG_STAT_DATABASE 937
18.3.145 PG_STAT_DATABASE_CONFLICTS 938
18.3.146 PG_STATS_EXT_EXPRS 939
18.3.147 PG_STAT_GET_MEM_MBYTES_RESERVED 941
18.3.148 PG_STAT_USER_FUNCTIONS 941
18.3.149 PG_STAT_USER_INDEXES 942
18.3.150 PG_STAT_USER_TABLES 942
18.3.151 PG_STAT_REPLICATION 943
18.3.152 PG_STAT_SYS_INDEXES 944
18.3.153 PG_STAT_SYS_TABLES 945
18.3.154 PG_STAT_XACT_ALL_TABLES 946
18.3.155 PG_STAT_XACT_SYS_TABLES 946
18.3.156 PG_STAT_XACT_USER_FUNCTIONS 947
18.3.157 PG_STAT_XACT_USER_TABLES 947
18.3.158 PG_STATIO_ALL_INDEXES 948
18.3.159 PG_STATIO_ALL_SEQUENCES 948
18.3.160 PG_STATIO_ALL_TABLES 949
18.3.161 PG_STATIO_SYS_INDEXES 949
18.3.162 PG_STATIO_SYS_SEQUENCES 950
18.3.163 PG_STATIO_SYS_TABLES 950
18.3.164 PG_STATIO_USER_INDEXES 951
18.3.165 PG_STATIO_USER_SEQUENCES 951
18.3.166 PG_STATIO_USER_TABLES 952
18.3.167 PG_THREAD_WAIT_STATUS 952
18.3.168 PG_TABLES 962
18.3.169 PG_TDE_INFO 963
18.3.170 PG_TIMEZONE_ABBREVS 964
18.3.171 PG_TIMEZONE_NAMES 964
18.3.172 PG_TOTAL_MEMORY_DETAIL 964
18.3.173 PG_TOTAL_SCHEMA_INFO 966
18.3.174 PG_TOTAL_USER_RESOURCE_INFO 966
18.3.175 PG_USER 968
18.3.176 PG_USER_MAPPINGS 969
18.3.177 PG_VIEWS 970
18.3.178 PG_WLM_STATISTICS 970
18.3.179 PGXC_BULKLOAD_PROGRESS 971
18.3.180 PGXC_BULKLOAD_INFO 972
18.3.181 PGXC_BULKLOAD_STATISTICS 975
18.3.182 PGXC_COLUMN_TABLE_IO_STAT 975
18.3.183 PGXC_COMM_CLIENT_INFO 976

 版权所有 © 华为云计算技术有限公司

18.3.184 PGXC_COMM_DELAY....977
18.3.185 PGXC_COMM_RECV_STREAM....977
18.3.186 PGXC_COMM_SEND_STREAM....978
18.3.187 PGXC_COMM_STATUS....979
18.3.188 PGXC_COMM_QUERY_SPEED....980
18.3.189 PGXC_DEADLOCK....980
18.3.190 PGXC_DISK_CACHE_STATS....982
18.3.191 PGXC_DISK_CACHE_ALL_STATS....982
18.3.192 PGXC_DISK_CACHE_PATH_INFO....984
18.3.193 PGXC_GET_STAT_ALL_TABLES....984
18.3.194 PGXC_GET_STAT_ALL_PARTITIONS....985
18.3.195 PGXC_GET_TABLE_SKEWNESS....986
18.3.196 PGXC_GLOBAL_TEMP_ATTACHED_PIDS....987
18.3.197 PGXC_GTM_SNAPSHOT_STATUS....987
18.3.198 PGXC_INSTANCE_TIME....988
18.3.199 PGXC_LOCKWAIT_DETAIL....988
18.3.200 PGXC_INSTR_UNIQUE_SQL....990
18.3.201 PGXC_LOCK_CONFLICTS....992
18.3.202 PGXC_LWLOCKS....993
18.3.203 PGXC_MEMORY_DEBUG_INFO....994
18.3.204 PGXC_NODE_ENV....996
18.3.205 PGXC_NODE_STAT_RESET_TIME....996
18.3.206 PGXC_OBS_IO_SCHEDULER_STATS....996
18.3.207 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS....998
18.3.208 PGXC_OS_RUN_INFO....1000
18.3.209 PGXC_OS_THREADS....1000
18.3.210 PGXC_POOLER_STATUS....1000
18.3.211 PGXC_PREPARED_XACTS....1001
18.3.212 PGXC_REDO_STAT....1001
18.3.213 PGXC_REL_IOSTAT....1002
18.3.214 PGXC_REPLICATION_SLOTS....1002
18.3.215 PGXC_RESPOOL_RUNTIME_INFO....1003
18.3.216 PGXC_RESPOOL_RESOURCE_INFO....1003
18.3.217 PGXC_RESPOOL_RESOURCE_HISTORY....1006
18.3.218 PGXC_ROW_TABLE_IO_STAT....1008
18.3.219 PGXC_RUNNING_XACTS....1009
18.3.220 PGXC_SETTINGS....1009
18.3.221 PGXC_SESSION_WLMSTAT....1010
18.3.222 PGXC_STAT_ACTIVITY....1012
18.3.223 PGXC_STAT_BAD_BLOCK....1015
18.3.224 PGXC_STAT_BGWRITER....1015
18.3.225 PGXC_STAT_DATABASE....1016

 版权所有 © 华为云计算技术有限公司

18.3.226 PGXC_STAT_OBJECT....1017
18.3.227 PGXC_STAT_REPLICATION....1021
18.3.228 PGXC_STAT_TABLE_DIRTY....1022
18.3.229 PGXC_STAT_WAL....1025
18.3.230 PGXC_SQL_COUNT....1026
18.3.231 PGXC_TABLE_CHANGE_STAT....1026
18.3.232 PGXC_TABLE_STAT....1027
18.3.233 PGXC_THREAD_WAIT_STATUS....1028
18.3.234 PGXC_TOTAL_MEMORY_DETAIL....1030
18.3.235 PGXC_TOTAL_SCHEMA_INFO....1032
18.3.236 PGXC_TOTAL_SCHEMA_INFO_ANALYZE....1032
18.3.237 PGXC_TOTAL_USER_RESOURCE_INFO....1033
18.3.238 PGXC_USER_TRANSACTION....1035
18.3.239 PGXC_VARIABLE_INFO....1036
18.3.240 PGXC_WAIT_DETAIL....1037
18.3.241 PGXC_WAIT_EVENTS....1039
18.3.242 PGXC_WLM_OPERATOR_HISTORY....1039
18.3.243 PGXC_WLM_OPERATOR_INFO....1041
18.3.244 PGXC_WLM_OPERATOR_STATISTICS....1043
18.3.245 PGXC_WLM_SESSION_INFO....1045
18.3.246 PGXC_WLM_SESSION_HISTORY....1050
18.3.247 PGXC_WLM_SESSION_STATISTICS....1056
18.3.248 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS....1060
18.3.249 PGXC_WLM_USER_RESOURCE_HISTORY....1061
18.3.250 PGXC_WLM_WORKLOAD_RECORDS....1064
18.3.251 PGXC_WORKLOAD_SQL_COUNT....1065
18.3.252 PGXC_WORKLOAD_SQL_ELAPSE_TIME....1065
18.3.253 PGXC_WORKLOAD_TRANSACTION....1066
18.3.254 PLAN_TABLE....1067
18.3.255 PV_FILE_STAT....1068
18.3.256 PV_INSTANCE_TIME....1068
18.3.257 PV_MATVIEW_DETAIL....1069
18.3.258 PV_OS_RUN_INFO....1070
18.3.259 PV_REDO_STAT....1070
18.3.260 PV_RUNTIME_ATTSTATS....1071
18.3.261 PV_RUNTIME_EXPRSTATS....1073
18.3.262 PV_RUNTIME_RELSTATS....1075
18.3.263 PV_SESSION_MEMORY....1075
18.3.264 PV_SESSION_MEMORY_DETAIL....1076
18.3.265 PV_SESSION_STAT....1077
18.3.266 PV_SESSION_TIME....1078
18.3.267 PV_TOTAL_MEMORY_DETAIL....1078

 版权所有 © 华为云计算技术有限公司

18.3.268 REDACTION_COLUMNS....1080
18.3.269 REDACTION_POLICIES....1080
18.3.270 REMOTE_TABLE_STAT....1081
18.3.271 SHOW_TSC_INFO....1082
18.3.272 SHOW_ALL_TSC_INFO....1082
18.3.273 USER_COL_COMMENTS....1083
18.3.274 USER_CONSTRAINTS....1083
18.3.275 USER_CONS_COLUMNS....1084
18.3.276 USER_INDEXES....1084
18.3.277 USER_IND_COLUMNS....1085
18.3.278 USER_IND_EXPRESSIONS....1085
18.3.279 USER_IND_PARTITIONS....1085
18.3.280 USER_JOBS....1086
18.3.281 USER_OBJECTS....1087
18.3.282 USER_PART_INDEXES....1088
18.3.283 USER_PART_TABLES....1088
18.3.284 USER_PROCEDURES....1089
18.3.285 USER_SEQUENCES....1089
18.3.286 USER_SOURCE....1089
18.3.287 USER_SYNONYMS....1090
18.3.288 USER_TAB_COLUMNS....1090
18.3.289 USER_TAB_COMMENTS....1091
18.3.290 USER_TAB_PARTITIONS....1091
18.3.291 USER_TABLES....1092
18.3.292 USER_TRIGGERS....1092
18.3.293 USER_VIEWS....1093
18.3.294 V$SESSION....1093
18.3.295 V$SESSION_LONGOPS....1093
19 DWS 数据库 GUC 参数....1094
19.1 查看 GUC 参数....1094
19.2 设置 GUC 参数....1094
19.3 GUC 使用说明....1096
19.4 连接和认证....1096
19.4.1 连接设置....1096
19.4.2 安全和认证（postgresql.conf）....1098
19.4.3 通信库参数....1099
19.5 资源消耗....1105
19.5.1 内存....1105
19.5.2 语句磁盘空间管控....1113
19.5.3 内核资源使用....1114
19.5.4 基于开销的清理延迟....1114
19.5.5 异步 IO....1116

 版权所有 © 华为云计算技术有限公司

19.5.6 磁盘缓存 1118
19.6 并行导入 1118
19.7 预写式日志 1120
19.7.1 设置 1120
19.7.2 检查点 1123
19.8 双机复制 1123
19.8.1 主服务器 1124
19.9 查询规划 1124
19.9.1 优化器方法配置 1124
19.9.2 优化器开销常量 1139
19.9.3 基因查询优化器 1141
19.9.4 其他优化器选项 1143
19.10 错误报告和日志 1162
19.10.1 记录日志的位置 1162
19.10.2 记录日志的时间 1162
19.10.3 记录日志的内容 1166
19.11 运行时统计 1170
19.11.1 查询和索引统计收集器 1170
19.11.2 性能统计 1174
19.12 资源管理 1175
19.13 自动清理 1185
19.14 客户端连接缺省设置 1188
19.14.1 语句行为 1188
19.14.2 区域和格式化 1193
19.14.3 其他缺省 1197
19.15 锁管理 1198
19.16 版本和平台兼容性 1201
19.16.1 历史版本兼容性 1201
19.16.2 平台和客户端兼容性 1204
19.17 容错性 1237
19.18 连接池参数 1238
19.19 集群事务 1240
19.20 开发人员选项 1242
19.21 审计 1260
19.21.1 审计开关 1260
19.21.2 操作审计 1261
19.22 事务监控 1262
19.23 GTM 相关参数 1263
19.24 其它选项 1263
20 DWS 开发者术语表 1272
21 SQL 语法参考 1287
21.1 DWS SQL 概述 1287

 版权所有 © 华为云计算技术有限公司

21.2 DWS SQL 语法格式说明 1288
21.3 与 PostgreSQL 的差异 1288
21.4 关键字 1291
21.5 数据类型 1321
21.5.1 数值类型 1321
21.5.2 货币类型 1325
21.5.3 布尔类型 1326
21.5.4 字符类型 1327
21.5.5 二进制类型 1330
21.5.6 日期/时间类型 1332
21.5.7 几何类型 1338
21.5.8 数组类型 1340
21.5.9 枚举类型 1344
21.5.10 网络地址类型 1345
21.5.11 位串类型 1347
21.5.12 文本搜索类型 1348
21.5.13 UUID 类型 1350
21.5.14 JSON 类型 1350
21.5.15 RoaringBitmap 类型 1353
21.5.16 HLL 数据类型 1354
21.5.17 对象标识符类型 1356
21.5.18 伪类型 1358
21.5.19 范围类型 1359
21.5.20 复合类型 1363
21.5.21 列存表支持的数据类型 1365
21.5.22 XML 类型 1366
21.6 常量与宏 1367
21.7 函数和操作符 1368
21.7.1 字符串处理函数和操作符 1368
21.7.1.1 字符串处理函数和操作符总体介绍 1368
21.7.1.2 获取字符串长度 1369
21.7.1.3 截取字符串 1372
21.7.1.4 拼接字符串 1383
21.7.1.5 替换和填充字符串 1385
21.7.1.6 删除指定字符 1390
21.7.1.7 分割字符串 1391
21.7.1.8 转换字符串 1393
21.7.1.9 比较字符串 1401
21.7.1.10 获取某一个字符或子字符串的位置 1401
21.7.1.11 判断某个字符串是否符合某种匹配模式(字符串相关的模式匹配函数) 1403
21.7.2 二进制字符串函数和操作符 1405
21.7.3 位串函数和操作符 1408

 版权所有 © 华为云计算技术有限公司

21.7.4 数字操作函数和操作符 1410
21.7.4.1 数字操作符 1410
21.7.4.2 数字操作函数 1413
21.7.5 时间、日期处理函数和操作符 1422
21.7.5.1 时间/日期操作符 1422
21.7.5.2 时间/日期函数 1424
21.7.5.3 EXTRACT 1441
21.7.5.4 date_part 1446
21.7.5.5 date_format 1446
21.7.5.6 time_format 1448
21.7.6 SEQUENCE 函数 1451
21.7.7 数组函数和操作符 1453
21.7.7.1 数组操作符 1453
21.7.7.2 数组函数 1455
21.7.8 逻辑操作符 1459
21.7.9 比较操作符 1460
21.7.10 模式匹配操作符 1460
21.7.11 聚集函数 1464
21.7.12 窗口函数 1479
21.7.13 类型转换函数 1484
21.7.14 JSON/JSONB 函数和操作符 1493
21.7.14.1 JSON/JSONB 操作符 1493
21.7.14.2 JSON/JSONB 函数 1496
21.7.15 安全函数 1516
21.7.16 条件表达式函数 1525
21.7.17 范围函数和操作符 1528
21.7.17.1 范围操作符 1529
21.7.17.2 范围函数 1532
21.7.18 数据脱敏函数 1534
21.7.19 位图函数和操作符 1535
21.7.19.1 Roaring Bitmap 操作符 1535
21.7.19.2 位图函数 1538
21.7.19.3 Roaring Bitmap 聚合函数 1545
21.7.19.4 使用场景 1547
21.7.20 UUID 函数 1549
21.7.21 文本检索函数和操作符 1551
21.7.21.1 文本检索操作符 1551
21.7.21.2 文本检索函数 1553
21.7.21.3 文本检索调试函数 1556
21.7.22 HLL 函数和操作符 1558
21.7.22.1 HLL 操作符 1558
21.7.22.2 哈希函数 1560

 版权所有 © 华为云计算技术有限公司

21.7.22.3 精度函数.....1563
21.7.22.4 聚合函数.....1565
21.7.22.5 功能函数.....1567
21.7.22.6 内置函数.....1569
21.7.23 返回集合的函数.....1570
21.7.23.1 序列号生成函数.....1570
21.7.23.2 下标生成函数.....1572
21.7.24 几何函数和操作符.....1573
21.7.24.1 几何操作符.....1573
21.7.24.2 几何函数.....1579
21.7.24.3 几何类型转换函数.....1582
21.7.25 网络地址函数和操作符.....1586
21.7.25.1 cidr 和 inet 操作符.....1586
21.7.25.2 网络地址函数.....1589
21.7.26 系统信息函数.....1592
21.7.26.1 会话信息函数.....1592
21.7.26.2 访问权限查询函数.....1597
21.7.26.3 模式可见性查询函数.....1601
21.7.26.4 系统表信息函数.....1602
21.7.26.5 系统函数信息函数.....1608
21.7.26.6 注释信息函数.....1609
21.7.26.7 事务 ID 和快照.....1610
21.7.26.8 计算子集群函数.....1611
21.7.26.9 锁信息函数.....1612
21.7.27 系统管理函数.....1612
21.7.27.1 配置设置函数.....1612
21.7.27.2 通用文件访问函数.....1613
21.7.27.3 服务器信号函数.....1615
21.7.27.4 快照同步函数.....1618
21.7.27.5 咨询锁函数.....1620
21.7.27.6 复制函数.....1622
21.7.27.7 资源管理函数.....1631
21.7.27.8 内存管理函数.....1644
21.7.27.9 其它函数.....1645
21.7.28 备份恢复控制函数.....1668
21.7.28.1 备份控制函数.....1668
21.7.28.2 恢复控制函数.....1673
21.7.29 数据库对象函数.....1677
21.7.29.1 数据库对象尺寸函数.....1677
21.7.29.2 数据库对象位置函数.....1680
21.7.29.3 分区管理函数.....1681
21.7.29.4 排序规则版本函数.....1682

 版权所有 © 华为云计算技术有限公司

21.7.29.5 冷热表用户函数....1682
21.7.30 残留文件管理函数....1685
21.7.30.1 获取残留文件列表函数....1685
21.7.30.2 扫描残留文件函数....1686
21.7.30.3 验证残留文件函数....1692
21.7.30.4 删除残留文件函数....1695
21.7.30.5 残留文件管理函数应用....1697
21.7.31 统计信息函数....1700
21.7.32 触发器函数....1738
21.7.33 XML 函数....1738
21.7.33.1 产生 XML 内容....1739
21.7.33.2 XML 谓词....1742
21.7.33.3 处理 XML....1744
21.7.33.4 将表映射到 XML....1746
21.7.34 调用栈记录函数....1748
21.7.35 Hudi 系统函数....1750
21.7.36 漏斗函数和留存函数....1757
21.7.37 EXTERNAL SCHEMA 系统函数....1761
21.7.38 存算分离函数....1762
21.7.38.1 数据共享函数....1762
21.7.38.2 获取 tablespace 垃圾桶文件信息函数....1763
21.7.39 时序函数....1764
21.8 表达式....1770
21.8.1 简单表达式....1771
21.8.2 条件表达式....1772
21.8.3 子查询表达式....1776
21.8.4 数组表达式....1779
21.8.5 行表达式....1780
21.9 类型转换....1781
21.9.1 概述....1781
21.9.2 操作符....1782
21.9.3 函数....1784
21.9.4 值存储....1786
21.9.5 UNION，CASE 和相关构造....1787
21.10 全文检索....1789
21.10.1 介绍....1789
21.10.1.1 全文检索概述....1789
21.10.1.2 文档概念....1790
21.10.1.3 基本文本匹配....1791
21.10.1.4 分词器....1792
21.10.2 表和索引....1792
21.10.2.1 搜索表....1792

 版权所有 © 华为云计算技术有限公司

21.10.2.2 创建索引 1794
21.10.2.3 索引使用约束 1795
21.10.3 控制文本搜索 1795
21.10.3.1 解析文档 1795
21.10.3.2 解析查询 1796
21.10.3.3 排序查询结果 1797
21.10.3.4 高亮搜索结果 1799
21.10.4 附加功能 1800
21.10.4.1 处理 tsvector 1800
21.10.4.2 处理查询 1801
21.10.4.3 查询重写 1801
21.10.4.4 收集文献统计 1802
21.10.5 解析器 1803
21.10.6 词典 1807
21.10.6.1 词典概述 1807
21.10.6.2 停用词 1808
21.10.6.3 Simple 词典 1808
21.10.6.4 Synonym 词典 1809
21.10.6.5 Thesaurus 词典 1811
21.10.6.6 Ispell 词典 1812
21.10.6.7 Snowball 词典 1813
21.10.7 配置示例 1813
21.10.8 测试和调试文本搜索 1815
21.10.8.1 分词器测试 1815
21.10.8.2 解析器测试 1816
21.10.8.3 词典测试 1817
21.10.9 限制约束 1817
21.11 系统操作 1817
21.12 事务管理 1818
21.13 DDL 语法 1823
21.13.1 DDL 语法一览表 1823
21.13.2 数据库&表空间&模式&数据类型 1829
21.13.2.1 CREATE DATABASE 1829
21.13.2.2 ALTER DATABASE 1833
21.13.2.3 DROP DATABASE 1835
21.13.2.4 CREATE SCHEMA 1837
21.13.2.5 ALTER SCHEMA 1838
21.13.2.6 DROP SCHEMA 1840
21.13.2.7 CREATE EXTERNAL SCHEMA 1841
21.13.2.8 ALTER EXTERNAL SCHEMA 1844
21.13.2.9 CREATE TYPE 1845
21.13.2.10 ALTER TYPE 1853

 版权所有 © 华为云计算技术有限公司

21.13.2.11 DROP TYPE....1855
21.13.3 用户和角色....1856
21.13.3.1 CREATE USER....1856
21.13.3.2 ALTER USER....1862
21.13.3.3 DROP USER....1868
21.13.3.4 CREATE ROLE....1869
21.13.3.5 ALTER ROLE....1876
21.13.3.6 SET ROLE....1882
21.13.3.7 DROP ROLE....1883
21.13.3.8 CREATE GROUP....1884
21.13.3.9 ALTER GROUP....1885
21.13.3.10 DROP GROUP....1886
21.13.3.11 DROP OWNED....1887
21.13.4 表和外部数据源....1887
21.13.4.1 CREATE TABLE....1887
21.13.4.2 CREATE TABLE PARTITION....1918
21.13.4.3 CREATE TABLE AS....1946
21.13.4.4 ALTER TABLE....1952
21.13.4.5 ALTER TABLE PARTITION....1968
21.13.4.6 RENAME TABLE....1982
21.13.4.7 DROP TABLE....1983
21.13.4.8 CREATE FOREIGN TABLE (SQL on OBS or Hadoop)....1984
21.13.4.9 CREATE FOREIGN TABLE (OBS 导入导出)....2011
21.13.4.10 CREATE FOREIGN TABLE (GDS 导入导出)....2025
21.13.4.11 ALTER FOREIGN TABLE (For HDFS or OBS)....2044
21.13.4.12 ALTER FOREIGN TABLE (GDS 导入导出)....2046
21.13.4.13 DROP FOREIGN TABLE....2048
21.13.4.14 CREATE SERVER....2049
21.13.4.15 ALTER SERVER....2053
21.13.4.16 DROP SERVER....2056
21.13.5 视图和物化视图....2057
21.13.5.1 CREATE VIEW....2057
21.13.5.2 ALTER VIEW....2059
21.13.5.3 DROP VIEW....2062
21.13.5.4 CREATE MATERIALIZED VIEW....2063
21.13.5.5 ALTER MATERIALIZED VIEW....2071
21.13.5.6 REFRESH MATERIALIZED VIEW....2073
21.13.5.7 DROP MATERIALIZED VIEW....2075
21.13.6 索引&序列&文件目录....2076
21.13.6.1 CREATE INDEX....2076
21.13.6.2 ALTER INDEX....2087
21.13.6.3 REINDEX....2092

 版权所有 © 华为云计算技术有限公司

21.13.6.4 DROP INDEX..... 2094
21.13.6.5 CREATE SEQUENCE..... 2095
21.13.6.6 ALTER SEQUENCE..... 2099
21.13.6.7 DROP SEQUENCE..... 2101
21.13.6.8 CREATE TEXT SEARCH CONFIGURATION..... 2101
21.13.6.9 ALTER TEXT SEARCH CONFIGURATION..... 2104
21.13.6.10 DROP TEXT SEARCH CONFIGURATION..... 2106
21.13.6.11 CREATE TEXT SEARCH DICTIONARY..... 2107
21.13.6.12 ALTER TEXT SEARCH DICTIONARY..... 2112
21.13.6.13 DROP TEXT SEARCH DICTIONARY..... 2114
21.13.6.14 CLUSTER..... 2115
21.13.7 存储过程&函数&触发器&游标..... 2117
21.13.7.1 CREATE FUNCTION..... 2117
21.13.7.2 ALTER FUNCTION..... 2132
21.13.7.3 DROP FUNCTION..... 2135
21.13.7.4 CREATE PROCEDURE..... 2136
21.13.7.5 DROP PROCEDURE..... 2142
21.13.7.6 CREATE TRIGGER..... 2142
21.13.7.7 ALTER TRIGGER..... 2147
21.13.7.8 DROP TRIGGER..... 2149
21.13.7.9 FETCH..... 2149
21.13.7.10 MOVE..... 2152
21.13.7.11 CURSOR..... 2154
21.13.7.12 CLOSE..... 2155
21.13.8 数据脱敏和安全相关..... 2156
21.13.8.1 CREATE ROW LEVEL SECURITY POLICY..... 2156
21.13.8.2 ALTER ROW LEVEL SECURITY POLICY..... 2161
21.13.8.3 DROP ROW LEVEL SECURITY POLICY..... 2162
21.13.8.4 CREATE REDACTION POLICY..... 2163
21.13.8.5 ALTER REDACTION POLICY..... 2167
21.13.8.6 DROP REDACTION POLICY..... 2169
21.13.9 资源池和过滤规则..... 2170
21.13.9.1 CREATE RESOURCE POOL..... 2170
21.13.9.2 ALTER RESOURCE POOL..... 2175
21.13.9.3 DROP RESOURCE POOL..... 2176
21.13.9.4 CREATE EXCEPT RULE..... 2177
21.13.9.5 ALTER EXCEPT RULE..... 2180
21.13.9.6 DROP EXCEPT RULE..... 2182
21.13.9.7 CREATE BLOCK RULE..... 2183
21.13.9.8 ALTER BLOCK RULE..... 2190
21.13.9.9 DROP BLOCK RULE..... 2194
21.13.10 性能和执行计划..... 2195

 版权所有 © 华为云计算技术有限公司

21.13.10.1 CREATE STATISTICS....2195
21.13.10.2 DROP STATISTICS....2196
21.13.11 连接&会话&资源清理....2197
21.13.11.1 CLEAN CONNECTION....2197
21.13.11.2 ALTER SYSTEM KILL SESSION....2198
21.13.11.3 ALTER SESSION....2199
21.13.11.4 DISCARD....2202
21.13.11.5 TRUNCATE....2204
21.13.11.6 VACUUM....2207
21.13.12 设置与注释....2211
21.13.12.1 SHOW....2212
21.13.12.2 SET....2212
21.13.12.3 RESET....2214
21.13.12.4 SET CONSTRAINTS....2215
21.13.12.5 SET SESSION AUTHORIZATION....2216
21.13.12.6 COMMENT....2217
21.13.13 发布和订阅....2220
21.13.13.1 CREATE PUBLICATION....2220
21.13.13.2 ALTER PUBLICATION....2222
21.13.13.3 DROP PUBLICATION....2223
21.13.13.4 CREATE SUBSCRIPTION....2224
21.13.13.5 ALTER SUBSCRIPTION....2226
21.13.13.6 DROP SUBSCRIPTION....2228
21.13.14 其他....2228
21.13.14.1 CREATE SYNONYM....2228
21.13.14.2 ALTER SYNONYM....2230
21.13.14.3 DROP SYNONYM....2231
21.13.14.4 ALTER LARGE OBJECT....2232
21.13.14.5 CREATE BARRIER....2233
21.14 DML 语法....2233
21.14.1 DML 语法一览表....2234
21.14.2 CALL....2234
21.14.3 COPY....2236
21.14.4 DELETE....2254
21.14.5 EXPLAIN....2257
21.14.6 EXPLAIN PLAN....2263
21.14.7 LOCK....2265
21.14.8 MERGE INTO....2269
21.14.9 INSERT 和 UPSERT....2273
21.14.9.1 INSERT....2273
21.14.9.2 UPSERT....2283
21.14.10 UPDATE....2287

 版权所有 © 华为云计算技术有限公司

21.14.11 VALUES....2295
21.15 DCL 语法....2297
21.15.1 DCL 语法一览表....2297
21.15.2 ALTER DEFAULT PRIVILEGES....2297
21.15.3 ANALYZE | ANALYSE....2300
21.15.4 DEALLOCATE....2305
21.15.5 DO....2305
21.15.6 EXECUTE....2306
21.15.7 EXECUTE DIRECT....2307
21.15.8 GRANT....2308
21.15.9 PREPARE....2315
21.15.10 REASSIGN OWNED....2316
21.15.11 REVOKE....2317
21.16 DQL 语法....2320
21.16.1 DQL 语法一览表....2320
21.16.2 SELECT....2320
21.16.3 SELECT INTO....2345
21.17 TCL 语法....2346
21.17.1 TCL 语法一览表....2346
21.17.2 ABORT....2347
21.17.3 BEGIN....2347
21.17.4 CHECKPOINT....2348
21.17.5 COMMIT | END....2349
21.17.6 COMMIT PREPARED....2350
21.17.7 PREPARE TRANSACTION....2350
21.17.8 SAVEPOINT....2351
21.17.9 SET TRANSACTION....2352
21.17.10 START TRANSACTION....2353
21.17.11 ROLLBACK....2355
21.17.12 RELEASE SAVEPOINT....2355
21.17.13 ROLLBACK PREPARED....2356
21.17.14 ROLLBACK TO SAVEPOINT....2357

 版权所有 © 华为云计算技术有限公司

1 使用前必读

文档面向的读者对象

数据库开发指南重点面向数据库的设计者、应用程序开发人员或DBA，提供设计、构建、查询和维护数据仓库所需的信息。

作为数据库管理员和应用程序开发人员，至少需要了解以下知识：

- 操作系统知识。这是一切的基础。
- SQL语法。这是操作数据库的必备能力。

前置条件

使用本指南前，需要完成以下任务。

- 创建DWS集群。
- 使用SQL编辑器或SQL客户端连接DWS数据库。

关于上述任务的详细指导，请参见《数据仓库服务用户指南》的“创建DWS集群”和“连接DWS集群”章节。

阅读指引

对于首次接触 DWS的用户，建议先阅读以下部分：

- 介绍DWS服务的特点、功能和适用场景。
- DWS入门包含一个示例，引导您完成创建数据仓库集群、创建数据库表、上传数据和测试查询这一过程。

如果打算或正在将应用程序从其他数据仓库向DWS迁移，您可能想知道DWS在实施方式上有什么区别。

DWS进行数据库应用程序开发过程中，下表将帮您找到对应的信息。

 版权所有 © 华为云计算技术有限公司

如果要..	查阅建议
快速开始使用DWS。	首先，按照《数据仓库服务用户指南》中“创建DWS集群”和“连接DWS集群”章节的步骤快速部署集群、连接到数据库并尝试进行一些查询。 准备好构建数据库后，将数据加载到表中并编写查询内容以操作数据仓库中的数据后，可以回到《数据仓库服务数据库开发指南》。
了解DWS数据仓库的内部架构。	如果您想要更全面地了解DWS服务，请转到DWS产品首页。
了解如何设计表以实现良好性能。	**DWS开发设计建议**介绍数据库应用程序开发过程中，应当遵守的设计规范。依据这些规范进行建模，能够更好的契合DWS的分布式处理架构，输出更高效的业务SQL代码。 对业务的执行效率不满意，期望通过调优加快业务执行的情况下，可以参考**DWS性能调优**进行调优。性能调优是一项复杂的工程，有些时候无法系统性地说明和解释，而是依赖于DBA的经验判断。尽管如此，**DWS性能调优**一节还是期望能尽量系统性的对性能调优方法加以说明，方便应用开发人员和刚接触DWS的DBA参考。
加载数据。	**导入数据**介绍数据入库DWS的方法和途径。
管理用户、组和数据库安全。	**DWS数据库安全管理**涵盖数据库安全主题。
监控和优化系统性能。	**DWS系统表和系统视图**详细介绍您可以从中查询数据库状态并监控查询内容与流程的系统表和视图。 您还应该查阅《数据仓库服务用户指南》中的“监控集群”了解如何使用DWS管理控制台检查系统运行状况、监控指标。

SQL 语法文本格式约定

为了方便对语法使用的理解，在文档中对SQL语法文本按如下格式进行表述。

格式	意义
大写	语法关键字（语句中保持不变、必须照输的部分）采用大写表示。
小写	参数（语句中必须由实际值进行替代的部分）采用小写表示。
[ ]	表示用“[ ]”括起来的部分是可选的。
...	表示前面的元素可重复出现。
[x | y | ...]	表示从两个或多个选项中选取一个或者不选。
{ x | y | ... }	表示从两个或多个选项中选取一个。

版权所有 © 华为云计算技术有限公司

格式	意义
[x | y | ...] [...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用空格分隔。
[x | y | ...] [,...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用逗号分隔。
{ x | y | ... } [...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间以空格分隔。
{ x | y | ... } [,...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间用逗号分隔。

声明

DWS的作者们在进行文档写作时努力基于商用角度，从使用场景和任务完成角度给出内容指引。即使这样，文档中依然可能存在对Postgres内容的引用和参考。对于这类内容，遵从如下的Postgres Copyright：

Postgres-XC is Copyright © 1996-2013 by the PostgreSQL Global Development Group.

PostgreSQL is Copyright © 1996-2013 by the PostgreSQL Global Development Group.

Postgres95 is Copyright © 1994-5 by the Regents of the University of California.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS-IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

 版权所有 © 华为云计算技术有限公司

2 DWS 数据迁移

2.1 迁移数据到 DWS

DWS提供了灵活的数据入库方式，可以将多种数据源的数据导入到DWS中。各导入方式具有不同的特点，如表2-1所示，用户可以根据其特点自行选择。建议用户配合云数据迁移（Cloud Data Migration，简称CDM）和数据治理中心（DataArts Studio）一起使用，CDM用于批量数据迁移，DataArts Studio可以对整个ETL过程进行编排调度，同时提供可视化的开发环境。

说明

- OBS、MRS、DLI为云服务。
- GDS、DSC、gs_restore、gs_dump为内部工具。

表 2-1 数据导入方式说明

数据导入方式	数据源	说明	优势
从OBS并行导入数据	OBS	支持将存储在OBS上的TXT、CSV、ORC及CARBONDATA格式的数据并行导入到DWS，支持导入后查询数据，也支持远程读OBS上的数据。 DWS优先推荐的导入方式。	并行拉取方式，性能好，横向扩展。
使用GDS从远端服务器导入数据	Servers（即远端服务器）	使用DWS提供的GDS工具，利用多DN并行的方式，将数据从远端服务器导入到DWS。这种方式导入效率高，适用于大批量数据入库。	
从MRS导入数据到集群	MRS（HDFS）	配置一个DWS集群连接到一个MRS集群，然后将数据从MRS的HDFS中读取到DWS。	并行拉取方式，性能好，横向扩展。

版权所有 © 华为云计算技术有限公司

数据导入方式	数据源	说明	优势
基于GDS的跨集群互联互通	-	通过GDS进行数据中转，实现多个集群之间的数据同步。	适用于多套DWS集群之间的数据同步。
使用gsql元命令\COPY导入数据	本地文件	与直接使用SQL语句COPY不同，该命令读取/写入的文件只能是gsql客户端所在机器上的本地文件。	操作简单，适用于小批量数据入库。
使用COPY FROM STDIN导入数据	其他文件或数据库	使用Java语言开发应用程序时，通过调用JDBC驱动的CopyManager接口，从文件或其他数据库向DWS写入数据。	从其他数据库直接写入DWS的方式，具有业务数据无需落地成文件的优势。
使用CDM迁移数据到DWS	数据库、NoSQL、文件系统、大数据平台	CDM提供同构/异构数据源之间批量数据迁移的功能，帮助用户实现从多种类型的数据源迁移数据到DWS。CDM在迁移数据到DWS时，采用的是COPY方式和GDS并行导入方式。	数据源丰富，操作简单。
使用DSC工具迁移SQL脚本	数据库、NoSQL、文件系统、大数据平台	请参考第三方ETL工具的相关文档。 DWS提供了DSC工具，可以将Teradata/Oracle脚本迁移到DWS。	通过OBS中转，数据源丰富，数据转换能力强。
使用gs_dump和gs_dumpall命令导出元数据	• 纯文本格式 • 自定义归档格式 • 目录归档格式 • tar归档格式	gs_dump支持导出单个数据库或其内的对象，而gs_dumpall支持导出集群中所有数据库或各库的公共全局对象。 通过导入工具将导出的元数据信息导入至需要的数据库，可以完成数据库信息的迁移。	适用于元数据迁移。

版权所有 © 华为云计算技术有限公司

数据导入方式	数据源	说明	优势
使用gs_restore导入数据	sql/tmp/tar文件格式	在数据库迁移场景下，支持使用gs_restore工具将事先使用gs_dump工具导出的文件格式，重新导入DWS集群，实现表定义、数据库对象定义等元数据的导入。导入数据主要包括以下内容： • 所有数据库对象定义。 • 单个数据库对象定义。 • 单个Schema定义。 • 单张表定义。	

2.2 数据类型映射关系

2.2.1 DWS 外表与 HIVE/SPARK、ORC 及 PARQUET 数据类型的映射关系

使用外表读取Hive/Spark数据或其他第三方组件的ORC/Parquet数据时，由于不同平台数据类型存在差异，可能导致读取过程中数据不兼容问题。本节阐述了DWS外表数据类型与这些数据源数据类型之间的映射关系，通过正确配置外表字段数据类型，确保数据在DWS中能被准确识别、处理和查询，从而避免数据丢失，保障业务数据完整获取。

具体的外表语法请参考《 数据仓库服务SQL语法参考 》中“CREATE FOREIGN TABLE (SQL on OBS or Hadoop)”章节。

DWS 外表与 HIVE/SPARK 数据类型映射关系

表2-2以DWS外表支持的类型为基准，某些在HIVE/SPARK中支持的类型比如INTERVAL、UNION等，在DWS外表中不支持。

表 2-2 DWS 外表与 HIVE/SPARK 数据类型之间的映射关系

类型名称	DWS外表类型	HIVE类型	SPARK类型
1字节整数	• TINYINT（不推荐） • SMALLINT（推荐）	TINYINT	BYTE，TINYINT
2字节整数	SMALLINT	SMALLINT	SHORT，SMALLINT
4字节整数	INTEGER	INT，INTEGER	INT，INTEGER
8字节整数	BIGINT	BIGINT	LONG，BIGINT

版权所有 © 华为云计算技术有限公司

类型名称	DWS外表类型	HIVE类型	SPARK类型
单精度浮点	FLOAT，REAL	FLOAT	FLOAT，REAL
双精度浮点	FLOAT8，DOUBLE	DOUBLE	DOUBLE
科学数据	DECIMAL[p (,s)]	DECIMAL	DECIMAL
日期类型	DATE	DATE	DATE
时间类型	TIMESTAMP WITHOUT TIME ZONE	TIMESTAMP	TIMESTAMP_NTZ
	TIMESTAMP WITH TIME ZONE		TIMESTAMP，TIMESTAMP_LTZ
布尔类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)	CHAR	CHAR(length)
Varchar类型	VARCHAR(n)	VARCHAR	VARCHAR(length)
	NVARCHAR2		
text类型	TEXT(CLOB)	STRING	STRING
二进制类型	BYTEA	BINARY	BINARY
复杂类型	JSON	ARRAY	ARRAY
		MAP	MAP
		STRUCT	STRUCT

DWS 外表与 ORC 数据类型映射关系

- “ORC类型”是ORC格式规范定义的内部类型，通常无需特别关注，在HIVE/SPARK定义不明确的场景下，可作为创建DWS外表的参考。
- DWS外表写入ORC文件时存在明确的一对一类型对应关系，但在（**读**）场景下，除了推荐的类型对应关系外，还存在一些可以兼容读取的场景，表2-3列出了这些兼容的情况。

表 2-3 DWS 外表与 ORC 数据类型的映射关系

类型分类	ORC类型	DWS外表类型（读）	DWS外表类型（写）
整数类型	boolean (1 bit)	BOOLEAN	BOOLEAN
	tinyint (8 bit)	• TINYINT（不推荐） • SMALLINT（推荐）	TINYINT（不推荐）
	smallint (16 bit)	SMALLINT	SMALLINT

版权所有 © 华为云计算技术有限公司

类型分类	ORC类型	DWS外表类型（读）	DWS外表类型（写）
	int (32 bit)	INTEGER	INTEGER
	bigint (64 bit)	BIGINT	BIGINT
浮点类型	float	FLOAT4	FLOAT4
	double	FLOAT8	FLOAT8
字符串类型	string	• TEXT（推荐） • VARCHAR (n) • CHAR (n) • DECIMAL[p (,s)]	• TEXT • DECIMAL[p (,s)] • CHAR (n) • VARCHAR (n)
	char	• CHAR (n)（推荐） • VARCHAR (n)	CHAR (n)
	varchar	VARCHAR(n)	VARCHAR (n)
二进制类型	binary	BYTEA	BYTEA
科学数据	decimal	DECIMAL[p (,s)]	DECIMAL[p (,s)]
时间类型	timestamp	• TIMESTAMP WITHOUT TIME ZONE • TIMESTAMP WITH TIME ZONE	• TIMESTAMP WITHOUT TIME ZONE • TIMESTAMP WITH TIME ZONE
	timestamp with local time zone	• TIMESTAMP WITHOUT TIME ZONE • TIMESTAMP WITH TIME ZONE	不支持
	date	• DATE • TIMESTAMP WITHOUT TIME ZONE • TIMESTAMP WITH TIME ZONE	DATE
复杂类型	struct	• JSON • JSONB	不支持
	list	• JSON • JSONB	不支持
	map	• JSON • JSONB	不支持

版权所有 © 华为云计算技术有限公司

类型分类	ORC类型	DWS外表类型（读）	DWS外表类型（写）
	union	不支持	不支持

DWS 外表与 PARQUET 数据类型映射关系

“PARQUET类型”指PARQUET格式规范定义的内部存储类型，PARQUET通过coverted_type（已废弃）或logical_type进一步描述类型行为。DWS（**写**）PARQUET仅使用logical_type，但（**读**）时兼容coverted_type。

表 2-4 DWS 外表与 PARQUET 数据类型的映射关系

PARQUET类型	DWS外表类型（读）	DWS外表类型（写）
BOOLEAN	BOOLEAN	BOOLEAN
INT32	• TINYINT（不推荐） • SMALLINT • INTEGER • DATE • TIMESTAMP[(p)] • DECIMAL[p (,s)]	• TINYINT（不推荐） • SMALLINT • INTEGER
INT64	• BIGINT • money • TIMESTAMP WITHOUT TIME ZONE] • TIMESTAMP WITH TIME ZONE • DECIMAL[p (,s)]	• BIGINT • DECIMAL[p (,s)] • money • TIMESTAMP WITHOUT TIME ZONE • TIMESTAMP WITH TIME ZONE
INT96	• TIMESTAMP[(p)] [WITHOUT TIME ZONE] • TIMESTAMP WITH TIME ZONE	不支持
FLOAT	FLOAT4	FLOAT4
DOUBLE	FLOAT8	FLOAT8
BYTE_ARRAY	• CHAR (n) • VARCHAR (n) • TEXT • NVARCHAR2 • BYTEA	• DECIMAL[p (,s)] • CHAR (n) • VARCHAR (n) • TEXT • NVARCHAR2 • BYTEA

版权所有 © 华为云计算技术有限公司

PARQUET类型	DWS外表类型（读）	DWS外表类型（写）
FIXED_LEN_BYTE_ARRAY	DECIMAL[p (,s)]	DECIMAL[p (,s)]

2.2.2 MySQL 迁移到 DWS 的数据类型映射

表 2-5 MySQL 迁移到 DWS 的数据类型映射

MySQL数据类型	DWS数据类型	是否支持映射
BIT[(M)]	BIT	支持
TINYINT[(M)]	SMALLINT	支持
TINYINT[(M)] [UNSIGNED]	SMALLINT	支持
SMALLINT[(M)]	SMALLINT	支持
SMALLINT[(M)] [UNSIGNED]	INTEGER	支持
MEDIUMINT[(M)]	INTEGER	支持
MEDIUMINT[(M)] [UNSIGNED]	INTEGER	支持
INT[(M)]	INTEGER	支持
INT[(M)] [UNSIGNED]	BIGINT	支持
BIGINT[(M)]	BIGINT	支持
BIGINT[(M)] [UNSIGNED]	NUMERIC	支持
DECIMAL[(M[,D])]	SMALLINT / INTEGER / BIGINT / NUMERIC 根据精度和标度找到合适的数据类型。	支持
FLOAT(p)	FLOAT4	支持
DOUBLE[(M,D)]	FLOAT8	支持
DATE	DATE	支持
DATETIME[(fsp)]	TIMESTAMP	支持
TIMESTAMP[(fsp)]	TIMESTAMP WITH TIME ZONE	支持
TIME[(fsp)]	TIME	支持
YEAR[(4)]	SMALLINT	支持
CHAR[(M)]	CHAR	支持

版权所有 © 华为云计算技术有限公司

MySQL数据类型	DWS数据类型	是否支持映射
VARCHAR(M)	CHARACTER VARYING() 根据源库列字符集，在目标库扩大相应倍数。	支持
BINARY[(M)]	BYTEA	支持
VARBINARY(M)	BYTEA	支持
TINYBLOB	BLOB	支持
TINYTEXT	TEXT	支持
BLOB	BLOB	支持
TEXT	TEXT	支持
MEDIUMBLOB	BLOB	支持
MEDIUMTEXT	TEXT	支持
LONGBLOB	BLOB	支持
LONGTEXT	TEXT	支持
ENUM	VARCHAR	支持
SET	VARCHAR	支持
JSON	JSONB	支持

说明

- DATE类型，MySQL的支持的范围是：'1000-01-01' to '9999-12-31'。

  DATETIME类型，MySQL的支持的范围是：'1000-01-01 00:00:00' to '9999-12-31 23:59:59'。

  TIMESTAMP类型，MySQL的支持的范围是：'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC。

  可参考MySQL的官方文档。
- TIME类型，MySQL的支持的范围是：'-838:59:59' to '838:59:59'，可参考MySQL的官方文档。而DWS的TIME类型的最小值：00:00:00，最大值：24:00:00。对于MySQL中小于00:00:00，或大于24:00:00的TIME类型的值，任务会同步失败。
- YEAR类型，MySQL的支持的范围是：1901 to 2155, 和 0000，可参考MySQL的官方文档。由于DWS中没有对应的类型，会将MySQL的YEAR类型映射成SMALLINT类型。
- 对于MySQL的DATE类型的'0000'任务会同步失败。
- 对于INT类型，如果该列含有自增属性（AUTO_INCREMENT）时，同步时会将设置该属性的字段同步为SERIAL类型。

版权所有 © 华为云计算技术有限公司

2.2.3 Oracle 迁移到 DWS 的数据类型映射

表 2-6 Oracle 迁移到 DWS 的数据类型映射

Oracle数据类型	DWS数据类型	是否支持映射
CHAR	CHAR	支持
VARCHAR	VARCHAR	支持
VARCHAR2	VARCHAR	支持
NCHAR	NCHAR	支持
NVARCHAR2	NVARCHAR	支持
NUMBER	NUMBER	支持
BINARY_FLOAT	REAL	支持
BINARY_DOUBLE	DOUBLE	支持
FLOAT	FLOAT/REAL	支持
DATE	TIMESTAMP	支持
TIMESTAMP	TIMESTAMP 如果指定精度值（*fsp*），则必须在0到6的范围内。值为0表示没有小数部分。如果省略，则默认精度为0。	支持
TIMESTAMP WITH TIME ZONE	TIMESTAMPTZ 如果指定精度值（*fsp*），则必须在0到6的范围内。值为0表示没有小数部分。如果省略，则默认精度为0。	支持
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMPTZ 如果指定精度值（*fsp*），则必须在0到6的范围内。值为0表示没有小数部分。如果省略，则默认精度为0。	支持
INTERVAL	INTERVAL	支持
BLOB	BYTEA	支持
CLOB	CLOB	支持
NCLOB	TEXT	支持
BFILE	不支持	不支持
LONG	TEXT	支持
LONG_RAW	BYTEA	支持

版权所有 © 华为云计算技术有限公司

Oracle数据类型	DWS数据类型	是否支持映射
RAW（非主键列和非唯一键列）	BYTEA	支持
RAW（主键列和唯一键列）	VARCHAR	支持
ROWID	CHARACTER(18)	支持
UROWID	不支持	不支持
XMLTYPE	TEXT	支持
SDO_GEOMETRY	不支持	不支持

2.2.4 Teradata 迁移到 DWS 的数据类型映射

表 2-7 Teradata 迁移到 DWS 的数据类型映射

Teradata数据类型	DWS数据类型	是否支持映射
BIGINT	BIGINT	支持
BYTEINT	SMALLINT	支持
DECIMAL [(n[,m])]	DECIMAL [(n[,m])]	支持
DOUBLE PRECISION	DOUBLE PRECISION	支持
FLOAT	DOUBLE PRECISION	支持
INT / INTEGER	INTEGER	支持
NUMBER / NUMERIC	NUMERIC	支持
NUMBER(n[,m])	NUMERIC (n[,m])	支持
REAL	REAL	支持
SMALLINT	SMALLINT	支持
CHAR[(n)] / CHARACTER [(n)]	CHAR(n)	支持
CLOB	CLOB	支持
LONG VARCHAR	TEXT	支持
VARCHAR(n) / CHAR VARYING(n) / CHARACTER VARYING(n)	VARCHAR(n)	支持
DATE	DATE	支持
TIME [(n)]	TIME [(n)]	支持

版权所有 © 华为云计算技术有限公司

Teradata数据类型	DWS数据类型	是否支持映射
TIME [(n)] WITH TIME ZONE	TIME [(n)] WITH TIME ZONE	支持
TIMESTAMP [(n)]	TIMESTAMP [(n)]	支持
TIMESTAMP [(n)] WITH TIME ZONE	TIMESTAMP [(n)] WITH TIME ZONE	支持
PERIOD(DATE)	daterange	支持
PERIOD(TIME [(n)])	tsrange [(n)]	支持
PERIOD(TIME WITH TIME ZONE)	tstzrange	支持
PERIOD(TIMESTAMP [(n)])	tsrange [(n)]	支持
PERIOD(TIMESTAMP WITH TIME ZONE)	tstzrange	支持
BLOB[(n)]	blob	支持
BYTE[(n)]	bytea	支持
VARBYTE[(n)]	bytea	支持

2.3 导入数据

2.3.1 从 OBS 并行导入数据

2.3.1.1 关于 OBS 并行导入

对象存储服务OBS（Object Storage Service）是云上提供的一个基于对象的海量存储服务，为客户提供安全、高可靠、低成本的数据存储能力。OBS为用户提供了超大存储容量的能力，适合存放任意类型的文件。

数据仓库服务DWS使用OBS作为集群数据与外部数据互相转化的平台，实现安全、高可靠和低成本的存储需求。

DWS支持将OBS上TXT、CSV、ORC、PARQUET、CARBONDATA以及JSON格式的数据导入到集群进行查询，也支持远程读OBS上的数据。因此对于经常查询的热数据建议直接导入DWS后再做查询。偶尔查询的冷数据可以存储在OBS上直接远程读以节省成本。

目前，导入数据有两种方式：

- 方式一：无需用户创建server，使用默认server创建外表，支持TXT、CSV格式的数据，参见**从OBS导入CSV、TXT数据**。

版权所有 © 华为云计算技术有限公司

- 方式二：需用户创建server，使用该server创建外表，支持ORC、CARBONDATA、TXT、CSV、PARQUET以及JSON格式的数据，参见**从OBS导入ORC、CARBONDATA、PARQUET和JSON数据**。

须知

- OBS导入导出数据时，不支持中文路径。
- OBS导入导出数据时，暂不支持跨Region进行OBS数据导入导出，必须确保OBS和DWS集群在同一个Region中。
- 在执行OBS导入导出时，为了确保数据导入或导出的正确性，需要在相同的兼容模式下操作。

  例如：在MySQL兼容模式下导入（导出）的数据，同样需要在MySQL兼容模式下才能正确导出（导入）。

概述

在数据迁移、ETL（Extract-Transform-Load）过程中，需要向DWS并行导入海量数据，使用普通方式会耗费大量的时间。DWS提供了OBS（Object Storage Service）及外表接口，通过OBS外表设置的导入URL路径、导入数据格式等信息来识别数据源文件，利用多DN（Datanode）并行的方式，实现了数据的快速并行导入。

优势：

- CN只负责任务的规划及下发，把数据导入的工作交给了DN，释放了CN的资源，使其有能力处理外部请求。
- 通过让各个DN都参与数据导入，充分利用各个设备的计算能力及网络带宽。
- 支持导入过程中对数据做预处理。
- 支持在导入过程中，针对数据格式错误设置导入容错性，并可在导入结束后根据错误信息定位错误数据。

劣势：

需要创建OBS外表，并且要在OBS服务器上存放导入数据。

适用场景：

高并发、大数据量导入。

相关概念

- **数据源文件**：存储有数据的TEXT、CSV、ORC、CARBONDATA、PARQUET、JSON文件。文件中保存的是待并行导入数据库的数据。
- **OBS**：对象存储服务，是一种可存储文档、图片、影音视频等非结构化数据的云存储服务。向DWS并行导入数据时，数据对象放置在OBS服务器上。
- **桶（Bucket）**：对OBS中的一个存储空间的形象称呼，是存储对象的容器。
  - 对象存储是一种非常扁平化的存储方式，桶中存储的对象都在同一个逻辑层级，去除了文件系统中的多层级树形目录结构。
  - 在OBS中，桶名必须是全局唯一的且不能修改，即用户创建的桶不能与自己已创建的其他桶名称相同，也不能与其他用户创建的桶名称相同。每个桶在创建时都会生成默认的桶ACL（Access Control List），桶ACL列表的每项包

版权所有 © 华为云计算技术有限公司

含了对被授权用户授予的权限，如读取权限、写入权限、完全控制权限等。用户需要对桶有相应的权限，才可以对桶进行操作，如创建、删除、显示、设置桶ACL等。

- 一个用户最多可创建100个桶，但每个桶中存放的总数据容量和对象/文件数量没有限制。

- **对象**：是存储在OBS中的基本数据单位。用户上传的数据以对象的形式存储在OBS的桶中。对象的属性包括名称Key，Metadata，Data。

  通常，将对象等同于文件来进行管理，但是由于OBS是一种对象存储服务，并没有文件系统中的文件和文件夹概念。为了使用户更方便进行管理数据，OBS提供了一种方式模拟文件夹。通过在对象的名称中增加“/”，如tpcds1000/stock.csv，tpcds1000可以等同于文件夹，stock.csv就可以等同于文件名，而对象名称（Key）仍然是tpcds1000/stock.csv、对象的内容就是stock.csv数据文件的内容。

- **Key**：对象的名称（键），是经过UTF-8编码的长度大于0且不超过1024的字符序列，一个桶里的每个对象必须拥有唯一的对象键值。用户可使用“桶名+对象名”来存储和获取对应的对象。
- **Metadata**：对象元数据，用来描述对象的信息。元数据又可分为系统元数据和用户元数据。这些元数据以键值对（Key-Value）的形式随http头域一起上传到OBS系统。
  - 系统元数据由OBS系统产生，在处理对象数据时使用。系统元数据包括：Date、Content-length、last-modify、Content-MD5等。
  - 用户元数据由用户上传对象时指定，是用户自己对对象的一些描述信息。
- **Data**：对象的数据内容，OBS对于数据的内容是无感知的，即认为对象内的数据为无状态的二进制数据。
- **数据库普通表**：数据库中的普通表，数据源文件中的数据最终并行导入到这些表中存储，包括行存表、列存表。
- **外表**：用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、编码格式、数据间的分隔符等信息。

导入数据原理

OBS导入原理如图2-1所示，CN负责任务的规划及下发，它是按文件给每个DN节点分配任务的。

分配算法如下：

例如，图2-1中，总共有4个节点DN0~DN3， OBS上有6个文件t1.data.0~t1.data.5，那么分配方式如下：

t1.data.0 -> DN0

t1.data.1 -> DN1

t1.data.2 -> DN2

t1.data.3 -> DN3

t1.data.4 -> DN0

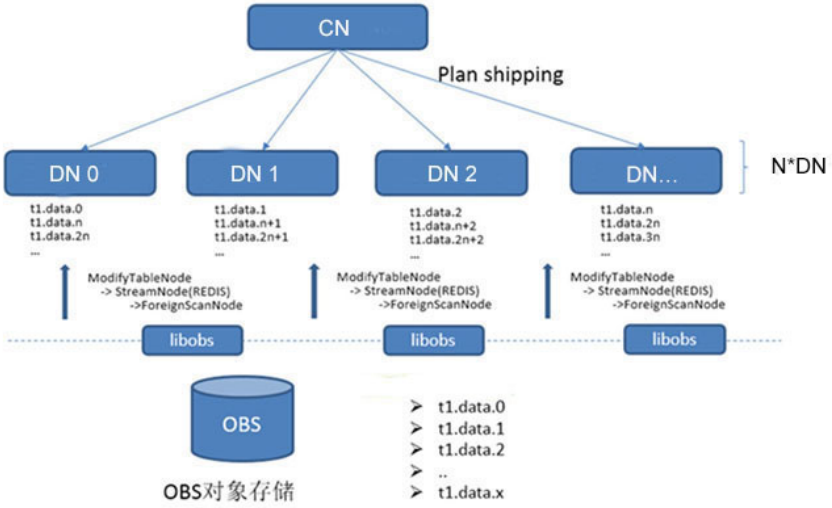
t1.data.5 -> DN1

其中DN0和DN1上分配了两个文件，其他DN分配了1个文件。

版权所有 © 华为云计算技术有限公司

如果OBS上文件大小都相同时，OBS上的文件数与DN节点数的比例为1:1时导入性能为最好，因为每个DN分配的任务都相同。因此建议将数据文件存储到OBS前，尽可能均匀地将文件切分成多个，文件的数量以DN的整数倍更适合。

图 2-1 通过 OBS 外表并行导入数据



 版权所有 © 华为云计算技术有限公司

导入流程图

图 2-2 并行导入流程

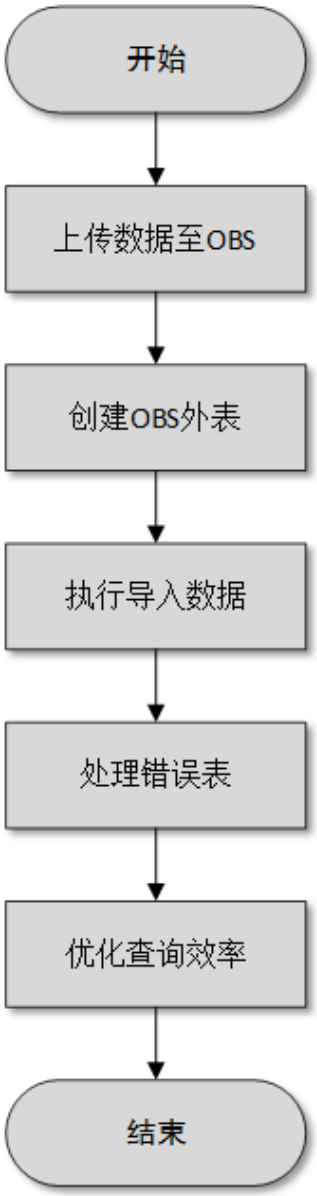


表 2-8 流程说明

流程	说明	子任务
上传数据至OBS	在OBS服务器上规划存储路径，并上传数据文件。 详细请参见**上传数据到OBS**。	-

版权所有 © 华为云计算技术有限公司

流程	说明	子任务
创建OBS外表	创建外表用于识别OBS服务器上的数据源文件。在OBS外表中保存了数据源在OBS服务器上的桶名、对象名、文件格式、存放位置、编码格式、数据间的分隔符等信息。 详细请参见**创建OBS外表**。	-
执行导入数据	在创建好外表后，通过INSERT语句，将数据快速、高效地导入到目标表中。 详细请参见**执行导入数据**。	-
处理错误表	在数据并行导入发生错误时，请根据错误信息，**处理导入错误**，以保证导入数据的完整性。	-
优化查询效率	导入数据后，通过ANALYZE语句生成表统计信息。ANALYZE语句会将统计结果自动存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以生成最有效的查询执行计划。	-

2.3.1.2 从 OBS 导入 CSV、TXT 数据

2.3.1.2.1 创建访问密钥（AK 和 SK）

在本示例中，将导入OBS数据到DWS集群数据库中。云平台用户通过客户端或API、SDK等方式访问OBS时，需要通过AK/SK认证方式进行认证鉴权。因此，当您需要通过客户端或JDBC/ODBC应用程序等方式连接DWS数据库访问OBS时，必须先获取访问密钥（AK和SK）。

- Access Key Id（AK）：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- Secret Access Key（SK）：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

创建访问密钥（AK 和 SK）

在创建访问密钥前，请确保登录控制台的账号已通过实名认证。

通过管理控制台创建访问密钥（AK和SK）操作步骤如下：

步骤1 登录DWS控制台。

版权所有 © 华为云计算技术有限公司

步骤2 单击右上角用户名，在下拉菜单中单击“我的凭证”。

步骤3 选择“管理访问密钥”页签。

如果在访问密钥列表中已经有访问密钥了，可以直接使用现有的访问密钥。但是，在访问密钥列表中只能查看到访问密钥ID（即Access Key ID），只有在新增访问密钥时，用户才可以下载到含有Access Key ID和Secret Access Key的密钥文件。如果您没有该密钥文件，可以单击“新增访问密钥”重新创建。

说明

- 每个用户最多可创建两个有效的访问密钥，如果当前已存在两个访问密钥，只能先删除现有的访问密钥，然后再重新创建。删除时，需要输入当前用户的登录密码、邮箱或手机短信的验证码，验证通过才能成功删除。
- 为了账号安全性，建议您定期更换并妥善保存访问密钥。

步骤4 单击“新增访问密钥”。

步骤5 在弹出的对话框中，输入登录密码和对应验证码，然后单击“确定”。

说明

- 用户如果未绑定邮箱和手机，则只需输入登录密码。
- 用户如果同时绑定了邮箱和手机，可以选择其中一种方式进行验证。

步骤6 在弹出的“下载确认”提示框中，单击“确定”后，密钥会直接保存到浏览器默认的下载文件夹中。

说明

- 为防止访问密钥泄露，建议您将其保存到安全的位置。
- 如果用户在此提示框中单击“取消”，则不会下载密钥，后续也将无法重新下载。如果需要使用访问密钥，可以重新创建新的访问密钥。

步骤7 打开下载下来的“credentials.csv”文件即可获取到访问密钥（AK和SK）。

----结束

注意事项

当用户发现访问密钥被异常使用（包括丢失，泄露等情况），或不再使用访问密钥时，建议在访问密钥列表中立即删除密钥或者通知管理员重置相关密钥。

删除访问密钥时，需要输入登录密码和邮箱或者手机验证码进行验证。

说明

删除的访问密钥将永久删除且无法恢复。

2.3.1.2.2 上传数据到 OBS

操作场景

从OBS导入数据到集群之前，需要提前准备数据源文件，并将数据源文件上传到OBS。如果您的数据文件已经在OBS上了，则只需完成**上传数据到OBS**中的**步骤2~步骤3**。

版权所有 © 华为云计算技术有限公司

准备数据文件

准备需要上传到OBS的数据源文件。DWS只支持CSV、TEXT、ORC和CARBONDATA格式的数据源文件。

如果用户数据无法以CSV格式保存，可以选择以文本类型保存为其他任意格式后缀的文件。

说明

根据**导入数据原理**，当数据源文件的数据量较大时，将数据文件存储到OBS前，尽可能均匀地将文件切分成多个，文件数量为DataNode的整数倍时，导入性能更好。

假设您已将3个CSV数据文件存储在OBS上，其原始数据分别如下：

- 数据文件“product_info.0”

  示例数据如下所示：

  ```
  100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt
  Women,red,M,328,2017-09-04,715,good!
  205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear
  Women,pink,L,584,2017-09-05,406,very good!
  300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt men,red,XL,1245,2017-09-03,502,Bad.
  310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket women,red,L,411,2017-09-05,436,It's
  really super nice.
  150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans
  Women,blue,M,1223,2017-09-06,1200,The seller's packaging is exquisite.
  ```

- 数据文件“product_info.1”

  示例数据如下所示：

  ```
  200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants
  men,black,L,997,2017-09-10,301,The clothes are of good quality.
  250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress
  women,black,S,841,2017-09-15,299,Follow the store for a long time.
  108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress women,red,M,85,2017-09-14,22,It's
  really amazing to buy.
  450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket
  women,white,M,114,2017-09-14,22,Open the package and the clothes have no odor.
  260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat
  women,red,L,2004,2017-09-15,826,Very favorite clothes.
  ```

- 数据文件“product_info.2”

  示例数据如下所示：

  ```
  980,"ZKDS-J",2017-09-13,"B","2017 Women's Cotton Clothing","red","M",112,,,
  98,"FKQB-I",2017-09-15,"B","2017 new shoes men","red","M",4345,2017-09-18,5473
  50,"DMQY-K",2017-09-21,"A","2017 pants men","red","37",28,2017-09-25,58,"good","good","good"
  80,"GKLW-l",2017-09-22,"A","2017 Jeans Men","red","39",58,2017-09-25,72,"Very comfortable."
  30,"HWEC-L",2017-09-23,"A","2017 shoes women","red","M",403,2017-09-26,607,"good!"
  40,"IQPD-M",2017-09-24,"B","2017 new pants Women","red","M",35,2017-09-27,52,"very good."
  50,"LPEC-N",2017-09-25,"B","2017 dress Women","red","M",29,2017-09-28,47,"not good at all."
  60,"NQAB-O",2017-09-26,"B","2017 jacket women","red","S",69,2017-09-29,70,"It's beautiful."
  70,"HWNB-P",2017-09-27,"B","2017 jacket women","red","L",30,2017-09-30,55,"I like it so much"
  80,"JKHU-Q",2017-09-29,"C","2017 T-shirt","red","M",90,2017-10-02,82,"very good."
  ```

上传数据到 OBS

步骤1 上传数据到OBS。

将待导入的数据源文件存储在OBS桶中。

1. 登录OBS控制台。

   单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。

版权所有 © 华为云计算技术有限公司

2. 创建桶。

   如何创建OBS桶，具体请参见《对象存储服务用户指南》中的“控制台指南 > 管理桶 > 创建桶”章节。

   例如，创建以下两个桶：“mybucket”和“mybucket02”。

3. 新建文件夹。

   具体请参见《对象存储服务用户指南》中的“控制台指南 > 管理对象 > 新建文件夹”章节。

   例如：

   - 在已创建的OBS桶“mybucket”中新建一个文件夹“input_data”。
   - 在已创建的OBS桶“mybucket02”中新建一个文件夹“input_data”。

4. 上传文件。

   具体请参见《对象存储服务用户指南》中的“控制台指南 > 管理对象 >上传文件”章节。

   例如：

   - 将以下数据文件上传到OBS桶“mybucket”的“input_data”目录中。

     ```
     product_info.0
     product_info.1
     ```

   - 将以下数据文件上传到OBS桶“mybucket02”的“input_data”目录中。

     ```
     product_info.2
     ```

步骤2 获取数据源文件的OBS路径。

数据源文件在上传到OBS桶之后，会生成全局唯一的访问路径。数据源文件的OBS路径用于创建外表时location参数设置。

location参数中OBS文件的路径由“obs://”、桶名和文件路径组成，即为：obs://<bucket_name>/<file_path>/

例如，在本例中，location参数中数据文件的OBS路径分别为：

```
obs://mybucket/input_data/product_info.0
obs://mybucket/input_data/product_info.1
obs://mybucket02/input_data/product_info.2
```

步骤3 为导入用户设置OBS桶的读取权限。

在从OBS导入数据到集群时，执行导入操作的用户需要取得数据源文件所在OBS桶的读取权限。通过配置桶的ACL权限，可以将读取权限授予指定的用户账号。

具体请参见《对象存储服务用户指南》中的“控制台指南 > 权限控制 > 配置桶ACL”章节。

----结束

2.3.1.2.3 创建 OBS 外表

操作步骤

步骤1 根据**上传数据到OBS**中规划的路径，由此确定创建外表时使用的参数**location**的值。

步骤2 用户获取OBS访问协议对应的AK值和SK值。

获取访问密钥，请登录DWS控制台，单击右上角的用户名并选择菜单“我的凭证”，然后在左侧导航树单击“访问密钥”。在访问密钥页面，可以查看已有的访问密钥ID

版权所有 © 华为云计算技术有限公司

（即AK），如果要同时获取AK和SK，可以单击“新增访问密钥”创建并下载访问密钥。

步骤3 梳理待导入数据的格式信息，确定创建外表时使用的数据格式参数的值。需要收集的主要数据源格式信息如下：

- **format**：外表中数据源文件的格式。OBS外表导入支持CSV、TEXT格式。缺省值为TEXT。
- **header**：指定导出数据文件是否包含标题行，header只能用于CSV格式的文件中。
- **delimiter**：指定数据文件行数据的字段分隔符，不指定则使用默认分隔符。
- 外表可以识别的更多参数，详细使用请参见《 数据仓库服务SQL语法参考 》中“CREATE FOREIGN TABLE (OBS导入导出)”章节的数据格式参数说明。

步骤4 **规划并行导入容错性**，以控制导入过程中处理错误的方式。

- **fill_missing_fields**：数据入库时，数据源文件中某行的最后一个字段缺失时，请选择是直接将字段设为Null，还是在错误表中报错提示。

  说明

  取值范围：true/on，false/off。
  - 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。
  - 参数为false/off，如果最后一个字段缺失会显示如下错误信息。
    ```
    missing data for column "tt"
    ```

  缺省值：false/off。

- **ignore_extra_data**：数据源文件中的字段比外表定义列数多时，请选择是忽略多出的列，还是在错误表中报错提示。

  说明

  取值范围：true/on、false/off。
  - 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。
  - 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。
    ```
    extra data after last expected column
    ```

  缺省值：false/off。

- **per node reject limit**：本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量。如果有一个DN实例上录入错误表中的错误数量超过设定值时，本次导入失败，报错退出。可以选择不做限制，也可以根据所能容忍的错误数量选择一个上限值。
- **compatible_illegal_chars**：导入时遇到非法字符，选择如何处理。是将非法字符按照转换规则转换后入库，还是报错中止导入。

  说明

  取值范围：true/on，false/off。
  - 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。
  - 参数为false/off，导入时遇到非法字符进行报错，中断导入。

  缺省值：false/off。

  非法字符容错转换规则如下：
  - 对于'\0'，容错后转换为空格。

版权所有 © 华为云计算技术有限公司

  - 对于其他非法字符，容错后转换为问号。
  - 对非法字符进行容错转换时，如遇NULL、DELIMITER、QUOTE、ESCAPE也设置成了空格或问号，DWS会通过如“illegal chars conversion may confuse COPY escape 0x20”等报错信息提示用户修改可能引起混淆的参数以避免导入错误。
- **error_table_name**：用于记录数据格式错误信息的错误表表名。并行导入结束后查询此错误信息表，能够获取详细的错误信息。
- 更多参数，详细使用请参见《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (OBS导入导出)”章节的容错性参数说明。

步骤5 根据前面步骤确定的参数，**创建OBS外表**。外表的创建语法以及详细使用，请参考《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (OBS导入导出)”章节。

----结束

示例

在DWS数据库中，创建一个外表。参数信息如下所示：

- **数据格式参数访问密钥（AK和SK）**
  - 用户获取OBS访问协议对应的AK值（access_key）。
  - 用户获取OBS访问协议对应的SK值（secret_access_key）。

  说明

  请根据用户实际获取的access_key和secret_access_key的密钥替换示例中的对应内容。
- **设置数据格式参数**
  - **数据源文件格式（format）**为“CSV”。
  - **编码格式（encoding）**为“UTF-8”。
  - **是否使用加密（encrypt），**默认为 “off”。
  - **字段分隔符（delimiter）**为“,”。
  - **引号字符（quote）**使用默认值双引号。
  - **null（数据文件中空值的表示）**为“一个没有引号的空字符串”。
  - **header（指定导出数据文件是否包含标题行）**为“false”，当数据文件第一行不是标题行（即表头），不需要设置。

    说明

    OBS导出数据时不支持该参数为true，使用缺省值false。
- **设置导入时的容错性参数**
  - **PER NODE REJECT LIMIT '*value*'** 为“unlimited”，即接受导入过程中所有数据格式错误。
  - **LOG INTO *error_table_name***指定为“product_info_err”，将数据导入过程中出现的数据格式错误信息写入表“product_info_err”。
  - **fill_missing_fields**为“true”，即当数据加载时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。
  - **ignore_extra_data**为“true”，当数据加载时，若数据源文件比外表定义列数多，则忽略行尾多出来的列，不报错。

 版权所有 © 华为云计算技术有限公司

根据以上信息，创建的外表如下所示：

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
DROP FOREIGN TABLE product_info_ext;

CREATE FOREIGN TABLE product_info_ext
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
SERVER gsmpp_server
OPTIONS(

LOCATION 'obs://mybucket/input_data/product_info | obs://mybucket02/input_data/product_info',
FORMAT 'CSV' ,
DELIMITER ',',
encoding 'utf8',
header 'false',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
fill_missing_fields 'true',
ignore_extra_data 'true'
)
READ ONLY
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

返回如下信息表示创建成功：

```
CREATE FOREIGN TABLE
```

2.3.1.2.4 执行导入数据

背景信息

在执行数据导入前，您可以参考以下优秀实践方法进行合理的设计部署，最大化的使用系统资源，以提高数据导入性能。

- OBS的数据导入性能，多数场景受限于网络的并发访问速率，因此在OBS服务器上最好部署多个桶，使用多桶并发导入，提高DN数据传输利用率。
- 并发导入场景，与单表导入相似，至少应保证I/O性能大于网络最大速率。
- 配置GUC参数“raise_errors_if_no_files”、“partition_mem_batch”和“partition_max_cache_size”，设置导入时是否区分“导入文件记录数为空”和“导入文件不存在”、导入时的缓存个数以及数据缓存区大小。
- 如果导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引。在数据导入完成后，再重新创建索引。

版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 在DWS数据库中，创建目标表，用于存储从OBS导入的数据。

目标表的表结构和OBS上将要导入的数据源文件的字段要保持一一对应，即字段个数、字段类型要一致。并且目标表和创建的外表的表结构也要保持一致，字段名称可以不一样。

步骤2 （可选）如果导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引。在数据导入完成后，再重新创建索引。

步骤3 执行数据导入。

```
INSERT INTO [目标表名] SELECT * FROM [foreign table 表名];
```

- 如果出现以下类似信息，说明数据导入成功。请查询错误信息表，查看是否存在数据格式错误，详细操作请参见**处理导入错误**。

  ```
  INSERT 0 20
  ```

- 如果出现数据加载错误，请参见**处理导入错误**，并重新执行数据导入。

----结束

示例

创建一个名为product_info的目标表，示例如下：

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

执行以下命令将外表*product_info*_ext的数据导入到目标表*product_info*中：

```
INSERT INTO product_info SELECT * FROM product_info_ext;
```

2.3.1.2.5 处理导入错误

操作场景

当数据导入发生错误时，请根据本文指引信息进行处理，错误表只能记录数据格式错误。

查询错误信息

数据导入过程中发生的错误，一般分为数据格式错误和非数据格式错误。

版权所有 © 华为云计算技术有限公司

- 数据格式错误

  在创建外表时，通过设置参数“LOG INTO error_table_name”，将数据导入过程中出现的数据格式错误信息写入指定的错误信息表error_table_name中。您可以通过以下SQL，查询详细错误信息。

  ```
  SELECT * FROM error_table_name;
  ```

  错误信息表结构如表2-9所示。

  表 2-9 错误信息表

列名称	类型	描述
nodeid	integer	报错节点编号。
begintime	timestamp with time zone	出现数据格式错误的时间。
filename	character varying	出现数据格式错误的数据源文件名。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。
detail	text	详细错误信息。

- 非数据格式错误

  对于非数据格式错误，一旦发生将导致整个数据导入失败，不会写入错误表。您可以根据执行数据导入过程中，界面提示的错误信息，帮助定位问题，处理错误表。

处理数据导入错误

根据获取的错误信息，请对照下表，处理数据导入错误。

版权所有 © 华为云计算技术有限公司

表 2-10 处理数据导入错误

错误信息	错误类型	原因	解决办法
missing data for column "r_reason_desc"	格式错误	1. 数据源文件中的列数比外表定义的列数少。 2. 对于TEXT格式的数据源文件，由于转义字符（\）导致delimiter（分隔符）错位或者quote（引号字符）错位造成的错误。 **示例：**目标表存在3列字段，导入的数据如下所示。由于存在转义字符“\”，分隔符“|”被转义为第二个字段的字段值，导致第三个字段值缺失。 `BE|Belgium\\|1`	1. 由于列数少导致的报错，选择下列办法解决： ● 在数据源文件中，增加列“r_reason_desc”的字段值。 ● 在创建外表时，将参数“fill_missing_fields”设置为“on”。即当导入过程中，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 2. 对由于转义字符导致的错误，需检查报错的行中是否含有转义字符（\）。若存在，建议在创建外表时，将参数“noescaping”（是否不对'\'和后面的字符进行转义）设置为true。
extra data after last expected column	格式错误	数据源文件中的列数比外表定义的列数多。	● 在数据源文件中，删除多余的字段值。 ● 在创建外表时，将参数“ignore_extra_data”设置为“on”。即在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。
invalid input syntax for type numeric: "a"	格式错误	数据类型错误。	在数据源文件中，修改输入字段的数据类型。根据此错误信息，请将输入的数据类型修改为numeric。
null value in column "staff_id" violates not-null constraint	非格式错误	非空约束。	在数据源文件中，增加非空字段信息。根据此错误信息，请增加“staff_id”列的值。

版权所有 © 华为云计算技术有限公司

错误信息	错误类型	原因	解决办法
duplicate key value violates unique constraint "reg_id_pk"	非格式错误	唯一约束。	• 删除数据源文件中重复的行。 • 通过设置关键字“DISTINCT”，从SELECT结果集中删除重复的行，保证导入的每一行都是唯一的。 **INSERT INTO** reasons **SELECT DISTINCT * FROM** foreign_tpcds_reasons;
value too long for type character varying(16)	格式错误	字段值长度超过限制。	在数据源文件中，修改字段值长度。根据此错误信息，字段值长度限制为VARCHAR2(16)。

2.3.1.3 从 OBS 导入 ORC、CARBONDATA、PARQUET 和 JSON 数据

2.3.1.3.1 OBS 上的数据准备

操作场景

使用SQL on OBS功能查询OBS数据之前：

1. 已将ORC数据存储在OBS上。

   例如，在使用Hive或Spark等组件时创建了ORC表，其表数据已经存储在OBS上的场景。

   假设有2个ORC数据文件“product_info.0”和“product_info.1”，其原始数据如**原始数据**所示，都已经存储在OBS桶“mybucket”的“demo.db/product_info_orc/”目录中。
2. 如果数据文件已经在OBS上了，请执行**获取源数据的OBS路径并设置读取权限**中的步骤。

   说明

   本小节以导入ORC格式为例，CARBONDATA、PARQUET和JSON数据的导入方法与ORC格式相似。

原始数据

假设您已将2个ORC数据文件存储在OBS上，其原始数据分别如下：

- 数据文件“product_info.0”

  示例数据如下所示：

  ```
  100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt Women,red,M,328,2017-09-04,715,good!
  ```


版权所有 © 华为云计算技术有限公司

```
205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear
Women,pink,L,584,2017-09-05,406,very good!
300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt men,red,XL,1245,2017-09-03,502,Bad.
310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket women,red,L,411,2017-09-05,436,It's
really super nice.
150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans
Women,blue,M,1223,2017-09-06,1200,The seller's packaging is exquisite.
```

- 数据文件“product_info.1”

  示例数据如下所示：

```
200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants
men,black,L,997,2017-09-10,301,The clothes are of good quality.
250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress
women,black,S,841,2017-09-15,299,Follow the store for a long time.
108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress women,red,M,85,2017-09-14,22,It's
really amazing to buy.
450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket
women,white,M,114,2017-09-14,22,Open the package and the clothes have no odor.
260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat
women,red,L,2004,2017-09-15,826,Very favorite clothes.
```

获取源数据的 OBS 路径并设置读取权限

步骤1 登录OBS控制台。

单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。

步骤2 获取数据源文件的OBS路径。

数据源文件在上传到OBS桶之后，会生成全局唯一的访问路径。在创建外表时需要指定数据源文件的OBS路径。

如何查看OBS路径，请参见《对象存储服务用户指南》中的“控制台指南 > 管理对象 > 通过对象URL访问对象”章节。

例如，在本例中，查看到数据文件的OBS路径分别为：

```
https://obs.example.com/mybucket/demo.db/product_info_orc/product_info.0
https://obs.example.com/mybucket/demo.db/product_info_orc/product_info.1
```

步骤3 为用户设置OBS桶的读取权限。

在使用SQL on OBS功能时，执行该功能的用户需要取得数据源文件所在OBS桶的读取权限。通过配置桶的ACL权限，可以将读取权限授予指定的用户账号。

具体请参见《对象存储服务用户指南》中的“控制台指南 > 权限控制 > 配置桶ACL”章节。

----结束

2.3.1.3.2 创建外部服务器

创建外部服务器，用于定义OBS服务器的信息，供外表调用，针对集群不同版本，创建方式不同。

- 8.2.0及以上集群版本：通过管理控制台创建OBS数据源实现以委托方式访问OBS桶数据。
- 8.2.0以下集群版本：通过手动创建OBS服务器实现。创建外部服务器的详细语法，请参见《数据仓库服务SQL语法参考》中“CREATE SERVER”章节。

版权所有 © 华为云计算技术有限公司

8.2.0 及以上集群版本

参见《 数据仓库服务用户指南 》的“管理OBS数据源”相关章节完成OBS服务器创建。

8.2.0 以下版本：（ 可选 ）新建用户及数据库并授予外表权限

如果您将使用普通用户在自定义数据库中创建外部服务器和外表，由于普通用户没有外表权限无法创建，所以，您必须参照以下步骤新建用户和数据库，并授予该用户外表权限。

以下示例，是新建一个普通用户dbuser并创建一个数据库mydatabase，然后使用管理员用户授予dbuser外表权限。

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库gaussdb。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d gaussdb -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
```

步骤2 新建一个普通用户，并用它创建一个数据库。

新建一个具有创建数据库权限的用户dbuser：

```
CREATE USER dbuser WITH CREATEDB PASSWORD 'password';
```

切换为新建的用户：

```
SET ROLE dbuser PASSWORD 'password';
```

执行以下命令创建数据库：

```
CREATE DATABASE mydatabase;
```

查询数据库：

```
SELECT * FROM pg_database;
```

返回结果中有**mydatabase**的信息表示创建成功：

```
datname   | datdba | encoding | datcollate | datctype | datistemplate | datallowconn | datconnlimit |
datlastsysoid | datfrozenxid | dattablespace | datcompatibility |                    datacl

------------+--------+----------+------------+----------+---------------+--------------+--------------+---------------
+---------------+---------------+------------------+--------------------------------------
--------------
 template1  |     10 |        0 | C          | C        | t             | t            |           -1 |        14146 |        1351
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 template0  |     10 |        0 | C          | C        | t             | f            |           -1 |        14146 |        1350
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 gaussdb    |     10 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1352 |
1663 | ORA             | {=Tc/Ruby,Ruby=CTc/Ruby,chaojun=C/Ruby,hu
obinru=C/Ruby}
 mydatabase |  17000 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1351
|          1663 | ORA             |
(4 rows)
```

步骤3 使用管理员用户给普通用户赋予创建外部服务器的权限和使用外表的权限。

使用数据库管理员用户通过DWS提供的数据库客户端连接新建的数据库。

例如，使用gsql客户端的用户可以直接使用如下语句切换为管理员用户去连接新建的数据库：

```
\c mydatabase dbadmin;
```

根据提示输入管理员用户密码。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

必须先使用管理员用户连接到**将要创建外部服务器和使用外表的数据库**，再对普通用户进行授权。

默认只有系统管理员才可以创建外部服务器，普通用户需要授权才可以创建，执行以下命令授权：

```
GRANT ALL ON SCHEMA public TO dbuser;
GRANT ALL ON FOREIGN DATA WRAPPER dfs_fdw TO dbuser;
```

其中fdw_name的名字可以是hdfs_fdw或者dfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令赋予用户使用外表的权限。

```
ALTER USER dbuser USEFT;
```

查看用户：

```
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvalidbegin, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolauditadmin
, r.rolsystemadmin
, r.roluseft
FROM pg_catalog.pg_roles r
ORDER BY 1;
```

返回结果中dbuser的信息中包含了UseFT权限，表示授权成功：

```
rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcanlogin | rolconnlimit | rolvalidbegin |
rolvaliduntil | memberof | rolreplication | rolauditadmin | rolsystemadmin | roluseft
-----------+----------+------------+---------------+-------------+-------------+--------------+---------------
+---------------+----------+----------------+---------------+----------------+----------
 dbuser    | f        | t          | f             | t           | t           |           -1 |               |           | {}       | f
| f           | f           | t
 lily      | f        | t          | f             | f           | t           |           -1 |               |           | {}       | f           |
f           | f           | f
 Ruby      | t        | t          | t             | t           | t           |           -1 |               |           | {}       | t
| t           | t           | t
```

----结束

8.2.0 以下版本：创建外部服务器

步骤1 使用即将创建外部服务器的用户去连接其对应的数据库。

在本示例中，将使用**8.2.0以下版本：（可选）新建用户及数据库并授予外表权限**中创建的普通用户dbuser连接其创建的数据库mydatabase 。用户需要通过DWS提供的数据库客户端连接数据库。

例如，使用gsql客户端的用户可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令切换数据库和用户：
  ```
  \c mydatabase dbuser;
  ```
  根据提示输入密码。

版权所有 © 华为云计算技术有限公司

- 如果尚未登录gsql客户端，或者已经登录了gsql客户端执行**\q**退出gsql后，执行以下命令重新进行连接：
  ```
  gsql -d mydatabase -h 192.168.2.30 -U dbuser -p 8000 -r
  ```
  根据提示输入密码。

步骤2 创建外部服务器。

执行以下命令创建外部服务器'obs_server'：

说明

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE SERVER obs_server FOREIGN DATA WRAPPER dfs_fdw
OPTIONS (
  address 'obs.example.com' ,
  ACCESS_KEY 'access_key_value_to_be_replaced',
  SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
  encrypt 'on',
  type 'obs'
);
```

以下为必选参数的说明：

- **外部服务器名称**
  允许用户自定义名字。
  在本例中指定为“obs_server” 。
- **FOREIGN DATA WRAPPER**
  fdw_name的名字可以是hdfs_fdw或者dfs_fdw，它在数据库中已经存在。
- **OPTIONS参数**
  - **address**
    指定OBS服务的终端节点。
    address的获取方法如下：
    i. 先通过**OBS上的数据准备**中的**步骤2**获取OBS路径。
    ii. 在OBS上查看到的OBS路径，为OBS服务终端节点（Endpoint）：obs.example.com。
  - **访问密钥（AK和SK）**（必选）
    DWS需要通过访问密钥（AK和SK）访问OBS，因此，必须先获取访问密钥。
    - “access_key”（必选）：表示用户的AK信息。
    - “secret_access_key”（必选）：表示用户的SK信息。

    获取访问密钥的具体步骤，请参见**创建访问密钥（AK和SK）**。
  - **type**
    取值为'obs'，表示dfs_fdw连接的是OBS。

步骤3 查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE srvname='obs_server';
```

返回结果如下所示，表示已经创建成功：

```
 srvname   | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                                    srvoptions
```

 版权所有 © 华为云计算技术有限公司

```
------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------
 obs_server |    24661 |  13686 |         |            |        |
{address=xxx.xxx.x.xxx,access_key=xxxxxxxxxxxxxxxxxxxx,type=obs,secret_access_key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}
(1 row)
```

----结束

2.3.1.3.3 创建外表

当完成**创建外部服务器**后，在DWS数据库中创建一个OBS外表，用来访问存储在OBS上的数据。OBS外表是只读的，只能用于查询操作，可直接使用SELECT查询其数据。根据集群版本不同，操作有所区别。

- 8.2.0及以上版本：参见《数据仓库服务用户指南》的“管理OBS数据源”相关章节完成。
- 8.2.0以前版本，参见以下步骤完成。

⚠ 注意

用户创建外表时如果报错“permission denied for foreign server xxx”，说明当前用户没有外部服务器权限，需要执行以下命令进行授权（obs_server和u1，请替换为实际的外部服务器名称和当前用户名称）：

```
GRANT usage ON foreign server obs_server TO u1;
```

创建外表

创建外表的语法格式如下。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER dfs_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为“*product_info_ext_obs*”的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  外表的表名。
- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。

  外表的字段个数和字段类型，需要与OBS上保存的数据完全一致。
- **SERVER dfs_server**

版权所有 © 华为云计算技术有限公司

外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器连接OBS读取数据。

此处应填写为参照**创建外部服务器**创建的外部服务器名称。

- **OPTIONS参数**

  用于指定外表数据的各类参数，关键参数如下所示。

  - “format”：表示对应的OBS服务上的文件格式，支持“orc”、“carbondata”和“parquet”格式。
  - “foldername”：必选参数。数据源文件的OBS路径，此处仅需要填写“*/桶名/文件夹目录层级/*”。

    可以先通过**OBS上的数据准备**中的**步骤2**获取数据源文件的完整的OBS路径，该路径为OBS服务的终端节点（Endpoint）。
  - “totalrows”：可选参数。该参数不是导入的总行数。由于OBS上文件可能很多，执行analyze可能会很慢，通过“totalrows”参数，让用户来设置一个预估的值，使优化器能通过这个值做大小表的估计。一般预估值与实际值的数量级差不多时，查询效率较高。
  - **“encoding”**：外表中数据源文件的编码格式名称，缺省为utf8。对于OBS外表此参数为必选项。

- **DISTRIBUTE BY**：

  这个子句是必须的，当前支持**ROUNDROBIN**和**REPLICATION**分布方式。缺省为**ROUNDROBIN**分布方式。

  ROUNDROBIN分布方式表示外表在从数据源读取数据时，DWS集群每一个节点随机读取一部分数据，并组成完整数据。

  REPLICATION分布方式表示外表在从数据源读取数据时，DWS集群选取一个节点读取全部数据。因为每个数据节点都有完整的表数据。

- **语法中的其他参数**

  其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中不需要设置。

根据以上信息，创建外表命令如下所示：

建立不包含分区列的OBS外表，表关联的外部服务器为obs_server，表对应的OBS服务上的文件格式为‘orc’，OBS上的数据存储路径为'/mybucket/demo.db/product_info_orc/'。

```
DROP FOREIGN TABLE IF EXISTS product_info_ext_obs;
CREATE FOREIGN TABLE product_info_ext_obs
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER obs_server
OPTIONS (
format 'orc',
foldername '/mybucket/demo.db/product_info_orc/',
encoding 'utf8',
totalrows '10'
)
DISTRIBUTE BY ROUNDROBIN;
```


版权所有 © 华为云计算技术有限公司

建立包含分区列的OBS外表，product_info_ext_obs外表使用product_manufacturer字段作为分区键，obs/mybucket/demo.db/product_info_orc/路径下有如下分区目录：

分区目录1：product_manufacturer=10001

分区目录2：product_manufacturer=10010

分区目录3：product_manufacturer=10086

...

```
DROP FOREIGN TABLE IF EXISTS product_info_ext_obs;
CREATE FOREIGN TABLE product_info_ext_obs
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)   ,
    product_manufacturer     integer
) SERVER obs_server
OPTIONS (
format 'orc',
foldername '/mybucket/demo.db/product_info_orc/',
encoding 'utf8',
totalrows '10'
)
DISTRIBUTE BY ROUNDROBIN
PARTITION BY (product_manufacturer) AUTOMAPPED;
```

2.3.1.3.4 通过外表查询 OBS 上的数据

直接查询外表查看 OBS 上的数据

如果数据量较少，可直接使用SELECT查询外表，即可查看到OBS上的数据。

步骤1 执行以下命令，则可以从外表查询数据。

```
SELECT * FROM product_info_ext_obs;
```

查询结果显示与**原始数据**显示相同，则表示导入成功。查询结果的结尾将显示以下信息：

```
(10 rows)
```

通过外表查询到数据后，用户可以将数据插入数据库的普通表。

----结束

导入数据后查询数据

步骤1 在DWS数据库中，创建导入数据的目标表，用于存储导入的数据。

该表的表结构必须与**创建外表**中创建的外表的表结构保持一致，即字段个数、字段类型要完全一致。

例如，创建一个名为product_info的表，示例如下：

```
DROP TABLE IF EXISTS product_info;
```


版权所有 © 华为云计算技术有限公司

```
CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

步骤2 执行“INSERT INTO .. SELECT ..”命令从外表导入数据到目标表。

示例：

```
INSERT INTO product_info SELECT * FROM product_info_ext_obs;
```

若出现以下类似信息，说明数据导入成功。

```
INSERT 0 10
```

步骤3 执行SELECT命令，查看从OBS导入到DWS中的数据。

```
SELECT * FROM product_info;
```

查询结果显示如**原始数据**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(10 rows)
```

----结束

2.3.1.3.5 清除资源

当完成本教程的示例后，如果不再需要使用本示例中创建的资源，可以删除这些资源，以免资源浪费或占用配额。步骤如下：

1. **删除外表和目标表**
2. **删除创建的外部服务器**
3. **删除数据库及其所属的用户**

   如果您执行了**8.2.0以下版本：（可选）新建用户及数据库并授予外表权限**中的步骤，请删除数据库及其所属的用户。

删除外表和目标表

步骤1 （可选）如果执行了**导入数据后查询数据**，请执行以下命令，删除目标表。

```
DROP TABLE product_info;
```

当结果显示为如下信息，则表示删除成功。

```
DROP TABLE
```

步骤2 执行以下命令，删除外表。

```
DROP FOREIGN TABLE product_info_ext_obs;
```

当结果显示为如下信息，则表示删除成功。

版权所有 © 华为云计算技术有限公司

```
DROP FOREIGN TABLE
```

----结束

删除创建的外部服务器

步骤1 使用创建外部服务器的用户连接到外部服务器所在的数据库。

在本示例中，使用的是普通用户dbuser在数据库mydatabase中创建了一个外部服务器 。用户需要通过DWS提供的数据库客户端连接数据库。例如，使用gsql客户端的用户，可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令进行切换：
  \c *mydatabase dbuser*;
  根据提示输入密码。
- 如果已经登录了gsql客户端，您也可以执行**\q**退出gsql后，再执行以下命令重新进行连接：
  gsql -d *mydatabase* -h *192.168.2.30* -U *dbuser* -p 8000 -r
  根据提示输入密码。

步骤2 删除创建的外部服务器。

执行以下命令进行删除。

DROP SERVER *obs_server*;

返回以下信息表示删除成功：

```
DROP SERVER
```

查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE srvname='obs_server';
```

返回结果如下所示，表示已经删除成功：

```
 srvname | srvowner | srvfdw | srvtype | srvversion | srvacl | srvoptions
---------+----------+--------+---------+------------+--------+------------
(0 rows)
```

----结束

删除数据库及其所属的用户

如果您执行了**8.2.0以下版本：（可选）新建用户及数据库并授予外表权限**中的步骤，请参照以下步骤删除数据库及其所属的用户。

步骤1 删除自定义数据库。

通过DWS提供的数据库客户端连接默认数据库gaussdb。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

先切换到默认数据库：

```
\c gaussdb
```

根据界面提示输入密码。

执行以下命令，删除自定义数据库：

DROP DATABASE *mydatabase*;

版权所有 © 华为云计算技术有限公司

返回以下信息表示删除成功：

```
DROP DATABASE
```

步骤2 使用管理员用户，删除本示例中创建的普通用户。

使用数据库管理员用户通过DWS提供的数据库客户端连接数据库。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

```
\c gaussdb dbadmin
```

执行以下命令回收创建外部服务器的权限：

```
REVOKE ALL ON FOREIGN DATA WRAPPER dfs_fdw FROM dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是dfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令删除用户：

```
DROP USER dbuser;
```

可使用**\du**命令查询用户，确认用户是否已经删除。

----结束

2.3.1.3.6 支持的数据类型

目前大数据领域，主流文件格式为ORC、PARQUET。用户利用HIVE将数据导出存储为ORC、PARQUET文件格式，使用DWS通过只读外表对ORC、PARQUET文件内的数据进行查询分析，因此，需要在ORC、PARQUET文件格式支持的数据类型与DWS自身支持数据类型间进行匹配，匹配状况如**表2-11**所示。同理，DWS可通过只写外表将数据导出存储为ORC、PARQUET文件格式，使用HIVE读取ORC、PARQUET文件内容，相互之间也需要类型匹配，匹配状况如**表2-12**所示：

表 2-11 ORC、PARQUET 格式的只读外表与 HIVE 数据类型匹配关系

类型名称	DWS外表支持类型	HIVE建表类型
1字节整数	TINYINT（不推荐）	TINYINT
	SMALLINT（推荐）	TINYINT
2字节整数	SMALLINT	SMALLINT
4字节整数	INTEGER	INT
8字节整数	BIGINT	BIGINT
单精度浮点数	FLOAT4 (REAL)	FLOAT
双精度浮点型	FLOAT8(DOUBLE PRECISION)	DOUBLE
科学数据类型	DECIMAL[p (,s)] 最大支持38位精度	DECIMAL最大支持38位（HIVE 0.11）
日期类型	DATE	DATE
时间类型	TIMESTAMP	TIMESTAMP

版权所有 © 华为云计算技术有限公司

类型名称	DWS外表支持类型	HIVE建表类型
BOOLEAN类型	BOOLEAN	BOOLEAN
Char类型	CHAR(n)	CHAR (n)
VarChar类型	VARCHAR(n)	VARCHAR (n)
字符串（文本大对象）	TEXT(CLOB)	STRING
二进制类型	BYTEA	BINARY

说明

二进制类型仅9.1.0.100及以上集群版本支持。

表 2-12 ORC、PARQUET 格式的只写外表与 HIVE 数据类型匹配关系

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
1字节整数	TINYINT	TINYINT（不推荐）	SMALLINT
		SMALLINT（推荐）	SMALLINT
2字节整数	SMALLINT	SMALLINT	SMALLINT
4字节整数	INTEGER、BINARY_INTEGER	INTEGER	INT
8字节整数	BIGINT	BIGINT	BIGINT
单精度浮点数	FLOAT4 、REAL	FLOAT4、REAL	FLOAT
双精度浮点型	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE
科学数据类型	DECIMAL、NUMERIC	DECIMAL[p (,s)] 最大支持38位精度	precision <=38时，DECIMAL, precision > 38时， STRING
日期类型	DATE	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP

版权所有 © 华为云计算技术有限公司

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
时间类型	TIME [(p)] [WITHOUT TIME ZONE]、TIME [(p)] [WITH TIME ZONE]	TEXT	STRING
	TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)][WITH TIME ZONE]、SMALLDATETIME	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP
	INTERVAL DAY (l) TO SECOND (p)、INTERVAL [FIELDS] [(p)]	VARCHAR(n)	VARCHAR(n)
BOOLEAN类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)、CHARACTER(n)、NCHAR(n)	CHAR(n)、CHARACTER(n)、NCHAR(n)	n<=255时，CHAR(n)，n>255 时，STRING
VarChar类型	VARCHAR(n)、CHARACTER VARYING(n)、VARCHAR2(n)、	VARCHAR(n)	n<=65535时，VARCHAR(n)，n>65535时，STRING
	NVARCHAR2(n)	TEXT	STRING
字符串（文本大对象）	TEXT、CLOB	TEXT、CLOB	STRING
二进制类型	BYTEA	BYTEA	BINARY
货币类型	MONEY	NUMERIC	BIGINT

版权所有 © 华为云计算技术有限公司

须知

1. DWS外表支持NULL定义，HIVE数据表支持并采用相对应的NULL定义。
2. HIVE数据表中的TINYINT的取值范围为[-128,127]，而 DWS 的TINYINT的取值范围为[0,255]，因此，HIVE表中的TINYINT类型在建DWS只读外表时建议采用SMALLINT类型，如果使用TINYINT有可能存在读取值与实际值不一致的情况。同样，DWS的TINYINT类型在导出时，只写外表和HIVE的建表类型也建议采用SMALLINT类型。
3. DWS外表的日期和时间类型，不支持时区定义，HIVE不支持时区定义。
4. HIVE中DATE类型只有日期，没有时间，DWS的DATE类型包含日期和时间。
5. DWS支持ORC的压缩格式，包括ZLIB、SNAPPY、LZ4及NONE压缩方式。其中FLOAT4格式本身存在不精准问题，求和等操作在不同环境下可能产生不同的结果，在高精度要求场景下建议使用DECIMAL类型代替。
6. 兼容Teradata数据库模式下，外表不支持DATE类型。
7. 二进制类型仅9.1.0.100及以上集群版本支持。

2.3.2 使用 GDS 从远端服务器导入数据

2.3.2.1 关于 GDS 并行导入

INSERT和COPY方式执行数据导入时，是一个串行执行的过程，导入性能低，因此适用于小数据量的导入。对于大数据量的导入，DWS支持使用GDS工具通过外表并行导入数据到集群。

当前版本的GDS已经支持从管道文件导入数据库，该功能使GDS的导入更加灵活多变。

- 当GDS用户的本地磁盘空间不足时，可直接将hdfs上的数据写入到管道文件而不需要占用额外的磁盘空间。
- 当用户导入前需要清洗数据时，用户可以根据自己的需求编写程序，将需要处理的数据流式实时的写入管道文件，完成导入的数据清洗工作。

  说明

  - 当前版本暂不支持SSL模式下GDS导入，请勿以SSL方式使用GDS。
  - 本章涉及的所有管道文件都是指Linux上的命名管道。
  - 在执行GDS导入导出时，为了确保数据导入或导出的正确性，需要在相同的兼容模式下操作。

    例如：在MySQL兼容模式下导入（导出）的数据，同样需要在MySQL兼容模式下才能正确导出（导入）。

概述

并行导入将存储在服务器普通文件系统中的数据导入到DWS数据库中。暂时不支持将存储在HDFS文件系统上的数据导入DWS。

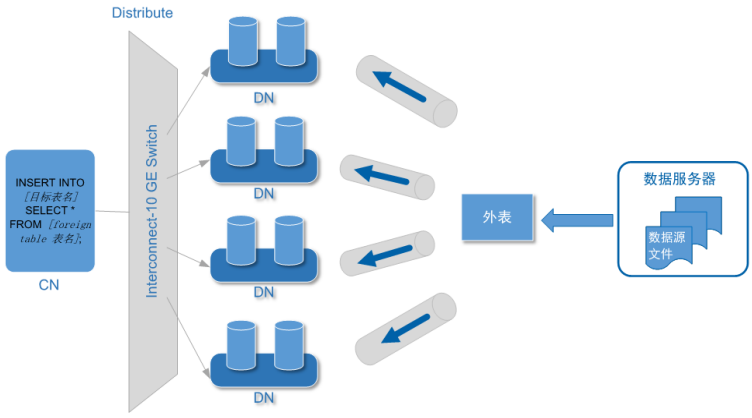
并行导入功能通过外表设置的导入策略、导入数据格式等信息来识别数据源文件，利用多DN并行的方式，将数据从数据源文件导入到数据库中，从而提高整体导入性能。如图2-3所示：

版权所有 © 华为云计算技术有限公司

- CN只负责任务的规划及下发，把数据导入的工作交给了DN，释放了CN的资源，使其有能力处理其他外部请求。
- 所有DN都参与数据导入，这样可以充分利用各设备的计算能力及网络带宽，提升导入效率。

外表灵活的OPTION设置，有利于在数据入库前对数据做预处理，例如非法字符替换、容错处理等。

图 2-3 数据并行导入示意图



上图中所涉及的相关概念说明如下：

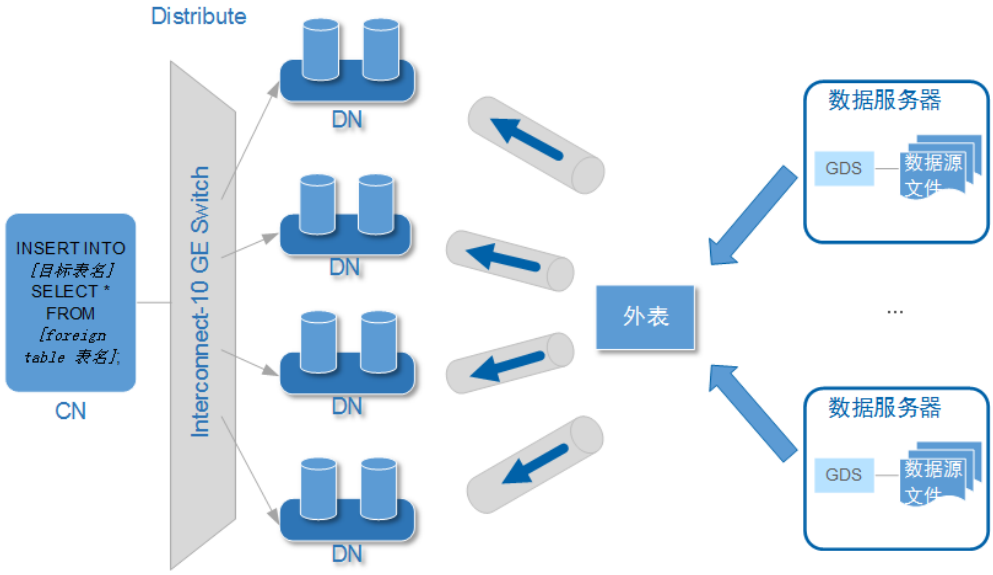
- **CN（Coordinator）**：DWS协调节点。在导入场景下，接收到应用或客户端的导入SQL指令后，负责任务的规划及下发到DN。
- **DN（Datanode）**：DWS数据节点。接收CN下发的导入任务，将数据源文件中的数据通过外表写入数据库目标表中。
- **数据源文件**：存有数据的文件。文件中保存的是待导入数据库的数据。
- **数据服务器**：数据源文件所在的服务器称为数据服务器。基于安全考虑，建议数据服务器和DWS集群处于同一内网。
- **外表Foreign Table**：用于识别数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。是关联数据文件与数据库实表（目标表）的对象。
- **目标表**：数据库中的实表。数据源文件中的数据最终导入到这些表中存储，包括行存表和列存表。

GDS 并发导入

- 数据量大，数据存储在多个服务器上时，在每个数据服务器上安装配置、启动GDS后，各服务器上的数据可以并行入库。如图2-4所示。

版权所有 © 华为云计算技术有限公司

图 2-4 多数据服务器并行导入



须知

GDS进程数目不能超过DN数目。如果超过，会出现一个DN连接多个GDS进程的情形，可能会导致部分GDS异常运行。

- 数据存储在一台数据服务器上时，如果DWS及数据服务器上的I/O资源均还有可利用空间时，可以采用GDS多线程来支持并发导入。

  GDS是根据导入事务并发数来决定服务运行线程数的，也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。

综上，多线程的使用场景如下：

- 多表并发导入时，采用多线程充分利用资源及提升并发导入效率。
- 对数据量大的某一事实表的导入进行提速。

  将该事实表对应的数据拆分为多个数据文件，通过多外表同时入库的方式实现多线程并发导入。注意需确保每个外表所能读取的数据文件不重复。

版权所有 © 华为云计算技术有限公司

导入流程

图 2-5 GDS 并行导入流程

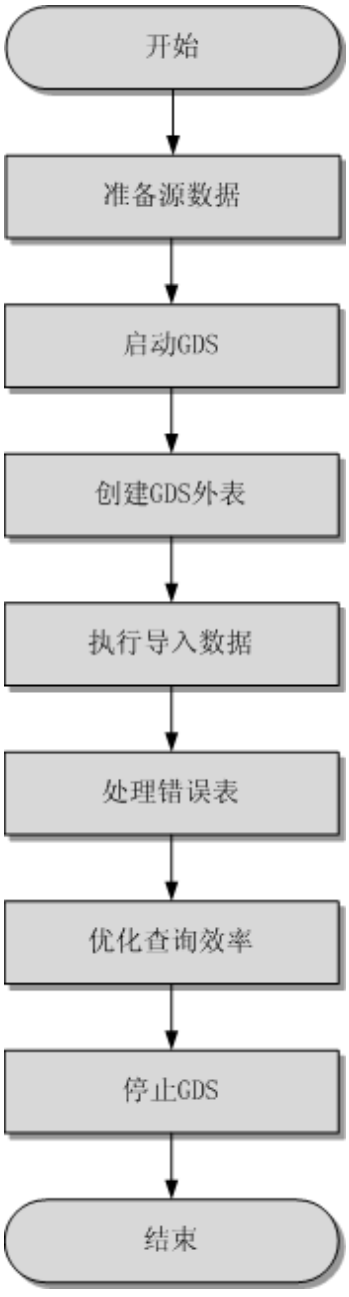


表 2-13 流程说明

流程	说明
准备源数据。	准备需要导入数据库的源数据文件，并上传至数据服务器。 详细内容请参见**准备源数据**。
启动GDS。	在数据服务器上安装配置并启动GDS。 详细内容请参见**安装配置和启动GDS**。

版权所有 © 华为云计算技术有限公司

流程	说明
创建外表。	创建外表用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。 详细内容请参见**创建GDS外表**。
执行导入数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导入到目标表中。详细内容请参见**执行导入数据**。
处理错误表。	在数据并行导入发生错误时，请根据具体的错误信息进行处理，以保证导入数据的完整性。 详细内容请参见**处理导入错误**。
优化查询效率。	导入数据后，通过ANALYZE语句生成表统计信息。ANALYZE语句会将统计结果自动存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以生成最有效的查询执行计划。
停止GDS	待数据导入完成后，登录每台数据服务器，分别停止GDS。 GDS的停止请参见**停止GDS**。

2.3.2.2 准备源数据

操作场景

通常在将数据导入数据库前，即将入库的数据已经在相关主机上了。这种保存着待入库数据的服务器为数据服务器。此时，只需检测以确认数据服务器和DWS集群能够正常通信，并查看和记录数据在数据服务器上的存放目录备用。

如果待入库数据还没有就绪，则请先参考如下步骤，将数据上传到数据服务器上。

操作步骤

步骤1 以root用户登录数据服务器。

步骤2 创建数据文件存放目录“/input_data”。

```
mkdir -p /input_data
```

步骤3 将数据源文件上传至上一步所创建的目录中。

GDS并行导入支持CSV、TEXT格式的数据导入。请确保数据源文件符合格式要求。

----结束

2.3.2.3 安装配置和启动 GDS

操作场景

DWS提供了数据服务工具GDS来帮助分发待导入的用户数据及实现数据的高速导入。GDS需部署到数据服务器上。

版权所有 © 华为云计算技术有限公司

数据量大，数据存储在多个服务器上时，在每个数据服务器上安装配置、启动GDS后，各服务器上的数据可以并行入库。GDS在各台数据服务器上的安装配置和启动方法相同，本节以一台服务器为例进行说明。

背景信息

GDS的版本需与集群版本保持一致（如：GDS V100R008C00版本与DWS 1.3.X版本配套），否则可能会出现导入导出失败或导入导出进程停止响应等情况。因此请勿使用历史版本的GDS进行导入。

数据库版本升级后，请按照**操作步骤**中的办法下载DWS软件包解压缩自带的GDS进行安装配置和启动。在导入导出开始时，DWS也会进行两端的版本一致性检测，不一致时会在屏幕上显示报错信息并终止对应操作。

GDS的版本号的查看办法为：在GDS工具的解压目录下执行如下命令。

```
gds -V
```

数据库版本的查看办法为：连接数据库后，执行如下SQL命令查看。

```
SELECT version();
```

操作步骤

步骤1 以root用户登录待安装GDS的数据服务器，创建存放GDS工具包的目录。

```
mkdir -p /opt/bin/dws
```

步骤2 将GDS工具包上传至上一步所创建的目录中。

以上传SUSE Linux版本的工具包为例 ，将GDS工具包“dws_client_8.x.x_suse_x64.zip”上传至上一步所创建的目录中。

步骤3 （可选）如果使用SSL加密传输，请一并上传SSL证书至**步骤1**所创建的目录下。

步骤4 在工具包所在目录下，解压工具包。

```
cd /opt/bin/dws
unzip dws_client_8.x.x_suse_x64.zip
```

步骤5 创建GDS专有用户及其所属的用户组。此用户用于启动GDS及读取源数据。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

步骤6 分别修改工具包和数据源文件目录属主为GDS专有用户。

```
chown -R gds_user:gdsgrp /opt/bin/dws/gds
chown -R gds_user:gdsgrp /input_data
```

步骤7 切换到gds_user用户。

```
su - gds_user
```

若当前集群版本为8.0.x及以前版本，请跳过**步骤8**，直接执行**步骤9**。

若当前集群版本为8.1.x版本，则正常执行以下步骤。

步骤8 执行环境依赖脚本（仅8.1.x版本适用）。

```
cd /opt/bin/dws/gds/bin
source gds_env
```

步骤9 启动GDS服务。

GDS是绿色软件，解压后启动即可。GDS启动方式有两种：

 版权所有 © 华为云计算技术有限公司

方式一：直接使用“gds”命令，在命令项中设置启动参数。

方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

对于集中一次性导入的场景推荐使用第一种方式。对于需要隔段时间再次导入的场景，推荐使用第二种方式以配置文件的形式提升启动效率。

- 方式一：直接使用“gds”命令，启动GDS。
  - 非SSL模式传输数据的情况下，启动GDS。
    gds -d *dir* **-p** *ip:port* **-H** *address_string* **-l** *log_file* **-D -t** *worker_num*

    示例：

    /opt/bin/dws/gds/bin/gds -d */input_data/* **-p** *192.168.0.90:5000* **-H** *10.10.0.1/24* **-l** */opt/bin/dws/gds/gds_log.txt* **-D -t** *2*
  - 使用SSL加密方式传输数据的情况下，启动GDS。
    gds -d *dir* **-p** *ip:port* **-H** *address_string* **-l** *log_file* **-D -t** *worker_num* **--enable-ssl --ssl-dir** *Cert_file*

    示例：

    以**步骤3**中SSL证书已上传至/opt/bin为例，命令如下。
    /opt/bin/dws/gds/bin/gds -d */input_data/* **-p** *192.168.0.90:5000* **-H** *10.10.0.1/24* **-l** */opt/bin/dws/gds/gds_log.txt* **-D --enable-ssl --ssl-dir** */opt/bin/*

  命令中的斜体部分请根据实际替换。
  - **-d** *dir*：保存有待导入数据的数据文件所在目录。本教程中为“/input_data/”。
  - **-p** *ip:port*：GDS监听IP和监听端口。默认值为：127.0.0.1，需要替换为能跟DWS通信的万兆网IP。监听端口的取值范围：1024~65535。默认值为：8098。本教程配置为：192.168.0.90:5000。
  - **-H** *address_string*：允许哪些主机连接和使用GDS服务。参数需为CIDR格式。此参数配置的目的是允许DWS集群可以访问GDS服务进行数据导入。所以请保证所配置的网段包含DWS集群各主机。
  - **-l** *log_file*：存放GDS的日志文件路径及文件名。本教程为“/opt/bin/dws/gds/gds_log.txt”。
  - **-D**：后台运行GDS。仅支持Linux操作系统下使用。
  - **-t** *worker_num*：设置GDS并发线程数。默认值为：8。取值范围为0<worker_num<=200，正整数。DWS及数据服务器上的I/O资源均充足时，可以加大并发线程数。

    GDS是根据导入事务并发数来决定服务运行线程数的。也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。
  - **--enable-ssl**：启用SSL加密方式传输数据。
  - **--ssl-dir** *Cert_file*：SSL证书所在目录。需与**步骤3**中的证书保存目录保持一致。

  关于更多参数的设置信息请参考《数据仓库服务工具指南》中的“GDS并行数据加载工具 > gds命令简介”。
- 方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。
  a. 使用如下命令，进入GDS工具包的“config”目录下，配置“gds.conf”文件。“gds.conf”配置详细信息请参考**表2-14**。
  vim */opt/bin/dws/gds/config/***gds.conf**

  示例：

  配置“gds.conf”文件如下：

 版权所有 © 华为云计算技术有限公司

```
<?xml version="1.0"?>
<config>
<gds name="gds1" ip="192.168.0.90" port="5000" data_dir="/input_data/" err_dir="/err"
data_seg="100MB" err_seg="100MB" log_file="/log/gds_log.txt" host="10.10.0.1/24"
daemon='true' recursive="true" parallel="32"></gds>
</config>
```

配置文件信息如下：

- 数据服务器所在IP为192.168.0.90，GDS监听端口为5000。
- 数据文件存放在“/input_data/”目录下。
- 错误日志文件存放在“/err”目录下。该目录需要拥有GDS读写权限的用户自行创建。
- 单个数据文件大小为100MB。
- 每个错误日志大小为100MB。
- 日志保存在“/log/gds_log.txt”文件中。该目录需要拥有GDS读写权限的用户自行创建。
- 只允许IP为10.10.0.*的节点进行连接。
- GDS进程以后台方式运行。
- 递归数据文件目录。
- 指定并发导入工作线程数目为2。

b. 执行如下命令启动GDS并确认GDS是否启动成功。

```
python3 gds_ctl.py start
```

示例：

```
cd /opt/bin/dws/gds/bin
python3 gds_ctl.py start
Start GDS gds1                  [OK]
gds [options]:
 -d dir            Set data directory.
 -p port           Set GDS listening port.
   ip:port         Set GDS listening ip address and port.
 -l log_file       Set log file.
 -H secure_ip_range
                   Set secure IP checklist in CIDR notation. Required for GDS to start.
 -e dir            Set error log directory.
 -E size           Set size of per error log segment.(0 < size < 1TB)
 -S size           Set size of data segment.(1MB < size < 100TB)
 -t worker_num     Set number of worker thread in multi-thread mode, the upper limit is 200. If
without setting, the default value is 8.
 -s status_file    Enable GDS status report.
 -D                Run the GDS as a daemon process.
 -r                Read the working directory recursively.
 -h                Display usage.
```

----结束

 版权所有 © 华为云计算技术有限公司

gds.conf 参数说明

表 2-14 gds.conf 配置说明

属性	说明	取值范围
name	标识名。	-
ip	监听ip地址。	IP需为合法IP地址。 IP的默认值：127.0.0.1
port	监听端口号。	取值范围：1024~65535，正整数。 默认值：8098。
data_dir	数据文件目录。	-
err_dir	错误日志文件目录。	默认值：数据文件目录
log_file	日志文件路径。	-
host	设置允许连接到GDS的主机IP地址（参数为CIDR格式，仅支持linux系统）。	-
recursive	是否递归数据文件目录。选择true时会递归读取location指定的目录层级下所有的同名文件。	取值范围： ● true：递归 。 ● false：不递归。 默认值：false。
daemon	是否以DAEMON（后台）模式运行。	取值范围： ● true：以DAEMON模式运行。 ● false：不以DAEMON模式运行。 默认值：false。
parallel	导入工作线程并发数目。	取值范围：0~200，正整数。 默认值：8。

2.3.2.4 创建 GDS 外表

外表中配置了数据源格式信息、GDS服务的访问信息，从而DWS最终可以通过外表将数据服务器上的数据引流进数据库实表中。

操作步骤

步骤1 收集数据源格式信息、GDS服务的访问信息。

需要收集的主要数据源格式信息如下：

- **format**：GDS外表导入支持CSV、TEXT和FIXED格式。请确认存放在数据服务器上待入库数据的格式。例如，待入库的数据为CSV格式。
- **header（仅支持CSV，FIXED格式）**：确认数据文件是否包含标题行。

版权所有 © 华为云计算技术有限公司

- **delimiter**：确认数据文件中，字段间的分隔符。例如，以英文逗号分隔的。
- **encoding**：数据源文件的数据编码格式。例如，为UTF-8。
- **eol**：确认数据文件中，行间的换行符。例如，默认的换行符，如0x0D0A、0X0A，或者自定义的换行符，如字符串!@#。该参数仅支持TEXT格式导入。
- 外表可识别的其他更多格式信息请参见《 数据仓库服务SQL语法参考 》中“CREATE FOREIGN TABLE (GDS导入导出)”章节的数据格式参数说明。

需要收集的GDS服务的访问信息如下：

location：GDS服务的访问地址。例如以**安装配置和启动GDS**中的GDS信息为例。非SSL模式下的location为：**gsfs**://*192.168.0.90:5000*/*/input_data/* 。SSL模式下的location为：**gsfss**://*192.168.0.90:5000*/*/input_data/* 。其中，“192.168.0.90:5000”为GDS服务的IP及端口号；“input_data”为GDS服务管理的数据源文件所在的路径。请根据实际情况替换。

步骤2 依据数据源文件中的数据情况，设计导入容错机制。

DWS支持如下的数据容错性处理，相当于数据入库前对数据做初步的简单清洗。

- **fill_missing_fields**：数据入库时，数据源文件中某行的最后一个字段缺失时，请选择是直接将字段设为Null，还是在错误表中报错提示。

  说明

  取值范围：true/on，false/off。
  - 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。
  - 参数为false/off，如果最后一个字段缺失会显示如下错误信息。
    ```
    missing data for column "tt"
    ```

  缺省值：false/off。

- **ignore_extra_data**：数据源文件中的字段比外表定义列数多时，请选择是忽略多出的列，还是在错误表中报错提示。

  说明

  取值范围：true/on、false/off。
  - 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。
  - 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。
    ```
    extra data after last expected column
    ```

  缺省值：false/off。

- **per node reject_limit**：本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量。如果有一个DN实例上录入错误表中的错误数量超过设定值时，本次导入失败，报错退出。可以选择不做限制，也可以根据所能容忍的错误数量选择一个上限值。
- **compatible_illegal_chars**：导入时遇到非法字符，选择如何处理。是将非法字符按照转换规则转换后入库，还是报错中止导入。

  说明

  取值范围：true/on，false/off。
  - 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。
  - 参数为false/off，导入时遇到非法字符进行报错，中断导入。

  缺省值：false/off。

 版权所有 © 华为云计算技术有限公司

非法字符容错转换规则如下：

- 对于'\0'，容错后转换为空格。
- 对于其他非法字符，容错后转换为问号。
- 对非法字符进行容错转换时，如遇NULL、DELIMITER、QUOTE、ESCAPE也设置成了空格或问号，DWS会通过如“illegal chars conversion may confuse COPY escape 0x20”等报错信息提示用户修改可能引起混淆的参数以避免导入错误。

- **error_table_name**：用于记录数据格式错误信息的错误表表名。并行导入结束后查询此错误信息表，能够获取详细的错误信息。
- **remote log 'name'**：数据导入过程中的数据格式错误信息是否同时在GDS服务器上以文件方式保存。name为错误数据文件的文件名前缀。
- 关于容错性参数的更多信息请参考《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (GDS导入导出)”章节的容错性参数说明。

步骤3 使用gsql或Data Studio连接数据库后，根据前面步骤所收集和规划的信息参数，创建GDS外表。

示例如下：

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
 SERVER gsmpp_server
 OPTIONS
(
LOCATION 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
FORMAT 'CSV' ,
DELIMITER ',',
ENCODING 'utf8',
HEADER 'false',
FILL_MISSING_FIELDS 'true',
IGNORE_EXTRA_DATA 'true'
)
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

示例中的各项说明如下：

- 外表字段需与数据库中即将存储数据的事实表保持一致。
- 对于GDS导入，SERVER gsmpp_server请保持不变。
- location参数请根据**步骤1**中收集的GDS服务访问信息修改。注意GDS使用SSL加密传输时，需要将“gsfs”替换为“gsfss”。
- FORMAT、DELIMITER、ENCODING、HEADER请根据**步骤1**中收集的数据源格式信息填写。
- FILL_MISSING_FIELDS、IGNORE_EXTRA_DATA、LOG INTO及PER NODE REJECT LIMIT请根据**步骤2**中设计的导入容错机制填写。注意LOG INTO是指将数据格式错误录入哪个错误表，即其取值为错误表表名。

----结束

任务示例

除了以下示例，更多外表创建的示例请参考**GDS导入示例**。

 版权所有 © 华为云计算技术有限公司

- **示例1：**创建GDS外表foreign_tpcds_reasons，数据格式为CSV。

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
 SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
FORMAT 'CSV',MODE 'Normal', ENCODING 'utf8', DELIMITER E'\x08', QUOTE E'\x1b', NULL '');
```

- **示例2：**创建GDS导入外表foreign_tpcds_reasons_SSL，使用SSL加密传输的模式传输，数据格式为CSV。

```
CREATE FOREIGN TABLE foreign_tpcds_reasons_SSL
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
 SERVER gsmpp_server OPTIONS (location 'gsfss://192.168.0.90:5000/* | gsfss://192.168.0.91:5000/*',
FORMAT 'CSV',MODE 'Normal', ENCODING 'utf8', DELIMITER E'\x08', QUOTE E'\x1b', NULL '');
```

- 示例3：创建GDS外表foreign_tpcds_reasons，数据格式为TEXT。

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
FORMAT 'TEXT', delimiter E'\x08',  null '',reject_limit '2',EOL '0x0D') WITH err_foreign_tpcds_reasons;
```

- 示例4：创建GDS外表foreign_tpcds_reasons，数据格式为FIXED。

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk       integer      position(1,2),
  r_reason_id       char(16)     position(3,16),
  r_reason_desc     char(100)    position(19,100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/*', FORMAT 'FIXED', ENCODING
'utf8',FIX '119');
```

2.3.2.5 执行导入数据

完成GDS的安装部署及外表创建后，本节介绍如何在DWS数据库中创建事实表并将数据导入事实表中。

对于记录数超过千万条的表，建议在执行全量数据导入前，先导入部分数据，以**进行数据倾斜检查和调整分布列**，避免导入大量数据后发现数据倾斜，调整成本高。

前提条件

GDS服务器和DWS集群之间网络可以互通。

- 需要创建一个弹性云服务器作为GDS服务器。
- 创建的弹性云服务器与DWS集群应处于同一区域、同一虚拟私有云和子网。

操作步骤

步骤1 在DWS中创建目标表，用于存储导入的数据。

步骤2 （可选）若导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引，但是如果不能保证数据唯一性不建议删除唯一索引。在数据导入完成后，再重新创建索引。

1. 假定在导入表“product_info”上的“product_id”字段上存在普通索引“product_idx”。在执行数据导入前，请先删除相关索引。

版权所有 © 华为云计算技术有限公司

```
DROP INDEX product_idx;
```

2. 在数据导入完成后，重建索引。

```
CREATE INDEX product_idx ON product_info(product_id);
```

步骤3 执行数据导入。

```
INSERT INTO [目标表名] SELECT * FROM [foreign table 表名];
```

- 如果出现以下类似信息，说明数据导入成功。请查询错误信息表，查看是否存在数据格式错误，详细操作请参见**处理导入错误**。

```
INSERT 0 9
```

- 如果出现数据加载错误，请参见**处理导入错误**，并重新执行数据导入。

说明

- 如果执行过程中出现数据加载错误，则数据全部导入失败，没有数据导入至目标表中。
- 编写批处理任务脚本，实现并发批量导入数据。并发量视机器资源使用情况而定。可通过几个表测试，监控资源利用率，根据结果提高或减少并发量。常用资源监控命令有：内存和CPU监控top命令，IO监控命令iostat，网络监控命令sar等。相关案例请参见**多线程导入**。
- 在资源许可的情况下，多台GDS服务器并发导入会很大程度上提高数据导入效率。相关案例请参见**多数据服务器并行导入**。
- 对于高并发的GDS导入场景，为了保持GDS和DN间的数据连接稳定，可以将GDS服务器环境和DN所在环境的TCP Keepalive检测时间增长（推荐增长至5分钟）。调整集群环境的TCP Keepalive参数会影响故障检测的响应时间。

----结束

任务示例

1. 创建一个名为reasons的目标表。

```
CREATE TABLE reasons
(
  r_reason_sk   integer  not null,
  r_reason_id   char(16) not null,
  r_reason_desc char(100)
)
DISTRIBUTE BY HASH (r_reason_sk);
```

2. 在执行数据导入前，先删除相关表的索引。

   假定在导入表“reasons”上的“r_reason_id”字段上存在普通索引“reasons_idx”。在执行数据导入前，请先删除相关索引。

```
DROP INDEX reasons_idx;
```

3. 将数据源文件中的数据通过外表“foreign_tpcds_reasons”导入到表“reasons”中。

```
INSERT INTO reasons SELECT * FROM foreign_tpcds_reasons ;
```

4. 在数据导入完成后，再重新创建索引。

```
CREATE INDEX reasons_idx ON reasons(r_reasons_id);
```

2.3.2.6 处理导入错误

操作场景

当数据导入发生错误时，请根据本文指引信息进行处理。

查询错误信息

数据导入过程中发生的错误，一般分为数据格式错误和非数据格式错误，错误表只能记录数据格式错误。

 版权所有 © 华为云计算技术有限公司

- 数据格式错误

  在创建外表时，通过设置参数“LOG INTO error_table_name”，将数据导入过程中出现的数据格式错误信息写入指定的错误信息表error_table_name中。您可以通过以下SQL，查询详细错误信息。

  ```
  SELECT * FROM error_table_name;
  ```

  错误信息表结构如表2-15所示。

  表 2-15 错误信息表

列名称	类型	描述
nodeid	integer	报错节点编号。
begintime	timestamp with time zone	出现数据格式错误的时间。
filename	character varying	出现数据格式错误的数据源文件名。 当GDS导入时，同时会包括对应GDS服务端的IP地址端口信息。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。
detail	text	详细错误信息。

- 非数据格式错误

  对于非数据格式错误，一旦发生将导致整个数据导入失败，不会写入错误表。您可以根据执行数据导入过程中，界面提示的错误信息，帮助定位问题，处理错误。

处理数据导入错误

根据获取的错误信息，请对照下表，处理数据导入错误。

版权所有 © 华为云计算技术有限公司

表 2-16 处理数据导入错误

错误信息	错误类型	原因	解决办法
missing data for column "r_reason_desc"	格式错误	1. 数据源文件中的列数比外表定义的列数少。 2. 对于TEXT格式的数据源文件，由于转义字符（\）导致delimiter（分隔符）错位或者quote（引号字符）错位造成的错误。 **示例：**目标表存在3列字段，导入的数据如下所示。由于存在转义字符“\”，分隔符“|”被转义为第二个字段的字段值，导致第三个字段值缺失。 `BE|Belgium\\|1`	1. 由于列数少导致的报错，选择下列办法解决： ● 在数据源文件中，增加列“r_reason_desc”的字段值。 ● 在创建外表时，将参数“fill_missing_fields”设置为“on”。即当导入过程中，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 2. 对由于转义字符导致的错误，需检查报错的行中是否含有转义字符（\）。若存在，建议在创建外表时，将参数“noescaping”（是否不对'\'和后面的字符进行转义）设置为true。
extra data after last expected column	格式错误	数据源文件中的列数比外表定义的列数多。	● 在数据源文件中，删除多余的字段值。 ● 在创建外表时，将参数“ignore_extra_data”设置为“on”。即在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。
invalid input syntax for type numeric: "a"	格式错误	数据类型错误。	在数据源文件中，修改输入字段的数据类型。根据此错误信息，请将输入的数据类型修改为numeric。
null value in column "staff_id" violates not-null constraint	非格式错误	非空约束。	在数据源文件中，增加非空字段信息。根据此错误信息，请增加“staff_id”列的值。

版权所有 © 华为云计算技术有限公司

错误信息	错误类型	原因	解决办法
duplicate key value violates unique constraint "reg_id_pk"	非格式错误	唯一约束。	• 删除数据源文件中重复的行。 • 通过设置关键字“DISTINCT”，从SELECT结果集中删除重复的行，保证导入的每一行都是唯一的。 **INSERT INTO** reasons **SELECT DISTINCT * FROM** foreign_tpcds_reasons;
value too long for type character varying(16)	格式错误	字段值长度超过限制。	在数据源文件中，修改字段值长度。根据此错误信息，字段值长度限制为VARCHAR2(16)。

2.3.2.7 停止 GDS

操作场景

待导入数据成功后，停止GDS。

操作步骤

步骤1 以gds_user用户登录安装GDS的数据服务器。

步骤2 请根据启动GDS的方式，选择停止GDS的方式。

- 若用户使用“gds”命令启动GDS，请使用以下方式停止GDS。
  a. 执行如下命令，查询GDS进程号。
  ```
  ps -ef|grep gds
  ```
  示例：其中GDS进程号为128954。
  ```
  ps -ef|grep gds
  gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data/ -p 192.168.0.90:5000 -l /log/gds_log.txt -D
  gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
  ```
  b. 使用“kill”命令，停止GDS。其中128954为上一步骤中查询出的GDS进程号。
  ```
  kill -9 128954
  ```
- 若用户使用“gds_ctl.py”命令启动GDS，请使用以下命令停止GDS。
  ```
  cd /opt/bin/dws/gds/bin
  python3 gds_ctl.py stop
  ```

----结束

 版权所有 © 华为云计算技术有限公司

2.3.2.8 GDS 导入示例

多数据服务器并行导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90和192.168.0.91。数据源文件格式为CSV。

1. 以root用户登录每台GDS数据服务器，在两台数据服务器上，分别创建数据文件存放目录“/input_data”。以下以IP为192.168.0.90的数据服务器为例进行操作，剩余服务器上的操作与它一致。
   ```
   mkdir -p /input_data
   ```
2. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该类用户及所属用户组已存在，可跳过此步骤。
   ```
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
3. 将准备好的CSV格式数据源文件均匀分发至相应数据服务器的“/input_data”目录中。
4. 修改每台数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。以下以IP为192.168.0.90的数据服务器为例，进行操作。
   ```
   chown -R gds_user:gdsgrp /input_data
   ```
5. 以gds_user用户登录每台数据服务器上分别启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90和192.168.0.91，GDS监听端口为5000，以后台方式运行。

   在IP为192.168.0.90的数据服务器上启动GDS。
   ```
   /opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
   ```
   在IP为192.168.0.91的数据服务器上启动GDS。
   ```
   /opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.91:5000 -H 10.10.0.1/24  -D
   ```
6. 使用工具连接数据库。
7. 创建导入的目标表tpcds.reasons。
   ```
   CREATE TABLE tpcds.reasons
   (
     r_reason_sk integer not null,
     r_reason_id char(16) not null,
     r_reason_desc char(100)
   );
   ```
8. 创建外表tpcds.foreign_tpcds_reasons用于接收数据服务器上的数据。

   其中设置**导入模式信息**如下所示：
   - 导入模式为Normal模式。
   - 由于启动GDS时，设置的数据源文件存放目录为“/input_data”，GDS监听端口为5000，所以设置参数“location”为“gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*”。

   设置**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：
   - 数据源文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为E'\x1b'。
   - 数据文件中空值（null）为没有引号的空字符串。

版权所有 © 华为云计算技术有限公司

- 逃逸字符（escape）默认和quote相同。
- 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

设置**导入容错性**如下所示：

- 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
- 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons。

根据以上信息，创建的外表如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
format 'CSV',mode 'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '', fill_missing_fields
'false') LOG INTO err_tpcds_reasons PER NODE REJECT LIMIT 'unlimited';
```

9. 通过外表tpcds.foreign_tpcds_reasons，将数据导入目标表tpcds.reasons。
   ```
   INSERT INTO tpcds.reasons SELECT * FROM tpcds.foreign_tpcds_reasons;
   ```
10. 查询错误信息表err_tpcds_reasons，处理数据导入错误。详细请参见**处理导入错误**。
    ```
    SELECT * FROM err_tpcds_reasons;
    ```
11. 待数据导入完成后，以gds_user用户登录每台数据服务器，分别停止GDS。

    以下以IP为192.168.0.90的数据服务器为例，停止GDS。其中GDS进程号为128954。
    ```
    ps -ef|grep gds
    gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data -p 192.168.0.90:5000 -D
    gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
    kill -9 128954
    ```

多线程导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导入的数据源文件格式为CSV，同时导入2个目标表。

1. 以root用户登录GDS数据服务器，创建数据文件存放目录“/input_data”，以及子目录“/input_data/import1/”和“/input_data/import2/”。
   ```
   mkdir -p /input_data
   ```
2. 将目标表tpcds.reasons1的数据源文件存放在数据服务器“/input_data/import1/”目录下，将目标表tpcds.reasons2的数据源文件存放在目录“/input_data/import2/”下。
3. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该用户及所属用户组已存在，可跳过此步骤。
   ```
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
4. 修改数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /input_data
   ```
5. 以gds_user用户登录数据服务器上启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行，设定并发度为2，并设定递归文件目录。
   ```
   /opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24  -D -t 2 -r
   ```


版权所有 © 华为云计算技术有限公司

6. 使用工具连接数据库。
7. 在数据库中创建导入的目标表tpcds.reasons1和tpcds.reasons2。

```
CREATE TABLE tpcds.reasons1
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) ;
CREATE TABLE tpcds.reasons2
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) ;
```

8. 在数据库中创建外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2用于接收数据服务器上的数据。

   以下以外表tpcds.foreign_tpcds_reasons1为例，讲解设置的导入外表参数信息。

   其中设置的**导入模式信息**如下所示：

   - 导入模式为Normal模式。
   - 由于启动GDS时，设置的数据源文件存放目录为“/input_data/”，GDS监听端口为5000，实际存放数据源文件目录为“/input_data/import1/”，所以设置参数“location”为“gsfs://192.168.0.90:5000/import1/*”。

   设置的**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：

   - 数据源文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为E'\x1b'。
   - 数据文件中空值（null）为没有引号的空字符串。
   - 逃逸字符（escape）默认和quote相同。
   - 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

   设置的**导入容错性**如下所示：

   - 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
   - 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons1。
   - 当数据源文件中一行的最后一个字段缺失（fill_missing_fields）时，自动设置为NULL。

   根据以上信息，创建的外表tpcds.foreign_tpcds_reasons1如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons1
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import1/*', format 'CSV',mode 'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '',fill_missing_fields 'on')LOG INTO err_tpcds_reasons1 PER NODE REJECT LIMIT 'unlimited';
```

   参考以上设置，创建的外表tpcds.foreign_tpcds_reasons2如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons2
(
  r_reason_sk integer not null,
```


版权所有 © 华为云计算技术有限公司

```
 r_reason_id char(16) not null,
 r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import2/*', format 'CSV',mode
'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '',fill_missing_fields 'on')LOG INTO
err_tpcds_reasons2 PER NODE REJECT LIMIT 'unlimited';
```

9. 通过外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2将数据分别导入tpcds.reasons1和tpcds.reasons2。

```
INSERT INTO tpcds.reasons1 SELECT * FROM tpcds.foreign_tpcds_reasons1;
INSERT INTO tpcds.reasons2 SELECT * FROM tpcds.foreign_tpcds_reasons2;
```

10. 查询错误信息表err_tpcds_reasons1和err_tpcds_reasons2，处理数据导入错误。详细请参见**处理导入错误**。

```
SELECT * FROM err_tpcds_reasons1;
SELECT * FROM err_tpcds_reasons2;
```

11. 待数据导入完成后，以gds_user用户登录数据服务器，停止GDS。

    其中GDS进程号为128954。

```
ps -ef|grep gds
gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data -p 192.168.0.90:5000 -D -t 2 -r
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

单个管道文件导入

步骤1 启动GDS：

```
/opt/bin/dws/gds/bin/gds -d /***/gds_data/ -D -p 192.168.0.1:7789 -l /***/gds_log/aa.log -H 0/0 -t 10 -D
```

如果需要设置管道文件的超时时间，则使用--pipe-timeout参数设置。

步骤2 执行数据导入。

1. 登录数据库创建内表：

```
CREATE TABLE test_pipe_1( id integer not null, gender text not null, name  text );
```

2. 创建只读外表：

```
CREATE FOREIGN TABLE foreign_test_pipe_tr( like test_pipe ) SERVER gsmpp_server OPTIONS
(LOCATION 'gsfs://192.168.0.1:7789/foreign_test_pipe.pipe', FORMAT 'text', DELIMITER ',',  NULL '',
EOL '0x0a' ,file_type 'pipe',auto_create_pipe 'false');
```

3. 执行导入语句，此时语句会阻塞：

```
INSERT INTO test_pipe_1 SELECT * FROM foreign_test_pipe_tr;
```

步骤3 通过GDS管道文件导入数据。

1. 登录GDS数据服务器进入GDS数据目录：

```
cd /***/gds_data/
```

2. 创建管道文件，如果auto_create_pipe设置为true跳过此步骤：

```
mkfifo foreign_test_pipe.pipe;
```

> **说明**
>
> 管道文件创建完成后，每执行完一次操作，业务会被自动清理。如果还需要执行其他业务，请参考该步骤重新创建管道文件。

3. 向管道文件中写入数据：

```
cat postgres_public_foreign_test_pipe_tw.txt > foreign_test_pipe.pipe
```

4. 若需要读取压缩文件到管道文件，执行：

```
gzip -d < out.gz > foreign_test_pipe.pipe
```

5. 若需要读取hdfs文件到管道文件，执行：

```
hdfs dfs -cat - /user/hive/***/test_pipe.txt > foreign_test_pipe.pipe
```

步骤4 查看导入语句返回的结果：

```
INSERT INTO test_pipe_1 select * from foreign_test_pipe_tr;
INSERT 0 4
```


版权所有 © 华为云计算技术有限公司

```
SELECT * FROM test_pipe_1;
id | gender |      name
----+------+----------------
3 | 2   | 11111111111111
1 | 2   | 11111111111111
2 | 2   | 11111111111111
4 | 2   | 11111111111111
(4 rows)
```

----结束

多进程管道文件导入

GDS支持多进程管道文件导入， 即启动一个外表对应多个GDS。

以本地文件的导入为例：

步骤1 启动多个GDS，如果已经启动跳过此步骤：

```
/opt/bin/dws/gds/bin/gds -d /***/gds_data/ -D -p 192.168.0.1:7789 -l /***/gds_log/aa.log -H 0/0 -t 10 -D
/opt/bin/dws/gds/bin/gds -d /***/gds_data_1/ -D -p 192.168.0.1:7790 -l /***/gds_log_1/aa.log -H 0/0 -t 10 -D
```

如果需要设置管道文件的超时时间，则使用--pipe-timeout参数设置。

步骤2 执行数据导入。

1. 登录数据库创建内表：
   ```
   CREATE TABLE test_pipe( id integer not null, gender text not null, name  text );
   ```
2. 创建只读外表：
   ```
   CREATE FOREIGN TABLE foreign_test_pipe_tr( like test_pipe ) SERVER gsmpp_server OPTIONS
   (LOCATION 'gsfs://192.168.0.1:7789/foreign_test_pipe.pipe|gsfs://192.168.0.1:7790/
   foreign_test_pipe.pipe', FORMAT 'text', DELIMITER ',', NULL '', EOL '0x0a' , file_type 'pipe',
   auto_create_pipe 'false');
   ```
3. 导入语句，此时语句会阻塞：
   ```
   INSERT INTO test_pipe_1 select * from foreign_test_pipe_tr;
   ```

步骤3 通过GDS管道文件导入数据。

1. 登录GDS数据服务器，分别进入GDS数据目录：
   ```
   cd /***/gds_data/
   cd /***/gds_data_1/
   ```
2. 创建管道文件，如果auto_create_pipe设置为true跳过此步骤：
   ```
   mkfifo foreign_test_pipe.pipe;
   ```
3. 分别读取管道文件并写入新文件：
   ```
   cat postgres_public_foreign_test_pipe_tw.txt > foreign_test_pipe.pipe
   ```

步骤4 查看导入语句返回的结果：

```
INSERT INTO test_pipe_1 select * from foreign_test_pipe_tr;
INSERT 0 4
SELECT * FROM test_pipe_1;
id | gender |      name
----+------+----------------
3 | 2   | 11111111111111
1 | 2   | 11111111111111
2 | 2   | 11111111111111
4 | 2   | 11111111111111
(4 rows)
```

----结束

版权所有 © 华为云计算技术有限公司

集群间不落地数据导入

步骤1 启动GDS。（如果已经启动跳过此步骤）

```
gds -d /***/gds_data/ -D -p GDS_IP:GDS_PORT -l /***/gds_log/aa.log -H 0/0 -t 10 -D
```

如果需要设置管道文件的超时时间，则使用--pipe-timeout参数设置。

步骤2 源数据库数据导出。

1. 登录目标数据库创建内表，并写入数据。

```
CREATE TABLE test_pipe( id integer not null, gender text not null, name  text );
INSERT INTO test_pipe values(1,2,'11111111111111');
INSERT INTO test_pipe values(2,2,'11111111111111');
INSERT INTO test_pipe values(3,2,'11111111111111');
INSERT INTO test_pipe values(4,2,'11111111111111');
INSERT INTO test_pipe values(5,2,'11111111111111');
```

2. 创建只写外表。

```
CREATE FOREIGN TABLE foreign_test_pipe( id integer not null, age text not null, name  text ) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://GDS_IP:GDS_PORT/', FORMAT 'text', DELIMITER ',', NULL '', EOL '0x0a' ,file_type 'pipe') WRITE ONLY;
```

3. 导入语句，此时语句会阻塞。

```
INSERT INTO foreign_test_pipe SELECT * FROM test_pipe;
```

步骤3 目标集群导入数据。

1. 创建内表。

```
CREATE TABLE test_pipe (id integer not null, gender text not null, name text);
```

2. 创建只读外表。

```
CREATE FOREIGN TABLE foreign_test_pipe(like test_pipe) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://GDS_IP:GDS_PORT/', FORMAT 'text', DELIMITER ',', NULL '', EOL '0x0a' , file_type 'pipe', auto_create_pipe 'false');
```

3. 执行导入语句：

```
INSERT INTO test_pipe SELECT * FROM foreign_test_pipe;
```

步骤4 查看目标集群导入语句返回的结果：

```
SELECT * FROM test_pipe;
 id | gender |      name
----+-----+----------------
  3 | 2   | 11111111111111
  6 | 2   | 11111111111111
  7 | 2   | 11111111111111
  1 | 2   | 11111111111111
  2 | 2   | 11111111111111
  4 | 2   | 11111111111111
  5 | 2   | 11111111111111
  8 | 2   | 11111111111111
  9 | 2   | 11111111111111
(9 rows)
```

----结束

版权所有 © 华为云计算技术有限公司

说明

GDS默认导出或者导入的管道文件命名规则为：“数据库名_模式名_外表名.pipe”，因此默认需要目标集群与源集群的数据库名及模式名保持一致。如果数据库或模式不一致，则可以在location的url中指定相同的管道文件。

示例：

- 只写外表指定管道名。
```
CREATE FOREIGN TABLE foreign_test_pipe(id integer not null, age text not null, name  text) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://GDS_IP:GDS_PORT/foreign_test_pipe.pipe', FORMAT 'text', DELIMITER ',',  NULL '', EOL '0x0a' ,file_type 'pipe') WRITE ONLY;
```
- 只读外表指定管道名。
```
CREATE FOREIGN TABLE foreign_test_pipe(like test_pipe) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://GDS_IP:GDS_PORT/foreign_test_pipe.pipe', FORMAT 'text', DELIMITER ',',  NULL '', EOL '0x0a' ,file_type 'pipe',auto_create_pipe 'false');
```

2.3.3 从 MRS 导入数据到集群

2.3.3.1 从 MRS 导入数据概述

MapReduce服务（MapReduce Service，简称MRS）是一个基于开源Hadoop生态环境而运行的大数据集群，对外提供大容量数据的存储和分析能力，可解决用户的数据存储和处理需求。具体信息可参考《MapReduce服务用户指南》。

用户可以将海量业务数据，存储在MRS的分析集群，即使用Hive/Spark组件保存。Hive/Spark的数据文件则保存在HDFS中。DWS支持在相同网络中，配置一个DWS集群连接到一个MRS集群，然后将数据从HDFS中的文件读取到DWS。

须知

确保MRS跟DWS网络互联互通，主要分以下几种场景：

场景一：MRS与DWS在同一个区域、同一个VPC下，默认网络互通。

场景二：MRS与DWS在同一个区域，不同VPC下，需要建立VPC对等连接，参见《虚拟私有云用户指南》的“对接连接简介”。

场景三：MRS与DWS不在一个区域，需要通过“云连接(CC)”打通网络，请参见对应服务的用户指南。

场景四：MRS属于云下场景，需要通过“云专线(DC)”或“虚拟专用网络(VPN)”打通网络，请参见对应服务的用户指南。

从 MRS 导入数据到集群的流程

1. **MRS集群上的数据准备**
2. （可选）**手动创建外部服务器**
3. **创建外表**
4. **执行数据导入**
5. **清除资源**

2.3.3.2 MRS 集群上的数据准备

从MRS导入数据到DWS集群之前，假设您已经完成了以下准备工作：

版权所有 © 华为云计算技术有限公司

1. 已创建MRS集群。
2. 在MRS集群上创建了Hive/Spark ORC表，且表数据已经存储到该表对应的HDFS路径上。

如果您已经完成上述准备，则可以跳过本章节。

为方便起见，以在MRS集群上创建Hive ORC表作为示例，完成上述准备工作。在MRS集群上创建Spark ORC表的大致流程和SQL语法，同Hive类似，在本文中不再展开描述。

数据文件

假设有数据文件product_info.txt，示例数据如下所示：

```
100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt Women,red,M,328,2017-09-04,715,good
205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear Women,pink,L,584,2017-09-05,406,very good!
300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt men,red,XL,1245,2017-09-03,502,Bad.
310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket women,red,L,411,2017-09-05,436,It's really super nice
150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans Women,blue,M,1223,2017-09-06,1200,The seller's packaging is exquisite
200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants men,black,L,997,2017-09-10,301,The clothes are of good quality.
250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress women,black,S,841,2017-09-15,299,Follow the store for a long time.
108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress women,red,M,85,2017-09-14,22,It's really amazing to buy
450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket women,white,M,114,2017-09-14,22,Open the package and the clothes have no odor
260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat women,red,L,2004,2017-09-15,826,Very favorite clothes
980,ZKDS-J-5490-#cW4,2017-09-13,B,2017 Autumn New Women's Cotton Clothing,red,M,112,2017-09-16,219,The clothes are small
98,FKQB-I-2564-#dA5,2017-09-15,B,2017 autumn new shoes men,green,M,4345,2017-09-18,5473,The clothes are thick and it's better this winter.
150,DMQY-K-6579-#eS6,2017-09-21,A,2017 autumn new underwear men,yellow,37,2840,2017-09-25,5831,This price is very cost effective
200,GKLW-l-2897-#wQ7,2017-09-22,A,2017 Autumn New Jeans Men,blue,39,5879,2017-09-25,7200,The clothes are very comfortable to wear
300,HWEC-L-2531-#xP8,2017-09-23,A,2017 autumn new shoes women,brown,M,403,2017-09-26,607,good
100,IQPD-M-3214-#yQ1,2017-09-24,B,2017 Autumn New Wide Leg Pants Women,black,M,3045,2017-09-27,5021,very good.
350,LPEC-N-4572-#zX2,2017-09-25,B,2017 Autumn New Underwear Women,red,M,239,2017-09-28,407,The seller's service is very good
110,NQAB-O-3768-#sM3,2017-09-26,B,2017 autumn new underwear women,red,S,6089,2017-09-29,7021,The color is very good
210,HWNB-P-7879-#tN4,2017-09-27,B,2017 autumn new underwear women,red,L,3201,2017-09-30,4059,I like it very much and the quality is good.
230,JKHU-Q-8865-#uO5,2017-09-29,C,2017 Autumn New Clothes with Chiffon Shirt,black,M,2056,2017-10-02,3842,very good
```

在 MRS 集群上创建 Hive ORC 表

1. 创建了MRS集群。

   具体操作请参见《 MapReduce服务用户指南 》的“创建集群 > 创建自定义集群”章节。

2. 下载客户端。

版权所有 © 华为云计算技术有限公司

a. 在MRS集群页面，单击集群名称进入“概览”，单击“前往Manager”，如果提示绑定公网IP，请先绑定公网IP。

b. 输入MRS Manager的用户名admin和密码，密码为创建MRS集群时输入的admin密码。

c. 登录成功后，选择“集群 > *待操作集群的名称* > 概览 > 更多 > 下载客户端”。界面显示“下载集群客户端”对话框。

说明

历史版本的客户端获取方法：请选择“服务管理 > 下载客户端”，“客户端类型”选择“仅配置文件”。

3. 登录MRS集群的Hive客户端。

a. 登录Master节点。

具体操作，请参见《MapReduce服务用户指南》中的“远程操作指南 > 登录Master节点”章节。

b. 执行以下命令切换用户：

```
sudo su - omm
```

c. 执行以下命令切换到客户端目录：

```
cd /opt/client
```

d. 执行以下命令配置环境变量：

```
source bigdata_env
```

e. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建Hive表的权限：

具体操作请参见《MapReduce服务用户指南》的“创建角色”章节。

f. 配置拥有对应权限的角色：

具体操作请参见《MapReduce服务用户指南》的“创建用户”章节。

g. 为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令：

```
kinit MRS集群用户
```

例如，kinit hiveuser。

版权所有 © 华为云计算技术有限公司

h. 执行以下命令启动Hive客户端：

```
beeline
```

4. 在Hive中创建数据库demo。

   执行以下命令创建数据库：

   ```
   CREATE DATABASE demo;
   ```

5. 在数据库demo中新建了一个Hive TEXTFILE类型的表product_info，并将**数据文件**（product_info.txt）导入到该表对应的HDFS路径中。

   执行以下命令切换到demo数据库：

   ```
   USE demo;
   ```

   执行以下命令，创建表product_info，表字段按照**数据文件**中的数据进行定义：

   ```
   DROP TABLE product_info;

   CREATE TABLE product_info
   (
       product_price               int           ,
       product_id                 char(30)       ,
       product_time                date          ,
       product_level              char(10)       ,
       product_name                 varchar(200)   ,
       product_type1               varchar(20)    ,
       product_type2               char(10)       ,
       product_monthly_sales_cnt    int           ,
       product_comment_time         date          ,
       product_comment_num          int           ,
       product_comment_content      varchar(200)
   )
   row format delimited fields terminated by ','
   stored as TEXTFILE;
   ```

   有关导入数据到MRS集群的操作，请参见《MapReduce服务用户指南》中的“集群操作指导 > 管理现有集群 > 管理数据文件”章节。

6. 在数据库demo中创建了一个Hive ORC表product_info_orc。

   执行以下命令，创建Hive ORC表product_info_orc，表字段与上一步创建的表product_info完全一致：

   ```
   DROP TABLE product_info_orc;

   CREATE TABLE product_info_orc
   (
       product_price               int           ,
       product_id                 char(30)       ,
       product_time                date          ,
       product_level              char(10)       ,
       product_name                 varchar(200)   ,
       product_type1               varchar(20)    ,
       product_type2               char(10)       ,
       product_monthly_sales_cnt    int           ,
       product_comment_time         date          ,
       product_comment_num          int           ,
       product_comment_content      varchar(200)
   )
   row format delimited fields terminated by ','
   stored as orc;
   ```

7. 将product_info表的数据插入到Hive ORC表product_info_orc中：

   ```
   INSERT INTO product_info_orc SELECT * FROM product_info;
   ```

   查询表product_info_orc：

   ```
   SELECT * FROM product_info_orc;
   ```

   如果查询到如**数据文件**所示的数据，表示已经成功将数据插入到ORC表。

版权所有 © 华为云计算技术有限公司

2.3.3.3 手动创建外部服务器

创建外表语法（CREATE FOREIGN TABLE (SQL on Hadoop or OBS)）中，需指定一个与MRS数据源连接相关联的外部服务器。

当您通过DWS管理控制台创建MRS数据源连接时，数据库管理员dbadmin会在默认数据库postgres中自动创建一个外部服务器。因此，如果您希望在默认数据库postgres中创建外表读取MRS数据，可以跳过本章节。

如果您希望使用普通用户在自定义数据库中创建外表读取MRS数据，必须先在自定义数据库中手动创建一个外部服务器。本章节将为您介绍，如何使用普通用户在自定义数据库中创建外部服务器。步骤如下：

1. 请确保DWS集群已创建MRS数据源连接。

   具体操作请参见《数据仓库服务用户指南》的“管理MRS数据源 > 创建MRS数据源连接”章节。
2. **创建用户和数据库并授予外表权限**
3. **手动创建外部服务器**

说明

需要注意的是，当您不再需要从该MRS数据源读取数据时，通过DWS管理控制台删除MRS数据源，仅会删除在默认数据库postgres中自动创建的外部服务器，手动创建的外部服务器需要您手动删除，具体操作请参见**删除手动创建的外部服务器**中的描述。

创建用户和数据库并授予外表权限

以下示例，是新建一个普通用户dbuser并创建一个数据库mydatabase，然后使用管理员用户授予dbuser外表权限。

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库gaussdb。

例如，使用gsql客户端的用户通过如下语句连接数据库：

```
gsql -d gaussdb -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
```

步骤2 新建一个普通用户，并用它创建一个数据库。

新建一个具有创建数据库权限的用户dbuser：

```
CREATE USER dbuser WITH CREATEDB PASSWORD 'password';
```

切换为新建的用户：

```
SET ROLE dbuser PASSWORD 'password';
```

执行以下命令创建数据库：

```
CREATE DATABASE mydatabase;
```

查询数据库：

```
SELECT * FROM pg_database;
```

返回结果中有**mydatabase**的信息表示创建成功：

```
datname   | datdba | encoding | datcollate | datctype | datistemplate | datallowconn | datconnlimit | datlastsysoid | datfrozenxid | dattablespace | datcompatibility |                     datacl

-----------+--------+----------+------------+----------+---------------+--------------+--------------+---------------+--------------+---------------+------------------+-------------------------------------
 template1 |     10 |        0 | C          | C        | t             | t            |           -1 |         14146 |         1351 |          1663 | ORA              | {=c/Ruby,Ruby=CTc/Ruby}
```


版权所有 © 华为云计算技术有限公司

```
 template0  |     10 |        0 | C          | C        | t            | f           |          -1 |        14146 |        1350
|         1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 gaussdb   |     10 |        0 | C          | C        | f            | t           |          -1 |        14146 |        1352 |
1663 | ORA             | {=Tc/Ruby,Ruby=CTc/Ruby,chaojun=C/Ruby,hu
obinru=C/Ruby}
 mydatabase |  17000 |        0 | C          | C        | f            | t           |          -1 |        14146 |        1351
|         1663 | ORA             |
(4 rows)
```

步骤3 使用管理员用户给普通用户赋予创建外部服务器的权限和使用外表的权限。

使用数据库管理员用户通过数据库客户端连接新建的数据库。

例如，使用gsql客户端的用户可以直接使用如下语句切换为管理员用户去连接新建的数据库：

```
\c mydatabase dbadmin;
```

根据提示输入用户密码。

说明

注意，必须先使用管理员用户连接到**将要创建外部服务器和使用外表的数据库**，再对普通用户进行授权。

默认只有系统管理员才可以创建外部服务器，普通用户需要授权才可以创建，执行以下命令授权：

```
GRANT ALL ON FOREIGN DATA WRAPPER hdfs_fdw TO dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是hdfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令赋予用户使用外表的权限：

```
ALTER USER dbuser USEFT;
```

查看用户：

```
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvalidbegin, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolauditadmin
, r.rolsystemadmin
, r.roluseft
FROM pg_catalog.pg_roles r
ORDER BY 1;
```

返回结果中，dbuser的信息中包含了UseFT权限，表示授权成功：

```
 rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcanlogin | rolconnlimit | rolvalidbegin |
rolvaliduntil | memberof | rolreplication | rolauditadmin | rolsystemadmin | roluseft
-----------+----------+------------+---------------+-------------+-------------+--------------+---------------
+---------------+----------+----------------+---------------+----------------+----------
 dbuser    | f        | t          | f             | t           | t           |           -1 |               |               | {}       | f
| f             | f              | t
 lily      | f        | t          | f             | f           | t           |           -1 |               |               | {}       | f             |
f             | f              | f
 Ruby      | t        | t          | t             | t           | t           |           -1 |               |               | {}       | t
| t             | t              | t
```

----结束

版权所有 © 华为云计算技术有限公司

手动创建外部服务器

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库postgres。

例如：通过gsql客户端登录数据库的用户可以使用以下两种方法中的一种进行连接：

可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令切换数据库和用户：
  ```
  \c postgres dbadmin;
  ```
  根据提示输入密码。
- 如果尚未登录gsql客户端，或者已经登录了gsql客户端执行**\q**退出gsql后，执行以下命令重新进行连接：
  ```
  gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
  ```

步骤2 执行以下命令查询自动创建的外部服务器的信息：

```
SELECT * FROM pg_foreign_server;
```

返回结果如下：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                srvoptions
------------------------------------------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------------------
 gsmpp_server                                   |       10 |  13673 |         |            |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(2 rows)
```

查询结果中，每一行代表一个外部服务器的信息。与MRS数据源连接相关联的外部服务器包含以下信息：

- srvname值包含“hdfs_server”字样以及MRS集群的ID，此ID与MRS管理控制台的集群列表MRS ID相同。
- srvoptions字段中的address参数为MRS集群的主备节点的IP地址及端口。

您可以根据上述信息找到您所要的外部服务器，并记录下它的srvname和srvoptions的值。

步骤3 切换为即将创建外部服务器的用户去连接其对应的数据库。

在本示例中，执行以下命令，使用**创建用户和数据库并授予外表权限**中创建的普通用户dbuser连接其创建的数据库mydatabase：

```
\c mydatabase dbuser;
```

步骤4 创建外部服务器。

创建外部服务器的详细语法，请参见《 数据仓库服务SQL语法参考 》中“CREATE SERVER”章节。示例如下：

```
CREATE SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692cahdfs_server FOREIGN DATA
WRAPPER HDFS_FDW
OPTIONS
(
address '192.168.1.245:25000,192.168.1.218:25000',
hdfscfgpath '/MRS/8f79ada0-d998-4026-9020-80d6de2692ca',
type 'hdfs'
);
```

以下为必选参数的说明：

 版权所有 © 华为云计算技术有限公司

- 外部服务器名称
  允许用户自定义名字。
  在本例中，指定为前面的步骤**步骤2**中记录下来的srvname字段的值，如'*hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca*'。
  不同的数据库之间资源是隔离的，因此在不同的数据库中外部服务器名称可以相同。
- FOREIGN DATA WRAPPER
  只能指定为HDFS_FDW，它在数据库中已经存在。
- OPTIONS参数
  以下参数请分别指定为步骤**步骤2**中记录下来的srvoptions中的参数值。
  - address
    指定HDFS集群的主备节点所在的IP地址以及端口。
  - hdfscfgpath
    指定HDFS集群配置文件路径。该参数仅支持type为HDFS时设置。只能设置一个路径。
  - type
    取值为'hdfs'，表示HDFS_FDW连接的是HDFS。

步骤5 查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

返回结果如下所示，表示已经创建成功：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                      srvoptions
------------------------------------------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------------------------------
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |      |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(1 row)
```

----结束

2.3.3.4 创建外表

在DWS数据库中创建一个Hadoop外表，用来访问存储在MRS HDFS文件系统上的Hadoop结构化数据。Hadoop外表是只读的，只能用于查询操作，可直接使用SELECT查询其数据。

前提条件

- 已创建MRS集群，并将数据导入Hive/Spark数据库中的ORC表。
  请参见**MRS集群上的数据准备**。
- DWS集群已创建MRS数据源连接。
  具体操作请参见《数据仓库服务用户指南》的“管理MRS数据源 > 创建MRS数据源连接”章节。

获取 MRS 数据源的 HDFS 路径

有两种方法可以查看：

 版权所有 © 华为云计算技术有限公司

- **方法一：**

  对于Hive数据，可以登录MRS的Hive客户端（参见2），执行以下命令查看表的详细信息，并记录下location参数中的数据存储路径。

  ```
  use <database_name>;
  desc formatted <table_name>;
  ```

  例如，返回结果中location参数值为“hdfs://hacluster/user/hive/warehouse/demo.db/product_info_orc/”，则记录HDFS路径为“/user/hive/warehouse/demo.db/product_info_orc/”。

- **方法二：**

  按以下步骤获取HDFS路径。

  a. 登录MRS控制台。

  b. 选择“集群列表 > 现有集群”，单击要查看的集群名称，进入集群基本信息页面。

  c. 单击“文件管理”，选择“HDFS文件列表”。

  d. 进入您要导入到DWS集群的数据的存储目录，并记录其路径。

获取 MRS 数据源连接的外部服务器信息

步骤1 使用创建外部服务器的用户去连接其对应的数据库。

是否使用普通用户在自定义数据库中创建外表，请根据需求进行选择：

- **是**

  a. 请先确保，您已按照**手动创建外部服务器**章节中的步骤，创建了普通用户dbuser和它的数据库mydatabase，并在mydatabase中手动创建了一个外部服务器。

  b. 使用用户dbuser通过DWS提供的数据库客户端连接数据库mydatabase。

  如果已经使用gsql客户端连接至数据库，可以直接执行如下命令进行用户和数据库切换：

  ```
  \c mydatabase dbuser;
  ```

  根据界面提示输入密码。

- **否**

  当您通过DWS管理控制台创建MRS数据源连接时，数据库管理员dbadmin会在默认数据库postgres中自动创建一个外部服务器。因此，如果使用数据库管理员dbadmin在默认数据库postgres中创建外表，需要通过DWS提供的数据库客户端工具连接数据库。例如，使用gsql客户端的用户通过如下命令连接数据库：

  ```
  gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
  ```

步骤2 执行以下命令，查看已创建的MRS数据源连接的外部服务器信息。

```
SELECT * FROM pg_foreign_server;
```

说明

也可以执行**\desc+**命令查看外部服务器信息。

返回结果如下：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                srvoptions
------------------------------------------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------
 gsmpp_server                           |       10 |  13673 |         |            |        |
```

 版权所有 © 华为云计算技术有限公司

```
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |   16476 | 13685 |       |         |      |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(2 rows)
```

查询结果中，每一行代表一个外部服务器的信息。与MRS数据源连接相关联的外部服务器包含以下信息：

- srvname值包含“hdfs_server”字样以及MRS集群的ID，此ID与MRS管理控制台的集群列表MRS ID相同。
- srvoptions字段中的address参数为MRS集群的主备节点的IP地址及端口。

您可以根据上述信息找到您所要的外部服务器，并记录下它的srvname和srvoptions的值。

----结束

创建外表

当完成**获取MRS数据源连接的外部服务器信息**和**获取MRS数据源的HDFS路径**后，就可以创建一个外表，用于读取MRS数据源数据。

创建外表的语法格式如下，详细的描述请参见《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (SQL on Hadoop or OBS)”章节。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER dfs_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为"*foreign_product_info*"的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  必选。外表的表名。
- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。

  外表的字段个数和字段类型，需要与MRS上保存的数据完全一致。定义字段的数据类型之前，您必须先了解**数据类型转换说明**。
- **SERVER dfs_server**

  外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器，从而关联MRS数据源连接并从MRS集群读取数据。

  此处应填写为通过**获取MRS数据源连接的外部服务器信息**查询到的“srvname”字段的值。
- **OPTIONS参数**

  用于指定外表数据的各类参数，关键参数如下所示。
  - **format**：必选参数。取值只支持“orc”。表示数据源文件的格式，只支持Hive的ORC数据文件。

版权所有 © 华为云计算技术有限公司

- **foldername**：必选参数。表示数据在HDFS的存储目录或数据文件路径。

  如果是启用了Kerberos认证的MRS分析集群，请确保MRS数据源连接的MRS用户，拥有此目录的读取权限。

  请按照**获取MRS数据源的HDFS路径**中的步骤获取HDFS路径，该路径作为**foldername**的参数值。
- **encoding**：可选参数。外表中数据源文件的编码格式名称，缺省为utf8。
- **DISTRIBUTE BY**

  表示外表的数据读取方式。有以下两种方式供选择，在本例中选择ROUNDROBIN。

  - **ROUNDROBIN**：表示外表在从数据源读取数据时，DWS集群每一个节点读取随机一部分数据，并组成完整数据。
  - **REPLICATION**：表示外表在从数据源读取数据时，DWS集群每一个节点都读取一份完整数据。
- **语法中的其他参数**

  其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中不需要设置。

根据以上信息，创建外表命令如下所示：

```
DROP FOREIGN TABLE IF EXISTS foreign_product_info;

CREATE FOREIGN TABLE foreign_product_info
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca
OPTIONS (
format 'orc',
encoding 'utf8',
foldername '/user/hive/warehouse/demo.db/product_info_orc/'
)
DISTRIBUTE BY ROUNDROBIN;
```

数据类型转换说明

当前用户导入到Hive/Spark的数据在HDFS存储为ORC文件格式，DWS实际读取HDFS中的ORC文件，并对文件内的数据进行查询分析。

由于Hive/Spark支持的数据类型与DWS自身支持的数据类型存在差异，在创建外表定义表字段时，您需要了解这两者之间数据类型的对应关系，具体如**表2-17**所示：

版权所有 © 华为云计算技术有限公司

表 2-17 数据类型匹配表

类型名称	DWS的HDFS/OBS外表支持的字段类型	Hive表字段类型	Spark表字段类型
2字节整数	SMALLINT	SMALLINT	SMALLINT
4字节整数	INTEGER	INT	INT
8字节整数	BIGINT	BIGINT	BIGINT
单精度浮点数	FLOAT4 (REAL)	FLOAT	FLOAT
双精度浮点型	FLOAT8(DOUBLE PRECISION)	DOUBLE	FLOAT
科学数据类型	DECIMAL[p (,s)] 最大支持38位精度	DECIMAL 最大支持38位（Hive 0.11）	DECIMAL
日期类型	DATE	DATE	DATE
时间类型	TIMESTAMP	TIMESTAMP	TIMESTAMP
Boolean类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)	CHAR (n)	STRING
VarChar类型	VARCHAR(n)	VARCHAR (n)	VARCHAR (n)
字符串	TEXT(CLOB)	STRING	STRING

2.3.3.5 执行数据导入

直接查询外表查看 MRS 数据源的数据

如果数据量较少，可直接使用SELECT查询外表，即可查看到MRS数据源的数据。

步骤1 执行以下命令，则可以从外表查询数据：

```
SELECT * FROM foreign_product_info;
```

查询结果显示如**数据文件**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(20 rows)
```

通过外表查询到数据后，用户可以将数据插入数据库的普通表。

----结束

导入数据后查询数据

也可以将MRS数据导入DWS后，再查询数据。

步骤1 在DWS数据库中，创建导入数据的目标表，用于存储导入的数据。

版权所有 © 华为云计算技术有限公司

该表的表结构必须与**创建外表**中创建的外表的表结构保持一致，即字段个数、字段类型要完全一致。

例如，创建一个名为product_info的表，示例如下：

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

步骤2 执行“INSERT INTO .. SELECT ..”命令从外表导入数据到目标表。

示例：

```
INSERT INTO product_info SELECT * FROM foreign_product_info;
```

若出现以下类似信息，说明数据导入成功。

```
INSERT 0 20
```

步骤3 执行SELECT命令，查看从MRS导入到DWS中的数据。

```
SELECT * FROM product_info;
```

查询结果显示如**数据文件**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(20 rows)
```

----结束

错误处理

数据导入过程中，对于如下错误信息，表示DWS期望读取ORC数据文件，但实际却是*.txt类型的数据文件。请先参见**在MRS集群上创建Hive ORC表**创建Hive ORC类型的表，并将数据存储到该Hive ORC表中。

```
ERROR:  dn_6009_6010: Error occurs while creating an orc reader for file /user/hive/warehouse/
products_info.txt, detail can be found in dn log of dn_6009_6010.
```

2.3.3.6 清除资源

当完成本教程的示例后，如果您不再需要使用本示例中创建的资源，您可以删除这些资源，以免资源浪费或占用您的配额。

删除外表和目标表

步骤1 （可选）如果执行了**导入数据后查询数据**，请执行以下命令，删除目标表。

```
DROP TABLE product_info;
```


版权所有 © 华为云计算技术有限公司

步骤2 执行以下命令，删除外表。

```
DROP FOREIGN TABLE foreign_product_info;
```

----结束

删除手动创建的外部服务器

如果执行了**手动创建外部服务器**，请按照以下步骤删除外部服务器、数据库和用户。

步骤1 使用创建外部服务器的用户通过DWS提供的数据库客户端连接到外部服务器所在的数据库。

例如，使用gsql客户端的用户可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令进行切换：
  ```
  \c mydatabase dbuser;
  ```
  根据提示输入密码。
- 如果已经登录了gsql客户端，您也可以执行**\q**退出gsql后，再执行以下命令重新进行连接：
  ```
  gsql -d mydatabase -h 192.168.2.30 -U dbuser -p 8000 -r
  ```
  根据提示输入密码。

步骤2 删除手动创建的外部服务器。

执行以下命令进行删除：

```
DROP SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca;
```

返回以下信息表示删除成功：

```
DROP SERVER
```

查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

返回结果如下所示，表示已经删除成功：

```
 srvname | srvowner | srvfdw | srvtype | srvversion | srvacl | srvoptions
---------+----------+--------+---------+------------+--------+------------
(0 rows)
```

步骤3 删除自定义数据库。

通过DWS提供的数据库客户端连接默认数据库postgres。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

```
\c postgres
```

根据界面提示输入密码。

执行以下命令，删除自定义数据库：

```
DROP DATABASE mydatabase;
```

返回以下信息表示删除成功：

```
DROP DATABASE
```

步骤4 使用管理员用户，删除本示例中创建的普通用户。

使用数据库管理员用户通过DWS提供的数据库客户端连接数据库。

版权所有 © 华为云计算技术有限公司

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

```
\c postgres dbadmin
```

执行以下命令回收创建外部服务器的权限：

```
REVOKE ALL ON FOREIGN DATA WRAPPER hdfs_fdw FROM dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是hdfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令删除用户：

```
DROP USER dbuser;
```

可使用**\du**命令查询用户，确认用户是否已经删除。

----结束

2.3.4 基于 GDS 的跨集群互联互通

功能描述

在“基于Foreign Table的数据处理”的基础上，通过GDS进行数据中转，实现多个集群之间的数据同步。

使用场景

- 将数据从一个集群同步到另外一个集群，支持全量数据同步、过滤条件数据同步。
- 目前互联互通仅支持以下使用方式，除以下语句外，其他类型的语法均不支持。
  - INSERT INTO 内表 SELECT ... FROM 互联互通外表1 [WHERE];
  - INSERT INTO 互联互通表 SELECT * FROM 内表1 [JOIN 内表2 | WHERE];
  - SELECT ... FROM 互联互通表;

注意事项

- 创建的互联互通外表与其对应的远端表的列名和类型名要完全一致，且远端表的类型为行存表或列存表。
- 执行同步语句时，要确保本地集群、远端集群的待同步表已存在。
- 使用期间，两个集群的状态应为Normal。
- 两个集群都需要具备基于GDS的跨集群互联互通功能。
- 建议两端集群的数据库编码保持一致，否则可能出现报错或者收到的数据为乱码。
- 两端集群所指定的数据库兼容类型要保持一致，否则可能报错或乱码。
- 确保执行数据同步的相关用户对待同步表有相应的访问权限。
- 互联互通外表只能用于跨集群数据同步场景，其他场景可能出错或无效。
- 互联互通外表不支持复杂的列上表达式，不支持复杂语法，包括join、排序、游标、with、集合等。
- 不下推的SQL语句无法使用本特性进行数据同步，否则会报错。
- 不支持EXPLAIN计划、逻辑集群。
- 不支持除了simple模式以外的JDBC模式。

版权所有 © 华为云计算技术有限公司

- 当本地集群同步数据到远端集群时，只支持内表查询。
- Foreign Server的syncsrv选项指定的GDS不支持SSL模式。
- 数据同步结束时只校验数据行数，不校验数据内容。
- 业务最大并发数不能大于GDS启动参数-t的一半，同时也不能大于max_active_statements，否则可能会导致业务超时失败。
- 互联互通在扩容期间因数据存在跨group重分布，数据同步性能会暂时下降，扩容结束后性能恢复。

使用前准备

- 配置两个集群互连。
- 规划部署GDS服务器，确保所有的GDS服务器可以和上面配置的两个集群所有节点网络连通，即GDS服务器的安全组入方向要放通对应的GDS端口（例如5000）和DWS端口（默认8000）。部署GDS请参考**安装配置和启动GDS**。

  说明

  启动GDS时，可指定任意目录作为数据中转的目录，例如/opt，启动命令示例如下：

  ```
  /opt/gds/bin/gds -d /opt -p 192.168.0.2:5000 -H 192.168.0.1/24 -l /opt/gds/bin/gds_log.txt -D -t 2
  ```

操作步骤

假设远端集群的待同步表名称是tbl_remote，用于数据同步的用户是user_remote，该用户须对表tbl_remote有访问权限；假设本地集群的待同步表名称是tbl_local。

步骤1 创建server。

```
CREATE SERVER server_remote FOREIGN DATA WRAPPER GC_FDW OPTIONS(
  address '192.168.178.207:8000',
  dbname 'db_remote',
  username 'user_remote',
  password 'xxxxxxxx',
  syncsrv 'gsfs://192.168.178.129:5000|gsfs://192.168.178.129:5000'
);
```

- server_remote为server名称，供互联互通外表使用。
- address为远端集群CN的IP地址和端口，仅允许填写一个地址。
- dbname为远端集群的数据库名。
- username为连接远端集群使用的用户名，注意该用户不能为系统管理员。
- password为连接远端集群使用的用户名的密码。
- syncsrv为GDS Server的IP地址和端口，如果有多个地址使用|分割，与GDS外表的location类似。

  说明

  DWS会对syncsrv所设置的GDS地址进行网络连接测试：

  - 只能判断本地执行集群与GDS的网络情况，无法判断远端集群与GDS的网络情况，需要注意报错提示。
  - 在移除不可用GDS后，从中选择不会导致业务hang的、数目适当的GDS进行数据同步。

步骤2 创建互联互通外表。

```
CREATE FOREIGN TABLE ft_tbl(
  col_1    type_name,
```

 版权所有 © 华为云计算技术有限公司

```
  col_2    type_name,
  ...
) SERVER server_remote OPTIONS (
  schema_name 'schema_remote',
  table_name 'tbl_remote',
  encoding 'utf8'
);
```

- schema_name为远端集群表所属schema，如果该option缺省，则schema_name预设为该外表所在的schema。
- table_name为远端集群表名，如果该option缺省，则table_name预设为该外表的表名。
- encoding为远端集群的编码，如果该option缺省，则编码使用本地集群数据库的默认编码。

说明

- 选项schema_name、table_name大小写敏感，必须与远端schema、table的名字大小写保持一致。
- 互联互通外表的列不允许带任何约束。
- 互联互通外表的列名、列类型必须与远端集群的表tbl_remote的列名和列类型完全一致。
- SERVER须设置为**步骤1**中新建的server，必须包含syncsrv属性。

步骤3 使用互联互通外表进行数据同步。

- 本地集群是目标集群时，发起数据同步业务：

  全列全量数据同步：

  ```
  INSERT INTO tbl_local SELECT * FROM ft_tbl;
  ```

  全列过滤条件数据同步：

  ```
  INSERT INTO tbl_local SELECT * FROM ft_tbl WHERE col_2 = XX;
  ```

  部分列全量数据同步：

  ```
  INSERT INTO tbl_local (col_1) SELECT col_1 FROM ft_tbl;
  ```

  部分列过滤条件数据同步：

  ```
  INSERT INTO tbl_local (col_1) SELECT col_1 FROM ft_tbl WHERE col_2 = XX;
  ```

- 本地集群是源集群时，发起数据同步业务：

  单表数据同步：

  ```
  INSERT INTO ft_tbl SELECT * FROM tbl_local;
  ```

  join结果集数据同步：

  ```
  INSERT INTO ft_tbl SELECT * FROM tbl_local1 join tbl_local2 ON XXX;
  ```


版权所有 © 华为云计算技术有限公司

说明

- 如遇到报错连接失败，请检查server的信息确认两个集群是否已经相互连通。
- 如遇到报错GDS连接失败，请检查syncsrv指定的GDS Server是否都已经启动，且与两个集群所有节点可以网络连通。
- 如遇到报错表不存在，请检查外表的option信息是否正确。
- 如遇到报错列不存在，请检查外表的列名是否与源表一致。
- 如遇到报错列重复定义，请检查是否相应列名超长，若超长建议使用AS别名精简。
- 如遇到报错无法解析列类型，请检查语句中是否有列上表达式。
- 如遇到报错列信息不匹配，请检查外表的列信息是否与远端集群对应表的列信息是否一致。
- 如遇到报错语法不支持，请检查是否使用了Join、distinct、排序等复杂用法。
- 如遇到乱码，请检查两端数据库的实际编码是否一致。
- 当本地集群是源集群时，存在极小的概率出现数据成功同步到远端集群，但是本地集群返回执行失败的情况，针对这种情况建议校验同步数据记录数。
- 当本地集群是源集群时，通过事务块、子事务等控制的数据同步，需要总事务提交后才能查询到数据同步结果。

步骤4 删除互联互通外表。

```
DROP FOREIGN TABLE ft_tbl;
```

----结束

2.3.5 使用 gsql 元命令\COPY 导入数据

DWS的gsql工具提供了元命令\copy进行数据导入。

\copy 命令

\copy命令格式以及说明参见**表 1 \copy元命令说明**。

表 2-18 \copy 元命令说明

语法	说明
\copy { table [(column_list)] | (query) } { from | to } { filename | stdin | stdout | pstdin | pstdout } [with] [binary] [oids] [delimiter [as] 'character'] [null [as] 'string'] [csv [header] [quote [as] 'character'] [escape [as] 'character'] [force quote column_list | *] [force not null column_list]]	在任何gsql客户端登录数据库成功后，可以使用该命令进行数据的导入/导出。但是与SQL的COPY命令不同，该命令读取/写入的文件是本地文件，而非数据库服务器端文件；所以，要操作的文件的可访问性、权限等，都是受限于本地用户的权限。 **说明** \COPY只适合小批量，格式良好的数据导入，容错能力较差。导入数据应优先选择GDS或COPY。

参数说明

- table

版权所有 © 华为云计算技术有限公司

表的名字（可以有模式修饰）。

取值范围：已存在的表名。

- column_list

  可选的待拷贝字段列表。

  取值范围：任意字段。如果没有声明字段列表，将使用所有字段。

- query

  其结果将被拷贝。

  取值范围：一个必须用圆括弧包围的SELECT或VALUES命令。

- filename

  文件的绝对路径。执行copy命令的用户必须有此路径的写权限。

- stdin

  声明输入是来自标准输入。

- stdout

  声明输出打印到标准输出。

- pstdin

  声明输入是来自gsql的标准输入。

- pstdout
- 声明输出打印到gsql的标准输出。
- binary

  使用二进制格式存储和读取，而不是以文本的方式。在二进制模式下，不能声明DELIMITER，NULL，CSV选项。指定binary类型后，不能再通过option或copy_option指定CSV、FIXED、TEXT等类型。

- oid

  为每行拷贝内部对象标识（oid）。

  说明

  若COPY FROM对象为query或者对于没有oid的表，指定oids标识报错。

  取值范围：true/on，false/off。

  默认值：false/off。

- delimiter [as] 'character'

  指定数据文件行数据的字段分隔符。

  说明

  - 分隔符不能是\r和\n。
  - 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。
  - TEXT格式数据的分隔符不能包含： \.abcdefghijklmnopqrstuvwxyz0123456789。
  - 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。
  - 分隔符推荐使用多字符和不可见字符。多字符例如'$^&'；不可见字符例如E'\x07'，E'\x08'，E'\x1b'等。

  取值范围：支持多字符分隔符，但分隔符不能超过10个字节。

  默认值：

版权所有 © 华为云计算技术有限公司

  - TEXT格式的默认分隔符是水平制表符（tab）。
  - CSV格式的默认分隔符为“,”。
  - FIXED格式没有分隔符。
- null [as] 'string'

  用来指定数据文件中空值的表示。

  取值范围：
  - null值不能是\r和\n，最大为100个字符。
  - null值不能和分隔符、quote参数相同。

  默认值：
  - CSV格式下默认值是一个没有引号的空字符串。
  - 在TEXT格式下默认值是\N。
- header

  指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV，FIXED格式的文件中。

  在导入数据时，如果header选项为on，则数据文本第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。

  在导出数据时，如果header选项为on，则需要指定fileheader。fileheader是指定导出数据包含标题行的定义文件。如果header为off，则导出数据文件不包含标题行。

  取值范围：true/on，false/off。

  默认值：false/off。
- quote [as] 'character'

  CSV格式文件下，指定一个数据值被引用时使用的引用字符。

  默认值：双引号。

  说明
  - quote参数不能和分隔符、null参数相同。
  - quote参数只能是单字节的字符。
  - 推荐不可见字符作为quote，例如E'\x07'，E'\x08'，E'\x1b'等。
- escape [as] 'character'

  CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。

  默认值：双引号。当与quote值相同时，会被替换为'\0'。
- force quote column_list | *

  在CSV COPY TO模式下，强制对每个声明的字段的对所有非NULL值都使用引用字符。NULL输出不会被引用。

  取值范围：已存在的字段。
- force not null column_list

  在CSV COPY FROM模式下，指定的字段输入不能为空。

  取值范围：已存在的字段。

示例

创建目标表copy_example。

 版权所有 © 华为云计算技术有限公司

```
create table copy_example
(
    col_1 integer,
    col_2 text,
    col_3 varchar(12),
    col_4 date,
    col_5 time
);
```

- 示例一：从stdin拷贝数据到目标表copy_example。
  ```
  \copy copy_example from stdin csv;
  ```
  出现>>符号提示时，输入数据，输入\.时结束。
  ```
  Enter data to be copied followed by a newline.
  End with a backslash and a period on a line by itself.
  >> 1,"iamtext","iamvarchar",2006-07-07,12:00:00
  >> \.
  ```
- 示例二：在本地目录'/local/data/'下有example.csv文件，包含header行，使用'|'作为delimiter，使用双引号作为quote。内容如下：

  iamheader

  1|"iamtext"|"iamvarchar"|2006-07-07|12:00:00

  2|"iamtext"|"iamvarchar"|2022-07-07|19:00:02

  从本地文件example.csv导入数据到目标表copy_example，header选项为'on'，自动忽略第一行。quote默认为双引号，因此可以不用指定。
  ```
  \copy copy_example from  '/local/data/example.csv' with(header 'on', format 'csv', delimiter '|', date_format 'yyyy-mm-dd',  time_format 'hh24:mi:ss');
  ```
- 示例三：在本地目录'/local/data/'下有example.csv文件，使用','作为delimiter，使用双引号作为quote，其中第一行缺少最后一个字段，第二行多一个字段。内容如下

  1,"iamtext","iamvarchar",2006-07-07

  2,"iamtext","iamvarchar",2022-07-07,19:00:02,12:00:00

  从本地文件example.csv导入数据到目标表copy_example，delimiter默认为','，因此可以不用指定，由于指定了容错参数IGNORE_EXTRA_DATA和FILL_MISSING_FIELDS，缺少的字段会用NULL替换，多出的字段被忽略。
  ```
  \copy copy_example from  '/local/data/example.csv' with( format 'csv', date_format 'yyyy-mm-dd', time_format 'hh24:mi:ss', IGNORE_EXTRA_DATA 'true', FILL_MISSING_FIELDS 'true');
  ```
- 示例四：将copy_example表的内容导出到stdout，格式为csv，使用双引号作为quote，第四列和第五列强制使用quote包围；
  ```
  \copy copy_example to stdout CSV quote as '"' force quote col_4,col_5;
  ```

2.3.6 使用 COPY FROM STDIN 导入数据

2.3.6.1 关于 COPY FROM STDIN 导入数据

这种方式适合数据写入量不太大， 并发度不太高的场景。

用户可以使用以下方式通过COPY FROM STDIN语句直接向DWS写入数据。

- 通过键盘输入向DWS写入数据。
- 通过JDBC驱动的CopyManager接口从文件或者数据库向DWS写入数据。此方法支持COPY语法中copy option的所有参数。

 版权所有 © 华为云计算技术有限公司

2.3.6.2 CopyManager 类简介

CopyManager是DWS JDBC驱动中提供的一个API接口类，用于批量向DWS集群中导入数据。

CopyManager 的继承关系

CopyManager类位于org.postgresql.copy Package中，继承自java.lang.Object类，该类的声明如下：

```
public class CopyManager
extends Object
```

构造方法

public CopyManager(BaseConnection connection)

throws SQLException

常用方法

表 2-19 CopyManager 常用方法

返回值	方法	描述	throws
CopyIn	copyIn(String sql)	-	SQLException
long	copyIn(String sql, InputStream from)	使用COPY FROM STDIN从InputStream中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, InputStream from, int bufferSize)	使用COPY FROM STDIN从InputStream中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, Reader from)	使用COPY FROM STDIN从Reader中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, Reader from, int bufferSize)	使用COPY FROM STDIN从Reader中快速向数据库中的表导入数据。	SQLException,IOException
CopyOut	copyOut(String sql)	-	SQLException

版权所有 © 华为云计算技术有限公司

返回值	方法	描述	throws
long	copyOut(String sql, OutputStream to)	将一个COPY TO STDOUT的结果集从数据库发送到OutputStream类中。	SQLException,IOException
long	copyOut(String sql, Writer to)	将一个COPY TO STDOUT的结果集从数据库发送到Writer类中。	SQLException,IOException

2.4 整库迁移

2.4.1 使用 CDM 迁移数据到 DWS

使用云数据迁移服务（Cloud Data Migration，简称CDM），可以将其他数据源（例如MySQL）的数据迁移到DWS集群的数据库中。

CDM提供同构/异构数据源之间批量数据迁移的功能，帮助用户实现从多种类型的数据源迁移数据到DWS。

说明

CDM在迁移数据到DWS时，采用的是COPY方式和GDS并行导入方式。

使用CDM迁移数据到DWS的典型场景，请参见《DataArts Studio用户指南》中的“DataArts迁移”章节。

图 2-6 使用 CDM 迁移数据到 DWS

2.4.2 使用 DSC 工具迁移 SQL 脚本

DSC（Database Schema Convertor）是一款运行在Linux或Windows操作系统上的命令行工具，致力于向客户提供简单、快速、可靠的应用程序SQL脚本迁移服务，通过内置的语法迁移逻辑解析源数据库应用程序SQL脚本，并迁移为适用于DWS 数据库的应用程序SQL脚本。DSC不需要连接数据库，可在离线模式下实现零停机迁移。在DWS中通过执行迁移后的SQL脚本即可恢复数据库，从而实现线下数据库轻松上云。

DSC支持迁移Teradata、Oracle、Netezza、MySQL和DB2数据库的SQL脚本。

下载 DSC SQL 语法迁移工具

步骤1 登录DWS控制台。

版权所有 © 华为云计算技术有限公司

步骤2 在左侧导航栏中，单击“管理 > 连接客户端”。

步骤3 在“下载客户端和驱动”区域，单击“这里”即可下载DSC迁移工具。

如果同时拥有不同版本的集群，系统会弹出对话框，提示您选择“集群版本”然后下载与集群版本相对应的客户端。在“集群管理”页面的集群列表中，单击指定集群的名称，再选择“基本信息”页签，可查看集群版本。

步骤4 下载到本机后，使用WinSCP工具，将DSC工具上传到一个需安装工具的Linux主机上。

执行上传操作的用户需要对Linux主机的目标存放目录有完全控制权限。

----结束

DSC SQL 语法迁移工具操作指导

详细指导请参见《数据仓库服务工具指南》中的“DSC SQL语法迁移工具”。

2.5 元数据迁移

2.5.1 使用 gs_dump 和 gs_dumpall 命令导出元数据

2.5.1.1 概述

DWS提供的gs_dump和gs_dumpall工具，能够帮助用户导出需要的数据库对象或其相关信息。通过导入工具将导出的元数据信息导入至需要的数据库，可以完成数据库信息的迁移。gs_dump支持导出单个数据库或其内的对象，而gs_dumpall支持导出集群中所有数据库或各库的公共全局对象。详细的使用场景见**表2-20**。

表 2-20 适用场景

适用场景	支持的导出粒度	支持的导出格式	配套的导入方法
导出单个数据库	**数据库级导出**。 • 导出全量信息。 使用导出的全量信息可以创建一个与当前库相同的数据库，且库中数据也与当前库相同。 • 仅导出库中所有对象的定义，包含库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。 使用导出的对象定义，可以快速创建一个相同的数据库，但是库中并无原数据库的数据。 • 仅导出数据。	• 纯文本格式 • 自定义归档格式 • 目录归档格式 • tar归档格式	• 纯文本格式数据文件导入请参见**使用gsql元命令\COPY导入数据**。 • 自定义归档格式、目录归档格式和tar归档格式数据文件导入请参见**使用gs_restore导入数据**。

版权所有 © 华为云计算技术有限公司

适用场景	支持的导出粒度	支持的导出格式	配套的导入方法
	模式级导出。 ● 导出模式的全量信息。 ● 仅导出模式中数据。 ● 仅导出对象的定义，包含表定义、存储过程定义和索引定义等。		
	表级导出。 ● 导出表的全量信息。 ● 仅导出表中数据。 ● 仅导出表的定义。		
导出所有数据库	**数据库级导出**。 ● 导出全量信息。 使用导出的全量信息可以创建与当前集群相同的一个集群，拥有相同数据库和公共全局对象，且库中数据也与当前各库相同。 ● 仅导出各数据库中的对象定义，包含表空间、库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。 使用导出的对象定义，可以快速创建与当前集群相同的一个集群，拥有相同的数据库和表空间，但是库中并无原数据库的数据。 ● 仅导出数据。	纯文本格式	数据文件导入请参见**使用gsql元命令\COPY导入数据**。
	各库公共全局对象导出 ● 仅导出表空间信息。 ● 仅导出角色信息。 ● 导出角色与表空间。		

gs_dump和gs_dumpall通过-U指定执行导出的用户账户。如果当前使用的账户不具备导出所要求的权限时，会无法导出数据。此时，可在导出命令中设置--role参数来指定具备权限的角色。在执行命令后，gs_dump和gs_dumpall会使用--role参数指定的角色，完成导出动作。可使用该功能的场景请参见**表2-20**，详细操作请参见**无权限角色导出数据**。

gs_dump和gs_dumpall通过对导出的数据文件加密，导入时对加密的数据文件进行解密，可以防止数据信息泄露，为数据库的安全提供保证。

gs_dump和gs_dumpall工具在进行数据导出时，导出的表会被加锁，会导致阻塞读写。

版权所有 © 华为云计算技术有限公司

gs_dump和gs_dumpall工具支持导出完整一致的数据。例如，T1时刻启动gs_dump导出A数据库，或者启动gs_dumpall导出整个集群数据库，那么导出数据结果将会是T1时刻A数据库或者该集群数据库的数据状态，T1时刻之后对A数据库或集群数据库的修改不会被导出。

gs_dump和gs_dumpall工具是通过“gsql命令行客户端”软件包解压缩获取。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，导出工具会对需要转储的表设置共享锁。如果表在别的事务中设置了共享锁，gs_dump和gs_dumpall会等待锁释放后锁定表。如果无法在指定时间内锁定某个表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。
- 由于gs_dumpall读取所有数据库中的表，因此必须以数据库集群管理员身份进行连接，才能导出完整文件。在使用gsql执行脚本文件导入时，同样需要管理员权限，以便添加用户和组，以及创建数据库。
- 由于DWS数据库所有视图的定义都默认带有表名或别名的前缀（即tab.col的形式），因此可能与原始定义不符，导致在极少的场景下会发生重建视图字段对应基表不准确而报错的情况。为避免此情况建议在导出视图定义时，设置guc参数behavior_compat_options='compat_display_ref_table'，使导出定义与原始语句一致。

2.5.1.2 导出单个数据库

2.5.1.2.1 导出数据库

DWS支持使用gs_dump工具导出某个数据库级的内容，包含数据库的数据和所有对象定义。可根据需要自定义导出如下信息：

- 导出数据库全量信息，包含数据和所有对象定义。

  使用导出的全量信息可以创建一个与当前库相同的数据库，且库中数据也与当前库相同。
- 仅导出所有对象定义，包括：库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。

  使用导出的对象定义，可以快速创建一个相同的数据库，但是库中并无原数据库的数据。
- 仅导出数据，不包含所有对象定义。

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤3 执行以下命令解压客户端工具。

 版权所有 © 华为云计算技术有限公司

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dump导出gaussdb数据库。

gs_dump -W *password* **-U** *jack* **-f** */home//backup/postgres_backup.tar* **-p 8000** gaussdb *-h 10.10.10.100* **-F** *t*

表 2-21 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，如果未填写则表示当前已连接的数据库用户。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*，此处密码需要用户自定义。
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home//backup/*postgres*_backup.tar
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
dbname	需要导出的数据库名称	gaussdb

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F t

其他参数说明请参见《数据仓库服务工具指南》中“gs_dump”章节。

----结束

示例

示例一：执行gs_dump，导出gaussdb数据库全量信息，并对导出文件进行压缩，导出文件格式为sql文本格式。

```
gs_dump -W password -U jack -f /home//backup/postgres_backup.sql -p 8000  -h 10.10.10.100 gaussdb -Z 8 -F p
gs_dump[port='']['gaussdb'][2017-07-21 15:36:13]: dump database gaussdb successfully
gs_dump[port='']['gaussdb'][2017-07-21 15:36:13]: total time: 3793  ms
```

示例二：执行gs_dump，仅导出gaussdb数据库中的数据，不包含数据库对象定义，导出文件格式为自定义归档格式。

```
gs_dump -W Password -U jack -f /home//backup/postgres_data_backup.dmp -p 8000 -h 10.10.10.100 gaussdb -a -F c
gs_dump[port='']['gaussdb'][2017-07-21 15:36:13]: dump database gaussdb successfully
gs_dump[port='']['gaussdb'][2017-07-21 15:36:13]: total time: 3793  ms
```

示例三：执行gs_dump，仅导出gaussdb数据库所有对象的定义，导出文件格式为sql文本格式。

```
--导出前，表nation有数据
select n_nationkey,n_name,n_regionkey from nation limit 3;
 n_nationkey |          n_name           | n_regionkey
-------------+---------------------------+-------------
           0 | ALGERIA                   |           0
           3 | CANADA                    |           1
          11 | IRAQ                      |           4
(3 rows)

gs_dump -W password -U jack -f /home//backup/postgres_def_backup.sql -p 8000 -h 10.10.10.100 gaussdb -s -F p
gs_dump[port='']['gaussdb'][2017-07-20 15:04:14]: dump database gaussdb successfully
gs_dump[port='']['gaussdb'][2017-07-20 15:04:14]: total time: 472 ms
```

示例四：执行gs_dump，仅导出gaussdb数据库的所有对象的定义，导出文件格式为文本格式，并对导出文件进行加密。

```
gs_dump -W password -U jack -f /home//backup/postgres_def_backup.sql -p 8000 -h 10.10.10.100 gaussdb --with-encryption AES128 --with-key 1234567812345678 -s -F p
gs_dump[port='']['gaussdb'][2018-11-14 11:25:18]: dump database gaussdb successfully
gs_dump[port='']['gaussdb'][2018-11-14 11:25:18]: total time: 1161  ms
```


版权所有 © 华为云计算技术有限公司

2.5.1.2.2 导出模式

DWS目前支持使用gs_dump工具导出模式级的内容，包含模式的数据和定义。用户可通过灵活的自定义方式导出模式内容，不仅支持选定一个模式或多个模式的导出，还支持排除一个模式或者多个模式的导出。可根据需要自定义导出如下信息：

- 导出模式全量信息，包含数据和对象定义。
- 仅导出数据，即模式包含表中的数据，不包含对象定义。
- 仅导出模式对象定义，包括：表定义、存储过程定义和索引定义等。

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤3 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dump同时导出hr和public模式。

```
gs_dump -W Password -U jack -f /home//backup/MPPDB_schema_backup -p 8000 -h 10.10.10.100 human_resource -n hr -F d
```

表 2-22 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，如果未填写则表示当前已连接的数据库用户。	-U jack
-W	指定用户连接的密码。 • 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 • 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*，此处密码需要用户自定义。

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home//backup/MPPDB_schema_backup
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
dbname	需要导出的数据库名称	human_resource
-n	只导出与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。 ● 单个模式：-n *schemaname* ● 多个模式：多次输入-n *schemaname*	● 单个模式：-n hr ● 多个模式：-n hr -n public
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F d

其他参数说明请参见《数据仓库服务工具指南》中“gs_dump”章节。

----结束

示例

示例一：执行gs_dump，导出hr模式全量信息，并对导出文件进行压缩，导出文件格式为文本格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_backup.sql -p 8000 -h 10.10.10.100
human_resource -n hr -Z 6 -F p
gs_dump[port='']human_resource][2017-07-21 16:05:55]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 16:05:55]: total time: 2425  ms
```

示例二：执行gs_dump，仅导出hr模式的数据，导出文件格式为tar归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_data_backup.tar -p 8000 -h 10.10.10.100
human_resource -n hr -a -F t
gs_dump[port='']human_resource][2018-11-14 15:07:16]: dump database human_resource successfully
gs_dump[port='']human_resource][2018-11-14 15:07:16]: total time: 1865  ms
```

示例三：执行gs_dump，仅导出hr模式的定义，导出文件格式为目录归档格式。

版权所有 © 华为云计算技术有限公司

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_def_backup -p 8000 -h 10.10.10.100
human_resource -n hr -s -F d
gs_dump[port='']human_resource][2018-11-14 15:11:34]: dump database human_resource successfully
gs_dump[port='']human_resource][2018-11-14 15:11:34]: total time: 1652  ms
```

示例四：执行gs_dump，导出human_resource数据库时，排除hr模式，导出文件格式为自定义归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_backup.dmp -p 8000 -h 10.10.10.100
human_resource -N hr -F c
gs_dump[port='']human_resource][2017-07-21 16:06:31]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 16:06:31]: total time: 2522  ms
```

示例五：执行gs_dump，同时导出hr和public模式，且仅导出模式定义，并对导出文件进行加密，导出文件格式为tar归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_backup1.tar -p 8000 -h 10.10.10.100
human_resource -n hr -n public -s --with-encryption AES128 --with-key 1234567812345678 -F t
gs_dump[port='']human_resource][2017-07-21 16:07:16]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 16:07:16]: total time: 2132  ms
```

示例六：执行gs_dump，导出human_resource数据库时，排除hr和public模式，导出文件格式为自定义归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_schema_backup2.dmp -p 8000 -h 10.10.10.100
human_resource -N hr -N public -F c
gs_dump[port='']human_resource][2017-07-21 16:07:55]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 16:07:55]: total time: 2296  ms
```

示例七：执行gs_dump，导出public模式下所有表（视图、序列和外表）和hr模式中staffs表，包含数据和表定义，导出文件格式为自定义归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_backup3.dmp -p 8000 -h 10.10.10.100
human_resource -t public.* -t hr.staffs -F c
gs_dump[port='']human_resource][2018-12-13 09:40:24]: dump database human_resource successfully
gs_dump[port='']human_resource][2018-12-13 09:40:24]: total time: 896  ms
```

2.5.1.2.3 导出表

DWS支持使用gs_dump工具导出表级的内容，包含表定义和表数据。视图、序列和外表属于特殊的表。用户可通过灵活的自定义方式导出表内容，不仅支持选定一个表或多个表的导出，还支持排除一个表或者多个表的导出。可根据需要自定义导出如下信息：

- 导出表全量信息，包含表数据和表定义。
- 仅导出数据，不包含表定义。
- 仅导出表定义。

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤3 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

 版权所有 © 华为云计算技术有限公司

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dump同时导出指定表hr.staffs和hr.employments。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup -p 8000 -h 10.10.10.100 human_resource -t hr.staffs -F d
```

表 2-23 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，如果未填写则表示当前已连接的数据库用户。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home//backup/MPPDB_table_backup
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
dbname	需要导出的数据库名称	human_resource

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例
-t	指定导出的表（或视图、序列、外表），可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。当使用通配符指定多个表对象时，注意给pattern打引号，防止shell扩展通配符。 ● 单个表：-t *schema.table* ● 多个表：多次输入-t *schema.table*	● 单个表：-t hr.staffs ● 多个表：-t hr.staffs -t hr.employments
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F d

其他参数说明请参见《数据仓库服务工具指南》中“gs_dump”章节。

----结束

示例

示例一：执行gs_dump，导出表hr.staffs的定义和数据，并对导出文件进行压缩，导出文件格式为文本格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup.sql -p 8000 -h 10.10.10.100
human_resource -t hr.staffs -Z 6 -F p
gs_dump[port='']human_resource][2017-07-21 17:05:10]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 17:05:10]: total time: 3116  ms
```

示例二：执行gs_dump，只导出表hr.staffs的数据，导出文件格式为tar归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_data_backup.tar -p 8000 -h 10.10.10.100
human_resource -t hr.staffs -a -F t
gs_dump[port='']human_resource][2017-07-21 17:04:26]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 17:04:26]: total time: 2570  ms
```

示例三：执行gs_dump，导出表hr.staffs的定义，导出文件格式为目录归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_def_backup -p 8000 -h 10.10.10.100
human_resource -t hr.staffs -s -F d
gs_dump[port='']human_resource][2017-07-21 17:03:09]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 17:03:09]: total time: 2297  ms
```

示例四：执行gs_dump，不导出表hr.staffs，导出文件格式为自定义归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup4.dmp -p 8000 -h 10.10.10.100
human_resource -T hr.staffs -F c
gs_dump[port='']human_resource][2017-07-21 17:14:11]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 17:14:11]: total time: 2450  ms
```

示例五：执行gs_dump，同时导出两个表hr.staffs和hr.employments，导出文件格式为文本格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup1.sql -p 8000 -h 10.10.10.100
human_resource -t hr.staffs -t hr.employments -F p
gs_dump[port='']human_resource][2017-07-21 17:19:42]: dump database human_resource successfully
gs_dump[port='']human_resource][2017-07-21 17:19:42]: total time: 2414  ms
```


版权所有 © 华为云计算技术有限公司

示例六：执行gs_dump，导出时，排除两个表hr.staffs和hr.employments，导出文件格式为文本格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup2.sql -p 8000 -h 10.10.10.100
human_resource -T hr.staffs -T hr.employments -F p
gs_dump[port='']['human_resource'][2017-07-21 17:21:02]: dump database human_resource successfully
gs_dump[port='']['human_resource'][2017-07-21 17:21:02]: total time: 3165  ms
```

示例七：执行gs_dump，导出表hr.staffs的定义和数据，只导出表hr.employments的定义，导出文件格式为tar归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup3.tar -p 8000 -h 10.10.10.100
human_resource -t hr.staffs -t hr.employments --exclude-table-data hr.employments -F t
gs_dump[port='']['human_resource'][2018-11-14 11:32:02]: dump database human_resource successfully
gs_dump[port='']['human_resource'][2018-11-14 11:32:02]: total time: 1645  ms
```

示例八：执行gs_dump，导出表hr.staffs的定义和数据，并对导出文件进行加密，导出文件格式为文本格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup4.sql -p 8000 -h 10.10.10.100
human_resource -t hr.staffs --with-encryption AES128 --with-key 1212121212121212 -F p
gs_dump[port='']['human_resource'][2018-11-14 11:35:30]: dump database human_resource successfully
gs_dump[port='']['human_resource'][2018-11-14 11:35:30]: total time: 6708  ms
```

示例九：执行gs_dump，导出public模式下所有表（包括视图、序列和外表）和hr模式中staffs表，包含数据和表定义，导出文件格式为自定义归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_table_backup5.dmp -p 8000 -h 10.10.10.100
human_resource -t public.* -t hr.staffs -F c
gs_dump[port='']['human_resource'][2018-12-13 09:40:24]: dump database human_resource successfully
gs_dump[port='']['human_resource'][2018-12-13 09:40:24]: total time: 896  ms
```

示例十： 执行gs_dump，仅导出依赖于t1模式下的test1表对象的视图信息，导出文件格式为目录归档格式。

```
gs_dump -W password -U jack -f /home//backup/MPPDB_view_backup6 -p 8000 -h 10.10.10.100
human_resource -t t1.test1 --include-depend-objs --exclude-self -F d
gs_dump[port='']['jack'][2018-11-14 17:21:18]: dump database human_resource successfully
gs_dump[port='']['jack'][2018-11-14 17:21:23]: total time: 4239  ms
```

2.5.1.3 导出所有数据库

2.5.1.3.1 导出所有数据库

DWS支持使用gs_dumpall工具导出所有数据库的全量信息，包含集群中每个数据库信息和公共的全局对象信息。可根据需要自定义导出如下信息：

- 导出所有数据库全量信息，包含集群中每个数据库信息和公共的全局对象信息（包含角色和表空间信息）。

  使用导出的全量信息可以创建与当前集群相同的一个集群，拥有相同数据库和公共全局对象，且库中数据也与当前各库相同。
- 仅导出数据，即导出每个数据库中的数据，且不包含所有对象定义和公共的全局对象信息。
- 仅导出所有对象定义，包括：表空间、库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。

  使用导出的对象定义，可以快速创建与当前集群相同的一个集群，拥有相同的数据库和表空间，但是库中并无原数据库的数据。

版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤3 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dumpall一次导出所有数据库信息。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -h 10.10.10.100
```

表 2-24 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，需要是集群管理员用户。	-U dbadmin
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*，此处密码需要用户自定义。
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/MPPDB_backup.sql
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100

其他参数说明请参见《数据仓库服务工具指南》中“gs_dumpall”章节。

----结束

示例

示例一：执行gs_dumpall，导出所有数据库全量信息（dbadmin用户为管理员用户），导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -h 10.10.10.100
gs_dumpall[port=''][2017-07-21 15:57:31]: dumpall operation successful
gs_dumpall[port=''][2017-07-21 15:57:31]: total time: 9627  ms
```

示例二：执行gs_dumpall，仅导出所有数据库定义（dbadmin用户为管理员用户），导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -h 10.10.10.100 -s
gs_dumpall[port=''][2018-11-14 11:28:14]: dumpall operation successful
gs_dumpall[port=''][2018-11-14 11:28:14]: total time: 4147  ms
```

示例三：执行gs_dumpall，仅导出所有数据库中数据，并对导出文件进行加密，导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -h 10.10.10.100 -a --with-encryption AES128 --with-key 1234567812345678
gs_dumpall[port=''][2018-11-14 11:32:26]: dumpall operation successful
gs_dumpall[port=''][2018-11-14 11:23:26]: total time: 4147  ms
```

2.5.1.3.2 导出全局对象

DWS支持使用gs_dumpall工具导出所有数据库公共的全局对象，包含数据库用户和组，表空间及属性（例如：适用于数据库整体的访问权限）信息。

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

版权所有 © 华为云计算技术有限公司

步骤3 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dumpall导出表空间对象信息。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_tablespace.sql -p 8000 -h 10.10.10.100 -t
```

表 2-25 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，需要是集群管理员用户。	-U dbadmin
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	--W *password*，此处密码需要用户自定义。
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home//backup/*MPPDB_tablespace*.sql
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
-t	或者--tablespaces-only，只转储表空间，不转储数据库或角色。	-

版权所有 © 华为云计算技术有限公司

其他参数说明请参见《数据仓库服务工具指南》中“gs_dumpall”章节。

----结束

示例

示例一：执行gs_dumpall，导出所有数据库的公共全局表空间信息和用户信息（dbadmin用户为管理员用户），导出文件为文本格式。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_globals.sql -p 8000 -h 10.10.10.100 -g
gs_dumpall[port='']2018-11-14 19:06:24]: dumpall operation successful
gs_dumpall[port='']2018-11-14 19:06:24]: total time: 1150  ms
```

示例二： 执行gs_dumpall，导出所有数据库的公共全局表空间信息（dbadmin用户为管理员用户），并对导出文件进行加密，导出文件为文本格式。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_tablespace.sql -p 8000 -h 10.10.10.100 -t --with-encryption AES128 --with-key 1212121212121212
gs_dumpall[port='']2018-11-14 19:00:58]: dumpall operation successful
gs_dumpall[port='']2018-11-14 19:00:58]: total time: 186  ms
```

示例三：执行gs_dumpall，导出所有数据库的公共全局用户信息（dbadmin用户为管理员用户），导出文件为文本格式。

```
gs_dumpall -W password -U dbadmin -f /home/dbadmin/backup/MPPDB_user.sql -p 8000 -h 10.10.10.100 -r
gs_dumpall[port='']2018-11-14 19:03:18]: dumpall operation successful
gs_dumpall[port='']2018-11-14 19:03:18]: total time: 162  ms
```

2.5.1.4 无权限角色导出数据

gs_dump和gs_dumpall通过-U指定执行导出的用户账户。如果当前使用的账户不具备导出所要求的权限时，会无法导出数据。此时，可在导出命令中设置--role参数来指定具备权限的角色。在执行命令后，gs_dump和gs_dumpall会使用--role参数指定的角色，完成导出动作。

操作步骤

步骤1 准备ECS作为gsql客户端主机，具体操作请参见《数据仓库服务用户指南》中“使用Linux gsql客户端连接集群”章节的“准备ECS作为gsql客户端主机”相关内容。

步骤2 请参见《数据仓库服务用户指南》中的“下载客户端”下载gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装gsql的Linux主机上。

执行上传gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤3 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.x.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_8.1.x_redhat_x86.zip：这是“RedHat x86”对应的客户端工具包名称，请替换为实际下载的包名。

步骤4 执行以下命令配置客户端。

```
source gsql_env.sh
```

 版权所有 © 华为云计算技术有限公司

提示以下信息表示客户端已配置成功：

```
All things done.
```

步骤5 使用gs_dump导出human_resource数据库数据。

用户jack不具备导出数据库human_resource的权限，而角色role1具备该权限，要实现导出数据库human_resource，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
gs_dump -U jack -W password -f /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 human_resource --role role1 --rolepassword password -F t
```

表 2-26 常用参数说明

参数	参数说明	举例dbadmin
-U	连接数据库的用户名。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*，此处密码需要用户自定义。
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home//backup/MPPDB_backup.tar
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
dbname	需要导出的数据库名称	human_resource
--role	指定导出使用的角色名。选择该选项，会使导出工具连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有导出工具要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。	-r role1
--rolepassword	指定具体角色用户的角色密码。	--rolepassword password

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例dbadmin
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F t

其他参数说明请参见《数据仓库服务工具指南》中“gs_dump”或“gs_dumpall”章节。

----结束

示例

示例一：执行gs_dump导出数据，用户jack不具备导出数据库human_resource的权限，而角色role1具备该权限，要实现导出数据库human_resource，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "password";

gs_dump -U jack -W password -f /home//backup/MPPDB_backup11.tar -p 8000 -h 10.10.10.100
human_resource --role role1 --rolepassword password -F t
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: total time: 4239  ms
```

示例二：执行gs_dump导出数据，用户jack不具备导出模式public的权限，而角色role1具备该权限，要实现导出模式public，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "1234@abc";

gs_dump -U jack -W password -f /home//backup/MPPDB_backup12.tar -p 8000 -h 10.10.10.100
human_resource -n public --role role1 --rolepassword password -F t
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: total time: 3278  ms
```

示例三：执行gs_dumpall导出数据，用户jack不具备导出所有数据库的权限，而角色role1具备该权限，要实现导出所有数据库，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为文本归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "password";

gs_dumpall -U jack -W password -f /home//backup/MPPDB_backup.sql -p 8000 -h 10.10.10.100 --role role1
--rolepassword password
gs_dumpall[port='8000'][human_resource][2018-11-14 17:26:18]: dumpall operation successful
gs_dumpall[port='8000'][human_resource][2018-11-14 17:26:18]: total time: 6437  ms
```


版权所有 © 华为云计算技术有限公司

2.5.2 使用 gs_restore 导入数据

操作场景

gs_restore是DWS提供的与gs_dump配套的导入工具。通过该工具，可将gs_dump导出的文件导入至数据库。gs_restore支持导入的文件格式包含自定义归档格式、目录归档格式和tar归档格式。

gs_restore具备如下两种功能。

- 导入至数据库

  如果指定了数据库，则数据将被导入到指定的数据库中。其中，并行导入必须指定连接数据库的密码。

- 导入至脚本文件

  如果未指定导入数据库，则创建包含重建数据库所需的SQL语句脚本，并将其写入至文件或者标准输出。该脚本文件等效于gs_dump导出的纯文本格式文件。

gs_restore工具在导入时，允许用户选择需要导入的内容，并支持在数据导入前对等待导入的内容进行排序。

操作步骤

说明

gs_restore默认是以追加的方式进行数据导入。为避免多次导入造成数据异常，在进行导入时，建议使用“-e”和“-c”参数，即导入前删除已存在于待导入数据库中的数据库对象，同时当出现导入错误时，忽略当前错误，继续执行导入任务，并在导入后会显示相应的错误信息。

步骤1 以root用户登录到服务器，执行如下命令进入数据存放路径。

```
cd /opt/bin
```

步骤2 使用gs_restore命令，从postgres整个数据库内容的导出文件中，将数据库的所有对象的定义导入到backupdb。

```
gs_restore -W password -U jack /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d backupdb -s -e -c
```

表 2-27 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W *password*，此处密码需要用户自定义。
-d	连接数据库dbname，并直接将数据导入到该数据库中。	-d backupdb

版权所有 © 华为云计算技术有限公司

参数	参数说明	举例
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-h	“集群地址”如果通过公网地址连接，请指定为集群“公网访问地址”或“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问地址”或“内网访问域名”。	-h 10.10.10.100
-e	当发送SQL语句到数据库时如果出现错误，退出当前出现错误的任务，并执行其他导入任务。即默认状态下会忽略错误任务并继续执行导入，且在导入后会显示一系列错误信息。	-
-c	在重新创建数据库对象前，清理（删除）已存在于将要导入的数据库中的数据库对象。	-
-s	只导入模式定义，不导入数据。当前的序列值也不会被导入。	-

其他参数说明请参见《数据仓库服务工具参考》中“服务端工具>gs_restore”章节。

----结束

示例

示例一：执行gs_restore，导入指定MPPDB_backup.dmp文件（自定义归档格式）中postgres数据库的数据和对象定义。

```
gs_restore -W password backup/MPPDB_backup.dmp -p 8000 -h 10.10.10.100 -d backupdb
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore: total time: 13053  ms
```

示例二：执行gs_restore，导入指定MPPDB_backup.tar文件（tar归档格式）中postgres数据库的数据和对象定义。

```
gs_restore backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d backupdb
gs_restore[2017-07-21 19:21:32]: restore operation successful
gs_restore[2017-07-21 19:21:32]: total time: 21203  ms
```

示例三：执行gs_restore，导入指定MPPDB_backup目录文件（目录归档格式）中postgres数据库的数据和对象定义。

```
gs_restore backup/MPPDB_backup -p 8000 -h 10.10.10.100 -d backupdb
gs_restore[2017-07-21 19:26:46]: restore operation successful
gs_restore[2017-07-21 19:26:46]: total time: 21003  ms
```

示例四：执行gs_restore，将postgres数据库的所有对象的定义导入至backupdb数据库。导入前，postgres存在完整的定义和数据，导入后，backupdb数据库只存在所有对象定义，表没有数据。

版权所有 © 华为云计算技术有限公司

```
gs_restore -W password /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d backupdb -s -e -c
gs_restore[2017-07-21 19:46:27]: restore operation successful
gs_restore[2017-07-21 19:46:27]: total time: 32993  ms
```

示例五：执行gs_restore，导入MPPDB_backup.dmp文件中PUBLIC模式的所有定义和数据。在导入时会先删除已经存在的对象，如果原对象存在跨模式的依赖则需手工强制干预。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -h 10.10.10.100 -d backupdb -e -c -n PUBLIC
gs_restore: [archiver (db)] Error while PROCESSING TOC:
gs_restore: [archiver (db)] Error from TOC entry 313; 1259 337399 TABLE table1 gaussdba
gs_restore: [archiver (db)] could not execute query: ERROR:  cannot drop table table1 because other objects depend on it
DETAIL:  view t1.v1 depends on table table1
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
Command was: DROP TABLE public.table1;
```

手工删除依赖，导入完成后再重新创建。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -h 10.10.10.100 -d backupdb -e -c -n PUBLIC
gs_restore[2017-07-21 19:52:26]: restore operation successful
gs_restore[2017-07-21 19:52:26]: total time: 2203  ms
```

示例六：执行gs_restore，导入MPPDB_backup.dmp文件中PUBLIC模式下表hr.staffs的定义。在导入之前，hr.staffs表不存在。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -h 10.10.10.100 -d backupdb -e -c -s -n PUBLIC -t hr.staffs
gs_restore[2017-07-21 19:56:29]: restore operation successful
gs_restore[2017-07-21 19:56:29]: total time: 21000  ms
```

示例七：执行gs_restore，导入MPPDB_backup.dmp文件中PUBLIC模式下表hr.staffs的数据。在导入之前，hr.staffs表不存在数据。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -h 10.10.10.100 -d backupdb -e -a -n PUBLIC -t hr.staffs
gs_restore[2017-07-21 20:12:32]: restore operation successful
gs_restore[2017-07-21 20:12:32]: total time: 20203  ms
```

示例八：执行gs_restore，导入指定表hr.staffs的定义。在导入之前，hr.staffs表的数据是存在的。

```
human_resource=# select * from hr.staffs;
 staff_id | first_name  |  last_name  |  email   |    phone_number    |      hire_date      | employment_id |
salary  | commission_pct | manager_id | section_id
----------+-------------+-------------+----------+--------------------+---------------------+---------------+----------
+----------------+------------+------------
      200 | Jennifer    | Whalen      | JWHALEN  | 515.123.4444       | 1987-09-17 00:00:00 | AD_ASST       |
4400.00 |                |        101 |         10
      201 | Michael     | Hartstein   | MHARTSTE | 515.123.5555       | 1996-02-17 00:00:00 | MK_MAN        |
13000.00 |                |        100 |         20

gsql -d human_resource -p 8000
gsql ((GaussDB x.x.x build 39137c2d) compiled at 2022-04-01 15:43:11 commit 3629 last mr 5138 release)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

human_resource=# drop table hr.staffs CASCADE;
NOTICE:  drop cascades to view hr.staff_details_view

gs_restore -W password /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100-d human_resource -n hr -t staffs -s -e
restore operation successful
total time: 904  ms

human_resource=# select * from hr.staffs;
 staff_id | first_name | last_name | email | phone_number | hire_date | employment_id | salary |
commission_pct | manager_id | section_id
----------+------------+-----------+-------+--------------+-----------+---------------+--------+----------------
+------------+------------
(0 rows)
```


版权所有 © 华为云计算技术有限公司

示例九：执行gs_restore，导入staffs和areas两个指定表的定义和数据。在导入之前，staffs和areas表不存在。

```
human_resource=# \d
                        List of relations
 Schema |        Name        | Type  |  Owner   |             Storage
--------+--------------------+-------+----------+---------------------------------
 hr     | employment_history | table |  | {orientation=row,compression=no}
 hr     | employments        | table |  | {orientation=row,compression=no}
 hr     | places             | table |  | {orientation=row,compression=no}
 hr     | sections           | table |  | {orientation=row,compression=no}
 hr     | states             | table |  | {orientation=row,compression=no}
(5 rows)

gs_restore -W password /home/mppdb/backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d
human_resource -n hr -t staffs -n hr -t areas
restore operation successful
total time: 724  ms

human_resource=# \d
                        List of relations
 Schema |        Name        | Type  |  Owner   |             Storage
--------+--------------------+-------+----------+---------------------------------
 hr     | areas              | table |  | {orientation=row,compression=no}
 hr     | employment_history | table |  | {orientation=row,compression=no}
 hr     | employments        | table |  | {orientation=row,compression=no}
 hr     | places             | table |  | {orientation=row,compression=no}
 hr     | sections           | table |  | {orientation=row,compression=no}
 hr     | staffs             | table |  | {orientation=row,compression=no}
 hr     | states             | table |  | {orientation=row,compression=no}
(7 rows)

human_resource=# select * from hr.areas;
 area_id |       area_name
---------+------------------------
       4 | Iron
       1 | Wood
       2 | Lake
       3 | Desert
(4 rows)
```

示例十：执行gs_restore，导入hr的模式，包含模式下的所有对象定义和数据。

```
gs_restore -W password /home//backup/MPPDB_backup1.sql -p 8000 -h 10.10.10.100 -d backupdb -n hr -e -
c
restore operation successful
total time: 702  ms
```

示例十一：执行gs_restore，同时导入hr和hr1两个模式，仅导入模式下的所有对象定义。

```
gs_restore -W password /home//backup/MPPDB_backup2.dmp -p 8000 -h 10.10.10.100 -d backupdb -n hr -
n hr1 -s
restore operation successful
total time: 665  ms
```

示例十二：执行gs_restore，将human_resource数据库导出文件进行解密并导入至backupdb数据库中。

```
create database backupdb;


gs_restore /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d backupdb --with-
key=1234567812345678
restore operation successful
total time: 23472  ms

gsql -d backupdb -p 8000 -r
gsql ((GaussDB x.x.x build 39137c2d) compiled at 2022-04-01 15:43:11 commit 3629 last mr 5138 release)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
```


版权所有 © 华为云计算技术有限公司

```
backupdb=# select * from hr.areas;
 area_id |       area_name
---------+------------------------
       4 | Iron
       1 | Wood
       2 | Lake
       3 | Desert
(4 rows)
```

示例十三：用户user1不具备将导出文件中数据导入至数据库backupdb的权限，而角色role1具备该权限，要实现将文件数据导入数据库backupdb，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。

```
human_resource=# CREATE USER user1 IDENTIFIED BY 'password';

gs_restore -U user1 -W password /home//backup/MPPDB_backup.tar -p 8000 -h 10.10.10.100 -d backupdb
--role role1 --rolepassword password
restore operation successful
total time: 554  ms

gsql -d backupdb -p 8000 -r
gsql ((GaussDB x.x.x build 39137c2d) compiled at 2022-04-01 15:43:11 commit 3629 last mr 5138 release)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

backupdb=# select * from hr.areas;
 area_id |       area_name
---------+------------------------
       4 | Iron
       1 | Wood
       2 | Lake
       3 | Desert
(4 rows)
```

2.6 导出数据

2.6.1 导出数据到 OBS

2.6.1.1 关于 OBS 并行导出

概述

DWS数据库支持通过OBS外表并行导出数据：通过OBS外表设置的导出模式、导出数据格式等信息来指定导出的数据文件，利用多DN并行的方式，将数据从DWS数据库导出到外部，存放在OBS对象存储服务器上，从而提高整体导出性能。

- CN只负责任务的规划及下发，数据导出工作由DN负责，并释放CN资源，使其有能力处理外部请求。
- 每个DN都参与数据导出，使各个设备的计算能力及网络带宽得到充分利用。
- 支持多个OBS服务并发导出，导出的桶和对象的路径必须不同并且不能为空。
- 选择OBS服务器与集群节点处于联网状态，导出速率会受网络带宽影响。
- 支持数据文件格式：TEXT、CSV、ORC、PARQUET。单行数据大小需<1GB。
- 在执行OBS导入导出时，为了确保数据导入或导出的正确性，需要在相同的兼容模式下操作。

 版权所有 © 华为云计算技术有限公司

例如：在mysql兼容模式下导入（导出）的数据，同样需要在mysql兼容模式下才能正确导出（导入）。

目前，导出数据有两种方式：

- 方式一：无需用户创建server，使用默认server创建外表，支持TXT、CSV格式的数据，参见**导出CSV、TXT数据到OBS**。
- 方式二：需用户创建server，使用该server创建外表，支持ORC、PARQUET格式的数据，参见**导出ORC和PARQUET数据到OBS**。其中PARQUET格式的数据仅9.1.0及以上版本支持。

须知

- OBS导入导出数据时，不支持中文路径。
- OBS导入导出数据时，暂不支持跨Region进行OBS数据导入导出，必须确保OBS和DWS集群在同一个Region中。
- 在执行OBS导入导出时，为了确保数据导入或导出的正确性，需要在相同的兼容模式下操作。

  例如：在MySQL兼容模式下导入（导出）的数据，同样需要在MySQL兼容模式下才能正确导出（导入）。

相关概念

- **数据源文件**：存储有数据的TEXT、CSV、ORC、PARQUET文件。
- **OBS**：对象存储服务，是一种可存储文档、图片、影音视频等非结构化数据的云存储服务。从DWS并行导出数据时，数据对象放置在OBS服务器上。
- **桶（Bucket）**：对OBS中的一个存储空间的形象称呼，是存储对象的容器。
  - 对象存储是一种非常扁平化的存储方式，桶中存储的对象都在同一个逻辑层级，去除了文件系统中的多层级树形目录结构。
  - 在OBS中，桶名必须是全局唯一的且不能修改，即用户创建的桶不能与自己已创建的其他桶名称相同，也不能与其他用户创建的桶名称相同。每个桶在创建时都会生成默认的桶ACL（Access Control List），桶ACL列表的每项包含了对被授权用户授予什么样的权限，如读取权限、写入权限、完全控制权限等。用户只有对桶有相应的权限，才可以对桶进行操作，如创建、删除、显示、设置桶ACL等。
  - 一个用户最多可创建100个桶，但每个桶中存放的总数据容量和对象/文件数量没有限制。
- **对象**：是存储在OBS中的基本数据单位。用户上传的数据以对象的形式存储在OBS的桶中。对象的属性包括名称Key，Metadata，Data。

  通常，将对象等同于文件来进行管理，但是由于OBS是一种对象存储服务，并没有文件系统中的文件和文件夹概念。为了使用户更方便进行管理数据，OBS提供了一种方式模拟文件夹。通过在对象的名称中增加“/”，如tpcds1000/stock.csv，tpcds1000可以等同于文件夹，stock.csv就可以等同于文件名，而对象名称（key）仍然是tpcds1000/stock.csv、对象的内容就是stock.csv数据文件的内容。
- **Key**：对象的名称（键），为经过UTF-8编码的长度大于0且不超过1024的字符序列，一个桶里的每个对象必须拥有唯一的对象键值。用户可使用桶名+对象名来存储和获取对应的对象。

版权所有 © 华为云计算技术有限公司

- **Metadata**：对象元数据，用来描述对象的信息。元数据又可分为系统元数据和用户元数据。这些元数据以键值对（Key-value）的形式随http头域一起上传到OBS系统。
  - 系统元数据由OBS系统产生，在处理对象数据时使用。系统元数据包括：Date、Content-length、last-modify、Content-MD5等。
  - 用户元数据由用户上传对象时指定，是用户自己对对象的一些描述信息。
- **Data**：对象的数据内容，OBS对于数据的内容是无感知的，即认为对象内的数据为无状态的二进制数据。
- **外表**：用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。

相关原理

下面分别从以下两类表介绍从集群导出数据到OBS的原理。

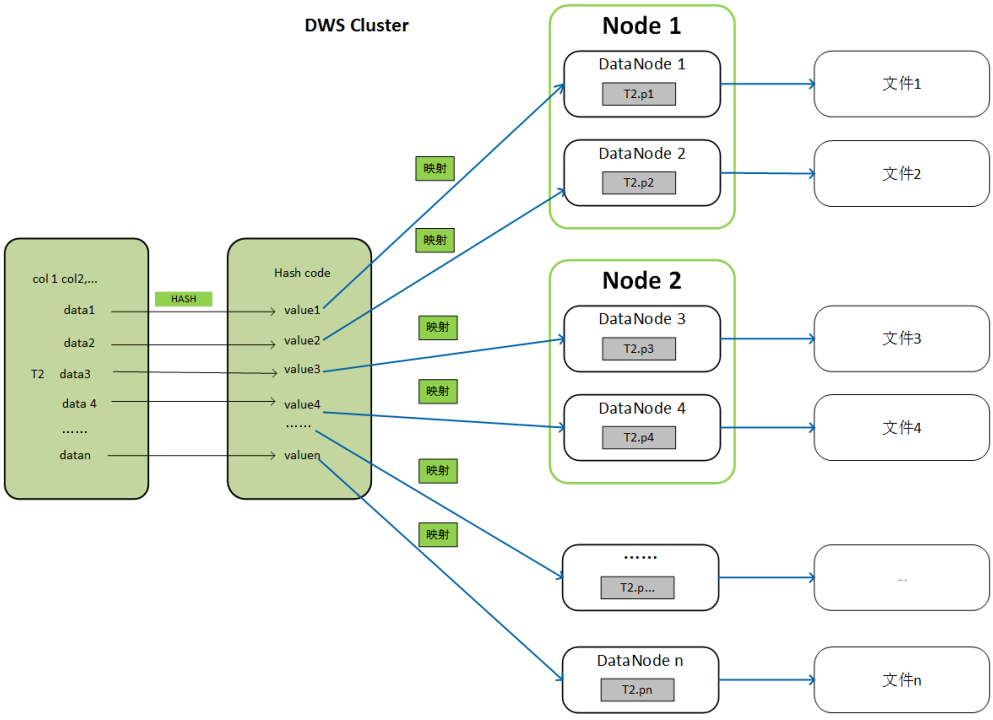
- Hash分布表：在建表语句中指定了DISTRIBUTE BY HASH (Column_Name)的表。

  对于Hash分布表而言，在存储表数据时，采用的是散列（Hash）方式的存储原理，如图2-7所示，图中以将表（T2）导出到OBS为例。

  在存储表数据时，将表（T2）中指定的Hash字段（col2）进行Hash运算后，生成相应的Hash值（value），根据DN与Hash值的映射关系，将该元组分发到相应的DN上进行存储。

  在导出数据到OBS时，每一个存储了导出表的（T2）数据的DN会直接向OBS导出属于自己的数据文件。多个节点将并行导出原始数据。

  图 2-7 Hash 分布原理



- Replication表：在建表语句中指定了DISTRIBUTE BY REPLICATION的表。

  Replication表在DWS集群的每个节点上都会存储一份完整的表数据。因此，在导出数据到OBS时，DWS只会随机选择一个DN节点向OBS导出数据。

版权所有 © 华为云计算技术有限公司

导出文件的命名规则

DWS向OBS导出数据的文件命名规则如下：

- 从DN节点导出数据时，以segment的格式存储在OBS服务中，文件命名规则为“**表名称_节点名称_segment.n**”。这里的“n”是从0开始按照自然数0、1、2、3递增。

  例如，表t1在datanode3里面的数据导出成文件“t1_datanode3_segment.0”、“t1_datanode3_segment.1”等等，以此类推。

  对于来自不同集群或不同数据库的数据，建议用户可以将数据导出到不同的OBS桶或者同一个OBS桶的不同路径下。

- 每个segment可以存储的最大数据为1GB，并且不能切断元组。如果segment超过1GB，超过1GB的数据会作为第二个segment进行存储。

  例如：

  datanode3节点将表（t1）导出到OBS时，一个segment里面已经存储了100条元组，大小是1023MB，如果再插入一条5MB的元组，大小就变成1028MB了，此时会以1023MB生成一个“t1_datanode3_segment.0”保存到OBS服务中，新插入的第101条元组作为下一个“t1_datanode3_segment.1”保存到OBS服务中。

- 导出Hash分布表时，每个DataNode节点生成的segment数量和集群的DataNode节点数无关，而是取决于每个DataNode节点上存储的数据量。按照Hash方式存储在各个DataNode节点上的数据分布不一定均匀。

  例如，一个有6个DataNode节点的集群，DataNode1到DataNode6分别有1.5GB、0.7GB、0.6GB、0.8GB、0.4GB、0.5GB的数据，则导出时会生成7个OBS segment文件，其中DataNode1会生成1GB和0.5GB两个segment文件。

版权所有 © 华为云计算技术有限公司

导出流程

图 2-8 并行导出流程

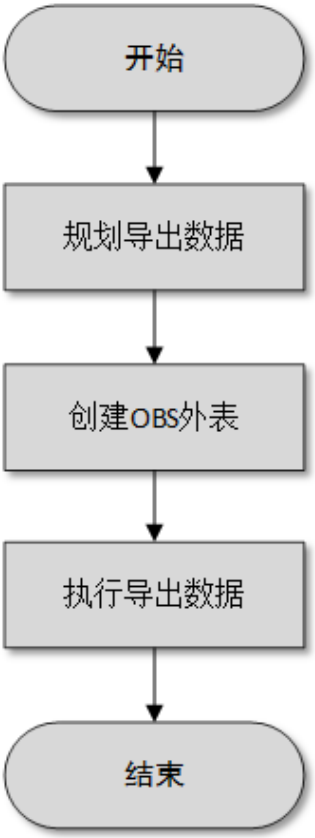


表 2-28 流程说明

流程	说明	子任务
规划导出数据	创建OBS桶，并在桶中创建导出后的数据文件的存放目录。 详细请参见**规划导出数据**。	-
创建OBS外表。	创建外表用于帮助OBS指定的待导出数据文件。外表中保存了数据源文件导出后的位置、文件格式、编码格式、数据间的分隔符等信息。 详细内容请参见**创建OBS外表**。	-
执行导出数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导出到数据文件中。 详细内容请参见**执行导出**。	-

版权所有 © 华为云计算技术有限公司

2.6.1.2 导出 CSV、TXT 数据到 OBS

2.6.1.2.1 规划导出数据

操作场景

在OBS上规划导出数据存放的位置。

规划 OBS 存储位置和文件

导出数据需要指定数据在OBS中的存储路径（需指定到目录），导出的数据可以按CSV解析格式保存到文件中。系统还支持TEXT类型的解析格式，将数据导出保存便于导入不同的应用程序。

导出路径的目标目录中不能存在任何文件。

规划 OBS 桶权限

在导出数据时，执行导出操作的用户需要具备以下条件：

- 已开通OBS服务。
- 具备数据导出路径所在的OBS桶的写入权限。

  通过配置桶的ACL权限，可以将写入权限授予指定的用户账号。

  具体操作请参见**根据规划准备OBS存储位置和OBS桶的写权限**。

规划导出数据和外表

提前在数据库的表中准备好待导出的数据，且单行数据大小需要小于1GB。根据导出数据，规划匹配用户数据的外表，外表的字段、字段类型以及长度等属性需要能够对应用户数据。

根据规划准备 OBS 存储位置和 OBS 桶的写权限

步骤1 创建OBS桶，并在OBS桶中新建文件夹作为导出数据的存放目录。

1. 登录OBS控制台。

   单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。
2. 创建桶。

   如何创建OBS桶，具体请参见《对象存储服务用户指南》中的“控制台指南 > 管理桶 > 创建桶”章节。

   例如，创建以下两个桶：“mybucket”和“mybucket02”。
3. 新建文件夹。

   具体请参见《对象存储服务用户指南》中的“控制台指南 > 管理对象 > 新建文件夹”章节。

   例如：

   - 在已创建的OBS桶“mybucket”中新建一个文件夹“output_data”。
   - 在已创建的OBS桶“mybucket02”中新建一个文件夹“output_data”。

步骤2 获取新建文件夹的OBS路径。

版权所有 © 华为云计算技术有限公司

在创建外表时需要指定导出数据文件的OBS存放目录的路径，用于创建外表时location参数设置。

location参数中OBS文件夹的路径由“obs://”、桶名和文件路径组成，即为：obs://<bucket_name>/<file_path>/

例如，在本例中，location参数中OBS文件夹路径为：

```
obs://mybucket/output_data/
```

说明

执行数据导出的时候，导出数据文件的OBS存放目录的路径必须为空。

步骤3 为导出用户设置OBS桶的写权限。

在导出数据时，执行导出操作的用户需要具备数据导出路径所在的OBS桶的写入权限。通过配置桶的ACL权限，可以将写入权限授予指定的用户账号。

具体请参见《对象存储服务用户指南》中的“控制台指南 > 权限控制 > 配置桶ACL”章节。

----结束

2.6.1.2.2 创建 OBS 外表

操作步骤

步骤1 根据**规划导出数据**中规划的路径，由此确定创建外表时使用的参数**location**的值。

步骤2 用户获取OBS访问协议对应的AK值和SK值。

获取访问密钥，请登录DWS控制台，单击右上角的用户名并选择菜单“我的凭证”，然后在左侧导航树单击“管理访问密钥”。在访问密钥页面，可以查看已有的访问密钥ID（即AK），如果要同时获取AK和SK，可以单击“新增访问密钥”创建并下载访问密钥。

步骤3 梳理待导出数据的格式信息，确定创建外表时使用的数据格式参数的值。详细使用请参见数据格式参数。

步骤4 根据前面步骤确定的参数，**创建OBS外表**。

----结束

示例一

例如，在DWS数据库中，创建一个format参数为text的只写外表，用于导出text文件。设置的参数信息如下所示：

- **location**

  在**规划导出数据**中，通过**获取数据源文件的OBS路径**已经获取到数据源文件的OBS路径。

  因此，设置参数“location”为：

  ```
  location 'obs://mybucket/output_data/',
  ```

- **访问密钥（AK和SK）**
  - 用户获取OBS访问协议对应的AK值（access_key）。

版权所有 © 华为云计算技术有限公司

- 用户获取OBS访问协议对应的SK值（secret_access_key）。

  说明

  用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的内容。

- **设置数据格式参数**
  - **数据源文件格式（format）**为“TEXT”。
  - **编码格式（encoding）**为“UTF-8”。
  - **是否使用加密（encrypt）**，默认为“off”。
  - **字段分隔符（delimiter）**为“|”。

根据以上信息，创建的外表如下所示：

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
DROP FOREIGN TABLE IF EXISTS product_info_output_ext1;
CREATE FOREIGN TABLE product_info_output_ext1
(
 c_bigint bigint,
 c_char char(30),
 c_varchar varchar(30),
 c_nvarchar2 nvarchar2(30) ,
 c_data date,
 c_time time ,
 c_test varchar(30))
 server gsmpp_server
 options (
 LOCATION 'obs://mybucket/output_data/',
 ACCESS_KEY 'access_key_value_to_be_replaced',
 SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
 format 'text',
 delimiter '|',
 encoding 'utf-8',
 encrypt 'on'
  )
 WRITE ONLY;
```

返回如下信息表示创建成功：

```
CREATE FOREIGN TABLE
```

示例二

例如，在DWS数据库中，创建一个format参数为CSV的只写外表，用于导出CSV文件。设置的参数信息如下所示：

- **location**

  在**规划导出数据**中，通过**获取数据源文件的OBS路径**，已经获取到数据源文件的OBS路径。

  因此，设置参数“location”为：

  ```
  location 'obs://mybucket/output_data/',
  ```

- **访问密钥（AK和SK）**

版权所有 © 华为云计算技术有限公司

  - 用户获取OBS访问协议对应的AK值（access_key）。
  - 用户获取OBS访问协议对应的SK值 （secret_access_key）。

  说明

  用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的对应内容。

- **设置数据格式参数**
  - **数据源文件格式（format）**为“CSV”。
  - **编码格式（encoding）**为“UTF-8”。
  - **是否使用加密（encrypt），**默认为 “off”。
  - **字段分隔符（delimiter）**为“,”。
  - **header（指定导出数据文件是否包含标题行）**

    指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。

    OBS导出数据时不支持该参数为true，使用缺省值false，不需要设置，表示导出的数据文件第一行不是标题行（即表头）。

根据以上信息，创建的外表如下所示：

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
DROP FOREIGN TABLE IF EXISTS product_info_output_ext2;
CREATE FOREIGN TABLE product_info_output_ext2
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
)
SERVER gsmpp_server
OPTIONS(
location 'obs://mybucket/output_data/',
FORMAT 'CSV' ,
DELIMITER ',',
encoding 'utf8',
header 'false',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)
WRITE ONLY ;
```

返回如下信息表示创建成功：

```
CREATE FOREIGN TABLE
```

 版权所有 © 华为云计算技术有限公司

2.6.1.2.3 执行导出

导出操作语法

执行数据导出语法：

```
INSERT INTO  [foreign table 表名] SELECT * FROM [源表名];
```

执行导出数据示例

- **示例1**：将表product_info_output的数据通过外表product_info_output_ext导出到数据文件中。

  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output;
  ```

  若出现以下类似信息，说明数据导出成功。

  ```
  INSERT 0 10
  ```

- **示例2**：通过条件过滤（WHERE *product_price>500*），向数据文件中导出部分数据。

  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output WHERE product_price>500;
  ```

  说明

  - 在需要执行多次数据导出时，导出数据的存放路径必须为空，否则将导出失败。
  - 对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。

2.6.1.2.4 示例

单表导出操作步骤

通过创建外表，将数据库中的单表导出至OBS的两个桶中。

步骤1 用户通过管理控制台登录到OBS数据服务器。在OBS数据服务器上，分别创建数据文件存放的两个桶“/input-data1”“/input-data2”，并创建每个桶下面的data目录“/input-data1/data”“/input-data2/data”。

步骤2 在DWS数据库上，创建外表tpcds.customer_address_ext1和tpcds.customer_address_ext2用于OBS数据服务器接收数据库导出数据。

OBS与集群处于同一区域，需要导出的表为DWS示例表tpcds.customer_address。

其中设置的**导出信息**如下所示：

- 由于OBS数据服务器上的数据源文件存放目录为“/input-data1/data/ ”和/input-data2/data/ ，所以设置tpcds.customer_address_ext1参数“location”为“obs://input-data1/data/ ”，设置tpcds.customer_address_ext2参数“location”为“ obs://input-data2/data/”。

设置的**数据格式信息**是根据表从数据库导出时需要的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（format）为“CSV”。
- 编码格式（encoding）为“UTF-8”。
- 字段分隔符（delimiter）为“E'\x08'”。
- 是否使用加密（encrypt），默认为“off”。
- （必选）用户获取OBS访问协议对应的AK值（access_key ）。

版权所有 © 华为云计算技术有限公司

- （必选）用户获取OBS访问协议对应的SK值（secret_access_key）。

  说明

  用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的内容。

根据以上信息，创建的外表如下所示：

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE FOREIGN TABLE tpcds.customer_address_ext1
(
ca_address_sk             integer                       ,
ca_address_id             char(16)                      ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                   ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                       ,
ca_zip                    char(10)                      ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)                  ,
ca_location_type          char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data1/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)Write Only;
CREATE FOREIGN TABLE tpcds.customer_address_ext2
(
ca_address_sk             integer                       ,
ca_address_id             char(16)                      ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                   ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                       ,
ca_zip                    char(10)                      ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)                  ,
ca_location_type          char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data2/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)Write Only;
```


版权所有 © 华为云计算技术有限公司

步骤3 在DWS数据库上，将数据表tpcds.customer_address并发导出到外表tpcds.customer_address_ext1和tpcds.customer_address_ext2中。

```
INSERT INTO tpcds.customer_address_ext1 SELECT * FROM tpcds.customer_address;
INSERT INTO tpcds.customer_address_ext2 SELECT * FROM tpcds.customer_address;
```

说明

OBS外表在设计上禁止往非空的路径下导出文件，但是在并发场景下会出现同一路径导出文件的情况，此时会发生异常。

异常场景：假如用户使用同一张表的数据并发导出到同一个OBS的外表，在一条SQL语句执行在OBS服务器上没有生成文件时，另一条SQL语句也执行导出，最终执行结果为两条SQL语句均执行成功，产生数据覆盖现象，建议用户在执行OBS外表导出任务时，不要往同一OBS外表并发导出。

----结束

多表并发导出操作步骤

通过创建的两个外表，将数据库中的两个表分别导出至OBS的桶中。

步骤1 用户通过管理控制台登录到OBS数据服务器。在OBS数据服务器上，分别创建数据文件存放的两个桶“/input-data1”“/input-data2”，并创建每个桶下面的data目录“/input-data1/data”“/input-data2/data”。

步骤2 在DWS数据库上，创建外表tpcds.customer_address_ext1和tpcds.customer_address_ext2分别用于OBS服务器接收导出的数据。

规划OBS与集群处于同一区域，需要导出的表为已存在的表tpcds.customer_address和tpcds.customer_demographics。

其中设置的**导出信息**如下所示：

- 由于OBS服务器上的数据源文件存放目录为“/input-data1/data/ ”和/input-data2/data/ ，所以设置tpcds.customer_address_ext1参数“location”为“obs://input-data1/data/ ”，设置tpcds.customer_address_ext2参数“location”为“ obs://input-data2/data/”。

设置的**数据格式信息**是根据表从DWS中导出时需要的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（format）为“CSV”。
- 编码格式（encoding）为“UTF-8”。
- 字段分隔符（delimiter）为“E'\x08'”。
- 是否使用加密（encrypt），默认为“off”。
- （必选）用户获取OBS访问协议对应的AK值（access_key ）。
- （必选）用户获取OBS访问协议对应的SK值（secret_access_key）。

说明

用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的内容。

根据以上信息，创建的外表如下所示：

版权所有 © 华为云计算技术有限公司

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE FOREIGN TABLE tpcds.customer_address_ext1
(
ca_address_sk             integer               ,
ca_address_id             char(16)              ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                     ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                  ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                      ,
ca_zip                    char(10)                     ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)                  ,
ca_location_type          char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data1/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)Write Only;
CREATE FOREIGN TABLE tpcds.customer_address_ext2
(
ca_address_sk             integer               ,
ca_address_id             char(16)              ,
ca_address_name           varchar(20)           ,
ca_address_code           integer               ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                     ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                  ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                      ,
ca_zip                    char(10)                     ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input_data2/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
QUOTE E'\x1b',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)Write Only;
```

步骤3 在DWS数据库上，将数据表tpcds.customer_address和tpcds.warehouse并发导出到外表tpcds.customer_address_ext1和tpcds.customer_address_ext2中。

```
INSERT INTO tpcds.customer_address_ext1 SELECT * FROM tpcds.customer_address;
INSERT INTO tpcds.customer_address_ext2 SELECT * FROM tpcds.warehouse;
```

----结束

 版权所有 © 华为云计算技术有限公司

2.6.1.3 导出 ORC 和 PARQUET 数据到 OBS

2.6.1.3.1 规划导出数据

OBS导出数据准备：请参见**规划导出数据**完成OBS导出数据准备。

OBS导出支持的数据类型请参见**表2-29**。

表 2-29 ORC、PARQUET 格式的只写外表与 HIVE 数据类型匹配关系

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
1字节整数	TINYINT	TINYINT（不推荐）	SMALLINT
		SMALLINT（推荐）	SMALLINT
2字节整数	SMALLINT	SMALLINT	SMALLINT
4字节整数	INTEGER、BINARY_INTEGER	INTEGER	INT
8字节整数	BIGINT	BIGINT	BIGINT
单精度浮点数	FLOAT4 、REAL	FLOAT4、REAL	FLOAT
双精度浮点型	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE
科学数据类型	DECIMAL、NUMERIC	DECIMAL[p (,s)] 最大支持38位精度	precision <=38时，DECIMAL, precision > 38时， STRING
日期类型	DATE	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP
时间类型	TIME [(p)] [WITHOUT TIME ZONE]、TIME [(p)] [WITH TIME ZONE]	TEXT	STRING

版权所有 © 华为云计算技术有限公司

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
	TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)][WITH TIME ZONE]、SMALLDATETIME	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP
	INTERVAL DAY (l) TO SECOND (p)、INTERVAL [FIELDS] [(p)]	VARCHAR(n)	VARCHAR(n)
BOOLEAN类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)、CHARACTER(n)、NCHAR(n)	CHAR(n)、CHARACTER(n)、NCHAR(n)	n<=255时，CHAR(n)，n>255 时，STRING
VarChar类型	VARCHAR(n)、CHARACTER VARYING(n)、VARCHAR2(n)、	VARCHAR(n)	n<=65535时，VARCHAR(n)，n>65535时，STRING
	NVARCHAR2(n)	TEXT	STRING
字符串（文本大对象）	TEXT、CLOB	TEXT、CLOB	STRING
二进制类型	BYTEA	BYTEA	BINARY
货币类型	MONEY	NUMERIC	BIGINT

HDFS导出数据准备：HDFS导出准备即配置MRS，具体信息可参考《MapReduce服务用户指南》。

说明

本小节以导出ORC格式为例，PARQUET数据的导出方法与ORC格式相似。9.1.0及以上集群版本支持导出PARQUET数据。

2.6.1.3.2 创建外部服务器

本章节包含创建OBS外部服务器和创建HDFS外部服务器。

创建OBS外部服务器，用于定义OBS服务器的信息，供外表调用，针对集群不同版本，创建方式不同。具体操作步骤请参见**创建外部服务器**。

版权所有 © 华为云计算技术有限公司

- 8.2.0及以上集群版本：通过管理控制台创建OBS数据源实现以委托方式访问OBS桶数据。
- 8.2.0以下集群版本：通过手动创建OBS服务器实现。创建外部服务器的详细语法，请参见《数据仓库服务SQL语法参考》中“CREATE SERVER”章节。

创建HDFS外部服务器的具体操作步骤请参见**手动创建外部服务器**。

2.6.1.3.3 创建外表

当完成**创建外部服务器**后，在DWS数据库中创建一个OBS/HDFS只写外表，用来访问存储在OBS/HDFS上的数据。此外表是只写的，只能用于导出操作。

创建外表的语法格式如下，详细的描述请参见《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (SQL on Hadoop or OBS)”章节。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER dfs_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   [ {WRITE ONLY }]
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为"*product_info_ext_obs*"的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  外表的表名。
- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。
- **SERVER dfs_server**

  外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器连接OBS/HDFS读取数据。

  此处应填写为参照**创建外部服务器**创建的外部服务器名称。
- **OPTIONS参数**

  用于指定外表数据的各类参数，关键参数如下所示。
  - “format”：表示导出的数据文件格式，支持“orc”和“parquet”格式。
  - “foldername”：必选参数。外表中数据源文件目录。OBS：数据源文件的OBS路径，此处仅需要填写“*/桶名/文件夹目录层级/*”。HDFS：HDFS文件系统上的路径。此选项对WRITE ONLY外表为必选项。
  - **“encoding”**：外表中数据源文件的编码格式名称，缺省为utf8。
  - **“filesize”**

    指定WRITE ONLY外表的文件大小，单位为MB。此选项为可选项，不指定该选项默认分布式文件系统配置中文件大小的配置值。此语法仅对WRITE ONLY的外表有效。

    取值范围：[1, 1024]的整数。

版权所有 © 华为云计算技术有限公司

> **说明**
>
> filesize参数只对ORC格式的WRITE ONLY的HDFS外表有效。

- **“compression”**

  指定文件的压缩方式，此选项为可选项。此语法仅对WRITE ONLY的外表有效。

  取值范围：zlib，snappy，lz4。

  指定ORC格式文件的压缩方式，缺省值为zlib。指定PARQUET格式文件的压缩方式，缺省值为snappy。

- **“version”**

  指定ORC格式的版本号，此选项为可选项。此语法仅对WRITE ONLY的外表有效。

  取值范围：目前仅支持0.12。缺省值为0.12。

- **“dataencoding”**

  在数据库编码与数据表的数据编码不一致时，该参数用于指定导出数据表的数据编码。比如数据库编码为Latin-1，而导出的数据表中的数据为UTF-8编码。此选项为可选项，如果不指定该选项，默认采用数据库编码。此语法仅对HDFS的WRITE ONLY外表有效。

  取值范围：该数据库编码支持转换的数据编码。

  > **说明**
  >
  > dataencoding参数只对ORC格式的WRITE ONLY的HDFS外表有效。

- **语法中的其他参数**

  其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中不需要设置。

  根据以上信息，创建外表命令如下所示：

```
DROP FOREIGN TABLE IF EXISTS product_info_ext_obs;

---建立不包含分区列的OBS外表，表关联的外部服务器为obs_server，表对应的OBS服务上的文件格式为‘orc’，OBS上的数据存储路径为'/mybucket/data/'。

CREATE FOREIGN TABLE product_info_ext_obs
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER obs_server
OPTIONS (
format 'orc',
foldername '/mybucket/demo.db/product_info_orc/',
   compression 'snappy',
    version '0.12'
) Write Only;
```


版权所有 © 华为云计算技术有限公司

2.6.1.3.4 执行导出

导出操作语法

执行数据导出语法：

```
INSERT INTO  [foreign table 表名] SELECT * FROM [源表名];
```

执行导出数据示例

- **示例1**：将表product_info_output的数据通过外表product_info_output_ext导出到数据文件中。

  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output;
  ```

  若出现以下类似信息，说明数据导出成功。

  ```
  INSERT 0 10
  ```

- **示例2**：通过条件过滤（WHERE *product_price>500*），向数据文件中导出部分数据。

  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output WHERE product_price>500;
  ```

  说明

  对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。

2.6.2 导出 ORC 和 PARQUET 数据到 MRS

2.6.2.1 导出 ORC 和 PARQUET 数据概述

DWS数据库支持通过HDFS外表导出ORC和PARQUET格式数据至MRS，通过外表设置的导出模式、导出数据格式等信息来指定导出的数据文件，利用多DN并行的方式，将数据从DWS数据库导出到外部，存放在HDFS文件系统上，从而提高整体导出性能。

- CN只负责任务的规划及下发，数据导出工作由DN负责，并释放CN资源，使其有能力处理外部请求。
- 每个DN都参与数据导出，使各个设备的计算能力及网络带宽得到充分利用。
- 支持多个hdfs server并发导出，导出的路径可以为空，命名规则需与导出文件一致。
- 选择MRS服务与集群节点处于联网状态，导出速率会受网络带宽影响。
- 支持数据文件格式：ORC、PARQUET。

说明

本小节以导出ORC格式为例，PARQUET数据的导出方法与ORC格式相似。9.1.0及以上集群版本支持导出PARQUET数据。

导出文件命名规则

DWS导出ORC和PARQUET数据的文件命名规则如下：

1. 导出至MRS（HDFS）：从DN节点导出数据时，以segment的格式存储在HDFS中，文件命名规则为“**mpp_数据库名_模式名_表名称_节点名称_n_UUID.数据格式名**”。这里的“n”是从0开始按照自然数0、1、2、3递增。这里的“UUID”是标准型式的UUID，由32个16进制字符组成，以连字号分为五段，形式为8-4-4-4-12共32个字符。

版权所有 © 华为云计算技术有限公司

2. 对于来自不同集群或不同数据库的数据，建议用户可以将数据导出到不同路径下。ORC和PARQUET格式单文件大小最大为256M左右（此处为软约束，实际业务可能会超出一点）。
3. 导出完成后会生成_SUCCESS标记文件。

2.6.2.2 规划导出数据

MRS导出支持的数据类型请参见**表2-30**。

表 2-30 ORC、PARQUET 格式的只写外表与 HIVE 数据类型匹配关系

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
1字节整数	TINYINT	TINYINT（不推荐）	SMALLINT
		SMALLINT（推荐）	SMALLINT
2字节整数	SMALLINT	SMALLINT	SMALLINT
4字节整数	INTEGER、BINARY_INTEGER	INTEGER	INT
8字节整数	BIGINT	BIGINT	BIGINT
单精度浮点数	FLOAT4 、REAL	FLOAT4、REAL	FLOAT
双精度浮点型	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE PRECISION、FLOAT8、BINARY_DOUBLE	DOUBLE
科学数据类型	DECIMAL、NUMERIC	DECIMAL[p (,s)] 最大支持38位精度	precision <=38时，DECIMAL, precision > 38时， STRING
日期类型	DATE	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP
时间类型	TIME [(p)] [WITHOUT TIME ZONE]、TIME [(p)] [WITH TIME ZONE]	TEXT	STRING

版权所有 © 华为云计算技术有限公司

类型名称	DWS内表支持类型（数据源表）	DWS只写外表对应的类型	HIVE建表类型
	TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)][WITH TIME ZONE]、SMALLDATETIME	TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP
	INTERVAL DAY (l) TO SECOND (p)、INTERVAL [FIELDS] [(p)]	VARCHAR(n)	VARCHAR(n)
BOOLEAN类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)、CHARACTER(n)、NCHAR(n)	CHAR(n)、CHARACTER(n)、NCHAR(n)	n<=255时，CHAR(n)，n>255 时，STRING
VarChar类型	VARCHAR(n)、CHARACTER VARYING(n)、VARCHAR2(n)、	VARCHAR(n)	n<=65535时，VARCHAR(n)，n>65535时，STRING
	NVARCHAR2(n)	TEXT	STRING
字符串（文本大对象）	TEXT、CLOB	TEXT、CLOB	STRING
二进制类型	BYTEA	BYTEA	BINARY
货币类型	MONEY	NUMERIC	BIGINT

HDFS导出数据准备：HDFS导出准备即配置MRS，具体信息可参考《 MapReduce服务用户指南 》。

2.6.2.3 创建外部服务器

HDFS创建外部服务器，需要先创建用户和数据库并授予外表权限。

创建用户和数据库并授予外表权限

以下示例，是新建一个普通用户dbuser并创建一个数据库mydatabase，然后使用管理员用户授予dbuser外表权限。

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库gaussdb。

版权所有 © 华为云计算技术有限公司

例如，使用gsql客户端的用户通过如下语句连接数据库：

```
gsql -d gaussdb -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
```

步骤2 新建一个普通用户，并用它创建一个数据库。

新建一个具有创建数据库权限的用户dbuser：

```
CREATE USER dbuser WITH CREATEDB PASSWORD 'password';
```

切换为新建的用户：

```
SET ROLE dbuser PASSWORD 'password';
```

执行以下命令创建数据库：

```
CREATE DATABASE mydatabase;
```

查询数据库：

```
SELECT * FROM pg_database;
```

返回结果中有**mydatabase**的信息表示创建成功：

```
datname   | datdba | encoding | datcollate | datctype | datistemplate | datallowconn | datconnlimit |
datlastsysoid | datfrozenxid | dattablespace | datcompatibility |                     datacl

------------+--------+----------+------------+----------+---------------+--------------+--------------+---------------
+--------------+---------------+------------------+-------------------------------------
--------------
 template1  |     10 |        0 | C          | C        | t             | t            |           -1 |        14146 |        1351
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 template0  |     10 |        0 | C          | C        | t             | f            |           -1 |        14146 |        1350
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 gaussdb    |     10 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1352 |
1663 | ORA             | {=Tc/Ruby,Ruby=CTc/Ruby,chaojun=C/Ruby,hu
obinru=C/Ruby}
 mydatabase |  17000 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1351
|          1663 | ORA             |
(4 rows)
```

步骤3 使用管理员用户给普通用户赋予创建外部服务器的权限和使用外表的权限。

使用数据库管理员用户通过数据库客户端连接新建的数据库。

例如，使用gsql客户端的用户可以直接使用如下语句切换为管理员用户去连接新建的数据库：

```
\c mydatabase dbadmin;
```

根据提示输入用户密码。

说明

注意，必须先使用管理员用户连接到**将要创建外部服务器和使用外表的数据库**，再对普通用户进行授权。

默认只有系统管理员才可以创建外部服务器，普通用户需要授权才可以创建，执行以下命令授权：

```
GRANT ALL ON FOREIGN DATA WRAPPER hdfs_fdw TO dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是hdfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令赋予用户使用外表的权限：

```
ALTER USER dbuser USEFT;
```

查看用户：

 版权所有 © 华为云计算技术有限公司

```
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvalidbegin, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolauditadmin
, r.rolsystemadmin
, r.roluseft
FROM pg_catalog.pg_roles r
ORDER BY 1;
```

返回结果中，dbuser的信息中包含了UseFT权限，表示授权成功：

```
 rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcanlogin | rolconnlimit | rolvalidbegin |
rolvaliduntil | memberof | rolreplication | rolauditadmin | rolsystemadmin | roluseft
-----------+----------+------------+---------------+-------------+-------------+--------------+---------------
+---------------+----------+----------------+---------------+----------------+----------
 dbuser    | f        | t          | f             | t           | t           |           -1 |               |               | {}       | f
| f             | f             | t
 lily      | f        | t          | f             | f           | t           |           -1 |               |               | {}       | f             |
f             | f             | f
 Ruby      | t        | t          | t             | t           | t           |           -1 |               |               | {}       | t
| t             | t             | t
```

----结束

手动创建外部服务器

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库postgres。

例如：通过gsql客户端登录数据库的用户可以使用以下两种方法中的一种进行连接：

可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令切换数据库和用户：
  ```
  \c postgres dbadmin;
  ```
  根据提示输入密码。
- 如果尚未登录gsql客户端，或者已经登录了gsql客户端执行**\q**退出gsql后，执行以下命令重新进行连接：
  ```
  gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -W password -r
  ```

步骤2 执行以下命令查询自动创建的外部服务器的信息：

```
SELECT * FROM pg_foreign_server;
```

返回结果如下：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                       srvoptions
------------------------------------------------+----------+--------+---------+------------+--------
+-----------------------------------------------------------------------------------------------------------------------------
 gsmpp_server                           |       10 |  13673 |         |            |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(2 rows)
```

查询结果中，每一行代表一个外部服务器的信息。与MRS数据源连接相关联的外部服务器包含以下信息：

- srvname值包含“hdfs_server”字样以及MRS集群的ID，此ID与MRS管理控制台的集群列表MRS ID相同。

版权所有 © 华为云计算技术有限公司

- srvoptions字段中的address参数为MRS集群的主备节点的IP地址及端口。

您可以根据上述信息找到您所要的外部服务器，并记录下它的srvname和srvoptions的值。

步骤3 切换为即将创建外部服务器的用户去连接其对应的数据库。

在本示例中，执行以下命令，使用**创建用户和数据库并授予外表权限**中创建的普通用户dbuser连接其创建的数据库mydatabase：

```
\c mydatabase dbuser;
```

步骤4 创建外部服务器。

创建外部服务器的详细语法，请参见《 数据仓库服务SQL语法参考 》中“CREATE SERVER”章节。示例如下：

```
CREATE SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692cahdfs_server FOREIGN DATA WRAPPER HDFS_FDW
OPTIONS
(
address '192.168.1.245:25000,192.168.1.218:25000',
hdfscfgpath '/MRS/8f79ada0-d998-4026-9020-80d6de2692ca',
type 'hdfs'
);
```

以下为必选参数的说明：

- 外部服务器名称

  允许用户自定义名字。

  在本例中，指定为前面的步骤**步骤2**中记录下来的srvname字段的值，如'*hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca*'。

  不同的数据库之间资源是隔离的，因此在不同的数据库中外部服务器名称可以相同。

- FOREIGN DATA WRAPPER

  只能指定为HDFS_FDW，它在数据库中已经存在。

- OPTIONS参数

  以下参数请分别指定为步骤**步骤2**中记录下来的srvoptions中的参数值。

  - address

    指定HDFS集群的主备节点所在的IP地址以及端口。

  - hdfscfgpath

    指定HDFS集群配置文件路径。该参数仅支持type为HDFS时设置。只能设置一个路径。

  - type

    取值为'hdfs'，表示HDFS_FDW连接的是HDFS。

步骤5 查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

返回结果如下所示，表示已经创建成功：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                srvoptions
------------------------------------------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------------------------
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
```


版权所有 © 华为云计算技术有限公司

```
d998-4026-9020-80d6de2692ca,type=hdfs}
(1 row)
```

----结束

2.6.2.4 创建外表

当完成**创建外部服务器**后，在DWS数据库中创建一个HDFS只写外表，用来访问存储在HDFS上的数据。此外表是只写的，只能用于导出操作。

创建外表的语法格式如下，详细的描述请参见《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (SQL on Hadoop or OBS)”章节。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER dfs_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   [ {WRITE ONLY }]
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为"*product_info_ext_obs*"的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  外表的表名。

- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。

- **SERVER dfs_server**

  外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器连接OBS/HDFS读取数据。

  此处应参考**创建外部服务器**中创建的外部服务器名称填写。

- **OPTIONS参数**

  用于指定外表数据的各类参数，关键参数如下所示。
  - “format”：表示导出的数据文件格式，支持“orc”和“parquet”格式。
  - “foldername”：外表中数据源文件目录，即表数据目录在HDFS文件系统上对应的文件目录。此选项对WRITE ONLY外表为必选项，对READ ONLY外表为可选项。
  - **“encoding”**：外表中数据源文件的编码格式名称，缺省为utf8。
  - **“filesize”**

    指定WRITE ONLY外表的文件大小。此选项为可选项，不指定该选项默认分布式文件系统配置中文件大小的配置值。此语法仅对WRITE ONLY的外表有效。

    取值范围：[1, 1024]的整数。

    说明

    filesize参数只对ORC格式的WRITE ONLY的HDFS外表有效。

版权所有 © 华为云计算技术有限公司

- **“compression”**

  指定文件的压缩方式，此选项为可选项。此语法仅对WRITE ONLY的外表有效。

  取值范围：zlib，snappy，lz4。

  指定ORC格式文件的压缩方式，缺省值为zlib。指定PARQUET格式文件的压缩方式，缺省值为snappy。

- **“version”**

  指定ORC格式的版本号，此选项为可选项。此语法仅对WRITE ONLY的外表有效。

  取值范围：目前仅支持0.12。缺省值为0.12。

- **“dataencoding”**

  在数据库编码与数据表的数据编码不一致时，该参数用于指定导出数据表的数据编码。比如数据库编码为Latin-1，而导出的数据表中的数据为UTF-8编码。此选项为可选项，如果不指定该选项，默认采用数据库编码。此语法仅对HDFS的WRITE ONLY外表有效。

  取值范围：该数据库编码支持转换的数据编码。

  说明

  dataencoding参数只对ORC格式的WRITE ONLY的HDFS外表有效。

- **语法中的其他参数**

  其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中不需要设置。详细的描述请参见《数据仓库服务SQL语法参考》中“CREATE FOREIGN TABLE (SQL on Hadoop or OBS)”章节。

  根据以上信息，创建外表命令如下所示：

```
DROP FOREIGN TABLE IF EXISTS product_info_ext_obs;

---建立不包含分区列的HDFS外表，表关联的外部服务器为hdfs_server，表对应的HDFS服务上的文件格式为‘orc’，HDFS上的数据存储路径为'/user/hive/warehouse/product_info_orc/'。

CREATE FOREIGN TABLE product_info_ext_obs
(
    product_price                integer        ,
    product_id                   char(30)       ,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER obs_server
OPTIONS (
format 'orc',
foldername '/user/hive/warehouse/product_info_orc/',
   compression 'snappy',
    version '0.12'
) Write Only;
```

2.6.2.5 执行导出

导出操作语法：

执行数据导出语法：

 版权所有 © 华为云计算技术有限公司

```
INSERT INTO  [foreign table 表名] SELECT * FROM [源表名];
```

执行导出数据示例

- **示例1**：将表product_info_output的数据通过外表product_info_output_ext导出到数据文件中。
  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output;
  ```
  若出现以下类似信息，说明数据导出成功。
  ```
  INSERT 0 10
  ```
- **示例2**：通过条件过滤（WHERE *product_price>500*），向数据文件中导出部分数据。
  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output WHERE product_price>500;
  ```

说明

对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。

2.6.3 使用 GDS 导出数据到远端服务器

2.6.3.1 关于 GDS 并行导出

使用GDS工具将数据从数据库导出到普通文件系统中，适用于高并发、大量数据导出的场景。

当前版本的GDS支持从数据库导出到管道文件，该功能使GDS的导出更加灵活多变。

- 当GDS用户的本地磁盘空间不足时：
  - 通过管道文件将从GDS导出的数据进行压缩减少磁盘空间。
  - 通过管道直接将导出来的数据放到hdfs服务器上。
- 当用户导出前需要清洗数据时：
  - 用户可以根据自己的需求编写程序，将需要处理的流式数据实时从管道中读取内容，完成导出的数据清洗工作。

  说明

  - 当前版本暂不支持SSL模式下GDS导出，请勿以SSL方式使用GDS。
  - 本章涉及的所有管道文件都是指linux上的命名管道。
- 在执行GDS导入导出时，为了确保数据导入或导出的正确性，需要在相同的兼容模式下操作。

  例如：在MySQL兼容模式下导入（导出）的数据，同样需要在MySQL兼容模式下才能正确导出（导入）。

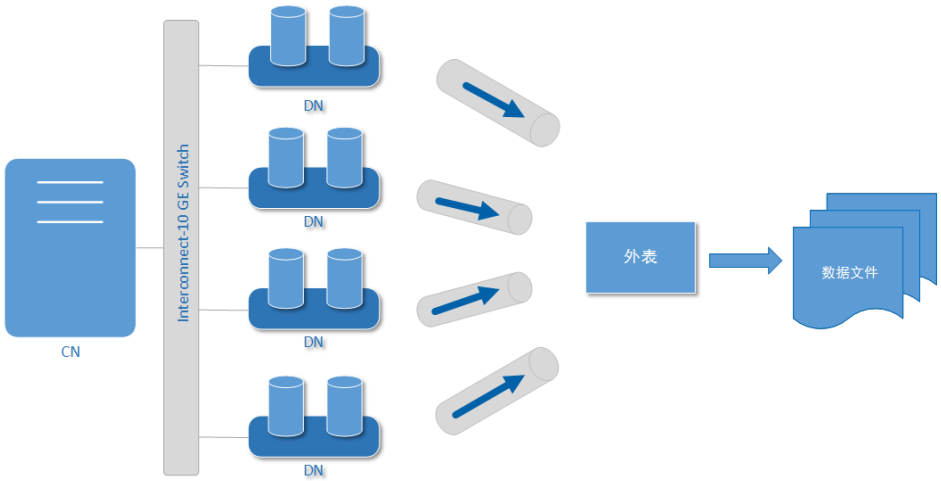
概述

通过外表导出数据：通过GDS外表设置的导出模式、导出数据格式等信息来指定待导出的数据文件，利用多DN并行的方式，将数据从数据库导出到数据文件中，从而提高整体导出性能。不支持直接导出文件到HDFS文件系统。

- CN只负责任务的规划及下发，把数据导出的工作交给了DN，释放了CN的资源，使其有能力处理外部请求。
- 通过让各个DN都参与数据导出，充分利用各个设备的计算能力及网络带宽。

版权所有 © 华为云计算技术有限公司

图 2-9 通过外表导出数据



相关概念

- **数据文件**：存储有数据的TEXT、CSV或FIXED文件。文件中保存的是从DWS数据库导出的数据。
- **外表**：用于规划导出数据文件的数据文件格式、存放位置、编码格式等信息。
- **GDS**：数据服务工具。在导出数据时，需要将此工具部署到数据文件所在的服务器上，使DN可以通过该工具导出数据。
- **表**：数据库中的表，包括行存表和列存表。数据文件中的数据从这些表中导出。
- **Remote导出模式**：将集群中的业务数据导出到集群之外的主机上。

导出模式

DWS支持的导出模式有Remote模式。

- **Remote模式：**将集群中的业务数据导出到集群之外的主机上。
  - 支持多个GDS服务并发导出，但1个GDS在同一时刻，只能为1个集群提供导出服务。
  - 配置与集群节点处于统一内网的GDS服务，导出速率受网络带宽影响，推荐的网络配置为10GE。
  - 支持数据文件格式：TEXT、CSV和FIXED。单行数据大小需<1GB。

版权所有 © 华为云计算技术有限公司

导出流程

图 2-10 并行导出流程

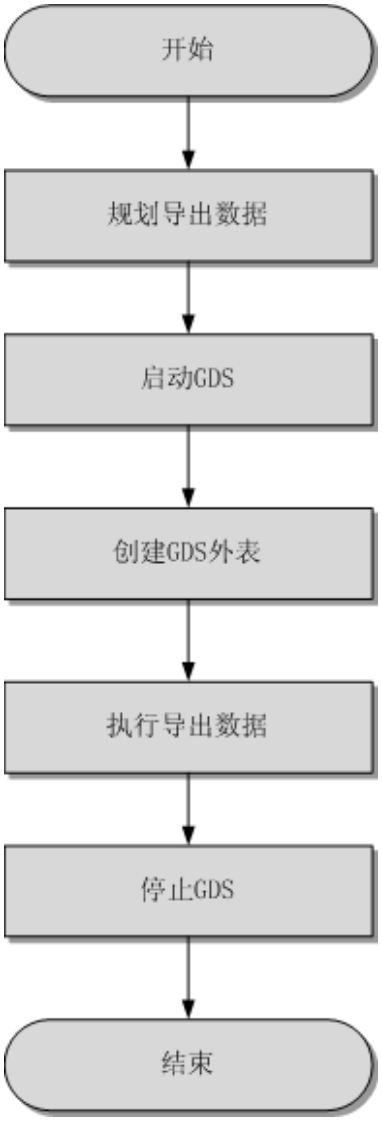


表 2-31 流程说明

流程	说明	子任务
规划导出数据。	根据所选模式，准备需要导出的数据并规划导出路径。 详细内容请参见**规划导出数据**	-
启动GDS。	若规划的导出模式为Remote模式，需在数据服务器上安装配置并启动GDS。 详细内容请参见**安装配置和启动GDS**。	-

版权所有 © 华为云计算技术有限公司

流程	说明	子任务
创建外表。	创建外表用于帮助GDS指定导出的数据文件。外表中保存了导出数据文件的位置、文件格式、编码格式、数据间的分隔符等信息。 详细内容请参见**创建GDS外表**。	-
执行导出数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导出到数据文件中。 详细内容请参见**执行导出数据**。	-
停止GDS。	数据导出完成后，停止GDS。 详细请参见**停止GDS**。	-

2.6.3.2 规划导出数据

操作场景

使用GDS从集群导出到数据之前，要提前准备需要导出的数据，并规划导出的路径。

规划导出路径

- **Remote**模式

步骤1 以root用户登录GDS数据服务器，创建导出的数据文件存放目录“/output_data”。

```
mkdir -p /output_data
```

步骤2 （可选）创建用户及所属的用户组。此用户为启动GDS的用户，该用户需要拥有导出数据文件存放目录的写权限。

```
groupadd gdsgrp
useradd -g gdsgrp gdsuser
```

若出现以下提示，说明数据库用户及所属用户组已存在，可跳过本步骤。

```
useradd: Account 'gdsuser' already exists.
groupadd: Group 'gdsgrp' already exists.
```

步骤3 修改数据文件目录属主为gdsuser。

```
chown -R gdsuser:gdsgrp /output_data
```

----结束

2.6.3.3 安装配置和启动 GDS

GDS是DWS提供的数据服务工具，通过和外表机制的配合，实现数据的高速导出。

安装配置和启动GDS操作步骤如下，详细内容请参见**安装配置和启动GDS**。

操作步骤

步骤1 以root用户登录待安装GDS的数据服务器，创建存放GDS工具包的目录。

```
mkdir -p /opt/bin/dws
```


版权所有 © 华为云计算技术有限公司

步骤2 将GDS工具包上传至上一步所创建的目录中。

以上传SUSE Linux版本的工具包为例，将GDS工具包“dws_client_8.x.x_suse_x64.zip”上传至上一步所创建的目录中。

步骤3 （可选）如果使用SSL加密传输，请一并上传SSL证书至**步骤1**所创建的目录下。

步骤4 在工具包所在目录下，解压工具包。

```
cd /opt/bin/dws
unzip dws_client_8.x.x_suse_x64.zip
```

步骤5 创建GDS专有用户及其所属的用户组。此用户用于启动GDS及读取源数据。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

步骤6 分别修改工具包和数据源文件目录属主为GDS专有用户。

```
chown -R gds_user:gdsgrp /opt/bin/dws/gds
chown -R gds_user:gdsgrp /input_data
```

步骤7 切换到gds_user用户。

```
su - gds_user
```

若当前集群版本为8.0.x及以前版本，请跳过**步骤8**，直接执行**步骤9**。

若当前集群版本为8.1.x版本，则正常执行以下步骤。

步骤8 执行环境依赖脚本（仅8.1.x版本适用）。

```
cd /opt/bin/dws/gds/bin
source gds_env
```

步骤9 启动GDS服务。

GDS是绿色软件，解压后启动即可。GDS启动方式有两种：

方式一：直接使用“gds”命令，在命令项中设置启动参数。

方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

对于集中一次性导入的场景推荐使用第一种方式。对于需要隔段时间再次导入的场景，推荐使用第二种方式以配置文件的形式提升启动效率。

- 方式一：直接使用“gds”命令，启动GDS。
  - 非SSL模式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D -t worker_num
    ```

    示例：

    ```
    /opt/bin/dws/gds/bin/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D -t 2
    ```

  - 使用SSL加密方式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D
    -t worker_num --enable-ssl --ssl-dir Cert_file
    ```

    示例：

    以**步骤3**中SSL证书已上传至/opt/bin为例，命令如下。

    ```
    /opt/bin/dws/gds/bin/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D --enable-ssl --ssl-dir /opt/bin/
    ```

  命令中的斜体部分请根据实际替换。

  - **-d** *dir*：保存有待导入数据的数据文件所在目录。本教程中为“/input_data/”。

版权所有 © 华为云计算技术有限公司

- **-p** *ip:port*：GDS监听IP和监听端口。默认值为：127.0.0.1，需要替换为能跟DWS通信的万兆网IP。监听端口的取值范围：1024~65535。默认值为：8098。本教程配置为：192.168.0.90:5000。
- **-H** *address_string*：允许哪些主机连接和使用GDS服务。参数需为CIDR格式。此参数配置的目的是允许DWS集群可以访问GDS服务进行数据导入。所以请保证所配置的网段包含DWS集群各主机。
- **-l** *log_file*：存放GDS的日志文件路径及文件名。本教程为“/opt/bin/dws/gds/gds_log.txt”。
- **-D**：后台运行GDS。仅支持Linux操作系统下使用。
- **-t** *worker_num*：设置GDS并发线程数。DWS及数据服务器上的I/O资源均充足时，可以加大并发线程数。

  GDS是根据导入事务并发数来决定服务运行线程数的。也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。
- **--enable-ssl**：启用SSL加密方式传输数据。
- **--ssl-dir** *Cert_file*：SSL证书所在目录。需与**步骤3**中的证书保存目录保持一致。
- 关于更多参数的设置信息请参考《数据仓库服务工具指南》中的“GDS并行数据加载工具 > gds命令简介”。

- 方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

  a. 使用如下命令，进入GDS工具包的“config”目录下，配置“gds.conf”文件。“gds.conf”配置详细信息请参考**表2-14**。

  ```
  vim /opt/bin/dws/gds/config/gds.conf
  ```

  示例：

  配置“gds.conf”文件如下：

  ```
  <?xml version="1.0"?>
  <config>
  <gds name="gds1" ip="192.168.0.90" port="5000" data_dir="/input_data/" err_dir="/err"
  data_seg="100MB" err_seg="100MB" log_file="/log/gds_log.txt" host="10.10.0.1/24"
  daemon='true' recursive="true" parallel="32"></gds>
  </config>
  ```

  配置文件信息如下：

  - 数据服务器所在IP为192.168.0.90，GDS监听端口为5000。
  - 数据文件存放在“/input_data/”目录下。
  - 错误日志文件存放在“/err”目录下。该目录需要拥有GDS读写权限的用户自行创建。
  - 单个数据文件大小为100MB。
  - 每个错误日志大小为100MB。
  - 日志保存在“/log/gds_log.txt”文件中。该目录需要拥有GDS读写权限的用户自行创建。
  - 只允许IP为10.10.0.*的节点进行连接。
  - GDS进程以后台方式运行。

 版权所有 © 华为云计算技术有限公司

- 递归数据文件目录。
- 指定并发导入工作线程数目为2。

b. 执行如下命令启动GDS并确认GDS是否启动成功。

```
python3 gds_ctl.py start
```

示例：

```
cd /opt/bin/dws/gds/bin
python3 gds_ctl.py start
Start GDS gds1                  [OK]
gds [options]:
 -d dir            Set data directory.
 -p port           Set GDS listening port.
    ip:port        Set GDS listening ip address and port.
 -l log_file       Set log file.
 -H secure_ip_range
                   Set secure IP checklist in CIDR notation. Required for GDS to start.
 -e dir            Set error log directory.
 -E size           Set size of per error log segment.(0 < size < 1TB)
 -S size           Set size of data segment.(1MB < size < 100TB)
 -t worker_num     Set number of worker thread in multi-thread mode, the upper limit is 200. If
without setting, the default value is 8.
 -s status_file    Enable GDS status report.
 -D                Run the GDS as a daemon process.
 -r                Read the working directory recursively.
 -h                Display usage.
```

----结束

2.6.3.4 创建 GDS 外表

操作步骤

步骤1 根据**规划导出数据**中规划的路径确定外表参数**location**的值。

- **Remote**模式

  请通过URL方式设置参数“location”，用于指定导出的数据文件存放路径。

  - 不需要指定文件名。
  - 当有多个路径时，若导出数据源数少于路径数时，多余的路径会只生成文件不写入数据。

  示例：

  GDS数据服务器IP为192.168.0.90，假定启动GDS时设置的监听端口为5000，设置的导出后文件存放目录为“/output_data/”。

  根据以上情况，在创建外表时，指定参数“location”为“gsfs://192.168.0.90:5000/”。

  说明

  - location可以指定子目录如“gsfs://192.168.0.90:5000/2019/11/”实现同一张表根据日期导出到不同目录下。
  - 现有版本在执行**导出**任务的时候会判断“/output_data/2019/11”目录是否存在，不存在则创建。导出时会将文件写入此目录下， 这样用户在创建或修改外表后就不需要再去手动执行“mkdir -p /output_data/2019/11”。

步骤2 梳理待导出数据的格式信息，确定创建外表时使用的数据格式参数的值。

步骤3 根据前面步骤确定的参数，**创建GDS外表**。

----结束

版权所有 © 华为云计算技术有限公司

示例

- **示例：**创建GDS导出外表foreign_tpcds_reasons，待导出数据格式为CSV，用于接收数据服务器上的数据。

  其中设置的**导出模式**信息如下所示：

  规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，待导出的数据文件格式为CSV，选择并行导出模式为Remote模式。

  假定启动GDS时，规划导出的数据文件存放目录为“/output_data/”，GDS监听端口为5000，所以设置参数“location”为“gsfs://192.168.0.90:5000/”。

  设置导出的**数据格式信息**，参数设置如下所示：

  - 导出数据文件格式（format）为CSV。
  - 编码格式（encoding）为UTF-8。
  - 字段分隔符（delimiter）为E'\x08'。
  - 引号字符（quote）为E'\x1b'。
  - 数据文件中空值（null）为没有引号的空字符串。
  - 逃逸字符（escape）默认和quote相同。
  - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。
  - 导出数据文件换行符样式（EOL）为0X0A。

  创建的外表如下所示：

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk    integer        not null,
  r_reason_id    char(16)       not null,
  r_reason_desc  char(100)
)
SERVER gsmpp_server
OPTIONS (LOCATION 'gsfs://192.168.0.90:5000/',
FORMAT 'CSV',
DELIMITER E'\x08',
QUOTE E'\x1b',
NULL '',
EOL '0x0a'
)
WRITE ONLY;
```

2.6.3.5 执行导出数据

前提条件

需要确保每一个CN和DN所在服务器到GDS服务器的IP和端口是互通的。

导出操作语法

执行数据导出语法：

```
INSERT INTO [foreign table 表名] SELECT * FROM [源表名];
```

说明

编写批处理任务脚本，实现并发批量导出数据。并发量视机器资源使用情况而定。可通过几个表测试，监控资源利用率，根据结果提高或减少并发量。常用资源监控命令有：内存和CPU监控top命令，I/O监控命令iostat，网络监控命令sar等。相关案例请参见**多线程导出**。

版权所有 © 华为云计算技术有限公司

任务示例

- **示例1**：将表reason的数据通过外表foreign_tpcds_reasons导出到数据文件中。
  ```
  INSERT INTO foreign_tpcds_reasons SELECT * FROM tpcds.reason;
  ```
- **示例2**：通过条件过滤（r_reason_sk =1），向数据文件中导出部分数据。
  ```
  INSERT INTO foreign_tpcds_reasons SELECT * FROM tpcds.reason WHERE r_reason_sk=1;
  ```
- **示例3**：对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。
  ```
  INSERT INTO foreign_tpcds_reasons SELECT RAWTOHEX(c) FROM tpcds.reason;
  ```

2.6.3.6 停止 GDS

GDS是DWS提供的数据服务工具，通过和外表机制的配合，实现数据的高速导出。

当不再使用GDS时，可通过以下步骤停止GDS。

操作步骤

步骤1 以gds_user用户登录安装GDS的数据服务器。

步骤2 请根据启动GDS的方式，选择停止GDS的方式。

- 若用户使用“gds”命令启动GDS，请使用以下方式停止GDS。
  a. 执行如下命令，查询GDS进程号。
  ```
  ps -ef|grep gds
  ```
  示例：其中GDS进程号为128954。
  ```
  ps -ef|grep gds
  gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data/ -p 192.168.0.90:5000 -l /log/gds_log.txt -D
  gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
  ```
  b. 使用“kill”命令，停止GDS。其中128954为上一步骤中查询出的GDS进程号。
  ```
  kill -9 128954
  ```
- 若用户使用“gds_ctl.py”命令启动GDS，请使用以下命令停止GDS。
  ```
  cd /opt/bin/dws/gds/bin
  python3 gds_ctl.py stop
  ```

----结束

2.6.3.7 GDS 导出示例

Remote 模式导出

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导出数据文件格式为CSV，所以规划的并行导出模式为Remote模式。

Remote模式并行导出数据操作示例如下所示：

1. 以root用户登录GDS数据服务器，创建数据文件存放目录“/output_data”，启动gds_user用户及所属的用户组。
   ```
   mkdir -p /output_data
   ```
2. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该类用户及所属用户组已存在，可跳过此步骤。
   ```
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```


版权所有 © 华为云计算技术有限公司

3. 修改数据服务器上数据文件目录“/output_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /output_data
   ```
4. 以gds_user用户登录数据服务器上分别启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，导出数据文件存放在“/output_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行。
   ```
   /opt/bin/dws/gds/bin/gds -d /output_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
   ```
5. 在数据库中创建外表foreign_tpcds_reasons用于接收数据服务器上的数据。

   其中设置的**导出模式信息**如下所示：
   - 由于启动GDS时，设置的导出数据文件存放目录为“/output_data/”，GDS监听端口为5000。创建的导出数据文件存放目录为“/output_data/”。所以设置参数“location”为“gsfs://192.168.0.90:5000/”。

   设置导出的**数据文件格式**信息如下所示：
   - 数据文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为E'\x1b'。
   - 数据文件中空值（null）为没有引号的空字符串。
   - 逃逸字符（escape）默认和quote相同。
   - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。

   根据以上信息，创建的外表如下所示：
   ```
   CREATE FOREIGN TABLE foreign_tpcds_reasons
   (
     r_reason_sk    integer        not null,
     r_reason_id    char(16)       not null,
     r_reason_desc  char(100)
   )
   SERVER gsmpp_server
   OPTIONS
   (
   LOCATION 'gsfs://192.168.0.90:5000/',
   FORMAT 'CSV',
   ENCODING 'utf8',
   DELIMITER E'\x08',
   QUOTE E'\x1b',
   NULL ''
   )
   WRITE ONLY;
   ```
6. 在数据库上，通过外表foreign_tpcds_reasons，将数据导出到数据文件中。
   ```
   INSERT INTO foreign_tpcds_reasons SELECT * FROM tpcds.reason;
   ```
7. 待数据导出完成后，以gds_user用户登录数据服务器，停止GDS。

   其中GDS进程号为128954。
   ```
   ps -ef|grep gds
   gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /output_data -p 192.168.0.90:5000 -D
   gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
   kill -9 128954
   ```

多线程导出

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导出的数据文件格式为CSV，同时导出2个目标表，所以规划使用Remote模式进行多线程导出。

Remote模式多线程导出数据操作示例如下所示：

 版权所有 © 华为云计算技术有限公司

1. 以root用户登录GDS数据服务器，创建导出数据文件存放目录“/output_data”，数据库用户及所属的用户组。
   ```
   mkdir -p /output_data
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
2. 修改数据服务器上数据文件目录“/output_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /output_data
   ```
3. 以gds_user用户登录数据服务器上启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，导出数据文件存放在“/output_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行，设定并发度为2。
   ```
   /opt/bin/dws/gds/bin/gds -d /output_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D -t 2
   ```
4. 在DWS上，创建外表foreign_tpcds_reasons1和foreign_tpcds_reasons2用于接收数据服务器上的数据。
   - 其中设置的**导出模式信息**如下所示：
     - 由于启动GDS时，设置的导出数据文件存放目录为“/output_data/”，GDS监听端口为5000。创建的导出数据文件存放目录为“/output_data/”。所以设置参数“location”为“gsfs://192.168.0.90:5000/”。
   - 设置导出的数据**文件格式信息**如下所示：
     - 数据文件格式（format）为CSV。
     - 编码格式（encoding）为UTF-8。
     - 字段分隔符（delimiter）为E'\x08'。
     - 引号字符（quote）为E'\x1b'。
     - 数据文件中空值（null）为没有引号的空字符串。
     - 逃逸字符（escape）默认和quote相同。
     - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。

   根据以上信息，创建的外表foreign_tpcds_reasons1如下所示：
   ```
   CREATE FOREIGN TABLE foreign_tpcds_reasons1
   (
     r_reason_sk    integer      not null,
     r_reason_id    char(16)     not null,
     r_reason_desc  char(100)
   )
   SERVER gsmpp_server
   OPTIONS
   (
   LOCATION 'gsfs://192.168.0.90:5000/',
   FORMAT 'CSV',
   ENCODING 'utf8',
   DELIMITER E'\x08',
   QUOTE E'\x1b',
   NULL ''
   )
   WRITE ONLY;
   ```
   参考以上设置，创建的外表foreign_tpcds_reasons2如下所示：
   ```
   CREATE FOREIGN TABLE foreign_tpcds_reasons2
   (
   ```


版权所有 © 华为云计算技术有限公司

```
 r_reason_sk    integer      not null,
 r_reason_id    char(16)     not null,
 r_reason_desc  char(100)
)
SERVER gsmpp_server
OPTIONS
(
LOCATION 'gsfs://192.168.0.90:5000/',
FORMAT 'CSV',
DELIMITER E'\x08',
QUOTE E'\x1b',
NULL ''
)
WRITE ONLY;
```

5. 在数据库中通过外表foreign_tpcds_reasons1和foreign_tpcds_reasons2，将表reasons1和reasons2中的数据导出到目录“/output_data”中：

```
INSERT INTO foreign_tpcds_reasons1 SELECT * FROM tpcds.reason;
INSERT INTO foreign_tpcds_reasons2 SELECT * FROM tpcds.reason;
```

6. 待数据导出完成后，以gds_user用户登录数据服务器，停止GDS。

   其中GDS进程号为128954。

```
ps -ef|grep gds
gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /output_data -p 192.168.0.90:5000 -D -t 2
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

单个管道文件导出

步骤1 启动GDS。

```
gds -d /***/gds_data/ -D -p 192.168.0.1:7789 -l /***/gds_log/aa.log -H 0/0 -t 10 -D
```

如果需要设置管道文件的超时时间，则使用--pipe-timeout参数设置。

步骤2 执行数据导出。

1. 登录数据库创建内表，并写入数据。

```
CREATE TABLE test_pipe( id integer not null, gender text not null, name text ) ;



INSERT INTO test_pipe values(1,2,'11111111111111');
INSERT INTO test_pipe values(2,2,'11111111111111');
INSERT INTO test_pipe values(3,2,'11111111111111');
INSERT INTO test_pipe values(4,2,'11111111111111');
INSERT 0 1
```

2. 创建只写外表。

```
CREATE FOREIGN TABLE foreign_test_pipe_tw( id integer not null, age text not null, name  text )
SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://192.168.0.1:7789/', FORMAT 'text', DELIMITER ',',
NULL '', EOL '0x0a' ,file_type 'pipe', auto_create_pipe 'false') WRITE ONLY;
```

3. 导出语句，此时语句会阻塞。

```
INSERT INTO foreign_test_pipe_tw select * from test_pipe;
```

步骤3 通过管道文件将数据从GDS导出。

1. 登录GDS，进入GDS数据目录：

```
cd /***/gds_data/
```

2. 创建管道文件，如果auto_create_pipe设置为true跳过此步骤：

```
mkfifo postgres_public_foreign_test_pipe_tw.pipe
```

   说明

   管道文件创建完成后，每执行完一次操作，业务会被自动清理。如果还需要执行其他业务，请参考该步骤重新创建管道文件。

3. 读取管道文件并写入新文件：

 版权所有 © 华为云计算技术有限公司

```
cat postgres_public_foreign_test_pipe_tw.pipe > postgres_public_foreign_test_pipe_tw.txt
```

4. 若需要对导出的文件进行压缩执行：

```
gzip -9 -c < postgres_public_foreign_test_pipe_tw.pipe  > out.gz
```

5. 若需要将管道文件的内容导出到hdfs服务器：

```
cat postgres_public_foreign_test_pipe_tw.pipe  | hdfs dfs -put -  /user/hive/***/test_pipe.txt
```

步骤4 验证导出的数据。

1. 查看文件是否导出正确。

```
cat postgres_public_foreign_test_pipe_tw.txt
3,2,11111111111111
1,2,11111111111111
2,2,11111111111111
4,2,11111111111111
```

2. 查看压缩后的文件。

```
vim out.gz
3,2,11111111111111
1,2,11111111111111
2,2,11111111111111
4,2,11111111111111
```

3. 查看导出到hdfs服务器上的数据。

```
hdfs dfs -cat /user/hive/***/test_pipe.txt
3,2,11111111111111
1,2,11111111111111
2,2,11111111111111
4,2,11111111111111
```

----结束

多进程管道文件导出

GDS也支持多进程管道文件导入导出， 即启动一个外表对应多个GDS。

以本地文件的导出为例：

步骤1 启动多个GDS。

```
gds -d /***/gds_data/ -D -p 192.168.0.1:7789 -l /***/gds_log/aa.log -H 0/0 -t 10 -D
gds -d /***/gds_data_1/ -D -p 192.168.0.1:7790 -l /***/gds_log/aa.log -H 0/0 -t 10 -D
```

如果需要设置管道文件的超时时间，则使用--pipe-timeout参数设置。

步骤2 执行数据导出。

1. 登录数据库创建内表。

```
CREATE TABLE test_pipe (id integer not null, gender text not null, name  text);
```

2. 写入数据。

```
INSERT INTO test_pipe values(1,2,'11111111111111');
INSERT INTO test_pipe values(2,2,'11111111111111');
INSERT INTO test_pipe values(3,2,'11111111111111');
INSERT INTO test_pipe values(4,2,'11111111111111');
```

3. 创建只写外表。

```
CREATE FOREIGN TABLE foreign_test_pipe_tw( id integer not null, age text not null, name  text )
SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://192.168.0.1:7789/|gsfs://192.168.0.1:7790/',
FORMAT 'text', DELIMITER ',',  NULL '', EOL '0x0a' ,file_type 'pipe', auto_create_pipe 'false') WRITE
ONLY;
```

4. 导出语句，此时语句会阻塞。

```
INSERT INTO foreign_test_pipe_tw select * from test_pipe;
```

步骤3 通过管道文件将数据从GDS导出。

1. 登录GDS，分别进入GDS数据目录。

```
cd /***/gds_data/
cd /***/gds_data_1/
```


版权所有 © 华为云计算技术有限公司

2. 创建管道文件，如果auto_create_pipe设置为true跳过此步骤。
```
mkfifo postgres_public_foreign_test_pipe_tw.pipe
```
3. 分别读取管道文件并写入新文件。
```
cat postgres_public_foreign_test_pipe_tw.pipe > postgres_public_foreign_test_pipe_tw.txt
```

步骤4 验证导出的数据。

```
cat /***/gds_data/postgres_public_foreign_test_pipe_tw.txt
3,2,11111111111111
cat /***/gds_data_1/postgres_public_foreign_test_pipe_tw.txt
1,2,11111111111111
2,2,11111111111111
4,2,11111111111111
```

----结束

2.7 其他操作

2.7.1 GDS 管道文件常见问题

注意事项

- GDS支持并发导入导出，**gds -t**参数用于设置GDS的工作线程池大小，控制并发场景下同时工作的工作线程数且不会加速单个SQL任务。**gds -t**缺省值为8，上限值为200。在使用管道功能进行导入导出时，**-t**参数应不低于业务并发数。如果是双集群互联互通场景，**-t**参数应不低于业务并发数的两倍。
- 由于管道“读取即删除”的特点，需确保导入或导出过程中除GDS程序外无其他程序读取管道文件，避免导入过程中数据丢失或者任务报错及导出的文件内容混乱。
- 不支持对具有相同Location的外表并发导入导出，即GDS的多个线程同时读取管道文件或者同时写入管道文件。
- GDS的单个导入导出任务只识别一个管道文件，因此不要对GDS外表设置带有通配符({}[]?)的Location地址。如：
```
CREATE FOREIGN TABLE foreign_test_pipe_tr( like test_pipe ) SERVER gsmpp_server OPTIONS
(LOCATION 'gsfs://192.168.0.1:7789/foreign_test_*', FORMAT 'text', DELIMITER ',',  NULL '', EOL
'0x0a' ,file_type 'pipe',auto_create_pipe 'false');
```
- GDS启用**-r**递归参数时只识别一个管道文件，即GDS只会识别当前数据目录下的一个管道文件而不会递归寻找，因此**-r**参数在管道文件导入导出场景下不生效。
- 管道文件的导入导出不支持CN Retry，因为GDS无法控制对端用户和程序操作管道的行为。
- GDS导入时默认对端程序超过1小时未向管道中写入数据导入任务将会超时报错。
- GDS导出时默认对端程序超过1小时未从管道中读数据导出任务将会超时报错。
- 需确保GDS版本和数据库内核版本都已经支持管道文件导入导出功能。
- 当外表参数auto_create_pipe设置为true时，GDS自动创建管道文件可能存在延迟，因此操作管道文件时建议先判断自动创建的管道文件是否存在，且是否为管道文件类型。
- GDS管道文件的导入导出任务结束后会自动删除管道文件，但是手动终止任务时，管道文件的删除会有延迟，直到到达超时时间后才会被删除。

版权所有 © 华为云计算技术有限公司

常见问题和定位方法：

- 问题1："/***/postgres_public_foreign_test_pipe_tr.pipe" must be named pipe.

  定位方法：GDS的外表file_type类型为pipe但是操作的文件却是一个普通文件类型。应该排查postgres_public_foreign_test_pipe_tr.pipe是否是为管道文件。

- 问题2：could not open pipe "/***/postgres_public_foreign_test_pipe_tw.pipe" cause by Permission denied.

  定位方法：GDS没有权限打开管道文件。

- 问题3：could not open source file /*****/ postgres_public_foreign_test_pipe_tw.pipe because timeout 300s for WRITING.

  定位方法：GDS导出时打开管道文件超时，一般由于auto_create_pipe为false时候，管道文件在300秒内未被创建，或者创建了但是300秒内没有程序读取该管道文件。

- 问题4：could not open source file /*****/ postgres_public_foreign_test_pipe_tw.pipe because timeout 300s for READING.

  定位方法：GDS导出时打开管道文件超时，一般由于auto_create_pipe为false时候，管道文件在300秒内未被创建或者创建了但是300秒之内没有程序写入该管道文件。

- 问题5：could not poll writing source pipe file "/****/ postgres_public_foreign_test_pipe_tw.pipe" timeout 300s.

  定位方法：GDS导出时超过300秒未等到管道上的写事件，一般由于该管道文件超过300秒没有被读取。

- 问题6：could not poll reading source pipe file "/****/ postgres_public_foreign_test_pipe_tw.pipe" timeout 300s.

  定位方法：GDS导入时超过300秒未等到管道上的读事件，一般由于该管道文件超过300秒没有被写入。

- 问题7：could not open pipe file "/***/ postgres_public_foreign_test_pipe_tw.pipe" for "WRITING" with error No such device or address.

  定位方法：表示当前"/***/postgres_public_foreign_test_pipe_tw.pipe"管道文件没有程序正在读取导致GDS无法以写的方式打开管道文件。

2.7.2 查看数据倾斜状态

操作场景

数据倾斜会造成查询表性能下降。对于记录数超过千万条的表，建议在执行全量数据导入前，先导入部分数据，以进行数据倾斜检查和调整分布列，避免导入大量数据后发现数据倾斜，调整成本高。

背景信息

DWS是采用Shared-nothing架构的MPP（Massive Parallel Processor，大规模并发处理）系统，采用水平分布的方式，将业务数据表的元组按合适的分布策略分散存储在所有的DN。

当前产品支持复制（Replication）、散列（Hash）和轮询（Roundrobin）三种用户表分布策略。

版权所有 © 华为云计算技术有限公司

- Replication方式：在每一个DN上存储一份全量表数据。对于数据量比较小的表建议采取Replication分布策略。
- Hash方式：采用这种分布方式，需要为用户表指定一个分布列（distribute key）。当插入一条记录时，系统会根据分布列的值进行Hash运算后，将数据存储在对应的DN中。对于数据量比较大的表建议采取Hash分布策略。
- Roundrobin方式：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。对于数据量比较大的表，如果Hash分布找不到一个合适的分布列，建议采用Roundrobin分布策略。

对于Hash分布策略，如果分布列选择不当，可能导致数据倾斜。因此在采用Hash分布策略之后会对用户表的数据进行数据倾斜性检查，以确保数据在各个DN上是均匀分布的。一般情况下分布列都是选择键值重复度小，数据分布比较均匀的列。

操作步骤

步骤1 分析数据源特征，选择若干个键值重复度小，数据分布比较均匀的备选分布列。

步骤2 从**步骤1**中选择一个备选分布列创建目标表。

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name
  ({ column_name data_type [ compress_mode ] [ COLLATE collation ] [ column_constraint [ ... ] ]
  | table_constraint   | LIKE source_table [ like_option [...] ] }
  [, ... ])   [ WITH ( {storage_parameter = value} [, ... ] ) ]
  [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
  [ COMPRESS | NOCOMPRESS ]   [ TABLESPACE tablespace_name ]
  [ DISTRIBUTE BY { REPLICATION
                  | ROUNDROBIN
                  | { HASH ( column_name [,...] ) } } ];
```

步骤3 参照前面章节中的办法向目标表中导入小批量数据。

对于单个数据源文件，在导入时，可通过均匀切割，导入部分切割后的数据源文件来验证数据倾斜性。

步骤4 检验数据倾斜性。命令中的table_name ，请填入实际的目标表名。

```
SELECT a.count,b.node_name FROM (SELECT count(*) AS count,xc_node_id FROM table_name GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
```

步骤5 若各DN上数据分布差小于10%，表明数据分布均衡，选择的分布列合适。请清理已导入小批量数据，导入全量数据，以完成数据迁移。

若各DN上数据分布差大于等于10%，表明数据分布倾斜，请从**步骤1**的备选分布列中删除该列，删除目标表，并重复**步骤2** 、**步骤3** 、**步骤4** 和**步骤5**。

说明

此处的数据分布差表示实际查询到DN上的数据量与DN平均数据量的差异。分布差可以通过视图**PGXC_GET_TABLE_SKEWNESS**的查看。

步骤6 （可选）如果上述步骤不能选出适合的分布列，需要从备选分布列选择多个列的组合作为分布列来完成数据迁移。

----结束

示例

对目标表staffs选择合适的分布列。

1. 分析表staffs的数据源特征，选择数据重复度低且分布均匀的备选分布列staff_ID、FIRST_NAME和LAST_NAME。

 版权所有 © 华为云计算技术有限公司

2. 先选择staff_ID作为分布列，创建目标表staffs。

```
CREATE TABLE staffs
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
DISTRIBUTE BY hash(staff_ID);
```

3. 向目标表staffs中导入部分数据。

   根据以下查询所得，集群环境中主DN数为8个，则建议导入的记录数为80000条。

```
SELECT count(*) FROM pgxc_node where node_type='D';
 count
-------
     8
(1 row)
```

4. 校验以staff_ID为分布列的目标表staffs的数据倾斜性。

```
SELECT a.count,b.node_name FROM (select count(*) as count,xc_node_id FROM staffs GROUP BY
xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
count | node_name
------+-----------
11010 | datanode4
10000 | datanode3
12001 | datanode2
 8995 | datanode1
10000 | datanode5
 7999 | datanode6
 9995 | datanode7
10000 | datanode8
(8 rows)
```

5. 根据上一步骤查询所得，各DN上数据分布差大于10%，数据分布倾斜。所以从步骤1的备选分布列中删除该列，并删除目标表staffs。

```
DROP TABLE staffs;
```

6. 尝试选择staff_ID、FIRST_NAME和LAST_NAME的组合作为分布列，创建目标表staffs。

```
CREATE TABLE staffs
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
DISTRIBUTE BY hash(staff_ID,FIRST_NAME,LAST_NAME);
```

7. 校验以staff_ID、FIRST_NAME和LAST_NAME的组合为分布列的目标表staffs的数据倾斜性。

```
SELECT a.count,b.node_name FROM (select count(*) as count,xc_node_id FROM staffs GROUP BY
xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
count | node_name
------+-----------
```


版权所有 © 华为云计算技术有限公司

```
10010 | datanode4
10000 | datanode3
10001 | datanode2
 9995 | datanode1
10000 | datanode5
 9999 | datanode6
 9995 | datanode7
10000 | datanode8
(8 rows)
```

8. 根据上一步骤查询所得，各DN上数据分布差小于10%，数据分布均衡，选择的分布列合适。
9. 清理已导入小批量数据。
   ```
   TRUNCATE TABLE staffs;
   ```
10. 导入全量数据，以完成数据迁移。

2.7.3 分析表

执行计划生成器需要使用表的统计信息，以生成最有效的查询执行计划，提高查询性能。因此数据导入完成后，建议执行ANALYZE语句生成最新的表统计信息。统计结果存储在系统表PG_STATISTIC中。

分析表

ANALYZE支持的表类型有行/列存表、HDFS表、ORC/CARBONDATA格式的OBS外表。ANALYZE同时也支持对本地表的指定列进行信息统计。

以表product_info为例，ANALYZE命令如下：

```
ANALYZE product_info;
```

表自动分析

DWS提供了三种场景下表的自动分析。

- 当查询中存在“统计信息完全缺失”或“修改量达到analyze阈值”的表，且执行计划不采取FQS（Fast Query Shipping）执行时，则通过GUC参数**autoanalyze**控制此场景下表统计信息的自动收集。此时，查询语句会等待统计信息收集成功后，生成更优的执行计划，再执行原查询语句。
- 当autovacuum设置为on时，系统会定时启动autovacuum线程，对“修改量达到analyze阈值”的表在后台自动进行统计信息收集。

表 2-32 表自动分析

触发方式	触发条件	触发频率	控制参数	备注
同步	统计信息完全缺失	查询时	autoanalyze	truncate主表时会清空统计信息。
同步	数据修改量达到analyze阈值	查询时	autoanalyze	先触发analyze，后选择最优计划。
异步	数据修改量达到analyze阈值	autovacuum线程轮询检查	autovacuum_mode, autovacuum_naptime	2s等锁超时, 5min执行超时。

版权所有 © 华为云计算技术有限公司

须知

- autoanalyze只支持默认采样方式，不支持百分比采样方式。
- 多列统计信息仅支持百分比采样，因此autoanalyze不收集多列统计信息。
- 查询过程因表的“统计信息完全缺失”和“修改量达到analyze阈值”而自动触发autoanalyze的场景，当前不支持对外表触发autoanalyze，不支持对带有ON COMMIT [DELETE ROWS | DROP]选项的临时表触发autoanalyze。
- 修改量达到analyze阈值是指：表的修改量超过autovacuum_analyze_threshold + autovacuum_analyze_scale_factor * reltuples，其中reltuples是pg_class中记录的表的估算行数。
- 基于定时启动的autovacuum线程触发的autoanalyze，仅支持行存表和列存表，不支持外表、HDFS表、OBS外表、临时表、unlogged表和toast表。
- 查询时触发analyze会对分区表的所有分区加四级锁，直到查询所在事务提交后才会放锁。四级锁不堵塞增删改查，但会堵塞分区的修改操作，比如分区的truncate，可以通过将object_mtime_record_mode设置为disable_partition，实现提前释放分区锁。
- autovacuum自动清理功能的生效还依赖于下面两个GUC参数：
  - **track_counts**参数需要设置为on，开启收集收据库统计数据功能。
  - **autovacuum_max_workers**参数需要大于0，该参数表示能同时运行的自动清理线程的最大数量。

版权所有 © 华为云计算技术有限公司

3 DWS 开发设计建议

3.1 DWS 总体开发设计规范

目的

本规范定义了在进行基于DWS数据库自主开发过程中应遵守的设计、开发准则，以提升开发效率，保证业务的连续性、稳定性。

适用范围

该规范适用于所有基于DWS数据库自主开发的场景，包含应用层面的设计与开发、数据库服务层面的设计与开发。

术语定义

规则：数据库设计与开发时必须遵守的约定。

建议：数据库设计与开发时建议考虑的约定。

说明：对规则/建议进行的详细说明和解释。

总体开发设计规范

下表是DWS开发过程中需遵循的**开发设计规范全集列表**，可以单击链接跳转到对应的规则下了解详细说明。

表 3-1 DWS 开发设计规范全集列表

编号	类别		规则/建议
1	连接管理规范	-	**规则1.1 DWS集群必须配置负载均衡**
2			**规则1.2 连接数据库完成所需操作后，必须关闭数据库连接（连接池场景除外）**
3			**规则1.3 开启的事务最后必须提交或回滚**

版权所有 © 华为云计算技术有限公司

编号	类别		规则/建议
4			规则1.4 应用侧使用连接池场景，其空闲超时配置必须小于服务侧的SESSION_TIMEOUT配置
5			规则1.5 应用侧使用连接池场景，如使用连接SET设置过参数，当将连接归还连接池前，必须进行参数重置
6			规则1.6 应用侧使用连接池场景，如使用连接创建过临时表，当将连接归还连接池前，必须手动清理所创建的临时表
7	对象设计规范	DATABASE对象设计	规则2.1 避免直接使用内置的DATABASE（如postgres、gaussdb等）
8			规则2.2 创建DATABASE时必须选择正确的数据库编码
9			规则2.3 创建DATABASE时必须选择正确的数据库兼容模式
10			建议2.4 存在关联计算的对象放在同一个DATABASE中
11		USER对象设计	规则2.5 禁止使用特殊权限用户运行业务，需遵循权限最小分配原则
12			规则2.6 禁止使用一个数据库用户运行所有业务
13		SCHEMA对象设计	建议2.7 不在其他USER的私有SCHEMA下创建对象
14		TABLESPACE对象设计	规则2.8 禁止自定义TABLESPACE表空间
15		TABLE对象设计（重点）	规则2.9 创建表时必须选择正确的分布方式和分布列
16			规则2.10 创建表时必须选择正确的存储方式
17			规则2.11 创建表时必须选择正确的分区策略
18			建议2.12 表字段的设计要遵循高效、准确原则
19			建议2.13 避免使用自增列或自增数据类型
20		INDEX对象设计（重点）	规则2.14 只创建必要的索引，创建索引必须选择合适的列和顺序
21			建议2.15 列存表通常可不建索引，极致性能场景需正确选择索引类型
22		VIEW对象设计	建议2.16 视图的嵌套需避免超过三层
23	SQL开发规范	DDL操作规范	建议3.1 DDL操作（CREATE除外）避免在业务高峰期和长事务中执行
24			规则3.2 DROP删除对象操作必须明确删除对象范围

版权所有 © 华为云计算技术有限公司

编号	类别		规则/建议
25		INSERT操作规范	规则3.3 INSERT多VALUES批插场景使用COPY替代
26			建议3.4 禁止针对普通列存表进行实时INSERT操作
27		UPDATE/DELETE操作规范	建议3.5 避免并发UPDATE/DELETE行存表的同一行
28			建议3.6 避免对列存表频繁或并发执行UPDATE/DELETE
29		SELECT操作（含所有语法中的查询部分）规范	规则3.7 禁止执行不下推的SQL
30			规则3.8 禁止多表关联时缺少关联条件
31			规则3.9 多表关联字段数据类型要保持一致
32			建议3.10 尽量避免对关联条件字段和过滤条件字段进行函数运算
33			建议3.11 资源高消耗型SQL需做好压测和并发管控
34			规则3.12 禁止针对行存大表的频繁COUNT
35			建议3.13 避免查询返回超大结果集（数据导出场景除外）
36			建议3.14 查询时避免使用“SELECT *”写法
37			建议3.15 谨慎使用递归语句(WITH RECURSIVE)，明确终止条件，确保递归可终止
38			建议3.16 访问对象（表，函数等）时带上SCHEMA名称
39			建议3.17 针对SQL标记注释，唯一标识SQL的归属
40			建议3.18 SQL语句的长度不要超过64K
41	外表功能开发规范	GDS外表	规则4.1 GDS服务须单独使用服务器部署在DWS集群外
42		协同分析外表	规则4.2 避免同时对多个协同分析外表进行跨集群并发访问
43	存储过程开发规范	-	建议5.1 避免使用复杂的存储过程，避免存储过程嵌套
44			规则5.2 存储过程内避免执行非CREATE类的DDL操作

版权所有 © 华为云计算技术有限公司

3.2 DWS 连接管理规范

规则 1.1 DWS 集群必须配置负载均衡

说明

违反规则的影响：

- 负载不均衡导致性能问题，严重情况会导致业务中断。
- CN故障场景，业务无法自恢复或恢复时间长。

方案建议：

- 配置ELB负载均衡，应用连接负载均衡IP。

规则 1.2 连接数据库完成所需操作后，必须关闭数据库连接（连接池场景除外）

说明

违反规则的影响：

- 空闲连接过多，触发连接上限，导致新建连接报错。
- 空闲连接过多，内部占用过多资源，导致资源过载。

方案建议：

- 应用侧与数据库建连并使用完成后，手动关闭连接。
- 服务侧设置session_timeout参数，连接空闲超时自动断开。

规则 1.3 开启的事务最后必须提交或回滚

说明

违反规则的影响：

- 事务长时间不提交，持锁阻塞ALTER等操作，进而阻塞所有业务。
- 大量“idle in transaction”连接，触发连接上限，导致新建连接报错。

方案建议：

- 默认使用autocommit自动提交方式，如关闭autocommit，则必须手动提交。
- 显式start transaction开启的事务，执行完相关操作后，必须显式commit/rollback结束事务。

规则 1.4 应用侧使用连接池场景，其空闲超时配置必须小于服务侧的 SESSION_TIMEOUT 配置

说明

违反规则的影响：

- 连接池的连接被服务侧的空闲超时机制清理，连接复用效果下降。

方案建议：

- 将连接池的空闲超时参数设置为小于服务侧的SESSION_TIMEOUT值，或调大SESSION_TIMEOUT为大于连接池超时参数（不推荐）。

版权所有 © 华为云计算技术有限公司

规则 1.5 应用侧使用连接池场景，如使用连接 SET 设置过参数，当将连接归还连接池前，必须进行参数重置

说明

违反规则的影响：

- 连接被其他业务复用时，可能会因复用其他业务设置的参数出现业务性能、业务报错等问题。

方案建议：

- 在将连接归还连接池之前，使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”重置参数。

注意：

在应用侧使用连接池的场景，如果在DWS服务侧通过GS_GUC RELOAD设置了全局的GUC参数，需要重启应用侧连接池该参数才可以生效，因为该设置只针对新建的连接生效，针对连接池中原有的连接不生效。

规则 1.6 应用侧使用连接池场景，如使用连接创建过临时表，当将连接归还连接池前，必须手动清理所创建的临时表

说明

违反规则的影响：

- 连接被其他业务复用时，可能出现创建临时表报错问题。

方案建议：

- 在将连接归还连接池之前，使用DROP清理当前SESSION创建的临时表。

3.3 DWS 对象设计规范

3.3.1 DATABASE 对象设计规范

规则 2.1 避免直接使用内置的 DATABASE（如 postgres、gaussdb 等）

说明

违反规则的影响：

- 内置的数据库编码、兼容模式不符合业务要求时，需要重新迁移数据。
- 所有业务均使用内置数据库，影响变更耗时。

方案建议：

- 根据实际业务需要，合理创建专用DATABASE并划分给业务使用。

规则 2.2 创建 DATABASE 时必须选择正确的数据库编码

说明

违反规则的影响：

- 选错数据库编码可能导致数据乱码，且不支持直接修改数据库编码，需重新建库重新导入数据。

方案建议：

- 通常建议建库时指定ENCODING为UTF-8编码，特殊场景根据实际情况而定。

版权所有 © 华为云计算技术有限公司

规则 2.3 创建 DATABASE 时必须选择正确的数据库兼容模式

说明

违反规则的影响：

- 选错数据库兼容模式，会导致从其他厂商的数据库迁移到DWS后出现行为不一致问题，且不支持直接修改数据库兼容模式，需重新建库重新导数

方案建议：

- 根据源库端数据库类型，在DWS中建库时通过DBCOMPATIBILITY指定兼容模式，当前支持Teradata、Oracle和MySQL等多种兼容模式。

建议 2.4 存在关联计算的对象创建在同一个 DATABASE 中

说明

违反规则的影响：

- 跨库访问无论使用哪种方案，性能均劣于同一DATABASE内的关联操作。

方案建议：

- 创建多个DATABASE的场景，将需要执行关联计算的对象，创建在同一DATABASE中。

3.3.2 USER 对象设计规范

规则 2.5 禁止使用特殊权限用户运行业务，需遵循权限最小分配原则

说明

违反规则的影响：

- 特殊权限用户如管理员用户，均具有特殊用途，使用此类用户运行业务有安全和管控风险。

方案建议：

- 使用普通用户运行业务，特殊权限类用户仅用于管理类操作。

规则 2.6 禁止使用一个数据库用户运行所有业务

说明

违反规则的影响：

- 同一个数据库用户运行所有业务不利于业务管控，异常场景无法针对特定用户做紧急隔离。

方案建议：

- 根据用途规划管理员、业务运行账号、运维账号等。
- 根据业务模块进行用户细分，以便通过用户进行业务和资源的划分和管控。

版权所有 © 华为云计算技术有限公司

3.3.3 SCHEMA 对象设计规范

建议 2.7 不在其他 USER 的私有 SCHEMA 下创建对象

说明

私有SCHEMA是指创建USER时自带的同名SCHEMA，该SCHEMA为该USER私有。

违反规则的影响：

- 在其他用户私有SCHEMA下创建对象，对象权限不受创建者控制，OWNER也非创建者。

方案建议：

- 在自身私有SCHEMA或自身创建SCHEMA下创建对象，不在其他用户私有SCHEMA下创建对象。

3.3.4 TABLESPACE 对象设计规范

规则 2.8 禁止自定义 TABLESPACE 表空间

说明

违反规则的影响：

- 分布式场景建表使用自定义表空间，导致表数据无法按照各DN分布式存储，出现存储倾斜。

方案建议：

- 创建表对象使用内置默认表空间。

3.3.5 TABLE 对象设计规范（重点）

规则 2.9 创建表时必须选择正确的分布方式和分布列

说明

违反规范的影响：

- 分布式和分布列选择错误，导致表数据存储倾斜，访问性能下降，严重情况会触发存储和计算资源过载。

方案建议：

- 创建表时通过DISTRIBUTE BY显式指定分布方式和分布列，分布列选择原则如下表所示。

表 3-2 分布列选择原则

分布方式	描述	适用场景
Hash	表数据按照分布列生成的hash值与DN实例的映射关系，将数据分布到各DN实例。 • **优点**：每个DN仅包含部分数据，占用整体空间小。 • **缺点**：数据分布的均匀程度强依赖分布列的选择；JOIN关联条件不包含各自分布列的场景存在节点间数据通信的消耗。	大表、事实表。

版权所有 © 华为云计算技术有限公司

分布方式	描述	适用场景
RoundRobin	表数据按照轮询的方式依次分布到各DN实例。 ● **优点**：每个DN仅包含部分数据，占用整体空间小；数据轮询均匀分布，不依赖分布列，不存在数据存储倾斜问题。 ● **缺点**：无法通过分布列条件消除和减少节点间通信，此类场景性能不如HASH。	大表、事实表，无合适分布列的表。
Replication	表中的全量数据在集群的每一个DN实例上保留一份。 ● **优点**：每个DN上都有此表的全量数据，JOIN操作中可以完全避免节点间数据通信，从而减小网络开销，同时减少了STREAM线程启停开销。 ● **缺点**：每个DN都保留了表的完整数据，数据的冗余，占用更多存储空间。	小表、维度表。

规则 2.10 创建表时必须选择正确的存储方式

说明

违反规范的影响：

- 行存表使用不当导致查询场景性能差，资源过载。
- 列存表使用不当导致CU膨胀，性能差，资源过载。

方案建议：

- 创建表时通过orientation参数显式指定表的存储类型，存储类型的选择原则如下表所示。

表 3-3 存储类型选择

存储方式	优势场景	劣势的场景
行存(row)	● DML增删改：UPDATE和DELETE操作多的场景。 ● DML查询：点查询(返回记录少，基于索引的简单查询)。	DML查询：统计分析类查询(group , join的数据量大的场景)。 **注意** **创建行存表（orientation = row）时，禁止指定compress属性，禁止使用行存压缩表。**

版权所有 © 华为云计算技术有限公司

存储方式	优势场景	劣势的场景
列存(column)	• DML增删改：INSERT批量导入场景（单次单分区入库量接近或大于6w*DN数）。 • DML查询：统计分析类查询(group，join的数据量大的场景)。	• DML增删改：UPDATE/DELETE多的场景、INSERT小批量插入的场景。 • DML查询：高并发的点查询。
行列混存(hstore&hstore_opt)	DWS的行列混存表仅**9.1.0**及以上集群版本支持，其提升了实时入库的能力，兼顾部分行存和列存的优点。 • DML增删改：INSERT/UPSERT/COPY实时入库场景和批量入库（除COPY微批场景外，hstore_opt全面优于hstore）。 • DML查询：统计分析类查询(group，join的数据量大的场景)。	主要是极端的TP（事务处理）类场景。 • DML增删改：高频并发UPDATE/DELETE场景。 • DML查询：高频并发点查场景。

规则 2.11 创建表时必须选择正确的分区策略

说明

违反规范的影响：

分区的优点如下，如不做分区，其查询性能和数据治理效率会下降，数据量越大这种劣化越大。

- 改善查询性能：对分区对象的查询可以仅搜索业务关注的分区，提高检索效率。
- 提升数据治理效率：如数据生命周期管理场景，针对历史分区执行TRUNCATE/DROP PARTITION，效率和效果远优于DELETE。

方案建议：

- 针对包含时间类型字段的表设计分区。

表 3-4 分区策略选择

分区策略	描述	适用场景
范围分区（Range Partitioning）	根据分区键值的范围，将数据存储到不同的分区中，分区键范围连续但不重叠。	1. 日期或者时间类的字段作为分区键。 2. 查询中大多包含分区键作为过滤条件。 3. 定期按照分区键清理数据。
列表分区（List Partitioning）	根据分区键值的列表进行分区，各分区的列表值不重复	1. 特定数量的枚举值作为分区键值。 2. 查询中大多包含分区键作为过滤条件。

版权所有 © 华为云计算技术有限公司

建议 2.12 表字段的设计要遵循高效、准确原则

说明

违反规范的影响：

- 存储空间大、查询效率降低。

方案建议：

1. **选择最高效的类型**。
   - 如果能选择整型就不选择浮点型或字符型。
   - 使用变长字符类型时根据数据特征指定最大长度。
2. **选择最准确的类型**。
   - 使用时间类型存储时间，不使用字符类型存储时间。
   - 使用满足需求的最小数值类型，如果int或smallint够用，就不用bigint而浪费空间。
3. **正确使用约束**。
   - 明确不存在NULL值的字段加上NOT NULL约束，优化器会在特定场景下对其进行自动优化。
   - 业务层面能补全的字段，不要使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
4. **避免非必要类型转换**。
   - 当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。
   - 不同类型的比较操作会导致数据类型转换，可能导致索引和分区剪枝失效，影响查询性能。

建议 2.13 避免使用自增列或自增数据类型

说明

违反规范的影响：

- 自增序列或自增数据类型在大量使用时，会造成GTM压力过大及序列生成速度慢。

方案建议：

- 如果只是想获取一个唯一标识，可以使用UUID。
- 如果必须使用自增序列，在没有严格递增的需求，可设置CACHE，比如1000，降低GTM压力。

版权所有 © 华为云计算技术有限公司

3.3.6 INDEX 对象设计规范（重点）

规则 2.14 只创建必要的索引，创建索引必须选择合适的列和顺序

说明

违反规范的影响：

- 冗余索引浪费空间，索引多影响入库效率。
- 组合索引中列顺序错误，影响查询效率。

最佳实践：

索引的使用需兼顾以下条件：

- 索引列必须是常用于过滤条件或JOIN关联条件的列。
- 索引列必须是DISTINCT值多的列。
- 创建多列组合索引时，DISTINCT值多的列往前放。
- 单表索引个数控制在5个以内，可通过组合索引控制索引的个数。
- 数据批量增删改场景，建议先删除索引，完成增删改后再加回索引，提升批量操作性能（实时访问会有影响）。

建议 2.15 列存表通常可不建索引，极致性能场景需正确选择索引类型

说明

违反规范的影响：

- 错误的索引对列存的访问无任何性能帮助，反而可能影响查询性能。

方案建议：

1. 创建索引时指定索引类型，避免使用默认的PSORT类型索引。
2. 极端点查（海量数据中检索极少数据）场景，可使用B-Tree类型索引。
3. 范围查询性能要求高的场景，可以创建Partial Cluster Key（局部聚簇，简称PCK）通过min/max稀疏索引实现事实表快速过滤扫描。PCK的选取遵循以下原则：
   - 【关注】一张表上只能建立一个PCK，一个PCK可以包含多列，但是一般不建议超过2列。
   - 【建议】针对表达式过滤条件列创建PCK（形如col op const，op为操作符 =、>、>=、<=、<，const为常量值）。

3.3.7 VIEW 对象设计规范

建议 2.16 视图的嵌套需避免超过三层

说明

违反规范的影响：

- 视图嵌套过深导致执行计划不稳定、耗时不稳定。
- 视图依赖的对象重建风险高，锁冲突发生概率增大。

方案建议：

- 创建视图直接基于物理表查询，不建议嵌套视图。

3.4 DWS SQL 开发规范

版权所有 © 华为云计算技术有限公司

3.4.1 DDL 操作规范

建议 3.1 DDL 操作（CREATE 除外）避免在业务高峰期和长事务中执行

说明

违反规范的影响：

DDL操作普遍持锁级别高，如ALTER、DROP、TRUNCATE、REINDEX、VACUUM FULL等，执行时会造成业务等锁阻塞。

- 高峰期执行持锁级别高的DDL操作，造成业务等锁阻塞。
- 长事务中执行持锁级别高的DDL操作，长时间持锁或等锁，均造成业务等锁阻塞。

方案建议：

- 根据业务周期，选择低峰期或运维时间窗执行DDL操作，明确DDL执行环境和耗时，避免锁阻塞。

规则 3.2 DROP 删除对象操作必须明确删除对象范围

危险

违反规范的影响：

DROP对象操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除。

- DROP DATABASE：整个DATABASE中所有对象被删除。
- DROP USER：USER对象、USER所拥有的SCHEMA、TABLE等对象均被删除。
- DROP SCHEMA：整个SCHEMA中的所有对象被删除。
- DROP TABLE：TABLE对象，依赖TABLE的INDEX、VIEW等对象均被删除。

方案建议：

- DROP操作谨慎，操作前考虑数据备份。

3.4.2 INSERT 操作规范

规则 3.3 INSERT 多 VALUES 批插场景使用 COPY 替代

说明

违反规范的影响：

- 多VALUES解析耗时、耗资源，入库效率低。

方案建议：

- 前端使用COPY类接口（如JDBC的CopyManger等）代替INSERT VALUES。

版权所有 © 华为云计算技术有限公司

建议 3.4 禁止针对普通列存表进行实时 INSERT 操作

说明

违反规范的影响：

- 针对普通列存表实时小批量入库会导致小CU膨胀严重，影响存储空间和查询性能。

方案建议：

- 实时INSERT场景评估单次入库数据量和数据总量，总量小的场景可以改为行存表。
- 实时INSERT场景前端攒批，保证单次、单表、单分区、单DN入库数据量接近6W，建议最低不少于5K。
- 实时INSERT场景使用Hstore列存表（8.3.0及以上集群版本）。

3.4.3 UPDATE&DELETE 操作规范

建议 3.5 避免并发 UPDATE/DELETE 行存表的同一行

说明

违反规范的影响：

- 并发UPDATE/DELETE行存表可能导致行锁阻塞和分布式死锁风险，导致业务报错和性能下降。

方案建议：

- 按主键或分布列进行分组UPDATE/DELETE，组间并行，组内串行。

建议 3.6 避免对列存表频繁或并发执行 UPDATE/DELETE

说明

违反规范的影响：

- 针对列存表频繁执行UPDATE/DELETE造成CU膨胀，导致空间膨胀和访问性能下降。
- 针对列存表并发执行UPDATE/DELETE，导致行锁阻塞和分布式死锁风险，导致业务报错和性能下降。

方案建议：

- 频繁执行UPDATE/DELETE操作的表需设计为行存表。
- 按主键或分布列进行分组UPDATE/DELETE，组间并行，组内串行。

3.4.4 SELECT 操作规范

规则 3.7 禁止执行不下推的 SQL

说明

DWS为分布式架构，SQL语句必须下推才能分利用分布式的计算资源，达到性能最优。

违反规范的影响：

- 不下推的SQL执行性能差，严重情况会导致CN资源瓶颈，影响整体业务。

方案建议：

- 不使用不下推的语法和不下推的函数，具体参考**语句下推调优**。

版权所有 © 华为云计算技术有限公司

规则 3.8 禁止多表关联时缺少关联条件

说明

违反规范的影响：

- 多表关联时不指定关联条件就是求笛卡尔积，极易导致结果集膨胀，造成性能和资源过载风险。

方案建议：

- 多表关联时明确每张表的过滤条件和关联条件，避免出现缺少过滤条件和关联条件的情况。

规则 3.9 多表关联字段数据类型要保持一致

说明

违反规范的影响：

- 关联字段类型不一致，导致额外的类型转换开销，且影响数据重分布的策略，无法生成最优计划。

方案建议：

- 多表关联场景，关联的字段均使用同样的数据类型。

建议 3.10 尽量避免对关联条件字段和过滤条件字段进行函数运算

说明

违反规范的影响：

- 对关联条件字段和过滤条件字段进行函数运算，导致优化器无法获取准确的字段统计信息，无法生成最优计划，影响执行性能。

方案建议：

- 关联条件字段之间直接比较，如有需要运算后比较的场景需在数据入库前进行预处理。
- 过滤条件和常量比较时，只对常量列进行函数运算，字段列不进行函数运算，例如：

```
SELECT id, from_image_id, from_person_id, from_video_id
FROM face_data
WHERE SS.DEL_FLAG = 'N'
AND NVL(SS.DELETE_FLAG, 'N') = 'N'
改写为：
SELECT id, from_image_id, from_person_id, from_video_id
FROM face_data
where SS.DEL_FLAG = 'N'
AND (SS.DELETE_FLAG = 'N' or SS.DELETE_FLAG is null)
```

建议 3.11 资源高消耗型 SQL 需做好压力测试和并发管控

说明

违反规范的影响：

- 存储和计算资源过载，整体运行性能下降。

方案建议：

高资源消耗型SQL的主要特征：

- 大量UNION ALL
- 大量AGG（ COUNT DISTINCT、MAX等 ）
- 大量表JOIN
- 大量STREAM算子（ 计划维度 ）

针对上述SQL需进行压力测试和并发管控，如果超出资源能力，则必须进行业务优化后再重新评估上线。

版权所有 © 华为云计算技术有限公司

规则 3.12 禁止针对行存大表的频繁 COUNT

说明

磁盘能力强的场景（如SSD），本规则可适当放宽，但仍需关注I/O消耗情况。

违反规范的影响：

- 行存表的COUNT需要扫描全表，大表场景频繁COUNT会消耗大量I/O，如触发I/O瓶颈会导致整体性能问题。

方案建议：

- 建议降低COUNT频率、使用结果缓存、分区级统计等方式，降低COUNT的I/O消耗。

建议 3.13 避免查询返回超大结果集（数据导出场景除外）

说明

违反规范的影响：

- 在实际不需要查看所有结果的场景，查询超大结果集会浪费大量资源。

方案建议：

- 查询使用LIMIT，只返回必要数量的结果。
- 真实需要查询大量结果集的场景，使用游标进行分段获取，合理设置FETCH SIZE。

建议 3.14 查询时避免使用“SELECT *”写法

说明

违反规范的影响：

- 查询实际不需要的列，增加计算负担，浪费计算资源。

方案建议：

- SELECT时明确列出查询所需字段，提升查询的有效性能。

建议 3.15 谨慎使用递归语句(WITH RECURSIVE)，明确终止条件，确保递归可终止

说明

违反规范的影响：

- 无明确终止条件，递归陷入死循环，无法完成。
- 重复数据过多，递归产生大量重复数据，占用大量资源。

方案建议：

- 根据业务表数据量和数据特征设计合理的递归终止条件。

建议 3.16 访问对象（表，函数等）时带上 SCHEMA 名称

说明

违反规范的影响：

- 不指定SCHEMA名称前缀，实际会根据当前search_path中表空间列表，依次搜索所有表空间直到找到匹配的表作为目标表，可能因SCHEMA切换导致访问到非预期的表。

方案建议：

- 访问表和函数对象时显式指定“SCHEMA.”前缀，增强可读性、稳定性、可移植性。

版权所有 © 华为云计算技术有限公司

建议 3.17 针对 SQL 标记注释，唯一标识 SQL 的归属

说明

违反规范的影响：

- 业务溯源能力较差，只能通过数据库、用户名、客户端IP信息找开发人员确认。

方案建议：

- 建议使用query_band，例如：
  ```
  SET query_band='JobName=abc;AppName=test;UserName=user';
  ```
- 每个SQL开头标记注释，唯一标识SQL的归属，方便问题定位及应用性能分析，命名建议为：
  /* 模块名_工具名_作业名_步骤 */，如: /* mca_python_xxxxxx_step1 */ insert into xxx select … from

建议 3.18 SQL 语句的长度不要超过 64K

说明

违反规范的影响：

- SQL解析耗时长、维护难度大，如频繁执行会导致日志膨胀严重。

方案建议：

- 控制业务SQL的长度，避免超过64K。

3.5 DWS 外表功能开发规范

规则 4.1 GDS 服务需单独使用服务器部署在 DWS 集群外

说明

违反规范的影响：

GDS如果部署在DWS集群内，会与DWS集群CN/DN节点发生资源争抢，导致双方性能同时劣化。

方案建议：

- GDS服务单独使用服务器部署在DWS集群外。
- GDS所在服务器的磁盘能力、GDS服务器与DWS集群间网络带宽都要按需规划。

规则 4.2 避免同时对多个协同分析外表进行跨集群并发访问

说明

原理说明：在A集群通过协同分析访问B集群数据时，A集群所有DN会与B集群CN建立连接和活跃会话。

违反规范的影响：

B集群（远端集群）中CN压力过大，导致连接和活跃会话资源超限，访问异常。

方案建议：

应尽量使用外表单表访问并避免并发，避免多外表关联查询；无法避免并发场景时，并发数需根据A集群DN数及B集群常规业务量进行计算和限制，并适当调大max_active_statements和max_connections值。

版权所有 © 华为云计算技术有限公司

3.6 DWS 存储过程开发规范

建议 5.1 避免使用复杂的存储过程，避免存储过程嵌套

说明

违反规范的影响：

- 复杂和嵌套的存储过程维护成本高，故障定位难度大，恢复耗时长。

方案建议：

- 不使用存储过程或只使用一层存储过程，不嵌套。
- 开发存储过程设计对应的日志表，将关键步骤前后的信息记录到日志表中，操作步骤如下。

保存并查看日志操作步骤。

步骤1 创建日志表。

```
CREATE TABLE func_exec_log
(
id varchar2(32) default lower(sys_guid()),
pro_name varchar2(60),
exec_times int,
log_date date,
deal_date date,
log_message text
);
```

步骤2 创建表和导入数据。

```
CREATE TABLE demo_table(data_id int, data_number int);
INSERT INTO demo_table values(generate_series(1,1000),generate_series(1,1000));
```

步骤3 创建业务存储过程。

```
CREATE OR REPLACE FUNCTION demo_table_process(out exe_info text)
LANGUAGE plpgsql
AS $$
declare v_count int;
pro_result text;
fun_name   text;
exec_times   int;
begin
fun_name := 'demo_table_process';
select nvl(max(exec_times), '0') + 1 into exec_times from func_exec_log where pro_name = fun_name;
--业务表插入数据
insert into demo_table values (dbms_random.value(1, 1000)::int,generate_series(1,
dbms_random.value(10000, 20000)::int));
get diagnostics v_count = ROW_COUNT;
exe_info = sysdate || '# step1:insert count:' || v_count || ' rows;';
--删除业务表指定数据
delete from demo_table where data_id = dbms_random.value(1, 1000)::int;
get diagnostics v_count = ROW_COUNT;
exe_info = exe_info || sysdate || '# step2:delete count:' || v_count || ' rows;';
--更新业务表数据
update demo_table set data_number = dbms_random.value(1, 100)::int where data_id =
dbms_random.value(1, 1000)::int;
exe_info = exe_info || sysdate || '# step3:update count:' || sql%rowcount || ' rows';
--在整个程序结束前记录日志，也可以在每个步骤结束后分别记录日志，也可以创建记录日志的函数供调用，灵活使用即可
insert into func_exec_log(pro_name, exec_times, log_date, deal_date, log_message) values
(fun_name,exec_times,sysdate,split_part(regexp_split_to_table(exe_info, ';'), '#',
1),split_part(regexp_split_to_table(exe_info, ';'), '#', 2));
--EXCEPTION用于保证当insert/update/delete等操作步骤异常退出时，也能够正常记录日志
EXCEPTION
WHEN OTHERS THEN
```


版权所有 © 华为云计算技术有限公司

```
pro_result := exe_info || sysdate || '# exception error message is: ' || sqlerrm;
insert into func_exec_log(pro_name, exec_times, log_date, deal_date, log_message)
values(fun_name,exec_times,sysdate,split_part(regexp_split_to_table(pro_result, ';'), '#',
1),split_part(regexp_split_to_table(pro_result, ';'), '#', 2));
END; $$;
```

步骤4 调用存储过程（正常执行）。

```
SELECT demo_table_process();
```

步骤5 查看日志（确认业务运行情况）。

```
SELECT * FROM func_exec_log ORDER BY log_date desc,deal_date,log_message;
```

```
demodb=# select * from func_exec_log order by log_date desc,deal_date,log_mesage;
               id                |      pro_name      | exec_times |      log_date       |      deal_date      |           log_mesage
---------------------------------+--------------------+------------+---------------------+---------------------+--------------------------------
 637343d9f2f10ec605c7687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:11 |  step1:insert count:19125 rows
 637343d9fe850e3105c8687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:33 |  step2:delete count:22 rows
 637343d9068a0fd805c9687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:34 |  step3:update count:15 rows
(3 rows)
```

步骤6 再次调用存储过程（构造执行异常）。

```
SELECT demo_table_process();  --先删除demo_table的data_number列构造异常，之后再调用
```

步骤7 查看日志（确认业务运行情况）。

----结束

规则 5.2 存储过程内避免执行非 CREATE 类的 DDL 操作

说明

违反规范的影响：

- 存储过程是一个大事务，如果有非CREATE类DDL操作（持锁级别高），整个存储过程执行时间窗内会阻塞外部对相关表的访问。

方案建议：

- 存储过程中不使用非CREATE类的DDL操作，如必须使用，严格评估存储过程耗时和DDL的影响耗时，与外部访问业务错峰执行。

3.7 DWS 各对象设计详细规则

3.7.1 DWS 数据库对象命名规则

数据库对象命名需要满足约束：长度不超过63个字符，以字母或下划线开头，中间字符可以是字母、数字、下划线、$。数据库字符集为GBK、UTF8、SQL_ASCII时对象命名支持中文汉字，其中UTF8、SQL_ASCII字符集下，一个中文汉字占3个字符，最多支持21个中文汉字；GBK字符集下，一个中文汉字占2个字符，最多支持31个中文汉字。Latin1字符集不支持中文汉字命名。字符集格式在创建数据库时指定，详情参见《数据仓库服务SQL语法参考》中“CREATE DATABASE”章节。

- 【建议】避免使用保留或者非保留关键字命名数据库对象。

  说明

  可以使用SELECT * FROM pg_get_keywords()查询DWS的关键字，或者参考《数据仓库服务SQL语法参考》中“关键字”章节。

- 【建议】避免使用双引号括起来的字符串来定义数据库对象名称，DWS中使用双引号将数据库对象名称括起来时表示对大小写敏感。数据库对象名称大小写敏感会使定位问题难度增加。

版权所有 © 华为云计算技术有限公司

- 【建议】数据库对象命名风格务必保持统一。
  - 增量开发的业务系统或进行业务迁移的系统，建议遵守历史的命名风格。
  - 数据库对象名称由字母、数字和下划线组成，并且不能由数字开头。建议使用多个单词组成，以下划线分割。
  - 数据库对象名称最好能够望文知意，尽量避免使用自定义缩写（可以使用通用的术语缩写进行命名）。例如，在命名中可以使用具有实际业务含义的英文词汇或汉语拼音，但规则应该在集群范围内保持一致。
  - 变量名的关键是要具有描述性，即变量名称要有一定的意义，变量名要有前缀标明该变量的类型。
- 【建议】表对象的命名应该可以表征该表的重要特征。例如，在表对象命名时区分该表是普通表、临时表还是非日志表：
  - 普通表名按照数据集的业务含义命名。
  - 临时表以“tmp_+后缀”命名。
  - 非日志表以“ul_+后缀”命名。
  - 外表以“f_+后缀”命名。

3.7.2 DWS 数据库对象设计规则

3.7.2.1 DWS Database 和 Schema 设计规则

DWS中可以使用Database和Schema实现业务的隔离，区别在于Database的隔离更加彻底，各个Database之间共享资源极少，可实现连接隔离、权限隔离等，Database之间无法直接互访。Schema隔离的方式共用资源较多，可以通过GRANT与REVOKE语法便捷地控制不同用户对各Schema及其下属对象的权限。

- 从便捷性和资源共享效率上考虑，推荐使用Schema进行业务隔离。
- 建议系统管理员创建Schema和Database，再赋予相关用户对应的权限。

Database 设计建议

- 【建议】在实际业务中，根据需要创建新的Database，不建议直接使用集群默认的gaussdb数据库。
- 【建议】一个集群内，用户自定义的Database数量建议不超过3个。
- 【建议】为了适应全球化的需求，使数据库编码能够存储与表示绝大多数的字符，建议创建Database的时候使用UTF-8编码。
- 【关注】创建Database时，需要重点关注字符集编码(ENCODING)和兼容性(DBCOMPATIBILITY)两个配置项。DWS支持Oracle、Teradata和MySQL三种兼容模式，分别兼容Oracle、Teradata和MySQL语法，不同兼容模式下的语法行为可能有一些差异。详细内容可参考**Oracle、Teradata和MySQL语法兼容性差异**。
- 【关注】Database的owner默认拥有该Database下所有对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

Schema 设计建议

- 【关注】如果该用户不具有sysadmin权限或者不是该Schema的owner，要访问Schema下的对象，需要同时给用户赋予Schema的usage权限和对象的相应权限。

版权所有 © 华为云计算技术有限公司

- 【关注】如果要在Schema下创建对象，需要授予操作用户该Schema的CREATE权限。
- 【关注】Schema的owner默认拥有该Schema下对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

3.7.2.2 DWS 表设计规则

DWS是分布式架构。数据分布在各个DN上。总体上讲，良好的表设计需要遵循以下原则：

- 【关注】将表数据均匀分布在各个DN上。数据均匀分布，可以防止数据在部分DN上集中分布，从而导致因存储倾斜造成集群有效容量下降。通过选择合适的分布列，可以避免数据倾斜。
- 【关注】将表的扫描压力均匀分散在各个DN上。避免扫描压力集中在部分DN上，而导致性能瓶颈。例如，在事实表上使用等值过滤条件时，将会导致扫描压力不均匀。
- 【关注】减少需要扫描的数据量。通过分区表的剪枝机制可以大幅减少数据的扫描量。
- 【关注】尽量减少随机I/O。通过聚簇/局部聚簇可以实现热数据的连续存储，将随机I/O转换为连续I/O，从而减少扫描的I/O代价。
- 【关注】尽量避免数据shuffle。shuffle是指在物理上，数据从一个节点传输到另一个节点。shuffle占用了大量宝贵的网络资源，减小不必要的数据shuffle，可以减少网络压力，使数据的处理本地化，提高集群的性能和可支持的并发度。通过对关联条件和分组条件的仔细设计，能够尽可能的减少不必要的数据shuffle。

选择存储方案

【建议】表的存储类型是表定义设计的第一步，用户业务类型是决定表的存储类型的主要因素，表存储类型的选择依据请参考**表3-5**。

表 3-5 表的存储类型及场景

存储类型	适用场景
行存	• 点查询(返回记录少，基于索引的简单查询)。 • 增、删、改操作较多的场景。
列存	• 统计分析类查询 (关联、分组操作较多的场景)。 • 即席查询（查询条件不确定，行存表扫描难以使用索引）。

对于分析场景，建表需显式设置ORIENTATION 选项为列存。

```
CREATE TABLE  public.t1
(
id integer not null,
data integer,
age integer
)
WITH (ORIENTATION =COLUMN);
```


版权所有 © 华为云计算技术有限公司

选择分布方案

【建议】表的分布方式的选择一般遵循以下原则：

表 3-6 表的分布方式及使用场景

分布方式	描述	适用场景
Hash	表数据通过Hash方式散列到集群中的所有DN上。	数据量较大的事实表。
Replication	集群中每一个DN都有一份全量表数据。	维度表、数据量较小的事实表。
Roundrobin	表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。	数据量较大的事实表，且使用Hash分布时找不到合适的分布列。

选择分区方案

当表中的数据量很大时，应当对表进行分区，一般需要遵循以下原则：

- 【建议】使用具有明显区间性的字段进行分区，比如日期、区域等字段上建立分区。
- 【建议】分区名称应当体现分区的数据特征。例如，关键字+区间特征。
- 【建议】将分区上边界的分区值定义为MAXVALUE，以防止可能出现的数据溢出。

典型的分区表定义如下：

```
CREATE TABLE staffS_p1
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(4,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
PARTITION BY RANGE (HIRE_DATE)
(
   PARTITION HIRE_19950501 VALUES LESS THAN ('1995-05-01 00:00:00'),
   PARTITION HIRE_19950502 VALUES LESS THAN ('1995-05-02 00:00:00'),
   PARTITION HIRE_maxvalue VALUES LESS THAN (MAXVALUE)
);
```

选择分布键

Hash表的分布键选取至关重要，如果分布键选择不当，可能会导致数据倾斜，从而导致查询时，I/O负载集中在部分DN上，影响整体查询性能。因此，在确定Hash表的分布策略之后，需要对表数据进行倾斜性检查，以确保数据的均匀分布。分布键的选择一般需要遵循以下原则：

版权所有 © 华为云计算技术有限公司

- 【建议】选作分布键的字段取值应该比较离散，以便数据能在各个DN上均匀分布。当单个字段无法满足离散条件时，可以考虑使用多个字段一起作为分布键。一般情况下，可以考虑选择表的主键作为分布键。例如，在人员信息表中选择证件号码作为分布键。
- 【建议】在满足第一条原则的情况下，尽量不要选取在查询中存在常量过滤条件的字段作为分布键。例如，在表dwcjk相关的查询中，字段zqdh存在常量过滤条件“zqdh='000001'”，那么就应当尽量不选择zqdh字段作为分布键。
- 【建议】在满足前两条原则的情况，尽量选择查询中的关联条件为分布键。当关联条件作为分布键时，Join任务的相关数据都分布在DN本地，将极大减少DN之间的数据流动代价。

3.7.2.3 DWS 字段设计规则

选择数据类型

在字段设计时，基于查询效率的考虑，一般遵循以下原则：

- 【建议】尽量使用高效数据类型。

  选择数值类型时，在满足业务精度的情况下，选择数据类型的优先级从高到低依次为整数、浮点数、NUMERIC。
- 【建议】当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。
- 【建议】对于字符串数据，建议使用变长字符串数据类型，并指定最大长度。请务必确保指定的最大长度大于需要存储的最大字符数，避免超出最大长度时出现字符截断现象。除非明确知道数据类型为固定长度字符串，否则，不建议使用CHAR(n)、BPCHAR(n)、NCHAR(n)、CHARACTER(n)。

  关于字符串类型的详细说明，请参见**常用字符串类型介绍**。

常用字符串类型介绍

在进行字段设计时，需要根据数据特征选择相应的数据类型。字符串类型在使用时比较容易混淆，下表列出了DWS中常见的字符串类型：

表 3-7 常用字符串类型

名称	描述	最大存储空间
CHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
CHARACTER(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
NCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB

版权所有 © 华为云计算技术有限公司

名称	描述	最大存储空间
BPCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
VARCHAR(n)	变长字符串，n描述了可以存储的最大字节长度。	10MB
CHARACTER VARYING(n)	变长字符串，n描述了可以存储的最大字节长度；此数据类型和VARCHAR(n)是同一数据类型的不同表达形式。	10MB
VARCHAR2(n)	变长字符串，n描述了可以存储的最大字节长度，此数据类型是为兼容Oracle类型新增的，行为和VARCHAR(n)一致。	10MB
NVARCHAR2(n)	变长字符串，n描述了可以存储的最大字符长度。	10MB
TEXT	不限长度(不超过1GB-8203字节)变长字符串。	1GB-8203字节

3.7.2.4 DWS 约束设计规则

DEFAULT 和 NULL 约束

- 【建议】如果能够从业务层面补全字段值，则不建议使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
- 【建议】给明确不存在NULL值的字段加上NOT NULL约束，优化器会在特定场景下对其进行自动优化。
- 【建议】给可以显式命名的约束显式命名。除了NOT NULL和DEFAULT约束外，其他约束都可以显式命名。

局部聚簇

Partial Cluster Key（局部聚簇，简称PCK）是列存表的一种局部聚簇技术，在DWS中，使用PCK可以通过min/max稀疏索引实现事实表快速过滤扫描。PCK的选取遵循以下原则：

- 【关注】一张表上只能建立一个PCK，一个PCK可以包含多列，但是一般不建议超过2列。
- 【建议】在查询中的简单表达式过滤条件上创建PCK。这种过滤条件一般形如col op const，其中col为列名，op为操作符 =、>、>=、<=、<，const为常量值。
- 【建议】在满足上面条件的前提下，选择在distinct值比较少的列上建PCK。

版权所有 © 华为云计算技术有限公司

唯一约束

- 【关注】行存表与列存表都支持唯一约束。
- 【建议】从命名上明确标识唯一约束，例如，命名为“UNI+构成字段”。

主键约束

- 【关注】行存表与列存表都支持主键约束。
- 【建议】从命名上明确标识主键约束，例如，将主键约束命名为 “PK+字段名”。

检查约束

- 【关注】行存表支持检查约束，而列存表不支持。
- 【建议】从命名上明确标识检查约束，例如，将检查约束命名为 “CK+字段名”。

3.7.2.5 DWS 视图和关联表设计规则

视图设计

- 【建议】除非视图之间存在强依赖关系，否则不建议视图嵌套。
- 【建议】视图定义中尽量避免排序操作。

关联表设计

- 【建议】表之间的关联字段应该尽量少。
- 【建议】关联字段的数据类型应该保持一致。
- 【建议】关联字段在命名上，尽可能体现出明显的关联关系。例如，采用同样名称来命名。

3.7.3 DWS JDBC 配置规则

目前，DWS相关的第三方工具可以通过JDBC进行连接的，此部分将介绍工具配置时的注意事项。

连接参数

- 【关注】第三方工具通过JDBC连接DWS时，JDBC向DWS发起连接请求，会默认添加以下配置参数，详见JDBC代码ConnectionFactoryImpl类的实现。

```
params = {
{ "user", user },
{ "database", database },
{ "client_encoding", "UTF8" },
{ "DateStyle", "ISO" },
{ "extra_float_digits", "2" },
{ "TimeZone",  createPostgresTimeZone() },
};
```

  这些参数可能会导致JDBC客户端的行为与gsql客户端的行为不一致，例如，Date数据显示方式、浮点数精度表示、timezone显示。

  如果实际期望和这些配置不符，建议在java连接设置代码中显式设定这些参数。

- 【建议】通过JDBC连接数据库时，应该保证下面两个时区设置一致：

版权所有 © 华为云计算技术有限公司

- JDBC客户端所在主机的时区。
- DWS集群所在主机的时区。

fetchsize

【关注】在应用程序中，如果需要使用fetchsize，必须关闭autocommit。开启autocommit，会使fetchsize配置失效。

autocommit

【建议】在JDBC向DWS申请连接的代码中，建议显式打开autocommit开关。如果基于性能或者其它方面考虑，需要关闭autocommit时，需要应用程序自己来保证事务的提交。例如，在指定的业务SQL执行完之后做显式提交，特别是客户端退出之前务必保证所有的事务已经提交。

释放连接

【建议】推荐使用连接池限制应用程序的连接数。每执行一条SQL就连接一次数据库，是一种不好SQL的编写习惯。

【建议】在应用程序完成作业任务之后，应当及时断开和DWS的连接，释放资源。建议在任务中设置session超时时间参数。

【建议】使用JDBC连接池，在将连接释放给连接池前，需要执行以下操作重置会话环境。否则，可能会因为历史会话信息导致的对象冲突。

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须执行“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

CopyManager

【建议】在不使用ETL工具、数据入库实时性要求又比较高的情况下，建议在开发应用程序时，使用DWS JDBC驱动的CopyManager接口进行微批量导入。

CopyManager的使用方法请参见**CopyManager**。

3.7.4 DWS SQL 编写规则

DDL

- 【建议】在DWS中，建议DDL（建表、comments等）操作统一执行，在批处理作业中尽量避免DDL操作。避免大量并发事务对性能的影响。
- 【建议】在非日志表（unlogged table）使用完后，立即执行数据清理（truncate）操作。因为在异常场景下，DWS不保证非日志表(unlogged table)数据的安全性。
- 【建议】临时表和非日志表的存储方式建议和基表相同。当基表为行存（列存）表时，临时表和非日志表也推荐创建为行存（列存）表，可以避免行列混合关联带来的高计算代价。
- 【建议】索引字段的总长度不超过50字节。否则，索引大小会膨胀比较严重，带来较大的存储开销，同时索引性能也会下降。
- 【建议】不要使用DROP…CASCADE方式删除对象，除非已经明确对象间的依赖关系，以免误删。

版权所有 © 华为云计算技术有限公司

数据加载和卸载

- 【建议】在insert语句中显式给出插入的字段列表。例如：
```
INSERT INTO task(name,id,comment) VALUES ('task1','100','第100个任务');
```
- 【建议】在批量数据入库之后，或者数据增量达到一定阈值后，建议对表进行analyze操作，防止统计信息不准确而导致的执行计划劣化。
- 【建议】如果要清理表中的所有数据，建议使用truncate table方式，不要使用delete table方式。delete table方式删除性能差，且不会释放那些已经删除了的数据占用的磁盘空间。

类型转换

- 【建议】在需要数据类型转换（不同数据类型进行比较或转换）时，使用强制类型转换，以防隐式类型转换结果与预期不符。
- 【建议】在查询中，对常量要显式指定数据类型，不要试图依赖任何隐式的数据类型转换。
- 【关注】在ORACLE兼容模式下，在导入数据时，空字符串会自动转化为NULL。如果需要保留空字符串需要新建兼容性为TD的数据库。

查询操作

- 【建议】除ETL程序外，应该尽量避免向客户端返回大量结果集的操作。如果结果集过大，应考虑业务设计是否合理。
- 【建议】使用事务方式执行DDL和DML操作。例如，truncate table、update table、delete table、drop table等操作，一旦执行提交就无法恢复。对于这类操作，建议使用事务进行封装，必要时可以进行回滚。
- 【建议】在查询编写时，建议明确列出查询涉及的所有字段，不建议使用“SELECT *”这种写法。一方面基于性能考虑，尽量减少查询输出列；另一方面避免增删字段对前端业务兼容性的影响。
- 【建议】在访问表对象时带上schema前缀，可以避免因schema切换导致访问到非预期的表。
- 【建议】超过8张表或视图进行关联（特别是full join）时，执行代价难以估算。建议使用WITH TABLE AS语句或者其它方式创建中间临时表的方式增加SQL语句的可读性。
- 【建议】尽量避免使用笛卡尔积和Full join。这些操作会造成结果集的急剧膨胀，同时其执行性能也很低。
- 【关注】NULL值的比较只能使用IS NULL或者IS NOT NULL的方式判断，其他任何形式的逻辑判断都返回NULL。例如：NULL<>NULL、NULL=NULL和NULL<>1返回结果都是NULL，而不是期望的布尔值。
- 【关注】需要统计表中所有记录数时，不要使用count(col)来替代count(*)。count(*)会统计NULL值（真实行数），而count(col)不会统计。
- 【关注】在执行count(col)时，将“值为NULL”的记录行计数为0。在执行sum(col)时，当所有记录都为NULL时，最终将返回NULL；当不全为NULL时，“值为NULL”的记录行将被计数为0。
- 【关注】count(多个字段)时，多个字段名必须用圆括号括起来。例如，count((col1,col2,col3))。注意：通过多字段统计行数时，即使所选字段都为NULL，该行也被计数，效果与count(*)一致。
- 【关注】count(distinct col)用来计算该列不重复的非NULL的数量，NULL将不被计数。

版权所有 © 华为云计算技术有限公司

- 【关注】count(distinct (col1,col2,...))用来统计多列的唯一值数量，当所有统计字段都为NULL时，也会被计数，同时这些记录被认为是相同的。
- 【关注】通过常量来过滤数据时，会根据常量的数据类型和匹配列的数据类型来查找用于这两种数据类型计算的函数，如果找不到对应的函数，则会相应的进行隐式数据类型转化，然后再根据转化后的数据类型查找用于转化后的数据类型计算的函数。

  ```
  SELECT * FROM test WHERE timestamp_col = 20000101;
  ```

  上述例子中，假设timestamp_col是timestamp类型，则会先查找支持timestamp类型和int类型（常量数字认为是int类型）“等于”运算的函数，如果找不到，则把timestamp_col和常量数字隐式类型转化成text类型来计算。
- 【建议】尽量避免标量子查询语句的出现。标量子查询是出现在select语句输出列表中的子查询，在下面例子中，括号内部分即为一个标量子查询语句：

  ```
  SELECT id, (SELECT COUNT(*) FROM films f WHERE f.did = s.id) FROM staffs_p1 s;
  ```

  标量子查询时常会导致查询性能急剧劣化，在应用开发过程中，应当根据业务逻辑，对标量子查询进行等价转换，将其写为表关联。
- 【建议】在where子句中，应当对过滤条件进行排序，把选择读较小（筛选出的记录数较少）的条件排在前面。
- 【建议】where子句中的过滤条件，尽量符合单边规则。即把字段名放在比较条件的一边，优化器在某些场景下会自动进行剪枝优化。形如col op expression，其中col为表的一个列，op为‘=’、‘>’的等比较操作符，expression为不含列名的表达式。例如，

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data WHERE
  current_timestamp(6) - time < '1 days'::interval;
  ```

  改写为：

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data where time >
  current_timestamp(6) - '1 days'::interval;
  ```
- 【建议】尽量避免不必要的排序操作。排序需要耗费大量的内存及CPU，如果业务逻辑许可，可以组合使用order by和limit，减小资源开销。DWS默认按照ASC & NULL LAST进行排序。
- 【建议】使用ORDER BY子句进行排序时，显式指定排序方式（ASC/DESC），NULL的排序方式（NULL FIRST/NULL LAST）。
- 【建议】不要单独依赖limit子句返回特定顺序的结果集。如果部分特定结果集，可以将ORDER BY子句与Limit子句组合使用，必要时也可以使用offset跳过特定结果。
- 【建议】在保障业务逻辑准确的情况下，建议尽量使用UNION ALL来代替UNION。
- 【建议】如果过滤条件只有OR表达式，可以将OR表达式转化为UNION ALL以提升性能。使用OR的SQL语句经常无法优化，导致执行速度慢。例如，将下面语句

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301) OR (cdp= 301 AND inline=302) OR (cdp= 302 AND inline=301);
  ```

  转换为：

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301)
  union all
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 301 AND inline=302)
  union all
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 302 AND inline=301);
  ```
- 【建议】当in(val1, val2, val3…)表达式中字段较多时，建议使用in (values (val1), (val2),(val3)…)语句进行替换。优化器会自动把in约束转换为非关联子查询，从而提升查询性能。

版权所有 © 华为云计算技术有限公司

- 【建议】在关联字段不存在NULL值的情况下，使用(not) exist代替(not) in。例如，在下面查询语句中，当T1.C1列不存在NULL值时，可以先为T1.C1字段添加NOT NULL约束，再进行如下改写。

```
SELECT * FROM T1 WHERE T1.C1 NOT IN (SELECT T2.C2 FROM T2);
```

可以改写为：

```
SELECT * FROM T1 WHERE NOT EXISTS (SELECT  * FROM T1,T2 WHERE T1.C1=T2.C2);
```

说明

- 如果不能保证T1.C1列的值为NOT NULL的情况下，就不能进行上述改写。
- 如果T1.C1为子查询的输出，要根据业务逻辑确认其输出是否为NOT NULL。

- 【建议】通过游标进行翻页查询，而不是使用LIMIT OFFSET语法，避免多次执行带来的资源开销。游标必须在事务中使用，执行完后务必关闭游标并提交事务。

3.7.5 自定义 DWS 外部函数(pgSQL/Java)使用规则

- 【关注】Java UDF可以实现一些java逻辑计算，禁止在Java UDF中封装业务。
- 【关注】禁止在Java函数中使用任何方式连接数据库，包括但不限于JDBC。
- 【关注】只能选择下表中的数据类型，不支持自定义类型、复杂数据类型（Java Array类及派生类）等：
- 【关注】不支持UDAF（用户定义聚合函数）、UDTF（用户自定义表生成函数）。

表 3-8 PL/Java 默认数据类型映射关系

DWS	Java
BOOLEAN	boolean
"char"	byte
bytea	byte[]
SMALLINT	short
INTEGER	int
BIGINT	long
FLOAT4	float
FLOAT8	double
CHAR	java.lang.String
VARCHAR	java.lang.String
TEXT	java.lang.String
name	java.lang.String
DATE	java.sql.Timestamp
TIME	java.sql.Time (stored value treated as local time)

版权所有 © 华为云计算技术有限公司

DWS	Java
TIMETZ	java.sql.Time
TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	java.sql.Timestamp

3.7.6 DWS PL/pgSQL 使用规则

总体开发原则

1. 应完全按照设计文档进行开发。
2. 程序模块应做到高内聚低耦合。
3. 应有正确、全面的故障对策。
4. 程序编写应做到结构合理，条理清晰。
5. 程序名称命名应按照统一的命名规则进行命名。
6. 应充分考虑程序的运行效率，包括程序的执行效率和数据库的查询、存储效率，在保证应用的同时应使用效率高的处理方法。
7. 程序注释应详细、正确、规范。
8. 除非应用特别需要控制commit和rollback的提交时机，否则应在存储过程结束时执行显式的commit或者rollback操作。
9. 程序处理应支持7*24小时；对于中断，应用程序应提供安全、简单的断点再续处理。
10. 应提供标准、简单的应用输出，为应用维护人员提供明确的进度显示、错误描述和运行结果；为业务人员提供明确、直观的报表、凭证输出。

程序编写原则

1. 在PL/PGSQL中的SQL语句宜使用绑定变量。
2. 在PL/PGSQL中的SQL语句宜使用RETURNING子句。
3. 存储过程使用原则：
   a. 对于单个存储过程中Varchar或者Varchar2类型输出参数个数不应超过50个。
   b. 不应使用long类型作为输入或输出参数。
   c. 对于大小超过10MB的字符串类型输出，应使用CLOB类型。
4. 变量声明原则：
   a. 变量声明时，如果含义和应用表某字段含义或某变量相同时，应使用%TYPE声明。
   b. 记录声明时，如果含义和某应用表行数据相同时，应使用%ROWTYPE声明。
   c. 变量声明每行应只包含一条语句。
   d. 不应声明LONG类型的变量。
5. 游标使用类型：
   a. 显式游标使用后应关闭。

版权所有 © 华为云计算技术有限公司

b. 游标变量使用后应关闭，若游标变量需要传递数据给调用的应用程序，应在应用程序中进行游标关闭处理；若游标变量仅在存储过程中使用，应显式关闭游标。

c. 在使用DBMS_SQL.CLOSE_CURSOR关闭游标前，应使用DBMS_SQL.IS_OPEN判断游标是否已打开。

6. 集合使用原则：

a. 引用集合中的元素时宜使用FORALL语句，不宜使用FOR循环语句。

7. 动态语句使用原则：

a. 联机系统的交易程序不宜使用动态SQL。

b. PL/PGSQL中要实现DDL语句和系统控制命令，可使用动态SQL。

c. 宜尽量使用变量绑定。

8. 拼装SQL的使用原则：

a. 拼装SQL宜使用绑定变量。

b. 拼装SQL语句的条件如果有外部输入源，应对输入条件进行字符检查，防止攻击。

c. 在PL/PGSQL脚本中，单行代码的长度，不宜超过2499字符。

9. Trigger使用原则：

a. Trigger可用于实现差量数据日志等与业务处理无关的可用性设计场景。

b. 不应使用Trigger实现业务处理相关功能。

异常处理原则

任何在PL/pgSQL函数中发生的错误会中止该函数的执行，而且实际上会中止其周围的事务。你可以使用一个带有EXCEPTION子句的BEGIN块俘获错误并且从中恢复。

1. 在使用PL/PGSQL块中，如果使用了不能返回确定结果的SQL语句，宜在EXCEPTION中对程序可能出现的异常进行处理，避免出现未处理的出错被传递到外层块，导致程序逻辑错误。
2. 对于系统已经定义了的异常，可以直接使用。DWS暂不支持自定义异常。
3. 进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大的多。因此，非必要场景不应使用EXCEPTION。

书写规范

1. 变量命名规则：

a. 过程、函数的输入参数格式宜为：IN_参数名，参数名宜使用大写。

b. 过程、函数的输出参数格式宜为：OUT_参数名，参数名宜使用大写。

c. 过程、函数的输入输出参数格式宜为：IO_参数名，参数名宜使用大写。

d. 过程、函数的程序中用到的变量宜由v_变量名组成，变量名宜使用小写。

e. 将查询语句做成字符串拼接时，where语句的拼接变量名宜统一为v_where，select语句的拼接变量名宜为v_select。

f. 记录（RECORD）的类型（TYPE）命名宜由T+变量名组成，名称宜使用大写。

g. 游标命名宜由CUR+变量名组成，名称宜使用大写。

h. 引用游标（REF CURSOR）的命名宜由REF+变量名组成，名称宜使用大写。

 版权所有 © 华为云计算技术有限公司

2. 变量类型定义：
   a. 变量类型声明时，如果其含义和应用表某字段含义相同时，应使用%TYPE声明。
   b. 记录类型声明时，如果其含义和某应用表行数据相同时，应使用%ROWTYPE声明。
3. 注释规范：
   a. 注释应该是有意义的，而不应是重述代码。
   b. 注释应简洁、易懂，以中文为主。为了表达准确，名词或操作上也可以使用英文。
   c. 应在每个存储过程、函数的开始加入注释，内容应包括：本程序的简要功能描述、编写者、编写日期、程序版本号信息和程序变更信息，而且各存储过程开头注释应保持统一格式。
   d. 应在输入输出参数的旁边添加注释，注明变量的意义。
   e. 每个块或大分支的开始宜添加注释，描述块的简要功能，若使用算法，宜添加注释简单描述算法的目的和结果。
4. 变量声明格式：

   每行应只包含一条语句，如同时需要赋初始值，应在同一行书写。
5. 大小写规范：

   除了变量名，应一律使用大写。
6. 缩进规范：

   创建存储过程语句中，同一层的CREATE、AS/IS、BEGIN、END这几个关键字应位于同一列，其他部分依次缩进。
7. 语句详述：
   a. 变量定义语句。每行应只包含一条语句。
   b. 同一层的IF、ELSEIF、ELSE和END关键字应开始于同一列，执行语句缩进。
   c. CASE和END关键字应位于同一列，WHEN和ELSE关键字应缩进。
   d. 同一层的LOOP和END LOOP关键字应位于同一列，层内语句或嵌套应依次缩进。

 版权所有 © 华为云计算技术有限公司

4 创建和管理 DWS 数据库对象

4.1 创建和管理 DWS 数据库

数据库（Database）是表、索引、视图、存储过程、操作符等对象的集合。DWS支持创建多个数据库，但是客户端程序一次只能连接并访问一个数据库，无法跨数据库进行查询。

模板和默认数据

- DWS提供了两个模板数据库template0、template1，以及一个默认的数据库gaussdb。
- 默认情况下，每个新创建的数据库都是基于一个模板数据库。DWS数据库默认使用template1作为模板，编码格式为SQL_ASCII，且不允许自定义字符编码。若创建数据库时需指定字符编码，请使用template0创建数据库。
- 请避免使用客户端或其他手段连接及操作两个模板数据库。

  说明

  通过“show server_encoding”命令可以查看当前数据库存储编码。

创建数据库

使用CREATE DATABASE语句创建一个新的数据库。

```
CREATE DATABASE mydatabase;
```

 版权所有 © 华为云计算技术有限公司

说明

- 创建数据库时，若数据库名称长度超过63字节，server端会对数据库名称进行截断，保留前63个字节，因此建议数据库名称长度不要超过63个字节，不要使用多字节字符作为对象名。（如果出现因为误操作导致在多字节字符的中间截断进而无法删除数据库对象的现象，请使用截断前的数据库对象名进行删除操作，或将该对象从各个数据库节点的相应系统表中依次删掉。）
- 数据库名称遵循SQL标识符的一般规则。当前用户自动成为此新数据库的所有者。
- 如果一个数据库系统用于承载相互独立的用户和项目，建议把它们放在不同的数据库里。
- 如果项目或者用户是相互关联的，并且可以相互使用对方的资源，则应该把它们放在同一个数据库里，但可以规划在不同的Schema中。
- 用户必须拥有数据库创建的权限或者是数据库的系统管理员权限才能创建数据库。

查看数据库

查看数据库的方式：

- 使用\l元命令查看数据库系统的数据库列表。
  ```
  \l
  ```
- 通过系统表pg_database查询数据库列表。
  ```
  SELECT datname FROM pg_database;
  ```

修改数据库

可以使用ALTER DATABASE语句修改数据库属性（比如：owner、名称和默认的配置属性）。

- 为数据库设置默认的模式搜索路径。
  ```
  ALTER DATABASE mydatabase SET search_path TO pa_catalog,public;
  ```
- 重新命名数据库。
  ```
  ALTER DATABASE mydatabase RENAME TO newdatabase;
  ```

删除数据库

可以使用DROP DATABASE语句删除数据库。该语句删除了数据库中的系统目录，并且删除了带有数据的磁盘上的数据库目录。用户必须是数据库的owner或者系统管理员才能删除数据库。当有用户连接数据库时，删除操作会失败。删除数据库时请先连接到其他的数据库。

使用DROP DATABASE语句删除数据库：

```
DROP DATABASE newdatabase;
```

4.2 创建和管理 DWS Schema

Schema又称作模式，从逻辑上组织一个数据库中的对象和数据。通过管理Schema，允许多个用户使用同一数据库而不相互干扰，同时便于将第三方应用添加到相应的Schema下而不引起冲突。

相同的数据库对象名称可以应用在同一数据库的不同Schema中，而没有冲突。例如，a_schema和b_schema都可以包含名为mytable的表。具有所需权限的用户可以访问数据库的多个Schema中的对象。

在当前数据库中创建用户时，系统会在当前数据库中为新用户创建一个同名Schema。

版权所有 © 华为云计算技术有限公司

public 模式

每个数据库都有一个名为public的模式。 所有的数据库角色（用户）都在public模式上拥有USAGE特权，但是普通角色（用户）没有在public模式上的CREATE权限。

创建 Schema

- 使用CREATE SCHEMA命令来创建一个新的Schema。
  ```
  CREATE SCHEMA myschema;
  ```
  如果需要在模式中创建或者访问对象，其完整的对象名称由模式名称和具体的对象名称组成。中间由符号“.”隔开。例如：myschema.table。
- 用户可以创建一个由他人拥有的schema。例如，创建名为myschema的Schema，并指定Schema的所有者为用户jack。
  ```
  CREATE SCHEMA myschema AUTHORIZATION jack;
  ```
  若不指定authorization username，则其所有者为执行该命令的用户。

修改 Schema

- 使用ALTER SCHEMA修改Schema名称，只有Schema所有者可以更改Schema。
  ```
  ALTER SCHEMA schema_name RENAME TO new_name;
  ```
- 使用ALTER SCHEMA修改Schema所有者：
  ```
  ALTER SCHEMA schema_name OWNER TO new_owner;
  ```

设置 Schema 搜索路径

GUC参数search_path设置Schema的搜索顺序，参数取值形式为采用逗号分隔的Schema名称列表。如果创建对象时未指定目标Schema，则该对象会被添加到搜索路径中列出的第一个Schema中。当不同Schema中存在同名的对象时，查询对象未指定Schema的情况下，将从搜索路径中包含该对象的第一个Schema中返回对象。

- 使用SHOW命令查看当前搜索路径。
  ```
  SHOW SEARCH_PATH;
   search_path
  ----------------
   "$user",public
  (1 row)
  ```
  search_path参数的默认值为："$user"，public。$user表示与当前会话用户名同名的Schema名，如果这样的模式不存在，$user将被忽略。所以默认情况下，用户连接数据库后，如果数据库下存在同名Schema，则对象会添加到同名Schema下，否则对象被添加到Public Schema下。
- 使用SET命令修改当前会话的默认Schema。例如，将搜索路径设置为myschema、public，首先搜索myschema。
  ```
  SET SEARCH_PATH TO myschema, public;
  ```
  也可以使用ALTER ROLE命令为特定的角色（用户）设置search_path。例如：
  ```
  ALTER ROLE jack SET search_path TO myschema, public;
  ```

使用 Schema

在特定Schema下创建对象或者访问特定Schema下的对象，需要使用有Schema修饰的对象名。名称包含Schema名以及对象名，之间用“.”号分开。

- 在myschema下创建mytable表。以schema_name.table_name格式创建表。
  ```
  CREATE TABLE myschema.mytable(id int, name varchar(20));
  ```
- 查询myschema下mytable表的所有数据。

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM myschema.mytable;
 id | name
----+------
(0 rows)
```

查看 Schema

- 使用current_schema()函数查看当前Schema：

```
SELECT current_schema();
 current_schema
----------------
 myschema
(1 row)
```

- 要查看Schema所有者，请对系统表PG_NAMESPACE和PG_USER执行如下关联查询。语句中的schema_name请替换为实际要查找的Schema名称。

```
SELECT s.nspname,u.usename AS nspowner FROM PG_NAMESPACE s, PG_USER u WHERE nspname='schema_name' AND s.nspowner = u.usesysid;
```

- 要查看所有Schema的列表，请查询PG_NAMESPACE系统表。

```
SELECT * FROM PG_NAMESPACE;
```

- 使用PGXC_TOTAL_SCHEMA_INFO视图查询整个集群的Schema空间使用情况。

```
SELECT * FROM PGXC_TOTAL_SCHEMA_INFO;
```

- 要查看属于某Schema下表的列表，请查询系统视图PG_TABLES。例如，以下查询会返回Schema PG_CATALOG中的表列表。

```
SELECT distinct(tablename),schemaname FROM PG_TABLES where schemaname = 'pg_catalog';
```

Schema 的权限控制

默认情况下，用户只能访问属于自己的Schema中的数据库对象。如需要访问其他Schema的对象，则需赋予对应Schema的usage权限。

通过将模式的CREATE权限授予某用户，被授权用户就可以在此模式中创建对象。

- 将myschema的usage权限赋给用户jack。

```
GRANT USAGE ON schema myschema TO jack;
```

- 将用户jack对于myschema的usage权限收回。

```
REVOKE USAGE ON schema myschema FROM jack;
```

删除 Schema

- 使用DROP SCHEMA命令删除一个空的Schema（即该Schema下没有数据库对象）。

```
DROP SCHEMA IF EXISTS myschema;
```

- 默认情况下，删除一个Schema前，它必须为空。 要删除一个Schema及其包含的所有对象（表、数据、函数等），需要使用CASCADE关键字。

```
DROP SCHEMA myschema CASCADE;
```

系统 Schema

- 每个数据库都包含一个pg_catalog schema，它包含系统表和所有内置数据类型、函数、操作符。pg_catalog是搜索路径中的一部分，始终在临时表所属的模式后面，并在search_path中所有模式的前面，即具有第二搜索优先级。这样确保可以搜索到数据库内置对象。如果用户需要使用和系统内置对象重名的自定义对象时，可以在操作自定义对象时带上自己的模式。
- information_schema由一个包含数据库中对象信息的视图集合组成。 这些视图以一种标准化的方式从系统目录表中得到系统信息。

 版权所有 © 华为云计算技术有限公司

4.3 创建和管理 DWS 表

创建表

CREATE TABLE命令创建一个表，创建表时可以定义以下内容：

- 表的列及数据类型。
- 表约束的定义，即任何用于限制列或者表中数据的表约束或者列约束。参见**表约束的定义**。
- 表分布的定义，即表的分布策略，它决定DWS数据库如何在片（Segment）之间划分数据。 参见**表分布的定义**。
- 表存储格式。参见**选择DWS表存储模型**。
- 分区表定义。参见**创建和管理DWS分区表**。

示例：CREATE TABLE创建了一个表web_returns_p1，使用wr_item_sk作为分布键，并基于wr_returned_date_sk设置了range分布功能。

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2019 START(20191231) END(20221231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
```

表约束的定义

可以在列和表上定义约束来限制表中的数据，但是有以下一些限制：

- 表中的主键约束和唯一约束必须包含分布列。
- 列存表支持PARTIAL CLUSTER KEY、主键和唯一表级约束，不支持外键表级约束。
- 列存表的字段约束只支持NULL、NOT NULL和DEFAULT常量值。

表 4-1 表约束

约束项	说明	示例
检查约束	检查约束允许用户指定一个特定列中的值必须满足一个布尔（真值）表达式。	创建表products，要求产品价格为正： CREATE TABLE products (product_no integer, name text, price numeric CHECK (price > 0));

版权所有 © 华为云计算技术有限公司

约束项	说明	示例
非空约束	非空约束指定一个列不能有空值。非空约束总是被写作为列约束。	创建表products，要求product_no和name不能为空： CREATE TABLE products (product_no integer NOT NULL, name text NOT NULL, price numeric);
唯一约束	唯一约束确保一列或者一组列中包含的数据对于表中所有的行都是唯一的。 如果没有声明DISTRIBUTE BY REPLICATION，则唯一约束的列集合中必须包含分布列。	创建表products，product_no值不能重复： CREATE TABLE products (product_no integer UNIQUE, name text, price numeric)DISTRIBUTE BY HASH(product_no);
主键约束	主键约束是一个UNIQUE约束和一个NOT NULL约束的组合。 如果没有声明DISTRIBUTE BY REPLICATION，则主键约束的列集合中必须包含分布列。如果一个表具有主键，这个列（或者这一组列）会被默认选中为该表的分布键。	创建表products，主键约束为product_no： CREATE TABLE products (product_no integer PRIMARY KEY, name text, price numeric)DISTRIBUTE BY HASH(product_no);
局部聚簇	局部聚簇通过min/max稀疏索引较快的实现基表扫描的filter过滤。Partial Cluster Key可以指定多列，但是一般不建议超过2列。	创建表products，PCK为product_no： CREATE TABLE products (product_no integer, name text, price numeric, PARTIAL CLUSTER KEY(product_no)) WITH (ORIENTATION = COLUMN);

表分布的定义

DWS支持的分布方式：复制表（Replication）、哈希表（Hash）和轮询表（Roundrobin）。

说明

轮询表（Roundrobin）分布方式仅8.1.2及以上集群版支持。

版权所有 © 华为云计算技术有限公司

策略	描述	适用场景	优势与劣势
复制表（Replication）	集群中每一个DN实例上都有一份全量表数据。	小表、维度表。	• Replication优点是每个DN上都有此表的全量数据，在join操作中可以避免数据重分布操作，从而减小网络开销，同时减少了plan segment(每个plan segment都会起对应的线程) • Replication缺点是每个DN都保留了表的完整数据，造成数据的冗余。一般情况下只有较小的维度表才会定义为Replication表。
哈希表（Hash）	表数据通过hash方式散列到集群中的所有DN实例上。	数据量较大的事实表。	• 在读/写数据时可以利用各个节点的IO资源，大幅度提升表的读/写速度。 • 一般情况下大表（1000000条记录以上）定义为Hash表。
轮询表（Roundrobin）	表的每一行被轮番地发送给各个DN，数据会被均匀地分布在各个DN中。	数据量较大的事实表，且使用Hash分布时找不到合适的分布列。	• Roundrobin优点是保证了数据不会发生倾斜，从而提高了集群的空间利用率。 • Roundrobin缺点是无法像Hash表一样进行DN本地化优化，查询性能通常不如Hash表。 • 一般在大表无法找到合适的分布列时，定义为Roundrobin表，若大表能够找到合适的分布列，优先选择性能更好的Hash分布。

选择分布列

采用Hash分布方式，需要为用户表指定一个分布列（distribute key）。当插入一条记录时，系统会根据分布列的值进行hash运算后，将数据存储在对应的DN中。

所以Hash分布列选取至关重要，需要满足以下原则：

1. **列值应比较离散，以便数据能够均匀分布到各个DN**。例如，考虑选择表的主键为分布列，如在人员信息表中选择身份证号码为分布列。
2. **在满足第一条原则的情况下尽量不要选取存在常量filter的列**。例如，表dwcjk相关的部分查询中出现dwcjk的列zqdh存在常量的约束(例如zqdh=’000001’)，那么就应当尽量不用zqdh做分布列。
3. **在满足前两条原则的情况，考虑选择查询中的连接条件为分布列**，以便Join任务能够下推到DN中执行，且减少DN之间的通信数据量。

   对于Hash分表策略，如果分布列选择不当，可能导致数据倾斜，查询时出现部分DN的I/O短板，从而影响整体查询性能。因此在采用Hash分表策略之后需对表的数据进行数据倾斜性检查，以确保数据在各个DN上是均匀分布的。可以使用以下SQL检查数据倾斜性

   ```
   select
   xc_node_id, count(1)
   from tablename
   ```

 版权所有 © 华为云计算技术有限公司

```
group by xc_node_id
order by xc_node_id desc;
```

其中xc_node_id对应DN，一般来说，**不同DN的数据量相差5%以上即可视为倾斜，如果相差10%以上就必须要调整分布列**。

4. 一般不建议用户新增一列专门用作分布列，尤其不建议用户新增一列，然后用SEQUENCE的值来填充作为分布列。因为SEQUENCE可能会带来性能瓶颈和不必要的维护成本。

查看表数据

- 使用系统表pg_tables查询数据库所有表的信息。
  ```
  SELECT * FROM pg_tables;
  ```
- 使用gsql的\d+命令查询表的属性。
  ```
  \d+ customer_t1;
  ```
- 执行如下命令查询表customer_t1的数据量。
  ```
  SELECT count(*) FROM customer_t1;
  ```
- 执行如下命令查询表customer_t1的所有数据。
  ```
  SELECT * FROM customer_t1;
  ```
- 执行如下命令只查询字段c_customer_sk的数据。
  ```
  SELECT c_customer_sk FROM customer_t1;
  ```
- 执行如下命令过滤字段c_customer_sk的重复数据。
  ```
  SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
  ```
- 执行如下命令查询字段c_customer_sk为3869的所有数据。
  ```
  SELECT * FROM customer_t1 WHERE c_customer_sk = 3869;
  ```
- 执行如下命令按照字段c_customer_sk进行排序。
  ```
  SELECT * FROM customer_t1 ORDER BY c_customer_sk;
  ```

删除表数据

⚠ 注意

请谨慎执行DROP TABLE和TRUNCATE TABLE命令，删除表后，数据将无法恢复。

- 从数据库中删除表customer_t1。
  ```
  DROP TABLE customer_t1;
  ```
- 清空一个表的行但不移除该表的定义，可使用DELETE或者TRUNCATE。

  删除表customer_t1中所有的行。
  ```
  TRUNCATE TABLE customer_t1;
  ```

  删除表customer_t1中所有的行。
  ```
  DELETE FROM customer_t1;
  ```

  删除表customer_t1中所有c_customer_sk为3869的记录：
  ```
  DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
  ```

管理 UNLOGGED 表

UNLOGGED表即非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动清理，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。

 版权所有 © 华为云计算技术有限公司

使用场景：非日志表不能保证数据的安全性，需要在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。在创建非日志表时应将cnretry关闭（即设置GUC参数max_query_retry_times=0）。

故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。

- 9.1.0版本后，UNLOGGED表默认存储在pg_unlogged表空间下，且不可迁移或指定到其他表空间。
- 从低版本升级到9.1.0版本后，旧版本创建的UNLOGGED表还存储在原表空间下。

9.1.0版本提供UNLOGGED表迁移脚本switch_unlogged_tablespace.py，配合GUC参数**enable_unlogged_tablespace_compat**，可以优化RTO。

1. 脚本位于 $GPHOME/script目录，可以通过-?获取帮助信息。

```
[perfadm@linux83108 script]$ python3 switch_unlogged_tablepace.py -?
Usage:
  python3 switch_unlogged_tablepace.py -? | --help
  python3 switch_unlogged_tablepace.py -t query [--dbname=DBNAME] [--verbose]
  python3 switch_unlogged_tablepace.py -t switch [--dbname=DBNAME] [--without-disable]

General options:
  -t                            Type of the command.
  -?, --help                    Show help information for this utility, and exit the command line mode.

Options for query:
  --dbname                      Database name.
  --verbose                     List all unlogged tables.

Options for switch:
  --dbname                      Database name.
  --without-disable             Do not disable unlogged tablespace compatibility after switch all unlogged tables.
```

2. 迁移所有UNLOGGED表（建议）

```
python3 switch_unlogged_tablespace.py -t switch
```

3. 迁移成功后，GUC参数**enable_unlogged_tablespace_compat**会自动设置为off。

须知

建议在升级到9.1.0版本后通过以下两步操作提升实例重启RTO：

1. 使用switch_unlogged_tablespace.py脚本将UNLOGGED表全部迁移到pg_unlogged表空间。
2. 如果旧版本没有使用UNLOGGED表，建议将GUC参数enable_unlogged_tablespace_compat设置为OFF。

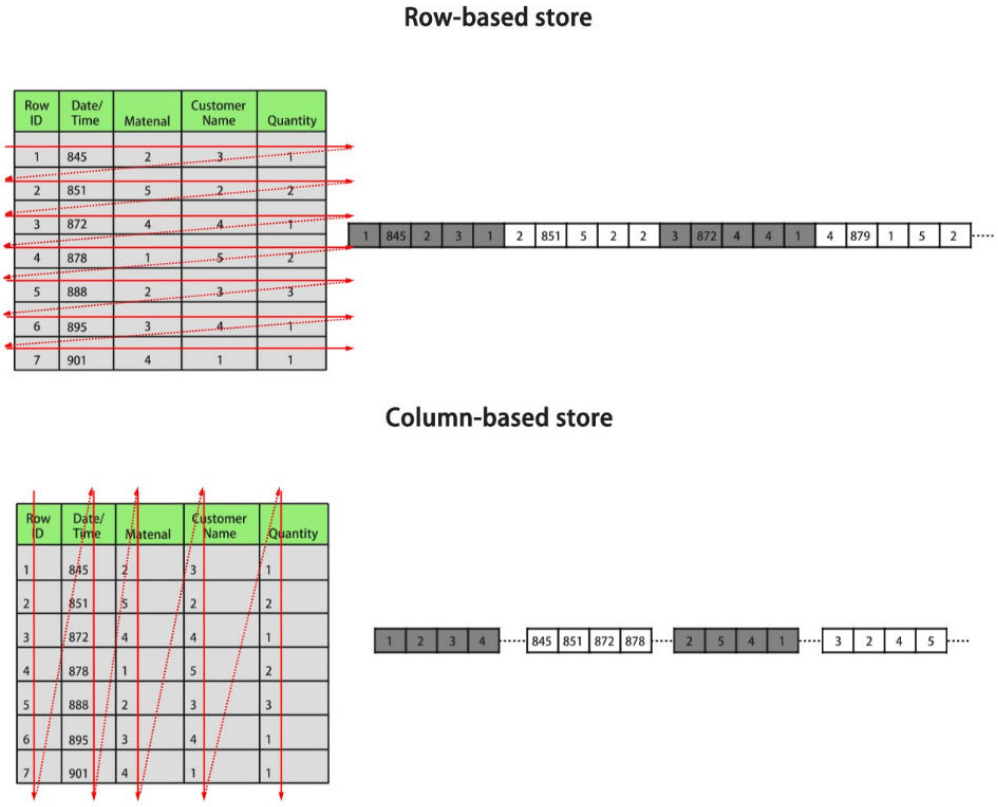
4.4 选择 DWS 表存储模型

DWS支持行列混合存储。当创建一个表时，可以选择表的存储格式为行存储或列存储。

行存储是指将表按行存储到硬盘分区上，列存储是指将表按列存储到硬盘分区上。默认情况下，创建的表为行存储。行存储和列存储的差异请参见图4-1。

版权所有 © 华为云计算技术有限公司

图 4-1 行存储和列存储的差异



上图中，左上为行存表，右上为行存表在硬盘上的存储方式。左下为列存表，右下为列存表在硬盘上的存储方式。

DWS表的行/列存储通过表定义的orientation属性定义。当指定orientation属性为row时，表为行存储；当指定orientation属性为column时，表为列存储；如果不指定，默认为行存储。行、列存储模型各有优劣，建议根据实际情况选择：

表 4-2 表的存储类型及场景

存储模型	优点	缺点	适用场景
行存	数据按照行进行存储，在查询某一行数据时，可以快速定位到目标位置。	查询时即使只涉及某几列，所有数据也都会被读取。	1. 表的字段个数比较少，查询表的大部分字段。 2. 点查询（返回记录少，基于索引的简单查询）。 3. 频繁进行增删改查操作且该涉及整行数据。

版权所有 © 华为云计算技术有限公司

存储模型	优点	缺点	适用场景
列存	1. 查询时只有涉及到的列会被读取。 2. 列数据特征比较相似，能够更有效地进行数据压缩。	不适合少量数据INSERT或UPDATE操作。	1. 表的字段比较多（大宽表），查询中涉及到的列不多。 2. 统计分析类查询 (关联、分组操作较多的场景)。 3. 即席查询（查询条件不确定，行存表扫描难以使用索引）。

创建一个行存表

例如，创建一个名为customer_t1的行存表：

```
CREATE TABLE customer_t1
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
);
```

创建一个列存表

例如，创建一个名为customer_t2的列存表：

```
CREATE TABLE customer_t2
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
)
WITH (ORIENTATION = COLUMN);
```

使用压缩

表压缩可以在创建表时开启，压缩表能够使表中的数据以压缩格式存储，意味着占用相对少的内存。

对于I/O读写量大，CPU富足（计算相对小）的场景，选择高压缩比；反之选择低压缩比。建议依据此原则进行不同压缩下的测试和对比，以选择符合自身业务情况的最优压缩比。压缩比通过COMPRESSION参数指定，其支持的取值如下：

- 列存表为：YES/NO/LOW/MIDDLE/HIGH，默认值为LOW。
- 行存表为：YES/NO，默认值为NO。（行存表压缩功能暂未商用，如需使用请联系技术支持工程师）

各压缩级别所适用的业务场景说明如下：

压缩级别	所适用的业务场景
低级别压缩	系统CPU使用率高，存储磁盘空间充足。
中度压缩	系统CPU使用率适中，但存储磁盘空间不是特别充足。

版权所有 © 华为云计算技术有限公司

压缩级别	所适用的业务场景
高级别压缩	系统CPU使用率低，磁盘空间不充裕。

例如，创建一个名为customer_t3的列存压缩表：

```
CREATE TABLE customer_t3
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
)
WITH (ORIENTATION = COLUMN,COMPRESSION=middle);
```

4.5 创建和管理 DWS 分区表

分区表就是把逻辑上的一张表根据分区策略分成几张物理块库进行存储，这张逻辑上的表称之为分区表，物理块称之为分区。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。当进行条件查询时，系统只会扫描满足条件的分区，避免全表扫描，从而提升查询性能。

关于分区表维护的详细语法介绍请参见《 SQL语法参考 》中“ALTER TABLE PARTITION”章节。

分区表的优势：

- 改善查询性能。对分区对象的查询可以仅搜索自己关心的分区，提高检索效率。
- 增强可用性。如果分区表的某个分区出现故障，表在其他分区的数据仍然可用。
- 提升可维护性。对于需要周期性删除的过期历史数据，可以通过drop/truncate分区的方式快速高效处理。

支持的表分区类型

- 范围分区（ Range Partitioning ），基于一个数值型范围划分数据，例如按日期或价格区间定义。
- 列表分区（ List Partitioning ），基于一个值列表划分数据，例如按销售范围或产品属性定义。仅8.1.3及以上集群版本支持。

分区策略选择

当表有以下特征时，可以考虑使用表分区策略：

- 数据具有明显区间性的字段。

  分区表需要根据有明显区间性字段进行表分区。比如按照日期、区域、数值等字段进行分区，时间字段是最常见的分区字段。
- 业务查询有明显的区间范围特征。

  查询数据可落到区间范围指定的分区内，这样才能通过分区剪枝，只扫描查询需要的分区，从而提升数据扫描效率，降低数据扫描的IO开销。
- 表数据量比较大。

版权所有 © 华为云计算技术有限公司

小表扫描本身耗时不大，分区表的性能收益不明显，因此只建议对大表采取分区策略。列存储模式下因为每个列是单独的文件存储，且最小的存储单元CU可存储6w行数据，因此对于列存分区表，建议每个分区的数据不小于DN个数*6w。

创建范围(range)分区表

示例：创建一个按wr_returned_date_sk范围分区的表web_returns_p1。

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE (wr_returned_date_sk)
(
    PARTITION p2016 VALUES LESS THAN(20161231),
    PARTITION p2017 VALUES LESS THAN(20171231),
    PARTITION p2018 VALUES LESS THAN(20181231),
    PARTITION p2019 VALUES LESS THAN(20191231),
    PARTITION pxxxx VALUES LESS THAN(maxvalue)
);
```

对于分区间隔固定、批量创建分区的场景。可使用如下示例：

```
CREATE TABLE web_returns_p2
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
```

示例：创建一个按时间日期作为分区的表web_returns_p2，其中time作为分区键。

```
CREATE TABLE web_returns_p2
(
   id integer,
   idle numeric,
   IO numeric,
   scope text,
   IP text,
   time timestamp
)
 WITH (TTL='7 days',PERIOD='1 day')
PARTITION BY RANGE(time)
 (
   PARTITION P1 VALUES LESS THAN('2022-01-05 16:32:45'),
   PARTITION P2 VALUES LESS THAN('2022-01-06 16:56:12')
 );
```

创建列表(list)分区表

LIST分区表可以使用任意允许值比较的列作为分区键列。创建LIST分区表时，必须要为每一个分区声明每一个值分区。

示例：创建LIST分区表sales_info。

```
CREATE TABLE sales_info
(
```

 版权所有 © 华为云计算技术有限公司

```
sale_time  timestamptz,
period     int,
city       text,
price      numeric(10,2),
remark     varchar2(100)
)
DISTRIBUTE BY HASH(sale_time)
PARTITION BY LIST (period, city)
(
PARTITION province1_202201 VALUES (('202201', 'city1'), ('202201', 'city2')),
PARTITION province2_202201 VALUES (('202201', 'city3'), ('202201', 'city4'), ('202201', 'city5'))
);
```

对已有的表进行分区

表只能在创建时被分区。 如果用户有一个表想要分区，用户必须创建一个分过区的表，把原始表的数据载入到新表，再删除原始表并且把分过区的表重命名为原始表的名称。 用户还必须重新授权表上的权限。例如：

```
CREATE TABLE web_returns_p2
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
INSERT INTO web_returns_p2 SELECT * FROM web_returns_p1;
DROP TABLE web_returns_p1;
ALTER TABLE web_returns_p2 RENAME TO web_returns_p1;
GRANT ALL PRIVILEGES ON web_returns_p1 TO dbadmin;
GRANT SELECT ON web_returns_p1 TO jack;
```

增加一个分区

使用ALTER TABLE语句为范围分区表增加一个分区。例如，为表web_returns_p1增加分区P2020。

```
ALTER TABLE web_returns_p1 ADD PARTITION P2020 VALUES LESS THAN (20201231);
```

为列表分区增加一个分区。例如，为表sales_info增加分区province3_202201。

```
ALTER TABLE sales_info ADD PARTITION province3_202201 VALUES (('202201', 'city6'), ('202201', 'city7'));
```

分割一个分区

范围分区表和列表分区表分割分语法有所区别：

- 使用ALTER TABLE语句为范围分区表分割一个分区。例如，将表*web_returns_p1*分区*pxxxx*以20201231为分割点分割为p2020和p20xx两个分区。
  ```
  ALTER TABLE web_returns_p1 SPLIT PARTITION pxxxx AT(20201231) INTO (PARTITION p2020,PARTITION p20xx);
  ```
- 使用ALTER TABLE语句为列表分区表分割一个分区。例如，将表*sales_info*分区*province2_202201*分割为*province3_202201*和*province4_202201*两个分区。
  ```
  ALTER TABLE sales_info SPLIT PARTITION province2_202201 VALUES(('202201', 'city5')) INTO (PARTITION province3_202201,PARTITION province4_202201);
  ```

 版权所有 © 华为云计算技术有限公司

合并一个分区

使用ALTER TABLE语句为分区表合并一个分区。例如，将表*web_returns_p1* 分区p2016和p2017合并为一个分区p20162017。

```
ALTER TABLE web_returns_p1 MERGE PARTITIONS p2016,p2017 INTO PARTITION p20162017;
```

删除一个分区

使用ALTER TABLE语句从分区表中删除一个分区。例如，删除表web_returns_p1的分区P2020。

```
ALTER TABLE web_returns_p1 DROP PARTITION P2020;
```

查询分区

- 查询分区p2019。

  ```
  SELECT * FROM web_returns_p1 PARTITION (p2019);
  SELECT * FROM web_returns_p1 PARTITION FOR (20201231);
  ```

- 查看分区表信息，可使用系统表dba_tab_partitions。

  ```
  SELECT * FROM dba_tab_partitions where table_name='web_returns_p1';
  ```

删除分区表

使用DROP TABLE语句删除一个分区表。

```
DROP TABLE web_returns_p1;
```

设置分区索引是否可用

给分区表customer_address创建LOCAL索引student_grade_index，并指定分区的索引名称。

```
CREATE INDEX customer_address_index ON customer_address(ca_address_id) LOCAL
(
    PARTITION P1_index,
    PARTITION P2_index,
    PARTITION P3_index
);
```

重建分区表customer_address中分区P1上的所有索引。

```
ALTER TABLE customer_address MODIFY PARTITION P1 REBUILD UNUSABLE LOCAL INDEXES;
```

设置分区表customer_address的分区P3上的所有索引不可用。

```
ALTER TABLE customer_address MODIFY PARTITION P3 UNUSABLE LOCAL INDEXES;
```

重建分区索引

长期运行的分区表，索引可能产生碎片，重建索引，可提升查询效率，以下演示分区插入数据后重建分区索引。

步骤1 创建分区表。

```
DROP TABLE IF EXISTS sales;
CREATE TABLE sales (
   id         INT,
   sale_date   DATE,
   amount      DECIMAL(10,2)
)
```

 版权所有 © 华为云计算技术有限公司

```
PARTITION BY RANGE (sale_date)
(
    PARTITION p202310 VALUES LESS THAN ('2023-11-01'),
    PARTITION p202311 VALUES LESS THAN ('2023-12-01')
);
```

步骤2 创建索引。

```
CREATE INDEX idx_sale_date ON sales (sale_date) LOCAL;
```

步骤3 插入10月、11月分区数据。

```
INSERT INTO sales PARTITION (p202310)
VALUES
(1, '2023-10-05', 100.50),
(2, '2023-10-10', 200.75);
INSERT INTO sales PARTITION (p202311)
VALUES
(3, '2023-11-15', 300.00);
```

步骤4 重建特定分区p202310的索引。

```
ALTER TABLE sales  REBUILD PARTITION p202310  WITHOUT UNUSABLE;
```

----结束

交换分区（EXCHANGE PARTITION）

如下示例将演示一个普通表math_grade数据迁移到分区表student_grade中分区（math）的操作。

1. 创建分区表student_grade。

```
CREATE TABLE student_grade (
        stu_name     char(5),
        stu_no       integer,
        grade        integer,
        subject      varchar(30)
)
PARTITION BY LIST(subject)
(
        PARTITION gym VALUES('gymnastics'),
        PARTITION phys VALUES('physics'),
        PARTITION history VALUES('history'),
        PARTITION math VALUES('math')
);
```

2. 插入数据到分区表student_grade中。

```
INSERT INTO student_grade VALUES
        ('Ann', 20220101, 75, 'gymnastics'),
        ('Jeck', 20220103, 60, 'math'),
        ('Anna', 20220108, 56, 'history'),
        ('Jann', 20220107, 82, 'physics'),
        ('Molly', 20220104, 91, 'physics'),
        ('Sam', 20220105, 72, 'math');
```

3. 查询分区表student_grade的math分区记录。

```
SELECT * FROM student_grade PARTITION (math);
```

查询结果如下：

```
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jeck     | 20220103 |    60 | math
 Sam      | 20220105 |    72 | math
(2 rows)
```

4. 创建一个与分区表student_grade定义匹配的普通表math_grade。

```
CREATE TABLE math_grade
(
        stu_name     char(5),
        stu_no       integer,
```

 版权所有 © 华为云计算技术有限公司

```
    grade       integer,
    subject     varchar(30)
);
```

5. 插入数据到表math_grade中。数据与分区表student_grade的math分区的分区规则一致。

```
INSERT INTO math_grade VALUES
    ('Ann', 20220101, 75, 'math'),
    ('Jeck', 20220103, 60, 'math'),
    ('Anna', 20220108, 56, 'math'),
    ('Jann', 20220107, 82, 'math');
```

6. 将普通表math_grade数据迁移到分区表student_grade的分区（math）。

```
ALTER TABLE student_grade EXCHANGE PARTITION (math) WITH TABLE math_grade;
```

7. 查询分区表student_grade，结果显示表math_grade中的数据已和分区表student_grade的分区math中的数据交换。

```
SELECT * FROM student_grade PARTITION (math);
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Anna     | 20220108 |    56 | math
 Jeck     | 20220103 |    60 | math
 Ann      | 20220101 |    75 | math
 Jann     | 20220107 |    82 | math
(4 rows)
```

8. 查询表math_grade，显示之前存储在分区表student_grade的分区math中的数据已交换到表math_grade中。

```
SELECT * FROM math_grade;
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jeck     | 20220103 |    60 | math
 Sam      | 20220105 |    72 | math
(2 rows)
```

4.6 创建和管理 DWS 索引

索引可以提高数据的访问速度，但同时也增加了插入、更新和删除操作的处理时间。所以是否要为表增加索引，索引建立在哪些字段上，是创建索引前必须要考虑的问题。需要分析应用程序的业务处理、数据使用、经常被用作查询的条件或者被要求排序的字段来确定是否建立索引。

索引类型

- B-Tree：B-tree索引使用一种类似于B+树的结构来存储数据的键值，通过这种结构能够快速地查找索引。B-Tree适合支持比较查询以及查询范围。
- GIN：GIN索引是倒排索引，可以处理包含多个键的值（比如数组）。
- GiST：GiST索引适用于几何和地理等多维数据类型和集合数据类型。
- PSORT：PSORT索引。针对列存表进行局部排序索引。

行存表支持的索引类型：B-Tree（行存表缺省值）、GIN、GiST。列存表支持的索引类型：PSORT（列存表缺省值）、B-Tree、GIN。

说明

对于点查询场景，推荐建立B-Tree索引。

索引的选择原则

索引建立在数据库表中的某些列上。因此，在创建索引时，应该仔细考虑在哪些列上创建索引。

版权所有 © 华为云计算技术有限公司

- 在经常需要搜索查询的列上创建索引，可以加快搜索的速度。
- 在作为主键的列上创建索引，强制该列的唯一性和组织表中数据的排列结构。
- 在经常使用连接的列上创建索引，这些列主要是一些外键，可以加快连接的速度。
- 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的。
- 在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间。
- 在经常使用WHERE子句的列上创建索引，加快条件的判断速度。
- 为经常出现在关键字ORDER BY、GROUP BY、DISTINCT后面的字段建立索引。

  说明

  - 索引创建成功后，系统会自动判断何时引用索引。当系统认为使用索引比顺序扫描更快时，就会使用索引。
  - 索引创建成功后，必须和表保持同步以保证能够准确地找到新数据，这样就增加了数据操作的负荷。因此请定期删除无用的索引。

创建索引

DWS支持4种创建索引的方式请参见表4-3。

说明

- 索引创建成功后，系统会自动判断何时引用索引。当系统认为使用索引比顺序扫描更快时，就会使用索引。
- 索引创建成功后，必须和表保持同步以保证能够准确地找到新数据，这样就增加了数据操作的负荷。因此请定期删除无用的索引。

表 4-3 索引方式

索引方式	描述
唯一索引	可用于约束索引属性值的唯一性，或者属性组合值的唯一性。如果一个表声明了唯一约束或者主键，则DWS自动在组成主键或唯一约束的字段上创建唯一索引（可能是多字段索引），以实现这些约束。目前，DWS只有B-Tree可以创建唯一索引。
多字段索引	一个索引可以定义在表中的多个属性上。目前，DWS中的B-Tree支持多字段索引，且最多可在32个字段上创建索引。
部分索引	建立在一个表的子集上的索引，这种索引方式只包含满足条件表达式的元组。
表达式索引	索引建立在一个函数或者从表中一个或多个属性计算出来的表达式上。表达式索引只有在查询时使用与创建时相同的表达式才会起作用。

- 创建一个普通表。

```
CREATE TABLE tpcds.customer_address_bak AS TABLE tpcds.customer_address;
```

 版权所有 © 华为云计算技术有限公司

- 创建普通索引

  如果对于tpcds.customer_address_bak表，需要经常进行以下查询。
  ```
  SELECT ca_address_sk FROM tpcds.customer_address_bak WHERE ca_address_sk=14888;
  ```

  通常，数据库系统需要逐行扫描整个tpcds.customer_address_bak表以寻找所有匹配的元组。如果表tpcds.customer_address_bak的规模很大，但满足WHERE条件的只有少数几个（可能是零个或一个），则这种顺序扫描的性能就比较差。如果让数据库系统在ca_address_sk属性上维护一个索引，用于快速定位匹配的元组，则数据库系统只需要在搜索树上查找少数的几层就可以找到匹配的元组，这将会大幅度提高数据查询的性能。同样，在数据库中进行更新和删除操作时，索引也可以提升这些操作的性能。

  使用以下命令创建索引。

  ```
  CREATE INDEX index_wr_returned_date_sk ON tpcds.customer_address_bak (ca_address_sk);
  ```

- 创建唯一索引

  如果一个表声明了唯一约束或者主键，则DWS自动在组成主键或唯一约束的字段上创建唯一索引（可能是多字段索引）。如果建表时未声明唯一约束或主键，可使用CREATE INDEX创建。

  ```
  CREATE UNIQUE INDEX unique_index ON tpcds.customer_address_bak(ca_address_sk);
  ```

- 创建多字段索引

  假如用户需要经常查询表tpcds.customer_address_bak中ca_address_sk是5050，且ca_street_number小于1000的记录，使用以下命令进行查询。
  ```
  SELECT ca_address_sk,ca_address_id FROM tpcds.customer_address_bak WHERE ca_address_sk = 5050 AND ca_street_number < 1000;
  ```

  使用以下命令在字段ca_address_sk和ca_street_number上定义一个多字段索引。
  ```
  CREATE INDEX more_column_index ON
  tpcds.customer_address_bak(ca_address_sk ,ca_street_number );
  ```

- 创建部分索引

  如果只需要查询ca_address_sk为5050的记录，可以创建部分索引来提升查询效率。

  ```
  CREATE INDEX part_index ON tpcds.customer_address_bak(ca_address_sk) WHERE ca_address_sk = 5050;
  ```

- 创建表达式索引

  假如经常需要查询ca_street_number小于1000的信息，执行如下命令进行查询。
  ```
  SELECT * FROM tpcds.customer_address_bak WHERE trunc(ca_street_number) < 1000;
  ```

  可以为上面的查询创建表达式索引：
  ```
  CREATE INDEX para_index ON tpcds.customer_address_bak (trunc(ca_street_number));
  ```

查看索引

- 执行如下命令查询系统和用户定义的所有索引。
  ```
  SELECT RELNAME FROM PG_CLASS WHERE RELKIND='i';
  ```
- 执行如下命令查询指定索引的信息。
  ```
  \di+ index_wr_returned_date_sk
  ```

重建索引

- 重建索引index_wr_returned_date_sk：
  ```
  REINDEX INDEX index_wr_returned_date_sk;
  ```
- 重建表上的所有索引：
  ```
  REINDEX TABLE tpcds.customer_address_bak;
  ```

 版权所有 © 华为云计算技术有限公司

删除索引

使用DROP INDEX命令删除索引：

```
DROP INDEX index_wr_returned_date_sk;
```

4.7 创建和使用 DWS 序列

序列Sequence是用来产生唯一整数的数据库对象。序列的值是按照一定规则自增的整数。因为自增所以不重复，因此说Sequence具有唯一标识性。这也是Sequence常被用作主键的原因。

通过序列使某字段成为唯一标识符的方法有两种：

- 一种是声明字段的类型为序列整型，由数据库在后台自动创建一个对应的Sequence。
- 另一种是使用CREATE SEQUENCE自定义一个新的Sequence，然后通过nextval('sequence_name')函数递增序列并返回新值，将nextval('sequence_name')函数读取的序列值，指定为某一字段的默认值，这样该字段就可以作为唯一标识符。

创建序列

方法一： 声明字段类型为序列整型来定义标识符字段。例如：

```
CREATE TABLE T1
(
    id    serial,
    name  text
);
```

方法二： 创建序列，并通过nextval('sequence_name')函数指定为某一字段的默认值。这种方式更灵活，可以为序列定义cache，一次预申请多个序列值，减少与GTM的交互次数，来提高性能。

1. 创建序列

   ```
   CREATE SEQUENCE seq1 cache 100;
   ```

2. 指定为某一字段的默认值，使该字段具有唯一标识属性。

   ```
   CREATE TABLE T2
   (
       id   int not null default nextval('seq1'),
       name text
   );
   ```

说明

- 除了为序列指定了cache，方法二所实现的功能基本与方法一类似。但是一旦定义cache，序列将会产生空洞(序列值为不连贯的数值，如：1.4.5)，并且不能保序。另外为某序列指定从属列后，该列删除，对应的sequence也会被删除。 虽然数据库并不限制序列只能为一列产生默认值，但最好不要多列共用同一个序列。
- 当前版本只支持在定义表的时候指定自增列，或者指定某列的默认值为nextval('seqname')， 不支持在已有表中增加自增列或者增加默认值为nextval('seqname')的列。
- 建议控制SEQUENCE的使用数量（含SERIAL类型），因分布式场景下SEQUENCE中序列值的产生依赖GTM全局管理，频繁的与GTM交互会严重影响性能。建议引用公共的SEQUENCE对象，或者针对需要唯一值的场景使用UUID进行替换。同时为尽量减少与GTM的交互，针对手动创建的SEQUENCE，建议指定CACHE值不小于10000。

版权所有 © 华为云计算技术有限公司

修改一个序列

ALTER SEQUENCE命令更改现有序列的属性，包括修改拥有者、归属列和最大值。

- 指定序列与列的归属关系。

  将序列和一个表的指定字段进行关联。在删除那个字段或其所在表的时候会自动删除已关联的序列。

  ```
  ALTER SEQUENCE seq1 OWNED BY T2.id;
  ```

- 将序列serial的最大值修改为300：

  ```
  ALTER SEQUENCE seq1 MAXVALUE 300;
  ```

删除一个序列

使用DROP SEQUENCE命令删除一个序列。 例如，将删除名为seq1的序列：

```
DROP SEQUENCE seq1;
```

注意事项

新序列值的产生是靠GTM维护的，默认情况下，每申请一个序列值都要向GTM发送一次申请，GTM在当前值的基础上加上步长值作为产生的新值返回给调用者。GTM作为全局唯一的节点，势必成为性能的瓶颈，所以对于需要大量频繁产生序列号的操作，如使用Bulkload（批量快速导入数据）功能进行数据导入场景，是非常不推荐产生默认序列值的。比如，在下面所示的场景中， INSERT FROM SELECT语句的性能会非常慢。

```
CREATE SEQUENCE newSeq1;
CREATE TABLE newT1
        (
          id   int not null default nextval('newSeq1'),
          name text
         );
INSERT INTO newT1(name) SELECT name from T1;
```

可以提高性能的写法是（假设T1表导入newT1表中的数据为10000行）：

```
INSERT INTO newT1(id, name) SELECT id,name from T1;
SELECT SETVAL('newSeq1',10000);
```

说明

序列操作函数nextval()，setval() 等均不支持回滚。另外setval设置的新值，会对当前会话的nextval立即生效，但对其他会话，如果定义了cache，不会立即生效，在用尽所有缓存的值后，其变动才被其他会话感知。所以为了避免产生重复值，要谨慎使用setval，设置的新值不能是已经产生的值或者在缓存中的值。

如果必须要在bulkload场景下产生默认序列值，则一定要为newSeq1定义足够大的cache，并且不要定义Maxvalue或者Minvalue。数据库会试图将nextval('sequence_name')的调用下推到Data Node，以提高性能。 目前GTM对并发的连接请求是有限制的，当Data Node很多时，将产生大量并发连接， 这时一定要控制bulkload的并发数目，避免耗尽GTM的连接资源。如果目标表为复制表(DISTRIBUTE BY REPLICATION)时下推将不能进行。当数据量较大时，这对数据库将是个灾难。除了性能问题之外，空间也可能会剧烈膨胀，在导入结束后，需要用vacuum full来恢复。最好的方式还是如上建议的，不要在bulkload的场景中产生默认序列值。

另外，序列创建后，在每个节点上都维护了一张单行表，存储序列的定义及当前值，但此当前值并非GTM上的当前值，只是保存本节点与GTM交互后的状态。如果其他节

 版权所有 © 华为云计算技术有限公司

点也向GTM申请了新值，或者调用了Setval修改了序列的状态，不会刷新本节点的单行表，但因每次申请序列值是向GTM申请，所以对序列正确性没有影响。

4.8 创建和管理 DWS 视图

视图允许用户保存常用的或者复杂的查询。视图在磁盘上并没有被物理存储，当用户访问视图时查询会作为一个子查询运行。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。视图每次被引用的时候都会运行一次。

创建视图

使用CREATE VIEW命令创建新视图。

```
CREATE OR REPLACE VIEW MyView AS SELECT * FROM tpcds.customer WHERE c_customer_sk < 150;
```

说明

CREATE VIEW中的OR REPLACE可有可无，当存在OR REPLACE时，表示若以前存在该视图就进行替换。

查看视图

- 查看MyView视图，查询结果为当前实时数据。

  ```
  SELECT * FROM myview;
  ```

- 查看当前用户下的视图。

  ```
  SELECT * FROM user_views;
  ```

- 查看所有视图。

  ```
  SELECT * FROM dba_views;
  ```

- 查看某视图的具体信息。

  执行如下命令查询dba_users视图的详细信息。

  ```
  \d+ dba_users
                        View "PG_CATALOG.DBA_USERS"
   Column  |         Type          | Modifiers | Storage  | Description
  ----------+-----------------------+-----------+----------+-------------
   USERNAME | CHARACTER VARYING(64) |           | extended |
  View definition:
   SELECT PG_AUTHID.ROLNAME::CHARACTER VARYING(64) AS USERNAME
     FROM PG_AUTHID;
  ```

重建视图

使用ALTER VIEW命令在不键入查询语句的情况下重建视图。

```
ALTER VIEW myview REBUILD;
```

删除视图

使用DROP VIEW命令删除一个视图。

```
DROP VIEW myview;
```

DROP VIEW ... CASCADE命令也可以级联删除依赖此视图的对象。 例如，如果A视图依赖于将要被删除的B视图，那么A视图也将被删除。 如果没有CASCADE选项，这个DROP VIEW命令将会失败。

版权所有 © 华为云计算技术有限公司

4.9 创建和管理 DWS 定时任务

DWS支持用户创建定时任务，当设置的任务时间点到达后可以自动触发任务的执行，从而可以减少用户运维的工作量。

数据库兼容Oracle定时任务功能主要通过DBMS.JOB高级包提供的接口，可以实现定时任务的创建、任务到期自动执行、任务删除、修改任务属性（包括：任务id、任务的关闭开启、任务的触发时间、触发时间间隔、任务内容等）。

说明

- 定时任务的执行语句暂不记录到系统的**实时TopSQL**日志中，仅8.2.1及以上集群版本支持记录。
- DWS 时间设置默认为UTC时间，定时任务的执行时间需根据用户所在时区差进行转换。

定时任务管理

步骤1 创建测试表：

```
CREATE TABLE test(id int, time date);
```

当结果显示为如下信息，则表示创建成功。

```
CREATE TABLE
```

步骤2 创建自定义存储过程：

```
CREATE OR REPLACE PROCEDURE PRC_JOB_1()
AS
N_NUM integer :=1;
BEGIN
FOR I IN 1..1000 LOOP
INSERT INTO test VALUES(I,SYSDATE);
END LOOP;
END;
/
```

当结果显示为如下信息，则表示创建成功。

```
CREATE PROCEDURE
```

步骤3 创建任务：

- 新创建的任务（未指定job_id）表示每隔1分钟执行一次存储过程PRC_JOB_1。

```
call dbms_job.submit('call public.prc_job_1(); ', sysdate, 'interval ''1 minute''', :a);
job
-----
1
(1 row)
```

- 指定job_id创建任务。

```
call dbms_job.isubmit(2,'call public.prc_job_1(); ', sysdate, 'interval ''1 minute''');
isubmit
---------


(1 row)
```

步骤4 通过USER_JOBS视图查看当前用户已创建的任务信息。

需要有系统管理员权限才可以访问此系统视图，字段说明详见**表18-346**。

```
select job,dbname,start_date,last_date,this_date,next_date,broken,status,interval,failures,what from
user_jobs;
 job |  dbname  |         start_date         |         last_date          |         this_date          |         next_date          |
```


版权所有 © 华为云计算技术有限公司

```
broken | status |       interval        | failures |            what

-----+----------+-----------------------------+-----------------------------+----------------------------
+--------------------+--------+--------+--------------------+----------+----------------
-----------
   1 | db_demo | 2022-03-25 07:58:01.829436 | 2022-03-25 07:58:03.174817 | 2022-03-25 07:58:01.829436 |
2022-03-25 07:59:01 | n      | s      | interval '1 minute' |        0 | call public.prc
_job_1();
   2 | db_demo | 2022-03-25 07:58:15.893383 | 2022-03-25 07:58:16.608959 | 2022-03-25 07:58:15.893383 |
2022-03-25 07:59:15 | n      | s      | interval '1 minute' |        0 | call public.prc
_job_1();
(2 rows)
```

步骤5 停止任务。

```
call dbms_job.broken(1,true);
broken
--------

(1 row)
```

步骤6 启动任务。

```
call dbms_job.broken(1,false);
broken
--------

(1 row)
```

步骤7 修改任务属性。

- 修改JOB的Next_date参数信息。例如，修改Job1的Next_date为1小时以后开始执行。

  ```
  call dbms_job.next_date(1, sysdate+1.0/24);
  next_date
  -----------

  (1 row)
  ```

- 修改JOB的Interval参数信息。例如，修改Job1的Interval为每隔1小时执行一次。

  ```
  call dbms_job.interval(1,'sysdate + 1.0/24');
  interval
  ----------

  (1 row)
  ```

- 修改JOB的What参数信息。例如，修改Job1的What为执行SQL语句“insert into public.test values(333, sysdate+5);”。

  ```
  call dbms_job.what(1,'insert into public.test values(333, sysdate+5);');
  what
  ------

  (1 row)
  ```

- 同时修改JOB的Next_date、Interval、What等多个参数信息。

  ```
  call dbms_job.change(1, 'call public.prc_job_1();', sysdate, 'interval ''1 minute''');
  change
  --------

  (1 row)
  ```

步骤8 删除JOB。

```
call dbms_job.remove(1);
remove
--------

(1 row)
```

步骤9 JOB的权限控制。

 版权所有 © 华为云计算技术有限公司

- 当创建一个JOB时，该JOB会和创建该JOB的数据库和用户绑定（即：pg_job系统视图新增的JOB记录中的dbname和log_user）。
- 如果当前用户是DBA用户、系统管理员、该JOB的创建用户（即：pg_job中的log_user），那么该用户有权限通过高级包接口remove、change、next_data、what、interval删除或修改JOB的参数信息。否则，会提示当前用户没有权限操作该JOB。
- 如果当前数据库是该JOB创建所属的数据库（即：为pg_job系统视图中的dbname），那么连接到当前数据库上可以通过高级包接口remove、change、next_data、what、interval删除或修改JOB的参数信息。
- 当删除JOB所属的数据库（即：为pg_job系统视图中的dbname）时，系统会关联删除该数据库从属的JOB记录。
- 当删除JOB所属的用户（即：为pg_job系统视图中的log_user）时，系统会关联删除该用户从属的JOB记录。

----结束

4.10 查看 DWS 系统表

除了创建的表以外，数据库还包含很多系统表。这些系统表包含集群安装信息以及DWS上运行的各种查询和进程的信息。可以通过查询系统表来收集有关数据库的信息。

查看数据库中包含的表

例如，在PG_TABLES系统表中查看public schema中包含的所有表。

```
SELECT distinct(tablename) FROM pg_tables WHERE SCHEMANAME = 'public';
```

结果类似如下这样：

```
    tablename
--------------------
 err_hr_staffs
 test
 err_hr_staffs_ft3
 web_returns_p1
 mig_seq_table
 films4
(6 rows)
```

查看数据库用户

通过PG_USER可以查看数据库中所有用户的列表，还可以查看用户ID（USESYSID）和用户权限。

```
SELECT * FROM pg_user;
usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd  | valbegin | valuntil |   respool
| parent | spacelimit | useconfig | nodegroup | tempspacelimit | spillspacelim
it
---------+----------+-------------+----------+-----------+---------+----------+----------+----------+--------------
+--------+------------+-----------+-----------+----------------+--------------
---
 Ruby    |       10 | t          | t       | t        | t       | ******** |          |          | default_pool |      0 |
|          |          |              |
 dbadmin |    16393 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 lily    |    16691 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
```


版权所有 © 华为云计算技术有限公司

```
 jack    |    70694 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
(4 rows)
```

DWS在内部使用Ruby执行日常管理和维护任务。可以向SELECT语句添加WHERE usesysid > 10来筛选查询，使其只显示用户定义的用户名称。

```
SELECT * FROM pg_user WHERE usesysid > 10;
 usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |   passwd   | valbegin | valuntil |
respool    | parent | spacelimit | useconfig | nodegroup | tempspacelimit | spillspacelim
it
---------+----------+-------------+----------+-----------+---------+----------+----------+----------+-------------
+--------+------------+-----------+-----------+----------------+--------------
---
 dbadmin |    16393 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 lily    |    16691 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 jack    |    70694 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
(3 rows)
```

查看和停止正在运行的查询语句

通过视图**PG_STAT_ACTIVITY**可以查看正在运行的查询语句。方法如下：

步骤1 设置参数track_activities为on。

```
SET track_activities = on;
```

当此参数为on时，数据库系统才会收集当前活动查询的运行信息。

步骤2 查看正在运行的查询语句。以查看正在运行的查询语句所连接的数据库名、执行查询的用户、查询状态及查询对应的PID为例：

```
SELECT datname, usename, state,pid FROM pg_stat_activity;
```

如果state字段显示为idle，则表明此连接处于空闲，等待用户输入命令。

如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pg_stat_activity WHERE state != 'idle';
```

步骤3 若需要取消运行时间过长的查询，通过PG_TERMINATE_BACKEND函数，根据线程ID结束会话。

```
SELECT PG_TERMINATE_BACKEND(139834759993104);
```

显示类似如下信息，表示结束会话成功。

```
PG_TERMINATE_BACKEND
----------------------
 t
(1 row)
```

显示类似如下信息，表示用户执行了结束当前会话的操作。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
```

 版权所有 © 华为云计算技术有限公司

说明

gsql客户端使用PG_TERMINATE_BACKEND函数结束当前会话后台线程时，客户端不会退出而是自动重连。即还会返回“The connection to the server was lost. Attempting reset: Succeeded.”

```
FATAL: terminating connection due to administrator command
FATAL: terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

----结束

 版权所有 © 华为云计算技术有限公司

5 Oracle、Teradata 和 MySQL 语法兼容性差异

DWS支持Oracle（以下简称ORA）、Teradata（以下简称TD）和MySQL三种兼容模式，分别兼容Oracle、Teradata和MySQL语法，不同兼容模式下的语法行为有一些差异。

数据库兼容模式可以在创建数据库时指定（由DBCOMPATIBILITY参数控制），语法示例如下，具体参见“CREATE DATABASE”章节。

```
CREATE DATABASE td_compatible_db DBCOMPATIBILITY 'TD'; --创建兼容Teradata的数据库
CREATE DATABASE ora_compatible_db DBCOMPATIBILITY 'ORA'; --创建兼容Oracle的数据库
CREATE DATABASE mysql_compatible_db DBCOMPATIBILITY 'MYSQL'; --创建兼容MySQL的数据库
```

查询结果：

```
SELECT datname,datcompatibility FROM PG_DATABASE WHERE datname LIKE '%compatible_db';
       datname       | datcompatibility
---------------------+------------------
 td_compatible_db    | TD
 ora_compatible_db   | ORA
 mysql_compatible_db | MYSQL
(3 rows)
```

表 5-1 兼容项差异

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
数据类型date	date会转为timestamp，包含年月日时分秒。	只有年月日。	只有年月日。
空串	只有null。	区分空串和null。	区分空串和null。
空串转数字	null。	转换为0。	转换为0。
超长字符自动截断	不支持。	支持(GUC参数td_compatible_truncation设置为ON)。	不支持。
varchar + int运算	转为bigint + int计算。	转为numeric + numeric计算。	转为bigint + int计算。

 版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
null拼接	非null对象与null拼接后返回非null对象。 例如，'abc'||null返回'abc'。	GUC参数behavior_compat_options增加strict_text_concat_td选项后，兼容TD行为，null类型拼接后返回null。 例如，'abc'||null返回null。	兼容MySQL行为，null类型拼接后返回null。 例如，'abc'||null返回null。
char(n)类型拼接	char(n)类型做拼接时移除右侧空格和占位。 例如，cast('a' as char(3))||'b'返回'ab'。	GUC参数behavior_compat_options增加bpchar_text_without_rtrim选项后，char(n)类型做拼接时，保留空格，并补足空格至指定的n长度。 当前不支持“比较字符串时忽略尾部空格”，拼接后结果如果存在尾部空格，进行比较时会对空格敏感。 例如，cast('a' as char(3))||'b'返回'a b'。	移除右侧空格和占位。
concat(str1,str2)	返回所有非null字符串的连接。	返回所有非null字符串的连接。	入参中存在null时，返回结果为null。
left和right负数处理	返回除最后/前|n|个字符以外的所有字符。	返回除最后/前|n|个字符以外的所有字符。	返回空串。
lpad(string text, length int [, fill text]) **rpad(string text, length int [, fill text])**	通过填充字符fill(缺省为空格)，把string填充到length长度，如果string已经比length长则将其尾部截断。如果fill为空串或者length为负数则返回null。	如果fill为空串且string长度小于length时，返回原字符串，如果length为负数则返回空串。	如果fill为空串且string长度小于length时，返回空串，如果length为负数则返回null。
substr(str, s[, n])	s = 0时，返回前n个字符。	s = 0时，返回前n个字符。	s = 0时，返回空串。

版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
substring(str, s[, n]) substring(str [from s] [for n])	s = 0时，返回前n - 1个字符 s < 0时，返回前s + n - 1个字符 n < 0时，报错。	s = 0时，返回前n - 1个字符。 s < 0时，返回前s + n - 1个字符。 n < 0时，报错。	s = 0时，返回空串。 s < 0时，倒数第|s|个字符位置开始截取n个字符。 n < 0时，返回空串。
trim、ltrim、rtrim、btrim(string[,characters])	从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。	从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。	从字符串string的指定位置删除等于characters的字符串（缺省为空格）。
log(x)	以10为底的对数。	以10为底的对数。	自然对数。
mod(x, 0)	除数为0时返回x。	除数为0时返回x。	除数为0时报错。
to_char(date)	入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为timestamp。	入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为date，且格式为'YYYY/MM/DD'(GUC参数convert_empty_str_to_null_td打开)。	入参最大值支持timestamp类型的最大值和date类型的最大值；返回值类型为date。
to_date, to_timestamp和to_number空串处理	返回null。	返回null(GUC参数convert_empty_str_to_null_td打开)。	to_date和to_timestamp返回null，to_number中参数为空串时，返回0。
last_day和next_day返回类型	timestamp类型。	timestamp类型。	date类型。
add_months返回类型	timestamp类型。	timestamp类型。	入参为date类型，返回date类型。 入参为timestamp类型，返回timestamp类型。 入参为timestamptz类型，返回timestamptz类型。
CURRENT_TIME CURRENT_TIME(p)	获取当前事务的时间，返回值类型为timetz。	获取当前事务的时间，返回值类型为timetz。	获取当前语句执行时的时间，返回值类型为time。

版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
CURRENT_TIMESTAMP CURRENT_TIMESTAMP(p)	获取当前语句执行时的时间，返回值类型为timestamptz。	获取当前语句执行时的时间，返回值类型为timestamptz。	获取当前语句执行时的时间，返回值类型为timestamp。
CURDATE	不支持。	不支持。	获取当前语句执行时的日期，返回值类型为date。
CURTIME(p)	不支持。	不支持。	获取当前语句执行时的时间，返回值类型为time。
LOCALTIME LOCALTIME(p)	获取当前事务的时间，返回值类型为time。	获取当前事务的时间，返回值类型为time。	获取当前语句执行时的时间，返回值类型为timestamp。
LOCALTIMESTAMP LOCALTIMESTAMP(p)	获取当前事务的时间，返回值类型为timestamp。	获取当前事务的时间，返回值类型为timestamp。	获取当前语句执行时的时间，返回值类型为timestamp。
SYSDATE SYSDATE(p)	获取当前语句执行时的时间，返回值类型为timestamp(0)。	获取当前语句执行时的时间，返回值类型为timestamp(0)。	获取当前系统的时间，返回值类型为timestamp(0)。此函数不可下推，建议用current_date代替。
now()	获取当前事务时间，返回值类型为timestamptz。	获取当前事务时间，返回值类型为timestamptz。	获取语句执行的时间，返回值类型为timestamptz。
操作符'^'	幂运算。	幂运算。	异或。
表达式greatest、least	返回所有非null入参的比较结果。	返回所有非null入参的比较结果。	入参中存在null时，返回结果为null。
表达式case、coalesce、if、ifnull入参类型不同	报错。	兼容TD行为，支持数字和字符串之间的类型转换，比如coalesce参数输入int和varchar类型，解析成varchar类型。	兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。
反引号	不支持。	不支持。	区分MySQL的保留字与普通字符。

举例说明

以下以兼容Teradata模式为例运行SQL，如果需要查看Oracle、MySQL兼容模式下的行为差异，可以切换到对应的数据库ora_compatible_db、mysql_compatible_db，然

版权所有 © 华为云计算技术有限公司

后运行以下相同的SQL语句（表名相应替换为ora_table、mysql_table），即可体会上表所列举的不同兼容模式下的行为差异。

```
CREATE TABLE td_table(a INT,b VARCHAR(5),c date);
INSERT INTO td_table VALUES(1,null,CURRENT_DATE);
INSERT INTO td_table VALUES(2,'',CURRENT_DATE);
```

区分空串和 NULL，date 类型只显示年月日

TD和MySQL模式下，区分空串和NULL，ORA模式下不区分空串和NULL，date类型会转为timestamp，包含年月日时分秒。

TD 模式

```
SELECT a, b, b IS NULL AS null, c FROM td_table;
```

```
 a | b | null |     c
---+---+------+------------
 2 |   | f    | 2025-08-12
 1 |   | t    | 2025-08-12
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
    date
------------
 2025-08-12
(1 row)
```

ORA 模式

```
SELECT a, b, b IS NULL AS null, c FROM ora_table;
```

```
 a | b | null |          c
---+---+------+---------------------
 1 |   | t    | 2025-08-12 00:00:00
 2 |   | t    | 2025-08-12 00:00:00
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
    date
------------
 2025-08-12
(1 row)
```

MySQL 模式

```
SELECT a, b, b IS NULL AS null, c FROM mysql_table;
```


版权所有 © 华为云计算技术有限公司

```
 a | b | null |     c
---+---+------+------------
 2 |   | f    | 2025-08-12
 1 |   | t    | 2025-08-12
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
 current_date
--------------
 2025-08-12
(1 row)
```

空串转 int，转为 0

TD数据库不同于Oracle，Oracle将空串当作NULL进行处理，TD在将空串转换为数值类型的时候，默认将空串转换为0进行处理，因此查询空串会查询到数值为0的数据。

同样地，在TD兼容模式下，字符串转换数值的过程中，也会将空串默认转换为相应数值类型的0值进行处理。除此之外，' - '、' + '、' '这些字符串也都会在TD兼容模式下默认转换为0进行处理，但是小数点字符串' . '会报错。

TD 模式

```
SELECT b::int FROM td_table WHERE b = '';
```

```
 b
---
 0
(1 row)
```

ORA 模式

```
SELECT b::int FROM ora_table WHERE b = '';
```

```
 b
---
(0 rows)
```

MySQL 模式

```
SELECT b::int FROM mysql_table WHERE b = '';
```

```
 b
---
 0
(1 row)
```


版权所有 © 华为云计算技术有限公司

超长字符自动截断

- TD模式下，**td_compatible_truncation**参数设置为on时，将启用超长字符串自动截断功能，在后续的insert语句中（不包含外表的场景下），对目标表中char和varchar类型的列上插入超长字符串时，系统会自动按照目标表中相应列定义的最大长度对超长字符串进行截断。
- ORA和MySQL模式下，插入超长字符串会报错。

TD 模式

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO td_table VALUES(3,'12345678',CURRENT_DATE);
SELECT * FROM td_table WHERE a = 3;
```

```
 a |   b   |     c
---+-------+------------
 3 | 12345 | 2025-08-12
(1 row)
```

ORA 模式

参数**td_compatible_truncation**设置无效，插入超长字符会报错。

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO ora_table VALUES(3,'12345678',CURRENT_DATE);
```

```
ERROR:  dn_6001_6002: value too long for type character varying(5)
CONTEXT:  referenced column: b
```

Mysql 模式

参数**td_compatible_truncation**设置无效，插入超长字符会报错。

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO mysql_table VALUES(3,'12345678',CURRENT_DATE);
```

```
ERROR:  dn_6001_6002: value too long for type character varying(5)
CONTEXT:  referenced column: b
```

varchar + int 运算，转为 numeric + numeric 计算

- TD模式下，varchar + int运算，转为numeric + numeric计算。
- ORA和MySQL模式下，转为bigint + int计算。

TD 模式

```
EXPLAIN VERBOSE SELECT b + a FROM td_table WHERE a = 3;
```


版权所有 © 华为云计算技术有限公司

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((td_table.b)::numeric + (td_table.a)::numeric))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::numeric + a::numeric AS "?column?" FROM public.td_table WHERE a = 3
```

ORA 模式

```
INSERT INTO ora_table VALUES(3,'12345',CURRENT_DATE);
EXPLAIN VERBOSE SELECT b + a FROM ora_table WHERE a = 3;
```

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((ora_table.b)::bigint + ora_table.a))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::bigint + a AS "?column?" FROM public.ora_table WHERE a = 3
```

MySQL 模式

```
INSERT INTO mysql_table VALUES(3,'12345',CURRENT_DATE);
EXPLAIN VERBOSE SELECT b + a FROM mysql_table WHERE a = 3;
```

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((mysql_table.b)::bigint + mysql_table.a))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::bigint + a AS "?column?" FROM public.mysql_table WHERE a = 3
```

null 拼接返回 null

- TD模式：GUC参数**behavior_compat_options**增加strict_text_concat_td选项，日期、时间、数字、字符串和null拼接均会返回null。
- ORA模式：非null对象与null拼接后返回非null对象。例如，'abc'||null返回'abc'。
- MySQL模式：兼容MySQL行为，null类型拼接后返回null。例如，'abc'||null返回null。

版权所有 © 华为云计算技术有限公司

TD 模式

```
SET behavior_compat_options = 'strict_text_concat_td';
SELECT '2024-02-07 12:12:12'::TIMESTAMP || NULL;
SELECT '12:12:12'::TIME || NULL;
SELECT '12'::TINYINT || NULL;
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------

(1 row)
```

ORA 模式

```
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------
 abc
(1 row)
```

MySQL 模式

```
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------

(1 row)
```

char(n)类型拼接

- TD模式下，GUC参数**behavior_compat_options**增加bpchar_text_without_rtrim选项，char(n)类型做拼接时，保留空格，并补足空格至指定的n长度。
- ORA和MySQL模式下，不保留空格。

TD 模式

例如，a后面带3个空格，强制转换成char(10)后，由于不足10个字符，则会用空格填充到10个字符，占用10个字节的存储空间，每个字节是8位，所以结果是80位。

```
SET behavior_compat_options = 'bpchar_text_without_rtrim';
SELECT bit_length('a   '::char(10));
```

```
 bit_length
------------
         80
(1 row)
```


版权所有 © 华为云计算技术有限公司

ORA 和 MySQL 模式

ORA和MySQL模式下，不保留空格，a只占了1个字符，占用1个字节的存储空间，所以结果是8。

```
SELECT bit_length('a   '::char(10));
```

```
 bit_length
------------
          8
(1 row)
```

concat(str1,str2)

TD和ORA模式下，返回所有非null字符串的连接。MySQL模式下，入参中存在null时，返回结果为null。

TD 和 ORA 模式

```
SELECT concat(null, 'World!');
```

```
 concat
--------
 World!
(1 row)
```

MySQL 模式

```
SELECT concat(null, 'World!');
```

```
 concat
--------

(1 row)
```

left 和 right 负数处理

TD和ORA兼容模式下，left和right函数分别返回除最后、最前|n|个字符以外的所有字符。MySQL模式下，返回空串。

TD 和 ORA 模式

```
SELECT left('abcde', -2);
```

```
 left
------
 abc
(1 row)
```

```
SELECT right('abcde', -2);
```


版权所有 © 华为云计算技术有限公司

```
 right
-------
 cde
(1 row)
```

MySQL 模式

```
SELECT left('abcde', -2);
```

```
 left
------


(1 row)
```

```
SELECT right('abcde', -2);
```

```
 right
-------


(1 row)
```

lpad 和 rpad 空串处理

- lpad和rpad函数在TD兼容模式下：如果fill为空串且string长度小于length时，返回原字符串。如果length为负数则返回空串。
- ORA模式下，都返回null。
- MySQL模式下，如果fill为空串且string长度小于length时，返回空串，如果length为负数则返回null。

TD 模式

```
SELECT lpad('a',7,'');
```

```
 lpad
------
 a
(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------


(1 row)
```

```
SELECT rpad('a',7,'');
```


版权所有 © 华为云计算技术有限公司

```
 rpad
------
 a
(1 row)
```

```
SELECT rpad('a',-7,'');
```

```
 rpad
------

(1 row)
```

ORA 模式

ORA模式下，返回null。 rpad示例类似，不再赘述。

```
SELECT lpad('a',7,'');
```

```
 lpad
------

(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------

(1 row)
```

MySQL 模式

```
SELECT lpad('a',7,'');
```

```
 lpad
------

(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------

(1 row)
```


版权所有 © 华为云计算技术有限公司

substr 返回子串

substr(str, s[, n])函数在TD和ORA模式下，s = 0时，返回前n个字符。在MySQL模式下，返回空串。

TD 和 ORA 模式

```
SELECT SUBSTR('Hello, World!', 0, 5);
```

```
 substr
--------
 Hello
(1 row)
```

MySQL 模式

```
SELECT SUBSTR('Hello, World!', 0, 5);
```

```
 substr
--------

(1 row)
```

substring 返回子串

substring(str [from s] [for n])函数在TD和ORA模式下：

- s = 0时，返回前n - 1个字符。
- s < 0时，返回前s + n - 1个字符。
- n < 0时，报错。

MySQL模式下：

- s = 0时，返回空串。
- s < 0时，倒数第|s|个字符位置开始截取n个字符。
- n < 0时，返回空串。

TD 和 ORA 模式

示例1：s=0，n=5时返回'Hello, World!'前4个字符。

```
SELECT SUBSTRING('Hello, World!' FROM 0 FOR 5);
```

```
 substring
-----------
 Hell
(1 row)
```

示例2：s=-1，n=4时返回'Hello, World!'前两个字符。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR 4);
```


版权所有 © 华为云计算技术有限公司

```
 substring
-----------
 He
(1 row)
```

示例3：n=-1时直接报错。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR -1);
```

```
ERROR:  negative substring length not allowed
CONTEXT:  referenced column: substring
```

MySQL 模式

示例1：s=0，n=5时返回空串。

```
SELECT SUBSTRING('Hello, World!' FROM 0 FOR 5);
```

```
 substring
-----------

(1 row)
```

示例2：s=-1，n=4时返回最后一个字符。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR 4);
```

```
 substring
-----------
 !
(1 row)
```

示例3：n=-1时返回空串。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR -1);
```

```
 substring
-----------

(1 row)
```

btrim(string[,characters])

- TD和ORA模式下，从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。
- MySQL模式下，从字符串string的指定位置删除等于characters的字符串（缺省为空格）。

TD 和 ORA 模式

```
SELECT BTRIM('xxHello Worldxx', 'xxz');
```


版权所有 © 华为云计算技术有限公司

```
    btrim
-------------
 Hello World
(1 row)
```

MySQL 模式

```
SELECT BTRIM('xxHello Worldxx', 'xxz');
```

```
      btrim
-----------------
 xxHello Worldxx
(1 row)
```

log(x)对数运算

- TD和ORA模式下log(x)表示以10为底的对数，例如log(100)等同于log(10,100)。
- MySQL模式下表示自然对数，即以e为底（e≈*2.71828*）的对数，例如log(100)约等同于log(2.71828,100)。

TD 和 ORA 模式

```
SELECT log(100);
```

```
 log
-----
   2
(1 row)
```

MySQL 模式

```
SELECT log(100);
```

```
       log
------------------
 4.60517018598809
(1 row)
```

mod(x,0)取模运算

TD和ORA模式下mod(x)除数为0时返回x。MySQL模式下报错。

TD 和 ORA 模式

```
SELECT mod(3,0);
```

```
 mod
-----
   3
(1 row)
```


版权所有 © 华为云计算技术有限公司

MySQL 模式

```
SELECT mod(3,0);
```

```
ERROR:  division by zero
CONTEXT:  referenced column: mod
```

to_char(date)

- TD模式下，入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为date，且格式为'YYYY/MM/DD'(GUC参数behavior_compat_options配置成convert_empty_str_to_null_td)。
- ORA模式下，入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为timestamp。
- MySQL模式下，入参最大值支持timestamp类型的最大值和date类型的最大值；返回值类型为date。

TD 模式

```
SET behavior_compat_options = convert_empty_str_to_null_td;
SELECT TO_CHAR(DATE '294276-12-31');
```

```
   to_char
--------------
 294276/12/31
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
ERROR:  date out of range for timestamp
CONTEXT:  referenced column: to_char
```

ORA 模式

```
SELECT TO_CHAR(DATE '294276-12-31');
```

```
        to_char
-----------------------
 294276-12-31 00:00:00
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
ERROR:  timestamp out of range: "5874897-12-31"
LINE 1: SELECT TO_CHAR(DATE '5874897-12-31');
                            ^
CONTEXT:  referenced column: to_char
```

MySQL 模式

```
SELECT TO_CHAR(DATE '294276-12-31');
```


版权所有 © 华为云计算技术有限公司

```
   to_char
--------------
 294276-12-31
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
    to_char
---------------
 5874897-12-31
(1 row)
```

to_date, to_timestamp 和 to_number 空串处理

- TD兼容模式下，GUC参数**behavior_compat_options**配置成convert_empty_str_to_null_td参数，这些函数都返回null。
- ORA模式下，都返回null。
- MySQL模式下，to_date和to_timestamp返回null，to_number中参数为空串时，返回0。

TD 和 ORA 模式

```
SET behavior_compat_options = 'convert_empty_str_to_null_td'; --仅对TD模式生效。
SELECT TO_DATE('');
SELECT TO_TIMESTAMP('');
SELECT TO_NUMBER('');
```

```
 to_number
-----------


(1 row)
```

MySQL 模式

```
SELECT TO_DATE('');
SELECT TO_TIMESTAMP('');
SELECT TO_NUMBER('');
```

```
 to_date
---------


(1 row)
```

```
 to_timestamp
--------------


(1 row)
```


版权所有 © 华为云计算技术有限公司

```
 to_number
-----------
         0
(1 row)
```

last_day 和 next_day

- TD和ORA模式下，返回均为timestamp类型。
- MySQL模式下，返回date类型。

TD 和 ORA 模式

```
SELECT last_day(to_date('2024-02-07', 'YYYY-MM-DD')) AS cal_result;
```

```
     cal_result
---------------------
 2024-02-29 00:00:00
(1 row)
```

```
SELECT next_day(TIMESTAMP '2024-02-07 00:00:00','Sunday')AS cal_result;
```

```
     cal_result
---------------------
 2024-02-11 00:00:00
(1 row)
```

MySQL 模式

```
SELECT last_day(to_date('2024-02-07', 'YYYY-MM-DD')) AS cal_result;
```

```
 cal_result
------------
 2024-02-29
(1 row)
```

```
SELECT next_day(TIMESTAMP '2024-02-07 00:00:00','Sunday')AS cal_result;
```

```
 cal_result
------------
 2024-02-11
(1 row)
```

add_months 时间加月

TD和ORA模式下，返回均为timestamp类型加月。

MySQL模式下：

- 入参为date类型，返回date类型。

版权所有 © 华为云计算技术有限公司

- 入参为timestamp类型，返回timestamp类型。
- 入参为timestamptz类型，返回timestamptz类型。

TD 和 ORA 模式

```
SELECT add_months('2024-02-07'::date,3);
```

```
     add_months
---------------------
 2024-05-07 00:00:00
(1 row)
```

MySQL 模式

```
SELECT add_months('2024-02-07'::date,3);
```

```
 add_months
------------
 2024-05-07
(1 row)
```

```
SELECT add_months('2024-02-07 00:00:00',3);
```

```
        add_months
-------------------------
 2024-05-07 00:00:00+08
(1 row)
```

操作符'^'

- TD和ORA模式下，表示幂运算。
- MySQL模式下，表示异或。

TD 和 ORA 模式

```
SELECT 10^3;
```

```
 ?column?
----------
     1000
(1 row)
```

MySQL 模式

```
SELECT 10^3;
```

```
 ?column?
----------
        9
(1 row)
```


版权所有 © 华为云计算技术有限公司

表达式 greatest、least

- TD和ORA模式下，返回所有非null入参的比较结果。
- MySQL模式下，入参中存在null时，返回结果为null。

TD 和 ORA 模式

```
SELECT greatest(1,2,3),least(1,2,3),greatest(1,null,3),least(1,null,3);
```

```
 greatest | least | greatest | least
----------+-------+----------+-------
        3 |     1 |        3 |     1
(1 row)
```

MySQL 模式

```
SELECT greatest(1,2,3),least(1,2,3),greatest(1,null,3),least(1,null,3);
```

```
 greatest | least | greatest | least
----------+-------+----------+-------
        3 |     1 |          |
(1 row)
```

case 和 coalesce 表达式

- TD模式：兼容TD行为，支持数字和字符串之间的类型转换，比如coalesce参数输入int和varchar类型，解析成varchar类型。
- ORA模式：报错。
- MySQL模式：兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。

TD 模式

对于case和coalesce，在TD兼容模式下的处理：

- 如果所有输入都是相同的类型，并且不是unknown类型，那么解析成这种类型。
- 如果所有输入都是unknown类型则解析成text类型。
- 如果输入字符串（包括unknown，unknown当text来处理）和数字类型，那么解析成字符串类型，如果是其他不同的类型范畴，则报错。
- 如果输入类型是同一个类型范畴，则选择该类型的优先级较高的类型。
- 把所有输入转换为所选的类型。如果从给定的输入到所选的类型没有隐式转换则失败。

示例1：Union中的待定类型解析。这里，unknown类型文本'b'将被解析成text类型。

```
SELECT text 'a' AS "text" UNION SELECT 'b';
```

```
 text
------
 a
 b
(2 rows)
```


版权所有 © 华为云计算技术有限公司

示例2：简单Union中的类型解析。文本1.2的类型为numeric，而且integer类型的1可以隐含地转换为numeric，因此使用这个类型。

```
SELECT 1.2 AS "numeric" UNION SELECT 1;
```

```
 numeric
---------
       1
     1.2
(2 rows)
```

示例3：转置Union中的类型解析。这里，因为类型real不能被隐含转换成integer，但是integer可以隐含转换成real，那么联合的结果类型将是real。

```
SELECT 1 AS "real" UNION SELECT CAST('2.2' AS REAL);
```

```
 real
------
    1
  2.2
(2 rows)
```

示例4：TD模式下，coalesce参数输入int和varchar类型，那么解析成varchar类型，ORA模式下会报错。查看coalesce参数输入int和varchar类型的查询语句的执行计划如下。

```
EXPLAIN VERBOSE select coalesce(a, b) FROM td_table;
```

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                        Targetlist Information (identified by plan id)
-----------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (COALESCE((td_table.a)::character varying, td_table.b))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT COALESCE(a::character varying, b) AS "coalesce" FROM public.td_table
```

ORA 模式

TD模式下，coalesce参数输入int和varchar类型，解析成varchar类型，ORA模式下会报错。查看coalesce参数输入int和varchar类型的查询语句的执行计划如下。

```
EXPLAIN VERBOSE select coalesce(a, b) FROM ora_table;
```

```
ERROR:  COALESCE types integer and character varying cannot be matched
LINE 1: EXPLAIN VERBOSE select coalesce(a, b) FROM ora_table;
                                                   ^
CONTEXT:  referenced column: coalesce
```


版权所有 © 华为云计算技术有限公司

MySQL 模式

兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。

版权所有 © 华为云计算技术有限公司

6 DWS 数据库安全管理

6.1 DWS 用户及权限管理

6.1.1 DWS 数据库用户类型

在非三权分立模式下，DWS支持管理员用户和普通用户两种类型的数据库账号。三权分立下，用户类型和权限请参考**DWS三权分立**。

- 管理员用户可以管理所有普通用户和数据库。
- 普通用户可连接和访问数据库，并在授权后进行特定的数据库操作以及执行SQL语句。

在用户登录DWS数据库时会对其进行身份验证。用户可以拥有数据库和数据库对象（例如表），并且可以向用户和角色授予对这些对象的权限以控制谁可以访问哪个对象。除管理员外，具有CREATEDB属性的用户可以创建数据库并授予对这些数据库的权限。

数据库用户类型

表 6-1 数据库用户类型

用户类型	描述	可进行的操作	如何创建
管理员dbadmin	管理员也称作系统管理员，是指具有SYSADMIN属性的账户。	非三权分立模式下，拥有系统的最高权限，能够执行所有的操作。系统管理员具有与对象所有者相同的权限。	• 在DWS 管理控制台创建集群时创建的用户dbadmin是系统管理员。 • 使用CREATE USER或ALTER USER语法创建和设置管理员用户。 CREATE USER *sysadmin* WITH SYSADMIN password '*{Password}*'; ALTER USER *u1* SYSADMIN;

 版权所有 © 华为云计算技术有限公司

用户类型	描述	可进行的操作	如何创建
普通用户	普通用户	• 使用工具连接数据库。 • 拥有数据库系统特定操作的属性，如CREATEDB、CREATEROLE、SYSADMIN。 • 访问数据库对象。 • 执行SQL语句。	使用CREATE USER语法创建普通用户。 CREATE USER *u1* PASSWORD '*{Password}*';
	私有用户	在非三权分立模式下，创建的具有INDEPENDENT属性的私有用户。 数据库管理员在未经其授权前，只能进行控制操作（DROP、ALTER、TRUNCATE），无权进行INSERT、DELETE、SELECT、UPDATE、COPY、GRANT、REVOKE、ALTER OWNER操作。	使用CREATE USER语法创建私有用户。 CREATE USER *user_independent* WITH INDEPENDENT IDENTIFIED BY '*{Password}*';

6.1.2 DWS 数据库用户管理

使用CREATE USER和ALTER USER可以创建和管理数据库用户。

- 非**三权分立**下，DWS用户账户只能由系统管理员或拥有CREATEROLE属性的安全管理员创建和删除。
- 三权分立时，用户账户只能由安全管理员创建。

创建用户

CREATE USER语句用于创建新的DWS用户。创建新用户后，可以使用该用户连接数据库。

- 创建普通用户u1，并设置用户拥有CREATEDB属性。
  CREATE USER *u1* WITH *CREATEDB* PASSWORD *'{Password}'*;
- 创建系统管理员mydbadmin，需指定参数SYSADMIN 。
  CREATE USER *mydbadmin* sysadmin PASSWORD '*{Password}*';
- 通过视图**PG_USER**查看已创建的用户。
  SELECT * FROM pg_user;
- 要查看用户属性，请查询系统表**PG_AUTHID**。
  SELECT * FROM pg_authid;

修改用户属性

ALTER USER语句用于更改用户属性（例如，更改用户密码或权限等内容）。

版权所有 © 华为云计算技术有限公司

示例：

- 用户u1重命名为u2：
  ```
  ALTER USER u1 RENAME TO u2;
  ```
- 为用户u1赋予CREATEROLE权限：
  ```
  ALTER USER u1 CREATEROLE;
  ```
- 修改用户密码可参考**密码设置和修改**。

锁定用户

ALTER USER语句中ACCOUNT LOCK | ACCOUNT UNLOCK参数用于锁定或者解锁用户，被锁定的用户不允许登录。若管理员发现某账户被盗、非法访问等异常情况，可手动锁定该账户；当管理员认为账户恢复正常后，可手动解锁该账户。

示例：

- 锁定用户u1：
  ```
  ALTER USER u1 ACCOUNT LOCK;
  ```
- 解锁用户u1：
  ```
  ALTER USER u1 ACCOUNT UNLOCK;
  ```

删除用户

DROP USER语句用于删除一个或多个DWS用户。当确认账户不再使用，管理员可以删除用户账户。用户删除后不可恢复。

- 同时删除多个用户时，用","隔开。
- 成功删除用户后，该用户的所有权限也会被一同删除。
- 当删除的用户正处于活动状态时，此会话状态不会立马断开，用户在会话状态断开后才会被完全删除。
- DROP USER语句指定CASCADE时，可级联删除依赖用户的表等对象。即删除owner是该用户的对象，并清理掉其他对象对该用户的授权信息。

示例：

- 删除用户u1：
  ```
  DROP USER u1;
  ```
- 级联删除账户u2：
  ```
  DROP USER u2 CASCADE;
  ```

6.1.3 自定义 DWS 密码策略

创建用户或修改用户时需要指定密码，DWS有默认的密码复杂度要求，也支持用户自定义数据库账户密码策略。

DWS 默认密码策略

DWS默认进行密码复杂度校验，默认密码策略要求如下：

- 长度为8~32个字符。
- 至少包含大写字母、小写字母、数字或特殊字符中三种的组合。
- 不能是用户名和用户名反序，此条要求为非大小写敏感。
- 不能是当前密码、当前密码的反序。

版权所有 © 华为云计算技术有限公司

自定义密码策略

密码策略包含：密码复杂度要求、密码有效期、密码重用设置以及密码的加密方式及密码重试与锁定，不同的策略项由对应的GUC参数控制，参见下表（详细内容也可参考**安全和认证（postgresql.conf）**）：

表 6-2 自定义密码策略及对应 GUC 参数

密码策略	对应参数名称	参数描述	参数取值范围	DWS默认值
是否进行密码复杂度检查	password_policy	创建或者修改DWS账户时，该参数决定是否进行密码复杂度检查。	整型，0、1 ● 0表示不采用任何密码复杂度策略。设置为0会存在安全风险，不建议设置为0 ● 1表示采用默认密码复杂度校验策略。	1
密码复杂度要求	password_min_length	密码的最小长度。	整型，6~999	8
	password_max_length	密码的最大长度。	整型，6~999	32
	password_min_uppercase	包含大写字母（A-Z）的最少个数。	整型，0~999 ● 0表示没有限制。 ● 1~999表示创建账户所指定的密码中至少需要包含大写字母个数。	0
	password_min_lowercase	包含小写字母（a-z）的最少个数。	整型，0~999 ● 0表示没有限制。 ● 1~999表示创建账户所指定的密码中至少需要包含小写字母个数。	0
	password_min_digital	包含数字（0-9）的最少个数。	整型，0~999 ● 0表示没有限制。 ● 1~999表示创建账户所指定的密码中至少需要包含数字个数。	0
	password_min_special	包含特殊字符的最少个数（特殊字符的列表请参见**表6-3**）。	整型，0~999 ● 0表示没有限制。 ● 1~999表示创建账户所指定的密码中至少需要包含特殊字符个数。	0

版权所有 © 华为云计算技术有限公司

密码策略	对应参数名称	参数描述	参数取值范围	DWS默认值
密码有效期	password_effect_time	提示用户密码是否过期。应与创建或修改用户时指定的密码有效期限保持一致。当修改密码时间加password_effect_time小于当前时间时，提示用户密码已过期。	浮点型，0~999，单位为天。 ● 0表示不开启密码有效期限制功能。 ● 1~999表示创建账户所指定的密码有效期，临近或超过有效期系统会提示用户修改密码。	90
	password_notify_time	密码到期前提醒的天数。	整型，0~999，单位为天。 ● 0表示不开启提醒功能。 ● 1~999表示账户密码到期前提醒的天数。	7
密码重用设置	password_reuse_time	不可重用天数。	浮点型，0~3650，单位为天。 ● 0表示不检查密码可重用天数。 ● 正数表示新密码不能为该值指定的天数内使用过的密码。	60
	password_reuse_max	不可重用次数。	整型，0~1000 ● 0表示不检查密码可重用次数。 ● 正整数表示新密码不能为该值指定的次数内使用过的密码。	0

版权所有 © 华为云计算技术有限公司

密码策略	对应参数名称	参数描述	参数取值范围	DWS默认值
加密方式	password_encryption_type	采用何种加密方式对用户密码进行加密存储。	0、1、2 ● 0表示采用md5方式对密码加密。密码存储方式为md5加密后密文。不建议用户使用。 ● 1表示采用sha256方式对密码加密，兼容postgres客户端的md5用户认证方式。密码存储方式为md5+sha256综合后的密文。 ● 2表示采用sha256方式对密码加密。密码存储方式为sha256加密后密文。	1
重试与锁定	password_lock_time	账户被锁定后自动解锁的时间	浮点型，0~365，单位为天。 ● 0表示密码验证失败时，自动锁定功能不生效。 ● 正数表示账户被锁定后，当锁定时间超过password_lock_time设定的值时，账户将会被自行解锁。 **说明** 参数password_lock_time的整数部分表示天数，小数部分可以换算成时、分、秒。	1
	failed_login_attempts	如果输入密码错误的次数达到failed_login_attempts则当前账户被锁定，password_lock_time秒后被自动解锁。	整型，0~1000 ● 0表示自动锁定功能不生效。 ● 正整数表示当错误密码次数达到failed_login_attempts设定的值时，当前账户将被锁定。	10

表 6-3 特殊字符

编号	字符	编号	字符	编号	字符	编号	字符
1	~	9	*	17	|	25	<

版权所有 © 华为云计算技术有限公司

编号	字符	编号	字符	编号	字符	编号	字符
2	!	10	(	18	[	26	.
3	@	11	)	19	{	27	>
4	#	12	-	20	}	28	/
5	$	13	_	21	]	29	?
6	%	14	=	22	;	-	-
7	^	15	+	23	:	-	-
8	&	16	\	24	,	-	-

示例二：配置密码有效期password_effect_time参数。

1. 使用如下命令连接数据库。

   ```
   gsql -d postgres -p 25308
   ```

   postgres为需要连接的数据库名称，25308为CN的端口号。

   连接成功后，系统显示类似如下信息：

   ```
   gsql ((GaussDB x.x.x build 39137c2d) compiled at 2022-04-01 15:43:11 commit 3629 last mr 5138 release)
   Non-SSL connection (SSL connection is recommended when requiring high-security)
   Type "help" for help.

   postgres=#
   ```

2. 查看已配置的参数。

   ```
   SHOW password_effect_time;
    password_effect_time
   ----------------------
    90
   (1 row)
   ```

   如果显示结果不为90，执行“\q”命令退出数据库。

3. 执行如下命令设置成默认值90（不建议设置为0）。

   ```
   gs_guc reload -Z coordinator -Z datanode -N all -I all -c "password_effect_time = 90"
   ```

密码设置和修改

- 建议系统管理员和普通用户都要定期修改自己的账户密码，避免账户密码被非法窃取。

  以修改用户user1密码为例，使用管理员用户连接数据库并执行如下命令：

  ```
  ALTER USER user1 IDENTIFIED BY 'newpassword' REPLACE 'oldpassword';
  ```

  说明

  密码要符合**DWS默认密码策略**，否则会执行失败。

- 管理员可以修改自己的或者其他账户的密码。通过修改其他账户的密码，解决用户密码遗失所造成无法登录的问题。

  以修改用户joe账户密码为例，命令格式如下：

  ```
  ALTER USER joe IDENTIFIED BY 'password';
  ```


版权所有 © 华为云计算技术有限公司

说明

- 系统管理员之间不允许互相修改对方密码。
- 系统管理员可以修改普通用户密码且不需要用户原密码。
- 系统管理员可以修改自己的密码但需要管理员原密码。

- 密码验证

  设置当前会话的用户和角色时，需要验证密码。如果输入密码与用户的存储密码不一致，则会报错。

  以设置用户joe为例，命令格式如下：

  ```
  SET ROLE joe PASSWORD 'password';
  ```

  显示如下命令表示设置成功：

  ```
  SET ROLE
  ```

6.1.4 DWS 数据库权限管理

权限概述

权限表示用户访问某个数据库对象（包括模式、表、函数、序列等）的操作（包括增、删、改、查、创建等）是否被允许。

DWS中的权限管理分为三种场景：

- 系统权限

  系统权限又称为用户属性，包括SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN和LOGIN。

  系统权限一般通过CREATE/ALTER ROLE语法来指定。其中，SYSADMIN权限可以通过GRANT/REVOKE ALL PRIVILEGE授予或撤销。但系统权限无法通过ROLE和USER的权限被继承，也无法授予PUBLIC。

- 数据对象权限

  将数据库对象（表和视图、指定字段、数据库、函数、模式等）的相关权限授予特定角色或用户。GRANT命令将数据库对象的特定权限授予一个或多个角色。这些权限会追加到已有的权限上。

- 用户权限

  将一个角色或用户的权限授予一个或多个其他角色或用户。在这种情况下，每个角色或用户都可视为拥有一个或多个数据库权限的集合。

  当声明了WITH ADMIN OPTION，被授权的用户可以将该权限再次授予其他角色或用户，以及撤销所有由该角色或用户继承到的权限。当授权的角色或用户发生变更或被撤销时，所有继承该角色或用户权限的用户拥有的权限都会随之发生变更。

  数据库系统管理员可以给任何角色或用户授予/撤销任何权限。拥有CREATEROLE权限的角色可以赋予或者撤销任何非系统管理员角色的权限。

层级权限管理

DWS通过Database、Schema和数据对象权限实现层级权限管理。

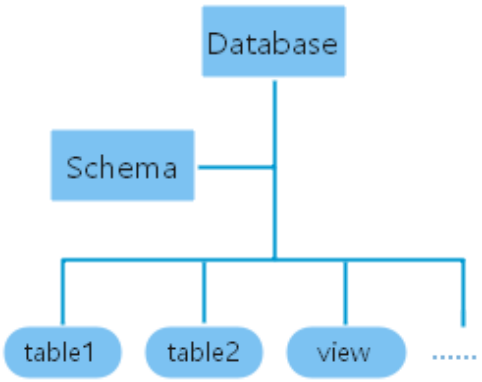
- Database之间无法直接互访，通过连接隔离实现彻底的权限隔离。各个Database之间共享资源极少，可实现连接隔离、权限隔离等。数据库集群包含一个或多个已命名数据库。用户和角色在整个集群范围内是共享的，但是其数据并不共享。

版权所有 © 华为云计算技术有限公司

即用户可以连接任何数据库，但当连接成功后，任何用户都只能访问连接请求里所声明的数据库。

- Schema隔离的方式共用资源较多，可以通过GRANT与REVOKE语法便捷地控制不同用户对各Schema及其下属对象的权限，从而赋给业务更多的灵活性。每个数据库包括一个或多个Schema。每个Schema包含表、函数等其他类型的对象。用户要访问包含在指定Schema中的对象，需要被授予Schema的USAGE权限。
- 对象创建后，默认只有对象所有者或者系统管理员可以查询、修改和删除对象。其他用户要访问包含具体的数据库对象，例如table1，需要首先被授予database的CONNECT权限，再被授予Schema的USAGE权限，最后授予table1的SELECT权限。用户要访问底层的对象，必须先赋予上层对象的权限。比如用户要创建或者删除Schema，需要首先被授予database的CREATE权限；

图 6-1 层级权限管理



角色

DWS的权限管理模型，是一种典型的RBAC（基于角色的权限控制）的实现。其将用户、角色、权限通过此模型管理起来。

角色是一组权限的集合。

- “用户”概念和“角色”概念实际是等同的，唯一的区别在于“用户”拥有login权限，而“角色”拥有nologin权限。
- 按照数据库系统中承担的责任划分具有不同权限的角色。角色是数据库权限的集合，代表了一个数据库用户、或一组数据用户的行为约束。
- 角色和用户可以转换，通过ALTER将角色拥有登录权限。
- 通过GRANT把角色授予用户后，用户即具有了角色的所有权限。推荐使用角色进行高效权限分配。例如，可以为设计、开发和维护人员创建不同的角色，将角色GRANT给用户后，再向每个角色中的用户授予其所需数据的差异权限。在角色级别授予或撤销权限时，这些权限更改会对角色下的所有成员生效。
- 非三权分立时，只有系统管理员和具有CREATEROLE属性的用户才能创建、修改或删除角色。三权分立下，只有具有CREATEROLE属性的用户才能创建、修改或删除角色。

要查看所有角色，请查询系统表PG_ROLES：

```
SELECT * FROM PG_ROLES;
```

具体的创建，修改和删除角色操作，可参考《数据仓库服务SQL语法参考》中“CREATE ROLE/ALTER ROLE/DROP ROLE”章节。

版权所有 © 华为云计算技术有限公司

预置角色

DWS提供了一组预置角色，以“gs_role_”开头命名，提供对特定的、通常需要高权限的操作的访问，可以将这些角色授权予数据库中的其他用户或角色，使这些用户能够访问或使用特定的信息和功能。请谨慎使用预置角色，以确保预置角色权限的安全使用。

预置角色允许的权限范围可参考下表：

表 6-4 预置角色允许的权限范围

角色	权限描述
gs_role_signal_backend	具有调用函数pg_cancel_backend、pg_terminate_backend、pg_terminate_query、pg_cancel_query、pgxc_terminate_query、pgxc_cancel_query来取消或终止其他会话的权限，但不能操作属于初始用户的会话。
gs_role_read_all_stats	读取系统状态视图并且使用与扩展相关的各种统计信息，包括有些通常只对系统管理员可见的信息。包括： 资源管理类： ● pgxc_wlm_operator_history ● pgxc_wlm_operator_info ● pgxc_wlm_operator_statistics ● pgxc_wlm_session_info ● pgxc_wlm_session_statistics ● pgxc_wlm_workload_records ● pgxc_workload_sql_count ● pgxc_workload_sql_elapse_time ● pgxc_workload_transaction 状态信息类： ● pgxc_stat_activity ● pgxc_get_table_skewness ● table_distribution ● pgxc_total_memory_detail ● pgxc_os_run_info ● pg_nodes_memory ● pgxc_instance_time ● pgxc_redo_stat
gs_role_analyze_any	具有系统级ANALYZE权限类似系统管理员用户，跳过schema权限检查，对所有的表可以执行ANALYZE。
gs_role_vacuum_any	具有系统级VACUUM权限类似系统管理员用户，跳过schema权限检查，对所有的表可以执行VACUUM。

版权所有 © 华为云计算技术有限公司

角色	权限描述
gs_redaction_policy	具有创建、修改、删除脱敏策略的权限，对所有的表都可以执行CREATE | ALTER | DROP REDACTION POLICY。9.1.0及以上集群版本支持。

预置角色的使用约束：

- 以gs_role_开头的角色名作为数据库的预置角色保留字，禁止新建以“gs_role_”开头的用户/角色，也禁止将已有的用户/角色重命名为以“gs_role_”开头。
- 禁止对预置角色执行ALTER和DROP操作。
- 预置角色默认没有LOGIN权限，不设置预置登录密码。
- gsql元命令\du和\dg不显示预置角色的相关信息，但若指定了PATTERN（用来指定要被显示的对象名称）则预置角色信息会显示。
- 三权分立关闭时，系统管理员和具有预置角色ADMIN OPTION权限的用户有权对预置角色执行GRANT/REVOKE管理；三权分立打开时，安全管理员（具有CREATEROLE属性）和具有预置角色ADMIN OPTION权限的用户有权对预置角色执行GRANT/REVOKE管理。例如：

```
GRANT gs_role_signal_backend TO user1;
REVOKE gs_role_signal_backend FROM user1;
```

权限授予或撤销

数据库对象创建后，进行对象创建的用户就是该对象的所有者。集群安装后的默认情况下，未开启三权分立，数据库系统管理员具有与对象所有者相同的权限。

也就是说对象创建后，默认只有对象所有者或者系统管理员可以查询、修改和删除对象，以及通过GRANT将对象的权限授予其他用户。为使其他用户能够使用对象，可以由对象所有者或管理员通过GRANT/REVOKE对其他用户或角色授予与撤销。

- 使用GRANT语句授予权限。

  例如，将模式myschema的权限赋给角色u1后，将表myschema.t1的SELECT权限授予角色u1。

  ```
  GRANT USAGE ON SCHEMA myschema TO u1;
  GRANT SELECT ON TABLE myschema.t1 to u1;
  ```

- 使用REVOKE撤销已经授予的权限。

  例如：撤销用户u1在指定表myschema.t1上的所有权限。

  ```
  REVOKE ALL PRIVILEGES ON myschema.t1 FROM u1;
  ```

6.1.5 DWS 三权分立

默认情况下拥有SYSADMIN属性的系统管理员，具备系统最高权限。在实际业务管理中，为了避免系统管理员拥有过度集中的权利带来高风险，可以设置三权分立，将系统管理员的权限分立给安全管理员和审计管理员。

- 三权分立后，系统管理员将不再具有CREATEROLE属性（安全管理员）和AUDITADMIN属性（审计管理员）能力。即不再拥有创建角色和用户的权限，并不再拥有查看和维护数据库审计日志的权限。关于CREATEROLE属性和AUDITADMIN属性的更多信息请参考《数据仓库服务SQL语法参考》中“CREATE ROLE”章节。

版权所有 © 华为云计算技术有限公司

- 三权分立后，系统管理员只会对自己作为所有者的对象有权限。

三权分立的设置办法请参考《 数据仓库服务用户指南 》中的“设置DWS集群三权分立”章节。

三权分立前的权限详情及三权分立后的权限变化，请分别参见**表6-5**和**表6-6**。

表 6-5 默认的用户权限

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	对表空间有创建、修改、删除、访问、分配操作的权限。	不具有对表空间进行创建、修改、删除、分配的权限，访问需要被赋权。		
表	对所有表有所有的权限。	仅对自己的表有所有的权限，对其他用户的表无权限。		
索引	可以在所有的表上建立索引。	仅可以在自己的表上建立索引。		
模式	对所有模式有所有的权限。	仅对自己的模式有所有的权限，对其他用户的模式无权限。		
函数	对所有的函数有所有的权限。	仅对自己的函数有所有的权限，对其他用户放在public这个公共模式下的函数有调用的权限，对其他用户放在其他模式下的函数无权限。		
自定义视图	对所有的视图有所有的权限。	仅对自己的视图有所有的权限，对其他用户的视图无权限。		
系统表和系统视图	可以查看所有系统表和视图。	只可以查看部分系统表和视图。详细请参见**DWS系统表和系统视图**。		

表 6-6 三权分立较非三权分立权限变化说明

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	无变化	无变化。		
表	权限缩小。 只对自己的表有所有权限，对其他用户放在属于各自模式下的表无权限。	无变化。		
索引	权限缩小。 只可以在自己的表上建立索引。	无变化。		

版权所有 © 华为云计算技术有限公司

对象名称	系统管理员	安全管理员	审计管理员	普通用户
模式	权限缩小。 只对自己的模式有所有的权限，对其他用户的模式无权限。	无变化。		
函数	权限缩小。 只对自己的函数有所有的权限，对其他用户放在属于各自模式下的函数无权限。	无变化。		
自定义视图	权限缩小。 只对自己的视图及其他用户放在public模式下的视图有所有的权限，对其他用户放在属于各自模式下的视图无权限。	无变化。		
系统表和系统视图	无变化。	无变化。	无变化。	无权查看任何系统表和视图。

6.2 DWS 敏感数据管理

6.2.1 DWS 行级访问控制

行级访问控制特性将数据库访问控制精确到数据表行级别，使数据库达到行级访问控制的能力。不同用户执行相同的SQL查询操作，读取到的结果是不同的。

用户可以在数据表创建行访问控制(Row Level Security)策略，该策略是指针对特定数据库用户、特定SQL操作生效的表达式。当数据库用户对数据表访问时，若SQL满足数据表特定的Row Level Security策略，在查询优化阶段将满足条件的表达式，按照属性(PERMISSIVE | RESTRICTIVE)类型，通过AND或OR方式拼接，应用到执行计划上。

行级访问控制的目的是控制表中行级数据可见性，通过在数据表上预定义Filter，在查询优化阶段将满足条件的表达式应用到执行计划上，影响最终的执行结果。当前受影响的SQL语句包括SELECT，UPDATE，DELETE。

场景一：某表中汇总了不同用户的数据，但是不同用户只能查看自身相关的数据信息，不能查看其他用户的数据信息。

```
--创建用户alice, bob, peter
CREATE ROLE alice PASSWORD 'password';
CREATE ROLE bob PASSWORD 'password';
CREATE ROLE peter PASSWORD 'password';

--创建表public.all_data，包含不同用户数据信息
CREATE TABLE public.all_data(id int, role varchar(100), data varchar(100));

--向数据表插入数据
INSERT INTO all_data VALUES(1, 'alice', 'alice data');
INSERT INTO all_data VALUES(2, 'bob', 'bob data');
INSERT INTO all_data VALUES(3, 'peter', 'peter data');
```


版权所有 © 华为云计算技术有限公司

```
--将表all_data的读取权限赋予alice，bob和peter用户
GRANT SELECT ON all_data TO alice, bob, peter;

--打开行访问控制策略开关
ALTER TABLE all_data ENABLE ROW LEVEL SECURITY;

--创建行访问控制策略，当前用户只能查看用户自身的数据
CREATE ROW LEVEL SECURITY POLICY all_data_rls ON all_data USING(role = CURRENT_USER);

--查看表详细信息
 \d+ all_data
                            Table "public.all_data"
 Column |          Type          | Modifiers | Storage  | Stats target | Description
--------+------------------------+-----------+----------+--------------+-------------
 id     | integer                |           | plain    |              |
 role   | character varying(100) |           | extended |              |
 data   | character varying(100) |           | extended |              |
Row Level Security Policies:
    POLICY "all_data_rls"
      USING (((role)::name = "current_user"()))
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, enable_rowsecurity=true

--切换至用户alice，执行SQL"SELECT * FROM all_data"
SET ROLE alice PASSWORD 'password';
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  1 | alice | alice data
(1 row)

EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                        QUERY PLAN
-----------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'alice'::name)
 Notice: This query is influenced by row level security feature
(5 rows)

--切换至用户peter，执行SQL"SELECT * FROM .all_data"
SET ROLE peter PASSWORD 'password';
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  3 | peter | peter data
(1 row)

 EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                        QUERY PLAN
-----------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'peter'::name)
 Notice: This query is influenced by row level security feature
(5 rows)
```

6.2.2 DWS 数据脱敏

DWS提供列级别的数据脱敏(Dynamic Data Masking)功能。针对某些敏感信息（如身份证号、手机号、银行卡号等），通过应用脱敏函数进行原始数据的变形改写，实现敏感隐私数据的可靠保护，从而增强产品在数据安全和隐私保护方面的能力。

- 创建表的数据脱敏策略

 版权所有 © 华为云计算技术有限公司

DWS通过使用“CREATE REDACTION POLICY”语法在表上创建数据脱敏策略并使用函数（MASK_NONE，不进行任何脱敏处理；MASK_FULL，全脱敏成固定值；MASK_PARTIAL，按指定的字符类型，数值类型或时间类型进行部分脱敏处理。）限定脱敏策略生效范围。

- 修改表的数据脱敏策略

  ALTER REDACTION POLICY语法用于修改脱敏策略生效表达式、重命名脱敏策略，以及新增、修改、删除脱敏列信息。

- 删除表的脱敏策略

  DROP REDACTION POLICY语法用于删除脱敏策略在表的所有列字段上的脱敏函数信息。

- 查看脱敏策略和脱敏列信息

  脱敏策略信息存储在系统表**PG_REDACTION_POLICY**，脱敏列信息存储在系统表**PG_REDACTION_COLUMN**。用户也可以通过系统视图**REDACTION_POLICIES**和**REDACTION_COLUMNS**，更加方便地查看脱敏策略及脱敏列信息。

注意事项

- 数据脱敏特性使用feature_support_options参数进行控制，当出现“REDACTION POLICY is not yet supported, please add enable_data_redaction into feature_support_options.“或者“Cannot use the feature of data redaction, please drop existed redaction policy or add enable_data_redaction into feature_support_options.“的报错信息时，说明没有开启数据脱敏特性，需联系技术支持人员设置feature_support_options参数，开启数据脱敏特性。
- 通常，用户可以执行SELECT语句查看敏感信息的脱敏效果。如果语句有如下特征，则可能存在故意套取敏感数据的可能性，造成语句执行报错。
  - GROUP BY子句引用与目标列一样含有脱敏列的Target Entry作为分组。
  - DISTINCT作用在输出的脱敏列上。
  - 带有CTE的语句。
  - 涉及集合操作。
  - 子查询的目标列不是基表的脱敏列，而是基表脱敏列的表达式或者函数调用。
- 支持脱敏数据的COPY TO或者GDS导出功能。由于脱敏数据的不可逆性，针对脱敏数据的二次运算无任何实际意义。
- 禁止UPDATE、MERGE INTO、DELETE语句的目标列涉及脱敏列。
- UPSERT语句允许通过EXCLUDED更新插入数据。如果引用脱敏列更新基表数据，存在误改数据的可能，执行会报错。
- 自8.2.1集群版本针对同一张表可以允许创建多个脱敏策略，实现敏感数据分级分类场景下的多样化配置。多脱敏策略的选择和应用原则如下：
  - 选择满足当前Session的多个候选策略中policy_order最大的那一个策略，即策略次序越大表示越晚创建。
  - 脱敏可算场景下，DML语句发生策略的继承，也仅继承源表相关的候选策略中policy_order最大的那一个策略。

示例

以员工表emp，表的属主alice及角色matu、july为例，简要介绍数据脱敏过程。其中，表emp包含员工的姓名、手机号、邮箱、银行卡号、薪资等隐私数据

 版权所有 © 华为云计算技术有限公司

步骤1 使用管理员用户连接数据库后，创建角色alice、matu和july。

```
CREATE ROLE alice PASSWORD 'password';
CREATE ROLE matu PASSWORD 'password';
CREATE ROLE july PASSWORD 'password';
```

步骤2 赋予alice、matu和july当前数据库的模式权限。

```
GRANT ALL PRIVILEGES on schema public to alice,matu,july;
```

步骤3 切换至角色alice，创建表emp并插入三条员工信息。

```
SET ROLE alice PASSWORD 'password';

CREATE TABLE emp(id int, name varchar(20), phone_no varchar(11), card_no number, card_string
varchar(19), email text, salary numeric(100, 4), birthday date);

INSERT INTO emp VALUES(1, 'anny', '13420002340', 1234123412341234, '1234-1234-1234-1234',
'smithWu@163.com', 10000.00, '1999-10-02');
INSERT INTO emp VALUES(2, 'bob', '18299023211', 3456345634563456, '3456-3456-3456-3456',
'66allen_mm@qq.com', 9999.99, '1989-12-12');
INSERT INTO emp VALUES(3, 'cici', '15512231233', NULL, NULL, 'jonesishere@sina.com', NULL,
'1992-11-06');
```

步骤4 alice将表emp的读取权限授予matu、july。

```
GRANT SELECT ON emp TO matu, july;
```

步骤5 创建脱敏策略mask_emp，仅alice可查看员工所有信息，matu和july对员工银行卡号和薪资数据不可见。字段card_no是数值类型，采用MASK_FULL全脱敏成固定值0；字段card_string是字符类型，采用MASK_PARTIAL按指定的输入输出格式对原始数据进行部分脱敏；字段salary是数值类型，采用MASK_PARTIAL指定数字9部分脱敏倒数第二位前的所有数位值。

```
CREATE REDACTION POLICY mask_emp ON emp WHEN (current_user IN ('matu', 'july'))
 ADD COLUMN card_no WITH mask_full(card_no),
 ADD COLUMN card_string WITH mask_partial(card_string, 'VVVVFVVVVFVVVVFVVVV','VVVV-VVVV-VVVV-
VVVV','#',1,12),
 ADD COLUMN salary WITH mask_partial(salary, '9', 1, length(salary) - 2);
```

步骤6 切换到matu和july，查看员工表emp。

```
SET ROLE matu PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |       email         |  salary   |      birthday
----+------+-------------+---------+---------------------+---------------------+-----------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com     | 99999.9990 |
1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com   |  9999.9990 |
1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |           | 1992-11-06 00:00:00
(3 rows)

SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |       email         |  salary   |      birthday
----+------+-------------+---------+---------------------+---------------------+-----------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com     | 99999.9990 |
1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com   |  9999.9990 |
1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |           | 1992-11-06 00:00:00
(3 rows)
```

步骤7 若需要matu也有员工所有信息的查看权限，只有july不可见，修改策略生效范围即可。

```
SET ROLE alice PASSWORD 'password';
ALTER REDACTION POLICY mask_emp ON emp WHEN(current_user = 'july');
```

步骤8 切换到matu和july，重新查看员工表emp。

 版权所有 © 华为云计算技术有限公司

```
SET ROLE matu PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   |     card_no      |     card_string     |        email         |  salary  |      birthday
----+------+-------------+------------------+---------------------+----------------------+----------+---------------------
  1 | anny | 13420002340 | 1234123412341234 | 1234-1234-1234-1234 | smithWu@163.com      | 10000.0000 | 1999-10-02 00:00:00
  2 | bob  | 18299023211 | 3456345634563456 | 3456-3456-3456-3456 | 66allen_mm@qq.com    | 9999.9900 | 1989-12-12 00:00:00
  3 | cici | 15512231233 |                  |                     | jonesishere@sina.com |           | 1992-11-06 00:00:00
(3 rows)

SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |        email         |  salary  |      birthday
----+------+-------------+---------+---------------------+----------------------+----------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com      | 99999.9990 | 1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com    |  9999.9990 | 1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |           | 1992-11-06 00:00:00
(3 rows)
```

步骤9 员工信息phone_no、email和birthday也是隐私数据，更新脱敏策略mask_emp，新增三个脱敏列。

```
SET ROLE alice PASSWORD 'password';
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN phone_no WITH mask_partial(phone_no, '*', 4);
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN email WITH mask_partial(email, '*', 1, position('@' in email));
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN birthday WITH mask_full(birthday);
```

步骤10 切换到july，查看表emp数据。

```
SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |        email         |  salary  |      birthday
----+------+-------------+---------+---------------------+----------------------+----------+---------------------
  1 | anny | 134******** |       0 | ####-####-####-1234 | ********163.com      | 99999.9990 | 1970-01-01 00:00:00
  2 | bob  | 182******** |       0 | ####-####-####-3456 | **********qq.com     |  9999.9990 | 1970-01-01 00:00:00
  3 | cici | 155******** |         |                     | **********sina.com   |           | 1970-01-01 00:00:00
(3 rows)
```

步骤11 通过视图redaction_policies和redaction_columns查看当前脱敏策略mask_emp的详细信息。

```
SELECT * FROM redaction_policies;
 object_schema | object_owner | object_name | policy_name |            expression             | enable | policy_description | inherited
---------------+--------------+-------------+-------------+-----------------------------------+--------+--------------------+-----------
 public        | alice        | emp         | mask_emp    | ("current_user"() = 'july'::name) | t      |                    | f
(1 row)

SELECT object_name, column_name, function_info FROM redaction_columns;
 object_name | column_name |                                                                                                function_info
-------------+-------------+-----------------------------------------------------------------------------------------------------------------------------------------
 emp         | card_no     | mask_full(card_no)
 emp         | card_string | mask_partial(card_string, 'VVVVFVVVVFVVVVFVVVV'::text, 'VVVV-VVVV-VVVV-VVVV'::text, '#'::text, 1, 12)
 emp         | email       | mask_partial(email, '*'::text, 1, "position"(email, '@'::text))
 emp         | salary      | mask_partial(salary, '9'::text, 1, (length((salary)::text) - 2))
 emp         | birthday    | mask_full(birthday)
 emp         | phone_no    | mask_partial(phone_no, '*'::text, 4)
(6 rows)
```

 版权所有 © 华为云计算技术有限公司

步骤12 新增一列salary_info，若需要将文本类型的薪资信息统一脱敏成“*.*”，可以创建自定义脱敏函数实现。此处采用PL/PGSQL语言定义脱敏函数mask_regexp_salary，创建脱敏列时，只需自定义脱敏的函数名和参数列表，详细内容可参考**DWS用户自定义函数**。

```
SET ROLE alice PASSWORD 'password';

ALTER TABLE emp ADD COLUMN salary_info TEXT;
UPDATE emp SET salary_info = salary::text;

CREATE FUNCTION mask_regexp_salary(salary_info text) RETURNS text AS
$$
 SELECT regexp_replace($1, '[0-9]+','*','g');
$$
 LANGUAGE SQL
STRICT SHIPPABLE;

ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN salary_info WITH
mask_regexp_salary(salary_info);

SET ROLE july PASSWORD 'password';
SELECT id, name, salary_info FROM emp;
 id | name | salary_info
----+------+-------------
  1 | anny | *.*
  2 | bob  | *.*
  3 | cici |
(3 rows)
```

步骤13 无需为表emp设置敏感策略，删除脱敏策略mask_emp。

```
SET ROLE alice PASSWORD 'password';
DROP REDACTION POLICY mask_emp ON emp;
```

----结束

6.2.3 DWS 字符串加解密

DWS支持使用以下函数对字符串进行加解密。

- gs_encrypt(encryptstr, keystr, cryptotype, cryptomode, hashmethod)

  描述：采用cryptotype和cryptomode组成的加密算法以及hashmethod指定的HMAC算法，以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。支持的cryptotype：aes128, aes192, aes256, sm4。支持的cryptomode：cbc。支持的hashmethod：sha256, sha384, sha512, sm3。支持的加密数据类型：目前数据库支持的数值类型，字符类型，二进制类型中的RAW，日期/时间类型中的DATE、TIMESTAMP、SMALLDATETIME。keystr的长度范围与加密算法相关，为1~KeyLen字节。当cryptotype为aes128和sm4时，KeyLen为16，aes192时KeyLen为24，aes256时KeyLen为32。

  返回值类型：text

  返回值长度：至少为 4*[(maclen + 56)/3] 字节，不超过 4*[(Len + maclen + 56)/3] 字节，其中Len为加密前数据长度（单位为字节），maclen为HMAC值的长度，当hashmethod为sha256和sm3时maclen为32，sha384时maclen为48，sha512时maclen为64。即当hashmethod为sha256和sm3时，返回值长度至少为120字节，不超过4*[(Len + 88)/3] 字节；当hashmethod为sha384时，返回值长度至少为140字节，不超过4*[(Len + 104)/3] 字节；当hashmethod为sha512时，返回值长度至少为160字节，不超过4*[(Len + 120)/3] 字节；

  示例：

  ```
  SELECT gs_encrypt('DWS', '1234', 'aes128', 'cbc',  'sha256');
                                                  gs_encrypt
  ----------------------------------------------------------------------------------------------------------------------
  ```


版权所有 © 华为云计算技术有限公司

```
---
AAAAAAAAAAArYr4+WFD1Tew23KhoB7H7cPComVRqy3O/3CnSglER3lAw662L6QfI
+ojFvHTRnqhS10KYbE+Xui5kYU5Hn56l7QaG6VuCKG84uZ9X8+nYXw==
(1 row)
```

说明

- 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
- 在同一张数据表中，加密函数ge_encrypt与gs_encrypt_aes128不要混合使用。

- gs_decrypt(decryptstr, keystr, cryptotype, cryptomode, hashmethod)

  描述：采用cryptotype和cryptomode组成的加密算法以及hashmethod指定的HMAC算法，以keystr为密钥对decryptstr字符串进行解密，返回解密后的字符串。解密使用的keystr必须保证与加密时使用的keystr一致才能正常解密。keystr不得为空。

  返回值类型：text

  示例：

```
SELECT gs_decrypt('AAAAAAAAAAArYr4+WFD1Tew23KhoB7H7cPComVRqy3O/3CnSglER3lAw662L6QfI
+ojFvHTRnqhS10KYbE+Xui5kYU5Hn56l7QaG6VuCKG84uZ9X8+nYXw==', '1234', 'aes128', 'cbc',
'sha256');
 gs_decrypt
--------------
 DWS
(1 row)
```

说明

- 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
- 此函数需要结合gs_encrypt加密函数共同使用，且加密算法和HMAC算法要保证一致。

- gs_encrypt_aes128(encryptstr,keystr)

  描述：以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。keystr的长度范围为1~16字节。支持的加密数据类型：目前数据库支持的数值类型，字符类型，二进制类型中的RAW，日期/时间类型中的DATE、TIMESTAMP、SMALLDATETIME。

  返回值类型：text

  返回值长度：至少为92字节，不超过 4*[(Len+68)/3] 字节，其中Len为加密前数据长度（单位为字节）。

  示例：

```
SELECT gs_encrypt_aes128('DWS','1234');
                                gs_encrypt_aes128
-----------------------------------------------------------------------------------------------
 ZrCp794vO5I9qJ+jHFf/sQqRyMBy0lKIDGP5S8RJXzgmpXoa/
e4EgmK82P5y5xe1bOXbJeoNxyHagK9OhPVVeJDbn/M=
(1 row)
```

说明

- 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
- 在同一张数据表中，加密函数gs_encrypt_aes128与ge_encrypt不要混合使用。

- gs_decrypt_aes128(decryptstr,keystr)

  描述：以keystr为密钥对decryptstr字符串进行解密，返回解密后的字符串。解密使用的keystr必须保证与加密时使用的keystr一致才能正常解密。keystr不得为空。

版权所有 © 华为云计算技术有限公司

返回值类型：text

示例：

```
SELECT gs_decrypt_aes128('ZrCp794vO5I9qJ+jHFf/sQqRyMBy0lKIDGP5S8RJXzgmpXoa/
e4EgmK82P5y5xe1bOXbJeoNxyHagK9OhPVVeJDbn/M=','1234');
 gs_decrypt_aes128
-------------------
 DWS
(1 row)
```

说明

- 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
- 此函数需要结合gs_encrypt_aes128加密函数共同使用。

- gs_hash(hashstr, hashmethod)

  描述：以hashmethod算法对hashstr字符串进行信息摘要，返回信息摘要字符串。支持的hashmethod：sha256, sha384, sha512, sm3。

  返回值类型：text

  返回值长度：sha256和sm3返回64字节，sha384返回96字节，sha512返回128字节。

  示例：

  ```
  SELECT gs_hash('DWS', 'sha256');
                                          gs_hash
  ----------------------------------------------------------------------------------------------------
   1d595d9e342bccc3b158fabfcdfece54d9f821cf9c7461c5d3b6c0157c442f94
  (1 row)
  ```

- md5(string)

  描述：将string使用MD5加密，并以16进制数作为返回值。

  说明

  MD5的安全性较低，不建议使用。

  返回值类型：text

  示例：

  ```
  SELECT md5('ABC');
                md5
  ----------------------------------
   902fbdd2b1df0c4f70b4a5d23525e932
  (1 row)
  ```

6.2.4 使用 pgcrypto 加密 DWS 数据

DWS数据库自8.2.0集群版本开始内置加密解密模块pgcrypto。pgcrypto模块允许数据库用户以加密形式存储数据的某些列，为敏感数据增加了一层额外的保护。因此在没有加密密钥的情况下，任何人都无法读取以加密形式存储在DWS数据库中的数据。

pgcrypto函数在数据库服务器内部运行，这意味着所有数据和密码都以明文形式在pgcrypto和客户端应用程序之间传输。为了获得最佳安全性，建议在客户端和DWS服务器之间使用SSL连接。

有关pgcrypto模块中各个函数的详细信息如下：

通用哈希函数

- digest()

 版权所有 © 华为云计算技术有限公司

digest()函数可以根据不同的算法生成数据的二进制哈希值，语法如下：

```
digest(data text, type text) returns bytea
digest(data bytea, type text) returns bytea
```

其中，data是原始数据，type是加密算法包括md5、sha1、sha224、sha256、sha384、sha512和sm3；函数的返回结果为二进制字符串。

示例：

使用digest() 函数对字符串DWS进行sha256加密存储：

```
select digest('DWS', 'sha256');
                              digest
------------------------------------------------------------------
 \x1d595d9e342bccc3b158fabfcdfece54d9f821cf9c7461c5d3b6c0157c442f94
(1 row)
```

- hmac()

  hmac()函数可以根据不同的算法为带有密钥的数据计算出MAC值，语法如下：

  ```
  hmac(data text, key text, type text) returns bytea
  hmac(data bytea, key bytea, type text) returns bytea
  ```

  其中，data是原始数据，key是加密密钥，type是加密算法，包括md5、sha1、sha224、sha256、sha384、sha512以及sm3；函数的返回结果为二进制字符串。

  示例：

  使用key123密钥和sha256加密算法对字符串DWS计算MAC值：

  ```
  select hmac('DWS', 'key123', 'sha256');
  hmac
  ------------------------------------------------------------------
  \xa2ee3b0d627a93166ac2e6a313deb86e0710ad4505cd7f7b88d563579174b8f9
  (1 row)
  ```

  对于digest() 函数，如果同时被修改了原始数据和加密结果，无法进行识别；hmac() 函数只要密钥没有泄露的话，可以发现被篡改的数据。

  如果该密钥大于哈希块的长度，它将先被哈希然后把结果用作密钥。

密码哈希函数

crypt()和gen_salt()函数专用于哈希密码。crypt()执行哈希用于加密数据，gen_salt()用于生成加盐哈希。

crypt()中的算法和普通的MD5或者SHA1哈希算法存在以下不同之处：

- crypt()中算法很慢。由于密码包含的数据量很小，这是增加暴力破解难度的唯一方法。
- 它们使用一个随机值（称为*salt*，即盐值），因此这样具有相同口令的用户将得到不同的密文口令。这也是针对破解算法提供一种额外的安全保护。
- 它们的结果中包括了算法类型，因此可以针对不同用户使用不同的算法对密码进行加密。
- 其中一些算法具有自适应性，意味着当计算机性能变得更快时，可以调整该算法使其变得更慢，而不会产生与已有密码的不兼容性。

crypt()函数所支持的算法如下表：

 版权所有 © 华为云计算技术有限公司

表 6-7 crypt()支持的算法

算法	密码最大长度	自适应性	Salt位数	输出结果长度	描述
bf	72	√	128	60	基于Blowfish的2a变种算法
md5	unlimited	×	48	34	基于MD5的加密算法
xdes	8	√	24	20	扩展DES
des	8	×	12	13	原生UNIX加密算法

- crypt()

  crypt()语法格式如下：

  ```
  crypt(password text, salt text) returns text
  ```

  该函数返回password字符串crypt(3)格式的哈希值，salt参数由gen_salt()函数生成。

  对于相同的密码，crypt()函数每次也会返回不同的结果，因为gen_salt()函数每次都会生成不同的salt。校验密码时可以将之前生成的哈希结果作为salt。

  例如：设置一个新密码：

  ```
  UPDATE ... SET pswhash = crypt('new password', gen_salt('bf',10));
  ```

  通过比较存储的密码哈希值验证输入的密码的正确性。

  ```
  SELECT (pswhash = crypt('entered password', pswhash)) AS pswmatch FROM ... ;
  ```

  如果输入的口令正确，这会返回true。

  示例：

  ```
  create table userpwd(userid int8, pwd text);
  CREATE TABLE

  insert into userpwd values (1, crypt('this is a pwd', gen_salt('bf',10)));
  INSERT 0 1

  select crypt('this is a pwd', pwd)=pwd as result from userpwd where userid =1;
   result
  --------
   t
  (1 row)

  select crypt('this is a wrong pwd', pwd)=pwd as result from userpwd where userid =1;
   result
  --------
   f
  (1 row)
  ```

- gen_salt()

  gen_salt()函数用来产生随机的参数给crypt，语法如下：

  ```
  gen_salt(type text [, iter_count integer ]) returns text
  ```

  该函数每次都会生成一个随机的盐值(salt)字符串，该字符串同时决定了crypt()函数使用的算法；type参数用于指定一个生成字符串的哈希算法，取值包括 des、xdes、md5 以及 bf。对于xdes和bf算法，iter_count指迭代次数, 数字越大加密时间越长, 被破解需要的时间也越长。

  ```
  SELECT gen_salt('des'), gen_salt('xdes'), gen_salt('md5'), gen_salt('bf');
   gen_salt | gen_salt  |  gen_salt   |          gen_salt
  ```


版权所有 © 华为云计算技术有限公司

```
----------+------------+-------------+------------------------------
 qh       | _J9..uEUi | $1$SNgqyKAi | $2a$06$B/Etc3J8zYBV49LrDU97MO
(1 row)
```

每种算法生成的salt拥有固定的格式，例如bf算法结果中的$2a$06$，2a表示Blowfish的2a变种算法，06表示迭代的次数。

如果忽略 iter_count，将会使用默认的迭代次数。允许的iter_count值与算法相关，如下表所示。对于xdes算法，迭代次数必须是一个奇数。

表 6-8 crypt()的迭代计数

算法	默认值	最小值	最大值
xdes	725	1	16777215
bf	6	4	31

PGP 加密函数

DWS的PGP加密函数遵循OpenPGP(RFC 4880)标准，包括对称密钥加密（私钥加密）和非对称密钥加密（公钥加密）。

加密后的PGP消息由两部分组成：

- 这个消息的会话密钥（加密过的对称密钥或者公钥）。
- 使用该会话密钥加密的数据。

对于对称密钥（也就是密码）加密：

1. 使用String2Key（S2K）算法对密钥进行加密，类似于执行一个特意减慢并且包含随机salt的crypt() 算法，生成一个完整长度的二进制密钥。
2. 如果要求一个单独的会话密钥，生成一个随机的密钥。否则使用上面的S2K密钥直接作为会话密钥。
3. 如果直接使用S2K密钥，只将S2K设置加入会话密钥包中。否则，使用S2K密钥对会话密钥进行加密，然后放入会话密钥包中。

对于公钥加密：

1. 生成一个随机的会话密钥。
2. 使用公钥对其进行加密后放入会话密钥包中。

无论哪种情况，数据的加密过程如下：

1. 执行可选的数据操作：压缩、转换成UTF-8或者换行符的转换。
2. 在数据前面增加一个随机字节组成的块，相当于使用了一个随机的初始值（IV）。
3. 追加随机前缀和数据的SHA1哈希值到数据后面。
4. 将所有内容使用会话密钥进行加密后放入数据包中。

支持的PGP加密函数：

- pgp_sym_encrypt()
  描述：用于对称密钥加密。
  语法格式：

 版权所有 © 华为云计算技术有限公司

```
pgp_sym_encrypt(data text, psw text [, options text ]) returns bytea
pgp_sym_encrypt_bytea(data bytea, psw text [, options text ]) returns bytea
```

其中，data是要加密的数据；psw是PGP对称密钥；options 参数用于设置选项，参考**表6-9**。

- pgp_sym_decrypt()

  描述：用于解密PGP对称密钥加密后的消息。

  语法格式：

  ```
  pgp_sym_decrypt(msg bytea, psw text [, options text ]) returns text
  pgp_sym_decrypt_bytea(msg bytea, psw text [, options text ]) returns bytea
  ```

  其中，msg是要解密的消息；psw是PGP对称密钥；options参数用于设置选项，参考**表6-9**。为了避免输出无效的字符，不允许使用pgp_sym_decrypt函数对bytea数据进行解密；可以使用 pgp_sym_decrypt_bytea 对原始文本数据进行解密。

- pgp_pub_encrypt()

  描述：用于公共密钥加密。

  语法格式：

  ```
  pgp_pub_encrypt(data text, key bytea [, options text ]) returns bytea
  pgp_pub_encrypt_bytea(data bytea, key bytea [, options text ]) returns bytea
  ```

  其中，data是要加密的数据；key是PGP公钥，如果传入一个私钥将会返回错误；options参数用于设置选项，参考**表6-9**。

- pgp_pub_decrypt()

  描述：用于解密PGP公共密钥加密后的消息。

  语法格式：

  ```
  pgp_pub_decrypt(msg bytea, key bytea [, psw text [, options text ]]) returns text
  pgp_pub_decrypt_bytea(msg bytea, key bytea [, psw text [, options text ]]) returns bytea
  ```

  解密一个公共密钥加密的消息。*key*必须是对应于用来加密的公钥的私钥。如果私钥是用口令保护的，必须在*psw*中给出该口令。如果没有口令，但想要指定选项，需要给出一个空口令。

  为了避免输出无效的字符，不允许使用pgp_pub_decrypt函数对bytea数据进行解密；可以使用 pgp_pub_decrypt_bytea 对原始文本数据进行解密。

  其中，key是公共密钥对应的私钥；如果私钥使用了密码保护功能，必须在psw参数中指定密码；如果没有使用密码保护，想要指定options参数时必须指定一个空的psw。options参数用于设置选项，参考**表6-9**。

- pgp_key_id()

  描述：用于提取PGP公钥或者私钥的密钥ID；如果传入一个加密后的消息，将会返回加密该消息使用的密钥ID。

  语法格式：

  ```
  pgp_key_id(bytea) returns text
  ```

  该函数可能返回两个特殊密钥ID：

  - SYMKEY，表示该消息使用对称密钥进行加密。
  - ANYKEY，表示该消息使用公钥进行加密，但是密钥ID已经被删除。这意味着需要尝试所有的密钥，查找可以解密该消息的私钥。pgcrypto不会产生这种加密消息。

  说明

  不同的密钥可能拥有相同的ID，这种情况很少见但可能存在。客户端应用程序需要自己尝试使用不同的密钥进行解密，就像处理ANYKEY一样。

- armor()

版权所有 © 华为云计算技术有限公司

描述：用于将二进制数据转换为PGP ASCII-armor格式，相当于Base64加上CRC以及额外的格式化。

语法格式：

```
armor(data bytea [ , keys text[], values text[] ]) returns text
```

- dearmor()

  描述：用于执行相反的转换。

  语法格式：

  ```
  dearmor(data text) returns bytea
  ```

  将加密后的数据bytea，转换成PGP ASCII-armor格式.或者反向转换。

  其中，data是需要转换的数据；如果指定了keys和values数值，每个key/value对都会生成一个armor header并添加到编码格式中；两个数组都是一维数组，长度相同，并且不能包含非ASCII字符。

- pgp_armor_headers()

  描述：函数用于返回数据中的armor header。

  ```
  pgp_armor_headers(data text, key out text, value out text) returns setof record
  ```

  返回结果是一个包含key和value两个字段的数据行集，如果其中包含任何非ASCII字符，都会被视作UTF-8字符。

用GnuPG生成PGP 密钥

要生成一个新密钥：

```
gpg --gen-key
```

更好的密钥类型是“DSA和Elgamal”。

对于RSA密钥，必须创建仅用于签名的DSA或RSA密钥作为主控密钥，然后用gpg --edit-key增加一个RSA加密子密钥。

要列举密钥：

```
gpg --list-secret-keys
```

要以ASCII-保护格式导出一个公钥：

```
gpg -a --export KEYID > public.key
```

要以 ASCII-保护格式导出一个私钥：

```
gpg -a --export-secret-keys KEYID > secret.key
```

在把这些密钥交给PGP函数之前，需要对它们使用dearmor()。或者如果你能处理二进制数据，可以从命令中去掉-a。

须知

PGP加密函数的存在以下限制：

- 不支持签名。这也意味着它不会检查加密子密钥是否属于主控密钥。
- 不支持加密密钥作为主控密钥。由于通常并不鼓励这种用法，因此这应该不是问题。
- 不支持多个子密钥。由于实际应用中经常需要多个子密钥，这可能是个问题。另一方面，不要使用常规GPG/PGP密钥作为pgcrypto加密密钥，而应该创建新的密钥，因为这是非常不同的使用场景。

PGP函数的选项

pgcrypto函数中的选项名称和GnuPG类似，选项的值使用等号设置，每个选项使用逗号进行分隔。例如：

 版权所有 © 华为云计算技术有限公司

```
pgp_sym_encrypt(data, psw, 'compress-algo=1, cipher-algo=aes256')
```

除了convert-crlf之外，其他选项仅适用于加密函数。解密函数会从PGP数据中获取参数。

最常设置的选项包括compress-algo和unicode-mode。其他选项通常使用默认值。

表 6-9 pgcrypto 加密选项

选项	描述	默认值	取值	适用函数
cipher-algo	使用的密码算法。	aes128	bf, aes128, aes192, aes256, 3des, cast5	pgp_sym_encrypt, pgp_pub_encrypt
compress-algo	使用的压缩算法。	0	• 0，表示不压缩。 • 1，表示ZIP压缩。 • 2，表示ZLIB压缩（= ZIP加上元数据和CRC）	pgp_sym_encrypt, pgp_pub_encrypt
compress-level	压缩级别。级别越高压缩得越小但速度也越慢。0表示不压缩。	6	0，1-9	pgp_sym_encrypt, pgp_pub_encrypt
convert-crlf	加密时是否将\n转换成\r\n并且解密时执行相反的转换是否把\r\n转换成\n。RFC4880指定文本数据存储时需要使用\r\n作为换行符。	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt, pgp_sym_decrypt, pgp_pub_decrypt
disable-mdc	不用 SHA-1 保护数据。仅用于兼容老旧的PGP产品。	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt
sess-key	使用单独的会话密钥。公钥加密总是使用一个单独的会话密钥。该选项用于对称密钥加密，因为对称密钥加密默认直接使用 S2K 密钥。	0	0，1	pgp_sym_encrypt

版权所有 © 华为云计算技术有限公司

选项	描述	默认值	取值	适用函数
s2k-mode	使用的S2K 算法。	3	• 0 ，表示不使用 salt。不推荐! • 1，表示使用 salt ，但是迭代固定次数。 • 3 ，可变的迭代计数。	pgp_sym_encrypt
s2k-count	S2K 算法的迭代次数。	65536 和 253952 之间的一个随机数值	大于等于1024并且小于等于65011712	pgp_sym_encrypt，并且 s2k-mode=3
s2k-digest-algo	S2K计算时的摘要算法。	sha1	md5, sha1	pgp_sym_encrypt
s2k-cipher-algo	加密单独会话密钥时使用的密码。	默认使用 cipher-algo 的算法	bf, aes, aes128, aes192, aes256	pgp_sym_encrypt
unicode-mode	是否将文本数据在数据库内部编码和 UTF-8 之间来回转换。如果当前数据库已经是UTF-8，不会执行转换，但是消息将被标记为 UTF-8。没有指定该选项就不会被标记	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt

原始加密函数

原始加密函数仅仅会对数据运行一次加密，不支持PGP加密的任何高级功能，因此存在以下问题：

- 直接将用户密钥作为加密密钥。
- 不提供任何完整性检查来校验加密后的数据是否被修改。
- 需要用户自己关联所有加密参数，包括初始值（ IV ）。

版权所有 © 华为云计算技术有限公司

- 不支持处理文本数据。

因此，在引入了PGP加密后，不建议使用这些原始加密函数。

```
encrypt(data bytea, key bytea, type text) returns bytea
decrypt(data bytea, key bytea, type text) returns bytea
encrypt_iv(data bytea, key bytea, iv bytea, type text) returns bytea
decrypt_iv(data bytea, key bytea, iv bytea, type text) returns bytea
```

其中，data是需要加密的数据；type用于指定加密/解密方法。type参数的语法如下：

```
algorithm [ - mode ] [ /pad: padding ]
```

其中algorithm的取值范围如下：

- bf，Blowfish算法。包括近义词：BF，BF-CBC；BLOWFISH，BF-CBC；BLOWFISH-CBC，BF-CBC；BLOWFISH-ECB，BF-ECB；BLOWFISH-CFB，BF-CFB。
- aes，AES算法(Rijndael-128, -192或-256)。包括近义词：AES，AES-CBC；RIJNDAEL，AES-CBC；RIJNDAEL，AES-CBC；RIJNDAEL-CBC，AES-CBC；RIJNDAEL-ECB，AES-ECB。
- DES算法。包括近义词：DES，DES-CBC；3DES，DES3-CBC；3DES-ECB，DES3-ECB；3DES-CBC，DES3-CBC
- sm4，SM4算法。包括近义词：SM4-CBC
- CAST5算法。包括近义词：CAST5-CBC

mode的可能取值范围如下：

- cbc，下一个块依赖前一个块（默认值）
- ecb，每个块独立加密（不推荐，仅用于测试）

padding的取值范围如下：

- pkcs，数据可以是任意长度（默认值）
- none，数据长度必须是密码块大小的倍数

例如，以下函数的加密结果相同：

```
encrypt(data, 'fooz', 'bf')
encrypt(data, 'fooz', 'bf-cbc/pad:pkcs')
```

对于函数encrypt_iv和decrypt_iv，参数iv表示CBC模式的初始值，ECB模式会忽略该参数。如果它的长度不是准确的块大小，可能会被截断或者使用0进行填充。对于没有该参数的两个函数，默认全部使用0填充。

随机数据函数

- gen_random_bytes()函数用于生成具有强加密性的随机字节。

  ```
  gen_random_bytes(count integer) returns bytea
  ```

  其中，count表示返回的字节数，取值为1到1024。

  示例：

  ```
  SELECT gen_random_bytes(16);
          gen_random_bytes
  ------------------------------------
   \x1f1eddc11153afdde0f9e1229f8f4caf
  (1 row)
  ```

- gen_random_uuid()函数用于返回一个version 4的随机UUID。

版权所有 © 华为云计算技术有限公司

```
SELECT gen_random_uuid();
gen_random_uuid
--------------------------------------
2bd664a2-b760-4859-8af6-8d09ccc5b830
```

 版权所有 © 华为云计算技术有限公司

7 查询 DWS 数据

7.1 单表查询

示例表：

```
CREATE TABLE newproducts
(
product_id INTEGER NOT NULL,
product_name VARCHAR2(60),
category VARCHAR2(60),
quantity  INTEGER
)
WITH (ORIENTATION = COLUMN) DISTRIBUTE BY HASH(product_id);


INSERT INTO newproducts VALUES (1502, 'earphones', 'electronics',150);
INSERT INTO newproducts VALUES (1601, 'telescope', 'toys',80);
INSERT INTO newproducts VALUES (1666, 'Frisbee', 'toys',244);
INSERT INTO newproducts VALUES (1700, 'interface', 'books',100);
INSERT INTO newproducts VALUES (2344, 'milklotion', 'skin care',320);
INSERT INTO newproducts VALUES (3577, 'dumbbell', 'sports',550);
INSERT INTO newproducts VALUES (1210, 'necklace', 'jewels', 200);
```

简单查询

通过SELECT ... FROM ... 语句从数据库中获取结果。

```
SELECT category FROM newproducts;
  category
------------
 electr
 sports
 jewels
 toys
 books
 skin care
 toys
(7 rows)
```

对结果进行筛选

通过WHERE语句对查询的结果进行过滤，找到想要查询的部分。

```
SELECT * FROM newproducts WHERE category='toys';
 product_id | product_name | category | quantity
```

 版权所有 © 华为云计算技术有限公司

```
------------+--------------+----------+----------
      1601 | telescope    | toys     |       80
      1666 | Frisbee      | toys     |      244
(2 rows)
```

对结果进行排序

使用ORDER BY语句可以让查询结果按照期望的方式进行排序。

```
SELECT product_id,product_name,category,quantity FROM newproducts ORDER BY quantity DESC;
 product_id | product_name |  category   | quantity
------------+--------------+-------------+----------
      3577 | dumbbell     | sports      |      550
      2344 | milklotion   | skin care   |      320
      1666 | Frisbee      | toys        |      244
      1210 | necklace     | jewels      |      200
      1502 | earphones    | electronics |      150
      1700 | interface    | books       |      100
      1601 | telescope    | toys        |       80
(7 rows)
```

限制结果查询数量

如果需要查询只返回部分结果，可以使用LIMIT语句限制查询结果返回的记录数。

```
SELECT product_id,product_name,category,quantity FROM newproducts ORDER BY quantity DESC limit 5;
 product_id | product_name |  category   | quantity
------------+--------------+-------------+----------
      3577 | dumbbell     | sports      |      550
      2344 | milklotion   | skin care   |      320
      1666 | Frisbee      | toys        |      244
      1210 | necklace     | jewels      |      200
      1502 | earphones    | electronics |      150
(5 rows)
```

聚合查询

可以通过使用GROUP BY语句配合聚合函数，构建一个聚合查询来关注数据的整体情况。

```
SELECT category, string_agg(quantity,',') FROM newproducts group by category;
  category   | string_agg
-------------+------------
 toys        | 80,244
 books       | 100
 sports      | 550
 jewels      | 200
 skin care   | 320
 electronics | 150
```

7.2 多表连接查询

连接类型介绍

通过SQL完成各种复杂的查询，多表之间的连接是必不可少的。连接分为：内连接和外连接两大类，每大类中还可进行细分。

- 内连接：标准内连接（INNER JOIN），交叉连接（CROSS JOIN）和自然连接（NATURAL JOIN）。
- 外连接：左外连接（LEFT OUTER JOIN），右外连接（RIGHT OUTER JOIN）和全外连接（FULL JOIN）。

版权所有 © 华为云计算技术有限公司

为了能更好的说明各种连接之间的区别，下面通过具体示例进行详细的阐述。

创建示例表student和math_score，并插入数据，设置enable_fast_query_shipping为off（默认为on）即查询优化器使用分布式框架；参数explain_perf_mode为pretty（默认值为pretty）指定explain的显示格式。

```
CREATE TABLE student(
  id INTEGER,
  name varchar(50)
);

CREATE TABLE math_score(
  id INTEGER,
  score INTEGER
);

INSERT INTO student VALUES(1, 'Tom');
INSERT INTO student VALUES(2, 'Lily');
INSERT INTO student VALUES(3, 'Tina');
INSERT INTO student VALUES(4, 'Perry');

INSERT INTO math_score VALUES(1, 80);
INSERT INTO math_score VALUES(2, 75);
INSERT INTO math_score VALUES(4, 95);
INSERT INTO math_score VALUES(6, NULL);

SET enable_fast_query_shipping = off;
SET explain_perf_mode = pretty;
```

内连接

- 标准内连接（INNER JOIN）

  语法：

  ```
  left_table [INNER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：表示left_table和right_table中满足join_condition的行拼接在一起作为结果输出，不满足条件的元组不会输出。

  示例1：查询学生的数学成绩。

  ```
  SELECT s.id, s.name, ms.score FROM student s JOIN math_score ms on s.id = ms.id;
   id | name  | score
  ----+-------+-------
    2 | Lily  |    75
    1 | Tom   |    80
    4 | Perry |    95
  (3 rows)


  EXPLAIN SELECT s.id, s.name, ms.score FROM student s JOIN math_score ms on s.id = ms.id;
                                    QUERY PLAN
  ------------------------------------------------------------------------------------
   id |                operation                | E-rows | E-memory | E-width | E-costs
  ----+-----------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)            |      4 |          |      13 | 19.47
    2 |    ->  Hash Join (3,4)                  |      4 | 1MB      |      13 | 11.47
    3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       8 | 10.10
    4 |       ->  Hash                          |     12 | 16MB     |       9 | 1.28
    5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
    6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Join (3,4)
          Hash Cond: (ms.id = s.id)

    ====== Query Summary =====
  -------------------------------
  ```

 版权所有 © 华为云计算技术有限公司

```
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 4400KB
(19 rows)
```

- 交叉连接（CROSS JOIN）

  语法：

  ```
  left_table CROSS JOIN right_table
  ```

  说明：表示left_table中所有行和right_table中的所有行分别进行连接，最终结果行数等于两边行数的乘积。又称笛卡尔积。

  示例2：学生表和数学成绩表的交叉连接。

  ```
  SELECT s.id, s.name, ms.score FROM student s CROSS JOIN math_score ms;
   id | name  | score
  ----+-------+-------
    3 | Tina  |    80
    2 | Lily  |    80
    1 | Tom   |    80
    4 | Perry |    80
    3 | Tina  |
    2 | Lily  |
    1 | Tom   |
    4 | Perry |
    3 | Tina  |    95
    2 | Lily  |    95
    1 | Tom   |    95
    4 | Perry |    95
    2 | Lily  |    75
    3 | Tina  |    75
    1 | Tom   |    75
    4 | Perry |    75
  (16 rows)

  EXPLAIN SELECT s.id, s.name, ms.score FROM student s CROSS JOIN math_score ms;
                                    QUERY PLAN
  ------------------------------------------------------------------------------------------
    id |                operation                | E-rows | E-memory | E-width | E-costs
   ----+-----------------------------------------+--------+----------+---------+---------
     1 | ->  Streaming (type: GATHER)            |    120 |          |      13 | 19.89
     2 |    ->  Nested Loop (3,4)                |    120 | 1MB      |      13 | 11.89
     3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       4 | 10.10
     4 |       ->  Materialize                   |     12 | 16MB     |       9 | 1.30
     5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
     6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

    ====== Query Summary =====
   -------------------------------
   System available mem: 1761280KB
   Query Max mem: 1761280KB
   Query estimated mem: 4144KB
  (14 rows)
  ```

- 自然连接（NATURAL JOIN）

  语法：

  ```
  left_table NATURAL JOIN right_table
  ```

  说明：表示left_table和right_table中列名相同的列进行等值连接，且自动将同名列只保留一份。

  示例3：学生表和数学成绩表的自然连接，两表的同名列为id列，将按照id列进行等值连接。

  ```
  SELECT * FROM student s NATURAL JOIN math_score ms;
   id | name  | score
  ----+-------+-------
    1 | Tom   |    80
    4 | Perry |    95
  ```

 版权所有 © 华为云计算技术有限公司

```
 2 | Lily |    75
(3 rows)

EXPLAIN SELECT * FROM student s NATURAL JOIN math_score ms;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
 id |               operation                | E-rows | E-memory | E-width | E-costs
----+----------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)           |      4 |          |      13 | 19.47
  2 |    ->  Hash Join (3,4)                 |      4 | 1MB      |      13 | 11.47
  3 |       ->  Seq Scan on math_score ms    |     30 | 1MB      |       8 | 10.10
  4 |       ->  Hash                         |     12 | 16MB     |       9 | 1.28
  5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
  6 |             ->  Seq Scan on student s  |      4 | 1MB      |       9 | 1.01

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
       Hash Cond: (ms.id = s.id)

  ====== Query Summary =====
-------------------------------
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 4400KB
(19 rows)
```

外连接

- 左外连接（LEFT JOIN）

  语法：

  ```
  left_table LEFT [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：左外连接的结果集包括left_table的所有行，而不仅是连接列所匹配的行。如果left_table的某行在right_table中没有匹配行，则在相关联的结果集行中输出right_table的列均为空值。

  示例4：学生表和数学成绩表进行左外连接，学生表中id为3的行在结果集中对应的右表数据用NULL填充。

  ```
  SELECT s.id, s.name, ms.score FROM student s LEFT JOIN math_score ms on (s.id = ms.id);
   id | name  | score
  ----+-------+-------
    3 | Tina  |
    1 | Tom   |    80
    2 | Lily  |    75
    4 | Perry |    95
  (4 rows)

  EXPLAIN SELECT s.id, s.name, ms.score FROM student s LEFT JOIN math_score ms on (s.id = ms.id);
                                    QUERY PLAN
  ---------------------------------------------------------------------------------------------
   id |                 operation                 | E-rows | E-memory | E-width | E-costs
  ----+-------------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)              |      4 |          |      13 | 10.26
    2 |    ->  Hash Left Join (3, 5)              |      4 | 1MB      |      13 | 2.26
    3 |       ->  Streaming(type: REDISTRIBUTE)   |      4 | 2MB      |       9 | 1.11
    4 |          ->  Seq Scan on student s        |      4 | 1MB      |       9 | 1.01
    5 |       ->  Hash                            |      4 | 16MB     |       8 | 1.11
    6 |          ->  Streaming(type: REDISTRIBUTE) |      4 | 2MB      |       8 | 1.11
    7 |             ->  Seq Scan on math_score ms |      4 | 1MB      |       8 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Left Join (3, 5)
         Hash Cond: (s.id = ms.id)

    ====== Query Summary =====
  ```

 版权所有 © 华为云计算技术有限公司

```
-------------------------------
 System available mem: 901120KB
 Query Max mem: 901120KB
 Query estimated mem: 7520KB
(20 rows)
```

- 右外连接（RIGHT JOIN）

  语法：

  ```
  left_table RIGHT [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：与左外连接相反，右外连接的结果集包括right_table的所有行，而不仅是连接列所匹配的行。如果right_table的某行在left_table中没有匹配行，则在相关联的结果集行中left_table的列均为空值。

  示例5：学生表和数学成绩表进行右外连接，数学成绩表中id为6的行在结果集中对应的左表数据用NULL填充。

  ```
  SELECT ms.id, s.name, ms.score FROM student s RIGHT JOIN math_score ms on (s.id = ms.id);
   id | name  | score
  ----+-------+-------
    1 | Tom   |    80
    6 |       |
    4 | Perry |    95
    2 | Lily  |    75

  EXPLAIN SELECT ms.id, s.name, ms.score FROM student s RIGHT JOIN math_score ms on (s.id = ms.id);
                                    QUERY PLAN
  ------------------------------------------------------------------------------------------
   id |                operation                | E-rows | E-memory | E-width | E-costs
  ----+-----------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)            |     30 |          |      13 | 19.47
    2 |    ->  Hash Left Join (3, 4)            |     30 | 1MB      |      13 | 11.47
    3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       8 | 10.10
    4 |       ->  Hash                          |     12 | 16MB     |       9 | 1.28
    5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
    6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Left Join (3, 4)
          Hash Cond: (ms.id = s.id)

    ====== Query Summary =====
  -------------------------------
   System available mem: 1761280KB
   Query Max mem: 1761280KB
   Query estimated mem: 5424KB
  (19 rows)
  ```

  对于右外连接，在join算子中却显示的是Left。这是因为，右外连接其实就是将左右表进行交互后的左外连接，所以数据库内部实现为了减少处理逻辑，会将右外连接转为左外连接。

- 全外连接（FULL JOIN）

  语法：

  ```
  left_table FULL [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：全外连接是左外连接和右外连接的综合。全外连接的结果集包括left_table和right_table的所有行，而不仅是连接列所匹配的行。如果left_table的某行在right_table中没有匹配行，则在相关联的结果集行中right_table的列均为空值。如果right_table的某行在left_table中没有匹配行，则在相关联的结果集行中left_table的列均为空值。

  示例6：学生表和数学成绩表进行全外连接，学生表中id为3的行在结果集中对应的右表数据用NULL填充，数学成绩表中id为6的行在结果集中对应的左表数据用NULL填充。

 版权所有 © 华为云计算技术有限公司

```
SELECT s.id, s.name, ms.id, ms.score FROM student s FULL JOIN math_score ms ON (s.id = ms.id);
 id | name  | id | score
----+-------+----+-------
  2 | Lily  |  2 |    75
  4 | Perry |  4 |    95
  1 | Tom   |  1 |    80
  3 | Tina  |    |
    |       |  6 |
(5 rows)

EXPLAIN SELECT s.id, s.name, ms.id, ms.score FROM student s FULL JOIN math_score ms ON (s.id =
ms.id);
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                operation                 | E-rows | E-memory | E-width | E-costs
----+------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)             |     30 |          |      17 | 20.24
  2 |    ->  Hash Full Join (3, 5)             |     30 | 1MB      |      17 | 12.24
  3 |       ->  Streaming(type: REDISTRIBUTE)  |     30 | 2MB      |       8 | 11.06
  4 |          ->  Seq Scan on math_score ms   |     30 | 1MB      |       8 | 10.10
  5 |       ->  Hash                           |      4 | 16MB     |       9 | 1.11
  6 |          ->  Streaming(type: REDISTRIBUTE) |   4 | 2MB      |       9 | 1.11
  7 |             ->  Seq Scan on student s    |      4 | 1MB      |       9 | 1.01

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Full Join (3, 5)
        Hash Cond: (ms.id = s.id)

  ====== Query Summary =====
--------------------------------
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 6496KB
(20 rows)
```

多表查询中 on 条件和 where 条件的区别

从上面各种连接语法中可见，除自然连接和交叉连接外，其他都需要有on条件（using在查询解析过程中会被转为on条件）来限制两表连接的结果。通常在查询的语句中也都会有where条件限制查询结果。这里说的on连接条件和where过滤条件是指不含可以下推到表上的过滤条件。on和where的区别是：

- on条件是两表连接的约束条件。
- where是对两表连接后产生的结果集再次进行过滤。

简单总结就是：on条件优先于where条件，在两表进行连接时被应用；生成两表连接结果集后，再应用where条件。

7.3 子查询表达式

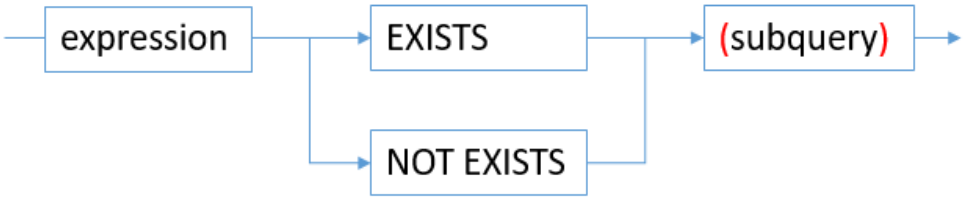
子查询允许在一个查询中嵌套另一个查询，从而实现更复杂的数据查询和分析。

子查询表达式

- EXISTS/NOT EXISTS

  首先执行子查询，然后根据子查询的结果是否满足条件来决定是否继续执行主查询。如果子查询至少返回一行，则EXISTS结果就为“真”。如果子查询没有返回任何行， NOT EXISTS的结果是“真”。

 版权所有 © 华为云计算技术有限公司

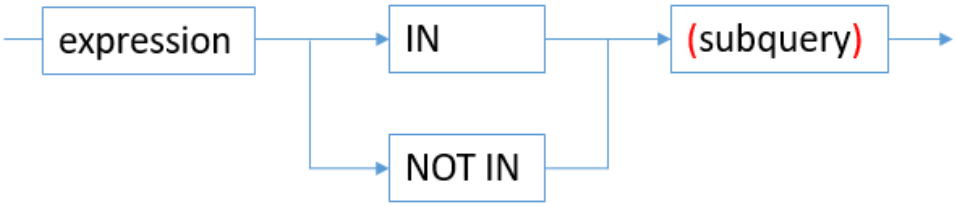


语法：

```
WHERE column_name EXISTS/NOT EXISTS (subquery)
```

- IN/NOT IN

  用于测试某个给定的比较值是否存在于某一组值里。如果外层查询里的行与子查询返回的某一个行相匹配，那么IN的结果为“真”。如果外层查询中的行与子查询返回的所有行都不匹配，那么NOT IN的结果为“真”。



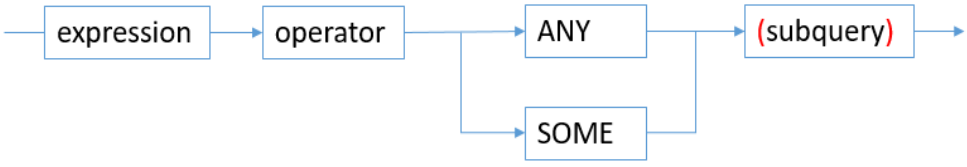
语法：

```
WHERE column_name IN/NOT IN (subquery)
```

- ANY/SOME

  ANY表示子查询中的任何值都可以与外部查询中的值匹配。SOME与ANY相同，只是在语法上的差别。

  右边的子查询必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较（=、<>、<、<=、>、>=），其结果必须是布尔值。如果至少获得一个真值，则ANY结果为“真”。如果全部获得假值，则结果是“假”（包括子查询没有返回任何行的情况）。

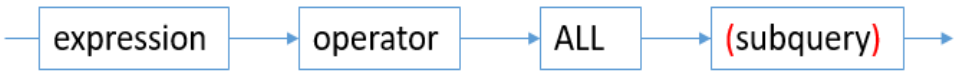


语法：

```
WHERE column_name operator ANY/SOME (subquery)
```

- ALL

  右边的子查询必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较（=、<>、<、<=、>、>=），其结果必须是布尔值。如果全部获得真值，ALL结果为“真”（包括子查询没有返回任何行的情况）。如果至少获得一个假值，则结果是“假”。



语法：

```
WHERE column_name operator ALL (subquery)
```

 版权所有 © 华为云计算技术有限公司

表 7-1 ALL 查询条件

条件	描述
column_name > ALL(…)	column_name列中的值必须大于要评估为true的集合中的最大值。
column_name >= ALL(…)	column_name列中的值必须大于或等于要评估为true的集合中的最大值。
column_name < ALL(…)	column_name列中的值必须小于要评估为true的集合中的最小值。
column_name <= ALL(…)	column_name列中的值必须小于或等于要评估为true的集合中的最小值。
column_name <> ALL(…)	column_name列中的值不得等于要评估为true的集合中的任何值。
column_name = ALL(…)	column_name列中的值必须等于要评估为true的集合中的任何值。

示例

创建记录课程信息的表course，并插入数据：

```
CREATE TABLE course(cid VARCHAR(10) COMMENT 'No.course',cname VARCHAR(10) COMMENT 'course name',teid VARCHAR(10) COMMENT 'No.teacher');

INSERT INTO course VALUES('01' , 'course1' , '02');
INSERT INTO course VALUES('02' , 'course2' , '01');
INSERT INTO course VALUES('03' , 'course3' , '03');
```

创建教师表teacher，并插入数据：

```
CREATE TABLE teacher(teid VARCHAR(10) COMMENT '教师编号',tname VARCHAR(10) COMMENT '教师姓名');

INSERT INTO teacher VALUES('01' , 'teacher1');
INSERT INTO teacher VALUES('02' , 'teacher2');
INSERT INTO teacher VALUES('03' , 'teacher3');
INSERT INTO teacher VALUES('04' , 'teacher4');
```

- EXISTS/NOT EXISTS示例

  查询在course表中的教师记录：

  ```
  SELECT * FROM teacher WHERE EXISTS (SELECT * FROM course WHERE course.teid = teacher.teid);
  ```

  ```
   teid |  tname
  ------+----------
   02   | teacher2
   01   | teacher1
   03   | teacher3
  (3 rows)
  ```

  查询没有在course表中的教师记录：

  ```
  SELECT * FROM teacher WHERE NOT EXISTS (SELECT * FROM course WHERE course.teid = teacher.teid);
  ```

  ```
   teid |  tname
  ------+----------
   04   | teacher4
  (1 row)
  ```


版权所有 © 华为云计算技术有限公司

- IN/NOT IN 示例

  根据教师id匹配course表，查询在course表的教师信息：

  ```
  SELECT * FROM course WHERE teid IN (SELECT teid FROM teacher );
  ```

  ```
   cid |  cname  | teid
  -----+---------+------
   01  | course1 | 02
   03  | course3 | 03
   02  | course2 | 01
  (3 rows)
  ```

  查询不在course表的教师信息：

  ```
  SELECT * FROM teacher WHERE teid NOT IN (SELECT teid FROM course );
  ```

  ```
   teid |  tname
  ------+----------
   04   | teacher4
  (1 row)
  ```

- ANY/SOME 示例

  左侧主句与右侧子查询进行字段比对，获取需要的结果集：

  ```
  SELECT * FROM course WHERE teid < ANY (SELECT teid FROM teacher where teid<>'04');
  ```

  或

  ```
  SELECT * FROM course WHERE teid < some (SELECT teid FROM teacher where teid<>'04');
  ```

  ```
  cid |  cname  | teid
  -----+---------+------
   01  | course1 | 02
   02  | course2 | 01
  (2 rows)
  ```

- ALL示例

  查询teid列中的值必须小于要评估为true的集合中的最小值：

  ```
  SELECT * FROM course WHERE teid < ALL(SELECT teid FROM teacher WHERE teid<>'01');
  ```

  ```
   cid |  cname  | teid
  -----+---------+------
   02  | course2 | 01
  (1 row)
  ```

注意事项

- 禁止一条SQL语句中，出现重复子查询语句。
- 尽量少用标量子查询（标量子查询指结果为1个值，并且条件表达式为等值的子查询）。
- 避免在SELECT目标列中使用子查询，可能导致计划无法下推影响执行性能。
- 子查询嵌套深度建议不超过2层。由于子查询会带来临时表开销，过于复杂的查询应考虑从业务逻辑上进行优化。

子查询可以在 SELECT 语句中嵌套其他查询，从而实现更复杂的查询。子查询还可以在WHERE子句中使用其他查询的结果，从而更好地过滤数据。但是子查询可能会导致查询性能问题和代码难阅读和理解。 所以在DWS数据库中使用SQL子查询时，请结合实际业务情况进行操作。

 版权所有 © 华为云计算技术有限公司

7.4 WITH 表达式

WITH表达式用于定义在大型查询中使用的辅助语句，这些辅助语句通常被称为公共表达式或CTE（即common table expr），可以理解为一个带名称的子查询，之后该子查询可以以其名称在查询中被多次引用。

WITH表达式中的辅助语句可以是SELECT、INSERT、UPDATE或DELETE，并且WITH子句本身也可以被附加到一个主语句中，主语句可以是SELECT、INSERT或DELETE。

WITH 中的 SELECT

在WITH子句中使用SELECT的相关信息。

语法格式

```
[WITH [RECURSIVE] with_query [,…] ] SELECT …
```

其中，with_query的语法为：

```
with_query_name [ ( column_name [, ...] ) ]
    AS [ [ NOT ] MATERIALIZED ] ( {select | values | insert | update | delete} )
```


版权所有 © 华为云计算技术有限公司

⚠ 注意

- 显示指定MATERIALIZED时，将子查询执行一次，并将其结果集进行物化；指定NOT MATERIALIZED时，则将其子查询替换到主查询中的引用处。
- 每个CTE的AS语句指定的SQL语句，必须是可以返回查询结果的语句，可以是普通的SELECT查询语句，也可以是INSERT、UPDATE、DELETE、VALUES等其它数据修改语句，使用数据修改语句时需要通过RETURNING子句返回元组。例如：
```
WITH s AS (INSERT INTO t VALUES(1) RETURNING a) SELECT * FROM s;
```
- 单个WITH表达式表示一个SQL语句块中的CTE定义，可以同时定义多个CTE，每个CTE可以指定列名，也可以默认使用查询输出列的别名。例如：
```
WITH s1(a, b) AS (SELECT x, y FROM t1), s2 AS (SELECT x, y FROM t2) SELECT * FROM s1 JOIN s2 ON s1.a=s2.x;
```
  该语句中定义了两个CTE，s1和s2，其中s1指定了列名为a，b，s2未指定列名，则列名为输出列名x，y。
- 每个CTE可以在主查询中引用0次、1次或多次。
- 同一个语句块中不能出现同名的CTE，但不同语句块中可以出现同名的CTE，此时，语句中引用的CTE则是距离引用位置最近的语句块中的CTE。
- 由于SQL语句中可能包含多个SQL语句块，每个语句块都可以包含一个WITH表达式，每个WITH表达式中的CTE可以在当前语句块、当前语句块的后续CTE中，以及子层语句块中引用，但不能在父层语句块中引用。由于每个CTE的定义也是个语句块，因此也支持在该语句块中定义WITH表达式。
- 当CTE由于指定MATERIALIZED或被引用多次而物化时，CTE整体将作为一个表看待，所以CTE中查询或filter条件无法单独提升到CTE外进行其他优化。如下示例，CTE是中的filter条件无法与jon条件进行等价类推导，所以无法使join的分区表采用分区剪枝优化：
```
CREATE TABLE date_part(ds date) distribute by hash(ds);

CREATE TABLE t1(id1 int, ds date) distribute by hash(ds)
partition by range(ds)
 ( partition p1 values less than('20250101'),
   partition p2 values less than('20250201'),
   partition p3 values less than('20250301'),
   partition p4 values less than('20250401'),
   partition p5 values less than('20250501'),
   partition p6 values less than('20250601'));

CREATE TABLE t3(id3 int, ds date) distribute by hash(ds)
partition by range(ds)
 ( partition p1 values less than('20250101'),
   partition p2 values less than('20250201'),
   partition p3 values less than('20250301'),
   partition p4 values less than('20250401'),
   partition p5 values less than('20250501'),
   partition p6 values less than('20250601'));

explain verbose -- CTE被物化时，无法将CTE中的filter条件提升与join条件进行等价类推导来使join的分区表应用分区剪枝
 with dt as (select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
 t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
 t2 as (select id3 from t3 join dt t2 on t3.ds = t2.ds) select * from t1 join t2 on t1.id1 = 100;

explain with dt as (select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
t2 as(select id3 from t3 join dt t2 on t3.ds = t2.ds)
select * from t1 join t2 on t1.id1 = 100;
                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 id |                       operation                       | E-rows | E-memory | E-width | E-costs
```


版权所有 © 华为云计算技术有限公司

```
----+-----------------------------------------------------------------+--------+----------+---------+---------
 1 | ->  Row Adapter                                    |      2 |          |       8 | 66.27
 2 |    ->  Vector Streaming (type: GATHER)                    |      2 |          |       8 | 66.27
 3 |       ->  Vector Nest Loop (4,15)                  |      2 | 1MB      |       8 | 60.27
 4 |         ->  Vector CTE Append(5, 7)                |      2 | 1MB      |      12 | 40.26
 5 |           ->  Vector Streaming(type: LOCAL GATHER)         |      1 | 2MB      |       8 | 20.01
 6 |             ->  CStore Scan on date_part  [5, CTE dt(0)] |      1 | 1MB      |       8 | 20.01
 7 |           ->  Vector Nest Loop (8,9)               |      2 | 1MB      |      12 | 20.25
 8 |             ->  Vector CTE Scan on dt(0) t2              |      1 | 1MB      |       8 | 0.02
 9 |             ->  Vector Materialize                 |      4 | 16MB     |       4 | 20.22
10 |               ->  Vector Streaming(type: BROADCAST)      |      4 | 2MB      |       4 | 20.21
11 |                 ->  Vector Nest Loop (12,14)             |      2 | 1MB      |       4 | 20.04
12 |                   ->  Vector Partition Iterator        |      1 | 1MB      |      12 | 20.00
13 |                     ->  Partitioned CStore Scan on t1 |      1 | 1MB      |      12 | 20.00
14 |                   ->  Vector CTE Scan on dt(0) t2      |      1 | 1MB      |       8 | 0.02
15 |         ->  Vector Partition Iterator              |      1 | 1MB      |      12 | 20.00
16 |           ->  Partitioned CStore Scan on t3              |      1 | 1MB      |      12 | 20.00

                                        Predicate Information (identified by plan id)

-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------
  3 --Vector Nest Loop (4,15)
        Join Filter: (t2.ds = t3.ds)
  6 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
 11 --Vector Nest Loop (12,14)
        Join Filter: (public.t1.ds = t2.ds)
 12 --Vector Partition Iterator
        Iterations: 6
 13 --Partitioned CStore Scan on t1
        CU Predicate Filter: (id1 = 100)
        Pushdown Predicate Filter: (id1 = 100)
        Partitions Selected by Static Prune: 1..6
 15 --Vector Partition Iterator
        Iterations: 6
 16 --Partitioned CStore Scan on t3
        Partitions Selected by Static Prune: 1..6

  ====== Query Summary =====
-------------------------------
System available mem: 2990080KB
Query Max mem: 2990080KB
Query estimated mem: 11312KB
Turbo Engine: true
(45 rows)


explain  -- CTE不被物化时，会将CTE被展开为子查询，其中的filter条件可与join条件进行等价类推导使join
的分区表应用分区剪枝
with dt as not materialized
(select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
t2 as(select id3 from t3 join dt t2 on t3.ds = t2.ds) select * from t1 join t2 on t1.id1 = 100;
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------
 id |                       operation                       | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                                    |      2 |          |       8 | 52.26
  2 |    ->  Vector Streaming (type: GATHER)                    |      2 |          |       8 | 52.26
  3 |       ->  Vector Nest Loop (4,6)                   |      2 | 1MB      |       8 | 46.26
  4 |         ->  Vector Streaming(type: BROADCAST)         |      2 | 2MB      |       8 | 20.10
  5 |           ->  CStore Scan on date_part               |      1 | 1MB      |       8 | 20.01
  6 |         ->  Vector Materialize                     |      2 | 16MB     |      16 | 26.15
  7 |           ->  Vector Nest Loop (8,11)                |      2 | 1MB      |      16 | 26.14
```


版权所有 © 华为云计算技术有限公司

```
 8 |              ->  Vector Streaming(type: BROADCAST)   |      2 | 2MB       |      12 | 3.10
 9 |                ->  Vector Partition Iterator        |      1 | 1MB       |      12 | 3.01
10 |                  ->  Partitioned CStore Scan on t3 |      1 | 1MB       |      12 | 3.01
11 |              ->  Vector Nest Loop (12,14)            |      2 | 1MB       |       4 | 23.03
12 |                ->  Vector Partition Iterator        |      1 | 1MB       |      12 | 3.01
13 |                  ->  Partitioned CStore Scan on t1 |      1 | 1MB       |      12 | 3.01
14 |                ->  CStore Scan on date_part         |      1 | 1MB       |       8 | 20.01

                                                  Predicate Information (identified by plan id)

-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------
  3 --Vector Nest Loop (4,6)
        Join Filter: (t3.ds = public.date_part.ds)
  5 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
  9 --Vector Partition Iterator
        Iterations: 1
 10 --Partitioned CStore Scan on t3
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
        Partitions Selected by Static Prune: 6
 11 --Vector Nest Loop (12,14)
        Join Filter: (public.t1.ds = public.date_part.ds)
 12 --Vector Partition Iterator
        Iterations: 1
 13 --Partitioned CStore Scan on t1
        CU Predicate Filter: ((id1 = 100) AND (ds >= '2025-05-04 00:00:00'::timestamp without time zone)
AND (ds < '2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((id1 = 100) AND (ds >= '2025-05-04 00:00:00'::timestamp without
time zone) AND (ds < '2025-05-06 00:00:00'::timestamp without time zone))
        Partitions Selected by Static Prune: 6
 14 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))

  ====== Query Summary =====
 ------------------------------
 System available mem: 2990080KB
 Query Max mem: 2990080KB
 Query estimated mem: 12352KB
 Turbo Engine: true
(48 rows)
```

WITH中SELECT的基本价值是将复杂的查询分解称为简单的部分。示例如下：

```
WITH regional_sales AS (
    SELECT region, SUM(amount) AS total_sales
    FROM orders
    GROUP BY region
), top_regions AS (
    SELECT region
    FROM regional_sales
    WHERE total_sales > (SELECT SUM(total_sales)/10 FROM regional_sales)
)
SELECT region,
       product,
       SUM(quantity) AS product_units,
       SUM(amount) AS product_sales
FROM orders
WHERE region IN (SELECT region FROM top_regions)
GROUP BY region, product;
```

 版权所有 © 华为云计算技术有限公司

WITH子句定义了两个辅助语句regional_sales和top_regions，其中regional_sales的输出用在top_regions中而top_regions的输出用在主SELECT查询。这个例子可以不用WITH来书写，但是就必须要用两层嵌套的子SELECT，使得查询更长更难以维护。

WITH 递归查询

通过声明RECURSIVE关键字，一个WITH查询可以引用它自己的输出。

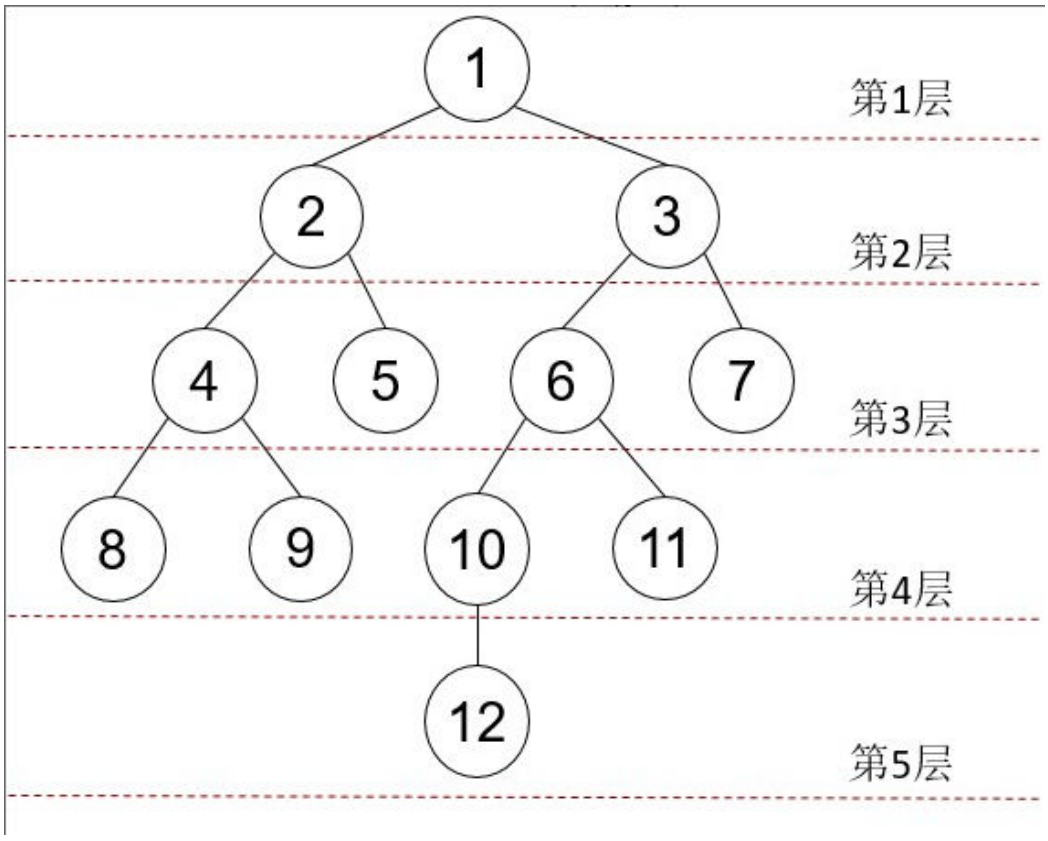
递归WITH查询的通常形式如下：

```
non_recursive_term UNION [ALL] recursive_term
```

其中：UNION在合并集合时会执行去重操作，而UNION ALL则直接将结果集合并、不执行去重；只有递归项能够包含对于查询自身输出的引用。

使用递归WITH时，必须确保查询的递归项最终不会返回元组，否则查询将无限循环。

使用表tree来存储下图中的所有节点信息：



表定义语句如下：

```
CREATE TABLE tree(id INT, parentid INT);
```

表中数据如下：

```
INSERT INTO tree VALUES(1,0),(2,1),(3,1),(4,2),(5,2),(6,3),(7,3),(8,4),(9,4),(10,6),(11,6),(12,10);

SELECT * FROM tree;
 id | parentid
----+----------
  1 |        0
  2 |        1
  3 |        1
```

 版权所有 © 华为云计算技术有限公司

```
 4 |       2
 5 |       2
 6 |       3
 7 |       3
 8 |       4
 9 |       4
10 |       6
11 |       6
12 |      10
(12 rows)
```

通过以下WITH RECURSIVE语句，可以返回从顶层1号节点开始，整个树的节点，以及层次信息：

```
WITH RECURSIVE nodeset AS
(
-- recursive initializing query
SELECT id, parentid, 1 AS level FROM tree
WHERE id = 1
UNION ALL
-- recursive join query
SELECT tree.id, tree.parentid, level + 1 FROM tree, nodeset
WHERE tree.parentid = nodeset.id
)
SELECT * FROM nodeset ORDER BY id;
```

上述查询中，我们可以看出，一个典型的WITH RECURSIVE表达式包含至少一个递归查询的CTE，该CTE中的定义为一个UNION ALL集合操作，第一个分支为递归起始查询，第二个分支为递归关联查询，需要自引用第一部分进行不断递归关联。该语句执行时，递归起始查询执行一次，关联查询执行若干次并将结果叠加到起始查询结果集中，直到某一些关联查询结果为空，则返回。

上述查询的执行结果如下：

```
 id | parentid | level
----+----------+-------
  1 |        0 |     1
  2 |        1 |     2
  3 |        1 |     2
  4 |        2 |     3
  5 |        2 |     3
  6 |        3 |     3
  7 |        3 |     3
  8 |        4 |     4
  9 |        4 |     4
 10 |        6 |     4
 11 |        6 |     4
 12 |       10 |     5
(12 rows)
```

从返回结果可以看出，起始查询结果包含level=1的结果集，关联查询执行了五次，前四次分别输出level=2,3,4,5的结果集，在第五次执行时，由于没有parentid和输出结果集id相等的记录，也就是再没有多余的孩子节点，因此查询结束。

说明

对于WITH RECURSIVE表达式，DWS支持其分布式执行。由于WITH RECURSIVE涉及到循环运算，DWS引入了参数max_recursive_times，用于控制WITH RECURSIVE的最大循环次数，默认值为200，超过该次数则报错。

WITH 中的数据修改语句

在WITH子句中使用数据修改命令INSERT、UPDATE、DELETE。这允许用户在同一个查询中执行多个不同操作。示例如下所示：

版权所有 © 华为云计算技术有限公司

```
WITH moved_tree AS (
    DELETE FROM tree
    WHERE parentid = 4
    RETURNING * )
 INSERT INTO tree_log
 SELECT * FROM moved_tree;
```

上述查询示例实际上从tree把行移动到tree_log。WITH中的DELETE删除来自tree的指定行，以它的RETURNING子句返回它们的内容，并且接着主查询读该输出并将它插入到tree_log。

WITH子句中的数据修改语句必须有RETURNING子句，用来返回RETURNING子句的输出，而不是数据修改语句的目标表，RETURNING子句形成了可以被查询的其余部分引用的临时表。如果一个WITH中的数据修改语句缺少一个RETURNING子句，则它形不成临时表并且不能在剩余的查询中被引用。

如果声明了RECURSIVE关键字，则不允许在数据修改语句中进行递归自引用。在某些情况中可以通过引用递归WITH的输出来绕过这个限制，例如：

```
WITH RECURSIVE included_parts(sub_part, part) AS (
    SELECT sub_part, part FROM parts WHERE part = 'our_product'
  UNION ALL
    SELECT p.sub_part, p.part
    FROM included_parts pr, parts p
    WHERE p.part = pr.sub_part
  )
DELETE FROM parts
  WHERE part IN (SELECT part FROM included_parts);
```

这个查询将会移除一个产品的所有直接或间接子部件。

WITH子句中的子语句与主查询同时执行。因此，在使用WITH中的数据修改语句时，指定更新的顺序实际是以不可预测的方式发生的。所有的语句都使用同一个快照中执行，语句的效果在目标表上不可见。这减轻了行更新的实际顺序的不可预见性的影响，并且意味着RETURNING数据是在不同WITH子语句和主查询之间传达改变的唯一方法。

本示例中外层SELECT可以返回更新之前的数据：

```
WITH t AS (
    UPDATE tree SET id = id + 1
    RETURNING * )
SELECT * FROM tree;
```

本示例中外部SELECT将返回更新过的数据：

```
WITH t AS (
UPDATE tree SET id = id + 1
    RETURNING * )
SELECT * FROM t;
```

不支持在单个语句中更新同一行两次。这种语句的效果是不可预测的。如果只有一个修改发生了，但却不容易（有时也不可能）预测哪一个发生了修改。

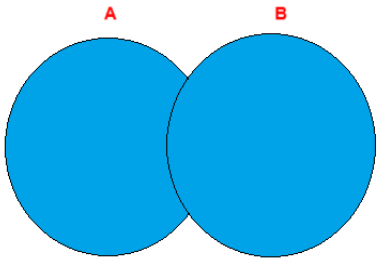
7.5 UNION 操作符的使用

在SQL中，UNION是一个非常强大的功能，UNION操作符用于合并两个或多个SELECT语句的结果集。合并时，两个表对应的列数和数据类型必须相同，并且相互对应。各个SELECT语句之间使用UNION或UNION ALL关键字分隔。

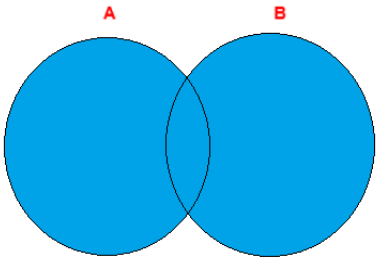
UNION在合并两个或多个表时会执行去重操作，而UNION ALL则直接将两个或者多个结果集合并，不执行去重。 另外，执行去重会消耗大量的时间，因此，在一些实际应

 版权所有 © 华为云计算技术有限公司

用场景中，如果通过业务逻辑已确认了两个集合不存在重复数据时，可直接用UNION ALL替代UNION，以便提升性能。



UNION操作符返回两个查询的结果集的并集，去除重复记录。



UNION ALL操作符返回两个查询的结果集的并集。对于两个结果集的重复部分，不去重。

语法格式

```
SELECT column,... FROM table1 UNION [ALL]SELECT column,... FROM table2
```

示例

步骤1 创建学生信息表student（ID、姓名、性别、学校）。

```
SET current_schema=public;
DROP TABLE IF EXISTS student;
CREATE table student(
sId VARCHAR(10) NOT NULL,
sname VARCHAR(10) NOT NULL,
sgender VARCHAR(10) NOT NULL,
sschool VARCHAR(10) NOT NULL);
```

步骤2 给表student插入数据。

```
INSERT INTO student VALUES('s01' , 'ZhaoLei' , 'male', 'NENU');
INSERT INTO student VALUES('s02' , 'QianDian' , 'male', 'SJTU');
INSERT INTO student VALUES('s03' , 'SunFeng' , 'male', 'Tongji');
INSERT INTO student VALUES('s04' , 'LiYun' , 'male', 'CCOM');
INSERT INTO student VALUES('s05' , 'ZhouMei' , 'female', 'FuDan');
INSERT INTO student VALUES('s06' , 'WuLan' , 'female', 'WHU');
INSERT INTO student VALUES('s07' , 'ZhengZhu' , 'female', 'NWAFU');
INSERT INTO student VALUES('s08' , 'ZhangShan' , 'female', 'Tongji');
```

步骤3 查看表student。

```
SELECT * FROM student;
```

回显如下：

```
 sid |   sname   | sgender | sschool
-----+-----------+---------+---------
 s02 | QianDian  | male    | SJTU
 s05 | ZhouMei   | female  | FuDan
 s08 | ZhangShan | female  | Tongji
 s03 | SunFeng   | male    | Tongji
 s06 | WuLan     | female  | WHU
 s01 | ZhaoLei   | male    | NENU
 s04 | LiYun     | male    | CCOM
 s07 | ZhengZhu  | female  | NWAFU
(8 rows)
```

步骤4 创建教师信息表teacher（ID、姓名、性别、学校）。

```
DROP TABLE IF EXISTS teacher;
CREATE table teacher(
```

 版权所有 © 华为云计算技术有限公司

```
tid VARCHAR(10) NOT NULL,
tname VARCHAR(10) NOT NULL,
tgender VARCHAR(10) NOT NULL,
tschool VARCHAR(10) NOT NULL);
```

步骤5 给表teacher插入数据。

```
INSERT INTO teacher VALUES('t01' , 'ZhangLei', 'male', 'FuDan');
INSERT INTO teacher VALUES('t02' , 'LiLiang', 'male', 'WHU');
INSERT INTO teacher VALUES('t03' , 'WangGang', 'male', 'Tongji');
```

步骤6 查询表teacher。

```
SELECT * FROM teacher;
```

```
 tid |  tname   |tgender| tschool
-----+----------+------+---------
 t03 | WangGang | male | Tongji
 t02 | LiLiang  | male | WHU
 t01 | ZhangLei | male | FuDan
(3 rows)
```

步骤7 使用UNION（合并且去重）获取学生和教师所在学校，并按学校名称首字母升序排序。

```
SELECT t.school  FROM (
    SELECT sschool AS school
     FROM student
     UNION
     SELECT tschool AS school
     FROM teacher
 ) t
 ORDER BY t.school ASC;
```

回显如下：

```
 school
--------
 CCOM
 FuDan
 NENU
 NWAFU
 SJTU
 Tongji
 WHU
(7 rows)
```

步骤8 使用UNION ALL（合并不去重）获取所有学生和教师所在学校，并按学校名称首字母升序排序。

```
SELECT t.school  FROM (
    SELECT sschool AS school
     FROM student
     UNION ALL
     SELECT tschool AS school
     FROM teacher
 ) t
 ORDER BY t.school ASC;
```

 版权所有 © 华为云计算技术有限公司

```
 school
--------
 CCOM
 FuDan
 FuDan
 NENU
 NWAFU
 SJTU
 Tongji
 Tongji
 Tongji
 WHU
 WHU
(11 rows)
```

步骤9 使用UNION ALL（合并带有WHERE子句SQL结果集）获取来自“Tongji”的学生和教师的所有信息，并按学生和教师的编号升序排序。

```
SELECT t.*  FROM  (
  SELECT Sid AS id,Sname AS name,Sgender AS gender,Sschool AS school
  FROM student
  WHERE Sschool='Tongji'
  UNION ALL
  SELECT Tid AS id,Tname AS name,Tgender AS gender,Tschool AS school
  FROM teacher
  WHERE Tschool='Tongji'
) t
  ORDER BY t.id ASC;
```

----结束

小结

在实际业务场景中，使用UNION和UNION ALL时需要注意以下几点：

- 左右两侧的SQL字段数量和字段类型需要保持一致。
- 业务需求是否需要考虑数据除重（合并前除重还是合并时除重）。
- 根据表中数据量的大小，需要对SQL的执行效率进行评估，从而考虑是否需要选择临时表进行过渡后再合并。
- 需要考虑SQL编写的复杂度，不能为了写SQL而写SQL，需要结合业务需求进行选择。

7.6 SQL on Hudi

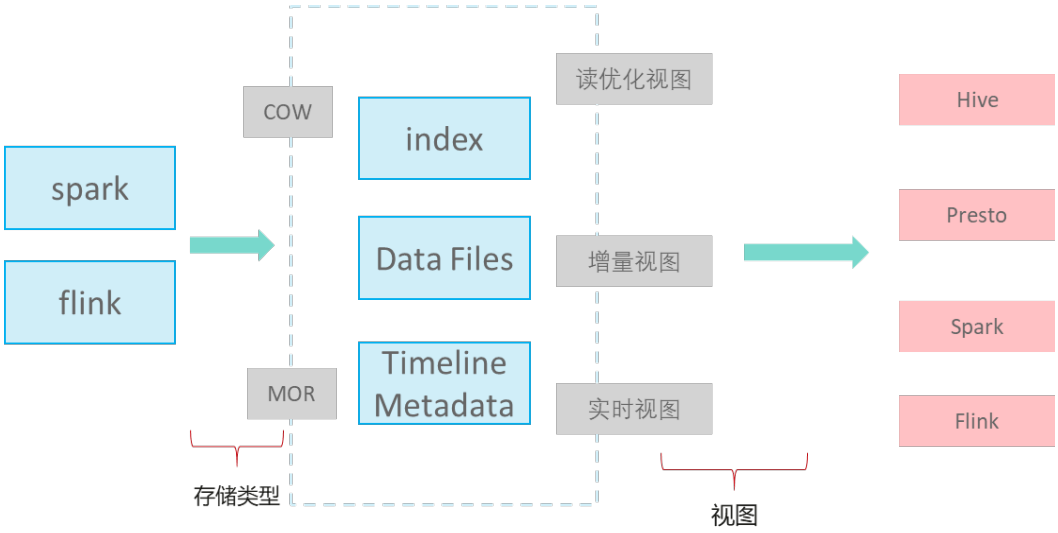
7.6.1 Hudi 简介

DWS的SQL on Hudi特性仅9.1.0.100及以上版本支持。

Apache Hudi表示Hadoop Upserts Deletes and Incrementals。用来管理Hadoop大数据体系下存储在DFS（Distributed File System，分布式文件系统）上大型分析数据集。

Hudi不是单纯的数据格式，而是一套数据访问方法（类似DWS存储的access层），在Apache Hudi 0.9版本，大数据的Spark，Flink等组件都单独实现各自客户端。Hudi的逻辑存储如下图所示：

 版权所有 © 华为云计算技术有限公司



- 写入模式

  COW：写时复制，适合更新少的场景。

  MOR：读时复制，对于UPDATE&DELETE增量写delta log文件，分析时进行base和delta log文件合并，异步compaction合并文件。
- 存储格式

  index：对主键进行索引，默认是file group级别的bloomfilter。

  data files：base file + delta log file（主要面向对base file的update&delete）。

  timeline metadata：版本log的管理。
- 视图

  读优化视图：读取Compaction后生成的base file，未Compaction数据时效性有一定延迟（高效读取）。

  实时视图：读取最新的数据，在读取时进行Base file和Delta file合并（频繁update场景）。

  增量视图：类似CDC方式持续读取增量写入Hudi的数据（流批一体）。

7.6.2 使用 Hudi 前准备

前提条件

- 已完成创建OBS委托以及创建OBS数据源，具体可参见《数据仓库服务用户指南》中的“管理OBS数据源”章节。
- 使用SQL on Hudi前需确保enable_dws_bigdata参数为打开状态。如果使用时出现类似“flight into error”报错，需联系技术支持开启该参数。

授权使用数据源

执行GRANT命令为用户授予使用数据源的权限：

```
GRANT USAGE ON FOREIGN SERVER server_name TO role_name;
```

示例：

给用户u1授予数据源hudi_server的访问权限：

```
GRANT USAGE ON FOREIGN SERVER hudi_server TO u1;
```


版权所有 © 华为云计算技术有限公司

授权使用外表

执行如下命令给用户授予使用外表的权限：

```
ALTER USER role_name USEFT;
```

示例：

给用户u1授予外表访问权限：

```
ALTER USER u1 USEFT;
```

7.6.3 Hudi 用户接口

实时视图与增量视图查询

DWS提供了类似spark-sql风格的表级参数，用于支持实时视图和增量视图。

具体参数说明如下，其中SCHEMA.FOREIGN_TABLE需要替换为实际的schema和外表名。

表 7-2 实时视图与增量视图查询的参数

参数	取值	说明
hoodie.SCHEMA.FOREIGN_TABLE.consume.mode	SNAPSHOT	查询实时视图。
	INCREMENTAL	查询增量视图。
hoodie.SCHEMA.FOREIGN_TABLE.consume.start.timestamp	hudi时间戳	指定增量同步的起始commit。
hoodie.SCHEMA.FOREIGN_TABLE.consume.ending.timestamp	hudi时间戳	指定增量同步的结束commit，不指定则采用最新commit。

说明

- 以上参数支持使用set命令设置，并且仅在当前SESSION中有效。可使用reset命令恢复缺省值。
- 可以通过系统函数pg_catalog.pg_show_custom_settings()来查询相关参数的设置详情。
- 查询MOR表的增量视图时，需要使用where条件过滤_hoodie_commit_time字段，避免读取到未合并的不符合条件的log文件数据。COW表无需该操作。

Hudi 外表信息查询与自动同步任务

DWS提供一系列系统函数来实现Hudi外表信息获取、创建Hudi自动同步任务等功能。其中，Hudi自动同步任务实现了从Hudi外表周期性同步数据到DWS内表功能。

版权所有 © 华为云计算技术有限公司

表 7-3 Hudi 系统函数

名称	类型	功能
pg_show_custom_settings()	内置函数	查询HUDI外表参数设置详情。
hudi_get_options(regclass)	内置函数	查询HUDI外表的属性信息（hoodie.properties）。
hudi_get_max_commit(regclass)	内置函数	获取当前HUDI外表最新commit的时间戳。
hudi_sync_task_submit(regclass, regclass)	内置函数	提交HUDI自动同步任务。
hudi_sync_task_submit(regclass, regclass, text, text)		
hudi_show_sync_state()	内置函数	获取HUDI自动同步任务的同步状态。
hudi_sync(regclass, regclass)	存储过程	HUDI自动同步任务调用入口。
hudi_sync_custom(regclass, regclass, text)	存储过程	HUDI自动同步任务调用入口，支持用户自定义目标表和数据源表的字段同步对应关系。
hudi_set_sync_commit(regclass, regclass, text)	内置函数	设置HUDI自动同步任务首次同步的起点时间戳，避免在已经同步了部分数据的情况下，重新同步已有数据。
hudi_set_sync_commit(text, text)		设置HUDI自动同步任务下一次同步的起点时间戳，可以用于重复同步历史数据或者跳过某些数据。

7.6.4 创建 Hudi 数据（外表）

DWS通过外表方式访问OBS上的Hudi数据，与一般外表相比，Hudi外表无需额外设置更多的参数，只需指定format参数为“hudi”，即可指定数据源文件的格式。具体参见《数据仓库服务SQL语法参考》中的“CREATE FOREIGN TABLE (SQL on OBS or Hadoop)”章节。

获取 MRS 上表定义

DWS Hudi外表为只读，因此在创建外表之前需要明确目标数据定义的字段及字段类型情况。

例如，对于MRS上的Hudi表，可以使用spark-sql来查询原始表定义：

```
SHOW create table rtd_mfdt_int_currency_t;
```

编写 DWS 表定义

复制MRS表所有列的定义，创建hudi外表，进行适当的类型转换以适配DWS语法：

```
CREATE FOREIGN TABLE rtd_mfdt_int_currency_ft(
_hoodie_commit_time text,
```


版权所有 © 华为云计算技术有限公司

```
_hoodie_commit_seqno text,
_hoodie_record_key text,
_hoodie_partition_path text,
_hoodie_file_name text,
...
)SERVER hudi_server OPTIONS (
foldername '/erpgc-test-01/s000/u1/rtd_mfdt_int_currency_t/',
format 'hudi',
encoding 'utf-8'
)distribute by roundrobin;
```

其中，foldername为hudi数据在OBS上存储路径，对应MRS中Spark-sql表定义中的LOCATION，末尾要以“/”结尾。

7.6.5 Hudi 任务同步

创建 Hudi 任务

迁移场景

如果DWS表已经通过CDL导入数据，改为用SQL on Hudi方式迁移数据。或者使用CDM做全量初始化后，继续使用SQL on Hudi方式同步增量数据。

步骤1 创建hudi.hudi_sync_state同步状态表，需要管理员权限。

```
SELECT pg_catalog.create_hudi_sync_table();
```

通常情况下，每个数据库中只创建一次hudi.hudi_sync_state。

步骤2 设置同步进度，用户需要有同步目标表的INSERT和UPDATE权限、HUDI外表的SELECT权限，否则无法正常设置同步进度。

```
SELECT hudi_set_sync_commit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE', 'LATEST_COMMIT');
```

其中：

- SCHEMA.TABLE，表示同步数据的目标表名，schema为模式名。
- SCHEMA.FOREIGN_TABLE，表示OBS外表名，schema为模式名。
- LATEST_COMMIT，表示已同步的Hudi数据时间截点。

示例：目标表public.in_rel，已经同步hudi的数据到20220913152131，切换到SQL on Hudi方式从OBS外表hudi_read1中继续导出数据。

```
SELECT hudi_set_sync_commit('public.in_rel', 'public.hudi_read1', '20220913152131');
```

步骤3 提交Hudi同步任务。

```
SELECT hudi_sync_task_submit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE');
```

例如：目标表public.in_rel，切换到SQL on Hudi方式从OBS外表hudi_read1中继续导出数据。

```
SELECT hudi_sync_task_submit('public.in_rel', 'public.hudi_read1');
```

----结束

新建场景

DWS表为空，第一次从Hudi同步数据，可执行以下命令直接创建任务。

```
SELECT hudi_sync_task_submit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE');
```


版权所有 © 华为云计算技术有限公司

查询 Hudi 同步任务

查询Hudi同步任务。

语法：

```
SELECT * FROM pg_task_show('SQLonHudi');
```

查询结果中的**task_id**是Hudi同步任务唯一标识。

示例：

```
SELECT * FROM pg_task_show('SQLonHudi');
                task_id                 |
what                                                
```

删除 Hudi 同步任务

通过指定的task_id，删除对应的Hudi同步任务。

语法：

```
SELECT pg_task_remove('task_id');
```

示例：

删除task_id为64479410-a04c-0700-d150-3037d700fffe的同步任务。

```
SELECT pg_task_remove('64479410-a04c-0700-d150-3037d700fffe');
```

查询历史 Hudi 同步任务信息

使用视图hudi_sync_state_history_view查询Hudi历史同步任务信息，该视图仅9.1.0及以上集群版本支持。

```
SELECT * FROM pg_catalog.hudi_sync_state_history_view;
```

表 7-4 hudi_sync_state_history_view 字段

名称	类型	描述
task_id	text	任务ID。
target_tbl	text	同步目标表名。
source_ftbl	text	同步源表名（外表）。
latest_commit	text	最近一次同步成功的时间戳。
latest_sync_count	bigint	最近一次同步成功的行数。
latest_sync_start	timestamp with time zone	最近一次同步任务开始的时间。
latest_sync_end	timestamp with time zone	最近一次同步任务结束的时间。
hudi_flushdisk_time	text	hudi文件落盘时间。

查询 Hudi 同步任务状态

使用函数hudi_show_sync_state()查询Hudi同步任务状态。

```
SELECT * FROM hudi_show_sync_state();
```

7.6.6 Hudi 外表查询

Hudi外表可以直接查询数据，默认查询实时视图，也可以通过设置增量查询参数实现增量视图查询。

hudi 外表增量查询

通过设置增量查询参数，进行hudi外表增量查询。

版权所有 © 华为云计算技术有限公司

```
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.mode=incremental;
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.start.timestamp=起始时间戳;
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.ending.timestamp=结束时间戳;
SELECT * FROM SCHEMA.FOREIGN_TABLE;
```

示例:

查询MOR hudi外表public.rtd_mfdt_int_currency_ft从20221207164617到20221207170234之间的增量数据。

```
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.mode=incremental;
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.start.timestamp=20221207164617;
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.ending.timestamp=20221207170234;
SELECT * FROM public.rtd_mfdt_int_currency_ft where _hoodie_commit_time>20221207164617 and
_hoodie_commit_time<=20221207170234;
```

查询 hudi 外表已设置的增量参数

通过函数pg_show_custom_settings()查看hudi外表已设置的增量参数，检查参数设置是否正确。

```
SELECT * FROM pg_show_custom_settings();
```

查询 hudi 外表属性（hoodie.properties）

查询OBS上hudi数据的外表属性hoodie.properties。

语法：

```
SELECT * FROM hudi_get_options('SCHEMA.FOREIGN_TABLE');
```

示例：查询当前schema下的OBS外表rtd_mfdt_int_currency_ft的hudi属性。

```
SELECT * FROM hudi_get_options('rtd_mfdt_int_currency_ft');
```

查询 hudi 外表最大时间线

查询OBS上hudi数据最大时间线，即最新的提交记录。

语法：

```
SELECT * FROM hudi_get_max_commit('SCHEMA.FOREIGN_TABLE');
```

示例：查询当前schema下的OBS外表rtd_mfdt_int_currency_ft的最大时间线。

```
SELECT * FROM hudi_get_max_commit('rtd_mfdt_int_currency_ft');
```

7.6.7 支持访问 MRS 上的 Hudi 表

SQL on Hudi支持对存储在MRS上的hudi表进行访问。

前提条件

已完成创建MRS数据源，具体可参见《数据仓库服务用户指南》中的“管理MRS数据源”章节。

说明

SQL on Hudi读取存储在MRS上的hudi表，在使用上仅创建数据源与OBS不同，其他无差异。

版权所有 © 华为云计算技术有限公司

对接多套 MRS 集群

通常情况下由于JDK的限制，同一JVM同时仅能保存一份kerberos配置文件信息，导致一套DWS集群无法同时通过SQL on Hudi并发访问多套MRS集群上的hudi表。但通过执行以下操作可访问多套MRS集群上的hudi表。

步骤1 获取各MRS集群已下载客户端中的krb5.conf文件。

步骤2 任取一套MRS集群的krb5.conf文件作为待合并文件，以下简称集群A。

步骤3 将集群B的KDC域信息添加到集群A配置文件realms中。

注意

实际操作时需要将集群realms或者domain_realm中实际的KDC域信息进行合并。

示例：

```
[realms]
CLUSTER.A.COM = {
admin_server =  ClusterA_SERVER_IP:PORT
kdc = ClusterA_KDC_IP:PORT
kdc = ClusterA_KDC_IP:PORT
}
CLUSTER.B.COM = {
admin_server = ClusterB_SERVER_IP:PORT
kdc = ClusterB_KDC_IP:PORT
kdc = ClusterB_KDC_IP:PORT
}
```

步骤4 将集群B的域信息添加到集群A配置文件domain_realm中。

示例：

```
[domain_realm]
.cluster.a.com = CLUSTER.A.COM
.cluster.b.com = CLUSTER.B.COM
```

步骤5 将合并后的krb5.conf配置文件更新到各集群各个节点原路径下替换原krb5.conf文件。

----结束

 版权所有 © 华为云计算技术有限公司

8 DWS 排序规则

排序规则(collation)是在字符集中指定数据排序顺序及对数据进行分类的规则。排序规则支持不再受限于数据库的LC_COLLATE和LC_CTYPE设置创建后就不能更改的约束。

概述

一种可排序数据类型的每一种表达式都有一个排序规则（系统内部的可排序数据类型可以是text、varchar和char等字符类型。用户定义的基础类型也可以被标记为可排序的，并且在一种可排序数据类型上的域也是可排序的）。如果该表达式是一个列引用，该表达式的排序规则就是列所定义的排序规则。如果该表达式是一个常量，排序规则就是该常量数据类型的默认排序规则。更复杂表达式的排序规则根据其输入的排序规则得来。

排序规则组合原则

- 当表达式的collation未指定时，则认为是默认的排序规则default，它表示数据库的区域设置。表达式的collation也可能是不确定的，此时，排序操作和其他不确定的排序规则的操作就会失败。
- 对于函数或操作符调用，其排序规则将通过检查所有参数的collation来决定。如果该函数或操作符调用的结果是一种可排序的数据类型，若有外层表达式要用到排序规则，那么该外层的表达式将继承对应函数和操作符所调用结果集的排序规则。
- 表达式的排序规则派生可以是显式或隐式。该区别会影响多个不同的排序规则出现在同一个表达式中时如何对collation进行组合。当执行语句使用COLLATE子句时，将发生显式派生，否则为隐式派生。当多个排序规则组合时，规则如下：
  - 如果输入表达式中存在显式COLLATE派生，则在输入表达式之间的所有显式派生的COLLATE必须相同，否则将产生冲突错误。如果存在显式COLLATE，那它就是排序规则组合的结果。
  - 如果不存在显式COLLATE，那所有输入表达式必须具有相同的隐式COLLATE或默认COLLATE。如果存在非默认COLLATE，那它就是排序规则组合的结果。否则，结果是默认COLLATE。
  - 如果在输入表达式之间存在多个冲突的非默认COLLATE，则组合被认为是具有不确定排序规则，这并非一种错误。如果被调用的函数或表达式需要用到排序规则，运行时将产生排序规则未知的错误。
- CASE表达式中，比较行为使用的规则以WHEN子句中的COLLATE设置为准。
- 显示COLLATE的派生仅在当前查询（CTE或SUBQUERY）中生效，查询外则降为隐式派生。

 版权所有 © 华为云计算技术有限公司

排序规则使用建议

- 同一条查询语句中，避免使用多种排序规则，可能导致非预期的结果集。
- 使用collate子句指定排序规则时，避免连续使用多个collate子句变更排序规则。

大小写不敏感排序规则支持

从集群8.1.3版本开始，DWS增加内置排序规则case_insensitive，即对字符类型的大小写不敏感行为（如排序、比较、哈希）。

约束条件：

- 支持字符类型：char/character/nchar、varchar/character varying/varchar2/nvarchar2/clob/text。
- 不支持字符类型：“char”和name。
- 不支持的编码：PG_EUC_JIS_2004、PG_MULE_INTERNAL、PG_LATIN10、PG_WIN874。
- 不支持CREATE DATABASE时指定到LC_COLLATE。
- 不支持正则表达式。
- 不支持字符类型的record比较（如record_eq）。
- 不支持时序表。
- 不支持倾斜优化。
- 不支持RoughCheck优化。

示例

语句中显示指定COLLATE子句。

```
SELECT 'a' = 'A', 'a' = 'A' COLLATE case_insensitive;
 ?column? | ?column?
----------+----------
 f        | t
(1 row)
```

建表时指定列属性为case_insensitive。

```
CREATE TABLE t1 (a text collate case_insensitive);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
\d t1
            Table "public.t1"
 Column | Type |         Modifiers
--------+------+---------------------------
 a      | text | collate case_insensitive

INSERT INTO t1 values('a'),('A'),('b'),('B');
INSERT 0 4
```

建表时指定，查询时无需指定。

```
SELECT a, a='a' FROM t1;
 a | ?column?
---+----------
 A | t
 B | f
 a | t
 b | f
(4 rows)
```


版权所有 © 华为云计算技术有限公司

```
SELECT a, count(1) FROM t1 GROUP BY a;
 a | count
---+-------
 a |     2
 B |     2
(2 rows)
```

CASE表达式，以WHEN子句中的COLLATE设置为准。

```
SELECT a,case a when 'a' collate case_insensitive then 'case1' when 'b' collate "C" then 'case2' else 'case3'
end FROM t1;
 a | case
---+-------
 A | case1
 B | case3
 a | case1
 b | case2
(4 rows)
```

跨子查询隐式派生。

```
SELECT * FROM (SELECT a collate "C" from t1) WHERE a in ('a','b');
 a
---
 a
 b
(2 rows)
SELECT * FROM t1,(SELECT a collate "C" from t1) t2 WHERE t1.a=t2.a;
ERROR:  could not determine which collation to use for string hashing
HINT:  Use the COLLATE clause to set the collation explicitly.
```

⚠ 注意

- 由于collate case_insensitive为不敏感排序，结果集不确定，再使用敏感排序筛选，会有结果集不稳定的问题，因此语句中避免出现敏感排序和不敏感排序混用。
- 使用collate case_insensitive指定字符类型为大小写不敏感后，性能较使用前会有所下降，因此性能敏感场景需谨慎评估后使用。

 版权所有 © 华为云计算技术有限公司

9 DWS 物化视图

9.1 物化视图概述

物化视图（Materialized View）是一种特殊的数据库对象。物化视图将复杂计算的查询结果预先进行计算并持久化到存储介质中，业务查询时直接查询已经预计算好的数据最终实现加速查询的目的。

DWS的物化视图特性仅8.2.1.220及以上集群支持，使用物化视图需确保**enable_matview**设置为on。

基本概念

表 9-1 物化视图基本概念

概念	说明
基表	• 物化视图定义中所依赖的表，物化视图的数据来自于这些表的计算结果。 • 基表支持DWS 内表、外表、分区表、视图或者物化视图。 • 不支持临时表。
刷新	创建物化视图后，物化视图中存储的数据只反映创建时刻的状态。当基表中的数据发生变化时，需要通过刷新物化视图更新数据变化。 根据刷新模式可划分为：全量刷新、分区刷新。 • 全量刷新：每次刷新重新计算物化视图中的数据。 • 分区刷新：每次刷新物化视图一个分区的数据。
查询重写	通过使用物化视图来加速查询，当输入一个不使用物化视图的查询时，系统会自动将查询重写为使用物化视图的查询。该方法的原理是将查询语句与物化视图的定义进行匹配，如果发现匹配的物化视图，则自动将查询重写为使用物化视图的查询，这样可以利用物化视图预计算的结果，大大提高查询性能和效率。

版权所有 © 华为云计算技术有限公司

异步物化视图和同步物化视图

创建物化视图后，当基表中的数据发生变化时，需要通过刷新物化视图更新数据变化。根据刷新方式，可分为异步物化视图和同步物化视图。

当前DWS只支持异步物化视图。

- 异步物化视图：基表数据变化后需要定期触发物化视图刷新，更新物化视图中数据。
  - 手动刷新：在业务数据修改后通过执行REFRESH MATERIALIZED VIEW进行相应物化视图的刷新。
  - 自动刷新：在物化视图创建时指定刷新开始时间和刷新间隔。由数据库负责定期刷新。
- 同步物化视图：基表数据变化后物化视图数据自动变化。

物化视图优势

- **提升查询性能**

  物化视图预先计算并存储查询结果，当发起查询时，可以查询物化视图直接返回结果，从而提高查询性能。

- **降低计算资源消耗**

  多次执行复杂查询通常需要显著的CPU和内存资源。物化视图通过避免这些重复计算，有助于减少数据库资源的负荷。

- **数仓建模**

  物化视图可以作为数据访问层，支持复杂的业务逻辑，允许用户访问经预处理的数据，无需关心底层复杂SQL逻辑。

物化视图相关语法

创建物化视图。

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```

刷新物化视图。

```
REFRESH MATERIALIZED VIEW
{[schema.]materialized_view_name} [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE
BACKWARD ] | [ CASCADE ALL ] ]
```

删除物化视图。

```
DROP MATERIALIZED VIEW [ IF EXISTS ]
{[schema.]materialized_view_name} [, ...] [ CASCADE | RESTRICT ];
```

修改物化视图。

```
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    [ ENABLE | DISABLE ] QUERY REWRITE;
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
```

 版权所有 © 华为云计算技术有限公司

```
    REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ];
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    REFRESH [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ;
ALTER MATERIALIZED VIEW { materialized_view_name }
    OWNER TO new_owner;
ALTER MATERIALIZED VIEW { materialized_view_name }
    SET ( {storage_parameter = value} [, ... ] )
    | RESET ( storage_parameter [, ... ] )
```

物化视图语法具体参数说明请参考《 SQL语法参考 》的CREATE MATERIALIZED VIEW和ALTER MATERIALIZED VIEW章节。

物化视图应用场景

物化视图适用如下场景：

- 查询结果不经常改变，即基表稳定且执行频次高的查询。
- 查询包含非常耗时的操作，比如聚合、连接操作等。对于涉及多个表连接和聚合操作的查询，物化视图可以将这些耗时的操作预先计算并存储起来，从而避免在每次查询时都重新执行这些操作。
- 查询结果仅涉及表中的很小部分数据。

表 9-2 物化视图和视图、表的对比

数据库对象	性能提升	持久化存储	业务逻辑简化
物化视图	√	√	√
视图	-	-	√
表	-	√	-

操作权限

- 创建物化视图时需要schema的CREATE权限和基表或列的SELECT权限。
- 查询物化视图需要物化视图的SELECT权限。
- 刷新需要物化视图的INSERT和基表或列的SELECT权限。
- 删除物化视图需要有DROP TABLE的权限。

使用限制

- 不支持包含临时表，包括全局临时表、volatile临时表和普通临时表。
- 不支持物化视图定义中的CTE包含除SELECT语句以外的其他DML。
- 不支持基表上存在脱敏策略和行级访问控制或者基表属于私有用户。脱敏策略和行级访问控制可能会导致结果集错误，而私有用户可能导致刷新物化视图失败等问题
- 9.1.0.200及以上版本支持stable和volatile函数。需注意：物化视图无法感知函数内容变化。如果需要函数变化尽快反馈到物化视图结果，指定REFRESH MATERIALIZED VIEW RESTRICT进行物化视图刷新或者跟随数据更新反馈到物化视图。

版权所有 © 华为云计算技术有限公司

- 一些影响查询执行时行为的参数修改，物化视图无法感知参数变化，需要再次刷新物化视图更新结果。

使用建议

- 存储资源：物化视图会占用额外的存储空间，需要考虑磁盘容量。
- 刷新维护：自动或手动刷新物化视图会消耗系统资源，如果基础表数据变化频繁，刷新操作可能会影响系统性能。
- 数据一致性和实时性：异步物化视图中的数据可能不是实时的，即数据不会随着原始数据的更新而自动更新。如果基础表数据发生变化，视图中的数据可能会过时，需要定期刷新以保持数据一致性。
- 设计复杂性：物化视图的设计和创建需要仔细考虑预期的查询模式和数据访问模式，以便实现最佳的性能优化。

9.2 异步物化视图

9.2.1 物化视图查询重写

9.2.1.1 概述

DWS不仅支持查询物化视图，还支持查询重写。物化视图查询重写，让用户在不改变业务情况下自动使用物化视图加速查询执行。通过查询重写功能，物化视图可以像索引一样添加或删除，不会影响业务。

使用场景

物化视图查询重写，能够避免查询中复杂的关联、聚集以及表达式运算操作。

文本匹配查询重写，查询与物化视图的定义基本一致（允许格式、注释有差异）。因为能够支持cte，窗口函数等复杂查询，复杂查询性能较差时可以通过定义与查询语句相同的物化视图，加快查询。

物化视图查询重写方式

基于文本匹配查询重写

当查询和物化视图的SQL或者语法树匹配时，可以使用物化视图进行查询改写。

设置物化视图查询重写

DWS通过GUC参数**mv_rewrite_rule**控制查询重写方式。

9.2.1.2 文本匹配查询重写

DWS支持文本匹配查询重写，包括JOIN、AGG、CTE、子查询等复杂查询。

使用约束

- 查询的子查询也可以使用物化视图进行文本匹配。

版权所有 © 华为云计算技术有限公司

- 如果查询语句有ORDER BY， 要求ORDER BY列在物化视图输出列中存在才可以进行重写。
- 支持查询和物化视图间有空格、回车、注释的差异。
- 物化视图输出列指定别名时，需要查询也指定别名才能匹配成功。
- 物化视图中的查询不能包含FOR UPDATE。

示例

以tpch Q1为例，基于Q1的复杂查询创建物化视图，查看查询计划时，显示语句已经被重写成了对物化视图的扫描。

```
CREATE MATERIALIZED VIEW mv_tpch_q1 enable query rewrite as
SELECT
        l_returnflag,
        l_linestatus,
        sum(l_quantity) AS sum_qty,
        sum(l_extendedprice) AS sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
        avg(l_quantity) AS avg_qty,
        avg(l_extendedprice) AS avg_price,
        avg(l_discount) AS avg_disc,
        count(*) AS count_order
FROM
        lineitem
WHERE l_shipdate <= DATE '1998-12-01' - INTERVAL '3 day'
GROUP BY l_returnflag,l_linestatus
ORDER BY sum_qty;

explain (costs off)
SELECT
        l_returnflag,
        l_linestatus,
        sum(l_quantity) AS sum_qty,
        sum(l_extendedprice) AS sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
        avg(l_quantity) AS avg_qty,
        avg(l_extendedprice) AS avg_price,
        avg(l_discount) AS avg_disc,
        count(*) AS count_order
FROM
        lineitem
WHERE l_shipdate <= DATE '1998-12-01' - INTERVAL '3 day'
GROUP BY l_returnflag,l_linestatus
ORDER BY sum_qty;
                QUERY PLAN
-----------------------------------------------
 id |              operation
 ----+--------------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |    ->  Sort
   3 |       ->  Seq Scan on mv_tpch_q1
```

查询除了ORDER BY部分如果与物化视图定义一致，也可以进行全文匹配。

```
CREATE TABLE t1 (a int,b int);
CREATE MATERIALIZED VIEW mv_t1 enable query rewrite AS SELECT sum(a), b FROM t1 GROUP BY b;

EXPLAIN (costs off) SELECT sum(a), b FROM t1 GROUP BY b ORDER BY 1;
            QUERY PLAN
-----------------------------------
 id |         operation
 ----+----------------------------
   1 | ->  Streaming (type: GATHER)
```

 版权所有 © 华为云计算技术有限公司

```
2 |   ->  Sort
3 |     ->  Seq Scan on mv_t1
```

9.2.2 分区物化视图

9.2.2.1 分区物化视图概述

为了实现物化视图能够按分区快速刷新，可以把物化视图建成分区表，并且与被查询的基表建立分区映射关系。当基表的部分分区发生变化时，刷新语句就可以只查询变化分区，然后把最新数据导入物化视图对应分区即可。

- **视图创建**

  创建时指定用哪张被查询的基表分区键作为物化视图的分区键，即可实现物化视图与基表的自动分区映射。

- **分区失效**
  - 当基表有分区发生数据变化时，会自动将物化视图对应的分区进行失效。
  - 基表并发数据修改时，可并行对物化视图标记失效，且已失效的不会重复标记。

- **增量刷新**
  - 物化视图刷新时会自动识别失效的分区，并仅对失效分区进行刷新。
  - 并发刷新时，如果当前分区正在被刷新则会自动跳过，避免重复刷新。
  - 物化视图的刷新不堵塞基表的数据修改。

- **分区DDL**

  基表分区增加或者删除时，物化视图的分区也会根据映射规则跟随着增加或删除。

9.2.2.2 创建分区物化视图

DWS提供多种场景的分区物化视图创建：

- 等比例对齐分区，物化视图分区条件与基表的分区条件一致。
- 时间粒度上卷对齐分区，例如：基表按天分区，物化视图按月分区。
- 多基表对齐分区，多个基表的分区条件完全一致时，都会自动和物化视图进行映射。
- 基表为多分区键时，物化视图选第一个键为分区键创建物化视图。

语法

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```


版权所有 © 华为云计算技术有限公司

使用约束

- 必须是range分区方式。
- 物化视图只支持单分区键且为基表第一个分区键，基表可以是组合分区键。
- 上卷方式，只支持时间类型且用date_trunc进行上卷计算，仅支持日，月，季，年的上卷。
- 不支持Outer join中会产生空值的列做分区键的情况。
- 不支持mysql兼容模式下的disable full group by。
- 预聚合物化视图不支持创建分区物化视图。
- 指定分区的查询不支持重写。
- 分区物化视图基表不支持SPLIT，MERGE。
- 如果union的两边都满足物化视图的分区约束检测，则创建多表绑定的物化视图。
- 使用JOIN 连接基表，并将分区键设置为公共列，并且连接的基表满足物化视图的分区约束检测，则创建多表绑定的物化视图。
- 不支持查询语句中含有子链接。
- 不支持查询语句中含有子计划。
- 不支持DML和DDL修改数据，只能靠刷新更新数据。

场景示例

创建事实表fact_table和fact_table1。

```
CREATE TABLE fact_table (start_time TIMESTAMPTZ NOT NULL, id int)PARTITION BY RANGE(start_time)(
PARTITION p1 VALUES LESS THAN ('2024-01-01'),
PARTITION p2 VALUES LESS THAN ('2024-01-02'),
PARTITION p3 VALUES LESS THAN ('2024-02-03'),
PARTITION p4 VALUES LESS THAN ('2024-03-01'),
PARTITION p5 VALUES LESS THAN ('2024-03-21'),
PARTITION p6 VALUES LESS THAN ('2024-04-02'));
CREATE TABLE fact_table1 (start_time TIMESTAMPTZ NOT NULL, id int)PARTITION BY RANGE(start_time)(
PARTITION p1 VALUES LESS THAN ('2024-01-01'),
PARTITION p2 VALUES LESS THAN ('2024-01-02'),
PARTITION p3 VALUES LESS THAN ('2024-02-03'),
PARTITION p4 VALUES LESS THAN ('2024-03-01'),
PARTITION p5 VALUES LESS THAN ('2024-03-21'),
PARTITION p6 VALUES LESS THAN ('2024-04-02'));
```

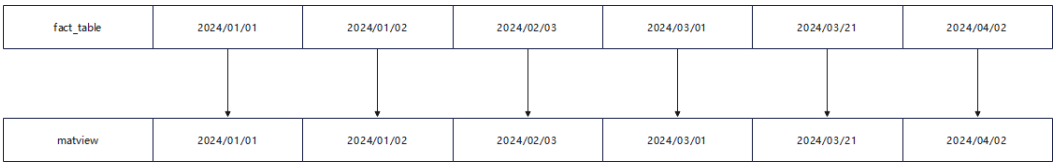
创建维度表dimen_table。

```
CREATE TABLE dimen_table (start_time TIMESTAMPTZ NOT NULL, id INT);
```

场景一：等比例对齐分区。

创建物化视图level_normal，它的分区与fact_table表分区，分区条件完全一样。对数据类型无特殊要求。

```
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
```



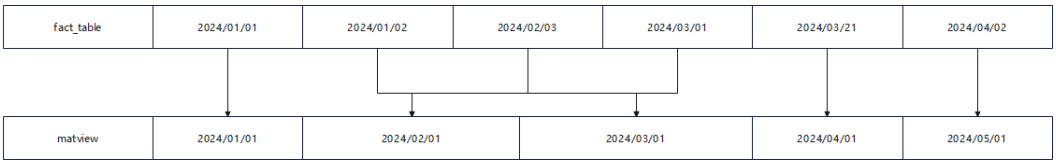
 版权所有 © 华为云计算技术有限公司

场景二：时间粒度上卷对齐分区。

通过在分区键上使用date_trunc参数，创建一个分区时间比基表更粗粒度的物化视图。当检测到基表分区中的数据变更后，DWS数据库将会刷新物化视图中对应的上卷分区。

通过date_trunc('time_unit', column_name)参数指定上卷，物化视图分区范围大于基表分区，比如：基表按天分区，物化视图按月分区。一般通过date_trunc参数，实现时间的向上截取。支持上卷的时间单位：年，季，月，日。数据类型：只支持time, timestamp, timestamptz类型。

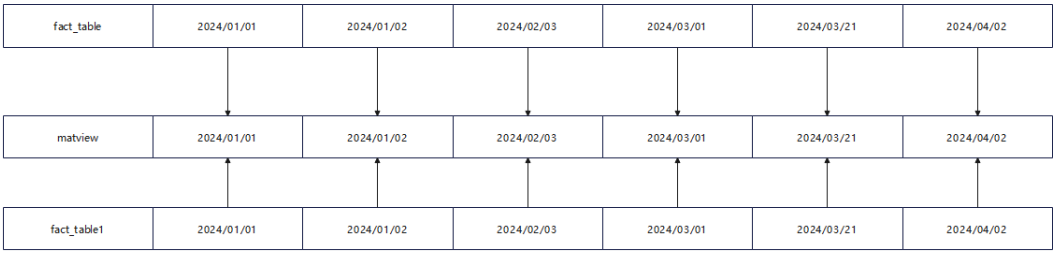
```
CREATE MATERIALIZED VIEW level_month enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY date_trunc('month', start_time)
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
```



场景三：多基表对齐分区。

当多个基表分区键的种类和分区范围一致，即基表使用相同的分区键，可以创建多基表绑定的物化视图。

```
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT start_time, id FROM fact_table union SELECT start_time, id FROM fact_table1;
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT fact_table1.start_time, fact_table.id, fact_table1.id as id1
FROM fact_table inner join fact_table1 on fact_table.start_time=fact_table1.start_time;
```



场景四：基表为多分区键时，支持以基表的第一个分区键创建物化视图。

```
CREATE TABLE IF NOT EXISTS  multi_time_base (
    new_start_time timestamp with time zone NOT NULL,
    id int,
    v1 int
) PARTITION BY RANGE(new_start_time,id) (
    PARTITION p1 VALUES LESS THAN ('2022-01-01',400),
    PARTITION p2 VALUES LESS THAN ('2022-01-05',400),
    PARTITION p3 VALUES LESS THAN ('2022-01-05',500),
    PARTITION p4 VALUES LESS THAN ('2022-02-15',500),
    PARTITION p5 VALUES LESS THAN ('2022-02-16',500),
    PARTITION p6 VALUES LESS THAN ('2022-03-10',500)
);
CREATE MATERIALIZED VIEW mult_time_mv
PARTITION BY (date_trunc('month', new_start_time))
AS SELECT new_start_time,id FROM multi_time_base;
```

 版权所有 © 华为云计算技术有限公司

9.2.2.3 刷新分区物化视图

当创建成功分区物化视图时，可以使用分区刷新减少刷新代价。

自动刷新

由系统内部识别自动刷新，通过以下命令刷新。

```
REFRESH MATERIALIZED VIEW materialized_view_name;
```

该命令会尝试刷新物化视图，如果满足分区刷新的条件则进行分区刷新，否则转为全量刷新。

手动刷新指定分区

通过以下命令指定需要刷新的物化视图分区。

```
REFRESH MATERIALIZED VIEW materialized_view_name PARTITION (partname1,partname2);
```

该命令会尝试刷新物化视图分区partname1, partname2，如果当前分区不需要刷新，则不会刷新，并会提示数据为最新。

强制手动刷新指定分区

通过以下命令强制刷新物化视图指定分区。

```
REFRESH MATERIALIZED VIEW materialized_view_name PARTITION (partname) RESTRICT;
```

查看分区物化视图

查看分区的数据是否更新。

通过pg_matview_partition的active字段可以查看当前分区是否已更新。

```
\d  pg_matview_partition;
Table "pg_catalog.pg_matview_partition"
     Column        |           Type           | Modifiers
-------------------+--------------------------+-----------
matpartid           | oid                      | not null
active              | boolean                  | not null
refresh_start_time  | timestamp with time zone | not null
refresh_finish_time | timestamp with time zone | not null
```

其中：

- matpartid，表示物化视图分区OID。
- active，表示当前分区是否已更新。 t表示已更新，f表示失效。
- refresh_start_time，表示最后一次分区刷新开始时间。
- refresh_finish_time，表示最后一次分区刷新结束时间。

示例

步骤1 创建时间上卷物化视图level_month并查看视图信息。

```
CREATE MATERIALIZED VIEW level_month enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY date_trunc('month', start_time)
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
```


版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17092  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
```

步骤2 向基表插入数据。

```
INSERT INTO fact_table (start_time, id) VALUES ('2023-05-01',1);
```

步骤3 查看分区失效情况。

```
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17092  | f      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
```

步骤4 刷新物化视图。

```
REFRESH MATERIALIZED VIEW level_month;
```

由于只是分区失效，则会触发分区刷新。

步骤5 查看刷新后的状态。

```
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17092  | t      | 2024-11-18 15:16:28.902902+08 | 2024-11-18 15:16:28.940159+08
```

----结束

9.2.3 嵌套物化视图

9.2.3.1 嵌套物化视图概述

DWS物化视图支持嵌套创建、嵌套物化视图刷新以及嵌套物化视图重写。嵌套物化视图的应用，可以有效地减少单个物化视图定义的复杂程度。

- 创建嵌套物化视图，可以基于物化视图构建物化视图。
- 级联刷新，嵌套物化视图刷新时，不仅能刷新当前物化视图，也支持自动级联刷新。
- 定时级联刷新，针对定时刷新的物化视图，支持用户指定和修改刷新方式。
- 嵌套物化视图重写，查询语句能自动被重写成查询物化视图的SQL。

9.2.3.2 创建嵌套物化视图

DWS支持创建嵌套物化视图，嵌套物化视图是指在物化视图的定义中使用其他物化视图的过程。 通过嵌套物化视图，可以将多个物化视图组合在一起，形成更复杂和综合的查询结果。

通过GUC参数**mv_max_cascade_depth**指定嵌套物化视图的最大深度，为了刷新性能上的考虑，当前最大深度为10，默认值为3。

 版权所有 © 华为云计算技术有限公司

使用约束

1. 不支持预聚合物化视图嵌套。
2. 不支持基于物化视图创建预聚合物化视图。
3. 不支持非定时刷新物化视图指定或者修改刷新方式。
4. 嵌套物化视图最大深度为10，暂不支持更深的嵌套层数。
5. 升级场景不支持创建嵌套物化视图。

语法格式

创建嵌套物化视图的语法基本同物化视图的语法一致，物化视图的定义语句中允许基表是物化视图。增加refresh_behavior用于定时刷新物化视图指定刷新方式。

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ refresh_behavior] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```

查看嵌套物化视图

系统表pg_matview_depend中的relid字段，存放物化视图的OID。

```
\d+ pg_matview_depend
Table "pg_catalog.pg_matview_depend"
Column            |   Type    | Modifiers | Storage | Stats target | Description
------------------+-----------+-----------+---------+--------------+-------------
matviewid         | oid       | not null  | plain   |              |
relid             | oid       | not null  | plain   |              |
partids           | oidvector | not null  | plain   |              |
contain_entire_rel | boolean   | not null  | plain   |              |
Indexes:
"pg_matview_depend_mvid_relid_index" UNIQUE, btree (matviewid, relid) TABLESPACE pg_default
"pg_matview_depend_relid_index" btree (relid) TABLESPACE pg_default
Replica Identity: NOTHING
Has OIDs: no
```

示例

1. 创建基表、物化视图和嵌套物化视图。

```
CREATE TABLE t1(a int, b int, c int);
CREATE TABLE t2(a int, b int, c int);
CREATE MATERIALIZED VIEW mv1 ENABLE QUERY REWRITE as select * from t1 WHERE a = 1;
CREATE MATERIALIZED VIEW mv2 ENABLE QUERY REWRITE as select * from t2 WHERE b = 1;
CREATE MATERIALIZED VIEW mv3 ENABLE QUERY REWRITE as select * from mv2 WHERE b = 1;
CREATE MATERIALIZED VIEW mv4 ENABLE QUERY REWRITE as select t1.a as t1a,t1.b as t1b,t1.c as t1c,t2.a as t2a,t2.b as t2b,t2.c as t2c from t1 left join t2 on t1.a = t2.a WHERE t2.b = 1;
CREATE MATERIALIZED VIEW mv5 ENABLE QUERY REWRITE as (select * from mv1) union all (select * from t1 WHERE a = 1);
CREATE MATERIALIZED VIEW mv6 ENABLE QUERY REWRITE as select * from mv4 left join (select * from mv1 union all select * from mv5)mv5 on mv4.t1a = mv5.a;
```

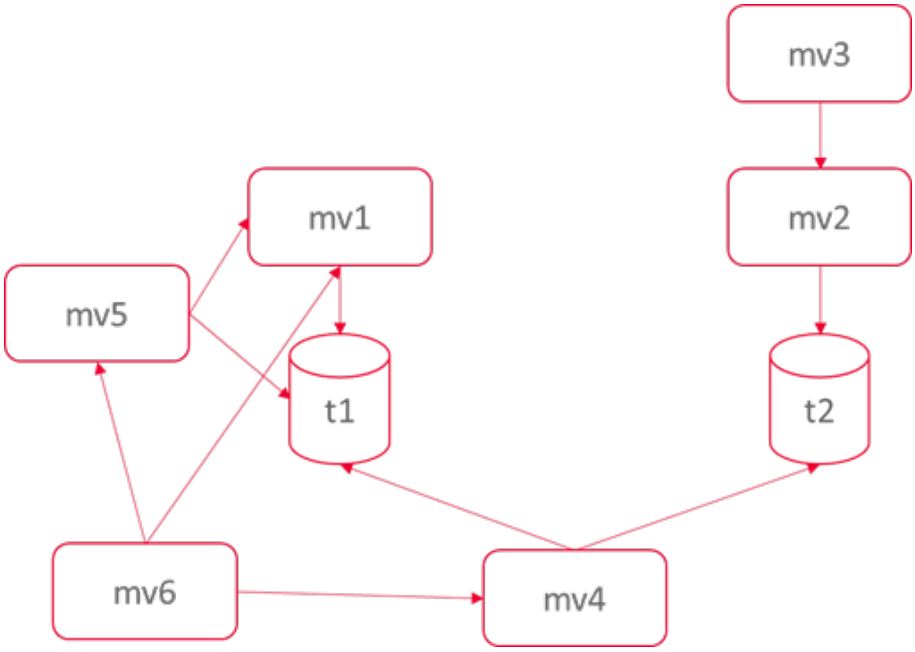
2. 上述SQL中先创建两个表t1和t2，然后以t1和t2为基表，创建了物化视图mv1和mv2；再以t1、t2、mv1、mv2为基表依次创建物化视图mv3、mv4、mv5和mv6。

版权所有 © 华为云计算技术有限公司

和单层物化视图相比，嵌套物化视图顶层的定义更为简洁，使用上更为灵活。

3. 可通过下图了解基表和物化视图的创建关系及依赖关系。

图 9-1 嵌套物化视图依赖关系



4. 当底层数据发生变更时，需要刷新物化视图，保证物化视图中的数据都是有效的。即t1表数据发生变化，mv1、mv4、mv5和mv6中的数据都需要更新。使用级联刷新单个物化视图，自动将相关联的物化视图都进行更新。

```
REFRESH MATERIALIZED VIEW mv6 cascade backward;
```

5. 当嵌套物化视图中的数据都是有效的，执行查询语句时，会进行自动识别和匹配，将查询语句自动重写为查询物化视图，大大减少查询语句中的计算。

9.2.3.3 刷新嵌套物化视图

级联刷新

嵌套物化视图中，为了避免单个物化视图刷新繁琐操作，支持自动级联刷新嵌套物化视图。

当前支持五种刷新方式，默认、强制、级联向上、级联向下和全部级联。即对应refresh_behavior取值：DEFAULT、RESTRICT、CASCADE FORWARD、CASCADE BACKWARD和CASCADE ALL。

- 默认方式：刷新单个物化视图，当物化视图的基表未发生数据变化时，不进行刷新操作，直接跳过。
- 强制方式：刷新单个物化视图。
- 级联向上：先刷新当前物化视图，然后依次刷新依赖当前物化视图的嵌套物化视图。
- 级联向下：找到当前物化视图依赖的所有物化视图，依次从最底层开始刷新，最后刷新当前物化视图。

版权所有 © 华为云计算技术有限公司

- 全部级联：先执行级联向下刷新，再执行级联向上刷新。

说明

级联刷新方式中，采用的是默认刷新方式，即下层物化视图/表数据未发生变化，不进行刷新操作。

示例：

以**创建嵌套物化视图**中创建的嵌套物化视图为例。

对物化视图mv5进行全部级联刷新。

```
REFRESH MATERIALIZED VIEW mv5 cascade all;
```

级联刷新流程：先刷新mv1，再刷新mv5，最后刷新mv6。

相当于先对mv5执行级联向下刷新，然后进行级联向上刷新。

```
REFRESH MATERIALIZED VIEW mv5 cascade backward;
REFRESH MATERIALIZED VIEW mv5 cascade forward;
```

对于级联向下刷新，会先找到mv5依赖的所有底层物化视图，然后从底层向上层，逐层进行物化视图刷新。即执行：

```
REFRESH MATERIALIZED VIEW mv5 cascade backward;
```

级联刷新流程：先刷新mv1，再刷新mv5。

对于级联向上刷新，会先找到所有依赖mv5的物化视图，然后从底层向上层，逐层进行物化视图刷新。

级联刷新流程：先刷新mv5，再刷新mv6。

定时刷新物化视图

对于后台定时刷新的物化视图，需要用户指定刷新方式，否则都采用默认的刷新方式，而不会进行级联刷新。

因此针对类物化视图，支持在创建时指定级联刷新方式，支持修改级联刷新方式。

示例：

1. 创建定时刷新物化视图。

```
CREATE TABLE test_table(a int2, b int, c int8, d int) WITH (ORIENTATION = COLUMN, COLVERSION = 2.0) DISTRIBUTE BY hash(c);
create materialized view mv_test_sync_default REFRESH START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s')  ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
create materialized view mv_test_sync_restrict REFRESH restrict START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
create materialized view mv_test_sync_forward REFRESH cascade forward START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
create materialized view mv_test_sync_backward REFRESH cascade backward START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
create materialized view mv_test_sync_all REFRESH cascade all START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
```

2. 查看定时刷新任务。

```
SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%' ORDER BY 1;
```


版权所有 © 华为云计算技术有限公司

```
postgres=# SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%' ORDER BY 1;
                                  what                                  |    category_id    | is_broken | interval
------------------------------------------------------------------------+-------------------+-----------+----------
 REFRESH MATERIALIZED VIEW public.mv_test_sync_all CASCADE ALL;         | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_backward CASCADE BACKWARD; | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_default;                 | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_forward CASCADE FORWARD; | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_restrict RESTRICT;       | MatViewKernelTask | f         | 00:00:01
(5 rows)
```

3. 修改级联刷新方式。

```
ALTER MATERIALIZED VIEW mv_test_sync_backward REFRESH START WITH('2020-01-01 15:15:15'::timestamptz) EVERY (interval '10 s') ;
ALTER MATERIALIZED VIEW mv_test_sync_restrict REFRESH cascade all START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '20 s') ;
ALTER MATERIALIZED VIEW mv_test_sync_all REFRESH  default;
ALTER MATERIALIZED VIEW mv_test_sync_forward REFRESH default START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '50 s') ;
```

```
postgres=# SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%' ORDER BY 1;
                                  what                                  |    category_id    | is_broken | interval
------------------------------------------------------------------------+-------------------+-----------+----------
 REFRESH MATERIALIZED VIEW public.mv_test_sync_all;                     | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_backward CASCADE BACKWARD; | MatViewKernelTask | f         | 00:00:10
 REFRESH MATERIALIZED VIEW public.mv_test_sync_default;                 | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_forward;                 | MatViewKernelTask | f         | 00:00:50
 REFRESH MATERIALIZED VIEW public.mv_test_sync_restrict CASCADE ALL;    | MatViewKernelTask | f         | 00:00:20
(5 rows)
```

 版权所有 © 华为云计算技术有限公司

10 DWS 用户自定义函数

10.1 DWS PL/Java 语言函数

DWS支持Java UDF (用户自定义函数)，它是一种在数据处理系统中扩展功能的方式，允许开发者使用Java语言编写自定义函数，然后在SQL查询或其他数据处理场景中调用这些函数。

使用限制

Java UDF可以实现一些java逻辑计算，强烈建议不要在Java UDF中封装业务

- 强烈建议不要在Java函数中使用任何方式连接数据库，包括但不限于JDBC。
- 暂不支持的数据类型：除表10-2内容之外的数据类型，包括自定义类型，复杂数据类型（Java Array类及派生类）。
- 暂不支持UDAF，UDTF。
- DWS PL/Java基于开源PL/Java 1.5.5开发，所使用的JRE版本为1.8.0.432。

Java UDF 使用流程

使用PL/Java函数时，用户可以使用Java IDE编写Java方法，并将包含这些方法的jar包文件部署到DWS数据库中，然后使用数据库管理员账号创建函数。考虑兼容性问题，请使用1.8.0.432版本的JRE进行编译。

步骤1 编译jar包。

Java方法的实现和出包可以借助IDE来实现，以下是一个通过命令行来进行编译和出包的简单的示例，通过这个简单示例可以创建出一个包含单个方法的jar包文件。

首先，编写一个Example.java文件（通常需要使用static关键字），在此文件中实现子字符串大写转换的方法，本例中类名为Example，方法名为upperString，内容如下

```
public class Example
{
    public static String upperString (String text, int beginIndex, int endIndex)
    {
        return text.substring(beginIndex, endIndex).toUpperCase();
    }
}
```

 版权所有 © 华为云计算技术有限公司

然后，创建manifest.txt清单文件，文件内容如下，各参数的解释如表10-1所示：

```
Manifest-Version: 1.0
Main-Class: Example
Specification-Title: "Example"
Specification-Version: "1.0"
Created-By: 1.6.0_35-b10-428-11M3811
Build-Date: 08/14/2018 10:09 AM
```

表 10-1 manifest.txt 清单文件

参数项	描述	示例
Manifest-Version	表示清单文件的版本，固定为1.0，与JAR规范兼容。 无需修改，保持默认即可。	1.0
Main-Class	指定JAR包的入口类（包含main方法的类），即主应用程序类的类名。当通过java -jar执行JAR包时，会调用此类。	Example
Specification-Title	项目或模块的名称标识符，用于描述JAR包的功能，例如本例：子字符串大写转换。	Example
Specification-Version	项目的规范版本号，遵循语义化版本，用户自定义，例如首次发布为1.0，后续迭代为2.0。	1.0
Created-By	定义了生成此清单文件所基于的Java实现的版本和供应商，此属性由jar工具生成，格式为 JDK版本号-构建号。 • 默认由构建工具（如 Maven）自动填充，通常无需手动修改。 • 确保构建环境和运行环境的JDK版本一致。	1.6.0_35-b10-428-11M3811
Build-Date	JAR 的构建时间戳，用于追踪构建历史，便于回滚或调试。	08/14/2018 10:09 AM

最后，编译java文件并打包得到javaudf-example.jar

```
javac Example.java
jar cfm javaudf-example.jar manifest.txt Example.class
```

须知

jar包的命名规则应符合JDK命名要求，如果含有非法字符，在部署或者使用函数时将出错。

步骤2 部署jar包。

Jar包首先需要放置到OBS服务器中，放置方法具体请参见《*对象存储服务控制台指南*》的上传文件章节。接着创建访问密钥AK/SK，获取访问密钥的具体步骤，请参见

版权所有 © 华为云计算技术有限公司

创建访问密钥（AK和SK）章节。登录数据库，使用具有管理员权限用户运行gs_extend_library函数，将文件导入到DWS中：

```
SELECT gs_extend_library('addjar', 'obs://bucket/path/javaudf-example.jar
accesskey=access_key_value_to_be_replaced  secretkey=secret_access_key_value_to_be_replaced
region=region_name libraryname=example');
```

gs_extend_library函数如何使用请参见 **管理jar包和文件**。函数中的AK/SK值，请用户根据实际获取值替换。region_name请用户根据实际所在的区域名称替换。bucket为OBS服务器上的桶名称。

步骤3 使用PL/Java函数。

首先，使用拥有sysadmin权限的数据库用户（例如：dbadmin）登录数据库并创建java_upperstring函数如下：

```
CREATE FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER)
    RETURNS VARCHAR
    AS 'Example.upperString'
LANGUAGE JAVA;
```

说明

- 函数java_upperstring中定义的数据类型为DWS的数据类型。该数据类型需要和**步骤1**中java定义的方法upperString中数据类型一一对应。DWS与Java数据类型的对应关系，请参见**表10-2**。
- AS子句用于指定该函数所调用的Java方法的类名和static方法名，格式为“类名.方法名”。该字段需要和**步骤1**中java定义的类名和方法名一致。
- 使用PL/Java函数时，LANGUAGE字段应指定为JAVA。
- CREATE FUNCTION更多说明，请参见**创建函数**。

然后，执行java_upperstring函数：

```
SELECT java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
----------------------
 T
(1 row)
```

步骤4 授权普通用户使用PL/Java函数。

创建普通用户，名称为udf_user。

```
CREATE USER udf_user PASSWORD 'password';
```

授权普通用户udf_user对java_upperstring函数的使用权限。注意，此处需要把函数所在模式和函数的使用权限同时赋予给用户，用户才可以使用此函数。

```
GRANT ALL PRIVILEGES ON SCHEMA public TO udf_user;
GRANT ALL PRIVILEGES ON FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER) TO udf_user;
```

以普通用户udf_user登录数据库。

```
SET SESSION SESSION AUTHORIZATION udf_user PASSWORD 'password';
```

执行java_upperstring函数：

```
SELECT public.java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
----------------------
```

 版权所有 © 华为云计算技术有限公司

```
 T
(1 row)
```

步骤5 删除函数。

如果不再使用该函数可以进行删除：

```
DROP FUNCTION java_upperstring;
```

步骤6 卸载jar包。

使用gs_extend_library函数卸载jar包：

```
SELECT gs_extend_library('rmjar', 'libraryname=example');
```

----结束

基本数据类型映射关系

表 10-2 PL/Java 默认数据类型映射关系

DWS	Java
BOOLEAN	boolean
"char"	byte
bytea	byte[]
SMALLINT	short
INTEGER	int
BIGINT	long
FLOAT4	float
FLOAT8	double
CHAR	java.lang.String
VARCHAR	java.lang.String
TEXT	java.lang.String
name	java.lang.String
DATE	java.sql.Timestamp
TIME	java.sql.Time (stored value treated as local time)
TIMETZ	java.sql.Time
TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	java.sql.Timestamp

版权所有 © 华为云计算技术有限公司

UDF 示例：SQL 定义与使用

- **管理jar包和文件**

  拥有sysadmin权限的数据库用户可以使用gs_extend_library函数来部署、查看和删除数据库中的jar包，函数语法如下：

  ```
  SELECT gs_extend_library('[action]', '[operation]');
  ```

  说明

  - **action**表明操作动作，值可以为：
    - ls表示查看数据库中jar包，会对各节点的文件进行MD5值一致性检查。
    - addjar表示将OBS服务器中的jar包部署到数据库中。
    - rmjar表示将数据库中的jar包删除。
  - **operation**表明操作字符串，格式为：

    obs://[bucket]/[source_filepath] accesskey=[accesskey] secretkey=[secretkey] region=[region] libraryname=[libraryname]
    - bucket: OBS文件所属桶名，不可缺省。
    - source_filepath: OBS服务器上的文件路径，仅支持jar文件。
    - accesskey: obs服务获得的accesskey，不可缺省。
    - secret_key: obs服务获得的secretkey，不可缺省。
    - region: 自定义函数jar包所存放的OBS桶所属的区域，不可缺省。
    - libraryname: 自定义库名，此自定义命名用于DWS内对jar文件的调用。当action为addjar和rmjar时，该参数不可缺省，当action为ls时，该参数可以缺省。注意，自定义库名不允许含有/|;&$<>\'{}"()[]~*?!等字符。

- **创建函数**

  PL/Java函数通过CREATE FUNCTION语法创建，并且定义为LANGUAGE JAVA，且包含RETURNS和AS子句。

  - CREATE FUNCTION时指定所创建函数的名称，以及参数类型；
  - RETURNS子句用于指定该函数的返回类型；
  - AS子句与用于指定该函数所调用的Java方法的类名和static方法名，如果需要向Java方法传递NULL值作为入参，还需要指定该参数类型所对应的Java封装类名（详见**UDF示例：NULL值处理**）。
  - 更多语法说明，请参见《数据仓库服务SQL语法参考》中“CREATE FUNCTION”章节。

  ```
  CREATE [ OR REPLACE ] FUNCTION function_name
  ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ]} [, ···] ])
  [ RETURNS rettype [ DETERMINISTIC ] ]
  LANGUAGE JAVA
  [
     { IMMUTABLE | STABLE | VOLATILE }
     | [ NOT ] LEAKPROOF
     | WINDOW
     | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT |STRICT }
     | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER | AUTHID CURRENT_USER}
     | { FENCED }
     | COST execution_cost
     | ROWS result_rows
     | SET configuration_parameter { {TO |=} value | FROM CURRENT}
  ] [···]
  {
     AS 'class_name.method_name' ( { argtype } [, ···] )
  }
  ```

- **使用函数**

版权所有 © 华为云计算技术有限公司

执行时，PL/Java会根据jar包名的字母序列，在所有部署的jar包中寻找函数指定的Java类，并调用首次找到的类中函数所指定的Java方法，并返回调用结果。

- **删除函数**

  PL/Java函数通过DROP FUNCTION语法删除函数。

  ```
  DROP FUNCTION [ IF EXISTS ] function_name [ ( [ {[ argmode ] [ argname ] argtype} [, ...] ] )
  [ CASCADE | RESTRICT ] ];
  ```

  需要注意的是，如果所删除的函数为重载函数（详见**重载函数**），则删除时需要指明该函数的参数类型，如为非重载函数，则可直接指定函数名进行删除。

- **函数授权**

  非sysadmin户无法创建PL/Java函数，sysadmin用户可以赋予其他类型用户使用函数的权限。更多语法说明，请参见《数据仓库服务SQL语法参考》中“GRANT”章节。

  ```
  GRANT { EXECUTE | ALL [ PRIVILEGES ] }
      ON { FUNCTION {function_name ( [ {[ argmode ] [ arg_name ] arg_type} [, ...] ] )} [, ...]
          | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
      TO { [ GROUP ] role_name | PUBLIC } [, ...]
      [ WITH GRANT OPTION ];
  ```

UDF 示例：数组类型处理

DWS支持基础数组类型的转换，只需要在创建函数时在数据类型后追加 [] 即可，例如：

```
CREATE FUNCTION java_arrayLength(INTEGER[])
    RETURNS INTEGER
    AS 'Example.getArrayLength'
LANGUAGE JAVA;
```

Java代码类似于：

```
public class Example
{
    public static int getArrayLength(Integer[] intArray)
    {
        return intArray.length;
    }
}
```

那么下面的调用的语句后：

```
SELECT java_arrayLength(ARRAY[1, 2, 3]);
```

得到预期结果应该如下所示：

```
java_arrayLength
---------------------
3
(1 row)
```

UDF 示例：NULL 值处理

对于默认与Java的简单类型进行映射转换的那些DWS数据类型，是无法处理NULL值的，如果希望在Java方法里能够获得并处理从DWS中传入的NULL值，可以使用Java的封装类，并通过以下方式在AS子句中指定该Java封装类：

```
CREATE FUNCTION java_countnulls(INTEGER[])
    RETURNS INTEGER
    AS 'Example.countNulls(java.lang.Integer[])'
LANGUAGE JAVA;
```

Java代码类似于：

 版权所有 © 华为云计算技术有限公司

```
public class Example
{
    public static int countNulls(Integer[] intArray)
    {
        int nullCount = 0;
        for (int idx = 0; idx < intArray.length; ++idx)
        {
            if (intArray[idx] == null)
            nullCount++;
        }
        return nullCount;
    }
}
```

那么下面的调用的语句后：

```
SELECT java_countNulls(ARRAY[null, 1, null, 2, null]);
```

得到的预期结果应该如下所示：

```
java_countNulls
---------------------
3
(1 row)
```

UDF 示例：重载函数

PL/Java支持重载函数，因此可以创建同名函数，或者调用Java代码中的重载方法。步骤如下：

步骤1 创建重载函数

例如，在Java中可以实现两个方法名相同，输入参数类型不同的方法dummy(int) 和dummy(String)

```
public class Example
{
    public static int dummy(int value)
    {
        return value*2;
    }
    public static String dummy(String value)
    {
        return value;
    }
}
```

并在DWS中创建两个同名函数分别指定为上述两个方法：

```
CREATE FUNCTION java_dummy(INTEGER)
    RETURNS INTEGER
    AS 'Example.dummy'
LANGUAGE JAVA;

CREATE FUNCTION java_dummy(VARCHAR)
    RETURNS VARCHAR
    AS 'Example.dummy'
LANGUAGE JAVA;
```

步骤2 调用重载函数

在调用重载函数时，DWS会根据输入的参数类型去调用匹配该类型的Java方法。因此上述两个函数的调用结果如下所示：

```
SELECT java_dummy(5);
 java_dummy
-----------------
```


版权所有 © 华为云计算技术有限公司

```
          10
(1 row)

SELECT java_dummy('5');
 java_dummy
------------
5
(1 row)
```

需要注意的是，由于DWS对数据类型存在隐式转换的情况，因此建议在调用重载函数时，指定输入参数的类型，例如：

```
SELECT java_dummy(5::varchar);
 java_dummy
------------
5
(1 row)
```

此时会优先匹配所指定的参数类型，如果不存在指定参数类型的Java方法，则会对参数进行隐式转换匹配转换后的参数类型对应的Java方法。

```
SELECT java_dummy(5::INTEGER);
 java_dummy
------------
10
(1 row)

DROP FUNCTION java_dummy(INTEGER);

SELECT java_dummy(5::INTEGER);
 java_dummy
------------
5
(1 row)
```

须知

隐式转换的数据类型包括：

- 可以默认转换为INTEGER类型的包括：SMALLINT
- 可以默认转换为BIGINT类型的包括：SMALLINT、INTEGER
- 可以默认转换为BOOL类型的包括：TINYINT、SMALLINT、INTEGER、BIGINT
- 可以默认转换为TEXT类型的包括：CHAR、NAME、BIGINT、INTEGER、SMALLINT、 TINYINT、RAW、FLOAT4、FLOAT8、BPCHAR、VARCHAR、NVARCHAR2、DATE、TIMESTAMP、TIMESTAMPTZ、NUMERIC、SMALLDATETIME
- 可以默认转换为VARCHAR类型的包括：TEXT、CHAR、BIGINT、INTEGER、SMALLINT、TINYINT、RAW、FLOAT4、FLOAT8、BPCHAR、DATE、NVARCHAR2、TIMESTAMP、NUMERIC、SMALLDATETIME

步骤3 删除重载函数

对于重载函数，删除时需要指定函数的参数类型，否则无法删除。

```
DROP FUNCTION java_dummy(INTEGER);
```

----结束

 版权所有 © 华为云计算技术有限公司

UDF 相关 GUC 参数

- **udf_memory_limit**

  系统级别的GUC参数，用于限制每个CN、DN执行UDF可以使用的物理内存量，默认为0.05 * max_process_memory。可通过修改postgresql.conf文件进行配置，配置后需要重启数据库服务后才可生效。

  须知

  - udf_memory_limit是max_process_memory的一部分。每个CN、DN启动时，会预留（udf_memory_limit - 200MB）内存供UDF Worker进程使用。CN、DN和UDF Worker是不同的进程，但CN、DN自动少用一部分内存，把这部分内存节省下来供UDF Worker进程使用。

    例如：在某DN上把max_process_memory设置为10GB，udf_memory_limit设置为4GB，则此DN最多使用10GB - (4GB - 200MB)=6.2GB内存。即使用户没有执行任何UDF，则此DN也最多只能使用6.2GB内存。默认情况下，udf_memory_limit为0.05 * max_process_memory。查询pv_total_memory_detail视图时可以发现，process_used_memory永远不会超过max_process_memory - (udf_memory_limit - 200MB)。
  - UDF进程断连时会有报错提示。当出现“memory in UDF Work Process is limited by cgroup: [usage: xxx, max_usage_history: xxx, limit: xxx]”时，可以根据提示判断当前内存使用情况。报错中的usage表示UDF进程被KILL后剩余UDF进程的总物理内存使用量；max_usage_history表示UDF实例启动后历史上使用过的最大内存量，limit表示UDF进程使用内存的最大限制量。如果max_usage_history值接近limit值说明当前集群存在内存超过限制的风险，需要优化业务或按实际情况调整udf_memory_limit参数限制。
  - 一个CN执行最简单的Java UDF函数，使用的物理内存量大约为50MB，用户可以根据自己Java函数的内存使用量和并发度设置此参数。新增此参数后，不再建议用户设置UDFWorkerMemHardLimit和FencedUDFMemoryLimit。
  - 当UDF进程并发度过大，内存超出udf_memory_limit设置值时会导致进程退出等非预期情况，该场景下执行结果可能不可靠，强烈建议根据实际情况进行参数设置，保留足够内存余量。如果系统记录有/var/log/messages，可查看该日志文件是否存在因超过cgroup内存限制而造成内存不足。内存严重不足时，甚至可能导致UDF master进程退出，可以查看UDF日志进行分析，默认的UDF日志路径在$GAUSSLOG/cm/cm_agent/pg_log下。例如，出现以下日志时就说明内存资源严重不足，导致了UDF master进程退出，需要检查udf_memory_limit参数设置。

    0 [BACKEND] FATAL: poll() failed: Bad address, please check the parameter:udf_memory_limit to make sure there is enough memory.

- **FencedUDFMemoryLimit**

  会话级别的GUC参数，用户限制会话发起的单个Fenced UDF Worker进程的最大虚拟内存使用量，设置方法如下：

  ```
  SET FencedUDFMemoryLimit='512MB';
  ```

  该参数的取值范围为 (150MB, 1G]，当设置大于1G时会立即报错，当设置小于等于150MB时，则会在调用函数时报错。

版权所有 © 华为云计算技术有限公司

须知

- FencedUDFMemoryLimit设置为0，表示不控制Fenced UDF Worker的虚拟内存使用量。
- 建议通过设置udf_memory_limit控制Fenced UDF Worker使用的物理内存量。不建议用户使用FencedUDFMemoryLimit，尤其在使用Java UDF时不建议用户设置此参数。但是如果用户非常清楚设置该参数带来的影响，可以参考下列信息进行设置：
  - C UDF worker启动之后，占用的虚拟内存约为200MB，占用的物理内存约为16MB。
  - Java UDF worker启动之后，占用的虚拟内存约为2.5GB，占用的物理内存约为50MB。

异常处理

如果在JVM中发生异常，PL/Java的异常处理机制会将异常时JVM的堆栈信息输出到客户端。

日志

PL/Java使用标准的Java Logger。 因此，用户可以通过如下方式记录日志：

```
Logger.getAnonymousLogger().config( "Time is " + new
Date(System.currentTimeMillis()));
```

初始化的Java Logger类会默认设置为CONFIG级别，对应为DWS的LOG级别。Java Logger类输出的日志消息都会重定向到DWS后端，并写入到服务器日志或显示在用户界面上。MPPDB服务器日志将记录LOG、WARNING、ERROR级别的信息，而SQL用户界面将显示WARNING和ERROR级别的日志消息。Java Logger级别与DWS的日志级别对应关系见下表。

表 10-3 PL/Java 日志级别

java.util.logging.Level	DWS 日志级别
SERVER	ERROR
WARNING	WARNING
CONFIG	LOG
INFO	INFO
FINE	DEBUG1
FINER	DEBUG2
FINEST	DEBUG3

用户可以通过以下方式更改Java Logger的记录级别。例如通过下面的Java代码修改Java Logger级别为SEVERE，此时再记录WARNING级别的日志时，日志消息（msg）就不会再写入到DWS日志中。

版权所有 © 华为云计算技术有限公司

```
Logger log = Logger.getAnonymousLogger();
Log.setLevel(Level.SEVERE);
log.log(Level.WARNING, msg);
```

安全问题

在DWS中，PL/Java是一种untrusted语言，PL/Java函数只能由数据库sysadmin用户进行创建，通过GRANT方式赋予其他用户使用权限（详见**函数授权**）。

同时PL/Java控制用户对文件系统的访问权限，不允许用户在Java方法中对大部分的系统文件进行读操作，不允许所有的写、删除和执行操作。

10.2 DWS PL/pgSQL 语言函数

PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。

用PL/pgSQL创建的函数可以被用在任何可以使用内建函数的地方。例如，可以创建复杂条件的计算函数并且后面用它们来定义操作符或把它们用于索引表达式。

SQL被大多数数据库用作查询语言。它是可移植的并且容易学习。但是每一个SQL语句必须由数据库服务器单独执行。

这意味着客户端应用必须发送每一个查询到数据库服务器、等待它被处理、接收并处理结果、做一些计算，然后发送更多查询给服务器。如果客户端和数据库服务器不在同一台机器上，所有这些会引起进程间通信并且将带来网络负担。

通过PL/pgSQL，可以将一整块计算和一系列查询分组在数据库服务器内部，这样就有了一种过程语言的能力并且使SQL更易用，同时能节省的客户端/服务器通信开销。

- 客户端和服务器之间的额外往返通信被消除。
- 客户端不需要的中间结果不必被整理或者在服务器和客户端之间传送。
- 多轮的查询解析可以被避免。

PL/pgSQL可以使用SQL中所有的数据类型、操作符和函数。

应用PL/pgSQL创建函数的语法为CREATE FUNCTION。正如前面所说，PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。其应用方法与**DWS存储过程**相似，只是存储过程无返回值，函数有返回值。

版权所有 © 华为云计算技术有限公司

11 DWS 存储过程

11.1 DWS 存储过程简介

什么是 DWS 存储过程

商业规则和业务逻辑可以通过程序存储在DWS中，这个程序就是存储过程。

存储过程是SQL，PL/SQL，Java语句的组合。存储过程使执行商业规则的代码可以从应用程序中移动到数据库。从而，代码存储一次能够被多个程序使用。

存储过程的创建及调用办法请参考《数据仓库服务SQL语法参考》中CREATE PROCEDURE。

DWS PL/pgSQL语言函数节所提到的PL/pgSQL语言创建的函数与存储过程的应用方法相同。下面各节中，除非特别声明，否则内容通用于存储过程和PL/pgSQL语言函数。

DWS 存储过程数据类型

数据类型是一组值的集合以及定义在这个值集上的一组操作。DWS数据库是由表的集合组成的，而各表中的列定义了该表，每一列都属于一种数据类型，DWS根据数据类型有相应函数对其内容进行操作，例如DWS可对数值型数据进行加、减、乘、除操作。

11.2 DWS 存储过程数据类型转换

数据库中允许有些数据类型进行隐式类型转换（赋值、函数调用的参数等），有些数据类型间不允许进行隐式数据类型转换，可尝试使用DWS提供的类型转换函数，例如CAST进行数据类型强转。

DWS数据库常见的隐式类型转换，请参见**表11-1**。

须知

DWS支持的DATE的效限范围是：公元前4713年到公元294276年。

版权所有 © 华为云计算技术有限公司

表 11-1 隐式类型转换表

原始数据类型	目标数据类型	备注
CHAR	VARCHAR2	-
CHAR	NUMBER	原数据必须由数字组成。
CHAR	DATE	原数据不能超出合法日期范围。
CHAR	RAW	-
CHAR	CLOB	-
VARCHAR2	CHAR	-
VARCHAR2	NUMBER	原数据必须由数字组成。
VARCHAR2	DATE	原数据不能超出合法日期范围。
VARCHAR2	CLOB	-
NUMBER	CHAR	-
NUMBER	VARCHAR2	-
DATE	CHAR	-
DATE	VARCHAR2	-
RAW	CHAR	-
RAW	VARCHAR2	-
CLOB	CHAR	-
CLOB	VARCHAR2	-
CLOB	NUMBER	原数据必须由数字组成。
INT4	CHAR	-

11.3 DWS 存储过程数组和 record

11.3.1 数组

数组类型的使用

在使用数组之前，需要自定义一个数组类型。

在存储过程中紧跟AS关键字后面定义数组类型。定义方法为：

```
TYPE array_type IS VARRAY(size) OF data_type [NOT NULL];
```

其中：

版权所有 © 华为云计算技术有限公司

- array_type：要定义的数组类型名。
- VARRAY：表示要定义的数组类型。
- size：取值为正整数，表示可以容纳的成员的最大数量。
- data_type：要创建的数组中成员的类型。
- NOT NULL: 可选约束，可以约束该数组中的元素均不为NULL。

说明

- 在DWS中，数组会自动增长，访问越界会返回一个NULL，不会报错。越界写入数组会提示：Subscript outside of limit.
- 在存储过程中定义的数组类型，其作用域仅在该存储过程中。
- 建议选择上述定义方法的一种来自定义数组类型，当同时使用两种方法定义同名的数组类型时，DWS会优先选择存储过程中定义的数组类型来声明数组变量。

DWS 8.1.0之前版本， 由于数组可以自动增长，系统不会校验数组越界以及数组元素的长度限制。当前版本为了兼容Oracle的用法增加了相关约束。如果已经存在越界写入等场景，可通过在**behavior_compat_options**参数中配置varray_verification，来兼容之前不校验的行为。

示例：

```
--演示在存储过程中对数组声明操作。
CREATE OR REPLACE PROCEDURE array_proc
AS
	TYPE ARRAY_INTEGER IS VARRAY(1024) OF INTEGER;--定义数组类型
	TYPE ARRAY_INTEGER_NOT_NULL IS VARRAY(1024) OF INTEGER NOT NULL;--定义非空数组类型
	ARRINT ARRAY_INTEGER := ARRAY_INTEGER();  --声明数组类型的变量
BEGIN
	ARRINT.extend(10);
	FOR I IN 1..10 LOOP
		ARRINT(I) := I;
	END LOOP;
	DBMS_OUTPUT.PUT_LINE(ARRINT.COUNT);
	DBMS_OUTPUT.PUT_LINE(ARRINT(1));
	DBMS_OUTPUT.PUT_LINE(ARRINT(10));
	DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.FIRST));
	DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.last));
END;
/

--调用该存储过程。
CALL array_proc();
10
1
10
1
10
--删除存储过程。
DROP PROCEDURE array_proc;
```

ROWTYPE 类型数组的声明及使用

除了上述示例中普通数组以及not null数组的声明及使用，数组中同时支持rowtype类型数组的声明及使用。

示例：

```
--演示在存储过程中对数组COUNT函数的用法。
CREATE TABLE tbl (a int, b int);
INSERT INTO tbl VALUES(1, 2),(2, 3),(3, 4);
CREATE OR REPLACE PROCEDURE array_proc
AS
```

 版权所有 © 华为云计算技术有限公司

```
    CURSOR all_tbl IS SELECT * FROM tbl ORDER BY a;
    TYPE tbl_array_type IS varray(50) OF tbl%rowtype; --定义rowtype类型的数组，tbl是任意表。
    tbl_array tbl_array_type;
    tbl_item tbl%rowtype;
    inx1 int;
BEGIN
    tbl_array := tbl_array_type();
    inx1 := 0;
    FOR tbl_item IN all_tbl LOOP
        inx1 := inx1 + 1;
        tbl_array(inx1) := tbl_item;
    END LOOP;
    WHILE inx1 IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('tbl_array(inx1).a=' || tbl_array(inx1).a || ' tbl_array(inx1).b=' ||
tbl_array(inx1).b);
        inx1 := tbl_array.PRIOR(inx1);
    END LOOP;
END;
/
```

执行结果：

```
call array_proc();
tbl_array(inx1).a=3 tbl_array(inx1).b=4
tbl_array(inx1).a=2 tbl_array(inx1).b=3
tbl_array(inx1).a=1 tbl_array(inx1).b=2
```

数组相关函数使用

在DWS中，提供了类似Oracle相关的数组函数的支持，可以通过以下函数获取数组的一些属性或者对数组内容进行操作。

COUNT

COUNT函数可以返回当前数组元素的数量，仅统计初始化过的元素或者经过EXTEND函数扩展后的元素。

用法如下：

varray.COUNT或*varray*.COUNT()

示例：

```
--演示在存储过程中对数组COUNT函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    v_varray.extend;
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count=3
v_varray.count=4
```

FIRST 和 LAST

FIRST函数可以返回第一个元素的下标。LAST函数可以返回最后一个元素的下标。

 版权所有 © 华为云计算技术有限公司

用法如下：

varray.FIRST或*varray*.FIRST()

varray.LAST或*varray*.LAST()

示例：

```
--演示在存储过程中对数组FIRST和LAST函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    DBMS_OUTPUT.PUT_LINE('v_varray.first=' || v_varray.first);
    DBMS_OUTPUT.PUT_LINE('v_varray.last=' || v_varray.last);
END;
/
```

执行结果：

```
call test_varray();
v_varray.first=1
v_varray.last=3
```

EXTEND

说明

EXTEND函数主要是为了兼容Oracle的两种用法。在DWS中，数组会自动增长，EXTEND函数不是必须的。如果是新写的存储过程，完全没有必要使用EXTEND函数。

EXTEND函数可以对数组进行扩展，EXTEND有两种调用方式。

- 方式一：

  EXTEND包含一个整型入参，表示数组向后扩展size大小的长度，EXTEND后COUNT和LAST函数的值也会有相应的变化。

  用法如下：

  varray.EXTEND(size)

  其中*varray*.EXTEND这种无参的调用默认会向后扩展1位等价于*varray*.EXTEND(1)

- 方式二：

  EXTEND包含两个整型入参，第一个参数代表向后扩展size大小的长度，第二个参数表示扩展后的数组元素值和之下标为index的元素相同。

  用法如下：

  varray.EXTEND(size, index)

示例：

```
--演示在存储过程中对数组EXTEND函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    v_varray.extend(3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    v_varray.extend(2,3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    DBMS_OUTPUT.PUT_LINE('v_varray(7)=' || v_varray(7));
```

 版权所有 © 华为云计算技术有限公司

```
    DBMS_OUTPUT.PUT_LINE('v_varray(8)=' || v_varray(7));
END;
/
```

执行结果：

```
call test_varray();
v_varray.count=6
v_varray.count=8
v_varray(7)=3
v_varray(8)=3
```

NEXT 和 PRIOR

NEXT函数和PRIOR函数主要用于数组的循环遍历中，NEXT函数会根据入参index值，返回下一个数组元素的下标，若已经到达数组下标最大值则返回NULL。PRIOR函数会根据入参index值，返回上一个数组元素的下标，若已经到达数组下标最小值则返回NULL。

用法如下：

varray.NEXT(index)

varray.PRIOR(index)

示例：

```
--演示在存储过程中对数组NEXT和PRIOR函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
    i int;
BEGIN
    v_varray := varray_type(1, 2, 3);

    i := v_varray.COUNT;
    WHILE i IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('test prior v_varray('||i||')=' || v_varray(i));
        i := v_varray.PRIOR(i);
    END LOOP;

    i := 1;
    WHILE i IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('test next v_varray('||i||')=' || v_varray(i));
        i := v_varray.NEXT(i);
    END LOOP;
END;
/
```

执行结果：

```
call test_varray();
test prior v_varray(3)=3
test prior v_varray(2)=2
test prior v_varray(1)=1
test next v_varray(1)=1
test next v_varray(2)=2
test next v_varray(3)=3
```

EXISTS

EXISTS函数可以判断数组下标是否存在。

用法如下：

 版权所有 © 华为云计算技术有限公司

varray.EXISTS(index)

示例：

```
--演示在存储过程中对数组EXISTS函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    IF v_varray.EXISTS(1) THEN
        DBMS_OUTPUT.PUT_LINE('v_varray.EXISTS(1)');
    END IF;
    IF NOT v_varray.EXISTS(10) THEN
        DBMS_OUTPUT.PUT_LINE('NOT v_varray.EXISTS(10)');
    END IF;
END;
/
```

执行结果：

```
call test_varray();
v_varray.EXISTS(1)
NOT v_varray.EXISTS(10)
```

TRIM

TRIM函数可以从数组尾部删除指定数量的元素。

用法如下：

varray.TRIM(size)

其中*varray*.TRIM这种无参的调用会默认入参为1，等价于*varray*.TRIM(1)

示例：

```
--演示在存储过程中对数组TRIM函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    v_varray.trim(3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count' || v_varray.count);
    v_varray.trim;
    DBMS_OUTPUT.PUT_LINE('v_varray.count:' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count:2
v_varray.count:1
```

DELETE

DELETE函数可以从数组删除数组中的所有元素。

用法如下：

varray.DELETE或*varray*.DELETE()

示例：

 版权所有 © 华为云计算技术有限公司

```
--演示在存储过程中对数组DELETE函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    v_varray.delete;
    DBMS_OUTPUT.PUT_LINE('v_varray.count:' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count:0
```

LIMIT

LIMIT函数可以返回数组的最大长度限制。

用法如下：

varray.LIMIT或*varray*.LIMIT()

示例：

```
--演示在存储过程中对数组LIMIT函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    DBMS_OUTPUT.PUT_LINE('v_varray.limit:' || v_varray.limit);
END;
/
```

执行结果：

```
call test_varray();
v_varray.limit:20
```

11.3.2 record

record 类型的变量

创建一个record变量的方式：

定义一个record类型 ， 然后使用该类型来声明一个变量。

语法

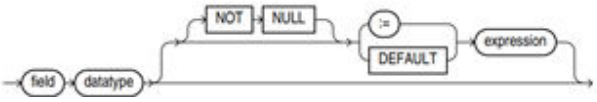
record类型的语法参见**图11-1**。

 版权所有 © 华为云计算技术有限公司

图 11-1 record 类型的语法

record_type_definition ::=

field_definition ::=



对以上语法格式的解释如下：

- record_type：声明的类型名称。
- field：record类型中的成员名称。
- datatype：record类型中成员的类型。
- expression：设置默认值的表达式。

说明

在DWS中：

- record类型的变量的赋值支持，
  - 在函数或存储过程的声明阶段，声明一个record类型，并且可以在该类型中定义成员变量。
  - 一个record变量到另一个record变量的赋值。
  - SELECT INTO和FETCH向一个record类型的变量中赋值。
  - 将一个NULL值赋值给一个record变量。
- 不支持INSERT和UPDATE语句使用record变量进行插入数据和更新数据。
- 如果成员有复合类型，在声明阶段不支持指定默认值，该行为同声明阶段的变量一样。

示例

```
下面存储过程中用到的表定义如下：
CREATE TABLE emp_rec
(
    empno           numeric(4,0),
    ename           character varying(10),
    job             character varying(9),
    mgr             numeric(4,0),
    hiredate        timestamp(0) without time zone,
    sal             numeric(7,2),
    comm            numeric(7,2),
    deptno          numeric(2,0)
)
with (orientation = column,compression=middle)
distribute by hash (sal);
\d emp_rec
              Table "public.emp_rec"
  Column  |            Type             | Modifiers
----------+-----------------------------+-----------
 empno    | numeric(4,0)                | not null
 ename    | character varying(10)       |
 job      | character varying(9)        |
 mgr      | numeric(4,0)                |
```

 版权所有 © 华为云计算技术有限公司

```
 hiredate | timestamp(0) without time zone |
 sal      | numeric(7,2)                   |
 comm     | numeric(7,2)                   |
 deptno   | numeric(2,0)                   |

--演示在存储过程中对数组进行操作。
CREATE OR REPLACE FUNCTION regress_record(p_w VARCHAR2)
RETURNS
VARCHAR2  AS $$
DECLARE

  --声明一个record类型.
  type rec_type is record (name  varchar2(100), epno int);
  employer rec_type;

  --使用%type声明record类型
  type rec_type1 is record (name  emp_rec.ename%type, epno int not null :=10);
  employer1 rec_type1;

  --声明带有默认值的record类型
  type rec_type2 is record (
        name varchar2 not null := 'SCOTT',
        epno int not null :=10);
   employer2 rec_type2;
   CURSOR C1 IS  select ename,empno from emp_rec order by 1 limit 1;

BEGIN
     --对一个record类型的变量的成员赋值。
    employer.name := 'WARD';
    employer.epno = 18;
    raise info 'employer name: % , epno:%', employer.name, employer.epno;

    --将一个record类型的变量赋值给另一个变量。
    employer1 := employer;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --将一个record类型变量赋值为NULL。
    employer1 := NULL;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --获取record变量的默认值。
    raise info 'employer2 name: % ,epno: %', employer2.name, employer2.epno;

     --在for循环中使用record变量
     for employer in select ename,empno from emp_rec order by 1  limit 1
         loop
             raise info 'employer name: % , epno: %', employer.name, employer.epno;
        end loop;

     --在select into 中使用record变量。
     select ename,empno  into employer2 from emp_rec order by 1 limit 1;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;

     --在cursor中使用record变量。
     OPEN C1;
     FETCH C1 INTO employer2;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;
     CLOSE C1;
     RETURN employer.name;
END;
$$
LANGUAGE plpgsql;

--调用该存储过程。
CALL regress_record('abc');
INFO:  employer name: WARD , epno:18
INFO:  employer1 name: WARD , epno: 18
INFO:  employer1 name: <NULL> , epno: <NULL>
INFO:  employer2 name: SCOTT ,epno: 10
```

 版权所有 © 华为云计算技术有限公司

```
--删除存储过程。
DROP PROCEDURE regress_record;
```

11.4 DWS 存储过程声明语法

基本结构

PL/SQL块中可以包含子块，子块可以位于PL/SQL中任何部分。PL/SQL块的结构如下：

- 声明部分：声明PL/SQL用到的变量，类型及游标，以及局部的存储过程和函数。
  ```
  DECLARE
  ```

  说明

  不涉及变量声明时声明部分可以没有。

  - 对匿名块来说，没有变量声明部分时，可以省去DECLARE关键字。
  - 对存储过程来说，没有DECLARE， AS相当于DECLARE。即便没有变量声明的部分，关键字AS也必须保留。
- 执行部分：过程及SQL语句，程序的主要部分。必选。
  ```
  BEGIN
  ```
- 执行异常部分：错误处理。可选。
  ```
  EXCEPTION
  ```
- 结束
  ```
  END;
  /
  ```

须知

禁止在PL/SQL块中使用连续的Tab，连续的Tab可能会造成在使用gsql工具带“-r”参数执行PL/SQL块时出现异常。

PL/SQL 块分类

PL/SQL块可以分为以下几类：

- 匿名块：动态构造，只能执行一次。语法请参考**匿名块**。
- 子程序：存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

匿名块

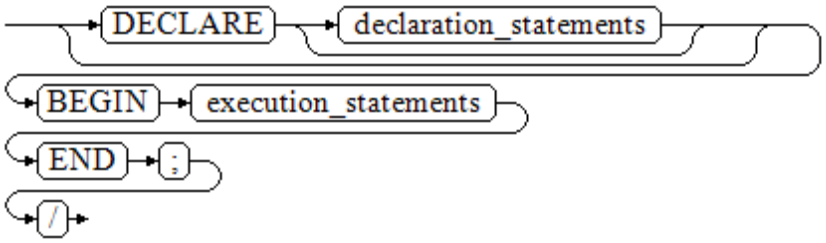
匿名块（Anonymous Block）一般用于不频繁执行的脚本或不重复进行的活动。它们在一个会话中执行，并不被存储。

语法

匿名块的语法参见**图11-2**。

版权所有 © 华为云计算技术有限公司

图 11-2 anonymous_block::=



对以上语法图的解释如下：

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。输入“/”按回车执行它。

  须知

  最后的结束符“/”必须独占一行，不能直接跟在END后面。

- 声明部分包括变量定义、类型、游标定义等。
- 最简单的匿名块不执行任何命令。但一定要在任意实施块里至少有一个语句，甚至是一个NULL语句。

示例

下面列举了基本的匿名块程序：

```
--空语句块
BEGIN
    NULL;
END;
/

--将信息打印到控制台：
BEGIN
    dbms_output.put_line('hello world!');
END;
/

--将变量内容打印到控制台：
DECLARE
    my_var VARCHAR2(30);
BEGIN
    my_var :='world';
    dbms_output.put_line('hello'||my_var);
END;
/
```

子程序

存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

 版权所有 © 华为云计算技术有限公司

11.5 DWS 存储过程基本语句

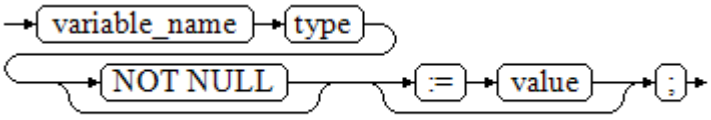
定义变量

介绍PL/SQL中变量的声明，以及该变量在代码中的作用域。

变量声明

变量声明语法请参见图11-3。

图 11-3 declare_variable::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- type，为变量类型。
- value，是该变量的初始值（如果不给定初始值，则初始为NULL）。value也可以是表达式。

示例

```
DECLARE
    emp_id  INTEGER := 7788;  --定义变量并赋值
BEGIN
    emp_id := 5*7784;  --变量赋值
END;
/
```

变量类型除了支持基本类型，还可使用%TYPE和%ROWTYPE去声明一些与其他表字段或表结构本身相关的变量。

%TYPE属性

%TYPE主要用于声明某个与其他变量类型（例如，表中某列的类型）相同的变量。假如想定义一个my_name变量，它的变量类型与employee的firstname类型相同，可使用如下定义：

```
my_name employee.firstname%TYPE
```

这样定义可以带来两个好处，首先，不用预先知道employee 表的firstname类型具体是什么。其次，即使之后firstname类型有了变化，也不需要再次修改my_name的类型。

%ROWTYPE属性

%ROWTYPE属性主要用于对一组数据的类型声明，用于存储表中的一行数据，或从游标匹配的结果。假如需要一组数据，该组数据的字段名称与字段类型都与employee表相同。可以通过如下定义：

```
my_employee employee%ROWTYPE
```


版权所有 © 华为云计算技术有限公司

须知

多个CN的环境下，存储过程中无法声明临时表的%ROWTYPE及%TYPE属性。因为临时表仅在当前session有效，在编译阶段其他CN无法看到当前CN的临时表。故多个CN的环境下，会提示该临时表不存在。

变量作用域

变量的作用域表示变量在代码块中的可访问性和可用性。只有在它的作用域内，变量才有效。

- 变量必须在declare部分声明，即必须建立BEGIN-END块。块结构也强制变量必须先声明后使用，即变量在过程内有不同作用域、不同的生存期。
- 同一变量可以在不同的作用域内定义多次，内层的定义会覆盖外层的定义。
- 在外部块定义的变量，可以在嵌套块中使用。但外部块不能访问嵌套块中的变量。

示例

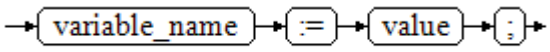
```
DECLARE
    emp_id  INTEGER :=7788;  --定义变量并赋值
    outer_var  INTEGER :=6688;  --定义变量并赋值
BEGIN
    DECLARE
        emp_id INTEGER :=7799;  --定义变量并赋值
        inner_var  INTEGER :=6688;  --定义变量并赋值
    BEGIN
        dbms_output.put_line('inner emp_id ='||emp_id);  --显示值为7799
        dbms_output.put_line('outer_var ='||outer_var);  --引用外部块的变量
    END;
    dbms_output.put_line('outer emp_id ='||emp_id);  --显示值为7788
END;
/
```

赋值语句

语法

给变量赋值的语法请参见图11-4。

图 11-4 assignment_value::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- value，可以是值或表达式。值value的类型需要和变量variable_name的类型兼容才能正确赋值。

示例

```
DECLARE
    emp_id  INTEGER := 7788;--赋值
BEGIN
    emp_id := 5;--赋值
    emp_id := 5*7784;
```


版权所有 © 华为云计算技术有限公司

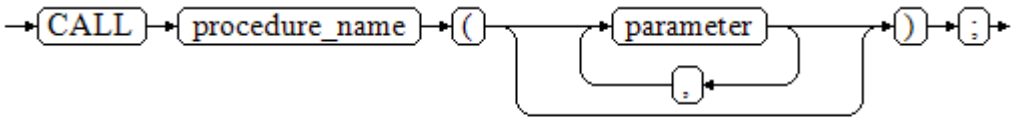
```
END;
/
```

调用语句

语法

调用一个语句的语法请参见图11-5。

图 11-5 call_clause::=



对以上语法格式的解释如下：

- procedure_name，为存储过程名。
- parameter，为存储过程的参数，可以没有或者有多个参数。

示例

```
--创建存储过程proc_staffs
CREATE OR REPLACE PROCEDURE proc_staffs
(
section     NUMBER(6),
salary_sum out NUMBER(8,2),
staffs_count out INTEGER
)
IS
BEGIN
SELECT sum(salary), count(*) INTO salary_sum, staffs_count FROM staffs where section_id = section;
END;
/

--创建存储过程proc_return.
CREATE OR REPLACE PROCEDURE proc_return
AS
v_num NUMBER(8,2);
v_sum INTEGER;
BEGIN
proc_staffs(30, v_sum, v_num);  --调用语句
dbms_output.put_line(v_sum||'#'||v_num);
RETURN;  --返回语句
END;
/

--调用存储过程proc_return.
CALL proc_return();

--清除存储过程
DROP PROCEDURE proc_staffs;
DROP PROCEDURE proc_return;

--创建函数func_return.
CREATE OR REPLACE FUNCTION func_return returns void
language plpgsql
AS $$
DECLARE
v_num INTEGER := 1;
BEGIN
dbms_output.put_line(v_num);
```

 版权所有 © 华为云计算技术有限公司

```
RETURN;  --返回语句
END $$;


-- 调用函数func_return
CALL func_return();
1

-- 清除函数
DROP FUNCTION func_return;
```

11.6 DWS 存储过程动态语句

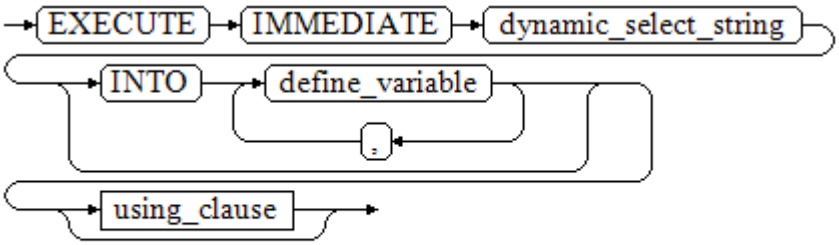
11.6.1 执行动态查询语句

介绍执行动态查询语句。DWS提供两种方式：使用EXECUTE IMMEDIATE、OPEN FOR实现动态查询。前者通过动态执行SELECT语句，后者结合了游标的使用。当需要将查询的结果保存在一个数据集用于提取时，可使用OPEN FOR实现动态查询。

EXECUTE IMMEDIATE

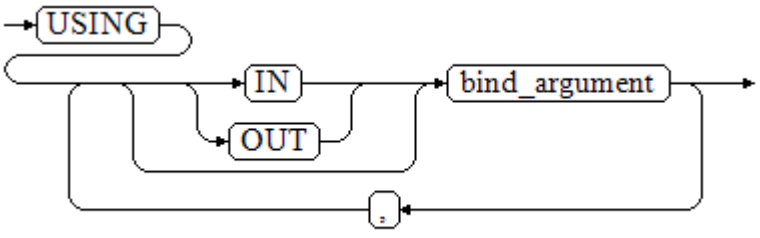
语法图请参见图11-6。

图 11-6 EXECUTE IMMEDIATE dynamic_select_clause::=



using_clause子句的语法图参见图11-7。

图 11-7 using_clause-1



对以上语法格式的解释如下：

- define_variable，用于指定存放单行查询结果的变量。
- USING IN bind_argument，用于指定存放传递给动态SQL值的变量，即在dynamic_select_string中存在占位符时使用。

版权所有 © 华为云计算技术有限公司

- USING OUT bind_argument，用于指定存放动态SQL返回值的变量。

须知

- 查询语句中，into和out不能同时存在；
- 占位符命名以“:”开始，后面可跟数字、字符或字符串，与USING子句的bind_argument一一对应；
- bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象，即不支持使用bind_argument为动态SQL语句传递模式对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause；
- 动态PL/SQL块允许出现重复的占位符，即相同占位符只能与USING子句的一个bind_argument按位置对应。

示例

```
--从动态语句检索值（INTO 子句）：
DECLARE
   staff_count  VARCHAR2(20);
BEGIN
   EXECUTE IMMEDIATE 'select count(*) from staffs'
      INTO staff_count;
   dbms_output.put_line(staff_count);
END;
/

--传递并检索值（INTO子句用在USING子句前）：
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
   EXECUTE IMMEDIATE 'select first_name, salary from staffs where staff_id = :1'
      INTO first_name, salary
      USING IN staff_id;
   dbms_output.put_line(first_name || ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

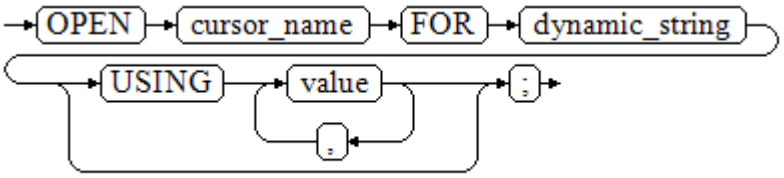
OPEN FOR

动态查询语句还可以使用OPEN FOR打开动态游标来执行。

语法参见图11-8。

 版权所有 © 华为云计算技术有限公司

图 11-8 open_for::=



参数说明：

- cursor_name：要打开的游标名。
- dynamic_string：动态查询语句。
- USING value：在dynamic_string中存在占位符时使用。

游标的使用请参考**DWS存储过程游标**。

示例

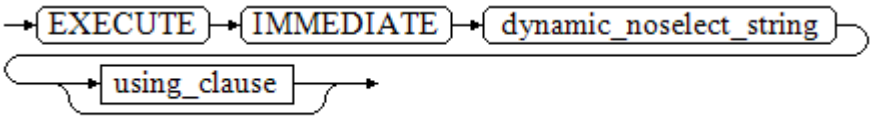
```
DECLARE
   name          VARCHAR2(20);
   phone_number  VARCHAR2(20);
   salary        NUMBER(8,2);
   sqlstr        VARCHAR2(1024);

   TYPE app_ref_cur_type IS REF CURSOR;  --定义游标类型
   my_cur app_ref_cur_type;  --定义游标变量

BEGIN
   sqlstr := 'select first_name,phone_number,salary from staffs
       where section_id = :1';
   OPEN my_cur FOR sqlstr USING '30';  --打开游标, using是可选的
   FETCH my_cur INTO name, phone_number, salary; --获取数据
   WHILE my_cur%FOUND LOOP
         dbms_output.put_line(name||'#'||phone_number||'#'||salary);
         FETCH my_cur INTO name, phone_number, salary;
   END LOOP;
   CLOSE my_cur;   --关闭游标
END;
/
```

11.6.2 执行动态非查询语句

语法

语法请参见**图11-9**。

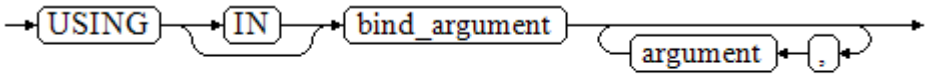
图 11-9 noselect::=



using_clause子句的语法参见**图11-10**。

版权所有 © 华为云计算技术有限公司

图 11-10 using_clause-2



对以上语法格式的解释如下：

USING IN bind_argument用于指定存放传递给动态SQL值的变量，在dynamic_noselect_string中存在占位符时使用，即动态SQL语句执行时，bind_argument将替换相对应的占位符。要注意的是，bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause。另外，动态语句允许出现重复的占位符，相同占位符只能与唯一一个bind_argument按位置一一对应。

示例

```
--创建表
CREATE TABLE sections_t1
(
   section       NUMBER(4) ,
   section_name  VARCHAR2(30),
   manager_id    NUMBER(6),
   place_id      NUMBER(4)
)
DISTRIBUTE BY hash(manager_id);

--声明变量
DECLARE
   section       NUMBER(4) := 280;
   section_name  VARCHAR2(30) := 'Info support';
   manager_id    NUMBER(6) := 103;
   place_id      NUMBER(4) := 1400;
   new_colname   VARCHAR2(10) := 'sec_name';
BEGIN
--执行查询
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :4)'
       USING section, section_name, manager_id,place_id;
--执行查询（重复占位符）
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :1)'
       USING section, section_name, manager_id;
--执行ALTER语句（建议采用“||”拼接数据库对象构造DDL语句）
    EXECUTE IMMEDIATE 'alter table sections_t1 rename section_name to ' || new_colname;
END;
/

--查询数据
SELECT * FROM sections_t1;

--删除表
DROP TABLE sections_t1;
```

11.6.3 动态调用存储过程

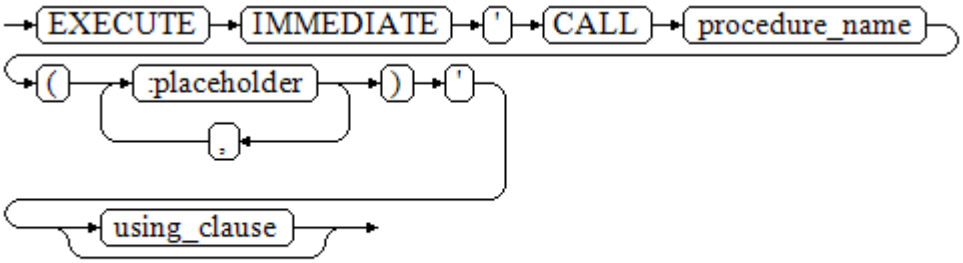
动态调用存储过程必须使用匿名的语句块将存储过程或语句块包在里面，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

语法请参见图11-11。

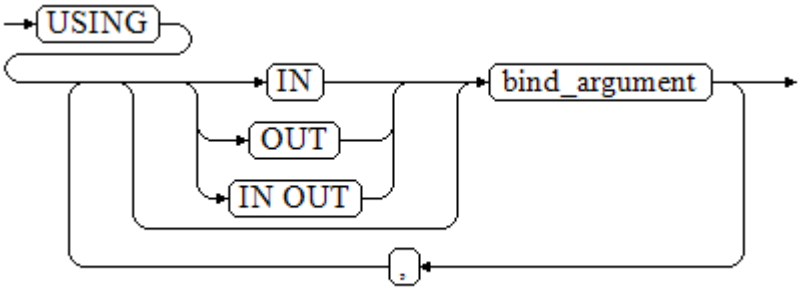
 版权所有 © 华为云计算技术有限公司

图 11-11 call_procedure::=



using_clause子句的语法参见图11-12。

图 11-12 using_clause-3



对以上语法格式的解释如下：

- CALL procedure_name，调用存储过程。
- [:placeholder1，:placeholder2，…]，存储过程参数占位符列表，占位符个数与参数个数相同。占位符命名以“:”或“$”开始，“:”后面可跟数字、字符或字符串（不能使用带引号的数字、字符或字符串），“$”后面仅可跟数字。占位符与USING子句的bind_argument一一对应。
- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。

示例

```
--创建存储过程proc_add。
CREATE OR REPLACE PROCEDURE proc_add
(
    param1    in   INTEGER,
    param2    out  INTEGER,
    param3    in   INTEGER
)
AS
BEGIN
   param2:= param1 + param3;
END;
/

DECLARE
    input1 INTEGER:=1;
    input2 INTEGER:=2;
    statement  VARCHAR2(200);
```

 版权所有 © 华为云计算技术有限公司

```
    param2     INTEGER;
BEGIN
  --声明调用语句
    statement := 'call proc_add(:col_1, :col_2, :col_3)'; ( 或者statement := 'call proc_add($1, $2, $3)'; )
    --执行语句
    EXECUTE IMMEDIATE statement
        USING IN input1, OUT param2, IN input2;
    dbms_output.put_line('result is: '||to_char(param2));
END;
/

--删除存储过程
DROP PROCEDURE proc_add;
```

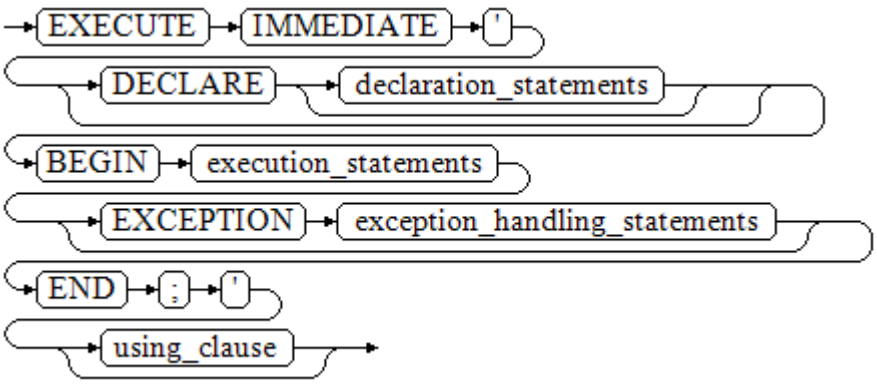
11.6.4 动态调用匿名块

动态调用匿名块是指在动态语句中执行匿名块，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

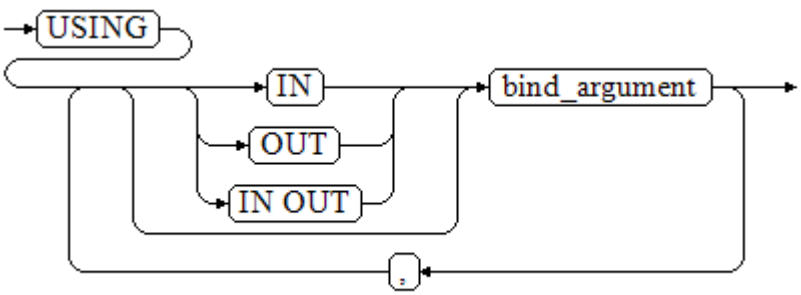
语法请参见图11-13。

图 11-13 call_anonymous_block::=



using_clause子句的语法参见图11-14。

图 11-14 using_clause-4



对以上语法格式的解释如下：

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。

版权所有 © 华为云计算技术有限公司

- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。
- 匿名块中间的输入输出参数使用占位符来指明，要求占位符个数与参数个数相同，并且占位符所对应参数的顺序和USING中参数的顺序一致。
- 目前DWS在动态语句调用匿名块时，EXCEPTION语句中暂不支持使用占位符进行输入输出参数的传递。

示例

```
--创建存储过程dynamic_proc
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
    --执行匿名块
    EXECUTE IMMEDIATE 'begin select first_name, salary into :first_name, :salary from staffs where
staff_id= :dno; end;'
       USING OUT first_name, OUT salary, IN staff_id;
   dbms_output.put_line(first_name|| ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

11.7 DWS 存储过程控制语句

11.7.1 返回语句

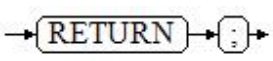
DWS提供两种方式返回数据：RETURN（或RETURN NEXT）及RETURN QUERY。其中，RETURN NEXT和RETURN QUERY只适用于函数，不适用存储过程。

RETURN

语法

返回语句的语法请参见图11-15。

图 11-15 return_clause::=



对以上语法的解释如下：

用于将控制从存储过程或函数返回给调用者。

示例

```
--创建存储过程proc_staffs
CREATE OR REPLACE PROCEDURE proc_staffs
(
section     NUMBER(6),
```


版权所有 © 华为云计算技术有限公司

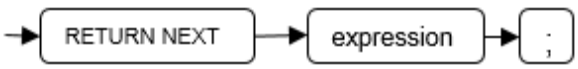
```
salary_sum out NUMBER(8,2),
staffs_count out INTEGER
)
IS
BEGIN
SELECT sum(salary), count(*) INTO salary_sum, staffs_count FROM staffs where section_id = section;
END;
/

--创建存储过程proc_return.
CREATE OR REPLACE PROCEDURE proc_return
AS
v_num NUMBER(8,2);
v_sum INTEGER;
BEGIN
proc_staffs(30, v_sum, v_num);  --调用语句
dbms_output.put_line(v_sum||'#'||v_num);
RETURN;  --返回语句
END;
/

--调用存储过程proc_return.
CALL proc_return();

--清除存储过程
DROP PROCEDURE proc_staffs;
DROP PROCEDURE proc_return;

--创建函数func_return.
CREATE OR REPLACE FUNCTION func_return returns void
language plpgsql
AS $$
DECLARE
v_num INTEGER := 1;
BEGIN
dbms_output.put_line(v_num);
RETURN;  --返回语句
END $$;


-- 调用函数func_return
CALL func_return();
1

-- 清除函数
DROP FUNCTION func_return;
```

RETURN NEXT 及 RETURN QUERY

语法

创建函数时需要指定返回值SETOF datatype。

return_next_clause::=



return_query_clause::=

对以上语法的解释如下：

 版权所有 © 华为云计算技术有限公司

当需要函数返回一个集合时，使用RETURN NEXT或者RETURN QUERY向结果集追加结果，然后继续执行函数的下一条语句。随着后续的RETURN NEXT或RETURN QUERY命令的执行，结果集中会有多个结果。函数执行完成后会一起返回所有结果。

RETURN NEXT可用于标量和复合数据类型。

RETURN QUERY有一种变体RETURN QUERY EXECUTE，后面还可以增加动态查询，通过USING向查询插入参数。

示例

```
CREATE TABLE t1(a int);
INSERT INTO t1 VALUES(1),(10);

--RETURN NEXT
CREATE OR REPLACE FUNCTION fun_for_return_next() RETURNS SETOF t1 AS $$
DECLARE
  r t1%ROWTYPE;
BEGIN
  FOR r IN select * from t1
  LOOP
    RETURN NEXT r;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE PLPGSQL;
call fun_for_return_next();
 a
---
 1
 10
(2 rows)

-- RETURN QUERY
CREATE OR REPLACE FUNCTION fun_for_return_query() RETURNS SETOF t1 AS $$
DECLARE
  r t1%ROWTYPE;
BEGIN
  RETURN QUERY select * from t1;
END;
$$
language plpgsql;
call fun_for_return_next();
 a
---
 1
 10
(2 rows)
```

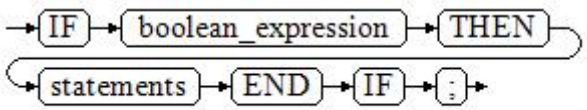
11.7.2 条件语句

条件语句的主要作用是判断参数或者语句是否满足已给定的条件，根据判定结果执行相应的操作。

DWS有五种形式的IF：

- IF_THEN

  图 11-16 IF_THEN::=

 版权所有 © 华为云计算技术有限公司

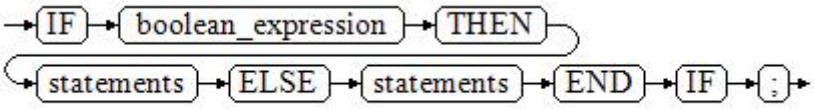
IF_THEN语句是IF的最简单形式。如果条件为真，statements将被执行。否则，将忽略它们的结果使该IF_THEN语句执行结束。

示例

```
IF v_user_id <> 0 THEN
    UPDATE users SET email = v_email WHERE user_id = v_user_id;
END IF;
```

- IF_THEN_ELSE

图 11-17 IF_THEN_ELSE::=



IF_THEN_ELSE语句增加了ELSE的分支，可以声明在条件为假的时候执行的语句。

示例

```
IF parentid IS NULL OR parentid = ''
THEN
    RETURN;
ELSE
    hp_true_filename(parentid);--表示调用存储过程
END IF;
```

- IF_THEN_ELSE IF

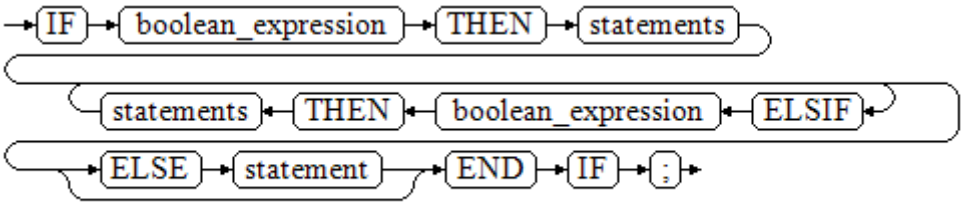
  IF语句可以嵌套，嵌套方式如下：

```
IF gender = 'm' THEN
    pretty_gender := 'man';
ELSE
    IF gender = 'f' THEN
        pretty_gender := 'woman';
    END IF;
END IF;
```

  这种形式实际上就是在一个IF语句的ELSE部分嵌套了另一个IF语句。因此需要给每个嵌套的IF一个END IF语句，另外还需要一个END IF语句结束父IF-ELSE。如果有多个选项，可使用IF_THEN_ELSIF_ELSE形式。

- IF_THEN_ELSIF_ELSE

图 11-18 IF_THEN_ELSIF_ELSE::=



示例

```
IF number_tmp = 0 THEN
    result := 'zero';
ELSIF number_tmp > 0 THEN
    result := 'positive';
ELSIF number_tmp < 0 THEN
    result := 'negative';
```

 版权所有 © 华为云计算技术有限公司

```
ELSE
    result := 'NULL';
END IF;
```

- IF_THEN_ELSEIF_ELSE

  ELSEIF是ELSIF的别名。

  综合示例

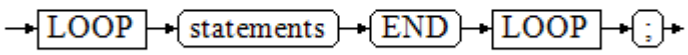
```
CREATE OR REPLACE PROCEDURE proc_control_structure(i in integer)
AS
    BEGIN
        IF i > 0 THEN
            raise info 'i:% is greater than 0. ',i;
        ELSIF i < 0 THEN
            raise info 'i:% is smaller than 0. ',i;
        ELSE
            raise info 'i:% is equal to 0. ',i;
        END IF;
        RETURN;
    END;
/

CALL proc_control_structure(3);

--删除存储过程
DROP PROCEDURE proc_control_structure;
```

11.7.3 循环语句

简单 LOOP 语句

语法图

图 11-19 loop::=



示例

```
CREATE OR REPLACE PROCEDURE proc_loop(i in integer, count out integer)
AS
    BEGIN
        count:=0;
        LOOP
        IF count > i THEN
            raise info 'count is %. ', count;
            EXIT;
        ELSE
            count:=count+1;
        END IF;
        END LOOP;
    END;
/

CALL proc_loop(10,5);
```


版权所有 © 华为云计算技术有限公司

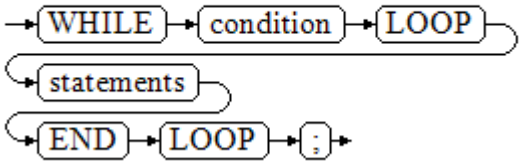
须知

该循环必须要结合EXIT使用，否则将陷入死循环。

WHILE_LOOP 语句

语法图

图 11-20 while_loop::=



只要条件表达式为真，WHILE语句就会不停地在一系列语句上进行循环，在每次进入循环体的时候进行条件判断。

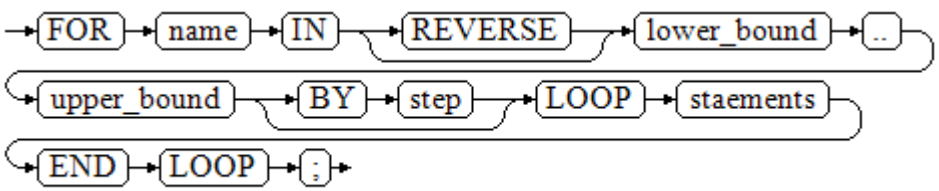
示例

```
CREATE TABLE integertable(c1 integer) DISTRIBUTE BY hash(c1);
CREATE OR REPLACE PROCEDURE proc_while_loop(maxval in integer)
AS
    DECLARE
    i int :=1;
    BEGIN
        WHILE i < maxval LOOP
            INSERT INTO integertable VALUES(i);
            i:=i+1;
        END LOOP;
    END;
/

--调用函数
CALL proc_while_loop(10);

--删除存储过程和表
DROP PROCEDURE proc_while_loop;
DROP TABLE integertable;
```

FOR_LOOP（integer 变量）语句

语法图

图 11-21 for_loop::=



 版权所有 © 华为云计算技术有限公司

说明

- 变量name会自动定义为integer类型并且只在此循环里存在。变量name介于lower_bound和upper_bound之间。
- 当使用REVERSE关键字时，lower_bound必须大于等于upper_bound，否则循环体不会被执行。

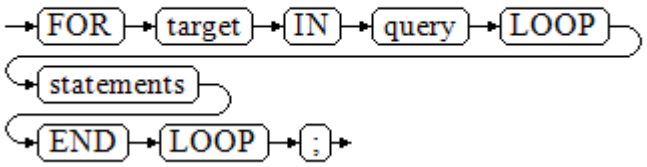
示例

```
--从0到5进行循环
CREATE OR REPLACE PROCEDURE proc_for_loop()
AS
    BEGIN
    FOR I IN 0..5 LOOP
        DBMS_OUTPUT.PUT_LINE('It is '||to_char(I) || ' time;') ;
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop();

--删除存储过程
DROP PROCEDURE proc_for_loop;
```

FOR_LOOP 查询语句

语法图

图 11-22 for_loop_query::=



说明

变量target会自动定义，类型和query的查询结果的类型一致，并且只在此循环中有效。target的取值就是query的查询结果。

示例

```
--循环输出查询结果。
CREATE OR REPLACE PROCEDURE proc_for_loop_query()
AS
    record VARCHAR2(50);
BEGIN
    FOR record IN SELECT spcname FROM pg_tablespace LOOP
    dbms_output.put_line(record);
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop_query();
```

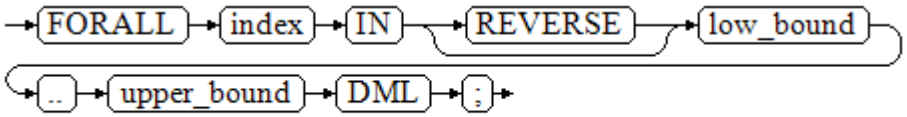
 版权所有 © 华为云计算技术有限公司

```
--删除存储过程
DROP PROCEDURE proc_for_loop_query;
```

FORALL 批量查询语句

语法图

图 11-23 forall::=



说明

变量index会自动定义为integer类型并且只在此循环里存在。index的取值介于low_bound和upper_bound之间。

示例

```
CREATE TABLE hdfs_t1 (
  title NUMBER(6),
  did VARCHAR2(20),
  data_period VARCHAR2(25),
  kind VARCHAR2(25),
  interval VARCHAR2(20),
  time DATE,
  isModified VARCHAR2(10)
)
DISTRIBUTE BY hash(did);

INSERT INTO hdfs_t1 VALUES( 8, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', to_date('21-06-1999',
'dd-mm-yyyy'), 'SH_CLERK' );

CREATE OR REPLACE PROCEDURE proc_forall()
AS
BEGIN
    FORALL i IN 100..120
        insert into hdfs_t1(title) values(i);
END;
/

--调用函数
CALL proc_forall();

--查询存储过程调用结果
SELECT * FROM hdfs_t1 WHERE title BETWEEN 100 AND 120;

--删除存储过程和表
DROP PROCEDURE proc_forall;
DROP TABLE hdfs_t1;
```

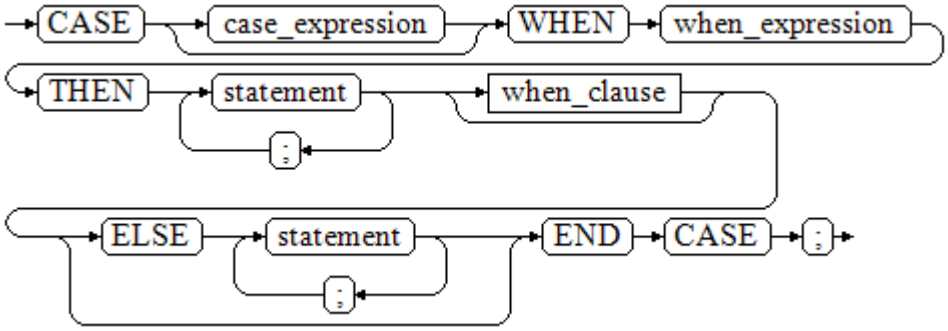
11.7.4 分支语句

语法

分支语句的语法请参见图11-24。

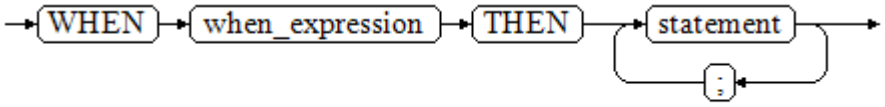
 版权所有 © 华为云计算技术有限公司

图 11-24 case_when::=



when_clause子句的语法图参见图11-25。

图 11-25 when_clause::=



参数说明：

- case_expression：变量或表达式。
- when_expression：常量或者条件表达式。
- statement：执行语句。

示例

```
CREATE OR REPLACE PROCEDURE proc_case_branch(pi_result in integer, pi_return out integer)
AS
   BEGIN
      CASE pi_result
         WHEN 1 THEN
            pi_return := 111;
         WHEN 2 THEN
            pi_return := 222;
         WHEN 3 THEN
            pi_return := 333;
         WHEN 6 THEN
            pi_return := 444;
         WHEN 7 THEN
            pi_return := 555;
         WHEN 8 THEN
            pi_return := 666;
         WHEN 9 THEN
            pi_return := 777;
         WHEN 10 THEN
            pi_return := 888;
         ELSE
            pi_return := 999;
      END CASE;
      raise info 'pi_return : %',pi_return ;
END;
/

CALL proc_case_branch(3,0);
```


版权所有 © 华为云计算技术有限公司

```
--删除存储过程
DROP PROCEDURE proc_case_branch;
```

11.7.5 空语句

在PL/SQL程序中，可以用NULL语句来说明“不用做任何事情”，即空语句。

空语句相当于一个占位符，可以使某些语句变得有意义，提高程序的可读性。

语法

空语句的用法如下：

```
DECLARE
   ...
BEGIN
   ...
   IF v_num IS NULL THEN
      NULL; -- 不需要处理任何数据。
   END IF;
END;
/
```

11.7.6 错误捕获语句

缺省时，当PL/SQL函数执行过程中发生错误时退出函数执行，并且周围的事务也会回滚。可以用一个带有EXCEPTION子句的BEGIN块捕获错误并且从中恢复。其语法是正常的BEGIN块语法的一个扩展：

```
[<<label>>]
[DECLARE
   declarations]
BEGIN
   statements
EXCEPTION
   WHEN condition [OR condition ...] THEN
      handler_statements
   [WHEN condition [OR condition ...] THEN
      handler_statements
   ...]
END;
```

如果没有发生错误，这种形式的块只是简单地执行所有语句，然后转到END之后的下一个语句。但是如果在执行的语句内部发生了一个错误，则这个语句将会回滚，然后转到EXCEPTION列表寻找匹配错误的第一个条件。若找到匹配，则执行对应的handler_statements，然后转到END之后的下一个语句。如果没有找到匹配，则会向事务的外层报告错误，和没有EXCEPTION子句一样。

也就是说该错误可以被一个包围块用EXCEPTION捕获，如果没有包围块，则进行退出函数处理。

condition的名字遵循SQL标准错误码编号说明的任意值。特殊的条件名OTHERS匹配除了QUERY_CANCELED之外的所有错误类型。

如果在选中的handler_statements里发生了新错误，则不能被这个EXCEPTION子句捕获，而是向事务的外层报告错误。一个外层的EXCEPTION子句可以捕获它。

如果一个错误被EXCEPTION捕获，PL/SQL函数的局部变量保持错误发生时的原值，但是所有该块中想写入数据库中的状态都回滚。

示例：

```
CREATE TABLE mytab(id INT,firstname VARCHAR(20),lastname VARCHAR(20)) DISTRIBUTE BY hash(id);
```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO mytab(firstname, lastname) VALUES('Tom', 'Jones');

CREATE FUNCTION fun_exp() RETURNS INT
AS $$
DECLARE
    x INT :=0;
    y INT;
BEGIN
    UPDATE mytab SET firstname = 'Joe' WHERE lastname = 'Jones';
    x := x + 1;
    y := x / 0;
EXCEPTION
    WHEN division_by_zero THEN
        RAISE NOTICE 'caught division_by_zero';
        RETURN x;
END;$$
LANGUAGE plpgsql;

CALL fun_exp();
NOTICE:  caught division_by_zero
 fun_exp
---------
       1
(1 row)

SELECT * FROM mytab;
 id | firstname | lastname
----+-----------+----------
    | Tom       | Jones
(1 row)

DROP FUNCTION fun_exp();
DROP TABLE mytab;
```

当控制执行到给y赋值的步骤时，若发生division_by_zero错误，该错误会被EXCEPTION子句捕获。此时，RETURN语句里返回的数值将是x的增量值。

说明

进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大的多。因此，不必要的时候不要使用EXCEPTION。

在下列场景中，无法捕获处理异常，整个存储过程回滚：节点故障、网络故障引起的存储过程参与节点线程退出以及COPY FROM操作中源数据与目标表的表结构不一致造成的异常。

示例：UPDATE/INSERT异常。

根据使用异常处理器执行恰当的UPDATE或INSERT 。

```
CREATE TABLE db (a INT, b TEXT);

CREATE FUNCTION merge_db(key INT, data TEXT) RETURNS VOID AS
$$
BEGIN
    LOOP

--第一次尝试更新key
        UPDATE db SET b = data WHERE a = key;
        IF found THEN
            RETURN;
        END IF;
--不存在，所以尝试插入key，如果其他人同时插入相同的key，可能得到唯一key失败。
        BEGIN
            INSERT INTO db(a,b) VALUES (key, data);
            RETURN;
        EXCEPTION WHEN unique_violation THEN
        --什么也不做，并且循环尝试再次更新。
        END;
    END LOOP;
```


版权所有 © 华为云计算技术有限公司

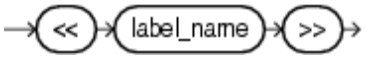
```
END;
$$
LANGUAGE plpgsql;

SELECT merge_db(1, 'david');
SELECT merge_db(1, 'dennis');

--删除FUNCTION和TABLE
DROP FUNCTION merge_db;
DROP TABLE db ;
```

11.7.7 GOTO 语句

GOTO语句可以实现从GOTO位置到目标语句的无条件跳转。GOTO语句会改变原本的执行逻辑，因此应该慎重使用，或者也可以使用EXCEPTION处理特殊场景。当执行GOTO语句时，目标Label必须是唯一的。

语法

label declaration ::=



goto statement ::=

示例

```
CREATE OR REPLACE PROCEDURE GOTO_test()
AS
DECLARE
    v1  int;
BEGIN
    v1  := 0;
        LOOP
        EXIT WHEN v1 > 100;
                v1 := v1 + 2;
                if v1 > 25 THEN
                        GOTO pos1;
                END IF;
        END LOOP;
<<pos1>>
v1 := v1 + 10;
raise info 'v1 is %. ', v1;
END;
/

call GOTO_test();
DROP PROCEDURE GOTO_test();
```

限制场景

GOTO使用有以下限制场景

- 不支持有多个相同的GOTO labels目标场景，无论是否在同一个block中。

  ```
  BEGIN
    GOTO pos1;
    <<pos1>>
    SELECT * FROM ...
    <<pos1>>
  ```


版权所有 © 华为云计算技术有限公司

```
  UPDATE t1 SET ...
END;
```

- 不支持GOTO跳转到IF语句、CASE语句、LOOP语句中。

```
BEGIN
  GOTO pos1;
  IF valid THEN
    <<pos1>>
    SELECT * FROM ...
  END IF;
END;
```

- 不支持GOTO语句从一个IF子句跳转到另一个IF子句，或从一个CASE语句的WHEN子句跳转到另一个WHEN子句。

```
BEGIN
  IF valid THEN
    GOTO pos1;
    SELECT * FROM ...
  ELSE
    <<pos1>>
    UPDATE t1 SET ...
  END IF;
END;
```

- 不支持从外部块跳转到内部的BEGIN-END块。

```
BEGIN
  GOTO pos1;
  BEGIN
    <<pos1>>
    UPDATE t1 SET ...
  END;
END;
```

- 不支持从异常处理部分跳转到当前的BEGIN-END块，但可以跳转到上层BEGIN-END块。

```
BEGIN
  <<pos1>>
  UPDATE t1 SET ...
  EXCEPTION
    WHEN condition THEN
      GOTO pos1;
END;
```

- 如果从GOTO到一个不包含执行语句的位置，需要添加NULL语句。

```
DECLARE
  done  BOOLEAN;
BEGIN
  FOR i IN 1..50 LOOP
    IF done THEN
      GOTO end_loop;
    END IF;
    <<end_loop>>  -- not allowed unless an executable statement follows
    NULL; -- add NULL statement to avoid error
  END LOOP;  -- raises an error without the previous NULL
END;
/
```

11.8 DWS 存储过程其他语句

锁操作

DWS提供了多种锁模式用于控制对表中数据的并发访问。这些模式可以用在MVCC（多版本并发控制）无法给出期望行为的场合。同样，大多数DWS命令自动施加恰当的锁，以保证被引用的表在命令的执行过程中不会以一种不兼容的方式被删除或者修改。比如，在存在其他并发操作的时候，ALTER TABLE是不能在同一个表上执行的。

版权所有 © 华为云计算技术有限公司

游标操作

DWS中游标（cursor）是系统为用户开设的一个数据缓冲区，存放着SQL语句的执行结果。每个游标区都有一个名字。用户可以用SQL语句逐一从游标中获取记录，并赋给主变量，交由主语言进一步处理。

游标的操作主要有游标的定义、打开、获取和关闭。

完整的游标操作示例可参考显式游标。

11.9 DWS 存储过程游标

11.9.1 游标概述

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

须知

当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。

游标的使用分为显式游标和隐式游标。对于不同的SQL语句，游标的使用情况不同，详细信息请参见表11-2。

表 11-2 游标使用情况

SQL语句	游标
非查询语句	隐式的
结果是单行的查询语句	隐式的或显式的
结果是多行的查询语句	显式的

11.9.2 显式游标

显式游标主要用于对查询语句的处理，尤其是查询结果为多条记录的情况。

处理步骤

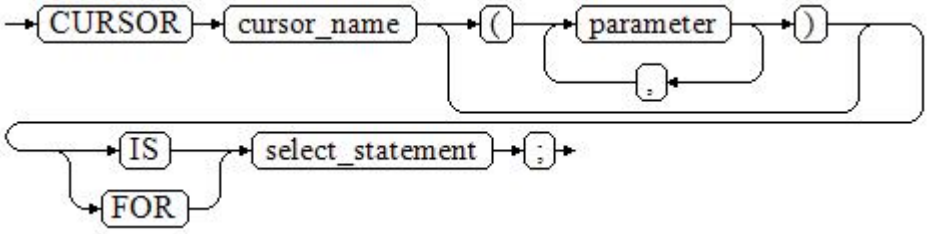
显式游标处理需六个PL/SQL步骤：

步骤1 **定义静态游标：**就是定义一个游标名，以及与其相对应的SELECT语句。

定义静态游标的语法图，请参见图11-26。

 版权所有 © 华为云计算技术有限公司

图 11-26 static_cursor_define::=



参数说明：

- cursor_name：定义的游标名。
- parameter：游标参数，只能为输入参数，其格式为：
  parameter_name datatype
- select_statement：查询语句。

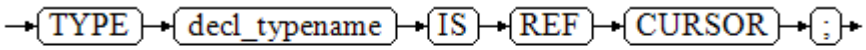
说明

根据执行计划的不同，系统会自动判断该游标是否可以用于以倒序的方式检索数据行。

定义动态游标：指ref游标，可以通过一组静态的SQL语句动态的打开游标。首先定义ref游标类型，然后定义该游标类型的游标变量，在打开游标时通过OPEN FOR动态绑定SELECT语句。

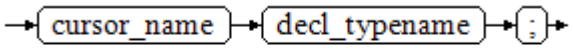
定义动态游标的语法图，请参见图11-27和图11-28。

图 11-27 cursor_typename::=



DWS支持sys_refcursor动态游标类型，函数或存储过程可以通过sys_refcursor参数传入或传出游标结果集合，函数也可以通过返回sys_refcursor来返回游标结果集合。

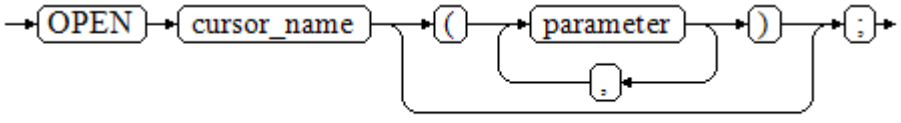
图 11-28 dynamic_cursor_define::=



步骤2 **打开静态游标：**就是执行游标所对应的SELECT语句，将其查询结果放入工作区，并且指针指向工作区的首部，标识游标结果集合。如果游标查询语句中带有FOR UPDATE选项，OPEN语句还将锁定数据库表中游标结果集合对应的数据行。

打开静态游标的语法图，请参见图11-29。

图 11-29 open_static_cursor::=

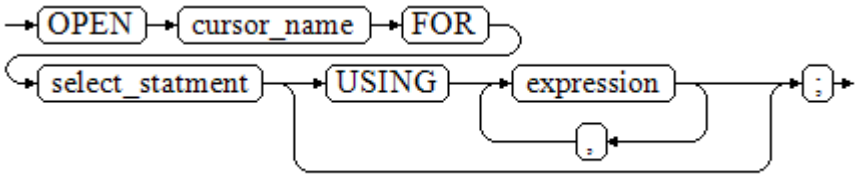


 版权所有 © 华为云计算技术有限公司

打开动态游标：可以通过OPEN FOR语句打开动态游标，动态绑定SQL语句。

打开动态游标的语法图，请参见图11-30。

图 11-30 open_dynamic_cursor::=

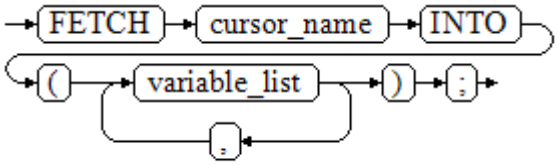


PL/SQL程序不能用OPEN语句重复打开一个游标。

步骤3 提取游标数据：检索结果集合中的数据行，放入指定的输出变量中。

提取游标数据的语法图，请参见图11-31。

图 11-31 fetch_cursor::=



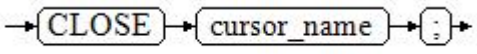
步骤4 对该记录进行处理。

步骤5 继续处理，直到活动集合中没有记录。

步骤6 关闭游标：当提取和处理完游标结果集合数据后，应及时关闭游标，以释放该游标所占用的系统资源，并使该游标的工作区变成无效，不能再使用FETCH语句获取其中数据。关闭后的游标可以使用OPEN语句重新打开。

关闭游标的语法图，请参见图11-32。

图 11-32 close_cursor::=



----结束

属性

游标的属性用于控制程序流程或者了解程序的状态。当运行DML语句时，PL/SQL打开一个内建游标并处理结果，游标是维护查询结果的内存中的一个区域，游标在运行DML语句时打开，完成后关闭。显式游标的属性为：

- %FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- %NOTFOUND布尔型属性：与%FOUND相反。
- %ISOPEN布尔型属性：当游标已打开时返回TRUE。

 版权所有 © 华为云计算技术有限公司

- %ROWCOUNT数值型属性：返回已从游标中读取的记录数。

示例

```
--游标参数的传递方法。
CREATE OR REPLACE PROCEDURE cursor_proc1()
AS
DECLARE
    DEPT_NAME VARCHAR(100);
    DEPT_LOC NUMBER(4);
    --定义游标
    CURSOR C1 IS
        SELECT section_name, place_id FROM sections WHERE section_id <= 50;
    CURSOR C2(sect_id INTEGER) IS
        SELECT section_name, place_id FROM sections WHERE section_id <= sect_id;
    TYPE CURSOR_TYPE IS REF CURSOR;
    C3 CURSOR_TYPE;
    SQL_STR VARCHAR(100);
BEGIN
    OPEN C1;--打开游标
    LOOP
        --通过游标取值
        FETCH C1 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C1;--关闭游标

    OPEN C2(10);
    LOOP
        FETCH C2 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C2%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C2;

    SQL_STR := 'SELECT section_name, place_id FROM sections WHERE section_id <= :DEPT_NO;';
    OPEN C3 FOR SQL_STR USING 50;
    LOOP
        FETCH C3 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C3%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C3;
END;
/

CALL cursor_proc1();

DROP PROCEDURE cursor_proc1;
--给工资低于3000的员工增加工资500。
CREATE TABLE staffs_t1 AS TABLE staffs;

CREATE OR REPLACE PROCEDURE cursor_proc2()
AS
DECLARE
  V_EMPNO  NUMBER(6);
  V_SAL    NUMBER(8,2);
  CURSOR C IS SELECT staff_id, salary FROM staffs_t1;
BEGIN
  OPEN C;
  LOOP
    FETCH C INTO V_EMPNO, V_SAL;
    EXIT WHEN C%NOTFOUND;
    IF V_SAL<=3000 THEN
         UPDATE staffs_t1 SET salary =salary + 500 WHERE staff_id = V_EMPNO;
    END IF;
  END LOOP;
  CLOSE C;
```

 版权所有 © 华为云计算技术有限公司

```
END;
/

CALL cursor_proc2();

--删除存储过程
DROP PROCEDURE cursor_proc2;
DROP TABLE staffs_t1;
--SYS_REFCURSOR类型作为函数参数
CREATE OR REPLACE PROCEDURE proc_sys_ref(O OUT SYS_REFCURSOR)
IS
C1 SYS_REFCURSOR;
BEGIN
OPEN C1 FOR SELECT section_ID FROM sections ORDER BY section_ID;
O := C1;
END;
/

DECLARE
C1 SYS_REFCURSOR;
TEMP NUMBER(4);
BEGIN
proc_sys_ref(C1);
LOOP
  FETCH C1 INTO TEMP;
  DBMS_OUTPUT.PUT_LINE(C1%ROWCOUNT);
  EXIT WHEN C1%NOTFOUND;
END LOOP;
END;
/

--删除存储过程
DROP PROCEDURE proc_sys_ref;
```

11.9.3 隐式游标

对于非查询语句，如修改、删除操作，则由系统自动地为这些操作设置游标并创建其工作区，这些由系统隐含创建的游标称为隐式游标，隐式游标的名字为SQL，这是由系统定义的。

简介

对于隐式游标的操作，如定义、打开、取值及关闭操作，都由系统自动地完成，无需用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。在隐式游标的工作区中，所存放的数据是最新处理的一条SQL语句所包含的数据，与用户自定义的显式游标无关。

格式调用为： SQL%

说明

INSERT，UPDATE，DROP，SELECT语句中不必明确定义游标。

属性

隐式游标属性为：

- SQL%FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- SQL%NOTFOUND布尔型属性：与%FOUND相反。
- SQL%ROWCOUNT数值型属性：返回已从游标中读取的记录数。
- SQL%ISOPEN布尔型属性：取值总是FALSE。SQL语句执行完毕立即关闭隐式游标。

 版权所有 © 华为云计算技术有限公司

示例

```
--删除EMP表中某部门的所有员工，如果该部门中已没有员工，则在DEPT表中删除该部门。
CREATE TABLE staffs_t1 AS TABLE staffs;
CREATE TABLE sections_t1 AS TABLE sections;

CREATE OR REPLACE PROCEDURE proc_cursor3()
AS
   DECLARE
   V_DEPTNO NUMBER(4) := 100;
   BEGIN
      DELETE FROM staffs WHERE section_ID = V_DEPTNO;
      --根据游标状态做进一步处理
      IF SQL%NOTFOUND THEN
      DELETE FROM sections_t1 WHERE section_ID = V_DEPTNO;
      END IF;
   END;
/

CALL proc_cursor3();

--删除存储过程和临时表
DROP PROCEDURE proc_cursor3;
DROP TABLE staffs_t1;
DROP TABLE sections_t1;
```

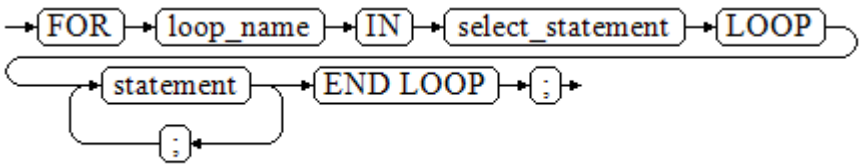
11.9.4 游标循环

游标在WHILE语句、LOOP语句中的使用称为游标循环，一般这种循环都需要使用OPEN、FETCH和CLOSE语句。下面要介绍的一种循环不需要这些操作，可以简化游标循环的操作，这种循环方式适用于静态游标的循环，不用执行静态游标的四个步骤。

语法

FOR AS循环的语法请参见**图11-33**。

图 11-33 FOR_AS_loop::=



注意事项

- 不能在该循环语句中对查询的表进行更新操作。
- 变量loop_name会自动定义且只在此循环中有效，类型和select_statement的查询结果类型一致。loop_name的取值就是select_statement的查询结果。
- 游标的属性中%FOUND、%NOTFOUND、%ROWCOUNT在DWS数据库中都是访问同一个内部变量，事务和匿名块不支持多个游标同时访问。

示例

```
BEGIN
FOR ROW_TRANS IN
      SELECT first_name FROM staffs
   LOOP
      DBMS_OUTPUT.PUT_LINE (ROW_TRANS.first_name );
```


版权所有 © 华为云计算技术有限公司

```
    END LOOP;
END;
/

--创建表
CREATE TABLE integerTable1( A INTEGER) DISTRIBUTE BY hash(A);
CREATE TABLE integerTable2( B INTEGER) DISTRIBUTE BY hash(B);
INSERT INTO integerTable2 VALUES(2);

--多游标共享游标属性的标量
DECLARE
  CURSOR C1 IS SELECT A FROM integerTable1;--声明游标
  CURSOR C2 IS SELECT B FROM integerTable2;
  PI_A INTEGER;
  PI_B INTEGER;
BEGIN
  OPEN C1;--打开游标
  OPEN C2;
  FETCH C1 INTO PI_A; ----  C1%FOUND 和 C2%FOUND 值为 FALSE
  FETCH C2 INTO PI_B; ----  C1%FOUND 和 C2%FOUND 的值都为 TRUE
  --判断游标状态
  IF C1%FOUND THEN
     IF C2%FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Dual cursor share parameter.');
    END IF;
  END IF;
  CLOSE C1;--关闭游标
  CLOSE C2;
END;
/

--删除临时表
DROP TABLE integerTable1;
DROP TABLE integerTable2;
```

11.10 DWS 存储过程高级包

11.10.1 DBMS_LOB

接口介绍

高级功能包DBMS_LOB支持的所有接口请参见**表11-3**。

表 11-3 DBMS_LOB

接口名称	描述
DBMS_LOB.GETLENGTH	获取并返回指定的LOB类型对象的长度。
DBMS_LOB.OPEN	打开一个LOB返回一个LOB的描述符。
DBMS_LOB.READ	根据指定的长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。
DBMS_LOB.WRITE	根据指定长度及起始位置偏移将BUFFER中内容写入到LOB中。
DBMS_LOB.WRITEAPPEND	根据指定长度将BUFFER中内容写入到LOB的尾部。

版权所有 © 华为云计算技术有限公司

接口名称	描述
DBMS_LOB.COPY	根据指定长度及起始位置偏移将BLOB内容写入到另一个BLOB中。
DBMS_LOB.ERASE	根据指定长度及起始位置偏移删除BLOB中的内容。
DBMS_LOB.CLOSE	关闭已经打开的LOB描述符。
DBMS_LOB.INSTR	返回一个字符串在LOB中第N次出现的位置。
DBMS_LOB.COMPARE	比较两个LOB或者两个LOB的某一部分。
DBMS_LOB.SUBSTR	用于读取一个LOB的子串，返回读取的字节个数或者字符个数。
DBMS_LOB.TRIM	用于截断指定长度的LOB，执行完会将LOB的长度设置为newlen参数指定的长度。
DBMS_LOB.CREATETEMPORARY	创建一个临时的BLOB或者CLOB。
DBMS_LOB.APPEND	将原LOB的内容拼接到目的LOB中。

- DBMS_LOB.GETLENGTH

  存储过程GETLENGTH获取并返回指定的LOB类型对象的长度。

  DBMS_LOB.GETLENGTH函数原型为：

  ```
  DBMS_LOB.GETLENGTH (
  lob_loc    IN   BLOB)
  RETURN INTEGER;

  DBMS_LOB.GETLENGTH (
  lob_loc    IN   CLOB)
  RETURN INTEGER;
  ```

  表 11-4 DBMS_LOB.GETLENGTH 接口参数说明

参数	描述
lob_loc	待获取长度的指定的LOB类型的对象。

- DBMS_LOB.OPEN

  存储过程打开一个LOB，并返回一个LOB描述符，该过程无实际意义，仅用于兼容。

  DBMS_LOB.OPEN函数原型为：

  ```
  DBMS_LOB.LOB (
  lob_loc    INOUT   BLOB,
  open_mode IN BINARY_INTEGER);

  DBMS_LOB.LOB (
  lob_loc  INOUT    CLOB,
  open_mode IN BINARY_INTEGER);
  ```


版权所有 © 华为云计算技术有限公司

表 11-5 DBMS_LOB.OPEN 接口参数说明

参数	描述
lob_loc	被打开的一个BLOB或者CLOB描述符。
open_mode IN BINARY_INTEGER	打开的模式，目前支持DBMS_LOB.LOB_READWRITE

- DBMS_LOB.READ

  存储过程READ根据指定长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。

  DBMS_LOB.READ函数原型为：

```
DBMS_LOB.READ (
lob_loc     IN          BLOB,
amount      IN          INTEGER,
offset      IN          INTEGER,
buffer      OUT         RAW);

DBMS_LOB.READ (
lob_loc    IN           CLOB,
amount     IN OUT       INTEGER,
offset     IN           INTEGER,
buffer     OUT          VARCHAR2);
```

表 11-6 DBMS_LOB.READ 接口参数说明

参数	说明
lob_loc	待读入的指定的LOB类型的对象。
amount	读入长度。 **说明** 如果读入长度为负，会收到错误提示“ERROR: argument 2 is null, invalid, or out of range.”
offset	指定从LOB内容的哪个位置开始读取的偏移（即相对LOB内容起始位置的字节数）。
buffer	读取LOB内容后存放的目标缓冲区。

- DBMS_LOB.WRITE

  存储过程WRITE根据指定长度及起始位置偏移将BUFFER中内容写入到LOB变量中。

  DBMS_LOB.WRITE函数原型为：

```
DBMS_LOB.WRITE (
lob_loc    IN OUT     BLOB,
amount     IN         INTEGER,
offset     IN         INTEGER,
buffer     IN         RAW);

DBMS_LOB.WRITE (
lob_loc   IN OUT      CLOB,
amount    IN          INTEGER,
offset    IN          INTEGER,
buffer    IN          VARCHAR2);
```


版权所有 © 华为云计算技术有限公司

表 11-7 DBMS_LOB.WRITE 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。
offset	指定从LOB内容的哪个位置开始写入的偏移（即相对LOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于LOB类型最大长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.WRITEAPPEND

  存储过程WRITEAPPEND根据指定长度将BUFFER中内容写入到LOB的尾部。

  DBMS_LOB.WRITEAPPEND函数原型为：

```
DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     BLOB,
amount      IN         INTEGER,
buffer      IN         RAW);

DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     CLOB,
amount      IN         INTEGER,
buffer      IN         VARCHAR2);
```

表 11-8 DBMS_LOB.WRITEAPPEND 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.COPY

  存储过程COPY根据指定长度及起始位置偏移将BLOB内容复制到另一个BLOB中。

  DBMS_LOB.COPY函数原型为：

```
DBMS_LOB.COPY (
dest_lob      IN OUT     BLOB,
src_lob       IN         BLOB,
amount        IN         INTEGER,
dest_offset   IN         INTEGER  DEFAULT 1,
src_offset    IN         INTEGER  DEFAULT 1);
```


版权所有 © 华为云计算技术有限公司

表 11-9 DBMS_LOB.COPY 接口参数说明

参数	说明
dest_lob	待拷入的BLOB类型对象。
src_lob	待拷出的BLOB类型对象。
amount	复制长度。 **说明** 如果拷入长度小于1或拷入长度大于BLOB类型最大长度，则报错。
dest_offset	指定从BLOB内容的哪个位置开始拷入的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。
src_offset	指定从BLOB内容的哪个位置开始拷出的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于复制来源BLOB的长度，则报错。

- DBMS_LOB.ERASE

  存储过程ERASE根据指定长度及起始位置偏移删除BLOB中的内容。

  DBMS_LOB.ERASE函数原型为：

```
DBMS_LOB.ERASE (
lob_loc          IN OUT   BLOB,
amount           IN OUT   INTEGER,
offset           IN       INTEGER DEFAULT 1);
```

表 11-10 DBMS_LOB.ERASE 接口参数说明

参数	说明
lob_loc	待删除内容的BLOB类型对象。
amount	待删除的长度。 **说明** 如果删除长度小于1或删除长度大于BLOB类型最大长度，则报错。
offset	指定从BLOB内容的哪个位置开始删除的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。

- DBMS_LOB.CLOSE

  存储过程CLOSE根据指定长度及起始位置偏移关闭已经打开的LOB内容。

  DBMS_LOB.CLOSE函数原型为：

```
DBMS_LOB.CLOSE(
src_lob      IN             BLOB);

DBMS_LOB.CLOSE (
src_lob      IN             CLOB);
```


版权所有 © 华为云计算技术有限公司

表 11-11 DBMS_LOB.CLOSE 接口参数说明

参数	说明
src_loc	待关闭的LOB类型对象。

- DBMS_LOB.INSTR

  该函数返回在LOB中第N次出现的位置，如果输入的是一些无效值会返回NULL值。offset < 1 or offset > LOBMAXSIZE，nth < 1，nth > LOBMAXSIZE。

  DBMS_LOB.INSTR函数原型为：

```
DBMS_LOB.INSTR (
lob_loc     IN     BLOB,
pattern     IN     RAW,
offset      IN     INTEGER := 1,
nth         IN     INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.INSTR (
lob_loc    IN     CLOB,
pattern    IN     VARCHAR2 ,
offset     IN     INTEGER := 1,
nth        IN     INTEGER := 1)
RETURN INTEGER;
```

表 11-12 DBMS_LOB.INSTR 接口参数说明

参数	说明
lob_loc	要查找的LOB描述符。
pattern	要匹配的模式，对于BLOB是由一组RAW类型的数据组成，对于CLOB是由一组text类型的数据组成。
offset	对于BLOB是以字节为单位的绝对偏移量，对于CLOB是以字符为单位的偏移量，模式匹配的起始位置是1。
nth	模式匹配的次数，最小值为1。

- DBMS_LOB.COMPARE

  这个函数比较两个LOB或者两个LOB的一部分。

  - 如果比较的结果相等返回0，否则返回非零的值。
  - 如果第一个CLOB比第二个小，返回-1；如果第一个CLOB比第二个大，返回1。
  - 如果amount，offset_1，offset_2这几个参数有无效参数返回NULL，有效的偏移量范围是1~LOBMAXSIZE。

  DBMS_LOB.READ函数原型为：

```
DBMS_LOB.COMPARE (
lob_1     IN     BLOB,
lob_2     IN     BLOB,
amount    IN     INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN     INTEGER := 1,
offset_2  IN     INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.COMPARE (
lob_1     IN     CLOB,
```

 版权所有 © 华为云计算技术有限公司

```
lob_2     IN     CLOB,
amount    IN     INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN     INTEGER := 1,
offset_2  IN     INTEGER := 1)
RETURN INTEGER;
```

表 11-13 DBMS_LOB.COMPARE 接口参数说明

参数	说明
lob_1	第一个要比较的LOB描述符。
lob_2	第二个要比较的LOB描述符。
amount	要比较的字符数或者字节数， 最大值为DBMS_LOB.LOBMAXSIZE。
offset_1	第一个LOB描述符的偏移量，初始位置是1。
offset_2	第二个LOB描述符的偏移量，初始位置是1。

- DBMS_LOB.SUBSTR

  用于读取一个LOB的子串，返回读取的字节个数或者字符个数，当amount > 1或者amount < 32767，offset < 1或者offset > LOBMAXSIZE的时候返回值是NULL。

  DBMS_LOB.SUBSTR函数原型为：

  ```
  DBMS_LOB.SUBSTR (
  lob_loc     IN     BLOB,
  amount      IN     INTEGER := 32767,
  offset      IN     INTEGER := 1)
  RETURN RAW;

  DBMS_LOB.SUBSTR (
  lob_loc     IN     CLOB,
  amount      IN     INTEGER := 32767,
  offset      IN     INTEGER := 1)
  RETURN VARCHAR2;
  ```

  表 11-14 DBMS_LOB.SUBSTR 接口参数说明

参数	说明
lob_loc	将要读取子串的LOB描述符，对于BLOB类型的返回值是读取的字节个数，对于CLOB类型的返回值是字符个数。
offset	要读取的字节数或者字符数量。
buffer	从开始位置偏移的字符数或者字节数量。

- DBMS_LOB.TRIM

  这个存储过程用于截断指定长度的LOB，执行完这个存储过程会将LOB的长度设置为newlen参数指定的长度。如果对一个空的LOB执行截断操作，不会有任何执行结果；如果指定的长度比LOB的长度长，会产生一个异常。

  DBMS_LOB.TRIM函数原型为：

  ```
  DBMS_LOB.TRIM (
  lob_loc     IN OUT     BLOB,
  ```

 版权所有 © 华为云计算技术有限公司

```
newlen      IN          INTEGER);

DBMS_LOB.TRIM (
lob_loc    IN          OUT CLOB,
newlen     IN          INTEGER);
```

表 11-15 DBMS_LOB.TRIM 接口参数说明

参数	说明
lob_loc	待读入的指定BLOB类型的对象。
newlen	截断后LOB的新长度对于BLOB是字节数，对于CLOB是字符数。

- DBMS_LOB.CREATETEMPORARY

  这个存储过程创建一个临时的BLOB或者CLOB，这个存储过程仅用于语法上的兼容，并无实际意义。

  DBMS_LOB.CREATETEMPORARY函数原型为：

```
DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT      BLOB,
cache      IN          BOOLEAN,
dur        IN          INTEGER);

DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT     CLOB,
cache      IN          BOOLEAN,
dur        IN         INTEGER);
```

表 11-16 DBMS_LOB.CREATETEMPORARY 接口参数说明

参数	说明
lob_loc	LOB描述符。
cache	仅用于语法上的兼容。
dur	仅用于语法上的兼容。

- DBMS_LOB.APPEND

  存储过程READ根据指定长度及起始位置偏移读取BLOB内容的一部分到BUFFER缓冲区。

  DBMS_LOB.APPEND函数原型为：

```
DBMS_LOB.APPEND (
dest_lob    IN OUT      BLOB,
src_lob     IN          BLOB);

DBMS_LOB.APPEND (
dest_lob    IN OUT      CLOB,
src_lob     IN          CLOB);
```

表 11-17 DBMS_LOB.APPEND 接口参数说明

参数	说明
dest_lob	要写入的LOB描述符。

 版权所有 © 华为云计算技术有限公司

参数	说明
src_lob	读取的LOB描述符。

示例

```
--获取字符串的长度
SELECT DBMS_LOB.GETLENGTH('12345678');

DECLARE
myraw  RAW(100);
amount INTEGER :=2;
buffer INTEGER :=1;
begin
DBMS_LOB.READ('123456789012345',amount,buffer,myraw);
dbms_output.put_line(myraw);
end;
/

CREATE TABLE blob_Table (t1 blob) DISTRIBUTE BY REPLICATION;
CREATE TABLE blob_Table_bak (t2 blob) DISTRIBUTE BY REPLICATION;
INSERT INTO blob_Table VALUES('abcdef');
INSERT INTO blob_Table_bak VALUES('22222');

DECLARE
str varchar2(100) := 'abcdef';
source raw(100);
dest blob;
copyto blob;
amount int;
PSV_SQL varchar2(100);
PSV_SQL1 varchar2(100);
a int :=1;
len int;
BEGIN
source := utl_raw.cast_to_raw(str);
amount := utl_raw.length(source);

PSV_SQL :='select * from blob_Table for update';
PSV_SQL1 := 'select * from blob_Table_bak for update';

EXECUTE IMMEDIATE PSV_SQL into dest;
EXECUTE IMMEDIATE PSV_SQL1 into copyto;

DBMS_LOB.WRITE(dest, amount, 1, source);
DBMS_LOB.WRITEAPPEND(dest, amount, source);

DBMS_LOB.ERASE(dest, a, 1);
DBMS_OUTPUT.PUT_LINE(a);
DBMS_LOB.COPY(copyto, dest, amount, 10, 1);
DBMS_LOB.CLOSE(dest);
RETURN;
END;
/

--删除表
DROP TABLE blob_Table;
DROP TABLE blob_Table_bak;
```

11.10.2 DBMS_RANDOM

接口介绍

高级功能包DBMS_RANDOM支持的所有接口请参见**表11-18**。

版权所有 © 华为云计算技术有限公司

表 11-18 DBMS_RANDOM 接口参数说明

接口名称	描述
DBMS_RANDOM.SEED	设置一个随机数的种子。
DBMS_RANDOM.VALUE	生成一个大小介于指定的low及high之间的随机数。

- DBMS_RANDOM.SEED

  存储过程SEED用于设置一个随机数的种子。DBMS_RANDOM.SEED函数原型为：

  ```
  DBMS_RANDOM.SEED (seed  IN  INTEGER);
  ```

  表 11-19 DBMS_RANDOM.SEED 接口参数说明

参数	描述
seed	用于产生一个随机数的种子。

- DBMS_RANDOM.VALUE

  存储过程VALUE生成一个大小介于指定的low及high之间的随机数。
  DBMS_RANDOM.VALUE函数原型为：

  ```
  DBMS_RANDOM.VALUE(
  low  IN  NUMBER,
  high IN  NUMBER)
  RETURN NUMBER;
  ```

  表 11-20 DBMS_RANDOM.VALUE 接口参数说明

参数	描述
low	指定随机数大小的下边界，生成的随机数大于或等于low。
high	指定随机数大小的上边界，生成的随机数小于high。

  说明

  实际上，只要求这里的参数类型是NUMERIC即可，对于左右边界的大小并没有要求。

示例

生成0到1之间的随机数：

```
SELECT DBMS_RANDOM.VALUE(0,1);
```

生成0到100之间的随机整数。随机整数大于等于low的指定值，小于high指定值。

```
SELECT TRUNC(DBMS_RANDOM.VALUE(0,100));
```

 版权所有 © 华为云计算技术有限公司

11.10.3 DBMS_OUTPUT

接口介绍

高级功能包DBMS_OUTPUT支持的所有接口请参见**表11-21**。

表 11-21 DBMS_OUTPUT

接口名称	描述
DBMS_OUTPUT.PUT_LINE	输出指定的文本，文本长度不能超过32767字节。
DBMS_OUTPUT.PUT	将指定的文本输出到指定文本的前面，不添加换行符，文本长度不能超过32767字节。
DBMS_OUTPUT.ENABLE	设置输出缓冲区的大小。若不指定，缓冲区最大只能容纳20000字节，缓冲区最小可设置为2000字节，若设置小于2000字节将按2000字节处理。

- DBMS_OUTPUT.PUT_LINE

存储过程PUT_LINE向消息缓冲区写入一行带有行结束符的文本。
DBMS_OUTPUT.PUT_LINE函数原型为：

```
DBMS_OUTPUT.PUT_LINE (
item IN VARCHAR2);
```

表 11-22 DBMS_OUTPUT.PUT_LINE 接口参数说明

参数	描述
item	写入消息缓冲区的文本。

- DBMS_OUTPUT.PUT

存储过程PUT将指定的文本输出到指定文本的前面，不添加换行符。
DBMS_OUTPUT.PUT函数原型为：

```
DBMS_OUTPUT.PUT (
item IN VARCHAR2);
```

表 11-23 DBMS_OUTPUT.PUT 接口参数说明

参数	描述
item	写入指定文本前的文本。

- DBMS_OUTPUT.ENABLE

版权所有 © 华为云计算技术有限公司

存储过程ENABLE设置输出缓冲区的大小，如果不指定的话缓冲区最大只能容纳20000字节。DBMS_OUTPUT.ENABLE函数原型为：

```
DBMS_OUTPUT.ENABLE (
buf IN INTEGER);
```

表 11-24 DBMS_OUTPUT.ENABLE 接口参数说明

参数	描述
buf	设置输出缓冲区的大小。

示例

```
BEGIN
    DBMS_OUTPUT.ENABLE(50);
    DBMS_OUTPUT.PUT ('hello, ');
    DBMS_OUTPUT.PUT_LINE('database!');--输出hello, database!
END;
/
```

11.10.4 UTL_RAW

接口介绍

高级功能包UTL_RAW支持的所有接口请参见**表11-25**。

表 11-25 UTL_RAW

接口名称	描述
UTL_RAW.CAST_FROM_BINARY_INTEGER	将INTEGER类型值转换为二进制表示形式（即RAW类型）。
UTL_RAW.CAST_TO_BINARY_INTEGER	将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。
UTL_RAW.LENGTH	获取RAW类型对象的长度。
UTL_RAW.CAST_TO_RAW	将VARCHAR2类型值转化为二进制表示形式（即RAW类型）。

须知

RAW类型的外部表现形式是十六进制，内部存储形式是二进制。例如一个RAW类型的数据11001011的表现形式为‘CB’，即在实际的类型转换中输入的是‘CB’。

- UTL_RAW.CAST_FROM_BINARY_INTEGER

  存储过程CAST_FROM_BINARY_INTEGER将INTEGER类型值转换为二进制表示形式（即RAW类型）。

  UTL_RAW.CAST_FROM_BINARY_INTEGER函数原型为：

 版权所有 © 华为云计算技术有限公司

```
UTL_RAW.CAST_FROM_BINARY_INTEGER (
n           IN  INTEGER,
endianess   IN  INTEGER)
RETURN RAW;
```

表 11-26 UTL_RAW.CAST_FROM_BINARY_INTEGER 接口参数说明

参数	描述
n	待转成RAW类型的整型数值。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.CAST_TO_BINARY_INTEGER

  存储过程CAST_TO_BINARY_INTEGER将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。

  UTL_RAW.CAST_TO_BINARY_INTEGER函数原型为：

  ```
  UTL_RAW.CAST_TO_BINARY_INTEGER (
  r           IN  RAW,
  endianess  IN  INTEGER)
  RETURN BINARY_INTEGER;
  ```

  表 11-27 UTL_RAW.CAST_TO_BINARY_INTEGER 接口参数说明

参数	描述
r	二进制表示形式的整型值（即RAW类型）。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.LENGTH

  存储过程LENGTH返回RAW类型对象的长度。

  UTL_RAW.LENGTH函数原型为：

  ```
  UTL_RAW.LENGTH(
  r      IN RAW)
  RETURN INTEGER;
  ```

  表 11-28 UTL_RAW.LENGTH 接口参数说明

参数	描述
r	RAW类型对象

- UTL_RAW.CAST_TO_RAW

  存储过程CAST_TO_RAW将VARCHAR2类型的对象转换成RAW类型。

  UTL_RAW.CAST_TO_RAW函数原型为：

  ```
  UTL_RAW.CAST_TO_RAW(
  c      IN VARCHAR2)
  RETURN RAW;
  ```

 版权所有 © 华为云计算技术有限公司

表 11-29 UTL_RAW.CAST_TO_RAW 接口参数说明

参数	描述
c	待转换的VARCHAR2类型对象

示例

在存储过程中操作RAW数据：

```
CREATE OR REPLACE PROCEDURE proc_raw
AS
str varchar2(100) := 'abcdef';
source raw(100);
amount integer;
BEGIN
source := utl_raw.cast_to_raw(str);--类型转换
amount := utl_raw.length(source);--获取长度
dbms_output.put_line(amount);
END;
/
```

调用存储过程：

```
CALL proc_raw();
```

11.10.5 DBMS_JOB

接口介绍

高级功能包DBMS_JOB支持的所有接口请参见**表11-30**。

表 11-30 DBMS_JOB

接口名称	描述
DBMS_JOB.SUBMIT	提交一个定时任务。作业号由系统自动生成。
DBMS_JOB.SUBMIT_NODE	提交一个定时任务。执行节点由用户指定，作业号由系统自动生成。
DBMS_JOB.ISUBMIT	提交一个定时任务。作业号由用户指定。
DBMS_JOB.REMOVE	通过作业号来删除定时任务。
DBMS_JOB.BROKEN	禁用或者启用定时任务。
DBMS_JOB.CHANGE	修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。
DBMS_JOB.WHAT	修改定时任务的任务内容属性。
DBMS_JOB.NEXT_DATE	修改定时任务的下次执行时间属性。

版权所有 © 华为云计算技术有限公司

接口名称	描述
DBMS_JOB.INTERVAL	修改定时任务的执行间隔属性。
DBMS_JOB.CHANGE_OWNER	修改定时任务的属主。
DBMS_JOB.CHANGE_NODE	修改定时任务的执行节点。

- DBMS_JOB.SUBMIT

  存储过程SUBMIT提交一个系统提供的定时任务。

  DBMS_JOB.SUBMIT函数原型为：

```
DBMS_JOB.SUBMIT(
what         IN   TEXT,
next_date    IN   TIMESTAMP DEFAULT sysdate,
job_interval IN   TEXT  DEFAULT 'null',
job          OUT  INTEGER);
```

  说明

  当创建一个定时任务（DBMS_JOB）时，系统默认将当前数据库和用户名与当前创建的定时任务（DBMS_JOB）绑定起来。该接口函数可以通过call或select调用，如果通过select调用，可以不填写出参。如果在存储过程中则需要用通过perform调用该接口函数。

  表 11-31 DBMS_JOB.SUBMIT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
what	text	IN	否	要执行的SQL语句。支持一个或多个‘DML’，‘匿名块’，‘调用存储过程的语句’或3种混合的场景。
next_date	timestamp	IN	否	下次作业运行时间。默认值为当前系统时间（sysdate）。如果是过去时间，在提交作业时表示立即执行。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式，可以是interval表达式，也可以是sysdate加上一个numeric值（例如：sysdate+1.0/24）。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。
job	integer	OUT	否	作业号。范围为1～32767。当使用select调用dbms.submit时，该参数可以省略。

  示例：

 版权所有 © 华为云计算技术有限公司

```
select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1');

select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1.0/24');

CALL DBMS_JOB.SUBMIT('INSERT INTO T_JOB  VALUES(1);  call pro_1(); call pro_2();',
add_months(to_date('201701','yyyymm'),1), 'date_trunc(''day'',SYSDATE) + 1 +(8*60+30.0)/
(24*60)' ,:jobid);
```

- DBMS_JOB.SUBMIT_NODE

  存储过程SUBMIT提交一个系统提供的定时任务。执行节点由用户指定。该接口仅8.3.0及以上集群版本支持。

  DBMS_JOB.SUBMIT_NODE函数原型为：

```
DBMS_JOB.SUBMIT_NODE(
what         IN   TEXT,
next_date    IN   TIMESTAMP DEFAULT sysdate,
job_interval IN   TEXT DEFAULT 'null',
job_node     IN   TEXT DEFAULT NULL,
job          OUT  INTEGER);
```

表 11-32 DBMS_JOB.SUBMIT_NODE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
what	text	IN	否	要执行的SQL语句。支持一个或多个DML、匿名块、调用存储过程的语句或3种混合的场景。
next_date	timestamp	IN	否	下次作业运行时间。默认值为当前系统时间（sysdate）。如果是过去时间，在提交作业时表示立即执行。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式，可以是interval表达式，也可以是sysdate加上一个numeric值（例如：sysdate+1.0/24）。如果为空值或字符串"null"则表示只执行一次，执行后JOB状态STATUS变成'd'则不再执行。
node	text	IN	是	作业执行节点名称。
job	integer	OUT	否	作业号。范围为1～32767。当使用select调用dbms.submit时，该参数可以省略。

  示例：

```
select DBMS_JOB.SUBMIT_NODE('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate
+1','coordinator1');

select DBMS_JOB.SUBMIT_NODE('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1.0/24');

CALL DBMS_JOB.SUBMIT('INSERT INTO T_JOB  VALUES(1); call pro_1(); call pro_2();',
add_months(to_date('201701','yyyymm'),1), 'date_trunc(''day'',SYSDATE) + 1 +(8*60+30.0)/(24*60)',
'coordinator1', :jobid);
```

- DBMS_JOB.ISUBMIT

 版权所有 © 华为云计算技术有限公司

ISUBMIT与SUBMIT语法功能相同，但其第一个参数是入参，即指定的作业号，SUBMIT最后一个参数是出参，表示系统自动生成的作业号。

示例：

```
CALL dbms_job.isubmit(101, 'insert_msg_statistic1;', sysdate, 'sysdate+3.0/24');
```

须知

DWS的pgstats持久化功能将内存中的统计信息写入pg_stat_object系统表，如果新安装的9.1.0.100及以上集群版本，会占用1为job_id。如果为低版本升级到9.1.0.100及以上集群版本，并且之前pg_job中有任务，那么会找到一个未被占用的job_id作为持久化任务的ID。所以在使用dbms_job.isubmit接口时需注意，不能和已经存在的pgstats持久化任务的id重复，否则会导致任务注册失败。

- DBMS_JOB.REMOVE

  存储过程REMOVE删除指定的定时任务。

  DBMS_JOB.REMOVE函数原型为：

  ```
  REMOVE(job IN INTEGER);
  ```

  表 11-33 DBMS_JOB.REMOVE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。

  示例：

  ```
  CALL dbms_job.remove(101);
  ```

- DBMS_JOB.BROKEN

  存储过程BROKEN禁用或者启用定时任务。

  DBMS_JOB.BROKEN函数原型为：

  ```
  DBMS_JOB.BROKEN(
  job          IN   INTEGER,
  broken       IN   BOOLEAN,
  next_date    IN   TIMESTAMP  DEFAULT  sysdate);
  ```

  表 11-34 DBMS_JOB.BROKEN 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。

版权所有 © 华为云计算技术有限公司

参数	类型	入参/出参	是否可以为空	描述
broken	boolean	IN	否	状态标志位，true代表禁用，false代表启用。具体true或false值更新当前job；如果为空值，则不改变原有job的状态。
next_date	timestamp	IN	是	下次运行时间，默认为当前系统时间。如果参数broken状态为true，则更新该参数为'4000-1-1'；如果参数broken状态为false，且如果参数next_date不为空值，则更新指定job的next_date值，如果next_date为空值，则不更新next_date值。该参数可以省略，为默认值。

示例：

```
CALL dbms_job.broken(101,true);
CALL dbms_job.broken(101,false,sysdate);
```

- DBMS_JOB.CHANGE

  存储过程CHANGE修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。

  DBMS_JOB.CHANGE函数原型为：

  ```
  DBMS_JOB.CHANGE(
  job          IN   INTEGER,
  what         IN   TEXT,
  next_date    IN   TIMESTAMP,
  interval     IN   TEXT);
  ```

  表 11-35 DBMS_JOB.CHANGE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	是	执行的存储过程名或者sql语句块。如果该参数为空值，则不更新指定job的what值，否则更新指定job的what值。
next_date	timestamp	IN	是	下次运行时间。如果该参数为空值，则不更新指定job的next_date值，否则更新指定job的next_date值。

版权所有 © 华为云计算技术有限公司

参数	类型	入参/出参	是否可以为空	描述
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果该参数为空值，则不更新指定job的interval值；如果该参数不为空值，会校验interval是否为有效的时间类型或interval类型，则更新指定job的interval值。如果为字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。

示例：

```
CALL dbms_job.change(101, 'call userproc();', sysdate, 'sysdate + 1.0/1440');
CALL dbms_job.change(101, 'insert into tbl_a values(sysdate);', sysdate, 'sysdate + 1.0/1440');
```

- DBMS_JOB.WHAT

  存储过程WHAT修改定时任务的任务内容属性。

  DBMS_JOB.WHAT函数原型为：

  ```
  DBMS_JOB.WHAT(
  job             IN     INTEGER,
  what            IN     TEXT);
  ```

  表 11-36 DBMS_JOB.WHAT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	否	执行的存储过程调用或者sql语句块。

  说明

  - 当what参数是一个或多个可以执行成功的sql语句/程序块/调用存储过程时，该接口函数才能被执行成功，否则会执行失败。
  - 若what参数为一个简单的insert、update等语句，需要在表前加模式名。

  示例：

  ```
  CALL dbms_job.what(101, 'call userproc();');
  CALL dbms_job.what(101, 'insert into tbl_a values(sysdate);');
  ```

- DBMS_JOB.NEXT_DATE

  存储过程NEXT_DATE修改定时任务的下次执行时间属性。

  DBMS_JOB.NEXT_DATE函数原型为：

  ```
  DBMS_JOB.NEXT_DATE(
  job          IN     INTEGER,
  next_date    IN     TIMESTAMP);
  ```


版权所有 © 华为云计算技术有限公司

表 11-37 DBMS_JOB.NEXT_DATE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
next_date	timestamp	IN	否	下次运行时间。

说明

如果输入的next_date的值小于当前日期值，该job会立即执行一次。

示例：

```
CALL dbms_job.next_date(101,sysdate);
```

- DBMS_JOB.INTERVAL

  存储过程INTERVAL修改定时任务的执行间隔属性。

  DBMS_JOB.INTERVAL函数原型为：

  ```
  DBMS_JOB.INTERVAL(
  job             IN   INTEGER,
  interval        IN   TEXT);
  ```

表 11-38 DBMS_JOB.INTERVAL 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。interval是否为有效的时间类型或interval类型。

示例：

```
CALL dbms_job.interval(101, 'sysdate + 1.0/1440');
```

说明

对于指定job正在运行状态（即job_status为'r'）时，不允许通过remove、change、next_date、what、interval等接口删除或修改job的参数信息。

- DBMS_JOB.CHANGE_OWNER

  存储过程CHANGE_OWNER修改定时任务的属主。

  DBMS_JOB.CHANGE_OWNER函数原型为：

  ```
  DBMS_JOB.CHANGE_OWNER(
  job             IN     INTEGER,
  new_owner       IN     NAME);
  ```


版权所有 © 华为云计算技术有限公司

表 11-39 DBMS_JOB.CHANGE_OWNER 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
new_owner	name	IN	否	新的用户名。

示例：

```
CALL dbms_job.change_owner(101, 'alice');
```

- DBMS_JOB.CHANGE_NODE

  存储过程CHANGE_NODE修改定时任务的执行节点。该接口仅8.3.0及以上集群版本支持。

  DBMS_JOB.CHANGE_NODE函数原型为：

```
DBMS_JOB.CHANGE_NODE(
job             IN     INTEGER,
new_node        IN     text);
```

表 11-40 DBMS_JOB.CHANGE_OWNER 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
new_node	text	IN	否	新的执行节点。

示例：

```
CALL dbms_job.change_node(101, 'coordinator2');
```

约束说明

1. 创建一个新job后，该job从属于当前coordinator（即：该job仅在当前coordinator上调度和执行），其他coordinator不会调度和执行该job。所有coordinator都可以查看、修改、删除其他CN创建的job。
2. job只能通过dbms_job高级包提供的接口进行创建、更新、删除操作，因为高级包的接口中会考虑所有CN间job信息的同步和pg_jobs表主键的关联操作，如果通过DML语句对pg_jobs表进行增删改，会导致job信息在CN间不一致和系统表无法关联变更的混乱问题，会严重影响job内部的管理。
3. 由于用户创建的每个任务和CN绑定，若不开启CN故障自动迁移功能，当任务运行过程中，该CN故障，则该任务的状态无法实时刷新。如果在任务未执行时CN故障，则该CN上的任务都得不到正常的调度和执行。建议开启CN故障自动迁移功能，故障CN上的作业会迁移至其他CN继续调度。
4. job在定时执行过程中，需要在当前job所属的CN上实时更新该job的运行状态、最近执行开始时间、最近执行结束时间、下次开始时间、失败次数（如果job执行失败）等相关参数信息到pg_jobs系统表中，并同步到其他CN，保证job信息的一致

版权所有 © 华为云计算技术有限公司

性。如果其他CN存在节点故障，那么job所属CN会同步超时重发的处理，导致job执行时间变长，但CN间同步超时失败后，原CN上pg_jobs表中job的相关信息仍然能正常更新，且job能正常执行成功。当故障CN恢复正常后，可能出现该CN上pg_jobs表中当前job的执行时间、运行状态等参数与原CN上不一致的情况，需要原CN上再次执行该job后才能保证job信息的同步。

5. 对于并发同时有多个job到达执行时间的场景，由于会为每个job创建一个线程来执行job，由于系统内部启动每个线程的时间会有延迟，因此会导致同时并发执行的job的开始时间有延迟，每个job的延迟时间在0.1ms左右。
6. job中待执行SQL语句有长度限制，最长为8K。

11.10.6 DBMS_SQL

接口介绍

高级功能包DBMS_SQL支持的接口请参见**表1 DBMS_SQL**。

表 11-41 DBMS_SQL

接口名称	描述
DBMS_SQL.OPEN_CURSOR	打开一个游标。
DBMS_SQL.CLOSE_CURSOR	关闭一个已打开的游标。
DBMS_SQL.PARSE	向游标传递一组SQL语句，目前只支持SELECT。
DBMS_SQL.EXECUTE	在游标上执行一组动态定义操作。
DBMS_SQL.FETCH_ROWS	读取游标一行数据。
DBMS_SQL.DEFINE_COLUMN	动态定义一个列。
DBMS_SQL.DEFINE_COLUMN_CHAR	动态定义一个char类型的列。
DBMS_SQL.DEFINE_COLUMN_INT	动态定义一个int类型的列。
DBMS_SQL.DEFINE_COLUMN_LONG	动态定义一个long类型的列。
DBMS_SQL.DEFINE_COLUMN_RAW	动态定义一个raw类型的列。
DBMS_SQL.DEFINE_COLUMN_TEXT	动态定义一个text类型的列。
DBMS_SQL.DEFINE_COLUMN_UNKNOWN	动态定义一个未知列（类型不识别时入此接口）。
DBMS_SQL.COLUMN_VALUE	读取一个已动态定义的列值。
DBMS_SQL.COLUMN_VALUE_CHAR	读取一个已动态定义的列值（指定char类型）。
DBMS_SQL.COLUMN_VALUE_INT	读取一个已动态定义的列值（指定int类型）。
DBMS_SQL.COLUMN_VALUE_LONG	读取一个已动态定义的列值（指定long类型）。

版权所有 © 华为云计算技术有限公司

接口名称	描述
DBMS_SQL.COLUMN_VALUE_RAW	读取一个已动态定义的列值（指定raw类型）。
DBMS_SQL.COLUMN_VALUE_TEXT	读取一个已动态定义的列值（指定text类型）。
DBMS_SQL.COLUMN_VALUE_UNKNOWN	读取一个已动态定义的列值（类型不识别时入此接口）。
DBMS_SQL.IS_OPEN	检查游标是否已打开。

说明

- 建议使用dbms_sql.define_column及dbms_sql.column_value定义参数列。
- 当结果集大于work_mem设定值时会触发结果集临时下盘，但最大阈值不超过512MB。

- DBMS_SQL.OPEN_CURSOR

  该函数用来打开一个游标，是后续dbms_sql各项操作的前提。该函数不传入任何参数，内部自动递增生成游标ID，并作为返回值返回给integer定义的变量。

  DBMS_SQL.OPEN_CURSOR函数原型为：

  ```
  DBMS_SQL.OPEN_CURSOR (
  )
  RETURN INTEGER;
  ```

- DBMS_SQL.CLOSE_CURSOR

  该函数用来关闭一个游标，是dbms_sql各项操作的结束。如果在存储过程结束时没有调用该函数，则该游标占用的内存仍然会保存，因此关闭游标非常重要。由于异常情况的发生会中途退出存储过程，导致游标未能关闭，因此建议存储过程中有异常处理，将该接口包含在内。

  DBMS_SQL.CLOSE_CURSOR函数原型为：

  ```
  DBMS_SQL.CLOSE_CURSOR (
  cursorid     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 11-42 DBMS_SQL.CLOSE_CURSOR 接口说明

参数名称	描述
cursorid	打算关闭的游标ID号

- DBMS_SQL.PARSE

  该函数用来解析给定游标的查询语句，被传入的查询语句会立即执行。目前仅支持SELECT查询语句的解析，且语句参数仅可通过text类型传递，长度不大于1G。

  DBMS_SQL.PARSE函数的原型为：

  ```
  DBMS_SQL.PARSE (
  cursorid     IN INTEGER,
  query_string IN TEXT,
  label        IN INTEGER
  )
  RETURN BOOLEAN;
  ```


版权所有 © 华为云计算技术有限公司

表 11-43 DBMS_SQL.PARSE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID
query_string	执行的查询语句
language_flag	版本语言号，目前只支持1

- DBMS_SQL.EXECUTE

  该函数用来执行一个给定的游标。该函数接收一个游标ID，运行后获得的数据用于后续操作。目前仅支持SELECT查询语句的执行。

  DBMS_SQL.EXECUTE函数的原型为：

```
DBMS_SQL.EXECUTE(
cursorid     IN INTEGER,
)
RETURN INTEGER;
```

表 11-44 DBMS_SQL.EXECUTE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID

- DBMS_SQL.FETCH_ROWS

  该函数返回符合查询条件的数据行数，每一次运行该接口都会获取到新的行数的集合，直到数据读取完毕获取不到新行为止。

  DBMS_SQL.FETCH_ROWS函数的原型为：

```
DBMS_SQL.FETCH_ROWS(
cursorid     IN INTEGER,
)
RETURN INTEGER;
```

表 11-45 DBMS_SQL.FETCH_ROWS 接口说明

参数名称	描述
cursorid	执行的游标ID。

- DBMS_SQL.DEFINE_COLUMN

  该函数用来定义从给定游标返回的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN(
cursorid     IN INTEGER,
position     IN INTEGER,
column_ref   IN ANYELEMENT,
column_size      IN INTEGER default 1024
)
RETURN INTEGER;
```

 版权所有 © 华为云计算技术有限公司

表 11-46 DBMS_SQL.DEFINE_COLUMN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_ref	任意类型的变量，可根据变量类型选择适当的接口动态定义列
column_size	定义的列的长度

- DBMS_SQL.DEFINE_COLUMN_CHAR

  该函数用来定义从给定游标返回的CHAR类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_CHAR函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT,
column_size     IN INTEGER
)
RETURN INTEGER;
```

表 11-47 DBMS_SQL.DEFINE_COLUMN_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	需要定义的某类型的参数变量
column_size	动态定义列长度

- DBMS_SQL.DEFINE_COLUMN_INT

  该函数用来定义从给定游标返回的INT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_INT函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_INT(
cursorid     IN INTEGER,
position     IN INTEGER
)
RETURN INTEGER;
```

表 11-48 DBMS_SQL.DEFINE_COLUMN_INT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

版权所有 © 华为云计算技术有限公司

- DBMS_SQL.DEFINE_COLUMN_LONG

  该函数用来定义从给定游标返回的长列类型（非数据类型long）的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。长列的大小限制为1G。

  DBMS_SQL.DEFINE_COLUMN_LONG函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_LONG(
cursorid     IN INTEGER,
position     IN INTEGER
)
RETURN INTEGER;
```

  表 11-49 DBMS_SQL.DEFINE_COLUMN_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.DEFINE_COLUMN_RAW

  该函数用来定义从给定游标返回的RAW类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_RAW函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_RAW(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN BYTEA,
column_size     IN INTEGER
)
RETURN INTEGER;
```

  表 11-50 DBMS_SQL.DEFINE_COLUMN_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	RAW类型的参数变量
column_size	列的长度

- DBMS_SQL.DEFINE_COLUMN_TEXT

  该函数用来定义从给定游标返回的TEXT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_TEXT函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
max_size     IN INTEGER
)
RETURN INTEGER;
```

 版权所有 © 华为云计算技术有限公司

表 11-51 DBMS_SQL.DEFINE_COLUMN_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
max_size	定义的TEXT类型的最大长度

- DBMS_SQL.DEFINE_COLUMN_UNKNOWN

  该函数用来处理从给定游标返回的未知数据类型的列，该接口仅用于类型不识别时的报错退出。

  DBMS_SQL.DEFINE_COLUMN_UNKNOWN函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT
)
RETURN INTEGER;
```

表 11-52 DBMS_SQL.DEFINE_COLUMN_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	动态定义的参数

- DBMS_SQL.COLUMN_VALUE

  该函数用来返回给定游标给定位置的游标元素值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE函数的原型为：

```
DBMS_SQL.COLUMN_VALUE(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT ANYELEMENT
)
RETURN ANYELEMENT;
```

表 11-53 DBMS_SQL.COLUMN_VALUE 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	定义的列的返回值

- DBMS_SQL.COLUMN_VALUE_CHAR

  该函数用来返回给定游标给定位置的游标CHAR类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

版权所有 © 华为云计算技术有限公司

DBMS_SQL.COLUMN_VALUE_CHAR函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_CHAR(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT CHARACTER,
err_num               INOUT NUMERIC default 0,
actual_length           INOUT INTEGER default 1024
)
RETURN RECORD;
```

表 11-54 DBMS_SQL.COLUMN_VALUE_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	返回值
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度

- DBMS_SQL.COLUMN_VALUE_INT

  该函数用来返回给定游标给定位置的游标INT类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。DBMS_SQL.COLUMN_VALUE_INT函数的原型为：

  ```
  DBMS_SQL.COLUMN_VALUE_INT(
  cursorid              IN    INTEGER,
  position              IN    INTEGER
  )
  RETURN INTEGER;
  ```

  表 11-55 DBMS_SQL.COLUMN_VALUE_INT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.COLUMN_VALUE_LONG

  该函数用来返回给定游标给定位置的游标长列（非long/bigint整型）类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_LONG函数的原型为：

  ```
  DBMS_SQL.COLUMN_VALUE_LONG(
  cursorid              IN    INTEGER,
  position              IN    INTEGER,
  length                IN    INTEGER,
  off_set               IN    INTEGER,
  column_value            INOUT TEXT,
  actual_length           INOUT INTEGER default 1024
  )
  RETURN RECORD;
  ```

 版权所有 © 华为云计算技术有限公司

表 11-56 DBMS_SQL.COLUMN_VALUE_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
length	返回值的长度
off_set	返回值的起始位置
column_value	返回值
actual_length	实际返回值的长度

- DBMS_SQL.COLUMN_VALUE_RAW

  该函数用来返回给定游标给定位置的游标RAW类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_RAW函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_RAW(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT BYTEA,
err_num               INOUT NUMERIC default 0,
actual_length          INOUT INTEGER default 1024
)
RETURN RECORD;
```

表 11-57 DBMS_SQL.COLUMN_VALUE_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	返回的列值
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度，不能长于此值，否则截断

- DBMS_SQL.COLUMN_VALUE_TEXT

  该函数用来返回给定游标给定位置的游标TEXT类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_TEXT函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_TEXT(
cursorid              IN    INTEGER,
position              IN    INTEGER
)
RETURN TEXT;
```


版权所有 © 华为云计算技术有限公司

表 11-58 DBMS_SQL.COLUMN_VALUE_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.COLUMN_VALUE_UNKNOWN

  该函数用来返回给定游标给定位置的游标未知类型的值，该接口为类型不支持时的报错处理接口。

  DBMS_SQL.COLUMN_VALUE_UNKNOWN函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_UNKNOWN(
cursorid              IN    INTEGER,
position              IN    INTEGER,
COLUMN_TYPE             IN    TEXT
)
RETURN TEXT;
```

表 11-59 DBMS_SQL.COLUMN_VALUE_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_type	返回的参数类型

- DBMS_SQL.IS_OPEN

该函数用来返回游标的当前状态：打开、解析、执行、定义。取值是为TRUE，关闭后为FALSE，未知时报错，其余默认为关闭。

DBMS_SQL.IS_OPEN函数的原型为：

```
DBMS_SQL.IS_OPEN(
cursorid              IN    INTEGER
)
RETURN BOOLEAN;
```

表 11-60 DBMS_SQL.IS_OPEN 接口说明

参数名称	描述
cursorid	被查询的游标ID

示例

```
--在存储过程中操作raw数据
create or replace procedure pro_dbms_sql_all_02(in_raw raw,v_in int,v_offset int)
as
cursorid int;
v_id int;
v_info bytea :=1;
query varchar(2000);
execute_ret int;
```

 版权所有 © 华为云计算技术有限公司

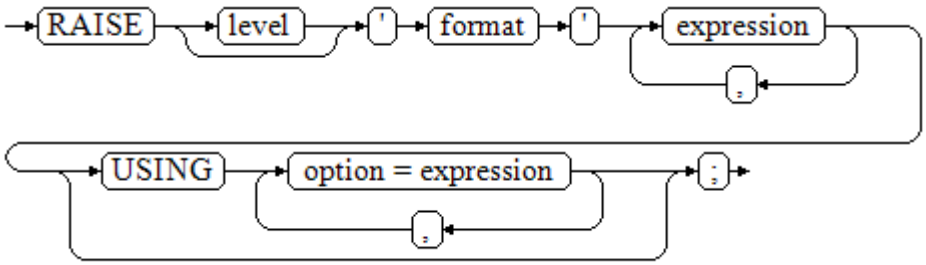
```
define_column_ret_raw bytea :='1';
define_column_ret int;
begin
drop table if exists pro_dbms_sql_all_tb1_02 ;
create table pro_dbms_sql_all_tb1_02(a int ,b blob);
insert into pro_dbms_sql_all_tb1_02 values(1,HEXTORAW('DEADBEEE'));
insert into pro_dbms_sql_all_tb1_02 values(2,in_raw);
query := 'select * from pro_dbms_sql_all_tb1_02 order by 1';
--打开游标
cursorid := dbms_sql.open_cursor();
--编译游标
dbms_sql.parse(cursorid, query, 1);
--定义列
define_column_ret:= dbms_sql.define_column(cursorid,1,v_id);
define_column_ret_raw:= dbms_sql.define_column_raw(cursorid,2,v_info,10);
--执行
execute_ret := dbms_sql.execute(cursorid);
loop
exit when (dbms_sql.fetch_rows(cursorid) <= 0);
--获取值
dbms_sql.column_value(cursorid,1,v_id);
dbms_sql.column_value_raw(cursorid,2,v_info,v_in,v_offset);
--输出结果
dbms_output.put_line('id:'|| v_id || ' info:' || v_info);
end loop;
--关闭游标
dbms_sql.close_cursor(cursorid);
end;
/
--调用存储过程
call pro_dbms_sql_all_02(HEXTORAW('DEADBEEF'),0,1);

--删除存储过程
DROP PROCEDURE pro_dbms_sql_all_02;
```

11.11 DWS 存储过程调试

语法

RAISE有以下五种语法格式：

图 11-34 raise_format::=



版权所有 © 华为云计算技术有限公司

图 11-35 raise_condition::=

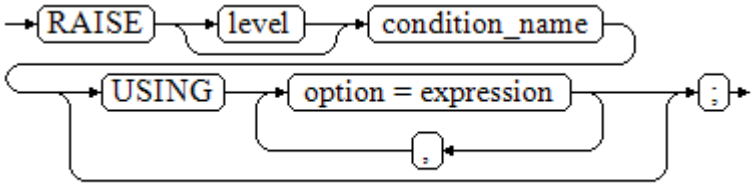


图 11-36 raise_sqlstate::=

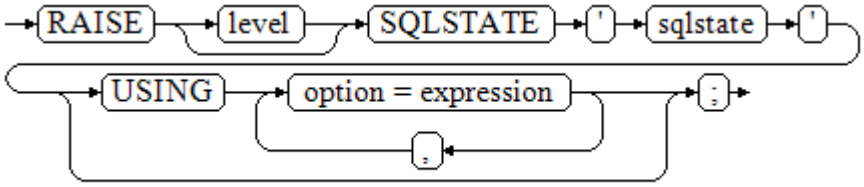


图 11-37 raise_option::=

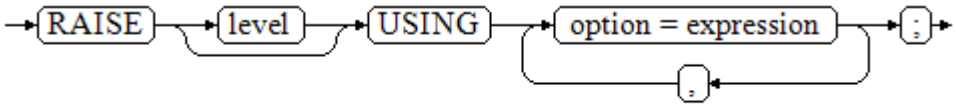
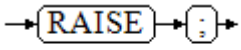


图 11-38 raise::=



参数说明：

- level选项用于指定错误级别，有DEBUG，LOG，INFO，NOTICE，WARNING以及EXCEPTION（默认值）。EXCEPTION上报一个正常终止当前事务的异常，其他的仅产生不同异常级别的信息。特殊级别的错误信息是否报告到客户端、写到服务器日志由**log_min_messages**和**client_min_messages**这两个配置参数控制。
- format：格式字符串，指定要报告的错误消息文本。格式字符串后可跟表达式，用于向消息文本中插入。在格式字符串中，%由format后面跟着的参数的值替换，%%用于打印出%。例如：

  ```
  --v_job_id 将替换字符串中的 %:
  RAISE NOTICE 'Calling cs_create_job(%)',v_job_id;
  ```

- option = expression：向错误报告中添加另外的信息。关键字option可以是MESSAGE、DETAIL、HINT以及ERRCODE，并且每一个expression可以是任意的字符串。
  - MESSAGE，指定错误消息文本，这个选项不能用于在USING前包含一个格式字符串的RAISE语句中。
  - DETAIL，说明错误的详细信息。
  - HINT，用于打印出提示信息。

 版权所有 © 华为云计算技术有限公司

  - ERRCODE，向报告中指定错误码（SQLSTATE）。可以使用条件名称或者直接用五位字符的SQLSTATE错误码。
- condition_name：错误码对应的条件名。
- sqlstate：错误码。

如果在RAISE EXCEPTION命令中既没有指定条件名也没有指定SQLSTATE，默认用RAISE EXCEPTION (P0001)。如果没有指定消息文本，默认用条件名或者SQLSTATE作为消息文本。

须知

当由SQLSTATE指定了错误码，则不局限于已定义的错误码，可以选择任意包含五个数字或者大写的ASCII字母的错误码，而不是00000。建议避免使用以三个0结尾的错误码，因为这种错误码是类别码，会被整个种类捕获。

说明

图11-38所示的语法不接任何参数。这种形式仅用于一个BEGIN块中的EXCEPTION语句，它使得错误重新被处理。

示例

终止事务时，给出错误和提示信息：

```
CREATE OR REPLACE PROCEDURE proc_raise1(user_id in integer)
AS
BEGIN
RAISE EXCEPTION 'Noexistence ID --> %',user_id USING HINT = 'Please check your user ID';
END;
/

call proc_raise1(300011);

--执行结果
ERROR:  Noexistence ID --> 300011
HINT:  Please check your user ID
```

两种设置SQLSTATE的方式：

```
CREATE OR REPLACE PROCEDURE proc_raise2(user_id in integer)
AS
BEGIN
RAISE 'Duplicate user ID: %',user_id USING ERRCODE = 'unique_violation';
END;
/

\set VERBOSITY verbose
call proc_raise2(300011);

--执行结果
ERROR:  Duplicate user ID: 300011
SQLSTATE: 23505
LOCATION:  exec_stmt_raise, pl_exec.cpp:3482
```

如果主要的参数是条件名或者是SQLSTATE，可以使用：

RAISE division_by_zero;

RAISE SQLSTATE '22012';

例如：

 版权所有 © 华为云计算技术有限公司

```
CREATE OR REPLACE PROCEDURE division(div in integer, dividend in integer)
AS
DECLARE
res int;
    BEGIN
    IF dividend=0 THEN
        RAISE division_by_zero;
        RETURN;
    ELSE
        res := div/dividend;
        RAISE INFO 'division result: %', res;
        RETURN;
    END IF;
    END;
/
call division(3,0);

--执行结果
ERROR:  division_by_zero
```

或者另一种方式：

```
RAISE unique_violation USING MESSAGE = 'Duplicate user ID: ' || user_id;
```


版权所有 © 华为云计算技术有限公司

12 使用 PostGIS Extension

12.1 PostGIS 概述

说明

- PostGIS Extension依赖的第三方软件需要用户进行单独安装，用户如需使用PostGIS功能，请提交工单或联系技术支持人员提交申请。
- 如果用户在使用中出现“ERROR：EXTENSION is not yet supported.”这种报错，则表示没有安装PostGIS软件包，请联系技术支持进行获取。

DWS提供PostGIS Extension（版本为PostGIS-2.4.2和PostGIS-3.2.2）。PostGIS Extension是PostgreSQL的空间数据库扩展，提供如下空间信息服务功能：空间对象、空间索引、空间操作函数和空间操作符。PostGIS Extension完全遵循OpenGIS规范。

DWS中PostGIS Extension依赖第三方开源软件如下。

- PostGIS 2.4.2依赖第三方开源软件：
  - Geos 3.6.2
  - Proj 4.9.2
  - Json 0.12.1
  - Libxml2 2.7.1
  - Gdal 1.11.0
- PostGIS 3.2.2依赖第三方开源软件：
  - Geos-3.11.0
  - Proj-6.0.0
  - Json 0.12.1
  - Libxml2 2.7.1
  - Sqlite3
  - protobuf-c 1.4.1
  - protobuf 3.6.1

 版权所有 © 华为云计算技术有限公司

12.2 PostGIS 使用

> **说明**
>
> PostGIS Extension依赖的第三方软件需要用户进行单独安装，用户如需使用PostGIS功能，请提交工单或联系技术支持人员提交申请。
>
> 如果用户在使用中出现“ERROR：EXTENSION is not yet supported.”这种报错，则表示没有安装PostGIS软件包，请联系技术支持进行获取。

创建 Extension

创建PostGIS Extension可直接使用CREATE EXTENSION命令进行创建：

```
CREATE EXTENSION postgis;
```

使用 Extension

PostGIS Extension函数调用格式为：

```
SELECT GisFunction (Param1, Param2,.....);
```

其中GisFunction为函数名，Param1、Param2等为函数参数名。下列SQL语句展示PostGIS的简单使用，对于各函数的具体使用，请参考**《 PostGIS-2.4.2用户手册 》**。

示例1：几何表的创建。

```
CREATE TABLE cities ( id integer, city_name varchar(50) );
SELECT AddGeometryColumn('cities', 'position', 4326, 'POINT', 2);
```

示例2：几何数据的插入。

```
INSERT INTO cities (id, position, city_name) VALUES (1,ST_GeomFromText('POINT(-9.5 23)',4326),'CityA');
INSERT INTO cities (id, position, city_name) VALUES (2,ST_GeomFromText('POINT(-10.6 40.3)',4326),'CityB');
INSERT INTO cities (id, position, city_name) VALUES (3,ST_GeomFromText('POINT(20.8 30.3)',4326), 'CityC');
```

示例3：计算三个城市间任意两个城市距离。

```
SELECT p1.city_name,p2.city_name,ST_Distance(p1.position,p2.position) FROM cities AS p1, cities AS p2
WHERE p1.id > p2.id;
```

删除 Extension

在DWS中删除PostGIS Extension的方法如下所示：

```
DROP EXTENSION postgis [CASCADE];
```

> **说明**
>
> 如果Extension被其它对象依赖（如创建的几何表），需要加入CASCADE（级联）关键字，删除所有依赖对象。

12.3 PostGIS 支持和限制

支持数据类型

DWS的PostGIS Extension支持如下数据类型:

版权所有 © 华为云计算技术有限公司

- box2d
- box3d
- geometry_dump
- geometry
- geography
- raster

说明

创建Postgis和使用Postgis非同一用户时，请设置如下GUC参数：

```
SET behavior_compat_options = 'bind_procedure_searchpath';
```

支持的操作符和函数列表

说明

PostGIS笛卡尔积ST_Intersects函数拥有缓存机制，外表的空间数据结构缓存命中率较高，在内表和外表的宽度有较大差异时，缓存宽表的数据可以避免多次加载大对象至缓存，对于性能的优化尤为明显，实际使用时可以通过**Join顺序的Hint**指定宽表作为外表使计划符合该场景。

表 12-1 PostGIS2.4.2 支持的操作符和函数列表

函数分类	包含函数
Management Functions	AddGeometryColumn、DropGeometryColumn、DropGeometryTable、PostGIS_Full_Version、PostGIS_GEOS_Version、PostGIS_Liblwgeom_Version、PostGIS_Lib_Build_Date、PostGIS_Lib_Version、PostGIS_PROJ_Version、PostGIS_Scripts_Build_Date、PostGIS_Scripts_Installed、PostGIS_Version、PostGIS_LibXML_Version、PostGIS_Scripts_Released、Populate_Geometry_Columns 、UpdateGeometrySRID
Geometry Constructors	ST_BdPolyFromText 、ST_BdMPolyFromText 、ST_Box2dFromGeoHash、ST_GeogFromText、ST_GeographyFromText、ST_GeogFromWKB、ST_GeomCollFromText、ST_GeomFromEWKB、ST_GeomFromEWKT、ST_GeometryFromText、ST_GeomFromGeoHash、ST_GeomFromGML、ST_GeomFromGeoJSON、ST_GeomFromKML、ST_GMLToSQL、ST_GeomFromText 、ST_GeomFromWKB、ST_LineFromMultiPoint、ST_LineFromText、ST_LineFromWKB、ST_LinestringFromWKB、ST_MakeBox2D、ST_3DMakeBox、ST_MakeEnvelope、ST_MakePolygon、ST_MakePoint、ST_MakePointM、ST_MLineFromText、ST_MPointFromText、ST_MPolyFromText、ST_Point、ST_PointFromGeoHash、ST_PointFromText、ST_PointFromWKB、ST_Polygon、ST_PolygonFromText、ST_WKBToSQL、ST_WKTToSQL

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Accessors	GeometryType、ST_Boundary、ST_CoordDim、ST_Dimension、ST_EndPoint、ST_Envelope、ST_ExteriorRing、ST_GeometryN、ST_GeometryType、ST_InteriorRingN、ST_IsClosed、ST_IsCollection、ST_IsEmpty、ST_IsRing、ST_IsSimple、ST_IsValid、ST_IsValidReason、ST_IsValidDetail、ST_M、ST_NDims、ST_NPoints、ST_NRings、ST_NumGeometries、ST_NumInteriorRings、ST_NumInteriorRing、ST_NumPatches、ST_NumPoints、ST_PatchN、ST_PointN、ST_SRID、ST_StartPoint、ST_Summary、ST_X、ST_XMax、ST_XMin、ST_Y、ST_YMax、ST_YMin、ST_Z、ST_ZMax、ST_Zmflag、ST_ZMin
Geometry Editors	ST_AddPoint、ST_Affine、ST_Force2D、ST_Force3D、ST_Force3DZ、ST_Force3DM、ST_Force4D、ST_ForceCollection、ST_ForceSFS、ST_ForceRHR、ST_LineMerge、ST_CollectionExtract、ST_CollectionHomogenize、ST_Multi、ST_RemovePoint、ST_Reverse、ST_Rotate、ST_RotateX、ST_RotateY、ST_RotateZ、ST_Scale、ST_Segmentize、ST_SetPoint、ST_SetSRID、ST_SnapToGrid、ST_Snap、ST_Transform、ST_Translate、ST_TransScale
Geometry Outputs	ST_AsBinary、ST_AsEWKB、ST_AsEWKT、ST_AsGeoJSON、ST_AsGML、ST_AsHEXEWKB、ST_AsKML、ST_AsLatLonText 、ST_AsSVG、ST_AsText、ST_AsX3D、ST_GeoHash
Operators	&&、&&&、&<、&<|、&>、<<、<<|、=、>>、@ 、|&> 、|>>、~、~=、<->、<#>
Spatial Relationships and Measurements	ST_3DClosestPoint、ST_3DDistance、ST_3DDWithin、ST_3DDFullyWithin、ST_3DIntersects、ST_3DLongestLine、ST_3DMaxDistance、ST_3DShortestLine、ST_Area、ST_Azimuth、ST_Centroid、ST_ClosestPoint、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_LineCrossingDirection、ST_Disjoint、ST_Distance、ST_HausdorffDistance、ST_MaxDistance、ST_DistanceSphere、ST_DistanceSpheroid、ST_DFullyWithin、ST_DWithin、ST_Equals、ST_HasArc、ST_Intersects、ST_Length、ST_Length2D、ST_3DLength、ST_Length_Spheroid、ST_Length2D_Spheroid、ST_3DLength_Spheroid、ST_LongestLine、ST_OrderingEquals、ST_Overlaps、ST_Perimeter、ST_Perimeter2D、ST_3DPerimeter、ST_PointOnSurface、ST_Project、ST_Relate、ST_RelateMatch、ST_ShortestLine、ST_Touches、ST_Within

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Processing	ST_Buffer、ST_BuildArea、ST_Collect、ST_ConcaveHull、ST_ConvexHull、ST_CurveToLine、ST_DelaunayTriangles、ST_Difference、ST_Dump、ST_DumpPoints、ST_DumpRings、ST_FlipCoordinates、ST_Intersection、ST_LineToCurve、ST_MakeValid、ST_MemUnion、ST_MinimumBoundingCircle、ST_Polygonize、ST_Node、ST_OffsetCurve、ST_RemoveRepeatedPoints、ST_SharedPaths、ST_Shift_Longitude、ST_Simplify、ST_SimplifyPreserveTopology、ST_Split、ST_SymDifference、ST_Union、ST_UnaryUnion
Linear Referencing	ST_LineInterpolatePoint、ST_LineLocatePoint、ST_LineSubstring、ST_LocateAlong、ST_LocateBetween、ST_LocateBetweenElevations、ST_InterpolatePoint、ST_AddMeasure
Miscellaneous Functions	ST_Accum、Box2D、Box3D、ST_Expand、ST_Extent、ST_3Dextent、Find_SRID、ST_MemSize
Exceptional Functions	PostGIS_AddBBox、PostGIS_DropBBox、PostGIS_HasBBox
Raster Management Functions	AddRasterConstraints、DropRasterConstraints、AddOverviewConstraints、DropOverviewConstraints、PostGIS_GDAL_Version、PostGIS_Raster_Lib_Build_Date、PostGIS_Raster_Lib_Version、ST_GDALDrivers、UpdateRasterSRID
Raster Constructors	ST_AddBand、ST_AsRaster、ST_Band、ST_MakeEmptyRaster、ST_Tile、ST_FromGDALRaster
Raster Accessors	ST_GeoReference、ST_Height、ST_IsEmpty、ST_MetaData、ST_NumBands、ST_PixelHeight、ST_PixelWidth、ST_ScaleX、ST_ScaleY、ST_RasterToWorldCoord、ST_RasterToWorldCoordX、ST_RasterToWorldCoordY、ST_Rotation、ST_SkewX、ST_SkewY、ST_SRID、ST_Summary、ST_UpperLeftX、ST_UpperLeftY、ST_Width、ST_WorldToRasterCoord、ST_WorldToRasterCoordX、ST_WorldToRasterCoordY
Raster Band Accessors	ST_BandMetaData、ST_BandNoDataValue、ST_BandIsNoData、ST_BandPath、ST_BandPixelType、ST_HasNoBand
Raster Pixel Accessors and Setters	ST_PixelAsPolygon、ST_PixelAsPolygons、ST_PixelAsPoint、ST_PixelAsPoints、ST_PixelAsCentroid、ST_PixelAsCentroids、ST_Value、ST_NearestValue、ST_Neighborhood、ST_SetValue、ST_SetValues、ST_DumpValues、ST_PixelOfValue

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Raster Editors	ST_SetGeoReference、ST_SetRotation、ST_SetScale、ST_SetSkew、ST_SetSRID、ST_SetUpperLeft、ST_Resample、ST_Rescale、ST_Reskew、ST_SnapToGrid、ST_Resize、ST_Transform
Raster Band Editors	ST_SetBandNoDataValue、ST_SetBandIsNoData
Raster Band Statistics and Analytics	ST_Count、ST_CountAgg、ST_Histogram、ST_Quantile、ST_SummaryStats、ST_SummaryStatsAgg、ST_ValueCount
Raster Outputs	ST_AsBinary、ST_AsGDALRaster、ST_AsJPEG、ST_AsPNG、ST_AsTIFF
Raster Processing	ST_Clip、ST_ColorMap、ST_Intersection、ST_MapAlgebra、ST_Reclass、ST_Union、ST_Distinct4ma、ST_InvDistWeight4ma、ST_Max4ma、ST_Mean4ma、ST_Min4ma、ST_MinDist4ma、ST_Range4ma、ST_StdDev4ma、ST_Sum4ma、ST_Aspect、ST_HillShade、ST_Roughness、ST_Slope、ST_TPI、ST_TRI、Box3D、ST_ConvexHull、ST_DumpAsPolygons、ST_Envelope、ST_MinConvexHull、ST_Polygon、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Disjoint、ST_Intersects、ST_Overlaps、ST_Touches、ST_SameAlignment、ST_NotSameAlignmentReason、ST_Within、ST_DWithin、ST_DFullyWithin
Raster Operators	&&、&<、&>、=、@、~=、~

表 12-2 PostGIS3.2.2 支持的操作符和函数列表

函数分类	包含函数
Management Functions	AddGeometryColumn、DropGeometryColumn、DropGeometryTable、PostGIS_Full_Version、PostGIS_GEOS_Version、PostGIS_Liblwgeom_Version、PostGIS_Lib_Build_Date、PostGIS_Lib_Version、PostGIS_PROJ_Version、PostGIS_Scripts_Build_Date、PostGIS_Scripts_Installed、PostGIS_Version、PostGIS_LibXML_Version、PostGIS_Scripts_Released、Populate_Geometry_Columns 、UpdateGeometrySRID 、PostGIS_Libprotobuf_Version 、PostGIS_Wagyu_Version

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Constructors	ST_BdPolyFromText 、ST_BdMPolyFromText 、ST_Box2dFromGeoHash、ST_GeneratePoints、ST_GeogFromText、ST_GeographyFromText、ST_GeogFromWKB、ST_GeomCollFromText、ST_GeomFromEWKB、ST_GeomFromEWKT、ST_GeometryFromText、ST_GeomFromGeoHash、ST_GeomFromGML、ST_GeomFromGeoJSON、ST_GeomFromKML、ST_GMLToSQL、ST_GeomFromText 、ST_GeomFromWKB、ST_LineFromMultiPoint、ST_LineFromText、ST_LineFromWKB、ST_LinestringFromWKB、ST_MakeBox2D、ST_3DMakeBox、ST_MakeEnvelope、ST_MakePolygon、ST_MakePoint、ST_MakePointM、ST_MLineFromText、ST_MPointFromText、ST_MPolyFromText、ST_Point、ST_Points、ST_PointFromGeoHash、ST_PointFromText、ST_PointFromWKB、ST_Polygon、ST_PolygonFromText、ST_WKBToSQL、ST_WKTToSQL、Geography_Distance_Knn、Geometry_Distance_Cpa、Geometry_Hash、ST_3Dlineinterpolate、ST_AsEncodedPolyline
Geometry Accessors	GeometryType、ST_Boundary、ST_CoordDim、ST_Dimension、ST_EndPoint、ST_Envelope、ST_ExteriorRing、ST_GeometryN、ST_GeometryType、ST_InteriorRingN、ST_IsClosed、ST_IsCollection、ST_IsEmpty、ST_IsPolygonCCW、ST_IsPolygonCW、ST_IsRing、ST_IsSimple、ST_IsValid、ST_IsValidReason、ST_IsValidDetail、ST_M、ST_NDims、ST_NPoints、ST_NRings、ST_NumGeometries、ST_NumInteriorRings、ST_NumInteriorRing、ST_NumPatches、ST_NumPoints、ST_PatchN、ST_PointN、ST_SRID、ST_StartPoint、ST_Summary、ST_X、ST_XMax、ST_XMin、ST_Y、ST_YMax、ST_YMin、ST_Z、ST_ZMax、ST_Zmflag、ST_ZMin、ST_Wrapx、ST_Asmvt

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Editors	ST_AddPoint、ST_Affine、ST_Force2D、ST_Force3D、ST_Force3DZ、ST_Force3DM、ST_Force4D、ST_ForceCollection、ST_ForcePolygonCCW、ST_ForcePolygonCW、ST_ForceSFS、ST_ForceRHR、ST_LineMerge、ST_CollectionExtract、ST_CollectionHomogenize、ST_Multi、ST_Normalize、ST_RemovePoint、ST_Reverse、ST_Rotate、ST_RotateX、ST_RotateY、ST_RotateZ、ST_Scale、ST_Segmentize、ST_SetPoint、ST_SetSRID、ST_SnapToGrid、ST_Snap、ST_Transform、ST_Translate、ST_TransScale、ST_AsmvtGeom、ST_isvalidTrajectory、ST_linefromencodedpolyline、ST_lineinterpolatepoints、ST_MaximuminScribedCircle、ST_OrientedEnvelope、ST_QuantizeCoordinates、ST_ReducePrecision、ST_Scroll、ST_SetEffectiveArea、ST_simplifyvw、ST_square、ST_squaregrid、ST_Swapordinates、ST_Voronoilines、ST_VoronoiPolygons
Geometry Outputs	ST_AsBinary、ST_AsEWKB、ST_AsEWKT、ST_AsGeoJSON、ST_AsGML、ST_AsHEXEWKB、ST_AsKML、ST_AsLatLonText 、ST_AsSVG、ST_AsText、ST_AsTwkb、ST_AsX3D、ST_GeoHash、Json、Jsonb、ST_GeomfromGeojson
Operators	&&、&&&、&<、&<|、&>、<<、<<|、=、>>、@ 、|&> 、|>>、~、~=、<->、<#>、<->、|=|、<<->>
Spatial Relationships and Measurements	ST_3DClosestPoint、ST_3DDistance、ST_3DDWithin、ST_3DDFullyWithin、ST_3DIntersects、ST_3DLongestLine、ST_3DMaxDistance、ST_3DShortestLine、ST_Area、ST_Azimuth、ST_Centroid、ST_ClosestPoint、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_LineCrossingDirection、ST_Disjoint、ST_Distance、ST_HausdorffDistance、ST_MaxDistance、ST_DistanceSphere、ST_DistanceSpheroid、ST_DFullyWithin、ST_DWithin、ST_Equals、ST_HasArc、ST_Intersects、ST_Length、ST_Length2D、ST_3DLength、ST_LengthSpheroid、ST_Length2DSpheroid、ST_LongestLine、ST_MinimumBoundingRadius、ST_OrderingEquals、ST_Overlaps、ST_Perimeter、ST_Perimeter2D、ST_3DPerimeter、ST_PointOnSurface、ST_Project、ST_Relate、ST_RelateMatch、ST_ShortestLine、ST_Touches、ST_Within、_ST_DistancerectTree、_ST_DistancerectTreeCached、_ST_SorTableHash

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Processing	ST_Buffer、ST_BuildArea、ST_ClipByBox2D、ST_ClusterDBSCAN、ST_ClusterIntersecting、ST_ClusterKMeans、ST_ClusterWithin、ST_Collect、ST_ConcaveHull、ST_ConvexHull、ST_CurveToLine、ST_DelaunayTriangles、ST_Difference、ST_Dump、ST_DumpPoints、ST_DumpRings、ST_FlipCoordinates、ST_Intersection、ST_LineToCurve、ST_MakeValid、ST_MemUnion、ST_MinimumBoundingCircle、ST_Polygonize、ST_Node、ST_OffsetCurve、ST_RemoveRepeatedPoints、ST_SharedPaths、ST_ShiftLongitude、ST_Simplify、ST_SimplifyPreserveTopology、ST_Split、ST_Subdivide、ST_SymDifference、ST_Union、ST_UnaryUnion、ST_BoundingDiagonal、ST_ChaikinsMoothing、ST_ClosestPointofApproach、ST_CollectionExtract、ST_CPAwithin、ST_DistanceCPA、ST_DumpSegments、ST_EstimatedExtent、ST_Filterbym、ST_SetEffectiveArea、ST_Forcecurve
Linear Referencing	ST_LineInterpolatePoint、ST_LineLocatePoint、ST_LineSubstring、ST_LocateAlong、ST_LocateBetween、ST_LocateBetweenElevations、ST_InterpolatePoint、ST_AddMeasure
Miscellaneous Functions	Array_Agg、Box2D、Box3D、ST_Expand、ST_Extent、ST_3Dextent、Find_SRID、ST_MemSize
Exceptional Functions	PostGIS_AddBBox、PostGIS_DropBBox、PostGIS_HasBBox

空间索引

DWS数据库的PostGIS Extension支持GiST (Generalized Search Tree) 空间索引（分区表除外）。相比于B-tree索引，GiST索引适应于任意类型的非常规数据结构，可有效提高几何和地理数据信息的检索效率。

使用如下命令创建GiST索引：

```
CREATE INDEX indexname ON tablename USING gist ( geometryfield );
```

扩展限制

- 只支持行存表，不支持列存索引。
- 只支持Oracle兼容格式数据库。
- 不支持拓扑对象管理模块Topology。
- 不支持BRIN索引。
- spatial_ref_sys表在扩容期间只支持查询操作。

版权所有 © 华为云计算技术有限公司

插件升级兼容性

PostGIS 2.4.2版本升级到3.2.2版本时，存在部分函数不兼容或没有完全的前向兼容，导致出现升级前后函数功能表现不一致的情况，所以在PostGIS升级时，需从业务自身角度来评估升级不兼容带来的影响。

不兼容函数列表如下。

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
3.2.2新增函数	无	ST_IsPolygonCW(geometry)
	无	ST_IsPolygonCCW(geometry)
	无	ST_PointInsideCircle(geometry,float8,float8,float8)
	无	ST_ForcePolygonCW(geometry)
	无	ST_ForcePolygonCCW(geometry)
	无	ST_Normalize(geom geometry)
	无	ST_AsTWKB(geom geometry, prec int4 default 0, prec_z int4 default 0, prec_m int4 default 0, with_sizes boolean default false, with_boxes boolean default false)
	无	ST_AsTWKB(geom geometry[], ids bigint[], prec int4 default 0, prec_z int4 default 0, prec_m int4 default 0, with_sizes boolean default false, with_boxes boolean default false)
	无	ST_MakeLine (geometry[])
	无	ST_TileEnvelope(zoom integer, x integer, y integer, bounds geometry DEFAULT 'SRID=3857;LINESTRING(-20037508.342789244 -20037508.342789244, 20037508.342789244 20037508.342789244)'::geometry, margin float8 DEFAULT 0.0)
	无	ST_ClusterIntersecting(geometry[])
	无	ST_ClusterWithin(geometry[], float8)
	无	ST_ClusterDBSCAN (geometry, eps float8, minpoints int)

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	无	ST_Scale(geometry,geometry,origin geometry)
	无	ST_GeneratePoints(area geometry, npoints integer, seed integer)
	无	ST_FrechetDistance(geom1 geometry, geom2 geometry, float8 default -1)
	无	ST_Points(geometry)
	无	ST_ClipByBox2d(geom geometry, box box2d)
	无	ST_Subdivide(geom geometry, maxvertices integer DEFAULT 256, gridSize float8 DEFAULT -1.0)
	无	ST_ClusterIntersecting (geometry)
	无	ST_ClusterWithin (geometry, float8)
	无	ST_ClusterKMeans(geom geometry, k integer, max_radius float8 default null)
	无	ST_AsText(geometry, int4)
	无	ST_AsEWKT(geography, int4)
	无	_ST_CoveredBy(geog1 geography, geog2 geography)
	无	ST_Point(float8, float8, srid integer)
3.2.2不再支持	ST_3DLength_spheroid(geometry, spheroid)	无
	ST_length2d_spheroid(geometry, spheroid)	无
	ST_locate_between_measures(geometry, float8, float8)	无
	ST_locate_along_measure(geometry, float8)	无
	ST_Buffer(geometry,float8, text)	无
	ST_GeneratePoints(area geometry, npoints integer)	无

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_Combine_BBox(box3d, geometry)	无
	ST_Combine_BBox(box2d, geometry)	无
	pgis_abs_in(cstring)	无
	pgis_abs_out(pgis_abs)	无
	pgis_abs (internallength = 16, input = pgis_abs_in, output = pgis_abs_out, alignment = double)	无
	ST_MemUnion(geometry)	无
	pgis_geometry_accum_finalfn(pgis_abs)	无
	ST_MakeLine (geometry)	无
	ST_Accum (geometry)	输入单geometry数据，会直接输出该geometry的值，多行geometry数据则会报错不支持。
	_ST_AsKML(int4,geometry, int4, text)	无
	ST_MemUnion(geometry)	无
	_ST_AsGeoJson(int4, geometry, int4, int4)	无
	ST_AsGeoJson(gj_version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0)	无
	_ST_DWithin(geography, geography, float8, boolean)	无
	ST_point_inside_circle(geometry,float8,float8,float8)	无
	ST_CurveToLine(geometry)	无
	ST_Shift_Longitude(geometry)	无，改用ST_ShiftLongitude
	ST_find_extent(text,text,text) , ST_find_extent(text,text)	无，改用ST_FindExtent

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_mem_size(geometry)	无，改用ST_MemSize
	ST_length_spheroid(geometry, spheroid)	无，改用ST_LengthSpheroid
	ST_distance_spheroid(geom1 geometry, geom2 geometry,spheroid)	无，改用ST_DistanceSpheroid
	ST_force_2d(geometry)	无，改用ST_Force2D
	ST_force_3dz(geometry)	无，改用ST_Force3DZ
	ST_force_3d(geometry)	无，改用ST_Force3D
	ST_force_3dm(geometry)	无，改用ST_Force3DM
	ST_force_4d(geometry)	无，改用ST_Force4D
	ST_force_collection(geometry)	无，改用ST_ForceCollection
	ST_line_locate_point(geom1 geometry, geom2 geometry)	无，改用ST_LineLocatePoint
	ST_line_interpolate_point(geometry, float8)	无，改用ST_LineInterpolatePoint
	ST_Buffer(geometry,float8)	无，改用ST_Buffer
3.2.2参数类型变化	pgis_geometry_accum_transfn(pgis_abs, geometry)	pgis_geometry_accum_transfn(internal, geometry)
	pgis_geometry_accum_transfn(pgis_abs, geometry, float8)	pgis_geometry_accum_transfn(internal, geometry, float8)
	pgis_geometry_accum_transfn(pgis_abs, geometry, float8, int)	pgis_geometry_accum_transfn(internal, geometry, float8, int)
	pgis_geometry_union_finalfn(pgis_abs)	pgis_geometry_union_finalfn(internal)
	pgis_geometry_collect_finalfn(pgis_abs)	pgis_geometry_collect_finalfn(internal)
	pgis_geometry_polygonize_finalfn(pgis_abs)	pgis_geometry_polygonize_finalfn(internal)
	pgis_geometry_clusterintersecting_finalfn(pgis_abs)	pgis_geometry_clusterintersecting_finalfn(internal)
	ST_Union (geometry)	ST_Union (geometry)

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_Collect (geometry)	ST_Collect (geometry)
	ST_Buffer(geometry,float8, integer)	ST_Buffer(geom geometry, radius float8, quadsegs integer)
3.2.2接口变更	ST_AsKML(version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix text DEFAULT null)	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix TEXT default ' ')
	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15)	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix TEXT default ' ')
	ST_AsKML(version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix text DEFAULT null)	ST_AsGML(version int4, geog geography, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0, nprefix text DEFAULT 'gml', id text DEFAULT '')
3.2.2默认参数变化	ST_SymDifference(geom1 geometry, geom2 geometry)	ST_SymDifference(geom1 geometry, geom2 geometry, gridSize float8 DEFAULT -1.0)
	ST_UnaryUnion(geometry)	ST_UnaryUnion(geometry, gridSize float8 DEFAULT -1.0)
	ST_AsGeoJson(geom geometry, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0)	ST_AsGeoJson(geom geometry, maxdecimaldigits int4 DEFAULT 9, options int4 DEFAULT 8)
	ST_Buffer(geometry,float8, cstring)	ST_Buffer(geom geometry, radius float8, options text DEFAULT ' ')
	_ST_DWithin(geography, geography, float8, boolean)	_ST_DWithin(geog1 geography, geog2 geography, tolerance float8, use_spheroid boolean DEFAULT true)
	ST_IsValidDetail(geometry)	ST_IsValidDetail(geom geometry, flags int4 DEFAULT 0)
	ST_CurveToLine(geom geometry, tol float8, toltype integer, flags integer)	ST_CurveToLine(geom geometry, tol float8 DEFAULT 32, toltype integer DEFAULT 0, flags integer DEFAULT 0)
3.2.2算子支持hash join和merge join	OPERATOR =	OPERATOR =

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
3.2.2功能变更	ST_ConcaveHull弱校验	ST_ConcaveHull添加强校验，校验更为严格。
3.2.2不定义commutor	OPERATOR &< ,OPERATOR &<|,OPERATOR |&>	OPERATOR &< ,OPERATOR &<|,OPERATOR |&>

12.4 OPEN SOURCE SOFTWARE NOTICE (For PostGIS)

This document contains open source software notice for the product. And this document is confidential information of copyright holder. Recipient shall protect it in due care and shall not disseminate it without permission.

Warranty Disclaimer

This document is provided “as is” without any warranty whatsoever, including the accuracy or comprehensiveness. Copyright holder of this document may change the contents of this document at any time without prior notice, and copyright holder disclaims any liability in relation to recipient’s use of this document.

Open source software is provided by the author “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of open source software, even if advised of the possibility of such damage.

Copyright Notice And License Texts

Software: postgis-2.4.2

Copyright notice:

"Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Copyright 2008 Kevin Neufeld

Copyright (c) 2009 Walter Bruce Sinclair

Copyright 2006-2013 Stephen Woodbridge.

Copyright (c) 2008 Walter Bruce Sinclair

Copyright (c) 2012 TJ Holowaychuk <tj@vision-media.ca>

Copyright (c) 2008, by Attractive Chaos <attractivechaos@aol.co.uk>

 版权所有 © 华为云计算技术有限公司

Copyright (c) 2001-2012 Walter Bruce Sinclair

Copyright (c) 2010 Walter Bruce Sinclair

Copyright 2006 Stephen Woodbridge

Copyright 2006-2010 Stephen Woodbridge.

Copyright (c) 2006-2014 Stephen Woodbridge.

Copyright (c) 2017, Even Rouault <even.rouault at spatialys.com>

Copyright (C) 2004-2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2015 Nicklas Avén <nicklas.aven@jordogskog.no>

Copyright 2008 Paul Ramsey

Copyright (C) 2012 Sandro Santilli <strk@kbt.io>

Copyright 2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2014 Sandro Santilli <strk@kbt.io>

Copyright 2013 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2011 Sandro Santilli <strk@kbt.io>

Copyright 2015 Daniel Baston

Copyright 2009 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Kashif Rasul <kashif.rasul@gmail.com> and

Shoaib Burq <saburq@gmail.com>

Copyright 2013 Sandro Santilli <strk@kbt.io>

Copyright 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright 2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Nicklas Avén

Copyright 2011-2016 Regina Obe

Copyright (C) 2008 Paul Ramsey

Copyright (C) 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2010-2012 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2015 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2013 Nicklas Avén

Copyright (C) 2016 Sandro Santilli <strk@kbt.io>

Copyright 2017 Darafei Praliaskouski <me@komzpa.net>

Copyright (c) 2016, Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011-2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007-2008 Mark Cave-Ayland

Copyright (C) 2001-2006 Refractions Research Inc.

Copyright 2015 Daniel Baston <dbaston@gmail.com>

Copyright 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2012-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2012-2015 Sandro Santilli <strk@kbt.io>

Copyright 2001-2006 Refractions Research Inc.

Copyright (C) 2004 Refractions Research Inc.

Copyright 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright 2009-2010 Sandro Santilli <strk@kbt.io>

Copyright 2015-2016 Daniel Baston <dbaston@gmail.com>

Copyright 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2007-2008 Mark Cave-Ayland

Copyright 2012-2013 Oslandia <infos@oslandia.com>

Copyright (C) 2015-2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2001-2003 Refractions Research Inc.

Copyright 2016 Sandro Santilli <strk@kbt.io>

Copyright 2011 Kashif Rasul <kashif.rasul@gmail.com>

Copyright (C) 2014 Nicklas Avén

Copyright (C) 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 1984, 1989-1990, 2000-2015 Free Software Foundation, Inc.

Copyright (C) 2011 Paul Ramsey

 版权所有 © 华为云计算技术有限公司

Copyright 2001-2003 Refractions Research Inc.

Copyright 2009-2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2010-2012 Oslandia

Copyright 2006 Corporacion Autonoma Regional de Santander

Copyright 2013 Nicklas Avén

Copyright 2011-2016 Arrival 3D, Regina Obe

Copyright (C) 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009-2012 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 - Oslandia

Copyright (C) 2006 Mark Leslie <mark.leslie@lisasoft.com>

Copyright (C) 2008-2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2009-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 Olivier Courtin <olivier.courtin@camptocamp.com>

Copyright 2010 Nicklas Avén

Copyright 2012 Paul Ramsey

Copyright 2011 Nicklas Avén

Copyright 2002 Thamer Alharbash

Copyright 2011 OSGeo

Copyright (C) 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2004-2007 Refractions Research Inc.

Copyright 2010 LISAsoft Pty Ltd

Copyright 2010 Mark Leslie

Copyright (c) 1999, Frank Warmerdam

Copyright 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 2007, Frank Warmerdam

Copyright 2008 OpenGeo.org

Copyright (C) 2008 OpenGeo.org

Copyright (C) 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2010 LISAsoft

Copyright (C) 2010 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 1999, 2001, Frank Warmerdam

Copyright (C) 2016-2017 Bj?rn Harrtell <bjorn@wololo.org>

Copyright (C) 2017 Danny G?tte <danny.goette@fem.tu-ilmenau.de>

Copyright 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

^copyright^

Copyright 2012 (C) Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2006 Refractions Research Inc.

Copyright 2009 Paul Ramsey <pramsey@opengeo.org>

Copyright 2001-2009 Refractions Research Inc.

Copyright (C) 2010 Olivier Courtin <olivier.courtin@oslandia.com>

By Nathan Wagner, copyright disclaimed,

this entire file is in the public domain

Copyright 2009-2011 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2001-2005 Refractions Research Inc.

Copyright 2001-2011 Refractions Research Inc.

Copyright 2009-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2010 Sandro Santilli <strk@kbt.io>

Copyright 2012 J Smith <dark.panda@gmail.com>

Copyright 2009 - 2010 Oslandia

Copyright 2009 Oslandia

Copyright 2001-2005 Refractions Research Inc.

Copyright 2016 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2016 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2011 OpenGeo.org

Copyright (c) 2003-2017, Troy D. Hanson http:troydhanson.github.com/uthash/

Copyright (C) 2011 Regents of the University of California

Copyright (C) 2011-2013 Regents of the University of California

Copyright (C) 2010-2011 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2010-2011 David Zwarg <dzwarg@azavea.com>

Copyright (C) 2009-2011 Pierre Racine <pierre.racine@sbf.ulaval.ca>

Copyright (C) 2009-2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2008-2009 Sandro Santilli <strk@kbt.io>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2013 Nathaneil Hunter Clay <clay.nathaniel@gmail.com

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

Copyright (C) 2013 Bborie Park <dustymugs@gmail.com>

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

(C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2012 Regents of the University of California

Copyright (C) 2013 Regents of the University of California

Copyright (C) 2012-2013 Regents of the University of California

Copyright (C) 2009 Sandro Santilli <strk@kbt.io>

"

License: The GPL v2 License.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate

 版权所有 © 华为云计算技术有限公司

to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.?

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

 版权所有 © 华为云计算技术有限公司

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

 版权所有 © 华为云计算技术有限公司

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of

 版权所有 © 华为云计算技术有限公司

this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

 版权所有 © 华为云计算技术有限公司

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301

 版权所有 © 华为云计算技术有限公司

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Software:Geos

Copyright notice：

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2013 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

 版权所有 © 华为云计算技术有限公司

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

 版权所有 © 华为云计算技术有限公司

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

 版权所有 © 华为云计算技术有限公司

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

 版权所有 © 华为云计算技术有限公司

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

 版权所有 © 华为云计算技术有限公司

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public

Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the

 版权所有 © 华为云计算技术有限公司

users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and

is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in

non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The

former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

 版权所有 © 华为云计算技术有限公司

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under

copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the

Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1

above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

 版权所有 © 华为云计算技术有限公司

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of

its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in

themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or

collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

 版权所有 © 华为云计算技术有限公司

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany

it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to

distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a

work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License.

Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object

file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6,

whether or not they are linked directly with the Library itself.

 版权所有 © 华为云计算技术有限公司

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work

under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system,

rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception,

the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

 版权所有 © 华为云计算技术有限公司

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot

use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you

 版权所有 © 华为云计算技术有限公司

may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the

integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot

impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time.

Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is

copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing and reuse of software generally.

 版权所有 © 华为云计算技术有限公司

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING

RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

 版权所有 © 华为云计算技术有限公司

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

Software: JSON-C

Copyright notice:

Copyright (c) 2004, 2005 Metaparadigm Pte. Ltd.

Copyright (c) 2009-2012 Eric Haszlakiewicz

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Copyright (c) 2009 Hewlett-Packard Development Company, L.P.

Copyright 2011, John Resig

Copyright 2011, The Dojo Foundation

Copyright (c) 2012 Eric Haszlakiewicz

Copyright (c) 2009-2012 Hewlett-Packard Development Company, L.P.

Copyright (c) 2008-2009 Yahoo! Inc. All rights reserved.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) 2013 Metaparadigm Pte. Ltd.

License: MIT License

Copyright (c) 2009-2012 Eric Haszlakiewicz

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,

and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Software: proj

Copyright notice:

"Copyright (C) 2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2007 Douglas Gregor <doug.gregor@gmail.com>

Copyright (C) 2007 Troy Straszheim

CMake, Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>)

Copyright (C) 2011 Nicolas David <nicolas.david@ign.fr>

Copyright (c) 2000, Frank Warmerdam

 版权所有 © 华为云计算技术有限公司

Copyright (c) 2011, Open Geospatial Consortium, Inc.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) Charles Karney (2012-2015) <charles@karney.com> and licensed

Copyright (c) 2005, Antonello Andrea

Copyright (c) 2010, Frank Warmerdam

Copyright (c) 1995, Gerald Evenden

Copyright (c) 2000, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2010, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2013, Frank Warmerdam

Copyright (c) 2003 Gerald I. Evenden

Copyright (c) 2012, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2002, Frank Warmerdam

Copyright (c) 2004 Gerald I. Evenden

Copyright (c) 2012 Martin Raspaud

Copyright (c) 2001, Thomas Flemming, tf@ttqv.com

Copyright (c) 2002, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2009, Frank Warmerdam

Copyright (c) 2003, 2006 Gerald I. Evenden

Copyright (c) 2011, 2012 Martin Lambers <marlam@marlam.de>

Copyright (c) 2006, Andrey Kiselev

Copyright (c) 2008-2012, Even Rouault <even dot rouault at mines-paris dot org>

Copyright (c) 2001, Frank Warmerdam

Copyright (c) 2001, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2008 Gerald I. Evenden

"

License: MIT License

Please see above

Software: libxml2

Copyright notice:

"See Copyright for the status of this software.

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Copyright (C) 2003 Daniel Veillard.

copy: see Copyright for the status of this software.

copy: see Copyright for the status of this software

copy: see Copyright for the status of this software.

Copyright (C) 2000 Bjorn Reese and Daniel Veillard.

Copy: See Copyright for the status of this software.

See COPYRIGHT for the status of this software

Copyright (C) 2000 Gary Pennington and Daniel Veillard.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006, 2007 Free Software Foundation, Inc.

Copyright (C) 1998 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese <breese@users.sourceforge.net>

Copyright (C) 2000 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese and Daniel Stenberg.

See Copyright for the status of this software

"

License: MIT License

Please see above

13 使用 JDBC 或 ODBC 进行 DWS 二次开发

13.1 开发前准备

如果用户在APP的开发中，使用了连接池机制，那么需要遵循如下规范：

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

否则，连接池里面的连接就是有状态的，会对用户后续使用连接池进行操作的正确性带来影响。

驱动下载

请参见《数据仓库服务用户指南》的“下载JDBC或ODBC驱动”章节。

13.2 基于 JDBC 开发

13.2.1 JDBC 开发流程

JDBC（Java Database Connectivity，java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问接口，应用程序可基于它操作数据。DWS库提供了对JDBC 4.0特性的支持，需要使用JDK1.6及以上版本编译程序代码，不支持JDBC桥接ODBC方式。JDBC开发应用程序的具体流程如下：

 版权所有 © 华为云计算技术有限公司

图 13-1 采用 JDBC 开发应用程序的流程

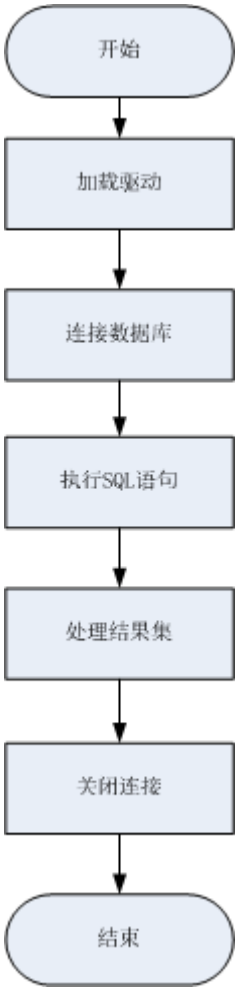


表 13-1 JDBC 开发流程

步骤	描述
加载驱动	下载JDBC驱动并在程序中编译和加载。
连接数据库	通过JDBC驱动连接数据库。
执行SQL语句	应用程序通过执行SQL语句来操作数据库的数据。
处理结果集	不同类型的结果集有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。
关闭连接	在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

13.2.2 JDBC 包与驱动类

JDBC 包

从管理控制台下载包名为dws_9.1.x_jdbc_driver.zip的软件包。

 版权所有 © 华为云计算技术有限公司

请参见《 数据仓库服务用户指南 》的“下载JDBC或ODBC驱动”章节。

解压后为JDBC的驱动jar包：

gsjdbc4.jar：与PostgreSQL保持兼容的驱动包，其中类名、类结构与PostgreSQL驱动完全一致，曾经运行于PostgreSQL的应用程序可以直接移植到当前系统使用。

驱动类

在创建数据库连接之前，需要加载数据库驱动类“org.postgresql.Driver”（ 对应包 gsjdbc4.jar ）。

13.2.3 加载驱动

在创建数据库连接之前，需要先加载数据库驱动程序。

加载驱动有两种方法：

- 在代码中创建连接之前任意位置隐含装载：
  Class.forName("org.postgresql.Driver");
- 在JVM启动时参数传递：java -Djdbc.drivers=org.postgresql.Driver jdbctest

13.2.4 连接数据库

在创建数据库连接之后，才能使用它来执行SQL语句操作数据。

前提条件

如果您使用的是开源的JDBC驱动程序，应确保数据库参数password_encryption_type取值设置为1，如果参数值不为1，可能会出现连接失败，典型的报错信息比如：“none of the server's SASL authentication mechanisms are supported”，可参考如下操作进行处理：

1. 联系技术支持人员检查数据库参数password_encryption_type是否为1，如果取值不为1，需要将该参数修改为1。
2. 新建一个数据库用户用于连接，或者重置准备使用的数据库用户的密码。
   - 如果您使用的是管理员账号，参考《 数据仓库服务用户指南 》的“重置DWS数据库系统管理员密码”章节进行密码重置。
   - 如果是普通用户，可以先通过其他客户端工具（ 例如Data Studio ）连接数据库后，使用ALTER USER语句来修改密码。
3. 再尝试连接数据库。

需要执行以上操作的原因：

- 调整参数的原因：当前MD5算法已被证实可以人工碰撞，已严禁将之用于密码校验算法。DWS采用默认安全设计，默认禁止MD5算法的密码校验，可能导致开源客户端无法正常连接的问题。所以需要调整密码算法参数password_encryption_type，打开MD5算法。
- 修改密码的原因：DWS中是不会存储您的密码原文的，而是存储的密码HASH摘要（ 默认是SHA256摘要 ），在密码校验时该摘要会与客户端发来的密码摘要进行比对（ 中间会有加盐操作 ）。故当您只是单纯调整了密码算法策略时，数据库是无法还原您的密码进而再生成MD5的摘要值的，必须要求您手动修改一次密码或者创建一个新用户，这时新的密码将会采用您设置的HASH算法进行摘要存储，用于下次连接认证。

 版权所有 © 华为云计算技术有限公司

函数原型

JDBC提供了三个方法，用于创建数据库连接。

- DriverManager.getConnection(String url);
- DriverManager.getConnection(String url, Properties info);
- DriverManager.getConnection(String url, String user, String password);

参数

表 13-2 数据库连接参数

参数	描述
url	gsjdbc4.jar数据库连接描述符。格式如下： ● jdbc:postgresql:database ● jdbc:postgresql://host/database ● jdbc:postgresql://host:port/database ● jdbc:postgresql://host:port[,host:port][...]/database **说明** 使用gsjdbc200.jar时，将“jdbc:postgresql”修改为“jdbc:gaussdb” ● database为要连接的数据库名称。 ● host为数据库服务器名称或IP地址，当集群绑定弹性负载均衡(ELB)时，应设置为ELB的IP地址。 由于安全原因，数据库CN禁止集群内部其他节点无认证接入。如果要在集群内部访问CN，请将JDBC程序部署在CN所在机器，host使用"127.0.0.1"。否则可能会出现“FATAL: Forbid remote connection with trust method!”错误。 建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。 ● port为数据库服务器端口。缺省情况下，会尝试连接到localhost的8000端口的database。 ● 支持多ip端口配置形式，jdbc自动实现了负载均衡，多ip端口配置形式是采取随机访问+failover的方式，这个过程系统会自动忽略不可达IP。 以","隔开，例如jdbc:postgresql://10.10.0.13:8000,10.10.0.14:8000/database ● 使用JDBC连接集群时集群链接地址只支持指定jdbc连接参数，不支持增加变量参数。

 版权所有 © 华为云计算技术有限公司

参数	描述
info	数据库连接属性。常用的属性如下： • user：String类型。表示创建连接的数据库用户。 • password：String类型。表示数据库用户的密码。 • ssl：Boolean类型。表示是否使用SSL连接。 • loggerLevel：string类型。为LogStream或LogWriter设置记录进DriverManager当前值的日志信息量。目前支持"OFF"、"DEBUG"和"TRACE"。 值为"DEBUG"时，表示只打印DEBUG级别以上的日志，将记录非常少的信息。值等于TRACE时，表示打印DEBUG和TRACE级别的日志，将产生详细的日志信息。默认值为OFF，表示不打印日志。 • prepareThreshold：integer类型。用于确定在转换为服务器端的预备语句之前，要求执行方法PreparedStatement的次数。缺省值是5。 • batchMode : boolean类型，用于确定是否使用batch模式连接。 • fetchsize： integer类型，用于设置数据库链接所创建statement的默认fetchsize。 • ApplicationName：string类型。应用名称，在不做设置时，缺省值为PostgreSQL JDBC Driver。 • allowReadOnly:boolean类型，用于设置connection是否允许设置readonly模式，默认为false，若该参数不被设置为true，则执行connection.setReadOnly不生效。 • blobMode:string类型，用于设置setBinaryStream方法为不同的数据类型赋值，设置为on时表示为blob数据类型赋值，设置为off时表示为bytea数据类型赋值，默认为on。 • connectionExtraInfo：Boolean类型。表示驱动是否上报当前驱动的部署路径、进程属主用户到数据库。 **说明** 取值范围：true或false，默认值为true。设置connectionExtraInfo为true，JDBC驱动会将当前驱动的部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。
user	数据库用户。
password	数据库用户的密码。

关闭连接

在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

关闭数据库连接可以直接调用其close方法即可。如：**conn.close()**

示例

```
//以下用例以gsjdbc4.jar为例。//以下代码将获取数据库连接操作封装为一个接口，可通过给定用户名和密码来连接数据库。

public static Connection GetConnection(String username, String passwd) {
```

 版权所有 © 华为云计算技术有限公司

```
    //驱动类。
    String driver = "org.postgresql.Driver";
    //数据库连接描述符。
    String sourceURL = "jdbc:postgresql://10.10.0.13:8000/postgres?currentSchema=test";
    Connection conn = null;

    try {
        //加载驱动。
        Class.forName(driver);
    } catch (ClassNotFoundException e ){
        e.printStackTrace();
        return null;
    }

    try {
         //创建连接。
        conn = DriverManager.getConnection(sourceURL, username, passwd);
        System.out.println("Connection succeed!");
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }

    return conn;
}
```

13.2.5 执行 SQL 语句

执行普通 SQL 语句

应用程序通过执行SQL语句来操作数据库的数据（不用传递参数的语句），需要按以下步骤执行：

步骤1 调用Connection的createStatement方法创建语句对象。

```
Statement stmt = con.createStatement();
```

步骤2 调用Statement的executeUpdate方法执行SQL语句。

```
int rc = stmt.executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name VARCHAR(32));");
```

说明

数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，事务块中不支持vacuum操作。如果其中有一个语句失败，那么整个请求都将会被回滚。

步骤3 关闭语句对象。

```
stmt.close();
```

----结束

执行预编译 SQL 语句

预编译语句是只编译和优化一次，然后可以通过设置不同的参数值多次使用。由于已经预先编译好，后续使用会减少执行时间。因此，如果多次执行一条语句，请选择使用预编译语句。可以按以下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
```

步骤2 调用PreparedStatement的setShort设置参数。

```
pstmt.setShort(1, (short)2);
```

 版权所有 © 华为云计算技术有限公司

步骤3 调用PreparedStatement的executeUpdate方法执行预编译SQL语句。

```
int rowcount = pstmt.executeUpdate();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

----结束

调用存储过程

DWS支持通过JDBC直接调用事先创建的存储过程，步骤如下：

步骤1 调用Connection的prepareCall方法创建调用语句对象。

```
CallableStatement cstmt = myConn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
```

步骤2 调用CallableStatement的setInt方法设置参数。

```
cstmt.setInt(2, 50);
cstmt.setInt(1, 20);
cstmt.setInt(3, 90);
```

步骤3 调用CallableStatement的registerOutParameter方法注册输出参数。

```
cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
```

步骤4 调用CallableStatement的execute执行方法调用。

```
cstmt.execute();
```

步骤5 调用CallableStatement的getInt方法获取输出参数。

```
int out = cstmt.getInt(4);  //获取out参数
```

示例：

```
//在数据库中已创建了如下存储过程，它带有out参数。
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
```

步骤6 调用CallableStatement的close方法关闭调用语句。

```
cstmt.close();
```

说明

- 很多的数据库例如Connection、Statement和ResultSet都有close()方法，在使用完对象后应把它们关闭。要注意的是，Connection的关闭将间接关闭所有与它关联的Statement，Statement的关闭间接关闭了ResultSet。
- 一些JDBC驱动程序还提供命名参数的方法来设置参数。命名参数的方法允许根据名称而不是顺序来设置参数，若参数有默认值，则可以不用指定参数值就可以使用此参数的默认值。即使存储过程中参数的顺序发生了变更，也不必修改应用程序。目前DWS数据库的JDBC驱动程序不支持此方法。
- DWS数据库不支持带有输出参数的函数，也不支持存储过程和函数参数默认值。

----结束

 版权所有 © 华为云计算技术有限公司

须知

- 当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。
- 存储过程不能和普通SQL在同一条语句中执行。

执行批处理

用一条预处理语句处理多条相似的数据，数据库只创建一次执行计划，节省了语句的编译和优化时间。可以按如下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("INSERT INTO customer_t1 VALUES (?)");
```

步骤2 针对每条数据都要调用setShort设置参数，以及调用addBatch确认该条设置完毕。

```
pstmt.setShort(1, (short)2);
pstmt.addBatch();
```

步骤3 调用PreparedStatement的executeBatch方法执行批处理。

```
int[] rowcount = pstmt.executeBatch();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

说明

在实际的批处理过程中，通常不终止批处理程序的执行，否则会降低数据库的性能。因此在批处理程序时，应该关闭自动提交功能，每几行提交一次。关闭自动提交功能的语句为：
conn.setAutoCommit(false);

----结束

13.2.6 处理结果集

设置结果集类型

不同类型的结果集有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。在执行SQL语句过程中，都需要先创建相应的语句对象，而部分创建语句对象的方法提供了设置结果集类型的功能。具体的参数设置如**表13-3**所示。涉及的Connection的方法如下：

```
//创建一个Statement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
createStatement(int resultSetType, int resultSetConcurrency);

//创建一个PreparedStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareStatement(String sql, int resultSetType, int resultSetConcurrency);

//创建一个CallableStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareCall(String sql, int resultSetType, int resultSetConcurrency);
```

 版权所有 © 华为云计算技术有限公司

表 13-3 结果集类型

参数	描述
resultSetType	表示结果集的类型，具体有三种类型： ● ResultSet.TYPE_FORWARD_ONLY：ResultSet只能向前移动。是缺省值。 ● ResultSet.TYPE_SCROLL_SENSITIVE：在修改后重新滚动到修改所在行，可以看到修改后的结果。 ● ResultSet.TYPE_SCROLL_INSENSITIVE：对可修改例程所做的编辑不进行显示。 **说明** 结果集从数据库中读取了数据之后，即使类型是ResultSet.TYPE_SCROLL_SENSITIVE，也不会看到由其他事务在这之后引起的改变。调用ResultSet的refreshRow()方法，可进入数据库并从其中取得当前游标所指记录的最新数据。
resultSetConcurrency	表示结果集的并发，具体有两种类型： ● ResultSet.CONCUR_READ_ONLY：如果不从结果集中的数据建立一个新的更新语句，不能对结果集中的数据进行更新。 ● ResultSet.CONCUR_UPDATEABLE：可改变的结果集。对于可滚动的结果集，可对结果集进行适当的改变。

在结果集中定位

ResultSet对象具有指向其当前数据行的光标。最初，光标被置于第一行之前。next方法将光标移动到下一行；因为该方法在ResultSet对象没有下一行时返回false，所以可以在while循环中使用它来迭代结果集。但对于可滚动的结果集，JDBC驱动程序提供更多的定位方法，使ResultSet指向特定的行。定位方法如表13-4所示。

表 13-4 在结果集中定位的方法

方法	描述
next()	把ResultSet向下移动一行。
previous()	把ResultSet向上移动一行。
beforeFirst()	把ResultSet定位到第一行之前。
afterLast()	把ResultSet定位到最后一行之后。
first()	把ResultSet定位到第一行。
last()	把ResultSet定位到最后一行。
absolute(int)	把ResultSet移动到参数指定的行数。
relative(int)	向前或者向后移动参数指定的行。

 版权所有 © 华为云计算技术有限公司

获取结果集中光标的位置

对于可滚动的结果集，可能会调用定位方法来改变光标的位置。JDBC驱动程序提供了获取结果集中光标所处位置的方法。获取光标位置的方法如表13-5所示。

表 13-5 获取结果集光标的位置

方法	描述
isFirst()	是否在一行。
isLast()	是否在最后一行。
isBeforeFirst()	是否在第一行之前。
isAfterLast()	是否在最后一行之后。
getRow()	获取当前在第几行。

获取结果集中的数据

ResultSet对象提供了丰富的方法，以获取结果集中的数据。获取数据常用的方法如表13-6所示，其他方法请参考JDK官方文档。

表 13-6 ResultSet 对象的常用方法

方法	描述
int getInt(int columnIndex)	按列标获取int型数据。
int getInt(String columnLabel)	按列名获取int型数据。
String getString(int columnIndex)	按列标获取String型数据。
String getString(String columnLabel)	按列名获取String型数据。
Date getDate(int columnIndex)	按列标获取Date型数据
Date getDate(String columnLabel)	按列名获取Date型数据。

13.2.7 常用 JDBC 开发示例

示例 1

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
```


版权所有 © 华为云计算技术有限公司

```
end;
/
```

此示例将演示如何基于DWS提供的JDBC接口开发应用程序。

```
//DBtest.java
//以下用例以gsjdbc4.jar为例。//演示基于JDBC开发的主要步骤，会涉及创建数据库、创建表、插入数据等。

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.CallableStatement;

public class DBTest {

 //创建数据库连接。
 public static Connection GetConnection(String username, String passwd) {
   String driver = "org.postgresql.Driver";
   String sourceURL = "jdbc:postgresql://localhost:/gaussdb";
   Connection conn = null;
   try {
     //加载数据库驱动。
     Class.forName(driver).newInstance();
   } catch (Exception e) {
     e.printStackTrace();
     return null;
   }

   try {
     //创建数据库连接。
     conn = DriverManager.getConnection(sourceURL, username, passwd);
     System.out.println("Connection succeed!");
   } catch (Exception e) {
     e.printStackTrace();
     return null;
   }

   return conn;
 };

 //执行普通SQL语句，创建customer_t1表。
 public static void CreateTable(Connection conn) {
   Statement stmt = null;
   try {
     stmt = conn.createStatement();

     //执行普通SQL语句。
     int rc = stmt
         .executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");

     stmt.close();
   } catch (SQLException e) {
     if (stmt != null) {
       try {
         stmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
   }
 }

 //执行预处理语句，批量插入数据。
 public static void BatchInsertData(Connection conn) {
   PreparedStatement pst = null;
```

 版权所有 © 华为云计算技术有限公司

```
  try {
    //生成预处理语句。
    pst = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?,?)");
    for (int i = 0; i < 3; i++) {
      //添加参数。
      pst.setInt(1, i);
      pst.setString(2, "data " + i);
      pst.addBatch();
    }
    //执行批处理。
    pst.executeBatch();
    pst.close();
  } catch (SQLException e) {
    if (pst != null) {
      try {
        pst.close();
      } catch (SQLException e1) {
      e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
 }

 //执行预编译语句，更新数据。
 public static void ExecPreparedSQL(Connection conn) {
   PreparedStatement pstmt = null;
   try {
     pstmt = conn
         .prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
     pstmt.setString(1, "new Data");
     int rowcount = pstmt.executeUpdate();
     pstmt.close();
   } catch (SQLException e) {
     if (pstmt != null) {
       try {
         pstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
   }
 }


//执行存储过程。
 public static void ExecCallableSQL(Connection conn) {
   CallableStatement cstmt = null;
   try {

     cstmt=conn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
     cstmt.setInt(2, 50);
     cstmt.setInt(1, 20);
     cstmt.setInt(3, 90);
     cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
     cstmt.execute();
     int out = cstmt.getInt(4);  //获取out参数
     System.out.println("The CallableStatment TESTPROC returns:"+out);
     cstmt.close();
   } catch (SQLException e) {
     if (cstmt != null) {
       try {
         cstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
```

 版权所有 © 华为云计算技术有限公司

```
    }
  }


  /**
   * 主程序，逐步调用各静态方法。
   * @param args
  */
  public static void main(String[] args) {
    //创建数据库连接。
    Connection conn = GetConnection("tester", "password");

    //创建表。
    CreateTable(conn);

    //批插数据。
    BatchInsertData(conn);

    //执行预编译语句，更新数据。
    ExecPreparedSQL(conn);

    //执行存储过程。
    ExecCallableSQL(conn);

    //关闭数据库连接。
    try {
      conn.close();
    } catch (SQLException e) {
      e.printStackTrace();
    }

  }

}
```

示例 2 客户端内存占用过多解决

此示例主要使用setFetchSize来调整客户端内存使用，它的原理是通过数据库游标来分批获取服务器端数据，但它会加大网络交互，可能会损失部分性能。

由于游标事务内有效，故需要先关闭自动提交。

```
// 关闭掉自动提交
conn.setAutoCommit(false);
Statement st = conn.createStatement();

// 打开游标，每次获取50行数据
st.setFetchSize(50);
ResultSet rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next()){
    System.out.print("a row was returned.");
}
rs.close();

// 关闭服务器游标。
st.setFetchSize(0);
rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next()){
    System.out.print("many rows were returned.");
}
rs.close();

// Close the statement.
st.close();
```

 版权所有 © 华为云计算技术有限公司

重新执行应用 SQL

当主DN故障且40s未恢复时，DWS会自动将对应的备DN升主，使集群正常运行。备升主期间正在运行的作业会失败；备升主后启动的作业不会再受影响。如果要做到DN主备切换过程中，上层业务不感知，可参考此示例构建业务层SQL重试机制。

以下用例以gsjdbc4.jar为例。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 *
 */

class ExitHandler extends Thread {
    private Statement cancel_stmt = null;

    public ExitHandler(Statement stmt) {
        super("Exit Handler");
        this.cancel_stmt = stmt;
    }
    public void run() {
        System.out.println("exit handle");
        try {
            this.cancel_stmt.cancel();
        } catch (SQLException e) {
            System.out.println("cancel query failed.");
            e.printStackTrace();
        }
    }
}

public class SQLRetry {
   //创建数据库连接。
   public static Connection GetConnection(String username, String passwd) {
     String driver = "org.postgresql.Driver";
     String sourceURL = "jdbc:postgresql://10.131.72.136:8000/gaussdb";
     Connection conn = null;
     try {
       //加载数据库驱动。
       Class.forName(driver).newInstance();
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     try {
       //创建数据库连接。
       conn = DriverManager.getConnection(sourceURL, username, passwd);
       System.out.println("Connection succeed!");
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     return conn;
}
```

执行普通SQL语句，创建jdbc_test1表。

```
public static void CreateTable(Connection conn) {
    Statement stmt = null;
    try {
```

 版权所有 © 华为云计算技术有限公司

```
      stmt = conn.createStatement();

      // add ctrl+c handler
      Runtime.getRuntime().addShutdownHook(new ExitHandler(stmt));

      //执行普通SQL语句。
      int rc2 = stmt
          .executeUpdate("DROP TABLE if exists jdbc_test1;");

      int rc1 = stmt
          .executeUpdate("CREATE TABLE jdbc_test1(col1 INTEGER, col2 VARCHAR(10));");

      stmt.close();
    } catch (SQLException e) {
      if (stmt != null) {
        try {
          stmt.close();
        } catch (SQLException e1) {
          e1.printStackTrace();
        }
      }
      e.printStackTrace();
    }
  }
```

执行预处理语句，批量插入数据。

```
public static void BatchInsertData(Connection conn) {
    PreparedStatement pst = null;

    try {
      //生成预处理语句。
      pst = conn.prepareStatement("INSERT INTO jdbc_test1 VALUES (?,?)");
      for (int i = 0; i < 100; i++) {
       //添加参数。
        pst.setInt(1, i);
        pst.setString(2, "data " + i);
        pst.addBatch();
      }
      //执行批处理。
      pst.executeBatch();
      pst.close();
    } catch (SQLException e) {
      if (pst != null) {
        try {
          pst.close();
        } catch (SQLException e1) {
        e1.printStackTrace();
        }
      }
      e.printStackTrace();
    }
  }
```

执行预编译语句，更新数据。

```
 private static boolean QueryRedo(Connection conn){
    PreparedStatement pstmt = null;
    boolean retValue = false;
    try {
      pstmt = conn
          .prepareStatement("SELECT col1 FROM jdbc_test1 WHERE col2 = ?");

          pstmt.setString(1, "data 10");
          ResultSet rs = pstmt.executeQuery();

          while (rs.next()) {
              System.out.println("col1 = " + rs.getString("col1"));
          }
          rs.close();
```

 版权所有 © 华为云计算技术有限公司

```
      pstmt.close();
       retValue = true;
     } catch (SQLException e) {
      System.out.println("catch..... retValue " + retValue);
      if (pstmt != null) {
        try {
         pstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
        }
      }
      e.printStackTrace();
    }

     System.out.println("finish.....");
    return retValue;
  }
```

查询语句，执行失败重试，重试次数可配置。

```
 public static void ExecPreparedSQL(Connection conn) throws InterruptedException {
        int maxRetryTime = 50;
        int time = 0;
        String result = null;
        do {
            time++;
            try {
 System.out.println("time:" + time);
 boolean ret = QueryRedo(conn);
 if(ret == false){
  System.out.println("retry, time:" + time);
  Thread.sleep(10000);
  QueryRedo(conn);
 }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } while (null == result && time < maxRetryTime);

  }

  /**
   * 主程序，逐步调用各静态方法。
   * @param args
 * @throws InterruptedException
  */
  public static void main(String[] args) throws InterruptedException {
    //创建数据库连接。
    Connection conn = GetConnection("testuser", "test@123");

    //创建表。
    CreateTable(conn);

    //批插数据。
    BatchInsertData(conn);

    //执行预编译语句，更新数据。
    ExecPreparedSQL(conn);

    //关闭数据库连接。
    try {
      conn.close();
    } catch (SQLException e) {
      e.printStackTrace();
    }

  }

}
```

 版权所有 © 华为云计算技术有限公司

通过本地文件导入导出数据

在使用JAVA语言基于DWS进行二次开发时，可以使用CopyManager接口，通过流方式，将数据库中的数据导出到本地文件或者将本地文件导入数据库中，文件格式支持CSV、TEXT等格式。

样例程序如下，执行时需要加载DWS jdbc驱动。

以下用例以gsjdbc4.jar为例。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.SQLException;
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Copy{

    public static void main(String[] args)
    {
     String urls = new String("jdbc:postgresql://10.180.155.74:8000/gaussdb"); //数据库URL
     String username = new String("jack");            //用户名
     String password = new String("********");        //密码
     String tablename = new String("migration_table"); //定义表信息
     String tablename1 = new String("migration_table_1"); //定义表信息
     String driver = "org.postgresql.Driver";
     Connection conn = null;

     try {
           Class.forName(driver);
           conn = DriverManager.getConnection(urls, username, password);
         } catch (ClassNotFoundException e) {
             e.printStackTrace(System.out);
         } catch (SQLException e) {
             e.printStackTrace(System.out);
         }
```

执行数据导入导出：

```
     // 将migration_table查询结果导出到本地文件d:/data.txt
     try {
    copyToFile(conn, "d:/data.txt", "(SELECT * FROM migration_table)");
  } catch (SQLException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  }
     //将d:/data.txt中的数据导入到migration_table_1中。
     try {
    copyFromFile(conn, "d:/data.txt", migration_table_1);
  } catch (SQLException e) {
 // TODO Auto-generated catch block
       e.printStackTrace();
} catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

     // 将migration_table_1中的数据导出到本地文件d:/data1.txt
     try {
    copyToFile(conn, "d:/data1.txt", migration_table_1);
  } catch (SQLException e) {
```

 版权所有 © 华为云计算技术有限公司

```
 // TODO Auto-generated catch block
 e.printStackTrace();
  } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
    }

 public static void copyFromFile(Connection connection, String filePath, String tableName)
        throws SQLException, IOException {

    FileInputStream fileInputStream = null;

    try {
        CopyManager copyManager = new CopyManager((BaseConnection)connection);
        fileInputStream = new FileInputStream(filePath);
        copyManager.copyIn("COPY " + tableName + " FROM STDIN", fileInputStream);
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

 public static void copyToFile(Connection connection, String filePath, String tableOrQuery)
         throws SQLException, IOException {

     FileOutputStream fileOutputStream = null;

     try {
         CopyManager copyManager = new CopyManager((BaseConnection)connection);
         fileOutputStream = new FileOutputStream(filePath);
         copyManager.copyOut("COPY " + tableOrQuery + " TO STDOUT", fileOutputStream);
     } finally {
         if (fileOutputStream != null) {
             try {
                 fileOutputStream.close();
             } catch (IOException e) {
                 e.printStackTrace();
             }
         }
     }
 }
}
```

从 MySQL 向 DWS 进行数据迁移

下面示例演示如何通过CopyManager从mysql向DWS进行数据迁移的过程。

以下用例以gsjdbc4.jar为例。

```
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Migration{

    public static void main(String[] args) {
```

 版权所有 © 华为云计算技术有限公司

```
    String url = new String("jdbc:postgresql://10.180.155.74:8000/gaussdb"); //数据库URL
    String user = new String("jack");            //DWS用户名
    String pass = new String("********");            //DWS密码
    String tablename = new String("migration_table"); //定义表信息
    String delimiter = new String("|");              //定义分隔符
    String encoding = new String("UTF8");            //定义字符集
    String driver = "org.postgresql.Driver";
    StringBuffer buffer = new StringBuffer();       //定义存放格式  化数据的缓存

    try {
        //获取源数据库查询结果集
        ResultSet rs = getDataSet();

        //遍历结果集，逐行获取记录
        //将每条记录中各字段值，按指定分隔符分割，由换行符结束，拼成一个字符串
        //把拼成的字符串，添加到缓存buffer
        while (rs.next()) {
            buffer.append(rs.getString(1) + delimiter
                    + rs.getString(2) + delimiter
                    + rs.getString(3) + delimiter
                    + rs.getString(4)
                    + "\n");
        }
        rs.close();

        try {
            //建立目标数据库连接
            Class.forName(driver);
            Connection conn = DriverManager.getConnection(url, user, pass);
            BaseConnection baseConn = (BaseConnection) conn;
            baseConn.setAutoCommit(false);

            //初始化表信息
            String sql = "Copy " + tablename + " from STDIN DELIMITER " + "'" + delimiter + "'" + "
ENCODING " + "'" + encoding + "'";

            //提交缓存buffer中的数据
            CopyManager cp = new CopyManager(baseConn);
            StringReader reader = new StringReader(buffer.toString());
            cp.copyIn(sql, reader);
            baseConn.commit();
            reader.close();
            baseConn.close();
        } catch (ClassNotFoundException e) {
            e.printStackTrace(System.out);
        } catch (SQLException e) {
            e.printStackTrace(System.out);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
```

从源数据库返回查询结果集：

```
  private static ResultSet getDataSet() {
    ResultSet rs = null;
    try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        Connection conn = DriverManager.getConnection("jdbc:mysql://10.119.179.227:3306/jack?
useSSL=false&allowPublicKeyRetrieval=true", "jack", "********");
        Statement stmt = conn.createStatement();
        rs = stmt.executeQuery("select * from migration_table");
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

 版权所有 © 华为云计算技术有限公司

```
        return rs;
    }
}
```

13.2.8 应用端加工 RoaringBitmap 结果集并入库 DWS 开发示例

DWS 自8.1.3版本后支持位图功能（RoaringBitmap），在使用JAVA语言基于DWS进行二次开发时，可以使用CopyManager接口，实现小批量RoaringBitmap的数据入库至DWS。

说明

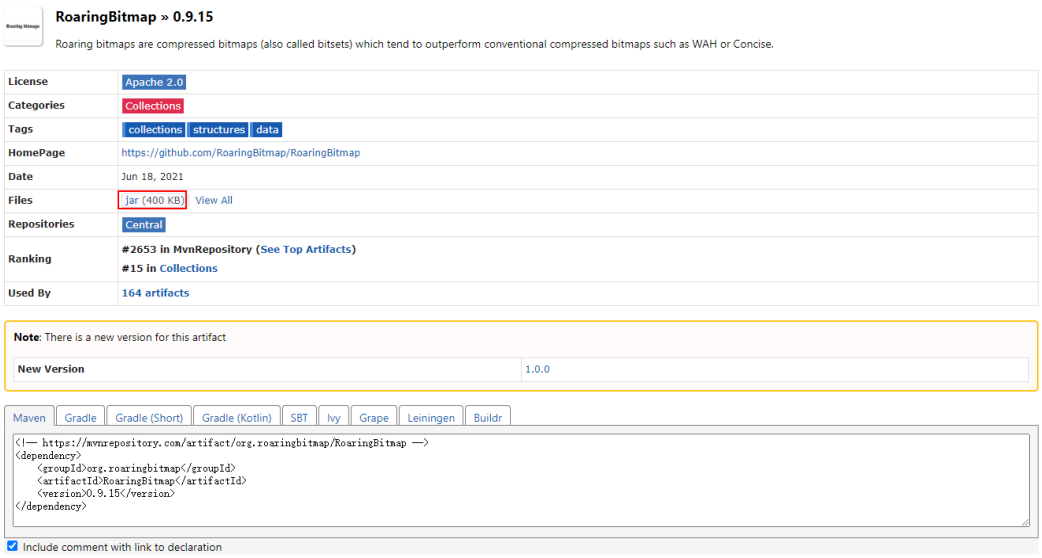
- 如针对大批量数据入库，需要在应用侧增加算力进行优化，否则会影响入库性能。
- Java生成的64位Roaringbitmap不支持入库。

加工 RoaringBitmap

步骤1 访问Maven下载开源的RoaringBitmap的jar包，推荐下载0.9.15版本。

POM文件依赖项配置如下：

```
<dependencies>
 <dependency>
 <groupId>org.roaringbitmap</groupId>
 <artifactId>RoaringBitmap</artifactId>
 <version>0.9.15</version>
 </dependency>
 </dependencies>
```



步骤2 调用jar包实现Roaringbitmap的转换。

大致过程是声明一个RoaringBitmap，调用add方法，将要转化的int转化成Roaringbitmap类型，再对数据进行序列化。代码示例如下：

```
RoaringBitmap rr2 = new RoaringBitmap ();
for (int i = 1; i < 10000000;  i++)  {
      rr2.add(i);
}
ByteArrayOutputStream a = new ByteArrayOutputStream();
DataOutputStream b = new DataOutputStream(a);
rr2.serialize(b);
```

----结束

 版权所有 © 华为云计算技术有限公司

数据入库

调用CopyManager进行入库，实现数据不落地，小批量RoaringBitmap入库。

```
//以下用例以gsjdbc4.jar为例。
package rb_demo;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;
import org.roaringbitmap.RoaringBitmap;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class rb_demo {

    private static String hexStr =  "0123456789ABCDEF";

    public static String bytesToHex(byte[] bytes) {
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < bytes.length; i++) {
            String hex = Integer.toHexString(bytes[i] & 0xFF);
            if (hex.length() < 2) {
                sb.append(0);
            }
            sb.append(hex);
        }
        return sb.toString();
    }

    public static Connection GetConnection(String username, String passwd) {
        String driver = "org.postgresql.Driver";
        String sourceURL = "jdbc:postgresql://10.185.180.161:8000/gaussdb";  //数据库URL
        Connection conn = null;
        try {
            // 加载数据库驱动。
            Class.forName(driver).newInstance();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        try {
            // 创建数据库连接。
            conn = DriverManager.getConnection(sourceURL, username, passwd);
            System.out.println("Connection succeed!");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        return conn;
    }

    public static void main(String[] args) throws IOException {

        RoaringBitmap rr2 = new RoaringBitmap();

        for (int i = 1; i < 10000000; i++) {
```

 版权所有 © 华为云计算技术有限公司

```
            rr2.add(i);
        }

        ByteArrayOutputStream a = new ByteArrayOutputStream();

        DataOutputStream b = new DataOutputStream(a);
        rr2.serialize(b);

        Connection conn = GetConnection("test", "password");  //数据库用户名、密码
        Statement pstmt = null;
        try {
            conn.setAutoCommit(true);
            pstmt = conn.createStatement();

            pstmt.execute("drop table if exists t_rb");
            pstmt.execute("create table t_rb(c1 int, c2 roaringbitmap) distribute by hash (c1);");

            StringReader sr = null;
            CopyManager cm = null;
            cm = new CopyManager((BaseConnection) conn);

            String delimiter = "|";
            StringBuffer tuples = new StringBuffer();
            tuples.append("1" + delimiter + "\\x" + bytesToHex(a.toByteArray()));


            StringBuffer sb = new StringBuffer();
            sb.append(tuples.toString());

            sr = new StringReader(tuples.toString());
            String sql = "copy t_rb from STDIN with (delimiter '|', NOESCAPING)";

            long rows = cm.copyIn(sql, sr);// 执行copy入库

            pstmt.close();
        } catch (SQLException e) {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }
}
```

13.2.9 JDBC 接口参考

JDBC接口是一套提供给用户的API方法，本节将对部分常用接口做具体描述，若涉及其他接口可参考JDK1.6（软件包）/JDBC4.0中相关内容。

java.sql.Connection

java.sql.Connection是数据库连接接口。

表 13-7 对 java.sql.Connection 接口的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes
commit()	void	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
createStatement()	Statement	Yes
getAutoCommit()	boolean	Yes
getClientInfo()	Properties	Yes
getClientInfo(String name)	String	Yes
getTransactionIsolation()	int	Yes
isClosed()	boolean	Yes
isReadOnly()	boolean	Yes
prepareStatement(String sql)	PreparedStatement	Yes
rollback()	void	Yes
setAutoCommit(boolean autoCommit)	void	Yes
setClientInfo(Properties properties)	void	Yes
setClientInfo(String name,String value)	void	Yes

须知

接口内部默认使用自动提交模式，若通过setAutoCommit(false)关闭自动提交，将会导致后面执行的语句都受到显式事务包裹，数据库中不支持事务中执行的语句不能在此模式下执行。

java.sql.CallableStatement

java.sql.CallableStatement是存储过程执行接口。

表 13-8 对 java.sql.CallableStatement 的支持情况

方法名	返回值类型	支持JDBC 4
registerOutParameter(int parameterIndex, int type)	void	Yes
wasNull()	boolean	Yes
getString(int parameterIndex)	String	Yes
getBoolean(int parameterIndex)	boolean	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getByte(int parameterIndex)	byte	Yes
getShort(int parameterIndex)	short	Yes
getInt(int parameterIndex)	int	Yes
getLong(int parameterIndex)	long	Yes
getFloat(int parameterIndex)	float	Yes
getDouble(int parameterIndex)	double	Yes
getBigDecimal(int parameterIndex)	BigDecimal	Yes
getBytes(int parameterIndex)	byte[]	Yes
getDate(int parameterIndex)	Date	Yes
getTime(int parameterIndex)	Time	Yes
getTimestamp(int parameterIndex)	Timestamp	Yes
getObject(int parameterIndex)	Object	Yes

说明

- 不允许含有OUT参数的语句执行批量操作。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows , setFetchSize。
- 以下方法是从java.sql.PreparedStatement继承而来：addBatch，clearParameters，execute，executeQuery，executeUpdate，getMetaData，setBigDecimal，setBoolean，setByte，setBytes，setDate，setDouble，setFloat，setInt，setLong，setNull，setObject，setString，setTime，setTimestamp 。

java.sql.DatabaseMetaData

java.sql.DatabaseMetaData是数据库对象定义接口。

表 13-9 对 java.sql.DatabaseMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	ResultSet	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	ResultSet	Yes
getTableTypes()	ResultSet	Yes
getUserName()	String	Yes
isReadOnly()	boolean	Yes
nullsAreSortedHigh()	boolean	Yes
nullsAreSortedLow()	boolean	Yes
nullsAreSortedAtStart()	boolean	Yes
nullsAreSortedAtEnd()	boolean	Yes
getDatabaseProductName()	String	Yes
getDatabaseProductVersion()	String	Yes
getDriverName()	String	Yes
getDriverVersion()	String	Yes
getDriverMajorVersion()	int	Yes
getDriverMinorVersion()	int	Yes
usesLocalFiles()	boolean	Yes
usesLocalFilePerTable()	boolean	Yes
supportsMixedCaseIdentifiers()	boolean	Yes
storesUpperCaseIdentifiers()	boolean	Yes
storesLowerCaseIdentifiers()	boolean	Yes
supportsMixedCaseQuotedIdentifiers()	boolean	Yes
storesUpperCaseQuotedIdentifiers()	boolean	Yes
storesLowerCaseQuotedIdentifiers()	boolean	Yes
storesMixedCaseQuotedIdentifiers()	boolean	Yes
supportsAlterTableWithAddColumn()	boolean	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
supportsAlterTableWithDropColumn()	boolean	Yes
supportsColumnAliasing()	boolean	Yes
nullPlusNonNullIsNull()	boolean	Yes
supportsConvert()	boolean	Yes
supportsConvert(int fromType, int toType)	boolean	Yes
supportsTableCorrelationNames()	boolean	Yes
supportsDifferentTableCorrelationNames()	boolean	Yes
supportsExpressionsInOrderBy()	boolean	Yes
supportsOrderByUnrelated()	boolean	Yes
supportsGroupBy()	boolean	Yes
supportsGroupByUnrelated()	boolean	Yes
supportsGroupByBeyondSelect()	boolean	Yes
supportsLikeEscapeClause()	boolean	Yes
supportsMultipleResultSets()	boolean	Yes
supportsMultipleTransactions()	boolean	Yes
supportsNonNullableColumns()	boolean	Yes
supportsMinimumSQLGrammar()	boolean	Yes
supportsCoreSQLGrammar()	boolean	Yes
supportsExtendedSQLGrammar()	boolean	Yes
supportsANSI92EntryLevelSQL()	boolean	Yes
supportsANSI92IntermediateSQL()	boolean	Yes
supportsANSI92FullSQL()	boolean	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
supportsIntegrityEnhancementFacility()	boolean	Yes
supportsOuterJoins()	boolean	Yes
supportsFullOuterJoins()	boolean	Yes
supportsLimitedOuterJoins()	boolean	Yes
isCatalogAtStart()	boolean	Yes
supportsSchemasInDataManipulation()	boolean	Yes
supportsSavepoints()	boolean	Yes
supportsResultSetHoldability(int holdability)	boolean	Yes
getResultSetHoldability()	int	Yes
getDatabaseMajorVersion()	int	Yes
getDatabaseMinorVersion()	int	Yes
getJDBCMajorVersion()	int	Yes
getJDBCMinorVersion()	int	Yes

java.sql.Driver

java.sql.Driver是数据库驱动接口。

表 13-10 对 java.sql.Driver 的支持情况

方法名	返回值类型	支持JDBC 4
acceptsURL(String url)	boolean	Yes
connect(String url, Properties info)	Connection	Yes
jdbcCompliant()	boolean	Yes
getMajorVersion()	int	Yes
getMinorVersion()	int	Yes

java.sql.PreparedStatement

java.sql.PreparedStatement是预处理语句接口。

 版权所有 © 华为云计算技术有限公司

表 13-11 对 java.sql.PreparedStatement 的支持情况

方法名	返回值类型	支持JDBC 4
clearParameters()	void	Yes
execute()	boolean	Yes
executeQuery()	ResultSet	Yes
executeUpdate()	int	Yes
getMetaData()	ResultSetMetaData	Yes
setBoolean(int parameterIndex, boolean x)	void	Yes
setBigDecimal(int parameterIndex, BigDecimal x)	void	Yes
setByte(int parameterIndex, byte x)	void	Yes
setBytes(int parameterIndex, byte[] x)	void	Yes
setDate(int parameterIndex, Date x)	void	Yes
setDouble(int parameterIndex, double x)	void	Yes
setFloat(int parameterIndex, float x)	void	Yes
setInt(int parameterIndex, int x)	void	Yes
setLong(int parameterIndex, long x)	void	Yes
setNString(int parameterIndex, String value)	void	Yes
setShort(int parameterIndex, short x)	void	Yes
setString(int parameterIndex, String x)	void	Yes
addBatch()	void	Yes
executeBatch()	int[]	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
clearBatch()	void	Yes

说明

- addBatch()、execute()必须在clearBatch()之后才能执行。
- 调用executeBatch()方法并不会清除batch。用户必须显式使用clearBatch()清除 。
- 在添加了一个batch的绑定变量后，用户若想重用这些值(再次添加一个batch)，无需再次使用set*()方法 。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows, setFetchSize。

java.sql.ResultSet

java.sql.ResultSet是执行结果集接口。

表 13-12 对 java.sql.ResultSet 的支持情况

方法名	返回值类型	支持JDBC 4
findColumn(String columnLabel)	int	Yes
getBigDecimal(int columnIndex)	BigDecimal	Yes
getBigDecimal(String columnLabel)	BigDecimal	Yes
getBoolean(int columnIndex)	boolean	Yes
getBoolean(String columnLabel)	boolean	Yes
getByte(int columnIndex)	byte	Yes
getBytes(int columnIndex)	byte[]	Yes
getByte(String columnLabel)	byte	Yes
getBytes(String columnLabel)	byte[]	Yes
getDate(int columnIndex)	Date	Yes
getDate(String columnLabel)	Date	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getDouble(int columnIndex)	double	Yes
getDouble(String columnLabel)	double	Yes
getFloat(int columnIndex)	float	Yes
getFloat(String columnLabel)	float	Yes
getInt(int columnIndex)	int	Yes
getInt(String columnLabel)	int	Yes
getLong(int columnIndex)	long	Yes
getLong(String columnLabel)	long	Yes
getShort(int columnIndex)	short	Yes
getShort(String columnLabel)	short	Yes
getString(int columnIndex)	String	Yes
getString(String columnLabel)	String	Yes
getTime(int columnIndex)	Time	Yes
getTime(String columnLabel)	Time	Yes
getTimestamp(int columnIndex)	Timestamp	Yes
getTimestamp(String columnLabel)	Timestamp	Yes
isAfterLast()	boolean	Yes
isBeforeFirst()	boolean	Yes
isFirst()	boolean	Yes
next()	boolean	Yes

 版权所有 © 华为云计算技术有限公司

说明

- 一个Statement不能有多个处于“open”状态的ResultSet。
- 用于遍历结果集(ResultSet)的游标(Cursor)在被提交后不能保持“open”的状态。

java.sql.ResultSetMetaData

java.sql.ResultSetMetaData是对ResultSet对象相关信息的具体描述。

表 13-13 对 java.sql.ResultSetMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getColumnCount()	int	Yes
getColumnName(int column)	String	Yes
getColumnType(int column)	int	Yes
getColumnTypeName(int column)	String	Yes

java.sql.Statement

java.sql.Statement是SQL语句接口。

表 13-14 对 java.sql.Statement 的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes
execute(String sql)	boolean	Yes
executeQuery(String sql)	ResultSet	Yes
executeUpdate(String sql)	int	Yes
getConnection()	Connection	Yes
getResultSet()	ResultSet	Yes
getQueryTimeout()	int	Yes
getUpdateCount()	int	Yes
isClosed()	boolean	Yes
setQueryTimeout(int seconds)	void	Yes
setFetchSize(int rows)	void	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
cancel()	void	Yes

说明

通过setFetchSize可以减少结果集在客户端的内存占用情况。它的原理是通过将结果集打包成游标，然后分段处理，所以会加大数据库与客户端的通信量，会有性能损耗。

由于数据库游标是事务内有效，所以，在设置setFetchSize的同时，需要将连接设置为非自动提交模式，setAutoCommit(false)。同时在业务数据需要持久化到数据库中时，在连接上执行提交操作。

javax.sql.ConnectionPoolDataSource

javax.sql.ConnectionPoolDataSource是数据源连接池接口。

表 13-15 对 javax.sql.ConnectionPoolDataSource 的支持情况

方法名	返回值类型	支持JDBC 4
getLoginTimeout()	int	Yes
getLogWriter()	PrintWriter	Yes
getPooledConnection()	PooledConnection	Yes
getPooledConnection(String user,String password)	PooledConnection	Yes
setLoginTimeout(int seconds)	void	Yes
setLogWriter(PrintWriter out)	void	Yes

javax.sql.DataSource

javax.sql.DataSource是数据源接口。

表 13-16 对 javax.sql.DataSource 接口的支持情况

方法名	返回值类型	支持JDBC 4
getConnection()	Connection	Yes
getConnection(String username,String password)	Connection	Yes
getLoginTimeout()	int	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getLogWriter()	PrintWriter	Yes
setLoginTimeout(int seconds)	void	Yes
setLogWriter(PrintWriter out)	void	Yes

javax.sql.PooledConnection

javax.sql.PooledConnection是由连接池创建的连接接口。

表 13-17 对 javax.sql.PooledConnection 的支持情况

方法名	返回值类型	支持JDBC 4
addConnectionEventListener (ConnectionEventListener listener)	void	Yes
close()	void	Yes
getConnection()	Connection	Yes
removeConnectionEventListener (ConnectionEventListener listener)	void	Yes
addStatementEventListener (StatementEventListener listener)	void	Yes
removeStatementEventListener (StatementEventListener listener)	void	Yes

javax.naming.Context

javax.naming.Context是连接配置的上下文接口。

表 13-18 对 javax.naming.Context 的支持情况

方法名	返回值类型	支持JDBC 4
bind(Name name, Object obj)	void	Yes
bind(String name, Object obj)	void	Yes
lookup(Name name)	Object	Yes
lookup(String name)	Object	Yes
rebind(Name name, Object obj)	void	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
rebind(String name, Object obj)	void	Yes
rename(Name oldName, Name newName)	void	Yes
rename(String oldName, String newName)	void	Yes
unbind(Name name)	void	Yes
unbind(String name)	void	Yes

javax.naming.spi.InitialContextFactory

javax.naming.spi.InitialContextFactory是初始连接上下文工厂接口。

表 13-19 对 javax.naming.spi.InitialContextFactory 的支持情况

方法名	返回值类型	支持JDBC 4
getInitialContext(Hashtable<?,?> environment)	Context	Yes

CopyManager

CopyManager是DWS JDBC驱动中提供的一个API接口类，用于批量向DWS集群中导入数据。

CopyManager的继承关系

CopyManager类位于org.postgresql.copy Package中，继承自java.lang.Object类，该类的声明如下：

```
public class CopyManager
extends Object
```

构造方法

```
public CopyManager(BaseConnection connection)
throws SQLException
```

常用方法

表 13-20 CopyManager 常用方法

返回值	方法	描述	throws
CopyIn	copyIn(String sql)	-	SQLException

 版权所有 © 华为云计算技术有限公司

返回值	方法	描述	throws
long	copyIn(String sql, InputStream from)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, InputStream from, int bufferSize)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from, int bufferSize)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
CopyOut	copyOut(String sql)	-	SQLException
long	copyOut(String sql, OutputStream to)	将一个COPY TO STDOUT的结果集从数据库发送到OutputStream类中。	SQLException,IOException
long	copyOut(String sql, Writer to)	将一个COPY TO STDOUT的结果集从数据库发送到Writer类中。	SQLException,IOException

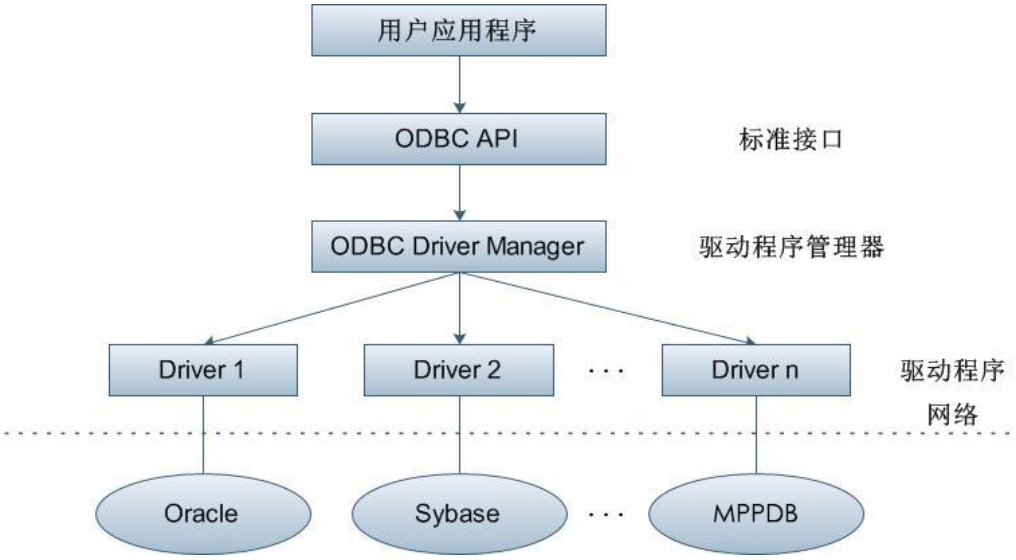
13.3 基于 ODBC 开发

ODBC（Open Database Connectivity，开放数据库互连）是由Microsoft公司基于X/OPEN CLI提出的用于访问数据库的应用程序编程接口。应用程序通过ODBC提供的API与数据库进行交互，在避免了应用程序直接操作数据库系统的同时，增强了应用程序的可移植性、扩展性和可维护性。

ODBC的系统结构参见**图13-2**。

 版权所有 © 华为云计算技术有限公司

图 13-2 ODBC 系统结构



DWS目前在以下环境中提供对ODBC3.5的支持。

表 13-21 ODBC 支持平台

操作系统	平台
SUSE Linux Enterprise Server 11 SP1/SP2/SP3/SP4 SUSE Linux Enterprise Server 12 及SP1/SP2/SP3/SP5	x86_64位
Red Hat Enterprise Linux 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4/7.5	x86_64位
Red Hat Enterprise Linux 7.5	ARM64位
CentOS 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4	x86_64位
CentOS 7.6	ARM64位
EulerOS 2.0 SP2/SP3	x86_64位
EulerOS 2.0 SP8	ARM64位
中标麒麟 7.5/7.6	ARM64位
Oracle Linux R7U4	x86_64位
Windows 7	32位
Windows 7	64位
Windows Server 2008	32位
Windows Server 2008	64位

以上操作系统平台是指ODBC程序所在的操作系统平台，可以与数据库部署的操作系统平台不同。

 版权所有 © 华为云计算技术有限公司

UNIX/Linux系统下的驱动程序管理器主要有unixODBC和iODBC，在这选择驱动管理器unixODBC-2.3.0作为连接数据库的组件。

Windows系统自带ODBC驱动程序管理器，在控制面板->管理工具中可以找到数据源（ODBC）选项。

说明

当前数据库ODBC驱动基于开源版本，对于DWS的数据类型，tinyint、smalldatetime、nvarchar2在获取数据类型的时候，可能会出现不兼容。

13.3.1 ODBC 包及依赖的库和头文件

从管理控制台下载ODBC的软件包。

请参见《数据仓库服务用户指南》的“下载JDBC或ODBC驱动”章节。

Linux 下的 ODBC 包

从软件包中获取，包名为dws_8.x.x_odbc_driver_for_xxx_xxx.zip。Linux环境下，开发应用程序要用到unixODBC提供的头文件（sql.h、sqlext.h等）和库libodbc.so。这些头文件和库可从unixODBC-2.3.0的安装包中获得。

Windows 下的 ODBC 包

从软件包中获取，包名为dws_8.x.x_odbc_driver_for_windows.zip。Windows环境下，开发应用程序用到的相关头文件和库文件都由系统自带。

13.3.2 Linux 下配置数据源

将DWS提供的ODBC DRIVER（psqlodbcw.so）配置到数据源中便可使用。配置数据源需要配置“odbc.ini”和“odbcinst.ini”两个文件（在编译安装unixODBC过程中生成且默认放在etc目录下），并在服务器端进行配置。

操作步骤

步骤1 获取unixODBC源码包。

获取参考地址：**https://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download**

步骤2 目前不支持unixODBC-2.2.1版本。以unixODBC-2.3.0版本为例，在客户端执行如下命令安装unixODBC。安装时通过--prefix=[your_path]指定安装目录，生成数据源文件到“[your_path]/etc”目录下，库文件生成在“[your_path]/lib”目录。

```
tar zxvf unixODBC-2.3.0.tar.gz
cd unixODBC-2.3.0
#修改configure文件(如果不存在，那么请修改configure.ac)，找到LIB_VERSION
#将它的值修改为"1:0:0"，这样将编译出*.so.1的动态库，与psqlodbcw.so的依赖关系相同。
vim configure

./configure --enable-gui=no --prefix=[your_path] #如果要在ARM服务器上编译，请追加一个configure参数：
--build=aarch64-unknown-linux-gnu
make
make install
```

安装unixODBC。如果机器上已经安装了其他版本的unixODBC，可以直接覆盖安装。

步骤3 替换客户端DWS驱动程序。

 版权所有 © 华为云计算技术有限公司

将dws_8.x.x_odbc_driver_for_xxx_xxx.zip解压，在“/dws_8.x.x_odbc_driver_for_xxx_xxx/odbc/lib”目录下得到“psqlodbcw.la”和“psqlodbcw.so”两个文件。

步骤4 配置数据源。

1. 配置ODBC驱动文件。

   在“[your_path]/etc/odbcinst.ini”文件中追加以下内容。

   ```
   [GaussMPP]
   Driver64=[your_path]/lib/odbc/psqlodbcw.so
   setup=[your_path]/lib/odbc/psqlodbcw.so
   ```

   odbcinst.ini文件中的配置参数说明如表13-22所示。

   表 13-22 odbcinst.ini 文件配置参数

参数	描述	示例
[DriverName]	驱动器名称，对应数据源DSN中的驱动名。	[DRIVER_N]
Driver64	驱动动态库的路径。	Driver64=/xxx/odbc/lib/odbc/psqlodbcw.so
setup	驱动安装路径，与Driver64中动态库的路径一致。	setup=/xxx/odbc/lib/odbc/psqlodbcw.so

2. 配置数据源文件。

   在“[your_path]/etc/odbc.ini ”文件中追加以下内容。

   ```
   [MPPODBC]
   Driver=GaussMPP
   Servername=10.10.0.13（数据库Server IP）
   Database=gaussdb （数据库名）
   Username=dbadmin （数据库用户名）
   Password= （数据库用户密码）
   Port=8000（数据库监听端口）
   Sslmode=allow
   ```

   odbc.ini文件配置参数说明如表13-23所示。

   表 13-23 odbc.ini 文件配置参数

参数	描述	示例
[DSN]	数据源的名称。	[MPPODBC]
Driver	驱动名，对应odbcinst.ini中的DriverName。	Driver=DRIVER_N
Servername	服务器的IP地址。	Servername=10.145.130.26
Database	要连接的数据库的名称。	Database=gaussdb
Username	数据库用户名称。	Username=dbadmin

 版权所有 © 华为云计算技术有限公司

参数	描述	示例
Password	数据库用户密码。	Password= **说明** ODBC驱动本身已经对内存密码进行过清理，以保证用户密码在连接后不会再在内存中保留。 但是如果配置了此参数，由于UnixODBC对数据源文件等进行缓存，可能导致密码长期保留在内存中。 推荐在应用程序连接时，将密码传递给相应API，而非写在数据源配置文件中。同时连接成功后，应当及时清理保存密码的内存段。
Port	服务器的端口号。	Port=8000
Sslmode	开启SSL模式。	Sslmode=allow
UseServerSidePrepare	是否开启数据库端扩展查询协议。 可选值0或1，默认为1，表示打开扩展查询协议。	UseServerSidePrepare=1
UseBatchProtocol	是否开启批量查询协议（打开可提高DML性能）；可选值0或者1，默认为1。 当此值为0时，不使用批量查询协议（主要用于与早期数据库版本通信兼容）。 当此值为1，并且数据库support_batch_bind参数存在且为on时，将打开批量查询协议。	UseBatchProtocol=1
ConnectionExtraInfo	GUC参数connection_info（参见**connection_info**）中显示驱动部署路径和进程属主用户的开关。	ConnectionExtraInfo=1 **说明** 默认值为1。当设置为0时，ODBC驱动会将当前驱动的名称、驱动版本上报到数据库中；当设置为1时，ODBC驱动会将当前驱动的名称、部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。

版权所有 © 华为云计算技术有限公司

参数	描述	示例
ForExtensionConnector	ETL工具性能优化参数，可进行内存优化，降低对端的CN内存占用，避免因CN内存使用过多导致系统不稳定。 可选值0或者1，默认为0，表示不开启优化项。 请勿在数据库系统之外的其他业务中配置此参数，以免影响业务的正确性。	ForExtensionConnector=1
KeepDisallowPremature	当UseDeclareFetch=1时，应用程序调用SQLPrepare后调用SQLNumResultCols、SQLDescribeCol或SQLColAttribute获取结果集列信息时，SQL语句中的游标是否具有with hold属性。 可选值0或者1，0表示具有with hold属性，1表示不具有with hold属性，默认为0。	KeepDisallowPremature=1 **说明** UseServerSidePrepare=1时，KeepDisallowPremature参数不生效，使用时需要指定UseServerSidePrepare为0，例如： UseDeclareFetch=1 KeepDisallowPremature=1 UseServerSidePrepare=0

其中关于sslmode的选项的允许值，具体信息见下表：

表 13-24 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机构签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。DWS不支持此模式。

步骤5 SSL模式

 版权所有 © 华为云计算技术有限公司

如果需要使用SSL证书连接，那么请将DWS安装包中的SSLCERT的证书包解压，在shell环境下，执行“source sslcert_env.sh”，即在**当前会话**完成证书的默认位置的部署。

或者手动声明如下环境变量，同时保证client.key*系列文件为600权限：

```
export PGSSLCERT="/YOUR/PATH/OF/client.crt" #请修改该路径到client.crt的绝对路径
export PGSSLKEY="/YOUR/PATH/OF/client.key"  #请修改该路径到client.key的绝对路径
```

同时将数据源中的Sslmode选项调整至“verify-ca”。

步骤6 将客户端所在主机的IP网段加入DWS的安全组规则，确保客户端主机与DWS网络互通。

步骤7 配置环境变量。

```
vim ~/.bashrc
```

在配置文件中追加以下内容。

```
export LD_LIBRARY_PATH=[your_path]/lib/:$LD_LIBRARY_PATH
export ODBCSYSINI=[your_path]/etc
export ODBCINI=[your_path]/etc/odbc.ini
```

说明

麒麟OS环境中，此处不建议追加LD_LIBRARY_PATH，可能造成libssl.so动态库冲突。集群9.1.0最新版本中已增加rpath，不需要LD_LIBRARY_PATH即可找到依赖。

步骤8 执行如下命令使设置生效。

```
source ~/.bashrc
```

----结束

测试数据源配置

执行isql -v GaussODBC(数据源名称)命令。

- 如果显示如下信息，表明配置正确，连接成功。

```
+----------------------------------------+
| Connected!                             |
|                                        |
| sql-statement                          |
| help [tablename]                       |
| quit                                   |
|                                        |
+----------------------------------------+
SQL>
```

- 如果显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

常见问题处理

- [UnixODBC][Driver Manager]Can't open lib 'xxx/xxx/psqlodbcw.so' : file not found.

  此问题的可能原因：

  - odbcinst.ini文件中配置的路径不正确

    确认的方法：执行'ls'查询错误信息中的路径，以确保该psqlodbcw.so文件存在，同时具有执行权限。

  - psqlodbcw.so的依赖库不存在，或者不在系统环境变量中

    确认的办法：**ldd**错误信息中的路径，如果是缺少libodbc.so.1等UnixODBC的库，那么按照“操作步骤”中的方法重新配置UnixODBC，并确保它的安装

 版权所有 © 华为云计算技术有限公司

路径下的lib目录添加到了LD_LIBRARY_PATH中；如果是缺少其他库，请将ODBC驱动包中的lib目录添加到LD_LIBRARY_PATH中。或者将psqlodbcw.so的依赖库放到psqlodbcw.so的rpath对应的路径中，通过readelf -d可查看rpath。

- [UnixODBC]connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

    如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

  - 防火墙及网闸设备

    请确认防火墙设置，将数据库的通信端口添加到可信端口中。

    如果有网闸设备，请确认相关的设置。

- [unixODBC]The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项。

  解决办法：

  请配置该选项至allow或以上选项。此配置的更多信息，见**表13-24**。

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因：

  使用了SSL加密的“verify-full”选项，驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。

  解决办法：

  碰到此问题可以使用“verify-ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- Driver's SQLAllocHandle on SQL_HANDLE_DBC failed

  此问题的可能原因：

  可执行文件（比如UnixODBC的isql，以下都以isql为例）与数据库驱动（psqlodbcw.so）依赖于不同的odbc的库版本：libodbc.so.1或者libodbc.so.2。此问题可以通过如下方式确认：

  ```
  ldd `which isql` | grep odbc
  ldd psqlodbcw.so | grep odbc
  ```

  这时，如果输出的libodbc.so最后的后缀数字不同或者指向不同的磁盘物理文件，那么基本就可以断定是此问题。isql与psqlodbcw.so都会要求加载libodbc.so，这时如果它们加载的是不同的物理文件，便会导致两套完全同名的函数列表，同时出现在同一个可见域里（UnixODBC的libodbc.so.*的函数导出列表完全一致），产生冲突，无法加载数据库驱动。

  解决办法：

  确定一个要使用的UnixODBC，然后卸载另外一个（比如卸载库版本号为.so.2的UnixODBC），然后将剩下的.so.1的库，新建一个同名但是后缀为.so.2的软链接，便可解决此问题。

- FATAL: Forbid remote connection with trust method!

  由于安全原因，数据库CN禁止集群内部其他节点无认证接入。

 版权所有 © 华为云计算技术有限公司

如果要在集群内部访问CN，请将ODBC程序部署在CN所在机器，服务器地址使用"127.0.0.1"。建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。

- [unixODBC][Driver Manager]Invalid attribute value

  在使用SQL on other GaussDB功能时碰到此问题，有可能是unixODBC的版本并非推荐版本，建议通过“odbcinst --version”命令排查环境中的unixODBC版本。

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

  说明

  - 数据库并不存储用户口令，只存储用户口令的哈希码。
  - 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
  - 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
  - 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

  要解决该问题，可以更新用户口令；或者新建一个用户，赋予同等权限，使用新用户连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

13.3.3 Windows 下配置数据源

Windows操作系统自带ODBC数据源管理器，无需用户手动安装管理器便可直接进行配置。

操作步骤

步骤1 替换客户端DWS驱动程序。

将GaussDB-9.1.0-Windows-Odbc.tar.gz解压后，根据需要，双击psqlodbc.msi（32位）或者psqlodbc_x64.msi（64 位）进行驱动安装。

步骤2 打开驱动管理器。

在配置数据源时，请使用对应的驱动管理器（假设操作系统安装盘符为C:盘，如果是其他盘符，请对路径做相应修改）：

- **64**位操作系统上进行**32**位程序开发，安装**32**位驱动程序后，使用**32**位的驱动管理器：C:\Windows**SysWOW64**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

  说明

  WoW64的全称是"Windows 32-bit on Windows 64-bit"，C:\Windows\SysWOW64\存放的是64位系统上的32位运行环境。

 版权所有 © 华为云计算技术有限公司

- **64**位操作系统上进行**64**位程序开发，安装**64**位驱动程序后，使用**64**位的驱动管理器：C:\Windows**System32**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

  说明

  C:\Windows\System32\存放的是与操作系统一致的运行环境，具体的技术信息请查阅Windows的相关技术文档。

- 32位操作系统请使用：C:\Windows\System32\odbcad32.exe

  或者单击计算机->控制面板->管理工具->数据源(ODBC)打开驱动管理器。

步骤3 配置数据源。

在打开的驱动管理器上，选择用户DSN->添加->PostgreSQL Unicode（如果是64位驱动，将会有64位标识），然后进行配置：

须知

此界面上配置的用户名及密码信息，将会被记录在Windows注册表中，再次连接数据库时就不再需要输入认证信息。但是出于安全考虑，建议在单击“Save”按钮保存配置信息前，清空相关敏感信息；在使用ODBC的连接API时，再传入所需的用户名、密码信息。

步骤4 SSL模式。

如果需要使用SSL证书连接，那么请将DWS安装包中的SSLCERT的证书包解压，双击“sslcert_env.bat”文件，即可完成证书的默认位置的部署。

须知

该sslcert_env.bat为了保证证书环境的纯净，在%APPDATA%\postgresql目录存在时，会提示是否需要移除相关目录。如果有需要，请备份该目录中的文件。

 版权所有 © 华为云计算技术有限公司

或者手动将client.crt、client.key、client.key.cipher、client.key.rand文件放至%APPDATA%\postgresql(该目录需手动建立)目录下，并且将文件名中的client改为postgres，例如client.key修改为postgres.key；将cacert.pem文件放至%APPDATA%\postgresql目录，并更名为root.crt。

同时将步骤2中的设置窗口的“SSL Mode”选项调整至“verify-ca”。

表 13-25 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机构签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。 **说明** 不支持此模式。

步骤5 将客户端所在主机的IP网段加入DWS的安全组规则，确保客户端主机与DWS网络互通。

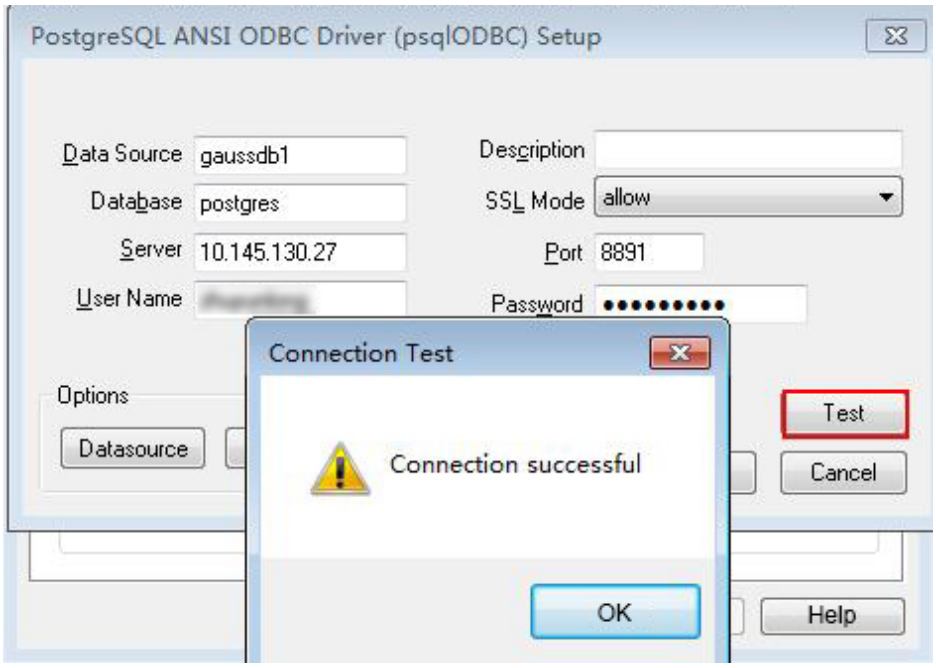
----结束

测试数据源配置

单击Test进行测试。

- 如果显示如下，则表明配置正确，连接成功。

 版权所有 © 华为云计算技术有限公司



- 如果显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

常见问题处理

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因是使用了SSL加密的“verify-full”选项，这时驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。碰到此问题可以使用“verify-ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

    如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

  - 防火墙及网闸设备

    请确认防火墙设置，将数据库的通信端口添加到可信端口中。

    如果有网闸设备，请确认相关的设置。

- 在指定的DSN中，驱动程序和应用程序之间的体系结构不匹配

  此问题可能的原因：在64位程序中使用了32位驱动，或者相反。

  C:\Windows\SysWOW64\odbcad32.exe：这是32位ODBC驱动管理器。

  C:\Windows\System32\odbcad32.exe：这是64位ODBC驱动管理器。

- The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项，请调整此项至allow或以上级别，允许SSL连接，此选项的更多说明，请见**表13-25**。

 版权所有 © 华为云计算技术有限公司

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

  说明

  - 数据库并不存储用户口令，只存储用户口令的哈希码。
  - 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
  - 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
  - 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

  要解决该问题，参见以下操作：

  a. 联系技术支持人员检查数据库参数password_encryption_type是否为1，如果取值不为1，需要将该参数修改为1。

  b. 新建一个数据库用户用于连接，或者重置准备使用的数据库用户的密码。

     - 如果您使用的是管理员账号，参考《数据仓库服务用户指南》的“重置DWS数据库系统管理员密码”章节进行密码重置。
     - 如果是普通用户，可以先通过其他客户端工具（例如Data Studio）连接数据库后，使用ALTER USER语句来修改密码。

  c. 再尝试连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

- FATAL: GSS authentication method is not allowed because XXXX user password is not disabled.

  或：GSSAPI authentication not supported.

  目标CN的pg_hba.conf里配置了当前客户端IP使用“gss”方式来做认证，该认证算法不支持用作客户端的身份认证，请修改到“sha256”后再试。

  同时请注意，数据库当前不支持在集群内跨节点连接数据库，如果是在集群内跨节点连接CN出现此问题，请将业务程序调整到集群外后重试。

13.3.4 ODBC 开发示例

常用功能示例代码

此示例演示如何通过ODBC方式获取DWS中的数据。

```
// DBtest.c (compile with: libodbc.so)
#include <stdlib.h>
#include <stdio.h>
#include <sqlext.h>
#ifdef WIN32
#include <windows.h>
#endif
SQLHENV      V_OD_Env;       // Handle ODBC environment
SQLHSTMT     V_OD_hstmt;     // Handle statement
```

 版权所有 © 华为云计算技术有限公司

```
SQLHDBC       V_OD_hdbc;       // Handle connection
char          typename[100];
SQLINTEGER    value = 100;
SQLINTEGER    V_OD_erg,V_OD_buffer,V_OD_err,V_OD_id;
SQLLEN        V_StrLen_or_IndPtr;
int main(int argc,char *argv[])
{
    // 1. 申请环境句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&V_OD_Env);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error AllocHandle\n");
        exit(0);
    }
    // 2. 设置环境属性（版本信息）
    SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
    // 3. 申请连接句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    // 4. 设置连接属性
    SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, 0);
    // 5. 连接数据源，这里的“userName”与“password”分别表示连接数据库的用户名和用户密码，请根据实际情况修改。
    // 如果odbc.ini文件中已经配置了用户名密码，那么这里可以留空（""）；但是不建议这么做，因为一旦odbc.ini权限管理不善，将导致数据库用户密码泄露。
    V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*) "gaussdb", SQL_NTS,
                          (SQLCHAR*) "userName", SQL_NTS,  (SQLCHAR*) "password", SQL_NTS);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
       printf("Error SQLConnect %d\n",V_OD_erg);
       SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
       exit(0);
    }
    printf("Connected !\n");
    // 6. 设置语句属性
    SQLSetStmtAttr(V_OD_hstmt,SQL_ATTR_QUERY_TIMEOUT,(SQLPOINTER *)3,0);
    // 7. 申请语句句柄
    SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);
    // 8. 直接执行SQL语句。
    SQLExecDirect(V_OD_hstmt,"drop table IF EXISTS customer_t1",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name VARCHAR(32));",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"insert into customer_t1 values(25,'li')",SQL_NTS);
    // 9. 准备执行
    SQLPrepare(V_OD_hstmt,"insert into customer_t1 values(?)",SQL_NTS);
    // 10. 绑定参数
    SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                     &value,0,NULL);
    // 11. 执行准备好的语句
    SQLExecute(V_OD_hstmt);
    SQLExecDirect(V_OD_hstmt,"select c_customer_sk from customer_t1",SQL_NTS);
    // 12. 获取结果集某一列的属性

SQLColAttribute(V_OD_hstmt,1,SQL_DESC_TYPE_NAME,typename,sizeof(typename),NULL,NULL);

    printf("SQLColAtrribute %s\n",typename);
    // 13. 绑定结果集
    SQLBindCol(V_OD_hstmt,1,SQL_C_SLONG, (SQLPOINTER)&V_OD_buffer,150,
            (SQLLEN *)&V_StrLen_or_IndPtr);
    // 14. 通过SQLFetch取结果集中数据
    V_OD_erg=SQLFetch(V_OD_hstmt);
    // 15. 通过SQLGetData获取并返回数据。
    while(V_OD_erg != SQL_NO_DATA)
    {
       SQLGetData(V_OD_hstmt,1,SQL_C_SLONG,(SQLPOINTER)&V_OD_id,0,NULL);
```

 版权所有 © 华为云计算技术有限公司

```
        printf("SQLGetData ----ID = %d\n",V_OD_id);
        V_OD_erg=SQLFetch(V_OD_hstmt);
    };
    printf("Done !\n");
    // 16. 断开数据源连接并释放句柄资源
    SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
    SQLDisconnect(V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC,V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
    return(0);
}
```

批量绑定示例代码

- 请在数据源中打开UseBatchProtocol，同时指定数据库中参数support_batch_bind为on。
- CHECK_ERROR的作用是检查并打印错误信息。
- 此示例将与用户交互式获取DSN、模拟的数据量，忽略的数据量，并将最终数据入库到test_odbc_batch_insert中。

```
#include <stdio.h>
#include <stdlib.h>
#include <sql.h>
#include <sqlext.h>
#include <string.h>

#include "util.c"

void Exec(SQLHDBC hdbc, SQLCHAR* sql)
{
    SQLRETURN retcode;                  // Return status
    SQLHSTMT hstmt = SQL_NULL_HSTMT;    // Statement handle
    SQLCHAR     loginfo[2048];

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
                hstmt, SQL_HANDLE_STMT);

    // Prepare Statement
    retcode = SQLPrepare(hstmt, (SQLCHAR*) sql, SQL_NTS);
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLExecute(hstmt);
    sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    sprintf((char*)loginfo, "SQLFreeHandle stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);
}

int main ()
{
    SQLHENV  henv  = SQL_NULL_HENV;
    SQLHDBC  hdbc  = SQL_NULL_HDBC;
    int      batchCount = 1000;
    SQLLEN   rowsCount = 0;
    int      ignoreCount = 0;

    SQLRETURN   retcode;
    SQLCHAR     dsn[1024] = {'\0'};
    SQLCHAR     loginfo[2048];

    // 交互获取数据源名称
    getStr("Please input your DSN", (char*)dsn, sizeof(dsn), 'N');
```

 版权所有 © 华为云计算技术有限公司

```
// 交互获取批量绑定的数据量
getInt("batchCount", &batchCount, 'N', 1);
do
{
    // 交互获取批量绑定的数据中，不要入库的数据量
    getInt("ignoreCount", &ignoreCount, 'N', 1);
    if (ignoreCount > batchCount)
    {
        printf("ignoreCount(%d) should be less than batchCount(%d)\n", ignoreCount, batchCount);
    }
}while(ignoreCount > batchCount);

retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_ENV)",
            henv, SQL_HANDLE_ENV);

// Set ODBC Version
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
                        (SQLPOINTER*)SQL_OV_ODBC3, 0);
CHECK_ERROR(retcode, "SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION)",
            henv, SQL_HANDLE_ENV);

// Allocate Connection
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_DBC)",
            henv, SQL_HANDLE_DBC);

// Set Login Timeout
retcode = SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (SQLPOINTER)5, 0);
CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_LOGIN_TIMEOUT)",
            hdbc, SQL_HANDLE_DBC);

// Set Auto Commit
retcode = SQLSetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT,
                        (SQLPOINTER)(1), 0);
CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_ATTR_AUTOCOMMIT)",
            hdbc, SQL_HANDLE_DBC);

// Connect to DSN
sprintf(loginfo, "SQLConnect(DSN:%s)", dsn);
retcode = SQLConnect(hdbc, (SQLCHAR*) dsn, SQL_NTS,
                     (SQLCHAR*) NULL, 0, NULL, 0);
CHECK_ERROR(retcode, loginfo, hdbc, SQL_HANDLE_DBC);

// init table info.
Exec(hdbc, "drop table if exists test_odbc_batch_insert");
Exec(hdbc, "create table test_odbc_batch_insert(id int primary key, col varchar2(50))");

// 下面的代码根据用户输入的数据量，构造出将要入库的数据:
{
    SQLRETURN retcode;
    SQLHSTMT hstmtinesrt = SQL_NULL_HSTMT;
    int          i;
    SQLCHAR      *sql = NULL;
    SQLINTEGER   *ids  = NULL;
    SQLCHAR      *cols = NULL;
    SQLLEN       *bufLenIds = NULL;
    SQLLEN       *bufLenCols = NULL;
    SQLUSMALLINT *operptr = NULL;
    SQLUSMALLINT *statusptr = NULL;
    SQLULEN      process = 0;

    // 这里是按列构造，每个字段的内存连续存放在一起。
    ids = (SQLINTEGER*)malloc(sizeof(ids[0]) * batchCount);
    cols = (SQLCHAR*)malloc(sizeof(cols[0]) * batchCount * 50);
    // 这里是每个字段中，每一行数据的内存长度。
    bufLenIds = (SQLLEN*)malloc(sizeof(bufLenIds[0]) * batchCount);
    bufLenCols = (SQLLEN*)malloc(sizeof(bufLenCols[0]) * batchCount);
    // 该行是否需要被处理，SQL_PARAM_IGNORE 或 SQL_PARAM_PROCEED
```

 版权所有 © 华为云计算技术有限公司

```
    operptr = (SQLUSMALLINT*)malloc(sizeof(operptr[0]) * batchCount);
    memset(operptr, 0, sizeof(operptr[0]) * batchCount);
    // 该行的处理结果。
    // 注：由于数据库中处理方式是同一语句隶属同一事务中，所以如果出错，那么待处理数据都将是出错的，并不会部分入库。
    statusptr = (SQLUSMALLINT*)malloc(sizeof(statusptr[0]) * batchCount);
    memset(statusptr, 88, sizeof(statusptr[0]) * batchCount);

    if (NULL == ids || NULL == cols || NULL == bufLenCols || NULL == bufLenIds)
    {
        fprintf(stderr, "FAILED:\tmalloc data memory failed\n");
        goto exit;
    }

    for (int i = 0; i < batchCount; i++)
    {
        ids[i] = i;
        sprintf(cols + 50 * i, "column test value %d", i);
        bufLenIds[i] = sizeof(ids[i]);
        bufLenCols[i] = strlen(cols + 50 * i);
        operptr[i] = (i < ignoreCount) ? SQL_PARAM_IGNORE : SQL_PARAM_PROCEED;
    }

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmtinesrt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
                hstmtinesrt, SQL_HANDLE_STMT);

    // Prepare Statement
    sql = (SQLCHAR*)"insert into test_odbc_batch_insert values(?, ?)";
    retcode = SQLPrepare(hstmtinesrt, (SQLCHAR*) sql, SQL_NTS);
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMSET_SIZE, (SQLPOINTER)batchCount,
sizeof(batchCount));
    CHECK_ERROR(retcode, "SQLSetStmtAttr", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 1, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER,
sizeof(ids[0]), 0,&(ids[0]), 0, bufLenIds);
    CHECK_ERROR(retcode, "SQLBindParameter for id", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 2, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR, 50, 50,
cols, 50, bufLenCols);
    CHECK_ERROR(retcode, "SQLBindParameter for cols", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMS_PROCESSED_PTR, (SQLPOINTER)&process,
sizeof(process));
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAMS_PROCESSED_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_STATUS_PTR, (SQLPOINTER)statusptr,
sizeof(statusptr[0]) * batchCount);
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_STATUS_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_OPERATION_PTR, (SQLPOINTER)operptr,
sizeof(operptr[0]) * batchCount);
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_OPERATION_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLExecute(hstmtinesrt);
    sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLRowCount(hstmtinesrt, &rowsCount);
    CHECK_ERROR(retcode, "SQLRowCount execution", hstmtinesrt, SQL_HANDLE_STMT);

    if (rowsCount != (batchCount - ignoreCount))
```

 版权所有 © 华为云计算技术有限公司

```
        {
            sprintf(loginfo, "(batchCount - ignoreCount)(%d) != rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
            CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
        }
        else
        {
            sprintf(loginfo, "(batchCount - ignoreCount)(%d) == rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
            CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
        }

        if (rowsCount != process)
        {
            sprintf(loginfo, "process(%d) != rowsCount(%d)", process, rowsCount);
            CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
        }
        else
        {
            sprintf(loginfo, "process(%d) == rowsCount(%d)", process, rowsCount);
            CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
        }

        for (int i = 0; i < batchCount; i++)
        {
            if (i < ignoreCount)
            {
                if (statusptr[i] != SQL_PARAM_UNUSED)
                {
                    sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_UNUSED", i, statusptr[i]);
                    CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
                }
            }
            else if (statusptr[i] != SQL_PARAM_SUCCESS)
            {
                sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_SUCCESS", i, statusptr[i]);
                CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
            }
        }

        retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmtinesrt);
        sprintf((char*)loginfo, "SQLFreeHandle hstmtinesrt");
        CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);
    }


exit:
    printf ("\nComplete.\n");

    // Connection
    if (hdbc != SQL_NULL_HDBC) {
        SQLDisconnect(hdbc);
        SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    }

    // Environment
    if (henv != SQL_NULL_HENV)
        SQLFreeHandle(SQL_HANDLE_ENV, henv);

    return 0;
}
```

13.3.5 ODBC 接口参考

ODBC接口是一套提供给用户的API函数，本节将对部分常用接口做具体描述，若涉及其他接口可参考**msdn网址**中ODBC Programmer's Reference项的相关内容。

 版权所有 © 华为云计算技术有限公司

SQLAllocEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocEnv已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLAllocConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocConnect已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLAllocHandle

功能描述

分配环境、连接、语句或描述符的句柄，它替代了ODBC 2.x函数SQLAllocEnv、SQLAllocConnect及SQLAllocStmt。

原型

```
SQLRETURN SQLAllocHandle(SQLSMALLINT   HandleType,
                         SQLHANDLE     InputHandle,
                         SQLHANDLE     *OutputHandlePtr);
```

参数

表 13-26 SQLAllocHandle 参数

关键字	参数说明
HandleType	由SQLAllocHandle分配的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV（环境句柄） • SQL_HANDLE_DBC（连接句柄） • SQL_HANDLE_STMT（语句句柄） • SQL_HANDLE_DESC（描述句柄） 申请句柄顺序为，先申请环境句柄，再申请连接句柄，最后申请语句句柄，后申请的句柄都要依赖它前面申请的句柄。
InputHandle	将要分配的新句柄的类型。 • 如果HandleType为SQL_HANDLE_ENV，则这个值为SQL_NULL_HANDLE。 • 如果HandleType为SQL_HANDLE_DBC，则这一定是一个环境句柄。 • 如果HandleType为SQL_HANDLE_STMT或SQL_HANDLE_DESC，则它一定是一个连接句柄。
OutputHandlePtr	**输出参数**：一个缓冲区的指针，此缓冲区以新分配的数据结构存放返回的句柄。

返回值

- SQL_SUCCESS：表示调用正确。

 版权所有 © 华为云计算技术有限公司

- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当分配的句柄并非环境句柄时，如果SQLAllocHandle返回的值为SQL_ERROR，则它会将OutputHandlePtr的值设置为SQL_NULL_HDBC、SQL_NULL_HSTMT或SQL_NULL_HDESC。之后，通过调用带有适当参数的**SQLGetDiagRec**，其中HandleType和Handle被设置为InputHandle的值，可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLAllocStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocStmt已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLBindCol

功能描述

将应用程序数据缓冲区绑定到结果集的列中。

原型

```
SQLRETURN SQLBindCol(SQLHSTMT       StatementHandle,
                     SQLUSMALLINT   ColumnNumber,
                     SQLSMALLINT    TargetType,
                     SQLPOINTER     TargetValuePtr,
                     SQLLEN         BufferLength,
                     SQLLEN         *StrLen_or_IndPtr);
```

参数

表 13-27 SQLBindCol 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要绑定结果集的列号。起始列号为0，以递增的顺序计算列号，第0列是书签列。若未设置书签页，则起始列号为1。
TargetType	缓冲区中C数据类型的标识符。
TargetValuePtr	**输出参数**：指向与列绑定的数据缓冲区的指针。SQLFetch函数返回这个缓冲区中的数据。如果此参数为一个空指针，则StrLen_or_IndPtr是一个有效值。
BufferLength	TargetValuePtr指向缓冲区的长度，以字节为单位。

版权所有 © 华为云计算技术有限公司

关键字	参数说明
StrLen_or_IndPtr	**输出参数**：缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLBindParameter

功能描述

将一条SQL语句中的一个参数标志和一个缓冲区绑定起来。

原型

```
SQLRETURN SQLBindParameter(SQLHSTMT       StatementHandle,
                           SQLUSMALLINT   ParameterNumber,
                           SQLSMALLINT    InputOutputType,
                           SQLSMALLINT    ValuetType,
                           SQLSMALLINT    ParameterType,
                           SQLULEN        ColumnSize,
                           SQLSMALLINT    DecimalDigits,
                           SQLPOINTER     ParameterValuePtr,
                           SQLLEN         BufferLength,
                           SQLLEN         *StrLen_or_IndPtr);
```

参数

表 13-28 SQLBindParameter

关键词	参数说明
StatementHandle	语句句柄。
ParameterNumber	参数序号，起始为1，依次递增。
InputOutputType	输入输出参数类型。
ValueType	参数的C数据类型。

 版权所有 © 华为云计算技术有限公司

关键词	参数说明
ParameterType	参数的SQL数据类型。
ColumnSize	列的大小或相应参数标记的表达式。
DecimalDigits	列的十进制数字或相应参数标记的表达式。
ParameterValuePtr	指向存储参数数据缓冲区的指针。
BufferLength	ParameterValuePtr指向缓冲区的长度，以字节为单位。
StrLen_or_IndPtr	缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用SQLGetDiagRec函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：示例

SQLColAttribute

功能描述

返回结果集中一列的描述符信息。

原型

```
SQLRETURN SQLColAttribute(SQLHSTMT        StatementHandle,
                          SQLUSMALLINT    ColumnNumber,
                          SQLUSMALLINT    FieldIdentifier,
                          SQLPOINTER      CharacterAttributePtr,
                          SQLSMALLINT     BufferLength,
                          SQLSMALLINT     *StringLengthPtr,
                          SQLPOINTER      NumericAttributePtr);
```

参数

 版权所有 © 华为云计算技术有限公司

表 13-29 SQLColAttribute 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要检索字段的列号，起始为1，依次递增。
FieldIdentifier	IRD中ColumnNumber行的字段。
CharacterAttributePtr	**输出参数**：一个缓冲区指针，返回FieldIdentifier字段值。
BufferLength	• 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个字符串或二进制缓冲区，则此参数为该缓冲区的长度。 • 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个整数，则会忽略该字段。
StringLengthPtr	**输出参数**：缓冲区指针，存放*CharacterAttributePtr中字符类型数据的字节总数，对于非字符类型，忽略BufferLength的值。
NumericAttributePtr	**输出参数**：指向一个整型缓冲区的指针，返回IRD中ColumnNumber行FieldIdentifier字段的值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLColAttribute返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLConnect

功能描述

在驱动程序和数据源之间建立连接。连接上数据源之后，可以通过连接句柄访问到所有有关连接数据源的信息，包括程序运行状态、事务处理状态和错误信息。

原型

```
SQLRETURN  SQLConnect(SQLHDBC        ConnectionHandle,
                      SQLCHAR        *ServerName,
```

 版权所有 © 华为云计算技术有限公司

```
SQLSMALLINT    NameLength1,
SQLCHAR        *UserName,
SQLSMALLINT    NameLength2,
SQLCHAR        *Authentication,
SQLSMALLINT    NameLength3);
```

参数

表 13-30 SQLConnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。
ServerName	要连接数据源的名称。
NameLength1	ServerName的长度。
UserName	数据源中数据库用户名。
NameLength2	UserName的长度。
Authentication	数据源中数据库用户密码。
NameLength3	Authentication的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLConnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLDisconnect

功能描述

关闭一个与特定连接句柄相关的连接。

原型

```
SQLRETURN SQLDisconnect(SQLHDBC    ConnectionHandle);
```

参数

 版权所有 © 华为云计算技术有限公司

表 13-31 SQLDisconnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当调用SQLDisconnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLExecDirect

功能描述

使用参数的当前值，执行一条准备好的语句。对于一次只执行一条SQL语句，SQLExecDirect是最快的执行方式。

原型

```
SQLRETURN SQLExecDirect(SQLHSTMT         StatementHandle,
                        SQLCHAR          *StatementText,
                        SQLINTEGER       TextLength);
```

参数

表 13-32 SQLExecDirect 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
StatementText	要执行的SQL语句。不支持一次执行多条语句。
TextLength	StatementText的长度。

返回值

 版权所有 © 华为云计算技术有限公司

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。
- SQL_NO_DATA：表示SQL语句不返回结果集。

注意事项

当调用SQLExecDirect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLExecute

功能描述

如果语句中存在参数标记的话，SQLExecute函数使用参数标记参数的当前值，执行一条准备好的SQL语句。

原型

```
SQLRETURN SQLExecute(SQLHSTMT    StatementHandle);
```

参数

表 13-33 SQLExecute 参数

关键字	参数说明
StatementHandle	要执行语句的语句句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：表示在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

 版权所有 © 华为云计算技术有限公司

当SQLExecute函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，可通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLFetch

功能描述

从结果集中取下一个行集的数据，并返回所有被绑定列的数据。

原型

```
SQLRETURN SQLFetch(SQLHSTMT    StatementHandle);
```

参数

表 13-34 SQLFetch 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLFreeStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeStmt已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

 版权所有 © 华为云计算技术有限公司

SQLFreeConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeConnect已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

SQLFreeHandle

功能描述

释放与指定环境、连接、语句或描述符相关联的资源，它替代了ODBC 2.x函数SQLFreeEnv、SQLFreeConnect及SQLFreeStmt。

原型

```
SQLRETURN SQLFreeHandle(SQLSMALLINT   HandleType,
                        SQLHANDLE     Handle);
```

参数

表 13-35 SQLFreeHandle 参数

关键字	参数说明
HandleType	SQLFreeHandle要释放的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC 如果HandleType不是这些值之一，SQLFreeHandle返回SQL_INVALID_HANDLE。
Handle	要释放的句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

如果SQLFreeHandle返回SQL_ERROR，句柄仍然有效。

示例

参见：**示例**

SQLFreeEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeEnv已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

 版权所有 © 华为云计算技术有限公司

SQLPrepare

功能描述

准备一个将要进行的SQL语句。

原型

```
SQLRETURN SQLPrepare(SQLHSTMT      StatementHandle,
                     SQLCHAR       *StatementText,
                     SQLINTEGER    TextLength);
```

参数

表 13-36 SQLPrepare 参数

关键字	参数说明
StatementHandle	语句句柄。
StatementText	SQL文本串。
TextLength	StatementText的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当SQLPrepare返回的值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLGetData

功能描述

SQLGetData返回结果集中某一列的数据。可以多次调用它来部分地检索不定长度的数据。

原型

```
SQLRETURN SQLGetData(SQLHSTMT       StatementHandle,
                     SQLUSMALLINT   Col_or_Param_Num,
```

 版权所有 © 华为云计算技术有限公司

```
SQLSMALLINT     TargetType,
SQLPOINTER      TargetValuePtr,
SQLLEN          BufferLength,
SQLLEN          *StrLen_or_IndPtr);
```

参数

表 13-37 SQLGetData 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
Col_or_Param_Num	要返回数据的列号。结果集的列按增序从1开始编号。书签列的列号为0。
TargetType	TargetValuePtr缓冲中的C数据类型的类型标识符。若TargetType为SQL_ARD_TYPE，驱动使用ARD中SQL_DESC_CONCISE_TYPE字段的类型标识符。若为SQL_C_DEFAULT，驱动根据源的SQL数据类型选择缺省的数据类型。
TargetValuePtr	**输出参数**：指向返回数据所在缓冲区的指针。
BufferLength	TargetValuePtr所指向缓冲区的长度。
StrLen_or_IndPtr	**输出参数**：指向缓冲区的指针，在此缓冲区中返回长度或标识符的值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLGetDiagRec

功能描述

返回诊断记录的多个字段的当前值，其中诊断记录包含错误、警告及状态信息。

 版权所有 © 华为云计算技术有限公司

原型

```
SQLRETURN  SQLGetDiagRec(SQLSMALLINT    HandleType,
                         SQLHANDLE      Handle,
                         SQLSMALLINT    RecNumber,
                         SQLCHAR        *SQLState,
                         SQLINTEGER     *NativeErrorPtr,
                         SQLCHAR        *MessageText,
                         SQLSMALLINT    BufferLength,
                         SQLSMALLINT    *TextLengthPtr);
```

参数

表 13-38 SQLGetDiagRec 参数

关键字	参数说明
HandleType	句柄类型标识符，它说明诊断所要求的句柄类型。必须为下列值之一： ● SQL_HANDLE_ENV ● SQL_HANDLE_DBC ● SQL_HANDLE_STMT ● SQL_HANDLE_DESC
Handle	诊断数据结构的句柄，其类型由HandleType来指出。如果HandleType是SQL_HANDLE_ENV，Handle可以是共享的或非共享的环境句柄。
RecNumber	指出应用从查找信息的状态记录。状态记录从1开始编号。
SQLState	**输出参数**：指向缓冲区的指针，该缓冲区存储着有关RecNumber的五字符的SQLSTATE码。
NativeErrorPtr	**输出参数**：指向缓冲区的指针，该缓冲区存储着本地的错误码。
MessageText	指向缓冲区的指针，该缓冲区存储着诊断信息文本串。
BufferLength	MessageText的长度。
TextLengthPtr	**输出参数**：指向缓冲区的指针，返回MessageText中的字节总数。如果返回字节数大于BufferLength，则MessageText中的诊断信息文本被截断成BufferLength减去NULL结尾字符的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

SQLGetDiagRec不发布自己的诊断记录。它用下列返回值来报告它自己的执行结果：

 版权所有 © 华为云计算技术有限公司

- SQL_SUCCESS：函数成功返回诊断信息。
- SQL_SUCCESS_WITH_INFO：*MessageText太小以致不能容纳所请求的诊断信息。没有诊断记录生成。
- SQL_INVALID_HANDLE：由HandType和Handle所指出的句柄是不合法句柄。
- SQL_ERROR：RecNumber小于等于0或BufferLength小于0。

如果调用ODBC函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO，可调用SQLGetDiagRec返回诊断信息值SQLSTATE，SQLSTATE值的如下表。

表 13-39 SQLSTATE 值

SQLSTATE	错误	描述
HY000	一般错误	未定义特定的SQLSTATE所产生的一个错误。
HY001	内存分配错误	驱动程序不能分配所需要的内存来支持函数的执行或完成。
HY008	取消操作	调用SQLCancel取消执行语句后，依然在StatementHandle上调用函数。
HY010	函数系列错误	在为执行中的所有数据参数或列发送数据前就调用了执行函数。
HY013	内存管理错误	不能处理函数调用，可能由当前内存条件差引起。
HYT01	连接超时	数据源响应请求之前，连接超时。
IM001	驱动程序不支持此函数	调用了StatementHandle相关的驱动程序不支持的函数

示例

参见：示例

SQLSetConnectAttr

功能描述

设置控制连接各方面的属性。

原型

```
SQLRETURN SQLSetConnectAttr(SQLHDBC      ConnectionHandle
                            SQLINTEGER   Attribute,
                            SQLPOINTER   ValuePtr,
                            SQLINTEGER   StringLength);
```

参数

 版权所有 © 华为云计算技术有限公司

表 13-40 SQLSetConnectAttr 参数

关键字	参数说明
ConnectionHandle	连接句柄。
Attribute	设置属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr是32位无符号整型值或指向以空结束的字符串。注意，如果ValuePtr参数是驱动程序指定值。ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetConnectAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_DBC的HandleType和ConnectionHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLSetEnvAttr

功能描述

设置控制环境各方面的属性。

原型

```
SQLRETURN SQLSetEnvAttr(SQLHENV      EnvironmentHandle
                        SQLINTEGER   Attribute,
                        SQLPOINTER   ValuePtr,
                        SQLINTEGER   StringLength);
```

参数

表 13-41 SQLSetEnvAttr 参数

关键字	参数说明
EnvironmentHandle	环境句柄。

 版权所有 © 华为云计算技术有限公司

关键字	参数说明
Attribute	需设置的环境属性，可为如下值： • SQL_ATTR_ODBC_VERSION：指定ODBC版本。 • SQL_CONNECTION_POOLING：连接池属性。 • SQL_OUTPUT_NTS：指明驱动器返回字符串的形式。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位整型值，或为以空结束的字符串。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetEnvAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_ENV的HandleType和EnvironmentHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLSetStmtAttr

功能描述

设置相关语句的属性。

原型

```
SQLRETURN SQLSetStmtAttr(SQLHSTMT      StatementHandle
                         SQLINTEGER    Attribute,
                         SQLPOINTER    ValuePtr,
                         SQLINTEGER    StringLength);
```

参数

表 13-42 SQLSetStmtAttr 参数

关键字	参数说明
StatementHandle	语句句柄。

 版权所有 © 华为云计算技术有限公司

关键字	参数说明
Attribute	需设置的属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位无符号整型值，或指向以空结束的字符串，二进制缓冲区，或者驱动定义值。注意，如果ValuePtr参数是驱动程序指定值。ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetStmtAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_STMT的HandleType和StatementHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

 版权所有 © 华为云计算技术有限公司

14 DWS 冷热数据管理

冷热数据简介

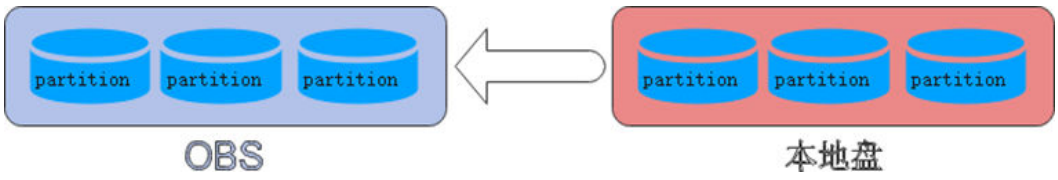
海量大数据场景下，随着业务和数据量的不断增长，数据存储与消耗的资源也日益增长。根据业务系统中用户对不同时期数据的不同使用需求，对膨胀的数据本身进行“冷热”分级管理，不仅可以提高数据分析性能还能降低业务成本。

例如，在网络流量分析系统中，用户可能对最近一个月内安全事件和网络访问情况感兴趣，而很少关注几个月前的数据。针对这样的一些场景，可以将数据按照时间分为：热数据、冷数据。

冷热数据主要从数据访问频率、更新频率进行划分。

- Hot（热数据）：访问、更新频率较高，未来被调用的概率较高的数据，对访问的响应时间要求很高的数据。
- Cold（冷数据）：不允许更新或更新频率比较低，访问频率比较低，对访问的响应时间要求不高的数据。

用户可以定义冷热管理表，将符合规则的冷数据切换至OBS上进行存储，可以按照分区自动进行冷热数据的判断和迁移。



冷热数据迁移

DWS列存数据写入时，数据首先进入热分区进行存储，分区数据较多后，可通过手动或自动的方式，将符合冷数据规则的数据切换至OBS上进行存储。在数据切换至OBS上后，其元数据、Desc表信息以及索引信息仍在本地进行存储，保证了读取的性能。

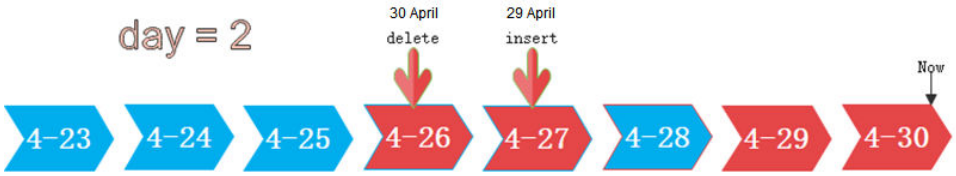
冷热切换策略

目前冷热切换的策略名称支持LMT（last modify time）和HPN（hot partition number），LMT指按分区的最后更新时间切换，HPN指保留热分区的个数切换。

- LMT：表示切换[day]时间前修改的热分区数据为冷分区，将该数据迁至OBS表空间中。其中[day]为整型，范围[0，36500]，单位为天。

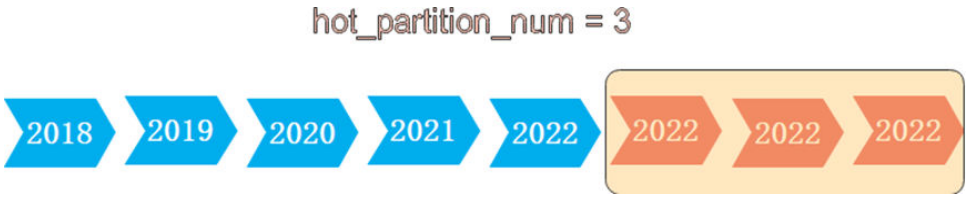
版权所有 © 华为云计算技术有限公司

如下图中，设置day为2，即在冷热切换时，根据分区数据的最晚修改时间，保留2日内所修改的分区为热分区，其余数据为冷分区数据。假设当前时间为4月30日，4月30日对[4-26]分区进行了delete操作，4月29日对[4-27]分区进行了insert操作，故在冷热切换时，保留[4-26][4-27][4-29][4-30]四个分区为热分区。



- HPN：表示保留HPN个有数据的分区为热分区。分区顺序按照分区的Sequence ID来确定，分区的Sequence ID是根据分区边界值的大小，内置生成的序号，此序号不对外呈现。对于RANGE分区，分区的边界值越大，分区对应的Sequence ID越大；对于LIST分区，分区边界枚举值中的最大值越大，分区对应的Sequence ID越大。在冷热切换时，需要将数据迁移至OBS表空间中。其中HPN为整型，范围为[0,1600]。其中HPN为0时，表示不保留热分区，在进行冷热切换时，将所有有数据的分区都转为冷分区并存储在OBS上。

  如下图中，设置HPN为3，即在冷热切换时，保留最新的3个有数据的分区为热分区数据，其余分区均切为冷分区。



冷热数据管理支持功能

- 支持对冷热表的insert、copy、delete、update、select等表相关的DML操作。
- 支持对冷热表的权限管理等DCL操作。
- 支持对冷热表进行analyze、vacuum、merge into等操作和一些分区的操作。
- 支持从普通列存分区表升级为冷热数据表。
- 支持带有冷热数据管理表的升级、扩容、缩容和重分布。
- 支持冷热分区互相转换。该功能仅8.3.0及以上版本支持。

冷热数据管理的约束限制

- 目前冷热表只支持列存2.0版本的分区表，外表不支持冷热分区。
- 对于已经切冷分区再次插入数据，数据直接会进入OBS，不会改变分区的冷热属性。
- 对于同一分区在同一DN只会存在冷或热的一种情况，对于同一分区在不同DN可能存在部分DN为热数据，部分DN为冷数据。
- 对于同时存在冷热分区的表，查询时会变慢，因为冷数据存储在OBS上，读写速度和时延都比在本地查询要慢。
- 只支持修改冷热表的冷热切换策略，不支持修改冷热表的冷数据的表空间。
- 冷热表的分区操作约束：
  - 不支持对冷分区的数据进行exchange操作。

 版权所有 © 华为云计算技术有限公司

  - Merge partition分区只支持热分区和热分区合并、冷分区和冷分区合并，不支持冷热分区合并。
  - ADD/Merge/Split Partition等分区操作不支持指定表空间为OBS表空间。
  - 不支持创建时指定和修改冷热表分区的表空间。
- 冷热切换不是只要满足条件就立刻进行冷热数据切换，依赖用户手动调用切换命令，或者通过调度器调用切换命令后才真正进行数据切换。目前自动调度时间为每日0点，可进行修改。
- 目前冷热切换规则只支持LMT和HPN两种。
- 冷热数据表不支持物理细粒度备份和恢复，由于物理备份时只备份热数据，在备份恢复前后OBS上冷数据为同一份，不支持truncate和drop table等涉及删除文件操作语句的备份恢复操作。

使用示例

1. 创建列存冷热数据管理表，指定热数据有效期LMT为100天。

```
CREATE TABLE lifecycle_table(i int, val text) WITH (ORIENTATION = COLUMN, storage_policy = 'LMT:100')
PARTITION BY RANGE (i)
(
PARTITION P1 VALUES LESS THAN(5),
PARTITION P2 VALUES LESS THAN(10),
PARTITION P3 VALUES LESS THAN(15),
PARTITION P8 VALUES LESS THAN(MAXVALUE)
)ENABLE ROW MOVEMENT;
```

2. 切换冷数据至OBS表空间。
   - 自动切换：每日0点调度框架自动触发，无需关注切换情况。

     可自定义自动切换时间：根据业务情况调整自动触发时间，修改为每天早晨6点30分。

     ```
     SELECT * FROM pg_obs_cold_refresh_time('lifecycle_table', '06:30:00');
     ```

   - 手动切换。

     执行如下操作手动切换单表：

     ```
     ALTER TABLE lifecycle_table refresh storage;
     ```

     执行如下操作批量切换所有冷热表：

     ```
     SELECT pg_catalog.pg_refresh_storage();
     ```

3. 将冷分区数据转换成热分区。该功能仅8.3.0及以上版本支持。

   将冷热表的所有冷分区转换成热分区：

   ```
   SELECT pg_catalog.reload_cold_partition('lifecycle_table');
   ```

   将冷热表的指定冷分区转换成热分区：

   ```
   SELECT pg_catalog.reload_cold_partition('lifecycle_table', 'cold_partition_name');
   ```

4. 查看冷热表数据分布情况。

   查看单表数据分布情况：

   ```
   SELECT * FROM pg_catalog.pg_lifecycle_table_data_distribute('lifecycle_table');
   ```

   查看所有冷热表数据分布情况：

   ```
   SELECT * FROM pg_catalog.pg_lifecycle_node_data_distribute();
   ```

5. 删除冷热表的分区及分区数据。

   删除冷热表的指定分区数据：

   ```
   ALTER TABLE lifecycle_table TRUNCATE PARTITION P1;
   ```

   删除冷热表的指定分区：

 版权所有 © 华为云计算技术有限公司

```
ALTER TABLE lifecycle_table DROP PARTITION P1;
```

 版权所有 © 华为云计算技术有限公司

15 DWS 资源监控

DWS为用户提供了多维度的资源监控视图。可以查看作业的实时资源记录和历史资源记录。

15.1 用户资源监控

在多租户管理的框架下，用户可以通过系统视图**PG_TOTAL_USER_RESOURCE_INFO**、**PGXC_TOTAL_USER_RESOURCE_INFO**和函数GS_WLM_USER_RESOURCE_INFO实时查询所有用户资源（包括内存，CPU核数，存储空间、临时空间、算子落盘空间和IO）实时使用情况，也可以通过系统表**GS_WLM_USER_RESOURCE_HISTORY**和系统视图**PGXC_WLM_USER_RESOURCE_HISTORY**查询用户资源的历史使用情况。

注意事项

- 用户监控可以同时监控快慢车道（快车道管控简单作业，慢车道管控复杂作业）所有作业的CPU、IO和内存使用情况，不再受限于仅监控慢车道作业。
- 当前快车道作业内存和CPU不受控，在快车道运行作业占用资源较多情况下，可能出现已用资源大于资源限制的情况。
- DN监控视图中，IO、内存和CPU显示的是本DN上资源池资源使用和资源限制信息。
- CN监控视图中，IO、内存和CPU显示的是集群内所有DN资源池资源使用和资源限制的累积和。
- DN每隔5s更新一次监控信息，CN每隔5s从DN收集一次用户监控信息，因为各实例单独更新/收集用户监控信息，因此各实例监控信息更新时间可能不一致。
- 辅助线程中每隔30s自动调用持久化函数，持久化用户监控数据，正常情况下不需要用户单独调用持久化函数持久化用户监控数据。
- 当用户数量较多，集群规模较大时，查询此类实时视图，因CN/DN间实时通信开销，会有一定的网络延时。
- 初始管理用户不进行资源监控。

操作步骤

- 查询所有用户的资源限额和资源实时使用情况。

```
SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO;
```

  得到的结果视图如下：

 版权所有 © 华为云计算技术有限公司

```
username       | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
---------------------+-------------+--------------+----------+-----------+------------+-------------
+-----------------+-----------------+-----------------+------------------+-------------+--------------
+-------------+--------------+------------+-------------+------------+------------
perfadm            |          0 |          0 |        0 |        0 |         0 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
usern              |          0 |      17250 |        0 |       48 |         0 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
(2 rows)
```

其中，IO资源监控字段(read_kbytes、write_kbytes、read_counts、write_counts、read_speed和write_speed)需要在GUC参数**enable_user_metric_persistent**开启时才有监控数据。

所查各字段说明详见**PG_TOTAL_USER_RESOURCE_INFO** 。

- 查询具体某个用户的资源限额和资源实时使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_INFO('username');
```

查询结果如下：

```
userid | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
--------+-------------+--------------+----------+-----------+------------+-------------
+-----------------+-----------------+-----------------+------------------+-------------+--------------
+-------------+--------------+------------+-------------+------------+------------
16407 |          18 |       1655 |        6 |         19 |         13787176 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
(1 row)
```

- 查询所有用户的资源限额和资源历史使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_HISTORY;
```

查询结果如下：

```
username       |          timestamp          | used_memory | total_memory | used_cpu | total_cpu |
used_space | total_space | used_temp_space | total_temp_space | used_spill_space | total_spill_space |
read_kbytes | write_kbytes | read_counts | write_counts | read_speed  | write_speed | send_speed |
recv_speed
----------------------+-------------------------------+-------------+--------------+----------+-----------
+------------+-------------+-----------------+------------------+------------------+-------------------
+-------------+--------------+-------------+--------------+-------------+-------------+------------+------------
usern              | 2020-01-08 22:56:06.456855+08 |           0 |        17250 |        0 |        48 |          0
|          -1 |             0 |             -1 |         88349078 |              -1 |        45680 |          34 |        5710
|           8 |         320 |           0 |          0 |          0
userg              | 2020-01-08 22:56:06.458659+08 |           0 |        15525 |    33.48 |        48 |          0
|          -1 |             0 |             -1 |        110169581 |              -1 |        17648 |          23 |
2206 |           5 |         123 |           0 |          0 |          0
userg1             | 2020-01-08 22:56:06.460252+08 |           0 |        13972 |    33.48 |        48 |          0
|          -1 |             0 |             -1 |        136106277 |              -1 |        17648 |          23 |
2206 |           5 |         123 |           0 |          0 |          0
```

对于系统表**GS_WLM_USER_RESOURCE_HISTORY**，仅当GUC参数**enable_user_metric_persistent**开启时，才会定期将视图**PG_TOTAL_USER_RESOURCE_INFO**中的数据保存到历史表中。

所查各字段说明详见**GS_WLM_USER_RESOURCE_HISTORY**。

 版权所有 © 华为云计算技术有限公司

15.2 资源池资源监控

简介

多租户管理框架下，用户关联资源池执行查询，用户执行查询所占用的资源将汇总至关联资源池上，通过资源池监控视图，用户可以直观的查询到所有资源池的实时资源使用情况，同时也可以通过资源池监控历史表查询资源池资源的历史使用情况。

资源池监控数据每5s更新一次，但是因为CN和DN时间差，实际监控数据更新时间可能会大于5s，正常不会超过10s。资源池监控数据每30s持久化一次，资源池监控和用户监控逻辑基本一致，因此共用GUC参数控制持久化和老化，使用GUC参数enable_user_metric_persistent控制是否进行资源池监控数据持久化，使用GUC参数user_metric_retention_time控制资源池监控数据老化。

资源池监控的资源包含：快慢车道作业运行和排队信息，CPU、内存以及逻辑IO资源监控信息。涉及的监控视图和历史表如下：

- 资源池实时运行信息监控视图(单CN)：**GS_RESPOOL_RUNTIME_INFO**。
- 资源池实时运行信息监控视图(所有CN)：**PGXC_RESPOOL_RUNTIME_INFO**。
- 资源池实时资源监控视图(单CN)：**GS_RESPOOL_RESOURCE_INFO**。
- 资源池实时资源监控视图(所有CN)：**PGXC_RESPOOL_RESOURCE_INFO**。
- 资源池历史资源监控表(单CN)：**GS_RESPOOL_RESOURCE_HISTORY**。
- 资源池历史资源监控视图(所有CN)：**PGXC_RESPOOL_RESOURCE_HISTORY**。

说明

- 资源池监控可以同时监控快慢车道所有作业的CPU、IO和内存使用情况，不再受限于仅监控慢车道作业。
- 当前快车道作业内存和CPU不受控，在快车道运行作业占用资源较多情况下，可能出现已用资源大于资源限制的情况。
- DN资源池监控视图中，IO、内存和CPU显示的是本DN上资源池资源使用和资源限制信息。
- CN资源池监控视图中，IO、内存和CPU显示的是集群内所有DN资源池资源使用和资源限制的累积和。
- DN每隔5s更新一次资源池监控信息，CN每隔5s从DN收集一次资源池监控信息，因为各实例单独更新/收集资源池监控信息，因此各实例监控信息更新时间可能不一致。
- 辅助线程中每隔30s自动调用持久化函数，持久化资源池监控数据，正常情况下不需要用户单独调用持久化函数持久化资源池监控数据。

操作步骤

- 查询资源池的作业实时运行情况。

```
SELECT * FROM GS_RESPOOL_RUNTIME_INFO;
```

得到的结果视图如下：

```
 nodegroup |     rpname     | ref_count | fast_run | fast_wait | slow_run | slow_wait
-----------+----------------+-----------+----------+-----------+----------+-----------
 vc1       | p2             |        10 |        0 |         0 |        0 |         0
 vc2       | p3             |        10 |        5 |         5 |        0 |         0
 vc2       | p4             |         0 |        0 |         0 |        0 |         0
 vc1       | default_pool   |         0 |        0 |         0 |        0 |         0
 vc2       | default_pool   |         0 |        0 |         0 |        0 |         0
 vc1       | p1             |        20 |        5 |         5 |        3 |         7
(6 rows)
```


版权所有 © 华为云计算技术有限公司

其中：

a. ref_count为引用当前资源池信息的作业数，语句从进入管控到结束一直占用该计数；

b. fast_run和slow_run为负载管理记账信息，只有管控(fast_limit/slow_limit大于0)时该值才有效；

c. 该视图仅在CN上有效，持久化信息保存在GS_RESPOOL_RESOURCE_HISTORY中；

d. 各字段说明详见**GS_RESPOOL_RUNTIME_INFO**。

- 查询资源池的资源限额和资源实时使用情况。

```
SELECT * FROM GS_RESPOOL_RESOURCE_INFO;
```

得到的结果视图如下：

```
nodegroup |     rpname     |       cgroup        | ref_count | fast_run | fast_wait | fast_limit | slow_run |
slow_wait | slow_limit | used_cpu | cpu_limit | used_mem | estimate_mem | mem_limit |read_kbytes |
write_kbytes | read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
-----------+----------------+---------------------+-----------+----------+-----------+------------+----------
+-----------+------------+----------+-----------+----------+--------------+-----------+-------------+--------------
+-------------+--------------+-------------+------------+------------+------------
 vc1       | p2             | DefaultClass:Rush   |        10 |        0 |         0 |         -1 |        0 |         0 |        10
|     9.97 |        48 |       20 |           0 |     11555 |         8 |        2880 |          1 |         360 |         1
|         589 |          0 |          0
 vc2       | p3             | DefaultClass:Rush   |        10 |        5 |         5 |          5 |        0 |         0 |        10
|     4.98 |        48 |       11 |           0 |     11555 |         0 |         848 |          0 |         106 |         0
|         173 |          0 |          0
 vc2       | p4             | DefaultClass:Rush   |         0 |        0 |         0 |         -1 |        0 |         0 |        10
|        0 |        48 |        0 |           0 |     11555 |         0 |           0 |          0 |           0 |         0
|           0 |          0 |          0
 vc1       | default_pool | DefaultClass:Medium |         0 |        0 |         0 |         -1 |        0 |         0
|         -1 |        0 |       48 |        0 |           0 |     11555 |         0 |           0 |           0 |          0
|          0 |          0 |          0 |          0
 vc2       | default_pool | DefaultClass:Medium |         0 |        0 |         0 |         -1 |        0 |         0
|         -1 |        0 |       48 |        0 |           0 |     11555 |         0 |           0 |           0 |          0
|          0 |          0 |          0 |          0
 vc1       | p1             | DefaultClass:Rush   |        20 |        5 |         5 |          5 |        3 |         7 |         3
|     7.98 |        48 |       16 |         768 |     11555 |         8 |        2656 |          1 |         332 |         1
|         543 |          0 |          0
(6 rows)
```

a. 该视图在CN和DN上均有效，DN上CPU、内存和IO为本DN资源消耗情况，CN上CPU、内存和IO为集群内所有DN上资源消耗的累加和；

b. estimate_mem仅在动态负载管理情况下CN上有效，显示资源池估算内存记账情况；

c. IO监控信息仅在enable_logical_io_statistics开启时才会记录；

d. 各字段说明详见**GS_RESPOOL_RESOURCE_INFO**。

- 查询资源池的资源限额和资源历史使用情况。

```
SELECT * FROM GS_RESPOOL_RESOURCE_HISTORY ORDER BY timestamp DESC;
```

得到的结果视图如下：

```
timestamp           |  nodegroup   |     rpname     |       cgroup        | ref_count | fast_run | fast_wait |
fast_limit | slow_run | slow_wait | slow_limit | used_cpu | cpu_limit | used_mem | estimate_mem |
mem_limit | read_kbytes | write_kbytes | read_counts | write_counts | read_speed | write_speed |
send_speed | recv_speed
-------------------------------+--------------+----------------+---------------------+-----------+----------+-----------+
+-----------+----------+-----------+------------+----------+-----------+----------+--------------+-----------+-------------
+--------------+-------------+--------------+-------------+--------------+------------+-------------+------------
+------------
 2022-03-04 09:41:57.53739+08  | vc1          | p2             | DefaultClass:Rush   |        10 |        0 |         0
|         -1 |        0 |         0 |         10 |     9.97 |        48 |   20 |           0 |     11555 |           0 |
2320 |          0 |         290 |          0 |         474 |          0 |          0
 2022-03-04 09:41:57.53739+08  | vc1          | p1             | DefaultClass:Rush   |        20 |        5 |         5
|          5 |        3 |         7 |          3 |     7.98 |        48 |   16 |         768 |     11555 |           0 |
```

 版权所有 © 华为云计算技术有限公司

```
1896 |          0 |         237 |          0 |         387 |          0 |          0
 2022-03-04 09:41:57.53739+08  | vc2          | default_pool | DefaultClass:Medium |        0 |       0
|        0 |        -1 |       0 |        0 |        -1 |       0 |       48 |    0 |          0 |     11555 |          0
|          0 |         0 |          0 |         0 |         0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc1          | default_pool | DefaultClass:Medium |        0 |       0
|        0 |        -1 |       0 |        0 |        -1 |       0 |       48 |    0 |          0 |     11555 |          0
|          0 |         0 |          0 |         0 |         0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc2          | p4           | DefaultClass:Rush   |        0 |       0 |       0
|        -1 |        0 |       0 |        10 |        0 |       48 |    0 |          0 |     11555 |          0 |          0
|          0 |         0 |          0 |         0 |         0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc2          | p3           | DefaultClass:Rush   |       10 |       5 |       5
|         5 |        0 |       0 |        10 |     4.99 |       48 |   11 |          0 |     11555 |          0 |        880
|          0 |         110 |          0 |         180 |          0 |          0
 2022-03-04 09:41:27.335234+08 | vc2          | p3           | DefaultClass:Rush   |       10 |       5 |       5
|         5 |        0 |       0 |        10 |     4.98 |       48 |   11 |          0 |     11555 |          0 |        856
|          0 |         107 |          0 |         175 |          0 |          0
```

a. 该监控信息来自资源池监控历史表，enable_user_metric_persistent开启时每30秒记录一次；

b. 该表数据保存时间由GUC参数user_metric_retention_time控制；

c. 各字段说明详见**GS_RESPOOL_RESOURCE_HISTORY**。

15.3 内存资源监控

内存监控

DWS支持在监控面板(DMS)查看集群内存使用情况，详情参考《数据仓库服务用户指南》中“在监控面板(DMS)查看DWS集群监控”章节。

DWS也提供了监控整个集群内存使用状态的视图：

查询pgxc_total_memory_detail视图，必须具有sysadmin权限。

```
SELECT * FROM pgxc_total_memory_detail;
```

如果查询该视图时出现以下错误，请开启内存管理功能。

```
SELECT * FROM pgxc_total_memory_detail;
ERROR:  unsupported view for memory protection feature is disabled.
CONTEXT:  PL/pgSQL function pgxc_total_memory_detail() line 12 at FOR over EXECUTE statement
```

用户可通过DWS控制台设置max_process_memory参数来开启内存管理功能，方法如下：

1. 登录DWS控制台。
2. 在左侧导航栏中，单击“专属集群 > 集群列表”。
3. 在集群列表中找到所需要的集群，单击集群名称，进入“集群详情”页面。
4. 单击“参数修改”页签，修改参数“max_process_memory”的值为合适的值，修改建议请参见**max_process_memory**，然后单击“保存”。
5. 在“修改预览”窗口，确认修改无误后，单击“保存”。修改完成后需要重启集群，参数才会生效。

15.4 实例资源监控

DWS提供了监控CN、DN实例资源使用状态（包括内存，CPU，磁盘IO，进程物理IO和进程逻辑IO）的系统表及监控整个集群资源使用状态的系统表。

关于系统表GS_WLM_INSTANCE_HISTORY的详细介绍，请参考**GS_WLM_INSTANCE_HISTORY**。

版权所有 © 华为云计算技术有限公司

说明

系统表GS_WLM_INSTANCE_HISTORY中的数据分布在对应的实例中，CN实例监控数据保存在CN实例中，DN实例监控数据保存在DN实例中；DN实例由于有备机，当主DN实例异常时，该DN实例的监控数据能够从备机恢复；但CN实例无备机，当某CN实例异常再恢复时，该CN实例的监控数据会丢失。

操作步骤

- 查询当前实例最近的资源使用情况。

```
SELECT * FROM GS_WLM_INSTANCE_HISTORY ORDER BY TIMESTAMP DESC;
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+----------+-----------
+------------+--------------+---------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2022-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |       357717 |        37667
dn_6015_6016 | 2022-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |       358103 |        38584
dn_6015_6016 | 2022-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |       316085 |        39079
dn_6015_6016 | 2022-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |       303351 |        38136
```

- 查询当前实例某一段时间内的资源使用情况。

```
SELECT * FROM GS_WLM_INSTANCE_HISTORY WHERE TIMESTAMP > '2022-01-10' AND TIMESTAMP
< '2020-01-11' ORDER BY TIMESTAMP DESC;
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+----------+-----------
+------------+--------------+---------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2022-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |       357717 |        37667
dn_6015_6016 | 2022-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |       358103 |        38584
dn_6015_6016 | 2022-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |       316085 |        39079
dn_6015_6016 | 2022-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |       303351 |        38136
```

- 查询集群最近的资源使用情况，可以在CN节点上调用pgxc_get_wlm_current_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_current_instance_info('ALL');
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+---------+-----------
+------------+--------------+---------------+--------------+---------------+-------------+--------------
coordinator2 | 2020-01-14 21:58:29.290894+08 |        0 |    12010 |      278 |  16.0445 | 7.19561 |
184.431 |    27959.3 |            0 |            10 |           0 |            0 |          0 |            0
coordinator3 | 2020-01-14 21:58:27.567655+08 |        0 |    12000 |      288 |  .964557 | 3.40659 |
332.468 |    3375.02 |           26 |            13 |           0 |            0 |          0 |            0
datanode1    | 2020-01-14 21:58:23.900321+08 |        0 |    11899 |      389 |  1.17296 |    3.25 |
329.6 |    2870.4 |           28 |             8 |          13 |            3 |         18 |            6
datanode2    | 2020-01-14 21:58:32.832989+08 |        0 |    11904 |      384 |   17.948 | 8.52148 |
214.186 |    25894.1 |           28 |            10 |          13 |            3 |         18 |            6
datanode3    | 2020-01-14 21:58:24.826694+08 |        0 |    11894 |      394 |  1.16088 |    3.15 |      328
|    2868.8 |           25 |            10 |          13 |            3 |         18 |            6
coordinator1 | 2020-01-14 21:58:33.367649+08 |        0 |    11988 |      300 |  9.53286 |   10.05 |
```

 版权所有 © 华为云计算技术有限公司

```
43.2 |      55232 |           0 |           0 |           0 |           0 |          0 |          0
coordinator1 | 2020-01-14 21:58:23.216645+08 |       0 |    11988 |      300 |  1.17085 | 3.21182 |
324.729 |    2831.13 |           8 |          13 |           0 |           0 |          0 |          0
(7 rows)
```

- 查询集群历史的资源使用情况，可以在CN节点上调用pgxc_get_wlm_history_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_history_instance_info('ALL', '2020-01-14 21:00:00', '2020-01-14
22:00:00', 3);
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await |  io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+-----------+-----------
+------------+--------------+--------------+--------------+--------------+-------------+--------------
coordinator2 | 2020-01-14 21:50:49.778902+08 |       0 |    12020 |      268 |  .127371 |   .789211 |
15.984 |    3994.41 |           0 |           0 |           0 |           0 |          0 |          0
coordinator2 | 2020-01-14 21:53:49.043646+08 |       0 |    12018 |      270 |  30.2902 |   8.65404 |
276.77 |    16741.8 |           3 |           1 |           0 |           0 |          0 |          0
coordinator2 | 2020-01-14 21:57:09.202654+08 |       0 |    12018 |      270 |   .16051 |   .979021 |
59.9401 |       5596 |           0 |           0 |           0 |           0 |          0 |          0
coordinator3 | 2020-01-14 21:38:48.948646+08 |       0 |    12012 |      276 | .0769231 | .00999001
|          0 |    35.1648 |           0 |           1 |           0 |           0 |          0 |          0
coordinator3 | 2020-01-14 21:40:29.061178+08 |       0 |    12012 |      276 |  .118421 |  .0199601
|          0 |    970.858 |           0 |           0 |           0 |           0 |          0 |          0
coordinator3 | 2020-01-14 21:50:19.612777+08 |       0 |    12010 |      278 |   24.411 |   11.7665 |
8.78244 |    44641.1 |           0 |           0 |           0 |           0 |          0 |          0
datanode1    | 2020-01-14 21:49:42.758649+08 |       0 |    11909 |      379 |  .798776 |      8.02 |
51.2 |    20924.8 |           0 |           0 |           0 |           0 |          0 |          0
datanode1    | 2020-01-14 21:49:52.760188+08 |       0 |    11909 |      379 |  23.8972 |      14.1 |
0 |      74760 |           0 |           0 |           0 |           0 |          0 |          0
datanode1    | 2020-01-14 21:50:22.769226+08 |       0 |    11909 |      379 |  39.5868 |       7.4 |         0
|    19760.8 |           0 |           0 |           0 |           0 |          0 |          0
datanode2    | 2020-01-14 21:58:02.826185+08 |       0 |    11905 |      383 |  .351648 |       .32 |
20.8 |      504.8 |           0 |           0 |           0 |           0 |          0 |          0
datanode2    | 2020-01-14 21:56:42.80793+08  |       0 |    11906 |      382 |  .559748 |       .04 |         0
|      326.4 |           0 |           0 |           0 |           0 |          0 |          0
datanode2    | 2020-01-14 21:45:21.632407+08 |       0 |    11901 |      387 |  12.1313 |   4.55544 |
3.1968 |    45177.2 |           0 |           0 |           0 |           0 |          0 |          0
datanode3    | 2020-01-14 21:58:14.823317+08 |       0 |    11898 |      390 |  .378205 |       .99 |
48 |    23353.6 |           0 |           0 |           0 |           0 |          0 |          0
datanode3    | 2020-01-14 21:47:50.665028+08 |       0 |    11901 |      387 |  1.07494 |       1.19 |
0 |    15506.4 |           0 |           0 |           0 |           0 |          0 |          0
datanode3    | 2020-01-14 21:51:21.720117+08 |       0 |    11903 |      385 |  10.2795 |       3.11 |
0 |    11031.2 |           0 |           0 |           0 |           0 |          0 |          0
coordinator1 | 2020-01-14 21:42:59.121945+08 |       0 |    12020 |      268 | .0857143 |  .0699301
|          0 |    6579.02 |           0 |           0 |           0 |           0 |          0 |          0
coordinator1 | 2020-01-14 21:41:49.042646+08 |       0 |    12020 |      268 |  20.9039 |   11.3786 |
6042.76 |    57903.7 |           0 |           0 |           0 |           0 |          0 |          0
coordinator1 | 2020-01-14 21:41:09.007652+08 |       0 |    12020 |      268 | .0446429 |    .03996 |
0 |    1109.29 |           0 |           0 |           0 |           0 |          0 |          0
(18 rows)
```

15.5 实时 TopSQL

系统提供了不同级别的资源监控实时视图用来查询实时TopSQL。资源监控实时视图记录了查询作业运行时的资源使用情况(包括内存、下盘、CPU时间等)以及性能告警信息。

实时视图具体的对外接口如下表所示：

 版权所有 © 华为云计算技术有限公司

表 15-1 资源监控实时视图

视图级别	节点范围	查询视图
query级别/perf级别	当前CN	**GS_WLM_SESSION_STATISTICS**
	所有CN	**PGXC_WLM_SESSION_STATISTICS**
operator_realtime级别/operator级别	当前CN	**GS_WLM_OPERATOR_STATISTICS**
	所有CN	**PGXC_WLM_OPERATOR_STATISTICS**

版权所有 © 华为云计算技术有限公司

说明

- 视图级别取决于资源监控的等级，即参数**resource_track_level**的配置。
- perf和operator级别会影响**GS_WLM_SESSION_STATISTICS**/**PGXC_WLM_SESSION_INFO**中的query_plan和warning字段的取值，详细内容参见**SQL自诊断**。
- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 查询此类实时视图时，因需要获取作业运行实时资源使用情况，会有一定的网络延时。
- 实例故障时，实时TopSQL视图有可能记录不全。
- 实时TopSQL中能够记录的SQL语句的规格是：
  - 不记录特殊数据定义语句，如：SET、RESET、SHOW、ALTER SESSION SET、SET CONSTRAINTS语句。
  - 记录数据定义语句，例如：执行CREATE、ALTER、DROP、GRANT、REVOKE和VACUUM语句。
  - 记录数据操作语句，例如：
    - 执行SELECT、INSERT、UPDATE和DELETE语句。
    - 执行explain analyze和explain performance场景。
    - 使用query级别/perf级别视图。
  - 记录函数与存储过程的调用入口语句，当GUC参数**enable_track_record_subsql**开启的情况下，可记录存储过程的部分内部语句(declare定义语句除外)，仅会记录其中下发到DN执行的内部语句，未下发到DN执行的内部语句会被过滤掉；
  - 记录匿名块语句，当GUC参数**enable_track_record_subsql**开启的情况下，可记录匿名块中的部分内部语句，仅会记录其中下发到DN执行的内部语句，未下发到DN执行的内部语句会被过滤掉。
  - 记录游标语句，当游标没有从缓存中读取数据，而触发语句下发到DN上执行的条件下，该游标语句会被记录，并且会进行语句、执行计划增强，但当游标从缓存中读取数据时，不进行记录；当游标语句在匿名块或者函数中使用且游标从DN上读取较多数据但不完全使用时，因当前架构限制，无法记录该游标在DN上的监控信息。对于With Hold游标，该语法执行逻辑特殊，会在事务提交阶段执行实际查询动作，当语句在该阶段执行报错时，作业的aborted状态无法反馈到TopSQL历史表中。
  - 重分布过程中的作业不统计。
  - JDBC执行的带占位符语句，通常会补齐参数内容，但当参数和原语句合起来长度超过64KB，则不记录参数，或者是轻量化语句直接下发到DN上执行，不记录参数。
  - 从8.1.3集群版本开始，query、perf级别TopSQL运行时监控功能已完全不影响查询性能，对当前会话的语句进行资源监控的GUC参数resource_track_cost默认值已修改为0，查询TopSQL实时监控视图时，默认会显示所有正在执行的语句。
  - 从8.1.3集群版本开始，对于存储过程中的子语句监控功能，如果在查询TopSQL实时监控视图的会话中，开启控制子语句记录归档功能的GUC参数enable_track_record_subsql，不论业务语句中是否开启子语句监控开关，查询TopSQL实时监控视图的结果都能看到执行语句的子语句运行信息。
  - 关于存储过程中子语句的监控功能即enable_track_record_subsql，8.1.3集群版本中建议不要全面开启，由于没有按时间过滤子语句的功能，全面开启可能会记录过多子语句，导致归档的监控表占用大量磁盘空间；8.1.3集群版本建议仅用于查询实时监控信息，或对个别存储过程业务做定位分析时，仅开启对应会话中的参数。8.2.1及以上集群版本新增GUC参数resource_track_subsql_duration（默认值为180秒），可以通过执行时间过滤需要归档的子语句，用户可以按需调整该值大小。
  - 由于规格限制，对于未下盘的主语句，TopSQL历史表中的记录会有延时，等待下次作业下发时才会显示在TopSQL历史表中。
  - 自8.2.1.200集群版本，新增operator_realtime级别TopSQL运行时监控，提供算子级实时监控的能力，开启此级别的监控可以查询语句的执行计划以及具体执行信息，查询TopSQL算子级实时监控视图时，默认会显示所有正在执行的语句。但是对于存储过程和游标场景，暂时不支持显示算子级实时监控信息。该版本暂时不支持算子历史信息查询。

 版权所有 © 华为云计算技术有限公司

- operator_realtime级别TopSQL运行时监控对于CN轻量化和存储过程的情况，暂时不支持。另由于算子执行速度较快的原因，对于算子信息的显示会有一定滞后性。
- query级别的作业监控和operator的算子监控中的spill_size字段，由于统计维度不同，会有一定差异，query级别监控的语句实际下盘文件大小，算子监控的是具体算子在逻辑层IO读写的数据量。
- 当GUC参数enable_stream_operator设置为off状态时，算子执行信息存在显示不准的情况。

前提条件

- GUC参数**enable_resource_track**为on（默认为on）。
- GUC参数**resource_track_level**为query、perf、operator_realtime或operator（默认为query）。
- 监控作业的类型为：
  - 优化器估算的执行代价大于或等于**resource_track_cost**取值的作业。
- Cgroups功能正常加载，可通过gs_cgroup -P查看控制组信息。
- GUC参数**enable_track_record_subsql**控制是否记录存储过程、匿名块内部语句。

在上述条件中，**enable_resource_track**为系统级参数，用于设置是否开启资源监控功能。**resource_track_level**为session级参数，可以对某个session的资源监控级别进行灵活设置。这两个参数的设置方法如下表：

表 15-2 设置资源监控信息统计级别

enable_resource_track	resource_track_level	query级别信息	算子级别信息
on(default)	none	不统计	不统计
	query(default)	统计	不统计
	perf	统计	不统计
	operator	统计	统计
on(default)	operator_realtime	统计	统计实时算子监控
off	none/query/operator	不统计	不统计

操作步骤

步骤1 通过视图gs_session_cpu_statistics查询实时CPU信息。

```
SELECT * FROM gs_session_cpu_statistics;
```

步骤2 通过视图gs_session_memory_statistics查询实时memory信息。

```
SELECT * FROM gs_session_memory_statistics;
```

步骤3 通过视图gs_wlm_session_statistics查询当前CN的实时资源。

```
SELECT * FROM gs_wlm_session_statistics;
```

步骤4 通过视图pgxc_wlm_session_statistics查询所有CN的实时资源。

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pgxc_wlm_session_statistics;
```

步骤5 通过视图gs_wlm_operator_statistics查询当前CN作业算子执行实时资源信息。

```
SELECT * FROM gs_wlm_operator_statistics;
```

步骤6 通过视图pgxc_wlm_operator_statistics查询所有CN作业算子执行实时资源信息。

```
SELECT * FROM pgxc_wlm_operator_statistics;
```

步骤7 通过视图pg_session_wlmstat查询当前用户执行作业正在运行时的负载管理信息。

```
SELECT * FROM pg_session_wlmstat;
```

步骤8 通过视图pgxc_wlm_workload_records（动态负载功能开启，即enable_dynamic_workload为on时该视图有效）查询当前用户在每个CN上作业执行时的状态信息。

```
SELECT * FROM pgxc_wlm_workload_records;
```

----结束

15.6 历史 TopSQL

系统提供了资源监控历史视图用来查询历史TopSQL。资源监控历史视图记录了查询作业运行结束时的资源使用情况(包括内存、下盘、CPU时间等)和运行状态信息(包括报错、终止、异常等)以及性能告警信息。但对于由于FATAL、PANIC错误导致查询异常结束时，状态信息列只显示aborted，无法记录详细异常信息。特别的，对于查询解析，优化阶段的状态信息则无法监控。

历史视图具体的对外接口如下表所示：

视图级别	节点范围	查询视图	
query级别/perf级别（推荐使用）	当前CN	历史（内部转储接口，仅显示最近三分钟内结束的语句）	GS_WLM_SESSION_HISTORY
		历史（全部语句）	GS_WLM_SESSION_INFO
	所有CN	历史（内部转储接口，仅显示最近三分钟内结束的语句）	PGXC_WLM_SESSION_HISTORY
		历史（全部语句）	PGXC_WLM_SESSION_INFO
operator级别	当前CN	历史（仅显示最近三分钟内结束的语句）	GS_WLM_OPERATOR_HISTORY
		历史（内部转储接口，全部语句）	GS_WLM_OPERATOR_INFO
	所有CN	历史（仅显示最近三分钟内结束的语句）	PGXC_WLM_OPERATOR_HISTORY
		历史（内部转储接口，全部语句）	PGXC_WLM_OPERATOR_INFO

版权所有 © 华为云计算技术有限公司

说明

- 视图级别取决于资源监控的等级，即参数**resource_track_level**的配置。
- perf和operator级别会影响**GS_WLM_SESSION_STATISTICS**/**PGXC_WLM_SESSION_INFO**中的query_plan和warning字段的取值，详细内容参见**SQL自诊断**。
- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 实例故障时，历史TopSQL视图有可能记录不全。
- 在某些异常的情况下，历史TopSQL中的状态信息列可能会显示为unknown，其记录的监控信息会导致不准确。
- 历史TopSQL能够记录的SQL语句的规格与实时TopSQL能够记录的SQL语句的规格一致。请参考**实时TopSQL中能够记录的SQL语句的规格**。
- 历史TopSQL只有在GUC参数**enable_resource_record**开启时才会记录数据。
- 查询历史TopSQL Query以及算子级别数据时，仅能通过postgres数据库进行访问。
- 历史TopSQL侧重于查询性能的定位定界辅助分析，不作为审计功能使用，不记录语法分析报错类语句。
- 8.2.1集群版本开始，新增GUC参数resource_track_subsql_duration（默认值为180秒），用于过滤存储过程中执行时间小于该参数的子语句，仅归档执行时间大于该参数的子语句。且从8.2.1版本开始，GUC参数enable_track_record_subsql默认值由off变更为on，默认将会记录存储过程中的子语句。如果一条子语句被记录，那么它必然满足以下几个条件：
  - 语句所在的会话中，参数enable_track_record_subsql开启。
  - 该子语句一定要下推到DN端执行（为避免TopSQL记录过多的子语句，不下推到DN端执行的子语句会被过滤）。
  - 该子语句的执行时间超过所在会话中resource_track_subsql_duration参数值。
- 关于history视图，默认会查询最近3分钟内结束的语句，该视图内部实际查询的是表，属于因性能考虑而实现的临时视图，从8.1.3集群版本开始，TopSQL监控功能中的实时监控和归档功能已有大幅度提升，不再有性能问题查询中不再建议优先使用history视图。
- 从8.1.3集群版本开始，TopSQL实时监控功能几乎对语句性能无影响，完全可以将GUC参数resource_track_cost设置为0来监控所有语句的运行时的信息；而TopSQL历史监控中对语句归档的能力，对语句执行性能也无影响，但当TPS较高时，需要考虑以下两个因素：
  - 记录所有语句的磁盘开销，用户可以将一条语句归档所需要的磁盘空间估算为8KB，按照业务TPS的峰值来计算相应的空间占用情况，来合理调整resource_track_duration和resource_track_subsql_duration参数。
  - 缓存所有语句的内存开销，用户可以将一条语句归档所需要的内存大小估算为16KB，语句批量归档的间隙为5秒，按照业务TPS的峰值来计算所需的内存峰值，计算方法为：5秒*TPS*16KB，需要session_history_memory GUC参数（默认值为100MB）要大于所需值，才能保证所有语句都能被记录。

前提条件

- GUC参数**enable_resource_track**为on（默认为on）。
- GUC参数**resource_track_level**为query、perf或operator（默认为query）。设置方法详见**表15-2**。
- GUC参数**enable_resource_record**为on（默认为on）。
- GUC参数**resource_track_duration**小于作业执行时间和排队时间之和（默认为60s）。
- GUC参数**enable_track_record_subsql**控制是否记录存储过程、匿名块内部语句（默认为on）。
- GUC参数**resource_track_subsql_duration**小于存储过程中内部语句的执行时间（默认为180s）。

版权所有 © 华为云计算技术有限公司

- 监控作业类型为：资源监控实时视图（参见**表15-1**）中记录的作业执行时间和排队时间之和大于或等于**resource_track_duration**的作业。
- Cgroups功能正常加载，可通过gs_cgroup -P查看控制组信息。

操作步骤

步骤1 通过视图gs_wlm_session_history查询当前CN最近执行作业结束后的负载记录。

```
SELECT * FROM gs_wlm_session_history;
```

步骤2 通过视图pgxc_wlm_session_history查询所有CN最近执行作业结束后的负载记录。

```
SELECT * FROM pgxc_wlm_session_history;
```

步骤3 通过数据表gs_wlm_session_info查询当前CN作业执行结束后的负载记录。要查到历史记录，必须保证**enable_resource_record**为on。

```
SELECT * FROM gs_wlm_session_info;
```

- 消耗内存最多的10个Query（可指定查询时间段）。

```
SELECT * FROM gs_wlm_session_info order by max_peak_memory desc limit 10;
SELECT * FROM gs_wlm_session_info WHERE start_time >= '2022-05-15 21:00:00' and finish_time <='2022-05-15 23:30:00' order by max_peak_memory desc limit 10;
```

- 消耗CPU最多的10个Query。

```
SELECT * FROM gs_wlm_session_info order by total_cpu_time desc limit 10;
SELECT * FROM gs_wlm_session_info WHERE start_time >= '2022-05-15 21:00:00' and finish_time <='2022-05-15 23:30:00' order by total_cpu_time desc limit 10;
```

步骤4 通过视图pgxc_wlm_session_info查询所有CN的作业执行结束后的负载记录。要查到历史记录，必须保证**enable_resource_record**为on。

```
SELECT * FROM pgxc_wlm_session_info;
```

- 查询所有CN消耗时间最多的10个query

```
SELECT * FROM pgxc_wlm_session_info order by duration desc limit 10;
```

- 查询已经完成执行的query语句的执行信息。如：查询queryid为76561193695026478的语句执行信息。

```
SELECT * FROM pgxc_wlm_session_info where queryid = '76561193695026478';
```

步骤5 通过函数pgxc_get_wlm_session_info_bytime对视图pgxc_wlm_session_info进行筛选查询，要查到历史记录，必须保证**enable_resource_record**为on。在统计数据量很大的场景中，建议使用该函数进行查询。

说明

DWS集群默认使用时区为UTC时间，与系统时间存在8h时差，请确保数据库时间与系统时间一致后进行以下查询。

- 查询所有CN上开始时间介于“2019-09-10 15:30:00”和“2019-09-10 15:35:00”之间的query，每个CN最多返回10条记录

```
SELECT * FROM pgxc_get_wlm_session_info_bytime('start_time', '2019-09-10 15:30:00', '2019-09-10 15:35:00', 10);
```

- 查询所有CN上结束时间介于“2019-09-10 15:30:00”和“2019-09-10 15:35:00”之间的query，每个CN最多返回10条记录

```
SELECT * FROM pgxc_get_wlm_session_info_bytime('finish_time', '2019-09-10 15:30:00', '2019-09-10 15:35:00', 10);
```

步骤6 通过视图gs_wlm_operator_history查询当前CN作业算子最近执行资源信息。要查到记录，必须保证**resource_track_level**为operator。

```
SELECT * FROM gs_wlm_operator_history;
```


版权所有 © 华为云计算技术有限公司

步骤7 通过视图pgxc_wlm_operator_history查询所有CN作业算子最近执行资源信息。要查到记录，必须保证**resource_track_level**为operator。

```
SELECT * FROM pgxc_wlm_operator_history;
```

步骤8 通过数据表gs_wlm_operator_info查询当前CN作业算子历史执行资源信息。要查到记录，必须保证**resource_track_level**为operator和**enable_resource_record**为on。

```
SELECT * FROM gs_wlm_operator_info;
```

步骤9 通过视图pgxc_wlm_operator_info查询所有CN作业算子历史执行资源信息。要查到记录，必须保证**resource_track_level**为operator和**enable_resource_record**为on。

```
SELECT * FROM pgxc_wlm_operator_info;
```

----结束

说明

- 对于上述的视图信息，由于预设内存的限制，内存中能够保留的数据记录数量有限，实时查询在结束后会导入到历史相关的视图中。关于记录上限，对于query级别视图，当新的需要记录的查询超过内存约束记录数上限时，则当前查询无法记录，下条查询重新进行规则判断；在每个CN上，记query级别历史视图的内存占用（默认100MB）可通过**PG_TOTAL_MEMORY_DETAIL**视图进行查询。
- 对于算子级别视图，当需要记录的查询的plan_node数量加上当前内存中已有的记录数量超过内存约束记录数上限时，则当前查询的所有算子节点不记录，下条查询重新按照算子规则判定。在每个CN上，算子级别视图在内存中可记录的最大实时和历史记录数分别为max_oper_realt_num(当前系统值为56987)，max_oper_hist_num(113975)；记当前用户业务系统的平均每个查询的节点数为num_plan_node，则在每个CN上，实时视图允许客户执行的最大并发数: num_realt_active = max_oper_realt_num/num_plan_node；历史视图允许客户执行的最大并发数：num_hist_active = max_oper_hist_num/(180/run_time)/num_plan_node。
- 如果并发过高，避免需要记录的查询数量超过query级别视图和算子级别视图的值，可以通过参数**session_history_memory**修改历史查询视图的内存，内存增大和查询数量成正比。

15.7 TopSQL 查询示例

本章节以查询TPC-DS样例数据的作业为例，演示如何查看**实时TopSQL**和**历史TopSQL**。

配置集群参数

查询TopSQL资源监控信息之前，需要先配置相关的GUC参数，以便能查询到作业的资源监控历史信息或归档信息。步骤如下：

1. 登录DWS控制台。
2. 在左侧导航栏中，单击“专属集群 > 集群列表”。
3. 在集群列表中找到所需要的集群，单击集群名称，进入“集群详情”页面。
4. 单击“参数修改”页签，并在“参数列表”模块修改GUC参数resource_track_duration的参数值，然后单击“保存”。
5. 在“修改预览”窗口，确认修改无误后，单击“保存”。

说明

enable_resource_record开关打开后，会引起存储空间膨胀及轻微性能影响，不用时请关闭。

版权所有 © 华为云计算技术有限公司

TopSQL 查询示例

本示例以TPC-DS样例数据为例。

步骤1 打开SQL客户端工具，连接到您的数据库。

步骤2 使用explain语句查询所要执行的SQL语句的预估代价，从而可以确定该SQL语句是否会进行资源监控。

默认执行代价大于**resource_track_cost**的查询才会进行资源监控，用户才可以查询到相关的资源监控信息。

例如，执行如下语句查询该SQL语句的预估执行代价：

```
SET CURRENT_SCHEMA = tpcds;
EXPLAIN WITH customer_total_return AS
( SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
FROM store_returns, date_dim
WHERE sr_returned_date_sk = d_date_sk AND d_year =2000
GROUP BY sr_customer_sk, sr_store_sk )
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

查询结果如下所示，第一行的E-costs列的值即为当前语句的预估代价。

图 15-1 explain 查询结果

```
id |                                operation                                 | E-rows | E-width | E-costs
---+--------------------------------------------------------------------------+--------+---------+---------
 1 | ->  Row Adapter                                                          |      6 |      20 | 153.06
 2 |    ->  Vector Limit                                                      |      6 |      20 | 153.06
 3 |       ->  Vector Streaming (type: GATHER)                                |      6 |      20 | 153.06
 4 |          ->  Vector Limit                                                |      6 |      20 | 152.84
 5 |             ->  Vector Sort                                              |      6 |      20 | 152.84
 6 |                ->  Vector Hash Join (7,26)                               |      6 |      20 | 152.83
 7 |                   ->  Vector Streaming(type: REDISTRIBUTE)               |      6 |       4 | 134.57
 8 |                      ->  Vector Hash Join (9,18)                         |      6 |       4 | 134.46
 9 |                         ->  Vector Hash Join (10,11)                     |      1 |      44 | 97.33
10 |                            ->  CStore Scan on store                      |      1 |       4 | 60.23
11 |                            ->  Vector Subquery Scan on ctr1              |      6 |      40 | 37.07
12 |                               ->  Vector Hash Aggregate                  |      6 |      54 | 37.06
13 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
14 |                                     ->  Vector Hash Join (15,17)         |      6 |      22 | 37.00
15 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
16 |                                           ->  CStore Scan on date_dim    |      1 |       4 | 18.06
17 |                                        ->  CStore Scan on store_returns  |     60 |      26 | 18.02
18 |                         ->  Vector Hash Aggregate                        |      6 |      68 | 37.09
19 |                            ->  Vector Subquery Scan on ctr2              |      6 |      36 | 37.07
20 |                               ->  Vector Hash Aggregate                  |      6 |      54 | 37.06
21 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
22 |                                     ->  Vector Hash Join (23,25)         |      6 |      22 | 37.00
23 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
24 |                                           ->  CStore Scan on date_dim    |      1 |       4 | 18.06
25 |                                        ->  CStore Scan on store_returns  |     60 |      26 | 18.02
26 |                   ->  CStore Scan on customer                            |     60 |      24 | 18.02
(26 rows)
```

本示例为了演示TopSQL的资源监控功能，需要将resource_track_cost参数设置为比explain查询结果中的预估代价小的一个值，即将resource_track_cost参数设置为100，设置方法请参见**resource_track_cost**。

说明

在完成本示例后，仍然要将**resource_track_cost**设置为原始的默认值100000或者一个比较合理的值，否则参数值太小会影响数据库性能。

 版权所有 © 华为云计算技术有限公司

步骤3 执行SQL语句。

```
SET CURRENT_SCHEMA = tpcds;
WITH customer_total_return AS
(SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
FROM store_returns,date_dim
WHERE sr_returned_date_sk = d_date_sk
AND d_year =2000
GROUP BY sr_customer_sk ,sr_store_sk)
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

步骤4 在SQL语句执行期间，查询该条SQL语句在当前CN上的Memory峰值的实时信息。

```
SELECT query,max_peak_memory,average_peak_memory,memory_skew_percent FROM
gs_wlm_session_statistics ORDER BY start_time DESC;
```

含义：查询**query级别**的SQL语句的Memory峰值**实时信息**（语句在所有DN上的每秒最大Memory峰值，在所有DN上的每秒平均Memory峰值，在DN间的Memory倾斜率）

实时TopSQL资源监控信息的更多查询示例，请参见**实时TopSQL**。

步骤5 等待**步骤3**中的SQL执行完成，然后查询该语句执行期间的资源监控历史信息。

```
SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_history ORDER BY start_time
desc;
```

含义：查询**query级别**的SQL语句执行期间的**历史信息**（语句执行的开始时间，结束时间，实际执行时间，执行状态），时间单位为ms。

历史TopSQL资源监控信息的更多查询示例，请参见**历史TopSQL**。

步骤6 等待**步骤3**中的SQL执行结束的三分钟后，在info视图中查询该语句的资源监控历史信息。

如果设置参数enable_resource_record为“on”，且**步骤3**中SQL的执行时间不小于resource_track_duration所设置的值，该条语句的历史信息将会在三分钟后被归档到gs_wlm_session_info视图中。

对于info视图，只支持在连接postgres数据库时查询。因此，请切换为连接postgres数据库后，再执行以下语句进行查询：

```
SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_info ORDER BY start_time desc;
```

----结束

 版权所有 © 华为云计算技术有限公司

16 DWS 性能调优

16.1 性能调优概述

数据库性能调优是指通过优化数据库系统的配置及SQL查询，以提高数据库性能和效率的过程。目的为消除性能瓶颈、减少响应时间、提高系统吞吐量和资源利用率，降低业务成本，从而提高系统稳定性，给用户带来更大的价值。

本章通过性能诊断、系统调优及SQL调优及常见SQL调优案例等性能调优的实际操作，为数据库性能调优人员提供全方位的指导。

注意事项

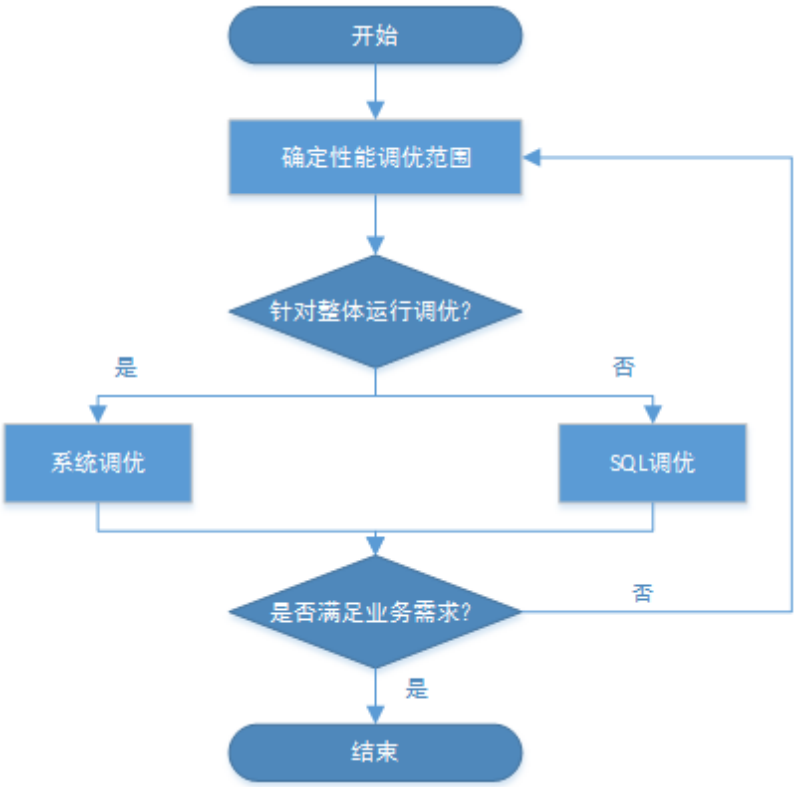
- 数据库调优是一个复杂和细致的过程，需熟悉数据库系统的内部工作原理和相关技术。它需要综合考虑硬件、软件、查询、配置和数据结构等多个方面的因素，以达到最佳的性能和效率。因此，要求调优人员应对系统软件架构、软硬件配置、数据库配置参数、并发控制、查询处理和数据库应用有广泛而深刻的理解。
- 性能调优过程有时需要重启集群，可能会中断当前业务。建议在业务低峰期进行需要重启集群的性能调优操作，避免业务异常中断。

调优流程

调优流程如图16-1所示。

 版权所有 © 华为云计算技术有限公司

图 16-1 DWS 性能调优流程



调优各阶段说明，如表16-1所示。

表 16-1 DWS 性能调优流程说明

阶段	描述
性能诊断	获取集群各节点的CPU、内存、I/O和网络资源使用情况，确认这些资源是否已被充分利用，是否存在瓶颈点。
系统调优	进行操作系统级以及数据库系统级的调优，更充分地利用机器的CPU、内存、I/O和网络资源，避免资源冲突，提升整个系统查询的吞吐量。

版权所有 © 华为云计算技术有限公司

阶段	描述
SQL调优	审视业务所用SQL语句是否存在可优化空间，包括： • 通过ANALYZE语句生成表统计信息：ANALYZE语句可收集与数据库中表内容相关的统计信息，统计结果存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以确定最有效的执行计划。 • 分析执行计划：EXPLAIN语句可显示SQL语句的执行计划，EXPLAIN PERFORMANCE语句可显示SQL语句中各算子的执行时间。 • 查找问题根因并进行调优：通过分析执行计划，找到可能存在的原因，进行针对性的调优，通常为调整数据库级SQL调优参数。 • 编写更优的SQL：介绍一些复杂查询中的中间临时数据缓存、结果集缓存、结果集合并等场景中的更优SQL语法。

16.2 性能诊断

16.2.1 集群性能分析

DWS不同集群规格的CPU核数、内存大小和节点存储容量不同，处理业务能力和性能也就不同，用户在创建集群前需要结合实际业务量和具体使用场景来选择集群规格。

在使用集群过程中，当用户的业务量过大，则需要更多的资源（CPU、内存、网络带宽等）来支撑逐渐增长的业务量，如果用户当前使用的集群资源不足，会降低数据库运行速度并影响数据库性能。

DWS监控信息功能提供了丰富的监控指标，您可以通过集群的各项监控指标（CPU使用率、内存使用率、磁盘使用率、磁盘I/O、网络I/O等），掌握集群的性能和运行状况。当您发现监控指标存在异常时，可以通过查看集群监控指标排查出现异常的原因。

如果需要更多的计算资源或存储资源以满足业务需要时，可以在管理控制台对已有集群，进行规格变更或扩容操作。

16.2.2 慢 SQL 分析

16.2.2.1 查询最耗性能的 SQL

系统中有些SQL语句运行了很长时间还没有结束，这些语句会消耗很多的系统性能，请根据本章内容查询长时间运行的SQL语句。

操作步骤

步骤1 查询系统中长时间运行的查询语句。

```
SELECT current_timestamp - query_start AS runtime, datname, usename, query FROM pg_stat_activity
where state != 'idle' ORDER BY 1 desc;
```


版权所有 © 华为云计算技术有限公司

查询后会按执行时间从长到短顺序返回查询语句列表，第一条结果就是当前系统中执行时间最长的查询语句。返回结果中包含了系统调用的SQL语句和用户执行SQL语句，请根据实际找到用户执行时间长的语句。

若当前系统较为繁忙，可以通过限制current_timestamp - query_start大于某一阈值来查看执行时间超过此阈值的查询语句。

```
SELECT query FROM pg_stat_activity WHERE current_timestamp - query_start > interval '1 days';
```

步骤2 设置参数track_activities为on。

```
SET track_activities = on;
```

当此参数为on时，数据库系统才会收集当前活动查询的运行信息。

步骤3 查看正在运行的查询语句。

以查看视图pg_stat_activity为例：

```
SELECT datname, usename, state FROM pg_stat_activity;
 datname  | usename | state  |
----------+---------+--------+
 postgres |   omm   | idle   |
 postgres |   omm   | active |
(2 rows)
```

如果state字段显示为idle，则表明此连接处于空闲，等待用户输入命令。

如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pg_stat_activity WHERE state != 'idle';
```

步骤4 分析长时间运行的查询语句状态。

- 若查询语句处于正常状态，则等待其执行完毕。
- 若查询语句阻塞，则通过如下命令查看当前处于阻塞状态的查询语句：

  ```
  SELECT datname, usename, state, query FROM pg_stat_activity WHERE waiting = true;
  ```

  查询结果中包含了当前被阻塞的查询语句，该查询语句所请求的锁资源可能被其他会话持有，正在等待持有会话释放锁资源。

  说明

  只有当查询阻塞在系统内部锁资源时，waiting字段才显示为true。尽管等待锁资源是数据库系统最常见的阻塞行为，但是在某些场景下查询也会阻塞在等待其他系统资源上，例如写文件、定时器等。但是这种情况的查询阻塞，不会在视图pg_stat_activity中体现。

----结束

16.2.2.2 分析作业是否被阻塞

数据库系统运行时，在某些业务场景下查询语句会被阻塞，导致语句运行时间过长，可以强制结束有问题的会话。

操作步骤

步骤1 查看阻塞的查询语句及阻塞查询的表、模式信息。

```
SELECT w.query as waiting_query,
w.pid as w_pid,
w.usename as w_user,
l.query as locking_query,
l.pid as l_pid,
l.usename as l_user,
t.schemaname || '.' || t.relname as tablename
from pg_stat_activity w join pg_locks l1 on w.pid = l1.pid
```


版权所有 © 华为云计算技术有限公司

```
and not l1.granted join pg_locks l2 on l1.relation = l2.relation
and l2.granted join pg_stat_activity l on l2.pid = l.pid join pg_stat_user_tables t on l1.relation = t.relid
where w.waiting;
```

该查询返回线程ID、用户信息、查询状态，以及导致阻塞的表、模式信息。

步骤2 使用如下命令结束相应的会话。其中，139834762094352为线程ID。

```
SELECT PG_TERMINATE_BACKEND(139834762094352);
```

显示类似如下信息，表示结束会话成功。

```
 PG_TERMINATE_BACKEND
----------------------
 t
(1 row)
```

显示类似如下信息，表示用户正在尝试结束当前会话，此时仅会重连会话，而不是结束会话。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

说明

gsql客户端使用PG_TERMINATE_BACKEND函数终止本会话后台线程时，客户端不会退出而是自动重连。

----结束

16.2.3 SQL 诊断

DWS集群提供了SQL诊断功能，支持通过多种条件检索出符合条件的SQL查询（如慢查询），并完整展示执行计划。

SQL诊断功能使用方式：

步骤1 登录DWS管理控制台。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的**“操作”**列，单击**“监控面板”**，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择**“工具 > SQL诊断”**，进入SQL诊断页面。

步骤5 在SQL诊断页面您可查看SQL的诊断信息。指标详情请参见**表16-2**。

表 16-2 SQL 诊断指标说明

指标名称	指标说明
查询ID	语句执行使用的内部query_ID。
数据库	所属数据库名称。
模式名	所属模式名称。
用户名称	所属用户名称。
客户端	连接到后端的应用名。

版权所有 © 华为云计算技术有限公司

指标名称	指标说明
客户端IP地址	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者是内部进程，如autovacuum。
运行时间（ms）	语句实际执行的时间，单位ms。
CPU时间（ms）	语句在所有DN上的CPU总时间，单位ms。
开始时间	语句执行开始时间。
完成时间	语句执行结束时间。
详情	单击“**查看**”键，可浏览到SQL的详细诊断结果。其中包括： ● 告警信息：该SQL语句的告警信息。 ● SQL语句：该SQL语句。 ● 执行计划：该SQL语句的执行计划分析。

----结束

16.2.4 表诊断

DWS提供了集群中数据表关键运行状态的统计数据与诊断工具。其中包括：

- 表倾斜率：对于集群中数据表统计信息进行监控分析，展示倾斜率高于5%且表大小TOP50的表信息。
- 表脏页率：对于集群中数据表统计信息进行监控分析，展示脏页率高于50%且表大小TOP50的表信息。

表倾斜率

不合理的分布列选择，将引发算子计算/数据下盘倾斜严重，导致不同DN的处理压力不同，影响业务性能，并容易造成单DN磁盘使用率过高。用户可通过查询表倾斜率，根据表的大小和倾斜率，对倾斜严重的表重新选择分布列。

操作步骤

步骤1 登录DWS管理控制台。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的“**操作**”列，单击“**监控面板**”，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择“**工具 > 表诊断 > 表倾斜率**”，页面将展示集群中符合统计条件的表信息（倾斜率高于5%且表大小Top50的表数据）。

该界面表倾斜率的算法和后台的算法不同，该算法为：**表倾斜率 =（表在各DN占空间的最大值-表在各DN占空间的最小值）/ 表在各DN占空间的平均值**。

其中该表在各DN占空间的最大值/最小值/平均值是通过查询视图**gs_table_distribution**的“**dnsize**”（表在该节点上的存储空间大小）列得出。

版权所有 © 华为云计算技术有限公司

步骤5 （可选）在表倾斜率页面单击**“一键导出”**，将会导出集群中符合统计条件的表信息到CSV文件，可直接在本地打开查看。

----结束

表脏页率

对于数据表的DML操作将影响数据表数据导致存在无用的脏数据， 过多的脏数据将占据磁盘空间，影响集群可用容量。用户可通过查询表的脏页率，根据表的大小和脏页率，对较大表和脏页率过高的表进行处理。

操作步骤

步骤1 登录DWS管理控制台。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的**“操作”**列，单击**“监控面板”**，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择**“工具 > 表诊断 > 表脏页率”**，页面将展示集群中符合统计条件的表信息。

步骤5 （可选）在表脏页率页面单击**“一键导出”**，将会导出集群中符合统计条件的表信息到CSV文件，可直接在本地打开查看。

----结束

16.3 系统调优

16.3.1 数据库系统参数调优

为了保证数据库尽可能高性能地运行，建议依据资源情况和业务实际进行数据库系统GUC参数的设置。本章节旨在介绍一些常用参数以及推荐配置，关于参数的详细设置方法请参考**设置GUC参数**。

版权所有 © 华为云计算技术有限公司

数据库内存相关参数

表 16-3 数据库内存相关参数

GUC参数	描述	建议
max_process_memory	max_process_memory设置单个CN/DN可用的最大物理内存。	• DN上该数值需要根据系统物理内存及单节点部署主DN个数决定的。计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.8 / (n+主DN个数)。该参数目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min_free_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即， max_process_memory=物理内存*0.8/(n+主DN个数)，其中，当集群规模小于256时，n=1；当集群规模大于256且小于512时， n=2；当集群规模超过512时，n=3。 • CN上该数值内存可设置与DN数值一样。 • RAM：集群规划时分配给集群的最大使用内存。
shared_buffers	设置DWS使用的共享内存大小。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。	建议设置shared_buffers值为内存的40%以内。行存列存分开对待。行存设大，列存设小。列存：(单服务器内存/单服务器DN个数)*0.4*0.25。 如果设置较大的shared_buffers需要同时增加checkpoint_segments的值，因为写入大量新增、修改数据需要消耗更多的时间周期。
cstore_buffers	设置列存和OBS、HDFS外表列存格式（orc、parquet、carbondata）所使用的共享缓冲区的大小。	列存表使用cstore_buffers设置的共享缓冲区，几乎不用shared_buffers。因此在列存表为主的场景中，应减少shared_buffers，增加cstore_buffers。 OBS、HDFS外表使用cstore_buffers设置ORC、Parquet、Carbondata的元数据和数据的缓存，元数据缓存大小为cstore_buffers的1/4，最大不超过2GB，其余缓存空间为列存数据和外表列存格式数据共享使用。

 版权所有 © 华为云计算技术有限公司

GUC参数	描述	建议
work_mem	设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。	该参数默认小规格内存为512MB，大规格内存为2GB（max_process_memory大于等于30GB为大规格内存，否则为小规格内存）。 建议参数work_mem依据查询特点和并发来确定，一旦work_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。 • 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work_mem=50%内存/10。 • 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work_mem=50%内存/5。 • 对于并发场景，建议work_mem=串行下的work_mem/物理并发数。
maintenance_work_mem	设置维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。	建议设置此参数的值等于work_mem，可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。 当自动清理进程运行时，autovacuum_max_workers倍数的内存将会被分配，所以此时设置maintenance_work_mem的值应该不小于work_mem。

 版权所有 © 华为云计算技术有限公司

数据库并发队列相关参数

GUC参数	描述	建议
max_active_statements（全局并发队列）	控制单个CN上运行并发执行的作业数量。	采用全局并发队列机制将控制所有普通用户的执行作业，不区分复杂度，即执行语句都将作为一个执行单元，当并发执行的作业数量达到此参数阈值时，将进入队列等待。对于管理员执行的作业，不受全局并发控制的限制。 需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持最大限度的并发作业，且防止并发执行作业过多，引起系统崩溃。
parctl_min_cost（局部并发队列）	控制在一个CN上同一资源池内的并发作业数量。	局部并发控制机制根据执行作业的cost，控制复杂查询的并发作业数量。

说明

全局并发队列参数max_active_statements调整需注意：

- 如果参数max_active_statements设置为-1，也就是不限制全局并发数，会导致通信大并发下存在连接断开报错。
- 在点查询的场景下，参数max_active_statements建议设置为100。
- 在分析类查询的场景下，参数max_active_statements的值设置为CPU的核数除以DN个数，一般可以设置5~8个。

数据库通信相关参数

数据库集群默认使用TCP代理通信库进行节点间通信。

表 16-4 数据库通信相关参数

GUC参数	描述	建议
comm_quota_size	参数comm_quota_size控制每个流通道每次数据传输的大小，默认数值1M。	大并发场景下，提升该数值时，可提升通信性能，但消耗更多的内存，需要根据实际场景进行调优。若通过查询DN的视图pg_total_memory_detail，发现通信层使用内存已达参数comm_usable_memory的阈值时，需要减少该数值，如修改为512K。

版权所有 © 华为云计算技术有限公司

GUC参数	描述	建议
comm_usable_memory	参数comm_usable_memory用于控制数据库传输时，整个DN上可以用于通信的内存使用量。	该参数的取值并不是用于总量控制，仅作为内存流控的手段，即默认流控数值为1M，若内存使用量超过了该参数取值的1/2时，流控数值自动调整为默认流控的一半，若仅剩20%时，则流控数值改为最小流控值8K。

数据库连接相关参数

表 16-5 数据库连接相关参数

GUC参数	描述	建议
max_connections	允许和数据库连接的最大并发连接数。此参数会影响集群的并发能力。	CN中此参数建议保持默认值。DN中此参数建议设置为CN的个数乘以CN中此参数的值。 增大这个参数可能导致DWS要求更多的System V共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。
max_prepared_transactions	设置可以同时处于“预备”状态的事务的最大数目。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。	max_connections取值的设置受max_prepared_transactions的影响，在设置max_connections之前，应确保max_prepared_transactions的值大于或等于max_connections的值，这样可确保每个会话都有一个等待中的预备事务。
session_timeout	客户端连接数据库后处于空闲状态时会根据该参数的默认值自动断开连接。	0-86400，最小单位为秒（s），0表示关闭超时设置。一般不建议设置为0。

版权所有 © 华为云计算技术有限公司

其他性能相关参数

表 16-6 其他性能相关参数

GUC参数	描述	建议
enable_dynamic_workload	设置是否开启动态负载管理。 动态负载管理指数据库内部根据用户负载情况，自动对复杂查询进行队列控制，不再需要手动设置参数，做到系统参数免调优。	该参数默认打开，需要注意以下几点： ● 简单查询作业（估算值<32MB）、非DML（即非INSERT、UPDATE、DELETE和SELECT）语句，不走自适应负载，需要通过max_active_statements来进行单CN的上限控制。 ● 在自适应负载特性下，参数work_mem的默认数值不能变大，否则会引起内存不受控（例如未做Analyze的语句）。 ● 以下场景或语句由于内存使用的特殊性和不确定性，可能导致大并发场景内存不受控，遇到需要降低并发数： – 单条元组占用内存过大，例如，基表包含超过MB级别的宽列。 – 完全下推语句的查询。 – 需要在CN上耗费大量内存的语句，例如，不能下推的语句、withhold cursor场景。 – 由于计划生成不当导致hashjoin算子建立的hash表重复值过多，占用大量内存。 – 包含UDF，且UDF中使用大量内存。 ● 该参数可配合query_dop=0使用，当query_dop设置为0（自适应），系统会根据资源情况和计划特征，动态为每个查询选取最优并行度。enable_dynamic_workload参数会动态分配内存。
bulk_write_ring_size	数据并行导入使用的环形缓冲区大小。	该参数主要影响入库性能，建议导入压力大的场景增加DN上的该参数配置。 默认值为2GB。
data_replicate_buffer_size	发送端与接收端传递数据页时，队列占用内存的大小。	该参数会影响主备之间复制的缓冲大小。 CN默认值为16MB，DN默认值为128MB。若服务器内存为256G，可适当增大到512MB。

 版权所有 © 华为云计算技术有限公司

16.3.2 SMP 并行执行

在复杂查询场景中，单个查询的执行较长，系统并发度低，通过SMP并行执行技术实现算子级的并行，能够有效减少查询执行时间，提升查询性能及资源利用率。

通过算子并行来提升性能，同时会占用更多的系统资源，包括CPU、内存、网络、I/O等等。本质上SMP是一种以资源换取时间的方式，在合适的场景以及资源充足的情况下，能够起到较好的性能提升效果；但是如果在不合适的场景下，或者资源不足的情况下，反而可能引起性能的劣化。同时，生成SMP需要考虑更多的候选计划，将会导致生成时间较长，相比串行场景也会引起性能的劣化。

DWS的SMP特性由GUC参数query_dop控制，该参数可设置用户自定义的查询并行度。

SMP 适用场景与限制

SMP适用场景：

- 支持并行的算子

  计划中存在以下算子支持并行：

  a. Scan：支持行存普通表和行存分区表顺序扫描、列存普通表和列存分区表顺序扫描、HDFS内外表顺序扫描；支持GDS数据导入的外表扫描并行。以上均不支持复制表。

  b. Join：HashJoin、NestLoop

  c. Agg：HashAgg、SortAgg、PlainAgg、WindowAgg(只支持partition by，不支持order by)

  d. Stream：Redistribute、Broadcast

  e. 其他：Result、Subqueryscan、Unique、Material、Setop、Append、VectoRow、RowToVec

- SMP特有算子

  为了实现并行，新增了并行线程间的数据交换Stream算子供SMP特性使用。以下新增的算子可以看做Stream算子的子类：

  a. Local Gather：实现DN内部并行线程的数据汇总

  b. Local Redistribute：在DN内部各线程之间，按照分布键进行数据重分布

  c. Local Broadcast：将数据广播到DN内部的每个线程

  d. Local RoundRobin：在DN内部各线程之间实现数据轮询分发

  e. Split Redistribute：在集群跨DN的并行线程之间实现数据重分布

  f. Split Broadcast：将数据广播到集群所有DN的并行线程

  上述新增算子可以分为Local与非Local两类，Local类算子实现了DN内部并行线程间的数据交换，而非Local类算子实现了跨DN的并行线程间的数据交换。

- 示例说明

  以TPCH Q1的并行计划为例：

 版权所有 © 华为云计算技术有限公司

```
 id |                                 operation                                  |
----+----------------------------------------------------------------------------+
  1 | ->  Row Adapter                                                            |
  2 |    ->  Vector Streaming (type: GATHER)                                     |
  3 |       ->  Vector Sort                                                      |
  4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/4)                 |
  5 |             ->  Vector Hash Aggregate                                      |
  6 |                ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 4/4)     |
  7 |                   ->  Vector Hash Aggregate                                |
  8 |                      ->  Vector Append(9, 10)                              |
  9 |                         ->  Dfs Scan on lineitem                           |
 10 |                         ->  Vector Adapter                                 |
 11 |                            ->  Seq Scan on pg_delta_1423863972 lineitem    |
(11 rows)
```

在这个计划中，实现了Hdfs Scan以及HashAgg算子的并行，并且新增了Local Gather和Split Redistribute数据交换算子。

其中6号算子为Split Redistribute算子，上面标有的“dop: 4/4”表明Split Redistribute的发送端和接收端线程的并行度均为4。4号算子为Local Gather，上面标有“dop: 1/4”，该算子的发送端线程并行度为4，而接收端线程并行度为1，即下层的5号Hash Aggregate算子按照4并行度执行，而上层的1~3号算子按照串行执行，4号算子实现了DN内并行线程的数据汇总。

通过计划Stream算子上标明的dop信息即可看出各个算子的并行情况。

非适用场景：

1. 生成计划时间占比很高的短查询场景。
2. 不支持CN上的算子并行。
3. 不支持不能下推的查询并行执行。
4. 不支持子查询subplan的并行，以及包含子查询的算子并行。

资源对 SMP 性能的影响

SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，包括CPU、内存、I/O和网络带宽等资源的消耗都会出现明显的增长，而且随着并行度的增大，资源消耗也随之增大。当上述资源成为瓶颈的情况下，SMP无法提升性能，反而可能导致集群整体性能的劣化。SMP支持自适应特性，该特性会根据当前资源和查询特征，动态选取最优的并行度。下面对各种资源对SMP性能的影响情况分别进行说明：

- **CPU资源**

  在一般客户场景中，系统CPU利用率不高的情况下，利用SMP并行架构能够更充分地利用系统CPU资源，提升系统性能。但当数据库服务器的CPU核数较少，CPU利用率已经比较高的情况下，如果打开SMP并行，不仅性能提升不明显，反而可能因为多线程间的资源竞争而导致性能劣化。

- **内存资源**

  查询并行后会导致内存使用量的增长，但每个算子使用内存上限仍受到work_mem等参数的限制。假设work_mem为4GB，并行度为2，那么每个并行线程所分配到的内存上限为2GB。在work_mem较小或者系统内存不充裕的情况下，使用SMP并行后，可能出现数据下盘，导致查询性能劣化的问题。

- **网络带宽资源**

  为了实现查询并行执行，会新增并行线程间的数据交换算子。对于Local类Stream算子，所需要进行数据交换的线程在同一个DN内，通过内存交换，不会增加网络

 版权所有 © 华为云计算技术有限公司

负担。而非Local类算子，需要通过网络进行数据交换，因此会加重网络负担。当网络资源成为瓶颈的情况下，并行可能会导致一定程度的劣化。

- **I/O资源**

  要实现并行扫描必定会增加I/O的资源消耗，因此只有在I/O资源充足的情况下，并行扫描才能够提高扫描性能。

其他因素对 SMP 性能的影响

除了资源因素外，还有一些因素也会对SMP并行性能造成影响。例如分区表中分区数据不均，以及系统并发度等因素。

- **数据倾斜对SMP性能的影响**

  当数据中存在严重数据倾斜时，并行效果较差。例如某表join列上某个值的数据量远大于其他值，开启并行后，根据join列的值对该表数据做hash重分布，使得某个并行线程的数据量远多于其他线程，造成长尾问题，导致并行后效果差。

- **系统并发度对SMP性能的影响**

  SMP特性会增加资源的使用，而在高并发场景下资源剩余较少。所以，如果在高并发场景下，开启SMP并行，会导致各查询之间严重的资源竞争问题。一旦出现了资源竞争的现象，无论是CPU、I/O、内存或者网络资源，都会导致整体性能的下降。因此在高并发场景下，开启SMP经常不能达到性能提升的效果，甚至可能引起性能劣化。

SMP 相关参数配置建议

如果要打开SMP自适应功能，要设置query_dop=0，需同步调整以下相关参数值，以获取更佳的dop选择：

- comm_usable_memory

  当系统内存较大时，max_process_memory设置较大，可适当调大该值，建议设置为max_process_memory的5%，默认值为4GB。

- comm_max_stream

  设置建议值为：comm_max_stream=Min(dop_limit * dop_limit * 20 * 2, max_process_memory(字节数) * 0.025 / 总DN数 / 260)，且该值在comm_max_stream取值范围内。

- max_connections

  设置建议值为：max_connections=dop_limit * 20 * 6 + 24，且该值在max_connections取值范围内。

⚠ 注意

公式中的dop_limit为集群中每个DN对应的CPU数，计算公式为：dop_limit = 单机器的CPU逻辑核数 / 单机器的DN数。

版权所有 © 华为云计算技术有限公司

SMP 配置方式

须知

系统的CPU、内存、I/O和网络带宽等资源充足。SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，当上述资源成为瓶颈的情况下，SMP无法提升性能，反而可能导致性能的劣化。同时，SMP计划的生成时间较串行要长。因此，在短查询为主的TP类业务中，或者出现资源瓶颈的情况下，建议关闭SMP，即设置query_dop=1。

配置步骤：

1. 观察当前系统负载情况，如果系统资源充足（资源利用率小于50%），执行步骤2；否则退出。
2. 设置query_dop=1，利用explain打出执行计划，观察计划是否符合**SMP适用场景与限制**适用场景。如果符合，进入下一步。
3. 设置query_dop=-value，在考虑资源情况和计划特征基础上，限制dop选取的范围为[1,value]。
4. 设置query_dop=value，不考虑资源情况和计划特征，强制选取dop为1或value。
5. 在符合条件的查询语句执行前设置合适的query_dop值，在语句执行结束后关闭query_dop。例如，

```
SET query_dop = 0;
SELECT COUNT(*) FROM t1 GROUP BY a;
......
SET query_dop = 1;
```

说明

- 资源许可的情况下，并行度越高，性能提升效果越好。
- SMP并行度支持会话级设置，推荐客户在执行符合要求的查询前，打开smp，执行结束后，关闭smp。以免在业务峰值时，对业务造成冲击。
- SMP自适应（query_dop<=0）依赖资源管理，如果资源管理禁用，那么只会产生1或2并行度的计划。

16.3.3 配置 LLVM

LLVM（Low Level Virtual Machine）动态编译技术可以为每个查询生成定制化的机器码用于替换原本的通用函数。通过减少实际查询时冗余的条件逻辑判断、虚函数调用并提高数据局域性，从而达到提升查询整体性能的目的。

由于LLVM需要消耗额外的时间预生成IR中间态表示并编译成机器码，因此在小数据量场景或查询本身耗时较少时，可能引起性能的劣化。

LLVM 适用场景与限制

适用场景

- 支持LLVM的表达式。查询语句中存在以下的表达式支持LLVM优化：
  a. Case…when… 表达式
  b. In表达式
  c. Bool表达式 (And/Or/Not)

版权所有 © 华为云计算技术有限公司

d. BooleanTest表达式 (IS_NOT_KNOWN/IS_UNKNOWN/IS_TRUE/IS_NOT_TRUE/IS_FALSE/IS_NOT_FALSE)
e. NullTest表达式 (IS_NOT_NULL/IS_NULL)
f. Operator表达式
g. Function表达式 (lpad, substring, btrim, rtrim, length)
h. Nullif表达式

表达式计算支持的数据类型包括bool，tinyint，smallint，int，bigint，float4，float8，numeric，date，time，timetz，timestamp，timestamptz，interval，bpchar，varchar，text，oid。

仅当表达式出现在以下场景时才会考虑是否使用LLVM动态编译优化：

- 向量化执行引擎中Scan节点的filter；
- Hash Join节点中的complicate hash condition、hash join filter、hash join target；
- Nested Loop节点中的filter、join filter；
- Merge Join节点的merge join filter、merge join target；
- Group节点中的filter表达式。

- 支持LLVM的算子：

  a. Join ： HashJoin
  b. Agg ： HashAgg
  c. Sort

  其中，

  - HashJoin算子仅支持Hash Inner Join，对应的hash cond仅支持int4、bigint、bpchar类型的比较；
  - HashAgg算子仅支持针对bigint、numeric类型的sum及avg操作，且group by语句仅支持int4、bigint、bpchar，text，varchar，timestamp类型操作，同时支持count(*)聚集操作；
  - Sort算子仅支持对int4，bigint，numeric，bpchar，text，varchar数据类型的比较操作；

  除此之外，无法使用LLVM动态编译优化，具体可通过explain performance语句进行查看。

非适用场景：

- 不支持CN上LLVM动态编译优化。
- 不支持小数据量表使用LLVM动态编译优化。
- 不支持生成非向量化执行路径的查询作业。

其他因素对 LLVM 性能的影响

LLVM优化效果不仅依赖于数据库内部具体的实现，还与当前所选择的硬件环境等有关。

- 表达式调用C-函数个数

  数据库内部针对表达式计算并未实现全codegen，即在整个表达式计算中部分表达式实现了codegen，部分直接调用原本的C代码。如果整个表达式计算中后者占据了主要部分，使用LLVM动态编译优化，可能会导致性能劣化。通过设置

 版权所有 © 华为云计算技术有限公司

log_min_messages的级别为DEBUG1可以查看到哪些表达式直接调用了C代码实现。

- 内存资源

  LLVM特性的一个重要思想是保障数据的局域特性，即数据应尽可能的存放在寄存器中。同时应减少数据加载，因此在使用LLVM优化时应设置足够大的work_mem，保证对应使用LLVM优化的执行代码整个过程在内存中实现，否则可能引起性能劣化。

- 优化器代价估算

  LLVM特性实现了简易的代价估算模型，即依据当前参与节点运算的表大小决定当前节点是否考虑使用LLVM动态编译优化。如果优化器低估了实际参与运算的行数，则原本可获得收益的未正常获得收益。反之亦然。

LLVM 使用建议

目前LLVM在数据库内核侧已默认打开，用户可结合上述的分析进行配置，总体建议如下：

1. 设置合理的work_mem，在允许的条件下尽可能设置较大的work_mem，如果出现大量下盘，则建议关闭LLVM动态编译优化（通过设置enable_codegen=off实现）。
2. 设置合理的codegen_cost_threshold(默认值为10000)，确保小数据量场景下避免使用LLVM动态编译优化。当codegen_cost_threshold的值设定后，因使用LLVM动态编译优化引入性能劣化，则建议增加codegen_cost_threshold的取值。
3. 对于表达式计算使用LLVM动态编译优化，如果存在大量的调用C-函数的场景，建议关闭LLVM动态编译优化。
4. In表达式后常量列表长度不能超过10，否则不能执行LLVM编译优化。

   说明

   在资源许可的情况下，数据量越大，可获得的性能提升效果越好。

16.4 SQL 调优

16.4.1 SQL 查询执行流程

SQL引擎从接收SQL语句到执行SQL语句需要经历的步骤如**图16-2**和**表16-7**所示。其中，红色字体部分为DBA可以介入实施调优的环节。

版权所有 © 华为云计算技术有限公司

图 16-2 SQL 引擎执行查询类 SQL 语句的流程

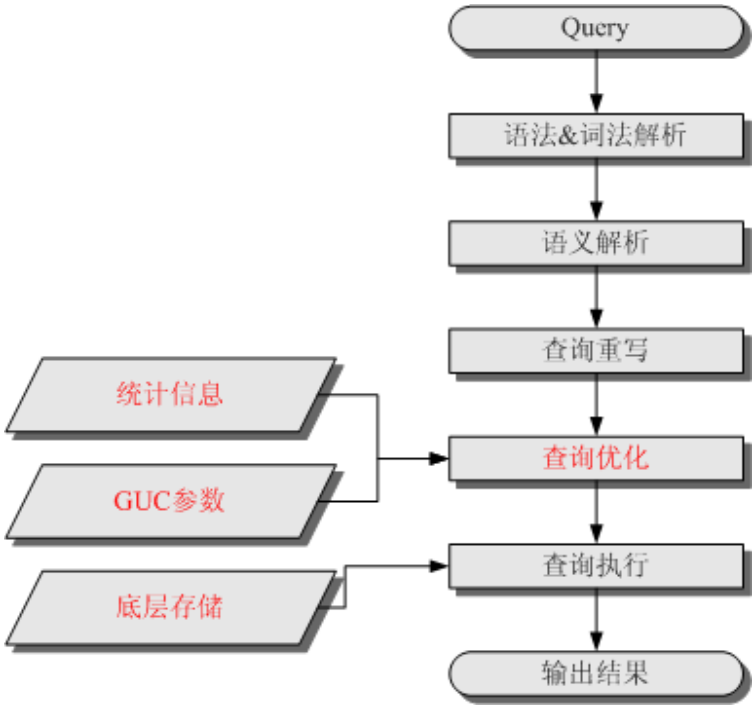


表 16-7 SQL 引擎执行查询类 SQL 语句的步骤说明

步骤	说明
1、语法&词法解析	按照约定的SQL语句规则，把输入的SQL语句从字符串转化为格式化结构(Stmt)。
2、语义解析	将“语法&词法解析”输出的格式化结构转化为数据库可以识别的对象。
3、查询重写	根据规则把“语义解析”的输出等价转化为执行上更为优化的结构。
4、查询优化	根据“查询重写”的输出和数据库内部的统计信息规划SQL语句具体的执行方式，也就是执行计划。统计信息和GUC参数对查询优化（执行计划）的影响，请参见**调优手段之统计信息**和**调优手段之GUC参数**。
5、查询执行	根据“查询优化”规划的执行路径执行SQL查询语句。底层存储方式的选择合理性，将影响查询执行效率。详见**调优手段之底层存储**。

调优手段之统计信息

DWS优化器是典型的基于代价的优化 (Cost-Based Optimization，简称CBO)。在这种优化器模型下，数据库根据表的元组数、字段宽度、NULL记录比率、distinct值、MCV值、HB值等表的特征值，以及一定的代价计算模型，计算出每一个执行步骤的不同执行方式的输出元组数和执行代价（cost），进而选出整体执行代价最小/首元组返回代价最小的执行方式进行执行。这些特征值就是统计信息。从上面描述可以看出统计信息是查询优化的核心输入，准确的统计信息将帮助优化器选择最合适的查询规

 版权所有 © 华为云计算技术有限公司

划，一般来说通过ANALYZE语法收集整个表或者表的若干个字段的统计信息，周期性地运行ANALYZE，或者在对表的大部分内容做了更改之后马上运行它是个好习惯。

调优手段之 GUC 参数

查询优化的主要目的是为查询语句选择高效的执行方式。

如下SQL语句:

```
SELECT count(1)
FROM customer inner join store_sales on (ss_customer_sk = c_customer_sk);
```

在执行customer inner join store_sales的时候，DWS支持Nested Loop、Merge Join和Hash Join三种不同的Join方式。优化器会根据表customer和表store_sales的统计信息估算结果集的大小以及每种join方式的执行代价，然后对比选出执行代价最小的执行计划。

正如前面所说，执行代价计算都是基于一定的模型和统计信息进行估算，当因为某些原因代价估算不能反映真实的cost的时候，就需要通过guc参数设置的方式让执行计划倾向更优规划。

调优手段之底层存储

DWS的表支持行存表、列存表，底层存储方式的选择严格依赖于客户的具体业务场景。一般来说计算型业务查询场景（以关联、聚合操作为主）建议使用列存表；点查询、大批量UPDATE/DELETE业务场景适合行存表。

对于每种存储方式还有对应的存储层优化手段，这部分会在后续的调优章节深入介绍。

调优手段之 SQL 重写

除了上述干预SQL引擎所生成执行计划的执行性能外，根据数据库的SQL执行机制以及大量的实践发现，有些场景下，在保证客户业务SQL逻辑的前提下，通过一定规则由DBA重写SQL语句，可以大幅度的提升SQL语句的性能。

这种调优场景对DBA的要求比较高，需要对客户业务有足够的了解，同时也需要扎实的SQL语句基本功，后续会介绍几个常见的SQL改写场景。

16.4.2 SQL 执行计划

SQL执行计划是一个节点树，显示DWS执行一条SQL语句时执行的详细步骤。

使用EXPLAIN命令可以查看优化器为每个查询生成的具体执行计划。EXPLAIN给每个执行节点都输出一行，显示基本的节点类型和优化器为执行这个节点预计的开销值。

执行计划显示信息

除了设置不同的执行计划显示格式外，还可以通过不同的EXPLAIN用法，显示不同详细程度的执行计划信息。常见有如下几种，关于更多用法请参见《数据仓库服务SQL语法参考》中“EXPLAIN”说明。

- EXPLAIN *statement*： 只生成执行计划，不实际执行。其中statement代表SQL语句。
- EXPLAIN ANALYZE *statement*：生成执行计划，进行执行，并显示执行的概要信息。显示中加入了实际的运行时间统计，包括在每个规划节点内部花掉的总时间(以毫秒计)和它实际返回的行数。

 版权所有 © 华为云计算技术有限公司

- EXPLAIN PERFORMANCE *statement*：生成执行计划，进行执行，并显示执行期间的全部信息。

为了测量运行时在执行计划中每个节点的开销，EXPLAIN ANALYZE或EXPLAIN PERFORMANCE会在当前查询执行上增加性能分析的开销。在一个查询上运行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE有时会比普通查询明显地花费更多的时间。超支的数量依赖于查询的本质和使用的平台。

因此，当定位SQL运行慢问题时，如果SQL长时间运行未结束，建议通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及其实际的运行信息，以便更精准地定位问题原因。

执行计划中的常见关键字说明：

1. 表访问方式
   - Seq Scan/CStore Scan

     全表顺序扫描。最基本的扫描算子，用于行/列存表的顺序扫描。
   - Index Scan/CStore Index Scan

     行/列存表的索引扫描。行/列存表上存在索引，条件列为索引列。

     优化器决定使用两步的规划：最底层的规划节点访问一个索引，找出匹配索引条件的行的位置，然后上层规划节点真实地从表中抓取出对应行。独立地抓取数据行比顺序地读取数据的开销高很多，但是因为并非所有表的页面都被访问了，这么做实际上仍然比一次顺序扫描开销要少。使用两层规划的原因是，上层规划节点在读取索引标识出来的行位置之前，会先将它们按照物理位置排序，这样可以最小化独立抓取的开销。

     如果在WHERE里面使用的好几个字段上都有索引，那么优化器可能会使用索引的AND或OR的组合。但是这么做要求访问两个索引，因此与只使用一个索引，而把另外一个条件只当作过滤器相比，这个方法未必是更优。

     索引扫描可以分为以下几类，其差异在于索引的排序机制。

     - Bitmap Index Scan

       使用位图索引抓取数据页，需要索引扫描获取位图后再到基表上扫描。

     - Index Scan using index_name

       使用简单索引搜索，该方式表的数据行是以索引顺序抓取的，这样就令读取它们的开销更大，但是这里的行少得可怜，因此对行位置的额外排序并不值得。最常见的就是看到这种规划类型只抓取一行，以及那些要求ORDER BY条件匹配索引顺序的查询。因为那时候没有多余的排序步骤是必要的以满足ORDER BY。
2. 表连接方式
   - Nested Loop

     嵌套循环，适用于被连接的数据子集较小的查询。在嵌套循环中，外表驱动内表，外表返回的每一行都要在内表中检索找到它匹配的行，因此整个查询返回的结果集不能太大（不能大于10000），要把返回子集较小的表作为外表，而且在内表的连接字段上建议要有索引。
   - (Sonic) Hash Join

     哈希连接，适用于数据量大的表的连接方式。优化器使用两个表中较小的表，利用连接键在内存中建立hash表，然后扫描较大的表并探测散列，找到与散列匹配的行。Sonic和非Sonic的Hash Join的区别在于所使用hash表结构不同，不影响执行的结果集。

 版权所有 © 华为云计算技术有限公司

- Merge Join

  归并连接或融合连接，是先将关联表的关联列各自做排序，然后从各自的排序表中抽取数据，到另一个排序表中做匹配。

  因为Merge join需要做更多的排序，所以消耗的资源更多，因此通常情况下执行性能差于Hash Join。 如果源数据已经被排序过，在执行融合连接时，并不需要再排序，此时Merge Join的性能优于Hash Join。

3. 运算符
   - sort

     对结果集进行排序。
   - filter

     EXPLAIN输出显示WHERE子句当作一个"filter"条件附属于顺序扫描计划节点。这意味着规划节点为它扫描的每一行检查该条件，并且只输出符合条件的行。预计的输出行数降低了，因为有WHERE子句。不过，扫描仍将必须访问所有 10000 行，因此开销没有降低；实际上它还增加了一些（确切的说，通过10000 * cpu_operator_cost）以反映检查WHERE条件的额外CPU时间。
   - LIMIT

     LIMIT限定了执行结果的输出记录数。如果增加了LIMIT，那么不是所有的行都会被检索到。

执行计划显示格式

DWS对执行计划提供了normal、pretty、summary、run四种显示格式。通过设置GUC参数explain_perf_mode，可以显示不同格式的执行计划。

- normal：代表使用默认的打印格式。图16-3中即为此显示格式。

  图 16-3 normal 格式执行计划示例

```
postgres=# explain select * from test where a < 1;
                            QUERY PLAN
------------------------------------------------------------------
 Streaming (type: GATHER)  (cost=0.25..19.16 rows=7 width=8)
   Node/s: All datanodes
   ->  Seq Scan on test  (cost=0.00..13.16 rows=7 width=8)
         Filter: (a < 1)
(4 rows)
```

- pretty：代表使用DWS改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。如图16-4。

  图 16-4 pretty 格式执行计划示例

```
postgres=# explain select cjxh, count(1) from dwcjk group by cjxh;
 id |              operation               | E-rows | E-memory | E-width | E-costs
----+--------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                      |      1 |          |      52 | 58.42
  2 |    ->  Vector Streaming (type: GATHER) |      1 |          |      52 | 58.42
  3 |       ->  Vector Hash Aggregate      |      1 | 16MB     |      52 | 58.02
  4 |          ->  CStore Scan on dwcjk    |      1 | 1MB      |      44 | 58.00
(4 rows)
```

- summary：是在pretty的基础上增加了对打印信息的分析。
- run：在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

 版权所有 © 华为云计算技术有限公司

常见类型计划

DWS中当前主要存在三类分布式计划：

- FQS(fast query shipping)计划

  CN直接将原语句下发到DN，各DN单独执行，并将执行结果在CN上进行汇总。

- Stream计划

  CN根据原语句生成计划并将计划下发给DN进行执行，各DN执行过程中使用Stream算子进行数据交互。

- Remote-Query计划

  CN生成计划后，将部分原语句下发到DN，各DN单独执行，执行后将结果发送给CN，CN执行剩余计划。

现有表tt01和tt02定义如下：

```
CREATE TABLE tt01(c1 int, c2 int) DISTRIBUTE BY hash(c1);
CREATE TABLE tt02(c1 int, c2 int) DISTRIBUTE BY hash(c2);
```

类型一：FQS计划，完全下推

两表JOIN，且其连接条件为各表的分布列，在关闭stream算子的情况下，CN会直接将该语句发送至各DN执行，最后结果在CN汇总。

```
SET enable_stream_operator=off;
SET explain_perf_mode=normal;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Data Node Scan on "__REMOTE_FQS_QUERY__"
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   Remote query: SELECT tt01.c1, tt01.c2, tt02.c1, tt02.c2 FROM dbadmin.tt01, dbadmin.tt02 WHERE tt01.c1
= tt02.c2
(4 rows)
```

类型二：非FQS计划，部分语句下推

两表JOIN，且连接条件中包含非分布列，此时在关闭stream算子的情况下，CN会将基表扫描语句下发至各DN，然后在CN上进行JOIN。

```
SET enable_stream_operator=off;
SET explain_perf_mode=normal;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c1;
                               QUERY PLAN
-------------------------------------------------------------------------------
 Hash Join
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Hash Cond: (tt01.c1 = tt02.c1)
   ->  Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"
         Output: tt01.c1, tt01.c2
         Node/s: All datanodes
         Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt01 WHERE true
   ->  Hash
         Output: tt02.c1, tt02.c2
         ->  Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_"
               Output: tt02.c1, tt02.c2
               Node/s: All datanodes
               Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt02 WHERE true
(13 rows)
```

类型三：Stream计划，DN之间无数据交换

版权所有 © 华为云计算技术有限公司

两表JOIN，且连接条件为各表的分布列，因此各DN无需数据交换。CN生成stream计划后，将除Gather Stream的计划下发给DN执行，在各个DN上进行基表扫描，并进行哈希连接后，发送给CN。

```
SET enable_fast_query_shipping=off;
SET enable_stream_operator=on;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
                QUERY PLAN
-----------------------------------------------------
 Streaming (type: GATHER)
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   ->  Hash Join
         Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
         Hash Cond: (tt01.c1 = tt02.c2)
         ->  Seq Scan on dbadmin.tt01
               Output: tt01.c1, tt01.c2
               Distribute Key: tt01.c1
         ->  Hash
               Output: tt02.c1, tt02.c2
               ->  Seq Scan on dbadmin.tt02
                     Output: tt02.c1, tt02.c2
                     Distribute Key: tt02.c2
(14 rows)
```

类型四：Stream计划，DN之间存在数据交换

两表JOIN，且连接条件包含非分布列，在开启stream算子(SET enable_stream_operator=on)的情况下，会生成stream计划，其DN间存在数据交换。此时对于tt02表，会在各DN进行基表扫描，扫描后会通过Redistribute Stream算子，按照JOIN条件中的tt02.c1进行哈希计算后重新发送给各DN，然后在各DN上做JOIN，最后汇总到CN。

```
postgres=> SET enable_stream_operator=on;
SET
postgres=> SET enable_fast_query_shipping=off;
SET
postgres=> SET explain_perf_mode=normal;
SET
postgres=> EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c1;
                      QUERY PLAN
-------------------------------------------------------
 Streaming (type: GATHER)
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   ->  Hash Join
         Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
         Hash Cond: (tt02.c1 = tt01.c1)
         ->  Streaming(type: REDISTRIBUTE)
               Output: tt02.c1, tt02.c2
               Distribute Key: tt02.c1
               Spawn on: All datanodes
               Consumer Nodes: All datanodes
               ->  Seq Scan on dbadmin.tt02
                     Output: tt02.c1, tt02.c2
                     Distribute Key: tt02.c2
         ->  Hash
               Output: tt01.c1, tt01.c2
               ->  Seq Scan on dbadmin.tt01
                     Output: tt01.c1, tt01.c2
                     Distribute Key: tt01.c1
(19 rows)
```

类型五：Remote-Query计划场景

因unship_func不能下推，且不满足部分下推要求（子查询下推），所以只能发送基表扫描的语句到DN，将基表数据收集到CN上来计算。

```
postgres=> CREATE FUNCTION unship_func(integer,integer) returns integer
postgres-> AS 'select $1 + $2;'
postgres-> LANGUAGE SQL volatile
postgres-> returns null on null input;
CREATE FUNCTION
```

 版权所有 © 华为云计算技术有限公司

```
postgres=> SET explain_perf_mode=pretty;
SET
postgres=> EXPLAIN VERBOSE SELECT unship_func(tt01.c1,tt01.c2) FROM tt01 JOIN tt02 on tt01.c1=tt02.c1;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 id |                       operation                       | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Hash Join (2,3)                                   |     30 |            |       8 | 0.86
  2 |    ->  Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"  |     30 |            |       8 | 0.00
  3 |    ->  Hash                                           |     30 |            |       4 | 0.00
  4 |       ->  Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_" |     30 |            |       4 | 0.00

            SQL Diagnostic Information
-------------------------------------------------
SQL is not plan-shipping
        reason: Function unship_func() can not be shipped

Predicate Information (identified by plan id)
-----------------------------------------------
  1 --Hash Join (2,3)
        Hash Cond: (tt01.c1 = tt02.c1)

           Targetlist Information (identified by plan id)
-------------------------------------------------------------------------
  1 --Hash Join (2,3)
        Output: (tt01.c1 + tt01.c2)
  2 --Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"
        Output: tt01.c1, tt01.c2
        Node/s: All datanodes
        Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt01 WHERE true
  3 --Hash
        Output: tt02.c1
  4 --Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_"
        Output: tt02.c1
        Node/s: All datanodes
        Remote query: SELECT c1 FROM ONLY dbadmin.tt02 WHERE true

====== Query Summary =====
---------------------------
Parser runtime: 0.055 ms
Planner runtime: 0.528 ms
Unique SQL Id: 1780774145
(37 rows)
```

EXPLAIN PERFORMANCE 详解

在SQL调优过程中经常需要执行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看SQL语句实际执行信息，通过对比实际执行与优化器的估算之间的差别来为优化提供依据。EXPLAIN PERFORMANCE相对于EXPLAIN ANALYZE增加了每个DN上的执行信息。

表定义如下：

```
CREATE TABLE tt01(c1 int, c2 int) DISTRIBUTE BY hash(c1);
CREATE TABLE tt02(c1 int, c2 int) DISTRIBUTE BY hash(c2);
```

以如下SQL查询语句为例：

```
SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
```

执行EXPLAIN PERFORMANCE输出的显示执行信息分为以下8个部分：

1. 执行计划

```
                                                
```

字段	描述
operation	具体的执行节点算子名称。 Vector前缀的算子是指向量化执行引擎算子，一般出现含有列存表的Query中。 Streaming是一个特殊的算子，它实现了分布式架构的核心数据shuffle功能，Streaming共有三种形态，分别对应了分布式结构下不同的数据shuffle功能： ● Streaming (type: GATHER)：作用是coordinator从DN收集数据。 ● Streaming(type: REDISTRIBUTE)：作用是DN根据选定的列把数据重分布到所有的DN。 ● Streaming(type: BROADCAST)：作用是把当前DN的数据广播给其他所有的DN。
A-time	各DN相应算子执行时间，一般DN上执行的算子的A-time是由[]括起来的两个值，分别表示此算子在所有DN上完成的最短时间和最长时间，包括下层算子执行时间。 注意：在整个计划中，除了叶子节点的执行时间是算子本身的执行时间，其余算子的执行时间均包含子节点的执行时间。
A-rows	表示相应算子输出的全局总行数。
E-rows	每个算子估算的输出行数。
E-distinct	表示hashjoin算子的distinct估计值。
Peak Memory	此算子在每个DN上执行时使用的内存峰值，[]中左侧为最小值，右侧为最大值。
E-memory	DN上每个算子估算的内存使用量，只有DN上执行的算子会显示。某些场景会在估算的内存使用量后使用括号显示该算子在内存资源充足下可以自动扩展的内存上限。
A-width	表示当前算子每行元组的实际宽度，仅对于重内存使用算子会显示，包括：(Vec)HashJoin、(Vec)HashAgg、(Vec) HashSetOp、(Vec)Sort、(Vec)Materialize算子等，其中(Vec)HashJoin计算的宽度是其右子树算子的宽度，会显示在其右子树上。
E-width	每个算子输出元组的估算宽度。
E-costs	每个算子估算的执行代价。 ● E-costs是优化器根据成本参数定义的单位来衡量的，习惯上以磁盘页面抓取为1个单位， 其它开销参数将参照它来设置。 ● 每个节点的开销（ E-costs值 ）包括它的所有子节点的开销。 ● 开销只反映了优化器关心的东西，并没有把结果行传递给客户端的时间考虑进去。虽然这个时间可能在实际的总时间里占据相当重要的分量，但是被优化器忽略了，因为它无法通过修改规划来改变。

 版权所有 © 华为云计算技术有限公司

2. SQL Diagnostic Information

   SQL自诊断信息。优化和执行过程中识别到的性能优化点，当对DML语句进行带VERBOSE属性的EXPLAIN（EXPLAIN PERFORMANCE内置自带VERBOSE属性）时，SQL自诊断信息也会输出，以辅助性能问题定位。

3. Predicate Information (identified by plan id)

```
Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
        Hash Cond: (tt01.c1 = tt02.c2)
  3 --Seq Scan on dbadmin.tt01
        Filter: (tt01.c1 >= 202007)
  5 --Seq Scan on dbadmin.tt02
        Filter: (tt02.c2 >= 202007)
```

   谓词过滤这部分主要显示的是对应执行算子节点的过滤条件，即在整个计划执行过程中不会变的信息，主要是一些join条件和一些filter信息。

   8.3.0及以上集群版本支持显示与字典计划相关的CU Predicate Filter和Pushdown Predicate Filter(will be pruned)信息。

4. Memory Information (identified by plan id)

```
          Memory Information (identified by plan id)
------------------------------------------------------------------
Coordinator Query Peak Memory:
        Query Peak Memory: 2MB
DataNode Query Peak Memory
        dn_6001_6002 Query Peak Memory: 0MB
        dn_6003_6004 Query Peak Memory: 0MB
        dn_6005_6006 Query Peak Memory: 0MB
  1 --Streaming (type: GATHER)
        Peak Memory: 56KB, Estimate Memory: 512MB
  2 --Hash Join (3,4)
        dn_6001_6002 Peak Memory: 8KB, Estimate Memory: 1024KB
        dn_6003_6004 Peak Memory: 8KB, Estimate Memory: 1024KB
        dn_6005_6006 Peak Memory: 8KB, Estimate Memory: 1024KB
  3 --Seq Scan on dbadmin.tt01
        dn_6001_6002 Peak Memory: 32KB, Estimate Memory: 1024KB
        dn_6003_6004 Peak Memory: 32KB, Estimate Memory: 1024KB
        dn_6005_6006 Peak Memory: 32KB, Estimate Memory: 1024KB
  4 --Hash
        dn_6001_6002 Buckets: 0  Batches: 0  Memory Usage: 0kB
        dn_6003_6004 Buckets: 0  Batches: 0  Memory Usage: 0kB
        dn_6005_6006 Buckets: 0  Batches: 0  Memory Usage: 0kB
```

   内存使用信息这部分显示的是整个计划中会将内存的使用情况打印出来的算子的内存使用信息，主要是Hash、Sort算子，包括算子峰值内存（peak memory），优化器预估的内存（estimate memory），控制内存（control memory），估算内存使用（operator memory），执行时实际宽度（width），内存使用自动扩展次数（auto spread num），是否提前下盘（early spilled），以及下盘信息，包括重复下盘次数（spill Time(s)），内外表下盘分区数（inner/outer partition spill num），下盘文件数（temp file num），下盘数据量及最小和最大分区的下盘数据量（written disk IO [min, max] ）。其中sort算子不会显示具体的下盘文件数，仅在显示排序方法时显示Disk。

5. Targetlist Information (identified by plan id)

 版权所有 © 华为云计算技术有限公司

```
 Targetlist Information (identified by plan id)
--------------------------------------------------------
 1 --Streaming (type: GATHER)
       Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
       Node/s: All datanodes
 2 --Hash Join (3,4)
       Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
 3 --Seq Scan on dbadmin.tt01
       Output: tt01.c1, tt01.c2
       Distribute Key: tt01.c1
 4 --Hash
       Output: tt02.c1, tt02.c2
 5 --Seq Scan on dbadmin.tt02
       Output: tt02.c1, tt02.c2
       Distribute Key: tt02.c2
```

这一部分显示的是每一个算子对应的输出目标列信息。

8.3.0及以上集群版本支持显示与字典相关的Dict Optimized和Dict Decoded，分别表示字典列和字典编码。

6. DataNode Information (identified by plan id)

```
                                                Datanode Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------------------------------------
 1 --Row Adapter
       (actual time=68637.768..68637.772 rows=1 loops=1)
       (CPU: ex c/r=7589244, ex row=1, ex cyc=7589244, inc cyc=205913250248)
 2 --Vector Aggregate
       (actual time=68635.241..68635.241 rows=1 loops=1) (projection time=0.014)
       (Buffers: shared hit=1)
       (CPU: ex c/r=1675582, ex row=2, ex cyc=3351164, inc cyc=205905661004)
 3 --Vector Streaming (type: GATHER)
       (actual time=54126.686..68634.124 rows=2 loops=1)
       (Buffers: shared hit=1)
       (CPU: ex c/r=102951154920, ex row=2, ex cyc=205902309840, inc cyc=205902309840)
 4 --Vector Aggregate
       dn_6001_6002 (actual time=54076.307..54076.308 rows=1 loops=1) (projection time=65.581)
       dn_6003_6004 (actual time=68597.121..68597.122 rows=1 loops=1) (projection time=64.801)
       dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
       dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
       dn_6001_6002 (CPU: ex c/r=44, ex row=29999729, ex cyc=1346672758, inc cyc=162228870602)
       dn_6003_6004 (CPU: ex c/r=44, ex row=29999728, ex cyc=1343711688, inc cyc=205791299646)
 5 --CStore Scan on public.lineitem_v3
       dn_6001_6002 (actual time=594.261..53628.924 rows=29999729 loops=1) (CU ScanInfo: smallCu: 0, totalCu: 500, avrCuRow: 59999, totalDeadRows: 0)
       dn_6003_6004 (actual time=865.099..68150.512 rows=29999728 loops=1) (CU ScanInfo: smallCu: 0, totalCu: 500, avrCuRow: 59999, totalDeadRows: 0)
       dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
       dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
       dn_6001_6002 (CStore Buffers: shared hit=29500 read=500)
       dn_6003_6004 (CStore Buffers: shared hit=29500 read=500)
       dn_6001_6002 (Disk Cache: miss=576 scanBytes=80MB remoteReadBytes=576MB)
       dn_6003_6004 (Disk Cache: miss=579 scanBytes=80MB remoteReadBytes=579MB)
       dn_6001_6002 (CPU: ex c/r=5362, ex row=29999729, ex cyc=160882197844, inc cyc=160882197844)
       dn_6003_6004 (CPU: ex c/r=6814, ex row=29999728, ex cyc=204447587958, inc cyc=204447587958)
```

这部分将各个算子的执行时间（若包含过滤及投影也会显示对应的执行时间）、CPU、buffer的使用情况全部打印出来。

- 算子执行信息

  ```
  4 --Vector Aggregate
        dn_6001_6002 (actual time=54076.307..54076.308 rows=1 loops=1) (projection time=65.581)
        dn_6003_6004 (actual time=68597.121..68597.122 rows=1 loops=1) (projection time=64.801)
  ```

  每个算子的执行信息都包含三个部分：

  - dn_6001_6002/dn_6003_6004表示具体执行的节点信息，括号中的信息是实际的执行信息。
  - actual time表示实际的执行时间，第一个数字表示执行时进入当前算子到输出第一条数据所花费的时间，第二个数字表示输出所有数据的总执行时间。
  - rows表示当前算子输出数据行数。
  - loops表示当前算子的执行次数。需要注意，对于分区表来说，每一个分区表的扫描就是一次完整的扫描操作，当切换到下一个分区的时候，又是一次新的扫描操作。

- CPU信息

 版权所有 © 华为云计算技术有限公司

```
dn_6001_6002 (CPU: ex c/r=44, ex row=29999729, ex cyc=1346672758, inc cyc=162228870602)
dn_6003_6004 (CPU: ex c/r=44, ex row=29999728, ex cyc=1343711688, inc cyc=205791299646)
```

每个算子执行的过程都有CPU信息，其中cyc代表的是CPU的周期数，ex cyc表示的是当前算子的周期数，不包含其子节点；inc cyc是包含子节点的周期数；ex row是当前算子输出的数据行数；ex c/r则是ex cyc/ex row得到的每条数据所用的平均周期数。

- Buffer信息

```
dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
```

buffers显示缓冲区信息，包括共享块和临时块的读和写。

共享块包含表和索引，临时块在排序和物化中使用的磁盘块。上层节点显示出来的块数据包含了其所有子节点使用的块数。

- 磁盘缓存信息（仅9.1.0.100及以上版本且colversion=3.0的存算分离V3表或外表支持）

```
dn_6001_6002 (Disk Cache: hit=145611 miss=47 error=47 errorCode=152 scanBytes=18325.02MB remoteReadBytes=959.00MB loadTime=13109.5)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1230.6 preloadWaitTime=1724.9 preloadWaitCount=28)
             (OBS I/O: count=349 averageRTT=238.7 averageLatency=233.0)
dn_6003_6004 (Disk Cache: hit=140641 miss=45 error=45 errorCode=152 scanBytes=17508.64MB remoteReadBytes=902.00MB loadTime=13271.7)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1202.9 preloadWaitTime=3061.5 preloadWaitCount=20)
             (OBS I/O: count=327 averageRTT=231.5 averageLatency=226.3 latencyGt1s=1)
dn_6005_6006 (Disk Cache: hit=139736 miss=47 error=47 errorCode=152 scanBytes=17494.91MB remoteReadBytes=909.00MB loadTime=12752.5)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1169.4 preloadWaitTime=2723.4 preloadWaitCount=13)
             (OBS I/O: count=327 averageRTT=194.3 averageLatency=188.8 latencyGt1s=3)
```

Disk Cache：表示磁盘缓存的命中信息和数据读取信息。（存算分离V3表或外表支持）

miss代表磁盘缓存未命中的次数，hit表示磁盘缓存命中的次数，errorCode见**表18-162**，disk_cache_error_code，error表示产生errorCode的次数。scanBytes表示scan查询的数据量，remoteReadBytes表示在OBS上读取的数据量，loadTime表示从磁盘缓存加载数据的时间。为了提升OBS的效率会对相邻的请求块合并，或者因为请求写磁盘缓存的最小粒度是block(默认1M)，可能会使得scanBytes会小于remoteReadBytes。

Column 3.0：存算分离V3表的预读参数与预读过程信息。（仅存算分离V3表支持，开启预读相关参数后显示）

preloadStep表示预读的步长，preloadSubmitTime表示预读流程中提交IO请求的时间，preloadWaitTime表示预读流程中等待IO请求的时间，preloadWaitCount表示预读流程中等待IO请求的次数。

OBS I/O ： 表示OBS IO请求的详细信息。（存算分离V3表或外表支持）

count表示OBS IO请求的总数量，averageRTT表示OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs，averageLatency表示OBS IO请求的平均延迟，单位为μs，latencyGt1s表示OBS IO请求延迟超过1s的数量，latencyGt10s表示OBS IO请求延迟超过10s的数量，retryCount表示OBS IO请求重试的总次数，rateLimitCount表示OBS IO请求被流控的总次数。

7. User Define Profiling

```
                  User Define Profiling
------------------------------------------------------------
Plan Node id: 1  Track name: coordinator get datanode connection
      cn_5001 (time=9.306 total_calls=1 loops=1)
Plan Node id: 1  Track name: coordinator begin transaction
      cn_5001 (time=0.002 total_calls=1 loops=1)
Plan Node id: 1  Track name: coordinator send command
      cn_5001 (time=0.113 total_calls=3 loops=1)
Plan Node id: 1  Track name: coordinator get the first tuple
      cn_5001 (time=0.091 total_calls=12 loops=1)
```

 版权所有 © 华为云计算技术有限公司

自定义信息，这一部分显示的是CN和DN、DN和DN建连的时间，以及存储层的一些执行信息。

8. Query Summary

```
                    ====== Query Summary =====
---------------------------------------------------------------------------
Datanode executor start time [dn_6005_6006, dn_6001_6002]: [0.360 ms,0.483 ms]
Datanode executor run time [dn_6001_6002, dn_6003_6004]: [0.008 ms,0.009 ms]
Datanode executor end time [dn_6003_6004, dn_6005_6006]: [0.036 ms,0.066 ms]
Remote query poll time: 2.649 ms, Deserialze time: 0.000 ms
System available mem: 1761280KB
Query Max mem: 1761280KB
Query estimated mem: 3328KB
Enqueue time: 0.030 ms
Coordinator executor start time: 0.083 ms
Coordinator executor run time: 13.044 ms
Coordinator executor end time: 0.034 ms
Parser runtime: 0.060 ms
Planner runtime: 0.539 ms
Query Id: 218706056932222840
Unique SQL Id: 2641724793
Total runtime: 13.906 ms
```

这一部分主要打印总的执行时间和网络流量，包括了各个DN上初始化和结束阶段的最大最小执行时间、CN上的初始化、执行、结束阶段的时间，以及当前语句执行时系统可用内存、语句估算内存等信息。

- DataNode executor start time：DN执行器开始时间，[min_node_name, max_node_name] : [min_time, max_time]。
- DataNode executor run time：DN执行器运行时间，[min_node_name, max_node_name] : [min_time, max_time]。
- DataNode executor end time：DN执行器结束时间，[min_node_name, max_node_name] : [min_time, max_time]。
- Remote query poll time：接收结果时用于poll等待的时间。
- System available mem：系统可用内存。
- Query Max mem：查询最大内存。
- Enqueue time：入队时间。
- Coordinator executor start time：CN执行器开始时间。
- Coordinator executor run time：CN执行器运行时间。
- Coordinator executor end time：CN执行器结束时间。
- Parser runtime：解析器运行时间。
- Planner runtime：优化器执行时间。
- 网络流量，stream算子发送的数据量。
- Query Id：查询ID。
- Unique SQL ID：约束SQL ID。
- Total runtime：总执行时间。

 版权所有 © 华为云计算技术有限公司

须知

- A-rows和E-rows的差异体现了优化器估算和实际执行的偏差度。一般情况下两者偏差越大，则可以认为优化器生成的计划的越不可信，人工干预调优的必要性越大。
- A-time中的两个值偏差越大，表明此算子的计算偏斜(在不同DN上执行时间差异)越大，人工干预调优的必要性越大。一般来说，两个相邻的算子，上层算子的执行时间包含下层算子的执行时间，但如果上层算子为stream算子，由于各线程不存在驱动关系，上层算子执行时间可能小于下层算子的执行时间，即不存在包含关系。
- Max Query Peak Memory经常用来估算SQL语句耗费内存，也被用来作为SQL语句调优时运行态内存参数设置的重要依据。一般会以EXPLAIN ANALYZE或EXPLAIN PERFORMANCE的输出作为进一步调优的输入。

16.4.3 执行计划算子

算子介绍

SQL执行计划中每一个步骤为一个数据库运算符，也叫作一个执行算子。DWS中算子是基本的数据处理单元，合理地组合算子、优化算子的顺序和执行方式，可以提升数据的处理效率。

DWS算子可分为：扫描算子、控制算子、物化算子、连接算子、其他算子等。

扫描算子

扫描算子用来扫描表中的数据，每次获取一条元组作为上层节点的输入， 存在于查询计划树的叶子节点，它不仅可以扫描表，还可以扫描函数的结果集、链表结构、子查询结果集。常见的扫描算子如下表所示：

表 16-9 扫描算子

算子	含义	场景
SeqScan	顺序扫描	最基本的扫描算子，用于扫描物理表（没有索引辅助的顺序扫描）。
IndexScan	索引扫描	选择条件涉及的属性上建立了索引。
IndexOnlyScan	直接从索引返回元组	索引列完全覆盖结果集列。
BitmapScan(BitmapIndexScan, BitmapHeapScan)	利用Bitmap获取元组	BitmapIndexScan利用属性上的索引进行扫描，返回结果为一个位图；BitmapHeapScan从BitmapIndexScan输出的位图中获取元组。
TidScan	通过元组tid获取元组	1. WHERE conditions(like CTID = tid or CTID IN (tid1, tid2, ···))； 2. UPDATE/DELETE ··· WHERE CURRENT OF cursor；

版权所有 © 华为云计算技术有限公司

算子	含义	场景
SubqueryScan	子查询扫描	以另一个查询计划树（子计划）为扫描对象进行元组的扫描。
FunctionScan	函数扫描	FROM function_name
ValuesScan	扫描values链表	对VALUES子句给出的元组集合进行扫描。
ForeignScan	外部表扫描	查询外部表。
CteScan	CTE表扫描	扫描SELECT查询中用WITH子句定义的子查询。

连接算子

连接算子对应了关系代数中的连接操作，以表 t1 join t2 为例，主要的集中连接类型如下：inner join、left join、right join、full join、semi join、 anti join，其实现方式包括Nestloop、HashJoin及MergeJoin。

表 16-10 连接算子

算子	含义	场景	实现特点
NestLoop	嵌套循环连接，暴力连接，对每一行都扫描内表。	Inner Join, Left Outer Join, Semi Join, Anti Join	适用于被连接的数据子集较小的查询。在嵌套循环中，外表驱动内表，外表返回的每一行都要在内表中检索找到它匹配的行，因此整个查询返回的结果集不能太大（不能大于10000），要把返回子集较小的表作为外表，而且在内表的连接字段上建议要有索引。
MergeJoin	归并连接（输入有序），内外表排序，定位首尾两端，一次性连接元组。等值连接。	Inner Join, Left Outer Join, Right Outer Join, Full Outer Join, Semi Join, Anti Join	也称作“融合连接”，是先将关联表的关联列各自做排序，然后从各自的排序表中抽取数据，到另一个排序表中做匹配。 因为Merge join需要做更多的排序，所以消耗的资源更多，因此通常情况下执行性能差于Hash Join。 如果源数据已经被排序过，在执行融合连接时，并不需要再排序，此时Merge Join的性能优于Hash Join。

 版权所有 © 华为云计算技术有限公司

算子	含义	场景	实现特点
(Sonic) HashJoin	哈希连接，内外表使用join列的hash值建立hash表，相同值的必在同一个hash桶。等值连接的连接两端必须为类型相同的等值连接，且支持hash散列。	Inner Join, Left Outer Join, Right Outer Join, Full Outer Join, Semi Join, Anti Join	哈希连接，适用于数据量大的表的连接方式。优化器使用两个表中较小的表，利用连接键在内存中建立hash表，然后扫描较大的表并探测散列，找到与散列匹配的行。Sonic和非Sonic的Hash Join的区别在于所使用hash表结构不同，不影响执行的结果集。

物化算子

物化算子是一类可缓存元组的节点。在执行过程中，很多扩展的物理操作符需要首先获取所有的元组才能进行操作（例如聚集函数操作、没有索引辅助的排序等），这是要用物化算子将元组缓存起来。

表 16-11 物化算子

算子	含义	场景
Material	物化	缓存子节点结果。
Sort	排序	ORDER BY子句，连接操作，分组操作，集合操作，配合Unique。
Group	分组操作	GROUP BY子句。
Agg	执行聚集函数。包含三种类型： • HashAggregate，哈希聚集算子，适用于大部分场景，以及数据无序情况下的聚集计算。 • GroupAggregate，排序聚集算子，基于排序或已排序输入做分组聚合。 • Aggregate，普通聚合算子，不分组，只简单聚集。	1. COUNT/SUM/AVG/MAX/MIN等聚集函数。 2. DISTINCT子句。 3. UNION去重。 4. GROUP BY子句。
WindowAgg	窗口函数	WINDOW子句。
Unique	去重（下层已排序）	1. DISTINCT子句。 2. UNION去重。
Hash	HashJoin辅助节点	构造Hash表，配合HashJoin。

 版权所有 © 华为云计算技术有限公司

算子	含义	场景
SetOp	处理集合操作	INTERSECT/INTERSECT ALL，EXCEPT/EXCEPT ALL
LockRows	处理行级锁	SELECT … FOR SHARE/ UPDATE

控制算子

控制算子是一类用于处理特殊情况的节点，用于实现特殊的执行流程。

表 16-12 控制算子

算子	含义	场景
Result	直接进行计算	1. 不包含表扫描。 2. INSERT语句中只有一个VALUES子句。
ModifyTable	INSERT/UPDATE/ DELETE上层节点	INSERT/UPDATE/DELETE
Append	追加	1. UNION(ALL)。 2. 继承表。
MergeAppend	追加（输入有序）	1. UNION(ALL)。 2. 继承表。
RecursiveUnion	处理WITH子句中递归定义的UNION子查询	WITH RECURSIVE … SELECT … 语句。
BitmapAnd	Bitmap逻辑与操作	多维索引扫描的BitmapScan。
BitmapOr	Bitmap逻辑或操作	多维索引扫描的BitmapScan。
Limit	处理LIMIT子句	OFFSET … LIMIT …

其他算子

其他算子包括Stream算子，以及RemoteQuery等算子。Stream算子主要有三种类型：Gather stream、Broadcast stream及Redistribute stream。

- Gather stream：每个源节点都将其数据发送给目标节点进行汇聚。
- Broadcast stream：由一个源节点将其数据发给N个目标节点进行运算。
- Redistribute stream：每个源节点将其数据根据连接条件计算Hash值，根据重新计算的Hash值进行分布，发给对应的目标节点。

 版权所有 © 华为云计算技术有限公司

表 16-13 其他算子

算子	含义	场景
Stream	多节点数据交换	执行分布式查询计划，节点间存在数据交换。
Partition Iterator	分区迭代器	分区表扫描，迭代扫描每个分区。
RowToVec	行转列	行列混合场景。
DfsScan / DfsIndexScan	HDFS表（索引）扫描	HDFS表扫描。

16.4.4 SQL 调优流程

对慢SQL语句进行分析，通常包括以下步骤：

操作步骤

步骤1 收集SQL中涉及到的所有表的统计信息。在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧会造成执行计划严重劣化，从而导致性能问题。从经验数据来看，10%左右性能问题是因为没有收集统计信息。具体请参见**更新统计信息**。

步骤2 **审视和修改表定义**。

步骤3 通常情况下，有些SQL语句可以通过查询重写转换成等价的，或特定场景下等价的语句。重写后的语句比原语句更简单，且可以简化某些执行步骤达到提升性能的目的。查询重写方法在各个数据库中基本是通用的。**SQL语句改写规则**介绍了几种常用的通过改写SQL进行调优的方法。

步骤4 通过查看执行计划来查找原因。如果SQL长时间运行未结束，通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及实际运行情况，以便更精准地定位问题原因。有关执行计划的详细介绍请参见**SQL执行计划**。

步骤5 针对EXPLAIN或EXPLAIN PERFORMANCE信息，定位SQL慢的具体原因以及改进措施，具体参见**SQL调优进阶**。

步骤6 用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息等多个手段来进行执行计划的调优，以提升查询的性能。详细请参见**使用Plan Hint进行调优**。

步骤7 为了保证数据库性能的持续优质，建议**例行维护表**和**例行重建索引**。

步骤8 （可选）DWS支持在资源富足的情况下，通过算子并行来提升性能。详细请参见**SMP并行执行**。

----结束

16.4.5 更新统计信息

在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧会造成执行计划严重劣化，从而导致性能问题。

版权所有 © 华为云计算技术有限公司

背景信息

ANALYZE语句可收集与数据库中表内容相关的统计信息，统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，以生成最有效的执行计划。

建议在执行了大批量插入/删除操作后，例行对表或全库执行ANALYZE语句更新统计信息。目前默认收集统计信息的采样比例是30000行（即：guc参数default_statistics_target默认设置为100），如果表的总行数超过一定行数（大于1600000），建议设置guc参数default_statistics_target为-2，即按2%收集样本估算统计信息。

对于在批处理脚本或者存储过程中生成的中间表，也需要在完成数据生成之后显式的调用ANALYZE。

对于表中多个列有相关性且查询中有同时基于这些列的条件或分组操作的情况，可尝试收集多列统计信息，以便查询优化器可以更准确地估算行数，并生成更有效的执行计划。

生成统计信息

- 更新单个表的统计信息。
  ```
  ANALYZE tablename;
  ```
- 更新全库的统计信息。
  ```
  ANALYZE;
  ```
- 多列统计信息相关操作。
  - 收集tablename表的column_1、column_2列的多列统计信息
    ```
    ANALYZE tablename ((column_1, column_2));
    ```
  - 添加tablename表的column_1、column_2列的多列统计信息声明
    ```
    ALTER TABLE tablename ADD STATISTICS ((column_1, column_2));
    ```
  - 收集单列统计信息，并收集已声明的多列统计信息
    ```
    ANALYZE tablename;
    ```
  - 删除tablename表的column_1、column_2列的多列统计信息或其声明
    ```
    ALTER TABLE tablename DELETE STATISTICS ((column_1, column_2));
    ```

须知

- 在使用ALTER TABLE tablename ADD STATISTICS语句添加了多列统计信息声明后，系统并不会立刻收集多列统计信息，而是在下次对该表或全库进行ANALYZE时，进行多列统计信息的收集。如果想直接收集多列统计信息，请使用ANALYZE命令进行收集。
- 使用EXPLAIN查看各SQL的执行计划时，如果发现某个表SEQ SCAN的输出中rows=10，rows=10是系统给的默认值，有可能该表没有进行ANALYZE，需要对该表执行ANALYZE。

提升统计信息质量

ANALYZE是按照随机采样算法从表上采样，根据样本计算表数据特征。采样数可以通过配置参数default_statistics_target进行控制，default_statistics_target取值范围为-100~10000，默认值为100。

版权所有 © 华为云计算技术有限公司

- 当default_statistics_target > 0时；采样的样本数为300*default_statistics_target，default_statistics_target取值越大，采样的样本也越大，样本占用的内存空间也越大，统计信息计算耗时也越长。
- 当default_statistics_target < 0时，采样的样本数为 (default_statistics_target)/100*表的总行数，default_statistics_target取值越小，采样的样本也越大。当default_statistics_target < 0时会把采样数据下盘，不存在样本占用的内存空间的问题，但是因为样本过大，计算耗时长的问题同样存在。

  default_statistics_target < 0时，实际采样数是 (default_statistics_target)/100*表的总行，所以又称之为百分比采样。

自动收集统计信息

当配置参数autoanalyze打开时，查询语句走到优化器发现表不存在统计信息或数据变化超过阈值时，会自动触发统计信息收集，以满足优化器的需求。

基于代价的优化器模型（CBO, cost base optimizer）中，统计信息决定了查询计划生成的好坏。因此，统计信息的及时有效很重要。

- 表级统计信息，存储在pg_class的relpages, reltuples中。
- 列级统计信息，存储在pg_statistics中，可以通过pg_stats视图查看。包括：NULL值比例，distinct值占比，高频值MCV，直方图histogram等。

收集条件：当数据量发生较大变化，默认是变化10%，认为数据特征已经有了变化，需要重新收集统计信息。

总体策略：开启动态采样保证统计信息及时性，开启轮询采样保证统计信息持久化，查询性能敏感（秒级响应）靠手动采样。

基本规则

表 16-14 统计信息收集相关基础功能

功能	介绍	特点	约束
手动采样	作业中修改大量数据后，手动执行ANALYZE	• normal模式，统计信息存系统表，全局共享。四级锁，同一张表不能并发。 • light模式，统计信息存内存，全局共享。一级锁，同一张表可以并发。 • force模式，在锁定统计信息情况下，也可以强制采样，行为与normal一致。 语法：ANALYZE tablename; ANALYZE (light|force) tablename;	-

 版权所有 © 华为云计算技术有限公司

功能	介绍	特点	约束
轮询采样	后台线程，根据阈值轮询维护统计信息	仅支持normal模式，统计信息存系统表，共享。四级锁，同一张表不能并发 相关GUC参数： ● autovacuum ● autovacuum_mode ● autovacuum_analyze_threshold ● autovacuum_analyze_scale_factor	异步轮询触发
动态采样	查询解析时，根据阈值，用几十秒的代价实时维护统计信息	● normal模式，统计信息存系统表，全局共享。四级锁，同一张表不能并发 ● light模式，统计信息存内存，全局共享。一级锁，同一张表可以并发 相关GUC参数： ● autoanalyze ● autoanalyze_mode	随查询实时触发 轻量化时依赖轮询采样进行持久化
强制采样	SQL中通过hint，强制每次查询都收集统计信息	用于数据特征敏感场景，确保查询时统计信息实时最新。 用法：select /*+ lightanalyze (t1 1) */ from t1; 1：强制采样，0：禁止采样	需要改SQL
分区统计信息	按分区收集，仅收增量，自动合并全局信息	用于超大分区表的场景，确保分区剪枝后的查询代价估算准确	多占存储，信息更准
多列统计信息	根据多个列组合收集统计信息	用于多列同时过滤场景，确保对多列组合的查询代价估算准确	手动识别，临时表方式
表达式统计信息	根据表达式函数对某列收集统计信息	用于批量的表达式过滤的场景，确保查询表达式的查询代价估算准确	手动识别
表达式索引信息	创建的表达式索引会自动收集统计信息	用于点查的表达式过滤的场景，确保查询表达式的查询代价估算准确	手动识别
冻结统计信息	将表级的统计信息冻结，防止发生变化	用于数据特征极稳定场景，禁止采样，防止查询计划跳变 用于数据特征极易变场景，强制采样，确保每次查询都采样 参数：表级属性analyze_mode	-

 版权所有 © 华为云计算技术有限公司

功能	介绍	特点	约束
修改统计信息	手动计算后，直接修改统计信息	继续低采样率，但可以手动计算后进行校准。用法： select approx_count_distinct(col_name) from table_name; alter table set (n_distinct=xxx)	-
分区信息拷贝	可将旧分区的统计信息拷贝到新分区	用于数据特征变化不大的分区表，减少统计信息收集的开销	-
统计信息推理	基于旧的统计信息自动推算更准确的统计信息	通过GUC参数enable_extrapolation_stats设置。	-
统计信息备份恢复	通过explain (stat on)将统计信息备份成SQL	用于场景复现或统计信息还原	导出SQL的形式

场景和策略

列举了常见的数据加工场景和对应的统计信息收集策略。

表 16-15 统计信息收集策略

场景	特点	策略
流式增量加工	数据流式增量变化，无合理ANALYZE时机	开启动态采样，查询按需自动收集统计信息，且全局共享

在线批量加工（数据湖）	数据加工与查询会并发，要求查询稳定	开启动态采样，或一个事务中完成数据加工和ANALYZE。
		begin;
		truncate table or partition;
		copy/merge/insert overwrite
		ANALYZE (light) tablename;
		end;

分区并行加工	不同分区并发加工数据	开启“动态采样”或“手动light采样”，同表可并发收集
宽表场景	百列以上的宽表	1. 启动动态采样的自动谓词管理。 2. 仅收集前N列的统计信息。 3. 根据查询中常用谓词，可列级设置是否参与采样。
大表场景	表的数据量大，变化难以达到阈值 统计信息易变	调低动态采样触发阈值。

版权所有 © 华为云计算技术有限公司

场景	特点	策略
特征敏感场景	数据特征易变，查询计划不稳定 需要强制收集	1. 调低动态采样触发阈值。 2. 在SQL中通过HINT方式强制light动态采样。 3. 清空并冻结统计信息，查询每次都会重新收集且不共享。
高并发场景	同一张表会高并发查询（10并发以上） 同时触发动态采样占用资源多	1. 关闭并发，其它查询不使用最新统计信息。 2. 其它查询等最新统计信息生成，再执行查询（开发中）。
流式性能敏感	流式加工，秒级查询或整体资源高水位	表级或SQL级禁止动态采样，依靠后台轮询采样。
批量性能敏感	批量加工，秒级查询或整体资源高水位	加工时手动收集统计信息。

资源消耗

表 16-16 资源消耗

类别	详细分类	说明
CPU	谓词列管理	自动管理谓词，仅对查询中使用的列收集统计信息。 手动屏蔽非谓词列，可手动设置哪些列不收集统计信息。
	超长列统计	可截断的数据类型，仅收取前1024字符计算统计信息。
IO	默认采集3W条样本	与表大小无关，与列数，分区数，小CU数有关。
内存	Buffer占用	最多只占用cstore buffer中的一个槽位。
	内存零拷贝	从buffer取出样本后，直接进行统计信息计算，无需组织成tuple再传递。
	内存自适应	可配置为内存不足时自动转为使用临时表采样。动态采样为防止查询触发建临时表，不会自适应。 通过GUC参数analyze_stats_mode设置。
	内存的大小	控制ANALYZE时最大使用内存，超过后或下盘或自动减少样本。 通过GUC参数maintenance_work_mem设置。

 版权所有 © 华为云计算技术有限公司

类别	详细分类	说明
锁	四级锁	normal模式，分布式申请四级锁。与增删改不冲突，与DDL，VACUUM，ANALYZE，REINDEX冲突。
	一级锁	light模式，仅本地一级锁。仅与DDL冲突。

准确性及可靠性

表 16-17 准确性及可靠性

准确性及可靠性	分类	说明
准确性	采样大小	可配置为按表大小自适应。由参数default_statistics_target控制。
	采样随机性	• analyze_sample_mode参数设置新支持优化蓄水池和range采样随机性更优。 • random_function_version参数增强了随机数计算函数的随机性。
	全局共享	统计信息可以跨会话和跨节点进行共享。
	修改计数广播	后台线程会轮询检全局查修改计数并进行广播。 作业线程也可直接广播修改计数。通过参数tuple_change_sync_threshold设置。 跨CN修改和查询也影响不大，修改计数会异步方式广播同步。
	调整CU采样比	CU填充率不高的场景，可调大CU采样比。通过列级参数cstore_cu_sample_ratio设置。
	distinct固定	随机采样后distinct值不稳定场景，不调高采样率情况下，可进行distinct值固定。通过列级参数n_distinct设置。
	统计信息推算	enable_extrapolation_stats参数可以控制估算失真时，基于旧的统计信息自动推算更准确的统计信息。
可靠性	CN故障	其它CN故障时，不影响动态采样，统计信息不同步，不影响当前CN查询的统计信息质量。
	CN恢复	其它CN恢复后，再次查询时会强制动态采样，并全局同步。
	DN故障	非当前逻辑集群的DN故障，不影响本逻辑集群的动态采样。

版权所有 © 华为云计算技术有限公司

运维监控

DWS通过在ANALYZE命令后面追加注释的方式，详细展示了ANALYZE运行模式和各种执行阶段，主要通过以下视图体现：

- 当前活跃会话视图pgxc_stat_activity的query字段。
- 当前线程等待视图pgxc_thread_wait_status的wait_status字段。

ANALYZE命令追加注释的内容格式为：--Action-RunMode-StatsMode-SyncMode

- Action取值及含义：
  {"begin", "finished", "lock FirstCN", "estimate rows", "statistics", "sample rows", "calc stats"};
  分别表示：开始，结束，在向FirstCN申锁，在一阶段估算行数信息，在二阶段真正执行ANALYZE，在采集样本，在计算统计信息。
- RunMode取值及含义：
  {"manual", "backend", "normal runtime", "light runtime", "light runtime inxact", "light estimate rows", "light manual"};
  分别表示：手动模式，后台轮询模式，normal动态采样，light动态采样，事务内的light动态采样，light仅估算函数，手动light模式。
- StatsMode取值及含义：
  {"dynamic", "memory", "smptbl"};
  分别表示：自适应选择内存或临时表放样本，仅使用内存放样本，仅使用临时表放样本。
- SyncMode取值及含义：
  {"sync", "nosync"};
  分别表示：向所有CN同步统计信息，不同步统计信息。

示例：

```
SELECT coorname,datid,datname,pid,usename,application_name,query_id,query
FROM pgxc_stat_activity WHERE query like '%analyze%' and query not like '%application_name%';
    coorname | datid | datname | pid | usename | application_name | query_id | query
--------------+-------+----------+-----------------+-----------+------------------+-------------------
+-----------------------------------------
coordinator1 | 15676 | postgres | 139919333779200 | test | gsql | 73183493944770822 | analyze t_1;
coordinator2 | 15676 | postgres | 140217336461056 | test | coordinator1 | 73183493944770822 | analyze
public.t_1;--push stats-manual-memory-sync
coordinator3 | 15676 | postgres | 139944245847808 | test | coordinator1 | 73183493944770822 | analyze
public.t_1;--push stats-manual-memory-sync
(3 rows)
```

统计信息查看

- 查看动态采样的内存统计信息。
  - 查内存中表级统计信息：
    SELECT * FROM pv_runtime_relstats;
  - 查内存中列级统计信息：
    SELECT * FROM pv_runtime_attstats;
- 查看系统表中统计信息。
  - 查系统表中表级统计信息：
    select relname, relpages, reltuples from pg_class;
  - 查系统表中列级统计信息：
    SELECT * FROM pg_stats;
- 查看统计信息最近收集时间。
  动态采样把统计信息放内存中，不更新该系统表的时间。

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pg_object;
```

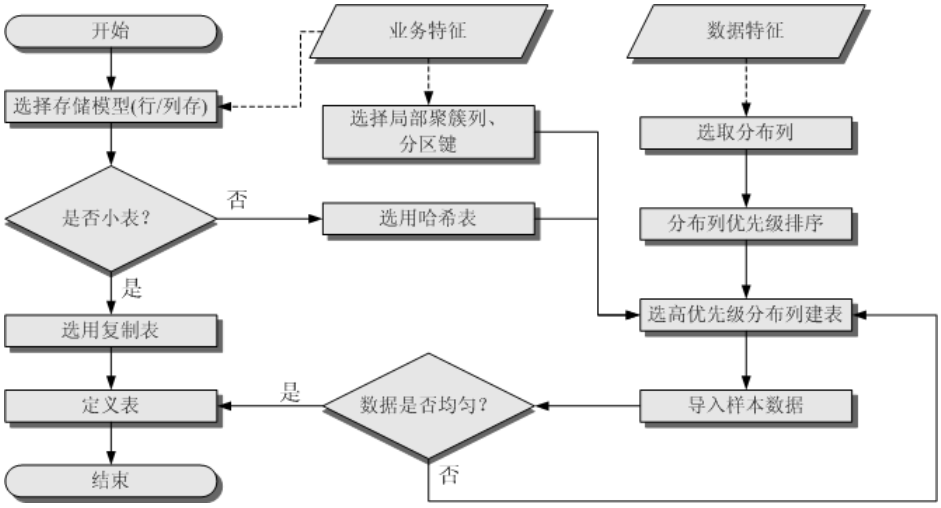
16.4.6 审视和修改表定义

在分布式框架下，数据分布在各个DN上。一个或者几个DN的数据存在一块物理存储设备上，较好的表定义需要满足以下要求：

1. **表数据均匀分布在各个DN上**，以防止单个DN对应的存储设备空间不足造成集群有效容量下降。选择合适分布列，避免数据分布倾斜可以实现该点。
2. **表Scan压力均匀分散在各个DN上**，以避免单DN的Scan压力过大，形成Scan的单节点瓶颈。分布列不选择基表上等值filter中的列可以实现该点。
3. **减少扫描数据量**。通过分区的剪枝机制可以实现该点。
4. **尽量减少随机IO**。通过聚簇/局部聚簇可以实现该点。
5. **尽量避免数据shuffle**，减小网络压力。通过选择join-condition或者group by列为分布列可以最大程度的实现这点。

从上述描述来看表定义中最重要的一点是分布列的选择。创建表定义一般遵循图16-5所示流程。表定义在数据库设计阶段创建，在SQL调优过程中进行审视和修改。

图 16-5 表定义流程



16.4.7 SQL 调优进阶

16.4.7.1 SQL 自诊断

用户在执行INSERT/UPDATE/DELETE/SELECT/MERGE INTO或者CREATE TABLE AS语句时，可能会遇到性能问题。产品内置集成了性能自动诊断功能，并把相关的诊断信息保存到**实时TopSQL**中，当配置参数**enable_resource_track**为on的时候，这些诊断信息会转存到**历史TopSQL**中。通过查询**GS_WLM_SESSION_STATISTICS**，**GS_WLM_SESSION_HISTORY**，**GS_WLM_SESSION_INFO**视图的warning字段可以获得对应的性能诊断信息，为性能调优提供参考。

- SQL自诊断的告警类型与**resource_track_level**的设置相关。

  当“resource_track_level”设置为query时，可以诊断多列/单列统计信息未收集、分区未剪枝告警和SQL不下推的告警等非执行态的诊断信息；
  “resource_track_level”设置为perf或operator时，可以诊断所有的告警场景。

版权所有 © 华为云计算技术有限公司

- SQL自诊断的诊断范围与**resource_track_cost**的设置相关。

  当SQL的代价大于“resource_track_cost”时，SQL才会被诊断。SQL的代价可以通过explain来确认。

- 执行EXPLAIN PERFORMANCE或者EXPLAIN VERBOSE的时候，除缺乏多列统计信息之外的SQL自诊断信息也会输出，具体请参考**SQL执行计划**。

SQL 执行性能相关告警场景

目前支持对以下10种导致性能问题的场景上报告警。

1. 多列/单列统计信息未收集。

   如果存在单列或者多列统计信息未收集，则上报相关告警。对于这种告警，建议的优化方案是对相关表进行ANALYZE，可参考**更新统计信息**和**统计信息调优**。

   需要特别注意的是，如果查询语句中的OBS外表和HDFS外表未收集统计信息，也会上报统计信息未收集的告警，因为OBS外表和HDFS外表的ANALYZE的性能比较差，一般不建议对其进行ANALYZE，但是建议使用ALTER FOREIGN TABLE的语法修改外表的totalrows属性，从而修正外表的行数估算。

   告警信息示例：

   整表的统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t1
   ```

   单列统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t2(c1)
   ```

   多列统计信息未收集:

   ```
   Statistic Not Collect
         schema_test.t3((c1,c2))
   ```

   单列和多列统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t4(c1)
         schema_test.t5((c1,c2))
   ```

2. 分区不剪枝。

   分区表查询时，常会期望通过分区键上的约束条件进行分区剪枝，从而提升分区表查询性能，但有时候会因为约束条件书写不当，导致分区表没有剪枝，出现查询性能问题，具体请参见**案例：改写SQL排除剪枝干扰**。

3. SQL不下推。

   对于不下推的SQL，尽可能详细上报导致不下推的原因。调优方法可参考案例**语句下推调优**。

   导致不下推的因素主要有以下两点：

   - 函数导致的不下推

     诊断信息中会给出具体的函数名称。函数下推是由函数的shippable属性决定，具体可参见《数据仓库服务SQL语法参考》中“CREATE FUNCTION”语法。

   - 语法导致的不下推

     诊断信息中会提示导致不下推的具体语法，例如：含有With Recursive，Distinct On，row表达式，返回值为record类型的，会告警相应语法不支持下推等等。

   告警信息示例：

 版权所有 © 华为云计算技术有限公司

```
SQL is not plan-shipping
        "enable_stream_operator" is off

SQL is not plan-shipping
        "Distinct On" can not be shipped

SQL is not plan-shipping
        "v_test_unshipping_log" is VIEW that will be treated as Record type can't be shipped
```

4. 不支持向量化计划

   对于不能走向量化计划的SQL，尽可能详细地上报不支持向量化的原因。

   常见的不支持向量化的因素主要有：

   - 目标列含有返回类型是集合类型的函数。
   - 目标列或查询条件、Stream算子的分布键、Limit和Offset子句含有不可向量化的表达式（地理空间类型、数组表达式、Row表达式、XML表达式、参数或返回值有refcursor类型的函数等）。
   - Group By子句含有数组等值判断表达式。
   - GC_FDW和LOG_FDW不支持向量化。
   - 计划含有Cte Scan、Recursive Union、Merge Append、Lock Rows等算子。

   告警信息示例：

   ```
   SQL is un-vectorized
           Function regexp_split_to_table that returns set is un-vectorized

   SQL is un-vectorized
           Array expression is un-vectorized

   SQL is un-vectorized
           Function array_agg is un-vectorized

   SQL is un-vectorized
           RecursiveUnion is un-vectorized
   ```

5. HashJoin中大表做内表。

   如果在表连接过程中使用了Hashjoin（可通过**GS_WLM_SESSION_HISTORY**的“query_plan”字段查看），且连接的内表行数是外表行数的10倍或以上；同时内表在每个DN上的平均行数大于10万行，且发生了下盘，则上报相关告警。针对这种场景，需要调整HashJoin内外表顺序，具体调优方法参考**Join顺序的Hint**。

   告警信息示例：

   ```
   Execute diagnostic information
           PlanNode[7] Large Table is INNER in HashJoin “Vector Hash Aggregate”
   ```

   其中，7为“query_plan”字段文本中编号为7的算子。

6. 大表等值连接使用Nestloop

   如果在表连接过程中使用了nestloop（可通过**GS_WLM_SESSION_HISTORY**的query_plan字段查看），并且两个表中较大表的行数平均每个DN上的行数大于10万行、表的连接中存在等值连接，则上报相关告警。针对这种场景，需要调整表关联方式，禁止当前内外表之间使用NestLoop的关联方式，具体调优方法参考**Join方式的Hint**。

   告警信息示例：

   ```
   Execute diagnostic information
           PlanNode[5] Large Table with Equal-Condition use Nestloop"Nested Loop"
   ```

7. 大表Broadcast

   如果在Broadcast算子中，平均每DN的行数大于10万行，则告警大表Broadcast。针对这种场景，需要禁止Broadcast下层算子做Broadcast动作，具体调优方法参考**Stream方式的Hint**。

 版权所有 © 华为云计算技术有限公司

告警信息示例：

```
Execute diagnostic information
    PlanNode[5] Large Table in Broadcast "Streaming(type: BROADCAST dop: 1/2)"
```

8. 数据倾斜

   某表在各DN上的分布，存在某DN上的行数是另一DN上行数的10倍或以上，且有DN中的行数大于10万行，则上报相关告警。该告警一般分为存储层倾斜和计算层倾斜，具体调优方法参考**数据倾斜调优**。

   告警信息示例：

   ```
   Execute diagnostic information
       PlanNode[6] DataSkew:"Seq Scan", min_dn_tuples:0, max_dn_tuples:524288
   ```

9. 索引不合理

   在基表扫描时，满足下述条件则上报相关告警：

   - 对于行存表：
     - 使用索引扫描，输出行数/扫描行数>1/1000且输出行数>1万行。
     - 使用顺序扫描，输出行数/扫描行数<1/1000且输出行数<=1万行、扫描行数>1万行。
   - 对于列存表：
     - 使用索引扫描，输出行数/扫描行数>1/10000且输出行数>100。
     - 使用顺序扫描，输出行数/扫描行数<1/10000且输出行数<=100、扫描行数>1万行。

   调优方法可参考**算子级调优**，也可参考案例**案例：建立合适的索引**和**案例：使用partial cluster key**。

   告警信息示例：

   ```
   Execute diagnostic information
       PlanNode[4] Indexscan is not properly used:"Index Only Scan", output:524288, filtered:0, rate:1.00000
       PlanNode[5] Indexscan is ought to be used:"Seq Scan", output:1, filtered:524288, rate:0.00000
   ```

   需要注意的是，这个诊断结果只是针对当前SQL的建议，是否创建索引要结合整体业务综合分析，对于高频的过滤条件才建议创建索引。

10. 估算不准

对于平均每DN行数如果优化器的估算行数和实际行数中的较大值大于10万行，并且估算行数和实际行数中较大值是较小值的10倍或以上，则上报相关告警。针对这种场景，可以参照**行数的Hint**修正行数估算，让优化器在正确的行数基础上重新规划执行计划。

告警信息示例：

```
Execute diagnostic information
    PlanNode[5] Inaccurate Estimation-Rows: "Hash Join" A-Rows:0, E-Rows:52488
```

规格约束

1. 告警字符串长度上限为2048。如果告警信息超过这个长度（例如存在大量未收集统计信息的超长表名，列名等信息）则不告警，只上报warning：
   ```
   WARNING, "Planner issue report is truncated, the rest of planner issues will be skipped"
   ```
2. 如果query存在limit节点（即查询语句中包含limit），则不会上报limit节点以下的Operator级别的告警。

版权所有 © 华为云计算技术有限公司

3. 对于“数据倾斜”和“估算不准”两种类型告警，在某一个plan树结构下，只上报下层节点的告警，上层节点不再重复告警。这主要是因为这两种类型的告警可能是因为底层触发上层的。例如，如果在scan节点已经存在数据倾斜，那么在上层的hashagg等其他算子很可能也出现数据倾斜。

16.4.7.2 语句下推调优

语句下推介绍

目前，DWS优化器在分布式框架下制定语句的执行策略时，有三种执行计划方式：生成下推语句计划、生成分布式执行计划、生成发送语句的分布式执行计划。

- 下推语句计划：指直接将查询语句从CN发送到DN进行执行，然后将执行结果返回给CN。
- 分布式执行计划：指CN对查询语句进行编译和优化，生成计划树，再将计划树发送给DN进行执行，并在执行完毕后返回结果到CN。
- 发送语句的分布式执行计划：上述两种方式都不可行时，将可下推的查询部分组成查询语句（多为基表扫描语句）下推到DN进行执行，获取中间结果到CN，然后在CN执行剩下的部分。

在“发送语句的分布式执行计划”策略中，要将大量中间结果从DN发送到CN，并且要在CN运行不能下推的部分语句，会导致CN成为性能瓶颈（带宽、存储、计算等）。在进行性能调优的时候，应尽量避免只能选择该策略的查询语句。

执行语句不能下推是因为语句中含有**不支持下推的函数**或者**不支持下推的语法**。一般都可以通过等价改写规避执行计划不能下推的问题。

查看执行计划是否下推

执行计划是否下推可以依靠如下方法快速判断：

步骤1 将GUC参数“**enable_fast_query_shipping**”设置为off，使查询优化器使用分布式框架策略。

```
SET enable_fast_query_shipping = off;
```

步骤2 查看执行计划。

如果执行计划中有Data Node Scan节点，那么此执行计划为不可下推的执行计划；如果执行计划中有Streaming节点，那么计划是可以下推的。

例如如下业务SQL：

```
select
count(ss.ss_sold_date_sk order by ss.ss_sold_date_sk)c1
from store_sales ss, store_returns sr
where
sr.sr_customer_sk = ss.ss_customer_sk;
```

执行计划如下，可以看出此SQL语句不能下推。

```
                          QUERY PLAN
------------------------------------------------------------------------
Aggregate
->  Hash Join
Hash Cond: (ss.ss_customer_sk = sr.sr_customer_sk)
->  Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
Node/s: All datanodes
->  Hash
->  Data Node Scan on store_returns "_REMOTE_TABLE_QUERY_"
```

 版权所有 © 华为云计算技术有限公司

```
Node/s: All datanodes
(8 rows)
```

----结束

不支持下推的语法

以如下三个表定义说明不支持下推的SQL语法。

```
CREATE TABLE CUSTOMER1
(
    C_CUSTKEY     BIGINT NOT NULL
  , C_NAME        VARCHAR(25) NOT NULL
  , C_ADDRESS     VARCHAR(40) NOT NULL
  , C_NATIONKEY   INT NOT NULL
  , C_PHONE       CHAR(15) NOT NULL
  , C_ACCTBAL     DECIMAL(15,2)   NOT NULL
  , C_MKTSEGMENT  CHAR(10) NOT NULL
  , C_COMMENT     VARCHAR(117) NOT NULL
)
DISTRIBUTE BY hash(C_CUSTKEY);
CREATE TABLE test_stream(a int,b float); --float不支持重分布
CREATE TABLE sal_emp ( c1 integer[] ) DISTRIBUTE BY replication;
```

- 不支持returning语句下推。

  ```
  explain update customer1 set C_NAME = 'a' returning c_name;
                                QUERY PLAN
  ------------------------------------------------------------------
   Update on customer1  (cost=0.00..0.00 rows=30 width=187)
     Node/s: All datanodes
     Node expr: c_custkey
     ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=187)
           Node/s: All datanodes
  (5 rows)
  ```

- count(distinct expr)中的字段不支持重分布，则不支持下推。

  ```
  explain verbose select count(distinct b) from test_stream;
                                       QUERY PLAN
  ------------------------------------------------------------------------ Aggregate  (cost=2.50..2.51 rows=1 width=8)
     Output: count(DISTINCT test_stream.b)
     ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=8)
           Output: test_stream.b
           Node/s: All datanodes
           Remote query: SELECT b FROM ONLY public.test_stream WHERE true
  (6 rows)
  ```

- 不支持distinct on用法下推。

  ```
  explain verbose select distinct on (c_custkey) c_custkey from customer1 order by c_custkey;
                                       QUERY PLAN
  ------------------------------------------------------------------------ Unique  (cost=49.83..54.83 rows=30 width=8)
     Output: customer1.c_custkey
     ->  Sort  (cost=49.83..52.33 rows=30 width=8)
           Output: customer1.c_custkey
           Sort Key: customer1.c_custkey
           ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
  width=8)
                 Output: customer1.c_custkey
                 Node/s: All datanodes
                 Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
  (9 rows)
  ```

- Fulljoin的join列如果不支持重分布，则不支持下推。

  ```
  explain select * from test_stream t1 full join test_stream t2 on t1.a=t2.b;
                                       QUERY PLAN
  ------------------------------------------------------------------------ Hash Full Join  (cost=0.38..0.82 rows=30
  width=24)
     Hash Cond: ((t1.a)::double precision = t2.b)
     ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=12)
           Node/s: All datanodes
  ```


版权所有 © 华为云计算技术有限公司

```
  ->  Hash  (cost=0.00..0.00 rows=30 width=12)
        ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
width=12)
              Node/s: All datanodes
(7 rows)
```

- 不支持数组表达式下推。

```
explain verbose select array[c_custkey,1] from customer1 order by c_custkey;

                        QUERY PLAN
------------------------------------------------------------------------ Sort  (cost=49.83..52.33 rows=30 width=8)
   Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
   Sort Key: customer1.c_custkey
   ->  Data Node Scan on "__REMOTE_SORT_QUERY__"  (cost=0.00..0.00 rows=30 width=8)
         Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
         Node/s: All datanodes
         Remote query: SELECT ARRAY[c_custkey, 1::bigint], c_custkey FROM ONLY public.customer1
WHERE true ORDER BY 2
(7 rows)
```

- Subplan在多个线程间共享不支持下推。

```
postgres=# explain verbose  select c_custkey in (select c_custkey from customer1) b from customer1;
                        QUERY PLAN
------------------------------------------------------------------------
Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=2.50..5.00 rows=1000 width=8)
   Output: (hashed SubPlan 1)
   Node/s: All datanodes
   Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
   SubPlan 1
     ->  Data Node Scan on customer "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=1000
width=8)
           Output: public.customer.c_custkey
           Node/s: All datanodes
           Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
(9 rows)
```

- With Recursive当前版本不支持下推的场景和原因如下：

序号	场景	不下推原因
1	包含外表、HDFS表的查询场景	LOG: SQL can't be shipped, reason: RecursiveUnion contains HDFS Table or ForeignScan is not shippable（LOG为CN日志中打印的不下推原因，下同） 外表、HDFS表，当前版本暂不支持下推。
2	多nodegroup场景	LOG: SQL can't be shipped, reason: With-Recursive under multi-nodegroup scenario is not shippable 基表存储nodegroup不相同，或者计算nodegroup与基表不相同，当前版本暂不支持下推。

版权所有 © 华为云计算技术有限公司

序号	场景	不下推原因
3	WITH recursive t_result AS (SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm > 10 UNION SELECT t2.dm,t2.sj_dm,t2.name||' > '|| t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive does not contain "ALL" to bind recursive & none-recursive branches UNION不带ALL，需要去重。
4	WITH RECURSIVE x(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 5), y(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 10) SELECT y.*, x.* FROM y LEFT JOIN x USING (id) ORDER BY 1;	LOG: SQL can't be shipped, reason: With-Recursive contains system table is not shippable 基表中有系统表。
5	WITH RECURSIVE t(n) AS (VALUES (1) UNION ALL SELECT n+1 FROM t WHERE n < 100) SELECT sum(n) FROM t;	LOG: SQL can't be shipped, reason: With-Recursive contains only values rte is not shippable 基表扫描只有VALUES子句，仅在CN上即可完成执行。
6	select a.ID,a.Name, (with recursive cte as (select ID, PID, NAME from b where b.ID = 1 union all select parent.ID,parent.PID,parent.NAME from cte as child join b as parent on child.pid=parent.id where child.ID = a.ID) select NAME from cte limit 1) cName from (select id, name, count(*) as cnt from a group by id,name) a order by 1,2;	LOG: SQL can't be shipped, reason: With-Recursive recursive term correlated only is not shippable 相关子查询的关联条件仅在递归部分，非递归部分无关联条件。

版权所有 © 华为云计算技术有限公司

序号	场景	不下推原因
7	WITH recursive t_result AS (select * from(SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm < 10 order by dm limit 6 offset 2) UNION all SELECT t2.dm,t2.sj_dm,t2.name||' > '||t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive contains conflict distribution in none-recursive(Replicate) recursive(Hash) 非递归部分带limit为Replicate计划，递归部分为Hash计划，计划存在冲突。
8	with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from (with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from rec_tb4 h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID) h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID,1,2,3;	LOG: SQL can't be shipped, reason: Recursive CTE references recursive CTE "cte" 多层Recursive嵌套，即recursive的递归部分又嵌套另一个recursive查询。

不支持下推的函数

首先介绍函数的易变性。在DWS中共分三种形态：

- **IMMUTABLE**

  表示该函数在给出同样的参数值时总是返回同样的结果。

- **STABLE**

  表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。

- **VOLATILE**

  表示该函数值可以在一次表扫描内改变，因此不会做任何优化。

函数易变性可以查询pg_proc的provolatile字段获得，i代表IMMUTABLE，s代表STABLE，v代表VOLATILE。另外，在pg_proc中的proshippable字段，取值范围为t/f/NULL，这个字段与provolatile字段一起用于描述函数是否下推。

- 如果函数的provolatile属性为i，则无论proshippable的值是否为t，则函数始终可以下推。
- 如果函数的provolatile属性为s或v，则仅当proshippable的值为t时，函数可以下推。

 版权所有 © 华为云计算技术有限公司

- random如果出现CTE中，也不下推。因为这种场景下下推可能出现结果错误。

对于用户自定义函数，可以在创建函数的时候指定provolatile和proshippable属性的值，详细请参考CREATE FUNCTION。

对于函数不能下推的场景：

- 如果是系统函数，建议根据业务等价替换这个函数。
- 如果是自定义函数，建议分析客户业务场景，看函数的provolatile和proshippable属性定义是否正确。

实例分析：自定义函数

对于自定义函数，如果对于确定的输入，有确定的输出，则应将函数定义为immutable类型。

利用TPCDS的销售信息举例，需要写一个函数来获取商品的打折情况，即定义一个计算折扣的函数：

```
CREATE FUNCTION func_percent_2 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
VOLATILE;
```

执行下列语句：

```
SELECT func_percent_2(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
   Output: func_percent_2(store_sales.ss_sales_price, store_sales.ss_list_price)
   Remote query: SELECT ss_sales_price, ss_list_price FROM ONLY store_sales WHERE true
(3 rows)
```

可见，func_percent_2并没有被下推，而是将ss_sales_price和ss_list_price收到CN上，再进行计算，消耗大量CN的资源，而且计算缓慢。

由于该自定义函数对确定的输入有确定的输出，如果将该自定义函数改为：

```
CREATE FUNCTION func_percent_1 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
IMMUTABLE;
```

执行语句：

```
SELECT func_percent_1(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan on "__REMOTE_FQS_QUERY__"  (cost=0.00..0.00 rows=0 width=0)
   Output: (func_percent_1(store_sales.ss_sales_price, store_sales.ss_list_price))
   Node/s: All datanodes
   Remote query: SELECT public.func_percent_1(ss_sales_price, ss_list_price) AS func_percent_1 FROM public.store_sales
(4 rows)
```

可见函数func_percent_1被下推到DN执行，提升了执行效率（TPCDS 1000X，3CN18DN，查询效率提升100倍以上）。

其他典型不下推案例集

请参考**案例：典型不下推语句整改案例集**。

 版权所有 © 华为云计算技术有限公司

16.4.7.3 子查询调优

子查询背景介绍

应用程序通过SQL语句来操作数据库时会使用大量的子查询，这种写法比直接对两个表做连接操作在结构上和思路上更清晰，尤其是在一些比较复杂的查询语句中，子查询有更完整、更独立的语义，会使SQL对业务逻辑的表达更清晰更容易理解，因此得到了广泛的应用。

DWS根据子查询在SQL语句中的位置把子查询分成了子查询、子链接两种形式。

- 子查询SubQuery：对应于查询解析树中的范围表RangeTblEntry，更通俗一些指的是出现在FROM语句后面的独立的SELECT语句。
- 子链接SubLink：对应于查询解析树中的表达式，更通俗一些指的是出现在where/on子句、targetlist里面的语句。

  综上，对于查询解析树而言，SubQuery的本质是范围表，而SubLink的本质是表达式。针对SubLink场景而言，由于SubLink可以出现在约束条件、表达式中，按照DWS对sublink的实现，sublink可以分为以下几类：
  - exist_sublink：对应EXIST、NOT EXIST语句
  - any_sublink：对应op Any(select…)语句，其中OP可以是IN,<,>,=操作符
  - all_sublink：对应op ALL(select…)语句，其中OP可以是IN,<,>,=操作符
  - rowcompare_sublink：对应record op (select …)语句
  - expr_sublink：对应(SELECT with single targetlist item ...)语句
  - array_sublink：对应ARRAY(select…)语句
  - cte_sublink：对应with query(…)语句

须知

在逻辑集群模式下，当执行的查询语句中存在子查询与父查询执行集群不一致，且子查询中存在与父查询关联的过滤条件时，由于相关条件无法跨逻辑集群下推，导致计划产生额外的算子开销。该场景下性能可能会劣于单逻辑集群场景。

其中OLAP、HTAP场景中常用的sublink为exist_sublink、any_sublink，在DWS的优化引擎中对其应用场景做了优化（子链接提升），由于SQL语句中子查询的使用的灵活性，会带来SQL子查询过于复杂而造成的性能问题。子查询从大类上来看，分为非相关子查询和相关子查询：

- **非相关子查询None-Correlated SubQuery**

  子查询的执行不依赖于外层父查询的任何属性值。这样子查询具有独立性，可独自求解，形成一个子查询计划先于外层的查询求解。

  例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c2 IN (2,3,4)
);
                        QUERY PLAN
------------------------------------------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
```


版权所有 © 华为云计算技术有限公司

```
  ->  Hash Right Semi Join
        Hash Cond: (t2.c2 = t1.c1)
        ->  Streaming(type: REDISTRIBUTE)
              Spawn on: All datanodes
              ->  Seq Scan on t2
                    Filter: (c2 = ANY ('{2,3,4}'::integer[]))
        ->  Hash
              ->  Seq Scan on t1
(10 rows)
```

- **相关子查询Correlated-SubQuery**

  子查询的执行依赖于外层父查询的一些属性值（如下列示例t2.c1 = t1.c1条件中的t1.c1）作为内层查询的一个AND-ed条件。这样的子查询不具备独立性，需要和外层查询按分组进行求解。

  例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = t1.c1 AND t2.c2 in (2,3,4)
);
                                QUERY PLAN
-------------------------------------------------------------------------
Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t2.c1 = t1.c1)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                             Spawn on: All datanodes
                             ->  Seq Scan on t2
                                   Filter: (c2 = ANY ('{2,3,4}'::integer[]))
(12 rows)
```

DWS 对 SubLink 的优化

针对SubLink的优化策略主要是让内层的子查询提升（pullup），能够和外表直接做关联查询，从而避免生成SubPlan+Broadcast内表的执行计划。判断子查询是否存在性能风险，可以通过explain查询语句查看Sublink的部分是否被转换成SubPlan+Broadcast的执行计划。

例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = t1.c1
);
```

```
QUERY PLAN
-------------------------------------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Seq Scan on t1
        Filter: (SubPlan 1)
        SubPlan 1
          ->  Result
                Filter: (t2.c1 = t1.c1)
                ->  Materialize
                      ->  Streaming(type: BROADCAST)
                            Spawn on: All datanodes
                            ->  Seq Scan on t2
(11 rows)
```

- **目前DWS支持的Sublink-Release场景**
  - IN-Sublink无相关条件
    - 不能包含上一层查询的表中的列（可以包含更高层查询表中的列）。

 版权所有 © 华为云计算技术有限公司

- 不能包含易变函数。

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = 1
);
```

```
QUERY PLAN
-----------------------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Nested Loop Semi Join
        Join Filter: (t1.c1 = t2.c2)
        ->  Seq Scan on t1
        ->  Materialize
              ->  Streaming(type: REDISTRIBUTE)
                    Spawn on: datanode1
                    ->  Seq Scan on t2
                          Filter: (c1 = 1)
(10 rows)
```

– Exist-Sublink包含相关条件

Where子句中必须包含上一层查询的表中的列，子查询的其它部分不能含有上层查询的表中的列。其它限制如下：

- 子查询必须有from子句。
- 子查询不能含有with子句。
- 子查询不能含有聚集函数。
- 子查询里不能包含集合操作、排序、limit、windowagg、having操作。
- 不能包含易变函数。

```
select t1.c1,t1.c2
from t1
where exists (
    select c2
    from t2
    where t2.c1 = t1.c1
);
```

```
QUERY PLAN
-----------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Hash Semi Join
        Hash Cond: (t1.c1 = t2.c1)
        ->  Seq Scan on t1
        ->  Hash
              ->  Seq Scan on t2
(7 rows)
```

– 包含聚集函数的等值相关子查询的提升

子查询的where条件中必须含有来自上一层的列，而且此列必须和子查询本层涉及表中的列做相等判断，且这些条件必须用and连接。其它地方不能包含上层的列。其它限制条件如下：

- 子查询中where条件包含的表达式（列名）必须是表中的列。
- 子查询的Select关键字后，必须有且仅有一个输出列，此输出列必须是聚集函数（如max），并且聚集函数的参数（t2.c2）不能是来自外层表（t1）中的列。聚集函数不能是count。

  下列示例可以提升：

  ```
  select * from t1 where c1 >(
      select max(t2.c1) from t2 where t2.c1=t1.c1
  );
  ```

  下列示例不能提升，因为子查询没有聚集函数：

  ```
  select * from t1 where c1 >(
      select  t2.c1 from t2 where t2.c1=t1.c1
  );
  ```

  下列示例不能提升，因为子查询有两个输出列：

  ```
  select * from t1 where (c1,c2) >(
      select  max(t2.c1),min(t2.c2) from t2 where t2.c1=t1.c1
  );
  ```

 版权所有 © 华为云计算技术有限公司

- 子查询必须是from子句。
- 子查询中不能有groupby、having、集合操作。
- 子查询只能是inner join。

  下列示例不能提升：

  ```
  select * from t1 where c1 >(
        select max(t2.c1) from t2 full join t3 on (t2.c2=t3.c2) where t2.c1=t1.c1
  );
  ```

- 子查询的targetlist中不能包含返回set的函数。
- 子查询的where条件中必须含有来自上一层的列，而且此列必须和子查询层涉及表中的列做相等判断，且这些条件必须用and连接。其它地方不能包含上层中的列。下列示例中的最内层子链接可以提升：

  ```
  select * from t3 where t3.c1=(
        select t1.c1
        from t1 where c1 >(
              select max(t2.c1) from t2 where t2.c1=t1.c1
  ));
  ```

  基于上面的示例，再加一个条件，则不能提升，因为最内侧子查询引用了上层中的列。示例如下：

  ```
  select * from t3 where t3.c1=(
        select t1.c1
        from t1 where c1 >(
              select max(t2.c1) from t2 where t2.c1=t1.c1 and t3.c1>t2.c2

  ));
  ```

– 提升OR子句中的SubLink

  当WHERE过滤条件中有OR连接的EXIST相关SubLink，

  例如：

  ```
  select a, c from t1
  where t1.a = (select avg(a) from t3 where t1.b = t3.b) or
  exists (select * from t4 where t1.c = t4.c);
  ```

  将OR-ed连接的EXIST相关子查询OR子句的提升过程：

  i. 提取where条件中，or子句中的opExpr。为：t1.a = (select avg(a) from t3 where t1.b = t3.b)

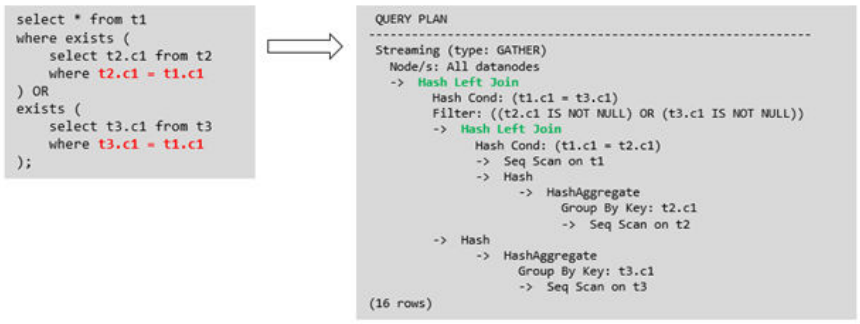
  ii. 这个op操作中包含subquery，判断是否可以提升，如果可以提升，重写subquery为：select avg(a), t3.b from t3 group by t3.b，生成not null条件t3.b is not null，并将这个opexpr用这个not null条件替换。此时SQL变为：

  ```
  select a, c
  from t1 left join (select avg(a) avg, t3.b from t3 group by t3.b)  as t3 on (t1.a = avg
  and t1.b = t3.b)
  where t3.b is not null or exists (select * from t4 where t1.c = t4.c);
  ```

  iii. 再次提取or子句中的exists sublink，exists (select * from t4 where t1.c = t4.c)，判断是否可以提升，如果可以提升，转换subquery为：select t4.c from t4 group by t4.c生成NotNull条件t4.c is not null提升查询，SQL变为：

  ```
  select a, c
  from t1 left join (select avg(a) avg, t3.b from t3 group by t3.b)  as t3 on (t1.a = avg and
  t1.b = t3.b)
  left join (select t4.c from t4 group by t4.c) where t3.b is not null or t4.c is not null;
  ```

 版权所有 © 华为云计算技术有限公司



- **目前DWS不支持的Sublink-Release场景**

  除了以上场景之外都不支持Sublink提升，因此关联子查询会被计划成SubPlan +Broadcast的执行计划，当inner表的数据量较大时则会产生性能风险。

  如果相关子查询中跟外层的两张表做join，那么无法提升该子查询，需要通过将父SQL创建成with子句，然后再跟子查询中的表做相关子查询。

  例如：

  ```
  select distinct t1.a, t2.a
  from t1 left join t2 on t1.a=t2.a and not exists (select a,b from test1 where test1.a=t1.a and test1.b=t2.a);
  ```

  改写为

  ```
  with temp as
  (
        select * from (select t1.a as a, t2.a as b from t1 left join t2 on t1.a=t2.a)

  )
  select distinct a,b
  from temp
  where not exists (select a,b from test1 where temp.a=test1.a and temp.b=test1.b);
  ```

  – 出现在targetlist里的相关子查询无法提升(不含count)

    例如：

    ```
    explain (costs off)
    select (select c2 from t2 where t1.c1 = t2.c1) ssq, t1.c2
    from t1
    where t1.c2 > 10;
    ```

    执行计划为：

    ```
    explain (costs off)
    select (select c2 from t2 where t1.c1 = t2.c1) ssq, t1.c2
    from t1
    where t1.c2 > 10;
                        QUERY PLAN
    ------------------------------------------------------------
     Streaming (type: GATHER)
       Node/s: All datanodes
       ->  Seq Scan on t1
             Filter: (c2 > 10)
             SubPlan 1
               ->  Result
                     Filter: (t1.c1 = t2.c1)
                     ->  Materialize
                           ->  Streaming(type: BROADCAST)
                                 Spawn on: All datanodes
                                 ->  Seq Scan on t2
    (11 rows)
    ```

    由于相关子查询出现在targetlist（查询返回列表）里，对于t1.c1=t2.c1不匹配的场景仍然需要输出值，因此使用left-outerjoin关联T1&T2确保t1.c1=t2.c1在不匹配时，子SSQ能够返回不匹配的补空值。

 版权所有 © 华为云计算技术有限公司

说明

SSQ和CSSQ的解释如下：

- SSQ：ScalarSubQuery一般指返回1行1列scalar值的sublink，简称SSQ。
- CSSQ：Correlated-ScalarSubQuery和SSQ相同不过是指包含相关条件的SSQ。

上述SQL语句可以改写为：

```
with ssq as
(
   select t2.c1, t2.c2 from t2
)
select ssq.c2, t1.c2
from t1 left join ssq on t1.c1 = ssq.c1
where t1.c2 > 10;
```

改写后的执行计划为：

```
                QUERY PLAN
-------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Hash Right Join
         Hash Cond: (t2.c1 = t1.c1)
         ->  Seq Scan on t2
         ->  Hash
               ->  Seq Scan on t1
                     Filter: (c2 > 10)
(8 rows)
```

可以看到出现在SSQ返回列表里的相关子查询SSQ，已经被提升成Right Join，从而避免当内表T2较大时出现SubPlan+Broadcast计划导致性能变差。

- 出现在targetlist里的相关子查询无法提升(带count)

  例如：

```
select (select count(*) from t2 where t2.c1=t1.c1) cnt, t1.c1, t3.c1
from t1,t3
where t1.c1=t3.c1 order by cnt, t1.c1;
```

  执行计划为：

```
                          QUERY PLAN
-------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Sort
         Sort Key: ((SubPlan 1)), t1.c1
         ->  Hash Join
               Hash Cond: (t1.c1 = t3.c1)
               ->  Seq Scan on t1
               ->  Hash
                     ->  Seq Scan on t3
               SubPlan 1
                 ->  Aggregate
                       ->  Result
                             Filter: (t2.c1 = t1.c1)
                             ->  Materialize
                                   ->  Streaming(type: BROADCAST)
                                         Spawn on: All datanodes
                                         ->  Seq Scan on t2
(17 rows)
```

  由于相关子查询出现在targetlist（查询返回列表）里，对于t1.c1=t2.c1不匹配的场景仍然需要输出值，因此使用left-outerjoin关联T1&T2确保t1.c1=t2.c1在不匹配时子SSQ能够返回不匹配的补空值，但是这里带了count语句及时在t1.c1=t2.t1不匹配时需要输出0，因此可以使用一个case-when NULL then 0 else count(*)来代替。

  上述SQL语句可以改写为：

 版权所有 © 华为云计算技术有限公司

```
with ssq as
(
   select count(*) cnt, c1 from t2 group by c1
)
select case when
          ssq.cnt is null then 0
          else ssq.cnt
      end cnt, t1.c1, t3.c1
from t1 left join ssq on ssq.c1 = t1.c1,t3
where t1.c1 = t3.c1
order by ssq.cnt, t1.c1;
```

改写后的执行计划为：

```
 QUERY PLAN
-----------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Sort
         Sort Key: (count(*)), t1.c1
         ->  Hash Join
               Hash Cond: (t1.c1 = t3.c1)
               ->  Hash Left Join
                     Hash Cond: (t1.c1 = t2.c1)
                     ->  Seq Scan on t1
                     ->  Hash
                           ->  HashAggregate
                                 Group By Key: t2.c1
                                 ->  Seq Scan on t2
               ->  Hash
                     ->  Seq Scan on t3
(15 rows)
```

- 相关条件为不等值场景

  例如：

  ```
  select t1.c1, t1.c2
  from t1
  where t1.c1 = (select agg() from t2.c2 > t1.c2);
  ```

  对于非等值相关条件的SubLink目前无法提升，从语义上可以通过做2次join（一次CorrelationKey，一次rownum自关联）达到提升改写的目的。

  改写方案有两种：

  - 子查询改写方式

    ```
    select t1.c1, t1.c2
    from t1, (
       select t1.rowid, agg() aggref
       from t1,t2
       where t1.c2 > t2.c2 group by t1.rowid
    ) dt /* derived table */
    where t1.rowid = dt.rowid AND t1.c1 = dt.aggref;
    ```

  - CTE改写方式

    ```
    WITH dt as
    (
       select t1.rowid, agg() aggref
       from t1,t2
       where t1.c2 > t2.c2 group by t1.rowid
    )
    select t1.c1, t1.c2
    from t1, dt
    where t1.rowid = dt.rowid AND
    t1.c1 = dt.aggref;
    ```

 版权所有 © 华为云计算技术有限公司

须知

- 目前DWS尚无高效的实现表、中间结果集的全局唯一rowid因此目前此类场景很难改写，建议通过业务层进行规避，或者可以使用t1.xc_node_id + t1.ctid进行rowid关联，但是xc_node_id的重复率较高会导致join关联效率变低，而xc_node_id+ctid类型无法作为hashjoin的关联条件。
- 对于AGG类型为count(*)时需要进行CASE-WHEN对没有match的场景补0处理，非COUNT(*)场景NULL处理。
- CTE改写方式如果有sharescan支持性能上能够更优。

更多优化示例

示例1：修改基表为REPLICATION表，并且在过滤列上创建索引。

```
create table master_table (a int);
create table sub_table(a int, b int);
select a from master_table group by a having a in (select a from sub_table);
```

上述示例中存在一个相关性子查询，为了提升查询的性能，可以将sub_table修改为一个REPLICATION表，并且在字段a上创建一个index。

示例2：修改select语句，将子查询修改为和主表的join，或者修改为可以提升的subquery，但是在修改前后需要保证语义的正确性。

```
explain (costs off)select * from master_table as t1 where t1.a in (select t2.a from sub_table as t2 where t1.a
= t2.b);
                      QUERY PLAN
------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on master_table t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t1.a = t2.b)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                             Spawn on: All datanodes
                             ->  Seq Scan on sub_table t2
(11 rows)
```

上面示例计划中存在一个subPlan，为了消除这个subPlan可以修改语句为：

```
explain(costs off) select * from master_table as t1 where exists (select t2.a from sub_table as t2 where t1.a
= t2.b and t1.a = t2.a);
                   QUERY PLAN
-------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Hash Semi Join
         Hash Cond: (t1.a = t2.b)
         ->  Seq Scan on master_table t1
         ->  Hash
               ->  Streaming(type: REDISTRIBUTE)
                     Spawn on: All datanodes
                     ->  Seq Scan on sub_table t2
(9 rows)
```

从计划可以看出，subPlan消除了，计划变成了两个表的semi join，这样会大幅度提高执行效率。

 版权所有 © 华为云计算技术有限公司

16.4.7.4 统计信息调优

统计信息调优介绍

DWS是基于代价估算生成的最优执行计划。优化器需要根据ANALYZE收集的统计信息行数估算和代价估算，因此统计信息对优化器行数估算和代价估算起着至关重要的作用。通过ANALYZE收集全局统计信息，主要包括：pg_class表中的relpages和reltuples；pg_statistic表中的stadistinct、stanullfrac、stanumbersN、stavaluesN、histogram_bounds等。

实例分析 1：未收集统计信息导致查询性能差

在很多场景中，由于查询中涉及到的表或列没有收集统计信息，对查询性能产生很大的影响。

示例表的表结构如下所示：

```
CREATE TABLE LINEITEM
(
L_ORDERKEY        BIGINT        NOT NULL
, L_PARTKEY       BIGINT        NOT NULL
, L_SUPPKEY       BIGINT        NOT NULL
, L_LINENUMBER    BIGINT        NOT NULL
, L_QUANTITY      DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT      DECIMAL(15,2) NOT NULL
, L_TAX           DECIMAL(15,2) NOT NULL
, L_RETURNFLAG    CHAR(1)       NOT NULL
, L_LINESTATUS    CHAR(1)       NOT NULL
, L_SHIPDATE      DATE          NOT NULL
, L_COMMITDATE    DATE          NOT NULL
, L_RECEIPTDATE   DATE          NOT NULL
, L_SHIPINSTRUCT  CHAR(25)      NOT NULL
, L_SHIPMODE      CHAR(10)      NOT NULL
, L_COMMENT       VARCHAR(44)   NOT NULL
) with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
(
O_ORDERKEY       BIGINT        NOT NULL
, O_CUSTKEY      BIGINT        NOT NULL
, O_ORDERSTATUS  CHAR(1)       NOT NULL
, O_TOTALPRICE   DECIMAL(15,2) NOT NULL
, O_ORDERDATE    DATE NOT NULL
, O_ORDERPRIORITY CHAR(15)     NOT NULL
, O_CLERK        CHAR(15)      NOT NULL
, O_SHIPPRIORITY  BIGINT       NOT NULL
, O_COMMENT      VARCHAR(79)   NOT NULL
)with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
```


版权所有 © 华为云计算技术有限公司

```
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
order by
numwait desc;
```

可以通过如下方法确认查询中涉及的表或列是否执行过ANALYZE收集统计信息。

1. 通过explain verbose执行query分析执行计划时会提示WARNING信息：
   ```
   WARNING:Statistics in some tables or columns(public.lineitem(l_receiptdate,l_commitdate,l_orderkey, l_suppkey), public.orders(o_orderstatus,o_orderkey)) are not collected.
   HINT:Do analyze for them in order to generate optimized plan.
   ```
2. 通过在pg_log目录下的日志文件中查找以下类似信息来确认是当前执行的query是否由于没有收集统计信息导致查询性能变差。
   ```
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] LOG:Statistics in some tables or columns(public.lineitem(l_receiptdate, l_commitdate,l_orderkey,
   .l_suppkey), public.orders(o_orderstatus,o_orderkey)) are not collected.
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] HINT:Do analyze for them in order to generate optimized plan.
   ```

查看和确认涉及的表或列没有执行ANALYZE后，可以通过对WARNING或日志中上报的表或列执行ANALYZE，以解决未收集统计信息导致查询变慢的问题。

实例分析 2：设置 cost_param 对查询性能优化

请参考**案例：设置cost_param对查询性能优化**。

实例分析 3：多表 join 的复杂查询存在中间结果不准调优

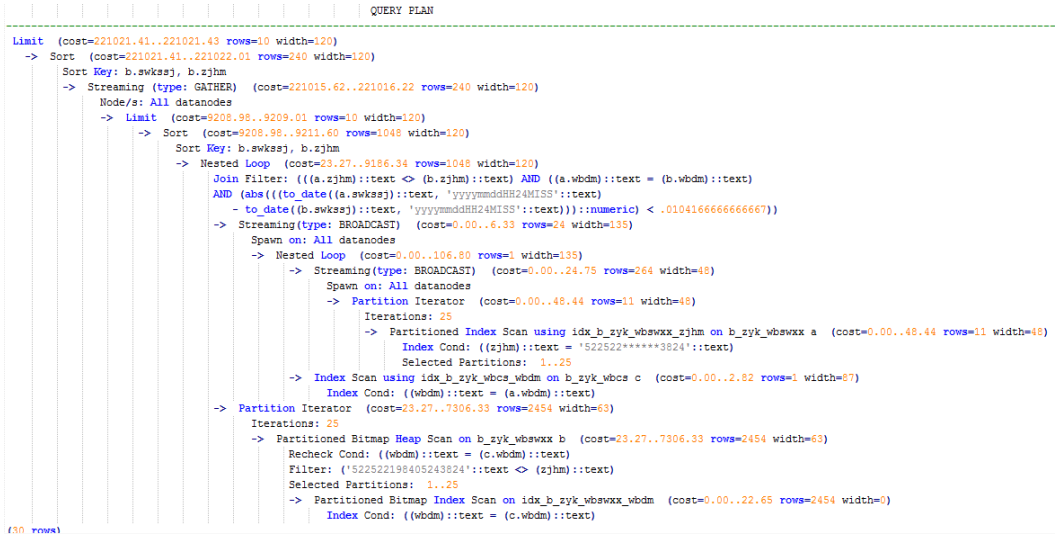
现象描述：查询与指定人在前后15分钟内、同一网吧登记上网的人员信息：

```
SELECT
C.WBM,
C.DZQH,
C.DZ,
B.ZJHM,
B.SWKSSJ,
B.XWSJ
FROM
b_zyk_wbswxx A,
b_zyk_wbswxx B,
b_zyk_wbcs C
WHERE
A.ZJHM = '522522******3824'
AND A.WBDM = B.WBDM
AND A.WBDM = C.WBDM
AND abs(to_date(A.SWKSSJ,'yyyymmddHH24MISS') - to_date(B.SWKSSJ,'yyyymmddHH24MISS')) <
INTERVAL '15 MINUTES'
ORDER BY
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

执行计划如**图16-6**所示。该查询实际耗时约12秒。

 版权所有 © 华为云计算技术有限公司

图 16-6 应用 unlogged table 案例（一）



```
                                                
```

```
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
order by
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

上述查询耗时约7秒，执行计划如图16-7所示。

图 16-7 应用 unlogged table 案例（二）

```
                                                
```

为解决上述问题，建议使用unlogged table保存结果集。

```
//创建保存结果集的unlogged table
CREATE UNLOGGED TABLE temp_result
(
WBM      NVARCHAR2(70),
DZQH     NVARCHAR2(6),
DZ       NVARCHAR2(70),
IPDZ     NVARCHAR2(39),
ZJHM     NVARCHAR2(18),
XM       NVARCHAR2(30),
SWKSSJ   date,
XWSJ     date,
SWZDH    NVARCHAR2(32)
);

//将结果集插入unlogged table，插入耗时约3秒。
INSERT INTO
temp_result
SELECT
A.WBM,
A.DZQH,
A.DZ,
A.IPDZ,
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
;

//查询结果集表进行分页显示，分页查询耗时约10ms。
SELECT
*
FROM
temp_result
ORDER BY
SWKSSJ,
ZJHM
LIMIT 10 OFFSET 0;
```

⚠ 注意

通过ANALYZE收集全局统计信息，通常会改善查询性能。

如果遇到性能问题：可以使用plan hint来调整到之前的查询计划，详情请参见**使用Plan Hint进行调优**。

16.4.7.5 算子级调优

算子级调优介绍

一个查询语句要经过多个算子步骤才会输出最终的结果。由于个别算子耗时过长导致整体查询性能下降的情况比较常见。这些算子是整个查询的瓶颈算子。通用的优化手段是EXPLAIN ANALYZE/PERFORMANCE命令查看执行过程的瓶颈算子，然后进行针对性优化。

版权所有 © 华为云计算技术有限公司

如下面的执行过程信息中，Hashagg算子的执行时间占总时间的：(51016-13535)/56476 ≈66%，此处Hashagg算子就是这个查询的瓶颈算子，在进行性能优化时应当优先考虑此算子的优化。

```
 id |                                operation                                |       A-time        |  A-rows   | E-rows |      Peak Memory      | E-memory | A-width | E-width |   E-costs
----+-------------------------------------------------------------------------+---------------------+-----------+--------+-----------------------+----------+---------+---------+-------------
  1 | ->  Row Adapter                                                         | 56476.397           |  10000000 |  237060 | 19KB                  |          |         |      20 | 20933222.75
  2 |   ->  Vector Streaming (type: GATHER)                                   | 55664.220           |  10000000 |  237060 | 243KB                 |          |         |      20 | 20933222.75
  3 |     ->  Vector Hash Aggregate                                           | [55124.685,55132.180] |  10000000 |  237060 | [29349KB, 29441KB]    | 16MB     | [20,20] |      20 | 20918406.50
  4 |       ->  Vector Streaming(type: REDISTRIBUTE)                          | [52519.781,53709.779] | 339364604 | 4856184 | [1219KB, 1219KB]      | 1MB      |         |      20 | 10461210.85
  5 |         ->  Vector Hash Aggregate                                       | [35675.636,51016.424] | 339364604 | 4856184 | [732850KB, 746894KB]  | 16MB     | [20,20] |      20 | 10457195.65
  6 |           ->  Vector Partition Iterator                                 | [9035.202,13565.884]  | 970000000 | 935838097 | [9KB, 9KB]          | 1MB      |         |      20 | 10195891.68
  7 |             ->  Partitioned CStore Scan on xuji.e_mp_day_energy_csv_1 | [9015.645,13535.346]  | 970000000 | 935838097 | [845KB, 845KB]      | 1MB      |         |      20 | 10195891.68
(7 rows)
```

算子级调优示例

示例1：基表扫描时，对于点查或者范围扫描等过滤大量数据的查询，如果使用SeqScan全表扫描会比较耗时，可以在条件列上建立索引选择IndexScan进行索引扫描提升扫描效率。

```
explain (analyze on, costs off) select * from store_sales where ss_sold_date_sk = 2450944;
 id |             operation             |       A-time        | A-rows | Peak Memory  | A-width
----+-----------------------------------+---------------------+--------+--------------+---------
  1 | ->  Streaming (type: GATHER)   | 3666.020            |   3360 | 195KB        |
  2 |    ->  Seq Scan on store_sales | [3594.611,3594.611] |   3360 | [34KB, 34KB] |

 Predicate Information (identified by plan id)
-------------------------------------------------
   2 --Seq Scan on store_sales
         Filter: (ss_sold_date_sk = 2450944)
         Rows Removed by Filter: 4968936
 create index idx on store_sales_row(ss_sold_date_sk);
CREATE INDEX
 explain (analyze on, costs off) select * from store_sales_row where ss_sold_date_sk = 2450944;
 id |                    operation                     |     A-time      | A-rows | Peak Memory  | A-width
----+--------------------------------------------------+-----------------+--------+--------------+----------
  1 | ->  Streaming (type: GATHER)                     | 81.524          |   3360 | 195KB        |
  2 |    ->  Index Scan using idx on store_sales_row | [13.352,13.352] |   3360 | [34KB, 34KB] |
```

上述例子中，全表扫描返回3360条数据，过滤掉大量数据，在ss_sold_date_sk列上建立索引后，使用IndexScan扫描效率显著提高，从3.6秒提升到13毫秒。

示例2：如果从执行计划中看，两表join选择了NestLoop，而实际行数比较大时，NestLoop Join可能执行比较慢。如下的例子中NestLoop耗时181秒，如果设置参数enable_mergejoin=off关掉Merge Join，同时设置参数enable_nestloop=off关掉NestLoop，让优化器选择HashJoin，则Join耗时提升至200多毫秒。

```
postgres=#  explain analyze select count(*) from store_sales ss, item i where ss.ss_item_sk = i.i_item_sk;
 id |                  operation                   |        A-time         |  A-rows   | E-rows  |  Peak Memory   | E-memory | A-width | E-width |   E-costs
----+----------------------------------------------+-----------------------+-----------+---------+----------------+----------+---------+---------+-------------
  1 | ->  Row Adapter                              | 184300.301            |         1 |       1 | 11KB           |          |         |       0 | 48629179.77
  2 |   ->  Vector Aggregate                       | 184300.280            |         1 |       1 | 181KB          |          |         |       0 | 48629179.77
  3 |     ->  Vector Streaming (type: GATHER)      | 184300.186            |         4 |       4 | 188KB          |          |         |       0 | 48629179.77
  4 |       ->  Vector Aggregate                   | [165575.394,184252.368] |         4 |       4 | [140KB, 140KB] | 1MB      |         |       0 | 48629179.61
  5 |         ->  Vector Nest Loop (6,7)           | [162918.848,181438.162] |   2880404 | 2880404 | [74KB, 74KB]   | 1MB      |         |       0 | 48627379.35
  6 |           ->  CStore Scan on store_sales ss  | [15.660,16.229]       |   2880404 | 2880404 | [490KB, 490KB] | 1MB      |         |       4 | 16683.10
  7 |           ->  Vector Materialize             | [118314.321,132478.454] | 12968211302 |   18000 | [869KB, 900KB] | 16MB     | [8,8]   |       4 | 3890.00
  8 |             ->  CStore Scan on item i        | [0.234,0.243]         |     18000 |   18000 | [476KB, 476KB] | 1MB      |         |       4 | 3867.50
(8 rows)



postgres=#  set enable_nestloop=off;
SET
postgres=#  set enable_mergejoin=off;
SET
ppostgres=# explain analyze select count(*) fpostgres=# ales ss, item i where ss.ss_item_sk = i.i_item_sk;
 id |                  operation                   |        A-time         | A-rows  | E-rows  |  Peak Memory   | E-memory | A-width | E-width | E-costs
----+----------------------------------------------+-----------------------+---------+---------+----------------+----------+---------+---------+----------
  1 | ->  Row Adapter                              | 291.066               |       1 |       1 | 11KB           |          |         |       0 | 32308.66
  2 |   ->  Vector Aggregate                       | 291.052               |       1 |       1 | 181KB          |          |         |       0 | 32308.66
  3 |     ->  Vector Streaming (type: GATHER)      | 290.973               |       4 |       4 | 188KB          |          |         |       0 | 32308.66
  4 |       ->  Vector Aggregate                   | [220.792,234.532]     |       4 |       4 | [140KB, 140KB] | 1MB      |         |       0 | 32308.50
  5 |         ->  Vector Hash Join (6,7)           | [209.987,223.345]     | 2880404 | 2880404 | [236KB, 241KB] | 16MB     | [8,8]   |       0 | 30508.24
  6 |           ->  CStore Scan on store_sales ss  | [13.132,13.717]       | 2880404 | 2880404 | [490KB, 490KB] | 1MB      |         |       4 | 16683.10
  7 |           ->  CStore Scan on item i          | [0.214,0.246]         |   18000 |   18000 | [477KB, 477KB] | 1MB      |         |       4 | 3867.50
(7 rows)
```

示例3：通常情况下Agg选择HashAgg性能较好，如果大结果集选择了Sort+GroupAgg，则需要设置enable_sort=off，HashAgg耗时明显优于Sort+GroupAgg。

```
postgres=#  explain analyze select count(*) from store_sales group by ss_item_sk;
 id |                  operation                   |        A-time         | A-rows  | E-rows  |      Peak Memory      | E-memory | A-width | E-width | E-costs
----+----------------------------------------------+-----------------------+---------+---------+-----------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                              | 1977.385              |   18000 |   17644 | 20KB                  |          |         |       4 | 92875.24
  2 |   ->  Vector Streaming (type: GATHER)        | 1973.617              |   18000 |   17644 | 1946KB                |          |         |       4 | 92875.24
  3 |     ->  Vector Sort Aggregate                | [1784.800,1883.243]   |   18000 |   17644 | [273KB, 273KB]        | 1MB      |         |       4 | 92186.02
  4 |       ->  Vector Sort                        | [1752.270,1848.357]   | 2880404 | 2880404 | [128466KB, 135135KB]  | 16MB     | [8,8]   |       4 | 88541.40
  5 |         ->  CStore Scan on store_sales       | [12.483,13.548]       | 2880404 | 2880404 | [490KB, 490KB]        | 1MB      |         |       4 | 16683.10
(5 rows)
```


版权所有 © 华为云计算技术有限公司

```
postgres=#  set enable_sort=off;
SET
postgres=#  explain analyze select count(*) from store_sales group by ss_item_sk;
 id |              operation               |       A-time        | A-rows | E-rows |        Peak Memory        | E-memory | A-width | E-width | E-costs
----+--------------------------------------+---------------------+--------+--------+---------------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                      | 838.218             |  18000 |  17644 | 20KB                      |          |         |       4 | 21016.93
  2 |     ->  Vector Streaming (type: GATHER)  | 834.264         |  18000 |  17644 | 228KB                     |          |         |       4 | 21016.93
  3 |       ->  Vector Hash Aggregate      | [585.017,758.204]   |  18000 |  17644 | [262552KB, 262564KB] | 16MB     | [8,8]   |       4 | 20327.72
  4 |         ->  CStore Scan on store_sales | [12.540,13.941]   | 2880404 | 2880404 | [490KB, 490KB]       | 1MB      |         |       4 | 16683.10
(4 rows)
```

16.4.7.6 数据倾斜调优

数据倾斜问题是分布式架构的重要难题，它破坏了MPP架构中各个节点对等的要求，导致单节点（倾斜节点）所存储或者计算的数据量远大于其他节点，所以会造成以下危害：

- 存储上的倾斜会严重限制系统容量，在系统容量不饱和的情况下，由于单节点倾斜的限制，使得整个系统容量无法继续增长。
- 计算上的倾斜会严重影响系统性能，由于倾斜节点所需要运算的数据量远大于其它节点，导致倾斜节点降低系统整体性能。
- 数据倾斜还严重影响了MPP架构的扩展性。由于在存储或者计算时，会将相同值的数据放到同一节点，因此当倾斜数据（大量数据的值相同）出现之后，即使增加节点，系统瓶颈仍然受限于倾斜节点的容量或者性能。

DWS数据库针对数据倾斜问题给出了完整的解决方案，包括存储倾斜和计算倾斜两大问题，下面分别进行介绍。

存储层数据倾斜

DWS数据库中，数据分布存储在各个DN上，通过分布式执行提高查询的效率。但是，如果数据分布存在倾斜，则会导致分布式执行某些DN成为瓶颈，影响查询性能。这种情况通常是由于分布列选择不合理，可以通过调整分布列的方式解决。

分析过程：

步骤1 登录DWS控制台。在“集群列表”页面，找到需要查看监控的集群。在指定集群所在行的“操作”列，单击“监控面板”。选择“监控 > 节点监控 > 磁盘”，查看磁盘使用率。

说明

各个数据磁盘的利用率，会有不均衡的现象。正常情况下，利用率最高和利用率最低的磁盘空间相差不大，如果磁盘利用率相差超过了5%就要注意是不是有资源倾斜的情况。

步骤2 连接数据库，通过等待视图查看作业的运行情况，发现作业总是等待部分DN或者个别DN。

```
SELECT wait_status, count(*) as cnt FROM pgxc_thread_wait_status WHERE wait_status not like '%cmd%' AND wait_status not like '%none%' and wait_status not like '%quit%' group by 1 order by 2 desc;
```

步骤3 执行慢语句的explain performance显示，发现各个DN的基表scan的时间和行数不均衡。

```
explain performance select avg(ss_wholesale_cost) from store_sales;
```

```
rmance select avg(ss_wholesale_cost) from store_sales;

      operation                              |       A-time        |  A-rows  | E-rows  | E-distinct |   Peak Memory    | E-memory | A-wi
---------------------------------------------+---------------------+----------+---------+------------+------------------+----------+-----
                                             | 2452.321            |        1 |       1 |            | 11KB             |          |
te                                           | 2452.238            |        1 |       1 |            | 184KB            |          |
aming (type: GATHER)                         | 2452.010            |        4 |       4 |            | 108KB            |          |
ggregate                                     | [12.219,2425.225]   |        4 |       4 |            | [183KB, 184KB]   | 1MB      |
r Partition Iterator                         | [12.139,1189.187]   | 22831616 | 2880404 |            | [17KB, 17KB]     | 1MB      |
rtitioned CStore Scan on public.store_sales  | [5.929,1173.555]    | 22831616 | 2880404 |            | [482KB, 499KB]   | 1MB      |


identified by plan id)
---------------------------
rator

can on public.store_sales
s:  1..7
```


版权所有 © 华为云计算技术有限公司

- 基表scan的时间：最快的DN耗时5ms，最慢的DN耗时1173ms。
- 数据分布情况：某些DN有22831616行，其他DN都是0行，数据有严重倾斜。

```
5 --Vector Partition Iterator
      datanode1 (actual time=0.620..1189.187 rows=22831616 loops=1)
      datanode2 (actual time=14.346..14.346 rows=0 loops=1)
      datanode3 (actual time=14.424..14.424 rows=0 loops=1)
      datanode4 (actual time=12.139..12.139 rows=0 loops=1)
      datanode1 (CPU: ex c/r=1, ex row=22831616, ex cyc=40540820, inc cyc=3088825848)
      datanode2 (CPU: ex c/r=0, ex row=0, ex cyc=20852952, inc cyc=37297912)
      datanode3 (CPU: ex c/r=0, ex row=0, ex cyc=20005612, inc cyc=37501436)
      datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=16147884, inc cyc=31560524)
6 --Partitioned CStore Scan on public.store_sales
      datanode1 (actual time=2.611..1173.555 rows=22831616 loops=7)
      datanode2 (actual time=6.327..6.327 rows=0 loops=7)
      datanode3 (actual time=6.732..6.732 rows=0 loops=7)
      datanode4 (actual time=5.929..5.929 rows=0 loops=7)
      datanode1 (Buffers: shared hit=1359)
      datanode2 (Buffers: shared hit=55)
      datanode3 (Buffers: shared hit=55)
      datanode4 (Buffers: shared hit=55)
      datanode1 (CPU: ex c/r=133, ex row=22831616, ex cyc=3048285028, inc cyc=3048285028)
      datanode2 (CPU: ex c/r=0, ex row=0, ex cyc=16444960, inc cyc=16444960)
      datanode3 (CPU: ex c/r=0, ex row=0, ex cyc=17495824, inc cyc=17495824)
      datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=15412640, inc cyc=15412640)
```

步骤4 通过倾斜检查接口可以发现数据倾斜。

```
SELECT table_skewness('store_sales');
```

```
tpcds1xcpm=# select table_skewness('store_sales');
                table_skewness
-------------------------------------------
 ("datanode1            ",22831616,100.000%)
 ("datanode2            ",0,0.000%)
 ("datanode3            ",0,0.000%)
 ("datanode4            ",0,0.000%)
(4 rows)
```

```
SELECT table_distribution('public','store_sales');
```

```
tpcds1xcpm=# select table_distribution('public','store_sales');
           table_distribution
-----------------------------------------
 (public,store_sales,datanode1,1011752960)
 (public,store_sales,datanode4,33792000)
 (public,store_sales,datanode2,32129024)
 (public,store_sales,datanode3,33349632)
(4 rows)
```

----结束

处理方法--如何找到倾斜的表：

1. 数据库中表个数少于1W的场景下，可直接使用倾斜视图查询当前库内所有表的数据倾斜情况。
   ```
   SELECT * FROM pgxc_get_table_skewness ORDER BY totalsize DESC;
   ```
2. 数据库中表个数非常多（至少大于1W）的场景，因PGXC_GET_TABLE_SKEWNESS视图涉及全库查并计算非常全面的倾斜字段，所以可能会花费比较长的时间（小时级），建议参考PGXC_GET_TABLE_SKEWNESS视图定义，执行以下操作：
   - 8.1.2及之前集群版本中使用table_distribution()函数自定义输出，减少输出列进行计算优化，例如：
     ```
     SELECT schemaname,tablename,max(dnsize) AS maxsize, min(dnsize) AS minsize
     FROM pg_catalog.pg_class c
     ```

 版权所有 © 华为云计算技术有限公司

```
INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
INNER JOIN pg_catalog.table_distribution() s ON s.schemaname = n.nspname AND s.tablename
= c.relname
INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype = 'H'
GROUP BY schemaname,tablename;
```

- 8.1.3及以上集群版本中支持使用gs_table_distribution()函数，全库查询所有表的数据倾斜情况。全库表查询时，gs_table_distribution()函数优于table_distribution()函数；在大集群大数据量场景下，如果进行全库表查询，建议优先使用gs_table_distribution()函数。

```
SELECT schemaname,tablename,max(dnsize) AS maxsize, min(dnsize) AS minsize
FROM pg_catalog.pg_class c
INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
INNER JOIN pg_catalog.gs_table_distribution() s ON s.schemaname = n.nspname AND
s.tablename = c.relname
INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype = 'H'
GROUP BY schemaname,tablename;
```

说明

使用如下语句可快速查询到大表：

```
SELECT schemaname||'.'||tablename as table, sum(dnsize) as size FROM
gs_table_distribution() group by 1 order by 2 desc limit 10;
```

使用如下语句可快速查询表的倾斜率：

```
WITH skew AS
(
	SELECT
		schemaname,
		tablename,
		pg_catalog.sum(dnsize) AS totalsize,
		pg_catalog.avg(dnsize) AS avgsize,
		pg_catalog.max(dnsize) AS maxsize,
		pg_catalog.min(dnsize) AS minsize,
		(pg_catalog.max(dnsize) - pg_catalog.min(dnsize)) AS skewsize,
		pg_catalog.stddev(dnsize) AS skewstddev
	FROM pg_catalog.pg_class c
	INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
	INNER JOIN pg_catalog.gs_table_distribution() s ON s.schemaname = n.nspname
AND s.tablename = c.relname
	INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype IN('H',
'N')
	GROUP BY schemaname,tablename
)
SELECT
	schemaname,
	tablename,
	totalsize,
	avgsize::numeric(1000),
	(maxsize/totalsize)::numeric(4,3)  AS maxratio,
	(minsize/totalsize)::numeric(4,3)  AS minratio,
	skewsize,
	(skewsize/avgsize)::numeric(4,3)  AS skewratio,
	skewstddev::numeric(1000)
FROM skew
WHERE totalsize > 0;
```

处理方法--表的分布键的选择方法：

1. 如果此列的distinct值比较大，并且没有明显的数据倾斜，也可以把多列定义成分布列。

   查看distinct的大小：

   ```
   SELECT count(distinct column1) FROM table;
   ```

   查看数据是否存在倾斜：

   ```
   SELECT count(*) cnt, column1 FROM table group by column1 order by cnt limit 100;
   ```

 版权所有 © 华为云计算技术有限公司

2. 选用经常做JOIN或group by的列，可以减少STREAM运算。
3. 不推荐以下分布键选择方式：
   a. 分布列用默认值（第一列）。
   b. 分布列用sequence自增生成。
   c. 分布列用随机数生成（除非任意列，或者任意两列的组合做分布键都是倾斜的，一般不选用这种方法）。

计算层数据倾斜

即使通过修改表的分布键，使得数据存储在各个节点上是均衡的，但是在执行查询的过程中，仍然可能出现数据倾斜的问题。在运算过程中某个算子在DN上输出的结果集出现倾斜，从而导致此算子上层的运算出现计算倾斜。一般来说，这是由于在执行过程中，数据重分布导致的。

在查询执行的过程中，join key、group by key等不是表的分布列，因此需要按照join key、group by key上数据的hash值，让数据在各个DN之间进行重新分布，这个过程对应于计划中的Redistribute算子。当重分布列上的数据存在倾斜时，就会导致运行时的数据倾斜，即重分布后部分节点的数据远大于其他。倾斜节点需要处理更多的数据，导致倾斜节点的计算性能远低于其他节点。

如下例中，s表和t表join，join条件中的s.x和t.x均不是表的分布列，因此需要重分布（REDISTRIBUTE算子）。其中s.x列上存在倾斜值，t.x上不存在倾斜。id=6的stream算子在datanode2节点输出的结果集是其他DN的3倍，从而导致了计算倾斜。

```
select * from skew s,test t where s.x = t.x order by s.a limit 1;
 id |                    operation                    |        A-time
----+-------------------------------------------------+-----------------------
  1 | ->  Limit                                       | 52622.382
  2 |    ->  Streaming (type: GATHER)                 | 52622.374
  3 |       ->  Limit                                 | [30138.494,52598.994]
  4 |          ->  Sort                               | [30138.486,52598.986]
  5 |             ->  Hash Join (6,8)                 | [30127.013,41483.275]
  6 |                ->  Streaming(type: REDISTRIBUTE)    | [11365.110,22024.845]
  7 |                   ->  Seq Scan on public.skew s     | [2019.168,2175.369]
  8 |                ->  Hash                         | [2460.108,2499.850]
  9 |                   ->  Streaming(type: REDISTRIBUTE) | [1056.214,1121.887]
 10 |                      ->  Seq Scan on public.test t  | [310.848,325.569]

6 --Streaming(type: REDISTRIBUTE)
        datanode1 (rows=5050368)
        datanode2 (rows=15276032)
        datanode3 (rows=5174272)
        datanode4 (rows=5219328)
```

和存储倾斜相比，计算倾斜更难以提前识别，因此DWS提出了RLBT(Runtime Load Balance Technology)方案，用于解决运行时的计算倾斜问题，该特性由参数**skew_option**控制。RLBT方案主要分为两个层面，第一步是计算倾斜识别，第二步是计算倾斜解决。下面分别进行介绍。

1. 倾斜识别

   计算倾斜的识别，即预先识别计算过程中的重分布列是否存在倾斜数据。RLBT方案中给出了三个解决手段，统计信息识别，hint方式指定以及规则识别：

   - 统计信息识别

     需要用户先执行ANALYZE收集各表的统计信息，然后优化器能够自动利用统计信息对重分布键上的倾斜数据进行提前识别，对于存在倾斜的查询，生成相应的优化计划。在重分布键有多列的情况，只有所有列都属于同一个基表才能利用统计信息进行识别。

 版权所有 © 华为云计算技术有限公司

统计信息只能给出基表的倾斜情况，当基表某一列存在倾斜，其他列上带有过滤条件，或者经过和其他表的join之后，无法准确判断倾斜列上倾斜数据是否依旧存在。当skew_option为normal时，这里认为倾斜数据依旧存在，仍然会对基表中识别到的倾斜进行优化；当skew_option为lazy时，这里认为倾斜数据已经不再存在，也就不会进行相应的优化。

- hint方式指定

  统计信息有着一定的局限性，对于较为复杂的查询，其中间结果难以通过统计信息进行估算和识别倾斜数据。对于这种情况，DWS设计了hint手段，通过用户手动指定的方式，给定倾斜信息。优化器根据用户给定的倾斜信息，来对查询进行优化。详细hint使用语法参见**运行倾斜的hint**。

- 规则识别

  现在BI系统会产生大量带有outer join（left join、right join、full join）的SQL，outer join在匹配失败的情况下会补空产生大量NULL值，如果接下来在补空列上进行join或者group by操作，就会导致NULL值倾斜。当前RLBT技术会自动识别这种场景，并生成相应的NULL值倾斜优化计划。

2. 计算倾斜解决

   在解决倾斜时，目前针对最常见的join和agg算子进行了优化。

   - join优化

   基本思路是将倾斜数据和非倾斜数据进行隔离处理。主要分为以下三种情况：

   a. join两侧都需要做重分布：

      对倾斜侧做PART_REDISTRIBUTE_PART_ROUNDROBIN，其中对倾斜数据做roundrobin，非倾斜数据做redistribute；

      对非倾斜侧做PART_REDISTRIBUTE_PART_BROADCAST，其中对倾斜数据做broadcast，非倾斜数据做redistribute；

   b. join一侧需要重分布，另一侧不需要重分布：

      对需要重分布的一侧做PART_REDISTRIBUTE_PART_ROUNDROBIN；

      对不需要重分布的一侧做PART_LOCAL_PART_BROADCAST，其中对等于倾斜值的部分做broadcast，其余数据保留在本地。

   c. 对于有补NULL值的表：

      对该表做PART_REDISTRIBUTE_PART_LOCAL，其中将NULL值保留在本地，其余数据做redistribute。

   以前面的查询为例，s.x列上存在倾斜数据，倾斜数据的值为0。优化器通过统计信息，识别到了该倾斜数据，生成了倾斜优化计划如下：

```
 id |                                operation                                 |        A-time
----+--------------------------------------------------------------------------+-----------------------
  1 | ->  Limit                                                                | 23642.049
  2 |    ->  Streaming (type: GATHER)                                          | 23642.041
  3 |       ->  Limit                                                          | [23310.768,23618.021]
  4 |          ->  Sort                                                        | [23310.761,23618.012]
  5 |             ->  Hash Join (6,8)                                          | [20898.341,21115.272]
  6 |                ->  Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)    |
[7125.834,7472.111]
  7 |                   ->  Seq Scan on public.skew s                          | [1837.079,1911.025]
  8 |                ->  Hash                                                  | [2612.484,2640.572]
  9 |                   ->  Streaming(type: PART REDISTRIBUTE PART BROADCAST) | [1193.548,1297.894]
 10 |                      ->  Seq Scan on public.test t                       | [314.343,328.707]

  5 --Vector Hash Join (6,8)
      Hash Cond: s.x = t.x
      Skew Join Optimized by Statistic
  6 --Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)
      datanode1  (rows=7635968)
```

 版权所有 © 华为云计算技术有限公司

```
        datanode2 (rows=7517184)
        datanode3 (rows=7748608)
        datanode4 (rows=7818240)
```

上述执行计划中，可以看到Skew Join Optimized by Statistic的字样，代表该计划为倾斜优化计划，其中Statistic关键字代表该倾斜优化来自于统计信息，除此之外还有Hint和Rule，分别代表倾斜优化来自于hint语句和规则。对比前面的计划可以看到，这里对于非倾斜数据和倾斜数据做了分别处理。对于s表中的非倾斜数据，依旧按照原有的方案，根据数据的hash值进行重分布；而对于倾斜数据（即等于0的数据），则通过轮询发送的方式，均衡地发送到所有节点。通过这样的方式，解决了倾斜数据分布不均衡的问题。

同时，为了保证结果的正确性，需要对t表做相应的处理。对于t表中等于0（s.x表中的倾斜值）的数据做广播，对于其他数据，依旧根据数据的hash值进行重分布。

通过这样的方式，就解决了join操作中，数据倾斜的问题。从上面的结果来看，id=6的stream算子各个DN的输出结果已经非常均衡，同时查询端到端性能提升了1倍。

如果执行计划中Stream算子类型显示为HYBRID，则表示对不同的倾斜数据所应用的stream方式不同，例如以下计划：

```
EXPLAIN (nodes OFF, costs OFF) SELECT COUNT(*) FROM skew_scol s, skew_scol1 s1 WHERE s.b =
s1.c;
QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------
id |                                                  operation
----
+-----------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------
1 | ->  Aggregate
2 |    ->  Streaming (type: GATHER)
3 |       ->  Aggregate
4 |          ->  Hash Join (5,7)
5 |             ->  Streaming(type: HYBRID)
6 |                ->  Seq Scan on skew_scol s
7 |             ->  Hash
8 |                ->  Streaming(type: HYBRID)
9 |                   ->  Seq Scan on skew_scol1 s1

Predicate Information (identified by plan id)
------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
4 --Hash Join (5,7)
Hash Cond: (s.b = s1.c)
Skew Join Optimized by Statistic
5 --Streaming(type: HYBRID)
Skew Filter: (b = 1)
Skew Filter: (b = 0)
8 --Streaming(type: HYBRID)
Skew Filter: (c = 0)
Skew Filter: (c = 1)
```

数据1在skew_scol表上有倾斜，对倾斜数据做roundrobin，非倾斜数据做redistribute。

数据0在skew_scol表上是非倾斜侧，对倾斜数据做broadcast，非倾斜数据做redistribute。

由此可以看到两个stream类型分别是PART REDISTRIBUTE PART ROUNDROBIN和PART REDISTRIBUTE PART BROADCAST，这里标记stream类型为HYBRID类型。

- agg优化

对于agg操作，解决倾斜的思路与join操作不同，这里是通过首先在本DN内按照group by key进行去重操作，然后再进行重分布。因为经过DN内部去重之后，从

 版权所有 © 华为云计算技术有限公司

全局来看，每个值的数量都不会超过DN数，因此不会出现严重的数据倾斜问题。以如下query为例：

```
select c1, c2, c3, c4, c5, c6, c7, c8, c9, count(*) from t group by c1, c2, c3, c4, c5, c6, c7, c8, c9 limit 10;
```

原执行结果如下：

```
 id |               operation                |         A-time         | A-rows
----+----------------------------------------+------------------------+----------
  1 | ->  Streaming (type: GATHER)           | 130621.783             |       12
  2 |   ->  GroupAggregate                   | [85499.711,130432.341] |       12
  3 |     ->  Sort                           | [85499.509,103145.632] | 36679237
  4 |       ->  Streaming(type: REDISTRIBUTE) | [25668.897,85499.050]  | 36679237
  5 |         ->  Seq Scan on public.t       | [9835.069,10416.388]   | 36679237

  4 --Streaming(type: REDISTRIBUTE)
      datanode1 (rows=36678837)
      datanode2 (rows=100)
      datanode3 (rows=100)
      datanode4 (rows=200)
```

其中存在大量倾斜数据，导致数据按照group by key进行重分布之后，datanode1的数据量是其他节点的数十万倍。在倾斜优化之后，首先在本DN进行一次group by操作，达到数据去重的效果，然后再进行重分布，可以发现基本没有数据倾斜的问题出现。

```
 id |               operation                |         A-time
----+----------------------------------------+------------------------
  1 | ->  Streaming (type: GATHER)           | 10961.337
  2 |   ->  HashAggregate                    | [10953.014,10953.705]
  3 |     ->  HashAggregate                  | [10952.957,10953.632]
  4 |       ->  Streaming(type: REDISTRIBUTE) | [10952.859,10953.502]
  5 |         ->  HashAggregate              | [10084.280,10947.139]
  6 |           ->  Seq Scan on public.t     | [4757.031,5201.168]

 Predicate Information (identified by plan id)
-----------------------------------------------
  3 --HashAggregate
      Skew Agg Optimized by Statistic

  4 --Streaming(type: REDISTRIBUTE)
      datanode1 (rows=17)
      datanode2 (rows=8)
      datanode3 (rows=8)
      datanode4 (rows=14)
```

适用范围

- join算子
  - 支持nest loop，merge join，hash join等join方式；
  - 当倾斜数据处于join的left侧时，支持inner join，left join，semi join，anti join；当倾斜属于位于join的right侧时，支持inner join，right join，right semi join，right anti join。
  - 通过统计信息得到的倾斜优化计划，优化器会根据代价判断该计划是否为最优计划。通过hint和规则会强制生成倾斜优化计划。
- agg算子
  - array_agg、string_agg、subplan in agg qual这几种场景不支持优化；
  - 通过统计信息识别到的倾斜优化计划会受到代价、plan_mode_seed参数、best_agg_plan参数影响，而hint、规则识别到的不会。

 版权所有 © 华为云计算技术有限公司

16.4.7.7 磁盘缓存主动预热调优

该功能仅9.1.0.200及以上版本支持。

背景介绍

当前存算分离架构为了降低存储成本会将用户数据存储到obs，这样会导致用户每次查询数据时都要发生网络IO去obs取数据，因此当前存算分离架构提供了磁盘缓存的能力，将用户预查询的数据缓存到本地磁盘，当实际查询数据流程时，数据已经缓存在本地，从而提升查询速度，在降低用户存储成本的同时尽可能的降低性能损耗。

当前采用了LRU2Q的算法来管理磁盘缓存。LRU2Q共包含3个队列，分别是A1in/A1out/Am，其中数据首次命中时会进入到A1in队列中，后续如果A1in队列满了（可调整guc参数，通常默认值为整个队列大小的0.25倍），会淘汰到A1out（实际并不存储数据到缓存，只是A1in队列淘汰的延长记忆），等A1out队列中的block被命中时，才会插入到Am队列。在Am队列中的数据，是最热的数据。

上述LRU2Q的淘汰策略对普通查询来说已经足够了，但是可能会存在的场景是：用户会将一些大表（比如查询数据接近磁盘缓存大小）和一些小表频繁join，此时如果继续按照普通LRU2Q的淘汰策略，对于频繁使用的小表来说可能会发生A1in满了然后淘汰到A1out，再次命中进入Am的过程，这样会频繁发生网络IO去重新缓存小表的数据，此时会极大的降低大表join小表的性能。

调优语法

当前提供了一种新的调优策略，即允许用户将上述描述场景中的小表数据直接入到Am队列中，保证小表的数据一直是热数据，join时不会频繁的发生网络IO而去重新加载小表数据，语法格式支持以下三种：

1. 发生实际查询操作，并将查询的数据直接进入到Am队列。
   ```
   explain warmup hot select ...;
   ```
   ```
   postgres=# explain warmup hot select * from orders13;
                          QUERY PLAN
   ------------------------------------------------

         ====== Cache Data =====
   ---------------------------------
    public.orders13:
          Read Cache Size : 39.095 MB
          Write Cache Size : 39.095 MB
           Avg Write Cache time : 5401.012ms
   (7 rows)

   postgres=# select * from pgxc_disk_cache_all_stats;
    node_name | total_read | local_read | remote_read | hit_rate | cache_size | fill_rate | temp_file_size | alin_size | alout_size | am_size | alin_fill_rate | alout_fill_rate | am_fill_rate | fd | pin_block_count
   -----------+------------+------------+-------------+----------+------------+-----------+----------------+-----------+------------+---------+----------------+-----------------+--------------+----+-----------------
    dn_1      |        239 |        239 |           0 |   100.00 |      81312 |      1.55 |              0 |         0 |          0 |   81312 |           0.00 |            0.00 |         2.07 | 80 |               0
    dn_2      |          0 |          0 |           0 |     0.00 |          0 |      0.00 |              0 |         0 |          0 |       0 |           0.00 |            0.00 |         0.00 |  0 |               0
   (2 rows)
   ```

2. 发生实际查询操作，查询的数据按照A1in > A1out > Am的顺序进行。
   ```
   explain warmup select ...;
   ```
   ```
   postgres=# explain warmup select * from orders13;
                          QUERY PLAN
   ------------------------------------------------

         ====== Cache Data =====
   ---------------------------------
    public.orders13:
          Read Cache Size : 39.095 MB
          Write Cache Size : 39.095 MB
          Avg Write Cache time : 22612.784ms
   (7 rows)

   postgres=# select * from pgxc_disk_cache_all_stats;
    node_name | total_read | local_read | remote_read | hit_rate | cache_size | fill_rate | temp_file_size | alin_size | alout_size | am_size | alin_fill_rate | alout_fill_rate | am_fill_rate | fd | pin_block_count
   -----------+------------+------------+-------------+----------+------------+-----------+----------------+-----------+------------+---------+----------------+-----------------+--------------+----+-----------------
    dn_1      |        239 |        239 |           0 |   100.00 |      81312 |      1.55 |              0 |     81312 |          0 |       0 |           6.20 |            0.00 |         0.00 | 80 |               0
    dn_2      |          0 |          0 |           0 |     0.00 |          0 |      0.00 |              0 |         0 |          0 |       0 |           0.00 |            0.00 |         0.00 |  0 |               0
   (2 rows)
   ```

3. 不发生实际的查询操作，和之前explain逻辑保持不变。
   ```
   explain select ...;
   ```

16.4.7.8 SQL 语句改写规则

根据数据库的SQL执行机制以及大量的实践，总结发现：通过一定的规则调整SQL语句，在保证结果正确的基础上，能够提高SQL执行效率。如果遵守下列规则，能够大幅度提升业务查询效率。

- **使用union all代替union**

 版权所有 © 华为云计算技术有限公司

union在合并两个集合时会执行去重操作，而union all则直接将两个结果集合并、不执行去重。执行去重会消耗大量的时间，因此，在一些实际应用场景中，如果通过业务逻辑已确认两个集合不存在重叠，可用union all替代union以便提升性能。

- **join列增加非空过滤条件**

  若join列上的NULL值较多，则可以加上is not null过滤条件，以实现数据的提前过滤，提高join效率。

- **not in转not exists**

  not in语句需要使用nestloop anti join来实现，而not exists则可以通过hash anti join来实现。在join列不存在null值的情况下，not exists和not in等价。因此在确保没有null值时，可以通过将not in转换为not exists，通过生成hash join来提升查询效率。

  如下所示，如果t2.d2字段中没有null值（t2.d2字段在表定义中not null）查询可以修改为

  ```
  SELECT * FROM t1 WHERE  NOT EXISTS (SELECT * FROM t2 WHERE t1.c1=t2.d2);
  ```

  产生的计划如下：

  图 16-9 not exists 执行计划

  ```
   id |                 operation
  ----+-------------------------------------------
    1 | ->  Streaming (type: GATHER)
    2 |    ->  Hash Right Anti Join (3, 5)
    3 |       ->  Streaming(type: REDISTRIBUTE)
    4 |          ->  Seq Scan on t2
    5 |       ->  Hash
    6 |          ->  Seq Scan on t1

  Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Right Anti Join (3, 5)
          Hash Cond: (t2.d2 = t1.c1)
  (13 rows)
  ```

- **选择hashagg。**

  查询中GROUP BY语句如果生成了groupagg+sort的plan性能会比较差，可以通过加大work_mem的方法生成hashagg的plan，因为不用排序而提高性能。

- **尝试将函数替换为case语句。**

  DWS函数调用性能较低，如果出现过多的函数调用导致性能下降很多，可以根据情况把可下推函数的函数改成CASE表达式。

- **避免对索引使用函数或表达式运算。**

  对索引使用函数或表达式运算会停止使用索引转而执行全表扫描。

- **尽量避免在where子句中使用!=或<>操作符、null值判断、or连接、参数隐式转换。**

- **对复杂SQL语句进行拆分。**

  对于过于复杂并且不易通过以上方法调整性能的SQL可以考虑拆分的方法，把SQL中某一部分拆分成独立的SQL并把执行结果存入临时表，拆分常见的场景包括但不限于：

  - 作业中多个SQL有同样的子查询，并且子查询数据量较大。

 版权所有 © 华为云计算技术有限公司

- Plan cost计算不准，导致子查询hash bucket太小，比如实际数据1000W行，hash bucket只有1000。
- 函数（如substr,to_number）导致大数据量子查询选择度计算不准。
- 多DN环境下对大表做broadcast的子查询。

16.4.8 优化器参数调整

本节将介绍影响DWS SQL调优性能的关键CN配置参数，配置方法参见**设置GUC参数**。

表 16-18 CN 配置参数

参数/参考值	描述
enable_nestloop=on	控制查询优化器对嵌套循环连接（Nest Loop Join）类型的使用。当设置为“on”后，优化器优先使用Nest Loop Join；当设置为“off”后，优化器在存在其他方法时将优先选择其他方法。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_nestloop to off;` 此参数默认设置为“on”，但实际调优中应根据情况选择是否关闭。一般情况下，在三种join方式（Nested Loop、Merge Join和Hash Join）里，Nested Loop性能较差，实际调优中可以选择关闭。
enable_bitmapscan=on	控制查询优化器对位图扫描规划类型的使用。设置为“on”，表示使用；设置为“off”，表示不使用。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行命令如下命令： `SET enable_bitmapscan to off;` bitmapscan扫描方式适用于“where a > 1 and b > 1”且a列和b列都有索引这种查询条件，但有时其性能不如indexscan。因此，现场调优如发现查询性能较差且计划中有bitmapscan算子，可以关闭bitmapscan，看性能是否有提升。
enable_fast_query_shipping=on	控制查询优化器是否使用分布式框架，执行快速执行计划。设置为“on”，表示执行计划在CN和DN上各自生成；设置为“off”，表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_fast_query_shipping to off;`
enable_hashagg=on	控制优化器对Hash聚集规划类型的使用。
enable_hashjoin=on	控制优化器对Hash连接规划类型的使用。

版权所有 © 华为云计算技术有限公司

参数/参考值	描述
enable_mergejoin=on	控制优化器对融合连接规划类型的使用。
enable_indexscan=on	控制优化器对索引扫描规划类型的使用。
enable_indexonlyscan=on	控制优化器对仅索引扫描规划类型的使用。
enable_seqscan=on	控制优化器对顺序扫描规划类型的使用。完全消除顺序扫描是不可能的，但是关闭这个变量会让优化器在存在其他方法的时候优先选择其他方法。
enable_sort=on	控制优化器使用的排序步骤。该设置不可能完全消除明确的排序，但是关闭这个变量可以让优化器在存在其他方法的时候优先选择其他方法。
enable_broadcast=on	控制查询优化器对于broadcast广播模式数据传输的使用。此方式网络传输数据量较大，因此当网络传输节点（Stream）实际数据量较大而估算不准时，可以将该参数设置为off，看性能是否有提升。
enable_redistribute=on	控制查询优化器对于local redistribute和split redistribute重分布模式数据传输的使用。此参数与enable_broadcast是对应关系。优化器可能会对local broadcast和split broadcast代价估计偏大，从而选择了local redistribute或者split redistribute重分布的计划。这有可能导致性能劣化。因此当网络传输节点（Stream）实际数据量较小时，可以将该参数设置为off，让优化器优先选择broadcast广播模式，看性能是否有提升。
rewrite_rule	控制优化器是否启用特定组合的重写规则。

16.4.9 使用 Plan Hint 进行调优

16.4.9.1 Plan Hint 调优概述

Plan Hint为用户提供了直接影响执行计划生成的手段，用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息，指定配置参数的值等多个手段来进行执行计划的调优，以提升查询的性能。

功能描述

Plan Hint支持在SELECT、INSERT、UPDATE、MERGE、DELETE关键字后通过如下形式指定：

```
/*+ <plan hint> */
```

可以同时指定多个hint，之间使用空格分隔。hint只能hint当前层的计划，对于子查询计划的hint，需要在子查询对应的关键字后指定hint。

例如：

版权所有 © 华为云计算技术有限公司

```
select /*+ <plan_hint1> <plan_hint2> */ * from t1, (select /*+ <plan_hint3> */ from t2) where 1=1;
```

其中<plan_hint1>，<plan_hint2>为外层查询的hint，<plan_hint3>为内层子查询的hint。

须知

如果在视图定义（CREATE VIEW）时指定hint，则在该视图每次被应用时会使用该hint。

当使用random plan功能（参数plan_mode_seed不为0）时，查询指定的plan hint不会被使用。

支持范围

当前版本Plan Hint支持的范围如下，后续版本会进行增强。

- 指定Join顺序的Hint - leading hint。
- 指定Join方式的Hint，仅支持除semi/anti join，unique plan之外的常用hint。
- 指定结果集行数的Hint。
- 指定Stream方式的Hint。
- 指定Scan方式的Hint，仅支持常用的tablescan，indexscan和indexonlyscan的hint。
- 指定子链接块名的Hint。
- 指定倾斜信息的Hint，仅支持Join与HashAgg的重分布过程倾斜。
- 指定Agg重分布列Hint。仅8.1.3.100及以上集群版本支持。
- 指定子查询不提升的Hint。仅8.2.0及以上集群版本支持。
- 指定配置参数值的Hint，仅支持部分配置参数，详见**配置参数的hint**。

注意事项

- 不支持Sort、Setop和Subplan的hint。
- 不支持SMP和Node Group场景下的Hint。
- 不支持对INSERT语句的目标表使用Hint。

示例

本章节使用同一个语句进行示例，便于Plan Hint支持的各方法作对比，示例语句及不带hint的原计划如下所示：

```
explain
select i_product_name product_name
,i_item_sk item_sk
,s_store_name store_name
,s_zip store_zip
,ad2.ca_street_number c_street_number
,ad2.ca_street_name c_street_name
,ad2.ca_city c_city
,ad2.ca_zip c_zip
,count(*) cnt
,sum(ss_wholesale_cost) s1
,sum(ss_list_price) s2
```

 版权所有 © 华为云计算技术有限公司

```
,sum(ss_coupon_amt) s3
FROM   store_sales
,store_returns
,store
,customer
,promotion
,customer_address ad2
,item
WHERE  ss_store_sk = s_store_sk AND
ss_customer_sk = c_customer_sk AND
ss_item_sk = i_item_sk and
ss_item_sk = sr_item_sk and
ss_ticket_number = sr_ticket_number and
c_current_addr_sk = ad2.ca_address_sk and
ss_promo_sk = p_promo_sk and
i_color in ('maroon','burnished','dim','steel','navajo','chocolate') and
i_current_price between 35 and 35 + 10 and
i_current_price between 35 + 1 and 35 + 15
group by i_product_name
,i_item_sk
,s_store_name
,s_zip
,ad2.ca_street_number
,ad2.ca_street_name
,ad2.ca_city
,ad2.ca_zip
;
```

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |  E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 3401632.49
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 3401632.49
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 3401630.82
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 3401630.78
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 16MB     |     169 | 3401630.42
  6 |                ->  Vector Hash Join (7,20)                               |          7 | 43MB     |     173 | 3400343.15
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)               |          7 | 1MB      |     123 | 3395775.64
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 3395775.48
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 3386294.72
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 3386294.56
 11 |                               ->  Vector Hash Join (12,14)               |          7 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator           |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)            |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 17 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 18 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                      ->  CStore Scan on customer_address ad2             |    6000000 | 1MB      |      58 | 5770.00
 21 |                   ->  CStore Scan on promotion                           |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```

16.4.9.2 Join 顺序的 Hint

功能描述

指明join的顺序，包括内外表顺序。

语法格式

- 单层圆括号()，仅指定join顺序，不指定内外表顺序。

```
leading(join_table_list)
leading(@block_name join_table_list)
```

- 双层圆括号(())，同时指定join顺序和内外表顺序，内外表顺序仅在最外层生效。

```
leading((join_table_list))
leading(@block_name (join_table_list))
```

- 单层方括号[]，同时指定[]这层的join顺序和内外表顺序。

```
leading[join_table_list]
leading[@block_name join_table_list]
```


版权所有 © 华为云计算技术有限公司

- 单层圆括号()和单层方括号[]混合使用，同时指定join顺序和任意层内外表顺序。圆括号()这一层只指定join顺序，不指定内外表顺序，方括号[]这一层同时指定join顺序和内外表顺序。

```
leading(join_table_list1 [join_table_list2])
leading[join_table_list1 [join_table_list2]]
leading[join_table_list1 (join_table_list2)]
leading(@block_name join_table_list1 [join_table_list2])
leading[@block_name join_table_list1 [join_table_list2]]
leading[@block_name join_table_list1 (join_table_list2)]
```

须知

单层方括号[]可以和单层圆括号()混合使用，指定任意层的内外表顺序。不支持单层[]和双层(())一起使用。

参数说明

- **join_table_list**

  为表示表join顺序的hint字符串，可以包含当前层的任意个表（别名），或对于子查询提升的场景，也可以包含子查询的hint别名，同时任意表可以使用括号指定优先级，表之间使用空格分隔。

  join table list中指定的表需要满足以下要求，否则会报语义错误：

  - list中的表必须在当前层或提升的子查询中存在。
  - list中的表在当前层或提升的子查询中必须是唯一的。如果不唯一，需要使用不同的别名进行区分。
  - 同一个表只能在list里出现一次。
  - 如果表存在别名，则list中的表必须使用别名。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

- **block_name**

  语句块的block_name。表示该hint在block_name对应的子查询语句块中生效。

 版权所有 © 华为云计算技术有限公司

须知

- 默认为语句生成block_name。
- CN轻量化不生成block_name。
- create table as select、select into、select、insert、update、delete、merge语句生成block_name。
- block_name的命名规则：
  - 为select、insert、update、delete、merge语句自动生成blockname，block_name的命名格式分别为sel$n、ins$n、upd$n、del$n、mer$n，n从1开始计数，不同类型语句之间，计数不累加，同一类型语句之间计数累加。

    例如：

    ```
    INSERT INTO t SELECT * FROM t1 WHERE a1 IN (select * from t2);
    ```

    --------sel$2-------

    ----------------------sel$1----------------------

    --------------------------------ins$1--------------------------
  - 在优化器之前递归为每个语句块分配block_name。

    首先为当前语句块根据语句类型分配block_name，然后按照下面的顺序进行遍历，并为其中的语句块分配block_name：

    1. 遍历目标列
    2. 遍历merge语句的源表中的目标列
    3. 遍历merge语句中的action（update/insert）
    4. 遍历returning子句
    5. 遍历from中的join条件和where条件（join条件优先于where条件）
    6. 如果是集合操作，遍历集合的各个分支（union/intersect/except）
    7. 遍历having子句
    8. 遍历limit offset子句
    9. 遍历limit count子句
    10. 遍历cte
    11. 遍历from后的表
    12. 遍历upsert子句
  - 在优化器的重写阶段，由于fulljoin、cte inline、物化视图重写、inlist2join、or转换、multi count(distinct)、magicset、lazyagg、子查询/子链接提升等重写优化，都会构造新的子查询，此时会对新构造的子查询也应用上面分配block_name的递归处理，block_name的编号计数在之前的基础上进行累加。
  - 优化器重写阶段，发生子查询提升时，内层子查询中的表被提升到外层查询中，内层子查询被消除。此时，被提升的表可能会和外层查询中的表同名，所以在表中记录其原本所属block_name来区分来自不同查询块中的两个相同表。

例如：

leading(t1 t2 t3 t4 t5)表示：t1, t2, t3, t4, t5先join，五表join顺序及内外表不限。

 版权所有 © 华为云计算技术有限公司

leading((t1 t2 t3 t4 t5))表示：t1和t2先join，t2做内表；再和t3 join，t3做内表；再和t4 join，t4做内表；再和t5 join，t5做内表。

leading(t1 (t2 t3 t4) t5)表示：t2, t3, t4先join，内外表不限；再和t1, t5 join，内外表不限。

leading((t1 (t2 t3 t4) t5))表示：t2, t3, t4先join，内外表不限；在最外层，t1再和t2, t3, t4的join表join，t1为外表，再和t5 join，t5为内表。

leading((t1 (t2 t3) t4 t5)) leading((t3 t2))表示：t2, t3先join，t2做内表；然后再和t1 join，t2, t3的join表做内表；然后再依次跟t4, t5做join，t4, t5做内表。

leading[t1 [t2 t3]]等价于leading((t1 (t2 t3))) leading((t2 t3))。

leading(t1 [t2 t3])等价于leading(t1 t2 t3) leading((t2 t3))。

leading[@sel$1 t1@sel$1 [t2@sel$2 t3@sel$2]]表示：t2、t3位于子查询中，子查询被提升后，和t1进行join，其表示先t2和t3 join，t2为外表，t3为内表；然后再和t1 join，t1为外表，t2, t3的join表做内表。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ leading((((((store_sales store) promotion) item) customer) ad2) store_returns) leading((store store_sales))*/ i_product_name product_name ...
```

该hint表示：表之间的join关系是：store_sales和store先join， store_sales做内表，然后依次跟promotion, item, customer, ad2, store_returns做join。生成计划如下所示：

```
WARNING:  Duplicated or conflict hint: Leading(store_sales store), will be discarded.
 id |                                operation                                |   E-rows   | E-memory | E-width |   E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+-------------
  1 | ->  Row Adapter                                                         |          6 |          |     273 | 16308094.34
  2 |    ->  Vector Streaming (type: GATHER)                                  |          6 |          |     273 | 16308094.34
  3 |       ->  Vector Hash Aggregate                                         |          6 | 16MB     |     273 | 16308092.67
  4 |          ->  Vector Hash Join (5,20)                                    |          6 | 585MB    |     169 | 16308092.63
  5 |             ->  Vector Streaming(type: REDISTRIBUTE)                    |    1320811 | 1MB      |     181 | 16069870.93
  6 |                ->  Vector Hash Join (7,19)                              |    1320811 | 43MB     |     181 | 16061891.00
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |    1320811 | 1MB      |     131 | 16056566.78
  8 |                      ->  Vector Hash Join (9,18)                        |    1320811 | 27MB     |     131 | 16048586.85
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)        |    1383248 | 1MB      |     131 | 16038321.62
 10 |                            ->  Vector Hash Join (11,17)                 |    1383248 | 16MB     |     131 | 16029664.50
 11 |                               ->  Vector Hash Join (12,16)              | 2626366951 | 16MB     |      73 | 15751384.88
 12 |                                  ->  Vector Hash Join (13,14)           | 2750085660 | 2156MB   |      77 | 14226077.19
 13 |                                     ->  CStore Scan on store            |      24048 | 1MB      |      19 | 2264.00
 14 |                                     ->  Vector Partition Iterator       | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 16 |                                  ->  CStore Scan on promotion           |      36000 | 1MB      |       4 | 1268.50
 17 |                               ->  CStore Scan on item                   |        158 | 1MB      |      58 | 4051.25
 18 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 19 |                         ->  CStore Scan on customer_address ad2         |    6000000 | 1MB      |      58 | 5770.00
 20 |                   ->  Vector Partition Iterator                         |  287999764 | 1MB      |      12 | 227383.99
 21 |                      ->  Partitioned CStore Scan on store_returns       |  287999764 | 1MB      |      12 | 227383.99
(21 rows)
```

图中计划顶端warning的提示详见**Hint的错误、冲突及告警**的说明。

16.4.9.3 Join 方式的 Hint

功能描述

指明Join使用的方法，可以为Nested Loop，Hash Join和Merge Join。

语法格式

```
[no] nestloop|hashjoin|mergejoin([@block_name] table_list)
```

 版权所有 © 华为云计算技术有限公司

参数说明

- **no**表示hint的join方式不使用。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table_list**为表示hint表集合的字符串，该字符串中的表与**join_table_list**相同，只是中间不允许出现括号指定join的优先级。

例如：

no nestloop(t1 t2 t3)表示：生成t1，t2，t3三表连接计划时，不使用nestloop。三表连接计划可能是t2 t3先join，再跟t1 join，或t1 t2先join，再跟t3 join。此hint只hint最后一次join的join方式，对于两表连接的方法不hint。如果需要，可以单独指定，例如：任意表均不允许nestloop连接，且希望t2 t3先join，则增加hint：no nestloop(t2 t3)。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ nestloop(store_sales store_returns item) */ i_product_name product_name ...
```

该hint表示：生成store_sales，store_returns和item三表的结果集时，最后的两表关联使用nestloop。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061693161.06
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061693161.06
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061693159.40
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061693159.36
  5 |             ->  Vector Hash Join (6,22)                                  |          6 | 43MB     |     169 | 100061693158.99
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061688591.48
  7 |                   ->  Vector Hash Join (8,21)                            |          6 | 16MB     |     119 | 100061688591.30
  8 |                      ->  Vector Hash Join (9,20)                         |          7 | 27MB     |     123 | 100061687304.04
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061677823.27
 10 |                            ->  Vector Hash Join (11,19)                  |          7 | 16MB     |     123 | 100061677823.12
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061675546.57
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB | 12 | 227383.99
 17 |                                  ->  Vector Materialize                  |        158 | 16MB     |      58 | 4051.28
 18 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 19 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 20 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 21 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 22 |                      ->  CStore Scan on customer_address ad2             |    6000000 | 1MB      |      58 | 5770.00
(22 rows)
```

16.4.9.4 行数的 Hint

功能描述

指明中间结果集的大小，支持绝对值和相对值的hint。

语法格式

```
rows([@block_name] table_list #|+|-|* const)
```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **#,+,-,***，进行行数估算hint的四种操作符号。#表示直接使用后面的行数进行hint。+,-,*表示对原来估算的行数进行加、减、乘操作，运算后的行数最小值为1

版权所有 © 华为云计算技术有限公司

行。table_list为hint对应的单表或多表join结果集，与**Join方式的Hint**中table_list相同。

- **const**可以是任意非负数，支持科学计数法。

例如：

rows(t1 #5)表示：指定t1表的结果集为5行。

rows(t1 t2 t3 *1000)表示：指定t1, t2, t3 join完的结果集的行数乘以1000。

建议

- 推荐使用两个表*的hint。对于两个表的采用*操作符的hint，只要两个表出现在join的两端，都会触发hint。例如：设置hint为rows(t1 t2 * 3)，对于(t1 t3 t4)和(t2 t5 t6)join时，由于t1和t2出现在join的两端，所以其join的结果集也会应用该hint规则乘以3。
- rows hint支持在单表、多表、function table及subquery scan table的结果集上指定hint。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ rows(store_sales store_returns *50) */ i_product_name product_name ...
```

该hint表示：store_sales，store_returns关联的结果集估算行数在原估算行数基础上乘以50。生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |        312 |          |     273 | 3401656.58
  2 |    ->  Vector Streaming (type: GATHER)                                  |        312 |          |     273 | 3401656.58
  3 |       ->  Vector Hash Aggregate                                         |        312 | 16MB     |     273 | 3401634.91
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |        313 | 1MB      |     169 | 3401634.39
  5 |             ->  Vector Hash Join (6,21)                                 |        313 | 43MB     |     169 | 3401633.06
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                 |        313 | 1MB      |     119 | 3397065.38
  7 |                   ->  Vector Hash Join (8,20)                           |        313 | 27MB     |     119 | 3397064.31
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)           |        328 | 1MB      |     119 | 3387583.37
  9 |                         ->  Vector Hash Join (10,19)                    |        328 | 16MB     |     119 | 3387582.18
 10 |                            ->  Vector Hash Join (11,18)                 |        344 | 16MB     |     123 | 3386294.74
 11 |                               ->  Vector Hash Join (12,14)              |        360 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator          |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |  12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)           |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator       | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 17 |                                     ->  CStore Scan on item             |        158 | 1MB      |      58 | 4051.25
 18 |                                ->  CStore Scan on store                 |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on promotion                 |      36000 | 1MB      |       4 | 1268.50
 20 |                      ->  CStore Scan on customer                        |   12000000 | 1MB      |       8 | 12923.00
 21 |                ->  CStore Scan on customer_address ad2                  |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

第11行算子的估算行数修正为360行，原估算行数为7行（四舍五入后取值）。

16.4.9.5 Stream 方式的 Hint

功能描述

指明stream使用的方法，可以为broadcast和redistribute以及指定AGG重分布的分布键。

说明

指定Agg重分布列Hint，仅8.1.3.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

语法格式

```
[no] broadcast | redistribute([@block_name] table_list) | redistribute ([@block_name] (*) (columns))
```

参数说明

- **no**表示hint的stream方式不使用，当指定AGG分布列的hint时指定no关键字无效。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table_list**为进行stream操作的单表或多表join结果集，见**参数说明**。
- 当指定分布列的hint时，*为固定写法，不支持指定表名。
- **columns**指定group by子句中的一个或者多个列，没有group by子句也可以指定distinct子句中的列。

  说明

  - 指定的分布列，需要用group by或distinct中的列序号或列名来表示，count（distinct）中的列只能通过列名指定。
  - 对于多层的查询，可以在每层指定对应层的分布列hint，只在当前层生效。
  - 指定的count(distinct)列仅针对生成双层hashagg的计划时才生效，否则指定的分布列无效。
  - 指定了分布列，如果优化器估算后发现不需要重分布，则指定的分布列无效。

建议

- 通常优化器会根据统计信息选择一组不倾斜的分布键进行数据重分布。当默认选择的分布键有倾斜时，可以手动指定重分布的列，避免数据倾斜。
- 在选择分布键的时候，通常要根据数据分布特征选取一组distinct值比较高的列作为分布列，这样可以保证重分布后，数据均匀的分布到各个DN。
- 在编写好hint后，可以通过explain verbose+SQL打印执行计划，查看指定的分布键是否有效，如果指定的分布键无效会有warning提示。

示例

- 对**示例**中原语句使用如下hint：

```
explain
select /*+ no redistribute(store_sales store_returns item store) leading(((store_sales store_returns item store) customer)) */ i_product_name product_name ...
```

原计划中，(store_sales store_returns item store)和customer做join时，前者做了重分布，此hint表示禁止前者混合表做重分布，但仍然保持join顺序，则生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |          6 |          |     273 | 5718448.94
  2 |    ->  Vector Streaming (type: GATHER)                                  |          6 |          |     273 | 5718448.94
  3 |       ->  Vector Hash Aggregate                                         |          6 | 16MB     |     273 | 5718447.27
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |          6 | 1MB      |     169 | 5718447.23
  5 |             ->  Vector Hash Join (6,21)                                 |          6 | 16MB     |     169 | 5718446.86
  6 |                ->  Vector Hash Join (7,20)                              |          7 | 43MB     |     173 | 5717159.60
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |          7 | 1MB      |     123 | 5712592.09
  8 |                      ->  Vector Hash Join (9,18)                        |          7 | 585MB    |     123 | 5712591.93
  9 |                         ->  Vector Hash Join (10,17)                    |          7 | 16MB     |     123 | 3386294.56
 10 |                            ->  Vector Hash Join (11,13)                 |          7 | 19MB     |     112 | 3384018.02
 11 |                               ->  Vector Partition Iterator             |  287999764 | 1MB      |      12 | 227383.99
 12 |                                  ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      12 | 227383.99
 13 |                               ->  Vector Hash Join (14,16)              |    1516824 | 16MB     |     124 | 3065686.08
 14 |                                  ->  Vector Partition Iterator          | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                     ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 16 |                                  ->  CStore Scan on item                |        158 | 1MB      |      58 | 4051.25
 17 |                            ->  CStore Scan on store                     |      24048 | 1MB      |      19 | 2264.00
 18 |                         ->  Vector Streaming(type: BROADCAST)           |  288000000 | 1MB      |       8 | 2176297.36
 19 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 20 |                      ->  CStore Scan on customer_address ad2            |    6000000 | 1MB      |      58 | 5770.00
 21 |                   ->  CStore Scan on promotion                          |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```


版权所有 © 华为云计算技术有限公司

- 指定Agg重分布的分布列。

```
explain (verbose on, costs off, nodes off)
select /*+ redistribute ((*) (2 3)) */ a1, b1, c1, count(c1)  from t1 group by a1, b1, c1 having
count(c1) > 10 and sum(d1) > 100
```

通过下面的示例可以看到指定的group by列的后两列作为分布键。

```
                           QUERY PLAN
-----------------------------------------------------------------
  id |                 operation
 ----+-------------------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |     ->  HashAggregate
   3 |        ->  Streaming(type: REDISTRIBUTE)
   4 |           ->  Seq Scan on public.t1

         Predicate Information (identified by plan id)
 ---------------------------------------------------------------
   2 --HashAggregate
         Filter: ((count(t1.c1) > 10) AND (sum(t1.d1) > 100))

 Targetlist Information (identified by plan id)
 -----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a1, b1, c1, (count(c1))
   2 --HashAggregate
         Output: a1, b1, c1, count(c1)
         Group By Key: t1.a1, t1.b1, t1.c1
   3 --Streaming(type: REDISTRIBUTE)
         Output: a1, b1, c1, d1
         Distribute Key: b1, c1
   4 --Seq Scan on public.t1
         Output: a1, b1, c1, d1

    ====== Query Summary =====
 ---------------------------------
 System available mem: 24862720KB
 Query Max mem: 24862720KB
 Query estimated mem: 3138KB
(30 rows)
```

- 当语句中不包含group by子句时，指定distinct列作为重分布列。

```
explain (verbose on, costs off, nodes off)
select /*+ redistribute ((*) (3 1)) */ distinct a1, b1, c1 from t1;
```

 版权所有 © 华为云计算技术有限公司

```
                    QUERY PLAN
-------------------------------------------------
 id |                 operation
----+-----------------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |     ->  HashAggregate
  3 |        ->  Streaming(type: REDISTRIBUTE)
  4 |           ->  Seq Scan on public.t1

Targetlist Information (identified by plan id)
-------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a1, b1, c1
  2 --HashAggregate
        Output: a1, b1, c1
        Group By Key: t1.a1, t1.b1, t1.c1
  3 --Streaming(type: REDISTRIBUTE)
        Output: a1, b1, c1
        Distribute Key: c1, a1
  4 --Seq Scan on public.t1
        Output: a1, b1, c1

   ====== Query Summary =====
---------------------------------
System available mem: 24862720KB
Query Max mem: 24862720KB
Query estimated mem: 3136KB
(25 rows)
```

16.4.9.6 Scan 方式的 Hint

功能描述

指明scan使用的方法，可以是tablescan、indexscan和indexonlyscan。

语法格式

```
[no] tablescan|indexscan|indexonlyscan([@block_name] table [index])
```

参数说明

- **no**表示hint的scan方式不使用。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示hint指定的表，只能指定一个表，如果表存在别名应优先使用别名进行hint。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

- **index**表示使用indexscan或indexonlyscan的hint时，指定的索引名称，当前只能指定一个。

 版权所有 © 华为云计算技术有限公司

说明

对于indexscan或indexonlyscan，只有hint的索引属于hint的表时，才能使用该hint。

scan hint支持在行列存表、hdfs内外表、obs表、子查询表上指定。对于hdfs内表，由主表和delta表组成，delta表对用户不可见，故hint仅作用在主表上。

指定indexscan可生效indexscan或indexonlyscan，indexscan或indexonlyscan也可同时出现。当indexscan和indexonlyscan hint同时出现，优先生效indexonlyscan。

示例

为了hint使用索引扫描，需要首先在表item的i_item_sk列上创建索引，名称为i：

```
create index i on item(i_item_sk);
```

对示例中原语句使用如下hint：

```
explain
select /*+ indexscan(item i) */ i_product_name product_name ...
```

该hint表示：item表使用索引i进行扫描。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061674938.26
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061674938.26
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061674936.59
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061674936.55
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 43MB     |     169 | 100061674936.19
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061670368.67
  7 |                   ->  Vector Hash Join (8,20)                            |          6 | 16MB     |     119 | 100061670368.50
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 100061669081.23
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061659600.47
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 100061659600.31
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061657323.77
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns |  287999764 | 1MB |      12 | 227383.99
 17 |                                  ->  CStore Index Scan using i on item   |          1 | 1MB      |      58 | 4.01
 18 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 21 |                   ->  CStore Scan on customer_address ad2                |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

16.4.9.7 子链接块名的 hint

功能描述

指明子链接块的名称。

语法格式

```
blockname ([@block_name] table)
```

注意事项

- block_name hint仅在对应的子链接块提升时才会被上层查询使用。目前支持的子链接提升包括IN子链接提升、EXISTS子链接提升和包含Agg等值相关子链接提升。该hint通常会和前面章节提到的hint联合使用。
- 对于FROM关键字后的子查询，则需要使用子查询的别名进行hint，block_name hint不会被用到。
- 如果子链接中含有多个表，则提升后这些表可与外层表以任意优化顺序连接，hint也不会被用到。

 版权所有 © 华为云计算技术有限公司

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示为该子链接块hint的别名的名称。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

示例

```
explain select /*+nestloop(store_sales tt) */ * from store_sales where ss_item_sk in (select /*
+blockname(tt)*/ i_item_sk from item group by 1);
```

该hint表示：子链接的别名为tt，提升后与上层的store_sales表关联时使用nestloop。生成计划如下所示：

```
 id |                       operation                       |   E-rows   | E-memory | E-width |     E-costs
----+-------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                       | 1439994000 |          |     216 | 325105765847.91
  2 |    ->  Vector Streaming (type: GATHER)                | 1439994000 |          |     216 | 325105765847.91
  3 |       ->  Vector Nest Loop Semi Join (4, 6)           | 1439994000 | 1MB      |     216 | 325026664615.00
  4 |          ->  Vector Partition Iterator                | 2879987999 | 1MB      |     216 | 2756066.50
  5 |             ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB      |     216 | 2756066.50
  6 |          ->  Vector Materialize                       |     300000 | 16MB     |       4 | 4176.25
  7 |             ->  Vector Hash Aggregate                 |     300000 | 16MB     |       4 | 3988.75
  8 |                ->  CStore Scan on item                |     300000 | 1MB      |       4 | 3832.50
(8 rows)
```

16.4.9.8 运行倾斜的 hint

功能描述

指明查询运行时重分布过程中存在倾斜的重分布键和倾斜值，针对Join和HashAgg运算中的重分布进行优化。

注意事项

- skew hint仅在需要重分布且指定的倾斜信息与查询执行过程中的重分布信息相匹配时才会被使用。
- skew hint受GUC参数**skew_option**限制，如果参数处于关闭状态，则无法进行skew hint倾斜调优。
- skew hint目前仅处理普通表和子查询类型的表关系，支持基表hint、子查询hint、with as子句hint。对于子查询，无论提升与否都支持在skew hint中使用，这点与其它hint不一样。
- 对于倾斜表，如果定义了别名，则在hint中必须使用别名。
- 对于倾斜列，在不产生歧义的情况下，可以使用原名也可以使用别名。skew hint的column不支持表达式，如果需要指定采用分布键为表达式的重分布存在倾斜，需要将重分布键指定为新的列，以新的列进行hint。
- 对于倾斜值，个数需为列数的整数倍并按列的顺序进行组合，组合的个数不能超过10个。如果各倾斜列的倾斜值的个数不一样，为了满足按列组合，值可以重复指定。如，表t1的c1和c2存在倾斜，c1列的倾斜值只有a1，而c2列的倾斜有b1和

 版权所有 © 华为云计算技术有限公司

b2，则skew hint如下：skew(t1 (c1 c2) ((a1 b1)(a1 b2)))。例中(a1 b1)为一个值组合，NULL可以作为倾斜值出现，每个hint中的值组合不超过十个， 且需为列的整数倍。

- 在Join的重分布优化中，skew hint中的value不可缺省，在HashAgg中可以缺省。
- 对于表、列、值中若指定多个，则同类间需以空格分离。
- 对于倾斜值，不支持在hint中进行类型强转；对于string类型，需要使用单引号。

语法格式

- 指定单表倾斜：
  ```
  skew([@block_name] table (column) [(value)])
  ```
- 指定中间结果倾斜：
  ```
  skew([@block_name] (join_rel) (column) [(value)])
  ```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示存在倾斜的单个表名。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。
- **join_rel**表示参与join的两个或多个表，如（t1 t2）表示t1和t2join后的结果存在倾斜。
- **column**表示倾斜表中存在倾斜的一个或多个列。
- **value**表示倾斜的列中存在倾斜的一个或多个值。

例如：

- 指定单表倾斜

  每一个skew hint用来表示一个表关系存在的倾斜信息，如果想要指定在查询中的多个表关系存在的倾斜信息，则通过指定多个skew hint实现。

  在指定skew时，包括以下四个场景的用法：

  - 单列单值： skew(t (c1) (v1))

    说明：表关系t的c1列中的v1值在查询执行中存在倾斜。
  - 单列多值：skew(t (c1) (v1 v2 v3 ...))

    说明：表关系t的c1列中的v1、v2、v3…等值在查询执行中存在倾斜。
  - 多列单值：skew(t (c1 c2) (v1 v2))

    说明：表关系t的c1列的v1值和c2列的v2值在查询执行中存在倾斜。
  - 多列多值：skew(t (c1 c2) ((v1 v2) (v3 v4) (v5 v6) ...))

    说明：表关系t的c1列的v1、v3、v5…值和c2列的v2、v4、v6…值在查询执行中存在倾斜。

 版权所有 © 华为云计算技术有限公司

须知

多列多值时，各组倾斜值间也可以不使用括号，如：skew(t (c1 c2) (v1 v2 v3 v4 v5 v6 ...))。是否使用括号必须统一，不可混合，

如：skew(t (c1 c2) (v1 v2 v3 v4 （ v5 v6 ） ...)) 将会产生语法报错。

- 指定中间结果倾斜

  如果基表不存在倾斜，而是查询执行中的中间结果出现倾斜，则需要通过指定中间结果倾斜的skew hint来进行倾斜的调优。skew((t1 t2) (c1) (v1))

  说明：表关系t1和t2 Join后的结果存在倾斜，倾斜的是t1表的c1列，c1列的倾斜值是v1。

  为了避免产生歧义，“c1”只能存在于join_rel的一个表关系中，如果存在同名列则通过别名进行规避。

建议

- 如果查询具有多层，则哪一层出现倾斜，则将hint写在哪一层中。
- 对于提升的子查询，skew hint支持直接使用子查询名进行hint。如果明确子查询提升后的哪一个基表存在倾斜，则直接使用基表进行hint的可用性更高。
- 无论对于表或列，若存在别名，则优先使用别名进行hint。

示例

指定单表倾斜

- 原query中进行hint。

  采用如下查询进行skew hint倾斜调优的举例，查询语句及不带hint的原计划如下所示：

```
explain
with customer_total_return as
(select sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```

 版权所有 © 华为云计算技术有限公司

```
 id |                                operation                                |  E-rows  |   E-memory   | E-width |  E-costs
----+-------------------------------------------------------------------------+----------+--------------+---------+-----------
  1 | ->  Row Adapter                                                         |      100 |              |      20 | 911254.47
  2 |    ->  Vector Limit                                                     |      100 |              |      20 | 911254.47
  3 |       ->  Vector Streaming (type: GATHER)                               |     2400 |              |      20 | 911325.75
  4 |          ->  Vector Limit                                               |     2400 | 1MB          |      20 | 911247.62
  5 |             ->  Vector Sort                                             |  3684816 | 16MB         |      20 | 911631.21
  6 |                ->  Vector Hash Join (7,29)                              |  3684817 | 41MB(12374MB)|      20 | 905379.41
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |  3684817 | 384KB        |       4 | 883010.31
  8 |                      ->  Vector Hash Join (9,19)                        |  3684817 | 16MB         |       4 | 861302.05
  9 |                         ->  Vector Hash Join (10,18)                    | 11054450 | 16MB         |      44 | 427109.71
 10 |                            ->  Vector Hash Aggregate                    | 50247501 | 397MB(12671MB)|      54 | 395302.57
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)  | 50247501 | 384KB        |      22 | 358663.76
 12 |                                  ->  Vector Hash Join (13,15)           | 50247501 | 16MB         |      22 | 294300.51
 13 |                                     ->  Vector Partition Iterator       |287999764 | 1MB          |      26 | 227383.99
 14 |                                        ->  Partitioned CStore Scan on store_returns |287999764 | 1MB |      26 | 227383.99
 15 |                                     ->  Vector Streaming(type: BROADCAST)|     8712 | 384KB        |       4 | 975.56
 16 |                                        ->  Vector Partition Iterator    |      363 | 1MB          |       4 | 910.65
 17 |                                           ->  Partitioned CStore Scan on date_dim |      363 | 1MB |       4 | 910.65
 18 |                               ->  CStore Scan on store                  |       44 | 1MB          |       4 | 1006.39
 19 |                            ->  Vector Hash Aggregate                    |      192 | 16MB         |      68 | 426707.38
 20 |                               ->  Vector Subquery Scan on ctr2          | 50247501 | 1MB          |      36 | 416239.03
 21 |                                  ->  Vector Hash Aggregate              | 50247501 | 397MB(12671MB)|      54 | 395302.57
 22 |                                     ->  Vector Streaming(type: REDISTRIBUTE)| 50247501 | 384KB |      22 | 358663.76
 23 |                                        ->  Vector Hash Join (24,26)     | 50247501 | 16MB         |      22 | 294300.51
 24 |                                           ->  Vector Partition Iterator |287999764 | 1MB          |      26 | 227383.99
 25 |                                              ->  Partitioned CStore Scan on store_returns |287999764 | 1MB |      26 | 227383.99
 26 |                                           ->  Vector Streaming(type: BROADCAST)|     8712 | 384KB |       4 | 975.56
 27 |                                              ->  Vector Partition Iterator|      363 | 1MB          |       4 | 910.65
 28 |                                                 ->  Partitioned CStore Scan on date_dim |      363 | 1MB |       4 | 910.65
 29 |                   ->  CStore Scan on customer                           | 12000000 | 1MB          |      24 | 12923.00
(29 rows)
```

对内层with子句中的HashAgg和外层的Hash Join进行hint指定，带hint的查询如下：

```
explain
with customer_total_return as
(select /*+ skew(store_returns(sr_store_sk sr_customer_sk)) */sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  /*+ skew(ctr1(ctr_customer_sk)(11))*/  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```

该hint表示：内层with子句中的group by在做HashAgg中进行重分布时存在倾斜，对应原计划的10和21号Hash Agg算子；外层ctr1表的ctr_customer_sk列在做Hash Join中进行重分布时存在倾斜，对应原计划的6号算子。生成计划如下所示：

```
 id |                                operation                                |  E-rows  |   E-memory   | E-width |  E-costs
----+-------------------------------------------------------------------------+----------+--------------+---------+-----------
  1 | ->  Row Adapter                                                         |      100 |              |      20 | 1061778.14
  2 |    ->  Vector Limit                                                     |      100 |              |      20 | 1061778.14
  3 |       ->  Vector Streaming (type: GATHER)                               |     2400 |              |      20 | 1061849.41
  4 |          ->  Vector Limit                                               |     2400 | 1MB          |      20 | 1061771.29
  5 |             ->  Vector Sort                                             |  3684816 | 16MB         |      20 | 1062154.87
  6 |                ->  Vector Hash Join (7,31)                              |  3684817 | 41MB(12344MB)|      20 | 1055903.08
  7 |                   ->  Vector Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)|  3684817 | 384KB |       4 | 1013056.49
  8 |                      ->  Vector Hash Join (9,20)                        |  3684817 | 16MB         |       4 | 1000006.10
  9 |                         ->  Vector Hash Join (10,19)                    | 11054450 | 16MB         |      44 | 496461.73
 10 |                            ->  Vector Hash Aggregate                    | 50247501 | 397MB(12010MB)|      54 | 464654.59
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)  | 50247501 | 384KB        |      54 | 428015.79
 12 |                                  ->  Vector Hash Aggregate              | 50247501 | 397MB(12010MB)|      54 | 330939.31
 13 |                                     ->  Vector Hash Join (14,16)        | 50247501 | 16MB         |      22 | 294300.51
 14 |                                        ->  Vector Partition Iterator    |287999764 | 1MB          |      26 | 227383.99
 15 |                                           ->  Partitioned CStore Scan on store_returns |287999764 | 1MB |      26 | 227383.99
 16 |                                        ->  Vector Streaming(type: BROADCAST)|     8712 | 384KB |       4 | 975.56
 17 |                                           ->  Vector Partition Iterator |      363 | 1MB          |       4 | 910.65
 18 |                                              ->  Partitioned CStore Scan on date_dim |      363 | 1MB |       4 | 910.65
 19 |                               ->  CStore Scan on store                  |       44 | 1MB          |       4 | 1006.39
 20 |                            ->  Vector Hash Aggregate                    |      192 | 16MB         |      68 | 496059.40
 21 |                               ->  Vector Subquery Scan on ctr2          | 50247501 | 1MB          |      36 | 485591.05
 22 |                                  ->  Vector Hash Aggregate              | 50247501 | 397MB(12010MB)|      54 | 464654.59
 23 |                                     ->  Vector Streaming(type: REDISTRIBUTE)| 50247501 | 384KB |      54 | 428015.79
 24 |                                        ->  Vector Hash Aggregate        | 50247501 | 397MB(12010MB)|      54 | 330939.31
 25 |                                           ->  Vector Hash Join (26,28)  | 50247501 | 16MB         |      22 | 294300.51
 26 |                                              ->  Vector Partition Iterator|287999764 | 1MB          |      26 | 227383.99
 27 |                                                 ->  Partitioned CStore Scan on store_returns |287999764 | 1MB |      26 | 227383.99
 28 |                                              ->  Vector Streaming(type: BROADCAST)|     8712 | 384KB |       4 | 975.56
 29 |                                                 ->  Vector Partition Iterator|      363 | 1MB          |       4 | 910.65
 30 |                                                    ->  Partitioned CStore Scan on date_dim |      363 | 1MB |       4 | 910.65
 31 |                   ->  Vector Streaming(type: PART LOCAL PART BROADCAST)  | 12000000 | 384KB        |      24 | 34485.50
 32 |                      ->  CStore Scan on customer                        | 12000000 | 1MB          |      24 | 12923.00
(32 rows)
```


版权所有 © 华为云计算技术有限公司

从优化后的计划可以看出：①对于Hash Agg，由于其重分布存在倾斜，所以优化为双层Agg；②对于Hash Join，同样由于其重分布存在倾斜，所以优化为采用新的重分布算子。

- 需要改写query后进行hint

  不带hint的查询和计划如下：

```
explain select count(*) from store_sales_1 group by round(ss_list_price);
```

```
 id |                                operation                                 |  E-rows  | E-memory | E-width | E-costs
----+--------------------------------------------------------------------------+----------+----------+---------+----------
  1 | ->  Row Adapter                                                          |    16672 |          |      14 | 62261.28
  2 |    ->  Vector Streaming (type: GATHER)                                   |    16672 |          |      14 | 62261.28
  3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)                  |    16672 | 32KB     |      14 | 61479.78
  4 |          ->  Vector Hash Aggregate                                       |    16672 | 16MB     |      14 | 61452.00
  5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2)      | 3112836 | 128KB    |       6 | 57498.43
  6 |                ->  CStore Scan on store_sales_1                          | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

  由于hint中列不支持表达式，在进行倾斜优化时需要借助subquery改写查询，改写后的查询和计划如下：

```
explain
select count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
 id |                                operation                                 |  E-rows  | E-memory | E-width | E-costs
----+--------------------------------------------------------------------------+----------+----------+---------+----------
  1 | ->  Row Adapter                                                          |    16672 |          |      14 | 62261.28
  2 |    ->  Vector Streaming (type: GATHER)                                   |    16672 |          |      14 | 62261.28
  3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)                  |    16672 | 32KB     |      14 | 61479.78
  4 |          ->  Vector Hash Aggregate                                       |    16672 | 16MB     |      14 | 61452.00
  5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2)      | 3112836 | 128KB    |       6 | 57498.43
  6 |                ->  CStore Scan on store_sales_1                          | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

  改写注意不要影响到业务逻辑。

  采用改写后的查询进行hint，带hint的查询和计划如下：

```
explain
select /*+ skew(tmp(a)) */ count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
 id |                                operation                                 |  E-rows  | E-memory | E-width | E-costs
----+--------------------------------------------------------------------------+----------+----------+---------+----------
  1 | ->  Row Adapter                                                          |    16672 |          |      14 | 27771.82
  2 |    ->  Vector Streaming (type: GATHER)                                   |    16672 |          |      14 | 27771.82
  3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)                  |    16672 | 32KB     |      14 | 26990.32
  4 |          ->  Vector Hash Aggregate                                       |    16671 | 16MB     |      14 | 26962.54
  5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2)      |    66216 | 128KB    |      14 | 26838.09
  6 |                ->  Vector Hash Aggregate                                 |    66216 | 16MB     |      14 | 25949.61
  7 |                   ->  CStore Scan on store_sales_1                       | 3112836 | 1MB      |       6 | 21810.25
(7 rows)
```

  从计划可以看出，对Hash Agg进行倾斜优化后，采用了双层agg实现，大大过滤了进行重分布时的数据量，减少了重分布时间。

  此外，需要说明的是，对于子查询，支持使用查询内部的列进行hint，如：

```
explain
select /*+ skew(tmp(b)) */ count(*)
from (select round(ss_list_price) b,ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

16.4.9.9 指定子查询不提升的 hint

功能描述

优化器在对查询进行逻辑优化时通常会将可以提升的子查询提升到上层以避免嵌套执行，但对于某些场景，嵌套执行不会导致性能下降过多，而提升之后扩大了查询路径的搜索范围，可能导致性能变差。对于此类情况，可以使用no merge hint指定子查询不提升进行调试。大多数情况下不建议使用此hint。

版权所有 © 华为云计算技术有限公司

语法格式

```
no merge[@block_name]
no merge ([@block_name1] subquery_name[@block_name2])
```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **subquery_name**为目标子查询名，亦可以是view或cte名，表示该子查询在逻辑优化时不会进行提升；当不指定subquery_name时，表示当前查询不提升。

示例

创建表t1、t2、t3：

```
create table t1(a1 int,b1 int,c1 int,d1 int);
create table t2(a2 int,b2 int,c2 int,d2 int);
create table t3(a3 int,b3 int,c3 int,d3 int);
```

原语句为：

```
explain select * from t3, (select a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
```

```
 id |           operation           | E-rows | E-width | E-costs
----+-------------------------------+--------+---------+---------
  1 | ->  Hash Join (2,6)           |  44450 |      32 | 754.31
  2 |    ->  Hash Join (3,4)        |   8885 |      28 | 182.11
  3 |       ->  Seq Scan on t3      |   1776 |      16 | 27.76
  4 |       ->  Hash                |   1776 |      12 | 27.76
  5 |          ->  Seq Scan on t2   |   1776 |      12 | 27.76
  6 |    ->  Hash                   |   1776 |       8 | 27.76
  7 |       ->  Seq Scan on t1      |   1776 |       8 | 27.76
```

上述查询中，可以使用以下两种方式禁止子查询s1进行提升：

- 方式一：
  ```
  explain select /*+ no merge(s1) */ * from t3, (select a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
  ```
- 方式二：
  ```
  explain select * from t3, (select /*+ no merge */ a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
  ```

提升后效果：

```
 id |           operation           | E-rows | E-width | E-costs
----+-------------------------------+--------+---------+---------
  1 | ->  Hash Join (2,6)           |   8880 |      32 | 443.03
  2 |    ->  Hash Join (3,4)        |   8885 |      16 | 182.11
  3 |       ->  Seq Scan on t1      |   1776 |       8 | 27.76
  4 |       ->  Hash                |   1776 |      12 | 27.76
  5 |          ->  Seq Scan on t2   |   1776 |      12 | 27.76
  6 |    ->  Hash                   |   1776 |      16 | 27.76
  7 |       ->  Seq Scan on t3      |   1776 |      16 | 27.76
```


版权所有 © 华为云计算技术有限公司

16.4.9.10 Drive 方式的 Hint

功能描述

优化器在进行查询计划生成时，会使用动态规划或遗传基因算法枚举可能的Join路径，并在其中选择最优的路径，当连接表的数目增大时，会导致搜索空间的过度膨胀。对于此类情况，可以通过Drive Hint指定查询中的事实表，使用启发式搜索的方式减少路径搜索空间的范围。该hint只有当GUC参数**join_search_mode**设置为heuristic时生效。

语法格式

```
[no] drive (table)
```

参数说明

- **no**表示hint对应的表为非Drive表。
- **table**表示hint指定的表，只能指定一个表，如果表存在别名应优先使用别名进行hint。

示例

创建表t1、t2、t3：

```
create table t1(a1 int,b1 int,c1 int,d1 int);
create table t2(a2 int,b2 int,c2 int,d2 int);
create table t3(a3 int,b3 int,c3 int,d3 int);
```

原语句为：

```
explain select * from t1,t2,t3 where t1.b1=t2.b2 and t1.c1=t3.c3 and t2.d2=t3.d3;
```

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      4 |          |      48 | 40.03
  2 |    ->  Hash Join (3,9)                    |      4 | 1MB      |      48 | 34.03
  3 |       ->  Streaming(type: BROADCAST)      |     40 | 2MB      |      32 | 23.64
  4 |          ->  Hash Join (5,7)              |     20 | 1MB      |      32 | 22.06
  5 |             ->  Streaming(type: BROADCAST) |     40 | 2MB      |      16 | 11.67
  6 |                ->  Seq Scan on t1         |     20 | 1MB      |      16 | 10.10
  7 |             ->  Hash                      |     20 | 16MB     |      16 | 10.10
  8 |                ->  Seq Scan on t2         |     20 | 1MB      |      16 | 10.10
  9 |       ->  Hash                            |     20 | 16MB     |      16 | 10.10
 10 |          ->  Seq Scan on t3               |     20 | 1MB      |      16 | 10.10
```

上述查询中，可以通过指定t3表为drive表，使得t3表优先和其它表join，减少计划生成路径搜索，同时更改join顺序。：

```
explain select /*+ drive(t3) */ * from t1, t2, t3 where t1.b1=t2.b2 and t1.c1=t3.c3 and t2.d2=t3.d3;
```

使用hint后的效果：

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      4 |          |      48 | 40.03
  2 |    ->  Hash Join (3,9)                    |      4 | 1MB      |      48 | 34.03
  3 |       ->  Streaming(type: BROADCAST)      |     40 | 2MB      |      32 | 23.64
  4 |          ->  Hash Join (5,7)              |     20 | 1MB      |      32 | 22.06
  5 |             ->  Streaming(type: BROADCAST) |     40 | 2MB      |      16 | 11.67
  6 |                ->  Seq Scan on t3(DRIVE)  |     20 | 1MB      |      16 | 10.10
  7 |             ->  Hash                      |     20 | 16MB     |      16 | 10.10
  8 |                ->  Seq Scan on t1         |     20 | 1MB      |      16 | 10.10
  9 |       ->  Hash                            |     20 | 16MB     |      16 | 10.10
 10 |          ->  Seq Scan on t2               |     20 | 1MB      |      16 | 10.10
```


版权所有 © 华为云计算技术有限公司

> **⚠ 注意**
>
> 当GUC参数**join_search_mode**取值为heuristic且连接表个数超过from_collapse_limit时，优化器会自动识别Drive表，并在计划中显示。如果识别优化器最终选择的Drive不对，可以使用no drive hint进行纠偏。

16.4.9.11 字典编码的 hint

功能描述

通过设置指定列来构建字典编码，将字典编码的字符串的比较转化为数字的比较，加快group by、filter等查询速度。该hint仅8.3.0及以上集群版本支持。

注意事项

- 目前只支持新版的hstore表（表级参数为enable_hstore_opt为on）。

语法格式

```
/* + ( no ) dict(table (column)) */
```

参数说明

- dict(table (column))
  使用字典编码表的某列。
- no dict(table (column))
  禁用字典编码表的某列。

示例

```
SELECT /*+ dict (bitmaptbl_high (server_ip)) */ distinct(server_ip) FROM bitmaptbl_high WHERE
scope_name='saetataetaeta' ORDER BY server_ip;
```

生成的计划如下，server_ip使用了字典编码：

版权所有 © 华为云计算技术有限公司

```
postgres=# explain (verbose on, costs off, nodes off)
select distinct(server_ip) from bitmaptbl_high where scope_name='saetataetaeta' order by server_ip;
                                  QUERY PLAN
------------------------------------------------------------------------------------
 id |                       operation
----+-----------------------------------------------------
  1 | ->  Row Adapter
  2 |    ->  Vector Sort
  3 |       ->  Vector Sonic Hash Aggregate
  4 |          ->  Vector Streaming (type: GATHER)
  5 |             ->  Vector Sonic Hash Aggregate
  6 |                ->  CStore Scan on public.bitmaptbl_high

                   Predicate Information (identified by plan id)
------------------------------------------------------------------------------------
  6 --CStore Scan on public.bitmaptbl_high
       CU Predicate Filter: (bitmaptbl_high.scope_name = 'saetataetaeta'::text)
       Pushdown Predicate Filter: (bitmaptbl_high.scope_name = 'saetataetaeta'::text)

 Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Row Adapter
       Output: server_ip
  2 --Vector Sort
       Output: server_ip
       Sort Key: bitmaptbl_high.server_ip
  3 --Vector Sonic Hash Aggregate
       Output: server_ip
       Group By Key: bitmaptbl_high.server_ip
  4 --Vector Streaming (type: GATHER)
       Output: server_ip
  5 --Vector Sonic Hash Aggregate
       Output: server_ip
       Group By Key: bitmaptbl_high.server_ip
       Dict Optimized: bitmaptbl_high.server_ip
       Dict Decoded: bitmaptbl_high.server_ip
  6 --CStore Scan on public.bitmaptbl_high
       Output: server_ip
       Dict Encoded: server_ip
       Distribute Key: scope_name
```

使用no dict可以禁用server_ip使用字典编码：

```
SELECT /*+ no dict (bitmaptbl_high (server_ip)) */ distinct(server_ip) FROM bitmaptbl_high WHERE
scope_name='saetataetaeta' ORDER BY server_ip;
```

16.4.9.12 配置参数的 hint

功能描述

指明计划生成时配置参数的值，又称作guc hint。

注意事项

- 如果hint设置的配置参数在语句级别生效，则该hint必须写在顶层查询中，而不能写在子查询中。对于UNION、INTERSECT、EXCEPT和MINUS语句，可以将在语句级别的guc hint写在参与集合运算的任意一个SELECT子句上，该guc hint设置的配置参数会在参与集合运算的每个SELECT子句上生效。
- 子查询提升时，该子查询上的所有guc hint会被丢弃。
- 如果一个配置参数既被语句级别的guc hint设置，又被子查询级别的guc hint设置，则子查询级别的guc hint在对应的子查询中生效，语句级别的guc hint在语句的其它子查询中生效。

语法格式

```
set [global]([@block_name] guc_name guc_value)
```

 版权所有 © 华为云计算技术有限公司

参数说明

- **global**表示hint设置的配置参数在语句级别生效，不加global表示hint设置的配置参数在子查询级别生效，即仅在hint所在的子查询中生效，在该语句的其它子查询中不生效。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **guc_name**表示hint指定的配置参数的名称。
- **guc_value**表示hint指定的配置参数的值。

guc hint当前仅支持部分配置参数，并且有些配置参数不支持在子查询级别设置，只能在语句级别设置，以下为支持的参数列表：

表 16-19 guc hint 支持的配置参数

配置参数名	是否支持在子查询级别设置
agg_max_mem	是
agg_redistribute_enhancement	是
best_agg_plan	是
cost_model_version	否
cost_param	否
enable_array_optimization	否
enable_bitmapscan	是
enable_broadcast	是
enable_csqual_pushdown	否
enable_redistribute	是
enable_extrapolation_stats	是
enable_fast_query_shipping	否
enable_force_vector_engine	否
enable_hashagg	是
enable_hashfilter	否
enable_hashjoin	是
enable_index_nestloop	是
enable_indexonlyscan	是
enable_indexscan	是
enable_join_pseudoconst	是
enable_mergejoin	是
enable_mixedagg	否

版权所有 © 华为云计算技术有限公司

配置参数名	是否支持在子查询级别设置
enable_nestloop	是
enable_nodegroup_debug	否
enable_partition_dynamic_pruning	是
enable_seqscan	是
enable_sonic_hashagg	否
enable_sonic_hashjoin	是
enable_sort	是
enable_stream_ctescan	否
enable_tidscan	是
enable_value_redistribute	是
enable_vector_engine	否
expected_computing_nodegroup	否
force_bitmapand	是
from_collapse_limit	是
join_collapse_limit	是
join_num_distinct	是
outer_join_max_rows_multipler	是
prefer_hashjoin_path	否
qrw_inlist2join_optmode	是
qual_num_distinct	是
query_dop	否
query_max_mem	否
query_mem	否
rewrite_rule	否
setop_optmode	是
skew_option	是
stream_ctescan_max_estimate_mem	否
stream_ctescan_pred_threshold	否
stream_ctescan_refcount_threshold	否
windowagg_pushdown_enhancement	否

 版权所有 © 华为云计算技术有限公司

配置参数名	是否支持在子查询级别设置
index_selectivity_cost	是
index_cost_limit	是

示例

对示例中原语句使用如下hint：

```
explain
select /*+ set global(query_dop 0) */ i_product_name product_name
...
```

该hint表示：在整个语句的计划生成过程中，将配置参数query_dop设置为0，即打开SMP自适应功能。生成的计划如下图所示：

```
 id |                                    operation                                    | E-rows  | E-memory | E-width | E-costs
----+---------------------------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                                                 |       1 |          |     230 | 19595.89
  2 |    ->  Vector Sonic Hash Aggregate                                              |       1 |          |     230 | 19595.89
  3 |       ->  Vector Streaming (type: GATHER)                                       |       3 |          |     230 | 19595.89
  4 |          ->  Vector Sonic Hash Aggregate                                        |       3 | 16MB     |     230 | 19595.66
  5 |             ->  Vector Nest Loop (6,28)                                         |       3 | 1MB      |     126 | 19595.62
  6 |                ->  Vector Nest Loop (7,27)                                      |       3 | 1MB      |     130 | 19291.57
  7 |                   ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)             |       3 | 4MB      |     118 | 19279.41
  8 |                      ->  Vector Nest Loop (9,24)                                |       3 | 1MB      |     118 | 19279.38
  9 |                         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) |       3 | 4MB      |      82 | 18117.66
 10 |                            ->  Vector Nest Loop (11,21)                         |       3 | 1MB      |      82 | 18117.61
 11 |                               ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) |   3 | 4MB      |      82 | 16195.20
 12 |                                  ->  Vector Sonic Hash Join (13,15)            |       3 | 16MB     |      82 | 16195.15
 13 |                                     ->  Vector Partition Iterator              |  287514 | 1MB      |      12 | 1110.42
 14 |                                        ->  Partitioned CStore Scan on store_returns | 287514 | 1MB  |      12 | 1110.42
 15 |                                     ->  Vector Streaming(type: LOCAL BROADCAST dop: 2/2) | 2764 | 4MB  |      94 | 14718.42
 16 |                                        ->  Vector Sonic Hash Join (17,19)       |    1382 | 16MB     |      94 | 14699.69
 17 |                                           ->  Vector Partition Iterator         | 2880404 | 1MB      |      39 | 11541.07
 18 |                                              ->  Partitioned CStore Scan on store_sales | 2880404 | 1MB | 39 | 11541.07
 19 |                                           ->  Vector Streaming(type: LOCAL BROADCAST dop: 2/2) |  16 | 4MB  |      55 | 1947.12
 20 |                                              ->  CStore Scan on item            |       8 | 1MB      |      55 | 1947.00
 21 |                               ->  Vector Materialize                            |  100000 | 16MB     |       8 | 1797.41
 22 |                                  ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 2/2) | 100000 | 4MB |  8 | 1714.07
 23 |                                     ->  CStore Scan on customer                |  100000 | 1MB      |       8 | 703.67
 24 |                         ->  Vector Materialize                                  |   50000 | 16MB     |      44 | 1099.22
 25 |                            ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 2/2) |  50000 | 4MB  |      44 | 1057.55
 26 |                               ->  CStore Scan on customer_address ad2           |   50000 | 1MB      |      44 | 552.33
 27 |                   ->  CStore Scan on store                                      |      36 | 1MB      |      20 | 12.01
 28 |                ->  CStore Scan on promotion                                     |     900 | 1MB      |       4 | 300.30
(28 rows)
```

16.4.9.13 Hint 的错误、冲突及告警

Plan Hint的结果会体现在计划的变化上，可以通过explain来查看变化。

Hint中的错误不会影响语句的执行，只是不能生效，该错误会根据语句类型以不同方式提示用户。对于explain语句，hint的错误会以warning形式显示在界面上，对于非explain语句，会以debug1级别日志显示在日志中，关键字为PLANHINT。

hint 的错误类型

- 语法错误

  语法规则树归约失败，会报错，指出出错的位置。

  例如：hint关键字错误，leading hint或join hint指定2个表以下，其它hint未指定表等。一旦发现语法错误，则立即终止hint的解析，所以此时只有错误前面的解析完的hint有效。

  例如：

  ```
  leading((t1 t2)) nestloop(t1) rows(t1 t2 #10)
  ```

  nestloop(t1)存在语法错误，则终止解析，可用hint只有之前解析的leading((t1 t2))。

 版权所有 © 华为云计算技术有限公司

- 语义错误
  - 表不存在，存在多个，或在leading或join中出现多次，均会报语义错误。
  - scanhint中的index不存在，会报语义错误。
  - 另外，如果子查询提升后，同一层出现多个名称相同的表，且其中某个表需要被hint，hint会存在歧义，无法使用，需要为相同表增加别名规避。
- hint重复或冲突

  如果存在hint重复或冲突，只有第一个hint生效，其它hint均会失效，会给出提示。
  - hint重复是指，hint的方法及表名均相同。例如：nestloop(t1 t2) nestloop(t1 t2)。
  - hint冲突是指，table list一样的hint，存在不一样的hint，hint的冲突仅对于每一类hint方法检测冲突。

    例如：nestloop (t1 t2) hashjoin (t1 t2)，则后面与前面冲突，此时hashjoin的hint失效。注意：nestloop(t1 t2)和no mergejoin(t1 t2)不冲突。

    须知

    leading hint中的多个表会进行拆解。例如：leading ((t1 t2 t3))会拆解成：leading((t1 t2)) leading(((t1 t2) t3))，此时如果存在leading((t2 t1))，则两者冲突，后面的会被丢弃。（例外：指定内外表的hint若与不指定内外表的hint重复，则始终丢弃不指定内外表的hint。）
- 子链接提升后hint失效

  子链接提升后的hint失效，会给出提示。通常出现在子链接中存在多个表连接的场景。提升后，子链接中的多个表不再作为一个整体出现在join中。
- 列类型不支持重分布
  - 对于skew hint来说，目的是为了进行重分布时的调优，所以当hint列的类型不支持重分布时，hint将无效。
- hint未被使用
  - 非等值join使用hashjoin hint或mergejoin hint
  - 不包含索引的表使用indexscan hint或indexonlyscan hint
  - 通常只有在索引列上使用过滤条件才会生成相应的索引路径，全表扫描将不会使用索引，因此使用indexscan hint或indexonlyscan hint将不会使用
  - indexonlyscan只有输出列仅包含索引列才会使用，否则指定时hint不会被使用
  - 多个表存在等值连接时，仅尝试有等值连接条件的表的连接，此时没有关联条件的表之间的路径将不会生成，所以指定相应的leading，join，rows hint将不使用，例如：t1 t2 t3表join，t1和t2，t2和t3有等值连接条件，则t1和t3不会优先连接，leading(t1 t3)不会被使用。
  - 生成stream计划时，如果表的分布列与join列相同，则不会生成redistribute的计划；如果不同，且另一表分布列与join列相同，只能生成redistribute的计划，不会生成broadcast的计划，指定相应的hint则不会被使用。
  - 对于AGG重分布列的hint，hint未被使用的可能原因如下：
    - 指定的分布键包含不支持重分布的数据类型。

 版权所有 © 华为云计算技术有限公司

- 执行计划中不需要重分布。
- 执行的分布键的序号有误。
- 对于使用grouping sets子句和cube子句的AP函数，window agg中的分布键，不支持hint。

  说明

  指定Agg重分布列Hint，仅8.1.3.100及以上集群版本支持。

– 如果子链接未被提升，则blockname hint不会被使用。
– 对于skew hint，hint未被使用的可能原因如下：
  - 计划中不需要进行重分布。
  - hint指定的列为包含分布键。
  - hint指定倾斜信息有误或不完整，如对于join优化未指定值。
  - 倾斜优化的GUC参数处于关闭状态。
– 对于guc hint，hint未被使用的可能原因如下：
  - 配置参数不存在。
  - 配置参数不支持guc hint。
  - 配置参数的值无效。
  - 语句级别的guc hint没有被写在顶层查询中。
  - 子查询级别的guc hint设置的配置参数不支持在子查询级别设置。
  - guc hint所在的子查询被提升。

16.4.9.14 Plan Hint 实际调优案例

本节以TPC-DS标准测试的Q24的部分语句为例，在1000X，24DN环境上，说明使用plan hint进行实际调优的过程。示例如下：

```
select avg(netpaid) from
(select c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size
,sum(ss_sales_price) netpaid
from store_sales
,store_returns
,store
,item
,customer
,customer_address
where ss_ticket_number = sr_ticket_number
and ss_item_sk = sr_item_sk
and ss_customer_sk = c_customer_sk
```


版权所有 © 华为云计算技术有限公司

```
and ss_item_sk = i_item_sk
and ss_store_sk = s_store_sk
and c_birth_country = upper(ca_country)
and s_zip = ca_zip
and s_market_id=7
group by c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size);
```

1. 该语句的初始计划如下，运行时间110s：

图 16-10 语句初始计划

```
 id |                                                                operation                                                                |        A-time         |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------------+-----------+-----------
  1 | ->  Row Adapter                                                                                                                         | 110324.107            |         1 |         1
  2 |    ->  Vector Aggregate                                                                                                                 | 110324.093            |         1 |         1
  3 |       ->  Vector Streaming (type: GATHER)                                                                                               | 110323.958            |        24 |        24
  4 |          ->  Vector Aggregate                                                                                                           | [110179.302,110309.653] |        24 |        24
  5 |             ->  Vector Hash Aggregate                                                                                                   | [110178.388,110308.515] |    647824 |     16656
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                                                 | [77616.177,96478.771] |    666834733 |     16664
  7 |                   ->  Vector Hash Join (8,22)                                                                                           | [81727.257,84728.519] |    666834733 |     16664
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)                                                                           | [78770.520,82021.087] |    666834733 |     16664
  9 |                         ->  Vector Hash Join (10,21)                                                                                    | [88066.755,90701.860] |    666834733 |     16664
 10 |                            ->  Vector Streaming(type: BROADCAST)                                                                        | [7940.962,21430.725]  |    591882336 |     51360
 11 |                               ->  Vector Hash Join (12,20)                                                                              | [2419.995,5319.606]   |     24661764 |      2140
 12 |                                  ->  Vector Streaming(type: REDISTRIBUTE)                                                               | [1750.448,4659.581]   |     25258268 |      2241
 13 |                                     ->  Vector Hash Join (14,18)                                                                        | [15240.666,17159.616] |     25258268 |      2241
 14 |                                        ->  Vector Hash Join (15,17)                                                                     | [12112.913,13563.366] |    252564412 |    472070592
 15 |                                           ->  Vector Partition Iterator                                                                 | [11148.731,12473.230] |  2879987999 |  2879987999
 16 |                                              ->  Partitioned CStore Scan on public.store_sales                                          | [11097.921,12412.596] |  2879987999 |  2879987999
 17 |                                           ->  CStore Scan on public.store                                                               | [0.447,0.689]         |        2064 |        2064
 18 |                                        ->  Vector Partition Iterator                                                                    | [296.805,319.014]     |    287999764 |    287999764
 19 |                                           ->  Partitioned CStore Scan on public.store_returns                                           | [292.938,314.787]     |    287999764 |    287999764
 20 |                                     ->  CStore Scan on public.customer                                                                  | [114.358,144.462]     |     12000000 |     12000000
 21 |                                  ->  CStore Scan on public.customer_address                                                             | [38.426,56.753]       |      6000000 |      6000000
 22 |                               ->  CStore Scan on public.item                                                                            | [3.160,5.026]         |       300000 |       300000
(22 rows)
```

该计划中，第10层算子使用broadcast性能较差，由于第11层算子估算行数为2140，比实际行数严重低估。错误行数估算主要来源于第13层算子的行数低估，根因是第13层hashjoin中，使用store_sales的(ss_ticket_number, ss_item_sk)列和store_returns的(sr_ticket_number, sr_item_sk)列进行关联，由于缺少多列相关性的估算导致行数严重低估。

2. 使用如下的rows hint进行调优后，计划如下，运行时间318s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns * 11270)*/ c_last_name ...
```

图 16-11 使用 rows hint 进行调优

```
 id |                                                                operation                                                                |        A-time         |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------------+-----------+-----------
  1 | ->  Row Adapter                                                                                                                         | 318585.246            |         1 |         1
  2 |    ->  Vector Aggregate                                                                                                                 | 318585.232            |         1 |         1
  3 |       ->  Vector Streaming (type: GATHER)                                                                                               | 318585.082            |        24 |        24
  4 |          ->  Vector Aggregate                                                                                                           | [318323.324,318499.290] |        24 |        24
  5 |             ->  Vector Hash Aggregate                                                                                                   | [318320.813,318497.054] |    647824 |  187770504
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                                                 | [288074.860,305601.698] |    666834733 |  187770507
  7 |                   ->  Vector Hash Join (8,22)                                                                                           | [253642.468,315808.664] |    666834733 |  187770507
  8 |                      ->  Vector Hash Join (9,18)                                                                                        | [250904.317,315684.018] |    666834733 |  187770507
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)                                                                        | [4552.500,310602.307] |    275042158 |  147106999
 10 |                            ->  Vector Hash Join (11,17)                                                                                 | [7658.951,14053.823]  |    275042158 |  147106999
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)                                                                  | [3953.255,10264.943]  |    287999764 |  154060900
 12 |                                  ->  Vector Hash Join (13,15)                                                                           | [28196.188,32838.794] |    287999764 |  154060900
 13 |                                     ->  Vector Partition Iterator                                                                       | [11477.673,12324.583] |  2879987999 |  2879987999
 14 |                                        ->  Partitioned CStore Scan on public.store_sales                                                | [11411.382,12250.209] |  2879987999 |  2879987999
 15 |                                     ->  Vector Partition Iterator                                                                       | [304.188,403.205]     |    287999764 |    287999764
 16 |                                        ->  Partitioned CStore Scan on public.store_returns                                              | [299.838,398.255]     |    287999764 |    287999764
 17 |                               ->  CStore Scan on public.customer                                                                        | [122.246,170.128]     |     12000000 |     12000000
 18 |                            ->  Vector Streaming(type: REDISTRIBUTE)                                                                     | [57.558,117.461]      |       492915 |       146467
 19 |                               ->  Vector Hash Join (20,21)                                                                              | [45.554,96.238]       |       492915 |       146467
 20 |                                  ->  CStore Scan on public.customer_address                                                             | [39.738,89.412]       |      6000000 |      6000000
 21 |                                  ->  CStore Scan on public.store                                                                        | [0.361,1.095]         |        2064 |        2064
 22 |                         ->  Vector Streaming(type: BROADCAST)                                                                           | [48.986,91.170]       |      7200000 |      7200000
 23 |                            ->  CStore Scan on public.item                                                                               | [4.506,6.602]         |       300000 |       300000
(23 rows)
```


版权所有 © 华为云计算技术有限公司

时间反而劣化了，原因是第8层hashjoin过慢引起第9层redistribute时间过慢导致，其中第9层redistribute并没有数据倾斜，hashjoin慢的原因是由于第18层redistribute后数据倾斜导致。

3. 经过实际数据查证，customer_address的两个join列的不同值数目较少，使用其进行join容易出现数据倾斜，故把customer_address放到最后进行join。使用如下的hint进行调优后，计划如下，运行时间116s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((store_sales store_returns store item customer) customer_address)*/
c_last_name ...
```

图 16-12 hint 调优

```
 id |                                   operation                                    |        A-time         |  A-rows  |  E-rows
----+--------------------------------------------------------------------------------+-----------------------+----------+----------+
  1 | ->  Row Adapter                                                                | 116326.597            |        1 |        1 |
  2 |   ->  Vector Aggregate                                                         | 116326.590            |        1 |        1 |
  3 |     ->  Vector Streaming (type: GATHER)                                        | 116326.473            |       24 |       24 |
  4 |       ->  Vector Aggregate                                                     | [116157.161,116236.494] |       24 |       24 |
  5 |         ->  Vector Hash Aggregate                                              | [116155.328,116233.946] |   647824 | 187770504 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                             | [84103.951,102052.326] |  666834733 | 187770507 |
  7 |             ->  Vector Hash Join (8,10)                                        | [23229.469,47484.697] |  666834733 | 187770507 |
  8 |               ->  Vector Streaming(type: REDISTRIBUTE)                         | [38.367,74.930]       |  6000000 |  6000000 |
  9 |                 ->  CStore Scan on public.customer_address                     | [69.877,121.460]      |  6000000 |  6000000 |
 10 |               ->  Vector Streaming(type: REDISTRIBUTE)                         | [17404.744,17567.550] |  24661764 |  24112909 |
 11 |                 ->  Vector Hash Join (12,22)                                   | [16123.627,16397.246] |  24661764 |  24112909 |
 12 |                   ->  Vector Streaming(type: REDISTRIBUTE)                     | [15320.663,15741.646] |  25258268 |  25252751 |
 13 |                     ->  Vector Hash Join (14,21)                               | [14962.342,16375.458] |  25258268 |  25252751 |
 14 |                       ->  Vector Hash Join (15,19)                             | [14449.031,15825.949] |  25258268 |  25252751 |
 15 |                         ->  Vector Hash Join (16,18)                           | [11439.959,12510.065] |  252564412 |  472070592 |
 16 |                           ->  Vector Partition Iterator                        | [10531.986,11536.213] |  2879987999 |  2879987999 |
 17 |                             ->  Partitioned CStore Scan on public.store_sales  | [10483.634,11474.944] |  2879987999 |  2879987999 |
 18 |                         ->  CStore Scan on public.store                        | [0.347,0.463]         |  2064 |  2064 |
 19 |                       ->  Vector Partition Iterator                            | [293.977,365.021]     |  287999764 |  287999764 |
 20 |                         ->  Partitioned CStore Scan on public.store_returns    | [289.936,360.808]     |  287999764 |  287999764 |
 21 |                     ->  CStore Scan on public.item                             | [3.109,5.245]         |  300000 |  300000 |
 22 |                 ->  CStore Scan on public.customer                             | [113.871,141.791]     |  12000000 |  12000000 |
(22 rows)
```

发现时间基本花在了第6层redistribute算子上，需要进一步优化。

4. 由于最后一层redistribute包含倾斜，所以时间较长。为了避免倾斜，需要将item表放在最后join，由于item表的join并不能使行数减少。修改hint如下并执行，计划如下，运行时间120s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
c_last_name ...
```

图 16-13 修改 hint 并执行语句

```
 id |                                   operation                                    |        A-time         |  A-rows  |  E-rows
----+--------------------------------------------------------------------------------+-----------------------+----------+----------+
  1 | ->  Row Adapter                                                                | 120377.258            |        1 |        1 |
  2 |   ->  Vector Aggregate                                                         | 120377.245            |        1 |        1 |
  3 |     ->  Vector Streaming (type: GATHER)                                        | 120377.091            |       24 |       24 |
  4 |       ->  Vector Aggregate                                                     | [120184.884,120301.704] |       24 |       24 |
  5 |         ->  Vector Hash Aggregate                                              | [120183.119,120297.845] |   647824 | 187770504 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                             | [87775.682,106070.878] |  666834733 | 187770507 |
  7 |             ->  Vector Hash Join (8,22)                                        | [22323.764,49878.523] |  666834733 | 187770507 |
  8 |               ->  Vector Hash Join (9,11)                                      | [21129.236,45208.255] |  666834733 | 187770507 |
  9 |                 ->  Vector Streaming(type: REDISTRIBUTE)                       | [37.859,75.412]       |  6000000 |  6000000 |
 10 |                   ->  CStore Scan on public.customer_address                   | [74.798,114.449]      |  6000000 |  6000000 |
 11 |                 ->  Vector Streaming(type: REDISTRIBUTE)                       | [15714.458,15824.928] |  24661764 |  24112909 |
 12 |                   ->  Vector Hash Join (13,21)                                 | [14637.516,14955.464] |  24661764 |  24112909 |
 13 |                     ->  Vector Streaming(type: REDISTRIBUTE)                   | [13898.593,14333.200] |  25258268 |  25252751 |
 14 |                       ->  Vector Hash Join (15,19)                             | [14166.917,15378.244] |  25258268 |  25252751 |
 15 |                         ->  Vector Hash Join (16,18)                           | [11272.239,12052.532] |  252564412 |  472070592 |
 16 |                           ->  Vector Partition Iterator                        | [10409.566,11127.981] |  2879987999 |  2879987999 |
 17 |                             ->  Partitioned CStore Scan on public.store_sales  | [10365.838,11077.601] |  2879987999 |  2879987999 |
 18 |                         ->  CStore Scan on public.store                        | [0.431,0.609]         |  2064 |  2064 |
 19 |                       ->  Vector Partition Iterator                            | [343.780,408.254]     |  287999764 |  287999764 |
 20 |                         ->  Partitioned CStore Scan on public.store_returns    | [339.844,403.923]     |  287999764 |  287999764 |
 21 |                     ->  CStore Scan on public.customer                         | [117.234,163.598]     |  12000000 |  12000000 |
 22 |               ->  Vector Streaming(type: BROADCAST)                            | [44.571,130.129]      |  7200000 |  7200000 |
 23 |                 ->  CStore Scan on public.item                                 | [4.169,6.347]         |  300000 |  300000 |
(23 rows)
```

该计划中的redistribute问题并没有解决，因为第22层item表做了broadcast，导致与customer_address表join后的倾斜并没有被消除掉。

5. 增加如下禁止item表做broadcast的hint，使与customer_address join的表做redistribute（也可以进行join表redistribute的hint），计划如下，运行时间105s：

 版权所有 © 华为云计算技术有限公司

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
no broadcast(item)*/
c_last_name ...
```

图 16-14 执行计划

```
 id |                                                                operation                                                                |          A-time          |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+--------------------------+-----------+-----------
  1 | ->  Row Adapter                                                                                                                         | 105854.957               |         1 |         1
  2 |   ->  Vector Aggregate                                                                                                                  | 105854.948               |         1 |         1
  3 |     ->  Vector Streaming (type: GATHER)                                                                                                 | 105854.825               |        24 |        24
  4 |       ->  Vector Aggregate                                                                                                              | [105706.709,105776.135]  |        24 |        24
  5 |         ->  Vector Hash Aggregate                                                                                                       | [105705.061,105773.013]  |    647824 | 187770504
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                                                                                      | [70701.966,89973.672]    |    666834733 | 187770507
  7 |             ->  Vector Hash Join (8,23)                                                                                                 | [71759.500,79018.433]    |    666834733 | 187770507
  8 |               ->  Vector Streaming(type: REDISTRIBUTE)                                                                                  | [69794.307,77269.178]    |    666834733 | 187770507
  9 |                 ->  Vector Hash Join (10,12)                                                                                            | [21443.307,46714.378]    |    666834733 | 187770507
 10 |                   ->  Vector Streaming(type: REDISTRIBUTE)                                                                              | [41.295,83.419]          |    6000000 |   6000000
 11 |                     ->  CStore Scan on public.customer_address                                                                          | [70.405,166.072]         |    6000000 |   6000000
 12 |                   ->  Vector Streaming(type: REDISTRIBUTE)                                                                              | [15689.053,15788.475]    |    24661764 |  24112909
 13 |                     ->  Vector Hash Join (14,22)                                                                                        | [14517.847,14712.929]    |    24661764 |  24112909
 14 |                       ->  Vector Streaming(type: REDISTRIBUTE)                                                                          | [13806.733,14089.770]    |    25258268 |  25252751
 15 |                         ->  Vector Hash Join (16,20)                                                                                    | [13709.384,15095.449]    |    25258268 |  25252751
 16 |                           ->  Vector Hash Join (17,19)                                                                                  | [10944.796,11827.285]    |    252564412 | 472070592
 17 |                             ->  Vector Partition Iterator                                                                               | [10070.316,10884.728]    |    2879987999 | 2879987999
 18 |                               ->  Partitioned CStore Scan on public.store_sales                                                         | [10018.966,10828.990]    |    2879987999 | 2879987999
 19 |                             ->  CStore Scan on public.store                                                                             | [0.447,0.568]            |    2064 |      2064
 20 |                           ->  Vector Partition Iterator                                                                                 | [293.042,329.056]        |    287999764 | 287999764
 21 |                             ->  Partitioned CStore Scan on public.store_returns                                                         | [288.631,324.782]        |    287999764 | 287999764
 22 |                         ->  CStore Scan on public.customer                                                                              | [113.735,138.235]        |    12000000 |  12000000
 23 |                   ->  CStore Scan on public.item                                                                                        | [3.127,5.357]            |    300000 |   300000
(23 rows)
```

6. 发现最后一层使用单层Agg，但行数缩减较多。使用相同的hint，同时结合参数best_agg_plan=3进行双层Agg调优，最终计划如下图所示，运行时间94s，完成调优。

图 16-15 最终调优计划

```
 id |                                                                operation                                                                |          A-time          |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+--------------------------+-----------+-----------
  1 | ->  Row Adapter                                                                                                                         | 94004.670                |         1 |         1
  2 |   ->  Vector Aggregate                                                                                                                  | 94004.655                |         1 |         1
  3 |     ->  Vector Streaming (type: GATHER)                                                                                                 | 94004.504                |        24 |        24
  4 |       ->  Vector Aggregate                                                                                                              | [93833.832,93928.052]    |        24 |        24
  5 |         ->  Vector Hash Aggregate                                                                                                       | [93832.460,93926.412]    |    647824 | 187770507
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                                                                                      | [93640.866,93787.939]    |    647824 | 183912384
  7 |             ->  Vector Hash Aggregate                                                                                                   | [93687.544,93791.242]    |    647824 | 183912384
  8 |               ->  Vector Hash Join (9,24)                                                                                               | [70025.469,72773.161]    |    666834733 | 187770507
  9 |                 ->  Vector Streaming(type: REDISTRIBUTE)                                                                                | [68242.223,71275.972]    |    666834733 | 187770507
 10 |                   ->  Vector Hash Join (11,13)                                                                                          | [21421.136,44830.306]    |    666834733 | 187770507
 11 |                     ->  Vector Streaming(type: REDISTRIBUTE)                                                                            | [35.444,71.328]          |    6000000 |   6000000
 12 |                       ->  CStore Scan on public.customer_address                                                                        | [67.246,119.224]         |    6000000 |   6000000
 13 |                     ->  Vector Streaming(type: REDISTRIBUTE)                                                                            | [16089.853,16212.570]    |    24661764 |  24112909
 14 |                       ->  Vector Hash Join (15,23)                                                                                      | [14822.972,15188.942]    |    24661764 |  24112909
 15 |                         ->  Vector Streaming(type: REDISTRIBUTE)                                                                        | [14061.867,14604.162]    |    25258268 |  25252751
 16 |                           ->  Vector Hash Join (17,21)                                                                                  | [13949.756,15492.311]    |    25258268 |  25252751
 17 |                             ->  Vector Hash Join (18,20)                                                                                | [10935.742,12160.719]    |    252564412 | 472070592
 18 |                               ->  Vector Partition Iterator                                                                             | [10052.958,11194.962]    |    2879987999 | 2879987999
 19 |                                 ->  Partitioned CStore Scan on public.store_sales                                                       | [10008.415,11143.984]    |    2879987999 | 2879987999
 20 |                               ->  CStore Scan on public.store                                                                           | [0.452,0.839]            |    2064 |      2064
 21 |                             ->  Vector Partition Iterator                                                                               | [298.235,332.736]        |    287999764 | 287999764
 22 |                               ->  Partitioned CStore Scan on public.store_returns                                                       | [294.067,327.629]        |    287999764 | 287999764
 23 |                           ->  CStore Scan on public.customer                                                                            | [114.377,145.156]        |    12000000 |  12000000
 24 |                     ->  CStore Scan on public.item                                                                                      | [3.150,3.530]            |    300000 |   300000
(24 rows)
```

如果有统计信息变更引起的查询劣化，可以考虑用plan hint来调整到之前的查询计划。这里以TPCH-Q17为例，在收集default_statistics_target设置为–2的统计信息之后，计划相比于默认统计信息发生劣化。

1. 默认统计信息（ default_statistics_target设置为100 ）的计划如下：

 版权所有 © 华为云计算技术有限公司

图 16-16 默认统计信息

```
id |                                                operation                                                |          A-time
---+---------------------------------------------------------------------------------------------------------+---------------------------
  1 | ->  Row Adapter                                                                                         | 265006.779
  2 |    ->  Vector Aggregate                                                                                 | 265006.764
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 265006.071
  4 |          ->  Vector Aggregate                                                                           | [263699.512,264503.084]
  5 |             ->  Vector Hash Join (6,17)                                                                 | [263676.665,264477.932]
  6 |                ->  Vector Streaming(type: LOCAL GATHER dop: 1/4)                                        | [1.998,7.594]
  7 |                   ->  Vector Hash Aggregate                                                             | [201775.393,202432.672]
  8 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 4/4)                            | [201567.130,202231.524]
  9 |                         ->  Vector Hash Join (10,12)                                                    | [170675.231,199908.410]
 10 |                            ->  Vector Partition Iterator                                                | [34847.797,51968.266]
 11 |                               ->  Partitioned CStore Scan on tpch10wx_col.lineitem                      | [33805.013,51137.657]
 12 |                            ->  Vector Hash Aggregate                                                    | [23283.387,25359.493]
 13 |                               ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4)                      | [12850.624,14608.515]
 14 |                                  ->  Vector Hash Aggregate                                              | [2690.439,3616.623]
 15 |                                     ->  Vector Partition Iterator                                       | [2659.700,3579.390]
 16 |                                        ->  Partitioned CStore Scan on tpch10wx_col.part                 | [2642.213,3559.093]
 17 |                ->  Vector Streaming(type: REDISTRIBUTE dop: 1/4)                                        | [262300.732,262961.078]
 18 |                   ->  Vector Hash Join (19,21)                                                          | [225749.727,260990.322]
 19 |                      ->  Vector Partition Iterator                                                      | [40046.078,56220.694]
 20 |                         ->  Partitioned CStore Scan on tpch10wx_col.lineitem                            | [39204.414,55328.448]
 21 |                      ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4)                               | [55748.177,61987.136]
 22 |                         ->  Vector Partition Iterator                                                   | [3042.864,3873.942]
 23 |                            ->  Partitioned CStore Scan on tpch10wx_col.part                             | [3027.023,3848.159]
(23 rows)
```

2. 统计信息变更（default_statistics_target设置为–2）的计划如下：

图 16-17 统计信息变更

```
id |                                                operation                                                |          A-time
---+---------------------------------------------------------------------------------------------------------+---------------------------
  1 | ->  Row Adapter                                                                                         | 1440492.994
  2 |    ->  Vector Aggregate                                                                                 | 1440492.982
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 1440491.021
  4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/6)                                              | [1439737.284,1440008.568]
  5 |             ->  Vector Aggregate                                                                        | [1439008.369,1439854.148]
  6 |                ->  Vector Hash Join (7,18)                                                              | [1439006.016,1439851.619]
  7 |                   ->  Vector Streaming(type: LOCAL BROADCAST dop: 6/6)                                  | [2.932,139.405]
  8 |                      ->  Vector Hash Aggregate                                                          | [190452.312,195910.748]
  9 |                         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                         | [190171.929,195653.119]
 10 |                            ->  Vector Hash Join (11,13)                                                 | [161076.195,178831.123]
 11 |                               ->  Vector Partition Iterator                                             | [27306.318,45564.565]
 12 |                                  ->  Partitioned CStore Scan on tpch10wx_col.lineitem                   | [26752.444,44912.020]
 13 |                               ->  Vector Hash Aggregate                                                 | [35601.624,39812.058]
 14 |                                  ->  Vector Streaming(type: SPLIT BROADCAST dop: 6/6)                   | [23096.460,27057.137]
 15 |                                     ->  Vector Hash Aggregate                                           | [2372.587,3052.445]
 16 |                                        ->  Vector Partition Iterator                                    | [2345.381,3012.732]
 17 |                                           ->  Partitioned CStore Scan on tpch10wx_col.part              | [2329.874,2989.393]
 18 |                   ->  Vector Hash Join (19,22)                                                          | [1437388.414,1438470.781]
 19 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                            | [1392693.529,1408571.859]
 20 |                         ->  Vector Partition Iterator                                                   | [29065.204,41264.514]
 21 |                            ->  Partitioned CStore Scan on tpch10wx_col.lineitem                         | [28212.219,40133.491]
 22 |                      ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 6/6)                            | [2570.841,3438.567]
 23 |                         ->  Vector Partition Iterator                                                   | [2447.569,3276.369]
 24 |                            ->  Partitioned CStore Scan on tpch10wx_col.part                             | [2432.124,3263.641]
(24 rows)
```

3. 经过对比，劣化的原因主要为lineitem和part表join时stream类型由BroadCast变更为Redistribute导致。可以对语句进行stream方式的hint来调整到之前的计划，例如：

图 16-18 调整语句

```
select /*+ no redistribute(part lineitem) */
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

 版权所有 © 华为云计算技术有限公司

16.4.10 例行维护表

为了保证数据库的有效运行，数据库必须在插入/删除操作后，定期执行VACUUM FULL和ANALYZE，更新统计信息，以便获得更优的性能。

相关概念

使用VACUUM、VACUUM FULL和ANALYZE命令定期对每个表进行维护，主要有以下原因：

- VACUUM FULL可回收已更新或已删除的数据所占据的磁盘空间，同时将小数据文件合并。
- VACUUM对每个表维护了一个可视化映射来跟踪包含对别的活动事务可见的数组的页。一个普通的索引扫描首先通过可视化映射来获取对应的数组，来检查是否对当前事务可见。若无法获取，再通过堆数组抓取的方式来检查。因此更新表的可视化映射，可加速唯一索引扫描。
- VACUUM可避免执行的事务数超过数据库阈值时，事务ID重叠造成的原有数据丢失。
- ANALYZE可收集与数据库中表内容相关的统计信息。统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，生成最有效的执行计划。

操作步骤

步骤1 使用VACUUM或VACUUM FULL命令，进行磁盘空间回收。

- **VACUUM**：

  对表执行VACUUM操作。

  ```
  VACUUM customer;
  ```

  可以与数据库操作命令并行运行。（执行期间，可正常使用的语句：SELECT、INSERT、UPDATE和DELETE。不可正常使用的语句：ALTER TABLE）。

  对表分区执行VACUUM操作。

  ```
  VACUUM customer_par PARTITION ( P1 );
  ```

- **VACUUM FULL**：

  ```
  VACUUM FULL customer;
  ```

  需要向正在执行的表增加排他锁，且需要停止其他所有数据库操作。

  在进行磁盘空间回收时，用户可以使用如下命令查询集群中最早事务对应session，再根据需要结束最早执行的长事务，从而更加高效的利用磁盘空间。

  a. 使用命令从GTM上查询oldestxmin

  ```
  SELECT * FROM pgxc_gtm_snapshot_status();
  ```

  b. 从CN上查询对应的session的pid，此处xmin为上一步的oldestxmin。

  ```
  SELECT * FROM pgxc_running_xacts() where xmin=1400202010;
  ```

步骤2 使用ANALYZE语句更新统计信息。

```
ANALYZE customer;
```

使用ANALYZE VERBOSE语句更新统计信息，并输出表的相关信息。

```
ANALYZE VERBOSE customer;
```

也可以同时执行VACUUM ANALYZE命令进行查询优化。

```
VACUUM ANALYZE customer;
```


版权所有 © 华为云计算技术有限公司

说明

VACUUM和ANALYZE会导致I/O流量的大幅增加，这可能会影响其他活动会话的性能。因此，建议通过“vacuum_cost_delay”参数设置。

----结束

维护建议

- 定期对部分大表做VACUUM FULL，在性能下降后为全库做VACUUM FULL，目前暂定每月做一次VACUUM FULL。
- 定期对系统表做VACUUM FULL，主要是PG_ATTRIBUTE。
- 启用系统自动清理进程（AUTOVACUUM）自动执行VACUUM和ANALYZE，回收被标识为删除状态的记录空间，并更新表的统计数据。

16.4.11 例行重建索引

背景信息

数据库经过多次删除操作后，索引页面上的索引键将被删除，造成索引膨胀。例行重建索引，可有效的提高查询效率。

数据库支持的索引类型包含B-tree索引、GIN索引和PSORT索引。

- 对于B-tree索引，例行重建索引可有效的提高查询效率。
  - 如果数据发生大量删除后，索引页面上的索引键将被删除，导致索引页面数量的减少，造成索引膨胀。重建索引可回收浪费的空间。
  - 新建的索引中逻辑结构相邻的页面，通常在物理结构中也是相邻的，所以一个新建的索引比更新了多次的索引访问速度要快。
- 对于非B-tree索引，不建议例行重建。

重建索引

重建索引有以下两种方式：

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）。

  在删除索引过程中，会在父表上增加一个短暂的排他锁，阻止相关读写操作。在创建索引过程中，会锁住写操作但是不会锁住读操作，此时读操作只能使用顺序扫描。
- 使用REINDEX语句重建索引。
  - 使用REINDEX TABLE语句重建索引，会在重建过程中增加排他锁，阻止相关读写操作。
  - 使用REINDEX INTERNAL TABLE语句重建desc表（包括）的索引，会在重建过程中增加排他锁，阻止相关读写操作。

操作步骤

假定在导入表“areaS”上的“area_id”字段上存在普通索引“areaS_idx”。重建索引有以下两种方式：

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）

版权所有 © 华为云计算技术有限公司

a. 删除索引。
```
DROP INDEX areaS_idx;
```
b. 创建索引。
```
CREATE INDEX areaS_idx ON areaS (area_id);
```

- 使用REINDEX重建索引。
  - 使用REINDEX TABLE语句重建索引。
    ```
    REINDEX TABLE areaS;
    ```
  - 使用REINDEX INTERNAL TABLE重建desc表（包括）的索引。
    ```
    REINDEX INTERNAL TABLE areaS;
    ```

16.4.12 SQL 语句出错自动重试

DWS支持在SQL语句执行出错时自动重试（下文简称CN Retry）。对于来自gsql客户端、JDBC、ODBC驱动的SQL语句，在SQL语句执行失败时，CN端能够自动识别语句执行过程中的报错，并重新下发任务进行自动重试。

该功能的限制和约束如下：

- 功能范围限制：
  - 仅能提高故障发生时SQL语句执行成功率，不能保证100%的执行成功。
  - CN Retry默认开启，开启后temp表会记录日志，关闭CN Retry后，temp表不会记录日志，因此不能在使用temp表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。
  - CN Retry默认开启，开启后新创建的unlogged表会忽视unlogged关键字，建成普通表。关闭CN Retry后，unlogged表不会记录日志，因此不能在使用unlogged表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。
  - 在使用gds进行数据导出时，支持CN Retry。现有机制导出时会对重复文件进行检测并删除相同的文件，因此建议不要对相同的外表重复导出数据，除非确定数据目录中相同文件名的文件需要删除。
- 错误类型约束：

  SQL语句出错时能够被识别和重试的错误，仅限在错误类型列表（请参考**表16-20**）中定义的错误。
- 语句类型约束：

  支持单语句CN Retry、存储过程、函数、匿名块。不支持事务块中的语句。
- 存储过程语句约束：
  - 包含EXCEPTION的存储过程，如果在执行过程中（包含语句块执行和EXCEPTION中的语句执行）错误被抛出，可以retry，且系统内部错误发生时，retry会先于EXCEPTION被执行，而如果报错被EXCEPTION捕获则不能retry。
  - 不支持使用全局变量的package。
  - 不支持DBMS_JOB。
  - 不支持UTL_FILE。
  - 如果存储过程中有输出打印信息（如dbms_output.put_line或raise info等），则发生retry时会重复输出已打印的消息，并会在重复消息前输出“Notice：Retry triggered, some message may be duplicated.”加以提示。
- 集群状态约束：

 版权所有 © 华为云计算技术有限公司

  - 仅支持DN、GTM实例故障。
  - CN Retry有次数限制，如果在CN Retry达到最大尝试次数（最大次数由max_query_retry_times控制）之前，集群状态无法从故障状态恢复到正常状态，那么CN Retry不能保证执行成功。
  - 扩容时不支持CN Retry。
- 数据导入约束：
  - 不支持COPY FROM STDIN语句。
  - 不支持gsql \copy from元命令。
  - 不支持JDBC CopyManager copyIn导入数据。

CN Retry支持的错误类型列表和对应的错误码信息见**表16-20**，可以通过GUC参数retry_ecode_list设置CN Retry支持的错误类型列表，但不建议用户直接修改该参数，如有修改需求请联系技术工程师协助处理。

表 16-20 CN Retry 支持的错误类型列表

错误类型	错误码	备注
对端连接重置（CONNECTION_RESET_BY_PEER）	YY001	TCP通信错误：Connection reset by peer（CN和DN间通信）
对端流重置（STREAM_CONNECTION_RESET_BY_PEER）	YY002	TCP通信错误：Stream connection reset by peer（DN和DN间通信）
锁等待超时（LOCK_WAIT_TIMEOUT）	YY003	锁超时，Lock wait timeout
连接超时（CONNECTION_TIMED_OUT）	YY004	TCP通信错误，Connection timed out
查询设置错误（SET_QUERY_ERROR）	YY005	SET命令发送失败，Set query
超出逻辑内存（OUT_OF_LOGICAL_MEMORY）	YY006	内存申请失败，Out of logical memory
通信库内存分配（SCTP_MEMORY_ALLOC）	YY007	SCTP通信错误，Memory allocate error
无通信库缓存数据（SCTP_NO_DATA_IN_BUFFER）	YY008	SCTP通信错误，SCTP no data in buffer
通信库释放内存关闭（SCTP_RELEASE_MEMORY_CLOSE）	YY009	SCTP通信错误，Release memory close
SCTP、TCP断开（SCTP_TCP_DISCONNECT）	YY010	SCTP通信错误，TCP disconnect
通信库断开（SCTP_DISCONNECT）	YY011	SCTP通信错误，SCTP disconnect

 版权所有 © 华为云计算技术有限公司

错误类型	错误码	备注
通信库远程关闭（SCTP_REMOTE_CLOSE）	YY012	SCTP通信错误，Stream closed by remote
等待未知通信库通信（SCTP_WAIT_POLL_UNKNOWN）	YY013	等待未知通信库通信，SCTP wait poll unknown
无效快照（SNAPSHOT_INVALID）	YY014	快照非法，Snapshot invalid
通讯接收信息错误（ERRCODE_CONNECTION_RECEIVE_WRONG）	YY015	连接获取错误，Connection receive wrong
内存耗尽（OUT_OF_MEMORY）	53200	内存耗尽，Out of memory
连接失败（CONNECTION_FAILURE）	08006	GTM出错，Connection failure
连接异常（CONNECTION_EXCEPTION）	08000	连接出现错误，和DN的通讯失败，Connection exception
管理员关闭系统（ADMIN_SHUTDOWN）	57P01	管理员关闭系统，Admin shutdown
关闭远程流接口（STREAM_REMOTE_CLOSE_SOCKET）	XX003	关闭远程套接字，Stream remote close socket
重复查询编号（ERRCODE_STREAM_DUPLICATE_QUERY_ID）	XX009	重复查询，Duplicate query id
stream查询并发更新同一行（ERRCODE_STREAM_CONCURRENT_UPDATE）	YY016	stream查询并发更新同一行，Stream concurrent update
LLVM内存分配错误（ERRCODE_LLVM_BAD_ALLOC_ERROR ）	CG003	内存分配错误， Allocate error
LLVM致命错误（ERRCODE_LLVM_FATAL_ERROR ）	CG004	致命错误，Fatal error
HashJoin临时文件读取错误(ERRCODE_HASHJOIN_TEMP_FILE_ERROR)	F0011	临时文件读取错误，File error
Buffer文件读取错误(ERRCODE_BUFFER_FILE_ERROR)	F0012	文件读取错误，File error

版权所有 © 华为云计算技术有限公司

错误类型	错误码	备注
分区个数发生变化(ERRCODE_PARTITION_NUM_CHANGED)	45003	在扫描LIST分区表时，发现此时的分区个数和优化阶段的分区个数不一致，一般出现在查询和ADD/DROP分区并发时。（此错误类型仅8.1.3及以上集群版本支持）
节点间对象SCHEMA名称不一致(ERRCODE_UNMATCH_OBJECT_SCHEMA)	42P30	对象SCHEMA名称不一致，Unmatched schema name

开启CN Retry功能需要设置如下GUC参数：

- 必选的GUC参数（CN和DN都需设置）

  max_query_retry_times

  注意

  CN Retry功能开启时会为临时表数据记录日志，为保证数据一致性，在使用临时表时不能切换CN Retry开关状态，保持使用临时表的会话中CN Retry开关始终处于打开状态或者关闭状态。

- 可选的GUC参数

  cn_send_buffer_size

  max_cn_temp_file_size

16.4.13 query_band 负载识别

概述

DWS实现基于query_band的负载识别和队列内优先级控制，一方面提供了更为灵活的负载识别手段，可根据作业类型、应用名称、脚本名称等识别负载队列，使用户根据业务场景可灵活配置query_band识别队列；另一方面实现了队列内作业下发优先级控制，后续将逐步实现队列内资源优先级控制。

管理员用户可根据业务场景及作业类别配置query_band所关联队列及估算内存限制等实现更为灵活的负载控制与资源管控。如果业务未配置query_band或用户未将query_band关联行为时，作业会默认使用用户关联队列和队列内优先级。

query_band 支持的负载行为

query_band是一个session级别的GUC参数，作为作业标识符，本身没有特殊含义，数据类型为字符型，支持赋值任何字符串。但是为方便区分和设置，query_band负载识别仅支持识别键值对形式的query_band，示例：

```
SET query_band='JobName=abc;AppName=test;UserName=user';
```

其中，‘JobName=abc’、‘AppName=test’以及‘UserName=user’都是一个独立的键值对。query_band键值对规格：

版权所有 © 华为云计算技术有限公司

- query_band使用键值对方式设置，即'key=value'；session内支持设置多个query_band键值对，多个键值对之间使用分号分隔。query_band键值对和query_band参数值长度的上限均为1024个字符。
- query_band键值对支持的有效字符包括：数字0~9、大写字母A~Z、小写字母a-z、'.'、'-'、'_' 以及'#'。

query_band负载识别以键值对为单位设置和识别负载行为，目前支持的负载行为，如**表16-21**所示：

表 16-21 QUERY_BAND 支持负载行为

类别	行为	行为表现
负载管理(workload)	资源池(respool)	query_band关联资源池
负载管理(workload)	优先级(priority)	队列内优先级
次序(order)	队列（respool） 目前为无效字段，主要用于后续扩展。	query_band搜索次序

其中，行为类别用于负载行为归类，不同的负载行为可能属于同一个类别，比如资源池和优先级同属于负载管理类别；负载行为代表query_band键值对关联的是哪种负载行为；行为表现用于记录具体的负载行为是什么；负载类别中的次序用于标记query_band负载行为识别的优先级，一个session内设置多个query_band键值对时，优先使用次序较小的query_band键值对识别负载行为。每一个query_band键值对都可以对应0个或多个负载行为，但是一种负载行为只能关联一个。query_band负载行为详细说明如下：

- 资源池：query_band支持关联资源池，作业执行时，若query_band指定了资源池，则使用query_band关联的资源池，否则使用用户关联的资源池。
  - query_band关联资源池时，资源池不存在报错退出，关联失败。
  - query_band关联资源池时，记录query_band与资源池依赖关系。
  - query_band关联资源池删除时，提示有query_band依赖，资源池删除失败。
- 队列内优先级：query_band支持关联作业优先级，支持高中低(High/Medium/Low)三个优先级，同时提供Rush作为特殊优先级（绿色通道），默认优先级为Medium。正常实践过程中，大部分作业使用Medium优先级，优先级较低作业使用Low优先级，特权作业使用High优先级，High作业不建议过多。Rush优先级作为特殊场景下应急使用，平时不建议使用。

  队列内优先级实现队列内排队优先级：
  - 静态负载管理场景下，CN并发不足时，触发CN全局队列排队，CN全局队列为优先级队列。
  - 动态负载管理场景下，DN内存不足时，触发CCN全局排队，CCN全局队列为优先级队列。
  - 资源池并发或内存不足时，触发资源池排队，资源池队列为优先级队列。

  以上优先级队列均遵守以下调度规则：

 版权所有 © 华为云计算技术有限公司

  - 优先级高作业优先调度。
  - 优先级高作业全部调度完之后调度优先级低作业。
  - 动态负载管理场景下，CN全局队列不支持query_band优先级。
- 次序：支持设置query_band识别次序，未设置识别次序的使用默认次序-1。除默认次序外，不存在次序相同的两个query_band。设置次序时对query_band次序进行校验，存在相同次序时，已存在次序递归+1直到不存在相同次序为止。
  - session内设置多个query_band键值对时，使用次序较小的query_band键值对作为负载识别的query_band。
  - 次序最小为0，默认次序-1为最大次序。
  - 次序都为默认次序时，使用设置靠前的query_band作为负载识别的query_band。
  - 示例：set query_band='b=1;a=3;c=1'; b=1，其中b=1次序-1，a=3次序4，c=1次序1，则使用c=1作为负载识别的query_band，此设计可提供负载管理员对负载调度调整能力。

query_band 应用与配置

- pg_workload_action跨库系统表用于存储query_band行为和次序，详见**PG_WORKLOAD_ACTION**。
- 默认行为和次序在系统表pg_workload_action不存储，query_band有设置非默认行为的，查询其行为默认行为也显示；查询行为和次序都为默认的query_band行为时，显示<query_band information not found>。
- gs_wlm_set_queryband_action函数用于设置query_band行为：其中第一个参数即queryband键值对的长度上限为63个字符；第二个参数action不区分大小写，多个action使用分号分隔；order为缺省参数，默认为-1。具体请参见《数据仓库服务SQL语法参考》中“资源管理函数”中的gs_wlm_set_queryband_action函数。
- gs_wlm_set_queryband_order函数设置query_band次序：其中第一个参数即queryband键值对的长度上限为63个字符；query_band次序必须大于等于-1，除默认次序-1外不存在两个次序相同的query_band。设置query_band次序时如果存在相同次序的query_band，则将原query_band次序+1。具体请参见《数据仓库服务SQL语法参考》中“资源管理函数”中的gs_wlm_set_queryband_order函数。
- gs_wlm_get_queryband_action函数用于查询query_band行为，具体请参见《数据仓库服务SQL语法参考》中“资源管理函数”中的gs_wlm_set_queryband_action函数。
- pg_queryband_action系统视图用于查询所有query_band行为，具体请参见**PG_QUERYBAND_ACTION**。
- query_band优先级在负载管理视图（**PG_SESSION_WLMSTAT**）中显示为int型，数字和优先级的对应关系如下：
  - 0：表示该作业不受负载管理管控；
  - 1：LOW；
  - 2：MEDIUM；
  - 4：HIGH；
  - 8：RUSH；
- 权限控制：除初始用户外，被授权用户才具有设置和查询query_band权限。

 版权所有 © 华为云计算技术有限公司

说明

- 批量取消所有运行作业或队列并发上限是1且只有一个队列有作业运行的情况下，可能会触发CN自动唤醒作业导致作业不按照优先级下发。

示例

步骤1 设置query_band“JobName=abc”关联资源池p1、队列内优先级Rush、次序为1。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=p1;priority=rush',1);
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤2 修改query_band“JobName=abc”的关联资源池为p2。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=p2');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤3 修改query_band“JobName=abc”的队列内优先级为High。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','priority=high');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤4 修改query_band“JobName=abc”的次序为3。

```
SELECT * FROM gs_wlm_set_queryband_order('JobName=abc',3);
gs_wlm_set_queryband_order
-----------------------------
 t
(1 row)
```

步骤5 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
    qband     | respool_id | respool | priority | qborder
--------------+------------+---------+----------+---------
 JobName=abc |      17119 | p2      | high     |       1
(1 row)
```

步骤6 设置query_band“AppName=test”队列内优先级Low，使用用户关联资源池，同时使用默认次序。

```
SELECT * FROM gs_wlm_set_queryband_action('AppName=test','priority=low');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤7 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
    qband     | respool_id | respool | priority | qborder
--------------+------------+---------+----------+---------
 AppName=test |          0 | NULL    | low      |      -1
 JobName=abc |      16754 | p2      | high     |       3
(2 rows)
```

步骤8 取消query_band“JobName=abc”关联的所有负载行为，设置为默认行为即可。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=null;priority=medium',-1);
NOTICE:  The respool of query_band(JobName=abc) will be removed.
NOTICE:  The priority of query_band(JobName=abc) will be removed.
 gs_wlm_set_queryband_action
-----------------------------
```

 版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
```

步骤9 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
   qband     | respool_id | respool | priority | qborder
-------------+------------+---------+----------+---------
 AppName=test |          0 | NULL    | low      |      -1
(1 row)
```

----结束

16.5 SQL 调优案例

16.5.1 案例：选择合适的分布列

分布列用于将数据分布到不同的节点上，划分均衡可以避免数据倾斜。

在进行关联查询时，尽量选择查询中的关联条件作为分布键。当关联条件作为分布键时，相关数据都分布在DN本地，将减少DN之间的数据流动代价，提升查询速度。

优化前

将a作为t1和t2的分布列，表定义如下：

```
CREATE TABLE t1 (a int, b int) DISTRIBUTE BY HASH (a);
CREATE TABLE t2 (a int, b int) DISTRIBUTE BY HASH (a);
```

执行如下查询：

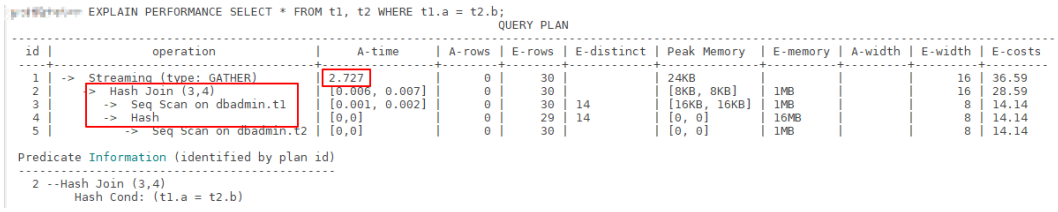
```
SELECT * FROM t1, t2 WHERE t1.a = t2.b;
```

则执行计划存在“Streaming(type: REDISTRIBUTE)”，即DN根据选定的列把数据重分布到所有的DN，这将导致DN之间存在较大通信数据量，如图16-19所示。

图 16-19 选择合适的分布列案例（一）

```
[redacted]=> EXPLAIN PERFORMANCE SELECT * FROM t1, t2 WHERE t1.a = t2.b;
                                                
```

图 16-20 选择合适的分布列案例（二）



```
EXPLAIN PERFORMANCE SELECT * FROM t1, t2 WHERE t1.a = t2.b;
                                                
```

```
CREATE INDEX idx_o_custkey ON orders (o_custkey) LOCAL;
```

执行SQL语句查询建立索引后的执行计划，发现执行时间为18毫秒。

```
gaussdb=> EXPLAIN PERFORMANCE SELECT * FROM orders WHERE o_custkey = '1186459';
                                                
```

优化后

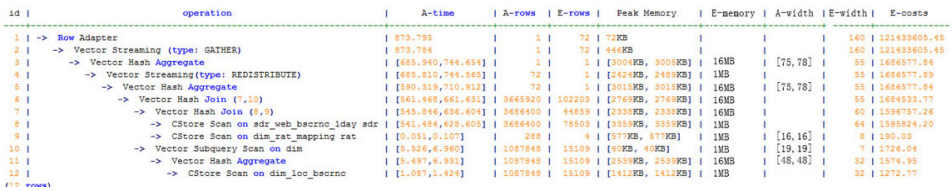
1. 分析执行计划图16-21可知，在顺序扫描阶段耗时较多。
2. 多表JOIN中，由于表PS.SDR_WEB_BSCRNC_1DAY的JOIN列“BSCRNC_ID”存在大量空值，JOIN性能差。

   建议在语句中手动添加JOIN列的非空判断，修改后的语句如下所示。

```
SELECT
 *
FROM
( ( SELECT
  STARTTIME STTIME,
  SUM(NVL(PAGE_DELAY_MSEL,0)) PAGE_DELAY_MSEL,
  SUM(NVL(PAGE_SUCCEED_TIMES,0)) PAGE_SUCCEED_TIMES,
  SUM(NVL(FST_PAGE_REQ_NUM,0)) FST_PAGE_REQ_NUM,
  SUM(NVL(PAGE_AVG_SIZE,0)) PAGE_AVG_SIZE,
  SUM(NVL(FST_PAGE_ACK_NUM,0)) FST_PAGE_ACK_NUM,
  SUM(NVL(DATATRANS_DW_DURATION,0)) DATATRANS_DW_DURATION,
  SUM(NVL(PAGE_SR_DELAY_MSEL,0)) PAGE_SR_DELAY_MSEL
 FROM
  PS.SDR_WEB_BSCRNC_1DAY SDR
  INNER JOIN (SELECT
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID
     FROM
      nethouse.DIM_LOC_BSCRNC
     GROUP BY
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID) DIM
  ON SDR.BSCRNC_ID = DIM.BSCRNC_ID
  AND DIM.ACCESS_TYPE_ID IN (0,1,2)
  INNER JOIN nethouse.DIM_RAT_MAPPING RAT
  ON (RAT.RAT = SDR.RAT)
 WHERE
  ( (STARTTIME >= 1461340800
  AND STARTTIME < 1461427200) )
  AND RAT.ACCESS_TYPE_ID IN (0,1,2)
  and SDR.BSCRNC_ID is not null
 GROUP BY
  STTIME ) ) A;
```

执行计划如图16-22所示。

图 16-22 增加 JOIN 列非空条件（二）



16.5.4 案例：典型不下推语句整改案例集

16.5.4.1 案例：使排序下推

在做场景性能测试时，发现某场景大部分时间是CN端在做window agg，占到总执行时间95%以上，系统资源不能充分利用。研究发现该场景的特点是：将两列分别求

版权所有 © 华为云计算技术有限公司

sum作为一个子查询，外层对两列的和再求和后做trunc，然后排序。可以尝试将语句改写为子查询，使排序下推。

优化前

表结构如下所示：

```
CREATE TABLE public.test(imsi int,L4_DW_THROUGHPUT int,L4_UL_THROUGHPUT int)
with (orientation = column) DISTRIBUTE BY hash(imsi);
```

查询语句如下所示：

```
SELECT COUNT(1) over() AS DATACNT,
IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))), 0) AS
DECIMAL(20)) AS TOTAL_VOLOME_KPIID
FROM public.test AS test
GROUP BY IMSI
ORDER BY TOTAL_VOLOME_KPIID DESC LIMIT 10;
```

执行计划如下：

```
QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 id |                  operation                  |      A-time      | A-rows  | E-rows  | E-distinct | Peak Memory  |   E-memory   | A-width | E-width | E-costs
----+---------------------------------------------+------------------+---------+---------+------------+--------------+--------------+---------+---------+---------
  1 | ->  Row Adapter                             | 2862.008         |      10 |      10 |            | 31KB         |              |      28 | 48360.42
  2 |    ->  Vector Limit                         | 2861.969         |      10 |      10 |            | 8KB          |              |      28 | 48360.42
  3 |       ->  Vector Sort                       | 2861.946         |      10 | 1000000 |            | 479KB        |              |      28 | 50860.39
  4 |          ->  Vector WindowAgg               | 2166.759         | 1000000 | 1000000 |            | 69987KB      |              |      28 | 26750.75
  5 |             ->  Vector Streaming (type: GATHER)  | 136.813          | 1000000 | 1000000 |            | 208KB        |              |      28 | 15500.75
  6 |                ->  Vector Sonic Hash Aggregate  | [71.374, 73.640] | 1000000 | 1000000 |            | [14MB, 14MB] | 96MB(2919MB) | [31,31] |      28 | 15032.00
  7 |                   ->  CStore Scan on public.test | [2.957, 2.994]   | 1000000 | 1000000 |            | [1MB, 1MB]   | 1MB          |         |      12 | 1282.00
```

可以看到window agg和sort全部在CN端执行，耗时非常严重。

优化后

尝试将语句改写为子查询：

```
SELECT COUNT(1) over() AS DATACNT, IMSI_IMSI, TOTAL_VOLOME_KPIID
FROM (SELECT IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))),
0) AS DECIMAL(20)) AS TOTAL_VOLOME_KPIID
FROM public.test AS test
GROUP BY IMSI
ORDER BY TOTAL_VOLOME_KPIID DESC LIMIT 10);
```

将trunc两列的和作为一个子查询，然后在子查询的外面做window agg，这样排序就可以下推了，执行计划如下：

```
 QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 id |                    operation                    |       A-time       | A-rows  | E-rows  | E-distinct |  Peak Memory   |   E-memory   | A-width | E-width | E-costs
----+-------------------------------------------------+--------------------+---------+---------+------------+----------------+--------------+---------+---------+---------
```

 版权所有 © 华为云计算技术有限公司

```
+----------------+--------------+---------+---------+----------
  1 | ->  Row Adapter                                | 955.277           |      10 |       5 |           | 31KB
|           |         |      24 | 25843.13
  2 |    ->  Vector WindowAgg                          | 955.261           |      10 |       5 |           | 1572KB
|           |         |      24 | 25843.13
  3 |       ->  Vector Streaming (type: GATHER)            | 955.015           |      10 |      10 |           |
127KB         |           |         |      24 | 25843.07
  4 |         ->  Vector Limit                         | [0.018, 0.018]    |      10 |      10 |           | [8KB, 8KB]    |
1MB           |         |      28 | 25836.97
  5 |           ->  Vector Streaming(type: BROADCAST)      | [0.014, 0.014]    |      20 |      20 |           |
[719KB, 719KB] | 2MB           |         |      28 | 25837.12
  6 |             ->  Vector Limit                       | [927.730, 934.283] |      20 |      20 |           | [8KB, 8KB]
| 1MB           |         |      28 | 25836.85
  7 |               ->  Vector Sort                    | [927.720, 934.269] |      20 | 1000000 |           | [463KB,
463KB] | 16MB          | [32,32] |      28 | 27086.82
  8 |                 ->  Vector Sonic Hash Aggregate   | [456.841, 461.077] | 1000000 | 1000000 |           |
[15MB, 15MB]   | 96MB(2916MB) | [31,31] |      28 | 15032.00
  9 |                   ->  CStore Scan on public.test | [2.959, 3.014]    | 1000000 | 1000000 |           | [1MB,
1MB]    | 1MB           |         |      12 | 1282.00
```

经过SQL改写，性能由2.862s提升0.955s，优化效果明显。需注意，本示例中优化结果仅供参考，由于WindowAgg的不确定性，优化后的结果集跟实际业务相关。

16.5.4.2 案例：With-Recursive contains only values rte is not shippable

问题根因

递归语句的某个分支中没有FROM子句（只有 VALUES 或者类似 SELECT 1 这样的语句）。

案例 1：递归驱动分支没有 FROM 子句

原始语句

```
SELECT  T.RPT_ITEM_ID, --报表项ID
   T.RPT_ITEM_CODE,
   T.USER_GROUP_CODE AS USER_GROUP_CODE --用户组
FROM BIF.BIF_RPT_ITEM_DEF_T T,
(WITH recursive cte AS (
       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, 1) + 1, INSTR('', ',', 1, 2) - INSTR('', ',', 1, 1) -
1)) AS cte_RPT_ITEM_CODE,
           1 AS level
       FROM (SELECT '') AS tb0

       UNION ALL

       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, cte.level + 1) + 1, INSTR('', ',', 1, cte.level + 2) -
INSTR('', ',', 1, cte.level + 1) - 1)),
           cte.level + 1
       FROM (SELECT '') AS tb0, cte
       WHERE cte.level + 1 <= LENGTH('') - LENGTH(REPLACE('', ',', '')) - 1
   )
   SELECT
       DISTINCT cte_RPT_ITEM_CODE AS RPT_ITEM_CODE
   FROM cte
) T5
WHERE NVL(INSTR(T.RPT_ITEM_FREQUENCE, 'M'), 0) > 0
AND T.RPT_ITEM_CODE = NVL(T5.RPT_ITEM_CODE, T.RPT_ITEM_CODE)
AND T.RPT_ITEM_TYPE = 1 --是否是叶子报表项，1=是，0=否，基本报表项
AND T.ENABLE_FLAG = 1
AND T.VERSION = '202308' --使用快照,增加条件限制
ORDER BY T.RPT_ITEM_ID
```

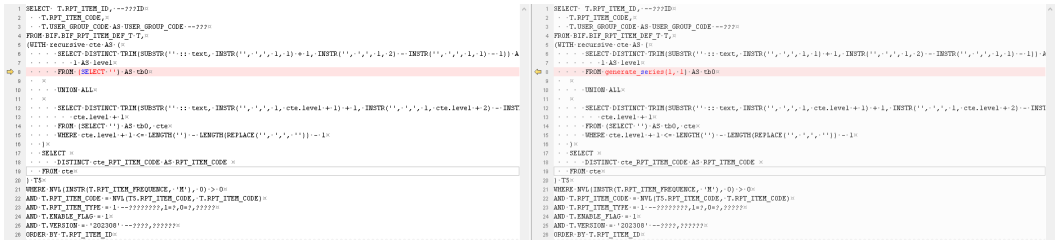
改写语句

```
SELECT  T.RPT_ITEM_ID, --报表项ID
   T.RPT_ITEM_CODE,
```

 版权所有 © 华为云计算技术有限公司

```
    T.USER_GROUP_CODE AS USER_GROUP_CODE --用户组
FROM BIF.BIF_RPT_ITEM_DEF_T T,
(WITH recursive cte AS (
        SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, 1) + 1, INSTR('', ',', 1, 2) - INSTR('', ',', 1, 1) -
1)) AS cte_RPT_ITEM_CODE,
            1 AS level
        FROM generate_series(1, 1) AS tb0

        UNION ALL

        SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, cte.level + 1) + 1, INSTR('', ',', 1, cte.level + 2) -
INSTR('', ',', 1, cte.level + 1) - 1)),
            cte.level + 1
        FROM (SELECT '') AS tb0, cte
        WHERE cte.level + 1 <= LENGTH('') - LENGTH(REPLACE('', ',', '')) - 1
    )
    SELECT
        DISTINCT cte_RPT_ITEM_CODE AS RPT_ITEM_CODE
    FROM cte
) T5
WHERE NVL(INSTR(T.RPT_ITEM_FREQUENCE, 'M'), 0) > 0
AND T.RPT_ITEM_CODE = NVL(T5.RPT_ITEM_CODE, T.RPT_ITEM_CODE)
AND T.RPT_ITEM_TYPE = 1 --是否是叶子报表项，1=是，0=否，基本报表项
AND T.ENABLE_FLAG = 1
AND T.VERSION = '202308' --使用快照,增加条件限制
ORDER BY T.RPT_ITEM_ID
```

修改点比对



案例 2：递归驱动分支没有 FROM 子句

原始语句

```
SELECT A.DYNM_COMP_ID,
    DECODE(B.LINE_NO, 1, '202308', A.VERSION)
FROM BIF.BIF_DYNM_COMP_SOU_TBL_V A,
(WITH recursive cte AS (
    SELECT 1 AS level
    UNION ALL
    SELECT cte.level + 1
    FROM cte
    WHERE cte.level + 1 < 3
    )
    SELECT
        level as LINE_NO
    FROM cte
) B
WHERE EXISTS (SELECT 1
    FROM BIF.BIF_RPT_ITEM_DEF_MV RPT,
    BIF.BIF_PUB_SUBJECT_AREA_T SBJ,
    BIF.BIF_SNAPSHORT_SUBJECT_V TYP
    WHERE A.DYNM_COMP_ID = RPT.DYNM_COMP_ID
    AND RPT.VERSION = 'current'
    AND RPT.SUBJECT_AREA_ID = SBJ.SUBJECT_AREA_ID
    AND SBJ.SUBJECT_AREA_CODE =TYP.SUBJECT_CODE
    AND TYP.SUBJECT_TYPE ='TAX')
AND A.VERSION = 'current'
```

改写语句

 版权所有 © 华为云计算技术有限公司

```
SELECT A.DYNM_COMP_ID,
   DECODE(B.LINE_NO, 1, '202308', A.VERSION)
FROM BIF.BIF_DYNM_COMP_SOU_TBL_V A,
(SELECT
   *
   FROM generate_series(1, 2) AS cte(LINE_NO)
) B
WHERE EXISTS (SELECT 1
   FROM BIF.BIF_RPT_ITEM_DEF_MV RPT,
   BIF.BIF_PUB_SUBJECT_AREA_T SBJ,
   BIF.BIF_SNAPSHORT_SUBJECT_V TYP
   WHERE A.DYNM_COMP_ID = RPT.DYNM_COMP_ID
   AND RPT.VERSION = 'current'
   AND RPT.SUBJECT_AREA_ID = SBJ.SUBJECT_AREA_ID
   AND SBJ.SUBJECT_AREA_CODE =TYP.SUBJECT_CODE
   AND TYP.SUBJECT_TYPE ='TAX')
AND A.VERSION = 'current'
```

修改点比对

案例 3：递归驱动分支是 VALUES 子句

原始语句

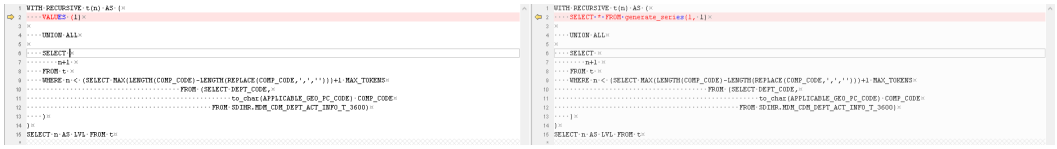
```
WITH RECURSIVE t(n) AS (
   VALUES (1)

   UNION ALL

   SELECT
      n+1
   FROM t
   WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
                              FROM (SELECT DEPT_CODE,
                                    to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
                                 FROM SDIHR.MDM_CDM_DEPT_ACT_INFO_T_3600)
   )
)
SELECT n AS LVL FROM t
```

改写语句

```
WITH RECURSIVE t(n) AS (
   SELECT * FROM generate_series(1, 1)

   UNION ALL

   SELECT
      n+1
   FROM t
   WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
                              FROM (SELECT DEPT_CODE,
                                    to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
                                 FROM SDIHR.MDM_CDM_DEPT_ACT_INFO_T_3600)
   )
)
SELECT n AS LVL FROM t
```

 版权所有 © 华为云计算技术有限公司

修改点比对



案例 4：递归驱动分支是 VALUES 子句

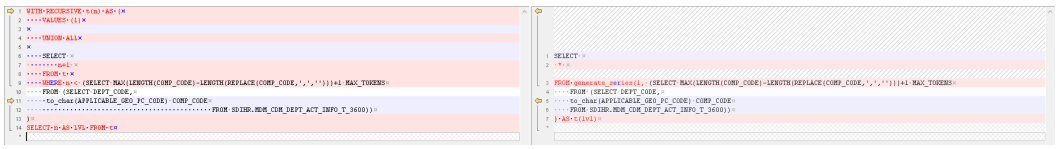
原始语句

```
WITH RECURSIVE t(n) AS (
    VALUES (1)

    UNION ALL

    SELECT
        n+1
    FROM t
    WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
    FROM (SELECT DEPT_CODE,
     to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
                                        FROM SDIHR.MDM_CDM_DEPT_ACT_INFO_T_3600))
)
SELECT n AS LVL FROM t
```

改写语句

```
SELECT
 *
FROM generate_series(1, (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
    FROM (SELECT DEPT_CODE,
    to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
    FROM SDIHR.MDM_CDM_DEPT_ACT_INFO_T_3600))
) AS t(lvl)
```

修改点比对



16.5.4.3 案例：With-Recursive contains system table is not shippable

问题根因

递归语句的某个分支中没有完整的From子句，仅使用系统表或者系统视图(DUAL也被视为系统视图)代替。

案例 1：递归驱动分支是 FROM DUAL 查询

原始语句

```
WITH recursive cte AS (
    SELECT
        TO_DATE(201701, 'YYYYMM') as level ,TO_DATE(20170131, 'YYYYMMDD')  LASTDAY
    FROM dual
```

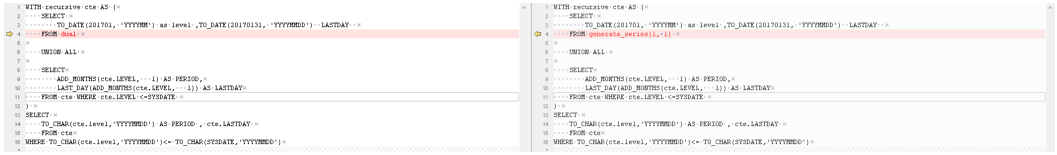

版权所有 © 华为云计算技术有限公司

```
   UNION ALL

   SELECT
      ADD_MONTHS(cte.LEVEL,   1) AS PERIOD,
      LAST_DAY(ADD_MONTHS(cte.LEVEL,   1)) AS LASTDAY
   FROM cte WHERE cte.LEVEL <=SYSDATE
)
SELECT
   TO_CHAR(cte.level,'YYYYMMDD') AS PERIOD , cte.LASTDAY
   FROM cte
WHERE TO_CHAR(cte.level,'YYYYMMDD')<= TO_CHAR(SYSDATE,'YYYYMMDD')
```

改写语句

```
WITH recursive cte AS (
   SELECT
      TO_DATE(201701, 'YYYYMM') as level ,TO_DATE(20170131, 'YYYYMMDD')  LASTDAY
   FROM generate_series(1, 1)

   UNION ALL

   SELECT
      ADD_MONTHS(cte.LEVEL,   1) AS PERIOD,
      LAST_DAY(ADD_MONTHS(cte.LEVEL,   1)) AS LASTDAY
   FROM cte WHERE cte.LEVEL <=SYSDATE
)
SELECT
   TO_CHAR(cte.level,'YYYYMMDD') AS PERIOD , cte.LASTDAY
   FROM cte
WHERE TO_CHAR(cte.level,'YYYYMMDD')<= TO_CHAR(SYSDATE,'YYYYMMDD')
```

修改点对比



16.5.4.4 案例：SubPlan exec on CN can't be shipped

问题根因

某个子查询语句只能在CN上执行，通常是这个子查询有不下推因素，比如有调用了系统表、系统视图，或者存在不下推函数等。

案例 1：子查询过程中存在系统表/系统视图的查询

原始语句

```
WITH error_log  AS NOT MATERIALIZED (
   SELECT
      upper(log_column_name) AS log_column_name,
      log_error_code,
      s.char_length AS data_length,
      s.data_type,s.nullable
   FROM (SELECT
         distinct
         unnest(string_to_array(bad_log_column_name,',')) AS log_column_name,
         unnest(string_to_array(bad_log_error_code,',')) AS log_error_code
      FROM stgltc.BAD_cfs_inv_invoice_ad_2500
   ) T,
   (SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')) S
   WHERE  upper(T.log_column_name)=upper(S.column_name)
)
```

 版权所有 © 华为云计算技术有限公司

```
SELECT
   CASE WHEN upper('ACTIVITY_NAME') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ACTIVITY_NAME,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ACTIVITY_NAME')))
      ELSE ACTIVITY_NAME
   END AS ACTIVITY_NAME,
   CASE WHEN upper('ADJUSTMENT_ID') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ADJUSTMENT_ID,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ADJUSTMENT_ID')))
      ELSE ADJUSTMENT_ID
   END AS ADJUSTMENT_ID
FROM stgltc.BAD_cfs_inv_invoice_ad_2500
```

改写语句

```
-- 识别不下推的子查询为WITH error_log字句中的
-- SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')
--
-- 因为这部分为系统表查询，无论如何都不能下推，所以此处把这部分结果转储到一个中间表中
-- 中间表创建成行存表
CREATE TEMP TABLE s WITH(orientation=row) DISTRIBUTE BY ROUNDROBIN AS
SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')

-- 因为整个查询涉及到的表都是列存表，之后前面创建的临时表s为行存表
-- 所以此处加一个强制走向量化的hint
WITH error_log  AS NOT MATERIALIZED (
   SELECT
      upper(log_column_name) AS log_column_name,
      log_error_code,
      s.char_length AS data_length,
      s.data_type,s.nullable
   FROM (SELECT
         distinct
         unnest(string_to_array(bad_log_column_name,',')) AS log_column_name,
         unnest(string_to_array(bad_log_error_code,',')) AS log_error_code
      FROM stgltc.bad_cfs_inv_invoice_ad_2500
   ) T,
   pg_temp.S
   WHERE  upper(T.log_column_name)=upper(S.column_name)
)

SELECT /*+ set global(enable_force_vector_engine on)*/
   CASE WHEN upper('ACTIVITY_NAME') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ACTIVITY_NAME,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ACTIVITY_NAME')))
      ELSE ACTIVITY_NAME
   END AS ACTIVITY_NAME,
   CASE WHEN upper('ADJUSTMENT_ID') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ADJUSTMENT_ID,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ADJUSTMENT_ID')))
      ELSE ADJUSTMENT_ID
   END AS ADJUSTMENT_ID
FROM stgltc.bad_cfs_inv_invoice_ad_2500
```

修改点对比

 版权所有 © 华为云计算技术有限公司

16.5.4.5 案例：Type of Record in TargetList can not be shipped

问题根因

输出列中存在record类型，这种类型的不下推一般是不会体现在最外层的输出列上，一般这类报错有两个场景：

1. SQL书写逻辑错误，导致输出列上出现了（ ... ）形式的输出列。
2. SQL业务逻辑正确， 这种场景需要了解业务含义，把record字段强转为text类型，再使用record字段的地方做特殊适配。

案例 1：输出列书写错误，出现（ ... ）形式的输出列

原始语句

```
SELECT
    d.id,
    coalesce(d.period, 'snull') AS period,
    (d.plan_unit_code, 'snull') AS plan_unit_code,
    coalesce(d.product_type_model, 'snull') AS product_type_model,
    coalesce(d.revision, 'snull') AS revision,
    d.start_date
FROM (SELECT *
    FROM cdcscm.cdc_mp_d_forecast_t_6120 t
    WHERE t.cdc_timestamp > to_date('2023-07-06 00:00:00', 'yyyy-mm-dd hh24:mi:ss') - 7
    AND t.cdc_timestamp < to_date('2023-08-08 00:00:00', 'yyyy-mm-dd hh24:mi:ss')
) t1,
sdiscm.mp_d_forecast_t_6120 d
WHERE (t1.audit_op_type = 'delete' AND t1.audit_op_option = 'before')
AND d.id = t1.id
```

改写语句

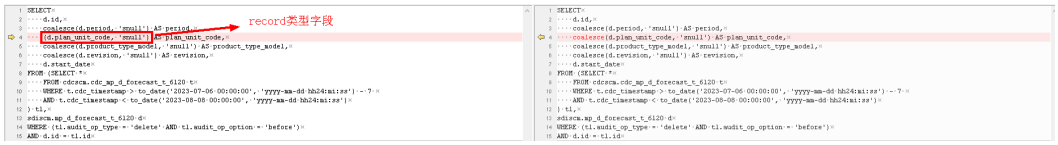
```
SELECT
    d.id,
    coalesce(d.period, 'snull') AS period,
    coalesce(d.plan_unit_code, 'snull') AS plan_unit_code,
    coalesce(d.product_type_model, 'snull') AS product_type_model,
    coalesce(d.revision, 'snull') AS revision,
    d.start_date
FROM (SELECT *
    FROM cdcscm.cdc_mp_d_forecast_t_6120 t
    WHERE t.cdc_timestamp > to_date('2023-07-06 00:00:00', 'yyyy-mm-dd hh24:mi:ss') - 7
    AND t.cdc_timestamp < to_date('2023-08-08 00:00:00', 'yyyy-mm-dd hh24:mi:ss')
) t1,
sdiscm.mp_d_forecast_t_6120 d
WHERE (t1.audit_op_type = 'delete' AND t1.audit_op_option = 'before')
AND d.id = t1.id
```


版权所有 © 华为云计算技术有限公司

说明

改写前后的语句不等价，不等价的原因是因为原始SQL书写有问题，正确的写法是就是coalesce(d.plan_unit_code, 'snull') AS plan_unit_code

修改点对比



16.5.4.6 案例：Type of Record in non-real table can not be shipped

问题根因

8.2.1.x特定版本下UPDATE语句+WITH语句场景存在下推问题。

案例 1：UPDATE 语句不下推

1. 通过GUC参数enable_stream_ctescan设置stream计划下不支持ctescan，在语句执行前进行设置。
   ```
   SET enable_stream_ctescan = off
   ```
2. 业务语句执行完再重置参数。
   ```
   SET enable_stream_ctescan = on
   ```
3. 语句中UPDATE关键字后加hint，具体方式为 UPDATE /*+ set global(enable_stream_ctescan off) */

16.5.4.7 案例：main plan exec on CN and SubPlan exec on DN can't be shipped

问题根因

根因：复杂语句中的主查询为不涉及用户表的查询，但相关的子查询是涉及用户表的查询语句。

案例 1：主查询来源表是维表 DUAL

原始语句

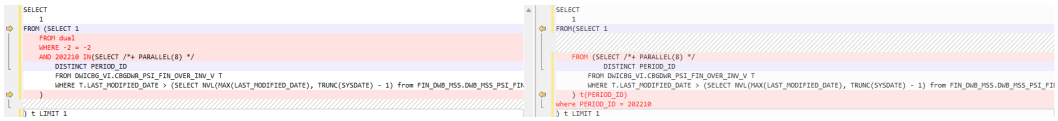
```
SELECT
    1
FROM (SELECT 1
    FROM dual
    WHERE -2 = -2
    AND 202210 IN(SELECT /*+ PARALLEL(8) */
        DISTINCT PERIOD_ID
        FROM DWICBG_VI.CBGDWR_PSI_FIN_OVER_INV_V T
        WHERE T.LAST_MODIFIED_DATE > (SELECT NVL(MAX(LAST_MODIFIED_DATE), TRUNC(SYSDATE) - 1)
from FIN_DWB_MSS.DWB_MSS_PSI_FIN_OVER_INV_M_F) AND T.LAST_MODIFIED_DATE <= (SELECT
TRUNC(SYSDATE) + 1)
    )
) t LIMIT 1
```

改写后语句

 版权所有 © 华为云计算技术有限公司

```
SELECT
    1
FROM(SELECT 1
    FROM (SELECT /*+ PARALLEL(8) */
            DISTINCT PERIOD_ID
        FROM DWICBG_VI.CBGDWR_PSI_FIN_OVER_INV_V T
        WHERE T.LAST_MODIFIED_DATE > (SELECT NVL(MAX(LAST_MODIFIED_DATE), TRUNC(SYSDATE) - 1)
from FIN_DWB_MSS.DWB_MSS_PSI_FIN_OVER_INV_M_F) AND T.LAST_MODIFIED_DATE <= (SELECT
TRUNC(SYSDATE) + 1)
    ) t(PERIOD_ID)
where PERIOD_ID = 202210
) t LIMIT 1
```

修改点对比



16.5.4.8 案例：With-Recursive under multi-nodegroup scenario is not shippable

问题根因

查询语句中所涉及的表分属于多个不同的逻辑集群中，这种场景的处理方案一般是通过中间表的方式让所有表在一个逻辑集群，或者把WITH RECURSIVE语句隔离出来以避免影响这个语句的性能。

案例 1：通过临时表把 WITH RECURSIVE 语句中的非当前逻辑集群的表转储到当前逻辑集群中

适用场景：语句中包含WITH RECURSIVE语句，且WITH RECURSIVE只有少部分（1~2个）表分属于其它逻辑集群，这些表数据量不大或者经过过滤条件之后结果集不大。

原始语句如下，语句中目标表dwltax.dwl_tax_tom_cit_tp_account_tmp在逻辑集群"LC_DW1"中，来源表dwtaxdi.dwi_tax_cit_tp_account_r_i中涉及的表来源于逻辑集群“LC_DL1”。

```
INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)
SELECT
    S.account_id,S.parentid
FROM (SELECT
        DISTINCT S.ACCOUNT_ID, S.PARENTID
    FROM (
        WITH RECURSIVE tmp AS (
            SELECT 1 AS LV,
                '/'||T.PARENTID AS path,
                account_ID AS account_ID,
                PARENTID AS PARENTID ,
                ROW_number() over() PATH_NUM,
                T.DEL_FLAG
            FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
            WHERE T.DEL_FLAG = 'N'
            AND T.ISLEAF_FLAG = 1

            UNION ALL

            SELECT 1+lv AS LV,
                p.path||'/'||t.PARENTID AS path,
                p.account_ID,
                t.PARENTID AS PARENTID,
                ROW_number() over() PATH_NUM,
```


版权所有 © 华为云计算技术有限公司

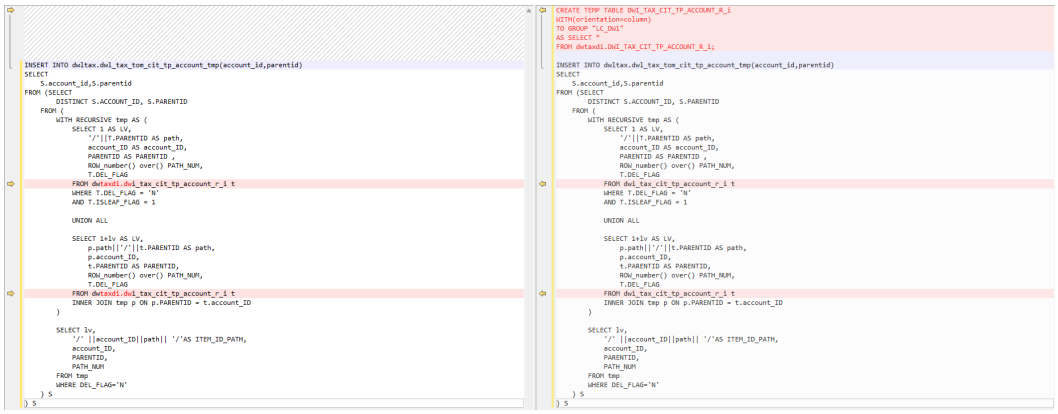
```
            T.DEL_FLAG
        FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
        INNER JOIN tmp p ON p.PARENTID = t.account_ID
    )

    SELECT lv,
        '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
        account_ID,
        PARENTID,
        PATH_NUM
    FROM tmp
    WHERE DEL_FLAG='N'
  ) S
) S
```

改写后语句

```
CREATE TEMP TABLE DWI_TAX_CIT_TP_ACCOUNT_R_i -- 把来源表数据导入临时表，临时表在逻辑集群
“LC_DW1”
WITH(orientation=column)
TO GROUP "LC_DW1"
AS SELECT *
FROM dwtaxdi.DWI_TAX_CIT_TP_ACCOUNT_R_i;

INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)  -- 来源表和目标表都在逻辑集群
“LC_DW1”
SELECT
   S.account_id,S.parentid
FROM (SELECT
      DISTINCT S.ACCOUNT_ID, S.PARENTID
   FROM (
      WITH RECURSIVE tmp AS (
         SELECT 1 AS LV,
            '/'||T.PARENTID AS path,
            account_ID AS account_ID,
            PARENTID AS PARENTID ,
            ROW_number() over() PATH_NUM,
            T.DEL_FLAG
         FROM dwi_tax_cit_tp_account_r_i t
         WHERE T.DEL_FLAG = 'N'
         AND T.ISLEAF_FLAG = 1

         UNION ALL

         SELECT 1+lv AS LV,
            p.path||'/'||t.PARENTID AS path,
            p.account_ID,
            t.PARENTID AS PARENTID,
            ROW_number() over() PATH_NUM,
            T.DEL_FLAG
         FROM dwi_tax_cit_tp_account_r_i t
         INNER JOIN tmp p ON p.PARENTID = t.account_ID
      )

      SELECT lv,
         '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
         account_ID,
         PARENTID,
         PATH_NUM
      FROM tmp
      WHERE DEL_FLAG='N'
   ) S
) S
```

修改点对比

 版权所有 © 华为云计算技术有限公司



```
INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)
SELECT
    S.account_id,S.parentid
FROM (SELECT
        DISTINCT S.ACCOUNT_ID, S.PARENTID
    FROM (
        WITH RECURSIVE tmp AS (
            SELECT 1 AS LV,
                '/'||T.PARENTID AS path,
                account_ID AS account_ID,
                PARENTID AS PARENTID ,
                ROW_number() over() PATH_NUM,
                T.DEL_FLAG
            FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
            WHERE T.DEL_FLAG = 'N'
            AND T.ISLEAF_FLAG = 1

            UNION ALL

            SELECT lv1v AS LV,
                p.path||'/'||t.PARENTID AS path,
                p.account_ID,
                t.PARENTID AS PARENTID,
                ROW_number() over() PATH_NUM,
                T.DEL_FLAG
            FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
            INNER JOIN tmp p ON p.PARENTID = t.account_ID
        )

        SELECT lv,
            '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
            account_ID,
            PARENTID,
            PATH_NUM
        FROM tmp
        WHERE DEL_FLAG='N'
    ) S
) S
```

```
CREATE TEMP TABLE DWI_TAX_CIT_TP_ACCOUNT_R_i
WITH(orientation=column)
TO GROUP "LC_DW1"
AS SELECT *
FROM dwtaxdi.DWI_TAX_CIT_TP_ACCOUNT_R_i;

INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)
SELECT
    S.account_id,S.parentid
FROM (SELECT
        DISTINCT S.ACCOUNT_ID, S.PARENTID
    FROM (
        WITH RECURSIVE tmp AS (
            SELECT 1 AS LV,
                '/'||T.PARENTID AS path,
                account_ID AS account_ID,
                PARENTID AS PARENTID ,
                ROW_number() over() PATH_NUM,
                T.DEL_FLAG
            FROM dwl_tax_cit_tp_account_r_i t
            WHERE T.DEL_FLAG = 'N'
            AND T.ISLEAF_FLAG = 1

            UNION ALL

            SELECT lv1v AS LV,
                p.path||'/'||t.PARENTID AS path,
                p.account_ID,
                t.PARENTID AS PARENTID,
                ROW_number() over() PATH_NUM,
                T.DEL_FLAG
            FROM dwl_tax_cit_tp_account_r_i t
            INNER JOIN tmp p ON p.PARENTID = t.account_ID
        )

        SELECT lv,
            '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
            account_ID,
            PARENTID,
            PATH_NUM
        FROM tmp
        WHERE DEL_FLAG='N'
    ) S
) S
```

优化后的执行计划

```
id |                                                                                operation                                                                                 |  E-rows  | E-distinct | E-memory | E-width | E-costs
---+------------------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+------------+----------+---------+---------
 1 | ->  Row Adapter                                                                                                                                                        |        1 |            |          |      64 | 92487.39
 2 |   ->  Vector Streaming (type: GATHER)                                                                                                                                  |        1 |            |          |      64 | 92487.39
 3 |     ->  Vector Insert on dwltax.dwl_tax_tom_cit_tp_account_tmp                                                                                                         |       72 |            |    16MB  |      64 | 92485.11
 4 |       ->  Vector Adapter                                                                                                                                               |       72 |            |     1MB  |      64 | 92483.10
 5 |         ->  Streaming(type: ROUNDROBIN)                                                                                                                                |       72 |            |     2MB  |      64 | 92483.10
 6 |           ->  HashAggregate                                                                                                                                            |        1 |            |    16MB  |      64 | 92480.84
 7 |             ->  Result                                                                                                                                                 |        1 |            |     1MB  |       0 | 92480.81
 8 |               ->  Streaming(type: BROADCAST)                                                                                                                           |        1 |            |     2MB  |       0 | 0.30
 9 |                 ->  CTE Scan on tmp                                                                                                                                    |        1 |            |     1MB  |       0 | 0.02
10 |                   ->  Recursive Union (11,15)                                                                                                                          |  2027304 |            |          |      78 | 92480.51
11 |                     ->  WindowAgg                                                                                                                                      |    33624 |            |          |      18 | 838.02
12 |                       ->  Streaming(type: BROADCAST)                                                                                                                   |  2421648 |            |          |      18 | 829.85
13 |                         ->  Row Adapter                                                                                                                                |    33624 |            |          |      18 | 534.32
14 |                           ->  CStore Scan on pg_temp_cn_5010_2869_708_140584651829760.dwi_tax_cit_tp_account_r_i t                                                     |    33634 |            |          |      18 | 534.32
15 |                     ->  WindowAgg                                                                                                                                      |   199368 |            |          |      78 | 9107.93
16 |                       ->  Streaming(type: BROADCAST) stream_level:1                                                                                                    | 14354280 |            |          |      78 | 9045.63
17 |                         ->  Hash Join (18,19) <<ruid:[10] ctlid:[10] (SYNC)>>                                                                                          |   199365 |            |          |      78 | 1452.52
18 |                           ->  WorkTable Scan on tmp p                                                                                                                  | 24209280 |  67248     |          |     100 | 38.64
19 |                           ->  Hash                                                                                                                                     |    39888 |  554       |          |      18 | 531.55
20 |                             ->  Row Adapter                                                                                                                            |    39888 |            |          |      18 | 531.55
21 |                               ->  CStore Scan on pg_temp_cn_5010_2869_708_140584651829760.dwi_tax_cit_tp_account_r_i t                                                 |    39873 |            |          |      18 | 531.55

   SQL Diagnostic Information
-------------------------------
CTE is inlined
    There are recursive CTEs in query
```

案例 2：把 WITH RECURSIVE 语句部分的结果转储到临时表

适用场景：当WITH RECURSIVE部分结果集比较小，且上一种整改方案的结果集仍比较大的时候，常见的有以下场景

- WITH RECURSIVE语句涉及到较多其它逻辑集中的表，转储比较麻烦。
- 当前语句没有直接调用WITH RECURSIVE语句，但是调用的视图中包含WITH RECURSIVE逻辑。

原始语句

```
INSERT INTO
dwltax.bif_rpt_item_htp_tax_t(PERIOD_ID,REPORT_ITEM_ID,CRT_CYCLE_ID,LAST_UPD_CYCLE_ID,UPD_JOB_IN
STANCE_ID, DEL_FLAG, CRT_JOB_INSTANCE_ID, UCCID )
SELECT
   DISTINCT T.PERIOD_ID ,158004503,-1,-1,-1,'N',-1,T.UCCID
FROM dwltax.dwl_tax_tom_taxpln_htp_tag_v T
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_account_v  TAX_E1502044 ON T.ACCOUNT_ID =
TAX_E1502044.ACCOUNT_ID
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_rf_tag_v  TAX_E1502045 ON T.RF_ID =  TAX_E1502045.RF_ID
WHERE  1 = 1
AND T.SOURCE_SYSTEM = 'HWHTPLOC01'
AND TAX_E1502044.PARENTID_ACCOUNT_CODE IN
('TARFAdjustmentsManual','TARFAdjustmentsExpenseDeferredReclass')
AND TAX_E1502045.PARENTID_RF_CODE = 'TARFTotal'
AND T.DATACATEGORY_CODE IN ('TARFDeferredNonCurrentLossNR')
AND T.PERIOD_ID=202308
```

分析发现如上语句的dwltax.dwl_tax_tom_cit_tp_rf_tag_v包含WITH RECURSIVE逻辑。

 版权所有 © 华为云计算技术有限公司

```
CREATE OR REPLACE VIEW dwltax.dwl_tax_tom_cit_tp_rf_tag_v AS  SELECT d2.rf_id AS parent_rf_id,
    d2.label AS parentid_rf_code, d1.rf_id, d1.label AS rf_code,
    d1.userdefined1, d1.userdefined2, d1.userdefined3, d1.en_us, d1.en_gb,
    d2.en_us AS parent_en_us, d2.en_gb AS parent_en_gb, d2.source_system
FROM (SELECT
        DISTINCT r1.rf_id, r1.parentid
    FROM (SELECT
                DISTINCT s.rf_id, s.parentid
            FROM (WITH RECURSIVE tmp AS (SELECT 1 AS lv,
                    '/'::text || t.parentid::text AS path,
                    t.rf_id, t.parentid
                FROM dwtaxdi.dwi_tax_cit_tp_rf_r_i t1
                RIGHT JOIN dwtaxdi.dwi_tax_cit_tp_rf_r_i t ON t.rf_id = t1.rf_id AND t.parentid = t1.parentid AND
t1.del_flag::text = 'N'::text
                WHERE t.del_flag::text = 'N'::text AND t.isleaf_flag = 1::numeric AND 1 = 1 AND 1 = 1 AND 1 = 1

                UNION ALL

                SELECT 1 + p.lv AS lv,
                    (p.path || '/'::text) || t.parentid::text AS path,
                    p.rf_id, t.parentid
                FROM tmp p, dwtaxdi.dwi_tax_cit_tp_rf_r_i t1
                RIGHT JOIN dwtaxdi.dwi_tax_cit_tp_rf_r_i t ON t.rf_id = t1.rf_id AND t.parentid = t1.parentid AND
t1.del_flag::text = 'N'::text
                WHERE p.parentid = t.rf_id AND t.del_flag::text = 'N'::text AND 1 = 1 AND 1 = 1 AND 1 = 1
            )

            SELECT (('/'::text || tmp.rf_id::text) || tmp.path) || '/'::text AS item_id_path,
                tmp.rf_id, tmp.parentid
            FROM tmp
        ) s

        UNION ALL

        SELECT r.rf_id, r.rf_id AS parentid
        FROM dwtaxdi.dwi_tax_cit_tp_rf_r_i r
        WHERE r.isleaf_flag = 1::numeric AND r.del_flag::text = 'N'::text) r1
) r, dwtaxdi.dwi_tax_cit_tp_rf_i d1, dwtaxdi.dwi_tax_cit_tp_rf_i d2
WHERE r.rf_id = d1.rf_id AND r.parentid = d2.rf_id;
```

视图dwltax.dwl_tax_tom_cit_tp_rf_tag_v的结果集只有1w行左右，可以尝试把事务dwltax.dwl_tax_tom_cit_tp_rf_tag_v的结果集转储到中间表，从而保证主查询可以下推。

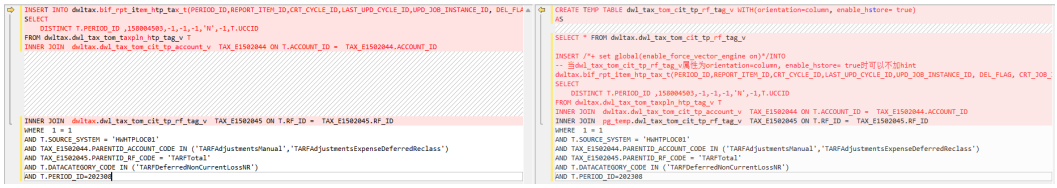
优化后语句：

```
-- 8.2.1及以上版本
CREATE TEMP TABLE dwl_tax_tom_cit_tp_rf_tag_v WITH(orientation=column, enable_hstore= true)
AS
SELECT * FROM dwltax.dwl_tax_tom_cit_tp_rf_tag_v

-- 8.2.0及以下版本
CREATE TEMP TABLE dwl_tax_tom_cit_tp_rf_tag_v WITH(orientation=row)
AS
SELECT * FROM dwltax.dwl_tax_tom_cit_tp_rf_tag_v

INSERT /*+ set global(enable_force_vector_engine on)*/INTO
-- 当dwl_tax_tom_cit_tp_rf_tag_v属性为orientation=column, enable_hstore= true时可以不加hint
dwltax.bif_rpt_item_htp_tax_t(PERIOD_ID,REPORT_ITEM_ID,CRT_CYCLE_ID,LAST_UPD_CYCLE_ID,UPD_JOB_IN
STANCE_ID, DEL_FLAG, CRT_JOB_INSTANCE_ID, UCCID )
SELECT
    DISTINCT T.PERIOD_ID ,158004503,-1,-1,-1,'N',-1,T.UCCID
FROM dwltax.dwl_tax_tom_taxpln_htp_tag_v T
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_account_v  TAX_E1502044 ON T.ACCOUNT_ID =
TAX_E1502044.ACCOUNT_ID
INNER JOIN  pg_temp.dwl_tax_tom_cit_tp_rf_tag_v  TAX_E1502045 ON T.RF_ID =  TAX_E1502045.RF_ID
WHERE  1 = 1
AND T.SOURCE_SYSTEM = 'HWHTPLOC01'
AND TAX_E1502044.PARENTID_ACCOUNT_CODE IN
```

 版权所有 © 华为云计算技术有限公司

```
('TARFAdjustmentsManual','TARFAdjustmentsExpenseDeferredReclass')
AND TAX_E1502045.PARENTID_RF_CODE = 'TARFTotal'
AND T.DATACATEGORY_CODE IN ('TARFDeferredNonCurrentLossNR')
AND T.PERIOD_ID=202308
```

修改点对比



16.5.4.9 案例：With-Recursive contains conflict distribution in none-recursive(xxx) recursive(xxx)

问题根因

非递归部分语句（UNION ALL前面部分）的执行计划为replicate计划，递归部分语句（UNION ALL后面部分）的执行计划为Hash计划，计划存在冲突。

处理方案：把非递归部分语句转储为临时表，促使临时表的分布形式和计划的分布式不一致即可。

案例 1：非递归部分语句的执行计划为 replicate 计划

原始语句

```
WITH RECURSIVE CTE AS (
   SELECT
      AOH.ORG_CODE,1 AS LEVEL,
      1 FLAG,
      LAST_UPDATE_DATE
   FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
   WHERE AOH.ORG_CODE IN (SELECT V.ORG_CODE
      FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 V
      WHERE V.ORG_CODE IS NOT NULL
      AND AOH.ORG_CODE = V.ORG_CODE
   )

   UNION ALL

   SELECT
     AOH.ORG_CODE,T.LEVEL + 1 AS LEVEL,
     1 FLAG,
     AOH.LAST_UPDATE_DATE
   FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
   INNER JOIN CTE T ON T.ORG_CODE = AOH.PARENT_ORG_CODE
)
SELECT
   ORG_CODE,
   MAX(LEVEL) -1 AS AOH_LEVEL,
   1 FLAG,
   LAST_UPDATE_DATE AS LAST_UPDATE_DATE1
FROM CTE
GROUP BY ORG_CODE, LAST_UPDATE_DATE
```

SQL自诊断信息

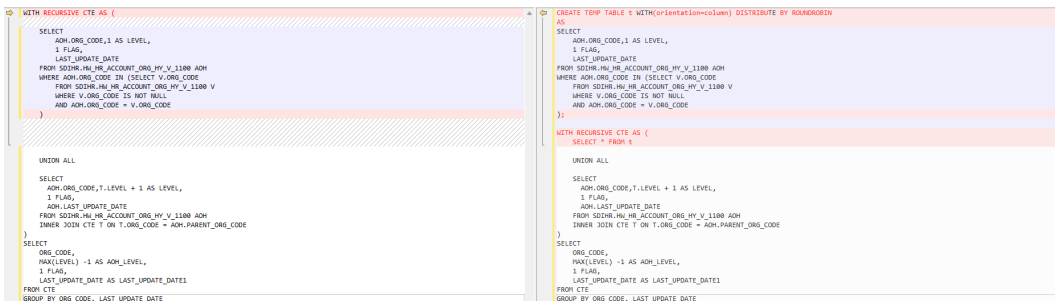
 版权所有 © 华为云计算技术有限公司

```
id |                                operation
---+-----------------------------------------------------------------------------------
 1 | ->  HashAggregate
 2 |   ->  CTE Scan on cte
 3 |   ->  Recursive Union  (4,7)  [2, CTE cte]
 4 |     ->  Subquery Scan on "*SELECT* 1"
 5 |       ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_"
 6 |         ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_"  [5, SubPlan 1]
 7 |     ->  Hash Join  (8,9)
 8 |       ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_"
 9 |       ->  Hash
10 |         ->  WorkTable Scan on cte t

                          SQL Diagnostic Information
-----------------------------------------------------------------------------------
SQL is not plan-shipping
    reason: With-Recursive contains conflict distribution in none-recursive(Replicate) recursive(Hash)
```

改写语句

```
CREATE TEMP TABLE t WITH(orientation=column) DISTRIBUTE BY ROUNDROBIN
AS
SELECT
   AOH.ORG_CODE,1 AS LEVEL,
   1 FLAG,
   LAST_UPDATE_DATE
FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
WHERE AOH.ORG_CODE IN (SELECT V.ORG_CODE
   FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 V
   WHERE V.ORG_CODE IS NOT NULL
   AND AOH.ORG_CODE = V.ORG_CODE
);

WITH RECURSIVE CTE AS (
   SELECT * FROM t

   UNION ALL

   SELECT
     AOH.ORG_CODE,T.LEVEL + 1 AS LEVEL,
     1 FLAG,
     AOH.LAST_UPDATE_DATE
   FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
   INNER JOIN CTE T ON T.ORG_CODE = AOH.PARENT_ORG_CODE
)
SELECT
   ORG_CODE,
   MAX(LEVEL) -1 AS AOH_LEVEL,
   1 FLAG,
   LAST_UPDATE_DATE AS LAST_UPDATE_DATE1
FROM CTE
GROUP BY ORG_CODE, LAST_UPDATE_DATE
```

修改点对比



 版权所有 © 华为云计算技术有限公司

16.5.4.10 案例：Table in TargetList can not be shipped

问题根因

表名或者表的别名出现在targetlist（输出列SELECT 和 FROM之间）中。

处理方案：改写SQL，消除对表名或者表的别名的直接引用。

案例 1：如场景描述，表 test 的别名 t，出现在 targetlist 中

原始语句

```
SELECT t, 1 FROM test t;
```

```
postgres=#  EXPLAIN VERBOSE SELECT t, 1 FROM test t;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                    operation                    | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on test "_REMOTE_TABLE_QUERY_" |     20 |            |      80 | 0.00

          SQL Diagnostic Information
-----------------------------------------------
SQL is not plan-shipping
       reason: Table in TargetList can not be shipped

                 Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------
  1 --Data Node Scan on test "_REMOTE_TABLE_QUERY_"
       Output: t.*, (1)
       Node/s: All datanodes (group_version1, bucket:16384)
       Remote query: SELECT t.*::public.test, 1 FROM ONLY public.test t WHERE true
```

改写语句：强转为text类型，或者优化SQL语句，输出列上去除表名或者表的别名。

```
SELECT t::text, 1 FROM test t;
```

```
postgres=#  EXPLAIN VERBOSE SELECT t::text, 1 FROM test t;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |           operation            | E-rows | E-distinct | E-memory | E-width | E-costs
----+--------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)   |     20 |            |          |      80 | 16.15
  2 |    ->  Seq Scan on public.test t |     20 |            | 1MB      |      80 | 10.15

       Targetlist Information (identified by plan id)
-------------------------------------------------------------
  1 --Streaming (type: GATHER)
       Output: ((t.*)::text), 1
       Node/s: All datanodes (group_version1, bucket:16384)
  2 --Seq Scan on public.test t
       Output: (t.*)::text, 1
```

16.5.4.11 案例："Row () Expression" can not be shipped

问题根因

语句中有row表达式，一般表现为在targetlist（输出列SELECT 和 FROM之间）上存在括号表达式"(.....)"，

版权所有 © 华为云计算技术有限公司

处理方案：改写SQL，消除row表达式。

案例 1：row 表达式(k4.businessID, arriveDate)出现在 targetlist 中

原始语句

```
SELECT
   date_format(arriveDate, '%Y-%m-%d') AS "date",
   sales,
   COUNT(DISTINCT (k4.businessID, arriveDate)/*row expression*/)
FROM (
   SELECT k1.businessID, k2.arriveDate, k1.sales
   FROM potential_customers k1
   LEFT JOIN flow_reception k2
      ON k1.businessID = k2.businessID
   WHERE k2.arriveDate BETWEEN '2024-08-31' AND '2024-09-02'
) k4
GROUP BY arriveDate, sales;
```

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
  id |                          operation                          | E-rows | E-distinct | E-memory | E-width | E-costs
 ----+-------------------------------------------------------------+--------+------------+----------+---------+---------
   1 | ->  GroupAggregate                                          |      2 |            |          |     104 | 20.63
   2 |    ->  Sort                                                 |      2 |            |          |      96 | 20.56
   3 |       ->  Subquery Scan on k4                               |      2 |            |          |      96 | 20.54
   4 |          ->  Streaming (type: SUBQUERY GATHER)              |      2 |            |          |      96 | 26.53
   5 |             ->  Nested Loop (6,8)                           |      2 |            | 1MB      |      96 | 20.53
   6 |                ->  Streaming(type: BROADCAST)               |      2 |            | 2MB      |      64 | 10.31
   7 |                   ->  Seq Scan on public.flow_reception k2  |      1 |            | 1MB      |      64 | 10.15
   8 |                ->  Seq Scan on public.potential_customers k1 |     20 |            | 1MB      |      64 | 10.10

          SQL Diagnostic Information
 -----------------------------------------------
 Partial SQL is not plan-shipping
        reason: "Row () Expression" can not be shipped

                          Predicate Information (identified by plan id)
 -------------------------------------------------------------------------------------------
   5 --Nested Loop (6,8)
        Join Filter: (k1.businessid = k2.businessid)
   7 --Seq Scan on public.flow_reception k2
        Filter: ((k2.arrivedate >= '2024-08-31'::text) AND (k2.arrivedate <= '2024-09-02'::text))
```

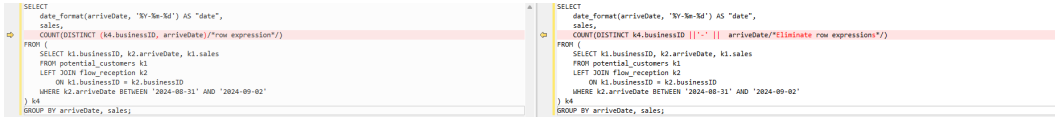
规避方案：不直接输出row表达式，因此改写row表达式为字符串拼接。

```
SELECT
   date_format(arriveDate, '%Y-%m-%d') AS "date",
   sales,
   COUNT(DISTINCT k4.businessID ||'-' ||  arriveDate/*Eliminate row expressions*/)
FROM (
   SELECT k1.businessID, k2.arriveDate, k1.sales
   FROM potential_customers k1
   LEFT JOIN flow_reception k2
      ON k1.businessID = k2.businessID
   WHERE k2.arriveDate BETWEEN '2024-08-31' AND '2024-09-02'
) k4
GROUP BY arriveDate, sales;
```

 版权所有 © 华为云计算技术有限公司

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 id |                       operation                        | E-rows | E-distinct | E-memory | E-width | E-costs
----+--------------------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                           |      2 |            |          |     112 | 26.71
  2 |    ->  HashAggregate                                   |      2 |            | 16MB     |     112 | 20.71
  3 |       ->  HashAggregate                                |      2 |            | 16MB     |     104 | 20.68
  4 |          ->  Streaming(type: REDISTRIBUTE)             |      2 |            | 2MB      |      96 | 20.64
  5 |             ->  Nested Loop (6,8)                      |      2 |            | 1MB      |      96 | 20.53
  6 |                ->  Streaming(type: BROADCAST)          |      2 |            | 2MB      |      64 | 10.31
  7 |                   ->  Seq Scan on public.flow_reception k2 |      1 |            | 1MB      |      64 | 10.15
  8 |                ->  Seq Scan on public.potential_customers k1 |     20 |            | 1MB      |      64 | 10.10

                      Predicate Information (identified by plan id)
-----------------------------------------------------------------------------------------------------------------------------------------
  5 --Nested Loop (6,8)
       Join Filter: (k1.businessid = k2.businessid)
  7 --Seq Scan on public.flow_reception k2
       Filter: ((k2.arrivedate >= '2024-08-31'::text) AND (k2.arrivedate <= '2024-09-02'::text))
```

修改点对比



16.5.4.12 案例：Merge Into can't be shipped in the sence temporarily

问题根因

MERGE INTO语句的USING子句部分还有非列表达式。

处理方案：改写SQL，去除非列表达式。

案例 1：子查询 d 中的 WHERE 条件含有 AND 1 = 2

原始语句

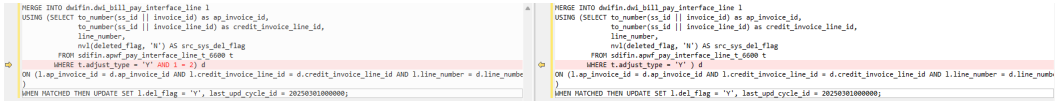
```
MERGE INTO dwifin.dwi_bill_pay_interface_line l
USING (SELECT to_number(ss_id || invoice_id) as ap_invoice_id,
              to_number(ss_id || invoice_line_id) as credit_invoice_line_id,
              line_number,
              nvl(deleted_flag, 'N') AS src_sys_del_flag
         FROM sdifin.apwf_pay_interface_line_t_6600 t
        WHERE t.adjust_type = 'Y' AND 1 = 2) d
ON (l.ap_invoice_id = d.ap_invoice_id AND l.credit_invoice_line_id = d.credit_invoice_line_id AND
l.line_number = d.line_number AND l.src_sys_del_flag = d.src_sys_del_flag AND l.ss_id = 6600
)
WHEN MATCHED THEN UPDATE SET l.del_flag = 'Y', last_upd_cycle_id = 20250301000000;
```

规避方案：这种情况一般属于SQL语句书写错误，去掉条件“AND 1 = 2”即可。

```
MERGE INTO dwifin.dwi_bill_pay_interface_line l
USING (SELECT to_number(ss_id || invoice_id) as ap_invoice_id,
              to_number(ss_id || invoice_line_id) as credit_invoice_line_id,
              line_number,
              nvl(deleted_flag, 'N') AS src_sys_del_flag
         FROM sdifin.apwf_pay_interface_line_t_6600 t
        WHERE t.adjust_type = 'Y' ) d
ON (l.ap_invoice_id = d.ap_invoice_id AND l.credit_invoice_line_id = d.credit_invoice_line_id AND
l.line_number = d.line_number AND l.src_sys_del_flag = d.src_sys_del_flag AND l.ss_id = 6600
)
WHEN MATCHED THEN UPDATE SET l.del_flag = 'Y', last_upd_cycle_id = 20250301000000;
```

修改点对比

 版权所有 © 华为云计算技术有限公司



16.5.4.13 案例：Type of Record in dual that is not a real table can not be shipped

问题根因

优化器识别到的record列，且record列对应的主表不是一个实体表。最常见的情况为MERGE INTO语句的USING部分使用了FROM dual字句导致问题出现。

处理方案：改写SQL，消除语句中的dual子句。

案例 1：子查询 f 为(SELECT 0 AS inv_id,0 AS ovs_pack_num,0 AS ovs_pack_id FROM DUAL)

原始语句

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
   SELECT
      f.inv_id,
      d.ovs_pack_id
   FROM (SELECT 0 AS inv_id,0 AS ovs_pack_num,0 AS ovs_pack_id FROM DUAL) f,
   dwicommon.dwi_inv_oversea_pack d
   WHERE f.ovs_pack_num = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```

```
                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 id |                                operation                                | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------------------------------+--------+------------+---------+--------
  1 | ->  Merge on dwiscm.dwi_inv_balance_onhand t                            |      1 |            |    3404 | 0.05
  2 |    ->  Nested Loop (3,6)                                               |      1 |            |    3404 | 0.05
  3 |       ->  Nested Loop (4,5)                                            |      1 |            |    3372 | 0.03
  4 |          ->  Data Node Scan on dwi_inv_balance_onhand "_REMOTE_TABLE_QUERY_" |      1 |            |    3372 | 0.00
  5 |          ->  Result                                                     |      1 |            |       0 | 0.01
  6 |       ->  Data Node Scan on dwi_inv_oversea_pack "_REMOTE_TABLE_QUERY_"  |      1 |            |      32 | 0.00

                     SQL Diagnostic Information
-----------------------------------------------------------------------------
SQL is not plan-shipping
       reason: Type of Record in dual that is not a real table can not be shipped
```

改写语句

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
   SELECT
      0 AS inv_id,
      d.ovs_pack_id
   FROM dwicommon.dwi_inv_oversea_pack d
   WHERE 0 = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```


版权所有 © 华为云计算技术有限公司

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 id |                           operation                                | E-rows | E-distinct |   E-memory   | E-width | E-costs
----+--------------------------------------------------------------------+--------+------------+--------------+---------+---------
  1 | ->  Row Adapter                                                    |      1 |            |              |    3414 | 25.78
  2 |   ->  Vector Streaming (type: GATHER)                              |      1 |            |              |    3414 | 25.78
  3 |     ->  Vector Merge on dwiscm.dwi_inv_balance_onhand t            |      2 |            | 36MB(3035MB) |    3414 | 23.28
  4 |       ->  Vector Nest Loop (5,6)                                   |      2 |            | 1MB          |    3414 | 23.28
  5 |         ->  CStore Scan on dwiscm.dwi_inv_balance_onhand t         |      1 |            | 1MB          |    3372 | 10.04
  6 |         ->  Vector Materialize                                     |      2 |            | 16MB         |      42 | 13.24
  7 |           ->  Vector Streaming(type: BROADCAST)                    |      2 |            | 2MB          |      42 | 13.24
  8 |             ->  CStore Scan on dwicommon.dwi_inv_oversea_pack d    |      1 |            | 1MB          |      42 | 13.08

    Predicate Information (identified by plan id)
-----------------------------------------------------------
  5 --CStore Scan on dwiscm.dwi_inv_balance_onhand t
      Filter: (t.inv_id = 0::numeric)
      Pushdown Predicate Filter: (t.inv_id = 0::numeric)
  8 --CStore Scan on dwicommon.dwi_inv_oversea_pack d
      Filter: (0 = (d.ovs_pack_num)::bigint)
```

修改点对比

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
    SELECT
        f.inv_id,
        d.ovs_pack_id
    FROM (SELECT 0 AS inv_id,0 AS ovs_pack_num,0 AS ovs_pack_id FROM DUAL) f,
    dwicommon.dwi_inv_oversea_pack d
    WHERE f.ovs_pack_num = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
    SELECT
        0 AS inv_id,
        d.ovs_pack_id
    FROM dwicommon.dwi_inv_oversea_pack d
    WHERE 0 = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```

16.5.5 案例：设置 cost_param 对查询性能优化

cost_param参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

场景一：优化前

cost_param的bit0（set cost_param=1）值为1时，表示对于求!=连接的选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确。下面查询的例子是cost_param的bit0为1时的优化场景。V300R002C00版本开始已弃用cost_param & 1不为0时的路径，默认选择已优化的估算公式。

说明

选择率是两表join时，满足join条件的行数在join结果集中所占的比率。

表结构如下所示：

```
CREATE TABLE LINEITEM
(
L_ORDERKEY BIGINT NOT NULL
, L_PARTKEY BIGINT NOT NULL
, L_SUPPKEY BIGINT NOT NULL
, L_LINENUMBER BIGINT NOT NULL
, L_QUANTITY DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL
, L_DISCOUNT DECIMAL(15,2) NOT NULL
, L_TAX DECIMAL(15,2) NOT NULL
, L_RETURNFLAG CHAR(1) NOT NULL
, L_LINESTATUS CHAR(1) NOT NULL
, L_SHIPDATE DATE NOT NULL
, L_COMMITDATE DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE CHAR(10) NOT NULL
, L_COMMENT VARCHAR(44) NOT NULL
) with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
```

 版权所有 © 华为云计算技术有限公司

```
(
O_ORDERKEY BIGINT NOT NULL
, O_CUSTKEY BIGINT NOT NULL
, O_ORDERSTATUS CHAR(1) NOT NULL
, O_TOTALPRICE DECIMAL(15,2) NOT NULL
, O_ORDERDATE DATE NOT NULL
, O_ORDERPRIORITY CHAR(15) NOT NULL
, O_CLERK CHAR(15) NOT NULL
, O_SHIPPRIORITY BIGINT NOT NULL
, O_COMMENT VARCHAR(79) NOT NULL
)with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
order by
numwait desc;
```

执行计划如下图所示（verbose条件下，新增distinct列，受cost off/on控制，hashjoin行显示内外表的distinct估值，其他行为空）：

```
 id |                         operation                          | E-rows | E-distinct | E-width | E-costs
----+------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Row Adapter                                            |      1 |            |       8 | 39.36
  2 |    ->  Vector Sort                                         |      1 |            |       8 | 39.36
  3 |       ->  Vector Aggregate                                 |      1 |            |       8 | 39.34
  4 |          ->  Vector Streaming (type: GATHER)               |      2 |            |       8 | 39.34
  5 |             ->  Vector Aggregate                           |      2 |            |       8 | 39.25
  6 |                ->  Vector Hash Anti Join (7, 10)           |      2 | 4, 5       |       0 | 39.24
  7 |                   ->  Vector Hash Join (8,9)               |      2 | 200, 1     |      16 | 26.12
  8 |                      ->  CStore Scan on public.lineitem l1 |      7 |            |      16 | 13.05
  9 |                      ->  CStore Scan on public.orders      |      1 |            |       8 | 13.05
 10 |                   ->  CStore Scan on public.lineitem l3    |      7 |            |      16 | 13.05
```

场景一：优化后

以上查询为lineitem表自连接的Anti Join，当使用cost_param的bit0为0时，估算Anti Join的行数与实际行数相差很大，导致查询性能下降。可以通过设置cost_param的bit0为1时，使Anti Join的行数估算更准确，从而提高查询性能。优化后的执行计划如下：

版权所有 © 华为云计算技术有限公司

id	operation	E-rows	E-memory	E-width	E-costs
1	-> Row Adapter	1		0	9104892.37 9
2	-> Vector Sort	1		0	9104892.37 9
3	-> Vector Aggregate	1		0	9104892.35 8
4	-> Vector Streaming (type: GATHER)	48		0	9104892.35 8
5	-> Vector Aggregate	48	1MB	0	9104890.82 5
6	-> Vector Hash Join (7.12)	2526630903	929MB	0	8973295.45 4
7	-> Vector Hash Anti Join (8. 10)	1999996587	3178MB	8	7198231.14
8	-> Vector Partition Iterator	1999996587	1MB	16	3000158.25
9	-> Partitioned CStore Scan on public.lineitem 11	1999996587	1MB	16	3000158.25 1
10	-> Vector Partition Iterator	1999996587	1MB	16	3000158.25
11	-> Partitioned CStore Scan on public.lineitem 13	1999996587	1MB	16	3000158.25
12	-> Vector Partition Iterator	730839014	1MB	8	589611.00
13	-> Partitioned CStore Scan on public.orders	730839014	1MB	8	589611.00

场景二：优化前

当cost_param的bit1（set cost_param=2）为1时，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确。下面查询的例子是cost_param的bit1为1时的优化场景。

表结构如下所示：

```
CREATE TABLE NATION
(
N_NATIONKEYINT NOT NULL
, N_NAMECHAR(25) NOT NULL
, N_REGIONKEYINT NOT NULL
, N_COMMENTVARCHAR(152)
) distribute by replication;
CREATE TABLE SUPPLIER
(
S_SUPPKEYBIGINT NOT NULL
, S_NAMECHAR(25) NOT NULL
, S_ADDRESSVARCHAR(40) NOT NULL
, S_NATIONKEYINT NOT NULL
, S_PHONECHAR(15) NOT NULL
, S_ACCTBALDECIMAL(15,2) NOT NULL
, S_COMMENTVARCHAR(101) NOT NULL
) distribute by hash(S_SUPPKEY);
CREATE TABLE PARTSUPP
(
PS_PARTKEYBIGINT NOT NULL
, PS_SUPPKEYBIGINT NOT NULL
, PS_AVAILQTYBIGINT NOT NULL
, PS_SUPPLYCOSTDECIMAL(15,2)NOT NULL
, PS_COMMENTVARCHAR(199) NOT NULL
)distribute by hash(PS_PARTKEY);
```

查询语句如下所示：

```
set cost_param=2;
explain verbose select
nation,
sum(amount) as sum_profit
from
(
select
n_name as nation,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
from
supplier,
lineitem,
partsupp,
nation
where
s_suppkey = l_suppkey
```


版权所有 © 华为云计算技术有限公司

```
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and s_nationkey = n_nationkey
) as profit
group by nation
order by nation;
```

当cost_param的bit1为0时，执行计划如下图所示：

```
 id |                            operation                            | E-rows | E-distinct | E-width | E-costs
----+-----------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Sort                                                        |      1 |            |     208 | 61.52
  2 |    ->  HashAggregate                                            |      1 |            |     208 | 61.51
  3 |       ->  Streaming (type: GATHER)                              |      2 |            |     208 | 61.51
  4 |          ->  HashAggregate                                      |      2 |            |     208 | 61.36
  5 |             ->  Hash Join (6,7)                                 |      2 | 20, 15     |     176 | 61.33
  6 |                ->  Seq Scan on public.nation                    |     40 |            |     108 | 20.20
  7 |                ->  Hash                                         |      2 |            |      76 | 41.04
  8 |                   ->  Hash Join (9,16)                          |      2 | 10, 13     |      76 | 41.04
  9 |                      ->  Streaming(type: REDISTRIBUTE)          |      2 |            |      88 | 27.73
 10 |                         ->  Hash Join (11,14)                   |      2 | 10, 13     |      88 | 27.62
 11 |                            ->  Streaming(type: REDISTRIBUTE)    |     20 |            |      70 | 14.19
 12 |                               ->  Row Adapter                   |     21 |            |      70 | 13.01
 13 |                                  ->  CStore Scan on public.lineitem |     20 |            |      70 | 13.01
 14 |                            ->  Hash                             |     21 |            |      34 | 13.13
 15 |                               ->  Seq Scan on public.partsupp   |     20 |            |      34 | 13.13
 16 |                      ->  Hash                                   |     21 |            |      12 | 13.13
 17 |                         ->  Seq Scan on public.supplier         |     20 |            |      12 | 13.13
```

场景二：优化后

在以上查询中，supplier、lineitem、partsupp三表做hashjoin的条件为(lineitem.l_suppkey = supplier.s_suppkey) AND (lineitem.l_partkey = partsupp.ps_partkey)，此hashjoin条件中存在两个过滤条件，这前一个过滤条件中的lineitem.l_suppkey和后一个过滤条件中的lineitem.l_partkey同为lineitem表的两列，这两列存在强相关的关联关系。在这种情况，估算hashjoin条件的选择率时，如果使用cost_param的bit1为0时，实际是将AND的两个过滤条件分别计算的2个选择率的值相乘来得到hashjoin条件的选择率，导致行数估算不准确，查询性能较差。所以需要将cost_param的bit1为1时，选择最小的选择率作为总的选择率估算行数比较准确，查询性能较好，优化后的计划如下图所示：

```
 id |                            operation                            | E-rows | E-distinct | E-width | E-costs
----+-----------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Sort                                                        |     10 |            |     208 | 64.42
  2 |    ->  HashAggregate                                            |     10 |            |     208 | 64.23
  3 |       ->  Streaming (type: GATHER)                              |     20 |            |     208 | 64.23
  4 |          ->  HashAggregate                                      |     20 |            |     208 | 62.71
  5 |             ->  Hash Join (6,7)                                 |     20 | 20, 10     |     176 | 62.46
  6 |                ->  Seq Scan on public.nation                    |     40 |            |     108 | 20.20
  7 |                ->  Hash                                         |     20 |            |      76 | 41.97
  8 |                   ->  Hash Join (9,16)                          |     20 | 10, 13     |      76 | 41.97
  9 |                      ->  Streaming(type: REDISTRIBUTE)          |     20 |            |      82 | 28.54
 10 |                         ->  Hash Join (11,14)                   |     20 | 10, 13     |      82 | 27.63
 11 |                            ->  Streaming(type: REDISTRIBUTE)    |     20 |            |      70 | 14.19
 12 |                               ->  Row Adapter                   |     21 |            |      70 | 13.01
 13 |                                  ->  CStore Scan on public.lineitem |     20 |            |      70 | 13.01
 14 |                            ->  Hash                             |     21 |            |      12 | 13.13
 15 |                               ->  Seq Scan on public.supplier   |     20 |            |      12 | 13.13
 16 |                      ->  Hash                                   |     21 |            |      34 | 13.13
 17 |                         ->  Seq Scan on public.partsupp         |     20 |            |      34 | 13.13
```

16.5.6 案例：调整局部聚簇键

局部聚簇 (Partial Cluster Key，简称PCK)，列存储下一种通过min/max稀疏索引实现基表快速扫描的索引技术。Partial Cluster Key可以指定多列，但是一般不建议超过2列。PCK适用于列存大表点查询加速。

优化前

创建一个无局部聚簇（以下称为PCK）的列存表orders_no_pck，表定义如下：

 版权所有 © 华为云计算技术有限公司

```
                                pg_get_tabledef
-------------------------------------------------------------------------------
SET search_path = dbadmin;                                                    +
CREATE  TABLE orders_no_pck (                                                 +
        o_orderkey bigint NOT NULL,                                           +
        o_custkey bigint NOT NULL,                                            +
        o_orderstatus character(1) NOT NULL,                                  +
        o_totalprice numeric(15,2) NOT NULL,                                  +
        o_orderdate timestamp(0) without time zone NOT NULL,                  +
        o_orderpriority character(15) NOT NULL,                               +
        o_clerk character(15) NOT NULL,                                       +
        o_shippriority bigint NOT NULL,                                       +
        o_comment character varying(79) NOT NULL                              +
)                                                                             +
WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)+
DISTRIBUTE BY HASH(o_orderkey)                                                +
TO GROUP group_version1;
(1 row)
```

执行以下SQL语句，查询某个点查询的执行计划：

```
EXPLAIN PERFORMANCE
SELECT * FROM orders_no_pck
WHERE o_orderkey = '13095143'
ORDER BY o_orderdate;
```

由下图可知执行时间为48毫秒，查看Datanode Information发现filter时间为19毫秒，CUNone比例为0。

```
gaussdb=>  EXPLAIN PERFORMANCE
gaussdb-> SELECT * FROM orders_no_pck
gaussdb-> WHERE o_orderkey = '13095143'
gaussdb-> ORDER BY o_orderdate;
                                                                
```

```
                                pg_get_tabledef
-------------------------------------------------------------------------------
 SET search_path = dbadmin;                                                   +
 CREATE  TABLE orders_pck (                                                   +
         o_orderkey bigint NOT NULL,                                          +
         o_custkey bigint NOT NULL,                                           +
         o_orderstatus character(1) NOT NULL,                                 +
         o_totalprice numeric(15,2) NOT NULL,                                 +
         o_orderdate timestamp(0) without time zone NOT NULL,                 +
         o_orderpriority character(15) NOT NULL,                              +
         o_clerk character(15) NOT NULL,                                      +
         o_shippriority bigint NOT NULL,                                      +
         o_comment character varying(79) NOT NULL                             +
 )                                                                            +
 WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)+
 DISTRIBUTE BY HASH(o_orderkey)                                               +
 TO GROUP group_version1;
(1 row)
```

使用ALTER TABLE将字段o_orderkey设置为PCK：

```
postgres=> ALTER TABLE orders_pck ADD PARTIAL CLUSTER KEY(o_orderkey);
ALTER TABLE
```

执行以下SQL语句，再次查询同样的点查询SQL语句的执行计划：

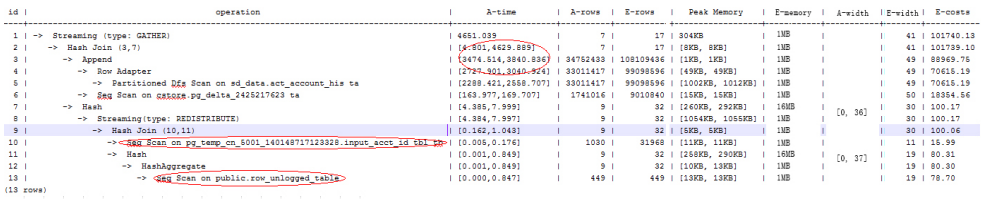
```
EXPLAIN PERFORMANCE
SELECT * FROM orders_pck
WHERE o_orderkey = '13095143'
ORDER BY o_orderdate;
```

由下图可知执行时间为5毫秒，查看Datanode Information发现filter时间为0.5毫秒，CUNone比例为82。CUNone比例越高，PCK的性能收益越明显。

```
gaussdb=> EXPLAIN PERFORMANCE
gaussdb-> SELECT * FROM orders_pck
gaussdb-> WHERE o_orderkey = '13095143'
gaussdb-> ORDER BY o_orderdate;
                                                                
```

优化前

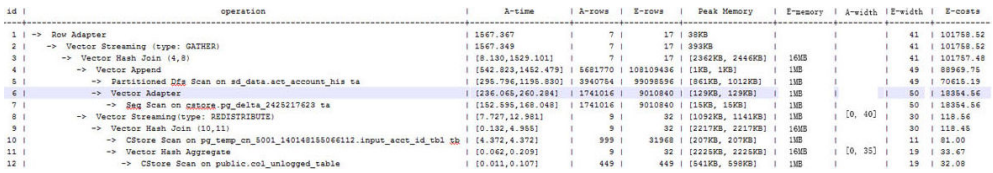
某局点测试过程遇到如下的执行计划，客户希望将性能提升至3s内返回结果。



优化后

经过分析发现计划走了行引擎。根本原因是：临时计划表input_acct_id_tbl和中间结果转储表row_unlogged_table使用了行存表。

修改这两个表为列存表之后，性能提升至1.6s。



16.5.8 案例：改建分区表

逻辑上的一张表根据某种策略分成多个物理块进行存储，这张逻辑上的表称之为分区表，每个物理块则称为一个分区。一般对数据和查询都有明显区间段特征的表使用分区策略可通过较小不必要的数据扫描，从而提升查询性能

在查询时，可通过分区剪枝技术尽可能减少底层数据扫描，即缩小表的扫描范围。分区剪枝是指对于分区表或分区索引来说，优化器可以自动从FROM和WHERE子句里根据分区键提取出需要扫描的分区，从而避免全表扫描，减少扫描的数据块，提高性能。

优化前

创建一个非分区表orders_no_part，表定义如下：

```
                                pg_get_tabledef
--------------------------------------------------------------------------------
SET search_path = dbadmin;
CREATE  TABLE orders_no_part (
        o_orderkey bigint NOT NULL,
        o_custkey bigint NOT NULL,
        o_orderstatus character(1) NOT NULL,
        o_totalprice numeric(15,2) NOT NULL,
        o_orderdate timestamp(0) without time zone NOT NULL,
        o_orderpriority character(15) NOT NULL,
        o_clerk character(15) NOT NULL,
        o_shippriority bigint NOT NULL,
        o_comment character varying(79) NOT NULL
)
WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)
DISTRIBUTE BY HASH(o_orderkey)
TO GROUP group_version1;
(1 row)
```

执行以下SQL语句查询非分区表的执行计划：

版权所有 © 华为云计算技术有限公司

```
EXPLAIN PERFORMANCE
SELECT count(*) FROM orders_no_part WHERE
o_orderdate >= '1996-01-01 00:00:00'::timestamp(0);
```

由下图可知执行时间为73毫秒，其中全表扫描的时间为44~45毫秒。

```
gaussdb=> EXPLAIN PERFORMANCE
gaussdb-> SELECT count(*) FROM orders_no_part WHERE
gaussdb-> o_orderdate >= '1996-01-01 00:00:00'::timestamp(0);
                                                
```

场景一：setA是setB的一个子集。

对于这种场景，直接对下层结果集进行汇聚的结果就是正确的汇聚结果，上层算子直接使用即可。如下图所示：

```
explain select a, count(1) from t1 group by a;
 id |          operation           | E-rows | E-width | E-costs
----+------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER) |     30 |       4 | 15.56
  2 |    ->  HashAggregate         |     30 |       4 | 14.31
  3 |       ->  Seq Scan on t1     |     30 |       4 | 14.14
(3 rows)
```

场景二：setA不是setB的一个子集。

对于这种场景，Stream执行框架分为如下三种计划形态：

hashagg+gather(redistribute)+hashagg

redistribute+hashagg(+gather)

hashagg+redistribute+hashagg(+gather)

DWS提供了guc参数best_agg_plan来干预执行计划，强制其生成上述对应的执行计划，此参数取值范围为0，1，2，3

- 取值为1时，强制生成第一种计划。
- 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
- 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
- 取值为0时，优化器会根据以上三种计划的估算代价选择最优的一种计划生成。

具体影响如下：

```
set best_agg_plan to 1;
SET
explain select b,count(1) from t1 group by b;
 id |           operation            | E-rows | E-width | E-costs
----+--------------------------------+--------+---------+---------
  1 | ->  HashAggregate              |      8 |       4 | 15.83
  2 |    ->  Streaming (type: GATHER) |     25 |       4 | 15.83
  3 |       ->  HashAggregate        |     25 |       4 | 14.33
  4 |          ->  Seq Scan on t1    |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 2;
SET
explain select b,count(1) from t1 group by b;
 id |               operation                | E-rows | E-width | E-costs
----+----------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)           |     30 |       4 | 15.85
  2 |    ->  HashAggregate                   |     30 |       4 | 14.60
  3 |       ->  Streaming(type: REDISTRIBUTE) |     30 |       4 | 14.45
  4 |          ->  Seq Scan on t1            |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 3;
SET
explain select b,count(1) from t1 group by b;
 id |               operation                | E-rows | E-width | E-costs
----+----------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)           |     30 |       4 | 15.84
  2 |    ->  HashAggregate                   |     30 |       4 | 14.59
  3 |       ->  Streaming(type: REDISTRIBUTE) |     25 |       4 | 14.59
  4 |          ->  HashAggregate             |     25 |       4 | 14.33
```

 版权所有 © 华为云计算技术有限公司

```
 5 |          ->  Seq Scan on t1          |     30 |       4 | 14.14
(5 rows)
```

总结

通常优化器总会选择最优的执行计划，但是众所周知代价估算，尤其是中间结果集的代价估算一般会有比较大的偏差，这种比较大的偏差就可能会导致agg的计算方式出现比较大的偏差，这时候就需要通过best_agg_plan进行agg计算模型的干预。

一般来说，当agg汇聚的收敛度很小时，即结果集的个数在agg之后并没有明显变少时（经验上以5倍为临界点），选择redistribute+hashagg执行方式，否则选择hashagg+redistribute+hashagg执行方式。

16.5.10 案例：改写 SQL 排除剪枝干扰

分区表查询中表达式一般不是单纯的分区键，而是包含分区键的表达式的Filter条件，这种类型的Filter条件是不能用来剪枝的。

优化前

t_ddw_f10_op_cust_asset_mon为分区表，分区键为year_mth，此字段是由年月两个值拼接而成的整数。

测试SQL如下：

```
SELECT
    count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth  < substr('20200722',1 ,6 )
AND b1.year_mth + 1 >= substr('20200722',1 ,6 );
```

测试结果显示此SQL的表Scan耗时长达10s，查询SQL语句的执行计划如下

```
EXPLAIN (ANALYZE ON, VERBOSE ON)
SELECT
    count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth < substr('20200722',1 ,6 )
AND b1.year_mth + 1 >= cast(substr('20200722',1 ,6 ) AS int);
                                                
```

```
-------------------------------------------------------------------------------
  4 --Partition Iterator
        Iterations: 6
  5 --Partitioned Seq Scan on public.t_ddw_f10_op_cust_asset_mon b1
        Filter: ((b1.year_mth < 202007::bigint) AND ((b1.year_mth + 1) >= 202007))
        Rows Removed by Filter: 81920000
        Partitions Selected by Static Prune: 1..6
```

优化后

分析语句的执行计划，查看执行计划中的SQL自诊断信息，发现如下诊断信息：

```
                          SQL Diagnostic Information
-----------------------------------------------------------------------------------------------
Partitioned table unprunable Qual
        table public.t_ddw_f10_op_cust_asset_mon b1:
        left side of expression "((year_mth + 1) > 202008)" invokes function-call/type-conversion
```

Filter条件中存在表达式(year_mth + 1) > 202008，这种**表达式一侧不是单纯的分区键、而是包含分区键的表达式的Filter条件是不能用来剪枝的**，因而导致查询语句扫描了几乎整个分区表的数据。

跟原始SQL语句对比，可以确定表达式 '(year_mth + 1) > 202008' 是从表达式 'b1.year_mth + 1 > substr('20200822',1 ,6)' 衍生而来，按照诊断信息把修改SQL语句为如下方式：

```
SELECT
    count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth <= substr('20200822',1 ,6 )
AND b1.year_mth > cast(substr('20200822',1 ,6 ) AS int) - 1;
```

改写之后，SQL语句的执行信息如下，可以看到不剪枝告警已经消除，剪枝后需要扫描分区数为1，执行时间从10s提升至3s。

```
EXPLAIN (analyze ON, verbose ON)
SELECT
    count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth < substr('20200722',1 ,6 )
AND b1.year_mth >= cast(substr('20200722',1 ,6 ) AS int) - 1;
                                                
```

16.5.11 案例：改写 SQL 消除 in-clause

优化前

in-clause/any-clause是常见的SQL语句约束条件，有时in或any后面的clause都是常量，类似于：

```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in (‘20120405’, ‘20130405’);
```

或者

```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in any(‘20120405’, ‘20130405’);
```

但是也有一些如下的特殊用法：

```
SELECT
ls_pid_cusr1,COALESCE(max(round((current_date-bthdate)/365)),0)
FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
WHERE t1.ls_pid_cusr1 = any(values(id),(id15))
GROUP BY ls_pid_cusr1;
```

其中，id、id15为p10_md_tmp_t2中的两列，“t1.ls_pid_cusr1 = any(values(id),(id15))”等价于“t1.ls_pid_cusr1 = id or t1.ls_pid_cusr1 = id15”。

因此join-condition实质上是一个不等式，这种不等值的join操作必须走nestloop，对应执行计划如下：

```
Streaming (type: GATHER)  (cost=1641429284.14..1641429523.98 rows=3840 width=49)
  Node/s: All datanodes
  ->  Insert on channel.calc_empfyc_c1_result_age_tmp  (cost=1641429280.14..1641429283.98 rows=3840 width=49)
        ->  HashAggregate  (cost=1641429280.14..1641429283.38 rows=3840 width=25)
              Output: t1.ls_pid_cusr1, COALESCE(max((max(round((((('2017-03-29 00:00:00'::timestamp without time zone - t2.bthdate) / 365::double precision))::numeric, 0)))), 0::numeric)
              Group By Key: t1.ls_pid_cusr1
              ->  Streaming(type: REDISTRIBUTE)  (cost=820714640.07..820714642.69 rows=3968 width=25)
                    Output: t1.ls_pid_cusr1, (max(round((((('2017-03-29 00:00:00'::timestamp without time zone - t2.bthdate) / 365::double precision))::numeric, 0)))
                    Distribute Key: t1.ls_pid_cusr1
                    Spawn on: All datanodes
                    ->  HashAggregate  (cost=820714640.07..820714640.69 rows=3968 width=25)
                          Output: t1.ls_pid_cusr1, max(round((((('2017-03-29 00:00:00'::timestamp without time zone - t2.bthdate) / 365::double precision))::numeric, 0))
                          Group By Key: t1.ls_pid_cusr1
                          ->  Nested Loop  (cost=0.00..615567760.93 rows=875293350960 width=25)
                                Output: t1.ls_pid_cusr1, t2.bthdate
                                Join Filter: (SubPlan 1)
                                ->  Seq Scan on channel.p10_md_tmp_t2 t2  (cost=0.00..127030.52 rows=443523360 width=64)
                                      Output: t2.id, t2.id15, t2.bthdate, t2.mandeg
                                ->  Materialize  (cost=0.00..147.29 rows=252608 width=17)
                                      Output: t1.ls_pid_cusr1
                                      ->  Streaming(type: BROADCAST)  (cost=0.00..127.56 rows=252608 width=17)
                                            Output: t1.ls_pid_cusr1
                                            Spawn on: All datanodes
                                            ->  Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1  (cost=0.00..1.62 rows=3947 width=17)
                                                  Output: t1.ls_pid_cusr1
                                SubPlan 1
                                  ->  Values Scan on "*VALUES*"  (cost=0.00..0.01 rows=64 width=38)
                                        Output: "*VALUES*".column1
```

优化后

测试发现由于两表结果集过大，导致nestloop耗时过长，超过一小时未返回结果，因此性能优化的关键是消除nestloop，让join走更高效的hashjoin。从语义等价的角度消除any-clause，SQL改写如下：

```
select
ls_pid_cusr1,COALESCE(max(round(ym/365)),0)
from
(
	(
		SELECT
			ls_pid_cusr1,(current_date-bthdate) as ym
		FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
		WHERE t1.ls_pid_cusr1 = t2.id and t1.ls_pid_cusr1 != t2.id15
	)
```


版权所有 © 华为云计算技术有限公司

```
        union all
        (
                SELECT
                        ls_pid_cusr1,(current_date-bthdate) as ym
                FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
                WHERE t1.ls_pid_cusr1 = id15
        )
)
GROUP BY ls_pid_cusr1;
```

注意：尽量使用union all代替union。union在合并两个集合时会执行去重操作，而union all则直接将两个结果集合并、不执行去重。执行去重会消耗大量的时间，因此，在一些实际应用场景中，如果通过业务逻辑已确认两个集合不存在重叠，可用union all替代union以便提升性能。

优化后的SQL查询由两个等值join的子查询构成，而每个子查询都可以走更适合此场景的hashjoin。优化后的执行计划如下

```
 id |                                  operation                                  |        A-time        |  A-rows   |  E-rows   |   Peak Memory    | E-memory |  A-width
----+-----------------------------------------------------------------------------+----------------------+-----------+-----------+------------------+----------+-----------
  1 | ->  Streaming (type: GATHER)                                                | 6737.281             |         0 |       192 | 292KB            |          |
  2 |   ->  Insert on channel.calc_empfyc_c1_result_age_tmp                       | [4665.024,4990.666]  |         0 |       192 | [1108KB, 1108KB] | 1MB      |
  3 |     ->  HashAggregate                                                       | [4664.996,4990.641]  |         0 |       192 | [12KB, 12KB]     | 16MB     |
  4 |       ->  Streaming(type: REDISTRIBUTE)                                     | [4664.991,4990.637]  |         0 |      3392 | [2090KB, 2090KB] | 1MB      |
  5 |         ->  HashAggregate                                                   | [3416.939,4958.348]  |         0 |      3392 | [14KB, 14KB]     | 16MB     |
  6 |           ->  Append                                                        | [3416.936,4958.340]  |         0 |      4011 | [1KB, 1KB]       | 1MB      |
  7 |             ->  Hash Join (8,9)                                             | [2011.226,3080.697]  |         0 |      3947 | [6KB, 6KB]       | 1MB      |
  8 |               ->  Seq Scan on channel.p10_md_tmp_t2 t2                      | [803.782,1238.984]   | 443525717 | 443523360 | [12KB, 12KB]     | 1MB      |
  9 |               ->  Hash                                                      | [4.357,328.979]      |    252608 |    252608 | [482KB, 482KB]   | 16MB     | [35,39]
 10 |                 ->  Streaming(type: BROADCAST)                              | [2.345,326.320]      |    252608 |    252608 | [2090KB, 2090KB] | 1MB      |
 11 |                   ->  Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1   | [0.011,0.030]        |      3947 |      3947 | [11KB, 11KB]     | 1MB      |
 12 |           ->  Hash Join (13,14)                                             | [1376.258,2066.110]  |         0 |        64 | [5KB, 5KB]       | 1MB      |
 13 |             ->  Seq Scan on channel.p10_md_tmp_t2 t2                        | [777.552,1388.499]   | 443525717 | 443523360 | [12KB, 12KB]     | 1MB      |
 14 |             ->  Hash                                                        | [2.812,4.217]        |    252608 |    252608 | [482KB, 482KB]   | 16MB     | [28,27]
 15 |               ->  Streaming(type: BROADCAST)                                | [1.276,1.868]        |    252608 |    252608 | [2090KB, 2090KB] | 1MB      |
 16 |                 ->  Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1     | [0.010,0.033]        |      3947 |      3947 | [11KB, 11KB]     | 1MB      |
(16 rows)
```

优化后，从超过1个小时未返回结果优化到7s返回结果。

16.5.12 案例：使用 partial cluster key

列存表可以选取某一列或几列设置为partial cluster key(column_name[, ...])。在导入数据时，按设置的列进行局部排序（默认每70个CU即420万行排序一次），生成的CU会聚集在一起，即CU的min，max会在一个较小的区间内。当查询时，where条件含有这些列时，可产生良好的过滤效果。

优化前

未使用partial cluster key。表定义如下：

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
)
with (orientation = column)
distribute by hash(L_ORDERKEY);

select
sum(l_extendedprice * l_discount) as revenue
```

 版权所有 © 华为云计算技术有限公司

```
from
lineitem
where
l_shipdate >= '1994-01-01'::date
and l_shipdate < '1994-01-01'::date + interval '1 year'
and l_discount between 0.06 - 0.01 and 0.06 + 0.01
and l_quantity < 24;
```

导入数据后执行查询，查看执行时间：

图 16-23 未使用 partial cluster key

```
 id |                 operation                 |       A-time        | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
----+-------------------------------------------+---------------------+--------+--------+----------------+---------+---------+-----------
  1 | ->  Row Adapter                           | 1653.156            |      1 |      1 | 12KB           |         |      44 | 205803.90
  2 |    ->  Vector Aggregate                   | 1653.146            |      1 |      1 | 184KB          |         |      44 | 205803.90
  3 |       ->  Vector Streaming (type: GATHER) | 1653.070            |      1 |      1 | 174KB          |         |      44 | 205803.90
  4 |          ->  Vector Aggregate             | [1481.497,1481.497] |      1 |      1 | [225KB, 225KB] |         |      44 | 205803.84
  5 |             ->  CStore Scan on public.lineitem | [1405.004,1405.004] | 114160 | 111485 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```

图 16-24 未使用 partial cluster key 后 CU 加载情况

```
  5 --CStore Scan on public.lineitem
        datanode1 (actual time=40.623..1405.004 rows=114160 loops=1)
        datanode1 (RoughCheck CU: CUNone: 0, CUSome: 101)
        datanode1 (LLVM Optimized)
        datanode1 (Buffers: shared hit=18385 read=23)
        datanode1 (CPU: ex c/r=31917, ex cyc=3643646206, inc cyc=3643646206)
```

优化后

where条件中l_shipdate和l_quantity的distinct值数量较少且可以做min max过滤，将字段l_shipdate、l_quantity设置为PCK修改表定义如下：

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
, partial cluster key(l_shipdate, l_quantity)
)
with (orientation = column)
distribute by hash(L_ORDERKEY);
```

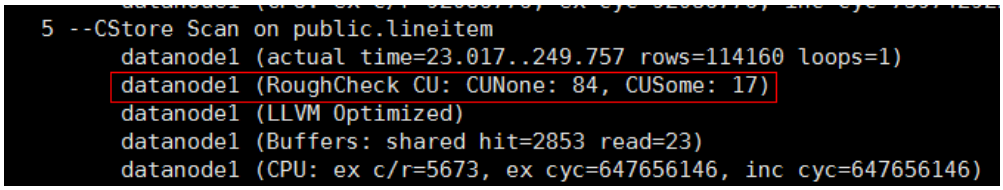
重新导入数据后执行查询，查看执行时间：

图 16-25 使用 partial cluster key

```
 id |                 operation                 |       A-time        | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
----+-------------------------------------------+---------------------+--------+--------+----------------+---------+---------+-----------
  1 | ->  Row Adapter                           | 459.539             |      1 |      1 | 12KB           |         |      44 | 205693.85
  2 |    ->  Vector Aggregate                   | 459.528             |      1 |      1 | 184KB          |         |      44 | 205693.85
  3 |       ->  Vector Streaming (type: GATHER) | 459.452             |      1 |      1 | 174KB          |         |      44 | 205693.85
  4 |          ->  Vector Aggregate             | [285.177,285.177]   |      1 |      1 | [225KB, 225KB] |         |      44 | 205693.79
  5 |             ->  CStore Scan on public.lineitem | [249.757,249.757] | 114160 |  89475 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```


版权所有 © 华为云计算技术有限公司

图 16-26 使用 partial cluster key 后 CU 加载情况



使用partial cluster key后，5-- CStore Scan on public.lineitem的时间减少了1.2s，得益于有84个CU被过滤掉了。

优化建议

- 选取partial cluster key列。
  - 列存表支持创建partial cluster key的类型character varying(n), varchar(n), character(n), char(n), text, nvarchar2, timestamp with time zone, timestamp without time zone, date, time without time zone, time with time zone。
  - 数据的distinct值数量较少，这样能产生较好的过滤效果。
  - 出现在查询where条件中，优先选取能过滤大量数据的列。
  - partial cluster key中设置多个列时，是先按第一个列排序，当第一个列值相同时，使用第二列比较，后续列依次类推。推荐不要超出3个列。
- 添加partial cluster key后，优化导入性能。

  由于添加了partial cluster key，在导入时会增加排序计算，会对导入性能产生影响。当排序完全在内存中进行时影响较小，如果无法在内存中完成排序时，会下盘写临时文件，这时就会产生较大的影响。

  排序使用的内存通过GUC参数psort_work_mem来设置，可以设置较大的值来使用更大的内存进行排序。

  排序的数据量是通过表的存储参数PARTIAL_CLUSTER_ROWS来设置，降低这个数值，可减少一次排序的数据量。这个参数通常与存储参数MAX_BATCHROW配置使用。PARTIAL_CLUSTER_ROWS设置值必须是MAX_BATCHROW的整数倍，MAX_BATCHROW是设置单个CU中数据的最大行数。

16.5.13 案例：NOT IN 转 NOT EXISTS

NOT IN语句需要使用nestloop anti join来实现，而NOT EXISTS则可以通过hash anti join来实现。在join列不存在null值的情况下，not exists和not in等价。因此在确保没有null值时，可以通过将not in转换为not exists，通过生成hash join来提升查询效率。

优化前

创建两个基表t1、t2：

```
CREATE TABLE t1(a int, b int, c int not null) WITH(orientation=row);
CREATE TABLE t2(a int, b int, c int not null) WITH(orientation=row);
```

执行下列SQL语句，查询NOT IN的执行计划：

```
EXPLAIN VERBOSE SELECT * FROM t1 WHERE t1.c NOT IN (SELECT t2.c FROM t2);
```

返回结果如图：

 版权所有 © 华为云计算技术有限公司

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
 id |               operation                | E-rows | E-distinct | E-width | E-costs
----+----------------------------------------+--------+------------+---------+---------
  1 | ->  Streaming (type: GATHER)           |      6 |            |      12 | 78.98
  2 |    ->  Nested Loop Anti Join (3, 4)    |      6 |            |      12 | 64.98
  3 |       ->  Seq Scan on public.t1        |     60 |            |      12 | 18.18
  4 |       ->  Materialize                  |    360 |            |       4 | 30.75
  5 |          ->  Streaming(type: BROADCAST) |    360 |            |       4 | 30.45
  6 |             ->  Seq Scan on public.t2  |     60 |            |       4 | 18.18

             Predicate Information (identified by plan id)
-------------------------------------------------------------------------
  2 --Nested Loop Anti Join (3, 4)
        Join Filter: ((t1.c = t2.c) OR (t1.c IS NULL) OR (t2.c IS NULL))
```

从返回结果可知执行计划走NestLoop，因为NULL值跟任意值的OR运算结果都是NULL，WHERE条件表达式：

```
t1.c NOT IN (SELECT t2.c FROM t2)
```

等价于：

```
t1.c <> ANY(t2.c) AND t1.c IS NOT NULL AND ANY(t2.c) IS NOT NULL
```

优化后

可将查询可以修改为：

```
SELECT * FROM t1 WHERE NOT EXISTS (SELECT * FROM t2 WHERE t2.c = t1.c);
```

执行下列语句，查询NOT EXISTS的执行计划：

```
EXPLAIN VERBOSE SELECT * FROM t1 WHERE NOT EXISTS (SELECT 1 FROM t2 WHERE t2.c = t1.c);
```

```
                                  QUERY PLAN
-----------------------------------------------------------------------------------------------
 id |                operation                 | E-rows | E-distinct | E-width | E-costs
----+------------------------------------------+--------+------------+---------+---------
  1 | ->  Streaming (type: GATHER)             |      6 |            |      12 | 54.56
  2 |    ->  Hash Anti Join (3, 5)             |      6 |            |      12 | 40.56
  3 |       ->  Streaming(type: REDISTRIBUTE)  |     60 | 10         |      12 | 20.12
  4 |          ->  Seq Scan on public.t1       |     60 |            |      12 | 18.18
  5 |       ->  Hash                           |     59 | 10         |       4 | 20.12
  6 |          ->  Streaming(type: REDISTRIBUTE) |     60 |            |       4 | 20.12
  7 |             ->  Seq Scan on public.t2    |     60 |            |       4 | 18.18

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Anti Join (3, 5)
        Hash Cond: (t1.c = t2.c)
```

 版权所有 © 华为云计算技术有限公司

17 实时数仓

17.1 实时数仓简介

实时数仓需要支持将insert+upsert+update等操作实时快速入库，数据来源于上游的其他数据库或者应用，同时要求入库后的数据要能及时查询，对于查询的效率要求很高。

目前DWS传统数仓已有的行存表或者列存表都无法同时满足实时入库和实时查询两个诉求。其中行存表实时入库能力强，支持高并发更新，但是磁盘占用高，查询效率低；列存表数据压缩率高，AP查询性能好，但是不能很好的支持并发更新，并发入库存在严重的锁冲突。

为了解决上面的问题，需要在使用列存储格式尽量降低磁盘占用的同时，支持高并发的更新操作入库以及高性能的查询效率。DWS的实时数仓中的HStore表就是针对这种情况设计和实现的，面向对于实时入库和实时查询有较强诉求的场景，同时拥有处理传统TP场景的事务能力。

DWS提供的实时数仓中实现了一种全新的HStore表，可以做到单条或者小批量IUD操作的高并发实时入库，也可以支持大批量的定期入库。数据入库提交后即可查询，无任何时延。支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询，也可以在TPCC这种强事务压力场景下保证数据强一致性。

说明

- 实时数仓的HStore表仅8.2.0.100及以上集群版本支持。
- 9.1.0及以上版本推荐使用HStore_opt表。除了微批copy无更新入库性能要求高的场景外，HStore表的场景都可以使用HStore_opt表代替，性能更优。
- 实时数仓为一库两用，生产即分析；适用于交易、分析混合型业务场景；关于如何创建实时数仓请参见《数据仓库服务用户指南》中“创建DWS存算一体2.0集群”章节。
- HStore表支持冷热数据管理，具体可参考**DWS冷热数据管理**，该功能仅8.2.0.101及以上集群版本支持。
- HStore表是实时数仓中设计的一种表类型，与SQL参数hstore没有任何关系。

与标准数仓的区别

实时数仓与标准数仓是DWS的两种规格不一样的数仓，在使用上也存在一定差异，具体可参考**表17-1**进行对比分析。

版权所有 © 华为云计算技术有限公司

表 17-1 实时数仓与标准数仓的差异

数仓类型	标准数仓	实时数仓
适用场景	融合分析业务，一体化OLAP分析场景。主要应用于金融、政企、电商、能源等领域。	实时入库+分析混合业务，上游数据实时入库+数据入库后实时高效查询场景。主要用于电商、金融等实时入库要求高的场景。
产品优势	性价比高，使用场景广泛。 支持冷热数据分析，存储、计算弹性伸缩，无限算力、无限容量等。	混合负载，入库性能强。 提供与列存相当的高性能查询效率与高压缩率的数据压缩能力。同时拥有处理传统TP场景的事务能力。
功能特点	支持海量数据离线处理和交互查询，数据规模大、复杂数据挖掘具有很好的性能优势。	支持海量数据高并发的更新操作入库以及高性能的查询效率。在数据规模大、入库并发高、查询要求高的场景下具有很好的性能优势。
SQL语法	SQL语法兼容性高，语法通用，易于使用。	兼容列存语法。
GUC参数	丰富的GUC参数，根据客户业务场景适配最适合客户的数仓环境。	兼容标准数仓GUC参数，同时支持实时数仓调优参数。

技术特点

- 完整的事务一致性

  体现在数据插入或者更新后提交即可见，不存在时延；并发更新后数据保证强一致，不会出现乱序导致的结果预期不一致。

- 查询性能好

  多表关联等复杂AP查询场景下，更完善的分布式查询计划与分布式执行器带来的性能优势，支持复杂的子查询和存储过程。

- 入库快

  彻底解决列存CU锁冲突问题，支持高并发的更新入库操作，典型场景下，并发更新性能是之前的百倍以上。

- 高压缩

  数据在MERGE进入列存主表后，按列存储具有天然的压缩优势，能极大地节省磁盘空间与IO资源。

- 查询加速

  支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询。

 版权所有 © 华为云计算技术有限公司

行存、列存、HStore 表对比

表 17-2 行存、列存、HStore 表对比

表类型	行存表	列存表	HStore表
数据存储方式	以元组为单位，将每一条数据的所有属性值存储到临近的空间里。	以CU（Compress Unit）为单位，将单个属性的所有值存储到临近的空间里。	数据主要以CU形式存储在列存主表上，对于被更新的列、小批量插入的数据将被序列化后存储到新设计的Delta表上。
数据写入	行存压缩暂未商用，数据按原始状态存储，磁盘空间占用较大。	按列存储时，由于属性值类型相同具有天然的压缩优势。数据写入时能极大节省IO资源与磁盘空间占用。	批量插入的数据直接写入CU，具有与列存一致的压缩优势。 被更新的列、小批量插入的数据会序列化后压缩。同时定期merge到主表CU。
数据更新	数据按行更新，没有CU锁问题，并发更新（update/upsert/delete等）性能好。	即使更新单条数据，也要获取整个CU的锁，基本无法支持并发更新（update/upsert/delete等）。	彻底解决列存更新的CU锁问题，并发更新（update/upsert/delete等）的性能达到行存的60%以上。
数据读取	按行读取，即使只需访问某一列的数据，也需要将一整行的数据取出。查询性能较差。	按列读取时只需访问该列的CU，再加上CU的压缩优势导致需要占用的IO资源更少，读取性能很好。	对于列存主表的数据按列读取，对于被更新的列、小批量插入的数据会反序列化后取出，数据merge到主表后具有与列存一致的数据读取优势。
优点	并发更新性能好。	查询性能好，磁盘占用空间少。	并发更新性能好，数据MERGE后具有与列存一致的查询性能优势与压缩优势。
缺点	占用磁盘空间多，查询性能差。	基本无法支持并发更新。	需要后台常驻线程对HStore表进行merge清理操作。先merge到CU主表再进行清理，与SQL语法中的Merge无关。
适用场景	1. 更新删除操作频繁的TP事务场景。 2. 点查询（基于索引的、返回数据量小的简单查询）。	1. 查询分析为主的AP场景。 2. 数据量大，存入后的更新删除操作少。	1. 实时并发入库场景。 2. 需要支持高并发的更新入库操作以及高性能的查询效率。

版权所有 © 华为云计算技术有限公司

17.2 支持与限制

实时数仓兼容所有列存语法，具体情况如下：

表 17-3 支持的语法

语法	是否支持
CREATE TABLE	支持
CREATE TABLE LIKE	支持
DROP TABLE	支持
INSERT	支持
COPY	支持
SELECT	支持
TRUNCATE	支持
EXPLAIN	支持
ANALYZE	支持
VACUUM	支持
ALTER TABLE DROP PARTITION	支持
ALTER TABLE ADD PARTITION	支持
ALTER TABLE SET WITH OPTION	支持
ALTER TABLE DROP COLUMN	支持
ALTER TABLE ADD COLUMN	支持
ALTER TABLE ADD NODELIST	支持
ALTER TABLE CHANGE OWNER	支持
ALTER TABLE RENAME COLUMN	支持
ALTER TABLE TRUNCATE PARTITION	支持
CREATE INDEX	支持
DROP INDEX	支持
DELETE	支持
ALTER TABLE 其他	支持
ALTER INDEX	支持
MERGE	支持

版权所有 © 华为云计算技术有限公司

语法	是否支持
SELECT INTO	支持
UPDATE	支持
CREATE TABLE AS	支持

约束限制

1. 当需要使用HStore表时，需要同步修改以下几个参数的默认值，否则会导致HStore表性能严重劣化。

   推荐的参数修改配置是：autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。
2. 8.2.1及以上版本支持列存索引脏数据清理，频繁更新入库的场景下能有效控制索引空间，提高入库与查询性能。
3. 当使用HStore异步排序功能时，有两点需要注意。
   - 异步排序期间可能会阻塞部分数据的DML操作，最大的阻塞粒度为异步排序的行数阈值，因此在反复DML的场景下不建议使用此功能。
   - 自动异步排序与列存vacuum互斥，当autovacuum流程中满足列存vacuum的条件，则直接跳过流程中的异步排序，等待下次触发（极端场景下因为反复大批量DML操作的执行，列存vacuum会不停被触发，因此异步排序总是无法被触发）。

17.3 实时数仓语法

17.3.1 CREATE TABLE

功能描述

在当前数据库中创建一个新的空白HStore表，该表由命令执行者所有。

实时数仓提供创建HStore表DDL语句。创建HStore表DDL需要指定enable_hstore为on，同时需要将orientation属性设置为column。

9.1.0及以上版本对HStore表做了优化，为保持前向兼容，保留了老的HStore表，优化后的HStore表为HStore_opt表。除了微批copy无更新入库性能要求高的场景外，HStore表的场景都可以使用HStore_opt表代替，性能更优。

HStore表一旦创建好，无法直接转换为HStore_opt表，若要使用HStore_opt表，需要重新建表。

注意事项

- 创建HStore表时，必须确保数据库GUC参数设置满足以下条件：
  - autovacuum设置为on。
  - autovacuum_max_workers_hstore取值大于0。
  - autovacuum_max_workers取值大于autovacuum_max_workers_hstore的取值。

版权所有 © 华为云计算技术有限公司

- 创建HStore表的用户需要拥有schema cstore的USAGE权限。
- 表级参数enable_delta与enable_hstore_opt无法同时开启，因为enable_delta用于控制普通列存表的delta开启，会与enable_hstore_opt冲突。
- 每一个HStore表绑定一张delta表，delta表的oid记录在pg_class中reldeltaidx字段（reldelta字段被列存表的delta表使用）。

语法格式

```
CREATE TABLE [ IF NOT EXISTS ] table_name
({ column_name data_type
    | LIKE source_table [like_option [...] ] }
}
    [, ... ])
[ WITH ( {storage_parameter = value}  [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY  HASH ( column_name [,...])]
[ TO { GROUP groupname | NODE ( nodename [, ... ] ) } ]
[ PARTITION BY {
       {RANGE (partition_key) ( partition_less_than_item [, ... ] )}
 } [ { ENABLE | DISABLE } ROW MOVEMENT ] ];
其中like选项like_option为：
{ INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES | STORAGE | COMMENTS | PARTITION
| RELOPTIONS | DISTRIBUTION | ALL }
```

列存表的 Delta 表差异

表 17-4 HStore 表与列存表的辅助 Delta 表差异

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
表结构	与列存主表的表定义一致。	与主表定义不一致。	不同于主表，与HStore表相同。
功能	用于暂存小批量insert的数据，满阈值后再merge到主表，避免直接insert到主表产生大量小CU。	用于持久化存储update/delete/insert信息。在产生故障后用于恢复内存更新链等管理并发更新的内存结构。	用于持久化存储update/delete/insert信息。在产生故障后用于恢复内存更新链等管理并发更新的内存结构。相比HStore进一步优化。
缺陷	来不及merge导致delta表膨胀，影响查询性能，同时无法解决并发update的锁冲突问题	依赖后台常驻autovacuum来做merge操作。	依赖后台常驻autovacuum来做merge操作。

版权所有 © 华为云计算技术有限公司

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
规格差异	不支持同一CU的并发，适合并发更新较少的情况。	1. 插入和更新限制 ● MERGE INTO操作不支持并发更新同一行或对同一键进行重复更新。 ● 不支持并发update或delete同一行，会触发报错。 2. 索引和查询限制 ● 索引不支持数组条件过滤、IN表达式过滤、部分索引或表达式索引。 ● 索引不支持失效处理。 3. 表结构和操作限制 ● 在进行分区交换或relfilenode操作时，确保交换的表都是HStore表。 ● 不支持update修改分布列，不建议update修改分区列（不报错，但性能非常差）。	1. 插入和更新限制 ● MERGE INTO操作不支持并发更新同一行或对同一键进行重复更新。 ● 不支持并发update或delete同一行，会触发报错。 ● hstore_opt不支持跨分区upsert更新。 2. 索引和查询限制 ● 支持bitmap索引。 ● 支持全局字典。 ● bitmap_columns必须在建表时确定，一旦设置后不允许修改。 ● opt版本不支持SMP streaming透传参数。在需要分区剪枝的多表关联查询中，避免使用复制表或设置query_dop。 3. 表结构和操作限制 ● 不支持update修改分布列，不支持update修改分区列。 ● enable_hstore_opt属性需在创建表时设定，且不可更改。

版权所有 © 华为云计算技术有限公司

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
入库建议	1. 综合入库、查询和空间情况，建议选择hstore_opt表，微批copy无更新入库场景性能要求高的情况下选择hstore表。 2. HStore/HStore Opt共同点： ● update入库性能差，建议修改为upsert。 ● delete入库，确定计划走索引扫描即可，用JDBC batch方式入库最佳。 ● merge into入库建议在单次入库数据量超过100W/dn，且无并发数据保证无重复的情况下使用。 ● 尽量避免对冷分区的数据进行修改和新增。		
点查建议	1. 综合点查场景，建议使用HStore Opt表。 2. HStore/HStore Opt共同点： 在等值过滤条件使用最多且distinct值分布相对均匀的一个列上创建二级分区。 3. HStore的点查建议： ● 使用主键之外的索引加速不能得到预期效果，不建议开启。 ● 如果数据类型多为numeric或者16字节以内的字符串，建议使用turbo加速。 4. HStore Opt的点查建议： ● 除了二级分区之外的等值过滤列，如果过滤条件涉及的列在查询中基本固定，使用cbtree索引，如果不断变化建议使用GIN索引，注意创建索引的列数不要超过5列。 ● 所有涉及等值过滤的字符串列，都可以建表时指定bitmap索引，不限列数，但后续不可修改。 ● 时间范围过滤的列，指定为分区列。 ● 点查返回数据量超过10W/dn的场景，索引扫描的性能提升有限，建议针对这种情况使用guc参数enable_seqscan对比测试，灵活选择。		

参数说明

- **IF NOT EXISTS**

  指定IF NOT EXISTS时，若不存在同名表，则可以成功创建表。若已存在同名表，创建时不会报错，仅会提示该表已存在并跳过创建。

- **table_name**

  要创建的表名。

  表名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。

- **column_name**

  新表中要创建的字段名。

  字段名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。

 版权所有 © 华为云计算技术有限公司

- **data_type**

  字段的数据类型。

- **LIKE source_table [like_option ...]**

  LIKE子句声明一个表，新表自动从这个表中继承所有字段名及其数据类型。

  新表与原表之间在创建动作完毕之后是完全无关的。在原表做的任何修改都不会传播到新表中，并且也不可能在扫描原表的时候包含新表的数据。

  被复制的列并不使用相同的名字进行融合。如果明确的指定了相同的名字或者在另外一个LIKE子句中，将会报错。

  HStore表只能从HStore表中进行继承。

- **WITH ({ storage_parameter = value } [, ...])**

  这个子句为表指定一个可选的存储参数。

  - ORIENTATION

    指定表数据的存储方式，即时序方式、行存方式、列存方式，该参数设置成功后就不再支持修改。对于HStore表，应当使用列存方式，同时设置enable_hstore为on。

    取值范围：

    - TIMESERIES，表示表的数据将以时序方式存储。
    - COLUMN，表示表的数据将以列存方式存储。
    - ROW，表示表的数据将以行方式存储。

    默认值：ROW。

  - COMPRESSION

    指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。

    取值范围：

    仅HStore表和列存表支持，有效值为LOW、MIDDLE或HIGH。默认值为LOW。

  - COMPRESSLEVEL

    指定表数据同一压缩级别下的不同压缩水平，它决定了同一压缩级别下表数据的压缩比以及压缩时间。对同一压缩级别进行了更加详细的划分，为用户选择压缩比和压缩时间提供了更多的空间。总体来讲，此值越大，表示同一压缩级别下压缩比越大，压缩时间越长；反之亦然。该参数只对时序表和列存表有效。

    取值范围：0~3

    默认值：0

  - MAX_BATCHROW

    指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对时序表和列存表有效。

    取值范围：10000~60000

    默认值：60000

  - PARTIAL_CLUSTER_ROWS

    指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对时序表和列存表有效。

 版权所有 © 华为云计算技术有限公司

取值范围：600000~2147483647

默认值：4,200,000

- enable_delta

  指定了在列存表是否开启delta表。对HStore表不能开启该参数。

  默认值：off

- enable_hstore

  指定了是否创建为HStore表（基于列存表实现）。该参数只对列存表有效。该参数仅8.2.0.100及以上集群版本支持。

  默认值：off

  说明

  打开该参数时必须设置以下GUC参数用于保证HStore表的清理，推荐值如下：

  autovacuum=true，autovacuum_max_workers=6，autovacuum_max_workers_hstore=3。

- enable_disaster_cstore

  指定了列存表是否开启细粒度容灾功能。该参数仅适用于COLVERSION为2.0的列存表，并且不能和enable_hstore同时打开。该参数仅8.2.0.100及以上集群版本支持。

  默认值：off

注意

设置该参数为on前需要先设置GUC参数enable_metadata_tracking=on，否则可能开启细粒度容灾功能失败。

- SUB_PARTITION_COUNT

  指定二级分区的个数。该参数用于设置在导入阶段二级分区个数。在建表时进行设置，建表后不支持修改。不建议用户随意设置该默认值，可能会影响导入和查询的性能。

  取值范围：1～1024

  默认值：32

- DELTAROW_THRESHOLD

  指定HStore表导入时小于多少行(SUB_PARTITION_COUNT * DELTAROW_THRESHOLD)的数据进入delta表。

  取值范围：0～60000

  默认值：60000

- COLVERSION

  指定存储格式的版本。 HStore表只支持2.0版本，enable_hstore_opt表支持2.0和3.0版本。

  取值范围：

  1.0：列存表的每列以一个单独的文件进行存储，文件名以relfilenode.C1.0、relfilenode.C2.0、relfilenode.C3.0等命名。

  2.0：列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名。

  默认值：2.0

版权所有 © 华为云计算技术有限公司

- enable_binlog

  用于控制hstore opt表是否开启binlog功能。该参数仅9.1.0及以上集群版本支持。

  取值范围：on/off

  默认值：off

- enable_binlog_timestamp

  用于控制hstore opt表是否开启带时间戳的binlog功能。该参数与enable_binlog无法同时打开，且仅9.1.0.200及以上集群版本支持。

  取值范围：on/off

  默认值：off

- DISTRIBUTE BY

  指定表如何在节点之间分布或者复制。

  取值范围：

  HASH (column_name)：对指定的列进行Hash，通过映射，把数据分布到指定DN。

- TO { GROUP groupname | NODE (nodename [, ...]) }

  TO GROUP指定创建表所在的Node Group，目前不支持hdfs表使用。TO NODE主要供内部扩容工具使用，一般用户不应该使用。

- PARTITION BY

  指定HStore表的初始分区。

- secondary_part_column

  指定列存表二级分区列的列名，仅能指定一列作为二级分区列且只适用于HStore opt列存表。该参数仅9.1.0及以上集群版本支持，其中v3表不支持该参数，会走hashbucket剪枝。

  说明

  - 被指定为二级分区的列不能被删除或者修改。
  - 只能在建表时指定二级分区列，建表后不支持修改。
  - 不建议指定分布列作为二级分区列。
  - 指定二级分区列后，该表将会在DN上按照二级分区列进行逻辑上的HASH分区存储，从而有效提升该列等值查询的性能。

- secondary_part_num

  指定列存表二级分区的数量，仅适用于HStore opt列存表。该参数仅9.1.0及以上集群版本支持，v3表不支持该参数，会走hashbucket剪枝。

  取值范围：1~32

  默认值：8

  说明

  - 只有secondary_part_column被指定时，该参数才可以指定。
  - 只能在建表时指定二级分区的数量，建表后不支持修改。
  - 不建议用户随意设置该默认值，否则会影响导入和查询的性能。

示例

创建简单的HStore_opt表:

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE warehouse_t1
(
    W_WAREHOUSE_SK            INTEGER               NOT NULL,
    W_WAREHOUSE_ID            CHAR(16)              NOT NULL,
    W_WAREHOUSE_NAME          VARCHAR(20)                   ,
    W_WAREHOUSE_SQ_FT         INTEGER                       ,
    W_STREET_NUMBER           CHAR(10)                      ,
    W_STREET_NAME             VARCHAR(60)                   ,
    W_STREET_TYPE             CHAR(15)                      ,
    W_SUITE_NUMBER            CHAR(10)                      ,
    W_CITY                    VARCHAR(60)                   ,
    W_COUNTY                  VARCHAR(30)                   ,
    W_STATE                   CHAR(2)                       ,
    W_ZIP                     CHAR(10)                      ,
    W_COUNTRY                 VARCHAR(20)                   ,
    W_GMT_OFFSET              DECIMAL(5,2)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE_OPT=ON);

CREATE TABLE warehouse_t2 (LIKE warehouse_t1 INCLUDING ALL);
```

17.3.2 INSERT

功能描述

往HStore表中插入一行或多行数据。

注意事项

- 当单次插入的数据量大于等于表级参数DELTAROW_THRESHOLD时，数据会直接插入主表生成CU（Compress Unit）。
- 当插入的数据量小于表级参数DELTAROW_THRESHOLD时，会在辅助Delta表上插入一条类型为I的记录，同时将数据序列化存储到这条记录的values字段。
- 插入到Delta表的数据跟主表使用全局统一分配的cuid。
- 插入到delta表的数据依赖autovacuum来merge到主表CU。

语法格式

```
INSERT [/*+ plan_hint */] [ IGNORE | OVERWRITE ] INTO table_name [ AS alias ] [ ( column_name [, ...] ) ]
    { DEFAULT VALUES
    | VALUES {( { expression | DEFAULT } [, ...] ) }[, ...] | query }
```

参数说明

- **table_name**

  要插入数据的目标表名。

  取值范围：已存在的表名。

- **AS**

  用于给目标表table_name指定别名。alias即为别名的名字。

- **column_name**

  目标表中的字段名。

- **query**

  一个查询语句（SELECT语句），将查询结果作为插入的数据。

 版权所有 © 华为云计算技术有限公司

示例

创建表reason_t1：

```
-- 创建表reason_t1。
CREATE TABLE reason_t1
(
    TABLE_SK          INTEGER               ,
    TABLE_ID          VARCHAR(20)           ,
    TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE_OPT=ON);
```

向表中插入一条记录：

```
INSERT INTO reason_t1(TABLE_SK, TABLE_ID, TABLE_NA) VALUES (1, 'S01', 'StudentA');
```

向表中插入多条记录：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
SELECT * FROM reason_t1 ORDER BY 1;
 TABLE_SK | TABLE_ID | TABLE_NAME
----------+----------+------------
        1 |      S01 |   StudentA
        2 |      T01 |   TeacherA
        3 |      T02 |   TeacherB
(3 rows)
```

17.3.3 DELETE

功能描述

删除HStore opt表中的数据。

注意事项

- 如果需要删除表上的所有数据，建议使用TRUNCATE语法，可以有效提高性能同时减少空间膨胀。
- HStore opt表上的单条DELETE操作，会往Delta中插入一条type是D的记录，同时再更新内存更新链用于管理并发。
- HStore opt表上的批量DELETE操作，对于每个CU上的连续delete，会插入一条type是MD的记录。
- 对于并发DELETE场景， 传统列存储格式由于同时操作相同CU时会阻塞所以并发性能较差， 对于HStore opt表由于不需要阻塞等待，并发DELETE性能可达到列存的百倍以上。

语法格式

```
DELETE FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ USING using_list ]
    [ WHERE condition ]
```

参数说明

- **ONLY**

  如果指定ONLY则只有该表被删除；如果没有声明，则该表和它的所有子表将都被删除。

- **table_name**

  目标表的名字（可以有模式修饰）。

  取值范围：已存在的表名。

 版权所有 © 华为云计算技术有限公司

- **alias**
  目标表的别名。
  取值范围：字符串，符合标识符命名规范。
- **using_list**
  using子句。
- **condition**
  一个返回boolean值的表达式，用于判断哪些行需要被删除。

示例

创建表reason_t2：

```
CREATE TABLE reason_t2
(
   TABLE_SK          INTEGER               ,
   TABLE_ID          VARCHAR(20)           ,
   TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE_OPT=ON);
INSERT INTO reason_t2 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
```

使用WHERE条件删除：

```
DELETE FROM reason_t2 WHERE TABLE_SK = 2;
DELETE FROM reason_t2 AS rt2 WHERE rt2.TABLE_SK = 2;
```

使用IN语法删除：

```
DELETE FROM reason_t2 WHERE TABLE_SK in (1,3);
```

17.3.4 UPDATE

功能描述

更新HStore opt表上指定的数据。

注意事项

- 与列存一样，当前版本HStore opt上的UPDATE操作始终先DELETE再INSERT。全局GUC参数可控制打开HStore opt的轻量化UPDATE，当前版本默认关闭。
- 对于并发UPDATE场景， 传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差， 对于HStore opt表由于不需要阻塞等待，并发UPDATE性能可达到列存的百倍以上。

语法格式

```
UPDATE [/*+ plan_hint */] [ ONLY ] table_name [ * ] [ [ AS ] alias ]
SET {column_name = { expression | DEFAULT }
   |( column_name [, ...] ) = {( { expression | DEFAULT } [, ...] ) |sub_query }}[, ...]
   [ FROM from_list] [ WHERE condition ];
```

参数说明

- **plan_hint子句**
  以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见《 数据仓库服务开发指南 》中的“优化查询性能>改进查询>使用Plan Hint进行调优”章节。
- **table_name**

 版权所有 © 华为云计算技术有限公司

要更新的表名，可以使用模式修饰。

取值范围：已存在的表名称。

- **alias**

  目标表的别名。

  取值范围：字符串，符合标识符命名规范。

- **expression**

  赋给字段的值或表达式。

- **DEFAULT**

  用对应字段的缺省值填充该字段。

  如果没有缺省值，则为NULL。

- **from_list**

  一个表的表达式列表，允许在WHERE条件里使用其他表的字段。与在一个SELECT语句的FROM子句里声明表列表类似。

  须知

  目标表禁止出现在from_list里，除非在使用一个自连接（此时它必须以from_list的别名出现）。

- **condition**

  一个返回boolean类型结果的表达式。只有这个表达式返回true的行才会被更新。

示例

创建表reason_update：

```
CREATE TABLE reason_update
(
   TABLE_SK         INTEGER              ,
   TABLE_ID         VARCHAR(20)          ,
   TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE_OPT=ON);
```

向表reason_update中插入数据：

```
INSERT INTO reason_update VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
```

对表reason_update执行UPDATE操作：

```
UPDATE reason_update SET TABLE_NA = 'TeacherD' where TABLE_SK = 3;
```

17.3.5 UPSERT

功能描述

HStore opt兼容UPSERT语法，向表中添加一行或多行数据。当出现主键或者唯一约束冲突时更新或者忽略冲突的数据。

注意事项

- 目标表上必须包含主键或者唯一索引才可以执行UPSERT的冲突更新语句。
- 与列存一样，当UPSERT触发更新操作时，当前版本HStore opt上的更新操作始终先DELETE再INSERT。

版权所有 © 华为云计算技术有限公司

- 对于并发UPSERT场景，传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差，对于HStore opt表由于不需要阻塞等待，并发UPSERT性能可达到列存的百倍以上。

语法格式

表 17-5 UPSERT 语法格式

语法格式	冲突更新	冲突忽略
第一种：不指定索引	INSERT INTO ON DUPLICATE KEY UPDATE	INSERT IGNORE INSERT INTO ON CONFLICT DO NOTHING
第二种：从指定列名或者约束上可以推断唯一约束	INSERT INTO ON CONFLICT(...) DO UPDATE SET INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO UPDATE SET	INSERT INTO ON CONFLICT(...) DO NOTHING INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO NOTHING

参数说明

第一种不指定索引。会在所有主键或唯一索引上检查冲突，有冲突就会忽略或者更新。

第二种指定索引。会从ON CONFLICT子句中指定列名、包含列名的表达式或者约束名上推断主键或者唯一索引。

- 唯一索引推断

  对于第二种语法形式，通过指定列名或者约束名推断主键或者唯一索引。列名可以是单一列名，或者由多个列名组成的表达式，比如column1，column2，column3。

- UPDATE子句

  UPDATE子句可以通过VALUES(colname)或者EXCLUDED.colname引用插入的数据。EXCLUDED表示因冲突原本该排除的数据行。

- WHERE子句
  - 用于在数据冲突时，判断是否满足指定条件。如果满足，则更新冲突数据。否则忽略。
  - 只有第二种语法形式的冲突更新语法可以指定WHERE子句。即INSERT INTO ON CONFLICT(...) DO UPDATE SET WHERE

示例

创建表reason_upsert并向表中插入数据：

```
CREATE TABLE reason_upsert
(
  a    int primary key,
  b    int,
  c    int
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE_OPT=ON);
INSERT INTO reason_upsert VALUES (1, 2, 3);
```

忽略冲突的数据：

 版权所有 © 华为云计算技术有限公司

```
INSERT INTO reason_upsert  VALUES (1, 4, 5),(2, 6, 7) ON CONFLICT(a) DO NOTHING;
```

更新冲突的数据：

```
INSERT INTO reason_upsert  VALUES (1, 4, 5),(3, 8, 9) ON CONFLICT(a) DO UPDATE SET b = EXCLUDED.b,
c = EXCLUDED.c;
```

17.3.6 MERGE INTO

功能描述

通过MERGE INTO语句，将目标表和源表中数据针对关联条件进行匹配，若关联条件匹配时对目标表进行UPDATE，无法匹配时对目标表执行INSERT。此语法可以很方便地用来合并执行UPDATE和INSERT，避免多次执行。

注意事项

对于并发MERGE INTO场景，触发UPDATE时，传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差，对于HStore opt表由于不需要阻塞等待，并发MERGE INTO性能可达到列存的百倍以上。

语法格式

```
MERGE INTO table_name [ [ AS ] alias ]
USING { { table_name | view_name } | subquery } [ [ AS ] alias ]
ON ( condition )
[
  WHEN MATCHED THEN
  UPDATE SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
  [ WHERE condition ]
]
[
  WHEN NOT MATCHED THEN
  INSERT { DEFAULT VALUES |
  [ ( column_name [, ...] ) ] VALUES ( { expression | DEFAULT } [, ...] ) [, ...] [ WHERE condition ] }
];
```

参数说明

- **INTO子句**

  指定正在更新或插入的目标表。

  - **table_name**

    目标表的表名。

  - **alias**

    目标表的别名。

    取值范围：字符串，符合标识符命名规范。

- **USING子句**

  指定源表，源表可以为表、视图或子查询。

- **ON子句**

  关联条件，用于指定目标表和源表的关联条件。不支持更新关联条件中的字段。ON关联条件可以是ctid, xc_node_id, tableoid这三个系统列。

- **WHEN MATCHED子句**

  当源表和目标表中数据针对关联条件可以匹配上时，选择WHEN MATCHED子句进行UPDATE操作。

 版权所有 © 华为云计算技术有限公司

> **说明**
>
> 不支持更新分布列、系统表以及系统列。

- **WHEN NOT MATCHED子句**

  当源表和目标表中数据针对关联条件无法匹配时，选择WHEN NOT MATCHED子句进行INSERT操作。

  > **说明**
  >
  > - 不支持INSERT子句中包含多个VALUES。
  > - WHEN MATCHED和WHEN NOT MATCHED子句顺序可以交换，可以缺省其中一个，但不能同时缺省。
  > - 不支持同时指定两个WHEN MATCHED或WHEN NOT MATCHED子句。

示例

创建进行MERGE INTO的目标：

```
CREATE TABLE target(a int, b int)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE_OPT = ON);
INSERT INTO target VALUES(1, 1),(2, 2);
```

创建数据源表：

```
CREATE TABLE source(a int, b int)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE_OPT = ON);
INSERT INTO source VALUES(1, 1),(2, 2),(3, 3),(4, 4),(5, 5);
```

执行MERGE INTO操作：

```
MERGE INTO target t
USING source s
ON (t.a = s.a)
WHEN MATCHED THEN
  UPDATE SET t.b = t.b + 1
WHEN NOT MATCHED THEN
  INSERT VALUES (s.a, s.b) WHERE s.b % 2 = 0;
```

17.3.7 SELECT

功能描述

从HStore opt表读取数据。

注意事项

- 列存表与HStore opt表都暂不支持SELECT FOR UPDATE语法。
- 对HStore opt表执行SELECT查询时，会扫描列存主表CU上的数据、delta表上的I记录中的数据、内存中每行数据更新信息，并将三种信息合并后返回。
- 在通过主键索引或唯一索引查询数据的场景中：

  对于传统列存表，唯一索引会同时存储行存Delta表上的数据位置信息（blocknum，offset）与列存主表的数据位置信息（cuid，offset），数据MERGE到主表后又会插入新的索引元组，索引会持续膨胀。

  对于HStore opt表，由于实现了全局CUID的统一分配，索引元组中始终只存储（cuid，offset）， 数据MERGE后不会产生新的索引元组。

语法格式

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [/*+ plan_hint */] [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
{ * | {expression [ [ AS ] output_name ]} [, ...] }
```

 版权所有 © 华为云计算技术有限公司

```
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY grouping_element [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]
[ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort_expression_clause ] [ NULLS { FIRST | LAST } ]} [, ...] ]
[ { [ LIMIT { count | ALL } ] [ OFFSET start [ ROW | ROWS ] ] } | { LIMIT start, { count | ALL } } ]
```

参数说明

- **DISTINCT [ON (expression [, ...])]**

  从SELECT的结果集中删除所有重复的行，使结果集中的每行都是唯一的。

  ON（ expression [, ...] ）只保留那些在给出的表达式上运算出相同结果的行集合中的第一行。

- **SELECT列表**

  指定查询表中列名，可以是部分列或者是全部（使用通配符*表示）。

  通过使用子句AS output_name可以为输出字段取个别名，这个别名通常用于输出字段的显示。

- **FROM子句**

  为SELECT声明一个或者多个源表。

  FROM子句涉及的元素如下所示。

- **WHERE子句**

  WHERE子句构成一个行选择表达式，用来缩小SELECT查询的范围。condition是返回值为布尔型的任意表达式，任何不满足该条件的行都不会被检索。

  WHERE子句中可以通过指定"(+)"操作符的方法将表的连接关系转换为外连接。但是不建议用户使用这种用法，因为这并不是SQL的标准语法，在做平台迁移的时候可能面临语法兼容性的问题。同时，使用"(+)"有很多限制：

- **GROUP BY子句**

  将查询结果按某一列或多列的值分组，值相等的为一组。

- **HAVING子句**

  与GROUP BY子句配合用来选择特殊的组。HAVING子句将组的一些属性与一个常数值比较，只有满足HAVING子句中的逻辑表达式的组才会被提取出来。

- **ORDER BY子句**

  对SELECT语句检索得到的数据进行升序或降序排序。对于ORDER BY表达式中包含多列的情况：

示例

创建表reason_select并向表中插入数据：

```
CREATE TABLE reason_select
(
  r_reason_sk integer,
  r_reason_id integer,
  r_reason_desc character(100)
)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE_OPT=ON);
INSERT INTO reason_select values(3, 1,'reason 1'),(10, 2,'reason 2'),(4, 3,'reason 3'),(10, 4,'reason 4');
```

执行GROUP BY分组操作：

```
SELECT COUNT(*), r_reason_sk FROM reason_select GROUP BY r_reason_sk;
```

执行HAVING过滤操作：

 版权所有 © 华为云计算技术有限公司

```
SELECT COUNT(*) c,r_reason_sk FROM reason_select GROUP BY r_reason_sk HAVING c > 1;
```

执行ORDER BY操作：

```
SELECT * FROM reason_select ORDER BY r_reason_sk;
```

17.3.8 ALTER TABLE

功能描述

修改表，包括修改表的定义、重命名表、重命名表中指定的列、添加/更新多个列、将列存改为HStore opt表等。

注意事项

- 通过ALTER修改enable_hstore值可以将列存表变成HStore表，或者将HStore修改成列存表。但需要注意enable_delta为on时，无法设置enable_hstore为on。
- 对于部分ALTER操作（修改列类型，分区合并，添加NOT NULL约束，添加主键约束），HStore表需要先将数据MERGE到主表然后再执行原有的ALTER逻辑，这可能会带来额外的性能开销， 性能影响大小与delta表的数据量相关。
- 当需要增加列时， 建议不要与其它类型的ALTER（比如修改列类型等）组合使用，在一条ALTER里只有ADD COLUMN情况下，由于不需要做FULL MERGE，性能会有很大提升。
- 不支持修改存储参数ORIENTATION。

修改表属性

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> SET （{ENABLE_HSTORE = ON} [, ... ] );
```

将列存表修改成HStore表：

```
CREATE TABLE alter_test(a int, b int) WITH(ORIENTATION = COLUMN);
ALTER TABLE alter_test SET (ENABLE_HSTORE = ON);
```

须知

当需要使用HStore表时，一定要同步修改如下参数的默认值，否则会导致HStore性能严重劣化，推荐的默认配置是

autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。

增加列

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> ADD COLUMN <new_column> <data_type> [ DEFAULT <default_value>];
```

示例：

创建表alter_test2并对其增加列：

```
CREATE TABLE alter_test2(a int, b int) WITH(ORIENTATION = COLUMN,ENABLE_HSTORE_OPT = ON);
ALTER TABLE alter_test ADD COLUMN c int;
```


版权所有 © 华为云计算技术有限公司

说明

增加列时不建议在同一个SQL中与其它ALTER组合使用。

重命名

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> RENAME TO <new_table_name>;
```

示例：

创建表alter_test3并对其重命名为alter_new：

```
CREATE TABLE alter_test3(a int, b int) WITH(ORIENTATION = COLUMN,ENABLE_HSTORE_OPT = ON);
ALTER TABLE alter_test3 RENAME TO alter_new;
```

17.4 实时数仓函数

hstore_light_merge(rel_name text)

描述：该函数用于手动对HStore表进行轻量化清理操作，持有目标表的三级锁。

返回值类型：int

示例：

```
SELECT hstore_light_merge('reason_select');
```

hstore_full_merge(rel_name text，partitionName text)

描述：该函数用于手动对HStore表进行全量清理操作。

partitionName为可选参数，用于指定单分区进行操作。

返回值类型：int

说明

- 执行该操作会强制将DELTA表上的所有可见操作Merge到主表，然后建一张新的空Delta表，期间持有该表的八级锁。
- 该操作的耗时长度与DELTA表上的数据量有关，务必打开HStore的清理线程，确保HStore表的及时清理。
- 仅在9.1.0及以上集群版本支持第二个可选参数partitionName，因此9.1.0及以上集群版本不支持使用call调用此函数（ call不支持重载 ）。

示例：

```
SELECT hstore_full_merge('reason_select'，'part1');
```

pgxc_get_small_cu_info(rel_name text, row_count int)

描述：该函数用于获取目标表的小CU信息。该函数仅8.2.1.200及以上集群版本支持。

row_count为可选参数，表示小CU的阈值，存活元组数小于该参数值会被认为是小CU，默认值为200。

返回值类型：record

 版权所有 © 华为云计算技术有限公司

返回值：

- node_name：DN节点名。
- part_name：分区名，非分区表此列为空。
- zero_cu_count：0CU的数量。当一个CU中的所有数据都被删除时，称之为0 CU。
- small_cu_count：小CU数量。当一个CU中**有存活数据**且**存活数量小于阈值**时，称之为小CU。
- total_cu_count：总的CU的数量。
- sec_part_cu_num：每个二级分区的CU数量。当secondary_part_column被指定时，该列才会显示。该字段仅8.3.0及以上集群版本支持。

说明

此处CU是跨列的概念，并非一列一个的CU。

示例：

```
SELECT * FROM pgxc_get_small_cu_info('hs');
node_name | part_name | zero_cu_count | small_cu_count | total_cu_count |
sec_part_cu_num
-----------+-----------+---------------+----------------+----------------+-----------------------------------------
 datanode1 |           |             0 |              4 |              4 | p1:1 p2:0 p3:1 p4:0 p5:1 p6:0 p7:1 p8:0
 datanode2 |           |             0 |              4 |              4 | p1:0 p2:1 p3:0 p4:1 p5:0 p6:1 p7:0 p8:1
(2 rows)
```

gs_hstore_compaction(rel_name text, row_count int)

描述：该函数用于合并目标表的小CU。该函数仅8.2.1.200及以上集群版本支持。

row_count为可选参数，表示小CU的阈值，存活元组数小于该参数值会被认为是小CU，默认值100。

返回值类型：int

返回值：numCompactCU: 合并小CU的数量。

说明

- 此处CU是跨列的概念，并非一列一个的CU。
- 该函数不支持传入分区名，即不支持指定单分区。

示例：

```
SELECT gs_hstore_compaction('hs'，10);
```

pgxc_get_hstore_delta_info(rel_name text)

描述：该函数用于获取目标表的delta表信息，包括delta表的大小，insert/delete/update各种类型记录的数量等。该函数仅8.2.1.100及以上集群版本支持。

返回值类型：record

返回值：

- node_name：DN节点名。
- part_name：分区名，非分区表此列为non partition table。

 版权所有 © 华为云计算技术有限公司

- live_tup：存活的元组数量。
- n_ui_type：type是ui (小cu合并和upsert走update插入) 的记录的数量。一条ui记录表示一次插入，可以是单插或者批插。该参数仅8.3.0.100及以上版本支持。
- n_i_type：type是i (insert) 的记录的数量。一条i记录表示一次插入，可以是单插或者批插。
- n_d_type：type是d (delete) 的记录的数量。一条d记录表示一次删除，可以是单条删或者批量删除。
- n_x_type：type是x (由update产生的delete) 记录的数量。
- n_u_type：type是u (轻量化update) 的记录数量。
- n_m_type：type是m (merge) 的记录数量。
- data_size：delta表的总大小，单位为字节（包括delta上的索引与toast数据的大小）。

示例：

```
SELECT * FROM pgxc_get_hstore_delta_info('hs_part');
 node_name | part_name | live_tup | n_ui_type | n_i_type | n_d_type | n_x_type | n_u_type | n_m_type |
data_size
-----------+-----------+----------+-----------+----------+----------+----------+----------+----------+-----------
 dn_1      | p1        |        2 |         0 |        2 |        0 |        0 |        0 |        0 |      8192
 dn_1      | p2        |        2 |         0 |        2 |        0 |        0 |        0 |        0 |      8192
 dn_1      | p3        |        2 |         0 |        2 |        0 |        0 |        0 |        0 |      8192
(3 rows)
```

pgxc_get_binlog_sync_point(rel_name text, slot_name text, checkpoint bool, node_id int)

描述：该函数用于从pg_binlog_slots系统表上获取槽位对应的同步点信息，只能对开启binlog或binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值：

- node_name：节点名。
- node_id：节点ID。
- last_sync_point：上次同步点。
- latest_sync_point：当前最新同步点。
- xmin：同步点对应xmin。

示例：

```
SELECT * FROM pg_catalog.pgxc_get_binlog_sync_point('hstore_binlog_source', 'slot1', false, 0);
 node_name |   node_id   | last_sync_point | latest_sync_point | xmin
-----------+-------------+-----------------+-------------------+-------
 dn_2      | -1051926843 |               0 |             10512 | 10507
 dn_1      | -1300059100 |               0 |             10512 | 10508
(2 rows)
```

pgxc_get_binlog_changes(rel_name text, node_id int, start_csn bigint, end_csn bigInt)

描述：该函数用于获取目标表在指定DN上指定同步点区间的增量数据(node_id给0表示指定所有DN)，只能对开启binlog或binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

返回值类型：record

返回值：

- gs_binlog_sync_point ： 同步点。
- gs_binlog_event_sequence：用于表示同一事务内的先后顺序。
- gs_binlog_event_type：binlog类型。
- gs_binlog_timestamp_us：binlog记录的时间戳，对于enable_binlog_timestamp为false的binlog表，该列返回空。
- value columns：目标表上各个用户字段的数据。

示例：

```
SELECT * FROM pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
 gs_binlog_sync_point | gs_binlog_event_sequence | gs_binlog_event_type | gs_binlog_timestamp_us | c1  | c2
| c3
----------------------+--------------------------+----------------------+------------------------+-----+----+----
              10516 |                        2 | I                    |      1731570520900211 | 100 |  1 |  1
              10517 |                        3 | d                    |      1731570520904425 | 100 |  1 |  1
              10518 |                        2 | I                    |      1731570520909055 | 200 |  1 |  1
              10519 |                        3 | B                    |      1731570520914102 | 200 |  1 |  1
              10519 |                        4 | U                    |      1731570520914154 | 200 |  2 |  1
```

pgxc_register_binlog_sync_point(rel_name text, slot_name text, node_id int, end_csn bigInt, checkpoint bool, xmin bigint)

描述：该函数用于登记同步点，只能对开启binlog或binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：int

返回值：登记成功的节点数量。

示例：

```
SELECT pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, false, 100);
 pgxc_register_binlog_sync_point
---------------------------------
                               2
(1 row)
```

pgxc_consumed_binlog_records(rel_name text, node_id int)

描述：该函数用于获取目标表在指定DN上的消费状态，只能对开启binlog或binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

其中，node_id为0，表示查询所有DN。

返回值类型：int

返回值：

- 0，表示目标表的binlog仍没有被消费完毕（包含所有槽位以及checkpoint同步点）。
- 1，表示目标表的binlog被全部消费完毕。

示例：

```
SELECT * FROM pgxc_consumed_binlog_records('hstore_binlog_source',0);
 pgxc_consumed_binlog_records
```

 版权所有 © 华为云计算技术有限公司

```
------------------------------
                           1
(1 row)
```

pgxc_get_binlog_slots(rel_name text, node_id int)

描述：该函数用于获取目标表所有槽位的进度信息。该函数仅9.1.0.223及以上集群版本支持。

返回值类型：record

其中，node_id为0，表示查询所有DN。

返回值：

- node_name：节点名。
- slot_name：槽位名。
- sync_point： 该槽位最近一次注册的同步点。
- sync_timestamp：该槽位最近一次注册的时间。
- oldest_xmin：该槽位最近一次注册时对应oldest_xmin。
- is_checkpoint：是否属于checkpoint点位。

示例：

```
SELECT * FROM pgxc_get_binlog_slots('binlog_view_part_t1', 0);
 node_name | slot_name | sync_point |         sync_timestamp          | oldest_xmin |  is_checkpoint
-----------+-----------+------------+---------------------------------+-------------+-----------------
 dn_1      |   slot1   |    210059  | 2025-09-19 10:31:38.894357+08 |       10054 |  f
(2 rows)
```

pgxc_binlog_clear_target_slot(rel_name text, slot_name text)

描述：该函数用于清理目标表的指定槽位。该函数仅9.1.0.223及以上集群版本支持。

返回值类型：int

返回值：返回所有节点被清理的槽位数量之和。

示例：

```
SELECT * FROM pgxc_binlog_clear_target_slot('binlog_view_part_t1',  'slot1');
 pgxc_binlog_clear_target_slot
------------------------------
                           4
(1 row)
```

pgxc_get_binlog_cursor_by_timestamp(rel_name text, timestamp timestampTz, node_id int)

描述：该函数用于获取目标表上指定时间点后的第一个binlog记录的信息，只能对开启binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值：

- node_name：节点名。
- node_id：节点ID。

 版权所有 © 华为云计算技术有限公司

- latest_sync_point：当前最新同步点。
- binlog_sync_point：时间点后的第一个binlog记录的同步点。
- binlog_timestamp_us：时间点后的第一个binlog记录的时间戳。
- binlog_xmin：时间点后的第一个binlog记录的xmin。

示例：

```
SELECT * FROM pgxc_get_binlog_cursor_by_timestamp('hstore_binlog_source','2024-11-14
15:48:40.900211+08', 0);
 node_name |   node_id   | latest_sync_point | binlog_sync_point | binlog_timestamp_us | binlog_xmin
-----------+-------------+-------------------+-------------------+---------------------+-------------
 dn_2      | -1051926843 |             10532 |             10516 |    1731570520900211 |       10510
 dn_1      | -1300059100 |             10532 |             10518 |    1731570520909055 |       10510
(2 rows)
```

pgxc_get_binlog_cursor_by_syncpoint(rel_name text, csn int8, node_id int)

描述：该函数用于获取目标表上指定同步点后的第一个binlog记录的信息，只能对开启binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值：

- node_name：节点名。
- node_id：节点ID。
- latest_sync_point：当前最新同步点。
- binlog_sync_point：同步点后的第一个binlog记录的同步点。
- binlog_timestamp_us：同步点后的第一个binlog记录的时间戳。
- binlog_xmin：同步点后的第一个binlog记录的xmin。

示例：

```
SELECT * FROM pgxc_get_binlog_cursor_by_syncpoint('hstore_binlog_source',10516,0);
 node_name |   node_id   | latest_sync_point | binlog_sync_point | binlog_timestamp_us | binlog_xmin
-----------+-------------+-------------------+-------------------+---------------------+-------------
 dn_1      | -1300059100 |             11187 |             10518 |    1731570520909055 |       10510
 dn_2      | -1051926843 |             11187 |             10516 |    1731570520900211 |       10510
(2 rows)
```

pgxc_get_binlog_consume_progress(rel_name text, node_id int)

描述：该函数用于获取目标表的消费进度信息，只能对开启binlog时间戳的表使用。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值：

- node_name：节点名。
- node_id：节点ID。
- slot_name：槽位名。
- checkpoint：是否属于checkpoint点位。
- latest_consumed_timestamp：已消费的最新的binlog的时间戳。

 版权所有 © 华为云计算技术有限公司

- latest_timestamp：所有binlog里最新的时间戳。
- latest_consumed_csn: 已消费的最新的binlog的csn点位。
- latest_csn：所有binlog里最新的csn点位。
- unconsumed_binlog_count：还未消费的binlog的数量。

示例：

```
SELECT * FROM pgxc_get_binlog_consume_progress('hstore_binlog_source', 0);
 node_name |   node_id   | slot_name | checkpoint | latest_consumed_timestamp |    latest_timestamp    |
latest_consumed_csn | latest_csn | unconsumed_binlog_count
-----------+-------------+-----------+------------+---------------------------+------------------------
+---------------------+------------+-------------------------
 dn_1      | -1300059100 | slot1     | f          | 2025-04-15 15:55:55+08    | 2025-04-15 15:55:55+08 |
417725 |     417725 |                     0
 dn_1      | -1300059100 | slot1     | t          | 2025-04-15 15:55:55+08    | 2025-04-15 15:55:55+08 |
417725 |     417725 |                     0
(2 rows)
```

pgxc_get_cstore_dirty_ratio(rel_name text, partition_name)

描述：该函数用于获取各个DN上目标表的cu、delta以及cudesc的脏页率以及大小，仅支持HStore_opt表。该函数仅9.1.0.100及以上集群版本支持。

partition_name为可选参数，如果传入分区名则只返回该分区对应的信息；如果没传入分区名且是主表时分区表返回所有分区对应的信息。

返回值类型：record

返回值：

- node_name：DN节点名。
- database_name：表所在数据库名称。
- rel_name：主表名。
- part_name：分区名。
- cu_dirty_ratio：cu文件的脏页率。
- cu_size：cu文件大小。
- delta_dirty_ratio：delta表脏页率。
- delta_size：delta表大小。
- cudesc_dirty_ratio：cudesc表脏页率。
- cudesc_size：cudesc表大小。

示例：

```
SELECT * FROM pgxc_get_cstore_dirty_ratio('hs_opt_part');
 node_name | database_name |      rel_name       | partition_name | cu_dirty_ratio | cu_size | delta_dirty_ratio
| delta_size | cudesc_dirty_ratio | cudesc_size
-----------+---------------+---------------------+----------------+----------------+---------+-------------------
+------------+--------------------+-------------
 dn_1      | postgres      | public.hs_opt_part | p1             |              0 |       0 |               0 |      16384
|               0 |       24576
 dn_1      | postgres      | public.hs_opt_part | p2             |              0 |       0 |               0 |      16384
|               0 |       24576
 dn_1      | postgres      | public.hs_opt_part | p3             |              0 |       0 |               0 |      16384
|               0 |       24576
 dn_1      | postgres      | public.hs_opt_part | p4             |              0 |       0 |               0 |      16384
|               0 |       24576
 dn_1      | postgres      | public.hs_opt_part | other          |              0 | 1105920 |               0 |     524288
|               0 |       40960
```

 版权所有 © 华为云计算技术有限公司

17.5 实时数仓 Binlog

17.5.1 订阅实时数仓 Binlog

Binlog 使用介绍

当用户需要捕获数据库事件用于数据增量导出Flink等第三方组件，并协同完成数据加工等任务时，DWS实时数仓中的HStore表提供了Binlog功能，通过消费Binlog数据来实现上下游的数据同步，提高数据加工的效率。

传统的数据比如MySQL数据库等，支持通过Binlog来记录数据库中所有数据的变化，但相比于MySQL的Binlog主要用于数据恢复与主从复制，DWS实时数仓Binlog一般只用于实时场景下的数据同步。同时DWS实时数仓Binlog并不会记录DDL操作，只记录INSERT/DELETE/UPDATE(UPSERT)等DML操作。

DWS的Binlog的优点如下：

- 表级按需开关：按需给指定表打开或关闭binlog功能，更为灵活。
- 全增量一体消费：支持Flink任务启动后，先全量同步source，再实时消费source端增量。
- 支持消费即清理：对于空间敏感且只关注实时同步与加工的客户，支持消费后即开始异步清理增量，有效减少空间使用。

利用Flink强大的实时处理能力和DWS的Binlog能力，可以快速构建实时数仓，且无需维护其他组件（如kafka），整体架构分层清晰，数据可以高效流动，并且整体任务链路都可以通过Flink SQL来驱动，便于业务人员理解和使用。

约束与限制

1. 当前仅8.3.0.100及以上版本支持HStore和Hstore_opt记录Binlog功能，且V3表处于试商用阶段，使用前需要联系技术支持进行评估。
2. 使用Binlog的前置条件是必须存在主键约束，并且为HStore表或者Hstore_opt表，分布方式只能是Hash分布。
3. Binlog表仅记录INSERT/DELETE/UPDATE(UPSERT)等DML操作进行记录，不会记录DDL。
4. 当前Binlog表不支持的操作：INSERT OVERWRITE、修改分布列、给临时表开启Binlog、EXCHANGE/MERGE/SPLIT PARTITION。
5. 当前Binlog表并不限制用户进行以下DDL操作，但进行操作后会导致增量数据与同步点位信息会被清空，需要评估后再执行:

   ADD COLUMN增加列、DROP COLUMN删除列、SET TYPE修改列、TRUNCATE清空表数据。
6. Binlog表在线或者离线扩容期间会等待Binlog记录的消费，只有Binlog记录消费完毕才可以继续进行接下来的扩缩容步骤，默认等待时间为1小时，如果等待超时或者等待出错都会退出扩缩容过程，认为该表扩缩容失败。
7. VACUUM FULL Binlog表时，会等待Binlog记录的消费，只有Binlog记录消费完毕才可以进行接下来的VACUUM FULL操作，默认等待时间为1小时，如果等待超时或者等待出错都会退出VACUUM FULL过程，认为该表VACUUM FULL失败。且由于需要等待Binlog记录消费完毕，所以即使VACUUM FULL一个分区表，也会对分区的主表上7级锁，阻塞整个表的插入更新或者删除。

版权所有 © 华为云计算技术有限公司

8. Binlog表在备份恢复期间，仅会被当做普通HStore表进行备份，恢复后辅助表的增量数据与同步点信息会清空，需要重新开始同步。
9. 支持Binlog时间戳功能，通过设置**enable_binlog_timestamp**打开，同样只有HStore和Hstore_opt两种表支持打开。该约束仅9.1.0.200及以上版本支持。

Binlog 格式与原理

表 17-6 binlog 字段格式

字段名称	字段类型	含义
gs_binlog_sync_point	BIGINT	Binlog系统字段，表示该记录的同步点值，普通GTM模式下，该值唯一且有序。
gs_binlog_event_sequence	BIGINT	Binlog的系统字段， 用于表示同一事务类操作的先后顺序。
gs_binlog_event_type	CHAR	Binlog的系统字段，表示当前记录的操作类型。 type可能有以下几种取值： • 'I' 即INSERT， 表示当前Binlog是插入一条新记录。 • 'd' 即DELETE，表示当前Binlog是删除一条记录。 • 'B' 即BEFORE_UPDATE，表示当前Binlog是更新前的记录。 • 'U'即AFTER_UPDATE，表示当前Binlog是更新后的记录。
gs_binlog_timestamp_us	BIGINT	Binlog的系统字段，表示当前记录入库时的时间戳。 只有开启binlog时间戳功能时会有，没开启binlog时间戳时为空。仅9.1.0.200及以上版本支持。
user_column_1	用户列	用户的自定义数据列。
...	...	...
user_column_n	用户列	用户的自定义数据列。

版权所有 © 华为云计算技术有限公司

说明

- UPDATE（或者UPSERT触发的更新操作）会产生两条Binlog，分别是BEFORE_UPDATE类型与AFTER_UPDATE类型，其中BEFORE_UPDATE主要用于保证Flink等第三方组件做数据加工后加工结果的正确性。
- 对于用户UPDATE/DELETE操作产生的BEFORE_UPDATE以及DELETE类型的Binlog，DWS实时数仓中这两类Binlog类型并不会在操作执行时就反查出所有用户列并填充，保证了入库的性能。
- 当在DWS实时数仓给某个HStore表开启Binlog功能时， 本质就是给HStore表再创建一张辅助表（也通过HStore实现），该辅助表包含gs_binlog_event_sync_point、gs_binlog_event_event_sequence、gs_binlog_event_type三个系统列以及一个将所有用户列序列化存储到一个字段的value列。
- 当开启binlog时间戳功能时（表级参数**enable_binlog_timestamp**），辅助表上记录的binlog记录会严格保留至超过TTL才会清理，这会带来数倍的额外空间开销（具体倍数与TTL时间内的更新入库量有关）。当开启普通binlog时（表级参数enable_binlog），辅助表上记录的binlog，只要被下游消费就会允许异步清理，能大幅度减少空间占用。仅9.1.0.200及以上版本支持。

开启 Binlog

- 通过在建HStore_opt表时指定表级参数enable_binlog，开启HStore_opt表的Binlog功能。

```
CREATE TABLE hstore_binlog_source (
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true,
    enable_binlog=on,
    binlog_ttl = 86400
);
```

说明

- 对于开启了Binlog的表，并不会立刻给入库的操作记录Binlog，需要再给同步任务注册同步点后，才会开始记录Binlog（开启Flink同步binlog任务后，会自动循环进行获取同步点、获取增量数据、注册同步点操作）。
  - binlog_ttl是可选参数，当不设置时将使用默认值86400， 单位为秒，当同步任务注册的同步点超过TTL没有进行增量同步时，该同步点位将被清理。最老的同步点位之前的Binlog（即被所有任务消费了的Binlog）会被异步清理来回收空间。
- 空间开销：对于开启普通binlog的表，如果能保证增量被下游及时消费，那么空间就能被及时清理回收。

- 通过执行ALTER命令给已有的HStore表开启binlog功能：

```
CREATE TABLE hstore_binlog_source (
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true
);
ALTER TABLE hstore_binlog_source SET (enable_binlog=on);
```

查询 binlog

- 通过DWS提供的系统函数，可以直接查询目标表在指定DN上binlog信息，以及是否被下游消费完毕等信息。

```
-- 模拟Flink调用系统函数获取同步点，参数分别表示 表名、槽位名、是否checkPoint点位，目标DN(为0表示所有DN)。
```

 版权所有 © 华为云计算技术有限公司

```
select * from pg_catalog.pgxc_get_binlog_sync_point('hstore_binlog_source', 'slot1', false, 0);
select * from pg_catalog.pgxc_get_binlog_sync_point('hstore_binlog_source', 'slot1', true, 0);
-- 进行增删改产生增量binlog。
INSERT INTO hstore_binlog_source VALUES(100, 1, 1);
delete hstore_binlog_source where c1 = 100;
INSERT INTO hstore_binlog_source VALUES(200, 1, 1);
update hstore_binlog_source set c2 =2 where c1 = 200;
-- 模拟Flink调用系统函数查询指定CSN区间的Binlog，参数分别表示表名，目标DN(为0表示所有DN),起始CSN点位, 终止CSN点位。
select * from pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
```

```
postgres=# select * from pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
 gs_binlog_sync_point | gs_binlog_event_sequence | gs_binlog_event_type | gs_binlog_timestamp_us | c1  | c2  | c3
----------------------+--------------------------+----------------------+------------------------+-----+-----+----
                10241 |                        2 | I                    |                        | 100 |   1 |  1
                10242 |                        3 | d                    |                        | 100 |   1 |  1
                10243 |                        4 | I                    |                        | 100 |   1 |  1
                10245 |                        5 | B                    |                        | 100 |   1 |  1
                10245 |                        6 | U                    |                        | 100 | 100 |  1
(5 rows)
```

可以看到两次INSERT操作产生了两个gs_binlog_event_type是'I'的记录，DELETE操作产生了type是'd'的记录，UPDATE产生了一行BeforeUpdate的'B'记录以及一条AfterUpdate的'U'记录，分别表示更新前的值以及更新后的值。

- 通过查询函数**pgxc_get_binlog_slots(rel_name text, node_id int)**可以获取目标表所有slot的同步信息。入参分别表示目标表名及目标DN（为0表示查询所有DN）。

```
postgres=# select * from pgxc_get_binlog_slots('binlog_view_part_t1', 0);
 node_name | slot_name | sync_point |         sync_timestamp          | oldest_xmin | is_checkpoint
-----------+-----------+------------+---------------------------------+-------------+---------------
 dn_1      | slot1     |     210059 | 2025-09-19 10:31:38.894357+08 |       10054 | f
(1 row)

postgres=#
```

- 通过调用系统函数**pgxc_consumed_binlog_records**可以查询目标表的binlog是否被所有槽位消费完毕。参数分别表示目标表名以及目标DN（为0表示查询所有DN）。

```
-- 模拟Flink调用系统函数注册同步点，参数分别表示表名，槽位名，注册的点位，是否属于checkPoint, 点位对应的xmin (获取同步点时会提供)。
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, false, 100);
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, true, 100);
-- 查询表上binlog是否被全部消费，返回 1 表示已经被下游槽位全部消费。
select * from pgxc_consumed_binlog_records('hstore_binlog_source',0);
```

```
postgres=# select * from pgxc_consumed_binlog_records('hstore_binlog_source',0);
 pgxc_consumed_binlog_records
------------------------------
                            1
(1 row)
```

- 通过调用系统函数**pgxc_binlog_clear_target_slot(rel_name text, slot_name text)**可以清理指定表上的指定槽位。参数分别表示目标表名及目标槽位名。

开启 Binlog 时间戳功能

如果需要读取指定时间点之后binlog的功能，通过在建HStore表时指定表级参数enable_binlog_timestamp，开启HStore表的Binlog时间戳功能。仅9.1.0.200及以上版本支持。

```
CREATE TABLE hstore_binlog_source(
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true,
    enable_binlog_timestamp =on,
    binlog_ttl = 86400
);
```

 版权所有 © 华为云计算技术有限公司

说明

- 对于开启了Binlog时间戳的表，并不会立刻给入库的操作记录Binlog，需要再给同步任务注册同步点后，才会开始记录Binlog（开启Flink同步binlog任务后，会自动循环进行获取同步点、获取增量数据、注册同步点操作）。
- binlog_ttl是可选参数，当不设置时将使用默认值86400， 单位为秒（即默认保留一天），当Binlog记录的时间戳距离现在大于TTL时，会被异步清理。
- 空间开销：对于开启binlog时间戳的表，辅助表上记录的binlog记录会严格保留至超过TTL才会清理，这会带来数倍的额外空间开销（具体倍数与TTL时间内的更新入库量有关）。

- 查询开启binlog时间戳功能的表上的Binlog。

```
postgres=# select * from pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
 gs_binlog_sync_point | gs_binlog_event_sequence | gs_binlog_event_type | gs_binlog_timestamp_us | c1  | c2 | c3
----------------------+--------------------------+----------------------+------------------------+-----+----+----
                10516 |                        2 | I                    |       1731570520900211 | 100 |  1 |  1
                10517 |                        3 | d                    |       1731570520904425 | 100 |  1 |  1
                10518 |                        2 | I                    |       1731570520909055 | 200 |  1 |  1
                10519 |                        3 | B                    |       1731570520914102 | 200 |  1 |  1
                10519 |                        4 | U                    |       1731570520914154 | 200 |  2 |  1
(5 rows)
```

将gs_binlog_timestamp_us从BigInt类型转成可读的时间戳：

```
SELECT to_timestamp(1731569598408661/1000000);
```

```
postgres=#  select to_timestamp(1731570520900211/1000000);
          to_timestamp
--------------------------------
 2024-11-14 15:48:40.900211+08
(1 row)
```

- 获取各个DN上，目标表指定时间点后的第一条binlog信息（为空表示这个时间点后没有binlog）。

```
SELECT * FROM pgxc_get_binlog_cursor_by_timestamp('hstore_binlog_source','2024-11-14 15:33:18.40866+08', 0);
```

```
postgres=# select * from pgxc_get_binlog_cursor_by_timestamp('hstore_binlog_source','2024-11-14 15:48:40.900211+08', 0);
 node_name |   node_id   | latest_sync_point | binlog_sync_point | binlog_timestamp_us | binlog_xmin
-----------+-------------+-------------------+-------------------+---------------------+-------------
 dn_2      | -1051926843 |             10532 |             10516 |     1731570520900211 |       10510
 dn_1      | -1300059100 |             10532 |             10518 |     1731570520909055 |       10510
(2 rows)
```

- 获取开启binlog时间戳功能的表的消费进度。

  返回的字段表示最近消费的一条binlog的时间戳，binlog上的最近时间戳，最近消费的一条binlog的CSN点位， binlog上最近CSN点位，未消费的binlog记录数量。

  查询目标表各个槽位的消费进度。

```
SELECT * FROM pgxc_get_binlog_consume_progress('hstore_binlog_source', 0);
```

```
postgres=# select * from pgxc_get_binlog_consume_progress('hstore_binlog_source', 0);
 node_name |   node_id   | slot_name | checkpoint | latest_consumed_timestamp |    latest_timestamp    | latest_consumed_csn | latest_csn | unconsumed_binlog_count
-----------+-------------+-----------+------------+---------------------------+------------------------+---------------------+------------+-------------------------
 dn_1      | -1300059100 | slot1     | f          | 2024-11-14 15:48:40+08    | 2024-11-14 15:48:40+08 |               10519 |      10519 |                       0
 dn_1      | -1300059100 | slot1     | t          | 2024-11-14 15:48:40+08    | 2024-11-14 15:48:40+08 |               10519 |      10519 |                       0
 dn_2      | -1051926843 | slot1     | f          | 2024-11-14 15:48:40+08    | 2024-11-14 15:48:40+08 |               10517 |      10517 |                       0
 dn_2      | -1051926843 | slot1     | t          | 2024-11-14 15:48:40+08    | 2024-11-14 15:48:40+08 |               10517 |      10517 |                       0
(4 rows)
```

控制 DML 不产生 Binlog

通过设置会话级参数**enable_generate_binlog**为off，可以控制当前会话的DML，在给开启binlog的表入库时，不产生binlog记录。

17.5.2 Flink 实时消费 Binlog

注意事项

- 当前仅8.3.0.100及以上的版本支持HStore和Hstore_opt记录Binlog功能。
- V3 HStore表不支持Binlog，仅V3 HStore_opt表支持Binlog，且V3处于试商用阶段，使用前需要进行评估。

版权所有 © 华为云计算技术有限公司

- 目前DWS只有HStore表和Hstore_opt支持Binlog功能，表需要包含主键且设置enable_binlog=on或者enable_binlog_timestamp=on。
- 消费的Binlog表名不要带有特殊字符，如.、""等。
- 如果多个任务消费同一张表的Binlog数据，需要保证每个任务的binlogSlotName唯一。
- 为了达到最高的消费速度，建议将任务的并发度和DWS集群DN数设置一致。
- 使用《数据仓库服务工具指南》中的“dws-connector-flink”的Sink能力来写入Binlog数据的话，需要注意以下几点：
  - 如果需要保证DN内的数据写入顺序则需要设置connectionSize设置为1。
  - 如果源端有更新主键操作或者需要flink进行聚合计算的话，将ignoreUpdateBefore设置为false，否则不建议将ignoreUpdateBefore设置为false（默认true）。

Flink 实时消费 Binlog

使用DWS Connector来实时消费Binlog，具体请参见《数据仓库服务工具指南》中的“DWS-Connector”。

如果已使用其他同步工具已经将全量数据同步到了目标端，后续只想进行增量同步。则可以调用以下系统函数来更新同步点。

```
SELECT * FROM pg_catalog.pgxc_register_full_sync_point('table_name', 'slot_name');
```

源表 DDL

Source端会根据操作类型自动为每行数据设置准确的Flink RowKind类型（INSERT、DELETE、UPDATE_BEFORE、UPDATE_AFTER)，这样就能镜像同步表的数据，类似MySQL和Postgres的CDC功能。

```
CREATE TABLE test_binlog_source (
   a int,
   b int,
   c int,
   primary key(a) NOT ENFORCED
) with (
   'connector' = 'dws',
   'url' = 'jdbc:gaussdb://ip:port/gaussdb',
   'binlog' = 'true',
   'tableName' = 'test_binlog_source',
   'binlogSlotName' = 'slot',
   'username'='xxx',
   'password'='xxx')
```

Binlog 相关参数说明

下表仅涉及消费Binlog时的参数。

表 17-7 消费 Binlog 时的参数

参数	说明	数据类型	默认值
binlog	是否读取Binlog信息	Boolean	false

 版权所有 © 华为云计算技术有限公司

参数	说明	数据类型	默认值
binlogSlotName	槽位信息，可以理解一个标识。由于可能存在多个Flink任务同时消费同一张表的Binlog信息，所以该场景需要保证每个任务的binlogSlotName不同。 如果指定了binlog，但没有指定binlogSlotName时会默认使用Flink映射表的表名，确保增量数据的正确读取。	String	Flink映射表的表名
binlogBatchReadSize	批量读取binlog的数据行数	Integer	5000
fullSyncBinlogBatchReadSize	全量读取binlog的数据行数	Integer	50000
binlogReadTimeout	增量消费Binlog数据时超时时间，单位毫秒	Integer	600000
fullSyncBinlogReadTimeout	全量消费Binlog数据时超时时间，单位毫秒	Long	1800000
binlogSleepTime	实时消费不到Binlog数据时休眠时间，单位毫秒。如果连续读取不到Binlog数据，则休眠时间为：binlogSleepTime * 次数，最大为binlogMaxSleepTime。读取到数据后，则重置。	Long	500
binlogMaxSleepTime	实时消费不到Binlog数据时最大休眠时间，单位毫秒。	Long	10000
binlogMaxRetryTimes	消费Binlog数据出错后的重试次数。	Integer	1
binlogRetryInterval	消费binlog数据出错后的重试时间间隔。重试时sleep时间：binlogRetryInterval * (1~binlogMaxRetryTimes) +Random（100）。单位毫秒。	Long	100
binlogParallelNum	消费Binlog数据时线程数，只有任务并发度小于DWS集群DN数时，该参数才有效，即此时一个并发度会消费多个DN上的数据，所以可以考虑设置该参数。	Integer	3
connectionPoolSize	JDBC连接池连接大小。	Integer	5

版权所有 © 华为云计算技术有限公司

参数	说明	数据类型	默认值
needRedistribution	表示是否兼容扩充重分布（需要升级到对应内核版本，如果是低版本则设置为false）；如果设置成true的话，flink的restart-strategy不能设置为none。	Boolean	true
newSystemValue	表示读取binlog数据时是否使用新的系统字段（需要升级到对应内核版本，如果是低版本则设置为false）。	Boolean	true
checkNodeChangeInterval	检测节点变化的间隔，只有needRedistribution=true才生效。	Long	10000
connectionSocketTimeout	连接处理超时时间（可以看成客户端执行SQL超时时间），单位毫秒；默认值为0，即不设置超时时间。	Integer	0
binlogIgnoreUpdateBefore	是否过滤Binlog记录中的before_update记录，以及delete记录是否只返回主键信息。该参数仅9.1.0.200及以上版本支持。	Boolean	false
binlogStartTime	设置从某个时间点开始消费Binlog（只能增量消费），格式为yyyy-MM-dd hh:mm:ss且表需要开启**enable_binlog_timestamp**。 该参数仅9.1.0.200及以上版本支持。	String	无
binlogSyncPointSize	增量读取binlog同步点区间的大小（增量读取binlog时，如果数据量过大可能涉及下盘，可通过调整该参数控制）。 该参数仅9.1.0.200及以上版本支持。	Integer	5000

数据同步示例

- DWS侧：

  说明

  新建binlog表时，**enable_hstore_binlog_table**参数需要设置为true，可以通过show enable_hstore_binlog_table来查询。

  -- 源表（产生binlog）

  ```
  CREATE TABLE test_binlog_source(a int, b int, c int, primary key(a)) with(orientation=column, enable_hstore_opt=on, enable_binlog=true);
  ```

  -- 目标表

版权所有 © 华为云计算技术有限公司

```
CREATE TABLE test_binlog_sink(a int, b int, c int, primary key(a)) with(orientation=column,
enable_hstore_opt=on);
```

- Flink侧：

  执行如下命令进行完整数据同步：

```
-- 建立源表的映射表
CREATE TABLE test_binlog_source (
  a int,
  b int,
  c int,
  primary key(a) NOT ENFORCED
) with (
  'connector' = 'dws',
  'url' = 'jdbc:gaussdb://ip:port/gaussdb',
  'binlog' = 'true',
  'tableName' = 'test_binlog_source',
  'binlogSlotName' = 'slot',
  'username'='xxx',
  'password'='xxx');

-- 建立目标表的映射表
CREATE TABLE test_binlog_sink (
  a int,
  b int,
  c int,
  primary key(a) NOT ENFORCED
) with (
  'connector' = 'dws',
  'url' = 'jdbc:gaussdb://ip:port/gaussdb',
  'tableName' = 'test_binlog_sink',
  'ignoreUpdateBefore'='false',
  'connectionSize' = '1',
  'username'='xxx',
  'password'='xxx');

INSERT INTO test_binlog_sink select * from test_binlog_source;
```

使用 java 程序示例

新建源表和目标表：

```
-- source
create table binlog_test_source(a int, b int, c int, primary key(a)) with(orientation=column,
enable_hstore_opt=on, enable_binlog=true);
-- sink
create table binlog_test_sink(a int, b int, c int, primary key(a)) with(orientation=column,
enable_hstore_opt=on, enable_binlog=true);
```

demo程序：

```
public class BinlogDemo {

    // binlog表的表名
    private static final String BINLOG_TABLE_NAME = "binlog_test_source";

    // binlog表的槽位名
    private static final String BINLOG_SLOT_NAME = "binlog_test_slot";

    // 写入表的表名
    private static final String SINK_TABLE_NAME = "binlog_test_sink";

    public static void main(String[] args) throws Exception {
        DwsConfig dwsConfig = buildDwsConfig();
        DwsClient dwsClient = new DwsClient(dwsConfig);

        TableSchema sourceTableSchema =
dwsClient.getTableSchema(TableName.valueOf(BINLOG_TABLE_NAME));
        TableSchema sinkTableSchema = dwsClient.getTableSchema(TableName.valueOf(SINK_TABLE_NAME));
```

 版权所有 © 华为云计算技术有限公司

```
        // 需要写入哪些列
        List<String> sinkColumns = sinkTableSchema.getColumnNames();

        // 线程池
        DwsConnectionPool dwsConnectionPool = new DwsConnectionPool(dwsConfig);
        // 存放数据的队列
        BlockingQueue<BinlogRecord> queue = new LinkedBlockingQueue<>();
        // 需要同步哪些列
        List<String> sourceColumnNames = sourceTableSchema.getColumnNames();

        BinlogReader binlogReader = new BinlogReader(dwsConfig, queue, sourceColumnNames,
dwsConnectionPool);

        // 启动读取任务
        binlogReader.start();
        binlogReader.getRecords();

        while (binlogReader.isStart()) {
            try {
                while (!queue.isEmpty() && !binlogReader.hasException()) {
                    // 读取数据
                    BinlogRecord record = queue.poll();
                    if (Objects.isNull(record)) {
                        continue;
                    }
                    BinlogRecordType type = BinlogRecordType.toBinlogRecordType(record.getType());
                    List<Object> columnValues = record.getColumnValues();

                    // 写入数据
                    if (BinlogRecordType.INSERT.equals(type) || BinlogRecordType.UPDATE_AFTER.equals(type)) {
                        Operate upsert = dwsClient.write(sinkTableSchema);
                        for (int i = 0; i < sinkColumns.size(); i++) {
                            upsert.setObject(i, columnValues.get(i), false);
                        }
                        upsert.commit();
                    } else if (BinlogRecordType.DELETE.equals(type) ||
BinlogRecordType.UPDATE_BEFORE.equals(type)) {
                        Operate delete = dwsClient.delete(sinkTableSchema);
                        for (int i = 0; i < sinkColumns.size(); i++) {
                            String field = sinkColumns.get(i);
                            if (!sinkTableSchema.isPrimaryKey(field)) {
                                continue;
                            }
                            delete.setObject(i, columnValues.get(i), false);
                        }
                        delete.commit();
                    }
                }
                binlogReader.checkException();
            } catch (Exception e) {
                throw new DwsClientException(ExceptionCode.GET_BINLOG_ERROR, "get binlog  has error", e);
            }
        }
    }

    private static DwsConfig buildDwsConfig() {
        // 初始化一些配置信息(只列举一些必要的配置,具体配置信息请参考本节文档)
        TableConfig tableConfig = new TableConfig().withBinlog(true)
                .withNewSystemValue(true)
                .withNeedRedistribution(false)
                .withBinlogSlotName(BINLOG_SLOT_NAME);
        return DwsConfig.builder()
                .withUrl("链接信息")
                .withUsername("用户名")
                .withPassword("密码")
                .withBinlogTableName(BINLOG_TABLE_NAME)
                .withTableConfig(BINLOG_TABLE_NAME, tableConfig)
                .build();
```

 版权所有 © 华为云计算技术有限公司

```
    }
}
```

 版权所有 © 华为云计算技术有限公司

18 DWS 系统表和系统视图

18.1 系统表和系统视图概述

系统表是DWS存放结构元数据，是DWS数据库系统运行控制信息的来源，也是数据库系统的核心组成部分。系统表包含集群安装信息以及DWS上运行的各种查询和进程的信息。可以通过查询系统表来收集有关数据库的信息。

系统视图提供了查询系统表和访问数据库内部状态的方法。当用户对数据库中的一张或者多张表的某些字段的组合感兴趣，而又不想每次键入这些查询时，用户就可以定义一个视图来解决这个问题。视图与基本表不同，不是物理上实际存在的，是一个虚表。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。视图每次被引用的时候都会运行一次。

三权分立下，非管理员无权查看系统表和视图。非三权分立下，系统表和系统视图要么只对管理员可见，要么对所有用户可见。下面的系统表和视图有些标识了需要管理员权限，这些系统表和视图只有管理员可以查询。

须知

- 禁止对系统表或系统视图进行增删改等操作，手动对系统表或系统视图的修改或破坏可能会导致系统信息不一致，造成系统控制异常甚至出现集群不可用的情况。
- 系统表不支持toast，无法跨页存储，一个页面大小为8K，系统表各个字段长度需小于8K。

18.2 系统表

18.2.1 GS_BLOCKLIST_QUERY

GS_BLOCKLIST_QUERY系统表存储作业黑名单信息和异常信息，该表以unique_sql_id作为唯一索引，进行作业异常信息统计和黑名单记录，可通过与GS_WLM_SESSION_INFO进行关联获取作业的query字段和执行信息。

版权所有 © 华为云计算技术有限公司

须知

- GS_BLOCKLIST_QUERY系统表的schema为dbms_om。
- GS_BLOCKLIST_QUERY系统表仅限在**postgres**数据库中查询，其它数据库中查询会直接报错。
- GS_BLOCKLIST_QUERY系统表包含唯一索引，使用哈希分布方式分布在DN上，分布列为unique_sql_id。
- 通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个unique_sql_id可能会对应一个或多个查询。

DWS同时提供了**GS_BLOCKLIST_QUERY**视图用于查询作业黑名单和异常信息，该视图可直接显示query字段信息，不过因为该视图与GS_WLM_SESSION_INFO为依赖关系，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长的时间。

表 18-1 GS_BLOCKLIST_QUERY 字段

名字	类型	引用	描述
unique_sql_id	bigint	-	基于查询解析树生成的查询ID。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。

18.2.2 GS_BLOCKLIST_SQL

GS_BLOCKLIST_SQL系统表存储作业黑名单信息和异常信息，该表以sql_hash作为唯一索引，进行作业异常信息统计和黑名单记录，可通过与**GS_WLM_SESSION_INFO**进行关联获取作业的query字段和执行信息。

DWS同时提供了**GS_BLOCKLIST_SQL**视图用于查询作业黑名单和异常信息，该视图可直接显示query字段信息，不过因为该视图与GS_WLM_SESSION_INFO为依赖关系，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长的时间。

该系统表仅9.1.0.200及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

须知

- GS_BLOCKLIST_SQL系统表的schema为dbms_om。
- GS_BLOCKLIST_SQL系统表仅限在**postgres**数据库中查询，其它数据库中查询会直接报错。
- GS_BLOCKLIST_SQL系统表包含唯一索引，使用哈希分布方式分布在DN上，分布列为sql_hash。
- 通常对于DML语句，在计算sql_hash的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个sql_hash可能会对应一个或多个查询。

表 18-2 GS_BLOCKLIST_SQL 字段

名字	类型	引用	描述
sql_hash	text	-	基于查询解析树生成的sql_hash。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。

18.2.3 GS_OBSSCANINFO

GS_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 18-3 GS_OBSSCANINFO 字段

名字	类型	描述
query_id	bigint	查询标识。
user_id	text	执行该查询的数据库用户。
table_name	text	OBS外表的表名。
file_type	text	底层数据保存的文件格式。
time_stamp	timestamp with time zone	扫描操作开始的时间。
actual_time	double	扫描操作执行时间，单位为秒。
file_scanned	bigint	扫描的文件数量。
data_size	double	扫描的数据量，单位为字节。
billing_info	text	保留字段。

版权所有 © 华为云计算技术有限公司

18.2.4 GS_RESPOOL_RESOURCE_HISTORY

GS_RESPOOL_RESOURCE_HISTORY表记录资源池监控历史信息，CN和DN上均进行记录。

表 18-4 GS_RESPOOL_RESOURCE_HISTORY 字段

名称	类型	描述
timestamp	timestamp	资源池监控信息持久化时间。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5s监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为GaussDB可用CPU，CPU限额管控情况下为关联控制组CPU可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5s监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5s监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5s监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5s监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5s监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5s监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5s监控周期内网络发送平均速率，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
recv_speed	double	资源池5s监控周期内网络接收平均速率，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。

18.2.5 GS_WLM_INSTANCE_HISTORY

GS_WLM_INSTANCE_HISTORY系统表存储与实例(CN或DN)相关的资源使用相关信息。该系统表里每条记录都是对应时间点某实例资源使用情况，包括：内存、CPU核数、磁盘IO、进程物理IO和进程逻辑IO信息。

表 18-5 GS_WLM_INSTANCE_HISTORY 字段

名称	类型	描述
instancename	text	实例名称。
timestamp	timestamp with time zone	时间戳。
used_cpu	int	实例使用CPU所占用的百分比。
free_mem	int	实例未使用的内存大小，单位MB。
used_mem	int	实例已使用的内存大小，单位MB。
io_await	real	实例所使用磁盘的io_wait值（10秒均值）。
io_util	real	实例所使用磁盘的io_util值（10秒均值）。
disk_read	real	实例所使用磁盘的读速率（10秒均值），单位KB/s。
disk_write	real	实例所使用磁盘的写速率（10秒均值），单位KB/s。
process_read	bigint	实例对应进程从磁盘读数据的读速率(不包括从磁盘pagecache中读取的字节数，10秒均值)，单位KB/s。
process_write	bigint	实例对应进程向磁盘写数据的写速率(不包括向磁盘pagecache中写入的字节数，10秒均值)，单位KB/s。
logical_read	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读字节速率，单位KB/s。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
logical_write	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写字节速率，单位KB/s。
read_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读操作次数之和，单位次。
write_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写操作次数之和，单位次。

18.2.6 GS_WLM_OPERATOR_INFO

GS_WLM_OPERATOR_INFO系统表显示执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表中的数据。当设置GUC参数**enable_resource_record**为on时，系统会定时将**GS_WLM_OPERATOR_HISTORY**中的记录导入此系统表，开启此功能会占用系统存储空间并对性能有一定影响，建议在性能定位及监控任务完成后及时关闭。

须知

- GS_WLM_OPERATOR_INFO系统表的schema为dbms_om。
- GS_WLM_OPERATOR_INFO系统表仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

表 18-6 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

 版权所有 © 华为云计算技术有限公司

18.2.7 GS_WLM_SESSION_INFO

GS_WLM_SESSION_INFO系统表显示当前CN执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。当设置GUC参数**enable_resource_record**为on时，系统会定时将**GS_WLM_SESSION_HISTORY**中的记录导入此系统表，开启此功能会占用系统存储空间并对性能有一定影响，建议在性能定位及监控任务完成后及时关闭。

须知

- GS_WLM_SESSION_INFO系统表的schema为dbms_om。
- GS_WLM_SESSION_INFO系统表仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

GS_WLM_SESSION_INFO系统表的字段同**表18-162**相同，具体字段内容如下：

表 18-7 GS_WLM_SESSION_HISTORY 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。
outline_name	text	该语句计划对应的outline的名字。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

18.2.8 GS_WLM_USER_RESOURCE_HISTORY

GS_WLM_USER_RESOURCE_HISTORY系统表存储与用户使用资源相关的信息，该表在CN和DN上均存有数据。该系统表的每条记录都是对应时间点某用户的资源使用情况，包括：内存、CPU核数、存储空间、临时空间、算子落盘空间、逻辑IO流量、逻辑IO次数和逻辑IO速率信息。其中，内存、CPU、IO相关监控项仅记录用户复杂作业的资源使用情况。

GS_WLM_USER_RESOURCE_HISTORY系统表的数据来源于**PG_TOTAL_USER_RESOURCE_INFO**视图。

表 18-8 GS_WLM_USER_RESOURCE_HISTORY 字段

名称	类型	描述
username	text	用户名。
timestamp	timestamp with time zone	时间戳。
used_memory	int	用户使用的内存大小，单位：MB。 • DN：显示当前DN上用户使用的内存大小。 • CN：显示所有DN上用户使用内存的累积和。
total_memory	int	资源池使用的内存大小，单位：MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存max_dynamic_memory。具体的计算公式为： total_memory = max_dynamic_memory * parent_percent * user_percent CN：显示所有DN上用户可用内存上限的累积和。
used_cpu	real	正在使用的CPU核数。
total_cpu	int	该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的存储空间大小，单位KB。
total_space	bigint	可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。
used_temp_space	bigint	已使用的临时存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时存储空间大小，单位KB，值为-1表示未限制最大临时存储空间。
used_spill_space	bigint	已使用的算子落盘存储空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘存储空间大小，单位KB，值为-1表示未限制最大算子落盘存储空间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_kbytes	bigint	监控周期内，读操作的字节流量，单位KB。
write_kbytes	bigint	监控周期内，写操作的字节流量，单位KB。
read_counts	bigint	监控周期内，读操作的次数，单位次。
write_counts	bigint	监控周期内，写操作的次数，单位次。
read_speed	real	监控周期内，读操作的字节速率，单位KB/s。
write_speed	real	监控周期内，写操作的字节速率，单位KB/s。
send_speed	double	监控周期内，网络发送速率，单位KB/s。
recv_speed	double	监控周期内，网络接收速率，单位KB/s。

18.2.9 PG_AGGREGATE

PG_AGGREGATE系统表存储与聚集函数有关的信息。PG_AGGREGATE里的每条记录都是一条pg_proc里面的记录的扩展。PG_PROC记录承载该聚集的名字、输入和输出数据类型，以及其它一些和普通函数类似的信息。

表 18-9 PG_AGGREGATE 字段

名字	类型	引用	描述
aggfnoid	regproc	PG_PROC.oid	此聚集函数的PG_PROC OID。
aggtransfn	regproc	PG_PROC.oid	转换函数。
aggcollectfn	regproc	PG_PROC.oid	收集函数。
aggfinalfn	regproc	PG_PROC.oid	最终处理函数（如果没有则为0）。
aggsortop	oid	PG_OPERATOR.oid	关联排序操作符（如果没有则为0）。
aggtranstype	oid	PG_TYPE.oid	此聚集函数的内部转换（状态）数据的数据类型。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
agginitval	text	-	转换状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则转换状态值从null开始。
agginitcollect	text	-	收集状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则收集状态值从null开始。

18.2.10 PG_AM

PG_AM系统表存储有关索引访问方法的信息。系统支持的每种索引访问方法都有一行。

表 18-10 PG_AM 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amname	name	-	访问方法的名称。
amstrategies	smallint	-	访问方法的操作符策略个数，或者如果访问方法没有一个固定的操作符策略集则为0。
amsupport	smallint	-	访问方法的支持过程个数。
amcanorder	boolean	-	这种访问方式是否支持通过索引字段值的命令扫描排序。
amcanorderbyop	boolean	-	这种访问方式是否支持通过索引字段上操作符的结果的命令扫描排序。
amcanbackward	boolean	-	访问方式是否支持向后扫描。
amcanunique	boolean	-	访问方式是否支持唯一索引。
amcanmulticol	boolean	-	访问方式是否支持多字段索引。
amoptionalkey	boolean	-	访问方式是否支持第一个索引字段上没有任何约束的扫描。
amsearcharray	boolean	-	访问方式是否支持ScalarArrayOpExpr搜索。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
amsearchnulls	boolean	-	访问方式是否支持IS NULL/NOT NULL搜索。
amstorage	boolean	-	允许索引存储的数据类型与列的数据类型是否不同。
amclusterable	boolean	-	是否允许在一个这种类型的索引上集群。
ampredlocks	boolean	-	是否允许这种类型的一个索引管理细粒度的谓词锁定。
amkeytype	oid	PG_TYPE.oid	存储在索引里数据的类型，如果不是一个固定的类型则为0。
aminsert	regproc	PG_PROC.oid	“插入此行”函数。
ambeginscan	regproc	PG_PROC.oid	“准备索引扫描” 函数。
amgettuple	regproc	PG_PROC.oid	“下一个有效行”函数，如果没有则为0。
amgetbitmap	regproc	PG_PROC.oid	“抓取所有的有效行” 函数，如果没有则为0。
amrescan	regproc	PG_PROC.oid	“（重新）开始索引扫描”函数。
amendscan	regproc	PG_PROC.oid	“索引扫描后清理” 函数。
ammarkpos	regproc	PG_PROC.oid	“标记当前扫描位置”函数。
amrestrpos	regproc	PG_PROC.oid	“恢复已标记的扫描位置”函数。
ammerge	regproc	PG_PROC.oid	“归并多个索引对象”函数。
ambuild	regproc	PG_PROC.oid	“建立新索引”函数。
ambuildempty	regproc	PG_PROC.oid	“建立空索引”函数。
ambulkdelete	regproc	PG_PROC.oid	批量删除函数。
amvacuumcleanup	regproc	PG_PROC.oid	VACUUM后的清理函数。
amcanreturn	regproc	PG_PROC.oid	检查是否索引支持唯一索引扫描的函数，如果没有则为0。
amcostestimate	regproc	PG_PROC.oid	估计一个索引扫描开销的函数。
amoptions	regproc	PG_PROC.oid	用于分析和验证索引的reloptions函数。

18.2.11 PG_AMOP

PG_AMOP系统表存储和访问方法操作符族关联的信息。如果一个操作符是一个操作符族中的成员，则在这个表中会占据一行。一个族成员是一个search操作符或一个

 版权所有 © 华为云计算技术有限公司

ordering操作符。一个操作符可以在多个族中出现，但是不能在一个族中的多个搜索位置或多个排序位置中出现。

表 18-11 PG_AMOP 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amopfamily	oid	PG_OPFAMILY.oid	该项的操作符族。
amoplefttype	oid	PG_TYPE.oid	操作符的左输入类型。
amoprighttype	oid	PG_TYPE.oid	操作符的右输入类型。
amopstrategy	smallint	-	操作符策略数。
amoppurpose	"char"	-	操作符目的，s为搜索或o为排序。
amopopr	oid	PG_OPERATOR.oid	该操作符的OID。
amopmethod	oid	PG_AM.oid	使用此操作符族的索引访问方法。
amopsortfamily	oid	PG_OPFAMILY.oid	如果是一个排序操作符，则该项会按照B-Tree操作符族排序；如果是一个搜索操作符，则为0。

search操作符表明这个操作符族的一个索引可以被搜索，找到所有满足WHERE indexed_column operator constant的行。显然，这样的操作符必须返回布尔值，并且它的左输入类型必须匹配索引的字段数据类型。

ordering操作符表明这个操作符族的一个索引可以被扫描，返回以ORDER BY indexed_column operator constant顺序表示的行。这样的操作符可以返回任意可排序的数据类型，它的左输入类型也必须匹配索引的字段数据类型。ORDER BY的确切语义是由amopsortfamily字段指定的，该字段必须为操作符的返回类型引用一个B-Tree操作符族。

18.2.12 PG_AMPROC

PG_AMPROC系统表存储与访问方法操作符族相关联的支持过程的信息。每个属于某个操作符族的支持过程都占有一行。

表 18-12 PG_AMPROC 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amprocfamily	oid	PG_OPFAMILY.oid	该项的操作符族。

 版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
amproclefttype	oid	PG_TYPE.oid	相关操作符的左输入数据类型。
amprocrighttype	oid	PG_TYPE.oid	相关操作符的右输入数据类型。
amprocnum	smallint	-	支持过程编号。
amproc	regproc	PG_PROC.oid	过程的OID。

amproclefttype和amprocrighttype字段的习惯解释，标识一个特定支持过程所支持的操作符的左右输入类型。对于某些访问方式，匹配支持过程本身的输入数据类型，对其他的则不会匹配。有一个对索引的“缺省”支持过程的概念，amproclefttype和amprocrighttype都等于索引操作符类的opcintype。

18.2.13 PG_ATTRDEF

PG_ATTRDEF系统表存储字段的默认值。

表 18-13 PG_ATTRDEF 字段

名称	类型	描述
adrelid	oid	该字段所属的表。
adnum	smallint	字段编号。
adbin	pg_node_tree	字段缺省值的内部表现形式。
adsrc	text	人类可读的缺省值的内部表现形式。
adbin_on_update	pg_node_tree	字段on_update_expr值的内部表现形式。
adsrc_on_update	text	人类可读的on_update_expr值的内部表现形式。

18.2.14 PG_ATTRIBUTE

PG_ATTRIBUTE系统表存储关于表字段的信息。

表 18-14 PG_ATTRIBUTE 字段

名称	类型	描述
attrelid	oid	该字段所属的表。
attname	name	字段名。
atttypid	oid	字段类型。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
attstattarget	integer	控制ANALYZE为该字段设置的统计细节的级别。 ● 零值表示不收集统计信息。 ● 负数表示使用系统缺省的统计对象。 ● 正数值的确切信息是和数据类型相关的。 对于标量数据类型，ATTSTATTARGET既是要收集的“最常用数值”的目标数目，也是要创建的柱状图的目标数量。
attlen	smallint	是本字段类型pg_type.typlen的复制。
attnum	smallint	字段编号。
attndims	integer	如果该字段是数组，该值表示数组的维数，否则是0 。
attcacheoff	integer	在磁盘上总是-1 ，但是如果加载入内存中的行描述器中，它可能会被更新为缓冲在行中字段的偏移量。
atttypmod	integer	记录创建新表时支持的类型特定的数据（比如，varchar字段的最大长度）。它传递给类型相关的输入和长度转换函数当做第三个参数。其值对那些不需要ATTTYPMOD的类型通常为-1。
attbyval	boolean	pg_type.typbyval字段值的复制。
attstorage	"char"	pg_type.typstorage字段值的复制。
attalign	"char"	pg_type.typalign字段值的复制。
attnotnull	boolean	代表一个非空约束。可以改变这个字段来打开或者关闭该约束。
atthasdef	boolean	该字段是否存在缺省值，此时它对应pg_attrdef表里实际定义此值的记录。
attisdropped	boolean	该字段是否已经被删除，不再有效。如果被删除，该字段物理上仍然存在表中，但会被分析器忽略，因此不能再通过SQL访问。
attislocal	boolean	该字段是否局部定义在对象中。一个字段可以同时是局部定义和继承的。
attcmprmode	tinyint	对某一列指定压缩方式。压缩方式包括： ● ATT_CMPR_NOCOMPRESS ● ATT_CMPR_DELTA ● ATT_CMPR_DICTIONARY ● ATT_CMPR_PREFIX ● ATT_CMPR_NUMSTR

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
attinhcount	integer	该字段所拥有的直接父表的个数。如果一个字段的父表个数非零，则它就不能被删除或重命名。
attcollation	oid	对此列定义的校对列。
attacl	aclitem[]	列级访问权限控制。
attoptions	text[]	属性级可选项。
attfdwoptions	text[]	属性级外数据选项。
attinitdefval	bytea	存储了此列默认的值表达式。行存表的ADD COLUMN需要使用此字段。
attkvtype	tinyint	该字段的kv_type属性。取值如下： ● 0表示默认值，用于非时序表。 ● 1表示维度属性(TSTAG)，仅用于时序表。 ● 2表示指标属性(TSFIELD)，仅用于时序表。 ● 3时间属性(TSTIME)，仅用于时序表。

应用示例

查询指定表中包含的字段名和字段编号。t1和public分别替换为实际的表名和schema名称。

```
SELECT attname,attnum FROM pg_attribute WHERE attrelid=(SELECT pg_class.oid FROM pg_class JOIN pg_namespace ON relnamespace=pg_namespace.oid WHERE relname='t1' and nspname='public') and attnum>0;
    attname      | attnum
-----------------+--------
 product_id      |      1
 product_name    |      2
 product_quantity |      3
(3 rows)
```

18.2.15 PG_AUTHID

PG_AUTHID系统表存储有关数据库认证标识符（角色）的信息。角色把“用户”的概念包含在内。一个用户实际上就是一个rolcanlogin标志被设置的角色。任何角色（不管rolcanlogin设置与否）都能够把其他角色作为成员。

在一个集群中只有一份pg_authid，不是每个数据库有一份。需要有系统管理员权限才可以访问此系统表。

表 18-15 PG_AUTHID 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rolname	name	角色名称。
rolsuper	boolean	角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	角色是否自动继承其所属角色的权限。
rolcreaterole	boolean	角色是否可以创建更多角色。
rolcreatedb	boolean	角色是否可以创建数据库。
rolcatupdate	boolean	角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	角色是否可以登录，即该角色是否能够作为初始会话授权标识符。
rolreplication	boolean	角色是一个复制的角色（适配作用，没有实际的功能）。
rolauditadmin	boolean	审计用户。
rolsystemadmin	boolean	管理员用户。
rolconnlimit	integer	限制单个用户在单个CN上的最大并发连接数。-1表示没有限制。
rolpassword	text	口令(可能是加密的)，如果没有口令，则为NULL。
rolvalidbegin	timestamp with time zone	账户的有效开始时间，如果没有开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	账户的有效结束时间，如果没有结束时间，则为NULL。
rolrespool	name	用户所能够使用的resource pool。
roluseft	boolean	角色是否可以操作外表。
rolparentid	oid	用户所在组用户的OID。
roltabspace	Text	用户永久表存储空间限额。
rolkind	char	特殊用户种类，包括私有用户、逻辑集群管理员、普通用户。
rolnodegroup	oid	用户所关联的Node Group OID，该Node Group必须是逻辑集群。
roltempspace	Text	用户临时表存储空间限额。
rolspillspace	Text	用户算子落盘空间限额。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rolexcpdata	text	保留字段未使用。
rolauthinfo	text	用户采用LDAP或OneAccess认证时的额外信息。如果是其他认证模式，则为NULL。
rolpwdexpire	integer	用户口令过期时间，在口令未过期时，用户自己修改口令。过期后请管理员修改口令。-1表示没有过期时间限制。
rolpwdtime	timestamp with time zone	口令的创建时间。
roluuid	bigint	角色标识符。该字段仅9.1.0及以上集群版本支持。

18.2.16 PG_AUTH_HISTORY

PG_AUTH_HISTORY系统表记录了角色的认证历史。需要有系统管理员权限才可以访问此系统表。

表 18-16 PG_AUTH_HISTORY 字段

名称	类型	描述
roloid	oid	角色标识。
passwordtime	timestamp with time zone	创建和修改密码的时间。
rolpassword	text	角色密码，使用MD5、SHA256加密或者不加密。

18.2.17 PG_AUTH_MEMBERS

PG_AUTH_MEMBERS系统表存储显示角色之间的成员关系。

表 18-17 PG_AUTH_MEMBERS 字段

名称	类型	描述
roleid	oid	拥有成员的角色ID。
member	oid	属于ROLEID角色的成员角色ID。
grantor	oid	赋予此成员关系的角色ID。
admin_option	boolean	如果有权限可以把ROLEID角色的成员关系赋予其他角色，则为真。

版权所有 © 华为云计算技术有限公司

18.2.18 PG_BLOCKLISTS

PG_BLOCKLISTS系统表记录了查询过滤规则相关信息。该系统表仅9.1.0.100及以上集群版本支持。

表 18-18 PG_BLOCKLISTS 字段

名称	类型	描述
block_name	name	查询过滤规则名称。
role	oid	查询过滤规则绑定的用户OID。
client_addr	inet	查询过滤规则绑定的客户端IP地址。
application_name	name	查询过滤规则绑定的客户端名称。
unique_sql_id	int8	查询过滤规则匹配的unique_sql_id。
sql_hash	name	查询过滤规则匹配的sql_hash。
block_type	int4	查询过滤规则绑定的语句类型，分别对应SELECT/UPDATE/INSERT/DELETE/MERGE。
partition_num	int4	计划预估扫描节点扫描的最大分区数。
table_num	int4	计划预估扫描的最大表数。
estimate_row	int4	计划预估扫描节点扫描的最大行数。
query_band	text	主动标识的作业类型。
sql	text	查询过滤规则匹配的SQL语句。
created_time	timestamp with time zone	查询过滤规则创建或者修改的时间戳。
resource_pool	name	查询过滤规则拦截到语句所切换到的指定资源池名称。该字段仅9.1.0.200及以上集群版本支持。
max_active_num	integer	查询过滤规则拦截到语句的最大并发数。低于该数值时正常执行，大于等于该数值时会报错拦截。 该字段仅9.1.0.200及以上集群版本支持。
is_warning	boolean	控制查询过滤规则拦截到语句时的行为。 ● false表示拦截到语句时报错。默认值为false。 ● true表示拦截到语句时产生告警信息。 该字段仅9.1.0.200及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

18.2.19 PG_CAST

PG_CAST系统表存储数据类型之间的转化关系。

表 18-19 PG_CAST 字段

名称	类型	描述
castsource	oid	源数据类型的OID。
casttarget	oid	目标数据类型的OID。
castfunc	oid	转化函数的OID。0表示不需要转化函数。
castcontext	"char"	源数据类型和目标数据类型间的转化方式： ● e表示只能进行显式转化（使用CAST或::语法）。 ● i表示只能进行隐式转化。 ● a表示类型间同时支持隐式和显式转化。
castmethod	"char"	转化方法： ● f表示使用castfunc字段中指定的函数进行转化。 ● b表示类型间是二进制强制转化，不使用castfunc。

18.2.20 PG_CLASS

PG_CLASS系统表存储数据库对象信息及其之间的关系。

表 18-20 PG_CLASS 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
relname	name	表、索引、视图等对象的名称。
relnamespace	oid	包含该关系的命名空间的OID。
reltype	oid	与该表的行类型对应的数据类型（索引为零，因为索引没有pg_type记录）。
reloftype	oid	复合类型的OID，0表示其他类型。
relowner	oid	关系所有者。
relam	oid	如果行是索引，则就是所用的访问模式（B-tree，hash等）。
relfilenode	oid	该关系在磁盘上的文件的名称，如果没有则为0。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
reltablespace	oid	该关系存储所在的表空间。如果为0，则使用该数据库的缺省表空间。如果关系无磁盘文件，该字段无意义。
relpages	double precision	以页(大小为BLCKSZ)为单位的此表在磁盘上的大小，只是优化器使用的一个近似值。
reltuples	double precision	表中行的数目，只是优化器使用的一个估计值。
relallvisible	integer	被标识为全可见的表中的页数。此字段是优化器用来做SQL执行优化使用的。VACUUM、ANALYZE和一些DDL语句（例如，CREATE INDEX）会引起此字段更新。
reltoastrelid	oid	与此表关联的TOAST表的OID ，如果没有则为0。 TOAST表在一个从属表里“离线”存储大字段。
reltoastidxid	oid	对于TOAST表是它的索引的OID，如果不是TOAST表则为0。
reldeltarelid	oid	Delta表的OID。 Delta表附属于列存表。用于存储数据导入过程中的甩尾数据。
reldeltaidx	oid	Delta表的索引表OID。
relcudescrelid	oid	CU描述表的OID。 CU描述表（Desc表）附属于列存表。用于控制表目录中存储数据的可见性。
relcudescidx	oid	CU描述表的索引表OID。
relhasindex	boolean	如果对象是一个表且至少有（或者最近建有）一个索引，则为真。 由CREATE INDEX设置，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，会清理relhasindex字段，将relhasindex值设置为假。
relisshared	boolean	如果该表在整个集群中由所有数据库共享则为真。只有某些系统表（比如pg_database）是共享的。
relpersistence	"char"	• p表示永久表。 • u表示非日志表。 • t表示本地临时表。 • g表示全局临时表。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
relkind	"char"	• r表示普通表。 • i表示索引。 • S表示序列。 • v表示视图。 • c表示复合类型。 • t表示TOAST表。 • f表示外表。 • m表示物化视图。
relnatts	smallint	关系中用户字段数目（除了系统字段以外）。在pg_attribute里肯定有相同数目对应行。
relchecks	smallint	表上检查约束的数目，参见**PG_CONSTRAINT**。
relhasoids	boolean	如果为关系中每行都生成一个OID，则为真。
relhaspkey	boolean	如果该表有一个（或曾有）主键，则为真。
relhasrules	boolean	如果表有规则，则为真。是否有规则可参考系统表**PG_REWRITE**。
relhastriggers	boolean	如果表有（或曾有）触发器，则为真。参见**PG_TRIGGER**。
relhassubclass	boolean	如果表有（或曾有）任何继承的子表，则为真。
relcmprs	tinyint	表示是否启用表的压缩特性。需要特别注意，当且仅当批量插入才会触发压缩，普通的CRUD并不能够触发压缩。 • 0表示其他不支持压缩的表（主要是指系统表，不支持压缩属性的修改操作）。 • 1表示表数据的压缩特性为NOCOMPRESS或者无指定关键字。 • 2表示表数据的压缩特性为COMPRESS。
relhasclusterkey	boolean	是否有局部聚簇存储。
relrowmovement	boolean	针对分区表进行update操作时，是否允许行迁移。 • true：表示允许行迁移。 • false：表示不允许行迁移。
parttype	"char"	表或者索引是否具有分区表的性质。 • p表示带有分区表性质。 • n表示没有分区表特性。 • v表示该表为HDFS的Value分区表。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
relfrozenxid	xid32	该表中所有在此之间的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
relacl	aclitem[]	访问权限。 查询的回显结果为以下形式： rolename=xxxx/yyyy --赋予一个角色的权限 =xxxx/yyyy --赋予public的权限 xxxx表示赋予的权限，yyyy表示授予该权限的角色。权限的参数说明请参见表18-21。
reloptions	text[]	特定的访问方法选项，用"keyword=value"字符串形式表示。
relfrozenxid64	xid	该表中所有在此之前的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。

数据库对象权限使用 ACL（Access Control List）进行控制。DWS使用aclitem来表示一个具体的数据库对象上的权限。

- 对于database和schema，aclitem存储在pg_database.datacl和pg_namespace.nspacl中。
- 对于table、view等其他数据库对象，aclitem存储在pg_class.relacl中。
- 对于列级别的权限，aclitem存储在pg_attribute.attacl中。

例如，test_user=a*r/test_user1，表示test_user这个用户对当前数据库对象拥有INSERT和SELECT权限，其中INSERT权限带有with grant option，表示可以授权其他用户该权限。/test_user1表示这条aclitem权限是test_user1授予的。

下表列出了DWS中所有的数据库对象权限及其缩写。

表 18-21 ACL 权限的参数说明

权限	缩写	适用的对象类型
SELECT（读）	r	LARGE OBJECT，SEQUENCE，TABLE (and table-like objects)，table column
UPDATE（写）	w	LARGE OBJECT，SEQUENCE，TABLE，table column
INSERT（插入）	a	TABLE，table column

 版权所有 © 华为云计算技术有限公司

权限	缩写	适用的对象类型
DELETE	d	TABLE
TRUNCATE	D	TABLE
REFERENCES	x	TABLE，table column
TRIGGER	t	TABLE
EXECUTE	X	FUNCTION，PROCEDURE
USAGE	U	DOMAIN，FOREIGN DATA WRAPPER，FOREIGN SERVER，LANGUAGE，SCHEMA，SEQUENCE，TYPE
CREATE	C	DATABASE，SCHEMA，TABLESPACE
CONNECT	c	DATABASE
TEMPORARY	T	DATABASE
ANALYZE|ANALYSE	A	TABLE
ALTER	L	TABLE
DROP	P	TABLE
VACUUM	v	TABLE
ALL PRIVILEGES	arwdDxtA，vLP	TABLE
给前面权限的授权选项	*	-

应用示例

查看某张表的oid及relfilenode。

```
SELECT oid,relname,relfilenode FROM pg_class WHERE relname = 'table_name';
```

统计行存表数量。

```
SELECT 'row count:'||count(1) as point FROM pg_class WHERE relkind = 'r' and oid > 16384 and reloptions::text not like '%column%' and reloptions::text not like '%internal_mask%';
```

统计列存表数量。

```
SELECT 'column count:'||count(1) as point FROM pg_class WHERE relkind = 'r' and oid > 16384 and reloptions::text like '%column%';
```

查询数据库中所有表的注释。

```
SELECT relname as tablename,obj_description(relfilenode,'pg_class') as comment FROM pg_class;
```

查看数据库中指定表的权限信息。

```
SELECT relacl FROM pg_class WHERE relname = 'table_name';
```

 版权所有 © 华为云计算技术有限公司

18.2.21 PG_COLLATION

PG_COLLATION系统表描述可用的排序规则，本质上从一个SQL名字映射到操作系统本地类别。

表 18-22 PG_COLLATION 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
collname	name	-	排序规则名（每个命名空间和编码唯一）。
collnamespace	oid	PG_NAMESPACE.oid	包含该排序规则的命名空间的OID。
collowner	oid	PG_AUTHID.oid	排序规则的所有者。
collencoding	integer	-	排序规则可用的编码，如果适用于任意编码为-1。
collcollate	name	-	排序规则对象的LC_COLLATE。
collctype	name	-	排序规则对象的LC_CTYPE。

18.2.22 PG_CONSTRAINT

PG_CONSTRAINT系统表存储表上的检查约束、主键、唯一约束和外键约束。

表 18-23 PG_CONSTRAINT 字段

名称	类型	描述
conname	name	约束名称（不一定是唯一的）。
connamespace	oid	包含约束的命名空间的OID。
contype	"char"	• c = 检查约束 • f = 外键约束 • p = 主键约束 • u = 唯一约束 • t = 触发器约束
condeferrable	boolean	该约束是否可以推迟。
condeferred	boolean	缺省时该约束是否可以推迟。
convalidated	boolean	约束是否有效。目前，只有外键和CHECK约束可将其设置为FALSE。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
conrelid	oid	该约束所在的表；如果不是表约束则为0。
contypid	oid	该约束所在的域；如果不是一个域约束则为0。
conindid	oid	与约束关联的索引ID。
confrelid	oid	如果是外键，则为参考的表；否则为0。
confupdtype	"char"	外键更新动作代码。 ● a = 没动作 ● r = 限制 ● c = 级联 ● n =设置为null ● d =设置为缺省
confdeltype	"char"	外键删除动作代码。 ● a = 没动作 ● r = 限制 ● c = 级联 ● n =设置为null ● d =设置为缺省
confmatchtype	"char"	外键匹配类型。 ● f = 全部 ● p = 部分 ● u = 简单（未指定）
conislocal	boolean	是否是为关系创建的本地约束。
coninhcount	integer	约束直接继承父表的数目。继承父表数非零时，不能删除或重命名该约束。
connoinherit	boolean	是否可以被继承。
consoft	boolean	是否为信息约束(Informational Constraint)。
conopt	boolean	是否使用信息约束优化执行计划。
conkey	smallint[]	如果是表约束，则是约束控制的字段列表。
confkey	smallint[]	如果是一个外键，则是参考的字段的列表。
conpfeqop	oid[]	如果是一个外键，是做PK=FK比较的相等操作符ID的列表。
conppeqop	oid[]	如果是一个外键，是做PK=PK比较的相等操作符ID的列表。
conffeqop	oid[]	如果是一个外键，是做FK=FK比较的相等操作符ID的列表。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
conexclop	oid[]	如果是一个排他约束，是列的排他操作符ID列表。
conbin	pg_node_tree	如果是检查约束，则是其表达式的内部形式。
consrc	text	如果是检查约束，则是表达式的人类可读形式。

须知

- 当被引用的对象改变时，consrc不能被更新。例如，它不会跟踪字段的重命名。建议还是使用pg_get_constraintdef()来抽取一个检查约束的定义，而不是依赖这个字段。
- pg_class.relchecks需要和每个关系在此目录中的检查约束数量保持一致。

应用示例

查询指定表是否有主键。

```
CREATE TABLE t1
(
    C_CUSTKEY     BIGINT       ,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT          ,
    C_PHONE       CHAR(15)     ,
    C_ACCTBAL     DECIMAL(15,2),
    CONSTRAINT C_CUSTKEY_KEY PRIMARY KEY(C_CUSTKEY,C_NAME)
)
DISTRIBUTE BY HASH(C_CUSTKEY,C_NAME);

SELECT conname FROM pg_constraint WHERE conrelid = 't1'::regclass AND contype = 'p';
    conname
----------------
 c_custkey_key
(1 row)
```

18.2.23 PG_CONVERSION

PG_CONVERSION系统表描述编码转换信息。

表 18-24 PG_CONVERSION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
conname	name	-	转换名（在一个命名空间里唯一）。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
connamespace	oid	PG_NAMESPACE.oid	包含此转换的命名空间的OID。
conowner	oid	PG_AUTHID.oid	编码转换的属主。
conforencoding	integer	-	源编码ID。
contoencoding	integer	-	目的编码ID。
conproc	regproc	PG_PROC.oid	转换过程。
condefault	boolean	-	如果为缺省转换则为真。

18.2.24 PG_DATABASE

PG_DATABASE系统表存储关于可用数据库的信息。

表 18-25 PG_DATABASE 字段

名称	类型	描述
datname	name	数据库名称。
datdba	oid	数据库所有者，通常为其创建者。
encoding	integer	数据库的字符编码方式。 pg_encoding_to_char()可以将此编号转换为编码名称。
datcollate	name	数据库使用的排序顺序。
datctype	name	数据库使用的字符分类。
datistemplate	boolean	是否允许作为模板数据库。
datallowconn	boolean	如果为假，则没有用户可以连接到这个数据库。这个字段用于保护template0数据库不被更改。
datconnlimit	integer	该数据库上允许的最大并发连接数，-1表示无限制。
datlastsysoid	oid	数据库里最后一个系统OID 。
datfrozenxid	xid32	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。 为保持前向兼容，保留此字段，新增datfrozenxid64用于记录此信息。
dattablespace	oid	数据库的缺省表空间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
datcompatibility	name	数据库兼容模式。 ● ORA，表示兼容Oracle数据库。 ● TD，表示兼容Teradata数据库。 ● MySQL，表示兼容MySQL数据库。
datacl	aclitem[]	数据库访问权限。
datfrozenxid64	xid	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。

18.2.25 PG_DB_ROLE_SETTING

PG_DB_ROLE_SETTING系统表存储数据库运行时每个角色与数据绑定的配置项的默认值 。

表 18-26 PG_DB_ROLE_SETTING 字段

名称	类型	描述
setdatabase	oid	配置项所对应的数据库，如果未指定数据库，则为0。
setrole	oid	配置项所对应的角色，如果未指定角色，则为0。
setconfig	text[]	运行时配置项的默认值。

18.2.26 PG_DEFAULT_ACL

PG_DEFAULT_ACL系统表存储为新建对象设置的初始权限。

表 18-27 PG_DEFAULT_ACL 字段

名称	类型	描述
defaclrole	oid	与此权限相关的角色ID。
defaclnamespace	oid	与此权限相关的命名空间，如果没有，则为0。
defaclobjtype	"char"	此权限的对象类型。 ● r表示表或视图。 ● S表示序列。 ● f表示函数。 ● T表示类型。
defaclacl	aclitem[]	创建该类型时所拥有的访问权限。

版权所有 © 华为云计算技术有限公司

应用示例

查看新建用户role1的初始权限：

```
select * from PG_DEFAULT_ACL;
 defaclrole | defaclnamespace | defaclobjtype |    defaclacl
------------+-----------------+---------------+-----------------
      16820 |           16822 | r             | {role1=r/user1}
```

也可使用如下语句进行转换后更直观的查看：

```
SELECT pg_catalog.pg_get_userbyid(d.defaclrole) AS "Granter",  n.nspname AS "Schema",  CASE d.defaclobjtype WHEN 'r' THEN 'table' WHEN 'S' THEN 'sequence' WHEN 'f' THEN 'function' WHEN 'T' THEN 'type' END AS "Type",  pg_catalog.array_to_string(d.defaclacl, E', ') AS "Access privileges" FROM pg_catalog.pg_default_acl d LEFT JOIN pg_catalog.pg_namespace n ON n.oid = d.defaclnamespace ORDER BY 1, 2, 3;
```

输出结果如下，表示通过用户user1授予用户role1对模式“user1”有读的权限。

```
 Granter | Schema | Type  | Access privileges
---------+--------+-------+-------------------
 user1   | user1  | table | role1=r/user1
(1 row)
```

18.2.27 PG_DEPEND

PG_DEPEND系统表记录数据库对象之间的依赖关系。这些信息允许DROP命令找出哪些其它对象必须由DROP CASCADE删除，或者是在DROP RESTRICT的情况下避免删除。

另请参考**PG_SHDEPEND**，对于记录那些在数据库集群之间共享的对象之间的依赖性关系提供了相似的功能。

表 18-28 PG_DEPEND 字段

名称	类型	引用	描述
classid	oid	**PG_CLASS**.oid	依赖对象所在系统表的OID。
objid	oid	任意OID属性	指定依赖对象的OID。
objsubid	integer	-	对于表字段，是该属性的字段数（objid和classid引用表本身）。对于所有其它对象类型，此字段是0。
refclassid	oid	**PG_CLASS**.oid	被引用对象所在的系统表的OID。
refobjid	oid	任意OID属性	指定的被引用对象的OID。
refobjsubid	integer	-	对于表字段，是该字段的字段号（refobjid和refclassid引用表本身）。对于所有其它对象类型，此字段是零。
deptype	"char"	-	定义此依赖关系特定语义的代码。

在所有情况下，一个PG_DEPEND记录表示被引用对象不能在没有删除依赖对象的情况下被删除。但是其中也有几种由deptype定义的情况：

- DEPENDENCY_NORMAL (n)：独立创建的对象之间的一般关系。依赖对象可以在不影响被引用对象的情况下删除。被引用对象只能通过指定CASCADE被删除，

 版权所有 © 华为云计算技术有限公司

这种情况下依赖对象也会被删除。例如：一个表字段对其数据类型有一般依赖关系。

- DEPENDENCY_AUTO (a)：依赖对象可以和被引用对象分别删除，且在被引用对象被删除时应自动被删除（不管是RESTRICT或CASCADE模式）。例如：一个表上的命名约束是该表上的自动依赖关系，因此如果删除了表，它也会被删除。
- DEPENDENCY_INTERNAL (i)：依赖对象作为被引用对象过程的一部分创建，且是其内部实现的一部分。 DROP依赖对象是不会直接允许的（会给户发出一个针对被引用对象的DROP）。不管是否指定CASCADE，一个被引用对象的DROP将被传播来删除其依赖对象。例如：一个用于强制外键约束的触发器将被设置为内部依赖于其约束的**PG_CONSTRAINT**项。
- DEPENDENCY_EXTENSION (e)：依赖对象作为被依赖对象extension的一个成员（请参见**PG_EXTENSION**）。依赖对象可以通过在被依赖对象上DROP EXTENSION删除。在功能上，这种依赖类型和内部依赖的作用相同，其存在只是为了清晰和简化gs_dump。
- DEPENDENCY_PIN (p)：没有依赖对象。这种类型的记录标志着系统本身依赖于被引用对象，因此这个对象绝不能被删除。这种类型的记录只有在initdb的时候创建。有依赖对象的字段都为0。

应用示例

查询名为serial1的数据库对象sequence和哪个表有依赖关系。

1. 先通过系统表PG_CLASS查询序列名为serial1的oid。

```
SELECT oid FROM pg_class WHERE relname ='serial1';
 oid
-------
 17815
(1 row)
```

2. 使用系统表PG_DEPEND根据所查询的序列serial1的oid获取依赖该序列的对象。

```
SELECT * FROM pg_depend WHERE objid ='17815';
 classid | objid | objsubid | refclassid | refobjid | refobjsubid | deptype
---------+-------+----------+------------+----------+-------------+---------
    1259 | 17815 |        0 |       2615 |     2200 |           0 | n
    1259 | 17815 |        0 |       1259 |    17812 |           1 | a
(2 rows)
```

3. 根据字段refobjid获取依赖该序列serial1的表的OID，并查询到表名。其结果表示，序列serial1依赖于表customer_address。

```
SELECT relname FROM pg_class where oid='17812';
     relname
------------------
 customer_address
(1 row)
```

18.2.28 PG_DESCRIPTION

PG_DESCRIPTION系统表可以给每个数据库对象存储一个可选的描述（注释）。许多内置的系统对象的描述提供了PG_DESCRIPTION的初始内容。

这个表的功能类似**PG_SHDESCRIPTION**，用于记录整个集群范围内共享对象的注释。

表 18-29 PG_DESCRIPTION 字段

名称	类型	引用	描述
objoid	oid	任意OID属性	描述所属对象的OID。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
classoid	oid	PG_CLASS.oid	对象显示的系统表的OID。
objsubid	integer	-	对于一个表字段的注释，为字段号（objoid和classoid指向表自身）。对于其它对象类型，为0。
description	text	-	对该对象描述的任意文本。

18.2.29 PG_ENUM

PG_ENUM系统表包含显示每个枚举类型值和标签的记录。给定枚举类型的内部表示实际上是PG_ENUM里面相关行的OID。

表 18-30 PG_ENUM 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
enumtypid	oid	PG_TYPE.oid	包含此枚举值的pg_type项的OID。
enumsortorder	real	-	此枚举值在其枚举类型中的排序位置。
enumlabel	name	-	此枚举值的文本标签。

PG_ENUM行的OID值遵循一种特殊规则：OID的数值被保证按照其枚举类型一样的排序顺序排序。即如果两个偶数OID属于同一枚举类型，那么较小的OID必然具有较小enumsortorder值。奇数OID不需要遵循排序顺序。这种规则使得枚举比较例程在很多常见情况下可以避免系统目录查找。创建和修改枚举类型的例程尝试尽可能地为枚举值分配偶数OID。

当一个枚举类型被创建后，其成员会被分配排序顺序位置1到n。但是后面增加的成员可能会分配负值或者分数值的enumsortorder。对于这些值的唯一要求是它们必须被正确的排序且在每个枚举类型中保持唯一。

18.2.30 PG_EXCEPT_RULE

PG_EXCEPT_RULE系统表存储关于异常规则的信息。一个异常规则集合由多个名称相同的异常规则组成。

表 18-31 PG_EXCEPT_RULE

名称	类型	描述
name	name	异常规则集合的名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rule	name	该异常规则集中某一个具体规则类型或者触发当前异常规则集时采取的操作。（如blocktime/elapsedtime/spillsize等类型或者触发异常规则后的操作）
value	name	该异常规则对应的规则阈值。如果是触发异常规则后的操作，那么该字段为abort。

18.2.31 PG_EXTERNAL_NAMESPACE

存储EXTERNAL SCHEMA相关的信息。该系统表仅8.3.0及以上版本支持。

表 18-32 PG_EXTERNAL_NAMESPACE 字段

名字	类型	描述
nspid	Oid	External Schema Oid。
srvname	text	外部服务器名称。
source	text	元数据服务类型。
address	text	元数据服务地址。
database	text	元数据服务器数据库。
confpath	text	元数据服务器配置文件路径。
ensoptions	text[]	保留字段，暂时为空。
catalog	text	元数据服务器catalog。

示例

查询创建的EXTERNAL SCHEMA ex1：

```
SELECT * FROM pg_external_namespace WHERE nspid = (SELECT oid FROM pg_namespace WHERE nspname = 'ex1');
```

18.2.32 PG_EXTENSION

PG_EXTENSION系统表存储关于所安装扩展的信息。DWS默认有三十四个扩展，即aio_scheduler、btree_gin、cudesckv、dimsearch、dist_fdw、functional_clog、functional_extension、functional_file、functional_hudi、functional_job、functional_largeobject、functional_memory、functional_other、functional_signal、functional_vacuum、gc_fdw、hdfs_fdw、hstore、log_fdw、operational_backup、operational_cgroup、operational_cudesc、operational_other、operational_replication、operational_restoration、operational_stats、operational_xlog、packages、pgcrypto、pldbgapi、plpgsql、roach_api、tsdb和uuid-ossp。

版权所有 © 华为云计算技术有限公司

表 18-33 PG_EXTENSION

名称	类型	描述
extname	name	扩展名。
extowner	oid	扩展的所有者。
extnamespace	oid	扩展导出对象的命名空间。
extrelocatable	boolean	如果扩展能够重定位到其他schema，则为true。
extversion	text	扩展的版本号。
extconfig	oid[]	扩展的配置信息。
extcondition	text[]	扩展配置信息的过滤条件。

18.2.33 PG_EXTENSION_DATA_SOURCE

PG_EXTENSION_DATA_SOURCE系统表存储外部数据源对象的信息。一个外部数据源对象（Data Source）包含了外部数据库的一些口令编码等信息，主要配合Extension Connector使用。

表 18-34 PG_EXTENSION_DATA_SOURCE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
srcname	name	-	外部数据源对象的名称。
srcowner	oid	PG_AUTHID.oid	外部数据源对象的所有者。
srctype	text	-	外部数据源对象的类型，缺省为空。
srcversion	text	-	外部数据源对象的版本，缺省为空。
srcacl	aclitem[]	-	访问权限。
srcoptions	text[]	-	外部数据源对象的指定选项，使用“keyword=value”格式的字符串。

18.2.34 PG_FINE_DR_INFO

PG_FINE_DR_INFO系统表用于记录细粒度容灾备表的回放状态。该系统表仅8.2.0.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-35 PG_FINE_DR_INFO 字段

名字	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
relid	oid	细粒度容灾备表的OID。
lastcsn	xid	上一次成功回放时的csn。
lastxmin	xid	上一次成功回放时的xmin。
lastxmax	xid	上一次成功回放时的xmax。
laststarttime	timestamp with time zone	上一次成功回放的开始时间。
lastendtime	timestamp with time zone	上一次成功回放的结束时间。

应用示例

在容灾集群上查看备表回放状态：

```
SELECT * FROM pg_fine_dr_info;
 relid | lastcsn | lastxmin | lastxmax |         laststarttime         |          lastendtime
-------+---------+----------+----------+-------------------------------+-------------------------------
 21132 | 1251610 |  1251609 |  1251611 | 2023-01-04 20:51:58.375136+08 | 2023-01-04 20:51:58.393986+08
(1 row)
```

18.2.35 PG_FOREIGN_DATA_WRAPPER

PG_FOREIGN_DATA_WRAPPER系统表存储外部数据封装器定义。一个外部数据封装器是在外部服务器上驻留外部数据的机制，是可以访问的。

表 18-36 PG_FOREIGN_DATA_WRAPPER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
fdwname	name	-	外部数据封装器名。
fdwowner	oid	PG_AUTHID.oid	外部数据封装器的所有者。
fdwhandler	oid	PG_PROC.oid	引用一个负责为外部数据封装器提供扩展例程的处理函数。如果没有提供处理函数则为0。
fdwvalidator	oid	PG_PROC.oid	引用一个验证器函数，这个验证器函数负责验证给予外部数据封装器的选项、外部服务器选项和使用外部数据封装器的用户映射的有效性。如果没有提供验证器函数则为0。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
fdwacl	aclitem[]	-	访问权限。
fdwoptions	text[]	-	外部数据封装器指定选项，使用“keyword=value”格式的字符串。

18.2.36 PG_FOREIGN_SERVER

PG_FOREIGN_SERVER系统表存储外部服务器定义。一个外部服务器描述了一个外部数据源，例如一个远程服务器。外部服务器通过外部数据封装器访问。

表 18-37 PG_FOREIGN_SERVER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
srvname	name	-	外部服务器名。
srvowner	oid	PG_AUTHID.oid	外部服务器的所有者。
srvfdw	oid	PG_FOREIGN_DATA_WRAPPER.oid	此外部服务器的外部数据封装器的OID。
srvtype	text	-	服务器的类型（可选）。
srvversion	text	-	服务器的版本（可选）。
srvacl	aclitem[]	-	访问权限。
srvoptions	text[]	-	外部服务器指定选项，使用“keyword=value”格式的字符串。

18.2.37 PG_FOREIGN_TABLE

PG_FOREIGN_TABLE系统表存储外部表的辅助信息。

表 18-38 PG_FOREIGN_TABLE 字段

名称	类型	描述
ftrelid	oid	外部表的OID。
ftserver	oid	外部表的所在服务器的OID。
ftwriteonly	boolean	外部表是否可写。
ftoptions	text[]	外部表的可选项。

版权所有 © 华为云计算技术有限公司

18.2.38 PG_INDEX

PG_INDEX系统表存储索引的一部分信息，其他的信息大多数在PG_CLASS中。

表 18-39 PG_INDEX 字段

名称	类型	描述
indexrelid	oid	此索引的pg_class项的OID。
indrelid	oid	使用该索引的表在pg_class项的OID。
indnatts	smallint	索引中的字段数目。
indisunique	boolean	如果为真，为唯一索引。
indisprimary	boolean	如果为真，该索引为该表的主键。该字段为真时，indisunique也总是为真。
indisexclusion	boolean	如果为真，该索引支持排他约束。
indimmediate	boolean	如果为真，在插入数据时会立即执行唯一性检查。
indisclustered	boolean	如果为真，则该表最后以此索引进行了聚簇。
indisusable	boolean	如果为真，此索引对INSERT/SELECT可用。
indisvalid	boolean	如果为真，则此索引可以用于查询。如果为假，则该索引可能不完整，仍然必须在INSERT/UPDATE操作时进行更新，但不能安全的被用于查询。如果是唯一索引，则唯一属性也不为真。
indcheckxmin	boolean	如果为真，查询不能使用此索引，直到pg_index此行的xmin低于其快照的TransactionXmin，因为该表可能包含它们可见的不兼容行断开的HOT链。
indisready	boolean	如果为真，表示此索引对插入数据可用。如果为假，在插入或修改数据时忽略此索引。
indkey	int2vector	这是一个包含indnatts值的数组，这些数组值表示此索引所建立的表字段。比如一个值为1 3的意思是第一个字段和第三个字段组成这个索引键。数组中的0表示对应的索引属性是一个表字段上的表达式，而不是一个简单的字段引用。
indcollation	oidvector	索引用到的各列的ID。
indclass	oidvector	对于索引键中的每个字段，该字段都包含要使用的操作符类的OID，详见PG_OPCLASS。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
indoption	int2vector	存储列前标识位，该标识位是由索引的访问方法定义。
indexprs	pg_node_tree	表达式树（以nodeToString()形式表现）用于那些非简单字段引用的索引属性。它是一个列表，个数与INDKEY中的零值个数相同。如果所有索引属性都是简单的引用，则为空。
indpred	pg_node_tree	部分索引谓词的表达式树（以nodeToString()的形式表现）。如果不是部分索引，则为空。
indnullstreatment	tinyint	表示唯一索引中NULL值的处理方式，该字段只有当indisunique为真时才起作用。 取值范围： • 0 表示NULLS DISTINCT，NULL互不相等，NULL值可重复插入。 • 1 表示NULLS NOT DISTINCT，NULL严格相等，NULL值不可重复插入。 • 2 表示NULLS IGNORE，在等值比较时忽略NULL值列。若索引列全为NULL，则NULL值可重复插入；若部分索引列为NULL，只有非NULL列不相等，才可插入。 默认值：0 **说明** • 若当前集群为低版本升级到8.2.0.100版本，对于之前已存在索引，该字段值均为NULL；对于新创建的索引，则根据[NULLS [NOT] DISTINCT | NULLS IGNORE]字段的指定情况确定该字段值，默认值为0。 • 若当前集群为新装的8.2.0.100版本，对于新创建的索引，则根据[NULLS [NOT] DISTINCT | NULLS IGNORE]字段的指定情况确定该字段值，默认值为0。

18.2.39 PG_INHERITS

PG_INHERITS系统表记录关于表继承层次的信息。数据库里每个直接的子系表都有一条记录。间接的继承可以通过追溯记录链来判断。

表 18-40 PG_INHERITS 字段

名字	类型	引用	描述
inhrelid	oid	PG_CLASS.oid	子表的OID。
inhparent	oid	PG_CLASS.oid	父表的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
inhseqno	integer	-	如果一个子表存在多个直系父表（多重继承），这个数字表示此继承字段的排列顺序。计数从1开始。

18.2.40 PG_JOB_INFO

PG_JOB_INFO系统表记录定时任务的执行结果信息。该系统表的schema是dbms_om。

表 18-41 dbms_om.pg_job_info 字段

名称	类型	描述
job_id	integer	任务ID。
job_db	oid	任务所在数据库OID。
start_time	timestamp with zone	任务执行开始时间。
status	character(8)	任务执行状态。
end_time	timestamp with zone	任务执行结束时间。
err_msg	text	任务执行出错信息。

18.2.41 PG_JOBS

PG_JOBS系统表存储用户创建的定时任务的任务详细信息，定时任务线程定时轮询pg_jobs系统表中的时间，当任务到期会触发任务的执行。该系统表属于Shared Relation，所有创建的job记录对所有数据库可见。

表 18-42 PG_JOBS 字段

名字	类型	描述
job_id	integer	作业ID，主键，是唯一的（有唯一索引）。
what	text	作业内容。
log_user	oid	创建者的UserID。
priv_user	oid	作业执行者的UserID。
job_db	oid	标识作业执行的数据库OID。
job_nsp	oid	标识作业运行时所在的命名空间OID。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
job_node	oid	标识当前作业是在哪个CN上创建和执行。
is_broken	boolean	标识当前作业是否为失效状态。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。

18.2.42 PG_LANGUAGE

PG_LANGUAGE系统表记录了可用于编写函数或存储过程的语言。

表 18-43 PG_LANGUAGE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
lanname	name	-	语言的名称。
lanowner	oid	PG_AUTHID.oid	语言的所有者。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
lanispl	boolean	-	内部语言为假（比如SQL），用户定义语言为真。目前，gs_dump仍然使用该字段判断哪些语言需要转储，但可能在将来会被其它机制所取代。
lanpltrusted	boolean	-	如果是可信语言则为真，即系统相信它不会被授予任何正常SQL执行环境之外的权限。只有初始用户可以在用非可信的语言创建函数。
lanplcallfoid	oid	PG_PROC.oid	对于非内部语言，这是指该语言处理器的引用，语言处理器是一个特殊函数，负责执行以某种语言写的所有函数。
laninline	oid	PG_PROC.oid	此字段引用一个负责执行“inline”匿名代码块的函数（DO块）。如果不支持内联块则为0。
lanvalidator	oid	PG_PROC.oid	此字段引用一个语言校验器函数，它负责检查新创建函数的语法和有效性。如果没有提供校验器，则为0。
lanacl	aclitem[]	-	访问权限。

18.2.43 PG_LARGEOBJECT

PG_LARGEOBJECT系统表保存那些标记着“大对象”的数据。一个大对象是使用其创建时分配的OID标识的。每个大对象都分解成足够小的小段或者“页面”以便以行的形式存储在PG_LARGEOBJECT里。每页的数据定义为LOBLKSIZE。

需要有系统管理员权限才可以访问此系统表。

表 18-44 PG_LARGEOBJECT 字段

名字	类型	引用	描述
loid	oid	PG_LARGEOBJECT_METADATA.oid	包含本页的大对象的标识符。
pageno	integer	-	本页在其大对象数据中的页码，从0开始计算。
data	bytea	-	实际存储在大对象中的数据。这些数据不会超过LOBLKSIZE字节，且可能更小。

PG_LARGEOBJECT的每一行保存一个大对象的一个页面，从该对象内部的字节偏移（pageno * LOBLKSIZE）开始。这种实现允许稀疏存储：页面可能丢失，并且可以比LOBLKSIZE字节少（即使它们不是对象的最后一页）。大对象中丢失的区域会被读为0。

 版权所有 © 华为云计算技术有限公司

18.2.44 PG_LARGEOBJECT_METADATA

PG_LARGEOBJECT_METADATA系统表存储与大数据相关的元数据。实际的大对象数据存储在PG_LARGEOBJECT里。

表 18-45 PG_LARGEOBJECT_METADATA 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
lomowner	oid	PG_AUTHID.oid	大对象的所有者。
lomacl	aclitem[]	-	访问权限。

18.2.45 PG_MATVIEW

PG_MATVIEW系统表提供获取当前节点的物化视图信息。

表 18-46 PG_MATVIEW 字段

字段名称	字段类型	描述
mvid	oid	物化视图OID。
build_mode	char	物化视图的build模式。 ● 'd'：代表deferred，表示创建物化视图时需要等到第一次refresh时才会包含数据。 ● 'i'：代表immediate，表示创建物化视图时即包含最新数据。
refresh_method	char	物化视图的刷新方法。 'c' ： 代表complete，表示完全刷新。
refresh_mode	char	物化视图的刷新模式。 'd'：代表demand，表示按需刷新。
rewrite_enable	boolean	是否支持物化视图的查询重写。
active	boolean	物化视图是否需要刷新。
relnum	Int	物化视图基表个数。
start_time	timestamptz	物化视图第一次定时刷新的时间，为空则第一次刷新时间是当前时间+interval。
interval	interval	物化视图定时刷新时间的间隔。
refresh_time	timestamptz	物化视图的最后一次刷新时间。

版权所有 © 华为云计算技术有限公司

字段名称	字段类型	描述
refresh_finish_time	timestamptz	物化视图的最后一次刷新结束时间。

18.2.46 PG_MATVIEW_DEPEND

PG_MATVIEW_DEPEND系统表用于查询物化视图包含的基表信息和基表对应的物化视图信息。该视图仅8.2.1.100及以上集群版本支持。

表 18-47 PG_MATVIEW_DEPEND 字段

字段名称	字段类型	描述
matviewid	oid	物化视图OID。
relid	oid	物化视图基表OID。
partids	oidvector	指定分区时候的基表分区OID。
contain_entire_rel	boolean	是否包含整个基表。
matpartid	oid	映射的物化视图的分区OID。
relpartid	oid	映射的基表的分区OID。

18.2.47 PG_MATVIEW_PARTITION

PG_MATVIEW_PARTITION系统表存储分区物化视图每个分区的数据信息。该系统表仅9.1.0.200及以上集群版本支持。

表 18-48 PG_MATVIEW_PARTITION 字段

名称	类型	描述
matpartid	oid	物化视图分区的OID。
active	boolean	分区数据是否为最新数据。 t，表示分区物化视图的数据为最新状态。 f，表示分区物化视图的数据不是最新状态。
refresh_start_time	timestamp with time zone	最后一次刷新的开始时间。
refresh_finish_time	timestamp with time zone	最后一次刷新的结束时间。

版权所有 © 华为云计算技术有限公司

18.2.48 PG_NAMESPACE

PG_NAMESPACE系统表存储命名空间，即存储schema相关的信息。

表 18-49 PG_NAMESPACE 字段

名称	类型	描述
nspname	name	命名空间的名称。
nspowner	oid	命名空间的所有者。
nsptimeline	bigint	在DN上创建此命名空间时的时间线。此字段为内部使用，仅在DN上有效。
nspacl	aclitem[]	访问权限。具体请参见《数据仓库服务SQL语法参考》中“GRANT”和“REVOKE”章节。
permspace	bigint	schema永久表空间限额。
usedspace	bigint	schema已用永久表空间大小。
nsptype	char	区分external schema和普通schema。该参数仅8.3.0及以上版本支持，适配Lakeformation特性。 **说明** 新增nsptype字段，用于区分external schema和普通schema。 ● external schema对应的值为“e” ● 普通schema对应的值为“i”。

18.2.49 PG_OBJECT

PG_OBJECT系统表存储限定类型对象(object_type中存在的类型)的创建用户、创建时间、最后修改时间和最后analyze时间。

表 18-50 PG_OBJECT 字段

名称	类型	描述
object_oid	oid	对象标识符。
object_type	"char"	对象类型： ● r表示表，包括普通表和临时表 ● i表示索引 ● s表示序列 ● v表示视图 ● p表示存储过程和函数 ● f表示外表
creator	oid	创建用户的标识符。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
ctime	timestamp with time zone	对象创建时间。
mtime	timestamp with time zone	对象最后修改时间，默认记录修改行为包括ALTER操作、COMMENT、GRANT/REVOKE和TRUNCATE操作。 **object_mtime_record_mode**参数可以细粒度控制ALTER、COMMENT、GRANT/REVOKE和TRUNCATE操作是否被记录。
last_analyze_time	timestamp with time zone	对象进行最后一次analyze的时间。

须知

- 仅针对用户正常操作行为进行记录，无法记录对象升级以前和initdb过程中的行为。
- ctime和mtime的时间记录为本次操作的事务起始时间。
- 由扩容引起的对象修改时间也会被记录。

18.2.50 PG_OBSSCANINFO

PG_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 18-51 PG_OBSSCANINFO 字段

名字	类型	引用	描述
query_id	bigint	-	查询标识。
user_id	text	-	执行该查询的数据库用户。
table_name	text	-	OBS外表的表名。
file_type	text	-	底层数据保存的文件格式。
time_stamp	timestamp with time zone	-	扫描操作开始的时间。
actual_time	double	-	扫描操作执行时间，单位为秒。
file_scanned	bigint	-	扫描的文件数量。
data_size	double	-	扫描的数据量，单位为字节。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
billing_info	text	-	保留字段。

18.2.51 PG_OPCLASS

PG_OPCLASS系统表定义索引访问方法操作符类。

每个操作符类为一种特定数据类型和一种特定索引访问方法定义索引字段的语义。一个操作符类本质上指定一个特定的操作符族适用于一个特定的可索引的字段数据类型。索引的字段实际可用的族中的操作符集是接受字段的数据类型作为它们的左边的输入的那个。

表 18-52 PG_OPCLASS 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
opcmethod	oid	PG_AM.oid	操作符类所属的索引访问方法。
opcname	name	-	操作符类的名称。
opcnamespace	oid	PG_NAMESPACE.oid	操作符类所属的命名空间。
opcowner	oid	PG_AUTHID.oid	操作符类所有者。
opcfamily	oid	PG_OPFAMILY.oid	包含此操作符类的操作符族。
opcintype	oid	PG_TYPE.oid	操作符类索引的数据类型。
opcdefault	boolean	-	如果操作符类是opcintype的缺省，则为真。
opckeytype	oid	PG_TYPE.oid	索引数据的类型，如果和opcintype相同则为0。

一个操作符类的opcmethod必须匹配包含它的操作符族的opfmethod。同样，对于任意给定的opcmethod和opcintype的组合，不能有超过一个PG_OPCLASS行有opcdefault为真。

18.2.52 PG_OPERATOR

PG_OPERATOR系统表存储有关操作符的信息。

版权所有 © 华为云计算技术有限公司

表 18-53 PG_OPERATOR 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
oprname	name	-	操作符的名称。
oprnamespace	oid	PG_NAMESPACE.oid	包含此操作符的命名空间的OID。
oprowner	oid	PG_AUTHID.oid	操作符所有者。
oprkind	"char"	-	• b=infix =中缀（两边） • l=前缀（左边） • r=后缀（右边）
oprcanmerge	boolean	-	该操作符是否支持合并连接。
oprcanhash	boolean	-	该操作符是否支持Hash连接。
oprleft	oid	PG_TYPE.oid	左操作数的类型。
oprright	oid	PG_TYPE.oid	右操作数的类型。
oprresult	oid	PG_TYPE.oid	结果类型。
oprcom	oid	PG_OPERATOR.oid	此操作符的交换符（如果存在）。
oprnegate	oid	PG_OPERATOR.oid	此操作符的反转器（如果存在）。
oprcode	regproc	PG_PROC.oid	实现该操作符的函数。
oprrest	regproc	PG_PROC.oid	此操作符的约束选择性计算函数。
oprjoin	regproc	PG_PROC.oid	此操作符的连接选择性计算函数。

18.2.53 PG_OPFAMILY

PG_OPFAMILY系统表定义操作符族。

每个操作符族是操作符和相关支持例程的集合，这些例程实现了为特定索引访问方法指定的语义。此外，按照访问方法指定的某种方式，一个族内的操作符都是“兼容的”。操作符族允许跨数据类型操作符与索引一起使用，并可以推理使用访问方法语义相关内容。

 版权所有 © 华为云计算技术有限公司

表 18-54 PG_OPFAMILY 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
opfmethod	oid	PG_AM.oid	操作符族使用的索引方法。
opfname	name	-	操作符族的名称。
opfnamespace	oid	PG_NAMESPACE.oid	操作符族的命名空间。
opfowner	oid	PG_AUTHID.oid	操作符族的所有者。

定义一个操作符族的大多数信息不在PG_OPFAMILY，而是在相关的PG_AMOP，PG_AMPROC和PG_OPCLASS中。

18.2.54 PG_PARTITION

PG_PARTITION系统表存储数据库内所有分区表(partitioned table)、分区(table partition)、分区上toast表和分区索引(index partition)四类对象的信息。分区表索引(partitioned index)的信息不在PG_PARTITION系统表中保存。

表 18-55 PG_PARTITION 字段

名称	类型	描述
relname	name	分区表、分区、分区上toast表和分区索引的名称。
parttype	"char"	对象类型： • 'r'：partitioned table • 'p'：table partition • 'x'：index partition • 't'：toast table
parentid	oid	当对象为分区表或分区时，此字段表示分区表在PG_CLASS中的OID。 当对象为index partition时，此字段表示所属分区表索引(partitioned index)的OID。
rangenum	integer	保留字段。
intervalnum	integer	保留字段。
partstrategy	"char"	分区表分区策略，现仅支持： 'r'：范围分区。 'v'：数值分区。 'l'：列表分区。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
relfilenode	oid	table partition、index partition、分区上toast表的物理存储位置。
reltablespace	oid	table partition、index partition、分区上toast表所属表空间的OID。
relpages	double precision	统计信息：table partition、index partition的数据页数量。
reltuples	double precision	统计信息：table partition、index partition的元组数。
relallvisible	integer	统计信息：table partition、index partition的可见数据页数。
reltoastrelid	oid	table partition所对应toast表的OID。
reltoastidxid	oid	table partition所对应toast表的索引的OID。
indextblid	oid	index partition对应table partition的OID。
indisusable	boolean	分区索引是否可用。
reldeltarelid	oid	Delta表的OID。
reldeltaidx	oid	Delta表的索引表的OID。
relcudescrelid	oid	CU描述表的OID。
relcudescidx	oid	CU描述表的索引表的OID。
relfrozenxid	xid32	冻结事务ID号。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
intspnum	integer	间隔分区所属表空间的个数。
partkey	int2vector	分区键的列号。
intervaltablespace	oidvector	间隔分区所属的表空间，间隔分区以round-robin方式落在这些表空间内。
interval	text[]	间隔分区的间隔值。
boundaries	text[]	范围分区和间隔分区的上边界。
transit	text[]	间隔分区的跳转点。
reloptions	text[]	设置partition的存储属性，与pg_class.reloptions的形态一样，用"keyword=value"格式的字符串来表示，目前用于在线扩容的信息搜集。
relfrozenxid64	xid	冻结事务ID号。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
boundexprs	pg_node_tree	分区边界表达式。 ● 对于范围分区来说是分区上边界表达式。 ● 对于列表分区来说是分区边界枚举值集合。 pg_node_tree数据类型是不可读的，可用如下表达式pg_get_expr把当前字段单翻译为可读信息。 SELECT pg_get_expr(boundexprs, 0) FROM pg_partition WHERE relname = 'country_202201'; pg_get_expr ---------- ROW(202201, 'city1'::text), ROW(202201, 'city2'::text) (1 row)
relmetaversion	xid	元数据版本信息，该字段仅9.1.0及以上集群版本支持。

应用示例

查询分区表web_returns_p2的分区信息。

```
CREATE TABLE web_returns_p2
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);

SELECT oid FROM pg_class WHERE relname ='web_returns_p2';
  oid
-------
 97628

SELECT relname,parttype,parentid,boundaries FROM pg_partition WHERE parentid = '97628';
    relname     | parttype | parentid | boundaries
----------------+----------+----------+------------
 web_returns_p2 | r        |    97628 |
 p2016_0        | p        |    97628 | {20161231}
 p2016_1        | p        |    97628 | {20171231}
 p2016_2        | p        |    97628 | {20181231}
 p2016_3        | p        |    97628 | {20191231}
 p0             | p        |    97628 | {NULL}
(6 rows)
```

18.2.55 PG_PLTEMPLATE

PG_PLTEMPLATE系统表存储过程语言的“模板”信息。

 版权所有 © 华为云计算技术有限公司

表 18-56 PG_PLTEMPLATE 字段

名称	类型	描述
tmplname	name	该模板所应用的语言的名称。
tmpltrusted	boolean	如果语言被认为是可信的，则为真。
tmpldbacreate	boolean	如果语言是由数据库所有者创建的，则为真。
tmplhandler	text	调用处理器函数的名称。
tmplinline	text	匿名块处理器的名称，如果没有则为NULL。
tmplvalidator	text	校验函数的名称，如果没有则为NULL。
tmpllibrary	text	实现语言的共享库的路径。
tmplacl	aclitem[]	模板的访问权限（未使用）。

18.2.56 PG_PROC

PG_PROC系统表存储函数或过程的信息。

表 18-57 PG_PROC 字段

名称	类型	描述
proname	name	函数名。
pronamespace	oid	此函数所在命名空间的OID。
proowner	oid	函数的所有者。
prolang	oid	实现语言或函数的调用接口。
procost	real	估计执行成本。
prorows	real	结果行估计数。
provariadic	oid	参数元素的数据类型。
protransform	regproc	此函数的简化调用方式。
proisagg	boolean	函数是否为聚集函数。
proiswindow	boolean	函数是否为窗口函数。
prosecdef	boolean	函数是否为一个安全定义器（例如，一个“setuid”函数）。
proleakproof	boolean	函数有无其他影响。如果函数没有对参数进行防泄露处理，则会抛出错误。
proisstrict	boolean	如果任意调用参数为空，函数是否返回空值。这种情况下函数实际上根本不会被调用。非“strict”的函数必须准备处理空值输入。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
proretset	boolean	函数是否返回一个集合（即，指定数据类型的多个数值）。
provolatile	"char"	说明该函数的结果是只依赖于它的输入参数，或者还会被外接因素影响。 ● i表示“不可变的”（immutable）函数，对于相同的输入总是输出相同的结果。 ● s表示“稳定的”（stable）函数，对于固定输入其结果在一次扫描里不变。 ● v表示“易变”（volatile）函数，其结果可能在任何时候都变化。
pronargs	smallint	参数个数。
pronargdefaults	smallint	有默认值的参数个数。
prorettype	oid	返回参数类型的OID。
proargtypes	oidvector	函数参数的数据类型的数组。数组里只包括输入参数（包括INOUT参数），因此也表现了函数的调用特征。
proallargtypes	oid[]	函数参数的数据类型的数组。数组里包括所有参数的类型（包括OUT和INOUT参数），如果所有参数都是IN参数，则这个字段就会为空。注意数组下标是以1为起点的，而因为历史原因，proargtypes的下标起点为0。
proargmodes	"char"[]	函数参数模式的数组。 ● i表示IN参数。 ● o表示OUT参数。 ● b表示INOUT参数。 ● v表示VARIADIC参数。 ● t表示TABLE参数。 如果所有参数都是IN参数，则这个字段为空。注意此数组下标对应的是proallargtypes的位置，而不是proargtypes。
proargnames	text[]	函数参数的名字的数组。没有名字的参数在数组里设置为空字符串。如果没有一个参数有名字，这个字段为空。注意此数组的下标对应proallargtypes而不是proargtypes。
proargdefaults	pg_node_tree	默认值的表达式树。是PRONARGDEFAULTS元素的列表。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
prosrc	text	描述函数或存储过程的定义。例如，对于解释型语言来说就是函数的源程序，或者一个链接符号，一个文件名，或者函数和存储过程创建时指定的其他任何函数体内容，具体取决于语言/调用习惯的实现。
probin	text	关于如何调用该函数的附加信息。同样，其含义也是和语言相关的。
proconfig	text[]	函数针对运行时配置变量的本地设置。
proacl	aclitem[]	访问权限。具体请参见《数据仓库服务SQL语法参考》中“GRANT”和“REVOKE”章节。
prodefaultargpos	int2vector	函数默认值的位置，不局限于能最后几个参数才可有默认值。
fencedmode	boolean	函数的执行模式，表示函数是在fence还是not fence模式下执行。如果是fence执行模式，函数的执行会在重新fork的进程中执行。默认值是fence。
proshippable	boolean	函数是否可以下推到DN上执行，默认值是false。 ● 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。 ● 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。
propackage	boolean	该函数是否支持重载，主要针对Oracle风格的函数，默认值是false。

应用示例

查询指定函数的OID。例如，获取函数justify_days的OID为1295。

```
SELECT oid FROM pg_proc where proname ='justify_days';
 oid
------
 1295
(1 row)
```

查询指定函数是否为聚集函数。例如，查询justify_days函数为非聚集函数。

```
SELECT proisagg FROM pg_proc where proname ='justify_days';
 proisagg
----------
 f
(1 row)
```

18.2.57 PG_PUBLICATION

PG_PUBLICATION系统表存储当前数据库中创建的所有发布。该系统表仅8.2.0.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 18-58 PG_PUBLICATION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
pubname	name	-	发布的名称。
pubowner	oid	PG_AUTHID.oid	发布的拥有者。
puballtables	boolean	-	如果为true，这个发布自动包括数据库中的所有表，包括未来将会创建的任何表。
pubinsert	boolean	-	如果为true，为发布中的表复制INSERT操作。
pubupdate	boolean	-	如果为true，为发布中的表复制UPDATE操作。
pubdelete	boolean	-	如果为true，为发布中的表复制DELETE操作。
pubtruncate	boolean	-	如果为true，为发布中的表复制TRUNCATE操作。

应用示例

查看所有发布：

```
SELECT * FROM pg_publication;
 pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete | pubtruncate
---------+----------+--------------+-----------+-----------+-----------+-------------
 mypub   |       10 | t            | t         | t         | t         | t
(1 row)
```

18.2.58 PG_PUBLICATION_NAMESPACE

PG_PUBLICATION_NAMESPACE系统表存储当前数据库中的发布和模式之间的映射，这是一种多对多映射。该系统表仅8.2.0.100及以上集群版本支持。

表 18-59 PG_PUBLICATION_NAMESPACE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prpubid	oid	PG_PUBLICATION.oid	映射的发布的OID。
pnnspid	oid	PG_NAMESPACE.oid	映射的模式的OID。

版权所有 © 华为云计算技术有限公司

应用示例

查看所有发布和模式的映射：

```
SELECT * FROM pg_publication_namespace;
 pnpubid | pnnspid
---------+---------
   16797 |   16796
(1 row)
```

18.2.59 PG_PUBLICATION_REL

PG_PUBLICATION_REL系统表存储当前数据库中的发布和表之间的映射，这是一种多对多映射。该系统表仅8.2.0.100及以上集群版本支持。

说明

查询时推荐使用视图**PG_PUBLICATION_TABLES**，可以展现更详细的信息。

表 18-60 PG_PUBLICATION_REL 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prpubid	oid	**PG_PUBLICATION**.oid	映射的发布的OID。
prrelid	oid	**PG_CLASS**.oid	映射的表的OID。

应用示例

查看所有发布和表的映射：

```
postgres=# SELECT * FROM pg_publication_rel;
 prpubid | prrelid
---------+---------
   16797 |   16757
   16797 |   16776
(2 rows)
```

18.2.60 PG_RANGE

PG_RANGE系统表存储关于范围类型的信息。

除了**PG_TYPE**里类型的记录。

表 18-61 PG_RANGE 字段

名字	类型	引用	描述
rngtypid	oid	**PG_TYPE**.oid	范围类型的OID。
rngsubtype	oid	**PG_TYPE**.oid	该范围类型的元素类型（子类型）的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
rngcollation	oid	PG_COLLATION.oid	用于范围比较的排序规则的OID，如果没有则为0。
rngsubopc	oid	PG_OPCLASS.oid	用于范围比较的子类型的操作符类的OID。
rngcanonical	regproc	PG_PROC.oid	转换范围类型为规范格式的函数的OID，如果没有则为0。
rngsubdiff	regproc	PG_PROC.oid	返回两个double precision元素值的不同的函数的OID，如果没有则为0。

rngsubopc（如果元素类型是可排序的，则加上rngcollation）决定用于范围类型的排序顺序。当元素类型是时使用rngcanonical用于离散类型的元素类型。

18.2.61 PG_REDACTION_COLUMN

PG_REDACTION_COLUMN系统表存储脱敏列的信息。

表 18-62 PG_REDACTION_COLUMN 字段

名称	类型	描述
object_oid	oid	脱敏对象OID。
column_attrno	smallint	脱敏列attrno。
function_type	integer	脱敏类型。 **说明** 保留字段，仅为向前兼容低版本的脱敏列信息，可取值为0（NONE）、1（FULL）。
function_parameters	text	脱敏类型为partial类型时的参数。（保留字段，无实际意义）
regexp_pattern	text	脱敏类型为regexp时，格式化字符串。（保留字段，无实际意义）
regexp_replace_string	text	脱敏类型为regexp时，替换串。（保留字段，无实际意义）
regexp_position	integer	脱敏类型为regexp时，起始替换位置。（保留字段，无实际意义）
regexp_occurrence	integer	脱敏类型为regexp时，替换次数。（保留字段，无实际意义）
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数。（保留字段，无实际意义）
column_description	text	脱敏列描述信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
function_expr	pg_node_tree	脱敏函数的内部表现形式。
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。
policy_oid	oid	所属脱敏策略OID。 该字段8.2.1.100及以上集群版本支持，用于查询时直接从系统表元数据检索脱敏列信息。

18.2.62 PG_REDACTION_POLICY

PG_REDACTION_POLICY系统表提供了脱敏对象的信息。

表 18-63 PG_REDACTION_POLICY 字段

名称	类型	描述
object_oid	oid	脱敏对象OID。
policy_name	name	脱敏策略名称。
enable	boolean	策略状态（开启、关闭）。 **说明** enable的取值： • true表示开启 • false表示关闭
expression	pg_node_tree	策略生效表达式（针对用户）。
policy_description	text	策略描述信息。
inherited	bool	说明脱敏策略是否“继承”自其他脱敏策略。
policy_order	float4	脱敏策略次序。该字段8.2.1.100及以上集群版本支持。

18.2.63 PG_RELFILENODE_SIZE

PG_RELFILENODE_SIZE系统表存储文件级空间统计信息，表中的每一条记录则对应磁盘上相应的物理文件和该文件的文件大小。

版权所有 © 华为云计算技术有限公司

表 18-64 PG_RELFILENODE_SIZE 字段

名称	类型	描述
databaseid	oid	物理文件所属database对应的OID。如果是跨库共享系统表，该值为0。
tablespaceid	oid	物理文件所属表空间对应的OID。
relfilenode	oid	物理文件的物理文件编号。
backendid	integer	创建物理文件的后台线程号，通常为-1。
type	integer	物理文件的文件类型。 ● 0为数据类型。 ● 1为FSM文件类型。 ● 2为VM文件类型。 ● 3为BCM文件类型。 ● 大于4为列存表对应列的数据文件和BCM文件大小之和。
filesize	bigint	物理文件的文件大小，单位Byte。

18.2.64 PG_RLSPOLICY

PG_RLSPOLICY系统表存储行级访问控制策略的信息。

表 18-65 PG_RLSPOLICY 字段

名称	类型	描述
polname	name	行访问控制策略名称。
polrelid	oid	行访问控制策略的表OID。
polcmd	char	行访问控制策略影响的SQL操作，包括：*(ALL)、r(SELECT)、w(UPDATE)、d(DELETE)。
polpermissive	boolean	行访问控制策略的类型。 **说明** polpermissive的取值： ● true表示PERMISSIVE，表示行访问控制策略是宽容性策略。 ● false表示RESTRICTIVE，表示行访问控制策略是限制性策略。
polroles	oid[]	行访问控制策略影响的数据库用户OID。
polqual	pg_node_tree	行访问控制策略的SQL条件表达式。

 版权所有 © 华为云计算技术有限公司

18.2.65 PG_RESOURCE_POOL

PG_RESOURCE_POOL系统表提供了数据库资源池的信息。

表 18-66 PG_RESOURCE_POOL 字段

名称	类型	描述
respool_name	name	资源池名称。
mem_percent	integer	内存配置的百分比，0代表资源池内存不管控。
cpu_affinity	bigint	保留字段，无实际意义。
control_group	name	资源池所在的control group名字。
active_statements	integer	资源池上最大的并发数。
max_dop	integer	资源池允许的简单作业最大并发数。-1和0代表不限制。
memory_limit	name	单个查询估算内存上限。
parentid	oid	父资源池OID。
io_limits	integer	保留字段，无实际意义。
io_priority	text	保留字段，无实际意义。
nodegroup	name	资源池关联的逻辑集群名称，非逻辑集群下为“installation”。
is_foreign	boolean	表示资源池是否用于逻辑集群之外的用户。如果为true，表示资源池用来控制不属于当前资源池的普通用户的资源。
short_acc	boolean	资源池是否开启短查询加速，默认开启。 ● 短查询加速开启，简单查询在快车道管控。 ● 短查询加速关闭，简单查询在慢车道管控。
except_rule	text	资源池关联的异常规则，支持关联多个异常规则，异常规则间用逗号分隔。
weight	integer	资源调度权重，目前仅用于网络调度。

18.2.66 PG_REWRITE

PG_REWRITE系统表存储为表和视图定义的重写规则。

表 18-67 PG_REWRITE 字段

名称	类型	描述
rulename	name	规则名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
ev_class	oid	使用该规则的表名。
ev_attr	smallint	该规则适用的字段（目前总是为0，表示整个表）。
ev_type	"char"	规则适用的事件类型： • 1 = SELECT • 2 = UPDATE • 3 = INSERT • 4 = DELETE
ev_enabled	"char"	用于控制复制的触发： • O =“origin”和“local”模式时触发。 • D =禁用触发。 • R =“replica”时触发。 • A =任何模式都会触发。
is_instead	boolean	如果是INSTEAD规则，则为真。
ev_qual	pg_node_tree	规则条件的表达式树（以nodeToString() 形式存在）。
ev_action	pg_node_tree	规则动作的查询树（以nodeToString() 形式存在）。
state_change	timestamp with time zone	ev_enabled字段的刷新时间。该字段仅9.1.0.200及以上集群版本支持。

18.2.67 PG_SECLABEL

PG_SECLABEL系统表存储数据对象上的安全标签。

PG_SHSECLABEL的作用类似，只是用于在一个数据库集群内共享的数据库对象的安全标签上。

表 18-68 PG_SECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象的系统目录的OID。
objsubid	integer	-	出现在此对象中的列的序号。
provider	text	-	与此标签相关的标签提供者。
label	text	-	应用于此对象的安全标签。

版权所有 © 华为云计算技术有限公司

18.2.68 PG_SHDEPEND

PG_SHDEPEND系统表记录数据库对象和共享对象（比如角色）之间的依赖关系。记录的这些信息使DWS可以确保对象在被删除时没有被其他对象引用。

PG_DEPEND的作用类似，只是它是针对单个数据库中对象之间的依赖。

和大多数其他系统表不同，PG_SHDEPEND在集群的所有数据库之间共享：每个数据库集群只有一个PG_SHDEPEND，并非每个数据库一个。

表 18-69 PG_SHDEPEND 字段

名字	类型	引用	描述
dbid	oid	PG_DATABASE.oid	依赖对象所在的数据库的OID，如果是共享对象，则为0。
classid	oid	PG_CLASS.oid	依赖对象所在的系统表的OID。
objid	oid	任意OID属性	指定的依赖对象的OID。
objsubid	integer	-	对于一个表字段，为字段号（objid和classid参考表本身）。对于所有其他对象类型，该字段为0。
refclassid	oid	PG_CLASS.oid	被引用对象所在的系统表的OID(必须是一个共享表)。
refobjid	oid	任意OID属性	指定的被引用对象的OID。
deptype	"char"	-	定义该依赖关系的特定语义的代码见表后说明。
objfile	text	-	用户定义C函数库文件路径。

在任何情况下，一条PG_SHDEPEND记录就表明被引用的对象不能在未删除依赖对象的前提下被删除。但是其中也有几种依赖类型由deptype定义的情况：

- SHARED_DEPENDENCY_OWNER (o)

  被引用的对象（必须是一个角色）是依赖对象的所有者。
- SHARED_DEPENDENCY_ACL (a)

  在依赖对象的ACL（访问控制列表，也就是权限列表）中提到被引用的对象（必须是一个角色）。不会为对象的所有者创建SHARED_DEPENDENCY_ACL，因为所有者将具有SHARED_DEPENDENCY_OWNER记录。
- SHARED_DEPENDENCY_PIN (p)

  没有依赖对象。这类记录标识系统自身依赖于被依赖对象，因此这种对象绝对不能被删除。此类型的记录只能被initdb创建，依赖对象的字段都为0。

18.2.69 PG_SHDESCRIPTION

PG_SHDESCRIPTION系统表存储共享数据库对象的可选注释。可以使用COMMENT命令操作注释的内容，使用gsql的\d命令查看注释内容。

PG_DESCRIPTION提供了类似的功能，它记录了单个数据库中对象的注释。

 版权所有 © 华为云计算技术有限公司

不同于大多数系统表，PG_SHDESCRIPTION在集群中所有数据库之间共享：每个数据库集群只有一个PG_SHDESCRIPTION，而不是每个数据库一个。

表 18-70 PG_SHDESCRIPTION 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此描述所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象所在的系统目录的OID。
description	text	-	作为对该对象描述的任意文本。

18.2.70 PG_SHSECLABEL

PG_SHSECLABEL系统表存储在共享数据库对象上的安全标签。安全标签可以用SECURITY LABEL命令操作。

查看安全标签的简单点的方法，请参阅PG_SECLABELS。

PG_SECLABEL的作用类似，只是它是用于在单个数据库内部的对象的安全标签的。

不同于大多数的系统表，PG_SHSECLABEL在一个集群中的所有数据库中共享：每个数据库集群只有一个PG_SHSECLABEL，而不是每个数据库一个。

表 18-71 PG_SHSECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属对象的OID。
classoid	oid	PG_CLASS.oid	对象所属系统目录的OID。
provider	text	-	与此标签关联的标签提供者。
label	text	-	应用于该对象的安全标签。

18.2.71 PG_STATISTIC

PG_STATISTIC系统表存储有关该数据库中表和索引列的统计数据。需要有系统管理员权限才可以访问此系统表。

表 18-72 PG_STATISTIC 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
staattnum	smallint	所描述的字段在表中的编号，从1开始。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	标识全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 ● 大于0的值是独立数值的实际数目。 ● 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。 n的取值范围：1～5
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则就是NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的办法。 n的取值范围：1～5
stadndistinct	real	标识dn1上字段里唯一的非NULL数据值的数目。 ● 大于0的值是独立数值的实际数目。 ● 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
staextinfo	text	统计信息的扩展信息。预留字段。

18.2.72 PG_STATISTIC_DATA

PG_STATISTIC_DATA系统表存储当前集群上表的扩展统计信息数据，包括表达式统计信息数据。该系统表仅8.2.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-73 PG_STATISTIC_DATA 字段

名称	类型	描述
statoid	oid	扩展统计信息对象的OID。
inherit	boolean	是否统计有继承关系的对象。 ● true表示该表有子表。 ● false表示该表无子表。
ndistinct	pg_ndistinct	N-distinct计数，序列化为pg_ndistinct类型。
dependencies	pg_dependencies	函数依赖统计信息，序列化为pg_dependencies类型。
mcv	pg_mcv_list	最常见值（MCV）统计信息，序列化为pg_mcv_list类型。
expr	pg_statistic[]	每个表达式的统计信息，序列化为pg_statistic类型的数组。

18.2.73 PG_STATISTIC_EXT

PG_STATISTIC_EXT系统表存储有关该数据库中表的扩展统计数据。收集的扩展统计数据范围由用户指定，需要有系统管理员权限才可以访问此系统表。

表 18-74 PG_STATISTIC_EXT 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	标识全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
stadndistinct	real	标识dn1上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	一个用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5
stakey	int2vector	所描述的字段编号的数组。
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。 n的取值范围：1～5
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则为NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的办法。 n的取值范围：1～5
staexprs	pg_node_tree	扩展统计信息对应的表达式。

18.2.74 PG_STATISTIC_INFO

PG_STATISTIC_INFO系统表存储当前集群上表的扩展统计信息对象，目前包括表达式统计信息对象。该系统表仅8.2.1.200及以上集群版本支持。

表 18-75 PG_STATISTIC_INFO 字段

名称	类型	描述
relid	oid	扩展统计信息对象中使用的表。
statname	name	扩展统计信息对象的名称。
statnamespace	oid	包含该扩展统计信息对象的命名空间的OID。
statowner	oid	该扩展统计信息对象的所有者OID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
stattarget	integer	用于控制统计信息的详细级别。
keys	int2vector	所描述的字段编号的数组，表示该统计信息对象包括哪些列。
kind	"char"[]	包含统计类型的数组，可能的值有： d：表示N-distinct计数统计信息。 f：表示函数依赖统计信息。 m：表示最常见值（MCV）统计信息。 e：表示表达式统计信息。
exprs	pg_node_tree	扩展统计信息对应的表达式。

18.2.75 PG_STAT_OBJECT

PG_STAT_OBJECT系统表存储当前实例上表的统计信息和autovacuum效率信息，并且对于databaseid, relid, partid字段创建索引。该系统表的更新受**enable_pg_stat_object**参数控制。该系统表仅8.2.1及以上集群版本支持。

表 18-76 PG_STAT_OBJECT 字段

名称	类型	引用	描述
databaseid	oid	**PG_DATABASE**.oid	数据库OID。
relid	oid	**PG_CLASS**.oid	表OID，分区表为主表OID。
partid	oid	**PG_PARTITION** .oid	分区OID，普通表此列为0。
numscans	bigint	-	启动顺序扫描的次数。
tuples_returned	bigint	-	顺序扫描抓取的可见元组条数。
tuples_fetched	bigint	-	抓取的可见元组条数。
tuples_inserted	bigint	-	插入条数。
tuples_updated	bigint	-	更新条数。
tuples_deleted	bigint	-	删除条数。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
tuples_hot_updated	bigint	-	HOT更新条数。
n_live_tuples	bigint	-	可见元组数。
last_autovacuum_begin_n_dead_tuple	bigint	-	Autovacuum执行前删除元组数。
n_dead_tuples	bigint	-	Autovacuum成功后删除元组数。
changes_since_analyze	bigint	-	Analyze后最近一次数据修改时间。
blocks_fetched	bigint	-	选中的页面数。
blocks_hit	bigint	-	扫描过的页面数。
cu_mem_hit	bigint	-	CU内存命中次数。
cu_hdd_sync	bigint	-	从磁盘同步读取CU次数。
cu_hdd_asyn	bigint	-	从磁盘异步读取CU次数。
data_changed_timestamp	timestamp with time zone	-	最近一次数据修改时间。
data_access_timestamp	timestamp with time zone	-	表的最后一次访问时间。
analyze_timestamp	timestamp with time zone	-	最近一次analyze时间。
analyze_count	bigint	-	Analyze总次数。
autovac_analyze_timestamp	timestamp with time zone	-	最近一次autoanalyze时间。
autovac_analyze_count	bigint	-	Autoanalyze总次数。
vacuum_timestamp	timestamp with time zone	-	最近一次vacuum的时间。
vacuum_count	bigint	-	vacuum总次数。
autovac_vacuum_timestamp	timestamp with time zone	-	最近一次autovacuum时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
autovac_vacuum_count	bigint	-	Autovacuum总次数。
autovacuum_success_count	bigint	-	成功执行的autovacuum总次数。
last_autovacuum_time_cost	bigint	-	最近一次成功的autovacuum花费时间，单位：微秒。
avg_autovacuum_time_cost	bigint	-	成功执行autovacuum的平均执行时间，单位：微秒。
last_autovacuum_failed_count	bigint	-	从上一次autovacuum成功到现在，autovacuum总失败次数。
last_autovacuum_trigger	smallint	-	最近一次autovacuum触发方式，用于辅助维护人员进行vacuum情况的判断。
last_autovacuum_oldestxmin	bigint	-	最近一次autovacuum成功执行后的oldestxmin。如果表级oldestxmin特性开启，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值。
last_autovacuum_scan_pages	bigint	-	最近一次autovacuum扫描的页面数（仅针对行存表）。
last_autovacuum_dirty_pages	bigint	-	最近一次autovacuum修改的页面数（仅针对行存表）。
last_autovacuum_clear_deadtuples	bigint	-	最近一次autovacuum清理的deadtuple数（仅针对行存表）。
sum_autovacuum_scan_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计扫描的页面数（仅针对行存表）。
sum_autovacuum_dirty_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计修改的页面数（仅针对行存表）。
sum_autovacuum_clear_deadtuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的deadtuple数（仅针对行存表）。
last_autovacuum_begin_cu_size	bigint	-	最近一次autovacuum前的CU文件大小（仅针对列存表）。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
last_autovacuum_cu_size	bigint	-	最近一次autovacuum后的CU文件大小（仅针对列存表）。
last_autovacuum_rewrite_size	bigint	-	最近一次autovacuum重写的列存文件大小（仅针对列存表）。
last_autovacuum_clear_size	bigint	-	最近一次autovacuum清理的列存文件大小（仅针对列存表）。
last_autovacuum_clear_cbtree_tuples	bigint	-	最近一次autovacuum清理的cbtree tuple数（仅针对列存表）。
sum_autovacuum_rewrite_size	bigint	-	从数据库初始化开始到现在，autovacuum累计重写的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_size	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_cbtree_tuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的cbtree tuple数（仅针对列存表）。
last_autovacuum_csn	bigint	-	如果表级oldestxmin特性打开，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值对应的CSN值。
last_automerge_timestamp	timestamp with time zone	-	上次表发生automerge的时间（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
last_automerge_time_cost	bigint	-	上次表发生automerge的耗时（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
last_automerge_count	bigint	-	上次表automerge的记录数（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
extra1	bigint	-	预留字段1。

18.2.76 PG_SUBSCRIPTION

PG_SUBSCRIPTION系统表存储所有现有的订阅。

 版权所有 © 华为云计算技术有限公司

表 18-77 PG_SUBSCRIPTION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
subdbid	oid	PG_DATABASE.oid	订阅所在的数据库的OID。
subname	name	-	订阅的名称。
subowner	oid	PG_AUTHID.oid	订阅的拥有者。
subenabled	boolean	-	如果为真，订阅被启用并且应该被复制。
subconninfo	text	-	到发布端数据库的连接信息。
subslotname	text	-	发布端数据库中复制槽的名称。空表示为NONE。
subpublications	text[]	-	被订阅的发布名称的数组。这些引用的是发布者服务器上的发布。

应用示例

查看所有订阅：

```
SELECT * FROM pg_subscription;
 subdbid | subname | subowner | subenabled |
subconninfo                                | subslotname | subpublications
---------+---------+----------+------------
+--------------------------------------------------------------------------------------------+-------------+-----------------
   15992 | mysub   |       10 | t          | host=1.1.1.1,2.2.2.2 port=10000,20000 dbname=postgres user=repusr1
password=password | mysub       | {mypub}
(1 row)
```

18.2.77 PG_SYNONYM

PG_SYNONYM系统表存储同义词对象名与其他数据库对象名间的映射信息。

表 18-78 PG_SYNONYM 字段

名称	类型	描述
synname	name	同义词名称。
synnamespace	oid	该同义词所在的命名空间的OID。
synowner	oid	同义词的所有者，通常是其创建者OID。
synobjschema	name	关联对象指定的模式名。
synobjname	name	关联对象名。

 版权所有 © 华为云计算技术有限公司

18.2.78 PG_TABLESPACE

PG_TABLESPACE系统表存储表空间信息。

表 18-79 PG_TABLESPACE 字段

名称	类型	描述
spcname	name	表空间名。
spcowner	oid	表空间的所有者，通常是其创建者。
spcacl	aclitem[]	访问权限。具体请参见《数据仓库服务SQL语法参考》中“GRANT”和“REVOKE”章节。
spcoptions	text[]	表空间的选项。
spcmaxsize	text	可使用的最大磁盘空间大小，单位Byte。

18.2.79 PG_TRIGGER

PG_TRIGGER系统表存储触发器信息。

名称	类型	描述
tgrelid	oid	触发器所在表的OID。
tgname	name	触发器名。
tgfoid	oid	触发器OID。
tgtype	smallint	触发器类型。
tgenabled	"char"	O表示触发器在“origin”和“local”模式下触发。 D表示触发器被禁用。 R表示触发器在“replica”模式下触发。 A表示触发器始终触发。
tgisinternal	boolean	内部触发器标识，如果为true表示内部触发器。
tgconstrrelid	oid	完整性约束引用的表。
tgconstrindid	oid	完整性约束的索引。
tgconstraint	oid	约束触发器在pg_constraint中的OID。
tgdeferrable	boolean	约束触发器是为DEFERRABLE类型。
tginitdeferred	boolean	约束触发器是否为INITIALLY DEFERRED类型。
tgnargs	smallint	触发器函数入参个数。
tgattr	int2vector	当触发器指定列时的列号，未指定则为空数组。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tgargs	bytea	传递给触发器的参数。
tgqual	pg_node_tree	表示触发器的WHEN条件，如果没有则为null。

18.2.80 PG_TS_CONFIG

PG_TS_CONFIG系统表包含表示文本搜索配置的选项。一个配置指定一个特定的文本搜索解析器和一个用于解析器输出类型的字典列表。

解析器在PG_TS_CONFIG记录中显示，但是字典映射的标记是由PG_TS_CONFIG_MAP中的辅助记录定义的。

表 18-80 PG_TS_CONFIG 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
cfgname	name	-	文本搜索配置名。
cfgnamespace	oid	PG_NAMESPACE.oid	此配置所在的命名空间的OID。
cfgowner	oid	PG_AUTHID.oid	配置的所有者。
cfgparser	oid	PG_TS_PARSER.oid	此配置的文本搜索解析器的OID。
cfoptions	text[]	-	分词相关配置选项。

18.2.81 PG_TS_CONFIG_MAP

PG_TS_CONFIG_MAP系统表包含为每个文本搜索配置的解析器的每种输出符号类型，显示有哪些文本搜索字典可供查询以及以哪种顺序搜索。

表 18-81 PG_TS_CONFIG_MAP 字段

名字	类型	引用	描述
mapcfg	oid	PG_TS_CONFIG.oid	拥有此映射记录的PG_TS_CONFIG的OID。
maptokentype	integer	-	由配置的解析器发出的一个符号类型。
mapseqno	integer	-	查询该项的顺序。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
mapdict	oid	PG_TS_DICT.oid	查询的文本搜索字典的OID。

18.2.82 PG_TS_DICT

PG_TS_DICT系统表包含定义文本搜索字典的项。字典取决于文本搜索模板，该模板显示所有需要实现的功能。字典本身提供了用户可设置参数的模板。

即允许字典通过非权限用户创建。参数由文本字符串dictinitoption指定，参数的格式和意义取决于模板。

表 18-82 PG_TS_DICT 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
dictname	name	-	文本搜索字典名。
dictnamespace	oid	PG_NAMESPACE.oid	此字典所在的命名空间的OID。
dictowner	oid	PG_AUTHID.oid	字典的所有者。
dicttemplate	oid	PG_TS_TEMPLATE.oid	此字典的文本搜索模板的OID。
dictinitoption	text	-	模板的初始化选项字符串。

18.2.83 PG_TS_PARSER

PG_TS_PARSER系统表包含定义文本解析器的项。解析器负责分割输入文本为词位，并且为每个词位分配标记类型。因为解析器必须通过C语言级别的函数实现，所以新解析器必须由数据库系统管理员创建。

表 18-83 PG_TS_PARSER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prsname	name	-	文本搜索解析器名称。
prsnamespace	oid	PG_NAMESPACE.oid	解析器所在的命名空间的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
prsstart	regproc	PG_PROC.oid	解析器的启动函数的OID。
prstoken	regproc	PG_PROC.oid	解析器的下一个标记函数的OID。
prsend	regproc	PG_PROC.oid	解析器的关闭函数的OID。
prsheadline	regproc	PG_PROC.oid	解析器的标题函数的OID。
prslextype	regproc	PG_PROC.oid	解析器的lextype函数的OID。

18.2.84 PG_TS_TEMPLATE

PG_TS_TEMPLATE系统表包含定义文本搜索模板的项。模板是文本搜索字典的类的实现框架。因为模板必须通过C语言级别的函数实现，索引新模板的创建必须由数据库系统管理员创建。

表 18-84 PG_TS_TEMPLATE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
tmplname	name	-	文本搜索模板名。
tmplnamespace	oid	PG_NAMESPACE.oid	模板所属的命名空间的OID。
tmplinit	regproc	PG_PROC.oid	模板的初始化函数的OID。
tmpllexize	regproc	PG_PROC.oid	模板的lexize函数的OID。

18.2.85 PG_TYPE

PG_TYPE系统表存储数据类型的相关信息。

表 18-85 PG_TYPE 字段

名称	类型	描述
typname	name	数据类型名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
typnamespace	oid	此类型所在的命名空间的OID。
typowner	oid	此类型的所有者。
typlen	smallint	对于定长类型是该类型内部表现形式的字节数。对于变长类型typlen为负值。 ● -1表示一种“变长”（有长度字属性的数据）。 ● -2表示一个以NULL结尾的C字符串。
typbyval	boolean	指定内部传递这个类型的数值时是传值还是传引用。如果该类型的TYPLEN不是1，2，4，8， TYPBYVAL建议设置为false。变长类型通常是传引用。即使TYPLEN允许传值，TYPBYVAL也可以为false。
typtype	"char"	● b表示基础类型。 ● c表示复合类型（比如，一个表的行类型）。 ● e表示枚举类型。 ● p表示伪类型。 参见typrelid和typbasetype。
typcategory	"char"	数据类型的模糊分类，可用于解析器使用的数据转换依据。
typispreferred	boolean	如果为真，则数据符合TYPCATEGORY所指定的转换规则时进行转换。
typisdefined	boolean	如果定义了类型，则为真。如果是一种尚未定义的类型的占位符，则为假。如果为假，则除了该类型名称，命名空间和OID之外没有可依赖的对象。
typdelim	"char"	分析数组输入时，分隔两个此类型数值的字符。请注意，分隔符是与数组元素数据类型相关联，而不是与数组数据类型相关联。
typrelid	oid	如果是复合类型（请参见typtype），则此字段指向pg_class中定义该表的行。对于独立的复合类型，pg_class记录并不表示一个表，但是总需要它来查找该类型连接的pg_attribute记录。非复合类型为0。
typelem	oid	如果不为0，则它标识pg_type中的另一行。当前类型可以像一个产生类型为typelem的数组来描述。“true”数组类型是变长的（typlen= -1），但是某些定长（typlen > 0）类型也有非零的typelem（比如name和point）。如果一个定长类型有typelem，则其内部形式必须是typelem数据类型的某个数目的个数值，不能有其他数据。变长数组类型有一个该数组子过程定义的头（文件）。
typarray	oid	如果不为0，则表示在pg_type中有对应的类型记录。
typinput	regproc	输入转换函数（文本格式）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
typoutput	regproc	输出转换函数（文本格式）。
typreceive	regproc	输入转换函数（二进制格式），如果没有则为0。
typsend	regproc	输出转换函数（二进制格式），如果没有则为0。
typmodin	regproc	输入类型修改符函数，如果为0，则不支持。
typmodout	regproc	输出类型修改符函数，如果为0，则不支持。
typanalyze	regproc	自定义的ANALYZE函数，如果使用标准函数，则为0。
typalign	"char"	当存储此类型的数值时要求的对齐方式。适用于磁盘存储以及该值在数据库中的大多数形式。如果数值是连续存储的，比如在磁盘上以完全的裸数据的形式存放时，则先在此类型的数据前填充空白，这样它就可以按照要求的边界存储。对齐引用是该序列中第一个数据的开头。可能的值包含： ● c = char对齐，即不需要对齐。 ● s = short对齐（在大多数机器上是2字节） ● i = int对齐（在大多数机器上是4字节） ● d = double对齐（在大多数机器上是8字节，但不一定是全部） **须知** 对于系统表里使用的类型，在pg_type里定义的尺寸和对齐方式要和编译器在表示表行的结构中布局方式保持一致。
typstorage	"char"	指明一个变长类型（那些有typlen = -1）是否准备好应付非常规值，以及对这种属性的类型的缺省策略是什么。可能的值包含： ● p：数值总是以简单方式存储。 ● e：数值可以存储在一个"次要"关系中（如果有该关系，请参见pg_class.reltoastrelid）。 ● m：数值可以以内联压缩方式存储。 ● x：数值可以以内联压缩方式或者在"次要"表里存储。 **须知** m域也可以移到从属表里存储，但只是最后的解决方法（首先移动e和x域）。
typnotnull	boolean	表示在某类型上的一个NOTNULL约束。目前只用于域。
typbasetype	oid	如果这是一个衍生类型（请参见typtype），则该标识作为这个类型的基础的类型。如果不是衍生类型则为零。
typtypmod	integer	域使用typtypmod记录要应用于其基础类型上的typmod（如果基础类型不使用typmod，则为-1）。如果此类型不是域，则为-1。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
typndims	integer	如果一个域是数组，则typndims是数组维数的数值（即typbasetype是一个数组类型；域的typelem将匹配基本类型的typelem）。除了数组类型的域以外的类型为0。
typcollation	oid	指定类型的排序规则。如果为0，则表示不支持排序规则。
typdefaultbin	pg_node_tree	如果不为NULL，则为该类型缺省表达式的nodeToString() 表现形式。目前这个字段只用于域。
typdefault	text	如果某类型没有相关缺省值，则为NULL。如果typdefaultbin不为NULL，则typdefault必须包含一个typdefaultbin代表的缺省表达式的人类可读版本。如果typdefaultbin为NULL但typdefault不为NULL，typdefault则是该类型缺省值的外部表现形式，可以将其输入到类型的转换器生成一个常量。
typacl	aclitem[]	访问权限。

18.2.86 PG_USER_MAPPING

PG_USER_MAPPING系统表存储从本地用户到远程的映射。

需要有系统管理员权限才可以访问此系统表。普通用户可以使用视图PG_USER_MAPPINGS进行查询。

表 18-86 PG_USER_MAPPING 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
umuser	oid	PG_AUTHID.oid	被映射的本地用户的OID，如果用户映射是公共的则为0。
umserver	oid	PG_FOREIGN_SERVER.oid	包含此映射的外部服务器的OID。
umoptions	text[]	-	用户映射指定选项，使用“keyword=value”格式的字符串。

18.2.87 PG_USER_STATUS

PG_USER_STATUS系统表提供了访问数据库用户的状态。需要有系统管理员权限才可以访问此系统表

 版权所有 © 华为云计算技术有限公司

表 18-87 PG_USER_STATUS 字段

名称	类型	描述
roloid	oid	角色的标识。
failcount	integer	尝试失败次数。
locktime	timestamp with time zone	角色被锁定的时间点。
rolstatus	smallint	角色的状态。 ● 0：正常状态。 ● 1：由于登录失败次数超过阈值被锁定了一定的时间。 ● 2：被管理员锁定。
permspace	bigint	角色在当前实例上已经使用的永久表存储空间大小。
tempspace	bigint	角色在当前实例上已经使用的临时表存储空间大小。

18.2.88 PG_WORKLOAD_ACTION

PG_WORKLOAD_ACTION系统表存储query_band的信息。

表 18-88 PG_WORKLOAD_ACTION 字段

名称	类型	描述
qband	name	query band键值对。
class	name	query band关联行为类别。
object	name	query band关联行为。
action	name	query band关联行为表现。

18.2.89 PGXC_CLASS

PGXC_CLASS系统表存储每张表的复制或分布信息。

表 18-89 PGXC_CLASS 字段

名称	类型	描述
pcrelid	oid	表的OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
pclocatortype	"char"	定位器类型。 ● H：hash ● M：Modulo ● N：Round Robin ● R：Replicate
pchashalgorithm	smallint	使用哈希算法分布元组。
pchashbuckets	smallint	哈希容器的值。
pgroup	name	节点组名称。
redistributed	"char"	表已经完成重分布。
redis_order	integer	重分布的顺序。
pcattnum	int2vector	用作分布键的列标号。
nodeoids	oidvector_extend	表分布的节点OID列表。
options	text	系统内部保留字段，存储扩展状态信息。

18.2.90 PGXC_GROUP

PGXC_GROUP系统表存储节点组信息，在存算分离3.0版本中，每个逻辑集群节点组称为一个VW（Virtual Warehouse），而在存储KV层，每一个VW会和一个vgroup相对应。

表 18-90 PGXC_GROUP 字段

名称	类型	描述
group_name	name	节点组名称。
in_redistribution	"char"	是否需要重分布： ● n表示NodeGroup没有再进行重分布。 ● y表示NodeGroup是重分布过程中的源节点组。 ● t表示NodeGroup是重分布过程中的目的节点组。 ● s表示NodeGroup不需要重分布，重分布过程将跳过此节点组。
group_members	oidvector_extend	节点组的节点OID列表。
group_buckets	text	分布数据桶的集合。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
is_installation	boolean	是否安装子集群。
group_acl	aclitem[]	访问权限。
group_kind	"char"	节点组类型： ● i表示安装节点组，包含所有DN节点。 ● n表示普通非逻辑集群节点组。 ● v表示逻辑集群节点组。 ● e表示弹性集群节点组 ● r表示复制表节点组，只能用于创建复制表，可以包含一个或多个逻辑集群节点组。
group_ckpt_csn	xid	节点组最近一次执行增量抽取的CSN。
vgroup_id	xid	节点组对应vgroup的ID标识。
vgroup_bucket_count	oid	节点组对应vgroup的桶数目。
group_ckpt_time	timestamp with time zone	节点组最近一次执行增量抽取的物理时间。
apply_kv_duration	integer	节点组最近一次执行增量抽取中增量扫描耗时(单位为秒)。
ckpt_duration	integer	节点组最近一次执行增量抽取中checkpoint耗时（单位为秒）。

18.2.91 PGXC_NODE

PGXC_NODE系统表存储集群节点信息。

表 18-91 PGXC_NODE 字段

名称	类型	描述
node_name	name	节点名称。
node_type	"char"	节点类型。 C：协调节点。 D：数据节点。
node_port	integer	节点的端口号。
node_host	name	节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
node_port1	integer	复制节点的端口号。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
node_host1	name	复制节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
hostis_primary	boolean	表明当前节点是否发生主备切换。
nodeis_primary	boolean	在replication表下，是否优选当前节点作为优先执行的节点进行非查询操作。
nodeis_preferred	boolean	在replication表下，是否优选当前节点作为首选的节点进行查询。
node_id	integer	节点标识符。
sctp_port	integer	主节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port	integer	主节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
sctp_port1	integer	备节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port1	integer	备节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
nodeis_central	boolean	当前节点为中心控制节点。

应用示例

查询集群的CN和DN信息：

```
SELECT * FROM pgxc_node;
  node_name   | node_type | node_port | node_host | node_port1 | node_host1 | hostis_primary |
nodeis_primary | nodeis_preferred |   node_id
 | sctp_port | control_port | sctp_port1 | control_port1 | nodeis_central | read_only
--------------+-----------+-----------+-----------+------------+------------+----------------+----------------
+------------------+------------
-+-----------+-------------+------------+--------------+----------------+-----------
 datanode1    | D        |    55504 | localhost |     55504 | localhost  | t            | f            | f            |
888802358
 |    55505 |       55507 |         0 |            0 | f            | f
 datanode2    | D        |    55508 | localhost |     55508 | localhost  | t            | f            | f            |
-905831925
 |    55509 |       55511 |         0 |            0 | f            | f
 coordinator1 | C        |    55500 | localhost |     55500 | localhost  | t            | f            | f            |
1938253334
 |         0 |           0 |         0 |            0 | t            | f
 datanode3    | D        |    55542 | localhost |     55542 | localhost  | t            | f            | f            |
-1894792127
 |    57552 |       55544 |         0 |            0 | f            | t
 datanode4    | D        |    55546 | localhost |     55546 | localhost  | t            | f            | f            |
-1307323892
 |    57808 |       55548 |         0 |            0 | f            | t
 datanode5    | D        |    55550 | localhost |     55550 | localhost  | t            | f            | f            |
1797586929
 |    58064 |       55552 |         0 |            0 | f            | t
 datanode6    | D        |    55554 | localhost |     55554 | localhost  | t            | f            | f            |
587455710
 |    58320 |       55556 |         0 |            0 | f            | t
```


版权所有 © 华为云计算技术有限公司

```
 datanode7    | D        |     55558 | localhost |     55558 | localhost  | t             | f             | f             |
-1685037427
 |     58576 |       55560 |         0 |           0 | f           | t
 datanode8    | D        |     55562 | localhost |     55562 | localhost  | t             | f             | f             |
-993847320
 |     58832 |       55564 |         0 |           0 | f           | t
(9 rows)
```

18.2.92 PLAN_TABLE_DATA

PLAN_TABLE_DATA系统表存储了用户通过执行EXPLAIN PLAN收集到的计划信息。与**PLAN_TABLE**视图不同的是PLAN_TABLE_DATA表存储了所有session和user执行EXPLAIN PLAN收集的计划信息。

表 18-92 PLAN_TABLE 字段

名称	类型	描述
session_id	text	表示插入该条数据的会话，由服务线程启动时间戳和服务线程ID组成。受非空约束限制。
user_id	oid	用户ID，用于标识触发插入该条数据的用户。受非空约束限制。
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	计划中的节点编号。
operation	varchar2(30)	操作描述。
options	varchar2(255)	操作选项。
object_name	name	操作对应的对象名，来自于用户定义。
object_type	varchar2(30)	对象类型。
object_owner	name	对象所属schema，来自于用户定义。
projection	varchar2(4000)	操作输出的列信息。

说明

- PLAN_TABLE_DATA中包含了当前节点所有用户、所有会话的数据，仅管理员有访问权限。普通用户可以通过**PLAN_TABLE**视图查看属于自己的数据。
- 对于不活跃（已退出）的会话，其在PLAN_TABLE_DATA中的数据会在一定时间（默认5min）后被gs_clean清理。用户也可以手动执行gs_clean -C选项对表中不活跃的会话数据进行清理。
- PLAN_TABLE_DATA中的数据是用户通过执行EXPLAIN PLAN命令后由系统自动插入表中，因此禁止用户手动对数据进行插入或更新，否则会引起表中的数据混乱。需要对表中数据删除时，建议通过**PLAN_TABLE**视图。
- statement_id、object_name、object_owner和projection字段内容遵循用户定义的大小写存储，其它字段内容采用大写存储。

 版权所有 © 华为云计算技术有限公司

18.2.93 SNAPSHOT

SNAPSHOT系统表记录每次创建性能视图快照的起止时间，设置enable_wdr_snapshot为on后，该表由后台快照线程创建并维护。需要有系统管理员权限才可以访问此系统表。

须知

- 此系统表的schema是dbms_om。
- 禁止从外部修改或删除此表，否则可能引起视图快照相关功能异常。

表 18-93 dbms_om.snapshot 字段

名称	类型	描述
snapshot_id	name	快照ID（此字段为主键和分布键）。
start_ts	timestamp with time zone	快照开始时间。
end_ts	timestamp with time zone	快照结束时间。

18.2.94 TABLES_SNAP_TIMESTAMP

TABLES_SNAP_TIMESTAMP系统表记录每次对每个性能视图创建快照的起止时间，设置enable_wdr_snapshot为on后，该表由后台快照线程创建并维护。需要有系统管理员权限才可以访问此系统表。

表 18-94 dbms_om.tables_snap_timestamp 字段

名称	类型	描述
snapshot_id	name	快照ID（此字段为主键和分布键）。
db_name	text	视图所属数据库名称。
tablename	text	视图名称。
start_ts	timestamp with time zone	快照开始时间。
end_ts	timestamp with time zone	快照结束时间。

版权所有 © 华为云计算技术有限公司

须知

- 此系统表的schema是dbms_om。
- 禁止从外部修改或删除此表，否则可能引起视图快照相关功能异常。

18.2.95 性能视图快照系统表

设置enable_wdr_snapshot为on后，后台快照线程会创建并维护以“SNAP_+视图名称”方式命名的系统表，用于记录各性能视图的快照结果。需要有系统管理员权限才可以访问下列系统表。

- SNAP_**PGXC_OS_RUN_INFO**
- SNAP_**PGXC_WAIT_EVENTS**
- SNAP_**PGXC_INSTR_UNIQUE_SQL**
- SNAP_**PGXC_STAT_BAD_BLOCK**
- SNAP_**PGXC_STAT_BGWRITER**
- SNAP_**PGXC_STAT_REPLICATION**
- SNAP_**PGXC_REPLICATION_SLOTS**
- SNAP_**PGXC_SETTINGS**
- SNAP_**PGXC_INSTANCE_TIME**
- SNAP_**GLOBAL_WORKLOAD_TRANSACTION**
- SNAP_**PGXC_WORKLOAD_SQL_COUNT**
- SNAP_**PGXC_STAT_DATABASE**
- SNAP_**GLOBAL_STAT_DATABASE**
- SNAP_**PGXC_REDO_STAT**
- SNAP_**GLOBAL_REDO_STAT**
- SNAP_**PGXC_REL_IOSTAT**
- SNAP_**GLOBAL_REL_IOSTAT**
- SNAP_**PGXC_TOTAL_MEMORY_DETAIL**
- SNAP_**PGXC_NODE_STAT_RESET_TIME**
- SNAP_**PGXC_SQL_COUNT**
- SNAP_**GLOBAL_TABLE_STAT**
- SNAP_**GLOBAL_TABLE_CHANGE_STAT**
- SNAP_**GLOBAL_COLUMN_TABLE_IO_STAT**
- SNAP_**GLOBAL_ROW_TABLE_IO_STAT**

此类系统表除增加snapshot_id字段（bigint类型）外，其余的字段定义与对应视图相同，且各表的分布键均为snapshot_id。

例如，SNAP_PGXC_OS_RUN_INFO，用于存储PGXC_OS_RUN_INFO视图的快照，其字段新增了snapshot_id，其余字段含义均与PGXC_OS_RUN_INFO视图相同。

版权所有 © 华为云计算技术有限公司

须知

- 以上系统表的schema都是dbms_om。
- 禁止从外部修改或删除以上系统表，否则可能引起视图快照相关功能异常。

18.3 系统视图

18.3.1 ALL_ALL_TABLES

ALL_ALL_TABLES视图存储当前用户所能访问的表或视图。

表 18-95 ALL_ALL_TABLES 字段

名称	类型	描述
owner	name	表或视图的所有者。
table_name	name	表或视图的名称。
tablespace_name	name	表或视图所在的表空间。

18.3.2 ALL_CONSTRAINTS

ALL_CONSTRAINTS视图存储当前用户可访问的约束的信息。

表 18-96 ALL_CONSTRAINTS 字段

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名。
constraint_type	text	约束类型。 • c表示检查约束。 • f表示外键约束。 • p表示主键约束。 • u表示唯一约束。
table_name	character varying(64)	约束相关的表名。
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

版权所有 © 华为云计算技术有限公司

18.3.3 ALL_CONS_COLUMNS

ALL_CONS_COLUMNS视图存储当前用户可访问的约束字段的信息。

表 18-97 ALL_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名。
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。
position	smallint	表中列的位置。

18.3.4 ALL_COL_COMMENTS

ALL_COL_COMMENTS视图存储当前用户可访问的表或视图中字段的注释信息。

表 18-98 ALL_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。
comments	text	注释。

18.3.5 ALL_DEPENDENCIES

ALL_DEPENDENCIES视图存储了当前用户可访问的函数、高级包之间的依赖关系。

须知

因为相关信息的限制，目前DWS中，此表为空表，表内没有任何记录。

表 18-99 ALL_DEPENDENCIES 字段

名称	类型	描述
owner	character varying(30)	对象的所有者。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
name	character varying(30)	对象的名称。
type	character varying(17)	对象的类型。
referenced_owner	character varying(30)	引用对象的所有者。
referenced_name	character varying(64)	引用对象的名称。
referenced_type	character varying(17)	引用对象的类型。
referenced_link_name	character varying(128)	引用对象的链接的名称。
schemaid	numeric	当前schema的ID。
dependency_type	character varying(4)	依赖类型（REF或HARD）。

18.3.6 ALL_IND_COLUMNS

ALL_IND_COLUMNS视图存储了当前用户可访问的所有索引的字段信息。

表 18-100 ALL_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_name	name	列名。
column_position	smallint	索引中列的位置。

18.3.7 ALL_IND_EXPRESSIONS

ALL_IND_EXPRESSIONS视图存储了当前用户可访问的表达式索引的信息。

表 18-101 ALL_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

18.3.8 ALL_INDEXES

ALL_INDEXES视图存储了当前用户可访问的索引信息。

表 18-102 ALL_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引的名称是否为系统生成。
partitioned	character(3)	表示索引是否具有分区表的性质。

18.3.9 ALL_OBJECTS

ALL_OBJECTS视图记录了当前用户可访问的数据库对象。

表 18-103 ALL_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。
object_id	oid	对象的OID。
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
last_ddl_time	timestamp with time zone	对象的最后修改时间。

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

18.3.10 ALL_PROCEDURES

ALL_PROCEDURES视图存储了当前用户可访问的所有存储过程或函数信息。

表 18-104 ALL_PROCEDURES 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象名称。

18.3.11 ALL_SEQUENCES

ALL_SEQUENCES视图存储当前用户能够访问的所有序列。

表 18-105 ALL_SEQUENCES 字段

名称	类型	描述
sequence_owner	name	序列所有者。
sequence_name	name	序列的名称。
min_value	bigint	序列最小值。
max_value	bigint	序列最大值。
increment_by	bigint	序列的增量。
cycle_flag	character(1)	表示序列是否是循环序列，取值为Y或N。 • Y表示是循环序列。 • N表示不是循环序列。

版权所有 © 华为云计算技术有限公司

18.3.12 ALL_SOURCE

ALL_SOURCE视图存储当前用户可访问的存储过程或函数信息，且提供存储过程或函数定义的字段。

表 18-106 ALL_SOURCE 字段

名称	类型	描述
owner	name	对象的所有者。
name	name	对象名称。
type	name	对象的类型。
text	text	对象的定义。

18.3.13 ALL_SYNONYMS

ALL_SYNONYMS视图存储了当前用户可访问的所有同义词信息。

表 18-107 ALL_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者。
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。
table_name	text	关联对象名。

18.3.14 ALL_TAB_COLUMNS

ALL_TAB_COLUMNS视图存储了当前用户可访问的表和视图的列的描述信息。

表 18-108 ALL_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	对象创建或增加列时列的序号。
data_length	integer	列的字节长度。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2，bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

18.3.15 ALL_TAB_COMMENTS

ALL_TAB_COMMENTS视图存储当前用户可访问的所有表和视图的注释信息。

表 18-109 ALL_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

18.3.16 ALL_TABLES

ALL_TABLES视图存储当前用户可访问的所有表。

 版权所有 © 华为云计算技术有限公司

表 18-110 ALL_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	表是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

18.3.17 ALL_USERS

ALL_USERS视图存储记录数据库中所有用户，但不对用户信息进行详细的描述。

表 18-111 ALL_USERS 字段

名称	类型	描述
username	name	用户名。
user_id	oid	用户的OID。

18.3.18 ALL_VIEWS

ALL_VIEWS视图存储了当前用户可访问的所有视图描述信息。

表 18-112 ALL_VIEWS 字段

名称	类型	描述
owner	name	视图的所有者。
view_name	name	视图的名称。
text_length	integer	视图文本长度。
text	text	视图文本。

 版权所有 © 华为云计算技术有限公司

18.3.19 DBA_DATA_FILES

DBA_DATA_FILES视图存储关于数据库文件的描述。需要有系统管理员权限才可以访问。

表 18-113 DBA_DATA_FILES 字段

名称	类型	描述
tablespace_name	name	文件所属的表空间的名称。
bytes	double precision	文件的字节长度。

18.3.20 DBA_USERS

DBA_USERS视图存储关于数据库所有用户名信息。需要有系统管理员权限才可以访问。

表 18-114 DBA_USERS 字段

名称	类型	描述
username	character varying(64)	用户名。

18.3.21 DBA_COL_COMMENTS

DBA_COL_COMMENTS视图存储关于数据库中表和视图中字段的注释信息。需要有系统管理员权限才可以访问。

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。
comments	text	注释。

18.3.22 DBA_CONSTRAINTS

DBA_CONSTRAINTS视图存储关于数据库表中约束的信息。需要有系统管理员权限才可以访问。

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
constraint_type	text	约束类型。 • c表示检查约束。 • f表示外键约束。 • p表示主键约束。 • u表示唯一约束。
table_name	character varying(64)	约束相关的表名。
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

18.3.23 DBA_CONS_COLUMNS

DBA_CONS_COLUMNS视图存储关于数据库表中约束字段的信息。需要有系统管理员权限才可以访问。

名称	类型	描述
table_name	character varying(64)	约束相关的表名。
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。
position	smallint	表中列的位置。

18.3.24 DBA_IND_COLUMNS

DBA_IND_COLUMNS视图存储关于数据库中所有索引的字段信息。需要有系统管理员权限才可以访问。

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
column_name	name	列名。
column_position	smallint	索引中列的位置。

18.3.25 DBA_IND_EXPRESSIONS

DBA_IND_EXPRESSIONS视图存储了数据库中的表达式索引的信息。需要有系统管理员权限才可以访问。

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

18.3.26 DBA_IND_PARTITIONS

DBA_IND_PARTITIONS视图存储数据库中所有索引分区的信息。数据库中每个分区表的每个索引分区（如果存在的话）在DBA_IND_PARTITIONS里都会有一行记录。需要有系统管理员权限才可以访问。

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称。
schema	character varying(64)	索引分区所属分区表索引的模式。
index_name	character varying(64)	索引分区所属分区表索引的名称。
partition_name	character varying(64)	索引分区的名称。
index_partition_usable	boolean	索引分区是否可用。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collaton、字段数据类型等信息。
def_tablespace_name	name	索引分区的表空间名称。

18.3.27 DBA_INDEXES

DBA_INDEXES视图存储关于数据库下的所有索引信息。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引名称是否为系统生成。
partitioned	character(3)	表示索引是否具有分区表的性质。

18.3.28 DBA_OBJECTS

DBA_OBJECTS视图存储了数据库中所有数据库对象。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
object_id	oid	对象的OID。
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。
last_ddl_time	timestamp with time zone	对象的最后修改时间。

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

18.3.29 DBA_PART_INDEXES

DBA_PART_INDEXES视图存储数据库中所有分区表索引的信息。需要有系统管理员权限才可以访问。

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称。
schema	character varying(64)	分区表索引的模式。
index_name	character varying(64)	分区表索引的名称。
table_name	character varying(64)	分区表索引所属的分区表名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表索引的索引分区的个数。
def_tablespace_name	name	分区表索引的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

版权所有 © 华为云计算技术有限公司

18.3.30 DBA_PART_TABLES

DBA_PART_TABLES视图存储数据中所有分区表的信息。需要有系统管理员权限才可以访问。

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称。
schema	character varying(64)	分区表的模式。
table_name	character varying(64)	分区表的名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表的分区个数。
def_tablespace_name	name	分区表的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

18.3.31 DBA_PROCEDURES

DBA_PROCEDURES视图存储关于数据库下的所有存储过程或函数信息。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
object_name	character varying(64)	存储过程或函数名称。
argument_number	smallint	存储过程入参个数。

18.3.32 DBA_SEQUENCES

DBA_SEQUENCES视图存储关于数据库下的所有序列信息。需要有系统管理员权限才可以访问。

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者。
sequence_name	character varying(64)	序列的名称。

版权所有 © 华为云计算技术有限公司

18.3.33 DBA_SOURCE

DBA_SOURCE视图存储关于数据库下的所有存储过程或函数信息，且提供存储过程或函数定义的字段。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
name	character varying(64)	存储过程或函数的名称。
text	text	存储过程或函数的定义。

18.3.34 DBA_SYNONYMS

DBA_SYNONYMS视图存储关于数据库下的所有同义词信息。需要有系统管理员权限才可以访问。

表 18-115 DBA_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者。
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。
table_name	text	关联对象名。

18.3.35 DBA_TAB_COLUMNS

DBA_TAB_COLUMNS视图存储关于表和视图的字段的信息。数据库里每个表的每个字段都在DBA_TAB_COLUMNS里有一行。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	创建表或视图时列的序号。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
data_length	integer	列的字节长度。
comments	text	注释。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2， bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

应用示例

查询指定schema下的所有表字段信息：

```
SELECT * FROM DBA_TAB_COLUMNS WHERE schema = 'public';
  owner   | table_name | column_name | data_type | column_id | data_length | comments | avg_col_len |
nullable | data_precision | data_scale | char_length | schema | kind
---------+------------+-------------+-----------+-----------+-------------+----------+-------------+----------
+----------------+------------+-------------+--------+-------
 dbadmin | t_customer | id          | int4      |         1 |           4 | 主键     |           | n        |              |         0
|              0 | public | table
 dbadmin | student    | sschool     | varchar   |         4 |          10 |          |           | n        |              |
0 |             10 | public | table
 dbadmin | student    | sgender     | varchar   |         3 |          10 |          |           | n        |              |
0 |             10 | public | table
 dbadmin | student    | sname       | varchar   |         2 |          10 |          |           | n        |              |
0 |             10 | public | table
 dbadmin | student    | sid         | varchar   |         1 |          10 |          |           | n        |              |         0
|             10 | public | table
 dbadmin | t_customer | email       | varchar   |         4 |          32 | email    |           | n        |
|             0 |          32 | public | table
 dbadmin | t_customer | gender      | varchar   |         3 |          10 | 性别     |           | n        |
|             0 |          10 | public | table
 dbadmin | t_customer | cust_name   | varchar   |         2 |          32 | 名字     |           | n        |
|             0 |          32 | public | table
(8 rows)
```

18.3.36 DBA_TAB_COMMENTS

DBA_TAB_COMMENTS视图存储关于数据库下的所有表和视图的注释信息。需要有系统管理员权限才可以访问。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

18.3.37 DBA_TAB_PARTITIONS

DBA_TAB_PARTITIONS视图提供数据库中所有分区的信息。

名称	类型	描述
table_owner	character varying(64)	分区所在表的所有者。
schema	character varying(64)	分区表模式。
table_name	character varying(64)	表名。
partition_name	character varying(64)	分区的名称。
high_value	text	范围分区的上边界，或列表分区的边界值集合。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	范围分区的上边界，或列表分区的边界值集合。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collaton、字段数据类型等信息。
tablespace_name	name	分区所在表空间的名称。

应用示例

查看分区表的分区信息：

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE (wr_returned_date_sk)
(
    PARTITION p2016 VALUES LESS THAN(20161231),
```

 版权所有 © 华为云计算技术有限公司

```
    PARTITION p2017 VALUES LESS THAN(20171231),
    PARTITION p2018 VALUES LESS THAN(20181231),
    PARTITION p2019 VALUES LESS THAN(20191231),
    PARTITION p2020 VALUES LESS THAN(maxvalue)
);

SELECT * FROM dba_tab_partitions where table_name='web_returns_p1';
 table_owner | schema |    table_name    | partition_name | high_value | pretty_high_value |  tablespace_name
-------------+--------+------------------+----------------+------------+-------------------+--------------------
 dbadmin     | public | web_returns_p1 | p2016          | 20161231   | 20161231          | DEFAULT TABLESPACE
 dbadmin     | public | web_returns_p1 | p2017          | 20171231   | 20171231          | DEFAULT TABLESPACE
 dbadmin     | public | web_returns_p1 | p2018          | 20181231   | 20181231          | DEFAULT TABLESPACE
 dbadmin     | public | web_returns_p1 | p2019          | 20191231   | 20191231          | DEFAULT TABLESPACE
 dbadmin     | public | web_returns_p1 | p2020          | MAXVALUE   | MAXVALUE          | DEFAULT
TABLESPACE
(5 rows)
```

18.3.38 DBA_TABLES

DBA_TABLES视图存储关于数据库下的所有表信息。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间的名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

18.3.39 DBA_TABLESPACES

DBA_TABLESPACES视图存储有关可用的表空间的信息。需要有系统管理员权限才可以访问。

表 18-116 DBA_TABLESPACES 字段

名称	类型	描述
tablespace_name	character varying(64)	表空间的名称。

版权所有 © 华为云计算技术有限公司

18.3.40 DBA_TRIGGERS

DBA_TRIGGERS视图存储关于数据库内的触发器信息。需要有系统管理员权限才可以访问。

名称	类型	描述
trigger_name	character varying(64)	触发器名称。
table_name	character varying(64)	定义触发器的表的名称。
table_owner	character varying(64)	定义触发器的表的所有者。

18.3.41 DBA_VIEWS

DBA_VIEWS视图存储关于数据库内的视图信息。需要有系统管理员权限才可以访问。

名称	类型	描述
owner	character varying(64)	视图的所有者。
view_name	character varying(64)	视图的名称。

18.3.42 DUAL

DUAL视图是数据库根据数据字典自动创建的，它只有一个文本字段，且只有一行，用于保存表达式计算结果。任何用户都可以访问它。

表 18-117 DUAL 字段

名称	类型	描述
dummy	text	表达式计算结果。

18.3.43 GET_ALL_TSC_INFO

重新获取所有节点TSC信息。该视图仅8.2.1及以上集群版本支持。

表 18-118 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

18.3.44 GET_TSC_INFO

重新获取当前节点TSC信息。该视图仅8.2.1及以上集群版本支持。

表 18-119 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

18.3.45 GLOBAL_COLUMN_TABLE_IO_STAT

GLOBAL_COLUMN_TABLE_IO_STAT视图提供当前数据库所有列存表的IO统计数据。其字段的名称、类型和顺序与GS_COLUMN_TABLE_IO_STAT视图相同，具体的字段请参考**表18-120**。各统计字段为所有节点对应字段之和。

 版权所有 © 华为云计算技术有限公司

表 18-120 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

18.3.46 GLOBAL_REDO_STAT

GLOBAL_REDO_STAT视图显示集群中所有节点上XLOG重做过程中的统计信息总和。除avgiotim（表示所有节点平均的重做写入时间）外，其余字段名称和PV_REDO_STAT视图相同，但其余字段含义为各节点上PV_REDO_STAT视图同名字段的数值之和。

表 18-121 GLOBAL_REDO_STAT 字段

名称	类型	描述
phywrts	bigint	所有节点物理写次数之和。
phyblkwrt	bigint	所有节点物理写块数之和。
writetim	bigint	所有节点物理写消耗时间之和。
avgiotim	bigint	所有节点平均的重做写入时间。
lstiotim	bigint	所有节点上一次写入时间之和。
miniotim	bigint	所有节点最小写入时间之和。
maxiowtm	bigint	所有节点最大写入时间之和。

说明

需要有系统管理员权限才可以访问此视图。

版权所有 © 华为云计算技术有限公司

18.3.47 GLOBAL_REL_IOSTAT

GLOBAL_REL_IOSTAT视图显示集群中所有节点上磁盘读写统计信息的总和。其各字段的名称与**GS_REL_IOSTAT**视图相同，但含义为各节点上GS_REL_IOSTAT视图同名字段的数值之和。

表 18-122 GLOBAL_REL_IOSTAT 字段

名称	类型	描述
phyrds	bigint	所有节点读磁盘次数之和。
phywrts	bigint	所有节点写磁盘次数之和。
phyblkrd	bigint	所有节点读磁盘页数之和。
phyblkwrt	bigint	所有节点写磁盘页数之和。

说明

需要有系统管理员权限才可以访问此视图。

18.3.48 GLOBAL_ROW_TABLE_IO_STAT

GLOBAL_ROW_TABLE_IO_STAT视图提供当前数据库所有行存表的IO统计数据。其字段的名称、类型和顺序与GS_ROW_TABLE_IO_STAT视图相同，具体的字段请参考**表18-123**。各统计字段为所有节点对应字段之和。

表 18-123 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

版权所有 © 华为云计算技术有限公司

18.3.49 GLOBAL_STAT_DATABASE

GLOBAL_STAT_DATABASE视图显示集群中所有节点上数据库的状态和统计信息之和。

- CN上查询GLOBAL_STAT_DATABASE视图，返回的结果除stats_reset字段（当前CN上的状态重置时间）之外，其余字段表示在集群内相关节点上的数值之和。需注意，因GLOBAL_STAT_DATABASE视图中各字段的逻辑含义不同，求和的范围也有所不同。
- 在DN上查询视图GLOBAL_STAT_DATABASE，所得结果与**表18-124**相同。

表 18-124 GLOBAL_STAT_DATABASE 字段

名称	类型	描述	求和范围
datid	oid	数据库OID。	-
datname	name	数据库名。	-
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。	CN
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。	CN
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。	CN
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。	DN
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。	DN
tup_returned	bigint	当前节点上该数据库查询返回的行数。	DN
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。	DN
tup_inserted	bigint	当前节点上该数据库插入的行数。	DN
tup_updated	bigint	当前节点上该数据库更新的行数。	DN
tup_deleted	bigint	当前节点上该数据库删除的行数。	DN
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。请参见PG_STAT_DATABASE_CONFLICTS获取更多信息。	CN和DN

版权所有 © 华为云计算技术有限公司

名称	类型	描述	求和范围
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。	DN
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。	DN
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。	CN和DN
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。	DN
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。	DN
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。	-

18.3.50 GLOBAL_TABLE_CHANGE_STAT

GLOBAL_TABLE_CHANGE_STAT视图显示当前数据库中所有表格（不包括外表）变更情况。表示次数的各字段为实例启动以来的累计值。

表 18-125 GLOBAL_TABLE_CHANGE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动Vacuum的时间。
vacuum_count	bigint	手动Vacuum的次数。为各CN节点上次数之和。
last_autovacuum	timestamp with time zone	最后一次自动Vacuum的时间。
autovacuum_count	bigint	自动Vacuum的次数。为各CN节点上次数之和。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
analyze_count	bigint	分析（包括手动和自动）的次数。由于analyze会同时在所有节点上进行，该字段为所有CN节点上的最大值。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。为各CN节点上次数之和。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

18.3.51 GLOBAL_TABLE_STAT

GLOBAL_TABLE_STAT视图显示当前数据库中所有表格（不包括外表）的统计信息。除live_tuples和dead_tuples为当前实时值外，其余各统计字段为实例启动以来的累计值。

表 18-126 GLOBAL_TABLE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
distribute_mode	char	表的分布方式，与系统表pgxc_class中的pclocatortype字段含义相同。
seq_scan	bigint	顺序扫描的次数。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。
index_scan	bigint	索引扫描的次数。
index_tuple_read	bigint	索引扫描的行数。
tuple_inserted	bigint	插入的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
tuple_updated	bigint	更新的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tuple_deleted	bigint	删除的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
tuple_hot_updated	bigint	热更新的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
live_tuples	bigint	活元组数量。显示各节点最大值；如果是分布表，显示各节点之和。 只适用行存表。
dead_tuples	bigint	死元组数量。显示各节点最大值；如果是分布表，显示各节点之和。 只适用行存表。

18.3.52 GLOBAL_WORKLOAD_SQL_COUNT

GLOBAL_WORKLOAD_SQL_COUNT视图显示集群中所有Workload控制组内SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 18-127 GLOBAL_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

18.3.53 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME

GLOBAL_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

 版权所有 © 华为云计算技术有限公司

表 18-128 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

18.3.54 GLOBAL_WORKLOAD_TRANSACTION

GLOBAL_WORKLOAD_TRANSACTION视图提供集群所有CN上WORKLOAD控制组相关的事务信息的总和。需要有系统管理员权限才可以访问。该视图仅在资源实时监控功能开启，即enable_resource_track为on时有效。

表 18-129 GLOBAL_WORKLOAD_TRANSACTION 字段

名称	类型	描述
workload	name	WORKLOAD控制组名称。
commit_counter	bigint	各CN上提交次数总和。
rollback_counter	bigint	各CN上回滚次数总和。
resp_min	bigint	集群总体最小响应时间。
resp_max	bigint	集群总体最大响应时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
resp_avg	bigint	各CN上平均响应时间。
resp_total	bigint	各CN上响应时间总和。

18.3.55 GS_ALL_CONTROL_GROUP_INFO

GS_ALL_CONTROL_GROUP_INFO视图显示数据库内所有的控制组信息。

表 18-130 GS_ALL_CONTROL_GROUP_INFO 字段

名称	类型	描述
name	text	控制组的名称。
type	text	控制组的类型。
gid	bigint	控制组ID。
classgid	bigint	Workload所属Class的控制组ID。
class	text	Class控制组。
workload	text	Workload控制组。
shares	bigint	控制组分配的CPU资源配额。
limits	bigint	控制组分配的CPU资源限额。
wdlevel	bigint	Workload控制组层级。
cpucores	text	控制组使用的CPU核的信息。

18.3.56 GS_BLOCKLIST_QUERY

GS_BLOCKLIST_QUERY视图用于查询作业黑名单信息和异常信息，此视图是由系统表GS_BLOCKLIST_QUERY和GS_WLM_SESSION_INFO关联所得，同时对查询结果进行了去重筛选，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长时间。

须知

- GS_BLOCKLIST_QUERY视图的schema为pg_catalog。
- GS_BLOCKLIST_QUERY视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。
- 通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个unique_sql_id可能会对应一个或多个查询。

版权所有 © 华为云计算技术有限公司

表 18-131 GS_BLOCKLIST_QUERY 视图字段

名字	类型	引用	描述
unique_sql_id	bigint	-	基于查询解析树生成的查询ID。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。
query	text	-	执行的查询语句。

18.3.57 GS_BLOCKLIST_SQL

GS_BLOCKLIST_SQL视图用于查询作业黑名单信息和异常信息，此视图是由系统表GS_BLOCKLIST_SQL和GS_WLM_SESSION_INFO关联所得，同时对查询结果进行了去重筛选，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长时间。

该视图仅9.1.0.200及以上集群版本支持。

须知

- GS_BLOCKLIST_SQL视图存储的schema为pg_catalog。
- GS_BLOCKLIST_SQL视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。
- 通常对于DML语句，在计算sql_hash的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个sql_hash可能会对应一个或多个查询。

表 18-132 GS_BLOCKLIST_QUERY 视图字段

名字	类型	引用	描述
sql_hash	text	-	基于查询解析树生成的sql_hash。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。
query	text	-	执行的查询语句。

18.3.58 GS_CLUSTER_RESOURCE_INFO

GS_CLUSTER_RESOURCE_INFO视图显示的是所有DN资源的汇总信息。

版权所有 © 华为云计算技术有限公司

表 18-133 GS_CLUSTER_RESOURCE_INFO 字段

名称	类型	描述
min_mem_util	integer	DN最小内存使用率。
max_mem_util	integer	DN最大内存使用率。
min_cpu_util	integer	DN最小CPU使用率。
max_cpu_util	integer	DN最大CPU使用率。
min_io_util	integer	DN最小IO使用率。
max_io_util	integer	DN最大IO使用率。
used_mem_rate	integer	物理节点最大内存使用率。

18.3.59 GS_COLUMN_TABLE_IO_STAT

GS_COLUMN_TABLE_IO_STAT视图显示当前数据库中所有列存表在当前节点上的IO情况。各统计字段为实例启动以来的累计值。

表 18-134 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

18.3.60 GS_OBS_READ_TRAFFIC

GS_OBS_READ_TRAFFIC视图，统计OBS读流量和平均读带宽，统计结果按10分钟聚集。该视图仅8.2.0及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
nodename	TEXT	集群节点。
hostname	TEXT	主机节点。
traffic_mb	float8	logtime之前10分钟OBS读流量统计。
bandwidth_mb_per_s	float8	平均带宽，单位MB/s。
reqcount	bigint	logtime之前10分钟OBS读次数。
logtime	timestamp with time zone	记录统计信息时刻。

应用示例

查询当前视图，统计OBS读流量和平均读带宽，统计结果按10分钟聚集。

```
select * from gs_obs_read_traffic;
 nodename |     hostname     |    traffic_mb     | bandwidth_mb_per_s | reqcount |        logtime
----------+------------------+-------------------+--------------------+----------+------------------------
 dn_1     | rhel_10_90_45_56 | 101.959338188171 |    5.14830159670447 |       23 | 2022-11-26 09:50:00+08
(1 row)
```

18.3.61 GS_OBS_WRITE_TRAFFIC

GS_OBS_WRITE_TRAFFIC视图，统计OBS写流量和平均写带宽，统计结果按10分钟聚集。该视图仅8.2.0及以上集群版本支持。

名称	类型	描述
nodename	TEXT	集群节点。
hostname	TEXT	主机节点。
traffic_mb	float8	logtime之前10分钟OBS写流量统计。
bandwidth_mb_per_s	float8	平均带宽，单位MB/s。
reqcount	bigint	logtime之前10分钟OBS写次数。
logtime	timestamp with time zone	记录统计信息时刻。

应用示例

查询当前视图，统计OBS写流量和平均写带宽，统计结果按10分钟聚集。

```
select * from gs_obs_write_traffic;
   nodename   |     hostname     |      traffic_mb      | bandwidth_mb_per_s  | reqcount |        logtime
--------------+------------------+----------------------+---------------------+----------+------------------------
 dn_1         | rhel_10_90_45_56 |  .000738143920898438 | .000289970820362525 |       12 | 2022-10-24
16:10:00+08
```


版权所有 © 华为云计算技术有限公司

```
 dn_1        | rhel_10_90_45_56 |  .000354766845703125 | .000386063466694153 |       7 | 2022-10-24
18:50:00+08
 dn_1        | rhel_10_90_45_56 | 9.34600830078125e-05 | .000143659648687162 |       2 | 2022-11-07
09:20:00+08
 dn_1        | rhel_10_90_45_56 | 4.10079956054688e-05 | .000186667253592502 |       1 | 2022-11-07
09:30:00+08
 dn_1        | rhel_10_90_45_56 |     2048.17834663391 |    27.2766632219637 |       2 | 2022-11-22
16:10:00+08
 dn_1        | rhel_10_90_45_56 |     3747.23722648621 |    28.0842938534546 |       4 | 2022-11-22
16:20:00+08
(6 row)
```

18.3.62 GS_INSTR_UNIQUE_SQL

Unique SQL 定义

数据库将接收到的每个SQL的文本字符串，都进行解析并生成内部解析树，遍历解析树并忽略其中的常数值，以一定的算法计算出来一个整数值作为Unique SQL ID，用来唯一标识这一类SQL，Unique SQL ID相同的一类SQL就叫做Unique SQL。

示例

假如，用户先后输入SQL:

```
select * from t1 where id = 1;
select * from t1 where id = 2;
```

那么，这两条SQL的统计信息会汇聚到同一个Unique SQL上：

```
select * from t1 where id = ?;
```

GS_INSTR_UNIQUE_SQL 视图

GS_INSTR_UNIQUE_SQL视图显示当前节点收集的Unique SQL的执行信息，主要包括以下内容：

- Unique SQL ID以及归一化后的SQL文本字符串，归一化后的SQL文本如**示例**中所示。通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。
- 执行次数（成功执行的次数），响应时间（数据库内部的SQL执行时间，包括最大、最小和总时间）。
- Cache/IO信息，包含block的物理读、逻辑读次数，仅统计执行成功的SQL在各DN节点上的相关信息。该统计值与查询执行当时所处理的数据量、所使用的内存、是否多次执行、内存管理策略、是否有其他并发查询等因素相关，反映整个查询执行过程中的buffer块物理读和逻辑读次数，不同时间执行可能统计值不同。
- 行活动，包含SELECT语句的结果集返回行数、更新行、插入行、删除行、顺序扫描行、随机扫描行等信息。除结果集返回行数与该SELECT语句的结果集行数一致、且仅在CN上记录外，其他行活动信息均在DN上记录，且统计数值反应的是整个查询执行过程中的行活动，包括对相关系统表、元数据表、数据表等做必要的扫描和修改，与对应数据量以及相关参数设置相关，即统计数值将会大于等于对实际数据表的扫描和修改。
- 时间分布，包含：DB_TIME/CPU_TIME/EXECUTION_TIME/PARSE_TIME/PLAN_TIME/REWRITE_TIME/PL_EXECUTION_TIME/PL_COMPILATION_TIME/NET_SEND_TIME/DATA_IO_TIME，相关定义见**表1**。该信息在CN和DN节点均有统计，视图查询时将汇总展示。

版权所有 © 华为云计算技术有限公司

- 软硬解析次数，包含软解析（缓存计划）、硬解析（生成计划）的次数，即如果本次执行的是之前缓存的计划，软解析次数+1，如果本次执行的计划是重新生成的，则硬解析次数+1。该次数在CN和DN节点上都会统计，视图查询时将汇总展示。

Unique SQL收集功能存在以下约束：

- 只有执行成功的SQL才会显示其详细的统计信息，否则可能只记录query、node、user等信息。
- 如果开启Unique SQL收集功能，CN节点将对所有接收到的查询进行统计收集，包括工具和用户的查询等。
- 若一条SQL语句内部包含执行多条SQL语句、类似存储过程执行等场景，仅会对最外层SQL生成一条Unique SQL，所有子SQL的统计信息都会汇总到该Unique SQL记录上。
- Unique SQL的响应时间统计中不完全包含NET_SEND_TIME阶段的时间，所以EXECUTION_TIME和elapse_time等时间之间不存在大小比较关系。
- 对于类似begin;...;commit;等形式的事务块，当前不支持统计子句的解析时间（parse_time）。

普通用户访问GS_INSTR_UNIQUE_SQL视图，只能看到该用户相关的Unique SQL信息，管理员用户可以看到当前节点所有的Unique SQL信息。CN和DN上均可查询GS_INSTR_UNIQUE_SQL视图，DN上显示的是本节点内的Unique SQL统计信息，CN上显示的是本节点Unique SQL完整统计信息，即该CN节点会收集其他CN和DN上对应该CN的Unique SQL的执行信息，进行汇总展示。通过查询GS_INSTR_UNIQUE_SQL视图，能够定位由于消耗不同资源导致的Top SQL，为集群性能调优和维护提供依据。

后台线程每隔1小时检查一次所有的Unique SQL，将last_time在instr_unique_sql_timeout小时之前的Unique SQL删除。

表 18-135 GS_INSTR_UNIQUE_SQL 字段

名称	类型	描述
node_name	name	接收SQL的CN节点名称。
node_id	integer	节点ID，等同于pgxc_node表中node_id。
user_name	name	用户名称。
user_id	oid	用户ID。
unique_sql_id	bigint	归一化的UNIQUE SQL ID。
query	text	归一化的SQL文本。
n_calls	bigint	成功执行次数。
min_elapse_time	bigint	SQL在数据库内的最小运行时间（单位：微秒）。
max_elapse_time	bigint	SQL在数据库内的最大运行时间（单位：微秒）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_elapse_time	bigint	SQL在数据库内的总运行时间（单位：微秒）。
n_returned_rows	bigint	行活动-SELECT语句返回的结果集行数。
n_tuples_fetched	bigint	行活动-随机扫描行（列存表/外表不统计）。
n_tuples_returned	bigint	行活动-顺序扫描行（列存表/外表不统计）。
n_tuples_inserted	bigint	行活动-插入行数。
n_tuples_updated	bigint	行活动-更新行数。
n_tuples_deleted	bigint	行活动-删除行数。
n_blocks_fetched	bigint	buffer的块访问次数，即物理读/IO。
n_blocks_hit	bigint	buffer的块命中次数，即逻辑读/Cache。
n_soft_parse	bigint	软解析次数（缓存计划）。
n_hard_parse	bigint	硬解析次数（生成计划）。
db_time	bigint	有效的DB执行时间，包含等待时间、网络发送时间等，若查询执行涉及到多线程，DB_TIME是多个线程的DB_TIME之和（单位：微秒）。
cpu_time	bigint	CPU的执行时间，不包含sleep时间（单位：微秒）。
execution_time	bigint	查询执行器内的SQL执行时间，DDL语句、以及某些不经过执行器执行的语句（例如Copy语句）不计数（单位：微秒）。
parse_time	bigint	SQL解析时间（单位：微秒）。
plan_time	bigint	SQL生成计划时间（单位：微秒）。
rewrite_time	bigint	SQL重写时间（单位：微秒）。
pl_execution_time	bigint	plpgsql过程化语言函数上的执行时间（单位：微秒）。
pl_compilation_time	bigint	plpgsql过程化语言函数上的编译时间（单位：微秒）。
net_send_time	bigint	网络时间，包含CN向客户端发送数据、DN向CN发送数据等时间（单位：微秒）。
data_io_time	bigint	IO时间，文件IO耗时（单位：微秒）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
first_time	timestamp with time zone	该SQL第一次执行的时间。
last_time	timestamp with time zone	该SQL上一次执行的时间。

18.3.63 GS_NODE_STAT_RESET_TIME

GS_NODE_STAT_RESET_TIME视图提供当前节点的统计信息重置时间，返回带时区的时间戳。

详细含义参考《数据仓库服务SQL语法参考》中“函数和操作符>系统管理函数>其他函数”章节的get_node_stat_reset_time()函数。

说明

实例正常运行过程中，内存中的各类统计数值会逐渐累加，以下情况会导致内存中的统计数值被重置为0：

- 实例重启或集群发生了切换；
- 数据库Database被删除（drop）；
- 用户执行了重置操作，如执行pgstat_recv_resetcounter函数会将数据库中的统计计数器清零，执行reset_instr_unique_sql函数会将Unique SQL数据清零。

如果发生了以上事件，DWS会记录统计信息被重置的时间，可通过get_node_stat_reset_time函数查询。

18.3.64 GS_OBS_LATENCY

GS_OBS_LATENCY记录logtime之前10分钟内OBS的平均延迟信息，延迟数据是根据相关OBS的操作进行估算的结果。该视图仅8.2.0及以上集群版本支持。

表 18-136 GS_OBS_LATENCY 字段

名称	类型	描述
nodename	text	集群节点。
hostname	text	主机节点。
latency_ms	double precision	logtime之前10分钟内OBS的平均延迟，单位ms。
reqcount	bigint	logtime之前10分钟内OBS的请求次数。
logtime	timestamp with time zone	记录延迟信息的时刻。

版权所有 © 华为云计算技术有限公司

18.3.65 GS_QUERY_MONITOR

GS_QUERY_MONITOR视图显示正在执行的查询运行/排队信息及资源使用信息，只显示排队和正在运行的作业，仅支持在CN上查询使用，仅显示主语句监控信息。该视图仅8.2.1.100及以上集群版本支持。

表 18-137 GS_QUERY_MONITOR 视图字段

名称	类型	描述
usename	name	执行该查询的用户名称。
nodename	name	执行该查询的CN名称。
nodegroup	name	执行该查询的集群名称，默认集群显示“installation”。
rpname	name	该查询关联的资源池名称。
priority	name	查询当前优先级，包含Rush/High/Medium/Low四个优先级。
xact_start	timestamp	查询所属事务的开启时间。
query_start	timestamp	查询执行开始时间。
block_time	bigint	作业累积已排队时间，存储过程、多语句可能多次排队。单位：秒。
duration	bigint	作业已运行时间，不包含排队时间。单位：秒。
query_band	text	显示作业标识，可通过GUC参数query_band设置，默认为空。
attribute	text	作业属性： ● Simple：简单作业； ● Complicated：复杂作业。 作业进入资源池管控前无意义，只有进入或已完成资源池管控该字段才有意义。
lane	text	作业排队/执行所处的资源池车道： ● fast：快车道； ● slow：慢车道。 作业进入资源池管控前无意义，只有进入或已完成资源池管控该字段才有意义。
status	text	作业当前状态，包含pending/running两种可能状态。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
queue	text	作业排队信息： ● None：作业正在运行； ● Global：作业在CN全局并发队列排队； ● Respool：作业在资源池队列排队； ● CCN：作业在CCN排队。
used_mem	integer	作业在所有DN上内存峰值的最大值，单位：MB。
estimate_mem	integer	作业估算内存，单位：MB。
used_cpu	double precision	作业开始运行至今，占用CPU核数的平均值。
read_speed	integer	作业当前在所有DN上逻辑IO读速率的平均值，单位：KB/s。
write_speed	integer	作业当前在所有DN上逻辑IO写速率的平均值，单位：KB/s。
send_speed	integer	作业开始运行至今，在所有DN上网络发送速率的平均值，单位：KB/s。
recv_speed	integer	作业开始运行至今，在所有DN上网络接收速率的平均值，单位：KB/s。
dn_count	bigint	执行该作业的DN数量。
stream_count	bigint	作业在所有DN上stream线程的数量之和。
pid	bigint	后端线程ID。
lwtid	integer	后台线程的轻量级线程号。
query_id	bigint	查询ID。
unique_sql_id	bigint	归一化的Unique SQL ID。
query	text	正在执行的查询。

18.3.66 GS_QUERY_RESOURCE_INFO

GS_QUERY_RESOURCE_INFO视图显示当前DN节点所有正在运行作业的资源信息。该参数仅9.1.0及以上集群版本支持。

说明

该视图只能在DN节点上查询。仅适用于运维操作定位问题，不建议用户使用。

版权所有 © 华为云计算技术有限公司

表 18-138 GS_QUERY_RESOURCE_INFO

名称	类型	描述
node_name	text	实例名称，只包含DN。
user_id	oid	用户ID。
queryid	bigint	语句执行使用的内部query id。
used_mem	int	语句在当前DN使用的内存大小，单位MB。
cpu_time	bigint	语句在当前DN上的CPU时间，单位ms。
used_cpu	double	语句在当前DN使用的CPU个数。
spill_size	bigint	若发生下盘，语句在当前DN下盘数据量，默认为0。单位MB。
read_bytes	bigint	语句在当前DN使用的逻辑读字节数，单位KB。
write_bytes	bigint	语句在当前DN使用的逻辑写字节数，单位KB。
read_count	bigint	语句在当前DN使用的逻辑读次数。
write_count	bigint	语句在当前DN使用的逻辑写次数。
read_speed	int	语句在当前DN使用的逻辑读速率，单位KB/s。
write_speed	int	语句在当前DN使用的逻辑写速率，单位KB/s。
curr_iops	int	语句在当前DN上的每秒IO数值（列存单位是次/s，行存单位是万次/s）。
send_pkg	bigint	语句在当前DN上的通信包发送总量，单位packages。
recv_pkg	bigint	语句在当前DN上的通信包接收总量，单位packages。
send_bytes	bigint	语句在当前DN上的通信流发送数据总量，单位Byte。
recv_bytes	bigint	语句在当前DN上的通信流接收数据总量，单位Byte。
send_speed	int	语句在当前DN使用的网络发送速率，单位KB/s。
recv_speed	int	语句在当前DN使用的网络接收速率，单位KB/s。

18.3.67 GS_REL_IOSTAT

GS_REL_IOSTAT视图提供当前节点上磁盘读写的统计信息。当前版本中，每次读/写磁盘只读/写一页，所以读写次数与页数相等。

 版权所有 © 华为云计算技术有限公司

表 18-139 GS_REL_IOSTAT 字段

名称	类型	描述
phyrds	bigint	读磁盘次数。
phywrts	bigint	写磁盘次数。
phyblkrd	bigint	读磁盘页数。
phyblkwrt	bigint	写磁盘页数。

18.3.68 GS_RESPOOL_RUNTIME_INFO

GS_RESPOOL_RUNTIME_INFO视图显示当前CN所有资源池作业运行信息。

表 18-140 GS_RESPOOL_RUNTIME_INFO 字段

名称	类型	描述
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数。
fast_run	int	资源池快车道运行作业数。
fast_wait	int	资源池快车道排队作业数。
slow_run	int	资源池慢车道运行作业数。
slow_wait	int	资源池慢车道排队作业数。

18.3.69 GS_RESPOOL_RESOURCE_INFO

GS_RESPOOL_RESOURCE_INFO视图显示CN上所有资源池作业运行信息以及当前实例（CN/DN）所有资源池资源使用信息。

说明

DN上仅显示当前DN所属逻辑集群的资源池监控信息。

表 18-141 GS_RESPOOL_RESOURCE_INFO 字段

名称	类型	描述
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，只在CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5s监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为GaussDB可用CPU，CPU限额管控情况下为关联控制组CPU可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池当前使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5s监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_kbytes	bigint	资源池5s监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5s监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5s监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5s监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5s监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5s监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5s监控周期内网络接收速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。

18.3.70 GS_RESPOOL_MONITOR

GS_RESPOOL_MONITOR视图显示所有资源池作业运行信息及资源使用信息，仅支持CN上查询。该视图仅8.2.1.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 18-142 GS_RESPOOL_MONITOR 字段

名称	类型	描述
rpname	name	资源池名称。
nodegroup	name	资源池所属逻辑集群的名称，默认集群显示“installation”。
cn_count	bigint	集群包含的CN数量，多CN环境下用于判断单CN管控结果是否合理。
short_acc	boolean	资源池是否开启短查询加速。
session_count	bigint	关联该资源池的会话数量，即关联该资源池的用户发起的会话数量，包含IDLE和ACTIVE会话。
active_count	bigint	关联该资源池的活跃会话数量，即正在执行查询的会话数量。
global_wait	bigint	关联该资源池的所有作业中，因单CN上并发超max_active_statements引起排队的作业数。
fast_run	bigint	关联该资源池的所有作业中，正在资源池快车道运行的作业数。
fast_wait	bigint	关联该资源池的所有作业中，在资源池快车道排队的作业数。
fast_limit	bigint	资源池快车道作业并发上限。
slow_run	bigint	关联该资源池的所有作业中，正在资源池慢车道运行的作业数。
slow_wait	bigint	关联该资源池的所有作业中，在资源池慢车道排队的作业数。
slow_limit	bigint	资源池慢车道作业并发上限。
used_mem	text	资源池在所有DN上已用内存的平均值，显示结果已使用pg_size_pretty格式化。
estimate_mem	text	资源池正在运行的作业估算内存之和，显示结果已使用pg_size_pretty格式化。
mem_limit	text	资源池可用内存的上限，显示结果已使用pg_size_pretty格式化。
query_mem_limit	name	资源池内单个查询可以使用的内存上限，主要用于限制查询估算内存，防止估算内存过大导致异常排队；通过估算内存限制查询实际使用内存，显示结果已使用pg_size_pretty格式化。
used_cpu	double precision	资源池在所有DN上占用CPU核数的平均值；CPU隔离以节点和资源池为单位，单个节点上包含多个DN时，资源池在单节点上占用的CPU核数需要乘以DN数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
cpu_limit	double precision	资源池在所有节点上可用CPU上限的平均值，CPU配额管控情况下为GaussDB全部可用CPU核数，CPU限额管控情况下为关联控制组的可用CPU核数。
read_speed	text	资源池在所有DN上逻辑IO读速率的平均值，显示结果已使用pg_size_pretty格式化。
write_speed	text	资源池在所有DN上逻辑IO写速率的平均值，显示结果已使用pg_size_pretty格式化。
send_speed	text	资源池在所有DN上网络发送速率的平均值，显示结果已使用pg_size_pretty格式化。
recv_speed	text	资源池在所有DN上网络接收速率的平均值，显示结果已使用pg_size_pretty格式化。

18.3.71 GS_ROW_TABLE_IO_STAT

GS_ROW_TABLE_IO_STAT视图显示当前数据库中所有行存表在当前节点上的IO情况。各统计字段为实例启动以来的累计值。

表 18-143 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

18.3.72 GS_SESSION_CPU_STATISTICS

GS_SESSION_CPU_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理CPU使用的信息。

版权所有 © 华为云计算技术有限公司

表 18-144 GS_SESSION_CPU_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
usename	name	登录到该后端的用户名。
pid	bigint	后端线程ID。
start_time	timestamp with time zone	语句执行的开始时间。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位为ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位为ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位为ms。
query	text	正在执行的语句。
node_group	text	语句所属用户对应的逻辑集群。

18.3.73 GS_SESSION_MEMORY_STATISTICS

GS_SESSION_MEMORY_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理内存使用的信息。

表 18-145 GS_SESSION_MEMORY_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
usename	name	登录到该后端的用户名。
pid	bigint	后端线程ID。
start_time	timestamp with time zone	语句执行的开始时间。
min_peak_memory	integer	语句在所有DN上的最小内存峰值大小，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值大小，单位MB。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
query	text	正在执行的语句。
node_group	text	语句所属用户对应的逻辑集群。

18.3.74 GS_SQL_COUNT

GS_SQL_COUNT视图显示数据库当前节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）统计信息，包括执行次数和响应时间（除MERGE INTO语句外，统计其他四类语句的最大、最小、平均和总响应时间，单位为微秒），以及DDL、DML、DCL类型语句的执行次数。

GS_SQL_COUNT视图中对DDL、DML、DCL类型语句分类与SQL语法中略有不同，具体如下：

- CREATE/ALTER/DROP USER，CREATE/ALTER/DROP ROLE等用户相关语句属于DCL类型。
- BEGIN/COMMIT/SET CONSTRAINTS/ROLLBACK/SAVEPOINT/START等事务相关语句属于DCL类型。
- ALTER SYSTEM KILL SESSION等价于SELECT pg_terminate_backend()语句，属于DML类型。

其余语句的分类与SQL语法中定义类似。

普通用户查询GS_SQL_COUNT视图仅能看到该用户当前节点的统计信息。管理员权限用户查询GS_SQL_COUNT视图则能看到所有用户当前节点的统计信息；当集群或该节点重启时，计数会清零，并重新开始计数 。计数以节点收到的查询数为准，包括集群内部进行的查询；GS_SQL_COUNT视图涉及的统计信息只在CN上统计，且不统计从其他CN发送过来的SQL。在DN上查询该视图返回结果为空。

表 18-146 GS_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称。
user_name	name	用户名。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
mergeinto_count	bigint	MERGE INTO数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_select_elapse	bigint	SELECT总响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。

18.3.75 GS_STAT_DB_CU

GS_STAT_DB_CU视图查询集群各个节点，每个数据库的CU命中情况。可以通过gs_stat_reset()进行清零。

版权所有 © 华为云计算技术有限公司

表 18-147 GS_STAT_DB_CU 字段

名称	类型	描述
node_name1	text	节点名称。
db_name	text	数据库名称。
mem_hit	bigint	内存命中次数。
hdd_sync_read	bigint	硬盘同步读次数。
hdd_asyn_read	bigint	硬盘异步读次数。

18.3.76 GS_STAT_SESSION_CU

GS_STAT_SESSION_CU视图查询当前集群各个节点，当前运行session的CU命中情况。session退出相应的统计数据会清零。集群重启后，统计数据也会清零。

表 18-148 GS_STAT_SESSION_CU 字段

名称	类型	描述
node_name1	text	节点名称。
mem_hit	integer	内存命中次数。
hdd_sync_read	integer	硬盘同步读次数。
hdd_asyn_read	integer	硬盘异步读次数。

18.3.77 GS_TABLE_CHANGE_STAT

GS_TABLE_CHANGE_STAT视图显示当前数据库中所有表格（不包括外表）在当前节点上的变更情况。表示次数的各字段为实例启动以来的累计值。

表 18-149 GS_TABLE_CHANGE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动vacuum的时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
vacuum_count	bigint	手动Vacuum的次数。
last_autovacuum	timestamp with time zone	最后一次自动vacuum的时间。
autovacuum_count	bigint	自动vacuum的次数。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
analyze_count	bigint	分析（包括手动和自动）的次数。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

18.3.78 GS_TABLE_STAT

GS_TABLE_STAT视图显示当前数据库中所有表格（不包括外表）在当前节点上的统计信息。除live_tuples和dead_tuples为当前实时值外，其余各统计字段为实例启动以来的累计值。

表 18-150 GS_TABLE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。
index_scan	bigint	索引扫描的次数。
index_tuple_read	bigint	索引扫描的行数。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

18.3.79 GS_TOTAL_NODEGROUP_MEMORY_DETAIL

GS_TOTAL_NODEGROUP_MEMORY_DETAIL视图统计当前数据库逻辑集群使用内存的信息，单位为MB。

表 18-151 GS_TOTAL_NODEGROUP_MEMORY_DETAIL 字段

名称	类型	描述
ngname	text	逻辑集群名称。
memorytype	text	内存类型，包括以下几种： • ng_total_memory：该逻辑集群的总内存大小。 • ng_used_memory：该逻辑集群的实际使用内存大小。 • ng_estimate_memory：该逻辑集群的估算使用内存大小。 • ng_foreignrp_memsize：该逻辑集群的外部资源池的总内存大小。 • ng_foreignrp_usedsize：该逻辑集群的外部资源池实际使用内存大小。 • ng_foreignrp_peaksize：该逻辑集群的外部资源池使用内存的峰值。 • ng_foreignrp_mempct：该逻辑集群的外部资源池占该逻辑集群总内存大小的百分比。 • ng_foreignrp_estmsize：该逻辑集群的外部资源池估算使用内存大小。
memorymbytes	integer	内存类型分配内存的大小。

18.3.80 GS_USER_MONITOR

GS_USER_MONITOR视图显示所有用户作业运行信息及资源使用信息，仅支持CN上查询。该视图仅8.2.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-152 GS_USER_MONITOR 字段

名称	类型	描述
usename	name	用户名称。
rpname	name	用户关联的资源池名称。
nodegroup	name	资源池所属逻辑集群的名称，默认集群显示“installation”。
session_count	bigint	该用户发起的会话数量，包含IDLE和ACTIVE会话。
active_count	bigint	该用户发起的活跃会话数量，即正在执行查询的会话数量。
global_wait	bigint	该用户执行的所有作业中，因单CN上并发超max_active_statements引起排队的作业数。
fast_run	bigint	该用户执行的所有作业中，正在资源池快车道运行的作业数。
fast_wait	bigint	该用户执行的所有作业中，在资源池快车道排队的作业数。
slow_run	bigint	该用户执行的所有作业中，正在资源池慢车道运行的作业数。
slow_wait	bigint	该用户执行的所有作业中，在资源池慢车道排队的作业数。
used_mem	bigint	用户在所有DN上已用内存的平均值，单位：MB。
estimate_mem	bigint	用户正在运行的作业估算内存之和，单位：MB。
used_cpu	double precision	用户在所有DN上使用CPU核数的平均值，单个节点上包含多个DN时，用户在单节点上占用的CPU核数需要乘以DN数。
read_speed	bigint	用户在所有DN上逻辑IO读速率的平均值，单位：KB/s。
write_speed	bigint	用户在所有DN上逻辑IO写速率的平均值，单位：KB/s。
send_speed	bigint	用户在所有DN上网络发送速率的平均值，单位：KB/s。
recv_speed	bigint	用户在所有DN上网络接收速率的平均值，单位：KB/s。
used_space	bigint	用户永久表已使用的空间大小，单位：KB。
space_limit	bigint	用户永久表可使用的空间大小上限，单位：KB。-1为不限制。
used_temp_space	bigint	用户临时表已使用的空间大小，单位：KB。
temp_space_limit	bigint	用户临时表可使用的空间大小上限，单位：KB。-1为不限制。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_spill_space	bigint	用户中间结果集落盘已使用的空间大小，单位：KB。
spill_space_limit	bigint	用户中间结果集落盘可使用的空间大小上限，单位：KB。-1为不限制。

18.3.81 GS_USER_TRANSACTION

GS_USER_TRANSACTION视图提供查询单CN上用户相关的事务信息。数据库记录每个用户事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 18-153 GS_USER_TRANSACTION 字段

名称	类型	描述
usename	name	用户名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

18.3.82 GS_VIEW_DEPENDENCY

GS_VIEW_DEPENDENCY视图提供查询当前用户可见的所有视图的直接依赖关系。

表 18-154 GS_VIEW_DEPENDENCY 字段

名称	类型	描述
objschema	name	视图空间名称。
objname	name	视图名称。
refobjschema	name	依赖对象的空间名称。
refobjname	name	依赖对象的名称。
relobjkind	char	依赖对象的类型。 • r表示依赖对象为表。 • v表示依赖对象为视图。

版权所有 © 华为云计算技术有限公司

18.3.83 GS_VIEW_DEPENDENCY_PATH

GS_VIEW_DEPENDENCY_PATH视图提供查询当前用户可见的所有视图的直接依赖关系。如果该视图依赖的基础表存在且各级视图依赖关系正常，通过该视图可以查询自基础表开始的各级视图的依赖关系。

表 18-155 GS_VIEW_DEPENDENCY_PATH 字段

名称	类型	描述
objschema	name	视图空间名称。
objname	name	视图名称。
refobjschema	name	依赖对象的空间名称。
refobjname	name	依赖对象的名称。
path	text	依赖路径。

18.3.84 GS_VIEW_INVALID

GS_VIEW_INVALID视图提供查询当前用户可见的所有不可用的视图。

如果该视图依赖的基础表或函数或同义词存在异常，该视图validtype列显示为“invalid”；如果该视图属于在升级过程中因依赖的系统对象变化，该视图validtype列显示为“invalidInUpgrade”。

表 18-156 GS_VIEW_INVALID 字段

名称	类型	描述
oid	oid	视图OID。
schemaname	name	视图空间名称。
viewname	name	视图名称。
viewowner	name	视图的所有者。
definition	text	视图定义。
validtype	text	视图有效性标识。
last_invalid_time	timestamp with time zone	视图无效的时间。该字段仅9.1.0.200及以上集群版本支持。

18.3.85 GS_WAIT_EVENTS

GS_WAIT_EVENTS视图显示当前节点上各类等待状态和事件的统计信息。

 版权所有 © 华为云计算技术有限公司

仅在GUC参数enable_track_wait_event为on的情况下，视图中各统计字段的数值才会被累加。若在运行过程中将enable_track_wait_event设置为off，则不再累加统计数值，但已有数值不受影响。enable_track_wait_event为off，查询该视图返回0行。

表 18-157 GS_WAIT_EVENTS 字段

名称	类型	描述
nodename	name	节点名称。
type	text	事件的类型，包括STATUS，LOCK_EVENT，LWLOCK_EVENT和IO_EVENT四种类型。
event	text	事件名称，可参考PG_THREAD_WAIT_STATUS视图。
wait	bigint	事件发生次数。该字段及以下字段均为进程运行中的累计值。
failed_wait	bigint	等待失败次数。当前版本中只有LOCK和LWLOCK等锁超时或失败才会使用该字段。
total_wait_time	bigint	该事件总持续时间。
avg_wait_time	bigint	该事件平均持续时间。
max_wait_time	bigint	该事件最大等待时间。
min_wait_time	bigint	该事件最小等待时间。

当前版本中，对于type='LOCK_EVENT'，'LWLOCK_EVENT'和'IO_EVENT'的事件，GS_WAIT_EVENTS视图显示范围与PG_THREAD_WAIT_STATUS视图对应事件相同。

对于type='STATUS'的事件GS_WAIT_EVENTS包含的等待状态列如下，其详细含义参见PG_THREAD_WAIT_STATUS视图。

- acquire lwlock
- acquire lock
- wait io
- wait pooler get conn
- wait pooler abort conn
- wait pooler clean conn
- wait transaction sync
- wait wal sync
- wait data sync
- wait producer ready
- create index
- analyze

 版权所有 © 华为云计算技术有限公司

- vacuum
- vacuum full
- gtm connect
- gtm begin trans
- gtm commit trans
- gtm rollback trans
- gtm create sequence
- gtm alter sequence
- gtm get sequence val
- gtm set sequence val
- gtm drop sequence
- gtm rename sequence

18.3.86 GS_WLM_OPERATOR_INFO

本视图显示当前CN上已经完成执行的query语句中的算子执行信息，此系统视图信息来源于系统表dbms_om.**gs_wlm_operator_info**。

须知

- GS_WLM_OPERATOR_INFO视图的schema为pg_catalog。
- GS_WLM_OPERATOR_INFO视图表仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

表 18-158 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

18.3.87 GS_WLM_OPERATOR_HISTORY

GS_WLM_OPERATOR_HISTORY视图显示的是当前用户在当前CN上执行作业结束后的算子的相关记录。

版权所有 © 华为云计算技术有限公司

此视图用于从DWS中查询数据，数据库中的数据会定时被清理。当GUC参数**enable_resource_record**为on时，视图中的记录每隔3分钟被转储到系统表**GS_WLM_OPERATOR_INFO**中一次，同时视图中的记录被删除；当GUC参数enable_resource_record为off时，记录在视图中的存留时间达到超期时间后会被删除。记录的数据同**表18-159**。

表 18-159 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

18.3.88 GS_WLM_OPERATOR_STATISTICS

GS_WLM_OPERATOR_STATISTICS视图显示当前用户正在执行的作业的算子相关信息。

表 18-160 GS_WLM_OPERATOR_STATISTICS 字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称，算子名称长度最大为127字符（不包含空格等格式化字符）。
start_time	timestamp with time zone	该算子第一次开始执行的时间。
duration	bigint	该算子从开始执行直到结束时总的执行时间(ms)。
status	text	当前算子的执行状态，包括waiting、running和finished。
query_dop	integer	当前算子执行时的并行度。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
estimated_rows	bigint	优化器估算的行数信息，若返回的预估行数超过int64_max时，显示为int64_max。
tuple_processed	bigint	当前算子在所有DN上的返回的元素个数总和，若返回的预估行数超过int64_max时，显示为int64_max。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小逻辑下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大逻辑下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均逻辑下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
parent_id	integer	该算子节点的父节点id。
exec_count	integer	该算子节点在所有DN上的最大执行次数。
progress	text	该算子的进度信息，第一个算子展示的是作业整体的进度。其他算子展示的是当前算子进度信息。
min_net_size	bigint	该算子在所有DN上的最小网络通信数据量(KB)，主要涉及网络算子。
max_net_size	bigint	该算子在所有DN上的最大网络通信数据量(KB)，主要涉及网络算子。
total_net_size	bigint	该算子在所有DN上的总网络通信数据量(KB)，主要涉及网络算子。
min_read_bytes	bigint	该算子在所有DN上的最小磁盘读取数据量(KB)。
max_read_bytes	bigint	该算子在所有DN上的最大磁盘读取数据量(KB)。
total_read_bytes	bigint	该算子在所有DN上的总磁盘读取数据量(KB)。
min_write_bytes	bigint	该算子在所有DN上的最小磁盘写入数据量(KB)。
max_write_bytes	bigint	该算子在所有DN上的最大磁盘写入数据量(KB)。
total_write_bytes	bigint	该算子在所有DN上的总磁盘写入数据量(KB)。

18.3.89 GS_WLM_SESSION_INFO

GS_WLM_SESSION_INFO视图显示当前CN上已经完成执行的query语句的执行信息，此系统视图信息来源于系统表dbms_om.**gs_wlm_session_info**。

须知

- GS_WLM_SESSION_INFO视图的schema为pg_catalog。
- GS_WLM_SESSION_INFO视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

GS_WLM_SESSION_INFO视图的字段同**表18-162**相同，具体字段内容如下：

版权所有 © 华为云计算技术有限公司

表 18-161 GS_WLM_SESSION_HISTORY 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。
outline_name	text	该语句计划对应的outline的名字。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。

18.3.90 GS_WLM_SESSION_HISTORY

GS_WLM_SESSION_HISTORY视图显示当前用户在当前CN上执行作业结束后的负载管理记录。此视图用于从DWS中查询数据，仅当GUC参数**enable_resource_track**为on时，视图会查询GS_WLM_SESSION_INFO表中3分钟内的数据进行返回。

版权所有 © 华为云计算技术有限公司

须知

GS_WLM_SESSION_HISTORY视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

表 18-162 GS_WLM_SESSION_HISTORY 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。
outline_name	text	该语句计划对应的outline的名字。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。

18.3.91 GS_WLM_SESSION_STATISTICS

GS_WLM_SESSION_STATISTICS视图显示当前用户在当前CN上正在执行的作业的负载管理记录。

表 18-163 GS_WLM_SESSION_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
dbname	name	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN节点名称。
username	name	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
pid	bigint	后端线程ID。
block_time	bigint	语句执行前的阻塞时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
duration	bigint	语句已经执行的时间，单位ms。
estimate_total_time	bigint	语句执行预估总时间，单位ms。
estimate_left_time	bigint	语句执行预估剩余时间，单位ms。
enqueue	text	工作负载管理资源状态。
resource_pool	name	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句在各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在各DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的最小IO读速率，单位KB/s。
max_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的最大IO读速率，单位KB/s。
average_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的平均IO读速率，单位KB/s。
min_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的最小IO写速率，单位KB/s。
max_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的最大IO写速率，单位KB/s。
average_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的平均IO写速率，单位KB/s。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
unique_sql_id	bigint	归一化的Unique SQL ID。
queryid	bigint	语句执行使用的内部query id。
query	text	正在执行的语句。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
stmt_type	text	语句所对应的查询类型。
except_info	text	语句触发的异常规则信息。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。 该字段仅8.3.0.100及以上版本支持。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对3.0表及外表生效。
block_name	text	与语句匹配的对应拦截规则名。该字段仅9.1.0.200及以上集群版本支持。

18.3.92 GS_WLM_SQL_ALLOW

GS_WLM_SQL_ALLOW视图显示已经设置的资源管理SQL白名单。

其中白名单包括两部分内容：

- 系统默认的SQL白名单。

 版权所有 © 华为云计算技术有限公司

- 通过GUC参数**wlm_sql_allow_list**设置的SQL白名单。

18.3.93 GS_WORKLOAD_SQL_COUNT

GS_WORKLOAD_SQL_COUNT视图显示当前节点上Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 18-164 GS_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

18.3.94 GS_WORKLOAD_SQL_ELAPSE_TIME

GS_WORKLOAD_SQL_ELAPSE_TIME视图显示当前节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

表 18-165 GS_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

18.3.95 GS_WORKLOAD_TRANSACTION

GS_WORKLOAD_TRANSACTION视图提供查询单CN上Workload控制组相关的事务信息。数据库记录每个Workload控制组事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 18-166 GS_WORKLOAD_TRANSACTION 字段

名称	类型	描述
workload	name	Workload控制组名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

18.3.96 MPP_TABLES

MPP_TABLES视图显示PGXC_CLASS中的表信息。

表 18-167 MPP_TABLES 字段

名称	类型	描述
schemaname	name	包含表的模式名。
tablename	name	表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tableowner	name	表的所有者。
tablespace	name	表所在的表空间。
pgroup	name	节点群的名称。
nodeoids	oidvector_extend	表分布的节点OID列表。

18.3.97 PG_AVAILABLE_EXTENSION_VERSIONS

PG_AVAILABLE_EXTENSION_VERSIONS视图显示数据库中某些特性的扩展版本信息。

表 18-168 PG_AVAILABLE_EXTENSION_VERSIONS 字段

名称	类型	描述
name	name	扩展名。
version	text	版本名。
installed	boolean	如果此扩展的版本当前已经安装，则为真。
superuser	boolean	如果只允许系统管理员安装此扩展，则为真。
relocatable	boolean	如果扩展可以重新加载到另一个模式，则为真。
schema	name	扩展必须安装到的模式名，如果部分或全部可重新定位，则为NULL。
requires	name[]	必备扩展的名称，如果没有则为NULL。
comment	text	扩展的控制文件的注释字符串。

18.3.98 PG_AVAILABLE_EXTENSIONS

PG_AVAILABLE_EXTENSIONS视图显示数据库中某些特性的扩展信息。

表 18-169 PG_AVAILABLE_EXTENSIONS 字段

名称	类型	描述
name	name	扩展名。
default_version	text	缺省版本名，如果没有指定则为NULL。
installed_version	text	当前安装的扩展版本，如果未安装则为NULL。
comment	text	扩展控制文件的注释字符串。

版权所有 © 华为云计算技术有限公司

18.3.99 PG_BULKLOAD_STATISTICS

在集群任一正常节点上，通过查询PG_BULKLOAD_STATISTICS视图可以获取当前登录节点正在进行的导入导出业务执行情况，其中每一个导入/导出业务对应一条记录。需要有系统管理员权限才可以访问此视图。

表 18-170 PG_BULKLOAD_STATISTICS 字段

名称	类型	描述
node_name	text	节点名称。
db_name	text	数据库名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
session_id	bigint	GDS的会话ID。
direction	text	业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
query	text	查询语句。
address	text	当前导入导出外表的location。
query_start	timestamp with time zone	导入/导出开始时间。
total_bytes	bigint	待处理数据的总大小。 仅GDS普通文件导入时，且该行记录来自CN节点才会显示，否则为空。
phase	text	当前业务导入导出执行阶段，包括：INITIALIZING、TRANSFER_DATA、RELEASE_RESOURCE。
done_lines	bigint	已传输行数。
done_bytes	bigint	已传输字节数。

18.3.100 PG_COMM_CLIENT_INFO

PG_COMM_CLIENT_INFO视图存储单个节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

版权所有 © 华为云计算技术有限公司

表 18-171 PG_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
query_id	bigint	查询ID，对应debug_query_id。
socket	integer	如果是物理连接，显示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，显示sid，显示-1时表示当前连接是物理连接。

18.3.101 PG_COMM_DELAY

PG_COMM_DELAY视图展示单个DN的通信库时延状态。

表 18-172 PG_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称。
remote_name	text	连接对端时延最大的节点名称。
remote_host	text	连接对端IP地址。
stream_num	integer	当前物理连接使用的stream逻辑连接数量。
min_delay	integer	当前物理连接探测到的最小时延，单位微秒。
average	integer	当前物理连接探测时延的平均值，单位微秒。
max_delay	integer	当前物理连接探测到的最大时延，单位微秒。 **说明** 取值为-1，表示时延探测超时失败，请重新建立节点间连接后再执行查询。

18.3.102 PG_COMM_STATUS

PG_COMM_STATUS视图展示单个DN的通信库状态。

 版权所有 © 华为云计算技术有限公司

表 18-173 PG_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck/s	integer	节点通信库接收速率，单位Byte/s。
txpck/s	integer	节点通信库发送速率，单位Byte/s。
rxkB/s	bigint	节点通信库接收速率，单位KByte/s。
txkB/s	bigint	节点通信库发送速率，单位KByte/s。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位KByte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位KByte。
%USED(PM)	integer	postmaster线程实时使用率。
%USED (sflow)	integer	gs_sender_flow_controller线程实时使用率。
%USED (rflow)	integer	gs_receiver_flow_controller线程实时使用率。
%USED (rloop)	integer	多个gs_receivers_loop线程中最高的实时使用率。
stream	integer	当前使用的逻辑连接总数。

18.3.103 PG_COMM_RECV_STREAM

PG_COMM_RECV_STREAM视图展示单个DN上所有的通信库接收流状态。

表 18-174 PG_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接接收报文正常。 ● HOLD：表示逻辑连接接收报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
recv_bytes	bigint	通信流接收的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均接收速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte。

18.3.104 PG_COMM_SEND_STREAM

PG_COMM_SEND_STREAM视图展示单个DN上所有的通信库发送流状态。

表 18-175 PG_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	通信流当前的状态。 ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接发送报文正常。 ● HOLD：表示逻辑连接发送报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

18.3.105 PG_COMM_QUERY_SPEED

PG_COMM_QUERY_SPEED视图展示单个节点上所有query对应的流量信息。

表 18-176 PG_COMM_QUERY_SPEED 字段

名称	类型	描述
node_name	text	节点名称。
query_id	bigint	通信流对应的debug_query_id编号。
rxkB/s	bigint	查询对应的通信流接收速率，单位Byte/s。
txkB/s	bigint	查询对应的通信流发送速率，单位Byte/s。
rxkB	bigint	查询对应的通信流接收数据总量，单位Byte。
txkB	bigint	查询对应的通信流发送数据总量，单位Byte。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rxpck/s	bigint	查询对应的通信包接收速率，单位packages/s。
txpck/s	bigint	查询对应的通信包发送速率，单位packages/s。
rxpck	bigint	查询对应的通信包接收总量，单位packages。
txpck	bigint	查询对应的通信包发送总量，单位packages。

18.3.106 PG_CONTROL_GROUP_CONFIG

PG_CONTROL_GROUP_CONFIG视图存储系统的控制组配置信息。

表 18-177 PG_CONTROL_GROUP_CONFIG 字段

名称	类型	描述
pg_control_group_config	text	控制组的配置信息。

18.3.107 PG_CURSORS

PG_CURSORS视图列出了当前可用的游标。

表 18-178 PG_CURSORS 字段

名称	类型	描述
name	text	游标名。
statement	text	声明该游标时的查询语句。
is_holdable	boolean	如果该游标是持久的（就是在声明该游标的事务结束后仍然可以访问该游标）则为TRUE，否则为FALSE。
is_binary	boolean	如果该游标被声明为BINARY则为TRUE，否则为FALSE。
is_scrollable	boolean	如果该游标可以滚动（就是允许以不连续的方式检索）则为TRUE，否则为FALSE。
creation_time	timestamp with time zone	声明该游标的时间戳。

版权所有 © 华为云计算技术有限公司

18.3.108 PG_EXT_STATS

PG_EXT_STATS视图提供对存储在PG_STATISTIC_EXT表里面的扩展统计信息的访问。扩展统计信息目前包括多列统计信息。

表 18-179 PG_EXT_STATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
attname	int2vector	PG_STATISTIC_EXT.stakey	统计信息扩展的多列信息。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段组合为空的百分比。
avg_width	integer	-	字段组合记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段组合中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 • 如果等于0，表示独立数值的数目未知。 **说明** 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个字段组合中独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段组合中非NULL数据的唯一值的数目。 • 如果大于0，表示独立数值的实际数目。 • 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段组合的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 • 如果等于0，表示独立数值的数目未知。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值均不为NULL。
most_common_freqs	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
most_common_vals_null	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值中至少有一个值为NULL。
most_common_freqs_null	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals_null是NULL，则为NULL。

18.3.109 PG_GET_INVALID_BACKENDS

PG_GET_INVALID_BACKENDS视图提供显示CN上连接到当前DN备机的后端线程信息。

表 18-180 PG_GET_INVALID_BACKENDS 字段

名称	类型	描述
pid	bigint	线程ID。
node_name	text	后端线程中连接的节点信息。
dbname	name	当前连接的数据库。
backend_start	timestamp with time zone	后端线程启动的时间。
query	text	后端线程正在执行的查询语句。

18.3.110 PG_GET_SENDERS_CATCHUP_TIME

PG_GET_SENDERS_CATCHUP_TIME视图显示单个DN上当前活跃的主备发送线程的追赶信息。

 版权所有 © 华为云计算技术有限公司

表 18-181 PG_GET_SENDERS_CATCHUP_TIME 字段

名称	类型	描述
pid	bigint	当前sender的线程ID。
lwpid	integer	当前sender的lwpid。
local_role	text	本地的角色。
peer_role	text	对端的角色。
state	text	当前sender的复制状态。
type	text	当前sender的类型。
catchup_start	timestamp with time zone	catchup启动的时间。
catchup_end	timestamp with time zone	catchup结束的时间。
catchup_type	text	catchup方式为全量还是增量。
catchup_bcm_filename	text	catchup当前执行的bcm文件。
catchup_bcm_finished	integer	catchup已经操作完成的bcm文件数量。
catchup_bcm_total	integer	catchup总共需要操作的bcm文件数量。
catchup_percent	text	catchup已经操作完成的百分比。
catchup_remaining_time	text	catchup预估剩余时间。

18.3.111 PG_GLOBAL_TEMP_ATTACHED_PIDS

查看全局临时表在当前节点占有资源的会话信息。该视图仅8.2.1.220及以上集群版本支持。

表 18-182 PG_GLOBAL_TEMP_ATTACHED_PIDS 字段

名称	类型	描述
schemaname	name	模式名。
tablename	name	表名。
pid	bigint	用于表示会话的PID。

18.3.112 PG_GROUP

PG_GROUP视图查看数据库认证角色及角色之间的成员关系。

 版权所有 © 华为云计算技术有限公司

表 18-183 PG_GROUP 字段

名称	类型	描述
groname	name	组的名称。
grosysid	oid	组的ID。
grolist	oid[]	一个数组，包含这个组里面所有角色的ID。

18.3.113 PG_INDEXES

PG_INDEXES视图提供对数据库中每个索引的有用信息的访问。

表 18-184 PG_INDEXES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表和索引的模式名。
tablename	name	PG_CLASS.relname	此索引所服务的表名。
indexname	name	PG_CLASS.relname	索引名。
tablespace	name	PG_TABLESPACE.spcname	包含索引的表空间名称。
indexdef	text	-	索引定义（一个重建的CREATE INDEX命令）。

应用示例

查询指定表的索引信息。

```
SELECT * FROM pg_indexes WHERE tablename = 'mytable';
 schemaname | tablename |   indexname    | tablespace |                                indexdef
------------+-----------+----------------+------------+-------------------------------------------------------------------------
+-------------------------------------------------------------------------
 public     | mytable   | idx_mytable_id |            | CREATE INDEX idx_mytable_id ON mytable USING btree
(id) TABLESPACE pg_default
(1 row)
```

查询当前数据库指定模式下所有表的索引信息。

```
SELECT tablename, indexname, indexdef FROM pg_indexes WHERE schemaname = 'public' ORDER BY
tablename,indexname;
 tablename |     indexname      |                                          indexdef
-----------+--------------------
+-------------------------------------------------------------------------------------------------
 books     | books_pkey         | CREATE UNIQUE INDEX books_pkey ON books USING btree (id) TABLESPACE
pg_default
 books     | idx_books_tags_gin | CREATE INDEX idx_books_tags_gin ON books USING gin (tags)
TABLESPACE pg_default
 customer  | c_custkey_key      | CREATE UNIQUE INDEX c_custkey_key ON customer USING btree
```


版权所有 © 华为云计算技术有限公司

```
(c_custkey, c_name) TABLESPACE pg_default
 mytable   | idx_mytable_id     | CREATE INDEX idx_mytable_id ON mytable USING btree (id) TABLESPACE
pg_default
 test1     | idx_test_id        | CREATE INDEX idx_test_id ON test1 USING btree (id) TABLESPACE pg_default
 v0        | v0_pkey            | CREATE UNIQUE INDEX v0_pkey ON v0 USING btree (c) TABLESPACE pg_default
(6 rows)
```

18.3.114 PG_JOB

PG_JOB视图存储用户创建的定时任务的任务详细信息。

PG_JOB视图用于代替历史版本的PG_JOB系统表，提供对之前版本的前向兼容。原PG_JOB系统表已经变更为PG_JOBS系统表，关于PG_JOBS系统表的描述详见**PG_JOBS**。

表 18-185 PG_JOB 字段

名字	类型	描述
job_id	bigint	作业ID。
current_postgres_pid	bigint	如果当前任务正在被执行，那么此处记录运行此任务的postgres线程ID。默认为-1，表示此任务未被执行或已执行完毕。
log_user	name	创建者的UserName。
priv_user	name	作业执行者的UserName。
dbname	name	标识作业执行的数据库名。
node_name	name	标识当前作业是在哪个CN上创建和执行。
job_status	text	当前任务的执行状态，取值范围：('r', 's', 'f', 'd','p','w','l')，默认为's'，取值含义： ● r=running ● s=successfully finished ● f=job failed ● d=disable ● p=pending ● w=waiting ● l=launching **说明** ● 当用户将定时任务关闭（即job_queue_processes为0时），由于监控job执行的线程不会启动，所以job_status不会根据job的实时状态进行设置，用户不需要关注job_status。 ● 只有当开启定时任务功能（job_queue_processes为非0时），系统才会根据当前job的实时状态刷新job_status的值。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。
nspname	name	作业运行时所在的命名空间的名称。
what	text	作业内容。

18.3.115 PG_JOB_PROC

PG_JOB_PROC视图用于代替之前版本的PG_JOB_PROC系统表，提供对之前版本的前向兼容。原PG_JOB_PROC系统表已经和原PG_JOB系统表一同并入当前版本的PG_JOBS系统表，关于PG_JOBS系统表的描述详见**PG_JOBS**。

表 18-186 PG_JOB_PROC 字段

名字	类型	描述
job_id	bigint	作业ID。
what	text	作业内容。

18.3.116 PG_JOB_SINGLE

PG_JOB_SINGLE视图用于显示当前节点的作业信息。

 版权所有 © 华为云计算技术有限公司

表 18-187 PG_JOB_SINGLE 字段

名字	类型	描述
job_id	bigint	作业ID。
current_postgres_pid	bigint	如果当前任务正在被执行，那么此处记录运行此任务的postgres线程ID。默认为-1，表示此任务未被执行或已执行完毕。
log_user	name	创建者的UserName。
priv_user	name	作业执行者的UserName。
dbname	name	标识作业执行的数据库名。
node_name	name	标识当前作业是在哪个CN上创建和执行。
job_status	text	当前任务的执行状态，取值范围：('r', 's', 'f', 'd','p','w','l')，默认为's'，取值含义： ● r=running ● s=successfully finished ● f=job failed ● d=disable ● p=pending ● w=waiting ● l=launching **说明** ● 当用户将定时任务关闭（即job_queue_processes为0时），由于监控job执行的线程不会启动，所以job_status不会根据job的实时状态进行设置，用户不需要关注job_status。 ● 只有当开启定时任务功能（job_queue_processes为非0时），系统才会根据当前job的实时状态刷新job_status的值。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。

 版权所有 © 华为云计算技术有限公司

名字	类型	描述
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。
nspname	name	作业运行时所在的命名空间的名称。
what	text	作业内容。

18.3.117 PG_LIFECYCLE_DATA_DISTRIBUTE

PG_LIFECYCLE_DATA_DISTRIBUTE视图查询OBS多温表中冷热数据分布情况。

表 18-188 PG_LIFECYCLE_DATA_DISTRIBUTE 字段

名称	类型	描述
schemaname	name	模式名。
tablename	name	当前表名。
nodename	name	节点名。
hotpartition	text	该DN节点上的热分区。
coldpartition	text	该DN节点上的冷分区。
switchablepartition	text	该DN节点上的可切分区。
hotdatasize	text	该DN节点上的热分区数据大小。
colddatasize	text	该DN节点上的冷分区数据大小。
switchabledatasize	text	该DN节点上的可切分区数据大小。

18.3.118 PG_LOCKS

PG_LOCKS视图存储各打开事务所持有的锁信息。

版权所有 © 华为云计算技术有限公司

表 18-189 PG_LOCKS 字段

名称	类型	引用	描述
locktype	text	-	被锁定对象的类型：relation，extend，page，tuple，transactionid，virtualxid，object，userlock，advisory。
database	oid	PG_DATABASE.oid	被锁定对象所在数据库的OID。 • 如果被锁定的对象是共享对象，则OID为0。 • 如果是一个事务ID，则为NULL。
relation	oid	PG_CLASS.oid	被锁定对象关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。
page	integer	-	关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。
tuple	smallint	-	页面里边的行编号，如果对象不是行，则为NULL。
virtualxid	text	-	事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。
transactionid	xid	-	事务的ID，如果对象不是一个事务ID，则为NULL。
classid	oid	PG_CLASS.oid	包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。
objid	oid	-	对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。
objsubid	smallint	-	对于表的某个字段对应为字段编号；对于其他对象类型，该字段为0；如果该对象不是普通数据库对象，则为NULL。
virtualtransaction	text	-	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	-	持有此锁或者等待此锁的服务器线程的逻辑ID。如果锁被一个预备事务持有，则为NULL。
mode	text	-	此线程持有的或者是期望持有的锁模式。更多有关锁模式的内容请参见《数据仓库服务SQL语法参考》中的“LOCK”章节。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
granted	boolean	-	● 如果锁是持有锁，则为TRUE。 ● 如果锁是等待锁，则为FALSE。
fastpath	boolean	-	如果通过fast-path获得锁，则为TRUE；如果通过主锁表获得，则为FALSE。
waittime	timestamp with time zone	-	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	-	开始持有锁的时间戳。该字段仅9.1.0.200及以上集群版本支持。

18.3.119 PG_LWLOCKS

PG_LWLOCKS视图提供当前实例正在持有的或等待的轻量级锁信息。该视图仅9.1.0.200及以上集群版本支持。

表 18-190 PG_LWLOCKS 字段

名称	类型	描述
pid	bigint	后端线程ID。
query_id	bigint	查询语句的ID。
lwtid	integer	后端线程的轻量级线程号。
reqlockid	integer	当前线程正在请求的轻量级锁ID。
reqlock	text	reqlockid对应的轻量级锁名称。
heldlocknums	integer	当前线程已经获得的轻量级锁的数量。
heldlockid	integer	当前线程已经获得的轻量级锁ID。
heldlock	text	heldlockid对应的轻量级锁名称。
heldlockmode	text	heldlockid对应的轻量级锁模式。

示例

使用PG_LWLOCKS视图查询当前实例正在持有的或等待的轻量级锁信息：

```
SELECT * FROM pg_lwlocks;
      pid       |     query_id      | lwtid | reqlockid | reqlock | heldlocknums | heldlockid |      heldlock      |
heldlockmode
----------------+-------------------+-------+-----------+---------+--------------+------------+--------------------
+--------------
 139810224192480 |                 0 | 54842 |           |         |            1 |          7 | WALWriteLock       | Exclusive
 139810224199520 | 78250043526306022 | 54963 |           |         |            1 |     193860 |
```


版权所有 © 华为云计算技术有限公司

```
BUFFER_POOL_LWLOCK | Exclusive
(2 rows)
```

18.3.120 PG_NODE_ENV

PG_NODE_ENV视图提供获取当前节点的环境变量信息。

表 18-191 PG_NODE_ENV 字段

名称	类型	描述
node_name	text	当前节点名称。
host	text	当前节点的主机名称。
process	integer	当前节点的进程号。
port	integer	当前节点的端口号。
installpath	text	当前节点的安装目录。
datapath	text	当前节点的数据目录。
log_directory	text	当前节点的日志目录。

18.3.121 PG_OS_THREADS

PG_OS_THREADS视图提供当前节点下所有线程的状态信息。

表 18-192 PG_OS_THREADS 字段

名称	类型	描述
node_name	text	当前节点名称。
pid	bigint	当前节点进程中正在运行的线程号。
lwpid	integer	与pid对应的轻量级线程号。
thread_name	text	与pid对应的线程名称。
creation_time	timestamp with time zone	与pid对应的线程创建的时间。

18.3.122 PG_POOLER_STATUS

PG_POOLER_STATUS视图查询pooler中的缓存连接状态。该视图只能在CN上执行查询，显示本地CN的pooler模块的连接缓存信息。

版权所有 © 华为云计算技术有限公司

表 18-193 PG_POOLER_STATUS 字段

名称	类型	描述
database	text	数据库名称。
user_name	text	用户名。
tid	bigint	连接CN的线程ID。
node_oid	bigint	连接的实例节点OID。
node_name	name	连接的实例节点名称。
in_use	boolean	连接是否正被使用。 ● t（true）：表示连接正在使用。 ● f（false）：表示连接没有使用。
fdsock	bigint	对端socket。
remote_pid	bigint	对端线程号。
session_params	text	由此连接下发的GUC session参数。

应用示例

查看pooler连接池信息：

```
select database,user_name,node_name,in_use,count(*) from pg_pooler_status group by 1, 2, 3 ,4 order by 5
desc limit 50;
 database | user_name |   node_name    | in_use | count
----------+-----------+----------------+--------+-------
 mydbdemo | user3     | cn_5001        | f      |     2
 mydbdemo | user3     | dn_6005_6006 | t      |     2
 mydbdemo | user3     | dn_6001_6002 | t      |     2
 mydbdemo | user3     | dn_6003_6004 | f      |     2
 mydbdemo | user3     | dn_6003_6004 | t      |     2
 mydbdemo | user3     | dn_6005_6006 | f      |     2
 mydbdemo | user3     | dn_6001_6002 | f      |     2
 mydbdemo | user3     | cn_5002        | f      |     2
 gaussdb  | user3     | dn_6003_6004 | f      |     1
 mydbdemo | user3     | cn_5001        | t      |     1
 music    | user2     | dn_6003_6004 | f      |     1
 music    | user2     | dn_6005_6006 | f      |     1
 gaussdb  | user1     | dn_6005_6006 | f      |     1
(13 rows)
```

18.3.123 PG_PREPARED_STATEMENTS

PG_PREPARED_STATEMENTS视图显示当前会话所有可用的预备语句。

表 18-194 PG_PREPARED_STATEMENTS 字段

名称	类型	描述
name	text	预备语句的标识符。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
statement	text	创建该预备语句的查询字符串。对于从SQL创建的预备语句而言是客户端提交的PREPARE语句；对于通过前/后端协议创建的预备语句而言是预备语句自身的文本。
prepare_time	timestamp with time zone	创建该预备语句的时间戳。
parameter_types	regtype[]	该预备语句期望的参数类型，以regtype类型的数组格式出现。与该数组元素相对应的OID可以通过把regtype转换为oid值得到。
from_sql	boolean	• 如果该预备语句是通过PREPARE语句创建的则为true。 • 如果是通过前/后端协议创建的则为false。

18.3.124 PG_PREPARED_XACTS

PG_PREPARED_XACTS视图显示当前准备好进行两阶段提交的事务的信息。

表 18-195 PG_PREPARED_XACTS 字段

名称	类型	引用	描述
transaction	xid	-	预备事务的数字事务标识。
gid	text	-	赋予该事务的全局事务标识。
prepared	timestamp with time zone	-	事务准备好提交的时间。
owner	name	PG_AUTHID.rolname	执行该事务的用户名。
database	name	PG_DATABASE.datname	执行该事务所在的数据库名。

18.3.125 PG_PUBLICATION_TABLES

PG_PUBLICATION_TABLES视图显示发布与其所发布的表之间的映射信息。和底层的系统表PG_PUBLICATION_REL不同，这个视图展开了定义为FOR ALL TABLES和FOR ALL TABLES IN SCHEMA的发布，对这类发布来说，每一个可发布的表都有一行。该系统视图仅8.2.0.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-196 PG_PUBLICATION_TABLES 字段

名称	类型	描述
pubname	name	发布的名称。
schemaname	name	表的模式的名称。
tablename	name	表的名称。

应用示例

查询所有发布表。

```
SELECT * FROM PG_PUBLICATION_TABLES;
 pubname | schemaname | tablename
---------+------------+-----------
 mypub   | public     | t1
 mypub   | public     | t2
(2 rows)
```

18.3.126 PG_QUERYBAND_ACTION

PG_QUERYBAND_ACTION视图显示query_band关联行为和次序。

表 18-197 PG_QUERYBAND_ACTION 字段

名称	类型	描述
qband	text	query_band键值对。
respool_id	oid	query_band关联资源池OID。
respool	text	query_band关联资源池名。
priority	text	query_band关联队列内优先级。
qborder	integer	query_band搜索次序。

18.3.127 PG_REPLICATION_SLOTS

PG_REPLICATION_SLOTS视图查看复制节点的信息。

表 18-198 PG_REPLICATION_SLOTS 字段

名称	类型	描述
slot_name	text	复制节点的名称。
plugin	name	逻辑复制槽对应的输出插件名。
slot_type	text	复制节点的类型。
datoid	oid	复制节点的数据库OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
database	name	复制节点的数据库名称。
active	boolean	复制节点是否为激活状态。
xmin	xid	复制节点事务标识。
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识。
restart_lsn	text	复制节点的Xlog文件信息。
dummy_standby	boolean	复制节点是否为假备。

18.3.128 PG_ROLES

PG_ROLES视图提供访问数据库角色的相关信息。

表 18-199 PG_ROLES 字段

名称	类型	引用	描述
rolname	name	-	角色名。
rolsuper	boolean	-	该角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	-	该角色是否继承角色的权限。
rolcreaterole	boolean	-	该角色是否可以创建其他的角色。
rolcreatedb	boolean	-	该角色是否可以创建数据库。
rolcatupdate	boolean	-	该角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	-	该角色是否可以登录数据库。
rolreplication	boolean	-	该角色是否可以复制。
rolauditadmin	boolean	-	该角色是否为审计管理员。
rolsystemadmin	boolean	-	该角色是否为系统管理员。
rolconnlimit	integer	-	限制单个用户在单个CN上的最大并发连接数。-1表示无限制。
rolpassword	text	-	不是口令，总是********。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
rolvalidbegin	timestamp with time zone	-	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	-	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
rolrespool	name	-	用户所能够使用的resource pool。
rolparentid	oid	PG_AUTHID.rolparentid	用户所在组用户的OID。
roltabspace	text	-	用户永久表存储空间限额。
roltempspace	text	-	用户临时表存储空间限额。
rolspillspace	text	-	用户算子落盘空间限额。
rolconfig	text[]	-	运行时配置变量的会话缺省。
oid	oid	PG_AUTHID.oid	角色的ID。
roluseft	boolean	PG_AUTHID.roluseft	角色是否可以操作外表。
nodegroup	name	-	角色所关联的逻辑集群名字，如果没有关联逻辑集群，该值为空。

18.3.129 PG_RULES

PG_RULES视图提供对查询重写规则的有用信息访问的接口。

表 18-200 PG_RULES 字段

名称	类型	描述
schemaname	name	包含表的模式名。
tablename	name	规则作用的表名。
rulename	name	规则的名称。
definition	text	规则定义（一个重新构造的创建命令）。

18.3.130 PG_RUNNING_XACTS

PG_RUNNING_XACTS视图显示当前节点运行事务的信息。

 版权所有 © 华为云计算技术有限公司

表 18-201 PG_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄。
gxid	xid	事务ID号。
state	tinyint	事务状态（3：prepared或者0：starting）。
node	text	节点名称。
xmin	xid	节点上当前数据涉及的最小事务号xmin。
vacuum	boolean	标志当前事务是否是lazy vacuum事务。
timeline	bigint	标志数据库重启次数。
prepare_xid	xid	处于prepared状态的事务的ID号，若不在prepared状态，值为0。
pid	bigint	事务对应的线程ID。
next_xid	xid	CN传给DN的事务ID号。

18.3.131 PG_SECLABELS

PG_SECLABELS视图提供关于安全标签的信息。

表 18-202 PG_SECLABELS 字段

名称	类型	引用	描述
objoid	oid	任意OID属性	安全标签所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象的系统表的OID。
objsubid	integer	-	对于某个在表字段上的安全标签，为字段编号（引用表本身的objoid和classoid）。对于所有其他对象类型，该字段为0。
objtype	text	-	标签出现的对象的类型。
objnamespace	oid	PG_NAMESPACE.oid	对象的命名空间的OID，如果适用；否则为NULL。
objname	text	-	标签适用的对象名。
provider	text	PG_SECLABEL.provider	与标签相关的标签提供者。
label	text	PG_SECLABEL.label	应用于此对象的安全标签。

版权所有 © 华为云计算技术有限公司

18.3.132 PG_SEQUENCES

PG_SEQUENCES视图显示当前用户有权限的序列属性信息。该视图仅9.1.0及以上集群版本支持。

表 18-203 PG_SEQUENCES 字段

名称	类型	描述
schemaname	name	命名空间的名称。
sequencename	name	序列的名称。
sequenceowner	name	序列的所有者。
start_value	bigint	序列的起始值。
min_value	bigint	序列生成的最小值。
max_value	bigint	序列生成的最大值。
increment_by	bigint	序列生成的步长，即每次生成的值增加的量。
cycle	boolean	如果设置为true，则在达到最大值后重新开始从最小值生成序列值；如果设置为false，则在达到最大值后停止生成序列值。
cache_size	bigint	序列缓存值的大小。
last_value	bigint	序列最近一次生成的值。

18.3.133 PG_SESSION_WLMSTAT

PG_SESSION_WLMSTAT视图显示和当前用户执行作业正在运行时的负载管理相关信息。

表 18-204 PG_SESSION_WLMSTAT 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
datname	name	连接后端的数据库名称。
threadid	bigint	后端线程ID。
processid	integer	后端线程的PID。
usesysid	oid	登录后端的用户OID。
appname	text	连接到后端的应用名。
usename	name	登录到该后端的用户名。
priority	bigint	语句所在Cgroups的优先级。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
attribute	text	语句的属性： ● Ordinary：语句发送到数据库后被解析前的默认属性。 ● Simple：简单语句。 ● Complicated：复杂语句。 ● Internal：数据库内部语句。
block_time	bigint	语句当前为止的pending的时间，单位s。
elapsed_time	bigint	语句当前为止的实际执行时间，单位s。
total_cpu_time	bigint	语句在上一时间周期内的DN上CPU使用的总时间，单位s。
cpu_skew_percent	integer	语句在上一时间周期内的DN上CPU使用的倾斜率。
statement_mem	integer	语句执行所需要的估算内存。
active_points	integer	语句占用的资源池并发点数。
dop_value	integer	语句的从资源池中获取的dop值。
control_group	text	语句当前所使用的Cgroups。
status	text	语句当前的状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active：非以上四种状态外的正常状态。 ● unknown：未知状态。
enqueue	text	语句当前的排队情况，包括： ● Global：全局排队。 ● Respool：资源池排队。 ● CentralQueue：在中心协调节点(CCN)中排队。 ● Transaction：语句处于一个事务块中。 ● StoredProc：语句处于一个存储过程中。 ● None：未在排队。 ● Forced None：事务块语句或存储过程语句由于超出设定的等待时间而强制执行。
resource_pool	name	语句当前所在的资源池。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
query	text	该后端的最新查询。如果state状态是active，此字段显示当前正在执行的查询。所有其他情况表示上一个查询。
isplana	bool	逻辑集群模式下，语句当前是否占用其他逻辑集群的资源执行。该值默认为f，表示不占用其他逻辑集群的资源执行。
node_group	text	语句所属用户对应的逻辑集群。
lane	text	表示语句查询的快慢车道。 • fast：快车道。 • slow：慢车道。 • none：未管控。

18.3.134 PG_SESSION_IOSTAT

PG_SESSION_IOSTAT视图8.1.2版本中已废弃，为兼容历史版本功能保留该视图，当前版本查询无效。可使用**PGXC_WLM_SESSION_STATISTICS**视图查看所有CN上正在执行的作业的负载管理信息。

表 18-205 PG_SESSION_IOSTAT 字段

名称	类型	描述
query_id	bigint	作业ID。
mincurriops	integer	该作业当前io在各DN中的最小值。
maxcurriops	integer	该作业当前io在各DN中的最大值。
minpeakiops	integer	在作业运行时，作业io峰值中，各DN的最小值。
maxpeakiops	integer	在作业运行时，作业io峰值中，各DN的最大值。
io_limits	integer	该作业所设GUC参数io_limits。
io_priority	text	该作业所设GUC参数io_priority。
query	text	作业。
node_group	text	作业所属用户对应的逻辑集群。

18.3.135 PG_SETTINGS

PG_SETTINGS视图显示数据库运行时参数的相关信息。

 版权所有 © 华为云计算技术有限公司

表 18-206 PG_SETTINGS 字段

名称	类型	描述
name	text	参数名称。
setting	text	参数当前值。
unit	text	参数的隐式结构。
category	text	参数的逻辑组。
short_desc	text	参数的简单描述。
extra_desc	text	参数的详细描述。
context	text	设置参数值的上下文，包括internal，postmaster，sighup，backend，superuser和user。
vartype	text	参数类型，包括bool，enum，integer，real和string。
source	text	参数的赋值方式。
min_val	text	参数最小值。如果参数类型不是数值型，那么该字段值为null。
max_val	text	参数最大值。如果参数类型不是数值型，那么该字段值为null。
enumvals	text[]	enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。
boot_val	text	数据库启动时参数默认值。
reset_val	text	数据库重置时参数默认值。
sourcefile	text	设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。
sourceline	integer	设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。

18.3.136 PG_SHADOW

PG_SHADOW视图显示了所有在PG_AUTHID中标记了rolcanlogin的角色的属性。

此系统视图的名称源于它不是所有用户可读的，因为包含口令。**PG_USER**是一个在PG_SHADOW上全局可读的视图，只是把口令域填充成了空白。

版权所有 © 华为云计算技术有限公司

表 18-207 PG_SHADOW 字段

名字	类型	引用	描述
usename	name	PG_AUTHID.rolname	用户名。
usesysid	oid	PG_AUTHID.oid	用户的ID。
usecreatedb	boolean	-	用户可以创建数据库。
usesuper	boolean	-	用户是系统管理员。
usecatupd	boolean	-	用户可以更新系统表。即使是系统管理员，如果此字段不为真，也不能更新系统表。
userepl	boolean	-	用户可以初始化流复制和使系统处于或不处于备份模式。
passwd	text	-	口令（可能是加密的）；如果没有则为null。参阅PG_AUTHID获取加密的口令是如何存储的信息。
valbegin	timestamp with time zone	-	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	-	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	-	用户使用的资源池。
parent	oid	-	父资源池。
spacelimit	text	-	永久表存储空间限额。
tempspacelimit	text	-	临时表存储空间限额。
spillspacelimit	text	-	算子落盘空间限额。
useconfig	text[ ]	-	运行时配置变量的会话缺省。

18.3.137 PG_SHARED_MEMORY_DETAIL

PG_SHARED_MEMORY_DETAIL视图查询所有已产生的共享内存上下文的使用信息。

 版权所有 © 华为云计算技术有限公司

表 18-208 PG_SHARED_MEMORY_DETAIL 字段

名字	类型	描述
contextname	text	内存上下文的名字。
level	smallint	当前上下文在整体内存上下文中的层级。
parent	text	上级内存上下文。
totalsize	bigint	共享内存总大小，单位Byte。
freesize	bigint	共享内存剩余大小，单位Byte。
usedsize	bigint	共享内存使用大小，单位Byte。

18.3.138 PG_STATS

PG_STATS视图提供对存储在pg_statistic表里面的单列统计信息的访问。

表 18-209 PG_STATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
attname	name	PG_ATTRIBUTE.attname	字段名。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
partname	name	-	分区名。该字段仅8.2.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

18.3.139 PG_STAT_ACTIVITY

PG_STAT_ACTIVITY视图显示和当前用户查询相关的信息。若有管理员权限或预置角色权限可以显示和所有用户查询相关的信息。

表 18-210 PG_STAT_ACTIVITY 字段

名称	类型	描述
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
lwtid	integer	轻量级线程ID。
usesysid	oid	登录该后端的用户OID。
usename	name	登录该后端的用户名。
application_name	text	连接到该后端的应用名。
client_addr	inet	连接到该后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，此字段是通过client_addr的反向DNS查找得到。此字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁或者等待节点则为t，否则为f。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
enqueue	text	语句当前排队状态。可能值是： ● waiting in global queue：表示语句在全局并发队列排队中，主要包含并发数超过单CN配置的max_active_statements。 ● waiting in respool queue：表示语句在资源池排队中，简单作业并发受限，主要是简单作业并发超过快车道并发上限max_dop。 ● waiting in ccn queue：表示作业在CCN排队中，包含全局内存排队和慢车道内存和并发排队，包含以下场景： 1. 全局可用内存超过上限，进行全局内存队列排队。 2. 资源池慢车道并发上限，即资源池并发超过active_statements上限。 3. 资源池慢车道内存上限，即资源池并发作业估算内存超过mem_percent计算的上限。 ● 空或no waiting queue：表示语句正在运行。
state	text	后端当前总体状态。可能值是： ● active：后台正在执行查询。 ● idle：后台正在等待新的客户端命令。 ● idle in transaction：后端在事务中，但事务中没有语句在执行。 ● idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 ● fastpath function call：后端正在执行一个fast-path函数。 ● disabled：如果后端禁用track_activities，则报告此状态。 **说明** 普通用户只能查看到自己账户所对应的会话状态。即其他账户的state信息为空。
resource_pool	name	用户使用的资源池。
stmt_type	text	语句类型。
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

 版权所有 © 华为云计算技术有限公司

18.3.140 PG_STAT_ALL_INDEXES

PG_STAT_ALL_INDEXES视图显示当前数据库中所有访问特定索引的统计信息。

索引可以通过简单的索引扫描或“位图”索引扫描进行使用。位图扫描中几个索引的输出可以通过AND或者OR规则进行组合， 因此当使用位图扫描的时候，很难将独立对行抓取与特定索引进行组合， 因此，一个位图扫描增加
pg_stat_all_indexes.idx_tup_read使用索引计数，并且增加
pg_stat_all_tables.idx_tup_fetch表计数，但不影响
pg_stat_all_indexes.idx_tup_fetch。

表 18-211 PG_STAT_ALL_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_scan	bigint	索引上开始的索引扫描数。
idx_tup_read	bigint	通过索引上扫描返回的索引项数。
idx_tup_fetch	bigint	通过使用索引的简单索引扫描抓取的活表行数。

18.3.141 PG_STAT_ALL_TABLES

PG_STAT_ALL_TABLES视图显示当前数据库中与表访问相关的统计信息（包括TOAST表）。

表 18-212 PG_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
vacuum_count	bigint	vacuum次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。
autoanalyze_count	bigint	autoanalyze次数。
last_data_changed	timestamp with time zone	记录该表最后一次数据发生变化的时间（引起数据变化的操作包括INSERT/UPDATE/DELETE、EXCHANGE/TRUNCATE/DROP partition），该列数据仅在本地CN记录。

应用示例

查询表table_test最后一次数据发生变化的时间：

```
SELECT last_data_changed FROM PG_STAT_ALL_TABLES WHERE relname ='table_test';
      last_data_changed
-------------------------------
 2024-03-27 10:28:16.277136+08
(1 row)
```

 版权所有 © 华为云计算技术有限公司

18.3.142 PG_STAT_BAD_BLOCK

PG_STAT_BAD_BLOCK视图显示自节点启动后，读取数据时出现Page/CU校验失败的统计信息。

表 18-213 PG_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称。
databaseid	integer	数据库OID。
tablespaceid	integer	表空间OID。
relfilenode	integer	文件对象ID。
forknum	integer	文件类型。
error_count	integer	出现校验失败的次数。
first_time	timestamp with time zone	第一次出现时间。
last_time	timestamp with time zone	最近一次出现时间。

18.3.143 PG_STAT_BGWRITER

PG_STAT_BGWRITER视图显示关于后端写进程活动的统计信息。

表 18-214 PG_STAT_BGWRITER 字段

名称	类型	描述
checkpoints_timed	bigint	定期执行的检查点数量。
checkpoints_req	bigint	请求执行的检查点数量。
checkpoint_write_time	double precision	检查点期间将文件写入磁盘花费的时间，以毫秒为单位。
checkpoint_sync_time	double precision	检查点期间数据同步到磁盘花费的时间，以毫秒为单位。
buffers_checkpoint	bigint	检查点期间写入缓冲区的数量。
buffers_clean	bigint	后端写进程写的缓冲区数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
maxwritten_clean	bigint	由于写入缓冲区太多，后端写进程停止清理扫描的次数。
buffers_backend	bigint	后端直接写入的缓冲区数量。
buffers_backend_fsync	bigint	后端需要fsync的次数。
buffers_alloc	bigint	分配的缓冲区数量。
stats_reset	timestamp with time zone	统计重置的时间。

18.3.144 PG_STAT_DATABASE

PG_STAT_DATABASE视图显示当前节点上每个数据库的状态和统计信息。

表 18-215 PG_STAT_DATABASE 字段

名称	类型	描述
datid	oid	数据库OID。
datname	name	数据库名。
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。
tup_returned	bigint	当前节点上该数据库查询返回的行数。
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。
tup_inserted	bigint	当前节点上该数据库插入的行数。
tup_updated	bigint	当前节点上该数据库更新的行数。
tup_deleted	bigint	当前节点上该数据库删除的行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。可参见PG_STAT_DATABASE_CONFLICTS。
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。

18.3.145 PG_STAT_DATABASE_CONFLICTS

PG_STAT_DATABASE_CONFLICTS视图显示数据库冲突状态的统计信息。

表 18-216 PG_STAT_DATABASE_CONFLICTS 字段

名称	类型	描述
datid	oid	数据库OID。
datname	name	数据库名。
confl_tablespace	bigint	冲突的表空间的数目。
confl_lock	bigint	冲突的锁数目。
confl_snapshot	bigint	冲突的快照数目。
confl_bufferpin	bigint	冲突的缓冲区数目。
confl_deadlock	bigint	冲突的死锁数目。

版权所有 © 华为云计算技术有限公司

18.3.146 PG_STATS_EXT_EXPRS

PG_STATS_EXT_EXPRS视图提供对存储在PG_STATISTIC_DATA表里面的表达式的统计信息。该视图仅8.2.1.200及以上集群版本支持。

表 18-217 PG_STATS_EXT_EXPRS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
statistics_schemaname	name	-	表达式统计信息对象所在模式名。
statistics_name	name	-	表达式统计信息对象的名称。
statistics_owner	name	-	表达式统计信息对象的所有者。
expr	text	-	表达式
inherited	boolean	-	如果为true，则包含继承的子列；如果为false，则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。

 版权所有 © 华为云计算技术有限公司

18.3.147 PG_STAT_GET_MEM_MBYTES_RESERVED

PG_STAT_GET_MEM_MBYTES_RESERVED视图显示线程在内存中保存的当前活动信息。该函数在调用时需要指定线程ID，线程ID的选取请参考**PG_STAT_ACTIVITY**中的pid，线程ID为0时表示选取当前线程ID，例如：

```
SELECT pg_stat_get_mem_mbytes_reserved(0);
```

表 18-218 PG_STAT_GET_MEM_MBYTES_RESERVED 信息

名称	描述
ConnectInfo	连接信息。
ParctlManager	并发管理信息。
GeneralParams	基本参数信息。
GeneralParams RPDATA	基本资源池信息。
ExceptionManager	异常管理信息。
CollectInfo	收集信息。
GeneralInfo	基本信息。
ParctlState	并发状态信息。
CPU INFO	CPU信息。
ControlGroup	控制组信息。
IOSTATE	IO状态信息。

18.3.148 PG_STAT_USER_FUNCTIONS

PG_STAT_USER_FUNCTIONS视图显示命名空间中用户自定义函数（函数语言为非内部语言）的状态信息。

表 18-219 PG_STAT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID。
schemaname	name	模式名。
funcname	name	函数名。
calls	bigint	函数被调用的次数。
total_time	double precision	函数的总执行时长。
self_time	double precision	当前线程调用函数的总的时长。

版权所有 © 华为云计算技术有限公司

18.3.149 PG_STAT_USER_INDEXES

PG_STAT_USER_INDEXES视图显示数据库中用户自定义普通表和TOAST表的索引状态信息。

表 18-220 PG_STAT_USER_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_scan	bigint	通过索引扫描的次数。
idx_tup_read	bigint	通过索引上扫描返回的索引条目数量。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。

18.3.150 PG_STAT_USER_TABLES

PG_STAT_USER_TABLES视图显示所有命名空间中用户自定义普通表和TOAST表的状态信息。

表 18-221 PG_STAT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上表启动的顺序扫描的次数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum的次数。
analyze_count	bigint	analyze的次数。
autoanalyze_count	bigint	autoanalyze的次数。

18.3.151 PG_STAT_REPLICATION

PG_STAT_REPLICATION视图用于描述日志同步状态信息，如发起端发送日志位置，收端接收日志位置等。

表 18-222 PG_STAT_REPLICATION 字段

名称	类型	描述
pid	bigint	线程的PID。
usesysid	oid	用户系统ID。
usename	name	用户名。
application_name	text	程序名称。
client_addr	inet	客户端地址。
client_hostname	text	客户端名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
client_port	integer	客户端端口号。
backend_start	timestamp with time zone	程序启动时间。
state	text	日志复制的状态（追赶状态，还是一致的流状态）。
sender_sent_location	text	发送端发送日志位置。
receiver_write_location	text	接收端write日志位置。
receiver_flush_location	text	接收端flush日志位置。
receiver_replay_location	text	接收端replay日志位置。
sync_priority	integer	同步复制的优先级（0表示异步）。
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步）。

18.3.152 PG_STAT_SYS_INDEXES

PG_STAT_SYS_INDEXES视图显示pg_catalog、information_schema模式中所有系统表的索引状态信息。

表 18-223 PG_STAT_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_scan	bigint	通过索引扫描的次数。
idx_tup_read	bigint	通过索引上扫描返回的索引条目数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。

版权所有 © 华为云计算技术有限公司

18.3.153 PG_STAT_SYS_TABLES

PG_STAT_SYS_TABLES视图显示pg_catalog、information_schema模式的所有命名空间中系统表的统计信息。

表 18-224 PG_STAT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上表启动的顺序扫描的次数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次analyze时间。
last_autoanalyze	timestamp with time zone	最后一次autoanalyze时间。
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum的次数。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
analyze_count	bigint	analyze的次数。
autoanalyze_count	bigint	autoanalyze的次数。

18.3.154 PG_STAT_XACT_ALL_TABLES

PG_STAT_XACT_ALL_TABLES视图显示命名空间中所有普通表和TOAST表的事务状态信息。

表 18-225 PG_STAT_XACT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在此表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

18.3.155 PG_STAT_XACT_SYS_TABLES

PG_STAT_XACT_SYS_TABLES视图显示命名空间中系统表的事务状态信息。

表 18-226 PG_STAT_XACT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在此表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。
n_tup_ins	bigint	插入行数。
n_tup_upd	bigint	更新行数。
n_tup_del	bigint	删除行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

18.3.156 PG_STAT_XACT_USER_FUNCTIONS

PG_STAT_XACT_USER_FUNCTIONS视图显示关于函数执行的统计信息。

表 18-227 PG_STAT_XACT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID。
schemaname	name	模式名。
funcname	name	函数名。
calls	bigint	函数被调用的次数。
total_time	double precision	函数的总执行时长。
self_time	double precision	当前线程调用函数的总的时长。

18.3.157 PG_STAT_XACT_USER_TABLES

PG_STAT_XACT_USER_TABLES视图显示命名空间中用户表的事务状态信息。

表 18-228 PG_STAT_XACT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
relname	name	表名。
seq_scan	bigint	在该表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在该表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。
n_tup_ins	bigint	插入行数。
n_tup_upd	bigint	更新行数。
n_tup_del	bigint	删除行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

18.3.158 PG_STATIO_ALL_INDEXES

PG_STATIO_ALL_INDEXES视图显示当前数据库所有索引的I/O的统计信息。

表 18-229 PG_STATIO_ALL_INDEXES 字段

名称	类型	描述
relid	oid	索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引缓冲区命中数量。

18.3.159 PG_STATIO_ALL_SEQUENCES

PG_STATIO_ALL_SEQUENCES视图显示当前数据库中相关的序列信息以及指定序列的I/O的统计信息。

表 18-230 PG_STATIO_ALL_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列缓冲区命中数量。

18.3.160 PG_STATIO_ALL_TABLES

PG_STATIO_ALL_TABLES视图显示当前数据库中所有表的I/O的统计信息（包括TOAST表）。

表 18-231 PG_STATIO_ALL_TABLES 字段

名称	类型	描述
relid	oid	表OID。
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从此表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引命中缓冲区数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

18.3.161 PG_STATIO_SYS_INDEXES

PG_STATIO_SYS_INDEXES视图显示命名空间中所有系统表索引的IO状态信息。

版权所有 © 华为云计算技术有限公司

表 18-232 PG_STATIO_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引的模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引缓冲区命中数量。

18.3.162 PG_STATIO_SYS_SEQUENCES

PG_STATIO_SYS_SEQUENCES视图显示命名空间中所有系统表为序列的IO状态信息。

表 18-233 PG_STATIO_SYS_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列缓冲区命中数量。

18.3.163 PG_STATIO_SYS_TABLES

PG_STATIO_SYS_TABLES视图显示命名空间中所有系统表的IO状态信息。

表 18-234 PG_STATIO_SYS_TABLES 字段

名称	类型	描述
relid	oid	表OID。
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引命中缓冲区数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

18.3.164 PG_STATIO_USER_INDEXES

PG_STATIO_USER_INDEXES视图显示命名空间中所有用户关系表索引的IO状态信息。

表 18-235 PG_STATIO_USER_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID。
indexrelid	oid	该索引的OID。
schemaname	name	该索引的模式名。
relname	name	该索引的表名。
indexrelname	name	索引名称。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引命中缓冲区数。

18.3.165 PG_STATIO_USER_SEQUENCES

PG_STATIO_USER_SEQUENCES视图显示命名空间中所有用户关系表类型为序列的IO状态信息。

表 18-236 PG_STATIO_USER_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列中缓存命中数。

18.3.166 PG_STATIO_USER_TABLES

PG_STATIO_USER_TABLES视图显示命名空间中所有用户关系表的IO状态信息。

表 18-237 PG_STATIO_USER_TABLES 字段

名称	类型	描述
relid	oid	表OID。
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从该表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引缓冲区命中数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

18.3.167 PG_THREAD_WAIT_STATUS

通过PG_THREAD_WAIT_STATUS视图可以检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。

表 18-238 PG_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见**表18-239**。
wait_event	text	如果wait_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则为空。

wait_status列的等待状态有以下状态。

表 18-239 等待状态列表

wait_status值	含义
none	未等任意事件。
acquire lock	等待加锁，要么加锁成功，要么加锁等待超时。
acquire lwlock	等待获取轻量级锁。
wait io	等待IO完成。
wait cmd	等待完成读取网络通信包。
wait pooler get conn	等待pooler完成获取连接。
wait pooler abort conn	等待pooler完成终止连接。
wait pooler clean conn	等待pooler完成清理连接。
pooler create conn: [nodename], total N	等待pooler建立连接，当前正在与nodename指定节点建立连接，且仍有N个连接等待建立。
get conn	获取到其他节点的连接。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
set cmd: [nodename]	在连接上执行SET/RESET/TRANSACTION BLOCK LEVEL PARA SET/SESSION LEVEL PARA SET，当前正在nodename指定节点上执行。
cancel query	取消某连接上正在执行的SQL语句。
stop query	停止某连接上正在执行的查询。
wait node: [nodename](plevel), total N, [phase]	等待接收与某节点的连接上的数据，当前正在等待nodename节点plevel线程的数据，且仍有N个连接的数据待返回。如果状态包含phase信息，则可能的阶段状态有： ● begin：表示处于事务开始阶段。 ● commit：表示处于事务提交阶段。 ● rollback：表示处于事务回滚阶段。
wait transaction sync: xid	等待xid指定事务同步。
wait wal sync	等待特定LSN的wal log完成到备机的同步。
wait data sync	等待完成数据页到备机的同步。
wait data sync queue	等待把行存的数据页或列存的CU放入同步队列。
flush data: [nodename](plevel), [phase]	等待向网络中nodename指定节点的plevel对应线程发送数据。如果状态包含phase信息，则可能的阶段状态为wait quota，即当前通信流正在等待quota值。
stream get conn: [nodename], total N	初始化stream flow时，等待与nodename节点的consumer对象建立连接，且当前有N个待建连对象。
wait producer ready: [nodename](plevel), total N	初始化stream flow时，等待每个producer都准备好，当前正在等待nodename节点plevel对应线程的producer对象准备好，且仍有N个producer对象处于等待状态。
synchronize quit	stream plan结束时，等待stream线程组内的线程统一退出。
nodegroup destroy	stream plan结束时，等待销毁stream node group。
wait active statement	等待作业执行，正在资源负载管控中。
wait global queue	等待作业执行，正在全局队列排队。
wait respool queue	等待作业执行，正在资源池上排队。
wait ccn queue	等待作业执行，正在中心协调节点(CCN)中排队。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
gtm connect	等待与GTM建连。
gtm get gxid	等待从GTM获取事务xid。
gtm get snapshot	等待从GTM获取事务快照snapshot。
gtm begin trans	等待GTM开始事务。
gtm commit trans	等待GTM提交事务。
gtm rollback trans	等待GTM执行事务回滚。
gtm create sequence	等待GTM创建sequence。
gtm alter sequence	等待GTM修改sequence。
gtm get sequence val	等待从GTM获取sequence的下一个值。
gtm set sequence val	等待GTM设置sequence的值。
gtm drop sequence	等待GTM删除sequence。
gtm rename sequence	等待GTM重命名sequence。
analyze: [relname], [phase]	当前正在对表relname执行analyze。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的analyze分析操作。
vacuum: [relname], [phase]	当前正在对表relname执行vacuum。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的vacuum清理操作。
vacuum full: [relname]	当前正在对表relname执行vacuum full清理。
create index	当前正在创建索引。
HashJoin - [build hash | write file]	当前是HashJoin算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashJoin算子正在建立哈希表。 ● write file：表示当前HashJoin算子正在将数据写入磁盘。
HashAgg - [build hash | write file]	当前是HashAgg算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashAgg算子正在建立哈希表。 ● write file：表示当前HashAgg算子正在将数据写入磁盘。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
HashSetop - [build hash | write file]	当前是HashSetop算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashSetop算子正在建立哈希表。 ● write file：表示当前HashSetop算子正在将数据写入磁盘。
Sort | Sort - write file	当前是Sort算子做排序，write file表示Sort算子正在将数据写入磁盘。
Material | Material - write file	当前是Material算子，write file表示Material算子正在将数据写入磁盘。
wait sync consumer next step	Consumer接收端同步等待下一轮迭代。
wait sync producer next step	Producer发送端同步等待下一轮迭代。
wait agent release	正在释放当前agent(8.1.2及以上版本支持)。
wait stream task	stream线程等待被复用(8.1.2及以上版本支持)。

当wait_status为acquire lwlock、acquire lock或者wait io时，表示有等待事件。正在等待获取wait_event列对应类型的轻量级锁、事务锁，或者正在进行IO。

其中，wait_status值为acquire lwlock（轻量级锁）时对应的wait_event等待事件类型与描述信息如下。（wait_event为extension时，表示此时的轻量级锁是动态分配的锁，未被监控。）

表 18-240 轻量级锁等待事件列表

wait_event类型	类型描述
ShmemIndexLock	用于保护共享内存中的主索引哈希表。
OidGenLock	用于避免不同线程产生相同的OID。
XidGenLock	用于避免两个事务获得相同的xid。
ProcArrayLock	用于避免并发访问或修改ProcArray共享数组。
SInvalReadLock	用于避免与清理失效消息并发执行。
SInvalWriteLock	用于避免与其它写失效消息、清理失效消息并发执行。
WALInsertLock	用于避免与其它WAL插入操作并发执行。
WALWriteLock	用于避免并发WAL写盘。
ControlFileLock	用于避免pg_control文件的读写并发、写写并发。
CheckpointLock	用于避免多个checkpoint并发执行。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
CLogControlLock	用于避免并发访问或者修改Clog控制数据结构。
MultiXactGenLock	用于串行分配唯一MultiXact id。
MultiXactOffsetControlLock	用于避免对pg_multixact/offset的写写并发和读写并发。
MultiXactMemberControlLock	用于避免对pg_multixact/members的写写并发和读写并发。
RelCacheInitLock	用于失效消息场景对init文件进行操作时加锁。
CheckpointerCommLock	用于向checkpointer发起文件刷盘请求场景，需要串行的向请求队列插入请求结构。
TwoPhaseStateLock	用于避免并发访问或者修改两阶段信息共享数组。
TablespaceCreateLock	用于确定tablespace是否已经存在。
BtreeVacuumLock	用于防止vacuum清理B-tree中还在使用的页面。
AutovacuumLock	用于串行化访问autovacuum worker数组。
AutovacuumScheduleLock	用于串行化分配需要vacuum的table。
SyncScanLock	用于确定heap扫描时某个relfilenode的起始位置。
NodeTableLock	用于保护存放CN和DN节点信息的共享结构。
PoolerLock	用于保证两个线程不会同时从连接池里取到相同的连接。
RelationMappingLock	用于等待更新系统表到存储位置之间映射的文件。
AsyncCtlLock	用于避免并发访问或者修改共享通知状态。
AsyncQueueLock	用于避免并发访问或者修改共享通知信息队列。
SerializableXactHashLock	用于避免对于可串行事务共享结构的写写并发和读写并发。
SerializableFinishedListLock	用于避免对于已完成可串行事务共享链表的写写并发和读写并发。
SerializablePredicateLockListLock	用于保护对于可串行事务持有的锁链表。
OldSerXidLock	用于保护记录冲突可串行事务的结构。
FileStatLock	用于保护存储统计文件信息的数据结构。
SyncRepLock	用于在主备复制时保护xlog同步信息。
DataSyncRepLock	用于在主备复制时保护数据页同步信息。
CStoreColspaceCacheLock	用于保护列存表的CU空间分配。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
CStoreCUCacheSweepLock	用于列存CU Cache循环淘汰。
MetaCacheSweepLock	用于元数据循环淘汰。
DfsConnectorCacheLock	用于保护缓存HDFS连接的句柄的全局哈希表。
dummyServerInfoCacheLock	用于保护缓存加速集群连接信息的全局哈希表。
ExtensionConnectorLibLock	用于初始化ODBC连接场景，在加载与卸载特定动态库时进行加锁。
SearchServerLibLock	用于GPU加速场景初始化加载特定动态库时，对读文件操作进行加锁。
DfsUserLoginLock	用于保护HDFS用户信息的全局链表。
DfsSpaceCacheLock	用于控制HDFS表导入时文件ID单调递增。
LsnXlogChkFileLock	用于串行更新特定结构中记录的主备机的xlog flush位置点。
GTMHostInfoLock	用于避免并发访问或者修改GTM主机信息。
ReplicationSlotAllocationLock	用于主备复制时保护主机端的流复制槽的分配。
ReplicationSlotControlLock	用于主备复制时避免并发更新流复制槽状态。
ResourcePoolHashLock	用于避免并发访问或者修改资源池哈希表。
WorkloadStatHashLock	用于避免并发访问或者修改包含CN侧的SQL请求构成的哈希表。
WorkloadIoStatHashLock	用于避免并发访问或者修改用于统计当前DN的IO信息的哈希表。
WorkloadCGroupHashLock	用于避免并发访问或者修改Cgroup信息构成的哈希表。
OBSGetPathLock	用于避免对obs路径的写写并发和读写并发。
WorkloadUserInfoLock	用于避免并发访问或修改负载管理的用户信息哈希表。
WorkloadRecordLock	用于避免并发访问或修改在内存自适应管理时对CN收到请求构成的哈希表。
WorkloadIOUtilLock	用于保护记录iostat，CPU等负载信息的结构。
WorkloadNodeGroupLock	用于避免并发访问或者修改内存中的nodegroup信息构成的哈希表。
JobShmemLock	用于MPP兼容ORACLE定时任务功能中保护定时读取的全局变量。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
OBSRuntimeLock	用于获取环境变量，如GAUSSHOME。
LLVMDumpIRLock	用于导出动态生成函数所对应的汇编语言。
LLVMParseIRLock	用于在查询开始处从IR文件中编译并解析已写好的IR函数。
RPNumberLock	用于加速集群的DN对正在执行计划的任务线程的计数。
ClusterRPLock	用于加速集群的CCN中维护的集群负载数据的并发存取控制。
CriticalCacheBuildLock	用于从共享或者本地缓存初始化文件中加载cache的场景。
WaitCountHashLock	用于保护用户语句计数功能场景中的共享结构。
BufMappingLock	用于保护对共享缓冲映射表的操作。
LockMgrLock	用于保护常规锁结构信息。
PredicateLockMgrLock	用于保护可串行事务锁结构信息。
OperatorRealTLock	用于避免并发访问或者修改记录算子级实时数据的全局结构。
OperatorHistLock	用于避免并发访问或者修改记录算子级历史数据的全局结构。
SessionRealTLock	用于避免并发访问或者修改记录query级实时数据的全局结构。
SessionHistLock	用于避免并发访问或者修改记录query级历史数据的全局结构。
CacheSlotMappingLock	用于保护CU Cache全局信息。
BarrierLock	用于保证当前只有一个线程在创建Barrier。

当wait_status值为wait io时对应的wait_event等待事件类型与描述信息如下。

表 18-241 IO 等待事件列表

wait_event类型	类型描述
BufFileRead	从临时文件中读取数据到指定buffer。
BufFileWrite	向临时文件中写入指定buffer中的内容。
ControlFileRead	读取pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
ControlFileSync	将pg_control文件持久化到磁盘。数据库初始化时发生。
ControlFileSyncUpdate	将pg_control文件持久化到磁盘。主要在数据库启动、执行checkpoint和主备校验过程中发生。
ControlFileWrite	写入pg_control文件。数据库初始化时发生。
ControlFileWriteUpdate	更新pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。
CopyFileRead	copy文件时读取文件内容。
CopyFileWrite	copy文件时写入文件内容。
DataFileExtend	扩展文件时向文件写入内容。
DataFileFlush	将表数据文件持久化到磁盘
DataFileImmediateSync	将表数据文件立即持久化到磁盘。
DataFilePrefetch	异步读取表数据文件。
DataFileRead	同步读取表数据文件。
DataFileSync	将表数据文件的修改持久化到磁盘。
DataFileTruncate	表数据文件truncate。
DataFileWrite	向表数据文件写入内容。
LockFileAddToDataDirRead	读取"postmaster.pid"文件。
LockFileAddToDataDirSync	将"postmaster.pid"内容持久化到磁盘。
LockFileAddToDataDirWrite	将pid信息写到"postmaster.pid"文件。
LockFileCreateRead	读取LockFile文件"%s.lock"。
LockFileCreateSync	将LockFile文件"%s.lock"内容持久化到磁盘。
LockFileCreateWRITE	将pid信息写到LockFile文件"%s.lock"。
RelationMapRead	读取系统表到存储位置之间的映射文件
RelationMapSync	将系统表到存储位置之间的映射文件持久化到磁盘。
RelationMapWrite	写入系统表到存储位置之间的映射文件。
ReplicationSlotRead	读取流复制槽文件。重新启动时发生。
ReplicationSlotRestoreSync	将流复制槽文件持久化到磁盘。重新启动时发生。
ReplicationSlotSync	checkpoint时将流复制槽临时文件持久化到磁盘。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
ReplicationSlotWrite	checkpoint时写流复制槽临时文件。
SLRUFlushSync	将pg_clog、pg_subtrans和pg_multixact文件持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRURead	读取pg_clog、pg_subtrans和pg_multixact文件。
SLRUSync	将脏页写入文件pg_clog、pg_subtrans和pg_multixact并持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRUWrite	写入pg_clog、pg_subtrans和pg_multixact文件。
TimelineHistoryRead	读取timeline history文件。在数据库启动时发生。
TimelineHistorySync	将timeline history文件持久化到磁盘。在数据库启动时发生。
TimelineHistoryWrite	写入timeline history文件。在数据库启动时发生。
TwophaseFileRead	读取pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileSync	将pg_twophase文件持久化到磁盘。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileWrite	写入pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
WALBootstrapSync	将初始化的WAL文件持久化到磁盘。在数据库初始化发生。
WALBootstrapWrite	写入初始化的WAL文件。在数据库初始化发生。
WALCopyRead	读取已存在的WAL文件并进行复制时产生的读操作。在执行归档恢复完后发生。
WALCopySync	将复制的WAL文件持久化到磁盘。在执行归档恢复完后发生。
WALCopyWrite	读取已存在WAL文件并进行复制时产生的写操作。在执行归档恢复完后发生。
WALInitSync	将新初始化的WAL文件持久化磁盘。在日志回收或写日志时发生。
WALInitWrite	将新创建的WAL文件初始化为0。在日志回收或写日志时发生。
WALRead	从xlog日志读取数据。两阶段文件redo相关的操作产生。
WALSyncMethodAssign	将当前打开的所有WAL文件持久化到磁盘。
WALWrite	写入WAL文件。

 版权所有 © 华为云计算技术有限公司

当wait_status值为acquire lock（事务锁）时对应的wait_event等待事件类型与描述信息如下。

表 18-242 事务锁等待事件列表

wait_event类型	类型描述
relation	对表加锁。
extend	对表扩展空间时加锁。
partition	对分区表加锁。
partition_seq	对分区表的分区加锁。
page	对表页面加锁。
tuple	对页面上的tuple加锁。
transactionid	对事务ID加锁。
virtualxid	对虚拟事务ID加锁。
object	加对象锁。
cstore_freespace	对列存空闲空间加锁。
userlock	加用户锁。
advisory	加advisory锁。

18.3.168 PG_TABLES

PG_TABLES视图提供了对数据库中每个表访问的有用信息。

表 18-243 PG_TABLES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
tableowner	name	pg_get_userbyid(PG_CLASS.relowner)	表的所有者。
tablespace	name	PG_TABLESPACE.spcname	包含表的表空间，默认为NULL。
hasindexes	boolean	PG_CLASS.relhasindex	如果表上有索引（或者最近拥有）则为TRUE，否则为FALSE。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
hasrules	boolean	PG_CLASS.relhasrules	如果表上有规则，则为TRUE，否则为FALSE。
hastriggers	boolean	PG_CLASS.RELHASTRIGGERS	如果表上有触发器，则为TRUE，否则为FALSE。
tablecreator	name	pg_get_userbyid(PG_OBJECT.creator)	表创建用户，如创建用户已删除，则返回空。
created	timestamp with time zone	PG_OBJECT.ctime	表创建时间。
last_ddl_time	timestamp with time zone	PG_OBJECT.mtime	表最后修改时间。

应用示例

查询指定模式下所有的表。

```
SELECT tablename FROM PG_TABLES WHERE schemaname = 'myschema';
  tablename
----------------
 inventory
 product
 sales_info
 test1
 mytable
 product_info
 customer_info
 newproducts
 customer_t1
(9 rows)
```

18.3.169 PG_TDE_INFO

PG_TDE_INFO视图提供了当前集群加密信息。

表 18-244 PG_TDE_INFO 字段

名称	类型	描述
is_encrypt	text	是否加密集群。 • f：非加密集群。 • t：加密集群。
g_tde_algo	text	加密算法。 • SM4-CTR-128 • AES-CTR-128
remain	text	保留字段。

版权所有 © 华为云计算技术有限公司

应用示例

查看当前集群是否加密/查看当前集群的加密算法：

```
SELECT * FROM PG_TDE_INFO;
 is_encrypt | g_tde_algo  | remain
------------+-------------+--------
 f          | AES-CTR-128 | remain
(1 row)
```

18.3.170 PG_TIMEZONE_ABBREVS

PG_TIMEZONE_ABBREVS视图提供了输入例程能够识别的所有时区缩写。

表 18-245 PG_TIMEZONE_ABBREVS 字段

名称	类型	描述
abbrev	text	时区缩写。
utc_offset	interval	相对于UTC的偏移量。
is_dst	boolean	如果这是一个夏时制时区缩写则为TRUE，否则为FALSE。

18.3.171 PG_TIMEZONE_NAMES

PG_TIMEZONE_NAMES视图提供了显示了所有能够被SET TIMEZONE识别的时区名及其缩写、UTC偏移量、是否夏时制。

表 18-246 PG_TIMEZONE_NAMES 字段

名称	类型	描述
name	text	时区名。
abbrev	text	时区名缩写。
utc_offset	interval	相对于UTC的偏移量。
is_dst	boolean	如果当前正处于夏令时范围则为TRUE，否则为FALSE。

18.3.172 PG_TOTAL_MEMORY_DETAIL

PG_TOTAL_MEMORY_DETAIL视图显示某个数据库节点内存使用情况。

表 18-247 PG_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存的名称，包括以下几种： ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● max_topsql_memory：TopSQL记录历史作业监控信息允许使用的最大内存。 ● topsql_used_memory：TopSQL记录历史作业监控信息已使用的内存大小。 ● topsql_peak_memory：TopSQL记录历史作业监控信息的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● gpu_max_dynamic_memory：GPU内存最大值。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● gpu_dynamic_peak_memory：GPU内存使用的最大内存。 ● pooler_conn_memory：pooler连接占用内存大小。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
		• pooler_freeconn_memory：pooler空闲连接占用的内存大小。 • storage_compress_memory：列存压缩和解压缩使用的内存大小。 • udf_reserved_memory：为UDF Worker进程预留的内存大小。 • mmap_used_memory：mmap使用的内存大小。
memorymbytes	integer	内存使用的大小，单位MB。

18.3.173 PG_TOTAL_SCHEMA_INFO

PG_TOTAL_SCHEMA_INFO视图显示各个数据库下所有Schema的存储资源使用情况。此视图在参数use_workload_manager为on时才有效。

名称	类型	描述
schemaid	oid	Schema的OID。
schemaname	text	Schema名称。
databaseid	oid	数据库的OID。
databasename	name	数据库名称。
usedspace	bigint	该Schema已使用的永久表存储空间大小，单位Byte。
permspace	bigint	该Schema的永久表存储空间上限值，单位Byte。

18.3.174 PG_TOTAL_USER_RESOURCE_INFO

PG_TOTAL_USER_RESOURCE_INFO视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在参数use_workload_manager为on时才有效。

表 18-248 PG_TOTAL_USER_RESOURCE_INFO 字段

名称	类型	描述
username	name	用户名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_memory	integer	用户使用的内存大小，单位：MB。 ● DN：显示当前DN上用户使用的内存大小。 ● CN：显示所有DN上用户使用内存的累积和。
total_memory	integer	资源池使用的内存大小，单位：MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存max_dynamic_memory。具体的计算公式为： total_memory = max_dynamic_memory * parent_percent * user_percent CN：显示所有DN上用户可用内存上限的累积和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。
total_cpu	integer	在该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑读的字节总数。单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑写的字节总数。单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的次数之和。 DN：实例启动至当前时间为止，该用户逻辑读的次数之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的次数之和。 DN：实例启动至当前时间为止，该用户逻辑写的次数之和。
read_speed	double precision	CN：过去5秒内，该用户在所有DN上逻辑读平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑读平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在所有DN上逻辑写平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑写平均速率。单位KB/s。
send_speed	double precision	CN：过去5秒内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：过去5秒内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络接收平均速率。单位KB/s。

18.3.175 PG_USER

PG_USER视图提供了访问数据库用户的信息。

表 18-249 PG_USER 字段

名称	类型	描述
usename	name	用户名。
usesysid	oid	此用户的ID。
usecreatedb	boolean	用户是否可以创建数据库。
usesuper	boolean	用户是否是拥有最高权限的初始系统管理员。
usecatupd	boolean	用户是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
userepl	boolean	用户是否可以复制数据流。
passwd	text	密文存储后的用户口令，始终为********。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
valbegin	timestamp with time zone	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	用户所在的资源池。
parent	oid	父用户OID
spacelimit	text	永久表存储空间限额。
tempspacelimit	text	临时表存储空间限额。
spillspacelimit	text	算子落盘空间限额。
useconfig	text[]	运行时配置参数的会话缺省。
nodegroup	name	用户关联的逻辑集群名字，如果该用户没有管理逻辑集群，则该字段为空。

应用示例

查看当前数据库用户列表：

```
SELECT usename  FROM pg_user;
  usename
-----------
 dbadmin
 u1
 u2
 u3
(4 rows)
```

18.3.176 PG_USER_MAPPINGS

PG_USER_MAPPINGS视图提供访问关于用户映射的信息的接口。

这个视图只是一个PG_USER_MAPPING的可读部分的视图化表现，如果用户无权使用它则查询此表时，有些选项字段会显示为空。

表 18-250 PG_USER_MAPPINGS 字段

名字	类型	引用	描述
umid	oid	PG_USER_MAPPING.oid	用户映射的OID。
srvid	oid	PG_FOREIGN_SERVER.oid	包含这个映射的外部服务器的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
srvname	name	PG_FOREIGN_SERVER.srvname	外部服务器的名字。
umuser	oid	PG_AUTHID.oid	被映射的本地角色的OID，如果用户映射是公共的则为0。
usename	name	-	被映射的本地用户的名字。
umoptions	text[]	-	如果当前用户是外部服务器的所有者，则为用户映射指定选项， 使用“keyword=value”字符串，否则为null。

18.3.177 PG_VIEWS

PG_VIEWS视图提供访问数据库中每个视图的有用信息。

表 18-251 PG_VIEWS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含视图的模式名。
viewname	name	PG_CLASS.relname	视图的名称。
viewowner	name	PG_AUTHID.rolname	视图的所有者。
definition	text	-	视图的定义。

应用示例

查询指定模式下的所有视图：

```
SELECT * FROM pg_views WHERE schemaname = 'myschema';
 schemaname | viewname | viewowner |                                                definition
------------+----------+-----------+-------------------------------------------------------------------------------------------------
 myschema   | myview   | dbadmin   | SELECT  * FROM pg_tablespace WHERE (pg_tablespace.spcname = 'pg_default'::name);
 myschema   | v1       | dbadmin   | SELECT  * FROM t1 WHERE (t1.c1 > 200);
(2 rows)
```

18.3.178 PG_WLM_STATISTICS

PG_WLM_STATISTICS视图显示作业结束后或已被处理异常后的负载管理相关信息。该视图8.1.2版本中已废弃，可使用PGXC_WLM_SESSION_INFO视图查看所有CN上执行作业结束后的负载管理记录。

版权所有 © 华为云计算技术有限公司

表 18-252 PG_WLM_STATISTICS 字段

名称	类型	描述
statement	text	执行了异常处理的语句。
block_time	bigint	语句执行前的阻塞时间。
elapsed_time	bigint	语句的实际执行时间。
total_cpu_time	bigint	语句执行异常处理时DN上CPU使用的总时间。
qualification_time	bigint	语句检查倾斜率的时间周期。
cpu_skew_percent	integer	语句在执行异常处理时DN上CPU使用的倾斜率。
control_group	text	语句执行异常处理时所使用的Cgroups。
status	text	语句执行异常处理后的状态，包括： ● pending：执行前预备状态。 ● running：执行进行状态。 ● finished：执行正常结束。 ● abort：执行异常终止。
action	text	语句执行的异常处理动作，包括： ● abort：执行终止操作。 ● adjust：执行Cgroups调整操作，目前只有降级操作。 ● finish：正常结束。
queryid	bigint	语句执行所使用的内部query_id。
threadid	bigint	后端线程ID。

18.3.179 PGXC_BULKLOAD_PROGRESS

PGXC_BULKLOAD_PROGRESS显示导入业务的执行进度，仅支持GDS普通文件导入业务。需要有系统管理员权限才可以访问此视图

表 18-253 PGXC_BULKLOAD_PROGRESS 字段

名称	类型	描述
session_id	bigint	GDS的会话ID。
query_id	bigint	查询ID，对应debug_query_id。
query	text	查询语句。
progress	text	进度百分比。

版权所有 © 华为云计算技术有限公司

18.3.180 PGXC_BULKLOAD_INFO

通过CN查看PGXC_BULKLOAD_INFO视图，可获取互联互通、GDS、COPY、\COPY等业务执行结束后的历史统计信息。该视图汇总当前集群上各个节点已经执行结束的导入导出类业务的历史执行情况（包括互联互通集群地址、导入导出业务类型、DN上落盘的最大、最小以及总和的行数与字节数等），从而可以获取导入导出类业务执行的历史信息，辅助进行性能问题排查。

该视图不会记录异常中断的导入导出作业，数据直接从系统表GS_WLM_SESSION_INFO获取，并将loader_status字段解析获取导入导出类业务信息。

需要有系统管理员权限才可以访问此PGXC_BULKLOAD_INFO视图。

表 18-254 PGXC_BULKLOAD_INFO 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
address	text	互联互通对端集群server的address，非空时即为互联互通业务，源集群会额外获取远端集群端口号。
direction	text	导入导出业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
min_done_lines	json	语句在所有DN上的最小行数。
max_done_lines	json	语句在所有DN上的最大行数。
total_done_lines	json	语句在所有DN上的总行数。
min_done_bytes	json	语句在所有DN上的最小字节数。
max_done_bytes	json	语句在所有DN上的最大字节数。
total_done_bytes	json	语句在所有DN上的总字节数。

 版权所有 © 华为云计算技术有限公司

说明

- 异常中断导入导出类业务不会记录在视图中。
- gds外表与互联互通外表实现机制不同，查询时gds会记录全量，互联互通会记录实际。
- 非全量导入导出外表，带limit时会因limit执行计划的特殊，数据显示为从一个DN上收取，即表现为max值为全部，min值为0。
- 若导入导出表为非分区表时：
  - gds分区表较小的情况下，在一个DN获取完数据后，若其余DN未开始获取数据，将不会在其余DN获取数据，所以在gds from非分区表数据量较小的情况下，min值可能为0，表数据量大时不为0。
  - 互联互通源集群导出非分区表时，所有DN都会被记录，且只会拿取一个DN的数据，所以min值为0。
  - 互联互通远端集群导出复制表时，只有一个DN都会被记录，所以相当于只有一个DN，min值max值相同。
- 导入导出历史监控通过复用历史TopSQL功能实现，同样遵守TopSQL的注意事项、前提条件、操作步骤等，详情请见**历史TopSQL**。
- 因TopSQL记录数据量较大，为提高查询性能建议结合start_time、finish_time等字段按需查询使用，或者降低查询频率。

应用示例

使用PGXC_BULKLOAD_INFO视图查询互联互通导入业务：

```
SELECT * FROM PGXC_BULKLOAD_INFO;
 datid |  dbname  |  schemaname  |  nodename  |  username  | application_name | client_addr |
client_hostname | client_port | query_band | block_time |          start_time          |          finish_time         |
status
s |      queryid
|
query
                                                
```

18.3.181 PGXC_BULKLOAD_STATISTICS

通过CN查看PGXC_BULKLOAD_STATISTICS视图，可获取GDS、COPY、\COPY等业务执行过程中的实时统计信息。该视图汇总当前集群上各个节点正在执行的导入/导出类业务的实时执行情况，从而可以监控导入导出类业务的实时进度，辅助进行性能问题排查。

PGXC_BULKLOAD_STATISTICS视图与PG_BULKLOAD_STATISTICS视图列定义完全相同。这是由于PGXC_BULKLOAD_STATISTICS视图本质是到集群中各个节点上查询PG_BULKLOAD_STATISTICS视图汇总的结果。

需要有系统管理员权限才可以访问此视图。

表 18-255 PGXC_BULKLOAD_STATISTICS 字段

名称	类型	描述
node_name	text	节点名称。
db_name	text	数据库名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
session_id	bigint	GDS的会话ID。
direction	text	业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
query	text	查询语句。
address	text	当前导入导出外表的location。
query_start	timestamp with time zone	导入/导出开始时间。
total_bytes	bigint	待处理数据的总大小。 仅GDS普通文件导入时，且该行记录来自CN节点才会显示，否则为空。
phase	text	当前阶段，包括：INITIALIZING、TRANSFER_DATA、RELEASE_RESOURCE。
done_lines	bigint	已传输行数。
done_bytes	bigint	已传输字节数。

18.3.182 PGXC_COLUMN_TABLE_IO_STAT

PGXC_COLUMN_TABLE_IO_STAT视图提供集群所有CN和DN节点上当前数据库所有列存表的IO统计数据。除在每一行前面增加name类型的nodename字段外，其余字段的

 版权所有 © 华为云计算技术有限公司

名称、类型和顺序与GS_COLUMN_TABLE_IO_STAT视图相同，具体的字段请参考**表18-256**。

表 18-256 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

18.3.183 PGXC_COMM_CLIENT_INFO

PGXC_COMM_CLIENT_INFO视图存储所有节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

表 18-257 PGXC_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
query_id	bigint	查询ID，对应debug_query_id。
socket	integer	如果是物理连接，展示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，展示sid，显示-1时表示当前连接是物理连接。

版权所有 © 华为云计算技术有限公司

18.3.184 PGXC_COMM_DELAY

PGXC_COMM_DELAY视图展示所有DN的通信库时延状态。

表 18-258 PGXC_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称。
remote_name	text	连接对端节点的对端时延最大的节点名称。
remote_host	text	连接对端IP的对端地址。
stream_num	integer	当前物理连接使用的stream逻辑连接数量。
min_delay	integer	当前物理连接探测到的最小时延，单位微秒。
average	integer	当前物理连接探测时延的平均值，单位微秒。
max_delay	integer	当前物理连接探测到的最大时延，单位微秒。 **说明** 取值为-1，表示时延探测超时失败，请重新建立节点间连接后再执行查询。

18.3.185 PGXC_COMM_RECV_STREAM

PGXC_COMM_RECV_STREAM视图展示所有DN上的通信库接收流状态。

表 18-259 PGXC_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接接收报文正常。 ● HOLD：表示逻辑连接接收报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
recv_bytes	bigint	通信流接收的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均接收速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte。

18.3.186 PGXC_COMM_SEND_STREAM

PGXC_COMM_SEND_STREAM视图展示所有DN上的通信库发送流状态。

表 18-260 PGXC_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接发送报文正常。 ● HOLD：表示逻辑连接发送报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

18.3.187 PGXC_COMM_STATUS

PGXC_COMM_STATUS视图展示所有DN的通信库状态。

表 18-261 PGXC_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck/s	integer	节点通信库接收速率，单位Byte/s。
txpck/s	integer	节点通信库发送速率，单位Byte/s。
rxkB/s	bigint	节点通信库接收速率，单位KByte/s。
txkB/s	bigint	节点通信库发送速率，单位KByte/s。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位KByte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位KByte。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
%USED(PM)	integer	postmaster线程实时使用率。
%USED (sflow)	integer	gs_sender_flow_controller线程实时使用率。
%USED (rflow)	integer	gs_receiver_flow_controller线程实时使用率。
%USED (rloop)	integer	多个gs_receivers_loop线程中最高的实时使用率。
stream	integer	当前使用的逻辑连接总数。

18.3.188 PGXC_COMM_QUERY_SPEED

PGXC_COMM_QUERY_SPEED视图展示所有节点上所有query对应的流量信息。

表 18-262 PGXC_COMM_QUERY_SPEED 字段

名称	类型	描述
node_name	text	节点名称。
query_id	bigint	通信流对应的debug_query_id编号。
rxkB/s	bigint	查询对应的通信流接收速率，单位Byte/s。
txkB/s	bigint	查询对应的通信流发送速率，单位Byte/s。
rxkB	bigint	查询对应的通信流接收数据总量，单位Byte。
txkB	bigint	查询对应的通信流发送数据总量，单位Byte。
rxpck/s	bigint	查询对应的通信包接收速率，单位packages/s。
txpck/s	bigint	查询对应的通信包发送速率，单位packages/s。
rxpck	bigint	查询对应的通信包接收总量，单位packages。
txpck	bigint	查询对应的通信包发送总量，单位packages。

18.3.189 PGXC_DEADLOCK

PGXC_DEADLOCK视图获取导致分布式死锁产生的锁等待信息。

目前，PGXC_DEADLOCK视图只收集locktype为relation、partition、page、tuple和transactionid的锁等待信息。

版权所有 © 华为云计算技术有限公司

表 18-263 PGXC_DEADLOCK 字段

名称	类型	描述
locktype	text	被锁定对象的类型。
nodename	name	被锁定对象的节点名称。
dbname	name	被锁定对象的数据库名称。如果被锁定对象是事务，则为NULL。
nspname	name	被锁定对象的命名空间名称。
relname	name	被锁定对象对应的关系名称。如果被锁定对象既不是关系，也不是关系的一部分，则为NULL。
partname	name	被锁定对象对应的分区名称。如果被锁定对象不是分区，则为NULL。
page	integer	被锁定对象对应的页面编号。如果被锁定对象既不是页面，也不是元组，则为NULL。
tuple	smallint	被锁定对象对应的元组编号。如果被锁定对象不是元组，则为NULL。
transactionid	xid	被锁定对象对应的事务ID。如果被锁定对象不是事务，则为NULL。
waitusername	name	等待锁的用户名称。
waitgxid	xid	等待锁的事务ID。
waitxactstart	timestamp with time zone	等待锁的事务的开始时间。
waitqueryid	bigint	等待锁的线程的最新查询ID。
waitquery	text	等待锁的线程的最新查询语句。
waitpid	bigint	等待锁的线程ID。
waitmode	text	等待的锁的级别。
holdusername	name	持有锁的用户名称。
holdgxid	xid	持有锁的事务ID。
holdxactstart	timestamp with time zone	持有锁的事务的开始时间。
holdqueryid	bigint	持有锁的线程的最新查询ID。
holdquery	text	持有锁的线程的最新查询语句。
holdpid	bigint	持有锁的线程ID。
holdmode	text	持有锁的级别。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始持有锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

18.3.190 PGXC_DISK_CACHE_STATS

PGXC_DISK_CACHE_STATS视图记录了文件缓存的使用情况。该系统视图仅9.1.0及以上集群版本支持。

表 18-264 PGXC_DISK_CACHE_STATS 字段

名称	类型	描述
node_name	text	节点名称。
total_read	bigint	访问disk cache的总次数。
local_read	bigint	disk cache读本地磁盘的总次数。
remote_read	bigint	disk cache读远端存储的总次数。
hit_rate	numeric(5,2)	disk cache的命中率。
cache_size	bigint	disk cache保存的数据总大小，单位kbytes。
fill_rate	numeric(5,2)	disk cache的填充率。

示例

查询每个节点disk cache的命中率：

```
SELECT hit_rate FROM pgxc_disk_cache_stats;
 hit_rate
----------
    56.91
    56.85
      NaN
      NaN
      NaN
      NaN
(6 rows)
```

18.3.191 PGXC_DISK_CACHE_ALL_STATS

PGXC_DISK_CACHE_ALL_STATS视图记录文件缓存所有的使用情况。该系统视图仅9.1.0及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 18-265 PGXC_DISK_CACHE_ALL_STATS 字段

名称	类型	描述
node_name	text	节点名称。
total_read	bigint	访问disk cache的总次数。
local_read	bigint	disk cache访问本地磁盘的总次数。
remote_read	bigint	disk cache访问远端存储的总次数。
hit_rate	numeric(5,2)	disk cache的命中率。
cache_size	bigint	disk cache保存的数据总大小，单位kbytes。
fill_rate	numeric(5,2)	disk cache的填充率。
temp_file_size	bigint	临时/冷缓存文件的总大小(kbytes)。
a1in_size	bigint	disk cache中a1in队列保存的数据的总大小，单位kbytes。
a1out_size	bigint	disk cache中a1out队列保存的数据的总大小，单位kbytes。
am_size	bigint	disk cache中am队列保存的数据的总大小，单位kbytes。
a1in_fill_rate	numeric(5,2)	disk cache中a1in队列的填充率。
a1out_fill_rate	numeric(5,2)	disk cache中a1out队列的填充率。
am_fill_rate	numeric(5,2)	disk cache中am队列的填充率。
fd	integer	disk cache正在使用的文件描述符数量。
pin_block_count	bigint	disk cache中被pin住block的数量。该字段仅9.1.0.100及以上集群版本支持。

示例

查询disk cache在每个节点使用的文件描述符的数量：

```
SELECT fd FROM pgxc_disk_cache_all_stats;
  fd
------
 1000
 1000
    0
    0
    0
```

 版权所有 © 华为云计算技术有限公司

```
  0
(6 rows)
```

18.3.192 PGXC_DISK_CACHE_PATH_INFO

PGXC_DISK_CACHE_PATH_INFO视图记录了文件缓存所在的硬盘的信息。该系统视图仅9.1.0及以上集群版本支持。

表 18-266 PGXC_DISK_CACHE_PATH_INFO 字段

名称	类型	描述
path_name	text	路径名称。
node_name	text	硬盘所属的节点名称。
cache_size	bigint	硬盘中cache文件所占的总大小，单位bytes。
disk_available	bigint	硬盘的可用空间，单位bytes。
disk_size	bigint	硬盘的总容量，单位bytes。
disk_use_ratio	double precision	硬盘空间的使用率。

示例

查询文件缓存所使用的硬盘的信息：

```
SELECT * FROM pgxc_disk_cache_path_info order by 1;
   path_name    |  node_name   | cache_size | disk_available |  disk_size   |  disk_use_ratio
----------------+--------------+------------+----------------+--------------+------------------
 dn_6001_6002_0 | dn_6001_6002 |      19619 |   137401716736 | 160982630400 | .146481105479564
 dn_6001_6002_1 | dn_6001_6002 |      35968 |   137401716736 | 160982630400 | .146481105479564
 dn_6003_6004_0 | dn_6003_6004 |      27794 |   121600655360 | 160982630400 | .244634933235629
 dn_6003_6004_1 | dn_6003_6004 |      26158 |   121600655360 | 160982630400 | .244634933235629
 dn_6005_6006_0 | dn_6005_6006 |      24533 |   134394839040 | 160982630400 | .165159379579873
 dn_6005_6006_1 | dn_6005_6006 |      31065 |   134394839040 | 160982630400 | .165159379579873
```

18.3.193 PGXC_GET_STAT_ALL_TABLES

PGXC_GET_STAT_ALL_TABLES视图获取各表的插入、更新、删除以及脏页率信息。

对于高脏页率的系统表，建议在确认当前没有用户操作该系统表时，再执行VACUUM FULL。

建议对脏页率超过80%的非系统表执行VACUUM FULL，用户也可根据业务场景自行选择是否执行VACUUM FULL。

说明

8.2.0.100及以上集群版本，查询脏页率推荐使用**PGXC_STAT_TABLE_DIRTY**。

版权所有 © 华为云计算技术有限公司

表 18-267 PGXC_GET_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
relname	name	表名。
schemaname	name	表的模式名。
n_tup_ins	numeric	插入的元组条数。
n_tup_upd	numeric	更新的元组条数。
n_tup_del	numeric	删除的元组条数。
n_live_tup	numeric	live元组的条数。
n_dead_tup	numeric	dead元组的条数。
dirty_page_rate	numeric(5,2)	表的脏页率信息(%)。

详细内容可参考《 数据仓库服务SQL语法参考 》的“函数和操作符>系统管理函数>其他函数”章节。

应用示例

使用PGXC_GET_STAT_ALL_TABLES视图查询数据库内脏页率大于30%的表：

```
SELECT * FROM PGXC_GET_STAT_ALL_TABLES WHERE dirty_page_rate>30;
 relid |         relname          | schemaname | n_tup_ins | n_tup_upd | n_tup_del | n_live_tup | n_dead_tup |
dirty_page_rate
-------+--------------------------+------------+-----------+-----------+-----------+------------+------------
+-----------------
  2840 | pg_toast_2619            | pg_toast   |      7415 |         0 |      7415 |          0 |        291 |          88.00
  9001 | pgxc_class               | pg_catalog |     56331 |         3 |     56285 |         54 |        143 |          72.59
 53860 | reason                   | dbadmin    |         9 |        19 |         0 |          9 |         19 |          67.86
  9025 | pg_object                | pg_catalog |    112858 |   1179707 |    112619 |        246 |        429 |
63.56
  9015 | pgxc_node                | pg_catalog |        15 |        24 |         0 |         15 |         24 |          61.54
  2606 | pg_constraint            | pg_catalog |        78 |         0 |        42 |         36 |         42 |          53.85
  1260 | pg_authid                | pg_catalog |         6 |         6 |         0 |          6 |          6 |          50.00
(7 rows)
```

18.3.194 PGXC_GET_STAT_ALL_PARTITIONS

PGXC_GET_STAT_ALL_PARTITIONS视图获取各分区表分区的插入、更新、删除以及脏页率信息。

该视图的统计信息依赖于ANALYZE，为获取最准确的信息请先对分区表进行ANALYZE。

说明

8.2.0.100及以上集群版本，查询脏页率推荐使用**PGXC_STAT_TABLE_DIRTY**。

版权所有 © 华为云计算技术有限公司

表 18-268 PGXC_GET_STAT_ALL_PARTITIONS 字段

名称	类型	描述
relid	oid	表的OID。
partid	oid	分区的OID。
schemaname	name	表的模式名。
relname	name	表名。
partname	name	分区名。
n_tup_ins	numeric	插入的元组条数。
n_tup_upd	numeric	更新的元组条数。
n_tup_del	numeric	删除的元组条数。
n_live_tup	numeric	live元组的条数。
n_dead_tup	numeric	dead元组的条数。
page_dirty_rate	numeric(5,2)	表的脏页率信息(%)。

应用示例

查询数据库内脏页率大于30%的分区表：

```
SELECT * FROM PGXC_GET_STAT_ALL_PARTITIONS WHERE dirty_page_rate>30;
 relid | partid |   schemaname    |      relname       | partname | n_tup_ins | n_tup_upd | n_tup_del | n_live_tup
| n_dead_tup | dirty_page_rate
-------+--------+-----------------+--------------------+----------+-----------+-----------+-----------+------------
+------------+-----------------
 58320 |  58626 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |          0 |          2
|         100.00
 58430 |  58706 | schema_subquery | store_hash_par_mor | p4       |         1 |         1 |         1 |          0 |
2 |         100.00
 58320 |  58644 | schema_subquery | store_hash_par     | p1       |         3 |         0 |         3 |          0 |          3
|         100.00
 58430 |  58770 | schema_subquery | store_hash_par_mor | p4       |         1 |         1 |         1 |          0 |
2 |         100.00
 58320 |  58643 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |          0 |          2
|         100.00
 58320 |  58625 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |          0 |          2
|         100.00
 58320 |  58579 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |          0 |          2
|         100.00
 58320 |  58619 | schema_subquery | store_hash_par     | p1       |         3 |         0 |         3 |          0 |          3
|         100.00
 58320 |  58627 | schema_subquery | store_hash_par     | p1       |         4 |         0 |         4 |          0 |          4
|         100.00
 58320 |  58657 | schema_subquery | store_hash_par     | p1       |         3 |         0 |         3 |          0 |          3
|         100.00
(10 rows)
```

18.3.195 PGXC_GET_TABLE_SKEWNESS

PGXC_GET_TABLE_SKEWNESS视图展示当前库中表的数据分布倾斜情况。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

 版权所有 © 华为云计算技术有限公司

使用PGXC_GET_TABLE_SKEWNESS视图进行数据倾斜查询，建议在数据库中表数量（小于10000）较少的场景使用。

表 18-269 PGXC_GET_TABLE_SKEWNESS 字段

名称	类型	描述
schemaname	name	表所在的模式名。
tablename	name	表名。
totalsize	numeric	表的总大小，单位Byte。
avgsize	numeric(1000, 0)	表大小平均值（totalsize/DN个数，该值为平均分布的理想情况下，表在各DN占用空间大小）。
maxratio	numeric(10,3)	单DN表大小最大值占比（表在各DN占用空间的最大值/avgsize）。
minratio	numeric(10,3)	单DN表大小最小值占比（表在各DN占用空间的最小值/avgsize）。
skewsize	bigint	表分布倾斜值（单DN表大小最大值 - 单DN表大小最小值）。
skewratio	numeric(10,3)	表分布倾斜率（skewsize/ avgsize）。
skewstddev	numeric(1000, 0)	表分布标准方差（在表大小一定的情况下，该值越大表明表的整体分布情况越倾斜）。

18.3.196 PGXC_GLOBAL_TEMP_ATTACHED_PIDS

查看全局临时表在CN上占有资源的会话信息。该视图仅8.2.1.220及以上集群版本支持。

表 18-270 PG_GLOBAL_TEMP_ATTACHED_PIDS 字段

名称	类型	描述
nodename	name	节点名。
schemaname	name	模式名。
tablename	name	表名。
pid	bigint	用于表示会话的PID。

18.3.197 PGXC_GTM_SNAPSHOT_STATUS

PGXC_GTM_SNAPSHOT_STATUS视图用于查看当前GTM上事务信息。

 版权所有 © 华为云计算技术有限公司

表 18-271 PGXC_GTM_SNAPSHOT_STATUS 字段

名称	类型	描述
xmin	xid	仍在运行的最小事务号。
xmax	xid	已完成的事务号最大的事务的下一个事务号。
csn	integer	待提交事务的序列号。
oldestxmin	xid	当前最早的活跃事务在其取快照时，所有运行事务号最小的事务。
xcnt	integer	当前活跃的事务个数。
running_xids	text	当前活跃的事务号。

18.3.198 PGXC_INSTANCE_TIME

PGXC_INSTANCE_TIME视图显示集群中各节点上进程的运行时间信息及各执行阶段所消耗时间，除新增node_name（节点名称）字段外，其余字段内容和PV_INSTANCE_TIME视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-272 PGXC_INSTANCE_TIME 字段

名称	类型	描述
node_name	text	节点名称。
stat_id	integer	类型编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

18.3.199 PGXC_LOCKWAIT_DETAIL

PGXC_LOCKWAIT_DETAIL视图显示集群中每个节点中锁等待链详细信息。如果节点中有多级的锁等待关系，会依次将整个锁等待链按照等待顺序显示出来。

该视图仅8.1.3.200及以上集群版本支持。

表 18-273 PGXC_LOCKWAIT_DETAIL 字段

名称	类型	描述
level	integer	锁等待链中的层级，以1开始，每显示一层等待关系level会加1。
node_name	name	节点名称，对应pgxc_node表中的node_name列。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
lock_wait_hierarchy	text	锁等待链，以节点名称：进程号->等待进程号->等待进程号->...。
lock_type	text	被锁定对象的类型。
database	oid	被锁定对象所在数据库的OID。
relation	oid	被锁定对象关系的OID。
page	integer	关系内部的页面编号。
tuple	smallint	页面的行编号。
virtual_xid	text	事务的虚拟ID。
transaction_id	xid	事务ID。
class_id	oid	包含该对象的系统表的OID。
obj_id	oid	对象在其系统表内的OID。
obj_subid	smallint	对于表的列字段编号。
virtual_transaction	text	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	持有此锁或者等待此锁的线程号。
mode	text	锁级别。
granted	boolean	是否持有锁。
fastpath	boolean	是否通过fastpath机制获得锁。
wait_for_pid	bigint	锁冲突线程的线程号。
conflict_mode	text	锁冲突线程持有的冲突锁级别。
query_id	bigint	查询语句的id。
query	text	查询语句。
application_name	text	连接到该后端的应用名。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_start	timestamp with time zone	开始当前活跃查询的时间。
state	text	后端当前总体状态。
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始获取锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

应用实例

步骤1 连接dn节点，开启一个事务，执行查询：

```
begin;select * from t1;
```

步骤2 另一个窗口连接cn，对表t1执行truncate：

```
truncate t1;
```

此时truncate会被阻塞。

步骤3 另开一个窗口连接cn，执行select * from pgxc_lockwait_detail;

```
SELECT * FROM PGXC_LOCKWAIT_DETAIL;
level | node_name |             lock_wait_hierarchy              | lock_type | database |  relation  | page | tuple |
virtual_xid | transaction_id | class_id | obj_id | obj_subid | virtual_transaction |       p
id       |        mode         | granted | fastpath |  wait_for_pid   |  conflict_mode  |     query_id
|                  query                   | application_name |        backend_start         |
xact_start          |         query_start          |        state
-------+-----------+----------------------------------------------+-----------+----------+------------+------+-------
+-------------+----------------+----------+--------+-----------+---------------------+--------
---------+---------------------+---------+----------+-----------------+-----------------+-------------------
+-------------------------------------------------+------------------+------------------------------+--
----------------------------+-----------------------------+--------------------
1 | datanode1 | datanode1:140378619314976                    | relation |    16049 | 2147484411 |      |
|           |         673638 |          |        |           | 19/297             | 1403786
19314976 | AccessExclusiveLock | f       | f        | 140378619263840 | AccessShareLock | 73183493945504391
| TRUNCATE t1                    | coordinator1     | 2023-03-13 12:13:52.530602+08 | 2
023-03-13 14:52:16.1456+08   | 2023-03-13 14:52:16.148693+08 | active
2 | datanode1 | datanode1:140378619314976 -> 140378619263840 | relation |    16049 | 2147484411 |
|       |           |               |          |        |           | 23/16067           | 1403786
19263840 | AccessShareLock     | t       | f        |               |               |               0 | begin;select * from t1;
| gsql            | 2023-03-13 14:19:26.325602+08 | 2
023-03-13 14:52:12.042741+08 | 2023-03-13 14:52:12.042741+08 | idle in transaction
(2 rows)
```

----结束

18.3.200 PGXC_INSTR_UNIQUE_SQL

PGXC_INSTR_UNIQUE_SQL视图展示集群中所有CN节点的Unique SQL的完整统计信息。

需要有系统管理员权限才可以访问此视图，该视图的字段与GS_INSTR_UNIQUE_SQL视图字段相同。具体的字段请参考**表18-274**。

 版权所有 © 华为云计算技术有限公司

表 18-274 GS_INSTR_UNIQUE_SQL 字段

名称	类型	描述
node_name	name	接收SQL的CN节点名称。
node_id	integer	节点ID，等同于pgxc_node表中node_id。
user_name	name	用户名称。
user_id	oid	用户ID。
unique_sql_id	bigint	归一化的UNIQUE SQL ID。
query	text	归一化的SQL文本。
n_calls	bigint	成功执行次数。
min_elapse_time	bigint	SQL在数据库内的最小运行时间（单位：微秒）。
max_elapse_time	bigint	SQL在数据库内的最大运行时间（单位：微秒）。
total_elapse_time	bigint	SQL在数据库内的总运行时间（单位：微秒）。
n_returned_rows	bigint	行活动-SELECT语句返回的结果集行数。
n_tuples_fetched	bigint	行活动-随机扫描行（列存表/外表不统计）。
n_tuples_returned	bigint	行活动-顺序扫描行（列存表/外表不统计）。
n_tuples_inserted	bigint	行活动-插入行数。
n_tuples_updated	bigint	行活动-更新行数。
n_tuples_deleted	bigint	行活动-删除行数。
n_blocks_fetched	bigint	buffer的块访问次数，即物理读/IO。
n_blocks_hit	bigint	buffer的块命中次数，即逻辑读/Cache。
n_soft_parse	bigint	软解析次数（缓存计划）。
n_hard_parse	bigint	硬解析次数（生成计划）。
db_time	bigint	有效的DB执行时间，包含等待时间、网络发送时间等，若查询执行涉及到多线程，DB_TIME是多个线程的DB_TIME之和（单位：微秒）。
cpu_time	bigint	CPU的执行时间，不包含sleep时间（单位：微秒）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
execution_time	bigint	查询执行器内的SQL执行时间，DDL语句、以及某些不经过执行器执行的语句（例如Copy语句）不计数（单位：微秒）。
parse_time	bigint	SQL解析时间（单位：微秒）。
plan_time	bigint	SQL生成计划时间（单位：微秒）。
rewrite_time	bigint	SQL重写时间（单位：微秒）。
pl_execution_time	bigint	plpgsql过程化语言函数上的执行时间（单位：微秒）。
pl_compilation_time	bigint	plpgsql过程化语言函数上的编译时间（单位：微秒）。
net_send_time	bigint	网络时间，包含CN向客户端发送数据、DN向CN发送数据等时间（单位：微秒）。
data_io_time	bigint	IO时间，文件IO耗时（单位：微秒）。
first_time	timestamp with time zone	该SQL第一次执行的时间。
last_time	timestamp with time zone	该SQL上一次执行的时间。

18.3.201 PGXC_LOCK_CONFLICTS

PGXC_LOCK_CONFLICTS视图提供集群中有冲突的锁的信息。

当某一个锁正在等待另一个锁，或正在被另一个锁等待，即该锁是有冲突的。

目前，PGXC_LOCK_CONFLICTS视图只收集locktype为relation、partition、page、tuple和transactionid的锁的信息。

表 18-275 PGXC_LOCK_CONFLICTS 字段

名称	类型	描述
locktype	text	被锁定对象的类型。
nodename	name	被锁定对象的节点的名称。
dbname	name	被锁定对象的数据库的名称。如果被锁定对象是事务，则为NULL。
nspname	name	被锁定对象的命名空间的名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
relname	name	被锁定对象对应的关系的名称。如果被锁定对象既不是关系，也不是关系的一部分，则为NULL。
partname	name	被锁定对象对应的分区的名称。如果被锁定对象不是分区，则为NULL。
page	integer	被锁定对象对应的页面的编号。如果被锁定对象既不是页面，也不是元组，则为NULL。
tuple	smallint	被锁定对象对应的元组的编号。如果被锁定对象不是元组，则为NULL。
transactionid	xid	被锁定对象对应的事务的ID。如果被锁定对象不是事务，则为NULL。
username	name	申请锁的用户的名称。
gxid	xid	申请锁的事务的ID。
xactstart	timestamp with time zone	申请锁的事务的开始时间。
queryid	bigint	申请锁的线程的最新查询ID。
query	text	申请锁的线程的最新查询语句。
pid	bigint	申请锁的线程的ID。
mode	text	锁的级别。
granted	boolean	• 如果锁已被持有，则为TRUE。 • 如果锁还在等待其它锁，则为FALSE。

18.3.202 PGXC_LWLOCKS

PGXC_LWLOCK视图提供当前集群中所有实例正在持有的或等待的轻量级锁信息。该视图仅9.1.0.200及以上集群版本支持。

表 18-276 PGXC_LWLOCKS 字段

名称	类型	描述
nodename	name	被锁定对象的节点的名称。
pid	bigint	后端线程ID。
query_id	bigint	查询语句的ID。
lwtid	integer	后端线程的轻量级线程号。
reqlockid	integer	当前线程正在请求的轻量级锁ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
reqlock	text	reqlockid对应的轻量级锁名称。
heldlocknums	integer	当前线程已经获得的轻量级锁的数量。
heldlockid	integer	当前线程已经获得的轻量级锁ID。
heldlock	text	heldlockid对应的轻量级锁名称。
heldlockmode	text	heldlockid对应的轻量级锁模式。

示例

使用PGXC_LWLOCKS视图查询当前集群中所有实例正在持有的或等待的轻量级锁信息：

```
SELECT * FROM pgxc_lwlocks;
 nodename  |       pid       |     query_id      | lwtid | reqlockid | reqlock | heldlocknums | heldlockid |
heldlock       | heldlockmode
-----------+-----------------+-------------------+-------+-----------+---------+--------------+------------
+--------------------+--------------
 datanode1 | 139810224193360 | 78250043525924188 | 54844 |           |         |            1 |      76390 |
BUFFER_POOL_LWLOCK | Shared
 datanode1 | 139810224198200 | 78250043525924886 | 54922 |           |         |            1 |     957438 |
PGPROC_LWLOCK      | Exclusive
 datanode2 | 140262654050288 |                 0 | 54832 |           |         |            1 |          7 |
WALWriteLock       | Exclusive
 datanode2 | 140262654052488 | 78250043525923195 | 54847 |           |         |            1 |      15862 |
BUFFER_POOL_LWLOCK | Shared
(4 rows)
```

18.3.203 PGXC_MEMORY_DEBUG_INFO

PGXC_MEMORY_DEBUG_INFO视图显示当前集群每个节点在执行作业时的内存报错信息，便于定位内存报错问题，当执行语句报错提示“memory is temporarily unavailable”时，通过该视图可查询到所有节点的内存报错信息，该报错信息跟日志中显示的内存报错信息相同。该视图8.3.0及以上集群版本支持。

须知

该视图仅显示最近一次报错的集群信息，重复报错信息会进行覆盖，同一个query多次申请内存报错，信息不会进行更新。

表 18-277 PGXC_MEMORY_DEBUG_INFO 字段

名称	类型	描述
node_name	text	实例名称，包含CN和DN。
query_id	bigint	正在申请内存的查询ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memory_info	text	当前实例的内存使用情况，主要包含： ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● shared_used_memory：已使用的共享内存。 ● cstore_used_memory：列存已使用的内存大小。 ● comm_used_memory：通信库已使用的内存大小。 ● comm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● topsql_used_memory：topsql已使用内存大小 ● large_storage_memory：列存压缩和解压缩使用的内存大小。 ● os_totalmem：操作系统总内存大小。 ● os_freeemem：操作系统剩余内存大小。
summary	text	包含实例上作业消耗的总估算内存和消耗的总实际内存大小。
abnormal_query	text	使用内存异常的线程ID和queryid，包含两种情况： 1. 当前使用内存最大的会话。 2. 估算内存和实际使用内存差别最大的会话。
abnormal_memory	text	使用内存异常的内存块，包含共享内存ctx使用最大的和通用内存ctx使用最大的
top_thread	text	使用内存最多的三个线程信息： context name：表示正在使用内存的内存块。 contextlevel：表示ctx的等级。 sessType：表示ctx顶层节点的类型。 totalsize[274,13,260]MB，表示当前内存ctx的总内存，已释放内存和使用内存大小，单位为MB。
create_time	timestamp with time zone	出现内存不足报错的时间点。

 版权所有 © 华为云计算技术有限公司

18.3.204 PGXC_NODE_ENV

PGXC_NODE_ENV视图提供获取集群中所有节点的环境变量信息。

表 18-278 PGXC_NODE_ENV 字段

名称	类型	描述
node_name	text	集群中所有节点的名称。
host	text	集群中所有节点的主机名称。
process	integer	集群中所有节点的进程号。
port	integer	集群中所有节点的端口号。
installpath	text	集群中所有节点的安装目录。
datapath	text	集群中所有节点的数据目录。
log_directory	text	集群中所有节点的日志目录。

18.3.205 PGXC_NODE_STAT_RESET_TIME

PGXC_NODE_STAT_RESET_TIME视图显示集群中各节点的统计信息重置时间，除新增node_name（节点名称）字段外，其余字段内容和**GS_NODE_STAT_RESET_TIME**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-279 PGXC_NODE_STAT_RESET_TIME 字段

名称	类型	描述
node_name	text	节点名称。
reset_time	timestamp	各节点的统计信息重置时间。

18.3.206 PGXC_OBS_IO_SCHEDULER_STATS

PGXC_OBS_IO_SCHEDULER_STATS视图显示OBS IO Scheduler读/写请求相关的近期实时统计信息。该系统视图仅9.1.0及以上版本支持。

表 18-280 PGXC_OBS_IO_SCHEDULER_STATS 字段

名称	类型	描述
node_name	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
io_type	char	IO类型： ● 'r'表示读。 ● 'w'表示写。 ● 's'表示文件操作。
current_bps	int8	当前带宽速率(KB/s)。
best_bps	int8	近期达到过的最佳带宽速率(KB/s)。
waiting_request_num	int	当前排队的请求数。
mean_request_size	int8	近期已处理请求的平均长度(KB)。
total_token_num	int	总的IO令牌数。
available_token_num	int	可用IO令牌数。
total_worker_num	int	总的工作线程数。
idle_worker_num	int	空闲的工作线程数。

示例

步骤1 查询OBS IO Scheduler在每个节点读请求相关的统计信息。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r       |       26990 |    26990 |                   0 |               215 |              18 |              16
|              12 |              10
 dn_6003_6004 | r       |       21475 |    21475 |                  10 |               190 |              30 |              30
|              20 |              20
 dn_6005_6006 | r       |       12384 |    12384 |                  36 |               133 |              30 |              27
|              20 |              17
```

查询结果显示，这是当前IO Scheduler在进行读取IO操作时的某个时刻统计信息的快照（snapshot），此时带宽处于上升阶段，current_bps与best_bps相等。以dn_6003_6004为例，可以观察到该DN当前队列中存在排队的请求，total_token_num与available_token_num相等，说明查询视图的时刻IO Scheduler还未开始处理这些请求。

步骤2 等待一段时间后，再次发起查询。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r       |       13228 |    26990 |                   0 |               609 |              18 |              18
|              12 |              12
 dn_6003_6004 | r       |       15717 |    21475 |                   0 |               622 |              30 |              30
|              20 |              20
```

 版权所有 © 华为云计算技术有限公司

```
 dn_6005_6006 | r       |       18041 |    21767 |                0 |             609 |              30 |                30
|              20 |             20
```

此时队列中已经没有了排队的请求，且available_token_num等于total_token_num，说明IO Scheduler已经处理完所有请求，且没有新的请求需要被处理；但观察到current_bps不为零，是因为统计bps的周期为3秒，此时显示的是3秒前的结果。

步骤3 短暂间隔后再次查询结果如下，current_bps会更新为0。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r       |           0 |    26990 |                   0 |               609 |              18 |                18
|              12 |             12
 dn_6003_6004 | r       |           0 |    21475 |                   0 |               622 |              30 |                30
|              20 |             20
 dn_6005_6006 | r       |           0 |    21767 |                   0 |               609 |              30 |                30
|              20 |             20
```

----结束

18.3.207 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS

PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS视图统计了OBS IO Scheduler不同请求类型（包括读/写/文件操作）下的请求数量、流控信息等内容。该系统视图仅9.1.0及以上版本支持。

其中第一次查询结果显示的是自集群启动以来到查询时刻之间的统计内容，详细字段见下表。

表 18-281 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS 字段

名称	类型	描述
node_name	name	CN或DN实例的名称，例如，dn_6001_6002。
io_type	char	IO类型，包括： • R（读） • W（写） • S（文件操作）
recent_throttled_req_num	int	两次查询视图之间的限流次数。
total_throttled_req_num	int	总的限流次数。
last_throttled_dur(s)	int8	距离上次发生限流的时间间隔。
waiting_req_num	int	当前有多少排队的请求数。
mean_tps	numeric(7,2)	两次查询视图之间的平均tps，tps为每秒处理的请求数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
mean_req_size(KB)	int8	两次查询视图之间的请求的平均长度，单位KB。
mean_req_latency(ms)	int8	两次查询视图之间的请求的平均时延，单位ms。
max_req_latency(ms)	int8	两次查询视图之前的请求的最大时延，单位ms。
mean_bps(KB/s)	int8	两次查询视图之间的平均读取请求or写入请求的速度，单位KB/s。
duration(s)	int	两次查询视图之间的时间间隔，单位s。

示例

使用语句SELECT * FROM pgxc_obs_io_scheduler_periodic_stats查询视图内容，结果示例：

```
SELECT * FROM pgxc_obs_io_scheduler_periodic_stats;

  node_name   | io_type | recent_throttled_req_num | total_throttled_req_num | last_throttled_dur(s) |
waiting_req_num | mean_tps | mean_req_size(KB) | mean_req_latency(ms) | max_req_latency(ms) |
mean_bps(KB/s) | duration(s)
--------------+---------+--------------------------+-------------------------+-----------------------+-----------------
+----------+------------------+---------------------+---------------------+----------------+-------------
 dn_6001_6002 | S       |                        0 |                       0 |                     0 |             0 |     0.00
|              0 |                0 |                   0 |            0 |        155
 dn_6001_6002 | R       |                        0 |                       0 |                     0 |             0 |     0.00
|              0 |                0 |                   0 |            0 |        155
 dn_6001_6002 | W       |                        0 |                       0 |                     0 |             0 |     0.00
|              0 |                0 |                   0 |            0 |        155
 cn_5001      | S       |                        0 |                       0 |                     0 |             0 |      .03 |
0 |              207 |              519 |                 0 |        155
 cn_5001      | R       |                        0 |                       0 |                     0 |             0 |     0.00 |
0 |                0 |                0 |                 0 |        155
 cn_5001      | W       |                        0 |                       0 |                     0 |             0 |      .01 |
0 |              288 |              288 |                 0 |        155
(6 rows)
```

如果需要显示“mean_tps”数值中小数点前的0，需要在behavior_compat_options参数中设置display_leading_zero（浮点数显示配置项）。

再执行查询视图语句“select * from pgxc_obs_io_scheduler_periodic_stats;” 显示结果如下：

```
SELECT * FROM pgxc_obs_io_scheduler_periodic_stats;

  node_name   | io_type | recent_throttled_req_num | total_throttled_req_num | last_throttled_dur(s) |
waiting_req_num | mean_tps | mean_req_size(KB) | mean_req_latency(ms) | max_req_latency(ms) |
mean_bps(KB/s) | duration(s)
--------------+---------+--------------------------+-------------------------+-----------------------+-----------------
+----------+------------------+---------------------+---------------------+----------------+-------------
 dn_6001_6002 | S       |                        0 |                       0 |                     0 |             0 |     0.36
|              0 |              132 |                 326 |            0 |        177
 dn_6001_6002 | R       |                        0 |                       0 |                     0 |             0 |     0.00
|              0 |                0 |                   0 |            0 |        177
 dn_6001_6002 | W       |                        0 |                       0 |                     0 |             0 |     0.00
|              0 |                0 |                   0 |            0 |        177
 cn_5001      | S       |                        0 |                       0 |                     0 |             0 |     0.00 |
0 |                0 |                0 |                 0 |        177
 cn_5001      | R       |                        0 |                       0 |                     0 |             0 |     0.00 |
```


版权所有 © 华为云计算技术有限公司

```
0 |                0 |                0 |                0 |       177
 cn_5001      | W      |                  0 |                  0 |                0 |             0 |     0.00 |
0 |                0 |                0 |                0 |       177
```

18.3.208 PGXC_OS_RUN_INFO

PGXC_OS_RUN_INFO视图显示集群中各节点上操作系统运行的状态信息，除新增node_name（节点名称）字段外，其余字段内容和PV_OS_RUN_INFO视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-282 PGXC_OS_RUN_INFO 字段

名称	类型	描述
node_name	text	节点名称。
id	integer	编号。
name	text	操作系统运行状态名称。
value	numeric	操作系统运行状态值。
comments	text	操作系统运行状态注释。
cumulative	boolean	操作系统运行状态的值是否为累加值。

18.3.209 PGXC_OS_THREADS

PGXC_OS_THREADS视图提供当前集群中所有正常节点下的线程状态信息。

表 18-283 PGXC_OS_THREADS 字段

名称	类型	描述
node_name	text	当前集群中所有正常节点的名称。
pid	bigint	当前集群中所有正常节点进程中正在运行的线程号。
lwpid	integer	与pid对应的轻量级线程号。
thread_name	text	与pid对应的线程名称。
creation_time	timestamp with time zone	与pid对应的线程创建的时间。

18.3.210 PGXC_POOLER_STATUS

PGXC_POOLER_STATUS视图显示当前集群中各CN节点的pooler中缓存连接状态。该视图只能在CN上执行查询，显示所有CN的pooler模块的连接缓存信息。
PGXC_POOLER_STATUS视图仅8.3.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-284 PGXC_POOLER_STATUS 字段

名称	类型	描述
coorname	text	CN节点名称。
database	text	数据库名称。
user_name	text	用户名。
tid	bigint	连接CN的线程ID。
node_oid	bigint	连接的实例节点OID。
node_name	name	连接的实例节点名称。
in_use	boolean	连接是否正被使用。 • t（true）：表示连接正在使用。 • f（false）：表示连接没有使用。
fdsock	bigint	对端socket。
remote_pid	bigint	对端线程号。
session_params	text	由此连接下发的GUC会话参数。

18.3.211 PGXC_PREPARED_XACTS

PGXC_PREPARED_XACTS视图显示当前处于prepared阶段的两阶段事务。

表 18-285 PGXC_PREPARED_XACTS 字段

名字	类型	描述
pgxc_prepared_xact	text	查看当前处于prepared阶段的两阶段事务。

18.3.212 PGXC_REDO_STAT

视图PGXC_REDO_STAT显示集群中各节点上XLOG重做过程中的统计信息，除新增node_name（节点名称）字段外，其余字段内容和**PV_REDO_STAT**视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-286 PGXC_REDO_STAT 字段

名称	类型	描述
node_name	text	节点名称。
phywrts	bigint	物理写次数。
phyblkwrt	bigint	物理写块数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
writetim	bigint	物理写消耗时间。
avgiotim	bigint	平均每次写入时间。
lstiotim	bigint	上一次写入时间。
miniotim	bigint	最小写入时间。
maxiowtm	bigint	最大写入时间。

18.3.213 PGXC_REL_IOSTAT

PGXC_REL_IOSTAT视图显示集群中各节点上磁盘读写的统计信息。需要有系统管理员权限才可以访问此视图。

表 18-287 PGXC_REL_IOSTAT 字段

名称	类型	描述
node_name	text	节点名称。
phyrds	bigint	读磁盘次数。
phywrts	bigint	写磁盘次数。
phyblkrd	bigint	读磁盘页数。
phyblkwrt	bigint	写磁盘页数。

18.3.214 PGXC_REPLICATION_SLOTS

PGXC_REPLICATION_SLOTS视图显示集群中DN上的复制信息，除增加表示节点名称的node_name字段外，其余字段内容和**PG_REPLICATION_SLOTS**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-288 PGXC_REPLICATION_SLOTS 字段

名称	类型	描述
node_name	text	节点名称。
slot_name	text	复制节点的名称。
plugin	name	逻辑复制槽对应的输出插件名。
slot_type	text	复制节点的类型。
datoid	oid	复制节点的数据库OID。
database	name	复制节点的数据库名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
active	boolean	复制节点是否为激活状态。
xmin	xid	复制节点事务标识。
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识。
restart_lsn	text	复制节点的Xlog文件信息。
dummy_standby	boolean	复制节点是否为假备。

18.3.215 PGXC_RESPOOL_RUNTIME_INFO

PGXC_RESPOOL_RUNTIME_INFO视图显示所有CN上所有资源池作业运行信息。

表 18-289 PGXC_RESPOOL_RUNTIME_INFO 字段

名称	类型	描述
nodename	name	CN名称。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数。
fast_run	int	资源池快车道运行作业数。
fast_wait	int	资源池快车道排队作业数。
slow_run	int	资源池慢车道运行作业数。
slow_wait	int	资源池慢车道排队作业数。

18.3.216 PGXC_RESPOOL_RESOURCE_INFO

PGXC_RESPOOL_RESOURCE_INFO视图显示所有实例上资源池实时监控信息。

版权所有 © 华为云计算技术有限公司

说明

- DN上仅显示当前DN所属逻辑集群的资源池监控信息。
- 从8.2.0集群版本开始，新增了内存负反馈机制功能，CCN节点会根据DN节点的实际内存使用情况辅助反向调节语句的估算内存统计值，缓解语句估算内存高估场景，当CCN节点触发通过负反馈机制来减少估算内存统计值来增发作业时，由于CCN节点更新了语句缩减后的估算内存统计值，而CN节点还保留了原先的估算内存统计值，会导致资源池监控视图中的估算内存超过资源池的上限。
- 语句的算子分为重内存算子和非重内存算子，对于语句的内存管控是重内存算子，非重内存算子的内存开销较小；由于非重内存算子的内存开销、线程初始化的开销、表达式的开销等这些内存不做管控，会导致资源池的used_mem会在一定程度上超过mem_limit值。

表 18-290 PGXC_RESPOOL_RESOURCE_INFO 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示"installation"。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5s监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为DWS可用CPU，CPU限额管控情况下为关联控制组CPU可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_mem	int	资源池当前使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5s监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5s监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5s监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5s监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5s监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5s监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
send_speed	double	资源池5s监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5s监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。

18.3.217 PGXC_RESPOOL_RESOURCE_HISTORY

PGXC_RESPOOL_RESOURCE_HISTORY用于查询所有实例上资源池监控历史信息。

表 18-291 PGXC_RESPOOL_RESOURCE_HISTORY 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
timestamp	timestamp	资源池监控信息持久化时间。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_cpu	double	资源池5s监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为GaussDB可用CPU，CPU限额管控情况下为关联控制组CPU可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5s监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5s监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5s监控周期内逻辑读次数 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5s监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_speed	double	资源池5s监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5s监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5s监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5s监控周期内网络接收速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。

18.3.218 PGXC_ROW_TABLE_IO_STAT

PGXC_ROW_TABLE_IO_STAT视图提供集群所有CN和DN节点上当前数据库所有行存表的IO统计数据。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与GS_ROW_TABLE_IO_STAT视图相同，具体的字段请参考**表18-292**。

表 18-292 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

18.3.219 PGXC_RUNNING_XACTS

PGXC_RUNNING_XACTS视图主要功能是显示集群中各个节点运行事务的信息，字段内容和**PG_RUNNING_XACTS**相同。

表 18-293 PGXC_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄。
gxid	xid	事务ID号。
state	tinyint	事务状态（3：prepared或者0：starting）。
node	text	节点名称。
xmin	xid	节点上当前数据涉及的最小事务号xmin。
vacuum	boolean	标志当前事务是否是lazy vacuum事务。
timeline	bigint	标识数据库重启次数。
prepare_xid	xid	处于prepared状态事务的id号，若不在prepared状态，值为0。
pid	bigint	事务对应的线程ID。
next_xid	xid	CN传给DN的事务ID号。

18.3.220 PGXC_SETTINGS

PGXC_SETTINGS视图显示集群中各节点数据库运行时参数的相关信息，除新增node_name（节点名称）字段外，其余字段内容和**PG_SETTINGS**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-294 PGXC_SETTINGS 字段

名称	类型	描述
node_name	text	节点名称。
name	text	参数名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
setting	text	参数当前值。
unit	text	参数的隐式结构。
category	text	参数的逻辑组。
short_desc	text	参数的简单描述。
extra_desc	text	参数的详细描述。
context	text	设置参数值的上下文，包括internal，postmaster，sighup，backend，superuser和user。
vartype	text	参数类型，包括bool，enum，integer，real和string。
source	text	参数的赋值方式。
min_val	text	参数最小值。如果参数类型不是数值型，那么该字段值为null。
max_val	text	参数最大值。如果参数类型不是数值型，那么该字段值为null。
enumvals	text[]	enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。
boot_val	text	数据库启动时参数默认值。
reset_val	text	数据库重置时参数默认值。
sourcefile	text	设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。
sourceline	integer	设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。

18.3.221 PGXC_SESSION_WLMSTAT

PGXC_SESSION_WLMSTAT视图显示当前集群中各CN节点用户执行作业正在运行时的负载管理相关信息。

表 18-295 PGXC_SESSION_WLMSTAT 字段

名称	类型	描述
nodename	name	节点名称。
datid	oid	连接后端的数据库OID。
datname	name	连接后端的数据库名称。
threadid	bigint	后端线程ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
processid	integer	后端线程的PID。
usesysid	oid	登录后端的用户OID。
appname	text	连接到后端的应用名。
username	name	登录到该后端的用户名。
priority	bigint	语句所在Cgroups的优先级。
attribute	text	语句的属性： ● Ordinary：语句发送到数据库后被解析前的默认属性。 ● Simple：简单语句。 ● Complicated：复杂语句。 ● Internal：数据库内部语句。
block_time	bigint	语句当前为止的pending的时间，单位s。
elapsed_time	bigint	语句当前为止的实际执行时间，单位s。
total_cpu_time	bigint	语句在上一时间周期内的DN上CPU使用的总时间，单位s。
cpu_skew_percent	integer	语句在上一时间周期内的DN上CPU使用的倾斜率。
statement_mem	integer	语句执行所需要的估算内存。
active_points	integer	语句占用的资源池并发点数。
dop_value	integer	从资源池中获取语句的dop值。
control_group	text	语句当前所使用的Cgroups。
status	text	语句当前的状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active：非以上四种状态外的正常状态。 ● unknown：未知状态。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
enqueue	text	语句当前的排队情况，包括： ● Global：全局排队。 ● Respool：资源池排队。 ● CentralQueue：在中心协调节点(CCN)中排队。 ● Transaction：语句处于一个事务块中。 ● StoredProc：语句处于一个存储过程中。 ● None：未在排队。 ● Forced None：事务块语句或存储过程语句由于超出设定的等待时间而强制执行。
resource_pool	name	语句当前所在的资源池。
query	text	该后端的最新查询。如果state状态是active，此字段显示当前正在执行的查询。所有其他情况表示上一个查询。
isplana	bool	逻辑集群模式下，语句当前是否占用其他逻辑集群的资源执行。该值默认为f，表示不占用其他逻辑集群的资源执行。
node_group	text	语句所属用户对应的逻辑集群。
lane	text	表示语句查询的快慢车道。 ● fast：快车道。 ● slow：慢车道。 ● none：未管控。

18.3.222 PGXC_STAT_ACTIVITY

PGXC_STAT_ACTIVITY视图显示当前集群下所有CN的当前用户查询相关的信息。

表 18-296 PGXC_STAT_ACTIVITY 字段

名称	类型	描述
coorname	text	当前集群下的CN名称。
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
lwtid	integer	后端线程的轻量级线程号。
usesysid	oid	登录此后端的用户OID。
usename	name	登录此后端的用户名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
application_name	text	连接到此后端的应用名。
client_addr	inet	连接到此后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，此字段是通过client_addr的反向DNS查找得到。仅在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁或者等待节点则为t，否则为f。
enqueue	text	语句当前排队状态。可能值是： ● waiting in global queue：表示语句在全局并发队列排队中。 ● waiting in respool queue：表示语句在资源池排队中，包含以下两类场景： 1. 动态负载开启的情况下，简单作业数量超过了快车道并发上限max_dop。 2. 动态负载关闭的情况下，简单作业数量超过了快车道并发上限max_dop或者复杂作业数量超过了慢车道并发上限。 ● waiting in ccn queue：表示作业在CCN排队中，包含全局内存排队和慢车道内存和并发排队。 ● 空或no waiting queue：表示语句正在运行。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	后端当前总体状态。可能值是： ● active：后端正在执行一个查询。 ● idle：后端正在等待一个新的客户端命令。 ● idle in transaction：后端在事务中，但事务中没有语句在执行。 ● idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 ● fastpath function call：后端正在执行一个fast-path函数。 ● disabled：如果后端禁用track_activities，则报告此状态。 **说明** 只有系统管理员能查看到自己账户所对应的会话状态。其他账户的state信息为空。
resource_pool	name	用户使用的资源池。
stmt_type	text	用户语句的语句类型。
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

应用实例

查看当前处于阻塞状态的查询语句。

```
SELECT datname,usename,state,query FROM PGXC_STAT_ACTIVITY WHERE waiting = true;
```

查看快照线程的工作状态。

```
SELECT application_name,backend_start,state_change,state,query FROM PGXC_STAT_ACTIVITY WHERE application_name='WDRSnapshot';
```

查看正在运行的查询语句。

```
SELECT datname,usename,state,pid FROM PGXC_STAT_ACTIVITY;
 datname  | usename | state  |       pid
----------+---------+--------+-----------------
 gaussdb | Ruby    | active | 140298793514752
 gaussdb | Ruby    | active | 140298718004992
 gaussdb | Ruby    | idle   | 140298650908416
 gaussdb | Ruby    | idle   | 140298625742592
 gaussdb | dbadmin | active | 140298575406848
(5 rows)
```

查看指定数据库postgres上已使用的会话连接数。其中1表示数据库postgres上已使用的会话连接数。

 版权所有 © 华为云计算技术有限公司

```
SELECT COUNT(*) FROM PGXC_STAT_ACTIVITY WHERE DATNAME='postgres';
 count
-------
     1
(1 row)
```

18.3.223 PGXC_STAT_BAD_BLOCK

PGXC_STAT_BAD_BLOCK视图显示集群所有节点从启动后，在读取数据时出现Page/CU校验失败的统计信息。

表 18-297 PGXC_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称。
databaseid	integer	数据库OID。
tablespaceid	integer	表空间OID。
relfilenode	integer	文件对象ID。
forknum	integer	文件类型。
error_count	integer	出现校验失败的次数。
first_time	timestamp with time zone	第一次出现的时间。
last_time	timestamp with time zone	最近一次出现的时间。

18.3.224 PGXC_STAT_BGWRITER

PGXC_STAT_BGWRITER视图显示集群中各节点上后端写进程活动的统计信息，除新增node_name（节点名称）字段外，其余字段内容和**PG_STAT_BGWRITER**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-298 PGXC_STAT_BGWRITER 字段

名称	类型	描述
node_name	text	节点名称。
checkpoints_timed	bigint	定期执行的检查点数量。
checkpoints_req	bigint	请求执行的检查点数量。
checkpoint_write_time	double precision	检查点期间将文件写入磁盘花费的时间，以毫秒为单位。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
checkpoint_sync_time	double precision	检查点期间数据同步到磁盘花费的时间，以毫秒为单位。
buffers_checkpoint	bigint	检查点期间写入缓冲区的数量。
buffers_clean	bigint	后端写进程写的缓冲区数量。
maxwritten_clean	bigint	由于写入缓冲区太多，后端写进程停止清理扫描的次数。
buffers_backend	bigint	后端直接写入的缓冲区数量。
buffers_backend_fsync	bigint	后端需要fsync的次数。
buffers_alloc	bigint	分配的缓冲区数量。
stats_reset	timestamp with time zone	统计重置的时间。

18.3.225 PGXC_STAT_DATABASE

视图PGXC_STAT_DATABASE显示集群中各节点上数据库的状态和统计信息，除新增node_name（节点名称）字段外，其余字段内容和**PG_STAT_DATABASE**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-299 PGXC_STAT_DATABASE 字段

名称	类型	描述
node_name	text	节点名称。
datid	oid	数据库OID。
datname	name	数据库名。
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tup_returned	bigint	当前节点上该数据库查询返回的行数。
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。
tup_inserted	bigint	当前节点上该数据库插入的行数。
tup_updated	bigint	当前节点上该数据库更新的行数。
tup_deleted	bigint	当前节点上该数据库删除的行数。
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。可参见PG_STAT_DATABASE_CONFLICTS。
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。

18.3.226 PGXC_STAT_OBJECT

PGXC_STAT_OBJECT视图显示集群中所有实例的表的统计信息和autovacuum效率信息。该系统视图仅8.2.1及以上集群版本支持。

表 18-300 PGXC_STAT_OBJECT 字段

名称	类型	引用	描述
nodename	name	-	节点名称。
datname	name	-	表所在数据库名称。
relnamespace	name	-	表所在schema名称。
relname	name	-	表名。
partname	name	-	分区表的分区名。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
databaseid	oid	PG_DATABASE.oid	数据库OID。
relid	oid	PG_CLASS.oid	表OID，分区表为主表OID。
partid	oid	PG_PARTITION .oid	分区OID，普通表此列为0。
numscans	bigint	-	启动顺序扫描的次数。
tuples_returned	bigint	-	顺序扫描抓取的可见元组条数。
tuples_fetched	bigint	-	抓取的可见元组条数。
tuples_inserted	bigint	-	插入条数。
tuples_updated	bigint	-	更新条数。
tuples_deleted	bigint	-	删除条数。
tuples_hot_updated	bigint	-	HOT更新条数。
n_live_tuples	bigint	-	可见元组数。
last_autovacuum_begin_n_dead_tuple	bigint	-	Autovacuum执行前删除元组数。
n_dead_tuples	bigint	-	Autovacuum成功后删除元组数。
changes_since_analyze	bigint	-	Analyze后最近一次数据修改时间。
blocks_fetched	bigint	-	选中的页面数。
blocks_hit	bigint	-	扫描过的页面数。
cu_mem_hit	bigint	-	CU内存命中次数。
cu_hdd_sync	bigint	-	从磁盘同步读取CU次数。
cu_hdd_asyn	bigint	-	从磁盘异步读取CU次数。
data_changed_timestamp	timestamp with time zone	-	最近一次数据修改时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
data_access_timestamp	timestamp with time zone	-	表的最后一次访问时间。
analyze_timestamp	timestamp with time zone	-	最近一次analyze时间。
analyze_count	bigint	-	Analyze总次数。
autovac_analyze_timestamp	timestamp with time zone	-	最近一次autoanalyze时间。
autovac_analyze_count	bigint	-	Autoanalyze总次数。
vacuum_timestamp	timestamp with time zone	-	最近一次vacuum的时间。
vacuum_count	bigint	-	vacuum总次数。
autovac_vacuum_timestamp	timestamp with time zone	-	最近一次autovacuum时间。
autovac_vacuum_count	bigint	-	Autovacuum总次数。
autovacuum_success_count	bigint	-	成功执行的autovacuum总次数。
last_autovacuum_time_cost	bigint	-	最近一次成功的autovacuum花费时间，单位：微秒。
avg_autovacuum_time_cost	bigint	-	成功执行autovacuum的平均执行时间，单位：微秒。
last_autovacuum_failed_count	bigint	-	从上一次autovacuum成功到现在，autovacuum总失败次数。
last_autovacuum_trigger	smallint	-	最近一次autovacuum触发方式，用于辅助维护人员进行vacuum情况的判断。
last_autovacuum_oldestxmin	bigint	-	最近一次autovacuum成功执行后的oldestxmin。如果表级oldestxmin特性开启，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
last_autovacuum_scan_pages	bigint	-	最近一次autovacuum扫描的页面数（仅针对行存表）。
last_autovacuum_dirty_pages	bigint	-	最近一次autovacuum修改的页面数（仅针对行存表）。
last_autovacuum_clear_deadtuples	bigint	-	最近一次autovacuum清理的deadtuple数（仅针对行存表）。
sum_autovacuum_scan_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计扫描的页面数（仅针对行存表）。
sum_autovacuum_dirty_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计修改的页面数（仅针对行存表）。
sum_autovacuum_clear_deadtuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的deadtuple数（仅针对行存表）。
last_autovacuum_begin_cu_size	bigint	-	最近一次autovacuum前的CU文件大小（仅针对列存表）。
last_autovacuum_cu_size	bigint	-	最近一次autovacuum后的CU文件大小（仅针对列存表）。
last_autovacuum_rewrite_size	bigint	-	最近一次autovacuum重写的列存文件大小（仅针对列存表）。
last_autovacuum_clear_size	bigint	-	最近一次autovacuum清理的列存文件大小（仅针对列存表）。
last_autovacuum_clear_cbtree_tuples	bigint	-	最近一次autovacuum清理的cbtree tuple数（仅针对列存表）。
sum_autovacuum_rewrite_size	bigint	-	从数据库初始化开始到现在，autovacuum累计重写的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_size	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_cbtree_tuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的cbtree tuple数（仅针对列存表）。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
last_autovacuum_csn	bigint	-	如果表级oldestxmin特性打开，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值对应的CSN值。
last_reference_timestamp	timestamp with time zone	-	表的最后一次访问时间（该字段仅8.3.0及以上集群版本支持）。 对应PG_STAT_OBJECT中data_changed_time_stamp（最后一次修改时间）和data_access_timestamp（最后一次访问时间）两者中距现在最近的时间。
extra1	bigint	-	预留字段1。
extra2	bigint	-	预留字段2。
extra3	bigint	-	预留字段3。
extra4	bigint	-	预留字段4。

18.3.227 PGXC_STAT_REPLICATION

PGXC_STAT_REPLICATION视图显示集群中各节点上日志同步的状态信息，除新增node_name（节点名称）字段外，其余字段内容和PG_STAT_REPLICATION视图相同。需要有系统管理员权限才可以访问此视图。

表 18-301 PGXC_STAT_REPLICATION 字段

名称	类型	描述
node_name	text	节点名称。
pid	bigint	线程的PID。
usesysid	oid	用户系统ID。
usename	name	用户名。
application_name	text	程序名称。
client_addr	inet	客户端地址。
client_hostname	text	客户端名。
client_port	integer	客户端端口号。
backend_start	timestamp with time zone	程序启动时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	日志复制的状态（追赶状态，还是一致的流状态）。
sender_sent_location	text	发送端发送日志位置。
receiver_write_location	text	接收端write日志位置。
receiver_flush_location	text	接收端flush日志位置。
receiver_replay_location	text	接收端replay日志位置。
sync_priority	integer	同步复制的优先级（0表示异步）。
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步）。

18.3.228 PGXC_STAT_TABLE_DIRTY

PGXC_STAT_TABLE_DIRTY显示当前集群中所有节点（CN和DN）上全部表的统计信息，并展示表在单节点（单CN级或者单DN级）的脏页率。该视图仅8.1.3及以上集群版本支持。

说明

该视图的统计信息依赖于ANALYZE，为获取最准确的信息请先对表进行ANALYZE。

表 18-302 PGXC_STAT_TABLE_DIRTY 字段

名称	类型	描述
nodename	text	节点名。
schema	name	表的模式名。
tablename	name	表名。
partname	name	分区表的分区名。
last_vacuum	timestampwith time zone	最后一次手动vacuum时间。
last_autovacuum	timestampwith time zone	最后一次autovacuum时间。
last_analyze	timestampwith time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestampwith time zone	最后一次自动轮询分析的时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
vacuum_count	bigint	vacuum次数。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。
autoanalyze_count	bigint	自动分析的次数。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	HOT更新的行数。
n_tup_change	bigint	analyze之后改变的行数。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
dirty_rate	bigint	单节点的脏页率（单CN或单DN）。
last_data_changed	timestampwith time zone	记录表最后一次数据变化的时间。 该字段在8.2.0及以上集群版本中固定为空，建议通过PG_STAT_ALL_TABLES查询。

使用建议

- 对于高脏页率的系统表，建议在确认当前没有用户操作该系统表时，再执行VACUUM FULL。
- 建议对脏页率超过80%的非系统表执行VACUUM FULL，用户也可根据业务场景自行选择是否执行VACUUM FULL。

使用场景

1. 查询全库所有用户表的整体脏页率：

```
select
    t1.schema,
    t1.tablename,
    t1.total_ins,
    t1.total_upd,
    t1.total_del,
    t1. total_tup_hot_upd,
    t1.total_change,
    t1.total_live,
    t1.total_dead,
    t1.total_dirty_rate,
    t1.max_dirty,
    t2.max_node,
    t1.min_dirty,
    t2.min_node
```


版权所有 © 华为云计算技术有限公司

```
from
    (select
        a.schema,
        a.tablename,
        sum(a.n_tup_ins) as total_ins,
        sum(a.n_tup_upd) as total_upd,
        sum(a.n_tup_del) as total_del,
        sum(a.n_tup_hot_upd) as total_tup_hot_upd,
        sum(a.n_tup_change) as total_change,
        sum(a.n_live_tup) as total_live,
        sum(a.n_dead_tup) as total_dead,
        Round((total_dead / (total_dead + total_live + 0.0001) * 100),2) AS total_dirty_rate,
        max(a.dirty_rate) as max_dirty,
        min(a.dirty_rate) as min_dirty
    from pg_catalog.pgxc_stat_table_dirty a where a.partname is null and a.schema not in
('pg_toast','cstore','gs_logical_cluster','sys','dbms_om','information_schema','pg_catalog','dbms_output','
dbms_random','utl_raw','utl_raw dbms_sql','dbms_lob') group by a.tablename, a.schema
    ) t1,
    (select distinct
    tablename, schema,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate) as min_node,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate desc) as max_node
    from (select * from pg_catalog.pgxc_stat_table_dirty)) t2
where t1.tablename = t2.tablename and t1.schema = t2.schema;
```

2. 查询全库所有表(用户表+系统表)的整体脏页率：

```
select
    t1.schema,
    t1.tablename,
    t1.total_ins,
    t1.total_upd,
    t1.total_del,
    t1. total_tup_hot_upd,
    t1.total_change,
    t1.total_live,
    t1.total_dead,
    t1.total_dirty_rate,
    t1.max_dirty,
    t2.max_node,
    t1.min_dirty,
    t2.min_node
from
    (select
        a.schema,
        a.tablename,
        sum(a.n_tup_ins) as total_ins,
        sum(a.n_tup_upd) as total_upd,
        sum(a.n_tup_del) as total_del,
        sum(a.n_tup_hot_upd) as total_tup_hot_upd,
        sum(a.n_tup_change) as total_change,
        sum(a.n_live_tup) as total_live,
        sum(a.n_dead_tup) as total_dead,
        Round((total_dead / (total_dead + total_live + 0.0001) * 100),2) AS total_dirty_rate,
        max(a.dirty_rate) as max_dirty,
        min(a.dirty_rate) as min_dirty
    from pg_catalog.pgxc_stat_table_dirty a where a.partname is null group by a.tablename, a.schema
    ) t1,
    (select distinct
    tablename, schema,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate) as min_node,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate desc) as max_node
    from (select * from pg_catalog.pgxc_stat_table_dirty)) t2
where t1.tablename = t2.tablename and t1.schema = t2.schema;
```

3. 查询全库系统表信息：

```
select * from pgxc_stat_table_dirty where schema in
('pg_toast','cstore','gs_logical_cluster','sys','dbms_om','information_schema','pg_catalog','dbms_output','
dbms_random','utl_raw','utl_raw dbms_sql','dbms_lob');
```

 版权所有 © 华为云计算技术有限公司

18.3.229 PGXC_STAT_WAL

PGXC_STAT_WAL视图显示当前query的wal日志和数据页的流量信息，该视图仅8.2.0及以上集群版本支持。

表 18-303 PGXC_STAT_WAL 字段

名称	类型	描述
query_id	bigint	当前query的ID。
query_start	timestamp	query起始时间。
global_wal	bigint	当前query在集群产生的wal日志总量，单位是Byte。
global_avg_wal_speed	bigint	当前query在集群产生wal日志的平均速率，单位是Byte/s。
global_datapage	bigint	当前query在集群产生的数据页总量，单位是Byte。
global_avg_datapage_speed	bigint	当前query在集群产生数据页的平均速率，单位是Byte/s。
min_wal_node	Text	当前query产生wal日志量最小的实例组名。
min_wal	bigint	最小node产生的wal日志量，单位是Byte。
max_wal_node	Text	当前query产生wal日志量最大的实例组名。
max_wal	bigint	最大node产生的wal日志量，单位是Byte。
min_datapage_node	Text	当前query产生数据页量最小的实例组名。
min_data_page	bigint	最小node产生的数据页量，单位是Byte。
max_datapage_node	Text	当前query产生数据页量最大的实例组名。
max_data_page	bigint	最大node产生的数据页量，单位是Byte。
avg_wal_per_node	bigint	平均每个node产生的wal日志量，单位是Byte。
avg_datapage_per_node	bigint	平均每个node产生的数据页量，单位是Byte。
query	Text	当前执行的语句。

说明

行存不带索引批量导入时，数据页copy导入会产生logical newpage相关的xlog日志，在xlog量大于默认值时同样会触发流控。

版权所有 © 华为云计算技术有限公司

应用示例

查询集群当前正在执行的语句，以及这些语句分别产生的wal日志的总量和平均速率、数据页总量和平均速率，以及这些信息各自DN上的分布情况。

```
SELECT * FROM PGXC_STAT_WAL;
     query_id      |          query_start          | global_wal | global_avg_wal_speed | global_datapage |
global_avg_datapage_speed | min_wal_node | min_wal  | max_wal_node | max_wal  | min_datapage_node |
min_data_page | max_datapage_node | max_data_page | avg_wal_per_node |
avg_datapage_per_node |                    query
-------------------+-------------------------------+------------+----------------------+-----------------
+--------------------------+--------------+----------+--------------+----------+------------------+---------------
+------------------+---------------+------------------+-
----------------------+-------------------------------------------------
 72620543991351767 | 2022-11-10 16:49:47.743291+08 |    7579052 |              419000 |       284057600
|                15740000 | datanode1    |  7579052 | datanode1    |  7579052 | datanode1         |     284057600
| datanode1         |     284057600 |         7579052 |
          284057600 | insert into mpptest3 select * from mpptest3;
 72620543991351781 | 2022-11-10 16:50:00.616697+08 |   55022176 |            10638000 |             0
|                       0 | datanode1    | 55022176 | datanode1    | 55022176 | datanode1         |             0 |
datanode1         |             0 |        55022176 |
                  0 | insert into mpptest1 select * from mpptest1;
(2 rows)
```

18.3.230 PGXC_SQL_COUNT

通过PGXC_SQL_COUNT视图，可以实时显示集群中各CN节点上SELECT、INSERT、UPDATE、DELETE、MERGE INTO五种SQL、以及DDL、DML、DCL语句的节点级和用户级统计结果，识别当前业务负载较重的query类型，衡量整个集群和单个节点执行某种类型查询的能力。通过对以上几类SQL查询进行计数和响应时间统计，获得指定时刻的统计结果，经计算可以得到指定QPS等统计信息。例如，T1时刻，USER1的SELECT计数结果为X1，T2时刻为X2，则可计算得到该用户SELECT查询的QPS值为(X2-X1)/(T2-T1)。由此，可获得集群用户级QPS曲线图和集群吞吐情况，监测每个用户的业务负载是否发生剧烈变化。如果有剧烈变化，可以定位具体的语句类型(SELECT/INSERT/UPDATE/DELETE/MERGE INTO)。同时观测QPS曲线可以获知问题发生时间点，结合其它工具，定位问题点。能够为集群性能调优、问题定位等提供依据。

PGXC_SQL_COUNT视图的字段与GS_SQL_COUNT一致，详见**表18-146**。

说明

当执行MERGE INTO语句时，若能下推，在DN上收到的是MERGE INTO语句，将在DN节点上进行MERGE INTO类型计数，相应mergeinto_count列计数增加；若不能下推，在DN上收到的是UPDATE或INSERT语句，将在DN节点上进行UPDATE或INSERT类型计数，相应的update_count列或insert_count列计数增加。

18.3.231 PGXC_TABLE_CHANGE_STAT

PGXC_TABLE_CHANGE_STAT视图提供集群所有CN节点上当前数据库所有表的变更情况。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与**GS_TABLE_CHANGE_STAT**视图相同。

表 18-304 PGXC_TABLE_CHANGE_STAT 字段

名称	类型	描述
nodename	name	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动vacuum的时间。
vacuum_count	bigint	手动Vacuum的次数。
last_autovacuum	timestamp with time zone	最后一次自动vacuum的时间。
autovacuum_count	bigint	自动vacuum的次数。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
analyze_count	bigint	分析（包括手动和自动）的次数。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

18.3.232 PGXC_TABLE_STAT

PGXC_TABLE_STAT视图提供集群所有CN和DN节点上当前数据库所有表的统计信息。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与**GS_TABLE_STAT**视图相同。

表 18-305 PGXC_TABLE_STAT 字段

名称	类型	描述
nodename	name	节点名称。
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。只统计行存表。如果是分区表，显示各个分区扫描次数的和。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
seq_tuple_read	bigint	顺序扫描的行数。只统计行存表。
index_scan	bigint	索引扫描的次数。只统计行存表。
index_tuple_read	bigint	索引扫描的行数。只统计行存表。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

18.3.233 PGXC_THREAD_WAIT_STATUS

通过CN节点查看PGXC_THREAD_WAIT_STATUS视图，可以查看集群全局各个节点上所有SQL语句产生的线程之间的调用层次关系，以及各个线程的阻塞等待状态，从而更容易定位进程停止响应问题以及类似现象的原因。

PGXC_THREAD_WAIT_STATUS视图和PG_THREAD_WAIT_STATUS视图列定义完全相同，这是由于PGXC_THREAD_WAIT_STATUS视图本质是到集群中各个节点上查询PG_THREAD_WAIT_STATUS视图汇总的结果。

表 18-306 PGXC_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见**表18-239**。
wait_event	text	如果wait_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则是空。

例如：

在coordinator1执行一条语句之后长时间没有响应。可以创建另外一个连接到coordinator1上，查询coordinator1上的线程状态。

```
 select * from pg_thread_wait_status where query_id > 0;
  node_name   | db_name  | thread_name  | query_id |       tid       | lwtid | ptid  | tlevel | smpid |
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+--------+-------
+----------------------
 coordinator1 | gaussdb | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |     0 | wait node:
datanode4 |
(1 rows)
```

此外，可以查看该语句在全局范围内各个节点上的工作情况。如下所示，每个DN上都没有在等待的阻塞资源，因为读取的数据太多而执行较慢。

```
select * from pgxc_thread_wait_status where query_id=20971544;
  node_name   | db_name  | thread_name  | query_id |       tid       | lwtid | ptid  | tlevel | smpid |
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+--------+-------
+----------------------
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902867994384 | 22735 |       |      0 |     0 | wait
node: datanode3 |
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902838634256 | 22970 | 22735 |      5 |     0 |
synchronize quit     |
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902607947536 | 22972 | 22735 |      5 |     1 |
synchronize quit     |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632156796688 | 22736 |       |      0 |     0 | wait
node: datanode3 |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632030967568 | 22974 | 22736 |      5 |     0 |
synchronize quit     |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632081299216 | 22975 | 22736 |      5 |     1 |
synchronize quit     |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323627988752 | 22737 |       |      0 |     0 | wait
node: datanode3 |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323523131152 | 22976 | 22737 |      5 |     0 | net
flush data       |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323548296976 | 22978 | 22737 |      5 |     1 | net
flush data
 datanode4    | gaussdb | coordinator1 | 20971544 | 140103024375568 | 22738 |       |      0 |     0 | wait
node: datanode3
 datanode4    | gaussdb | coordinator1 | 20971544 | 140102919517968 | 22979 | 22738 |      5 |     0 |
synchronize quit     |
 datanode4    | gaussdb | coordinator1 | 20971544 | 140102969849616 | 22980 | 22738 |      5 |     1 |
synchronize quit     |
 coordinator1 | gaussdb | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |     0 | wait node:
datanode4  |
(13 rows)
```

 版权所有 © 华为云计算技术有限公司

18.3.234 PGXC_TOTAL_MEMORY_DETAIL

PGXC_TOTAL_MEMORY_DETAIL视图显示集群内存使用情况。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-307 PGXC_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存使用的名称。 ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● gpu_max_dynamic_memory：GPU内存最大值。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● gpu_dynamic_peak_memory：GPU内存使用的最大内存。 ● pooler_conn_memory：pooler连接占用内存大小。 ● pooler_freeconn_memory：pooler空闲连接占用的内存大小。 ● storage_compress_memory：列存压缩和解压缩使用的内存大小。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
		• udf_reserved_memory：为UDF Worker进程预留的内存大小。 • mmap_used_memory：mmap使用的内存大小。
memorymbytes	integer	内存使用的大小，单位为MB。

18.3.235 PGXC_TOTAL_SCHEMA_INFO

PGXC_TOTAL_SCHEMA_INFO视图提供了集群所有实例上的Schema空间信息，便于用户获悉集群各个实例上的Schema空间使用情况，仅支持在CN节点上查询。

表 18-308 PGXC_TOTAL_SCHEMA_INFO 字段

名称	类型	描述
schemaname	text	模式名称。
schemaid	oid	模式OID。
databasename	text	数据库名称。
databaseid	oid	数据库OID。
nodename	text	实例名称。
nodegroup	text	节点组名称。
usedspace	bigint	已使用的空间大小。
permspace	bigint	空间上限值。

18.3.236 PGXC_TOTAL_SCHEMA_INFO_ANALYZE

PGXC_TOTAL_SCHEMA_INFO_ANALYZE视图提供了集群整体的Schema空间信息，包括：集群空间总值、各实例空间平均值、倾斜率、单实例空间最大值、单实例空间最小值以及最大最小空间所在的实例名，便于用户获悉集群整体的Schema空间使用情况，仅支持在CN节点上查询。

表 18-309 PGXC_TOTAL_SCHEMA_INFO_ANALYZE 字段

名称	类型	描述
schemaname	text	模式名称。
databasename	text	数据库名称。
nodegroup	text	节点组名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_value	bigint	该模式的集群空间总值。
avg_value	bigint	该模式的各实例空间平均值。
skew_percent	integer	倾斜率。
extend_info	text	提供信息包括：单实例空间最大值、单实例空间最小值以及最大最小空间所在的实例名。

18.3.237 PGXC_TOTAL_USER_RESOURCE_INFO

PGXC_TOTAL_USER_RESOURCE_INFO视图显示所有实例上用户实时资源消耗信息。该视图仅8.2.0及以上集群版本支持。

表 18-310 PGXC_TOTAL_USER_RESOURCE_INFO 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
username	name	用户名。
used_memory	integer	正在使用的内存大小，单位MB。 DN：显示当前DN上对应用户正在使用的内存大小。 CN：显示所有DN上对应用户正在使用的内存累加和。
total_memory	integer	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 DN：显示当前DN上对应用户可以使用的内存大小。 CN：显示所有DN上对应用户可以使用的内存大小之和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。 DN：显示当前DN上对应用户正在使用的CPU核数。 CN：显示所有DN上对应用户正在使用的CPU核数之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_cpu	integer	用户关联控制组的CPU核数总和。 DN：显示当前DN上对应用户能够使用的CPU核数。 CN：显示所有DN上对应用户能够使用的CPU核数之和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的永久表存储空间大小。 CN：显示所有DN上对应用户已使用的永久表存储空间大小之和。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。 DN：显示当前DN上对应用户可使用的永久表存储空间大小。 CN：显示所有DN上对应用户可使用的永久表存储空间大小之和。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的临时表存储空间大小。 CN：显示所有DN上对应用户已使用的临时表存储空间大小之和。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。 DN：显示当前DN上对应用户可使用的临时表存储空间大小。 CN：显示所有DN上对应用户可使用的临时表存储空间大小之和。
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。 DN：显示当前DN上对应用户已使用的算子落盘空间大小 CN：显示所有DN上对应用户已使用的算子落盘空间大小之和
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。 DN：显示当前DN上对应用户可使用的算子落盘空间大小。 CN：显示所有DN上对应用户可使用的算子落盘空间大小之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑读的字节总数。单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑写的字节总数。单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的次数之和。 DN：实例启动至当前时间为止，该用户逻辑读的次数之和。
write_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的次数之和。 DN：实例启动至当前时间为止，该用户逻辑写的次数之和。
read_speed	double precision	CN：过去5秒内，该用户在单个DN上逻辑读平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑读平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在单个DN上逻辑写平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑写平均速率。单位KB/s。
send_speed	double precision	CN：过去5秒内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：过去5秒内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络接收平均速率。单位KB/s。

18.3.238 PGXC_USER_TRANSACTION

PGXC_USER_TRANSACTION视图提供查询所有CN上用户相关的事务信息。需要有系统管理员权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

版权所有 © 华为云计算技术有限公司

表 18-311 PGXC_USER_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称。
usename	name	用户名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

18.3.239 PGXC_VARIABLE_INFO

PGXC_VARIABLE_INFO视图用于查询集群中所有节点的xid、oid的状态。

表 18-312 PGXC_VARIABLE_INFO 字段

名称	类型	描述
node_name	text	节点名称。
nextOid	oid	该节点下一次生成的OID。
nextXid	xid	该节点下一次生成的事务号。
oldestXid	xid	该节点最早的事务号。
xidVacLimit	xid	强制autovacuum的临界点。
oldestXidDB	oid	该节点datafrozenxid最小的数据库OID。
lastExtendCSNLogpage	integer	最后一次扩展csnlog的页面号。
startExtendCSNLogpage	integer	csnlog扩展的起始页面号。
nextCommitSeqNo	integer	该节点下次生成的csn号。
latestCompletedXid	xid	该节点提交或者回滚后节点上的最新事务号。
startupMaxXid	xid	该节点关机前的最后一个事务号。

 版权所有 © 华为云计算技术有限公司

18.3.240 PGXC_WAIT_DETAIL

PGXC_WAIT_DETAIL视图显示集群中所有节点SQL的详细等待链信息。该视图仅8.1.3.200及以上集群版本支持。

表 18-313 PGXC_WAIT_DETAIL 字段

名称	类型	描述
level	integer	等待链中的层级，以1开始，每显示一层等待关系level会加1。
lock_wait_hierarchy	text	等待链，以节点名称：进程号->节点名称：等待进程号->节点名称：等待进程号->...。
node_name	text	节点名称。
db_name	text	database名称。
thread_name	text	线程名称。
query_id	bigint	查询语句的id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。
wait_event	text	持有此锁或者在等待此锁的事务的虚拟id。
exec_cn	boolean	是否执行sql语句的cn节点。
wait_node	text	锁级别。
query	text	查询语句。
application_name	text	连接到该后端的应用名。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间。
query_start	timestamp with time zone	开始当前活跃查询的时间。
waiting	boolean	是否正处于等待状态。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	后端当前总体状态。
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始获取锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

应用实例

步骤1 连接CN节点，开启一个事务，执行update操作：

```
begin;update td set c2=6 where c1=1;
```

步骤2 重开一个窗口连接CN节点，开启另一个事务，执行update操作（注意不要并发更新同一条记录）：

```
begin;update td set c2=6 where c1=7;
```

此时update操作会被阻塞。

步骤3 再开一个窗口连接CN节点，创建一个索引：

```
create index c2_key on td(c2);
```

步骤4 执行select * from pgxc_wait_detail;

```
SELECT * FROM PGXC_WAIT_DETAIL;
level |                lock_wait_hierarchy                 | node_name | db_name  | thread_name |     query_id
|       tid       | lwtid  | ptid | tlevel | sm
pid | wait_status  | wait_event | exec_cn | wait_node |                query                 | application_name |
backend_start         |          xact_st
art           |          query_start          | waiting | state
-------+------------------------------------------------------+-----------+----------+-------------+-------------------
+-----------------+--------+------+--------+---
----+--------------+------------+---------+-----------+---------------------------------------+------------------
+-------------------------------+-----------------
--------------+-------------------------------+---------+--------
1 | cn_5001:139870843444360                              | cn_5001   | postgres | workload    | 73183493945299462 |
139870843444360 | 578531 |      |      0 |
0 | wait node    |           | t       |           | WLM fetch collect info from data nodes | workload         |
2023-03-13 13:56:56.611486+08 | 2023-03-14 11:54
:33.562808+08 | 2023-03-13 13:57:00.262736+08 | t       | active
1 | cn_5001:139870843654544                              | cn_5001   | postgres | gsql        | 73183493945299204 |
139870843654544 | 722259 |      |      0 |
0 | wait node    |           | t       |           | update td set c2=6 where c1=1;         | gsql             | 2023-03-14
11:52:05.176588+08 | 2023-03-14 11:52
:19.054727+08 | 2023-03-14 11:53:58.114794+08 | t       | active
1 | cn_5001:139870843655296                              | cn_5001   | postgres | gsql        | 73183493945299218 |
139870843655296 | 722301 |      |      0 |
0 | wait node    |           | t       |           | update td set c2=6 where c1=7;         | gsql             | 2023-03-14
11:52:08.084265+08 | 2023-03-14 11:52
:42.978132+08 | 2023-03-14 11:53:59.459575+08 | t       | active
1 | cn_5001:139870843656424                              | cn_5001   | postgres | gsql        | 73183493945299223 |
139870843656424 | 722344 |      |      0 |
0 | acquire lock | relation    | t       |           | create index c2_key on td(c2);          | gsql             | 2023-03-14
11:52:10.967028+08 | 2023-03-14 11:52
:53.463227+08 | 2023-03-14 11:54:00.25203+08  | t       | active
2 | cn_5001:139870843656424 -> cn_5001:139870843655296 | cn_5001   | postgres | gsql        |
73183493945299218 | 139870843655296 | 722344 |      |        |
|             |           | f       |           | update td set c2=6 where c1=7;         | gsql             | 2023-03-14
```

 版权所有 © 华为云计算技术有限公司

```
11:52:08.084265+08 | 2023-03-14 11:52
:42.978132+08 | 2023-03-14 11:53:59.459575+08 | t       | active
(5 rows)
```

----结束

18.3.241 PGXC_WAIT_EVENTS

PGXC_WAIT_EVENTS视图显示集群中各节点各类等待状态和事件的统计信息，其字段内容和**GS_WAIT_EVENTS**视图相同。需要有系统管理员权限才可以访问此视图。

表 18-314 PGXC_WAIT_EVENTS 字段

名称	类型	描述
nodename	name	节点名称。
type	text	事件的类型，包括STATUS，LOCK_EVENT，LWLOCK_EVENT和IO_EVENT四种类型。
event	text	事件名称，可参考**PG_THREAD_WAIT_STATUS**视图。
wait	bigint	事件发生次数。该字段及以下字段均为进程运行中的累计值。
failed_wait	bigint	等待失败次数。当前版本中只有LOCK和LWLOCK等锁超时或失败才会使用该字段。
total_wait_time	bigint	该事件总持续时间。
avg_wait_time	bigint	该事件平均持续时间。
max_wait_time	bigint	该事件最大等待时间。
min_wait_time	bigint	该事件最小等待时间。

18.3.242 PGXC_WLM_OPERATOR_HISTORY

PGXC_WLM_OPERATOR_HISTORY视图显示在所有CN上执行作业结束时的算子信息。此视图用于从DWS数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟。

需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。具体的字段请参考**表18-315**。

表 18-315 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

18.3.243 PGXC_WLM_OPERATOR_INFO

PGXC_WLM_OPERATOR_INFO视图显示在所有CN上执行作业结束时的算子信息。此视图的数据直接从系统表GS_WLM_OPERATOR_INFO获取。

需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

须知

PGXC_WLM_OPERATOR_INFO视图表仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

PGXC_WLM_OPERATOR_INFO视图的字段同GS_WLM_OPERATOR_INFO相同，具体字段内容如下：

表 18-316 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

 版权所有 © 华为云计算技术有限公司

18.3.244 PGXC_WLM_OPERATOR_STATISTICS

PGXC_WLM_OPERATOR_STATISTICS视图显示在所有CN上正在执行作业的算子信息。系统管理员权限可以查询集群所有用户的作业算子信息，普通用户仅可查询自己的作业算子信息。

PGXC_WLM_OPERATOR_STATISTICS视图的字段同**表18-160**相同，具体字段内容如下：

表 18-317 GS_WLM_OPERATOR_STATISTICS 字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称，算子名称长度最大为127字符（不包含空格等格式化字符）。
start_time	timestamp with time zone	该算子第一次开始执行的时间。
duration	bigint	该算子从开始执行直到结束时总的执行时间(ms)。
status	text	当前算子的执行状态，包括waiting、running和finished。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息，若返回的预估行数超过int64_max时，显示为int64_max。
tuple_processed	bigint	当前算子在所有DN上的返回的元素个数总和，若返回的预估行数超过int64_max时，显示为int64_max。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小逻辑下盘数据量(MB)，默认为0。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_spill_size	integer	若发生下盘，所有下盘DN的最大逻辑下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均逻辑下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin
parent_id	integer	该算子节点的父节点id。
exec_count	integer	该算子节点在所有DN上的最大执行次数。
progress	text	该算子的进度信息，第一个算子展示的是作业整体的进度。其他算子展示的是当前算子进度信息。
min_net_size	bigint	该算子在所有DN上的最小网络通信数据量(KB)，主要涉及网络算子。
max_net_size	bigint	该算子在所有DN上的最大网络通信数据量(KB)，主要涉及网络算子。
total_net_size	bigint	该算子在所有DN上的总网络通信数据量(KB)，主要涉及网络算子。
min_read_bytes	bigint	该算子在所有DN上的最小磁盘读取数据量(KB)。
max_read_bytes	bigint	该算子在所有DN上的最大磁盘读取数据量(KB)。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_read_bytes	bigint	该算子在所有DN上的总磁盘读取数据量(KB)。
min_write_bytes	bigint	该算子在所有DN上的最小磁盘写入数据量(KB)。
max_write_bytes	bigint	该算子在所有DN上的最大磁盘写入数据量(KB)。
total_write_bytes	bigint	该算子在所有DN上的总磁盘写入数据量(KB)。

18.3.245 PGXC_WLM_SESSION_INFO

PGXC_WLM_SESSION_INFO视图显示在所有CN上执行作业结束后的负载管理记录。此视图信息来源于系统表**GS_WLM_SESSION_INFO**。

须知

PGXC_WLM_SESSION_INFO视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

表 18-318 PGXC_WLM_SESSION_INFO 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的PID。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上版本支持。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对3.0表及外表生效。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对3.0表及外表生效。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对3.0表及外表生效。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对3.0表生效。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对3.0表生效。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对3.0表生效。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对3.0表及外表生效。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对3.0表及外表生效。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对3.0表及外表生效。
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码。该字段仅对3.0表及外表生效。
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对3.0表及外表生效。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对3.0表及外表生效。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对3.0表及外表生效。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对3.0表及外表生效。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对3.0表及外表生效。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对3.0表及外表生效。

18.3.246 PGXC_WLM_SESSION_HISTORY

PGXC_WLM_SESSION_HISTORY视图显示在所有CN上执行作业结束后的负载管理记录。此视图用于从DWS数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟。

须知

PGXC_WLM_SESSION_HISTORY视图仅支持在**postgres**数据库中查询，其它数据库中查询会直接报错。

PGXC_WLM_SESSION_HISTORY视图的字段同**GS_WLM_SESSION_HISTORY**相同，具体字段内容如下：

表 18-319 GS_WLM_SESSION_HISTORY 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。
outline_name	text	该语句计划对应的outline的名字。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。

18.3.247 PGXC_WLM_SESSION_STATISTICS

PGXC_WLM_SESSION_STATISTICS视图显示在所有CN上正在执行的作业的负载管理信息。

表 18-320 PGXC_WLM_SESSION_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据OID。
dbname	name	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN节点名称。
username	name	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
pid	bigint	后端线程ID。
block_time	bigint	语句执行前的阻塞时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
duration	bigint	语句已经执行的时间，单位ms。
estimate_total_time	bigint	语句执行预估总时间，单位ms。
estimate_left_time	bigint	语句执行预估剩余时间，单位ms。
enqueue	text	工作负载管理资源状态。
resource_pool	name	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句在各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在各DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
min_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的最小IO读速率，单位KB/s。
max_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的最大IO读速率，单位KB/s。
average_read_speed	integer	一个监控周期（5s）内，语句在所有DN上的平均IO读速率，单位KB/s。
min_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的最小IO写速率，单位KB/s。
max_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的最大IO写速率，单位KB/s。
average_write_speed	integer	一个监控周期（5s）内，语句在所有DN上的平均IO写速率，单位KB/s。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
unique_sql_id	bigint	归一化的Unique SQL ID。
queryid	bigint	语句执行使用的内部query id。
query	text	正在执行的语句。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
stmt_type	text	语句所对应的查询类型。
except_info	text	语句触发的异常规则信息。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上版本支持。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
disk_cache_hit_ratio	numeric(5, 2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
block_name	text	与语句匹配的对应拦截规则名

18.3.248 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS

PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图展示当前库中表的数据分布倾斜情况。支持快速查看当前库中所有表在各节点的存储空间分布倾斜大小情况。该视图仅8.2.1及以上集群版本支持。

使用建议：

- 在大集群大数据量业务场景下进行全库各表所占磁盘空间倾斜率分析时，PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图的查询性能优于gs_table_distribution()函数和PGXC_GET_TABLE_SKEWNESS视图。建议先使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览，再使用gs_table_distribution(schemaname text, tablename text)获取指定表在各个节点所占磁盘空间大小的分布情况。
- 使用本视图查询指定表存储分布信息，需要具备指定表的SELECT权限。
- 该函数基于**PG_RELFILENODE_SIZE**系统表上的物理文件存储空间记录，需确保GUC参数use_workload_manager和enable_perm_space为开启状态。

表 18-321 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS 字段

名称	类型	描述
schema_name	name	表所在的模式名称。
table_name	name	表名。
total_size	numeric	表在各个节点上的存储空间大小总和值，单位：字节。
avg_size	numeric(1000,0)	表在各个节点上的存储空间大小的平均值，单位：字节。
max_percent	numeric	表在各个节点上的存储空间的最大值占总和值的百分比(%)。
min_percent	numeric	表在各个节点上的存储空间的最小值占总和值的百分比(%)。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
skew_percent	numeric	表的倾斜率信息(%)。 倾斜率skew_percent的计算方式：倾斜率(SKEW_PERCENT)= (最大值dnsize - 平均值dnsize) * 100/最大值dnsize

应用示例

使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览，再使用gs_table_distribution(schemaname text, tablename text)函数获取指定表在各个节点所占磁盘空间大小的分布情况。

步骤1 使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览。

```
tpcds_col=# select * from pgxc_wlm_table_distribution_skewness;
```

查询结果如下：

```
tpcds_col=# select * from pgxc_wlm_table_distribution_skewness;
    schema_name     |                table_name                | total_size  |  avg_size  | max_percent | min_percent | skew_percent
--------------------+------------------------------------------+-------------+------------+-------------+-------------+--------------
 pg_catalog         | pg_namespace_oid_index                   |       98304 |      16384 |       16.67 |       16.67 |         0.00
 public             | customer_demographics                    |     4595712 |    1531904 |       33.33 |       33.33 |         0.00
 pg_catalog         | pg_type_oid_index                        |      245760 |      40960 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_partition                             |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_user_mapping_user_server_index        |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_attrdef                               |      294912 |      49152 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_global_temp_parent_nsp_index          |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_namespace_nspname_index               |       98304 |      16384 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_synonym_name_nsp_index                |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_proc_proname_args_nsp_index           |     2015232 |     335872 |       16.67 |       16.67 |         0.00
 public             | dbgen_version                            |       32768 |      10923 |      100.00 |        0.00 |        66.67
 pg_catalog         | pg_amop_opr_fam_index                    |      294912 |      49152 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_global_temp                           |           0 |          0 |        0.00 |        0.00 |         0.00
 pg_catalog         | pg_workload_group_name_index             |           0 |          0 |        0.00 |        0.00 |         0.00
 public             | date_dim                                 |     7159808 |    2386603 |       34.10 |       32.95 |         2.24
 pg_catalog         | pg_operator_oid_index                    |      245760 |      40960 |       16.67 |       16.67 |         0.00
```

显示表dbgen_version的数据倾斜程度较为严重。

步骤2 使用gs_table_distribution(schemaname text, tablename text)函数查询表dbgen_version在各个节点所占磁盘空间大小的分布情况。

```
tpcds_col=# select * from gs_table_distribution('public','dbgen_version');
```

查询结果如下：

```
tpcds_col=# select * from gs_table_distribution('public','dbgen_version');
 schemaname |   tablename    | relkind | relpersistence |   nodename   | dnsize | sessionid
------------+----------------+---------+----------------+--------------+--------+-----------
 public     | dbgen_version | r       | p              | dn_6001_6002 |      0 |
 public     | dbgen_version | r       | p              | dn_6005_6006 |  32768 |
 public     | dbgen_version | r       | p              | dn_6003_6004 |      0 |
(3 rows)
```

显示该表在DN上所占磁盘空间确实存在数据倾斜现象，数据集中在dn_6005_6006上。

----结束

18.3.249 PGXC_WLM_USER_RESOURCE_HISTORY

PGXC_WLM_USER_RESOURCE_HISTORY视图显示所有用户在对应实例上资源消耗的历史信息。该视图仅8.2.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 18-322 PGXC_WLM_USER_RESOURCE_HISTORY 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
username	text	用户名。
timestamp	timestamp with time zone	时间戳。
used_memory	integer	正在使用的内存大小，单位MB。 DN：显示当前DN上对应用户正在使用的内存大小。 CN：显示所有DN上对应用户正在使用的内存累加和。
total_memory	integer	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 DN：显示当前DN上对应用户可以使用的内存大小。 CN：显示所有DN上对应用户可以使用的内存大小之和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。 DN：显示当前DN上对应用户正在使用的CPU核数。 CN：显示所有DN上对应用户正在使用的CPU核数之和。
total_cpu	integer	用户关联控制组的CPU核数总和。 DN：显示当前DN上对应用户能够使用的CPU核数。 CN：显示所有DN上对应用户能够使用的CPU核数之和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的永久表存储空间大小。 CN：显示所有DN上对应用户已使用的永久表存储空间大小之和。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。 DN：显示当前DN上对应用户可使用的永久表存储空间大小。 CN：显示所有DN上对应用户可使用的永久表存储空间大小之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的临时表存储空间大小。 CN：显示所有DN上对应用户已使用的临时表存储空间大小之和。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。 DN：显示当前DN上对应用户可使用的临时表存储空间大小 CN：显示所有DN上对应用户可使用的临时表存储空间大小之和
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。 DN：显示当前DN上对应用户已使用的算子落盘空间大小。 CN：显示所有DN上对应用户已使用的算子落盘空间大小之和。
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。 DN：显示当前DN上对应用户可使用的算子落盘空间大小。 CN：显示所有DN上对应用户可使用的算子落盘空间大小之和。
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的字节总数，单位KB。 DN：实例启动至当前时间为止，该用户复杂作业read的字节总数，单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的字节总数，单位KB。 DN：实例启动至当前时间为止，该用户复杂作业write的字节总数，单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的次数之和。 DN：实例启动至当前时间为止，该用户复杂作业read的次数之和。
write_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的次数之和。 DN：实例启动至当前时间为止，该用户复杂作业write的次数之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业read平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上复杂作业read平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业write平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上复杂作业write平均速率。单位KB/s。
send_speed	double precision	CN：一个5s监控周期内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：一个5s监控周期内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：一个5s监控周期内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：一个5s监控周期内，该用户在该DN上网络接收平均速率。单位KB/s。

18.3.250 PGXC_WLM_WORKLOAD_RECORDS

PGXC_WLM_WORKLOAD_RECORDS视图显示当前用户在每个CN上执行作业时在CN上的状态信息。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。该视图仅在动态负载功能开启，即enable_dynamic_workload为on时有效。

表 18-323 PGXC_WLM_WORKLOAD_RECORDS 字段

名称	类型	描述
node_name	text	作业执行所在的CN的名称。
thread_id	bigint	后端线程ID。
processid	integer	线程的lwpid。
timestamp	bigint	语句执行的开始时间。
username	name	登录到该后端的用户名。
memory	integer	语句所需的内存大小。
active_points	integer	语句在资源池上消耗的资源点数。
max_points	integer	资源在资源池上的最大资源数。
priority	integer	作业的优先级。
resource_pool	text	作业所在资源池。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
status	text	作业执行的状态，包括： pending：阻塞状态。 running：执行状态。 finished：结束状态。 aborted：终止状态。 unknown：未知状态。
control_group	text	作业所使用的Cgroups。
enqueue	text	作业的排队信息，包括： GLOBAL：全局排队。 RESPOOL：资源池排队。 ACTIVE：不排队。
query	text	正在执行的语句。

18.3.251 PGXC_WORKLOAD_SQL_COUNT

PGXC_WORKLOAD_SQL_COUNT视图显示集群中所有CN节点上的Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-324 PGXC_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

18.3.252 PGXC_WORKLOAD_SQL_ELAPSE_TIME

PGXC_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有CN节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE

 版权所有 © 华为云计算技术有限公司

语句的最大、最小、平均、以及总响应时间，单位为微秒。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 18-325 PGXC_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

18.3.253 PGXC_WORKLOAD_TRANSACTION

PGXC_WORKLOAD_TRANSACTION视图提供查询所有CN上Workload控制组相关的事务信息。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

表 18-326 PGXC_WORKLOAD_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间，单位微秒。
resp_max	bigint	最大响应时间，单位微秒。
resp_avg	bigint	平均响应时间，单位微秒。
resp_total	bigint	响应时间总和，单位微秒。

18.3.254 PLAN_TABLE

PLAN_TABLE视图显示用户通过执行EXPLAIN PLAN收集到的计划信息。计划信息的生命周期是session级别，session退出后相应的数据将被清除。同时不同session和不同user间的数据是相互隔离的。

表 18-327 PLAN_TABLE 字段

名称	类型	描述
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	查询生成的计划中的每一个执行算子的编号。
operation	varchar2(30)	计划中算子的操作描述。
options	varchar2(255)	操作选项。
object_name	name	操作对应的对象名，非查询中使用到的对象别名。来自于用户定义。
object_type	varchar2(30)	对象类型。
object_owner	name	对象所属schema，来自于用户定义。
projection	varchar2(4000)	操作输出的列信息。

 版权所有 © 华为云计算技术有限公司

说明

- object_type取值范围为PG_CLASS中定义的relkind类型（TABLE普通表，INDEX索引，SEQUENCE序列，VIEW视图，FOREIGN TABLE外表，COMPOSITE TYPE复合类型，TOASTVALUE TOAST表）和计划使用到的rtekind(SUBQUERY, JOIN, FUNCTION, VALUES, CTE, REMOTE_QUERY)。
- object_owner对于RTE来说是计划中使用的对象描述，非用户定义的类型不存在object_owner。
- statement_id、object_name、object_owner、projection字段内容遵循用户定义的大小写存储，其它字段内容采用大写存储。
- 支持用户对PLAN_TABLE进行SELECT和DELETE操作，不支持其它DML操作。

18.3.255 PV_FILE_STAT

PV_FILE_STAT视图通过对数据文件IO的统计，反映数据的IO性能，用于发现IO操作异常等性能问题。

表 18-328 PV_FILE_STAT 字段

名称	类型	描述
filenum	oid	文件标识。
dbid	oid	数据库标识。
spcid	oid	表空间标识。
phyrds	bigint	读物理文件的数目。
phywrts	bigint	写物理文件的数目。
phyblkrd	bigint	读物理文件块的数目。
phyblkwrt	bigint	写物理文件块的数目。
readtim	bigint	读文件的总时长，单位微秒。
writetim	bigint	写文件的总时长，单位微秒。
avgiotim	bigint	读写文件的平均时长，单位微秒。
lstiotim	bigint	最后一次读文件时长，单位微秒。
miniotim	bigint	读写文件的最小时长，单位微秒。
maxiowtm	bigint	读写文件的最大时长，单位微秒。

18.3.256 PV_INSTANCE_TIME

PV_INSTANCE_TIME视图用于统计进程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

提供当前节点下的各种时间消耗信息，主要分为以下类型：

- DB_TIME: 进程在多核并行计算下的时间开销。

版权所有 © 华为云计算技术有限公司

- CPU_TIME：进程在CPU核心上的时间开销。
- EXECUTION_TIME：执行器内的时间开销。
- PARSE_TIME：SQL解析的时间开销。
- PLAN_TIME：生成Plan的时间开销。
- REWRITE_TIME：SQL重写的时间开销。
- PL_EXECUTION_TIME ： plpgsql（存储过程）的执行时间.
- PL_COMPILATION_TIME：plpgsql（存储过程）编译时间。
- NET_SEND_TIME：网络上的时间开销。
- DATA_IO_TIME：IO上的时间开销。

表 18-329 PV_INSTANCE_TIME 字段

名称	类型	描述
stat_id	integer	类型编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

18.3.257 PV_MATVIEW_DETAIL

PV_MATVIEW_DETAIL视图显示物化视图的具体信息。该视图仅9.1.0 200及以上集群版本支持。

表 18-330 PV_MATVIEW_DETAIL 字段

名称	类型	描述
matview	text	物化视图名。
baserel	text	基表名。
partids	oidvector	指定分区时的分区OID。
contain_entire_rel	boolean	是否以基表的整表建物化视图。
build_mode	text	物化视图的build模式。 ● 'd'：代表deferred，表示创建物化视图时需要等到第一次refresh时才会包含数据。 ● 'i'：代表immediate，表示创建物化视图时即包含最新数据。
refresh_mode	text	物化视图的刷新模式。 'd'：代表demand，表示按需刷新。
refresh_method	text	物化视图的刷新方法。 'c' ： 表示完全刷新。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
mapping	text	基表分区和物化视图分区的映射关系。
active	boolean	物化视图是否需要刷新。
refresh_start_time	timestamp with time zone	最后一次刷新的开始时间。
refresh_finish_time	timestamp with time zone	最后一次刷新的结束时间。

18.3.258 PV_OS_RUN_INFO

PV_OS_RUN_INFO视图显示当前操作系统运行的状态信息。

表 18-331 PV_OS_RUN_INFO 字段

名称	类型	描述
id	integer	编号。
name	text	操作系统运行状态名称。
value	numeric	操作系统运行状态值。
comments	text	操作系统运行状态注释。
cumulative	boolean	操作系统运行状态的值是否为累加值。

18.3.259 PV_REDO_STAT

PV_REDO_STAT视图提供当前节点上XLOG重做过程中的统计信息。

表 18-332 PV_REDO_STAT 字段

名称	类型	描述
phywrts	bigint	物理写次数。
phyblkwrt	bigint	物理写块数。
writetim	bigint	物理写消耗时间。
avgiotim	bigint	平均每次写入时间。
lstiotim	bigint	上一次写入时间。
miniotim	bigint	最小写入时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
maxiowtm	bigint	最大写入时间。

18.3.260 PV_RUNTIME_ATTSTATS

PV_RUNTIME_ATTSTATS视图显示autoanalyze产生的内存中表级统计信息，各字段含义与**PG_STATS**视图一样。该视图仅8.2.0及以上集群使用。

表 18-333 PV_RUNTIME_ATTSTATS 字段

名称	类型	引用	描述
schemaname	name	**PG_NAMESPACE**.nspname	包含表的模式名。
tablename	name	**PG_CLASS**.relname	表名。
attname	name	**PG_ATTRIBUTE**.attname	字段名。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
partname	name	-	分区名。该字段仅8.2.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

18.3.261 PV_RUNTIME_EXPRSTATS

PV_RUNTIME_EXPRSTATS视图显示autoanalyze产生的内存中表达式统计信息。该视图仅8.2.1.200及以上集群版本支持。

表 18-334 PV_RUNTIME_EXPRSTATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
statistics_schemaname	name	-	表达式统计信息对象所在模式名。
statistics_name	name	-	表达式统计信息对象的名称。
statistics_owner	name	-	表达式统计信息对象的所有者。
expr	text	-	表达式
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
level	text	-	轻量化autoanalyze生成的内存统计信息当前所处的阶段。包含: local, sendlist, global三个阶段。

 版权所有 © 华为云计算技术有限公司

18.3.262 PV_RUNTIME_RELSTATS

PV_RUNTIME_RELSTATS视图显示autoanalyze产生的内存中表级统计信息，各字段含义与PG_CLASS视图一样。该视图仅8.2.0及以上集群使用。

表 18-335 PV_RUNTIME_RELSTATS 字段

名称	类型	描述
nspname	name	模式名。
relname	name	表、索引等对象的名称。
relpages	double precision	以页(大小为BLCKSZ)为单位的此表在磁盘上的大小，只是优化器使用的一个近似值。
reltuples	double precision	表中行的数目，只是优化器使用的一个估计值。
relallvisible	integer	被标识为全可见的表中的页数。此字段是优化器用来做SQL执行优化使用的。
relhasindex	boolean	如果对象是一个表且至少有（或者最近建有）一个索引，则为真。 由CREATE INDEX设置，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，会清理relhasindex字段，将relhasindex值设置为假。
changes	bigint	触发轻量化autoanalyze时，表的历史累计修改条数。
level	text	轻量化autoanalyze生成的内存统计信息当前所处的阶段。包含: local, sendlist, global三个阶段。
partname	name	分区名。该字段仅8.2.1.200及以上集群版本支持。

18.3.263 PV_SESSION_MEMORY

PV_SESSION_MEMORY视图统计Session级别的内存使用情况，包含执行作业在数据节点上Postgres线程和Stream线程分配的所有内存。

表 18-336 PV_SESSION_MEMORY 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename） **说明** 8.2.1集群版本之前的sessid仅包含线程线程启动时间+线程标识，8.2.1及以上集群版本sessid新增了节点名称，格式为：timestamp.threadid.nodename

版权所有 © 华为云计算技术有限公司

名称	类型	描述
init_mem	integer	当前正在执行作业进入执行器前已分配的内存，单位MB。
used_mem	integer	当前正在执行作业已分配的内存，单位MB。
peak_mem	integer	当前正在执行作业已分配的内存峰值，单位MB。

18.3.264 PV_SESSION_MEMORY_DETAIL

PV_SESSION_MEMORY_DETAIL统计线程的内存使用情况，以MemoryContext节点来统计。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

可通过"select * from pv_session_memctx_detail(threadid, '');"将某个线程所有内存上下文信息记录到“/tmp/dumpmem”目录下的“threadid_timestamp.log”文件中。其中threadid可通过下表sessid中获得。

表 18-337 PV_SESSION_MEMORY_DETAIL 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）。
sesstype	text	线程名称。
contextname	text	内存上下文名称。
level	smallint	当前上下文在整体内存上下文中的层级。
parent	text	父内存上下文名称。
totalsize	bigint	当前内存上下文的内存总数，单位Byte。
freesize	bigint	当前内存上下文中已释放的内存总数，单位Byte。
usedsize	bigint	当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。

应用示例

查询当前节点上所有MemoryContext的使用情况。

 版权所有 © 华为云计算技术有限公司

根据sessid定位到该MemoryContext是在哪个线程中创建和使用的，依据totalsize，freesize及usedsize来确认内存的使用情况是否符合预期，预先判断是否可能存在内存泄露的风险。

```
SELECT * FROM pv_session_memory_detail;
                sessid                |       sesstype       |                contextname                | level |
parent           | totalsize | freesize | usedsize
--------------------------------------+----------------------+-------------------------------------------+-------
+------------------------------+-----------+----------+----------
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TempSmallContextGroup                 |     0 |                          |    646520 |   125184 |      149
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TopMemoryContext                      |     0 |                          |    372672 |    19488 |   353184
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Record information
cache                    |     1 | TopMemoryContext            |     24576 |    13632 |    10944
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Operator class
cache                    |     1 | TopMemoryContext            |     24576 |    15168 |     9408
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | smgr relation
table                    |     1 | TopMemoryContext            |     24576 |    11040 |    13536
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TransactionAbortContext               |     1 | TopMemoryContext            |     32768 |    32704 |       64
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Portal
hash                     |     1 | TopMemoryContext            |     24576 |    16192 |     8384
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
CacheMemoryContext                    |     1 | TopMemoryContext            |   1858848 |    10640 |
1848208
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
InvalidCacheMemoryContext             |     2 | CacheMemoryContext          |   2088960 |   828928 |
1260032
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | WAL record
construction             |     1 | TopMemoryContext            |     49792 |     6528 |    43264
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | LOCAL LOCK
hash                     |     1 | TopMemoryContext            |     24576 |    15936 |     8640
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
Timezones                             |     1 | TopMemoryContext            |     24576 |    15808 |     8768
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | SELF MEMORY
CONTEXT                  |     1 | TopMemoryContext            |     66600 |    17824 |    48776
 1760444600.140326303562480.coordinator1 | file_recyclebin       |
TempSmallContextGroup                 |     0 |                          |    642688 |   129040 |      147
 1760444600.140326303562480.coordinator1 | file_recyclebin       | TopMemoryContext
|     0 |                          |   1437632 |    14256 |  1423376
 1760444600.140326303562480.coordinator1 | file_recyclebin       | Operator class cache
|     1 | TopMemoryContext            |     24576 |    15168 |     9408
 1760444600.140326303562480.coordinator1 | file_recyclebin       | smgr relation table
|     1 | TopMemoryContext            |     24576 |    11040 |    13536
 1760444600.140326303562480.coordinator1 | file_recyclebin       | TransactionAbortContext
|     1 | TopMemoryContext            |     32768 |    32704 |       64
 1760444600.140326303562480.coordinator1 | file_recyclebin       | Portal hash                               |
1 | TopMemoryContext            |     24576 |    16192 |     8384
 1760444600.140326303562480.coordinator1 | file_recyclebin       |
CacheMemoryContext                    |     1 | TopMemoryContext            |   1858848 |    12096 |
1846752
```

18.3.265 PV_SESSION_STAT

PV_SESSION_STAT视图以会话线程或AutoVacuum线程为单位，统计会话状态信息。

表 18-338 PV_SESSION_STAT 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）。
statid	integer	统计编号。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
statname	text	统计会话名称。
statunit	text	统计会话单位。
value	bigint	统计会话值。

18.3.266 PV_SESSION_TIME

PV_SESSION_TIME视图用于统计会话线程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

表 18-339 PV_SESSION_TIME 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）。
stat_id	integer	统计编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

18.3.267 PV_TOTAL_MEMORY_DETAIL

PV_TOTAL_MEMORY_DETAIL视图统计当前数据库节点使用内存的信息，单位为MB。

表 18-340 PV_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存类型，包括以下几种： ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● gpu_max_dynamic_memory：GPU内存最大值。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● gpu_dynamic_peak_memory：GPU内存使用的最大内存。 ● pooler_conn_memory：pooler连接占用内存大小。 ● pooler_freeconn_memory：pooler空闲连接占用的内存大小。 ● storage_compress_memory：列存压缩和解压缩使用的内存大小。 ● udf_reserved_memory：为UDF Worker进程预留的内存大小。 ● mmap_used_memory：mmap使用的内存大小。
memorymbytes	integer	内存类型分配内存的大小。

 版权所有 © 华为云计算技术有限公司

18.3.268 REDACTION_COLUMNS

REDACTION_COLUMNS视图展示当前数据库内所有脱敏列信息。

表 18-341 REDACTION_COLUMNS 字段

名称	类型	描述
object_schema	name	脱敏对象的schema。
object_owner	name	脱敏对象的owner。
object_name	name	脱敏对象名称。
column_name	name	脱敏列名称。
function_type	integer	脱敏类型。
function_parameters	text	脱敏类型为partial类型时的参数（保留字段，无实际意义）。
regexp_pattern	text	脱敏类型为regexp时，pattern串（保留字段，无实际意义）。
regexp_replace_string	text	脱敏类型为regexp时，替换串（保留字段，无实际意义）。
regexp_position	integer	脱敏类型为regexp时，起始替换位置（保留字段，无实际意义）。
regexp_occurrence	integer	脱敏类型为regexp时，替换次数（保留字段，无实际意义）。
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数（保留字段，无实际意义）。
function_info	text	脱敏函数信息。
column_description	text	脱敏列描述信息。
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。
policy_name	name	所属的脱敏策略名称。该字段仅8.2.1.100及以上集群版本支持。

18.3.269 REDACTION_POLICIES

REDACTION_POLICIES视图展示当前数据库内所有脱敏对象信息。

 版权所有 © 华为云计算技术有限公司

表 18-342 REDACTION_POLICIES 字段

名称	类型	描述
object_schema	name	脱敏对象的schema。
object_owner	name	脱敏对象的owner。
object_name	name	脱敏对象名称。
policy_name	name	脱敏策略名称。
expression	text	策略生效表达式（针对用户）。
enable	boolean	策略状态（开启、关闭）。
policy_description	text	策略描述信息。
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。

18.3.270 REMOTE_TABLE_STAT

REMOTE_TABLE_STAT视图提供集群所有DN节点上当前数据库所有表的统计信息。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与GS_TABLE_STAT视图相同。

表 18-343 REMOTE_TABLE_STAT 字段

名称	类型	描述
nodename	name	节点名称。
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。只统计行存表。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。只统计行存表。
index_scan	bigint	索引扫描的次数。只统计行存表。
index_tuple_read	bigint	索引扫描的行数。只统计行存表。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

18.3.271 SHOW_TSC_INFO

查询当前节点TSC信息。该视图仅8.2.1及以上集群版本支持。

表 18-344 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

18.3.272 SHOW_ALL_TSC_INFO

查询所有节点TSC信息。该视图仅8.2.1及以上集群版本支持。

表 18-345 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

18.3.273 USER_COL_COMMENTS

USER_COL_COMMENTS视图存储当前用户下表和视图的列注释信息。

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。
comments	text	注释。

18.3.274 USER_CONSTRAINTS

USER_CONSTRAINTS视图存储当前用户下表中约束的信息。

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名。
constraint_type	text	约束类型。 ● c表示检查约束。 ● f表示外键约束。 ● p表示主键约束。 ● u表示唯一约束。
table_name	character varying(64)	约束相关的表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

应用示例

查询当前用户下指定表的约束信息。t1替换为实际的表名。

```
SELECT * FROM USER_CONSTRAINTS WHERE table_name='t1';
 constraint_name | constraint_type | table_name | index_owner |  index_name
-----------------+-----------------+------------+-------------+---------------
 c_custkey_key   | p               | t1         | u1          | c_custkey_key
(1 row)
```

18.3.275 USER_CONS_COLUMNS

USER_CONS_COLUMNS视图存储当前用户下表中约束列的信息。

名称	类型	描述
table_name	character varying(64)	约束相关的表名。
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。
position	smallint	表中列的位置。

18.3.276 USER_INDEXES

USER_INDEXES视图存储关于本模式下的索引信息。

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引名称是否为系统生成。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
partitioned	character(3)	表示索引是否具有分区表的性质。

18.3.277 USER_IND_COLUMNS

USER_IND_COLUMNS视图存储当前用户下所有索引的字段信息。

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_name	name	列名。
column_position	smallint	索引中列的位置。

18.3.278 USER_IND_EXPRESSIONS

USER_IND_EXPRESSIONS视图存储当前用户下基于函数的表达式索引的信息。

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

18.3.279 USER_IND_PARTITIONS

USER_IND_PARTITIONS视图存储当前用户下的索引分区信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称。
schema	character varying(64)	索引分区所属分区表索引的模式。
index_name	character varying(64)	索引分区所属分区表索引的名称。
partition_name	character varying(64)	索引分区的名称。
index_partition_usable	boolean	索引分区是否可用。
high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collaton、字段数据类型等信息。
def_tablespace_name	name	索引分区的表空间名称。

18.3.280 USER_JOBS

USER_JOBS视图为当前用户所属定时任务的详细信息。需要有系统管理员权限才可以访问此系统视图。

表 18-346 USER_JOBS 字段

名字	类型	描述
job	int4	作业ID。
log_user	name not null	创建者的UserName。
priv_user	name not null	作业执行者的UserName。
dbname	name not null	作业创建数据库名字。
start_date	timestamp without time zone	作业的开始时间。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
start_suc	text	作业成功执行的开始时间。
last_date	timestamp without time zone	上次运行开始时间。
last_suc	text	上次成功运行的开始时间。
this_date	timestamp without time zone	正在运行任务的开始时间。
this suc	text	正在运行任务成功的开始时间。
next_date	timestamp without time zone	任务下次执行时间。
next suc	text	任务下次成功执行时间。
broken	text	任务状态 如果为Y，不尝试运行此任务。 如果为N，将尝试执行此任务。
status	char	当前任务的执行状态，取值范围：('r', 's', 'f', 'd')，默认为'r'，取值含义： ● r=running ● s=successfully finished ● f= job failed ● d=aborted
interval	text	用来计算下次运行时间的时间表达式，如果为nul，l则表示定时任务只执行一次。
failures	smallint	连续失败计数。
what	text	可执行的作业。

18.3.281 USER_OBJECTS

USER_OBJECTS视图描述了当前用户拥有的数据库对象。

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。
object_id	oid	对象的OID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。
last_ddl_time	timestamp with time zone	对象的最后修改时间。

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

18.3.282 USER_PART_INDEXES

USER_PART_INDEXES视图存储当前用户下分区表索引的信息。

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称。
schema	character varying(64)	分区表索引的模式。
index_name	character varying(64)	分区表索引的名称。
table_name	character varying(64)	分区表索引所属的分区表名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表索引的索引分区的个数。
def_tablespace_name	name	分区表索引的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

18.3.283 USER_PART_TABLES

USER_PART_TABLES视图存储当前用户下分区表的信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称。
schema	character varying(64)	分区表的模式。
table_name	character varying(64)	分区表的名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表的分区个数。
def_tablespace_name	name	分区表的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

18.3.284 USER_PROCEDURES

USER_PROCEDURES视图存储关于本模式下的存储过程或函数信息。

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
object_name	character varying(64)	存储过程或函数名称。
argument_number	smallint	存储过程入参个数。

18.3.285 USER_SEQUENCES

USER_SEQUENCES视图存储关于本模式下的序列信息。

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者。
sequence_name	character varying(64)	序列的名称。

18.3.286 USER_SOURCE

USER_SOURCE视图存储关于本模式下的存储过程或函数信息，且提供存储过程或函数定义的字段。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
name	character varying(64)	存储过程或函数的名称。
text	text	存储过程或函数的定义。

18.3.287 USER_SYNONYMS

USER_SYNONYMS视图存储当前用户可访问的同义词信息。

表 18-347 USER_SYNONYMS 字段

名称	类型	描述
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。
table_name	text	关联对象名。

18.3.288 USER_TAB_COLUMNS

USER_TAB_COLUMNS视图存储当前用户可访问的表和视图字段信息。

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	创建表或视图时列的序号。
data_length	integer	列的字节长度。
comments	text	注释。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2， bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

18.3.289 USER_TAB_COMMENTS

USER_TAB_COMMENTS视图存储当前用户所有表和视图的注释信息。

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

18.3.290 USER_TAB_PARTITIONS

USER_TAB_PARTITIONS视图存储当前用户下所有分区的信息。当前用户下每个分区表的每个分区在USER_TAB_PARTITIONS中都会有一条记录。

名称	类型	描述
table_owner	character varying(64)	分区所在表的所有者。
schema	character varying(64)	分区表模式。
table_name	character varying(64)	表名。
partition_name	character varying(64)	分区的名称。
high_value	text	范围分区的上边界，或列表分区的边界值集合。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
pretty_high_value	text	范围分区的上边界，或列表分区的边界值集合。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collaton、字段数据类型等信息。
tablespace_name	name	分区所在表空间的名称。

18.3.291 USER_TABLES

USER_TABLES视图存储关于当前模式下的表信息。

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间的名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

18.3.292 USER_TRIGGERS

USER_TRIGGERS视图存储关于当前用户下的触发器信息。

名称	类型	描述
trigger_name	character varying(64)	触发器名称。
table_name	character varying(64)	定义触发器的表的名称。
table_owner	character varying(64)	定义触发器的表的所有者。

版权所有 © 华为云计算技术有限公司

18.3.293 USER_VIEWS

USER_VIEWS视图存储关于当前模式下的所有视图信息。

名称	类型	描述
owner	character varying(64)	视图的所有者。
view_name	character varying(64)	视图的名称。

18.3.294 V$SESSION

V$SESSION视图存储当前会话的所有会话信息。

表 18-348 V$SESSION 字段

名称	类型	描述
sid	bigint	当前活动的后台进程的OID。
serial#	integer	当前活动的后台进程的序号，在DWS中为0。
user#	oid	登录此后台进程的用户的OID。
username	name	登录此后台进程的用户名。

18.3.295 V$SESSION_LONGOPS

V$SESSION_LONGOPS视图存储当前正在执行的操作的进度。

表 18-349 V$SESSION_LONGOPS 字段

名称	类型	描述
sid	bigint	当前正在执行的后台进程的OID。
serial#	integer	当前正在执行的后台进程的序号，在DWS中为0。
sofar	integer	目前完成的工作量，在DWS中为空。
totalwork	integer	工作总量，在DWS中为空。

版权所有 © 华为云计算技术有限公司

19 DWS 数据库 GUC 参数

19.1 查看 GUC 参数

DWS的GUC参数影响数据库的系统行为，用户可根据业务场景和数据量查看并调整GUC参数取值。

- 查看GUC参数方式一：集群创建成功后，用户可在DWS管理控制台上查看常用的数据库参数。
- 查看GUC参数方式二：成功连接集群后，通过SQL命令的方式查看数据库GUC参数。
  - 使用SHOW命令。

    说明

    方式二只能查CN的GUC参数值，DN的GUC参数值可通过方式一：通过管理控制台查看。

    使用如下命令查看单个参数：

    ```
    SHOW server_version;
    ```

    server_version显示数据库版本信息的参数。

    使用如下命令查看所有参数：

    ```
    SHOW ALL;
    ```

  - 使用pg_settings视图。

    使用如下命令查看单个参数：

    ```
    SELECT * FROM pg_settings WHERE NAME='server_version';
    ```

    使用如下命令查看所有参数：

    ```
    SELECT * FROM pg_settings;
    ```

19.2 设置 GUC 参数

为确保DWS的最优性能，用户可根据业务需求对数据库中的GUC参数进行调整。

参数类型和值

- DWS的GUC参数类型分为以下五类：

 版权所有 © 华为云计算技术有限公司

  - SUSET，数据库管理员参数。设置后立即生效，无需重启集群。若在当前会话中设置该类型参数仅当前会话生效。
  - USERSET，普通用户参数。设置后立即生效，无需重启集群。若在当前会话中设置该类型参数仅当前会话生效。
  - POSTMASTER，数据库服务端参数。设置后需要重启集群才能生效，确认修改后系统会提示集群状态为待重启，建议在非业务高峰期手动重启集群，使参数生效。
  - SIGHUP，数据库全局参数。设置后全局生效，无法会话级生效。
  - BACKEND，数据库全局参数。设置后全局生效，无法会话级生效。
- 所有的参数名称不区分大小写。 参数取值有整型、浮点型、字符串、布尔型和枚举型五类。
  - 布尔值可以是（on，off）、（true，false）、（yes，no）或者（1，0），且不区分大小写。
  - 枚举类型的取值由系统表pg_settings的enumvals字段取值所定义。
- 对于有单位的参数，在设置时请指定单位，否则将使用默认的单位。
  - 参数的默认单位由系统表pg_settings的unit字段所定义。
  - 内存单位有：KB（千字节）、MB（兆字节）和GB（吉字节）。
  - 时间单位：ms（毫秒）、s（秒）、min（分钟）、h（小时）和d（天）。

GUC 参数设置

GUC参数设置有两种方式：

- 方式一：集群创建成功后，用户可以登录DWS控制台，根据实际需要修改集群的数据库参数。 具体操作请参见《 数据仓库服务用户指南 》中的“修改DWS集群GUC参数”章节。
- 方式二：成功连接集群后，通过SQL命令的方式设置SUSET或USERSET类型的参数。

  修改指定数据库，用户，会话级别的参数。
  - 设置数据库级别的参数

    ALTER DATABASE *dbname* **SET** *paraname* **TO** *value;*

    在下次会话中生效。
  - 设置用户级别的参数

    ALTER USER *username* **SET** *paraname* **TO** *value;*

    在下次会话中生效。
  - 设置会话级别的参数

    SET *paraname* **TO** *value;*

    修改本次会话中的取值。退出会话后，设置将失效。

操作步骤

设置参数，以设置explain_perf_mode参数为例。

步骤1 查看explain_perf_mode参数。

```
SHOW explain_perf_mode;
 explain_perf_mode
--------------------
 normal
(1 row)
```

 版权所有 © 华为云计算技术有限公司

步骤2 设置explain_perf_mode参数。

使用以下任意方式进行设置：

- 设置数据库级别的参数
  ALTER DATABASE gaussdb SET *explain_perf_mode* **TO** *pretty*;
  当结果显示为如下信息，则表示设置成功。
  ```
  ALTER DATABASE
  ```
  在下次会话中生效。
- 设置用户级别的参数
  ALTER USER dbadmin **SET** *explain_perf_mode* **TO** *pretty*;
  当结果显示为如下信息，则表示设置成功。
  ```
  ALTER USER
  ```
  在下次会话中生效。
- 设置会话级别的参数
  SET *explain_perf_mode* **TO** *pretty*;
  当结果显示为如下信息，则表示设置成功。
  ```
  SET
  ```

步骤3 检查参数设置的正确性。

```
SHOW explain_perf_mode;
 explain_perf_mode
--------------
 pretty
(1 row)
```

----结束

19.3 GUC 使用说明

数据库提供了许多运行参数，配置这些参数可以影响数据库系统的行为。在修改这些参数时请确保已了解对应参数对数据库的影响，否则可能会导致无法预料的结果。

注意事项

- 参数中如果取值范围为字符串，此字符串应遵循操作系统的路径和文件名命名规则。
- 取值范围最大值为INT_MAX的参数，此选项最大值跟所在的操作系统有关。
- 取值范围最大值为DBL_MAX的参数，此选项最大值跟所在的操作系统有关。

19.4 连接和认证

19.4.1 连接设置

介绍设置客户端和服务器连接方式相关的参数。

max_connections

参数说明：允许和数据库连接的最大并发连接数。此参数会影响集群的并发能力。

 版权所有 © 华为云计算技术有限公司

参数类型：POSTMASTER

取值范围：整型。CN最小值为100，最大值为16384；DN最小值为100， 最大值为262143，由于集群内部存在着各种连接，设置时通常达不到最大值，若日志中出现'invalid value for parameter "max_connections"'，需要调小DN的max_connections值。

默认值：CN节点为800，DN节点为5000，如果该默认值超过内核支持的最大值（在执行gs_initdb的时候判断），系统会提示错误。

设置建议：

CN中此参数建议保持默认值。DN中此参数按照如下公式计算：

dop_limit * 20 * 6 + 24，公式中的dop_limit为集群中每个DN对应的CPU数，计算公式为：dop_limit = 单机器的CPU逻辑核数 / 单机器的DN数。

最小值5000。

增大这个参数可能导致DWS要求更多的SystemV共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。

须知

max_connections取值的设置受**max_prepared_transactions**的影响，在设置max_connections之前，应确保max_prepared_transactions的值大于或等于max_connections的值，这样可确保每个会话都有一个等待中的预备事务。

application_name

参数说明：连接数据库的客户端程序名称。

参数类型：USERSET

取值范围：字符串。

默认值：gsql

connection_info

参数说明：连接数据库的驱动类型、驱动版本号、当前驱动的部署路径和进程属主用户。（运维类参数，不建议用户设置）

参数类型：USERSET

取值范围：字符串

默认值：空字符串

 版权所有 © 华为云计算技术有限公司

说明

- 空字符串，表示当前连接数据库的驱动不支持自动设置connection_info参数或应用程序未设置。
- 驱动连接数据库的时候自行拼接的connection_info参数格式如下：
  ```
  {"driver_name":"ODBC","driver_version": "(GaussDB x.x.x build 39137c2d) compiled at 2022-09-23 15:43:11 commit 3629 last mr 5138 debug","driver_path":"/usr/local/lib/psqlodbcw.so","os_user":"omm"}
  ```
  ODBC，JDBC，gsql连接默认显示driver_name和driver_version，driver_path，os_user，其他接口连接默认显示driver_name和driver_version，driver_path和os_user的显示由用户控制（参见**连接数据库**和**Linux下配置数据源**）。

19.4.2 安全和认证（postgresql.conf）

介绍设置客户端和服务器的安全认证方式的相关参数。

session_timeout

参数说明：表明与服务器建立连接后，不进行任何操作的最长时间。

参数类型：USERSET

取值范围：整型，0-86400，最小单位为秒（s），0表示关闭超时设置。

默认值：10min

须知

- DWS gsql客户端中有自动重连机制，所以针对初始化用户本地连接，超时后gsql表现的现象为断开后重连。
- pooler连接池到其它CN和DN的连接，不受session_timeout参数控制。

ssl_renegotiation_limit

参数说明：指定在会话密钥重新协商之前，通过SSL加密通道可以传输的流量。这个重新协商流量限制机制可以减少攻击者针对大量数据使用密码分析法破解密钥的几率，但是也带来较大的性能损失。流量是指发送和接受的流量总和。

参数类型：USERSET

说明

参数建议保持默认设置，即禁用重协商机制。不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置ssl_renegotiation_limit参数，即使设置也不会生效。

取值范围：整型，0~INT_MAX，单位为KB。其中0表示禁用重新协商机制。

默认值：0

failed_login_attempts

参数说明：在任意时候，如果输入密码错误的次数达到failed_login_attempts则当前账户被锁定，password_lock_time秒后被自动解锁。 例如，登录时输入密码失败，ALTER USER时修改密码失败等。

版权所有 © 华为云计算技术有限公司

参数类型：SIGHUP

取值范围：整型，0~1000

- 0，表示自动锁定功能不生效。
- 正整数，表示当错误密码次数达到failed_login_attempts设定的值时，当前账户将被锁定。

默认值：10

须知

- failed_login_attempts和password_lock_time必须都为正数时锁定和解锁功能才能生效。
- failed_login_attempts会与客户端SSL连接模式共同决定用户的密码错误次数。当PGSSLMODE取值是allow或prefer时，客户的一次密码连接请求会生成两次连接请求：一次是尝试SSL连接，另一次是尝试非SSL连接。此时，用户感知到的密码错误次数是failed_login_attempts除以2。

19.4.3 通信库参数

本节介绍通信库相关的参数设置及取值范围等内容。

tcp_keepalives_idle

参数说明：在支持TCP_KEEPIDLE套接字选项的系统上，设置发送活跃信号的间隔秒数。不设置发送保持活跃信号，连接就会处于闲置状态。

参数类型：USERSET

须知

- 如果操作系统不支持TCP_KEEPIDLE选项 ，这个参数的值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。

取值范围：整型，0~3600，单位为秒（s）。

默认值：0

tcp_keepalives_interval

参数说明：在支持TCP_KEEPINTVL套接字选项的操作系统上，以秒数声明在重新传输之间等待响应的时间。

参数类型：USERSET

取值范围：整型，0~180，单位为秒（s）。

默认值：0

 版权所有 © 华为云计算技术有限公司

须知

- 如果操作系统不支持TCP_KEEPINTVL选项，这个参数的值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。

tcp_keepalives_count

参数说明：在支持TCP_KEEPCNT套接字选项的操作系统上，设置DWS服务端在断开与客户端连接之前可以等待的保持活跃信号个数。

参数类型：USERSET

须知

- 如果操作系统不支持TCP_KEEPCNT选项，这个参数的值必须为0。
- 在通过Unix域套接字进行连接的操作系统上，这个参数将被忽略。

取值范围：整型，0~100，其中0表示DWS未收到客户端反馈的保持活跃信号则立即断开连接。

默认值：0

comm_max_datanode

参数说明：通信库支持的最大DN数。

参数类型：USERSET

取值范围：整型，1~8192

默认值：实际DN数

须知

该参数值的DN数由小调大时立即生效，由大调小时需重启集群生效。

comm_max_stream

参数说明：通信库缓存的最大逻辑连接数据结构量。

参数类型：SIGHUP

取值范围：整型，1~65535

默认值：1024

说明

如果**comm_max_datanode**参数值较小，进程内存充足，可以适当将comm_max_stream值调大。

 版权所有 © 华为云计算技术有限公司

max_stream_pool

参数说明：设置stream线程池能够容纳stream线程的最大个数。该参数8.1.2及以上版本支持。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，-1和0表示关闭stream线程池。

默认值：

- 新安装集群默认值计算公式为：max_stream_pool=MIN(max_connections, max_process_memory/16/5MB, 1024)
- 旧集群升级到8.3.0.100及以上新集群版本的默认值计算公式为：max_stream_pool=MIN(max_connections, max_process_memory/16/5MB, 1024，旧集群值)，升级时也强制应用新装集群的设置值，当旧值小时取旧值。

说明

- 支持实时减少线程池中stream线程的个数；若增大该参数值，stream线程的个数由业务驱动向上增长。
- 通常情况下不建议调整该参数值，因为stream线程池有自动清理功能。
- 如果stream空闲线程过多造成内存占用，可以调小此值以节约内存。

enable_stream_sync_quit

参数说明：设置stream计划结束时，stream线程是否同步退出。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示stream计划结束后，等待stream线程组内的线程统一退出。
- off表示stream计划结束后，stream线程直接退出，无需等待stream线程组内的线程统一退出。

默认值：off

enable_connect_standby

参数说明：设置CN连接DN备机。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示CN连接备机。
- off表示CN连接DN主机。

默认值：off

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- 不建议在日常业务中使用此参数，仅适用于运维操作，更不建议使用gs_guc工具进行全局设置。否则可能会导致数据不一致，结果集错误等问题。
- 已创建临时表的session打开此参数后将造成DN上的临时表数据丢失，后续无法进行临时表相关操作。

comm_quota_size

参数说明：通信库最大可连续发送包总大小。使用1GE网卡时，建议取较小值，推荐设置为20KB~40KB。

参数类型：USERSET

取值范围：整型，0~102400，默认单位为KB。0表示不使用quota机制。

默认值：1MB

comm_usable_memory

参数说明：单个DN内通信库缓存最大可使用内存。

参数类型：SIGHUP

取值范围：整型，1GB~256GB，默认单位为GB。安装时最小值不得小于1GB。

默认值：max_process_memory/8

须知

此参数需根据环境内存及部署方式具体配置，过大会造成OOM，过小会降低通信库性能。

comm_client_bind

参数说明：通信库客户端发起连接时是否使用bind绑定指定IP。

参数类型：USERSET

取值范围：布尔型

- on表示绑定指定IP。
- off表示不绑定指定IP。

须知

如果集群某一节点存在多个IP处于同一通信网段时，需设置为on。此时将绑定本地listen_addresses指定的IP发起通信，随机端口号不能重复使用，集群并发数量会受到可用随机端口号数量的限制。

默认值：off

版权所有 © 华为云计算技术有限公司

comm_no_delay

参数说明：是否使用通信库连接的NO_DELAY属性，重启集群生效。

参数类型：USERSET

取值范围：布尔型

默认值：off

须知

如果集群出现因每秒接收数据包过多导致的丢包时，需设置为off，以便小包合并成大包发送，减少数据包总数。

comm_debug_mode

参数说明：通信库debug模式开关，该参数设置是否打印通信层详细日志，session级别生效。

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细debug日志。
- off表示不打印通信库详细debug日志。

默认值：off

comm_ackchk_time

参数说明：无数据包接收情况下，该参数设置通信库服务端主动ACK触发时长。

参数类型：USERSET

取值范围：整型，0~20000，单位为毫秒（ms）。取值为0表示关闭此功能。

默认值：2000

comm_timer_mode

参数说明：通信库timer模式开关，该参数设置是否打印通信层各阶段时间桩，session级别生效。

 版权所有 © 华为云计算技术有限公司

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细时间桩日志。
- off表示不打印通信库详细时间桩日志。

默认值：off

comm_stat_mode

参数说明：通信库stat模式开关，该参数设置是否打印通信层的统计信息，session级别生效。

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库统计信息日志。
- off表示不打印通信库统计信息日志。

默认值：off

client_connection_check_interval

参数说明：客户端连接状态检测时间间隔。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒。0表示不检测客户端连接状态。

默认值：10000

须知

通过gsql/jdbc/odbc等客户端直连CN执行长查询，在长查询执行期间：

- CN每隔client_connection_check_interval时间检测一次客户端连接状态，若检测到客户端与CN的连接已经断开，则服务端主动终止长查询的执行，释放相关资源，避免集群资源浪费。
- DN每隔client_connection_check_interval时间检测一次CN与DN的连接状态，若检测到CN与DN的连接已经断开，则DN主动终止长查询的执行，释放相关资源，避免集群资源浪费。

 版权所有 © 华为云计算技术有限公司

conn_recycle_timeout

参数说明：用于控制将CN和其他节点间空闲连接回收到连接池的时间间隔。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0 ~ 3600，单位为秒(s)。0表示关闭空闲连接回收功能。

默认值：30

19.5 资源消耗

19.5.1 内存

介绍与内存相关的参数设置。

须知

本节涉及的参数仅在数据库服务重新启动后生效。

max_process_memory

参数说明：设置一个数据库节点可用的最大物理内存。

参数类型：SIGHUP

取值范围：整型，2*1024*1024～INT_MAX/2，单位为KB。

默认值：非从备DN节点自动适配，一个机器部署多个DN情况下，公式为（物理内存大小）* 0.8 / (1+主DN个数)；一个机器部署单个DN情况下，公式为（物理内存大小）* 0.6；当结果不足2GB时，默认取2GB。从备DN默认为12GB。

设置建议：

⚠ 注意

单机部署的场景仅适用于存量用户。

- DN上该数值需要根据系统物理内存及单节点部署主DN个数决定的。

  一个机器部署多个DN情况下，max_process_memory计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.8 / (n+主DN个数)；一个机器部署单个DN情况下，max_process_memory计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.6。

  该参数目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min_free_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即，max_process_memory=物理内存*0.8/(n+主DN个数)，其中，当集群规模小于256时， n=1；当集群规模大于256且小于512时， n=2；当集群规模超过512时，n=3。

 版权所有 © 华为云计算技术有限公司

- 不推荐设置为最小阈值。
- CN上该数值内存可设置与DN数值一样。
- RAM：集群规划时分配给集群的最大使用内存。
- DWS从8.2.0版本开始，为了提升内存资源利用率，增大了单机器单DN部署形态的max_process_memory初始值，但当出现集群状态不均衡的情况下，两个主DN节点会同时在一个机器上，如果仍然使用max_process_memory初始值，机器可能出现OOM报错的情况。为此，8.2.0版本开始，max_process_memory参数更改为SIGHUP类型，可以通过手动设置的方式来动态调整；同时，新增max_process_memory_auto_adjust参数，当出现集群状态不均衡的情况下，CM会根据集群状态来动态调整max_process_memory，调整max_process_memory的计算公式为：（物理内存大小 – vm.min_free_kbytes）* 0.8 / 主DN个数。
- DWS从8.2.1版本开始，扩大了max_process_memory动态内存调整的适用范围，由原先仅支持一个机器部署单个DN的集群形态，变更为支持所有部署形态。
  - max_process_memory_auto_adjust为on时，max_process_memory将会在上限和下限两值之间动态调整；其中，max_process_memory下限值的计算公式为：（物理内存大小）* 0.8 / (1+主DN个数)；max_process_memory上限值由GUC参数max_process_memory_balanced设置（max_process_memory_balanced参数设置请联系技术支持工程师）。
  - 集群在均衡模式下，max_process_memory将使用上限值，提高节点整体的内存资源利用率，相对之前版本，内存利用率会提升。
  - 集群在非均衡模式下，max_process_memory将使用下限值，节点整体的内存资源利用率和8.2.1集群之前的版本一致。
  - 升级场景下，该参数为保持前向兼容，系统不会设置max_process_memory_balanced，max_process_memory默认使用升级前设置的值。

max_process_memory_auto_adjust

参数说明：设置是否开启max_process_memory参数的自动调整功能（该参数仅8.2.0及以上集群版本支持）。

参数类型：SIGHUP

取值范围：布尔型

默认值：on

shared_buffers

参数说明：设置DWS使用的共享内存大小。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。

参数类型：POSTMASTER

取值范围：整型，128～INT_MAX，单位为8KB。

改变BLCKSZ的值会改变最小值。

默认值：CN节点为DN节点值的1/2，DN节点取公式计算：POWER(2,ROUND(LOG(2,max_process_memory/18),0))。如果操作系统支持的共享内存小于32MB，则在初始化数据存储区时会自动调整为操作系统支持的最大值。

设置建议：

 版权所有 © 华为云计算技术有限公司

由于DWS大部分查询下推，建议DN中此参数设置比CN大。

建议设置shared_buffers值为内存的40%以内。行存列存分开对待。行存设大，列存设小。列存：(单服务器内存/单服务器DN个数)*0.4*0.25。

如果设置较大的shared_buffers需要同时增加checkpoint_segments的值，因为写入大量新增、修改数据需要消耗更多的时间周期。

bulk_write_ring_size

参数说明：数据并行导入使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，16384～INT_MAX，单位为KB。

默认值：2GB

设置建议：建议导入压力大的场景中增加DN中此参数配置。

buffer_ring_ratio

参数说明：设置并行导出时使用环形缓冲区的阈值大小。

参数类型：USERSET

取值范围：整型，1～1000

默认值：250

说明

- 默认值表示阈值为shared_buffers的250/1000即1/4。
- 最小为shared_buffers的1/1000。
- 最大为shared_buffers的大小。

设置建议：导出时出现缓存命中率不符合预期的场景建议在DN中设置此参数。

enable_cstore_ring_buffer

参数说明：设置列存RingBuffer的开关，该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

默认值：off

设置建议：建议在业务运行一段时间，当CStoreBuffer内已存入大量高频查询的数据后，进行低频大表查询前打开，查询完大表后关闭。

temp_buffers

参数说明：设置每个数据库会话使用的LOCAL临时缓冲区的大小。

参数类型：USERSET

取值范围：整型，800~INT_MAX/2，单位为8KB。

 版权所有 © 华为云计算技术有限公司

默认值：8MB

说明

- 在每个会话的第一次使用临时表之前可以改变temp_buffers的值，之后的设置将是无效的。
- 一个会话将按照temp_buffers给出的限制，根据需要分配临时缓冲区。如果在一个并不需要大量临时缓冲区的会话里设置一个大的数值，其开销只是一个缓冲区描述符的大小。当缓冲区被使用，就会额外消耗8192字节。

max_prepared_transactions

参数说明：设置可以同时处于"预备"状态的事务的最大数目。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。

当DWS部署为主备双机时，在备机上此参数的设置必须要高于或等于主机上的，否则无法在备机上进行查询操作。

参数类型：POSTMASTER

取值范围：整型，0~536870911，其中CN取值为0表示关闭预备事务的特性。

默认值：CN节点为800， DN节点为5000

说明

为避免在准备步骤失败，此参数的值不能小于**max_connections**。

work_mem

参数说明：设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。ORDER BY，DISTINCT和merge joins都要用到排序操作。Hash表在散列连接、散列为基础的聚集、散列为基础的IN子查询处理中都要用到。

对于复杂的查询，可能会同时并发运行好几个排序或者散列操作，每个都可以使用此参数所声明的内存量，不足时会使用临时文件。同样，好几个正在运行的会话可能会同时进行排序操作。因此使用的总内存可能是work_mem的好几倍。

参数类型：USERSET

取值范围：整型，64~INT_MAX，单位为KB。

默认值：小规格内存为512MB， 大规格内存为2GB（**max_process_memory**大于等于30GB为大规格内存，否则为小规格内存）。

设置建议：

依据查询特点和并发来确定，一旦work_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。

- 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work_mem=50%内存/10。
- 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work_mem=50%内存/5。
- 对于并发场景，建议work_mem=串行下的work_mem/物理并发数。

 版权所有 © 华为云计算技术有限公司

须知

开启内存自适应后，收集统计信息后不再需要使用work_mem进行算子内存使用调优，由系统根据当前负载情况，为每个语句生成计划，并估算每个算子的内存使用量和整个语句的内存使用量。系统根据负载情况和整个语句内存使用量进行队列调度，所以多并发场景会出现语句排队的情况。

query_mem

参数说明：设置执行作业所使用的内存。如果设置的query_mem值大于0，在生成执行计划时，优化器会将作业的估算内存调整为该值。

参数类型：USERSET

取值范围：整型，0，或大于32MB的整型，默认单位为KB。如果设置值小于32MB，系统会自动将该参数设置为默认值0，此时优化器不会根据该值调整作业的估算内存。

默认值：0

query_max_mem

参数说明：设置执行作业所能够使用的最大内存。如果设置的query_max_mem值大于0，在生成执行计划时，优化器会根据该值来设置算子的可用内存。当作业执行时所使用内存超过该值时，将报错退出。

参数类型：USERSET

取值范围：整型，0，或大于32MB的整型，单位为KB。如果设置值为小于32MB，系统会自动将该参数设置为默认值0，此时优化器不会根据该值限制作业的内存使用。

默认值：0

agg_max_mem

参数说明：设置执行作业中的Agg算子的聚集列超过5列时，该Agg算子所能够使用的最大内存。当agg_max_mem大于0时生效。（该参数仅8.1.3.200及以上集群版本支持）

参数类型：USERSET

取值范围：整型，0，或大于32MB的整型，单位为KB。如果设置值小于32MB，系统会自动将该参数设置为默认值0，此时不会根据该值限制Agg算子的内存使用。

默认值：

- 若当前集群为低版本升级到8.1.3及以上版本，继承升级前参数，默认值为INT_MAX。
- 若当前集群为新装的8.1.3及以上版本，默认值为2GB。

enable_rowagg_memory_control

参数说明：控制行存agg算子内存使用上限。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示开启控制行存agg算子内存使用限制。设置为on可避免行存agg内存过度使用导致系统内存不可用，但可能导致agg性能劣化。
- off表示关闭控制行存agg算子内存使用限制。设置为off可能导致系统内存不可用。

默认值：on

maintenance_work_mem

参数说明：设置在维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。该参数的设置会影响VACUUM、VACUUM FULL、CLUSTER、CREATE INDEX的执行效率。

参数类型：USERSET

取值范围：整型，1024~INT_MAX，单位为KB。

默认值：小规格内存为512MB， 大规格内存为2GB（**max_process_memory**大于等于30GB为大规格内存，否则为小规格内存）。

设置建议：

- 建议设置此参数的值等于**work_mem**，可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。
- 当**自动清理**进程运行时，**autovacuum_max_workers**倍数的内存将会被分配，所以此时设置maintenance_work_mem的值应该不小于**work_mem**。
- 如果进行大数据量的cluster等，可以在session中调大该值。

psort_work_mem

参数说明：设置列存表在进行局部排序中在开始写入临时磁盘文件之前使用的内存大小。带partial cluster key的表、带索引的表插入，创建表索引，删除表和更新表都会用到。

参数类型：USERSET

须知

多个正在运行的会话可能会同时进行表的局部排序操作，因此使用的总内存可能是psort_work_mem的好几倍。

取值范围：整型，64~INT_MAX，单位为KB。

默认值：512MB

max_loaded_cudesc

参数说明：设置列存表在做扫描时，每列缓存cudesc信息的个数。增大设置会提高查询性能，但也会增加内存占用，特别是当列存表的列非常多时。

参数类型：USERSET

取值范围：整型，100~INT_MAX/2

 版权所有 © 华为云计算技术有限公司

默认值：1024

须知

max_loaded_cudesc设置过高时，有可能引起内存分配不足。

max_stack_depth

参数说明：设置DWS执行堆栈的最大安全深度。需要这个安全界限是因为在服务器里，并非所有程序都检查了堆栈深度，只是在可能递规的过程，比如表达式计算这样的过程里面才进行检查。

参数类型：SUSET

设置原则：

- 此参数的最佳设置是等于操作系统内核允许的最大值（就是ulimit -s的设置）。
- 如果设置此参数的值大于实际的内核限制，则一个正在运行的递归函数可能会导致一个独立的服务器进程崩溃。在DWS能够检测内核限制的操作系统上（SLES上），将自动限制设置为一个不安全的值。
- 因为并非所有的操作都能够检测，所以建议用户在此设置一个明确的值。

取值范围：整型，100~INT_MAX，单位为KB。

默认值：2MB

说明

默认值2MB，这个值相对比较小，不容易导致系统崩溃。但是可能会因为该值较小，导致无法执行复杂的函数。

cstore_buffers

参数说明：设置列存和OBS、HDFS外表列存格式（orc、parquet、carbondata）所使用的共享缓冲区的大小。

参数类型：POSTMASTER

取值范围：整型，16384～INT_MAX，单位为KB。

默认值：CN为32MB，DN取公式计算：
POWER(2,ROUND(LOG(2,max_process_memory/18),0))

设置建议：

列存表使用cstore_buffers设置的共享缓冲区，几乎不用**shared_buffers**。因此在列存表为主的场景中，应减少shared_buffers，增加cstore_buffers。

OBS、HDFS外表使用cstore_buffers设置ORC、Parquet、Carbondata的元数据和数据的缓存，元数据缓存大小为cstore_buffers的1/4，最大不超过2GB，其余缓存空间为列存数据和外表列存格式数据共享使用。

dfs_max_memory

参数说明：设置orc导出时能占用的最大内存。如果导出大宽表时出现内存不足问题，可以调大此参数后重试。该参数仅8.3.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，131072 ~ 10485760，单位为KB

默认值：262144KB（256MB）

schedule_splits_threshold

参数说明：设置HDFS外表schedule阶段能够在内存中存储的最大文件个数，超过该限制时，将把文件列表下盘处理。

参数类型：USERSET

取值范围：整型，1~INT_MAX

默认值：60000

bulk_read_ring_size

参数说明：并行导出，使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，256~INT_MAX，单位为KB。

默认值：16MB

check_cu_size_threshold

参数说明：列存表插入数据时，如果一个CU中已插入的数据量大于该参数时，开始进行行级大小校验，避免生成非压缩态大于1G的CU。

参数类型：USERSET

取值范围：整型，0~1048576，单位为KB。

默认值：1GB

须知

在进行行级大小校验时，如果多次校验不通过，建议先会话级设置该参数为0进行规避。

memory_spread_strategy

参数说明：调整自定义资源池DN内存扩展的策略。建议在内存充裕的业务场景进行调整，该参数设置后会提升查询性能，内存最高可以和默认资源池保持一致，但若业务场景中内存较小的情况下可能出现内存不足报错。该参数仅8.1.3及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none表示不扩展内存

 版权所有 © 华为云计算技术有限公司

- negative表示按需扩展内存，算子按需扩展，估算的多扩展多，估算的小扩展的小。
- crazy表示直接扩展内存，等同于默认资源池的内存扩展策略，但在业务场景内存较小的情况下可能出现内存不足报错。

默认值：none

19.5.2 语句磁盘空间管控

介绍与语句磁盘空间管控相关的参数，用于限制语句磁盘空间使用。

sql_use_spacelimit

参数说明：限制单个SQL在单个DN上，触发写盘操作时，所有类型写盘文件的总空间大小，管控的空间包括普通表、临时表以及中间结果集落盘占用的空间。系统管理员用户也受该参数限制。

参数类型：USERSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

默认值：配置sql_use_spacelimit为实例所在磁盘空间总容量的10%。

说明

例如，执行语句中配置参数sql_use_spacelimit=100，当出现单DN写盘超过100kB时，DWS会主动终止该query的运行，并提示用户单DN写盘量超阈值。

```
insert into user1.t1 select * from user2.t1;
ERROR:  The space used on DN (104 kB) has exceeded the sql use space limit (100 kB).
```

建议处理方式：

- 优化语句，减少语句写盘占用空间。
- 如果磁盘空间充足可以适当调大该参数。

temp_file_limit

参数说明：语句执行过程中触发落盘操作时，限制语句中单个线程落盘文件的总空间大小。例如，排序和哈希表使用的临时文件或者游标占用的临时文件。

此设置为会话级别的落盘文件控制。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

默认值：配置temp_file_limit为实例所在磁盘空间总容量的10%。

须知

SQL查询执行时使用的临时表空间不在此限制。

bi_page_reuse_factor

参数说明：行存表批量插入场景下，主备DN使用页复制进行数据同步时，可以复用的旧页面空闲空间的百分比。

版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0~100，单位为%。其中0表示不对页面进行复用，全部申请新页面。

默认值：70

须知

- 不建议将此值设置为50以下（0除外），如果复用页面的空闲空间较小的话，会使主备DN间传输过多的旧页面数据，从而导致批量插入性能下降。
- 不建议将此值设置为90以上，如果此值设置过高，会导致频繁查询空闲页面，但又无法复用旧页面，得不偿失。

19.5.3 内核资源使用

介绍与操作系统内核相关的参数，这些参数是否生效依赖于操作系统的设置。

max_files_per_node

参数说明：限制单个节点上单个SQL打开的文件最大数量。通常情况下，不需要设置此参数。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，其中-1表示限制最大数量。

默认值：50000

说明

- 新安装9.1.0及以上集群版本中该参数的默认值为50000。
- 升级场景下若原集群已支持max_files_per_node参数，该参数的默认值为保持前向兼容维持原值。
- 升级场景下若原集群未支持max_files_per_node参数，升级后集群中该参数的默认值为-1。
- 执行语句报错“The last file name is [%s] and %d files have already been opened on data node [%s] with a maximum of %d files.”时，请尝试增大max_files_per_node值。

19.5.4 基于开销的清理延迟

基于开销的清理延迟的目的是允许管理员减少VACUUM和ANALYZE语句在并发活动的数据库上的I/O影响。 例如，VACUUM和ANALYZE此类维护语句并不需要迅速完成，且运行时要求此类语句不能严重干扰系统执行其他的数据库操作，管理员就可以使用该功能实现此目的。

 版权所有 © 华为云计算技术有限公司

须知

有些清理操作会持有关键的锁，这些操作应该尽快结束并释放锁。所以DWS的机制是，在这类操作过程中，基于开销的清理延迟不会发生作用。为了避免在这种情况下的长延时，实际的开销限制取下面两者之间的较大值：

- vacuum_cost_delay*accumulated_balance/vacuum_cost_limit
- vacuum_cost_delay*4

在ANALYZE | ANALYSE和VACUUM语句执行过程中，系统维护一个内部的记数器，跟踪所执行的各种I/O操作的近似开销。如果积累的开销达到了vacuum_cost_limit声明的限制，则执行这个操作的进程将睡眠vacuum_cost_delay指定的时间。然后它会重置记数器然后继续执行。

这个特性是缺省关闭的。要想打开它，把vacuum_cost_delay变量设置为一个非零值。

vacuum_cost_delay

参数说明：指定开销超过vacuum_cost_limit的值时，进程睡眠的时间。

参数类型：USERSET

取值范围：整型，0~100，单位为毫秒（ms）。正数值表示打开基于开销的清理延迟特性；0表示关闭基于开销的清理延迟特性。

默认值：0

须知

- 许多系统上，睡眠的有效分辨率是10毫秒。因此把vacuum_cost_delay设置为一个不是10的整数倍的数值与将它设置为下一个10的整数倍作用相同。
- 此参数一般设置较小，常见的设置是10或20毫秒。调整此特性资源占用率时，建议是调整其他参数，而不是该参数。

vacuum_cost_page_hit

参数说明：清理一个在共享缓存里找到的缓冲区的预计开销。它代表锁住缓冲池、查找共享的Hash表、扫描页面内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

默认值：1

vacuum_cost_page_miss

参数说明：清理一个要从磁盘上读取的缓冲区的预计开销。它代表锁住缓冲池、查找共享Hash表、从磁盘读取需要的数据块、扫描它的内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

版权所有 © 华为云计算技术有限公司

默认值：2

vacuum_cost_page_dirty

参数说明：清理修改一个原先是干净的块的预计开销。它代表把一个脏的磁盘块再次刷新到磁盘上的额外开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

默认值：20

vacuum_cost_limit

参数说明：导致清理进程休眠的开销限制。

参数类型：USERSET

取值范围：整型，1~10000，单位为毫秒（ms）。

默认值：200

19.5.5 异步 IO

enable_adio_debug

参数说明：设置是否开启ADIO相关的日志，便于定位ADIO相关问题。运维类参数不建议普通用户设置。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此日志开关。
- off/false表示关闭此日志开关。

默认值：off

enable_fast_allocate

参数说明：磁盘空间快速分配开关。只有在XFS文件系统上才能开启该开关。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此功能。
- off/false表示关闭此功能。

默认值：off

prefetch_quantity

参数说明：描述行存储使用ADIO预读取IO量的大小。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：整型，1024～1048576，单位为8KB。

默认值：32MB

backwrite_quantity

参数说明：描述行存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为8KB。

默认值：8MB

cstore_prefetch_quantity

参数说明：描述列存储使用ADIO预取IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：32MB

cstore_backwrite_quantity

参数说明：描述列存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

cstore_backwrite_max_threshold

参数说明：描述列存储使用ADIO写入数据库可缓存最大的IO量。

参数类型：USERSET

取值范围：整型，4096～INT_MAX/2，单位为KB。

默认值：2GB

fast_extend_file_size

参数说明：描述列存储使用ADIO预扩展磁盘的大小。

参数类型：SUSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

effective_io_concurrency

参数说明：磁盘子系统可以同时有效处理的请求数。对于RAID阵列，此参数应该是阵列中驱动器主轴的数量。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0~1000

默认值：1

cu_preload_max_distance

参数说明：用于控制预读CU Group个数的上限。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型， 0~1024，其中0表示关闭预读功能。

默认值：20

cu_preload_count

参数说明：设置预读CU个数的最大值。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型。0~10000，其中0表示关闭预读功能。

默认值：600

19.5.6 磁盘缓存

以下参数仅9.1.0及以上集群版本支持。

enable_disk_cache

参数说明：控制是否打开文件缓存。该参数受enable_aio_scheduler和obs_worker_pool_size的前置控制；只有在enable_aio_scheduler=on以及obs_worker_pool_size>=4的前提下，此参数为on才会生效。

参数类型：USERSET

取值范围：布尔型

默认值： off

enable_disk_cache_recovery

参数说明：控制是否允许在重启集群时恢复文件缓存。

参数类型：USERSET

取值范围：布尔型

默认值： off

19.6 并行导入

DWS提供了并行导入功能，以快速、高效地完成大量数据导入。介绍DWS并行导入的相关参数。

 版权所有 © 华为云计算技术有限公司

raise_errors_if_no_files

参数说明：导入时是否区分“导入文件记录数为空”和“导入文件不存在”。raise_errors_if_no_files=TRUE，则“导入文件不存在”的时候，DWS将上报“文件不存在的”错误。

参数类型：SUSET

取值范围：布尔型

- on表示导入时区分“导入文件记录数为空”和“导入文件不存在”。
- off表示导入时不区分“导入文件记录数为空”和“导入文件不存在”。

默认值：off

partition_max_cache_size

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。通过partition_max_cache_size可指定数据缓存区大小。该值设置过大，将消耗较多系统内存资源；设置过小，将降低列存分区表批量插入性能。

参数类型：USERSET

取值范围：4096~ INT_MAX / 2，最小单位为KB。

默认值：2GB

partition_mem_batch

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。在partition_max_cache_size设置的情况下，通过partition_mem_batch可指定缓存个数。该参数值设置越大，每个分区可用的缓存越小，降低列存分区表批量插入性能；设置越小，每个分区可用的缓存会越大，会消耗较多系统内存资源。

参数类型：USERSET

取值范围：1~ 65535

默认值：256

gds_debug_mod

参数说明：为了增强对数据并行加载工具General Data Service（以下简称GDS）相关问题的分析定位能力，可以通过此参数选择是否开启GDS的debug功能。参数开启后，将在集群节点对应的日志中输出GDS每次收发的包裹类型、命令交互的对端以及其他交互相关的细节信息，方便记录Gaussdb端状态机的状态跳转，以及目前所处的状态信息。此参数打开会输出额外日志，增加日志IO开销，进而影响性能和日志的信息有效性，因此请仅在定位GDS问题时开启。

参数类型：USERSET

取值范围：布尔型

- on表示开启GDS debug功能。
- off表示不开启GDS debug功能。

默认值：off

 版权所有 © 华为云计算技术有限公司

max_copy_data_display

参数说明：为copy错误表rawrecord字段长度增加guc管控，该字段为text类型，最大为1G减去8203B（即1073733621B）。该参数仅8.2.1.100及以上版本支持。

设置该参数时，表示能显示的最长字符数，超过该长度结尾显示‘...’。

参数类型：USERSET

取值范围：0~1073733616

默认值：1024

enable_parallel_batch_insert

参数说明：控制行存表和列存表是否开启并发入库功能。该参数仅8.2.0.102及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启并发入库功能。开启并发入库功能可以有效提升行存表和列存表insert、upsert、update和delete语句入库执行速度。
- off表示不开启并发入库功能。

默认值：off

19.7 预写式日志

19.7.1 设置

synchronous_commit

参数说明：设置当前事务的同步方式。

参数类型：USERSET

取值范围：枚举类型

- on表示将备机的同步日志刷新到磁盘。
- off表示异步提交。
- local表示为本地提交。
- remote_write表示要备机的同步日志写到磁盘。
- remote_receive表示要备机同步日志接收数据。

默认值：on

wal_skip_fpw_level

参数说明：设置在检查点（checkpoint）之后DN节点对页面的第一次修改，会将整个页面的全部内容写到WAL日志中。该参数用于控制是否将各类型FPW记录到WAL日志中，以避免WAL日志过多导致空间不足或主备同步过慢的问题。该参数8.3.0及以上集群版本支持，8.3.0以下版本请参考write_fpi_hint。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型

- 0：打开全部FPW。
- 1：设置heap页面元组的hintbits标记位时，如果是checkpoint后的第一次修改，则不写FPW日志。此功能是否生效不受enable_crc_check及wal_log_hints参数影响。设置为1时等效于低版本的write_fpi_hint=off。参考write_fpi_hint。
- 2：不记录索引的FPW日志和heap页面hintbits标记位被修改后的FPW日志。

默认值：1

wal_decelerate_policy

参数说明：触发限速后的行为策略，该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- warning表示触发限速条件后仅告警，但不降速。
- decelerate触发限速条件后，根据配速对当前业务进行降速处理。

默认值：warning

说明

warning不影响性能，decelerate会根据配速，对当前业务进行降速处理。

wal_write_speed

参数说明：单DN的每个query每秒最大允许作业触发WAL写入速率（byte/s），该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1024～10240000，单位为KB

默认值：30MB

说明

对于带索引Copy、Delete等大量的作业，会根据配速进行作业速率限制。

wal_decelerate_trigger_threshold

参数说明：单DN每个query触发wal写入限速的阈值，该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1024～100000000000，单位为KB

默认值：128MB

 版权所有 © 华为云计算技术有限公司

说明

对于单query产生的xlog量大于该参数设置值时，即触发wal写入限速，对于DDL、少量的DML操作无影响。

commit_delay

参数说明：表示一个已经提交的数据在WAL缓冲区中存放的时间。

参数类型：USERSET

取值范围：整型， 0～100000（微秒），其中0表示无延迟。

默认值：0

须知

- 设置为非 0 值时事务执行commit后不会立即写入WAL中，而仍存放在WAL缓冲区中，等待WalWriter进程周期性写入磁盘。
- 如果系统负载很高，在延迟时间内，其他事务可能已经准备好提交。但如果没有事务准备提交，这个延迟就是在浪费时间。

commit_siblings

参数说明：当一个事务发出提交请求时，如果数据库中正在执行的事务数量大于此参数的值，则该事务将等待一段时间（**commit_delay**的值），否则该事务则直接写入WAL。

参数类型：USERSET

取值范围：整型， 0～1000

默认值：5

wal_compression

参数说明：控制是否对FPI页面进行压缩。

参数类型：USERSET

取值范围：布尔型

- on表示开启FPI压缩。
- off表示关闭FPI压缩。

默认值：on

 版权所有 © 华为云计算技术有限公司

须知

- 当前压缩算法为zlib，暂不支持设置为其他压缩算法。
- 对于通过从低版本升级成为当前版本的集群，此参数默认关闭（off）。如果用户需要，可以通过gs_guc命令打开FPI压缩功能。
- 当前版本若为全新安装版本，此参数默认打开（on）。
- 从低版本升级上来的集群，如果手动开启了此参数，不允许再进行集群回滚操作。

wal_compression_level

参数说明：当打开wal_compression参数时，设置zlib压缩算法的压缩级别。

参数类型：USERSET

取值范围：整型， 0～9

- 0表示不压缩。
- 1表示最低的压缩率。
- 9表示最高的压缩率。

默认值：9

19.7.2 检查点

checkpoint_segments

参数说明：设置checkpoint_timeout周期内所保留的最少WAL日志段文件数量。每个日志文件大小为16MB。

参数类型：SIGHUP

取值范围：整型，最小值为1。

默认值：64

须知

提升此参数可加快大数据的导入速度，但需要结合checkpoint_timeout、shared_buffers这两个参数统一考虑。这个参数同时影响WAL日志段文件复用数量，通常情况下pg_xlog文件夹下最大的复用文件个数为2倍的checkpoint_segments个，复用的文件被改名为后续即将使用的WAL日志段文件，不会被真正删除。

19.8 双机复制

 版权所有 © 华为云计算技术有限公司

19.8.1 主服务器

enable_data_replicate

参数说明：当数据库在数据导入行存表时，主机与备机的数据同步方式可以进行选择。

参数类型：USERSET

取值范围：布尔型

- on表示导入数据行存表时主备数据采用数据页的方式进行同步。当replication_type参数为1时，不允许设置为on。
- off表示导入数据行存表时主备数据采用日志（Xlog）方式进行同步。

默认值：on

ha_module_debug

参数说明：用于查看数据复制时具体数据块的复制状态日志。

参数类型：USERSET

取值范围：布尔型

- on表示日志中将打印数据复制时每个数据块的状态。
- off表示日志中不打印数据复制时每个数据块的状态。

默认值：off

19.9 查询规划

19.9.1 优化器方法配置

这些配置参数提供了影响查询优化器选择查询规划的原始方法。如果优化器为特定的查询选择的缺省规划并不是最优的，可以通过使用这些配置参数强制优化器选择一个不同的规划来临时解决这个问题。更好的方法包括调节优化器开销常量、手动运行ANALYZE、增加配置参数**default_statistics_target**的值、增加使用ALTER TABLE SET STATISTICS为指定列增加收集的统计信息。

enable_bitmapscan

参数说明：控制优化器对位图扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_hashagg

参数说明：控制优化器对Hash聚集规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_mixedagg

参数说明：控制优化器对Mixed Agg聚集规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用，为符合条件的Grouping Sets语句（包括Rollup或Cube）生成Mixed Agg查询计划。
- off表示不使用。

默认值：on

须知

- 新装9.1.0.200及以上版本集群中该参数的默认值为on，升级场景该参数的默认值为保持前向兼容维持原值。
- 通常在大数量场景（单个DN表的数据量在100GB以上），采用Mixed Agg查询计划可以提升语句的执行性能。

  不支持Mixed Agg的场景如下：
  - GROUP BY子句里列的数据类型不支持哈希操作。
  - 聚集函数中包含DISTINCT去重或ORDER BY排序。
  - 使用GROUPING SETS子句时不包含空的分组。

enable_hashjoin

参数说明：控制优化器对Hash连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_indexscan

参数说明：控制优化器对索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_indexonlyscan

参数说明：控制优化器对仅索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_material

参数说明：控制优化器对实体化的使用。消除整个实体化是不可能的，但是可以关闭这个变量以防止优化器插入实体节点。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_mergejoin

参数说明：控制优化器对融合连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_nestloop

参数说明：控制优化器对内表全表扫描嵌套循环连接规划类型的使用。完全消除嵌套循环连接是不可能的，但是关闭这个变量就会让优化器在存在其他方法的时候优先选择其他方法。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_index_nestloop

参数说明：控制优化器对内表参数化索引扫描嵌套循环连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：新安装集群的场景下是on，如果集群是从R8C10版本升级上来的，则保持前向兼容。如果是从R7C10或者更早的版本升级上来，则默认值是off。

left_join_estimation_enhancement

参数说明：控制left join的行数估计值是否采用优化后的值。该参数仅8.3.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_seqscan

参数说明：控制优化器对顺序扫描规划类型的使用。完全消除顺序扫描是不可能的，但是关闭这个变量会让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_sort

参数说明：控制优化器使用的排序步骤。完全消除明确的排序是不可能的，但是关闭这个变量可以让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

max_opt_sort_rows

参数说明：控制order by子句中最大优化的limit+offset行数。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0～INT_MAX

- 取值为0时：表示参数不生效。
- 取值为其他值时：表示order by子句中limit+offset行数小于该值时，优化生效，大于该值时，优化不生效。优化后耗时减小，但内存消耗可能增大。

默认值：0

enable_tidscan

参数说明：控制优化器对TID扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_kill_query

参数说明：CASCADE模式删除用户时，会删除此用户拥有的所有对象。此参数标识是否允许在删除用户的时候，取消锁定此用户所属对象的query。

参数类型：SUSET

取值范围：布尔型

- on表示允许取消锁定。
- off表示不允许取消锁定。

默认值：off

enforce_oracle_behavior

参数说明：控制正则表达式的规则匹配模式。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示正则表达式采用ORACLE格式的匹配规则。
- off表示正则表达式采用POSIX格式的匹配规则。

默认值：on

enable_stream_concurrent_update

参数说明：控制优化器在并发更新场景下对stream的使用，该参数受限于**enable_stream_operator**参数。

参数类型：USERSET

取值范围：布尔型

- on表示允许优化器对update语句生成stream计划。
- off表示优化器对update语句仅能生成非stream计划。

默认值：on

enable_stream_ctescan

参数说明：控制stream计划是否支持ctescan。

参数类型：USERSET

取值范围：布尔型

- on表示stream计划下支持ctescan。
- off表示stream计划下不支持ctescan。

默认值：on

说明

升级场景该参数的默认值为保持前向兼容维持原值。

enable_stream_operator

参数说明：控制优化器对stream的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_stream_recursive

参数说明：控制是否将with recursive关联查询下推DN分布式执行。

该参数在8.2.1.210及以上集群版本中废弃。为了保留前向兼容，可以设置成功，但实际不生效。建议使用**stream_recursive_option**参数。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示支持使用with recursive关联查询下推DN分布式执行。
- off表示不支持使用with recursive下推。

默认值：on

stream_recursive_option

参数说明：控制是否将with recursive关联查询下推DN分布式执行。该参数仅8.2.1.210及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- none表示不支持使用with recursive下推。
- normal表示在单逻辑集群场景下，支持使用with recursive关联查询下推DN分布式执行。
- nodegroup表示在多逻辑集群场景下，支持使用with recursive关联查询下推DN分布式执行。

默认值：normal

enable_value_redistribute

参数说明：控制是否开启生成value redistribute优化计划，8.2.0及以上集群版本中，该参数针对不带Partition by子句的rank、dense_rank、row_number是否生成value redistribute优化计划生效。

参数类型：USERSET

取值范围：布尔型

- on表示支持使用value redistribute生成优化计划。
- off表示不支持使用value redistribute生成优化计划。

默认值：on

max_recursive_times

参数说明：控制with recursive的最大迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX。

默认值：200

enable_vector_engine

参数说明：控制优化器对向量化执行引擎的使用。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示使用。
- off表示不使用。

默认值：on

enable_broadcast

参数说明：控制优化器对stream代价估算时对broadcast分布方式的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_redistribute

参数说明：控制优化器对stream代价估算时对local redistribute和split redistribute分布方式的使用。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_change_hjcost

参数说明：控制优化器在Hash Join代价估算路径选择时，是否使用将内表运行时代价排除在Hash Join节点运行时代价外的估算方式。如果使用，则有利于选择条数少，但运行代价大的表做内表。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_fstream

参数说明：控制优化器下发语句时对stream的使用，该参数只限定于HDFS外表使用。

该参数现在已经废弃。为了保留前向兼容，可以设置成功，但是实际不起任何作用。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示使用。
- off表示不使用。

默认值：off

enable_hashfilter

参数说明：控制是否允许对包含复制表(包括dual表和常量表)的计划生成hashfilter。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许生成hashfilter。
- off表示任何情况下都不允许生成hashfilter。

默认值：on

best_agg_plan

参数说明：对于stream下的Agg操作，优化器会生成三种计划：

1. hashagg+gather(redistribute)+hashagg。
2. redistribute+hashagg(+gather)。
3. hashagg+redistribute+hashagg(+gather)。

本参数用于控制优化器生成哪种hashagg的计划。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为1时，强制生成第一种计划。
- 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
- 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
- 取值为0时，优化器会根据以上三种计划的估算cost选择最优的一种计划生成。

默认值：0

turbo_engine_version

参数说明：对于建表指定turbo存储格式表（表属性中enable_turbo_store参数设置为on），且当查询不涉及merge join或sort agg算子时，执行器可走turbo执行引擎，执行器部分性能可获得成倍性能提升。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为0时，表示turbo执行引擎关闭。
- 取值为1时，表示仅针对单表agg查询场景使用turbo执行引擎。

 版权所有 © 华为云计算技术有限公司

- 取值为2时，表示仅针对单表agg或多表join关联查询场景使用turbo执行引擎。
- 取值为3时，对于大多常用算子可使用turbo执行引擎加速，不支持算子如merge join，sort agg等算子。数据量较大且turbo_engine_version取值为3时，merge join，sort agg算子出现的情况较少，因此基本可以实现任意SQL语句的turbo执行引擎加速。

默认值：3

enable_bucket_stream_opt

参数说明：对于二级分区表或3.0 hash分布表使用bucket agg和bucket join执行策略。通过避免数据的local重分布或广播，提升SQL语句执行效率。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- true，表示优化器在满足策略生效条件时使用bucket agg和bucket join执行策略生成计划。如使用该优化策略，执行explain末尾会显示字段Bucket Stream: true。
- false，表示不使用bucket agg和bucket join执行策略生成计划。

默认值：true

须知

- 新装9.1.0.200及以上版本集群中该参数的默认值为true，升级场景该参数的默认值为保持前向兼容维持原值。
- bucket agg和bucket join执行策略生效要求当前query可用CPU个数小于等于16，且建表满足以下条件之一：

  1. 二级分区要求分布列与二级分区的secondary_part_column相同，推荐二级分区个数为DN个数*12，支持倍数包括4, 6, 8, 12, 16。

  2. 3.0表要求表为hash分布，且bucketnums/DN个数>10。
- 当计划中local stream代价较低，query可能不会选择bucket agg和bucket join执行策略。

spill_compression

参数说明：控制执行器算子运行数据由于内存不足下盘时的压缩算法。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- 'lz4'，表示使用lz4压缩算法，对于下盘量较小场景性能更优，但占用更多存储空间。
- 'zstd'，表示使用zstd压缩算法，对于下盘量较大场景，IO为主要瓶颈，其性能更优，且存储空间约为lz4的2/3。

默认值：'lz4'

 版权所有 © 华为云计算技术有限公司

index_selectivity_cost

参数说明：控制列存表索引扫描时cbtree的cost计算（选择率>0.001），该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，-1，0～1000，负数仅支持设置-1，且不允许设置其他负数。

- 取值为0时，不受索引选择率阈值0.001的影响。
- 取值为-1时，受disable_cost影响。
- 为其他值时，为cbtree cost计算的系数。

默认值：-1

index_cost_limit

参数说明：控制列存表索引扫描时cbtree的cost计算不生效阈值，该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~2147483647。

- 取值为0时，表示参数不生效。
- 取值为其他值时，表行数少于该值时不受索引选择率阈值0.001的影响。

默认值：0

volatile_shipping_version

参数说明：控制volatile函数下推执行的范围。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为3时，在2的基础上，扩展支持InlineCTE在只被引用一次时支持下推，扩展支持复制表UPSERT中使用包含volatile函数时的下推。
- 取值为2时，在1的基础上，扩展支持在复制CTE结果的目标列中包含volatile函数时的下推。
- 取值为1时，扩展支持nextval、uuid_generate_v1、sys_guid、uuid函数出现在语句目标列时的完全下推。
- 取值为0时，支持random类函数的完全下推，nextval、uuid_generate_v1函数仅在INSERT含有简单查询语句的部分下推。

默认值：3

agg_redistribute_enhancement

参数说明：当进行Agg操作时，如果包含多个group by列且均不为分布列，进行重分布时会选择某一group by列进行重分布。本参数控制选择重分布列的策略。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示会选择估算distinct值最多的一个可重分布列作为重分布列。
- off表示会选择第一个可重分布列为重分布列。

默认值：off

enable_valuepartition_pruning

参数说明：是否对DFS（Distributed File System，分布式文件系统）分区表进行静态/动态优化。

参数类型：USERSET

取值范围：布尔型

- on表示对DFS分区表进行静态/动态优化。
- off表示不对DFS分区表进行静态/动态优化。

默认值：on

expected_computing_nodegroup

参数说明：标识选定的计算Node Group模式或目标计算Node Group。Node Group目前为内部用机制，用户无需设置。

共4种计算Node Group模式，用于关联操作和聚集操作时选定计算Node Group。在每一种模式中，优化器有针对性地选定几个候选计算Node Group，然后根据代价，从中为当前算子挑选更合适的计算Node Group。

参数类型：USERSET

取值范围：字符串

- optimal：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前用户具有COMPUTE权限的所有Node Group包含的所有DN构成的Node Group。
- query：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前查询涉及的所有基表所在Node Group包含的所有DN构成的Node Group。
- bind：当前session用户是逻辑集群用户时，候选计算Node Group为当前用户关联的逻辑集群的Node Group；当session用户不是逻辑集群用户时，候选计算Node Group选取规则和参数设置为query时的规则一致。
- Node Group名：
  - **enable_nodegroup_debug**为off时：候选计算Node Group列表包含算子操作对象所在的Node Group和该指定的Node Group。
  - **enable_nodegroup_debug**为on时：候选计算Node Group为指定的Node Group。

默认值：bind

enable_nodegroup_debug

参数说明：控制优化器在多Node Group环境下，是否使用强制弹性计算。Node Group目前为内部用机制，用户无需设置。

该参数只在**expected_computing_nodegroup**被设置为具体Node Group时生效。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示强制将计算弹性到expected_computing_nodegroup所指定的Node Group进行计算。
- off表示不强制使用某个Node Group进行计算。

默认值：off

stream_multiple

参数说明：设置优化器计算Stream算子的开销时的加权。

在原代价模型的基础上，最终Stream代价将被乘以此加权参数。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：1

须知

此参数仅对Redistribute和Broadcast类型的Stream有效。

qrw_inlist2join_optmode

参数说明：控制是否使用inlist-to-join查询重写。

参数类型：USERSET

取值范围：字符串

- disable：关闭inlist2join查询重写。
- cost_base：基于代价的inlist2join查询重写。
- rule_base：基于规则的inlist2join查询重写，即强制使用inlist2join查询重写。
- 任意正整数：inlist2join查询重写阈值，即list内元素个数大于该阈值，进行inlist2join查询重写。

默认值：disable

enable_inlist_hashing

参数说明：控制是否使用inlist哈希优化。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启inlist哈希优化。
- off表示关闭inlist哈希优化。

默认值：on

 版权所有 © 华为云计算技术有限公司

setop_optmode

参数说明：控制不带ALL选项的集合操作(UNION/EXCEPT/INTERSECT)语句的各个查询分支语句是否执行去重操作。

参数类型：USERSET

取值范围：枚举型

- disable：查询分支不执行去重操作。
- force：强制查询分支执行去重操作。
- cost：优化器在查询分支去重和不去重这两种执行方式中，选择代价比较小的执行方式。

默认值：cost

须知

- 新装9.1.0.200及以上版本集群中该参数的默认值为cost，升级场景该参数的默认值为保持前向兼容维持原值。
- 此参数配置仅在SQL语句的执行计划满足以下条件时生效：
  - SQL语句中的UNION/EXCEPT/INTERSECT操作不带ALL选项。
  - 执行集合操作的各个查询分支在进行集合操作前执行过数据重分布动作。

skew_option

参数说明：控制是否使用优化策略。

参数类型：USERSET

取值范围：字符串

- off：关闭策略。
- normal：采用激进策略。对于不确定是否出现倾斜的场景，认为存在倾斜，并进行相应优化。
- lazy：采用保守策略。对于不确定是否出现倾斜场景，认为不存在倾斜，不进行优化。

默认值：normal

enable_expr_skew_optimization

参数说明：控制是否在倾斜优化策略中使用表达式统计信息。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示在倾斜优化策略中使用表达式统计信息来判断是否存在倾斜。
- off表示在倾斜优化策略中不使用表达式统计信息来判断是否存在倾斜。

默认值：on

 版权所有 © 华为云计算技术有限公司

prefer_hashjoin_path

参数说明：控制是否优先生成hashjoin路径，以便通过将代价较大的其它路径预剪枝掉的方法来提升整体计划生成时间。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用hashjoin路径提前生成的优化。
- off表示关闭hashjoin路径提前生成的优化。

默认值：on

enable_hashfilter_test

参数说明：该参数用于控制是否为基表扫描增加分布列的hashfilter，以便确认结果是否符合预期。同时，在数据插入时，控制是否进行DN准确性校验（即校验当前数据是否应该插入当前DN）。

参数类型：USERSET

取值范围：布尔型

- on表示为基表扫描添加分布列的hashfilter，并在数据插入时进行DN准确性校验。
- off表示不为基表扫描添加分布列的hashfilter，并在数据插入时不进行DN准确性校验。

默认值：on

须知

- 此参数仅对hash分布的表有效。
- 当该参数设置为on后，因在数据插入时会进行DN准确性校验，会影响数据插入性能。

enable_cu_align_8k

参数说明：设置V3表CU是否8K字节对齐。该参数仅9.1.0及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示V3表CU按照8192字节对齐。
- off表示V3表CU按照512字节对齐。

默认值：off

enable_cu_batch_insert

参数说明：设置V2表是否开启多列CU批量写功能。该参数仅9.1.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示V2表开启多列CU批量写功能。
- off表示V2表未开启多列CU批量写功能。

默认值：off

enable_topk_optimization

参数说明：控制是否开启TopK排序优化。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启TopK排序优化。
- off表示关闭TopK排序优化。

默认值：on

19.9.2 优化器开销常量

介绍优化器开销常量。这里描述的开销可以按照任意标准度量。只关心其相对值，因此以相同的系数缩放它们将不会对优化器的选择产生任何影响。缺省时，它们以抓取顺序页的开销为基本单位。也就是说将seq_page_cost设为1.0，同时其他开销参数以它为基准设置。也可以使用其他基准，比如以毫秒计的实际执行时间。

seq_page_cost

参数说明：设置优化器计算一次顺序磁盘页面抓取的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：1

random_page_cost

参数说明：设置优化器计算一次非顺序抓取磁盘页面的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：4

说明

- 虽然服务器允许将random_page_cost设置的比seq_page_cost小，但是物理上实际不受影响。如果所有数据库都位于随机访问内存中时，两者设置为相等很合理。因为在此种情况下，非顺序抓取页并没有副作用。同样，在缓冲率很高的数据库上，应该相对于CPU参数同时降低这两个值，因为获取内存中的页要比通常情况下开销小很多。
- 对于特别表空间中的表和索引，可以通过设置同名的表空间的参数来覆盖这个值。
- 相对于seq_page_cost，减少这个值将导致系统更倾向于使用索引扫描，而增加这个值使得索引扫描开销比较高。可以通过同时增加或减少这两个值来调整磁盘I/O相对于CPU的开销。

 版权所有 © 华为云计算技术有限公司

cpu_tuple_cost

参数说明：设置优化器计算在一次查询中处理每一行数据的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.01

cpu_index_tuple_cost

参数说明：设置优化器计算在一次索引扫描中处理每条索引的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.005

cpu_operator_cost

参数说明：设置优化器计算一次查询中执行一个操作符或函数的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.0025

effective_cache_size

参数说明：设置优化器在一次单一的查询中可用的磁盘缓冲区的有效大小。

设置这个参数，还要考虑DWS的共享缓冲区以及内核的磁盘缓冲区。另外，还要考虑预计的在不同表之间的并发查询数目，因为它们将共享可用的空间。

这个参数对DWS分配的共享内存大小没有影响，它也不会使用内核磁盘缓冲，它只用于估算。数值是用磁盘页来计算的，通常每个页面是8192字节。

参数类型：USERSET

取值范围：整型，1～INT_MAX，单位为8KB。

比默认值高的数值可能会导致使用索引扫描，更低的数值可能会导致选择顺序扫描。

默认值：128MB

allocate_mem_cost

参数说明：设置优化器计算Hash Join创建Hash表开辟内存空间所需的开销，供Hash join估算不准时调优使用。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0

 版权所有 © 华为云计算技术有限公司

smp_thread_cost

参数说明：设置优化器计算一个算子并行线程所需的开销，用于在生成query_dop不合适时调优使用。（该参数仅8.2.0及以上集群版本支持）

参数类型：USERSET

取值范围：浮点型，1～10000。

默认值：1000

19.9.3 基因查询优化器

介绍基因查询优化器相关的参数。基因查询优化器（GEQO）是一种启发式的查询规划算法。这个算法减少了对复杂查询规划的时间，而且生成规划的开销有时也小于正常的详尽的查询算法。

geqo

参数说明：控制基因查询优化的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

须知

通常情况下在执行过程中不要关闭，geqo_threshold变量提供了更精细的控制GEQO的方法。

geqo_threshold

参数说明：如果执行语句的数量超过设计的FROM的项数，则会使用基因查询优化来执行查询。

参数类型：USERSET

取值范围：整型，2～INT_MAX。

默认值：12

须知

- 对于简单的查询，通常用详尽搜索方法，当涉及多个表的查询的时候，用GEQO可以更好的管理查询。
- 一个FULL OUTER JOIN构造仅作为一个FROM项。

 版权所有 © 华为云计算技术有限公司

geqo_effort

参数说明：控制GEQO在规划时间和规划质量之间的平衡。

参数类型：USERSET

取值范围：整型，1～10

默认值：5

须知

- 比默认值大的数值增加了查询规划的时间，但是也增加了选中有效查询的几率。
- geqo_effort实际上并没有直接作用，只是用于计算其他影响GEQO的变量的默认值。如有需求，可以手动设置其他相关参数。

geqo_pool_size

参数说明：控制GEQO使用池的大小，也就是基因全体中的个体数量。

参数类型：USERSET

取值范围：整型，0～INT_MAX

须知

至少是2，且有用的值一般在100到1000之间。设置为0，表示使用系统自适应方式，DWS会基于geqo_effort和表的个数选取合适的值。

默认值：0

geqo_generations

参数说明：控制GEQO使用的算法的迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX

须知

必须至少是1，且有用的值介于100和1000之间。如果设置为0，则基于geqo_pool_size选取合适的值。

默认值：0

geqo_selection_bias

参数说明：控制GEQO的选择性偏好，即一个种群中的选择性压力。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：浮点型，1.5～2.0

默认值：2

说明

数值越大表示选择性压力越高，更优的个体被选中的概率更高。数值越小表示选择性压力越低，允许更多多样性。

geqo_seed

参数说明：控制GEQO使用的随机数生产器的初始化值，用来从顺序连接在一起的查询空间中查找随机路径。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

须知

不同的值会改变搜索的连接路径，从而影响了所找路径的优劣。

默认值：0

19.9.4 其他优化器选项

default_statistics_target

参数说明：为没有用ALTER TABLE SET STATISTICS设置字段目标的表设置缺省统计目标。此参数设置为正数是代表统计信息的样本数量，为负数时，代表使用百分比的形式设置统计目标，负数转换为对应的百分比，即-5代表5%。采样时，会将default_statistics_target * 300作为随机抽样的大小，例如默认值为100时，会随机读取30000个页面再从中随机取30000条数据来完成随机抽样。

参数类型：USERSET

取值范围：浮点型，-100～10000。

须知

- 比默认值大的正数数值增加了ANALYZE所需的时间，但是可能会改善优化器的估计质量。
- 调整此参数可能存在性能劣化的风险，如果某个查询劣化，可以考虑
  1. 恢复默认的统计信息。
  2. 使用plan hint来调整到之前的查询计划。（详细参见**使用Plan Hint进行调优**）
- 当此guc参数设置为负数时，如果计算的采样样本数大于等于总数据量的2%，且用户表的数据量小于1600000时，ANALYZE所需时间相比guc参数为默认值的时间会有所增加。
- autoanalyze不支持临时表采样方式设置采样大小，采样过程使用参数默认值。
- 当强制使用内存方式计算统计信息时，采样大小受maintenance_work_mem参数限制。

版权所有 © 华为云计算技术有限公司

默认值：100

random_function_version

参数说明：控制analyze在进行数据采样时选取的random函数版本。该参数仅8.1.2及以上版本支持。

参数类型：USERSET

取值范围：枚举类型

- 0 表示采用C标准库提供的random函数。
- 1 表示采用优化增强的random函数。

默认值：

- 若当前集群为低版本升级到8.2.0.100及以上集群版本，为保持和前向兼容，默认值为0。
- 若当前集群为新装的8.2.0.100及以上集群版本，默认值为1。

constraint_exclusion

参数说明：控制查询优化器使用表约束查询的优化。

参数类型：USERSET

取值范围：枚举类型

- on表示检查所有表的约束。
- off表示不检查约束。
- partition表示只检查继承的子表和UNION ALL子查询。

须知

当constraint_exclusion为on，优化器用查询条件和表的CHECK约束比较，并且在查询条件和约束冲突的时候忽略对表的扫描。

默认值：partition

说明

目前， constraint_exclusion缺省被打开，通常用来实现表分区。为所有的表打开它时，对于简单的查询强加了额外的规划，并且对简单查询没有什么好处。如果不用分区表，可以关掉它。

cursor_tuple_fraction

参数说明：优化器估计游标获取行数在总行数中的占比。

参数类型：USERSET

取值范围：浮点型，0.0～1.0。

 版权所有 © 华为云计算技术有限公司

须知

比默认值小的值与使用 “fast start” 为游标规划的值相偏离，从而使得前几行恢复的很快而抓取全部的行需要很长的时间。比默认值大的值加大了总的估计的时间。在最大的值1.0处，像正常的查询一样规划游标，只考虑总的估计时间和传送第一行的时间。

默认值：0.1

from_collapse_limit

参数说明：根据生成的FROM列表的项数来判断优化器是否将把子查询合并到上层查询，如果FROM列表项个数小于等于该参数值，优化器会将子查询合并到上层查询。

参数类型：USERSET

取值范围：整型，1～INT_MAX。

默认值：8

⚠ 注意

比默认值小的数值会减少规划时间，但是可能生成较差的执行计划。

join_collapse_limit

参数说明：根据得出的列表项数来判断优化器是否执行把除FULL JOINS之外的JOIN构造重写到FROM列表中。

参数类型：USERSET

取值范围：整型，1～INT_MAX。

默认值：8

须知

- 设置为1会避免任何JOIN重排。这样就使得查询中指定的连接顺序就是实际的连接顺序。查询优化器并不是总能选取最优的连接顺序，高级用户可以选择暂时把这个变量设置为1，然后指定它们需要的连接顺序。
- 比默认值小的数值会减少规划时间，但也降低了执行计划的质量。

join_search_mode

参数说明：标识计划路径搜索的方式。

参数类型：USERSET

取值范围：枚举类型

- exhaustive：使用传统动态规划和遗传基因方式进行计划路径搜索。

 版权所有 © 华为云计算技术有限公司

- heuristic：使用启发式方法进行计划路径搜索。该方式会加快计划生成的性能，但存在忽略掉最优计划的可能性。该设置仅对于指定Drive Hint或连接表个数大于from_collapse_limit的场景生效。

默认值：heuristic

enable_from_collapse_hint

参数说明：标识是否优先以hint生效的方式重写FROM列表，其次再根据from_collapse_limit、join_collapse_limit参数进行重写。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示优先以hint生效的方式重写FROM列表。
- off表示无差别重写FROM列表。

须知

- 该参数启用时，优化器会优先以hint生效的方式重写FROM列表，但hint是否生效在计划生成后才能知道，因此有可能出现该hint不生效的情况。
- 该参数关闭时，回退到8.2.0版本之前的计划，即生成计划时不区分表是否有hint。

默认值：on

plan_mode_seed

参数说明：该参数为调测参数，目前仅支持OPTIMIZE_PLAN和RANDOM_PLAN两种。其中：OPTIMIZE_PLAN表示通过动态规划算法进行代价估算的最优plan，参数值设置为0；RANDOM_PLAN表示随机生成的plan；如果设置为-1，表示用户不指定随机数的种子标识符seed值，由优化器随机生成[1, 2147483647]范围整型值的随机数，并根据随机数生成随机的执行计划；如果用户指定guc参数值为[1, 2147483647]范围的整型值，表示指定的生成随机数的种子标识符seed，优化器需要根据seed值生成随机的执行计划。

参数类型：USERSET

取值范围：整型，-1~ 2147483647

默认值：0

须知

- 当该参数设置为随机执行计划模式时，优化器会生成不同的随机执行计划，该执行计划可能不是最优计划。因此在随机计划模式下，会对查询性能产生影响，所以建议在升级、扩容、缩容等正常业务操作或运维过程中将该参数保持为默认值0。
- 当该参数不为0时，查询指定的plan hint不会生效。

 版权所有 © 华为云计算技术有限公司

enable_hdfs_predicate_pushdown

参数说明：标识是否启用谓词下推至原生数据层的功能。

参数类型：SUSET

取值范围：布尔型

- on表示启用谓词下推至原生数据层的功能。
- off表示不启用谓词下推至原生数据层的功能。

默认值：on

windowagg_pushdown_enhancement

参数说明：标识是否在聚集场景下启用窗口函数谓词下推增强功能。（该参数仅8.2.0及以上集群版本支持）

参数类型：SUSET

取值范围：布尔型

- on表示聚集场景启用窗口函数谓词下推增强功能。
- off表示聚集场景不启用窗口函数谓词下推增强功能。

默认值：on

implied_quality_optmode

参数说明：标识语句中等值列的条件传递优化策略。（该参数仅8.2.0及以上集群版本支持）

参数类型：SUSET

取值范围：枚举类型

- normal表示向前兼容8.1.3及以前版本，即推导出的表达式行为优化。
- negative表示推导出的表达式行为不优化。
- positive在normal的基础上增加了类型转换表达式的优化。

默认值：normal

enable_random_datanode

参数说明：标识是否允许开启复制表DN随机查找功能，复制表在每个DN存放一份完整数据，随机选取可以缓解节点压力。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启复制表DN随机查找功能。
- off表示不允许开启复制表DN随机查找功能。

默认值：on

 版权所有 © 华为云计算技术有限公司

hashagg_table_size

参数说明：用于设置执行HASH AGG操作时HASH表的大小。

参数类型：USERSET

取值范围：整型，0～INT_MAX/2。

默认值：0

enable_codegen

参数说明：标识是否允许开启代码生成优化，目前代码生成使用的是LLVM优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启代码生成优化。
- off表示不允许开启代码生成优化。

默认值：off

注意

- 9.1.0.218及以上集群版本中（新安装和升级场景），该参数默认值为off。9.1.0.218以下低版本的9.1.0.xxx集群中该参数默认值为on。
- 目前LLVM优化仅支持向量化执行引擎特性和SQL on Hadoop特性，其他场景下建议此参数保持关闭。

codegen_strategy

参数说明：标识在表达式codegen化过程中所使用的代码生成优化策略。

参数类型：USERSET

取值范围：枚举类型

- partial表示当所计算表达式中即使包含部分未被codegen化的函数时，仍可借助表达式全codegen框架调用LLVM动态编译优化策略。
- pure表示当所计算表达式整体可被codegen化时，才考虑调用LLVM动态编译优化策略。

须知

在开启代码生成优化会导致查询性能下降的场景下可以设置此参数为pure，其他场景下建议不改变此参数的默认值partial。

默认值：partial

enable_codegen_print

参数说明：标识是否允许在log日志中打印所生成的LLVM IR函数。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示允许在log日志中打印IR函数。
- off表示不允许在log日志中打印IR函数。

默认值：off

codegen_cost_threshold

参数说明：由于LLVM编译生成最终的可执行机器码需要一定时间，因此只有当实际执行的代价大于编译生成机器码所需要的代码和优化后的执行代价之和时，利用代码生成才有收益。codegen_cost_threshold标识代价的阈值，当执行估算代价大于该代价时，使用LLVM优化。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：10000

llvm_compile_expr_limit

参数说明：用于标识编译表达式个数的阈值，当表达式个数高于该阈值时，只将阈值之内的表达式使用LLVM编译，高于阈值之外的表达式不使用LLVM编译，同时生成编译告警提示（告警提示需要在执行explain performance前SET analysis_options="on(LLVM_COMPILE)"）。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

默认值：500

llvm_compile_time_limit

参数说明：LLVM编译时间在执行器运行时间中的占比超过llvm_compile_time_limit所设置的阈值，说明LLVM编译时间占比过高，生成告警提示（告警提示需要在执行explain performance前SET analysis_options="on(LLVM_COMPILE)"）。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0.2

enable_constraint_optimization

参数说明: 标识是否允许HDFS外表使用信息约束（Informational Constraint）优化执行计划。

参数类型：SUSET

取值范围：布尔型

- on表示允许使用信息约束优化执行计划。

 版权所有 © 华为云计算技术有限公司

- off表示不允许使用信息约束优化执行计划。

默认值：on

enable_bloom_filter

参数说明: 标识是否允许使用BloomFilter优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许使用BloomFilter优化。
- off表示不允许使用BloomFilter优化。

默认值：on

须知

适用场景：外表侧同线程包含有HDFS内外表或列存表的HASH JOIN会触发Bloom Filter。

使用限制：

1. JOIN类型仅限于INNER JOIN、SEMI JOIN、RIGHT JOIN、RIGHT SEMI JOIN、RIGHT ANTI JOIN、RIGHT ANTI FULL JOIN。
2. JOIN内表侧关联条件：对于HDFS内外表不能为表达式；对于列存表可以为表达式，但仅限于非JOIN层计算的表达式。
3. JOIN外表侧关联条件必须为简单列关联。
4. JOIN内表侧与外表侧关联条件均为简单列关联时，计划层估算必须可以去除1/3以上的数据（仅针对HDFS内外表）。
5. JOIN不能包含null值关联。
6. 数据类型：
   - HDFS内外表字段类型支持SMALLINT、INTEGER、BIGINT、REAL/FLOAT4、DOUBLE PRECISION/FLOAT8、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、CLOB、TEXT。
   - 列存表字段类型支持SMALLINT、INTEGER、BIGINT、OID、"char"、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、NVARCHAR2(n)、CLOB、TEXT、DATE、TIME、TIMESTAMP、TIMESTAMPTZ，其中字符类型其排序规则必须指定为"C"。

runtime_filter_type

参数说明: 标识使用的runtime filter类型，仅在打开**enable_bloom_filter**时生效。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- All，表示应用所有场景下的runtime filter。
- Topn_filter，表示应用join场景以及带有limit的order by场景下的runtime filter。

 版权所有 © 华为云计算技术有限公司

- Bloom_filter，表示仅应用join场景下的runtime filter，且满足条件后join会生成bloom filter进行过滤。
- Min_max，表示仅应用join场景下的runtime filter，且join场景仅会生成min_max过滤器。
- None，表示不使用runtime filter，此时仅对原版bloom filter生效场景具有过滤效果。

默认值：All

须知

- 适用场景：列存表的HASH JOIN外表侧的计划类型以及列存表带有limit的order by计划类型。
- 使用限制：
  - JOIN场景使用限制同**enable_bloom_filter**参数的限制。
  - 带有limit的order by场景下，order by字段类型仅支持SMALLINT、INTEGER、BIGINT、"char"、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、NVARCHAR2(n)、TEXT、DATE、TIME、TIMESTAMP、TIMESTAMPTZ，其中字符类型其排序规则必须指定为"C"。

runtime_filter_ratio

参数说明: 控制runtime filter在join场景中是否使用bloom filter进行细粒度行级过滤的阈值，仅在**runtime_filter_type**设置值大于等于Bloom_filter时生效。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0.01

须知

- 适用场景：列存表的HASH JOIN，并且满足：内表estimate_join_rows/外表estimate_join_rows <= runtime_filter_ratio。细粒度行级过滤仅推荐在join场景下内表外表数据量差距明显时使用，不合理的runtime_filter_ratio设置可能会导致join场景性能劣化。
- 使用限制：细粒度行级过滤仅支持join字段类型为SMALLINT、INTEGER、BIGINT、FLOAT类型时使用。

runtime_filter_cost_options

参数说明: 控制在生成计划时，是否根据cost选择生成应用runtime filter计划。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

 版权所有 © 华为云计算技术有限公司

- apply_partial：build端只要包含probe端某个runtime filter需要的表，就可以生成runtime filter path。
- apply_all：限制build端必须包含probe端可应用的runtime filter需要的所有表，才可以生成runtime filter path。

默认值：''，即生成计划时不考虑根据cost选择应用runtime filter的计划。

须知

若apply_partial和apply_all同时设置，则以apply_all设置生效。

enable_extrapolation_stats

参数说明: 标识是否允许基于历史统计信息使用推理估算的逻辑。使用该逻辑对于未及时收集统计信息的表可以增大估算准确的可能性，但也存在错误推理导致估算过大的可能性。

参数类型：USERSET

取值范围：布尔型

- on表示允许基于历史统计信息使用推理估算的逻辑。
- off表示不允许基于历史统计信息使用推理估算的逻辑。

默认值:

- 若当前集群为低版本升级到8.2.0.100及以上集群版本，为保持和前向兼容，默认值为off。
- 若当前集群为新装的8.2.0.100及以上集群版本，默认值为on。

autoanalyze

参数说明: 标识是否允许在生成计划的时候，对于“统计信息完全缺失”或“修改量达到analyze阈值”的表进行统计信息自动收集，当前不支持对外表触发autoanalyze，不支持对带有ON COMMIT [DELETE ROWS|DROP]选项的临时表触发autoanalyze，如需收集，需用户手动执行analyze操作。如果在auto analyze某个表的过程中数据库发生异常，当数据库正常运行之后再执行语句有可能仍提示需要收集此表的统计信息。此时需要用户对该表手动执行一次analyze操作，以同步统计信息数据。

须知

表的修改量达到analyze阈值是指：表的修改量超过autovacuum_analyze_threshold + autovacuum_analyze_scale_factor * reltuples，其中reltuples是pg_class中记录的表的估算行数。

参数类型：SUSET

取值范围：布尔型

- on表示允许自动进行统计信息收集。
- off表示不允许自动进行统计信息收集。

 版权所有 © 华为云计算技术有限公司

默认值：on

enable_analyze_partition

参数说明: 控制是否支持对表的某个分区收集统计信息。开启该参数后，可以通过ANALYZE table_name PARTITION（partition_name）收集某个分区的统计信息，查询表的这个分区上的数据时，优化器会选择使用分区统计信息。

参数类型：USERSET

取值范围：布尔型

- on表示支持对表的某个分区收集统计信息。
- off表示不支持对表的某个分区收集统计信息。

默认值：off

analyze_use_dn_correlation

参数说明：设置ANALYZE时CN是否使用DN的相关系数统计信息。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示CN使用DN的相关系数统计信息。
- off表示CN不使用DN的相关系数统计信息。

默认值：on

analyze_predicate_column_threshold

参数说明：控制是否开启谓词列ANALYZE及限定支持的最小列数。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0～10000

- 0表示关闭谓词列ANALYZE，不会收集谓词列以及对谓词列进行ANALYZE。
- 大于0表示开启谓词列收集功能，且仅对列数大于等于此值的表进行谓词列ANALYZE。

默认值：10

enable_runtime_analyze_concurrent

参数说明：控制是否支持对一个表进行并发RUNTIME ANALYZE。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示支持并发。

 版权所有 © 华为云计算技术有限公司

- off表示不支持并发。

默认值：on

analyze_max_columns_count

参数说明：控制ANALYZE能支持最大的列数。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1～10000

- -1，表示不限制ANALYZE的列数。
- 大于-1，将只收集此值之前的列，超过的列不收集。

默认值：-1

query_dop

参数说明: 用户自定义的查询并行度。

参数类型：USERSET

取值范围：整型，-64-64

[1,64]：打开固定SMP功能，系统会使用固定并行度。

0：打开SMP自适应功能，系统会根据资源情况和计划特征动态为每个查询选取[1,8]之间（x86平台），[1,64]之间（鲲鹏平台）的最优的并行度。

[-64,-1]：打开SMP自适应功能，并限制自适应选取的最大并行度。

说明

- 对于短查询为主的TP类业务中，如果不能通过CN轻量化或下发语句进行业务的调优，则生成SMP计划的时间较长，建议设置query_dop=1。对于AP类复杂语句的场景，建议设置query_dop=0。
- 在开启并行查询后，请保证系统CPU、内存、网络、I/O等资源充足，以达到良好效果。
- 为了避免用户设置不合理的过大值造成性能劣化，系统会计算出该DN可用最大CPU核数，并以此来作为query_dop的上限。如果用户设置query_dop超过4并且同时超过该上限，那么系统会重置query_dop为该上限值。

默认值：1（64GB及以上内存规格默认值为0）

query_dop_ratio

参数说明: 用于当query_dop取值为0时，设置在系统中给定的最优dop基础上，调整dop的倍数。即最终dop=系统设置dop * query_dop_ratio（最小值为1，最大值为64）。当设置为1时，表示不调整。

参数类型：USERSET

取值范围：浮点型，0-64

默认值：1

 版权所有 © 华为云计算技术有限公司

debug_group_dop

参数说明：当query_dop取值为0时，针对生成的执行计划划分的以Stream算子为顶点的group，均分配统一的dop并行度。此参数用于人为指定特定group的dop进行性能调优，格式为G1,D1,G2,D2,...，其中：G1,G2为group的ID，可以从日志中获得，D1,D2为指定的dop值，可以为任意正整数。

参数类型：USERSET

取值范围：字符型

默认值：空

须知

该参数仅供内部调优使用，不允许用户进行设置，建议保持默认值。

enable_analyze_check

参数说明: 标识是否允许在生成计划的时候，对于在pg_class中显示reltuples和relpages均为0的表，检查该表是否曾进行过统计信息收集。**该参数8.1.3及以上集群版本中已废弃，为兼容历史版本功能保留该参数，设置不会生效**。

参数类型：SUSET

取值范围：布尔型

- on表示允许检查。
- off表示不允许检查。

默认值：on

enable_sonic_hashagg

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Agg算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Agg算子。
- off表示不使用面向列的hash表设计的Hash Agg算子。

说明

- 在开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，查询对应的Hash Agg算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景(**enable_codegen**打开后获得较大性能提升)，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Aggregation”，而未达到该约束条件时，算子名称将显示为“Hash Aggregation”。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_sonic_hashjoin

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Join算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Join算子。
- off表示不使用面向列的hash表设计的Hash Join算子。

说明

- 当前开关仅适用于Inner Join的场景。
- 在开启enable_sonic_hashjoin，查询对应的Hash Inner算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashjoin，且查询达到约束条件使用基于面向列的hash表设计的Hash Join算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Join”，而未达到该约束条件时，算子名称将显示为“Hash Join”。

默认值：on

enable_sonic_optspill

参数说明：标识是否优化sonic场景下HashJoin或者HashAgg的下盘文件个数。仅在enable_sonic_hashjoin或者 enable_sonic_hashagg开启情况下生效。

参数类型：USERSET

取值范围：布尔型

- on表示开启下盘文件数优化。
- off表示关闭下盘文件数优化。

说明

满足sonic条件下的HashJoin或者HashAgg算子，在关闭此参数（设置为off）时每列会产生1个下盘文件。开启此参数（设置为on）时如果不同列数据类型相似，只会有1个下盘文件（最多5个文件）。

默认值：on

expand_hashtable_ratio

参数说明：控制Hash Agg和Hash Join算子执行过程中hash表的大小扩大比例。

参数类型：USERSET

取值范围：浮点型，0, 0.5~10

说明

- 默认值设置为0时表示hash表大小会根据当前内存进行自适应扩展。
- 默认值设置为0.5~10之间时，显式的是指定hash表扩大的倍数，通常hash表越大性能越好，但会占用更多内存空间，在内存不足场景可能造成数据提前下盘，带来性能劣化。

默认值：0

 版权所有 © 华为云计算技术有限公司

plan_cache_mode

参数说明：标识在prepare语句中，选择生成执行计划的策略。

参数类型：USERSET

取值范围：枚举类型

- auto表示按照默认的方式选择custom plan或者generic plan。
- force_generic_plan表示强制走generic plan。
- force_custom_plan表示强制走custom plan。

说明

- 此参数只对prepare语句生效，一般用在prepare语句中参数化字段存在比较严重的数据倾斜的场景下。
- custom plan是指对于prepare语句，在执行execute的时候，把execute语句中的参数嵌套到语句之后生成的计划。custom plan会根据execute语句中具体的参数生成计划，这种方案的优点是每次都按照具体的参数生成优选计划，执行性能比较好；缺点是每次执行前都需要重新生成计划，存在大量的重复的优化器开销。
- generic plan是指对于prepare语句生成计划，该计划策略会在执行execute语句的时候把参数bind到plan中，然后执行计划。这种方案的优点是每次执行可以省去重复的优化器开销；缺点是当bind参数字段上数据存在倾斜时该计划可能不是最优的，部分bind参数场景下执行性能较差。

默认值：auto

wlm_query_accelerate

参数说明：标识在短查询加速打开时，查询是否需要加速。

参数类型：USERSET

取值范围：整型，-1~1

- -1：短查询由快车道管控，长查询由慢车道管控。
- 0：查询不加速，短查询和长查询均由慢车道管控。
- 1：查询加速，短查询和长查询均由快车道管控。

默认值：-1

show_unshippable_warning

参数说明：标识是否将语句不下推的告警打印到客户端。

参数类型：USERSET

取值范围：布尔型

- on：将语句不下推的原因以WARNING打印到日志和客户端
- off：仅将语句不下推的原因以LOG打印到日志

默认值：off

hashjoin_spill_strategy

参数说明：选择hashjoin下盘策略。（ 该参数8.1.2及以上版本支持 ）

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0～6

- 0：当内表较大且无法在数据库可用内存放下所有数据时，会将数据划分成不同的子分区，直到多次划分后无法分开且仍无法在内存放下所有数据时，尝试外表是否可以放到可用内存中建立哈希表。若外表可以放到可用内存中建立哈希表，则执行HashJoin。反之，则执行NestLoop。
- 1：当内表较大且无法在数据库可用内存放下所有数据时，会将数据划分成不同的子分区，直到多次划分后无法分开且仍无法在内存放下所有数据时，尝试外表是否可以放到可用内存中建立哈希表。如果内外表均很大，强制执行HashJoin。
- 2：当内表较大，并且多次下盘无法分开时，强制执行HashJoin。
- 3：当内表较大，并且多次下盘无法分开时，尝试外表是否可以放到数据库可用内存建立哈希表。如果内外表均很大，则报错。
- 4：当内表较大，并且多次下盘无法分开时，则报错。
- 5：当内表较大且无法在数据库可用内存放下所有数据时，如果外表数据可以放到内存中，则使用外表建立哈希表执行HashJoin。如果外表数据无法存放到内存中，则将数据划分成不同的子分区，直到内外表多次划分均无法分开时，执行NestLoop。
- 6：当内表较大且无法在数据库可用内存放下所有数据时，如果外表数据可以放到内存中，则使用外表建立哈希表执行HashJoin。如果外表数据无法存放到内存中，则将数据划分成不同的子分区，直到内外表多次划分均无法分开时，强制执行HashJoin。

说明

- 此参数只对向量化HashJoin生效。
- 对于数据distinct值很小且数据量很大的场景，可能出现无法下盘导致使用内存过大产生内存不受控的问题。 取值0时通过尝试内外表交换或者Nestloop可以避免出现此类内存不受控问题。 执行Nestloop可能造成某些场景性能劣化。遇到此种场景，该参数可取值1、2、6强制执行HashJoin。
- 取值0对向量化Full Join不生效，行为与取值1相同。只尝试外表是否可建立哈希表，不执行NestLoop。
- 取值5和6相对于取值0和1的优势是如果内表数据量大到无法直接放于可用内存中，但是外表可以，则直接使用外表进行Hashjoin，减少后续多次下盘划分数据的时间消耗。当外表数据distinct较少时，使用外表建立哈希表可能导致性能劣化。此时可以将参数取值调整到0或者1。

默认值：0

max_streams_per_query

参数说明：控制查询计划中Stream节点的数目。（该参数仅8.1.1及以上集群版本支持）

参数类型：SUSET

取值范围：整型，-1～10000

- -1，表示查询计划中Stream节点数目无限制。
- 0~10000，表示查询计划中Stream节点数目超过设定值后报错，查询计划不会被执行。

 版权所有 © 华为云计算技术有限公司

说明

- 此参数只控制DN上的Stream节点，不考虑CN上的Gather节点。
- 此参数不影响Explain查询计划，但是对Explain analyze和Explain performance有影响。

默认值：-1

enable_agg_limit_opt

参数说明：标识是否对select distinct col from table limit N 场景优化，其中N小于16384时生效, table为列存表。该参数仅8.2.0.101及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启该优化。开启后可保证查询结果来自不同DN，且各个DN无需全部建立全量hash表，可显著提升查询性能。
- off表示关闭该优化。

默认值：on

stream_ctescan_pred_threshold

参数说明：当enable_stream_ctescan开启、CTE中仅包含单表过滤时，控制CTE中至少包含的过滤条件个数。大于等于该值时使用share scan方式执行；小于该值时将使用inline方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：SUSET

取值范围：整型，0～INT_MAX

默认值：2

stream_ctescan_max_estimate_mem

参数说明：当enable_stream_ctescan开启时，控制CTE的最大估算内存值。该参数需与stream_ctescan_refcount_threshold共同使用。当CTE的估算内存大于stream_ctescan_max_estimate_mem且引用次数小于stream_ctescan_refcount_threshold时，将使用inline方式执行；反之，则使用sharescan方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：SUSET

取值范围：整型，32 * 1024(32MB)~INT_MAX，单位KB

默认值：256MB

stream_ctescan_refcount_threshold

参数说明：当enable_stream_ctescan开启时，控制CTE的最大引用次数。该参数需与stream_ctescan_max_estimate_mem共同使用。当CTE的估算内存大于stream_ctescan_max_estimate_mem且引用次数小于stream_ctescan_refcount_threshold时，将使用inline方式执行；反之，则使用share scan方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：SUSET

 版权所有 © 华为云计算技术有限公司

取值范围：整型，0~INT_MAX

默认值：4

说明

该参数仅在取值大于0时生效，取值为0时仅依赖stream_ctescan_max_estimate_mem控制inline行为。

inlist_rough_check_threshold

参数说明：当enable_csqual_pushdown开启时、过滤条件为IN进行rough check条件下推时，控制IN条件中值的个数的最大值。IN过滤条件中值的个数超过该参数时，将获取IN条件中值的最大/最小值进行条件下推。该参数仅8.2.0.101及以上集群版本支持。

参数类型：SUSET

取值范围：整型，0～10000

默认值：100

说明

如果IN条件在表的单列分布列上进行，可以在DN进行值的过滤，此时IN条件中值的个数的最大值为inlist_rough_check_threshold的DN倍。

enable_array_optimization

参数说明：控制是否将IN/ANY/ALL条件生成的Array类型拆分为普通表达式执行。此优化将支持向量化执行、rough check剪枝、分区剪枝等多重优化。该参数仅8.2.1及以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on：进行Array类型表达式的拆分优化。
- off：不进行Array类型表达式的拆分优化。

默认值：on

max_skew_num

参数说明：控制优化器允许进行重分布优化的倾斜值个数。该参数仅8.2.1及以上集群版本支持。

参数类型：SUSET

取值范围：整型，0~INT_MAX

默认值：10

enable_dict_plan

参数说明：控制优化器字典编码的总开关，即是否字典编码来加快group by、filter等查询速度。该参数仅8.3.0及以上集群支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示打开优化器字典总开关。
- off表示关闭优化器字典总开关。

默认值：off

dict_plan_distinct_limit

参数说明：用于设置表的某一列的distinct值，小于等于此阈值才会构建字典编码。该参数仅8.3.0及以上集群支持。

参数类型：USERSET

取值范围：0~INT_MAX

默认值：10000

说明

构建字典编码需同时满足dict_plan_distinct_limit和dict_plan_duplicate_ratio参数阈值。

dict_plan_duplicate_ratio

参数说明：用于设置表的某一列的重复率大小，大于等于此阈值才会构建字典编码。字典的典型场景是该列distinct很少，重复率很大。该参数仅8.3.0及以上集群支持。

参数类型：USERSET

取值范围：0.0~100，单位：百分比

默认值：90

说明

构建字典编码需同时满足dict_plan_distinct_limit和dict_plan_duplicate_ratio参数阈值。

enable_cu_predicate_pushdown

参数说明：

1. 功能概述：控制简单过滤条件是否下推到CU来过滤。启用后，能够普遍提升查询性能，尤其是在涉及bitmap_columns列和pck排序列时性能会显著提升。适用于特定的WHERE条件、IS NULL条件、IN条件等场景。该参数仅8.3.0及以上集群支持。
2. 支持的列类型：
   - 整数类型：INT2、INT4、INT8
   - 日期和时间类型：DATE、TIMESTAMP、TIMESTAMPTZ
   - 字符串类型：VARCHAR、TEXT
   - 数值类型：NUMERIC（长度在19以内）
3. 支持查询条件：该功能支持多种 WHERE 表达式，主要包括：
   - **IN 表达式**：用于匹配多个值

 版权所有 © 华为云计算技术有限公司

- **IS NULL / IS NOT NULL 条件**：检查列值是否为空或非空
- **比较表达式**：如大于（>）、小于（<）、等于（=）、不等于（<>）等条件，用于范围查询和精确匹配。

参数类型：USERSET

取值范围：布尔型

- on表示简单过滤条件下推到CU来过滤。
- off表示简单过滤条件不下推到CU来过滤。

默认值：on

说明

字典列中的简单过滤条件指的是等值（“=”）运算、IN表达式及is (not) null。因为此过滤条件下推到存储层在CU填充VectorBatch时提前过滤，称此过滤为CU Predicate Filter。

info_constraint_options

参数说明：用于设置可创建的信息约束。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none，表示不支持任何信息约束的创建。
- foreign_key，表示支持创建外键约束。

默认值：none

19.10 错误报告和日志

19.10.1 记录日志的位置

log_statement_length_limit

参数说明：控制单独打印的SQL语句的长度。超过设置长度的SQL语句，单独打印到statement-*时间戳*.log中。该参数仅9.1.0.200及以上版本支持。

参数类型：SUSET

取值范围：整型，0~INT_MAX。

默认值：1024

19.10.2 记录日志的时间

client_min_messages

参数说明：控制发送到客户端的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，发送给客户端的消息就越少。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

须知

当client_min_messages和**log_min_messages**取相同值时，其值所代表的级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error。参数的详细信息请参见**表19-1**。

默认值：notice

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

参数类型：SUSET

须知

当**client_min_messages**和log_min_messages取相同值log时所代表的消息级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表19-1**。

默认值：warning

log_min_error_statement

参数说明：控制在服务器日志中记录错误的SQL语句。

参数类型：SUSET

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表19-1**。

说明

- 设置为error ，表示导致错误、日志消息、致命错误、panic的语句都将被记录。
- 设置为panic，表示关闭此特性。

默认值：error

log_min_duration_statement

参数说明：当某条语句的持续时间大于或者等于特定的毫秒数时，log_min_duration_statement参数用于控制记录每条完成语句的持续时间。

设置log_min_duration_statement可以很方便地跟踪需要优化的查询语句。对于使用扩展查询协议的客户端，语法分析、绑定、执行每一步所花时间被独立记录。

参数类型：SUSET

 版权所有 © 华为云计算技术有限公司

须知

当此选项与log_statement同时使用时，已经被log_statement记录的语句文本不会被重复记录。在没有使用syslog情况下，推荐使用log_line_prefix记录PID或会话ID，方便将当前语句消息连接到最后的持续时间消息。

取值范围：整型，-1 ~ INT_MAX，单位为毫秒。

- 设置为250，所有运行时间不短于250ms的SQL语句都会被记录。
- 设置为0，输出所有语句的持续时间。
- 设置为-1，关闭此功能。

默认值：30min

backtrace_min_messages

参数说明：控制当产生该设置参数级别相等或更高级别的信息时，会打印函数的堆栈信息到服务器日志文件中。

参数类型：SUSET

须知

该参数作为客户现场问题定位手段使用，且由于频繁的打印函数栈会对系统的开销及稳定性有一定的影响，因此如果需要进行问题定位时，建议避免将backtrace_min_messages的值设置为fatal及panic以外的级别。

取值范围：枚举类型

有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见表19-1。

默认值：panic

表19-1解释DWS中使用的消息安全级别。当日志输出到syslog或者eventlog时，DWS进行如表中的转换。

表 19-1 信息严重程度分类

信息严重程度类型	详细说明	系统日志	事件日志
debug[1-5]	报告详细调试信息。	DEBUG	INFORMATION
log	报告对数据库管理员有用的信息， 比如检查点操作统计信息。	INFO	INFORMATION
info	报告用户可能需求的信息， 比如在VACUUM VERBOSE过程中的信息。	INFO	INFORMATION

 版权所有 © 华为云计算技术有限公司

信息严重程度类型	详细说明	系统日志	事件日志
notice	报告可能对用户有帮助的信息， 比如， 长标识符的截断， 作为主键一部分创建的索引等。	NOTICE	INFORMATION
warning	报告警告信息， 比如在事务块范围之外的COMMIT。	NOTICE	WARNING
error	报告导致当前命令退出的错误。	WARNING	ERROR
fatal	报告导致当前会话终止的原因。	ERR	ERROR
panic	报告导致整个数据库被关闭的原因。	CRIT	ERROR

plog_merge_age

参数说明：该参数用于控制性能日志数据输出的周期。

参数类型：SUSET

须知

该参数以毫秒为单位的，建议在使用过程中设置值为1000的整数倍，即设置值以秒为最小单位。该参数所控制的性能日志文件以prf为扩展名，文件放置在$GAUSSLOG/gs_profile/<node_name> 目录下面，其中node_name是由postgres.conf文件中的pgxc_node_name的值，不建议外部使用该参数。

取值范围：0~INT_MAX，单位为毫秒（ ms ）。

- 当设置为0时，当前会话不再输出性能日志数据。
- 当设置为非0时，当前会话按照指定的时间周期进行输出性能日志数据。该参数设置的越小，输出的日志数据越多，对性能的负面影响越大。

默认值：3s

profile_logging_module

参数说明：用于设置记录性能日志的类型，使用该参数时需确保plog_merge_age参数值非0。该参数属于会话级参数，不建议通过gs_guc工具来设置。仅8.1.3及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

默认值：默认打开OBS、HADOOP、REMOTE_DATANODE，关闭MD。可由SHOW profile_logging_module查看。

 版权所有 © 华为云计算技术有限公司

设置方法：首先，可以通过SHOW profile_logging_module来查看哪些模块是支持可控制的。例如，查询输出结果为：

```
show profile_logging_module;
profile_logging_module
-----------------------------------------------
ALL,on(OBS,HADOOP,REMOTE_DATANODE),off(MD)(1 row)
```

打开MD性能日志，并查看设置结果。其中ALL标识是相当于一个快捷操作，即对所有模块的日志可输出进行开启或关闭。

```
set profile_logging_module='on(md)';
SET

show profile_logging_module;
profile_logging_module
-----------------------------------------------
ALL,on(MD,OBS,HADOOP,REMOTE_DATANODE),off()(1 row)
```

19.10.3 记录日志的内容

debug_pretty_print

参数说明：设置此选项对debug_print_parse、debug_print_rewritten和debug_print_plan产生的日志进行缩进，会生成易读但比设置为off时更长的输出格式。

参数类型：USERSET

取值范围：布尔型

- on表示进行缩进。
- off表示不进行缩进。

默认值：on

log_duration

参数说明：控制记录每个已完成SQL语句的执行时间。对使用扩展查询协议的客户端、会记录语法分析、绑定和执行每一步所花费的时间。

参数类型：SUSET

取值范围：布尔型

- 设置为off ，该选项与**log_min_duration_statement**的不同之处在于log_min_duration_statement强制记录查询文本。
- 设置为on并且log_min_duration_statement大于零，记录所有持续时间，但是仅记录超过阈值的语句。这可用于在高负载情况下搜集统计信息。

默认值：on

log_error_verbosity

参数说明：控制服务器日志中每条记录的消息写入的详细度。

参数类型：SUSET

取值范围：枚举类型

 版权所有 © 华为云计算技术有限公司

- terse输出不包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录。
- verbose输出包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。
- default输出包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录，不包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。

默认值：default

log_lock_waits

参数说明：当一个会话的等待获得一个锁的时间超过**deadlock_timeout**的值时，此选项控制在数据库日志中记录此消息。这对于决定锁等待是否会产生一个坏的行为是非常有用的。

参数类型：SUSET

取值范围：布尔型

- on表示记录此信息。
- off表示不记录此信息。

默认值：off

log_statement

参数说明：控制记录SQL语句。对于使用扩展查询协议的客户端，记录接收到执行消息的事件和绑定参数的值（内置单引号要双写）。

参数类型：SUSET

须知

即使log_statement设置为all，包含简单语法错误的语句也不会被记录，因为仅在完成基本的语法分析并确定了语句类型之后才记录日志。在使用扩展查询协议的情况下，在执行阶段之前（语法分析或规划阶段）同样不会记录。将log_min_error_statement设为ERROR或更低才能记录这些语句。

取值范围：枚举类型

- none表示不记录语句。
- ddl表示记录所有的数据定义语句，比如CREATE、ALTER和DROP语句。
- mod表示记录所有DDL语句，还包括数据修改语句INSERT、UPDATE、DELETE、TRUNCATE和COPY FROM 。
- all表示记录所有语句，PREPARE、EXECUTE和EXPLAIN ANALYZE语句也同样被记录。

默认值：none

log_temp_files

参数说明：控制记录临时文件的删除信息。临时文件可以用来排序、哈希及临时查询结果。当一个临时文件被删除时，将会产生一条日志消息。

 版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB

- 正整数表示只记录比log_temp_files设定值大的临时文件的删除信息。
- 0表示记录所有的临时文件的删除信息。
- -1表示不记录任何临时文件的删除信息。

默认值：-1

logging_module

参数说明：用于设置或者显示模块日志在服务端的可输出性。该参数属于会话级参数，不建议通过gs_guc工具来设置。

参数类型：USERSET

取值范围：字符串

默认值：所有模块日志在服务端是不输出的，可由SHOW logging_module查看。

设置方法：首先，可以通过SHOW logging_module来查看哪些模块是支持可控制的。例如，查询输出结果为：

```
show logging_module;
logging_module
----------------------
```

```
show
logging_module;

                                          logging_module
-------------------
```

log_statement_filter_list

参数说明：控制记录SQL语句。设置错误码的集合，多个错误码通过“，”分隔，例如：GS_001,GS_002。该错误码日志中不打印SQL语句。该参数仅9.1.0.200及以上版本支持。

参数类型：SUSET

取值范围：字符串

默认值：空字符串

19.11 运行时统计

19.11.1 查询和索引统计收集器

查询和索引统计收集器负责收集数据库系统运行中的统计数据，如在一个表和索引上进行了多少次插入与更新操作、磁盘块的数量和元组的数量、每个表上最近一次执行清理和分析操作的时间等。可以通过查询系统视图pg_stats和pg_statistic查看统计数据。下面的参数设置服务器范围内的统计收集特性。

track_activities

参数说明：控制收集每个会话中当前正在执行命令的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：on

track_counts

参数说明：控制收集数据库活动的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

说明

在autovacuum自动清理进程中选择清理的数据库时，需要数据库的统计数据，故默认值设为on。

默认值：on

track_io_timing

参数说明：控制收集数据库I/O调用时序的统计数据。I/O时序统计数据可以在pg_stat_database中查询。

 版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能，开启时，收集器会重复地去查询当前时间的操作系统，这可能会引起某些平台的重大开销，故默认值设置为off。
- off表示关闭收集功能。

默认值：off

track_functions

参数说明：控制收集函数的调用次数和调用耗时的统计数据。

参数类型：SUSET

须知

当SQL语言函数设置为调用查询的“内联”函数时，不管是否设置此选项，这些SQL语言函数无法被追踪到。

取值范围：枚举类型

- pl表示只追踪过程语言函数。
- all表示追踪SQL和C语言函数。
- none表示关闭函数追踪功能。

默认值：none

update_process_title

参数说明：控制收集每次服务器接收到一个新的SQL语句时产生的进程名称更新的统计数据。

进程名称可以通过ps命令进行查看，在Windows下通过任务管理器查看。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：off

track_thread_wait_status_interval

参数说明：用来定期收集thread状态信息的时间间隔。

参数类型：SUSET

取值范围：整型，0~1440，单位为min。

默认值：30min

 版权所有 © 华为云计算技术有限公司

enable_save_datachanged_timestamp

参数说明：控制是否收集insert/update/delete, exchange/truncate/drop partition操作对表数据改动的时间。

参数类型：USERSET

取值范围：布尔型

- on表示允许收集相关操作对表数据改动的时间。
- off表示禁止收集相关操作对表数据改动的时间。

默认值：on

enable_save_dataaccess_timestamp

参数说明：控制是否记录表的最后一次访问时间。该参数仅8.2.1.210及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许记录表的最后一次访问时间。
- off表示禁止记录表的最后一次访问时间。

默认值：off

instr_unique_sql_count

参数说明：控制是否收集整个集群的Unique SQL以及收集数量限制。

参数类型：SIGHUP

取值范围：整型，0～INT_MAX

- 值为0时，表示不收集Unique SQL统计信息。
- 值大于0时，在CN节点上，将会控制收集的Unique SQL数量不超过该设置值。当收集数量达到限制时，不再收集新的Unique SQL，此时可通过reload调大设置值，继续收集新的Unique SQL。

默认值：0

⚠ 注意

如果新设置值小于原设置值，将会清空对应CN节点已收集的Unique SQL统计信息。

track_sql_count

参数说明：控制对每个会话中当前正在执行的SELECT、INSERT、UPDATE、DELETE、MERGE INTO语句是否进行计数统计，对SELECT、INSERT、UPDATE、DELETE语句的响应时间进行统计，以及对DDL、DML、DCL语句进行计数的统计。

参数类型：SUSET

取值范围：布尔型

版权所有 © 华为云计算技术有限公司

- on表示开启统计功能。
- off表示关闭统计功能。

默认值：on

说明

- track_sql_count参数受track_activities约束：
  - track_activities开启而track_sql_count关闭时，如果查询了gs_sql_count、pgxc_sql_count、gs_workload_sql_count、pgxc_workload_sql_count、global_workload_sql_count、gs_workload_sql_elapse_time、pgxc_workload_sql_elapse_time、或global_workload_sql_elapse_time视图，将会有LOG提示track_sql_count是关闭的。
  - track_activities和track_sql_count同时关闭，那么此时将会有两条LOG，分别提示track_activities是关闭的和track_sql_count是关闭的。
  - track_activities关闭而track_sql_count开启，此时将仅有LOG提示track_activities是关闭。
- 当参数关闭时，查询视图的结果为0行。

enable_parallel_analyze

参数说明：控制内外表analyze采样时是否使用并行的采样方式。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- true表示内外表analyze采样时使用并行的采样方式。
- false表示内外表analyze采样时不使用并行的采样方式。

默认值：true

注意

- 开启enable_parallel_analyze的情况下，对外表进行analyze时，尽量避免对目标外表列增加NOT NULL约束条件，防止外表数据源变更时约束失效导致analyze失败；同时目前并行采样不支持物化视图；当发生由于此类原因导致的analyze失败时，可以设置该参数为false，成功执行analyze。
- 目前并行采样仅支持普通列存内表analyze，当内表使用hstore/hstore_opt或声明为复制表时此优化不会生效。
- 目前并行采样仅支持parquet/orc格式存储的外表，当外表为其他格式时此优化不会生效。

parallel_analyze_workers

参数说明：用于设置并行的analyze采样时并发的线程数量。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~64

 版权所有 © 华为云计算技术有限公司

默认值：10

说明

该参数取值应与集群负载相对应，当集群负载较小时可以根据集群配置适当提高该参数值，进一步增加analyze执行效率。

analyze_sample_multiplier

参数说明：用于设置外表analyze采样的stripe采样率的扩大倍数。该参数仅9.1.0及以上集群版本支持。

参数类型：SUSET

取值范围：整型，0~100，0表示stripe采样率为100%。

默认值：3

19.11.2 性能统计

在数据库的运行过程中，会涉及到锁的访问、磁盘IO操作、无效消息的处理，这些操作都可能是数据库的性能瓶颈，通过DWS提供的性能统计方法，可以方便定位性能问题。

输出性能统计日志

参数说明：对每条查询，以下4个选项控制在服务器日志里记录相应模块的性能统计数据，具体含义如下：

- log_parser_stats控制在服务器日志里记录解析器的性能统计数据。
- log_planner_stats控制在服务器日志里记录查询优化器的性能统计数据。
- log_executor_stats控制在服务器日志里记录执行器的性能统计数据。
- log_statement_stats控制在服务器日志里记录整个语句的性能统计数据。

这些参数只能辅助管理员进行粗略分析，类似Linux中的操作系统工具getrusage()。

参数类型：SUSET

须知

- log_statement_stats记录总的语句统计数据，而其他参数只记录针对每个模块的统计数据。
- log_statement_stats不能和其他任何针对每个模块统计的选项一起打开。

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

版权所有 © 华为云计算技术有限公司

19.12 资源管理

未对数据库资源做控制时，容易出现并发任务抢占资源导致操作系统过载甚至最终崩溃。操作系统过载时，其响应用户任务的速度会变慢甚至无响应；操作系统崩溃时，整个系统将无法对用户提供任何服务。DWS的负载管理功能能够基于可用资源的多少均衡数据库的负载，以避免数据库系统过载。

space_once_adjust_num

参数说明：空间管控和空间统计功能中，控制慢速构建与细粒度校准操作中每次处理的文件个数阈值。该参数8.1.3及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~INT_MAX

- 0表示不启用慢速构建和细粒度校准功能。

默认值：300

说明

文件个数阈值影响数据库资源，建议合理设置。

default_partition_cache_strategy

参数说明：控制分区缓存的默认策略。该参数8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- cache_each_partition_as_possible表示尽可能的缓存插入的数据，插入时遇到不同分区的数据并不一定将数据刷入CU。
- flush_when_switch_partition表示插入时数据属于不同分区就将数据刷入CU。

默认值：cache_each_partition_as_possible

max_active_statements

参数说明：设置全局的最大并发数量。该参数只在CN上应用且针对一个CN上的执行作业。

数据库管理员需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持最大限度的并发作业，且防止并发执行作业过多，引起系统崩溃。

参数类型：SIGHUP

取值范围：整型，-1 ~ INT_MAX。设置为-1和0表示对最大并发数不做限制。

默认值：60

cgroup_name

参数说明：设置当前使用的Cgroups的名字或者调整当前group下排队的优先级。

版权所有 © 华为云计算技术有限公司

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

参数类型：USERSET

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：字符串

默认值：DefaultClass:Medium

说明

DefaultClass:Medium表示DefaultClass下Timeshare控制组中的Medium控制组。

enable_cgroup_switch

参数说明：是否控制数据库执行语句时根据类型自动切换到TopWD组。

参数类型：USERSET

取值范围：布尔型

- on表示控制数据库执行语句时根据类型自动切换到TopWD组。
- off表示控制数据库执行语句时根据类型不自动切换到TopWD组。

默认值：off

memory_tracking_mode

参数说明：设置记录内存信息的模式。

参数类型：USERSET

取值范围：

- none，不启动内存统计功能。
- normal，仅做内存实时统计，不生成文件。
- executor，生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。
- fullexec，生成文件包含执行层申请过的所有内存上下文信息。

默认值：none

memory_detail_tracking

参数说明：设置需要的线程内分配内存上下文的顺序号以及当前线程所在query的plannodeid。

参数类型：USERSET

取值范围：字符型

默认值：空

 版权所有 © 华为云计算技术有限公司

须知

该参数不允许用户进行设置，建议保持默认值。

enable_resource_track

参数说明：设置是否开启资源实时监控功能。该参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开资源监控。
- off表示关闭资源监控。

默认值：on

enable_resource_record

参数说明：设置是否开启资源监控记录归档功能。开启时，对于执行结束的记录，会分别被归档到相应的INFO视图（**GS_WLM_SESSION_INFO**和**GS_WLM_OPERATOR_INFO**）。此参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启资源监控记录归档功能。
- off表示关闭资源监控记录归档功能。

默认值：on

说明

新建集群默认值为on，升级场景该参数的默认值为保持前向兼容维持原值。

enable_track_record_subsql

参数说明：设置是否开启子语句记录归档功能。开启时，存储过程、匿名块内部的子语句会被记录归档到相应的INFO表（**GS_WLM_SESSION_INFO**）。此参数为会话级参数，可在与CN的连接会话中设置生效，仅影响该会话连接中的语句；也可在CN和DN上同时设置，能全局生效。

参数类型：USERSET

取值范围：布尔型

- on表示开启子语句资源监控记录归档功能。
- off表示关闭子语句资源监控记录归档功能。

默认值：on

block_rule_cost

参数说明：设置查询过滤器触发的最小代价。该参数仅9.1.0.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 值为-1时，表示不根据cost判断是否对语句进行拦截，对所有语句进行查询过滤。
- 值为0时，表示对所有cost大于0的语句进行拦截，但特殊类型语句（cost为0）不进行语句拦截。
- 值大于0时，如果语句cost小于block_rule_cost值，则不进行拦截；否则，对语句进行拦截。

默认值：-1

enable_user_metric_persistent

参数说明：设置是否开启用户/资源池历史资源监控转存功能。开启时，对于**PG_TOTAL_USER_RESOURCE_INFO**视图中数据，会定期采样保存到**GS_WLM_USER_RESOURCE_HISTORY**系统表中；对于**GS_RESPOOL_RESOURCE_INFO**视图中数据，会定期采样保存到**GS_RESPOOL_RESOURCE_HISTORY**系统表中。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启用户/资源池历史资源监控转存功能。
- off表示关闭用户/资源池历史资源监控转存功能。

默认值：on

user_metric_retention_time

参数说明：设置用户历史资源监控数据的保存天数。

参数类型：SIGHUP

取值范围：整型，0～3650，单位为天。

- 值等于0时，用户历史资源监控数据将永久保存。
- 值大于0时，用户历史资源监控数据将保存对应天数。

默认值：7

resource_track_level

参数说明：设置当前会话的资源监控的等级。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：枚举型

- none，不开启资源监控功能。
- query，开启query级别资源监控功能，开启此功能会把SQL语句的计划信息（类似explain输出信息）记录到TopSQL中。
- perf，开启perf级别资源监控功能，开启此功能会把包含实际执行时间和执行行数的计划信息（类似explain analyze输出信息）记录到TopSQL中。

 版权所有 © 华为云计算技术有限公司

- operator_realtime, 开启operator级别资源监控功能，开启此功能后会记录实时运行的作业算子信息到TopSQL中，但不会持久化到历史TopSQL中。
- operator，开启operator级别资源监控功能，开启此功能不仅会把包含实际执行时间和执行行数的信息记录到TopSQL中，还会把算子级别执行信息刷新到TopSQL中。

默认值：query

fast_obs_tablesize_method

参数说明：设置快速计算列存v3和v3 hstore_opt表大小的方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- 0：通过list obs文件的方式计算表大小。
- 1：wlm后台统计的方式，通过pg_relfilenode_size计算表大小。
- 2：通过cudesc的每个文件的最大偏移和近似估算表大小。

默认值：2

fast_obs_dbsize_method

参数说明：设置快速计算OBS上db数据大小的方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- 0：直接根据obs桶来估算db大小。
- 1： 普通模式正常计算整个库的大小。

默认值：0

resource_track_cost

参数说明：设置对当前会话的语句进行资源监控的最小执行代价。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 值为-1时，不进行资源监控。
- 值大于或等于0时，对执行代价超过该参数值的语句进行资源监控。

默认值：0

说明

新建集群默认值为0，升级场景该参数的默认值为保持前向兼容维持原值。

 版权所有 © 华为云计算技术有限公司

resource_track_duration

参数说明：设置资源监控实时视图（参见**表15-1**）中记录的语句执行结束后进行历史信息转存的最小执行时间。当执行完成的作业，其执行时间不小于此参数值时，作业信息会从实时视图（以statistics为后缀的视图）转存到相应的历史视图（以history为后缀的视图）中。该参数只有当enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：整型，0～INT_MAX，单位为秒。

- 值为0时，资源监控实时视图（**表15-1**）中记录的所有语句都进行历史信息归档。
- 值大于0时，资源监控实时视图（**表15-1**）中记录的语句的执行时间和排队时间之和超过这个值就会进行历史信息归档。

默认值：60s

resource_track_subsql_duration

参数说明：设置过滤存储过程中子语句的最小执行时间。该参数仅8.2.1及以上集群版本支持。

当存储过程中子语句执行完成时，其执行时间大于此参数设置值时，作业信息会转储到TopSQL归档表中。该参数只有当**enable_track_record_subsql**为on时生效。

参数类型：USERSET

取值范围：整型，0～INT_MAX，单位为秒。

- 值为0时，存储过程中所有子语句都进行历史信息归档。
- 值大于0时，存储过程中子语句记录的执行时间超过设置值就会进行历史信息归档。

默认值：180s

query_exception_count_limit

参数说明：设置作业触发异常规则次数的上限。作业触发异常规则次数达到上限后该作业会被自动加入黑名单且禁止再执行，只有移除黑名单后才能恢复运行。

参数类型：USERSET

取值范围：整型，-1～INT_MAX。

- 值为-1时，表示对作业触发异常规则的次数不做限制，即作业触发异常规则次数再多也不会自动将该作业加入黑名单。
- 值大于等于0时，表示作业触发异常规则次数达到阈值时该作业立即被加入黑名单。其中，0和1含义相同，表示只要作业触发异常规则即会将作业加入黑名单。

默认值：-1

disable_memory_protect

参数说明：禁止内存保护功能。当系统内存不足时如果需要查询系统视图，可以先将此参数置为on，禁止内存保护功能，保证视图可以正常查询。该参数只适用于在系统内存不足时进行系统诊断和调试，正常运行时请保持该参数配置为off。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示禁止内存保护功能。
- off表示启动内存保护功能。

默认值：off

query_band

参数说明：用于标识当前会话的作业类型，由用户自定义。

参数类型：USERSET

取值范围：字符型

默认值：空

注意

此参数在修改时需注意：

1. 当关闭此参数时，代表用户不需要CCN管控功能，此时CCN内存负反馈机制失效。
2. 当作业正在运行时，若此时该GUC参数从关闭修改为开启，此时CCN内存负反馈机制生效，在高并发情况下，会出现暂时的内存不可用情况，待当前正在运行的作业运行完成后，动态负载功能会恢复。
3. 当作业正在运行时，此时修改该GUC参数从开启到关闭状态，若作业大多是属于默认资源池的用户下发的，不建议直接修改，可能会出现内存报错问题。待无从属于默认资源池的用户下发的作业时，可以修改。建议用户绑定用户资源池后，再下发作业。

wlm_memory_feedback_adjust

参数说明：是否开启动态负载管理中的内存负反馈功能。（该参数仅8.2.0及以上集群版本支持）

语句在CN节点上会计算出估算内存，用该值来预占内存资源，语句内存过高估算场景时，语句预占内存资源过大，会引发后续作业排队，导致系统资源利用率下降，内存负反馈机制，会基于DN上的实际使用内存情况，判断如果集群连续一段时间处于高估场景时，会由CCN节点动态降低语句在CN预占的内存资源，将高估的内存腾让出来，给后续作业使用，从一定程度上缓解了语句严重高估导致系统资源利用率下降的问题。

参数类型：SIGHUP

取值范围：字符串

- on表示启用内存负反馈功能。
- off表示关闭内存负反馈功能。
- on()表示启用内存负反馈功能，并且指定触发负反馈需要的时间以及触发负反馈需要的估算内存百分比参数。例如：on(60,50)表示启用内存负反馈功能，触发负反馈机制生效需要连续60秒都是高估场景，还需要语句预占的估算内存总值要超

 版权所有 © 华为云计算技术有限公司

过系统可用内存的50%。默认触发负反馈机制生效的时间长度为50秒，默认触发负反馈生效的最低估算内存总值要超过系统可用内存的40%。

默认值：on

bbox_dump_count

参数说明：在bbox_dump_path定义的路径下，允许存储的DWS所产生core文件最大数。超过此数量，旧的core文件会被删除。此参数只有当enable_bbox_dump为on时才生效。

参数类型：USERSET

取值范围：整型，1～20

默认值：8

说明

在并发产生core文件时，core文件的产生个数可能大于bbox_dump_count。

io_limits

参数说明：该参数8.1.2版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。

参数类型：USERSET

取值范围：整型，0～1073741823

默认值：0

io_priority

参数说明：该参数8.1.2版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。

参数类型：USERSET

取值范围：枚举型

- None
- Low
- Medium
- High

默认值：None

session_respool

参数说明：当前的session关联的resource pool。

参数类型：USERSET

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

 版权所有 © 华为云计算技术有限公司

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：字符串，通过create resource pool所设置的资源池。

默认值：invalid_pool

enable_transaction_parctl

参数说明：是否管控事务块语句和存储过程语句。

参数类型：USERSET

取值范围：布尔型

- on表示对事务块及存储过程语句进行管控。
- off表示对事务块及存储过程语句不进行管控。

默认值：on

session_history_memory

参数说明：设置历史查询视图的内存大小。

参数类型：SIGHUP

取值范围：整型，10MB～max_process_memory的50%。

默认值：100MB

topsql_retention_time

参数说明：设置历史TopSQL中gs_wlm_session_info和gs_wlm_operator_info表中数据的保存时间。

参数类型：SIGHUP

取值范围：整型，0~3650，单位为天。

- 值为0时，表示数据永久保存。
- 值大于0时，表示数据能够保存的对应天数。

默认值：30

注意

- 设置此GUC参数启用数据保存功能前，请先清理gs_wlm_session_info和gs_wlm_operator_info表中的数据。
- 新建集群默认值为30，升级场景该参数的默认值为保持前向兼容维持原值。

transaction_pending_time

参数说明：当enable_transaction_parctl为on时，事务块语句和存储过程语句排队的最大时间。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，-1～INT_MAX，单位为秒。

- 值为-1或0：事务块语句和存储过程语句无超时判断，排队至资源满足可执行条件。
- 值大于0：事务块语句和存储过程语句排队超过所设数值的时间后，无视当前资源情况强制执行。

默认值：0

须知

此参数仅对存储过程及事务块的内部语句有效，即**PG_SESSION_WLMSTAT**中enqueue字段显示为Transaction或StoredProc的语句才会生效。

enable_concurrency_scaling

参数说明：设置是否打开弹性并发扩展功能。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启弹性并发扩展功能。
- off表示关闭弹性并发扩展功能。

默认值：off

concurrency_scaling_max_idle_time

参数说明：用于指定并发扩展创建的弹性逻辑集群的最大空闲时间，如果超过该参数的设置值则进入弹性逻辑集群销毁流程。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~60，单位为分钟。其中，0表示该并发扩展创建的弹性逻辑集群不会被销毁，请谨慎设置。

默认值：5

concurrency_scaling_limit_per_main_vw

参数说明：用于限制每个经典逻辑集群所能创建的最大并发扩展弹性逻辑集群的数量。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~32

默认值：5

 版权所有 © 华为云计算技术有限公司

concurrency_scaling_max_vw_active_statements

参数说明：用于指定并发扩展弹性逻辑集群上所能执行的最大并发数。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，-1~INT_MAX。

- 0，表示该并发扩展弹性逻辑集群不执行任何作业，请谨慎设置。
- -1，表示该并发扩展弹性逻辑集群不受并发管控，请谨慎设置。该取值仅9.1.0.200及以上集群版本支持。

默认值：60

concurrency_scaling_max_waiting_statements

参数说明：用于指定全局排队队列中，触发并发扩展弹性逻辑集群创建流程的弹性作业排队数量。该参数仅9.1.0.100及以上集群版本支持。

如果大于等于该参数的设置值且当前已创建的并发扩展逻辑集群未超过concurrency_scaling_limit_per_main_vw设置值，则自动进入创建并发扩展弹性逻辑集群流程。

参数类型：SIGHUP

取值范围：整型，0~INT_MAX。其中0表示未排队也会创建弹性逻辑集群。设置为0时会消耗大量资源，请谨慎设置。

默认值：10

19.13 自动清理

系统自动清理进程自动执行VACUUM和ANALYZE命令，回收被标识为删除状态的记录空间，并更新表的统计数据。

autovacuum_compaction_rows_limit

参数说明：小CU的阈值，存活元组数小于这个值的就会被认为是小CU。

参数类型：USERSET

取值范围：整型，-1~5000

默认值：2500

⚠ 注意

- **该参数不建议自行修改，如需修改请联系技术支持。**
- 版本低于9.1.0.100，禁止设置该参数，否则可能会导致**主键数据重复**。
- 版本低于9.1.0.100，默认值为-1，表示关闭0CU开关。
- 9.1.0.100版本，该参数默认值为0。
- 9.1.0.200及以上版本，该参数默认值为2500。

版权所有 © 华为云计算技术有限公司

autoanalyze_mode

参数说明：设置autoanalyze的模式。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- normal表示普通的autoanalyze。
- light表示轻量化的autoanalyze。

默认值：

- 若当前集群为低版本升级到8.2.0及以上集群版本，为保持和前向兼容，默认值为normal。
- 若当前集群为新装的8.2.0及以上集群版本，默认值为light。

analyze_stats_mode

参数说明：设置analyze计算统计信息的模式。

参数类型：USERSET

取值范围：枚举类型

- memory表示强制使用内存计算统计信息，不计算多列统计信息。
- sample_table表示强制使用临时采样表计算统计信息，临时表不支持使用该模式。
- dynamic表示按内存maintenance_work_mem大小自适应选择统计信息计算模式，若**maintenance_work_mem**可放下样本，则使用内存方式，否则使用临时采样表方式。

默认值：

- 若当前集群为低版本升级到8.2.0.100及以上集群版本，为保持和前向兼容，默认值为memory。
- 若当前集群为新装的8.2.0.100及以上集群版本，默认值为dynamic。

analyze_sample_mode

参数说明：设置analyze时使用的采样模型。

参数类型：USERSET

取值范围：整型，0～2。

- 0 表示使用默认蓄水池采样模型。
- 1 表示使用优化的蓄水池采样模型。
- 2 表示使用range采样模型。

默认值：0

autovacuum_max_workers

参数说明：设置能同时运行的自动清理线程的最大数量。

 版权所有 © 华为云计算技术有限公司

参数类型：SIGHUP

取值范围：整型，0～128 。其中0表示不会自动进行autovacuum。

默认值：6

说明

该参数与autovacuum共同发挥作用，对系统表和用户表的清理规则如下：

- autovacuum_max_workers = 0时，autovacuum被彻底关闭，不会对任何表做清理。
- autovacuum_max_workers > 0和autovacuum = off，只对系统表和开了delta表的列存表做清理（如vacuum delta表，vacuum cudesc表和delta merge）。
- autovacuum_max_workers > 0和autovacuum = on，会对所有表做清理。

autovacuum_max_workers_hstore

参数说明：设置Autovacuum_max_workers里面，能同时运行的专用于清理hstore的自动清理线程的最大数量。

参数类型：SIGHUP

取值范围：整型，0 ~ 128。其中0表示关闭hstore的自动清理功能。

默认值：3

说明

当需要使用hstore表时，一定要同步修改以下几个参数的默认值，否则会导致hstore表性能严重劣化，推荐的修改配置是：

autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。

autovacuum_naptime

参数说明：设置两次自动清理操作的时间间隔。

参数类型：SIGHUP

取值范围：整型，1～2147483 ，单位为秒（s）。

默认值：60s

autovacuum_vacuum_cost_delay

参数说明：设置在自动VACUUM操作里使用的开销延迟数值。

参数类型：SIGHUP

取值范围：整型，-1～100，单位为毫秒（ms）。其中-1表示使用常规的vacuum_cost_delay。

默认值：2ms

check_crossvw_write

参数说明：控制是否要开启跨VW写场景的检测。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：整型，-1、1。

- -1表示兼容9.0.3版本能力。对于v3表vacuum，只清理所有epoch的非最后一个文件。
- 1表示检查是否为跨VW写场景。对于v3表vacuum，若检查为非跨VW写的场景，清理所有epoch的非最后一个文件，清理当前epoch的最后一个文件，清理小于当前epoch的最后一个文件。若检查为跨VW写的场景，CN节点会获取所有DN节点的epoch信息包装成epochList下发给元数据VW，v3表vacuum会清理所有epoch的非最后一个文件，清理小于max{epochList}且不在epochList中的epoch最后一个文件。

默认值：1

enable_pg_stat_object

参数说明：控制auto vacuum是否更新**PG_STAT_OBJECT**系统表。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示auto vacuum更新PG_STAT_OBJECT系统表。
- off表示auto vacuum不更新PG_STAT_OBJECT系统表。

默认值：on

19.14 客户端连接缺省设置

19.14.1 语句行为

介绍SQL语句执行过程的相关默认参数。

search_path

参数说明：当一个被引用对象没有指定模式时，此参数设置模式搜索顺序。它的值由一个或多个模式名构成，不同的模式名用逗号隔开。

参数类型：USERSET

- 当前会话存放临时表的模式时，可以使用别名pg_temp将它列在搜索路径中，如'pg_temp，public' 。 存放临时表的模式始终会作为第一个被搜索的对象，排在pg_catalog和search_path中所有模式的前面，即具有第一搜索优先级。建议用户不要在search_path中显示设置pg_temp。如果在search_path中指定了pg_temp，但不是在最前面，系统会提示设置无效，pg_temp仍被优先搜索。通过使用别名pg_temp，系统只会在存放临时表的模式中搜索表、视图和数据类型这样的数据库对象，不会在里面搜索函数或运算符这样的数据库对象。
- 系统表所在的模式pg_catalog，总是排在search_path中指定的所有模式前面被搜索，即具有第二搜索优先级（pg_temp具有第一搜索优先级）。建议用户不要在search_path中显式设置pg_catalog。如果在search_path中指定了pg_catalog，但不是在最前面，系统会提示设置无效，pg_catalog仍被第二优先搜索。
- 当没有指定一个特定模式而创建一个对象时，它们被放置到以search_path为命名的第一个有效模式中。当搜索路径为空时，会报错误。

 版权所有 © 华为云计算技术有限公司

- 通过SQL函数current_schema可以检测当前搜索路径的有效值。这和检测search_path的值不尽相同，因为current_schema显示search_path中首位有效的模式名称。

取值范围：字符串

说明

- 设置为"$user"，public时，支持共享数据库（没有用户具有私有模式和所有共享使用public），用户私有模式和这些功能的组合使用。可以通过改变默认搜索路径来获得其他效果，无论是全局化的还是私有化的。
- 设置为空串（''）的时候，系统会自动转换成一对双引号。
- 设置的内容中包含双引号，系统会认为是不安全字符，会将每个双引号转换成一对双引号。

默认值："$user",public

说明

$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

current_schema

参数说明：设置当前的模式。

参数类型：USERSET

取值范围：字符串

默认值："$user",public

说明

$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

default_tablespace

参数说明：当CREATE命令没有明确声明表空间时，所创建对象(表和索引等)的缺省表空间。

- 值是一个表空间的名字或者一个表示使用当前数据库缺省表空间的空字符串。若指定的是一个非默认表空间，用户必须具有它的CREATE权限，否则尝试创建会失败。
- 临时表不使用此参数，可以用**temp_tablespaces**代替。
- 创建数据库时不使用此参数。默认情况下，一个新的数据库从模板数据库继承表空间配置。

参数类型：USERSET

取值范围：字符串，其中空表示使用默认表空间。

默认值：空

default_storage_nodegroup

参数说明：设置当前的默认建表所在的Node Group，目前只适用普通表。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：字符串

- installation，表示默认建表在安装的Node Group上。
- random_node_group，表示默认建表在随机选择的NodeGroup上，该配置8.1.2及以上版本支持，仅用于测试环境。
- roach_group，表示默认建表在所有节点上，该值为roach工具预留，不能用于其他场景。
- 取值为其他字符串，表示默认建表在设置的Node Group上。

默认值：installation

temp_tablespaces

参数说明：当一个CREATE命令没有明确指定一个表空间时，temp_tablespaces指定了创建临时对象（临时表和临时表的索引）所在的表空间。在这些表空间中创建临时文件用来做大型数据的排序工作。

其值是一系列表空间名的列表。如果列表中有多个表空间时，每次临时对象的创建，DWS会在列表中随机选择一个表空间；如果在事务中，连续创建的临时对象被放置在列表里连续的表空间中。如果选择的列表中的元素是一个空串，DWS将自动将当前的数据库设为默认的表空间。

参数类型：USERSET

取值范围：字符串。空字符串表示所有的临时对象仅在当前数据库默认的表空间中创建，请参见**default_tablespace**。

默认值：空

check_function_bodies

参数说明：设置是否在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。为了避免产生问题（比如避免从转储中恢复函数定义时向前引用的问题），偶尔会禁用验证。

参数类型：USERSET

取值范围：布尔型

- on表示在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。
- off表示在CREATE FUNCTION执行过程中不进行函数体字符串的合法性验证。

默认值：on

default_transaction_isolation

参数说明：设置默认的事务隔离级别。

参数类型：USERSET

取值范围：枚举型

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
- read uncommitted：读未提交隔离级别，DWS不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。

 版权所有 © 华为云计算技术有限公司

- repeatable read：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
- serializable：事务可序列化，DWS不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

default_transaction_read_only_probe

参数说明：数据库即将只读（磁盘使用率达到90%）时，控制是否终止特殊语句（下盘语句、产生新表或者新物理文件的语句）的执行。该参数由CM模块检测磁盘使用阈值并设置，不建议用户设置。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示终止特殊语句执行。
- off表示不终止特殊语句执行。

默认值：off

default_transaction_deferrable

参数说明：控制每个新事务的默认延迟状态。只读事务或者那些比序列化更加低的隔离级别的事务除外。

DWS不支持可串行化的隔离级别，因此，该参数无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示默认延迟。
- off表示默认不延迟。

默认值：off

session_replication_role

参数说明：控制当前会话与复制相关的触发器和规则的行为。

参数类型：USERSET

须知

设置此参数会丢弃之前所有缓存的执行计划。

取值范围：枚举型

- origin表示从当前会话中复制插入、删除、更新等操作。
- replica表示从其他地方复制插入、删除、更新等操作到当前会话。
- local表示函数执行复制时会检测当前登录数据库的角色并采取相应的操作。

默认值：origin

 版权所有 © 华为云计算技术有限公司

statement_timeout

参数说明：当语句执行时间超过该参数设置的时间（从服务器收到命令时开始计时）时，该语句将会报错并退出执行。

参数类型：USERSET

取值范围：整型，0～2147483647，单位为毫秒（ms）。

默认值：

- 若当前集群为低版本升级到8.2.0版本，继承升级前参数，默认值为0。
- 若当前集群为新装的8.2.0版本，默认值为24h。

vacuum_freeze_min_age

参数说明：指定VACUUM在扫描一个表时用于判断是否用FrozenXID替换事务ID的中断寿命(在同一个事务中)。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

说明

尽管随时可以将此参数设为0到10亿之间的任意值，但是，VACUUM将默认其有效值范围限制在**autovacuum_freeze_max_age**的50%以内。

默认值：5000000000

vacuum_freeze_table_age

参数说明：指定VACUUM对全表的扫描冻结元组的时间。如果表的**pg_class**.relfrozenxid字段的值已经达到了参数指定的时间，VACUUM对全表进行扫描。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

说明

尽管随时可以将此参数设为零到20亿之间的值，但是，VACUUM将默认其有效值范围限制在**autovacuum_freeze_max_age**的95%以内。定期的手动VACUUM可以在对此表的反重叠自动清理启动之前运行。

默认值：15000000000

bytea_output

参数说明：设置bytea类型值的输出格式。

参数类型：USERSET

取值范围：枚举型

- hex：将二进制数据编码为每字节2位十六进制数字。
- escape：传统化的PostgreSQL格式。采用以ASCII字符序列表示二进制串的方法，同时将那些无法表示成ASCII字符的二进制串转换成特殊的转义序列。

 版权所有 © 华为云计算技术有限公司

默认值：hex

xmlbinary

参数说明：设置二进制值是如何在XML中进行编码的。

参数类型：USERSET

取值范围：枚举型

- base64
- hex

默认值：base64

xmloption

参数说明：当XML和字符串值之间进行转换时，设置document或content是否是隐含的。

参数类型：USERSET

取值范围：枚举型

- document：表示HTML格式的文档。
- content：普通的字符串。

默认值：content

gin_pending_list_limit

参数说明：设置当GIN索引启用快速更新FASTUPDATE时，Pending List（待处理列表）容量的最大值。

Pending List是GIN索引特有的数据结构，用于临时存储索引更新操作。当GIN索引启用FASTUPDATE参数时，新的索引项不会直接写入主索引结构，而是先存入Pending List，待满足特定条件时再批量合并到主索引中。

当Pending List的容量大于设置值时，会把Pending List中数据批量移动到GIN索引数据结构中以进行清理。单个GIN索引可通过更改索引存储参数覆盖此设置值。

参数类型：USERSET

取值范围：整型，64～INT_MAX，单位为KB。

默认值：4MB

19.14.2 区域和格式化

介绍时间格式设置的相关参数。

DateStyle

参数说明：设置日期和时间值的显示格式，以及有歧义的输入值的解析规则。

这个变量包含两个独立的部分：输出格式声明（ISO、Postgres、SQL、German）和输入输出的年/月/日顺序（DMY、MDY、YMD）。这两个可以独立设置或者一起设

 版权所有 © 华为云计算技术有限公司

置。关键字Euro和European等价于DMY；关键字US、NonEuro、NonEuropean等价于MDY 。

参数类型：USERSET

取值范围：字符串

默认值：ISO, MDY

说明

gs_initdb会将这个参数初始化成与**lc_time**一致的值。

设置建议：优先推荐使用ISO格式。Postgres、SQL和German均采用字母缩写的形式来表示时区，例如“EST、WST、CST”等。

IntervalStyle

参数说明：设置区间值的显示格式。

参数类型：USERSET

取值范围：枚举型

- sql_standard表示产生与SQL标准规定匹配的输出。
- postgres表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为ISO时。
- postgres_verbose表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为non_ISO时。
- iso_8601表示产生与在ISO 8601中定义的“格式与代号”相匹配的输出。
- oracle表示产生于Oracle中与numtodsinterval函数相匹配的输出结果，详细请参考numtodsinterval。

须知

IntervalStyle参数也会影响不明确的间隔输入的说明。

默认值：postgres

TimeZone

参数说明：设置显示和解释时间类型数值时使用的时区。

参数类型：USERSET

取值范围：字符串，可查询视图**pg_timezone_names**获得。

默认值：UTC

说明

gs_initdb将设置一个与其系统环境一致的时区值。

 版权所有 © 华为云计算技术有限公司

timezone_abbreviations

参数说明：设置服务器接受的时区缩写值。

参数类型：USERSET

取值范围：字符串，可查询视图pg_timezone_names获得。

默认值：Default

说明

Default表示通用时区的缩写。但也有其他诸如 'Australia' 和 'India' 等用来定义特定的安装。

extra_float_digits

参数说明：调整浮点值显示的数据位数，浮点类型包括float4、float8 以及几何数据类型。参数值加在标准的数据位数上（FLT_DIG或DBL_DIG中合适的）。

参数类型：USERSET

取值范围：整型，-15～3

说明

- 设置为3，表示包括部分有效的数据位。对转储需要精确恢复的浮点数据尤其有用。
- 设置为负数，表示摒弃不需要的数据位。

默认值：0

client_encoding

参数说明：设置客户端的字符编码类型。

请根据前端业务的情况确定。尽量客户端编码和服务器端编码一致，提高效率。

参数类型：USERSET

取值范围：兼容PostgreSQL所有的字符编码类型。其中UTF8表示使用数据库的字符编码类型。

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。
- 参数建议保持默认值，不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置client_encoding参数，即使设置也不会生效，以保证集群内部通信编码格式一致。

默认值：UTF8

推荐值：SQL_ASCII/UTF8

lc_messages

参数说明：设置信息显示的语言。

 版权所有 © 华为云计算技术有限公司

可接受的值是与系统相关的；在一些系统上，这个区域范畴并不存在，不过仍然允许设置这个变量，只是不会有任何效果。同样，也有可能是所期望的语言的翻译信息不存在。在这种情况下，用户仍然能看到英文信息。

参数类型：SUSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_monetary

参数说明：设置货币值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_numeric

参数说明：设置数值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_time

参数说明：设置时间和区域的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

 版权所有 © 华为云计算技术有限公司

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

default_text_search_config

参数说明：设置全文检索的配置信息。

如果设置为不存在的文本搜索配置时将会报错。如果default_text_search_config对应的文本搜索配置被删除，需要重新设置default_text_search_config，否则会报设置错误。

- 其被文本搜索函数使用，这些函数并没有一个明确指定的配置。
- 当与环境相匹配的配置文件确定时，gs_initdb会选择一个与环境相对应的设置来初始化配置文件。

参数类型：USERSET

取值范围：字符串

说明

DWS支持pg_catalog.english，pg_catalog.simple两种配置。

默认值：pg_catalog.english

19.14.3 其他缺省

主要介绍数据库系统默认的库加载参数。

dynamic_library_path

参数说明：设置数据查找动态加载的共享库文件的路径。当需要打开一个可以动态装载的模块并且在CREATE FUNCTION或LOAD命令里面声明的名字没有目录部分时，系统将搜索这个目录以查找声明的文件。

用于dynamic_library_path的数值必须是一个冒号分隔（Windows下是分号分隔）的绝对路径列表。当一个路径名字以特殊变量$libdir为开头时，会替换为DWS发布提供的模块安装路径。例如：

```
dynamic_library_path = '/usr/local/lib/postgresql:/opt/testgs/lib:$libdir'
```

参数类型：SUSET

取值范围：字符串

说明

设置为空字符串，表示关闭自动路径搜索。

默认值：$libdir

gin_fuzzy_search_limit

参数说明：设置GIN索引返回的集合大小的上限。

版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0～INT_MAX，0表示没有限制。

默认值：0

19.15 锁管理

在DWS中，并发执行的事务由于竞争资源可能会导致单机死锁或分布式死锁。本节介绍的参数主要管理事务锁的机制。

deadlock_timeout

参数说明：设置死锁超时检测时间，以毫秒为单位。当申请的锁超过设定值时，系统会检查是否产生了死锁。

- 死锁的检查代价是比较高的，服务器不会在每次等待锁的时候都运行这个过程。在系统运行过程中死锁是不经常出现的，因此在检查死锁前只需等待一个相对较短的时间。增加这个值就减少了无用的死锁检查浪费的时间，但是会减慢真正的死锁错误报告的速度。在一个负载过重的服务器上，用户可能需要增大它。这个值的设置应该超过事务持续时间，这样就可以减少在锁释放之前就开始死锁检查的问题。
- 设置**log_lock_waits**时，这个选项也决定了在一个日志消息发出关于锁等待以前要等待的时间。当需要调查锁延迟时，请设置比正常deadlock_timeout更小的值。

参数类型：SUSET

取值范围：整型，1~2147483647，单位为毫秒（ms）。

默认值：1s

ddl_lock_timeout

参数说明：通过该参数单独指定阻塞DDL语句锁等待的时间，当申请的锁等待时间超过设定值时，系统会报错。该参数仅8.1.3.200及以上版本支持。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

- 如果该参数的值等于0，表示该参数不生效。
- 如果该参数的值大于0，DDL锁阻塞时间为该参数的值，其它锁等待时间为lockwait_timeout参数值。

默认值：0

说明

该参数优先级高于lockwait_timeout，只针对AccessExclusiveLock生效。

ddl_select_concurrent_mode

参数说明：通过该参数控制DDL语句和SELECT语句并发的模式。该参数仅8.1.3.320、8.2.1及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：字符串

- none：表示该参数不生效，DDL语句和select语句不能并发，保持锁等待状态。
- truncate：表示truncate语句被select语句阻塞时，truncate会中断select语句，优先执行。
- exchange：表示exchange语句被select语句阻塞时，exchange会中断select语句，优先执行。
- vacuum_full：表示vacuum full语句被select语句阻塞时，vacuum full会中断select语句，优先执行。（该参数仅9.1.0.200及以上集群版本支持）
- insert_overwrite：表示insert overwrite语句被select语句阻塞时，insert overwrite会中断select语句，优先执行。（该参数仅9.1.0.200及以上集群版本支持）

默认值：none

说明

- 为了给SELECT语句预留响应信号的时间，当前版本中设置的ddl_lock_timeout的值不足1秒时按照1s处理。
- 与高级别的锁冲突（大于1级），不支持并发（比如autoanalyze_mode=normal时，同时SELECT触发了autoanalyze）。
- 该参数支持与单语句中的select或事务块中的select并发，其它版本中仅支持与单语句的select并发。对于单语句或事务块中的select并发处理，参考**enable_cancel_select_in_txnblock**参数。
- 除none以外的其他几个参数可以组合使用，比如配置为“truncate, exchange”表示truncate和exchange语句被select语句阻塞时，二者会中断select语句，优先执行。

enable_cancel_select_in_txnblock

参数说明：控制在事务块中的select是否可以被中断。

参数类型：USERSET

取值范围：布尔型

- on表示开启，即事务块中的select可以被中断。
- off表示关闭，即事务块中的select不可以被中断。

默认值：on

说明

- 该参数8.2.1.236及以上的8.2.1.2xx集群版本支持；9.1.0.200及以上集群版本支持。
- 本参数用于控制处于事务块中的SELECT是否可以被ddl_select_concurrent_mode中指定的DDL操作中断。
- ddl_select_concurrent_mode参数控制truncate、exchange等DDL语句，enable_cancel_select_in_txnblock参数控制select语句。

lockwait_timeout

参数说明：控制单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

 版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

默认值：20min

update_lockwait_timeout

参数说明：允许并发更新参数开启情况下，该参数控制并发更新同一行时单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

默认值：2min

partition_lock_upgrade_timeout

参数说明：分区上的锁级别由允许读的ExclusiveLock升级到读写阻塞的AccessExclusiveLock时，会进行尝试性的锁升级，partition_lock_upgrade_timeout指示了尝试锁升级的超时时间。

- 在分区表上进行MERGE PARTITION和CLUSTER PARTITION操作时，都利用了临时表进行数据重排和文件交换，为了最大程度提高分区上的操作并发度，在数据重排阶段给相关分区加锁ExclusiveLock，在文件交换阶段加锁AccessExclusiveLock。
- 常规加锁方式是等待加锁，直到加锁成功，或者等待时间超过**lockwait_timeout**发生超时失败。
- 在分区表上进行MERGE PARTITION或CLUSTER PARTITION操作时，进入文件交换阶段需要申请加锁AccessExclusiveLock，加锁方式是尝试性加锁，加锁成功了则立即返回，不成功则等待50ms后继续下次尝试，加锁超时时间使用会话级设置参数partition_lock_upgrade_timeout。
- 特殊值：若partition_lock_upgrade_timeout取值-1，表示无限等待，即不停地尝试锁升级，直到加锁成功。

参数类型：USERSET

取值范围：整型，-1 ~ 3000，单位为秒（s）。

默认值：1800

enable_release_scan_lock

参数说明：控制SELECT语句是否在语句执行结束后将一级锁释放。 打开此开关，可以缓解DDL等与事务块中的SELECT锁冲突问题。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启。SELECT语句在语句执行结束后释放一级锁，不会等事务提交才进行释放。
- off表示关闭。

默认值：off

 版权所有 © 华为云计算技术有限公司

vacuum_full_interruptible

参数说明：通过该参数控制VACUUM FULL语句给其它语句的让锁行为。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启。当VACUUM FULL阻塞其它语句时，VACUUM FULL会中断执行，给其它语句让锁。
- off表示关闭。当VACUUM FULL阻塞其它语句时，VACUUM FULL不会中断执行，其它语句需要等待VACUUM FULL执行完毕放锁后才能执行。

默认值：off

19.16 版本和平台兼容性

19.16.1 历史版本兼容性

DWS介绍数据库的向下兼容性和对外兼容性特性的参数控制。数据库系统的向后兼容性能够为旧版本的数据库应用提供支持。本节介绍的参数主要控制数据库的向后兼容性。

array_nulls

参数说明：控制数组输入解析器是否将未用引用的NULL识别为数组的一个NULL元素。

参数类型：USERSET

取值范围：布尔型

- on表示允许向数组中输入空元素。
- off表示向下兼容旧式模式。仍然能够创建包含NULL值的数组。

默认值：on

backslash_quote

参数说明：控制字符串文本中的单引号是否能够用\'表示。

参数类型：USERSET

须知

在字符串文本符合SQL标准的情况下，\没有任何其他含义。这个参数影响的是如何处理不符合标准的字符串文本，包括明确的字符串转义语法是（ E'...' ）。

取值范围：枚举类型

- on表示一直允许使用\'表示。

 版权所有 © 华为云计算技术有限公司

- off表示拒绝使用\'表示。
- safe_encoding表示仅在客户端字符集编码不会在多字节字符末尾包含\的ASCII值时允许。

默认值：safe_encoding

default_with_oids

参数说明：在没有声明WITH OIDS和WITHOUT OIDS的情况下，这个选项控制在新创建的表中CREATE TABLE和CREATE TABLE AS是否包含一个OID字段。它还决定SELECT INTO创建的表里面是否包含OID 。

不推荐在用户表中使用OID，故默认设置为off。需要带有OID字段的表应该在创建时声明WITH OIDS 。

参数类型：USERSET

取值范围：布尔型

- on表示在新创建的表中CREATE TABLE和CREATE TABLE AS可以包含一个OID字段。
- off表示在新创建的表中CREATE TABLE和CREATE TABLE AS不可以包含一个OID字段。

默认值：off

enable_parquet_integer_decimal

参数说明：控制parquet外表写decimal(p,s)类型且p<=18时，选择使用INT64类型或还是FIXED_LEN_BYTE_ARRAY类型存储。

参数类型：USERSET

取值范围：布尔型

- on表示使用INT64类型存储。
- off表示使用FIXED_LEN_BYTE_ARRAY类型存储。

默认值：off

说明

- 此参数在9.1.0集群版本中仅9.1.0.222及以上版本支持。为了兼容hive行为（hive使用FIXED_LEN_BYTE_ARRAY），此参数默认使用FIXED_LEN_BYTE_ARRAY类型存储，并保留历史行为兼容性。
- 暂未支持此参数的集群版本中，DWS parquet外表默认将p<=18的decimal存储为INT64类型。
- INT64或FIXED_LEN_BYTE_ARRAY，此处均表示parquet内部定义的类型，不同的外部引擎可能会选择写不同的类型。

escape_string_warning

参数说明：警告在普通字符串中直接使用反斜杠转义。

- 如果需要使用反斜杠作为转义，可以调整为使用转义字符串语法(E'...')来做转义，因为在每个SQL标准中，普通字符串的默认行为现在将反斜杠作为一个普通字符。

 版权所有 © 华为云计算技术有限公司

- 这个变量可以帮助定位需要改变的代码。

参数类型：USERSET

取值范围：布尔型

默认值：on

lo_compat_privileges

参数说明：控制是否启动对大对象权限检查的向后兼容模式。

参数类型：SUSET

取值范围：布尔型

on表示当读取或修改大对象时禁用权限检查，与PostgreSQL 9.0以前的版本兼容。

默认值：off

quote_all_identifiers

参数说明：当数据库生成SQL时，此选项强制引用所有的标识符（包括非关键字）。这将影响到EXPLAIN的输出及函数的结果，例如pg_get_viewdef。详细说明请参见《数据仓库服务工具指南》中“gs_dump”章节的--quote-all-identifiers选项。

参数类型：USERSET

取值范围：布尔型

- on表示打开强制引用。
- off表示关闭强制引用。

默认值：off

sql_inheritance

参数说明：控制继承语义。

参数类型：USERSET

取值范围：布尔型

off表示各种命令不能访问子表，即默认使用ONLY关键字。这是为了兼容7.1之前版本而设置的。

默认值：on

standard_conforming_strings

参数说明：控制普通字符串文本（'...'）中是否按照SQL标准把反斜扛当普通文本。

- 应用程序通过检查这个参数可以判断字符串文本的处理方式。
- 建议明确使用转义字符串语法（E'...'）来转义字符。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示打开控制功能。
- off表示关闭控制功能。

默认值：on

synchronize_seqscans

参数说明：控制启动同步的顺序扫描。在大约相同的时间内并行扫描读取相同的数据块，共享I/O负载。

参数类型：USERSET

取值范围：布尔型

- on表示扫描可能从表的中间开始，然后选择"环绕"方式来覆盖所有的行，为了与已经在进行中的扫描活动同步。这可能会造成没有用ORDER BY子句的查询得到行排序造成不可预测的后果。
- off表示确保顺序扫描是从表头开始的。

默认值：on

enable_beta_features

参数说明：控制开启某些功能受限的特性，例如GDS表关联操作。这些特性在历史版本未明确禁止，但某些场景功能上存在问题，不建议用户使用。

参数类型：USERSET

取值范围：布尔型

- on表示开启这些功能受限的特性，保持前向兼容。但某些场景可能存在功能上的问题。
- off表示禁止使用这些特性。

默认值：off

19.16.2 平台和客户端兼容性

很多平台都使用数据库系统，数据库系统的对外兼容性给平台提供了很大的方便。

transform_null_equals

参数说明：控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。如果expr得出NULL值则返回真，否则返回假。

- 正确的SQL标准兼容的expr = NULL总是返回NULL（未知）。
- Microsoft Access里的过滤表单生成的查询使用expr = NULL来测试空值。打开这个选项，可以使用该接口来访问数据库。

参数类型：USERSET

取值范围：布尔型

- on表示控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。
- off表示不控制，即expr = NULL总是返回NULL（未知）。

版权所有 © 华为云计算技术有限公司

默认值：off

说明

新用户经常在涉及NULL的表达式上语义混淆，故默认值设为off。

td_compatible_truncation

参数说明：控制是否开启与Teradata数据库相应兼容的特征。该参数在用户连接上与TD兼容的数据库时，可以将参数设置成为on（即超长字符串自动截断功能启用），该功能启用后，在后续的insert语句中，对目标表中char和varchar类型的列插入超长字符串时，会按照目标表中相应列定义的最大长度对超长字符串进行自动截断。保证数据都能插入目标表中，而不是报错。

说明

- 超长字符串自动截断功能不适用于insert语句包含外表的场景。
- 如果向字符集为字节类型编码（SQL_ASCII，LATIN1等）的数据库中插入多字节字符数据（如汉字等），且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

参数类型：USERSET

取值范围：布尔型

- on表示启动超长字符串自动截断功能。
- off表示停止超长字符串自动截断功能。

默认值：off

behavior_compat_options

参数说明：数据库兼容性行为配置项，该参数的值由若干个配置项用逗号隔开构成。不同的兼容性配置项，适用的兼容模式（ORA/TD/MySQL）不同，参见**表19-2**。

参数类型：USERSET

取值范围：字符串

默认值：升级场景下保持前向兼容，即与升级前的集群中该参数的默认值保持一致。新安装集群场景下，该参数默认值为check_function_conflicts,check_function_shippable,unsupported_set_function_case，以防止用户定义错误的函数属性导致严重的问题。

说明

- 当前只支持**表19-2**。
- 配置多个兼容性配置项时，相邻配置项用逗号隔开，例如：set behavior_compat_options='end_month_calculate,display_leading_zero';
- 此参数选项中strict_concat_functions和strict_text_concat_td不能同时设置。
- Oracle兼容模式下不建议设置behavior_compat_options= 'return_null_string' ，如果设置了此选项，应避免将查询结果插入到表里。

 版权所有 © 华为云计算技术有限公司

表 19-2 兼容性配置项

兼容性配置项	兼容性行为控制	适用兼容模式
display_leading_zero	浮点数显示配置项。 ● 不设置此配置项时，对于-1~0和0~1之间的小数，不显示小数点前的0。比如，0.25显示为.25。 ● 设置此配置项时，对于-1~0和0~1之间的小数，显示小数点前的0。比如，0.25显示为0.25。	ORA TD

end_month_calculate	add_months函数计算逻辑配置项。
	假定函数add_months的两个参数分别为param1和param2，param1的月份和param2的月份和为result。
	● 不设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期小，计算结果中的日期字段（Day字段）和param1的日期字段保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-28 00:00:00`
	`(1 row)`
	● 设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期比小，计算结果中的日期字段（Day字段）和result的月末日期保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-31 00:00:00`
	`(1 row)`
		ORA
		TD

compat_analyze_sample	analyze采样行为配置项。 设置此配置项时，会优化analyze的采样行为，主要体现在analyze时全局采样会更精确的控制在3万条左右，更好的控制analyze时Coordinator端的内存消耗，保证analyze性能的稳定性。	ORA TD MySQL
bind_schema_tablespace	绑定模式与同名表空间配置项。 如果存在与模式名schema_name相同的表空间名，那么如果设置search_path为schema_name， default_tablespace也会同步切换到schema_name。	ORA TD MySQL
bind_procedure_searchpath	未指定模式名的数据库对象的搜索路径配置项。 在存储过程中如果不显示指定模式名，会优先在存储过程所属的模式下搜索。 如果找不到，则有两种情况： ● 若不设置此参数，报错退出。 ● 若设置此参数，按照search_path中指定的顺序继续搜索。如果还是找不到，报错退出。	ORA TD MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
correct_to_number	控制to_number()结果兼容性的配置项。 若设置此配置项，则to_number()函数结果与PG11保持一致，否则默认与Oracle保持一致。	ORA

unbind_divide_bound	控制对整数除法的结果进行范围校验。
	● 不设置此配置项时，将对除法结果进行校验，超出范围则报错。例如，示例中INT_MIN/(-1)会因为超过结果大于INT_MAX而报越界错误：
	`SELECT (-2147483648)::int / (-1)::int;`
	`ERROR:  integer out of range`
	● 若设置此配置项，则不需要对除法结果进行范围校验。例如，示例中INT_MIN/(-1)可以得到输出结果INT_MAX+1：
	`SELECT (-2147483648)::int / (-1)::int;`
	` ?column?`
	`------------`
	` 2147483648`
	`(1 row)`
		ORA
		TD

merge_update_multi	控制行存表merge into匹配多行时是否进行update操作。 若设置此配置项，匹配多行时update不报错，否则默认与Oracle保持一致，报错。	ORA TD
disable_row_update_multi	控制行存表update匹配多行时是否进行update操作。 若设置此配置项，匹配多行时update报错，否则默认可以进行多行匹配更新。	ORA TD

return_null_string	控制函数lpad()、rpad()、repeat()、regexp_split_to_table()和split_part()的结果为空字符串''的显示配置项。
	● 不设置此配置项时，空字符串显示为NULL。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	
	`(1 row)`
	● 设置此配置项时，空字符串显示为''。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	`0`
	`(1 row)`
		ORA

compat_concat_variadic	控制函数concat()和concat_ws()对variadic类型结果兼容性的配置项。 若设置此配置项，当concat函数参数为variadic类型时，保留Oracle和Teradata兼容模式下不同的结果形式；否则默认Oracle和Teradata兼容模式下结果相同，且与Oracle保持一致。	ORA TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
convert_string_digit_to_numeric	控制CHAR类型和INT类型进行二元BOOL运算时类型转换优先级的配置项。 ● 不设置此配置项时，类型转换优先级与PG9.6一致。 ● 设置此配置项时，所有CHAR类型和INT类型的二元BOOL运算均强制转换为NUMERIC类型进行计算。设置此配置项后会被影响的CHAR类型包括BPCHAR、VARCHAR、NVARCHAR2、TEXT四种类型，会被影响的INT类型包括INT1、INT2、INT4、INT8四种类型。 **注意** 此配置项只对二元BOOL运算生效，例如，INT2>TEXT、INT4=BPCHAR，非BOOL运算不会受到影响，该配置项暂不支持INT>'1.1'这类UNKNOWN类型运算的转换。由于该配置项开启后，CHAR类型与INT类型的BOOL运算会优先转换为NUMERIC类型进行计算，因此会影响数据库计算性能，当JOIN列为受影响的类型组合时，还会影响执行计划。	ORA TD MyS QL

check_function_conflicts	控制是否检查自定义plpgsql/SQL函数的属性。
	● 不设置此配置项时，不检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性。
	● 设置此配置项时，会检查自定义函数的IMMUTABLE属性，如果函数中含有表，或者是有STABLE/VOLATILE函数时，在执行时会报错。因为函数中如果有表或者STABLE/VOLATILE函数时，与函数定义中的IMMUTABLE属性冲突，即这种场景下，函数的行为非IMMUTABLE。
	例如：设置此参数时，以下场景下会执行报错：
	`CREATE OR replace FUNCTION sql_immutable (INTEGER)`
	`RETURNS INTEGER AS 'SELECT a+$1 from shipping_schema.t4 where a=1;'`
	`LANGUAGE SQL IMMUTABLE`
	`RETURNS NULL`
	`ON NULL INPUT;`
	`select sql_immutable(1);`
	`ERROR:  IMMUTABLE function cannot contain SQL statements with relation or Non-IMMUTABLE function.`
	`CONTEXT:  SQL function "sql_immutable" during startup`
	`referenced column: sql_immutable`
		ORA
		TD
		MyS
		QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

varray_verification	控制是否校验数组长度以及数组类型长度。用于兼容DWS 8.1.0之前的版本。
	若设置此配置项，不会校验数组长度以及数组类型长度。
	-- 场景1
	CREATE OR REPLACE PROCEDURE varray_verification
	AS
	TYPE org_varray_type IS varray(5) OF VARCHAR2(2);
	v_org_varray org_varray_type;
	BEGIN
	v_org_varray(1) := '111'; --例如赋值已经超过了VARCHAR2(2)的限制，配置该选项后将和历史版本保持一致不进行校验
	END;
	/
	--场景2
	CREATE OR REPLACE PROCEDURE varray_verification_i3_1
	AS
	TYPE org_varray_type IS varray(2) OF NUMBER(2);
	v_org_varray org_varray_type;
	BEGIN
	v_org_varray(3) := 1; --例如赋值已经超过了varray(2)的数组长度限制，配置该选项后将和历史版本保持一致不进行校验
	END;
	/
		ORA
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

strict_concat_functions	控制函数textanycat()和anytextcat()在参数存在空值时，对返回值兼容性的配置项。此参数不能和strict_text_concat_td同时设置。
	MySQL兼容模式下，此参数无影响。
	● 不设置此配置项时，函数textanycat()和anytextcat()的返回值默认与Oracle保持一致。
	● 设置此配置项时，若函数textanycat()和anytextcat()的参数存在空值，则返回值也为空值，保留与Oracle和Teradata兼容模式下不同的结果。
	例如，不设置此配置项时，函数textanycat()和anytextcat()的返回值与Oracle保持一致：
	`SELECT textanycat('gauss', cast(NULL as BOOLEAN));`
	` textanycat`
	`------------`
	` gauss`
	`(1 row)`
	
	`SELECT 'gauss' || cast(NULL as BOOLEAN); --这种情况下，||运算符会被转换为函数textanycat`
	` ?column?`
	`----------`
	` gauss`
	`(1 row)`
	设置此配置项时，保留与Oracle和Teradata兼容模式下不同的结果：
	`SELECT textanycat('gauss', cast(NULL as BOOLEAN));`
	` textanycat`
	`------------`
	
	`(1 row)`
	
	`SELECT 'gauss' || cast(NULL as BOOLEAN); --这种情况下，||运算符会被转换为函数textanycat`
	` ?column?`
	`----------`
	
	`(1 row)`
		ORA
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

strict_text_concat_td	Teradata兼容模式下，控制函数textcat()、textanycat()和anytextcat()在参数存在空值时，对返回值兼容性的配置项。此参数不能和strict_concat_functions同时设置。
	● 不设置此配置项时，Teradata兼容模式下函数textcat()、textanycat()和anytextcat()的返回值与DWS一致。
	● 设置此配置项时，若Teradata兼容模式下函数textcat()、textanycat()和anytextcat()的参数存在空值，则返回值为空值。
	例如，不设置此配置项时，函数textcat()、textanycat()和anytextcat()的返回值与DWS保持一致：
	td_compatibility_db=# SELECT textcat('abc', NULL);
	textcat
	--------- 	
	abc
	(1 row)
	td_compatibility_db=# SELECT 'abc' || NULL; --这种情况下，||运算符会被转换为函数textcat()
	?column?
	---------- 	
	abc
	(1 row)
	设置此配置项时，若函数textcat()、textanycat()和anytextcat()的返回值有空值，则返回NULL：
	td_compatibility_db=# SELECT textcat('abc', NULL);
	textcat
	--------- 	
	
	(1 row)
	td_compatibility_db=# SELECT 'abc' || NULL;
	?column?
	---------- 	
	
	(1 row)
		TD

compat_display_ref_table	设置视图中列的显示格式。 ● 不设置该选项时默认带前缀，即tab.col的格式。 ● 设置该选项时与原始定义一致，原始定义带前缀则显示，否则不显示。 SET behavior_compat_options='compat_display_ref_table'; CREATE OR REPLACE VIEW viewtest2 AS SELECT a.c1, **c2**, a.c3, 0 AS c4 FROM viewtest_tbl a; SELECT pg_get_viewdef('viewtest2'); pg_get_viewdef ---------- SELECT a.c1, **c2**, a.c3, 0 AS c4 FROM viewtest_tbl a; (1 row)	ORA TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
para_support_set_func	列存表中控制函数COALESCE()、NVL()、GREATEST()、LEAST()入参是否支持多结果集表达式。 ● 不设置此配置项时，函数入参包含多结果集表达式时，直接报错不支持。 SELECT COALESCE(regexp_split_to_table(c3,'#'), regexp_split_to_table(c3,'#')) FROM regexp_ext2_tb1 ORDER BY 1 LIMIT 5; ERROR: set-valued function called in context that cannot accept a set ● 设置此配置项时，支持函数入参包含多结果集表达式。 SELECT COALESCE(regexp_split_to_table(c3,'#'), regexp_split_to_table(c3,'#')) FROM regexp_ext2_tb1 ORDER BY 1 LIMIT 5; coalesce ---------- a a a a a (5 rows)	ORA TD
disable_select_truncate_parallel	控制分区表的truncate等ddl的锁等级。 ● 设置此配置项时，将禁止分区表的不同分区上truncate与DML(如select)的并发，允许分区表上select的FQS(快速下发)。在OLTP场景下分区表上的简单查询较多，并且没有分区表不同分区truncate与DML并发的需求，可以考虑设置此配置项。 ● 不设置此配置项时，分区表上不同分区的select与truncate可以并发进行，同时关闭分区表的FQS（快速下发）来避免可能的不一致问题。	ORA TD MySQL

bpchar_text_without_rtrim	Teradata兼容模式下，设置此参数时，控制bpchar到text转换保留右侧空格，如果实际长度不足bpchar指定的长度，对其进行补空格操作，兼容Teradata对bpchar字符串的处理风格。
	当前不支持“比较字符串时忽略尾部空格”，拼接后结果如果存在尾部空格，进行比较时会对空格敏感。
	例如，设置参数时：
	td_compatibility_db=# select length('a'::char(10)::text);
	length
	-------- 	
	10
	(1 row)
	
	td_compatibility_db=# select length('a'||'a'::char(10));
	length
	-------- 	
	11
	(1 row)
		TD

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

convert_empty_str_to_null_td	Teradata兼容模式下，设置此参数时，控制to_date, to_timestamp和to_number类型转换函数处理空串时，返回null；同时控制to_char函数处理date类型入参时返回的格式。
	例如：
	未设置此参数时：
	td_compatibility_db=# select to_number('');
	 to_number
	----------- 	
	0
	(1 row)
	
	td_compatibility_db=# select to_date('');
	ERROR: the format is not correct
	DETAIL: invalid date length "0", must between 8 and 10.
	CONTEXT: referenced column: to_date
	
	td_compatibility_db=# select to_timestamp('');
	to_timestamp
	------------------------ 	
	0001-01-01 00:00:00 BC
	(1 row)
	
	td_compatibility_db=# select to_char(date '2020-11-16');
	to_char
	------------------------ 	
	2020-11-16 00:00:00+08
	(1 row)
	设置此参数，若to_number, to_date, to_timestamp函数的参数有空串时：
	td_compatibility_db=# select to_number('');
	 to_number
	----------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_date('');
	 to_date
	--------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_timestamp('');
	 to_timestamp
	-------------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_char(date '2020-11-16');
	 to_char
	------------ 	
	2020/11/16
	(1 row)
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_case_specific	控制字符类型匹配时是否忽略大小写。仅在Teradata兼容模式下生效。 ● 不设置此配置项时，字符类型匹配时，字符的大小写敏感。 ● 设置此配置项时，字符类型匹配时，字符的大小写不敏感。 设置此配置项后会影响的字符类型包括CHAR、TEXT、BPCHAR、VARCHAR、NVARCHAR五种类型，会被影响的操作符包括<、>、=、>=、<=、!=、<>、!=、like、not like、in、not in共12种操作符以及case when、decode 表达式。 **注意** 由于该配置项开启后，字符类型前会增加UPPER函数进而会影响估算逻辑，需要使用增强的估算模型。（建议设置：cost_param=16、cost_model_version = 1、join_num_distinct=-20、qual_num_distinct=200 ）	TD
enable_interval_to_text	控制interval到text类型的隐式转换功能。 ● 设置此选项时，支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- f (1 row) ● 不设置此选项时，不支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- t (1 row)	ORA TD MySQL
case_insensitive	MySQL兼容模式下，设置此参数，控制locate，strpos，instr字符串函数入参大小写不敏感。 目前默认未设置该参数，即入参大小写敏感。 例如： ● 未设置此选项时，入参大小写敏感。 mysql_compatibility_db=# SELECT LOCATE('sub', 'Substr'); locate -------- 0 (1 row) ● 设置此选项时，入参大小写不敏感。 mysql_compatibility_db=# SELECT LOCATE('sub', 'Substr'); locate -------- 1 (1 row)	MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
inherit_not_null_strict_func	控制函数原有的strict属性，参数为1个的函数可以传递NOT NULL属性的行为。即：对于func(x)，如果func()为strict属性，且x包含NOT NULL约束，则认为func(x)也是包含NOT NULL约束的。 该兼容配置项在某些优化场景，例如：NOT IN优化、COUNT(DISTINCT)优化，会有特定的优化效果，但特定场景可能导致结果错误。 目前默认未设置该参数，保证结果正确，但可能导致性能回退，如果出现问题可设置该参数回退到历史版本行为。	ORA TD MyS QL
disable_compat_minmax_expr_mysql	MySQL兼容模式下，控制greatest/least表达式对null入参的处理方式。 默认兼容MySQL。可通过设置此参数，回退到历史版本行为。 ● 不设置此选项时，兼容MySQL行为，入参为null时返回null。 mysql_compatibility_db=# SELECT greatest(1, 2, null), least(1, 2, null); greatest | least ----------+------- | (1 row) ● 设置此选项时，返回非null参数中的最大/小值。 mysql_compatibility_db=# SELECT greatest(1, 2, null), least(1, 2, null); greatest | least ----------+------- 2 | 1 (1 row)	MyS QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

disable_compat_substr_mysql	MySQL兼容模式下，控制substr/substring函数在起始位置pos <= 0时的行为。
	默认兼容MySQL。可通过设置此参数，回退到历史版本行为。
	● 不设置此选项时，兼容MySQL行为，即pos = 0时返回空串，pos < 0时从倒数第 |pos| 个位置开始截取字符。
	mysql_compatibility_db=# SELECT substr('helloworld',0);
	 substr
	-------- 	
	
	(1 row)
	mysql_compatibility_db=# SELECT substring('helloworld',0),substring('helloworld',-2,4);
	 substring | substring
	-----------+-----------
	 | ld
	(1 row)
	● 设置此选项时，pos <= 0时仍然从左侧开始截取字符。
	mysql_compatibility_db=# SELECT substr('helloworld',0);
	 substr
	------------ 	
	 helloworld
	(1 row)
	mysql_compatibility_db=# SELECT substring('helloworld',0),substring('helloworld',-2,4);
	 substring | substring
	------------+-----------
	 helloworld | h
	(1 row)
		MySQL

disable_compat_trim_mysql	MySQL兼容模式下，控制trim/ltrim/rtrim函数对入参的处理方式。 默认兼容MySQL。可通过设置此参数，回退到历史版本行为。 ● 不设置此选项时，兼容MySQL行为，匹配完整子串。 mysql_compatibility_db=# SELECT trim('{}{name}{}','{}'),trim('xyznamezyx','xyz'); btrim | btrim ---------+--------- {name} | namezyx (1 row) ● 设置此选项时，匹配字符集合中的单个字符。 mysql_compatibility_db=# SELECT trim('{}{name}{}','{}'),trim('xyznamezyx','xyz'); btrim | btrim --------+------- name | name (1 row)	MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
light_object_mtime	控制pg_object系统表mtime字段是否会记录对象行为的操作。 ● 设置此选项时，GRANT/REVOKE/TRUNCATE操作不被mtime记录即不更新mtime字段。 ● 不设置此选项时（默认行为），ALTER操作、COMMENT、GRANT/REVOKE和TRUNCATE均会被mtime记录即更新mtime字段。	ORA TD MyS QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

disable_including_all_mysql	MySQL兼容模式下，控制CREATE TABLE ... LIKE语法是否为INCLUDING_ALL模式。
	默认不设置此参数，即MySQL兼容模式下，CREATE TABLE ... LIKE语法默认为INCLUDING_ALL模式。
	可通过设置此参数，回退到历史版本行为。
	● 不设置此选项，MySQL兼容模式下，CREATE TABLE ... LIKE语法为INCLUDING_ALL模式。
	mysql_compatibility_db=# CREATE TABLE mysql_like(id int, name varchar(10), score int) DISTRIBUTE BY hash(id) COMMENT 'mysql_like';
	CREATE TABLE
	mysql_compatibility_db=# CREATE INDEX index_like ON mysql_like(name);
	CREATE INDEX
	mysql_compatibility_db=# \d+ mysql_like;
	Table "public.mysql_like"
	Column | Type | Modifiers | Storage | Stats target | Description
	--------+----------------------+-----------+----------+--------------+-------------
	id | integer | | plain | |
	name | character varying(10) | | extended | |
	score | integer | | plain | |
	Indexes:
	"index_like" btree (name) TABLESPACE pg_default
	Has OIDs: no
	Distribute By: HASH(id)
	Location Nodes: ALL DATANODES
	Options: orientation=row, compression=no
	
	mysql_compatibility_db=# CREATE TABLE copy_like like mysql_like;
	CREATE TABLE
	mysql_compatibility_db=# \d+ copy_like;
	Table "public.copy_like"
	Column | Type | Modifiers | Storage | Stats target | Description
	--------+----------------------+-----------+----------+--------------+-------------
	id | integer | | plain | |
	name | character varying(10) | | extended | |
	score | integer | | plain | |
	Indexes:
	"copy_like_name_idx" btree (name) TABLESPACE pg_default
	Has OIDs: no
	Distribute By: HASH(id)
	Location Nodes: ALL DATANODES
	Options: orientation=row, compression=no
	● 设置此选项，MySQL兼容模式下，CREATE TABLE ... LIKE语法为空模式。
	mysql_compatibility_db=# SET behavior_compat_options = 'disable_including_all_mysql';
	SET
	mysql_compatibility_db=# CREATE TABLE mysql_copy LIKE mysql_like;
	NOTICE: The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
	HINT: Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
	CREATE TABLE
	mysql_db=# \d+ mysql_copy;
		MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
	Table "public.mysql_copy" Column | Type | Modifiers | Storage | Stats target | Description --------+-----------------------+-----------+----------+--------------+------------- id | integer | | plain | | name | character varying(10) | | extended | | score | integer | | plain | | Has OIDs: no Distribute By: ROUND ROBIN Location Nodes: ALL DATANODES Options: orientation=row, compression=no	
cte_onetime_inline	控制非stream计划是否执行inline。 ● 设置此选项时，非stream计划且只被引用一次的CTE执行inline。 ● 不设置此选项时，非stream计划且只被引用一次的CTE不执行inline。	ORA TD MyS QL
skip_first_after_mysql	MySQL兼容模式下，控制是否在ALTER TABLE ADD/MODIFY/CHANGE COLUMN中忽略FIRST/AFTER colname语法。 ● 设置此选项时，将忽略FIRST/AFTER colname语法，执行不报错。 mysql_compatibility_db=# SET behavior_compat_options = 'skip_first_after_mysql'; mysql_compatibility_db=# ALTER TABLE t1 ADD COLUMN b text after a; ALTER TABLE ● 不设置此选项时，将不支持FIRST/AFTER colname语法，执行报错。 mysql_compatibility_db=# SET behavior_compat_options = ''; mysql_compatibility_db=# ALTER TABLE t1 ADD COLUMN b text after a; ERROR: FIRST/AFTER is not yet supported.	MyS QL

enable_division_by_zero_mysql	MySQL兼容模式下，除数为0时，控制除法或取余操作是否报错。（该配置项仅8.1.3.110及以上集群版本支持。）
	● 设置此选项时，除法或取余操作除数为0时，返回NULL。
	compatible_mysql_db=# SET behavior_compat_options = 'enable_division_by_zero_mysql';
	SET
	compatible_mysql_db=# SELECT 1/0 AS test;
	test
	---------- 	
	
	(1 row)
	● 不设置此选项时，除法或取余操作除数为0时，执行报错。
	compatible_mysql_db=# SELECT 1/0;
	ERROR: division by zero
		MyS
		QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
normal_session_id	控制是否生成normal格式的会话ID。 ● 设置此选项时，生成normal格式的会话ID，兼容8.1.3及之前集群版本的会话ID。 SET behavior_compat_options='normal_session_id'; SELECT pg_current_sessionid(); pg_current_sessionid ---------- 1660268184.140594655524608 (1 row) ● 不设置此选项时，生成pretty格式的会话ID。 SET behavior_compat_options=''; SELECT pg_current_sessionid(); pg_current_sessionid ---------- 1660268184.140594655524608.coordinator1 (1 row)	ORA TD MyS QL
disable_jsonb_exact_match	控制操作符匹配规则，在对二元操作符进行模糊匹配时是否考虑jsonb类型。 ● 设置此选项时，当对操作符进行模糊匹配时，会在所有备选（即不区分jsonb类型）中匹配，兼容8.1.1之后集群版本的匹配规则。 SET behavior_compat_options='disable_jsonb_exact_match'; select '2022' - '2'::text; ERROR: cannot delete from scalar ● 不设置此选项时，当对操作符进行模糊匹配时，会在不含jsonb类型的备选中匹配，兼容8.1.1之前集群版本的匹配规则。 SET behavior_compat_options=''; select '2022' - '2'::text; ?column? ---------- 2020 (1 row)	ORA TD MyS QL
merge_into_with_trigger	控制是否支持对有触发器的表执行MERGE INTO操作。 ● 设置此选项时，可以对有触发器的表执行MERGE INTO操作。注意在MERGE INTO操作执行时，表上的触发器不会被触发执行。 ● 不设置此选项时，对有触发器的表执行MERGE INTO操作时报错。	ORA TD MyS QL
add_column_default_v_func	控制alter table add column default expression中expression是否支持volatile类型的函数。 ● 设置此选项时，alter table add column default expression中的expression支持volatile类型的函数。 ● 不设置此选项时，alter table add column default expression中expression不支持volatile类型的函数，如果expression中有volatile的函数，该语句执行会报错。	ORA TD MyS QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_full_group_by_mysql	MySQL兼容模式下，控制查询中GROUP BY后是否可以不体现任何非聚合函数查询字段。 ● 设置此选项时，查询允许GROUP BY后不体现任何非聚合函数查询字段。 SET behavior_compat_options='disable_full_group_by_mysql'; SELECT a,b FROM t1 GROUP BY a; a | b ---+--- 1 | 1 2 | 2 (2 rows) ● 不设置此选项时，查询不允许GROUP BY后不体现任何非聚合函数查询字段，查询报错。 SET behavior_compat_options=''; SELECT a,b FROM t1 GROUP BY a; ERROR: column "t1.b" must appear in the GROUP BY clause or be used in an aggregate function LINE 1: SELECT a,b FROM t1 GROUP BY a; **注意** 该参数需结合full_group_by_mode使用，具体请参考**full_group_by_mode**。 此配置项配置后，若full_group_by_mode为notpadding，对于非GROUP BY后非聚合查询字段需要保证分组后数据一致，否则该列将为随机值。	MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_gc_fdw_filter_partial_pushdown	协同分析外表（类型为gc_fdw）场景下，控制使用过滤条件查询外表数据时过滤条件的下推情况。 ● 设置此选项时，过滤条件中若存在不满足下推条件的因素（如非immutable函数），为了保证结果集文档，则全部过滤条件不下推，此行为兼容8.2.1版本之前的行为。 --源端集群建表 CREATE TABLE t1(c1 INT, c2 INT, c3 INT) DISTRIBUTE BY HASH(c1); --本地集群建相同结构外表 CREATE SERVER server_remote FOREIGN DATA WRAPPER gc_fdw options(ADDRESS 'address', DBNAME 'dbname', USERNAME 'username', PASSWORD 'password'); CREATE FOREIGN TABLE t1(c1 INT, c2 INT, c3 INT) SERVER server_remote; --打开参数时，条件下推情况 SET behavior_compat_options = 'disable_gc_fdw_filter_partial_pushdown'; EXPLAIN (verbose on,costs off) SELECt * FROM t1 WHERE c1>3 AND c2 <100 AND now() - '20230101' < c3; QUERY PLAN --- --- Streaming (type: GATHER) Output: c1, c2, c3 Node/s: All datanodes -> Foreign Scan on ca_schema.t1 Output: c1, c2, c3 Filter: ((t1.c1 > 3) AND (t1.c2 < 100) AND ((now() - '2023-01-01 00:00:00-08'::timestamp with time zone) < (t1.c3)::interval)) Remote SQL: SELECT c1, c2, c3 FROM ca_schema.t1 (7 rows) ● 不设置此选项时，过滤条件中可下推的部分将下推到源端集群执行，不可下推部分将在本地集群执行，如此可提升外表查询效率。 --关闭参数时，条件下推情况 SET behavior_compat_options = ''; EXPLAIN (verbose on,costs off) SELECT * FROM t1 WHERE c1>3 AND c2 <100 AND now() - '20230101' < c3; QUERY PLAN --- --- Streaming (type: GATHER) Output: c1, c2, c3 Node/s: All datanodes -> Foreign Scan on ca_schema.t1 Output: c1, c2, c3 Filter: ((now() - '2023-01-01 00:00:00-08'::timestamp with time zone) < (t1.c3)::interval) Remote SQL: SELECT c1, c2, c3 FROM ca_schema.t1 WHERE ((c1 > 3)) AND ((c2 < 100)) (7 rows)	ORA TD MyS QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
ignore_unshipped_concurrent_update	并发更新场景下，当前会话的语句不下推时，如果更新的元组被其它会话更新成新元组后，控制当前会话执行UPDATE/DELETE语句时是否忽略处理新的元组。默认不再处理新的元组。 ● 设置此选项时，当前会话执行UPDATE/DELETE语句时忽略处理新的元组，当前UPDATE/DELETE语句执行成功，此行为会导致并发更新场景数据不一致。此行为兼容8.2.1版本之前的行为。 ● 不设置此选项时，当前会话执行UPDATE/DELETE语句检测元组已被更新时，会重新执行当前会话的UPDATE/DELETE语句，以保证数据一致性。语句执行重试次数受**max_query_retry_times**参数控制。	ORA TD MySQL
disable_set_global_var_on_datanode	控制set_config函数是否可以在DN上设置全局变量。 ● 设置此选项时，禁止set_config函数在DN上设置全局变量。此行为默认兼容8.2.1版本之前的行为。 ● 不设置此选项时，允许set_config函数在DN上设置全局变量，这将会使得CN与DN上全局变量值不一致，导致read_global_var函数下推时可能出错。	ORA TD MySQL

variadic_null_check	控制variadic参数是否能传入NULL参数的选项，默认不开启。（该参数仅8.3.0及以上集群版本支持）
	● 设置此选项时，禁止variadic传入NULL参数，传入后会报错。
	`SET behavior_compat_options = 'variadic_null_check';`
	`SELECT format ( 'array', VARIADIC NULL);`
	`ERROR:  VARIADIC parameter must be an array`
	说明
	为了兼容mysql，compat_concat_variadic开启时，对concat函数和concat_ws函数不会生效，仍然可以传入NULL参数。
	● 不设置此选项时，允许variadic传入NULL参数。
	`SET behavior_compat_options = '';`
	`SELECT format ( 'array', VARIADIC NULL);`
	` format`
	`--------`
	` array`
	`(1 row)`
		ORA
		TD
		MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_use_syscol_in_replicate_table	控制复制表在INSERT、UPDATE、MERGE INTO和DELETE时是否可以采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件。该选项默认不设置。 ● 不设置该选项时，如果复制表在INSERT、UPDATE、MERGE INTO和DELETE时采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件，会报如下错误： ERROR: Can not use system column oid/ctid/tableoid/xc_node_id in Replication Table. ● 设置该选项时，可在复制表中使用系统列id/ctid/tableoid/xc_node_id进行INSERT、UPDATE、MERGE INTO和DELETE。 **注意** 复制表在INSERT、UPDATE、MERGE INTO和DELETE时采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件，语句有导致集群core的风险，请慎重考虑配置该选项。	ORA TD MYS QL
enable_force_add_batch	当参数support_batch_bind设置为on且参数enable_fast_query_shipping和enable_light_proxy设置off时，该选项控制DWS是否接受addbatch模式的U报文。该选项默认不设置。 ● 当参数support_batch_bind设置为on且参数enable_fast_query_shipping和enable_light_proxy设置off，不设置该选项时，DWS不再接收addbatch模式的U报文。 ● 当参数support_batch_bind设置为on且参数enable_fast_query_shipping和enable_light_proxy设置off，设置该选项时，DWS接收addbatch模式的U报文。但是入库速度较慢，有内存不足风险，需谨慎设置该选项。	ORA TD MYS QL
disable_mergesort_without_material	控制当前stream片段是否含有物化算子时stream算子采用merge sort。 ● 设置此选项时，若当前stream片段含有物化算子（material、sort、agg、CteScan），则可采用merge sort，否则不可采用merge sort。 ● 不设置此选项时，是否采用merge sort不需要判断当前stream片段是否含有物化算子。	ORA TD MYS QL
enable_pushdown_groupingset_subquery	当子查询中含有grouping set时，该选项控制是否可将外层查询中仅与该子查询相关的条件下推到子查询中。 ● 当子查询中含有grouping set时，设置此选项，则不可将外层查询的条件下推到子查询中。 ● 当子查询中含有grouping set时，不设置此选项，则将外层查询的条件下推到子查询中。	ORA TD MYS QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_whole_row_var	该参数主要涉及两个场景。1. 控制是否允许表或视图出现在SQL表达式中，包括但不限于查询的目标列表、GROUP BY列表等；2. 控制是否允许非表的record出现的SQL表达式。该选项仅8.3.0及以上集群版本支持。 ● 设置此选项时，允许表或视图出现在SQL表达式中。 SET behavior_compat_options = 'enable_whole_row_var'; SELECT a1 FROM t a1; a1 ---- (0 rows) SELECT t FROM (SELECT 1) as t; t ----- (1) (1 rows) ● 不设置此选项时，不允许表或视图出现在SQL表达式中，如果SQL中出现则报错。 SET behavior_compat_options = ''; SELECT a1 FROM t a1; ERROR: Table or view cannot appear in expression. Table/view name: t, alias: a1. Please check targetList, groupClause etc. SELECT t FROM (SELECT 1) as t; ERROR: Non-table records cannot appear in expression. alias: t. Please check targetList, groupClause etc.	ORA TD MYSQL
enable_unknown_datatype	控制是否允许创建含有unknown类型列的表。该选项仅8.3.0及以上集群版本支持。 ● 设置此选项时，允许创建含有unknown类型列的表。 SET behavior_compat_options = 'enable_unknown_datatype'; CREATE TABLE t(a unknown); WARNING: column "a" has type "unknown" DETAIL: Proceeding with relation creation anyway. CREATE TABLE ● 不设置此选项时，不允许创建含有unknown类型列的表，如果建表SQL含有unknown列，则报错。 SET behavior_compat_options = ''; create table t(a unknown); ERROR: column "a" has type "unknown"	ORA TD MYSQL
alter_distribute_key_by_partition	控制ALTER TABLE修改分区表分布列时INSERT INTO是否按分区执行。 ● 设置此选项时，按分区执行INSERT INTO，使用内存降低但性能劣化。 ● 不设置此选项时，将分区表整表进行INSERT INTO，性能较好但使用内存较多。	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_update_returning_check	控制是否禁用涉及到多表关联，更新复制表且带returning语句场景。该选项仅8.3.0及以上集群版本支持。 ● 不设置此选项时，更新复制表且带returning语句场景，若涉及到多表关联，则报下述错误： ERROR: Unsupported FOR UPDATE replicated table joined with other table. ● 设置此选项，与旧版本前向兼容，但是在更新复制表且带returning语句场景，若涉及到多表关联，存在结果集不一致问题。	ORA TD MYSQL

check_function_shippable	控制是否检查自定义plpgsql/SQL函数的属性。该选项仅8.3.0及以上集群版本支持。
	● 不设置此配置项时，不检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性。
	● 设置此配置项时，会遵循以下逻辑检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性：
	– 设置白名单：对于**DBMS_OUTPUT**的3个函数，跳过check_function_shippable的检测。
	– 如果自定义函数内部有DML语句，外层是IMMUTABLE或者SHIPPABLE，均是下推，故报错。
	– 如果自定义函数外层是shippable，内层是immutable，则通过检查。
	– 如果自定义函数外层是shippable，内层是非immutable且内层是shippable，则通过检查。
	– 如果函数外层是shippable，但是内层非上述的情况，则报错。
	例如：设置此参数时，以下场景下会执行报错：
	CREATE OR replace function func_ship(a int)
	returns int
	LANGUAGE plpgsql
	NOT FENCED SHIPPABLE
	AS $function$
	begin
	perform test_ship();
	return a;
	EXCEPTION WHEN OTHERS THEN
	return a;
	end $function$;
	select func_ship(a) from tt3;
	ERROR: parent function is shippable but child is not immutable or shippable.
		ORA
		TD
		MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_full_string_agg	控制string_agg(a, delimiter) over (partition by b order by c)场景行为，采用窗口内的全量聚合逻辑还是增量聚合逻辑。该参数8.3.0及以上集群版本支持。 不设置此选项时，采用增量聚合逻辑。设置此选项时，采用窗口内的全量聚合逻辑。默认不设置此选项。 CREATE TABLE string_agg_dn_col(c1 int, c2 text) WITH(orientation = column) distribute by hash(c1); INSERT INTO string_agg_dn_col values(1, 'test'); INSERT INTO string_agg_dn_col values(1, 'haidian'); INSERT INTO string_agg_dn_col values(1, 'nanjing'); SELECT t.c1 AS c1, string_agg(t.c2, ',') OVER(PARTITION BY t.c1 ORDER BY t.c2) AS c2 FROM string_agg_dn_col t ORDER BY c2; c1 | c2 ----+---------------------- 1 | haidian 1 | haidian,nanjing 1 | haidian,nanjing,test (3 rows) SET behavior_compat_options='enable_full_string_agg'; SELECT t.c1 AS c1, string_agg(t.c2, ',') OVER(PARTITION BY t.c1 ORDER BY t.c2) AS c2 FROM string_agg_dn_col t ORDER BY c2; c1 | c2 ----+---------------------- 1 | haidian,nanjing,test 1 | haidian,nanjing,test 1 | haidian,nanjing,test (3 rows)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

enable_banker_round	控制数值类型舍入行为使用四舍五入还是银行家算法。该选项仅8.3.0及以上集群版本支持。
	受到参数控制的行为包括：
	● INSERT INTO和::xxx指定类型时自动触发的类型转换：所有整数类型（int1，int2，int4，int8）、所有任意精度类型（decimal，numeric，number）和money类型。
	● 对numeric类型的舍入和转换函数：round(xxx.xx,s),cast('xxx.xx',numeric),to_char(xxx.xx,'xxx')。
	● numeric类型的数学计算。
	说明
	银行家算法舍入规则：舍入位后的值大于5时，进位；舍入位后的值小于5时，舍去不进位；舍入位后的值等于5且全为零时，则按照修约前数字奇偶性决定舍入，奇数进位，偶数舍去；舍入位后的值等于5且存在非零数字，则直接进位。
	● 设置选项时，舍入行为采用银行家算法：
	`SET behavior_compat_options = enable_banker_round;`
	`SELECT 1.5::int1,1.5::int2,1.5::int4,1.5::int8,1.5::numeric(10,0),1.115::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 2 | 2 | 2 | 2 | 2 | $1.12`
	`SELECT 0.5::int1,0.5::int2,0.5::int4,0.5::int8,0.5::numeric(10,0),1.105::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 0 | 0 | 0 | 0 | 0 | $1.10`
	`SELECT round(1.05,1),round(1.15,1),cast('1.05',numeric(10,1)),cast('1.15',numeric(10,1)),to_char(1.05,'9D9'),to_char(1.15,'9D9');`
	` round | round | numeric | numeric | to_char | to_char`
	`-------+-------+---------+---------+---------+---------`
	` 1.0 | 1.2 | 1.0 | 1.2 | 1.0 | 1.2`
	● 不设置选项时，舍入行为采用四舍五入：
	`SET behavior_compat_options = '';`
	`SELECT 1.5::int1,1.5::int2,1.5::int4,1.5::int8,1.5::numeric(10,0),1.115::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 2 | 2 | 2 | 2 | 2 | $1.12`
	`SELECT 0.5::int1,0.5::int2,0.5::int4,0.5::int8,0.5::numeric(10,0),1.105::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 1 | 1 | 1 | 1 | 1 | $1.11`
	`SELECT round(1.05,1),round(1.15,1),cast('1.05',numeric(10,1)),cast('1.15',numeric(10,1)),to_char(1.05,'9D9'),to_char(1.15,'9D9');`
	` round | round | numeric | numeric | to_char | to_char`
	`-------+-------+---------+---------+---------+---------`
	` 1.1 | 1.2 | 1.1 | 1.2 | 1.1 | 1.2`
		ORACLE
		TD
		MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
create_partition_local_index	控制在分区表上默认创建的索引是全局（GLOBAL）索引还是本地（LOCAL）索引。该选项仅8.2.1.210及以上集群版本支持。 ● 不设置此选项时，在分区表上默认创建全局（GLOBAL）索引。 SET behavior_compat_options = ''; CREATE INDEX sale_id_idx ON sales(sale_id); ERROR: partitioned table does not support global index HINT: please set behavior_compat_options = 'create_partition_local_index'to create local index by default. ● 设置此选项时，在分区表上默认创建本地（LOCAL）索引。 SET behavior_compat_options = create_partition_local_index; CREATE INDEX sale_id_idx ON sales(sale_id); CREATE INDEX	ORA
enable_int_division_by_truncate	控制整数除法行为结果集输出整数还是浮点数，行为兼容PG或者兼容ORA。 ● 设置此选项时，整数除法结果输出整数，小数位截断，兼容PG行为。 SET behavior_compat_options = 'enable_int_division_by_truncate'; SELECT 8::int8 / 3::int8, 8::int4 / 3::int4, 8::int2 / 3::int2, 8::int1 / 3::int1; ?column? | ?column? | ?column? | ?column? ----------+----------+----------+---------- 2 | 2 | 2 | 2 (1 row) ● 不设置此选项时，整数除法结果输出浮点数，包含小数位，兼容ORA行为。 SET behavior_compat_options = ''; SELECT 8::int8 / 3::int8, 8::int4 / 3::int4, 8::int2 / 3::int2, 8::int1 / 3::int1; ?column? | ?column? | ?column? | ?column? ------------------+------------------+------------------+------------------ 2.66666666666667 | 2.66666666666667 | 2.66666666666667 | 2.66666666666667 (1 row)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
select_into_allow_multi_result	控制存储过程中SELECT INTO是否允许接收多行结果或者没有结果集。 ● 不设置此选项时，存储过程中使用SELECT INTO语句插入多值或者空时，执行报错： CREATE TABLE PersonTmpTable (id int primary key, name varchar(64), age int, city varchar(512) default null, update_time timestamp default null); INSERT INTO PersonTmpTable VALUES(1,'zhangsan', 23, 'wuhan', '2022-12-10 15:39:32'); INSERT INTO PersonTmpTable VALUE(2,'lisi', 11, 'beijing', '2022-12-13 15:39:32'); INSERT INTO PersonTmpTable VALUE(3,'wangwu', 46, 'xian', '2022-12-1 15:39:32'); INSERT INTO PersonTmpTable VALUE(4,'zhaoliu', 46, 'wuhan', '2022-12-31 15:39:32'); SET behavior_compat_options = ''; CREATE OR REPLACE function func_test1 RETURNS int LANGUAGE plpgsql AS $$DECLAREtmp INTEGER;BEGINSELECT id INTO tmp FROM PersonTmpTable;dbms_output.put_line(tmp);return tmp;END;$$;select func_test1(); ERROR: query returned no rows when process INTO. ERROR: query returned 2 rows more than one row. ● 设置此选项，存储过程中使用SELECT INTO语句插入多值或者空时，成功插入第一行或者空。 SET behavior_compat_options = 'select_into_allow_multi_result'; CREATE OR REPLACE function func_test1 RETURNS int LANGUAGE plpgsql AS $$DECLAREtmp INTEGER;BEGINSELECT id INTO tmp FROM PersonTmpTable;dbms_output.put_line(tmp);return tmp;END;$$;select func_test1(); CREATE FUNCTION	ORA TD MYS QL
orderby_null_first	控制ORDER BY排序时NULL值是否默认视为最小值。 ● 设置此选项时，ORDER BY排序时NULL值默认视为最小值。 SET behavior_compat_options = 'orderby_null_first'; SELECT * FROM test ORDER BY a; a | b ---+--- | 1 1 | 2 ● 不设置此选项时，ORDER BY排序时NULL值默认视为最大值。 SET behavior_compat_options = ''; SELECT * FROM test ORDER BY a; a | b ---+--- 1 | 2 | 1	TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
unsupported_set_function_case	控制是否支持case when条件中含有返回为多结果集函数。该选项仅8.3.0.100及以上集群版本支持。9.1.0及以上新安装的集群版本中默认开启该参数。 ● 设置此选项时，列存不支持case when条件中含有返回为多结果集函数： CREATE TABLE t1(id int, c1 text) with(orientation=column); INSERT INTO t1 values(1, 'a#1'); SET behavior_compat_options = 'unsupported_set_function_case'; SELECT CASE split_part(regexp_split_to_table(c1, E'\,'),'#',1) when 'a' then c1 else null end from t1; ERROR: set-valued function called in context that cannot accept a set ● 不设置此选项时，列存支持case when条件中含有返回为多结果集函数： SET behavior_compat_options = ''; SELECT CASE split_part(regexp_split_to_table(c1, E'\,'),'#',1) when 'a' then c1 else null end from t1; case ------ a#1 (1 row)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

enable_change_search_path	控制在形成通用计划generic_plan后能否修改搜寻路径。该选项仅9.1.0及以上集群版本支持。
	● 不设置此选项时，当设置新的搜索路径（search_path）并执行EXECUTE语句，数据库将仍然从原表的schema下寻找对应表。
	CREATE SCHEMA s1
	  CREATE TABLE abc(f1 INT);
	CREATE SCHEMA s2
	  CREATE TABLE abc(f1 INT);
	SET search_path = s1;
	INSERT INTO s1.abc VALUES(123);INSERT INTO s2.abc VALUES(456);
	SET search_path = s1;
	PREPARE p1 AS SELECT f1 FROM abc;
	EXECUTE p1;
	 f1
	----- 	
	 123
	(1 row)
	SET search_path = s2;
	SELECT f1 FROM abc;
	 f1
	----- 	
	 456
	(1 row)
	EXECUTE p1;
	 f1
	----- 	
	 123
	(1 row)
	● 设置此选项时，当设置新的搜索路径（search_path）并执行EXECUTE语句，数据库将从新设置的搜索路径中寻找对应表。
	SET behavior_compat_options = 'enable_change_search_path';
	EXECUTE p1;
	 f1
	----- 	
	 456
	(1 row)
	
	SET search_path = s1;
	EXECUTE p1;
	 f1
	----- 	
	 123
	(1 row)
		TD

enable_varchar_to_nvarchar2	控制通过DDL语句创建及更新的varchar字段是否自动切换为nvarchar2字段。该选项仅9.1.0及以上集群版本支持。 ● 设置此选项时，通过DDL语句创建/更新的varchar字段自动切换为nvarchar2字段。 ● 不设置此选项时，通过DDL语句创建/更新的varchar字段不会切换为nvarchar2字段。	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
normalize_negative_zero	控制ceil()，round()函数在处理float类型特定值时返回-0与否。 ● 设置此选项时，ceil()处理(-1,0)，round()处理[-0.5, 0)时返回值会返回0。 SET behavior_compat_options='normalize_negative_zero'; SELECT ceil(cast(-0.1 as float)); ceil ------ 0 (1 row) SELECT round(cast(-0.1 as FLOAT)); round ------- 0 (1 row) ● 不设置此选项时，ceil()处理(-1,0)，round()处理[-0.5, 0)时返回值会返回-0。 SET behavior_compat_options = ''; SELECT ceil(cast(-0.1 as FLOAT)); ceil ------ -0 (1 row) SELECT round(cast(-0.1 as FLOAT)); round ------- -0 (1 row)	ORA TD MyS QL
disable_client_detection_commit	控制在每次事务提交之前，检测与客户端的连接是否存在。如果不存在，则报错，回滚该事务，防止因断连未感知重复下发导致的数据重复问题。 ● 不设置此选项时，每次事务提交之前检测一次客户端连接是否存在。 ● 设置此选项时，事务提交之前不检测客户端连接是否存在。	ORA TD MyS QL
change_illegal_char	控制gds读取UTF8非法字符的显示问题。该选项仅8.3.0.100及以上集群版本支持。 该选项打开时，gds读取到UTF8不兼容的非法字符从“？”变为显示“&#xfffd;”。	MyS QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
row_use_pseudo_name	控制row相关表达式是否对匿名列生成伪列名。该选项仅9.1.0.100及以上集群版本支持。 ● 不设置此选项时，row表达式中若存在对应的真实列名，则使用真实的列名，若本身为匿名列，则生成f1、f2...fn的伪列名。 SELECT row_to_json(row(1,'foo')); row_to_json ---------- {"f1":1,"f2":"foo"} (1 row) CREATE TABLE json_tbl(id INT, x INT, y text) WITH (ORIENTATION = COLUMN); INSERT INTO json_tbl VALUES (1, 1, 'txt1'), (2, 2, 'txt2'), (3, 3, 'txt3'); SELECT to_json(t.*) FROM json_tbl t; to_json ---------- {"id":3,"x":3,"y":"txt3"} {"id":1,"x":1,"y":"txt1"} {"id":2,"x":2,"y":"txt2"} (3 rows) ● 设置此选项时，row表达式中将在列存表条件下生成f1、f2...fn的伪列名。 SET behavior_compat_options='ROW_USE_PSEUDO_NAME'; SELECT to_json(t.*) FROM json_tbl t; to_json ---------- {"f1":3,"f2":3,"f3":"txt3"} {"f1":1,"f2":1,"f3":"txt1"} {"f1":2,"f2":2,"f3":"txt2"} (3 rows)	ORA TD MySQL
enable_trunc_orc_string	控制orc格式外表字段为varchar(n)，但是orc文件字段类型为string，且string长度超过n时，外表查询的行为。 ● 不设置此选项时，查询报错，提示字段超长。 ● 设置此选项时，查询正常，按照varchar(n)定义长度进行截断。	ORA TD MySQL
gds_fill_multi_missing_fields	控制GDS外表容错性参数fill_missing_fields设置为true/on时的行为。在GDS外表fill_missing_fields设置为true/on时，允许数据源文件一行中最后若干字段缺失，处理方式是将这些字段设置为NULL。而在此之前，仅允许数据源文件一行中最后一个字段缺失，否则报错。 ● 设置此选项时，GDS外表允许数据源文件一行最后多个字段缺失。 ● 不设置此选项时，GDS外表允许数据源文件一行最后一个字段缺失。兼容历史行为。	ORA TD MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
correct_convert_tz	控制convert_tz函数查询结果是否与MySQL行为一致。该选项仅9.1.0.210及以上集群版本支持。 ● 设置此选项时，convert_tz函数在时区转换时返回预期结果，与MySQL行为一致。 set behavior_compat_options = 'correct_convert_tz'; select convert_tz('2020-1-1 12:00', '+00:00', 'PRC'); convert_tz --------- 2020-01-01 20:00:00 (1 row) ● 不设置此选项时，convert_tz函数在时区转换时结果不符合预期，与MySQL行为不一致。 set behavior_compat_options = ''; select convert_tz('2020-1-1 12:00', '+00:00', 'PRC'); convert_tz --------- 2020-01-01 04:00:00 (1 row)	ORA TD MySQL
set_timezone_mysql_style	控制set timezone='+08:00' / '-07:00'等类型时（'+/- 时:分'形式）是否与MySQL行为一致。该选项仅9.1.0.210及以上集群版本支持。 ● 设置此选项时，set timezone='+08:00'表示时区设置为东八区，与MySQL行为一致。 set behavior_compat_options = 'set_timezone_mysql_style'; set timezone='UTC'; select * from t; a --------- 2020-01-01 12:00:00+00 (1 row) set timezone='+08:00'; select * from t; a --------- 2020-01-01 20:00:00+08 (1 row) ● 不设置此选项时，set timezone='+08:00'表示时区设置为西八区，与MySQL行为不一致，为PG行为。 set behavior_compat_options = ''; set timezone='UTC'; select * from t; a --------- 2020-01-01 12:00:00+00 (1 row) set timezone='+08:00'; select * from t; a --------- 2020-01-01 04:00:00-08 (1 row)	ORA TD MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
parquet_int96_is_localtime	控制读取parquet格式外表的int96类型数据（timestamp专用）时，是否将存储的原始时间戳使用数据库时区修正。该选项仅9.1.0.210及以上集群版本支持。 ● 设置此选项时，认为parquet int96类型存储的是本地时间（非UTC+0），读取结果不会再进行时区修正。 set behavior_compat_options = 'parquet_int96_is_localtime'; set time zone 'Asia/Shanghai'; select * from test_parquet_int96; a --------------------- 2025-01-25 00:00:00 (1 row) set time zone 'Atlantic/Cape_Verde'; select * from test_parquet_int96; a --------------------- 2025-01-25 00:00:00 (1 row) ● 不设置此选项时，认为parquet int96类型存储是UTC+0时间，为了兼容spark、hive等引擎，读取结果会使用数据库时区进行修正。 set behavior_compat_options = ''; set time zone 'Asia/Shanghai'; select * from test_parquet_int96; a --------------------- 2025-01-25 08:00:00 (1 row) set time zone 'Atlantic/Cape_Verde'; select * from test_parquet_int96; a --------------------- 2025-01-24 23:00:00 (1 row)	ORA TD MySQL
enable_vacuum_update_datfrozenxid	控制在VACUUM FULL过程中是否执行更新datfrozenxid64字段的操作。 该选项9.1.0.210及以上集群版本支持。 设置此选项时需要开启autovacuum，如果autovacuum参数没有开启，即使没有添加enable_vacuum_update_datfrozenxid配置项也会执行更新datfrozenxid64字段的操作。在数据库中表较多时，设置该选项可能会拖慢VACUUM FULL操作在小表上的执行时间。 ● 设置此选项时，VACUUM FULL会执行更新datfrozenxid64字段的操作。 ● 不设置此选项时，默认VACUUM FULL不执行更新datfrozenxid64字段的操作。	ORA TD MySQL

enable_matview

参数说明：控制是否开启物化视图功能。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示物化视图功能开启。
- off表示物化视图功能关闭。

默认值：off

mv_max_cascade_depth

参数说明：设置嵌套物化视图最大深度。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：整型， 1～10

默认值：3

hive_compat_options

参数说明：用于DWS的MySQL模式下，控制INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时是否将空串转为NULL。该参数仅9.1.0.210及以上集群版本支持。

设置示例，可参见《SQL语法参考》的“ INSERT”章节。

参数类型：USERSET

取值范围：枚举型

- empty_str_to_null，表示INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时，会将空串转为NULL。
- 空，表示INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时，对空串不做任何处理。

默认值：空

19.17 容错性

当数据库系统发生错误时，以下参数控制服务器处理错误的方式。

exit_on_error

参数说明：控制终止会话。

参数类型：SUSET

取值范围：布尔型

- on表示任何错误都会终止当前的会话。
- off表示只有FATAL级别的错误才会终止会话。

默认值：off

 版权所有 © 华为云计算技术有限公司

omit_encoding_error

参数说明：数据库进行字符编码转换，在出现字符编码错误时，若目标端字符集编码为UTF-8，可将有转换错误的被转换字符忽略，并以"?"代替。

参数类型：USERSET

取值范围：布尔型

- on表示有转换错误的字符将被忽略，以"?"代替，打印错误信息到日志中。
- off表示有转换错误的字符不能被转换，打印错误信息到终端。

默认值：off

max_query_retry_times

参数说明：指定SQL语句出错自动重试功能的最大重跑次数，目前支持重跑的错误类型为“Connection reset by peer”、“Lock wait timeout”和“Connection timed out”等，设定为0时关闭重跑功能。

参数类型：USERSET

取值范围：整型，0~20

默认值：6

retry_ecode_list

参数说明：指定SQL语句出错自动重试功能支持的错误类型列表。

参数类型：USERSET

取值范围：字符串

默认值：YY001 YY002 YY003 YY004 YY005 YY006 YY007 YY008 YY009 YY010 YY011 YY012 YY013 YY014 YY015 53200 08006 08000 57P01 XX003 XX009 YY016 CG003 CG004 F0011 F0012 45003 42P30

19.18 连接池参数

当使用连接池访问数据库时，在系统运行过程中，数据库连接是被当作对象存储在内存中的，当用户需要访问数据库时，并非建立一个新的连接，而是从连接池中取出一个已建立的空闲连接来使用。用户使用完毕后，数据库并非将连接关闭，而是将连接放回连接池中，以供下一个请求访问使用。

max_pool_size

参数说明：CN的连接池与其它某个CN/DN的最大连接数。

参数类型：POSTMASTER

取值范围：整型，1~65535

默认值：CN为800， DN为5000

 版权所有 © 华为云计算技术有限公司

persistent_datanode_connections

参数说明：会话是否会释放获得的连接。

参数类型：USERSET

取值范围：布尔型

- off表示会释放获得连接。
- on表示不会释放获得连接。

须知

打开此开关后，会存在会话持有连接但并未运行查询的情况，导致其他查询申请不到连接报错。出现此问题时，需约束会话数量小于等于max_active_statements。

默认值：off

cache_connection

参数说明：是否回收连接池的连接。

参数类型：USERSET

取值范围：布尔型

- on表示回收连接池的连接。
- off表示不回收连接池的连接。

默认值：on

enable_force_reuse_connections

参数说明：会话是否强制重用新的连接。

参数类型：USERSET

取值范围：布尔型

- on表示强制使用新连接。
- off表示使用现有连接。

默认值：off

syscache_clean_policy

参数说明：设置DN空闲连接内存及数量清理策略。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：字符串

该参数策略由以下三个值组成：

 版权所有 © 华为云计算技术有限公司

1. 第一个值的取值为0~1，表示DN已使用内存占总可用内存比例。如果已使用内存占用达到该值时，清理1/4的stream线程，继续执行第二个取值的判断。
2. 第二个值的取值为0~1，表示DN上syscache内存占总可用内存比例。如果syscache内存占用达到该值时，继续执行第三个取值的判断。
3. 第三个值的取值为0~INT_MAX，单位为MB。表示空闲线程的syscache内存占用大小。如果空闲线程syscache内存占用达到该值时，则清理该线程占用的syscache。

默认值：0.8,0.3,64

须知

- 设置该参数前，需使用PV_SESSION_MEMORY_DETAIL视图及PV_TOTAL_MEMORY_DETAIL视图查看内存占用情况，进行评估。
- 设置该参数值时，需遵循设置格式，确保三个值之间以英文逗号分隔，且中间没有空格。
- 如果没有遵循规定的参数值设置格式导致设置失败，会在日志中产生WARNING类型日志，且使用SHOW命令查询该参数时，显示的为上一次设置成功的参数值；如果设置失败重启，该参数会按照默认值进行设置。
- CN上处于Readcommand阶段的线程，超时30s之后，判断syscache大于256MB清理DN。有以下两个操作：
  1. 由辅助线程监控内存占用，如果整体内存占用到80%时，清理1/4的stream线程，并判断syscache占用是否超过总占用内存的30%，如果大于30%，则清理大于64MB的Readcommand阶段pg线程的syscache。
  2. stream线程在空闲大于30s，且syscache占用大于64MB，则清理syscache。

19.19 集群事务

介绍集群事务隔离、事务只读、最大prepared事务数、集群维护模式目的参数设置及取值范围等内容。

transaction_isolation

参数说明：设置当前事务的隔离级别。

参数类型：USERSET

取值范围：

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
- read uncommitted：读未提交隔离级别，DWS不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
- serializable：事务可序列化，DWS不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

 版权所有 © 华为云计算技术有限公司

transaction_read_only

参数说明：设置当前事务是只读事务。

参数类型：USERSET

取值范围：布尔型

- on表示设置当前事务为只读事务。
- off表示该事务可以是非只读事务。

默认值：CN节点为off，DN节点为on。

xc_maintenance_mode

参数说明：设置系统进入维护模式。

参数类型：SUSET

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

默认值：off

须知

谨慎打开这个开关，避免引起集群数据不一致。

allow_concurrent_tuple_update

参数说明：设置是否允许并发更新。

参数类型：USERSET

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

默认值：on

gtm_backup_barrier

参数说明：指定是否为GTM启动点创建还原点。

参数类型：SUSET

取值范围：布尔型

- on表示创建还原点。
- off表示不创建还原点。

默认值：off

 版权所有 © 华为云计算技术有限公司

transaction_deferrable

参数说明：指定是否允许一个只读串行事务延迟执行，使其不会执行失败。该参数设置为on时，当一个只读事务发现读取的元组正在被其他事务修改，则延迟该只读事务直到其他事务修改完成。目前，DWS暂时未用到这个参数。与该参数类似的还有一个**default_transaction_deferrable**，设置它来指定一个事务是否允许延迟。

参数类型：USERSET

取值范围：布尔型

- on表示允许执行。
- off表示不允许执行。

默认值：off

enforce_two_phase_commit

参数说明：为了兼容历史版本功能保留该参数，当前版本设置无效。

enable_show_any_tuples

参数说明：该参数只有在只读事务中可用，用于分析。当这个参数被置为on/true时，表中元组的所有版本都会可见。

参数类型：USERSET

取值范围：布尔型

- on/true表示表中元组的所有版本都会可见。
- off/false表示表中元组的所有版本都不可见。

默认值：off

idle_in_transaction_timeout

参数说明: 设置允许事务处于idle空闲状态的时间。当事务处于idle状态超过该时间后，会终止本事务。功能仅对直连CN的客户端连接生效，对直连DN或内部连接不生效。该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：0 ~ 86400，单位为秒。

默认值：3600

19.20 开发人员选项

enable_light_colupdate

参数说明：控制是否使用列存轻量化UPDATE。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示开启列存轻量化UPDATE。
- off表示关闭列存轻量化UPDATE。

默认值：off

说明

列存轻量化UPDATE与后台列存AUTOVACUUM并发会小概率报错，可以通过ALTER TABLE设置表级参数enable_column_autovacuum_garbage为off来避免。需要注意的是设置表级参数enable_column_autovacuum_garbage为off会关闭该表的后台列存AUTOVACUUM。

enable_fast_query_shipping

参数说明：控制查询优化器是否使用分布式框架。

参数类型：USERSET

取值范围：布尔型

- on表示执行计划在CN和DN上各自生成。
- off表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。

默认值：on

enable_trigger_shipping

参数说明：控制触发器场景是否允许将触发器下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许将触发器下推到DN执行。
- off表示不允许将触发器下推到DN执行，在CN执行。

默认值：on

enable_remotejoin

参数说明：设置是否允许连接操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许连接操作计划下推到DN执行。
- off表示不允许连接操作计划下推到DN执行。

默认值：on

enable_remotegroup

参数说明：设置是否允许group by与aggregates执行计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示允许group by与aggregates执行计划下推到DN执行。
- off表示不允许group by与aggregates执行计划下推到DN执行。

默认值：on

enable_remotelimit

参数说明：设置是否允许LIMIT子句执行计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许LIMIT子句执行计划下推到DN执行。
- off表示不允许LIMIT子句执行计划下推到DN执行。

默认值：on

enable_limit_stop

参数说明：控制LIMIT语句是否启用early stop优化。对于LIMIT n语句，若使用early stop优化则CN收取到n条数据后会传递请求使DN提前结束执行，适用于复杂查询且带有LIMIT的场景。该参数仅8.1.3.320及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许LIMIT语句使用early stop优化。
- off表示不允许LIMIT语句使用early stop优化。

默认值：on

enable_remotesort

参数说明：设置是否允许ORDER BY子句操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许ORDER BY子句操作计划下推到DN执行。
- off表示不允许ORDER BY子句操作计划下推到DN执行。

默认值：on

enable_join_pseudoconst

参数说明：设置是否允许与伪常数进行join。伪常数是指join两侧的变量都等于同一个常量。

参数类型：USERSET

取值范围：布尔型

- on表示允许与伪常数进行join。

 版权所有 © 华为云计算技术有限公司

- off表示不允许与伪常数进行join。

默认值：off

cost_model_version

参数说明：控制应用场景中估算时cost使用的模型。该参数的影响范围主要涵盖：表达式distinct估算、HashJoin代价模型、行数估算、重分布时分布键的选择及Aggregate的行数估算等。

参数类型：USERSET

取值范围：0、1、2、3、4

- 0表示使用原始的cost估算模型。
- 1表示在0的基础上，使用增强的表达式distinct估算、HashJoin代价模型、行数估算、重分布时分布键的选择及Aggregate的行数估算。
- 2表示在1的基础上，使用随机性更优的analyze采样算法，以提高统计信息准确性。
- 3表示在2的基础上，优化大集群场景下的broadcast代价估算，以便优化器选择更优计划。该选项仅8.3.0及以上集群版本支持。
- 4表示在3的基础上，优化了hashjoin并行化代价、倾斜代价、列存索引有序性代价的代价估算以及coalesce表达式的行数估算，并支持子查询常量输出列进行join时的倾斜优化识别。

默认值：4

debug_assertions

参数说明：控制打开各种断言检查。能够协助调试，当遇到奇怪的问题或者崩溃，请把此参数打开，因为它能暴露编程的错误。要使用这个参数，必须在编译DWS的时候定义宏USE_ASSERT_CHECKING（通过configure选项 --enable-cassert完成）。

参数类型：USERSET

取值范围：布尔型

- on表示打开断言检查。
- off表示不打开断言检查。

说明

当启用断言选项编译DWS时，debug_assertions缺省值为on 。

默认值：off

distribute_test_param

参数说明：控制分布式测试框架打桩点是否生效。通常开发人员在进行故障注入测试时会在代码中预埋一些打桩点，使用唯一的名称进行标识，使用此参数可以控制代码中预埋的打桩点是否生效。参数采用逗号分隔的三元组形式，分别指定线程级别、测试桩名称和注入故障的错误级别。

参数类型：USERSET

取值范围：字符串，任一个已预埋的测试桩名称。

 版权所有 © 华为云计算技术有限公司

默认值：-1, default, default

ignore_checksum_failure

参数说明： 设置读取数据时是否忽略校验信息检查失败（但仍然会告警），继续执行。该参数仅在enable_crc_check为on时有效。继续执行可能导致崩溃，传播或隐藏损坏数据，无法从远程节点恢复数据及其他严重问题。不建议用户修改设置。

参数类型：SUSET

取值范围：布尔型

- on表示忽略数据校验错误。
- off表示数据校验错误正常报错。

默认值：off

default_table_behavior

参数说明：支持设置默认表的行为类型，该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：column_btree_index、column_high_compress、column_middle_compress、column_low_compress

- column_btree_index，表示列存表默认创建索引为B-Tree。
- column_high_compress，表示列存表默认压缩级别为high。
- column_middle_compress，表示列存表默认压缩级别为middle。
- column_low_compress，表示列存表默认压缩级别为low。

默认值：空字符串

enable_colstore

参数说明：创建表时，当不指定存储方式时，默认创建为列存表。使用时，各节点值需要保持一致。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

默认值：off

enable_force_vector_engine

参数说明：对于支持向量化的执行器算子，如果其子节点是非向量化的算子，通过设置此参数为on，强制生成向量化的执行计划。当打开enable_force_vector_engine开关时，无论是行存表、列存表或者是行列混存，如果plantree中不包含不支持向量化的场景，则强制走向量化执行引擎。

参数类型：USERSET

取值范围：布尔型

默认值：off

 版权所有 © 华为云计算技术有限公司

enable_csqual_pushdown

参数说明：进行查询时，是否要将过滤条件下推，进行Rough Check。

参数类型：USERSET

取值范围：布尔型

- on表示进行查询时，要将过滤条件下推，进行Rough Check。
- off表示进行查询时，不要将过滤条件下推，进行Rough Check。

默认值：on

explain_dna_file

参数说明：指定**explain_perf_mode**为run，导出的csv信息的目标文件。

参数类型：USERSET

须知

这个参数的取值必须是绝对路径加上.csv格式的文件名。

取值范围：字符串

默认值：NULL

explain_perf_mode

参数说明：此参数用来指定explain的显示格式。

参数类型：USERSET

取值范围：normal、pretty、summary、run

- normal：代表使用默认的打印格式。
- pretty：代表使用DWS改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。
- summary：是在pretty的基础上增加了对打印信息的分析。
- run：在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

默认值：pretty

join_num_distinct

参数说明：控制应用场景中Join列或表达式的默认distinct值。

参数类型：USERSET

取值范围：双精度浮点型，大于或等于-100，客户端显示小数时可能会有截断。

- 值大于0时，表示使用该值作为默认distinct值。
- 值大于等于-100且小于0时，表示估算默认distinct时使用的百分比。

 版权所有 © 华为云计算技术有限公司

- 值为0时，表示使用200作为默认distinct值。

默认值：-20

outer_join_max_rows_multipler

参数说明：控制应用场景中外连接估算行数的最大值。

参数类型：USERSET

取值范围：双精度浮点型，0或大于等于1，客户端显示小数时可能会有截断。

- 值为0时，表示不限制outer join的估算行数。
- 值大于等于1时，表示限制估算行数不超过外连接中外表行数的倍数。

默认值：1.1

qual_num_distinct

参数说明：控制应用场景中过滤列或表达式的默认distinct值。

参数类型：USERSET

取值范围：双精度浮点型，大于或等于-100，客户端显示小数时可能会有截断。

- 值大于0时，表示使用该值作为默认distinct值。
- 值大于等于-100且小于0时，表示估算默认distinct时使用的百分比。
- 值为0时，表示使用200作为默认distinct值。

默认值：200

trace_notify

参数说明：为LISTEN和NOTIFY命令生成大量调试输出。**client_min_messages**或**log_min_messages**级别必须是DEBUG1或者更低时，才能把这些输出分别发送到客户端或者服务器日志。

参数类型：USERSET

取值范围：布尔型

- on表示打开输出功能。
- off表示关闭输出功能。

默认值：off

trace_sort

参数说明：控制是否在日志中打印排序操作中的资源使用相关信息。这个选项只有在编译DWS的时候定义了TRACE_SORT宏的时候才可用，不过目前TRACE_SORT是由缺省定义的。

参数类型：USERSET

取值范围：布尔型

- on表示打开控制功能。

 版权所有 © 华为云计算技术有限公司

- off表示关闭控制功能。

默认值：off

zero_damaged_pages

参数说明：控制检测导致DWS报告错误的损坏的页头，中止当前事务。

参数类型：SUSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

说明

- 设置为on时，系统报告一个警告，把损坏的页面填充为零然后继续处理。该行为会破坏数据，即被损坏页面上的所有行。但是它允许绕开损坏页面然后从表中存在的未损坏页面上继续检索数据行。因此该参数在硬件或者软件错误导致的数据损坏中进行恢复是有作用的。通常不建议该参数设置为on，除非不需要从损坏的页面中恢复数据。
- 对于列存表，会将整个CU跳过然后继续处理。支持的场景包括crc校验失败、magic校验失败以及读取的CU长度错误。

默认值：off

replication_test

参数说明：此参数用于数据复制的内部功能测试。

参数类型：USERSET

取值范围：布尔型

- on表示启用功能测试。
- off表示关闭功能测试。

默认值：off

cost_param

参数说明：该参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

当cost_param & 1 不为0，表示对于求不等值连接选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确，V300R002C00版本开始，已弃用cost_param & 1 不为0时的路径，默认选择更优的估算公式；

当cost_param & 2 不为0，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确；

当cost_param & 4 不为0，表示在进行stream节点估算时，选用调试模型，该模型不推荐用户使用。

 版权所有 © 华为云计算技术有限公司

当cost_param & 16不为0，表示在计算两个及以上过滤条件或Join条件的综合选择率时，采用介于完全相关和完全不相关之间的一种模型，过滤条件较多时倾向于相关性较强的模型。

参数类型：USERSET

取值范围：整型，1～INT_MAX

默认值：16

convert_string_to_digit

参数说明：设置隐式转换优先级，是否优先将字符串转为数字。

参数类型：USERSET

取值范围：布尔型

- on表示优先将字符串转为数字。
- off表示不优先将字符串转为数字。

默认值：on

须知

请谨慎调整该参数，调整该参数会修改内部数据类型转换规则并可能导致不可预期的行为。

nls_timestamp_format

参数说明：设置时间戳默认格式。

参数类型：USERSET

取值范围：字符串

默认值：DD-Mon-YYYY HH:MI:SS.FF AM

enable_partitionwise

参数说明：分区表连接操作是否选择智能算法。

参数类型：USERSET

取值范围：布尔型

- on表示选择智能算法。
- off表示不选择智能算法。

默认值：off

enable_partition_dynamic_pruning

参数说明：分区表扫描是否支持动态剪枝。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示分区表扫描开启动态剪枝。
- off表示分区表扫描关闭动态剪枝。

默认值：on

max_user_defined_exception

参数说明：异常最大个数，默认值不可更改。

参数类型：USERSET

取值范围：整型

默认值：1000

datanode_strong_sync

参数说明：目前该参数已废弃，无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示启用stream节点间强同步。
- off表示不启用stream节点间强同步。

默认值：off

enable_global_stats

参数说明：标识当前统计信息模式，为便于进行全局统计信息生成计划和单DN统计信息计划对比使用，默认创建为全局统计信息模式。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

- on/true表示全局统计信息。
- off/false表示单DN统计信息。

默认值：on

enable_fast_numeric

参数说明：标识是否开启Numeric类型数据运算优化。Numeric数据运算是较为耗时的操作之一，通过将Numeric转化为int64/int128类型，提高Numeric运算的性能。

参数类型：USERSET

取值范围：布尔型

- on/true表示开启Numeric优化。
- off/false表示关闭Numeric优化。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_row_fast_numeric

参数说明：标识行存表numeric数据落盘的格式。

参数类型：USERSET

取值范围：布尔型

- on/true表示行存表numeric落盘格式为bigint格式。
- off/false表示行存表numeric落盘格式为原始格式。

须知

参数值设置为on时，建议同步打开enable_force_vector_engine, 可提升大数据集query的查询性能。但相比于原始格式，大概率会占用更多磁盘空间。以TPC-H测试集为例，大约多占7%空间（不同环境参考值可能有差异）。

默认值：off

rewrite_rule

参数说明：标识开启的可选查询重写规则。有部分查询重写规则是可选的，开启它们并不能总是对查询效率有提升效果。在特定的客户场景中，通过此GUC参数对查询重写规则进行设置，使得查询效率最优。

此参数可以控制查询重写规则的组合，比如有多个重写规则：rule1、rule2、rule3、rule4。可以设置：

```
set rewrite_rule=rule1;          --启用查询重写规则rule1
set rewrite_rule=rule2,rule3;    --启用查询重写规则rule2和rule3
set rewrite_rule=none;           --关闭所有可选查询重写规则
```

参数类型：USERSET

取值范围：字符串

- none：不使用任何可选查询重写规则。
- lazyagg：使用Lazy Agg查询重写规则（消除子查询中的聚集运算）。
- magicset：使用Magic Set查询重写规则（从主查询中下推条件到提升的子链接）。
- uniquecheck：使用Unique Check重写规则（允许目标列不含聚集函数的表达式子链接场景提升，需在子链接按关联列聚集后目标列值唯一才能开启，建议专业调优人员使用）。
- disablerep：使用禁止复制表的子链接提升规则（针对复制表禁止子链接提升）。
- projection_pushdown：使用Projection Pushdown重写规则（子查询中消除父查询不使用的列）。
- or_conversion：使用OR转换重写规则（消除执行效率低下的关联OR条件）。
- plain_lazyagg: 使用Plain Lazy Agg查询重写规则（消除单子查询中的聚集操作）。该选项仅8.1.3.100及以上集群版本支持。
- eager_magicset: 使用eager_magicset查询重写规则（从主查询中下推条件到子查询）。该选项仅8.2.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

- casewhen_simplification：使用casewhen语句改写的重写规则，开启后对(case when xxx then const1 else const2)=const1场景进行改写，用于简化CASE WHEN语句，减少冗余计算从而提升查询性能。该选项仅8.3.0及以上集群版本支持。
- outer_join_quality_imply：left outer join和right outer join存在等值关联条件时，外表关联列上的表达式条件下推到内表的关联列上。该选项仅8.3.0及以上集群版本支持。
- inlist_merge：使用inlist_or_inlist查询重写规则，支持基表相同列的or语句合并，开启后对(where a in (list1) or a in (list2))进行合并改写，合并后可支持inlist2join。该选项仅8.3.0及以上集群版本支持。
- subquery_qual_pull_up：对于无法提升的子查询，若子查询与其它表进行关联的关联列在子查询内部有过滤条件，支持从子查询中提取过滤条件并传递到关联条件另一侧。目前只支持不存在类型转换的var op const形式，例如a > 2。打开该开关时，则认为outer_join_quality_imply也是打开的。该选项仅9.1.0及以上集群版本支持。
- not_distinct_from_opt：开启后，将is not distinct from表达式通过临时改写为等号表达式的方式，使包含该语法的语句能够生成hashjoin或mergejoin计划，从而提升执行性能。该选项仅9.1.0.218及以上集群版本支持。

默认值：magicset, or_conversion, projection_pushdown, plain_lazyagg, subquery_qual_pull_up

mv_rewrite_rule

参数说明：标识开启的可选物化视图查询重写规则。

参数类型：USERSET

取值范围：字符串

- none，不使用任何物化视图重写规则。该取值仅8.2.1.100及以上集群版本支持。
- text，使用文本匹配的物化视图重写规则。该取值仅8.2.1.100及以上集群版本支持。
- general，使用结构匹配。该取值仅9.1.0.200及以上集群版本支持。
- predicate，结构匹配重写使用表达式框架，用于过滤条件和关联条件的预处理和正则化。仅在general开启时生效。该取值仅9.1.0.210及以上集群版本支持。
- view_delta，通过主外键关系消除物化视图中的冗余join后与查询进行匹配。该取值仅9.1.0.210及以上集群版本支持。
- single，单表物化视图使用物化视图重写规则。该取值仅9.1.0.222及以上集群版本支持。

默认值：text, general, predicate, single

须知

general、predicate和view_delta功能受限商用，如需使用请联系技术支持。

enable_compress_spill

参数说明：标识是否开启下盘压缩功能。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on/true表示开启下盘优化。
- off/false表示关闭下盘优化。

默认值：on

analysis_options

参数说明：设置是否开启对应的功能选项使用其定位功能，包括数据校验，性能统计等。

参数类型：USERSET

取值范围：字符串

- LLVM_COMPILE，表示在explain performance显示界面中显示每个线程的codegen编译时间。
- HASH_CONFLICT，表示在DN进程的pg_log目录中的log日志中显示hash表的统计信息，包括hash表大小、hash链长及hash冲突情况。
- STREAM_DATA_CHECK，表示对网络传输前后的数据进行CRC校验。
- TURBO_DATA_CHECK，表示对turbo的ScalarVector和VectorBatch的算子的数据上下文进行校验。该选项仅8.3.0.100及以上集群支持。
- KEEP_SAMPLE_DATA，表示以临时表的形式保留每次analyze操作中使用的采样数据。该选项仅9.1.0及以上集群版本支持。
- BLOCK_RULE，表示在explain performance显示界面中显示查询过滤器检查耗时。该选项仅9.1.0.100及以上集群版本支持。

默认值：off(ALL)，不开启任何定位功能。

resource_track_log

参数说明：控制自诊断的日志级别。目前仅对多列统计信息进行控制。

参数类型：USERSET

取值范围：字符串

- summary：显示简略的诊断信息。
- detail：显示详细的诊断信息。

目前这两个参数值只在显示多列统计信息未收集的告警的情况下有差别，summary不显示未收集多列统计信息的告警，detail会显示这类告警。

默认值：summary

hll_default_log2m

参数说明：该参数可以指定hll数据结构桶的个数。桶的个数会影响hll计算distinct值的精度，桶的个数越多，误差越小。误差范围为：$[-1.04/2^{log2m*1/2},+1.04/2^{log2m*1/2}]$。

参数类型：USERSET

取值范围：整型，10~16。

 版权所有 © 华为云计算技术有限公司

默认值：11

hll_default_regwidth

参数说明：该参数可以指定hll数据结构每个桶的位数，该值越大，hll所占内存越高。hll_default_regwidth和hll_default_log2m可以决定当前hll能够计算的最大distinct value。具体对应关系可以参见**表1 hll_default_log2m和hll_default_regwidth与当前能计算的最大distinct value值的关系**。

参数类型：USERSET

取值范围：整型，1~5。

默认值：5

表 19-3 hll_default_log2m 和 hll_default_regwidth 与当前能计算的最大 distinct value 值的关系

log2m	regwidth = 1	regwidth = 2	regwidth = 3	regwidth = 4	regwidth = 5
10	7.4e+02	3.0e+03	4.7e+04	1.2e+07	7.9e+11
11	1.5e+03	5.9e+03	9.5e+04	2.4e+07	1.6e+12
12	3.0e+03	1.2e+04	1.9e+05	4.8e+07	3.2e+12
13	5.9e+03	2.4e+04	3.8e+05	9.7e+07	6.3e+12
14	1.2e+04	4.7e+04	7.6e+05	1.9e+08	1.3e+13
15	2.4e+04	9.5e+04	1.5e+06	3.9e+08	2.5e+13

hll_default_expthresh

参数说明：该参数可以用来设置从Explicit模式到Sparse模式的默认阈值大小。

参数类型：USERSET

取值范围：整型，-1~7。-1表示自动模式，0表示跳过Explicit模式，取1-7表示在基数到达$2^{hll_default_expthresh}$时切换模式。

默认值：-1

hll_default_sparseon

参数说明：该参数可用来指定是否默认开启Sparse模式。

参数类型：USERSET

取值范围：0，1。0表示默认关闭，1表示默认开启。

默认值：1

hll_max_sparse

参数说明：该参数可以用来指定max_sparse的大小。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，-1~INT_MAX

默认值：-1

enable_compress_hll

参数说明：该参数可以用来指定是否对hll开启内存优化模式。

参数类型：USERSET

取值范围：布尔型

- on/true表示对hll开启内存优化模式。
- off/false表示不开启内存优化模式。

默认值：off

approx_count_distinct_precision

参数说明：该参数表示HyperLogLog++ (HLL++)算法中分桶个数，可以用来调整approx_count_distinct聚集函数的误差率。桶的个数会影响distinct值估算的精度，桶的个数越多，误差越小。误差范围为：$[-1.04/2^{log2m*1/2},+1.04/2^{log2m*1/2}]$。

参数类型：USERSET

取值范围：整型，10~20。

默认值：17

udf_memory_limit

参数说明：控制每个CN、DN执行UDF时可用的最大物理内存量。

参数类型：POSTMASTER

取值范围：整型，200*1024～**max_process_memory**，单位为KB。

默认值：0.05 * max_process_memory

FencedUDFMemoryLimit

参数说明：控制每个fenced udf worker进程使用的虚拟内存。

参数类型：USERSET

设置建议：不建议设置此参数，可用**udf_memory_limit**代替。

取值范围：整数，可带单位（KB，MB，GB）。其中0表示不做内存控制。

默认值：0

enable_pbe_optimization

参数说明：设置优化器是否对以PBE（Parse Bind Execute）形式执行的语句进行查询计划的优化。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型。

- on表示优化器将优化PBE语句的查询计划。
- off表示不使用优化。

默认值：on

enable_light_proxy

参数说明：设置优化器是否对简单查询在CN上优化执行。

参数类型：USERSET

取值范围：布尔型。

- on表示优化器将优化CN上简单查询的执行。
- off表示不使用优化。

默认值：on

enable_parallel_ddl

参数说明：控制多CN对同一数据库对象是否能安全的并发执行DDL操作。

参数类型：USERSET

取值范围：布尔型

- on表示可以安全的并发执行DDL操作，不会出现分布式死锁。
- off表示不能安全的并发执行DDL操作，可能会出现分布式死锁。

默认值：on

gc_fdw_verify_option

参数说明：在协同分析特性中，控制是否启用结果集行数校验逻辑。该参数仅8.1.3.310及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启结果集行数校验逻辑，额外发送select count语句获取预期行数，与实际获取结果集进行比对。
- off表示关闭结果集行数校验逻辑，仅获取所需结果集。

默认值：on

说明

- 该参数开启时性能会有轻微劣化，性能敏感场景可通过关闭该参数提升性能。
- 结果集行数校验失败时会上报异常，可通过设置参数log_min_messages=debug1和logging_module='on(COOP_ANALYZE)'来打开协同分析日志。

show_acce_estimate_detail

参数说明：在DWS集群使用加速集群场景下（即**acceleration_with_compute_pool**设置为on），控制explain命令是否显示用于评估执行计划下推到加速集群的评估信

 版权所有 © 华为云计算技术有限公司

息。评估信息一般用于运维人员在维护工作中使用，因此该参数默认关闭，此外为了避免这些信息干扰正常的explain信息显示，只有在explain命令的verbose选项打开的情况下才显示评估信息。

参数类型：USERSET

取值范围：布尔型

- on表示可以在explain命令的输出中显示评估信息。
- off表示不在explain命令的输出中显示评估信息。

默认值：off

full_group_by_mode

参数说明：结合**behavior_compat_options**中disable_full_group_by_mysql使用，用于控制disable_full_group_by_mysql语法开关打开后的两种不同行为。

参数类型：USERSET

取值范围：字符串

- nullpadding表示对于非聚集列而言，对该列NULL值进行填充，取该列非NULL值，结果集可能为不同行。
- notpadding表示对于非聚集列而言，不处理NULL值，取该行整行数据，非聚集列结果集为随机的一行。

默认值：notpadding

须知

该参数生效前提为MySQL兼容库下打开disable_full_group_by_mysql，且查询中出现非聚集列场景生效。该参数的两种行为也仅针对查询中的非聚集列生效。

enable_cudesc_streaming

参数说明：在存算分离架构下，控制跨逻辑集群访问是否选择cudesc streaming路径（从表所在逻辑集群获取cudesc、delta表数据等信息）。该参数仅9.1.0及以上集群版本支持。

参数类型：SUSET

取值范围：枚举型

- off表示关闭cudesc streaming。
- on表示开启cudesc streaming。
- only_read_on表示只在数据读取时支持cudesc streaming。

默认值： on

force_read_from_rw

参数说明：在存算分离架构下，强制从其他逻辑集群上读取数据（从表所在逻辑集群上读取数据）。该参数仅9.0.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

默认值：off

kv_sync_up_timeout

参数说明：在存算分离架构下，设置KV同步等待超时时间。该参数仅9.0.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~2147483647

默认值： 10min

enable_insert_foreign_table_dop

参数说明：控制OBS外表写出时是否启用dop加速。每个DN的dop线程数量由query_dop决定，可以设置query_dop控制并行度。该参数仅9.1.0.200及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用外表dop加速。
- off表示禁用外表dop加速。

默认值：off

enable_insert_foreign_table_dop_opt

参数说明：控制外表insert dop启用后，是否启用分区重分布优化。导出分区数较多时（导出分区数大于10倍分区数），建议开启，可以减少单分区内的小文件同时提高导出性能。该参数仅9.1.0.200及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用分区外表insert dop重分布优化。
- off表示禁用分区外表insert dop重分布优化。

默认值：off

enable_hstore_binlog_table

参数说明：用于控制是否可以创建binlog表。

参数类型：SIGHUP

取值范围：布尔型

- on表示可以创建binlog表。
- off表示不可以创建binlog表。

 版权所有 © 华为云计算技术有限公司

默认值：off

enable_generate_binlog

参数说明：用于控制当前会话的binlog表上的DML操作是否产生binlog。该参数仅9.1.0.200以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示产生binlog。
- off表示不产生binlog。

默认值：on

binlog_consume_timeout

参数说明：用于控制在线扩缩容binlog表或者vacuum full binlog表时，循环判断binlog记录是否都被消费的超时时间。该参数仅8.3.0.100及以上版本支持。单位秒。

参数类型：SIGHUP

取值范围：整型，0~86400

默认值： 3600

19.21 审计

19.21.1 审计开关

audit_enabled

参数说明：控制审计进程的开启和关闭。审计进程开启后，将从管道读取后台进程写入的审计信息，并写入审计文件。

参数类型：SIGHUP

取值范围：布尔型

- on表示启动审计功能。
- off表示关闭审计功能。

默认值：on

audit_space_limit

参数说明：审计文件占用的磁盘空间总量。

参数类型：SIGHUP

取值范围：整型，1024~1073741824，单位为KB。

默认值：1GB

 版权所有 © 华为云计算技术有限公司

audit_object_name_format

参数说明：控制审计日志object_name字段所显示的对象名的格式。

参数类型：USERSET

取值范围：枚举类型

- single，表示object_name字段显示单个对象名，且为目标对象的名称。
- all，表示object_name字段显示多个对象名称。

默认值：single

说明

当默认值设置为all时，显示多个对象名的场景有：SELECT、DELETE、UPDATE、INSERT、MERGE、CREATE TABLE AS、CREATE VIEW AS、DROP USER ... CASCADE、DROP OWNED BY ... CASCADE、DROP SCHEMA ... CASSCADE、DROP TABLE ... CASCADE、DROP FOREIGN TABLE ... CASCADE、DROP VIEW ... CASCADE。

audit_object_details

参数说明：控制审计日志中是否记录object_details字段，该字段为审计语句中的表名、列名以及列使用的类型。该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔类型

- on表示审计记录object_details字段。
- off表示审计不记录object_details字段。

默认值：off

说明

- 当设置为on时，会对语句中的表名、列名以及列使用的类型进行审计，可能会对性能存在一定的影响，请谨慎开启。
- 当设置为on时，object_details字段记录的场景有：SELECT、DELETE、UPDATE、INSERT、MERGE、CREATE TABLE AS SELECT，GRANT、DECLARE CURSOR，其中对于执行失败的GRANT语句不予记录。

19.21.2 操作审计

security_enable_options

参数说明：控制是否允许使用grant_to_public、grant_with_grant_option和foreign_table_options这三种功能，出于安全考虑，这些操作默认禁止，用户可根据实际需求进行配置。该参数仅8.2.0及以上集群版本支持。

参数类型：SIGHUP

取值范围：字符串

- grant_to_public，表示允许使用grant to public功能。
- grant_with_grant_option，表示允许使用with grant option功能。

版权所有 © 华为云计算技术有限公司

- foreign_table_options，表示允许使用外表操作功能，不需要显式赋予用户useft权限。

默认值：空

说明

- 新安装集群场景下，该参数默认值为空，表示不允许使用grant_to_public、grant_with_grant_option和foreign_table_options中任何一种功能。
- 升级场景下，该参数的默认值保持前向兼容，若用户升级前原版本中GUC参数enable_grant_public和enable_grant_option默认为ON，那么升级后security_enable_options参数的默认值为“grant_to_public, grant_with_grant_option”。

19.22 事务监控

通过设置事务超时预警，可以监控自动回滚的事务并定位其中的语句问题，也可以监控执行时间过长的语句。

transaction_sync_naptime

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_naptime时会主动触发gs_clean进行清理，缩短不一致时的阻塞时长。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒。

默认值：5s

说明

若该值设为0，则不会在阻塞达到时长时主动调用gs_clean进行清理，而是靠gs_clean_timeout间隔来调用gs_clean，默认是5分钟。

transaction_sync_timeout

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_timeout时会报错，回滚事务，避免由于sync lock等其他情况长时间进程停止响应造成对系统的阻塞。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒。

默认值：10min

说明

- 若该值设为0，则不会在阻塞超时时报错，回滚事务。
- 该值必须大于gs_clean_timeout，避免DN上由于还未被gs_clean清理的残留事务阻塞超时引起的不必要的事务回滚。

 版权所有 © 华为云计算技术有限公司

19.23 GTM 相关参数

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

须知

当client_min_messages和log_min_messages取值相同时，其值所代表的级别不同。

参数类型：SUSET

取值范围：枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见表19-1。

默认值：warning

19.24 其它选项

server_version

参数说明：报告服务器版本号（字符串形式）。

参数类型：INTERNAL（固定参数用于查看，不建议设置该参数，错误值可能导致兼容性问题。）

取值范围：字符串

默认值：9.2.4

server_version_num

参数说明：报告服务器版本号（整数形式）。

参数类型：INTERNAL（固定参数用于查看，不建议设置该参数，错误值可能导致兼容性问题。）

取值范围：整数

默认值：90204

enable_cluster_resize

参数说明：标识当前会话是否为扩容重分布会话。该参数应仅限用于扩容重分布会话，其他业务会话不要设置该参数。

参数类型：SUSET

取值范围：布尔型

版权所有 © 华为云计算技术有限公司

- on表示标识当前会话为扩容重分布会话，可以执行重分布专有SQL语句。
- off表示标识当前会话为非扩容重分布会话，不能执行重分布专有SQL语句。

默认值：off

说明

此参数用于内部运维场景，请勿随意开启。

dfs_partition_directory_length

参数说明：在HDFS文件系统上，构造HDFS VALUE分区表的分区目录时，目录名长度的上限值。

参数类型：USERSET

取值范围：92～7999

默认值：512

enable_hadoop_env

参数说明：设置使用Hadoop特性时，是否允许在数据库中创建本地行存表和列存表。DWS集群中，集群安装好后，该参数默认设为off。以支持本地行列存储和跨集群访问Hadoop特性。不推荐用户调整enable_hadoop_env的值。

参数类型：USERSET

取值范围： 布尔型

- on/true，表示使用Hadoop特性时，不允许在数据库中创建本地行存表和列存表。
- off/false，表示使用Hadoop特性时，可以在数据库中创建本地行存表和列存表。

默认值：off

enable_upgrade_merge_lock_mode

参数说明：当该参数设置为on时，通过提升deltamerge内部实现的锁级别，避免和update/delete并发操作时的报错。

参数类型：USERSET

取值范围： 布尔型

- on，提升deltamerge内部实现的锁级别，并发执行deltamerge和update/delete操作时，一个操作先执行，另一个操作被阻塞，在前一个操作完成后，后一个操作再执行。
- off，在对HDFS表的delta table的同一行并发执行deltamerge和update/delete操作时，后一个对同一行数据更新的操作会报错退出。

默认值：off

job_queue_processes

参数说明：表示系统可以并发执行的job数目。

 版权所有 © 华为云计算技术有限公司

参数类型：POSTMASTER

取值范围：0～1000

功能：

- 当job_queue_processes设置为0值，表示不启用定时任务功能，任何job都不会被执行（因为开启定时任务的功能会对系统的性能有影响，有些局点可能不需要定时任务的功能，可以通过设置为0不启用定时任务功能）。
- 当job_queue_processes为大于0时，表示启用定时任务功能且系统能够并发处理的最大任务数。

启用定时任务功能后，job_scheduler线程会在定时时间间隔轮询pg_jobs系统表，系统设置定时任务检查周期默认为1s。

由于并行运行的任务数太多会消耗更多的系统资源，因此需要设置系统并发处理的任务数，当前并发的任务数达到job_queue_processes时，且此时又有任务到期，那么这些任务本次得不到执行而延期到下一轮询周期。因此，建议用户需要根据每个任务的执行时长合理地设置任务的时间间隔（即submit接口中的interval参数），来避免由于任务执行时间太长而导致下个轮询周期无法正常执行。

注：如果同一时间内并行的job数很多，过小的参数值会导致job等待。而过大的参数值则消耗更多的系统资源，建议设置此参数为100，用户可以根据系统资源情况合理调整。

默认值：10

ngram_gram_size

参数说明：ngram解析器分词的长度。

参数类型：USERSET

取值范围：整型 ，1～4

默认值：2

ngram_grapsymbol_ignore

参数说明：ngram解析器是否忽略图形化字符。

参数类型：USERSET

取值范围：布尔型

- on表示忽略图形化字符。
- off表示不忽略图形化字符。

默认值：off

ngram_punctuation_ignore

参数说明：ngram解析器是否忽略标点符号。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示忽略标点符号。
- off表示不忽略标点符号。

默认值：on

zhparser_multi_duality

参数说明：Zhparser解析器设定是否将长词内的文字自动以二字分词法聚合。

参数类型：USERSET

取值范围：布尔型

- on表示将长词内的文字自动以二字分词法聚合。
- off表示不将长词内的文字自动以二字分词法聚合。

默认值：off

zhparser_multi_short

参数说明：Zhparser解析器分词执行时是否执行针对长词复合切分。

参数类型：USERSET

取值范围：布尔型

- on表示执行针对长词复合切分。
- off表示不执行针对长词复合切分。

默认值：on

zhparser_multi_zall

参数说明：Zhparser解析器是否将全部单字单独显示。

参数类型：USERSET

取值范围：布尔型

- on表示将全部单字单独显示。
- off表示不将全部单字单独显示。

默认值：off

zhparser_multi_zmain

参数说明：Zhparser解析器是否将重要单字单独显示。

参数类型：USERSET

取值范围：布尔型

- on表示将重要单字单独显示。
- off表示不将重要单字单独显示。

默认值：off

 版权所有 © 华为云计算技术有限公司

zhparser_punctuation_ignore

参数说明：Zhparser解析器分词结果是否忽略所有的标点等特殊符号（不会忽略\r和\n）。

参数类型：USERSET

取值范围：布尔型

- on：忽略所有的标点等特殊符号。
- off：不忽略所有的标点等特殊符号。

默认值：on

zhparser_seg_with_duality

参数说明：Zhparser解析器是否将闲散文字自动以二字分词法聚合。

参数类型：USERSET

取值范围：布尔型

- on表示将闲散文字自动以二字分词法聚合。
- off表示不将闲散文字自动以二字分词法聚合。

默认值：off

acceleration_with_compute_pool

参数说明：在查询包含OBS时，通过该参数决定查询是否通过计算资源池进行加速。

参数类型：USERSET

取值范围：布尔型

- on表示包含有OBS的查询在计算资源池可用时，会根据代价评估决定是否通过计算资源池对查询加速。
- off表示任何查询都不会通过计算资源池进行加速。

默认值：off

redact_compat_options

参数说明：设置数据脱敏可算不可见兼容性行为配置项。该参数仅8.1.3.310及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none表示未配置兼容项。
- disable_comparison_operator_mask表示允许不存在暴露原始数据风险的比较操作符绕过脱敏检查，输出实际计算结果。
- relaxed_redact_check，表示对数据脱敏列类型校验进行弱管控，支持修改依赖脱敏的表字段列类型。该选项在9.1.0集群版本中仅9.1.0.223及以上版本支持。

默认值：none

 版权所有 © 华为云计算技术有限公司

table_skewness_warning_threshold

参数说明：设置用于表倾斜告警的阈值。

参数类型：SUSET

取值范围：浮点型，0～1

默认值：1

table_skewness_warning_rows

参数说明：设置用于表倾斜告警的行数。

参数类型：SUSET

取值范围：整型，0～INT_MAX

默认值：100000

enable_view_update

参数说明：用于设置是否开启视图更新功能。

参数类型：POSTMASTER

取值范围：布尔型

- on表示启用视图更新功能。
- off表示关闭视图更新功能。

默认值：off

view_independent

参数说明：用于设置是否开启视图与表、函数、同义词的解耦功能。基表恢复后目前已支持自动关联重建。

参数类型：SIGHUP

取值范围：布尔型

- on表示启用视图解耦功能，存在视图依赖的表、函数、同义词及其他视图可以单独删除（临时表及临时视图除外），关联视图保留但不可用。
- off表示关闭视图解耦功能，存在视图依赖的表、函数、同义词及其他视图不可以单独删除，仅可使用cascade级联删除。

默认值：off

assign_abort_xid

参数说明：查询时将指定的xid判断为需要abort的事务。

参数类型：USERSET

取值范围：指定xid的字符串

 版权所有 © 华为云计算技术有限公司

⚠ 注意

此参数只用于用户误删数据(delete操作)后进行快速恢复。其他场景禁止使用，否则造成事务可见性错误问题。

default_distribution_mode

参数说明：用于设置表的默认分布方式。该参数仅8.1.2及以上版本支持。

参数类型：USERSET

取值范围：枚举类型

- roundrobin，创建表不指定分布方式时，按如下规则选取默认分布方式：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，则选取ROUNDROBIN分布。
- hash，创建表不指定分布方式时，按如下规则选取默认分布方式：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，但存在数据类型支持作分布列的列，则选取HASH分布，分布列为第一个数据类型支持作分布列的列。
  c. 若建表时不包含主键/唯一约束，也不存在数据类型支持作分布列的列，选取ROUNDROBIN分布。

默认值：roundrobin

说明

新建8.1.2集群版本默认值为roundrobin，升级到8.1.2集群版本场景该参数的默认值为hash。

object_mtime_record_mode

参数说明：用于设置PG_OBJECT系统表中mtime字段的更新行为。

参数类型：SIGHUP

取值范围：字符串

- default，表示默认行为包括ALTER、COMMENT、GRANT/REVOKE和TRUNCATE操作会更新mtime字段。
- disable，表示不更新mtime字段。
- disable_acl，表示GRANT/REVOKE操作不更新mtime字段。
- disable_truncate，表示TRUNCATE操作不更新mtime字段。
- disable_partition，表示分区表相关ALTER操作不更新mtime字段。

默认值：default

max_volatile_tables

参数说明：指定每个session创建的volatile临时表最大个数，其中包括volatile临时表自身及其附属表。该参数仅8.2.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0~INT_MAX

默认值：300

query_cache_refresh_time

参数说明：用于控制enable_accelerate_select生效的查询的缓存刷新时间。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0~10000.0，单位为秒。

默认值：60.0

vector_engine_strategy

参数说明：用于控制向量化增强策略。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- force表示向量化增强策略设置为存在不支持向量化场景时，强制回退为行存计划。
- improve表示向量化增强策略设置为存在不支持向量化场景时，启用向量化增强，尽可能将计划向量化。

默认值：improve

default_temptable_type

参数说明：用于控制CREATE TABLE创建临时表时在TEMP或TEMPORARY前未指定表类型所创建临时表的类型。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- local：表示在未指定类型的情况下创建local型临时表。
- volatile：表示在未指定类型的情况下创建volatile型临时表。

默认值：local

pgxc_node_readonly

参数说明：标记CN、DN实例是否为弹性或经典DN节点。该参数仅9.1.0及以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on表示标记CN、DN实例为弹性节点。
- off表示标记CN、DN实例为经典节点。

 版权所有 © 华为云计算技术有限公司

默认值：off

hudi_sync_max_commits

参数说明：设置Hudi同步任务单次同步的最大commits的数量。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，-1~INT_MAX

- -1，表示不限制。
- 0，表示不限制。
- 其他值，表示最大commits的数量。

默认值：-1

foreign_table_default_rw_options

参数说明：控制创建外表时未指定权限时的默认权限。仅9.0.3及以上版本支持。

参数类型：USERSET

取值范围：字符串

- READ_ONLY，表示只读权限。
- WRITE_ONLY，表示只写权限。
- READ_WRITE，表示读写权限。

默认值：READ_ONLY

 版权所有 © 华为云计算技术有限公司

20 DWS 开发者术语表

A~E

表 20-1 A~E 术语解释

术语	解释
ACID	在可靠数据库管理系统（DBMS）中，事务（transaction）所应该具有的四个特性：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durability）。
安全环	每一个环都由若干物理机组成，环内的DN形成主、备、从备关系，不向环外延伸。也就是说，环内的任何一个节点的主，或者备，或从备，依然在环内。扩容与缩容时，是以环为最小单位进行的。
Bgwriter	数据库启动时创建的一个后台写线程，此线程用于将数据库中脏页面写入到持久性设备（例如磁盘）中。
bit	比特。计算机处理的最小的信息单位。比特用来表示二进制数字1或0，或者一种逻辑条件真或假。在物理上，比特表示一个电路上高或低的电压点或者磁盘上的磁化单程或其它。一个单独的比特位所传达的信息很少有意义的。然而，一个8位组却构成了一个字节，可用于表示如一个英文字母，十进制数字，或其它字符等多种类型的信息。
Bloom Filter	布隆过滤器。由Howard Bloom在1970年提出的二进制向量数据结构，它具有很好的空间和时间效率，被用来检测一个元素是不是集合中的一个成员，这种检测只会对在集合内的数据错判，而不会对不是集合内的数据进行错判，这样每个检测请求返回有“在集合内（可能错误）”和“不在集合内（绝对不在集合内）”两种情况，可见Bloom filter是牺牲了正确率换取时间和空间。
CCN	Central Coordinator ，DWS动态负载管理中心协调节点。负责进行各CN中复杂作业是否可以执行的中心判断、排队和调度，以实现动态负载管理。

版权所有 © 华为云计算技术有限公司

术语	解释
CIDR	Classless Inter-Domain Routing，无类域间路由IP编址方案。CIDR摒弃传统的基于类（A类：8，B类：16，C类：24）的地址分配方式，允许使用任意长度的地址前缀，有效提高地址空间的利用率。CIDR表示方法：IP地址/网络ID的位数。比如192.168.23.35/21，其中“21”表示前面地址中的前21位代表网络部分，其余位代表主机部分。
Cgroups	Control Groups，控制组（DWS中也称之为优先级组）。SUSE Linux和RedHat内核提供的一种可以限制、记录、隔离进程组所使用的物理资源的机制。
CLI	Command-line Interface，命令行界面。应用程序和用户交互的一种方式，完全基于文本输入和输出。命令通过键盘或类似装置输入，由程序编译并执行。结果是以文本或图形的方式呈现在终端界面。
CM	Cluster Manager，集群管理模块。管理和监控分布式系统中各个功能单元和物理资源的运行情况，确保整个系统的稳定运行。
CMS	Cluster Management Service，集群管理服务。是用于管理集群状态的部件。
CN	Coordinator，负责数据库系统元数据存储、查询任务的分解和部分执行，以及将DN中查询结果汇聚在一起。
CTE	Common Table Expression，公共表达式是一种在SQL查询中定义的临时命名结果集，通常通过 WITH关键字创建。CTE仅存在于单个查询的执行范围内，可被后续的SELECT、INSERT、UPDATE或DELETE语句多次引用。CTE主要作用是简化复杂查询的编写，提升代码可读性和维护性，合理使用CTE，能显著提升复杂SQL的可维护性，尤其在递归或分步处理数据时优势明显。
CU	Compression Unit，压缩单元。列存表的最小存储单位。
core文件	当程序出现内存越界、断言失败或者访问非法内存时，操作系统会中止进程，并将当前内存状态导出到core文件中，以便进一步分析。 core文件包含内存转储，支持全二进制和指定端口格式。core文件名称由字符串core以及操作系统进程ID组成。 core文件不依赖于任何平台。
Core Dump	通常在程序异常终止时，核心转储（Core Dump）、内存转储或系统转储用于记录特定时间计算机程序工作内存的状态。实际上，其它关键程序的状态经常在同一时间进行转储，例如处理器寄存器，包括程序指标和栈指针、内存管理信息、其它处理器和操作系统标记及信息。Core Dump经常用于辅助诊断和纠错计算机程序问题。
DBA	Database Administrator，数据库管理员。指导或执行所有和维护数据库环境相关的操作。
DBLINK	DBLINK是定义一个数据库到另一个数据库路径的对象，通过它可以查询远程数据库对象。

版权所有 © 华为云计算技术有限公司

术语	解释
DBMS	Database Management System，数据库管理系统。数据库管理系统是为了访问数据库中的信息而使用的一个管理系统软件。它包含一组程序使用户可以进入、管理、查询数据库中数据。基于真实数据的位置，可以分为内存数据库管理系统和磁盘数据库管理系统。
DCL	Data Control Language，数据控制语言。
DDL	Data Definition Language，数据定义语言。
DFS	Distributed File System，分布式文件系统。DFS是一个通用术语，指代任何将文件存储在多个物理节点上并通过网络提供统一访问接口的系统。它不特指具体技术，而是一类系统的统称。 例如HDFS，Hadoop 分布式文件系统，它属于DFS中的一种，专为大数据场景设计，适合高吞吐量、批处理任务。
DML	Data Manipulation Language，数据操纵语言。
DN	Data Node，和CN对应的概念。负责实际执行表数据的存储、查询操作。
ETCD	Editable Text Configuration Daemon，分布式键值存储系统，用于共享配置和服务发现（服务注册和查找）。
ETL	Extract-Transform-Load，描述将数据从来源端经过抽取（extract）、转换（transform）、加载（load）至目的端的过程。
Extension Connector	Extension Connector是DWS提供的功能模块，使用它可以将SQL语句发送到集群外部的Spark，并在当前库中返回执行结果，实现跨集群处理数据。
备份	备份件或者备份过程。指复制并归档计算机数据，当发生数据丢失事件时，可以用该复制并归档的数据来恢复原始数据。
备份和恢复	保护数据库防止由于媒介失效或人为错误造成的数据丢失过程中涉及的一组概念、过程及策略。
备机	DWS双机方案中的一个节点，用于作为主机的备份，在主机异常时，备机会切换到主机状态，以确保能正常提供数据服务。
崩溃	崩溃（或系统崩溃）指计算机或程序（例如软件应用程序或操作系统）异常终止的事件。出现错误后，通常会自动退出。有时出现恶意程序冻结或挂起直到崩溃上报服务记录崩溃的详细信息。对于操作系统内核关键部分的程序，整个计算机可能瘫痪（可能造成致命的系统错误）。
编码	编码是指用代码来表示各组数据资料，使其成为可利用计算机进行处理和分析的信息。用预先规定的方法将文字、数字或其它对象编成数码，或将信息、数据转换成规定的电脉冲信号。
编码技术	呈现计算机软硬件识别的特定字符集数据的技术。

版权所有 © 华为云计算技术有限公司

术语	解释
表	表是由行与列组合成的。每一列被当作是一个字段。每个字段中的值代表一种类型的数据。例如，一个表可能有3个字段：姓名、城市和国家。这个表就会有3列：一列代表姓名，一列代表城市，一列代表国家。表中的每一行包含3个字段的内容，姓名字段包含姓名，城市字段包含城市，国家字段包含国家。
表空间	包含表、索引、大对象、长数据等数据的逻辑存储结构。表空间在物理数据和逻辑数据间提供了抽象的一层，为所有的数据库对象分配存储空间。表空间创建好后，创建数据库对象时可以指定该对象所属的表空间。
并发控制	在多用户环境下同时执行多个事务并保证数据完整性的一个DBMS服务。并发控制是DWS提供的一种多线程管理机制，用来保证多线程环境下在数据库中执行的操作是安全的和一致的。
查询	向数据库发出的信息请求，包含更新、修改、查询或删除信息的请求。
查询操作符	Query Operator，也称为查询迭代算子（Iterator）或查询节点（Query Tree Node）。一个查询的执行可以分解为一个或多个查询操作符，是构成一个查询执行的最基本单位。常见的查询操作符包括表扫描（Scan），表关联（Join），表聚集（Aggregation）等。
查询片段	每一个查询任务都可以分解成为一个或者多个查询片段。每个查询片段由一个或多个查询操作符构成，可独立在节点上运行。通过数据流操作符与其它查询片段块交换数据。
持久性	数据库事务的ACID特性之一。在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。
存储过程	存储过程（StoredProcedure）是在大型数据库系统中，一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名称并设置参数（如果该存储过程带有参数）来执行它。
操作系统	操作系统OS（operating system）由引导程序加载到计算中，对计算机中其它程序进行管理。其它程序叫做应用或应用程序。
从备	Secondary，为了保证集群的高可靠性，主、备间无法正常同步数据时，主节点会将日志同步到从备。如果主节点突然故障不可用，备节点会升主，并且升主成功后从从备节点上同步之前异常期间的日志。
大对象	大对象（Blob）在数据库中指使用二进制方式存储的数据。它通常可以用于存储视频、音频和图像等多媒体数据。
递归查询	递归查询（RECURSIVE Query）是数据库查询中的一种高级技术，主要用于处理层次结构数据或需要逐层迭代计算的场景。它通过递归调用自身来实现对树形结构、图结构或链式关系的遍历和分析。在SELECT语法中，常通过WITH RECURSIVE格式来声明递归查询，递归查询的使用示例可以参见SQL语法的“SELECT”章节。

版权所有 © 华为云计算技术有限公司

术语	解释
动态负载	DWS动态负载是指基于系统中CPU、I/O、内存等资源的使用情况，自动调节并发作业的运行数量，避免因为系统资源过载导致业务报错或无响应。
段	数据库中，一段指包含一个或多个区域的数据库中的一部分。区域是数据库的最小范围，由单元调用块组成。一个或多个段组成一个表空间。

F~J

表 20-2 F~J 术语解释

术语	解释
Failover	指当某个节点出现故障时，自动切换到备节点上的过程。反之，从备节点上切换回来的过程称为Failback。
FDW	Foreign Data Wrapper，外部数据封装器。是Postgres提供的一个SQL接口，用于访问远程数据存储中的大数据对象，使DBA可以整合来自不相关数据源的数据，将它们存入数据库中的一个公共模型。
Freeze	在事务ID耗尽时由AutoVacuum Worker进程自动执行的操作。DWS会把事务ID记在行头，在一个事务取得一行时，通过比较行头的事务ID和事务本身的ID判断这行是否可见，而事务ID是一个无符号整数，如果事务ID耗尽，事务ID会跨过整数的界限重新计算，此时原先可见的行就会变成不可见的行，为了避免这个问题，Freeze操作会将行头的事务标记为一个特殊的事务ID，标记了这个特殊的事务ID的行将对所有事务可见，以此避免事务ID耗尽产生的问题。
GDB	GNU工程调试器，可以监控其它程序运行时的内部情况，或者其它程序要崩溃时发生了什么。GDB支持如下四种主要操作（使PDK功能更加强大），辅助查找缺陷。 ● 启动程序，指定可能影响行为的任何因素。 ● 特定条件下，停止程序。 ● 程序停止时，检查发生了什么。 ● 修改程序内容，尝试纠正一个缺陷并继续下一个。
GDS	General Data Service（简称GDS），数据并行加载工具。向DWS导入数据时，需要将此工具部署到源数据所在的服务器上，使DN可以通过该工具获取数据。
GIN索引	Generalized Inverted Index，通用倒排索引。作用为处理索引项为组合值的情况，查询时需要通过索引搜索出出现在组合值中的特定元素值。

版权所有 © 华为云计算技术有限公司

术语	解释
GNU	GNU计划，又称革奴计划，是由RichardStallman在1983年9月27日公开发起的。它的目标是创建一套完全自由的操作系统。GNU是“GNU's NotUnix”的递归缩写。Stallman宣布GNU应当发音为Guh-NOO以避免与new这个单词混淆（注：Gnu在英文中原意为非洲牛羚，发音与new相同）。Unix是一种广泛使用的商业操作系统的名称。技术上讲，GNU类似Unix。但是GNU却给了用户自由。
gsql	DWS交互终端。通过gsql能够以交互的方式输入查询，下发查询到DWS，然后查看查询结果。或者，也可以从文件中输入。此外，gsql还提供许多元命令和各种类似shell命令，协助脚本编写及自动化各种任务。
GTM	Global Transaction Manager，全局事务管理器。用于管理事务状态的部件。
GUC	Grand Unified Configuration，数据库运行参数。配置这些参数可以影响数据库系统的行为。
HA	高可用性（HighAvailability），通过尽量缩短因日常维护操作（计划）和突发的系统崩溃（非计划）所导致的停机时间，以提高系统和应用的可用性。
HBA	host-based authentication，主机认证。主机鉴权允许主机鉴权部分或全部系统用户。适用于系统所有用户或者使用Match指令的子集。该类型鉴权对于管理计算集群以及其它完全同质设备非常有用。总之，服务器上的三个文件以及客户端上的一个文件必须修改，为主机鉴权做准备。
HDFS	Hadoop Distributed File System，Apache Hadoop项目的一个子项目。一个高度容错的分布式文件系统，设计用于在低成本硬件上运行。HDFS提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。
服务器	为客户端提供服务的软硬件的组合。单独使用时，指运行服务器操作系统的计算机，也可以指提供服务的软件或者专用硬件。
高级包	DWS提供的具有一定逻辑和功能的存储过程、函数，这些具备功能的存储过程、函数统称为高级包。
隔离性	数据库事务的ACID特性之一。它是指一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰。
关系型数据库	创建在关系模型基础上的数据库。关系型数据库借助于集合代数等数学概念和方法来处理数据库中的数据。
归档线程	数据库打开归档功能时启动的一个线程，此线程用于将数据库日志归档到指定的路径。
故障接管	功能对等的系统部件对于故障部件的自动替换过程。系统部件包含处理器、服务器、网络、数据库等。
环境变量	定义进程操作环境某一方面的变量。例如，环境变量可以为主目录，命令搜索路径，使用终端或当前时区。

版权所有 © 华为云计算技术有限公司

术语	解释
检查点	将数据库内存中某一时刻的数据存到磁盘的机制。DWS定期将已提交的事务数据和未提交的事务数据存到磁盘，这些数据用来和Redo日志一起在数据库重启和崩溃时恢复数据库。
加密	用于传输数据的功能。通过该功能，可以隐藏信息内容，防止非法使用。
节点	将构成DWS集群环境的各台服务器（物理机或虚拟机）称为集群节点，简称节点。
纠错	系统自动识别软件和数据流上的错误并自动修正错误的能力，提升系统的稳定性和可靠性。
进程	在单个计算机上执行程序的实例。一个进程由一个或多个线程组成。其它进程不能接入某个进程已占用的线程。
基于时间点恢复	PITR（Point-In-Time Recovery），基于时间点恢复是DWS备份恢复的一个特性，是指在备份数据和WAL日志正常的情况下，数据可以恢复到指定时间点。
记录	在关系型数据库中，每一条记录对应表中的每一行数据。
集群	集群是由一组服务器和其它资源组成的一个单独的系统，可以实现高可用性。有的情况下，可以实现负载均衡及并行处理。

K~O

表 20-3 K~O 术语解释

术语	解释
LLVM	LLVM命名最早源自于底层虚拟机（Low Level Virtual Machine）的缩写。LLVM是构架编译器（compiler）的框架系统，以C++编写而成，用于优化以任意程序语言编写的程序的编译时间（compile-time）、链接时间（link-time）、运行时间（run-time）以及空闲时间（idle-time），对开发者保持开放，并兼容已有脚本。 DWS LLVM动态编译技术可以为每个查询生成定制化的机器码用于替换原本的通用函数。通过减少实际查询时冗余的条件逻辑判断、虚函数调用并提高数据局域性，从而达到提升查询整体性能的目的。
LVS	Linux Virtual Server，虚拟服务器集群系统，用于负责集群的负载均衡。
逻辑复制	数据库主备或两个集群间的数据同步方式。区别于通过物理日志回放方式的物理复制，逻辑复制在两个集群间传输逻辑日志或通过逻辑日志对应的SQL语句实现数据同步。
逻辑日志	数据库修改的日志记录，可直接对应为SQL语句，一般为行级记录。区别于物理日志，物理日志是记录物理页面修改的日志。

版权所有 © 华为云计算技术有限公司

术语	解释
逻辑解码	逻辑解码是一种通过对xlog日志的反解实现将数据库表的所有持久更改抽取到一种清晰、易于理解的格式的处理过程。
逻辑复制槽	在逻辑复制的环境下，逻辑复制槽用于防止Xlog被系统或VACUUM回收。DWS中用于记录逻辑解码位置的对象，提供创建、删除、读取、推进等多个SQL接口函数。
MPP	Massive Parallel Processing，大规模并行处理。指利用多个机器构成集群的架构方式，也称为集群（Cluster）系统。
MVCC	Multi-Version Concurrency Control，多版本并发控制。数据库并发控制协议的一种，它的基本算法是一个元组可以有多个版本，不同的查询可以工作在不同的版本上。一个基本的好处是读和写可以不冲突。
NameNode	NameNode是Hadoop系统中的一个中心服务器，负责管理文件系统的名字空间(namespace)以及客户端对文件的访问。
mmap	memory map，也就是内存映射。mmap是一种内存映射文件的方法，即将一个文件或者其它对象映射到进程的地址空间，文件或设备的内容可以直接作为内存中的一个区域进行访问，而不需要显式地进行读写操作，实现高效文件读写和进程间通信。
Node Group	在DWS集群里Node Group指DN的集合，是集群中的子集群。从性质上可以分为存储子集群Storage Node Group和计算子集群Computing Node Group。存储子集群用来承载本地表的数据存储，而计算子集群用来承载查询的聚集、关联计算。
NULL FIRST	在 SQL中，NULL FIRST是用于显式控制排序时NULL值位置的子句，通常出现在ORDER BY语句中。其核心作用是强制让NULL值的行排在非NULL值的行之前（即优先显示 NULL）。 当在ORDER BY中使用NULL FIRST时，无论排序是升序（ASC）还是降序（DESC），所有NULL值的行都会优先出现在结果集的最前面，非NULL值的行紧随其后（按排序规则排列）。
OLAP	Online Analytical Processing，联机分析处理。是数据仓库系统最主要的应用，专门设计用于支持复杂的分析操作，侧重对决策人员和高层管理人员的决策支持，可以根据分析人员的要求快速、灵活地进行大数据量的复杂查询处理，并且以一种直观而易懂的形式将查询结果提供给决策人员，以便准确掌握企业（公司）的经营状况，了解对象的需求，制定正确的方案。
OM	Operations Management，运维管理模块。提供集群日常运维、配置管理的管理接口、工具。
ORC	Optimized Row Columnar，一种广泛使用的Hadoop系统结构化数据的文件格式，由Hadoop HIVE项目引入。
客户端	连接或请求其它计算机或程序服务的计算机或程序。
空闲空间管理	管理表内空闲空间的机制，通过记录每个表内空闲空间信息，并建立易于查找的数据结构，可以加速对空闲空间进行的操作（例如INSERT）。

版权所有 © 华为云计算技术有限公司

术语	解释
跨集群	DWS支持通过外表和Extension Connector访问当前DWS集群外的其他DBMS中的数据，这种行为称为跨集群。
垃圾元组	是指使用DELETE和UPDATE语句删除的元组，DWS在删除元组时只是打个删除标记，由Vacuum线程清理这种垃圾元组。
列	字段的等效概念。在数据库中，表由一列或多列组成。
逻辑节点	一个物理节点上可以安装多个逻辑节点。一个逻辑节点是一个数据库实例。
模式	数据库对象集，包括逻辑结构，例如表、视图、序、存储过程、同义名、索引、集群及数据库连接。
模式文件	用于决定数据库结构的SQL文件。

P~T

表 20-4 P~T 术语解释

术语	解释
Page	DWS数据库关系对象结构中行存的最小存储单元。一个页（Page）大小为默认为8KB。默认情况下，页大小在数据库初始化时确定，并且无法动态更改。
PostgreSQL	PostgreSQL是一个开源的关系数据库管理系统(DBMS)，由全球志愿者团队开发。PostgreSQL不受任何公司或个体所控制，源代码免费使用。
Postgres-XC	一款多节点同步，读写可扩展的PostgreSQL集群数据库。
Postmaster	数据库服务启动时启动的一个线程。用于监听来自集群其它节点或客户端的连接请求。 主机上监听到备机连接请求，并接受后，就会创建一个WAL Sender线程，用于处理与备机的交互。
RHEL	Red Hat Enterprise Linux，红帽企业Linux。
REDO日志	记录对数据库进行操作的日志，这些日志包含重新执行这些操作所需要的信息。当数据库故障时，可以利用REDO日志将数据库恢复到故障前的状态。
SCTP	Stream Control Transmission Protocol，流控制传输协议。是IETF于2000年新定义的一个传输层协议。是提供基于不可靠传输业务的协议之上的可靠的数据报传输协议。SCTP的设计用于通过IP网传输SCN窄带信令消息。

版权所有 © 华为云计算技术有限公司

术语	解释
Savepoint	保存点。是一种在关系数据库管理系统中实现子事务（也称为嵌套事务）的方法。在一个长事务中，可以把操作过程分成几部分，前面部分执行成功后，可以建一个保存点，若后面的执行失败，则回滚到这个保存点即可，无需回滚整个事务。保存点对于在数据库应用程序中实现复杂错误恢复很有用。如果在多语句事务中发生错误，则应用程序能够从错误中恢复（通过回滚到保存点）而无需中止整个事务。
Session	数据库系统在接收到应用程序的连接请求时，为该连接创建的一个任务。它被Session Manager管理，完成一些初始化任务，执行用户的所有操作。
Shared-nothing architecture	无共享架构是一种分布式计算架构，这种架构中不存在集中共享CPU、存储的状态，这种架构具有非常强的扩展性。
SIMD	Single Instruction Multiple Data，即单指令多数据，是一种并行计算技术，允许一条指令同时处理多个数据，从而显著提升计算密集型任务的性能。
SLES	SUSE Linux Enterprise Server，由SUSE提供的企业级Linux操作系统。
SMP	Symmetric Multi-Processing，对称多处理技术，是指在一台计算机上汇集了一组处理器（多CPU），各CPU之间共享内存子系统以及总线结构。操作系统必须支持多任务和多线程处理，以使得SMP系统发挥高效的性能。数据库领域的SMP并行技术，一般指利用多线程技术实现查询的并行执行，以充分利用CPU资源，从而提升查询性能。
SQL	Structure Query Language，结构化查询语言。数据库的标准查询语言。它可以分为数据定义语言（DDL），数据操纵语言（DML）和数据控制语言（DCL）。
SSL	Secure Socket Layer，安全套接层。SSL是Netscape公司率先采用的网络安全协议。它是在传输通信协议（TCP/IP）上实现的一种安全协议，采用公开密钥技术。SSL广泛支持各种类型的网络，同时提供三种基本的安全服务，它们都使用公开密钥技术。SSL支持服务通过网络进行通信而不损害安全性。它在客户端和服务器之间创建一个安全连接。然后通过该连接安全地发送任意数据量。
收敛比	交换机下行带宽与上行带宽的比值。收敛比越高，流量收敛程度越大，丢包越严重。
TCP	Transmission Control Protocol，传输控制协议。用于将数据信息分解成信息包，使之经过IP协议发送；并对利用IP协议接收来的信息包进行校验并将其重新装配成完整的信息。TCP是面向连接的可靠协议，能够确保信息的无误发送。
trace	一种特殊的日志记录方法，用来记录程序执行的信息。程序员使用该信息进行纠错。另外，根据trace日志中信息的类型和内容，有经验的系统管理员或技术支持人员以及软件监控工具诊断软件常见问题。

版权所有 © 华为云计算技术有限公司

术语	解释
逃逸字符	在数据库中，逃逸字符（Escape Character）是一种特殊字符，用于转义其他字符，它的主要作用是让数据库系统将原本具有特殊含义的字符当作普通字符处理，让数据库识别并正确处理那些可能与SQL语法冲突的字符（如引号、特殊符号等）。常见的逃逸字符有反斜杠\或双引号"。
全备份	备份整个数据库集群。
全量同步	DWS双机方案中的一种数据同步机制，是指把主机中的所有数据同步给备机。
日志文件	计算机记录自身活动的记录。
事务	数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成，事务必须满足ACID原则。
数据	事实或指令的一种表达形式，适用于人为或自动的通信、解释或处理。数据包含常量、变量、阵列和字符串。
数据重分布	用户改变数据的分布方式后，数据表在节点间重新分布的过程。
数据分布	表数据在分布式环境中的分布方式（Distribution），即数据表以何种方式打散存储到各个数据库实例上去。具体的分布方式可以有：散列（Hash）方式，复制方式（Replication）和随机方式（Random）。散列方式根据元组中指定字段的取值算得哈希值，根据节点与哈希值的映射关系获得该元组的目标存储位置。复制方式将元组复制到所有节点上。随机方式将数据随机分布到各节点。
数据分区	数据分区是指在一个数据库实例内部，将表按照划分为多个数据互不重叠的部分（Partition）。具体的分区方式可以有：范围分区（Range），它根据元组中指定字段的取值所处的范围映射到目标存储位置。
数据库	数据库是存储在一起的相关数据的集合，这些数据可以被访问，管理以及更新。同一视图中，数据库可以根据存储内容类型分为以下几类：数目类、全文本类、数字类及图像类。
数据库实例	一个数据库实例是一个DWS进程以及它控制的数据库文件。DWS在一个物理节点上安装多个数据库实例，集群各节点上所安装的GTM、CM、CN、DN统称为实例。一个数据库实例也被称为一个逻辑节点。
数据库双机	DWS提供的高可靠性双机方案。在此方案中，每个DWS逻辑节点标识为主机或备机。在同一时间内，只有一个DWS被标识为主机。双机初次建立时，主机会对每个备机数据做全量同步，然后做增量同步。双机建立之后的运行过程中，主机能接受数据读和写的操作请求，备机只做日志同步。
数据库文件	保存用户数据和数据库系统内部数据的二进制文件。

版权所有 © 华为云计算技术有限公司

术语	解释
数据流操作符	负责查询片段间交换数据的操作符。根据数据流的输入、输出关系，可以细分为聚合流（Gather）、广播流（Broadcast）和重分布流（Redistribution）。聚合流将数据从多个查询片段聚合到一个。广播流将数据从一个查询片段向多个传输。重分布流则将多个查询片段的数据，按照一定规则重组后向多个传输。
数据字典	数据字典是一系列只读的表，用来提供数据库的信息。这些信息包括：数据库设计信息、存储过程信息、用户权限、用户统计数据、数据库进程信息、数据库增长统计数据和数据库性能统计数据。
死锁	为使用同一资源而产生的无法解决的争用状态。
索引	数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。
统计信息	数据库使用统计信息估算查询代价，以查找代价最小的执行计划，统计信息一般是数据库自动收集的，包括表级信息（元组数、页面数等）和列级信息（列的值域分布直方图）。
停用词	在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为Stop Words（停用词）。

U~Z

表 20-5 U~Z 术语解释

术语	解释
UNLOGGED表	即非日志表，Unlogged表是一种特殊类型的表，其数据操作不记录预写日志（WAL，Write-Ahead Logging）。这类表在特定场景下能显著提升写入性能，但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动截断，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。 CREATE TABLE语法中，指定UNLOGGED参数，则创建为非日志表。
Vacuum	数据库定期启动的清理垃圾元组的线程，根据配置参数可以同时启动多个。
V2表	指的是建表环节时，执行CREATE TABLE语法中定义的colversion取值为2.0的表，表示列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名，数据存储在本地盘。存算一体场景下，不指定colversion取值时，用户创建的列存表默认为V2表。

版权所有 © 华为云计算技术有限公司

术语	解释
V3表	指的是建表环节时，执行CREATE TABLE语法中定义的colversion取值为3.0的表，即存算分离表，表示列存表的每列合并存储在一个文件中，文件名以C1_field.0命名，数据存储在OBS文件系统。存算分离场景下，不指定colversion取值时，用户创建的列存表默认为V3表。
verbose	verbose选项指定显示在屏幕上的处理信息。
WAL	Write-Ahead Logging，预写日志系统。实现事务日志的标准方法，是指对数据文件（表和索引的载体）持久化修改之前必须先持久化相应的日志。
WAL Receiver	数据库复制时备机创建的一个线程的名称。此线程用于从主机接收数据、命令，并反馈确认信息至主机。一个备机只有一个WAL Receiver线程。
WAL Sender	数据库复制过程中，主机接收到备机的连接请求后创建的一个线程的名称。此线程用于发送命令、数据到备机，并从备机接收信息。一个主机可能会有多个WAL Sender线程，每一个WAL Sender线程对应一个备机的一个连接请求。
WAL Writer	数据库启动时创建的一个写Redo日志的线程，用于将内存中的日志写入到持久性设备（如：磁盘）。
WLM	WorkLoad Manager，负载管理。DWS中系统资源控制和分配的模块。
Xlog	表示事务日志，一个逻辑节点中只有一个，不允许创建多个Xlog文件。
xDR	详单。用户面和信令面详单的统称，包括CDR和UFDR、TDR和SDR。
网络备份	网络备份为各种平台提供一套完整的、灵活的数据保护方案。平台包含Windows、UNIX及Linux。网络备份支持备份、归档、恢复计算机上的文件、文件夹或目录、卷或分区。
谓词	谓词（Predicate）指的是一个逻辑表达式，用于筛选或限定数据。它本质上是一个返回布尔值(TRUE/FALSE/NULL)的条件表达式。像WHERE、CHECK及JOIN子句中都会用到谓词。例如WHERE salary > 5000，“salary > 5000”就是谓词。
谓词列	谓词列指的是在谓词表达式里被引用的列。例如，在salary > 5000这个谓词中，salary就是谓词列。若查询中的谓词列存在索引，查询优化器可能会利用该索引来加速数据的过滤，这种情况被称为“索引扫描”。DWS会收集谓词列的统计信息（9.1.0.100及以上集群版本支持），以此来辅助生成更优的查询计划。
物理节点	一个物理机器称为一个物理节点。
物化视图	物化视图（Materialized View）是数据库中一种特殊类型的数据库对象。它将复杂计算的查询结果预计算并存储在数据库中，实现加速查询的目的。

版权所有 © 华为云计算技术有限公司

术语	解释
系统表	存储数据库元信息的表，元信息包括数据库中的用户表、索引、列、函数和数据类型等。
下推	DWS是分布式数据库，其可以利用多DN分布式并行执行查询计划，即将CN中的查询计划下发到各DN中并行执行。这种行为称为下推。与将数据抽取到CN上执行查询的方式相比，下推可以大幅提升查询性能。
压缩	数据压缩，信源编码，或比特率降低涉及使用相比原来较少比特的编码信息。压缩可以是有损或无损。无损压缩通过识别和消除统计冗余降低比特位。无损压缩中没有信息丢失。有损压缩识别并删除次要信息，减少了比特位。减少数据文件大小的方法被普遍称为数据压缩，尽管其正式名称为源编码（数据源的编码，然后将其存储或传输）。
一致性	数据库事务的ACID特性之一。在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
元数据	用来定义数据的数据。主要是描述数据自身信息，包含源、大小、格式或其它数据特征。数据库字段中，元数据用于理解以及诠释数据仓库的内容。
原子性	数据库事务的ACID特性之一。整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚到事务开始前的状态，就像这个事务从来没有执行过一样。
Zhparser	Zhparser是一个针对中文全文检索的扩展插件。它的主要功能是提供中文分词支持，使DWS能够更高效地处理中文文本的搜索和分析。
在线扩容	在线扩容是指DWS扩容重分布过程中支持数据持续入库、查询业务不中断。
脏页面	已经被修改且未写入持久性设备的页面。
增量备份	基于上次有效备份之后对文件修改的备份。
增量同步	DWS双机方案中的一种数据同步机制，是指把主机中数据增量同步给备机，即只同步主备间有差异的数据。
主机	DWS数据库双机系统中接受数据读写操作的节点，和所有备机一起协同工作。在同一时间内，双机系统中只有一个节点被标识为主机。
主题词	在标引和检索中用于表达文献主题的规范化的词或词组。
转储文件	转储文件是一种特定类型的trace文件。转储文件为响应事件过程中一次性输出的诊断数据，trace文件指诊断数据的连续输出。
资源池	资源池是DWS提供的一种资源划分的配置机制。通过将用户绑定到资源池，来限定其所执行作业的优先级及能够利用到的资源。

版权所有 © 华为云计算技术有限公司

术语	解释
租户	数据库业务用户在给定的计算资源（cpu，内存和io）和存储资源下执行业务，通过资源管理和隔离，达成业务的服务等级协定（SLA）。
最小恢复点	最小恢复点是DWS提供的数据一致性保障手段之一。最小恢复点特性可以在DWS启动时检查出WAL日志和持久化到磁盘的数据的不一致性，并提示用户进行处理。

版权所有 © 华为云计算技术有限公司

21 SQL 语法参考

21.1 DWS SQL 概述

什么是 SQL

SQL是用于访问和处理数据库的标准计算机语言。

SQL提供了各种任务的语句，包括：

- 查询数据。
- 在表中插入、更新和删除行。
- 创建、替换、更改和删除对象。
- 控制对数据库及其对象的访问。
- 保证数据库的一致性和完整性。

SQL语言由用于处理数据库和数据库对象的命令和函数组成。该语言还会强制实施有关数据类型、表达式和文本使用的规则。因此在SQL参考章节，除了SQL语法参考外，还会看到有关数据类型、表达式、函数和操作符等信息。

SQL 发展简史

SQL发展简史如下：

- 1986年，ANSI X3.135-1986，ISO/IEC 9075:1986，SQL-86
- 1989年，ANSI X3.135-1989，ISO/IEC 9075:1989，SQL-89
- 1992年，ANSI X3.135-1992，ISO/IEC 9075:1992，SQL-92（SQL2）
- 1999年，ISO/IEC 9075:1999，SQL:1999（SQL3）
- 2003年，ISO/IEC 9075:2003，SQL:2003（SQL4）
- 2011年，ISO/IEC 9075:200N，SQL:2011（SQL5）

DWS 支持的 SQL 标准

DWS兼容Postgres-XC，默认支持SQL2、SQL3和SQL4的主要特性、SQL5的部分特性。

 版权所有 © 华为云计算技术有限公司

DWS 支持的语言扩展

DWS支持的语言扩展包括PL/pgSQL和PL/Java。

SQL 示例说明

手册中的部分SQL示例是基于TPC-DS模型开发的，如果需要运行手册中的示例，请先参考官网说明（**http://www.tpc.org/tpcds/**），安装TPC-DS benchmark。

21.2 DWS SQL 语法格式说明

为了方便对语法使用的理解，在文档中对SQL语法文本按如下格式进行表述。

表 21-1 SQL 语法格式说明

格式	意义
大写	语法关键字（语句中保持不变、必须与语法格式一致的部分）采用大写表示。
小写	参数（语句中必须由实际值进行替代的部分）采用小写表示。
[ ]	可选语法项。表示用“[ ]”括起来的部分是可选的。
{ }	必选语法项。
...	表示前面的元素可重复出现。
[x | y | ...]	表示从两个或多个选项中选取一个或者不选。
{ x | y | ... }	表示从两个或多个选项中选取一个。
[x | y | ...] [...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用空格分隔。
[x | y | ...] [,...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用逗号分隔。
{ x | y | ... } [...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间以空格分隔。
{ x | y | ... } [,...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间用逗号分隔。

21.3 与 PostgreSQL 的差异

DWS与PostgreSQL的差异基于PostgreSQL 9.X版本整理，具体差异如下：

客户端差异

DWS gsql相较于PostgreSQL psql做了如下安全加固变更：

版权所有 © 华为云计算技术有限公司

- 取消通过元命令\password设置用户密码。
- 新增元命令\i+、 \ir+、\include_relative+和输入输出参数-k，以支持给导入导出的文件加密。
- 取消打印命令行历史到文件的元命令\s。
- 涉及敏感操作SQL历史不再记录，如含有密码操作。即用户通过翻页/上下键查阅SQL历史将不能查到对应的记录。
- 支持连接后在屏幕上给出用户密码过期提示以及版本信息。

gsql在psql基础上还增加了如下功能：

- 新增输出格式参数-r。支持用户输入命令时的tab补齐和方向键调整焦点。
- 新增并行操作元命令\parallel，以提升执行性能。
- 新增\set RETRY支持语句出错重试。
- 新增PLSQL默认结束符功能，将“/”作为PLSQL语句（ create or replace function/procedure ）的默认结束符，增加便利性。

libpq：

DWS在开发某些功能，如客户端连接工具gsql时，对PostgreSQL libpq进行了较大修改，但并未对此接口在应用程序开发场景下的使用做验证。因此对使用此接口做应用程序开发存在的风险未知，故不推荐用户使用此套接口做应用程序开发。推荐用户使用ODBC或JDBC接口来替代。

SQL 差异

表 21-2 DWS 不支持的 PostgreSQL 数据库语言

分类	DWS不支持	说明
数据类型	几何类型line	DWS所支持的数据类型参见**数据类型**。
	pg_node_tree	
函数	枚举支持函数： • enum_first(anyenum) • enum_last(anyenum) • enum_range(anyenum) • enum_range(anyenum, anyenum)	DWS所支持的函数参见**函数和操作符**。
	访问权限查询函数： • has_sequence_privilege(user, sequence, privilege) • has_sequence_privilege(sequence, privilege)	
	系统目录信息函数： • pg_get_triggerdef(trigger_oid) • pg_get_triggerdef(trigger_oid, pretty_bool)	

版权所有 © 华为云计算技术有限公司

分类	DWS不支持	说明
	几何类型转换函数： line(point, point)	
	pg_node_tree函数	
SQL语法	CREATE TABLE子句： INHERITS (parent_table [, ...])	继承表。
	CREATE TABLE的列约束： REFERENCES reftable [(refcolumn)] [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] [ON DELETE action] [ON UPDATE action]	列约束中用REFERENCES reftable [(refcolumn)] [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] [ON DELETE action] [ON UPDATE action] 为表创建外键约束。
	CREATE TABLE的表约束： EXCLUDE [USING index_method] (exclude_element WITH operator [, ...])	表约束中用EXCLUDE [USING index_method] (exclude_element WITH operator [, ...])为表创建排除约束。
	CREATE/ALTER/DROP EXTENSION	扩展的加载、修改和删除。
	CREATE/ALTER/DROP AGGREGATE	聚集函数的定义、修改和删除。
	CREATE/ALTER/DROP OPERATOR	操作符（OPERATOR）的创建、修改和删除。
	CREATE/ALTER/DROP OPERATOR CLASS	操作符类（OPERATOR CLASS）的创建、修改和删除。
	CREATE/ALTER/DROP OPERATOR FAMILY	操作符族（OPERATOR FAMILY）的创建、修改和删除。
	CREATE/ALTER/DROP TEXT SEARCH PARSER	文本检索解析器（TEXT SEARCH PARSER）的创建、修改和删除。
	CREATE/ALTER/DROP TEXT SEARCH TEMPLATE	文本检索模板（TEXT SEARCH TEMPLATE）的创建、修改和删除。
	CREATE/ALTER/DROP COLLATION	排序规则（COLLATION）的创建、修改和删除
	CREATE/ALTER/DROP CONVERSION	字符集编码转换（CONVERSION）的定义、修改和删除。

版权所有 © 华为云计算技术有限公司

分类	DWS不支持	说明
	CREATE/ALTER/DROP RULE	规则（RULE）的创建、修改和删除。
	CREATE/ALTER/DROP LANGUAGE	过程语言（LANGUAGE）的注册、修改和删除。
	CREATE/ALTER/DROP DOMAIN	域（DOMAIN）的创建、修改和删除。
	CREATE/DROP CAST	类型转换（CAST）的定义和删除。
	CREATE/ALTER/DROP USER MAPPING	用户映射（USER MAPPING）的定义、修改和删除。
	SECURITY LABEL	定义或更改对象的安全标签。
	NOTIFY	生成一个通知。
	LISTEN	监听一个通知。
	UNLISTEN	停止监听通知信息。
	LOAD	加载或重新加载一个共享库文件。
	DISCARD	释放一个数据库的会话资源（8.2.0及以上集群版本已支持DISCARD）。
	MOVE BACKWARD	反向移动游标。
	COPY的COPY FROM FILE和COPY TO FILE	为了权限的隔离，DWS禁用COPY FROM FILE和COPY TO FILE。
其他	自定义C函数	DWS支持的用户自定义函数可参见《数据仓库服务开发指南》中“用户自定义函数”章节。

21.4 关键字

SQL里有保留关键字和非保留关键字之分。根据标准，保留关键字绝不能用做其他标识符。非保留关键字只是在特定的环境里有特殊的含义，而在其他环境里是可以用做标识符的，即便如此，也尽量避免使用非保留关键字。

可以使用**SELECT * FROM pg_get_keywords();**查询DWS的关键字。

版权所有 © 华为云计算技术有限公司

举例：“AUTHORIZATION”为保留关键字，当使用“AUTHORIZATION”作为表名执行建表语句时系统会报错，不允许创建名称为“AUTHORIZATION”的表。

但是有例外情况，“AUTHORIZATION”可以作为函数或类型名称使用（下表有说明）。

图 21-1 使用关键字作为表名

图 21-2 使用关键字作为函数名

标识符命名规范

数据库标识符是用来唯一标识数据库中对象（如表、列、索引等）的名称。在数据库中，标识符在SQL查询语句中经常被用来引用数据库对象，确保数据库操作的准确性和一致性。合理命名和使用标识符非常重要，DWS标识符的命名需遵守如下规范：

- 标识符需要由字母（a-z）、下划线（_）、数字（0-9）或$组成。
- 标识符建议以字母（a-z）或下划线（_）开头。
- 标识符长度不超过63个字符。
- 标识符不能是保留的SQL关键字，非保留关键字请慎用。
- 数据库字符集为GBK、UTF8、SQL_ASCII时标识符支持中文汉字，其中UTF8、SQL_ASCII字符集下，一个中文汉字占3个字符，最多支持21个中文汉字；GBK字符集下，一个中文汉字占2个字符，最多支持31个中文汉字。Latin1字符集不支持中文汉字命名。字符集格式在创建数据库时指定，详情参见**CREATE DATABASE**。

 版权所有 © 华为云计算技术有限公司

表 21-3 SQL 关键字列表

关键字	DWS	SQL:1999	SQL:1992
ABORT	非保留	-	-
ABS	-	非保留	-
ABSOLUTE	非保留	保留	保留
ACCESS	非保留	-	-
ACCOUNT	非保留	-	-
ACTION	非保留	保留	保留
ADA	-	非保留	非保留
ADD	非保留	保留	保留
ADMIN	非保留	保留	-
AFTER	非保留	保留	-
AGGREGATE	非保留	保留	-
ALIAS	-	保留	-
ALL	保留	保留	保留
ALLOCATE	-	保留	保留
ALSO	非保留	-	-
ALTER	非保留	保留	保留
ALWAYS	非保留	-	-
ANALYSE	保留	-	-
ANALYZE	保留	-	-
AND	保留	保留	保留
ANY	保留	保留	保留
APP	非保留	-	-
ARE	-	保留	保留
ARRAY	保留	保留	-
AS	保留	保留	保留
ASC	保留	保留	保留
ASENSITIVE	-	非保留	-
ASSERTION	非保留	保留	保留
ASSIGNMENT	非保留	非保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
ASYMMETRIC	保留	非保留	-
AT	非保留	保留	保留
ATOMIC	-	非保留	-
ATTRIBUTE	非保留	-	-
AUTHID	保留	-	-
AUTHINFO	非保留	-	-
AUTHORIZATION	保留(可以是函数或类型)	保留	保留
AUTO	非保留	-	-
AUTOEXTEND	非保留	-	-
AUTOMAPPED	非保留	-	-
AUTOREFRESH	非保留	-	-
AUTOSCALE	非保留	-	-
AVG	-	非保留	保留
BACKWARD	非保留	-	-
BARRIER	非保留	-	-
BEFORE	非保留	保留	-
BEGIN	非保留	保留	保留
BETWEEN	非保留(不能是函数或类型)	非保留	保留
BIGINT	非保留(不能是函数或类型)	-	-
BINARY	保留(可以是函数或类型)	保留	-
BINARY_DOUBLE	非保留(不能是函数或类型)	-	-
BINARY_INTEGER	非保留(不能是函数或类型)	-	-
BIT	非保留(不能是函数或类型)	保留	保留
BITVAR	-	非保留	-
BIT_LENGTH	-	非保留	保留
BLOB	非保留	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
BLOCK	非保留	-	-
BOOLEAN	非保留(不能是函数或类型)	保留	-
BOTH	保留	保留	保留
BUCKETS	保留	-	-
BUILD	非保留	-	-
BREADTH	-	保留	-
BY	非保留	保留	保留
C	-	非保留	非保留
CACHE	非保留	-	-
CALL	非保留	保留	-
CALLED	非保留	非保留	-
CARDINALITY	-	非保留	-
CASCADE	非保留	保留	保留
CASCADED	非保留	保留	保留
CASE	保留	保留	保留
CAST	保留	保留	保留
CATALOG	非保留	保留	保留
CATALOG_NAME	-	非保留	非保留
CHAIN	非保留	非保留	-
CHANGE	非保留	-	-
CHAR	非保留(不能是函数或类型)	保留	保留
CHARACTER	非保留(不能是函数或类型)	保留	保留
CHARACTERISTICS	非保留	-	-
CHARACTER_LENGTH	-	非保留	保留
CHARACTER_SET_CATALOG	-	非保留	非保留
CHARACTER_SET_NAME	-	非保留	非保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
CHARACTER_SET_SCHEMA	-	非保留	非保留
CHAR_LENGTH	-	非保留	保留
CHECK	保留	保留	保留
CHECKED	-	非保留	-
CHECKPOINT	非保留	-	-
CLASS	非保留	保留	-
CLEAN	非保留	-	-
CLASS_ORIGIN	-	非保留	非保留
CLOB	非保留	保留	-
CLOSE	非保留	保留	保留
CLUSTER	非保留	-	-
COALESCE	非保留(不能是函数或类型)	非保留	保留
COBOL	-	非保留	非保留
COLLATE	保留	保留	保留
COLLATION	保留(可以是函数或类型)	保留	保留
COLLATION_CATALOG	-	非保留	非保留
COLLATION_NAME	-	非保留	非保留
COLLATION_SCHEMA	-	非保留	非保留
COLUMN	保留	保留	保留
COLUMNS	非保留	-	-
COLUMN_NAME	-	非保留	非保留
COMMAND_FUNCTION	-	非保留	非保留
COMMAND_FUNCTION_CODE	-	非保留	-
COMMENT	非保留	-	-
COMMENTS	非保留	-	-
COMMIT	非保留	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
COMMITTED	非保留	非保留	非保留
COMPACT	保留(可以是函数或类型)	-	-
COMPATIBLE_ILLEGAL_CHARS	非保留	-	-
COMPLETE	非保留	-	-
COMPRESS	非保留	-	-
COMPLETION	-	保留	-
CONCURRENTLY	保留(可以是函数或类型)	-	-
CONDITION	-	-	-
CONDITION_NUMBER	-	非保留	非保留
CONFIGURATION	非保留	-	-
CONFLICT	非保留	-	-
CONNECT	-	保留	保留
CONNECTION	非保留	保留	保留
CONNECTION_NAME	-	非保留	非保留
CONSTRAINT	保留	保留	保留
CONSTRAINTS	非保留	保留	保留
CONSTRAINT_CATALOG	-	非保留	非保留
CONSTRAINT_NAME	-	非保留	非保留
CONSTRAINT_SCHEMA	-	非保留	非保留
CONSTRUCTOR	-	保留	-
CONTAINS	-	非保留	-
CONTENT	非保留	-	-
CONTINUE	非保留	保留	保留
CONVERSION	非保留	-	-
CONVERT	-	非保留	保留
COORDINATOR	非保留	-	-

 版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
COPY	非保留	-	-
CORRESPONDING	-	保留	保留
COST	非保留	-	-
COUNT	-	非保留	保留
CREATE	保留	保留	保留
CROSS	保留(可以是函数或类型)	保留	保留
CSV	非保留	-	-
CUBE	非保留	保留	-
CURRENT	非保留	保留	保留
CURRENT_CATALOG	保留	-	-
CURRENT_DATE	保留	保留	保留
CURRENT_PATH	-	保留	-
CURRENT_ROLE	保留	保留	-
CURRENT_SCHEMA	保留(可以是函数或类型)	-	-
CURRENT_TIME	保留	保留	保留
CURRENT_TIMESTAMP	保留	保留	保留
CURRENT_USER	保留	保留	保留
CURSOR	非保留	保留	保留
CURSOR_NAME	-	非保留	非保留
CYCLE	非保留	保留	-
DATA	非保留	保留	非保留
DATE_FORMAT	非保留	-	-
DATABASE	非保留	-	-
DATAFILE	非保留	-	-
DATETIME_INTERVAL_CODE	-	非保留	非保留
DATETIME_INTERVAL_PRECISION	-	非保留	非保留
DAY	非保留	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
DBCOMPATIBILITY	非保留	-	-
DBCOMPATIBILITY_BEHAVIOR	非保留	-	-
DEALLOCATE	非保留	保留	保留
DEC	非保留(不能是函数或类型)	保留	保留
DECIMAL	非保留(不能是函数或类型)	保留	保留
DECLARE	非保留	保留	保留
DECODE	非保留(不能是函数或类型)	-	-
DEFAULT	保留	保留	保留
DEFAULTS	非保留	-	-
DEFERRABLE	保留	保留	保留
DEFERRED	非保留	保留	保留
DEFINED	-	非保留	-
DEFINER	非保留	非保留	-
DELETE	非保留	保留	保留
DELIMITER	非保留	-	-
DELIMITERS	非保留	-	-
DELTA	非保留	-	-
DELTAMERGE	保留(可以是函数或类型)	-	-
DEMAND	非保留	-	-
DEPTH	-	保留	-
DEREF	-	保留	-
DESC	保留	保留	保留
DESCRIBE	-	保留	保留
DESCRIPTOR	-	保留	保留
DESTROY	-	保留	-
DESTRUCTOR	-	保留	-
DETERMINISTIC	非保留	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
DIAGNOSTICS	-	保留	保留
DICTIONARY	非保留	保留	-
DIRECT	非保留	-	-
DIRECTORY	非保留	-	-
DISABLE	非保留	-	-
DISCARD	非保留	-	-
DISCONNECT	-	保留	保留
DISPATCH	-	非保留	-
DISTINCT	保留	保留	保留
DISTRIBUTE	非保留	-	-
DISTRIBUTION	非保留	-	-
DO	保留	-	-
DOCUMENT	非保留	-	-
DOMAIN	非保留	保留	保留
DOUBLE	非保留	保留	保留
DROP	非保留	保留	保留
DROPCOLUMNS	非保留	-	-
DUPLICATE	非保留	-	-
DYNAMIC	-	保留	-
DYNAMIC_FUNCTION	-	非保留	非保留
DYNAMIC_FUNCTION_CODE	-	非保留	-
EACH	非保留	保留	-
ELASTIC	非保留	-	-
ELSE	保留	保留	保留
ENABLE	非保留	-	-
ENCODING	非保留	-	-
ENCRYPTED	非保留	-	-
END	保留	保留	保留
END-EXEC	-	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
ENFORCED	非保留	-	-
ENUM	非保留	-	-
EOL	非保留	-	-
EQUALS	-	保留	-
ERRORS	非保留	-	-
ESCAPE	非保留	保留	保留
ESCAPING	非保留	-	-
EVERY	非保留	保留	-
EXCEPT	保留	保留	保留
EXCEPTION	-	保留	保留
EXCHANGE	非保留	-	-
EXCLUDE	非保留	-	-
EXCLUDING	非保留	-	-
EXCLUSIVE	非保留	-	-
EXEC	-	保留	保留
EXECUTE	非保留	保留	保留
EXISTING	-	非保留	-
EXISTS	非保留(不能是函数或类型)	非保留	保留
EXPIRATION	非保留	-	-
EXPLAIN	非保留	-	-
EXTENSION	非保留	-	-
EXTERNAL	非保留	保留	保留
EXTRACT	非保留(不能是函数或类型)	非保留	保留
FALSE	保留	保留	保留
FAMILY	非保留	-	-
FAST	非保留	-	-
FENCED	保留	-	-
FETCH	保留	保留	保留
FILEHEADER	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
FILL_AVG	非保留(不能是函数或类型)	-	-
FILL_FIRST	非保留(不能是函数或类型)	-	-
FILL_LAST	非保留(不能是函数或类型)	-	-
FILL_MISSING_FIELDS	非保留	-	-
FILTER	非保留	-	-
FINAL	-	非保留	-
FIRST	非保留	保留	保留
FIXED	非保留	保留	保留
FLOAT	非保留(不能是函数或类型)	保留	保留
FOLLOWING	非保留	-	-
FOR	保留	保留	保留
FORCE	非保留	-	-
FOREIGN	保留	保留	保留
FORMATTER	非保留	-	-
FORTRAN	-	非保留	非保留
FORWARD	非保留	-	-
FOUND	-	保留	保留
FREE	-	保留	-
FREEZE	保留(可以是函数或类型)	-	-
FROM	保留	保留	保留
FULL	保留(可以是函数或类型)	保留	保留
FUNCTION	非保留	保留	-
FUNCTIONS	非保留	-	-
G	-	非保留	-
GENERAL	-	保留	-
GENERATED	-	非保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
GET	-	保留	保留
GLOBAL	非保留	保留	保留
GO	-	保留	保留
GOTO	-	保留	保留
GRANT	保留	保留	保留
GRANTED	非保留	非保留	-
GREATEST	非保留(不能是函数或类型)	-	-
GROUP	保留	保留	保留
GROUPING	非保留(不能是函数或类型)	保留	-
HANDLER	非保留	-	-
HAVING	保留	保留	保留
HDFSDIRECTORY	保留(可以是函数或类型)	-	-
HEADER	非保留	-	-
HIERARCHY	-	非保留	-
HOLD	非保留	非保留	-
HOST	-	保留	-
HOT	保留	-	-
HOUR	非保留	保留	保留
IDENTIFIED	非保留	-	-
IDENTITY	非保留	保留	保留
IF	非保留(不能是函数或类型)	-	-
IFNULL	非保留(不能是函数或类型)	-	-
IGNORE	非保留	保留	-
IGNORE_EXTRA_DATA	非保留	-	-
ILIKE	保留(可以是函数或类型)	-	-
IMMEDIATE	非保留	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
IMMUTABLE	非保留	-	-
IMPLEMENTATION	-	非保留	-
IMPLICIT	非保留	-	-
IN	保留	保留	保留
INCLUDING	非保留	-	-
INCREMENT	非保留	-	-
INDEX	非保留	-	-
INDEXES	非保留	-	-
INDICATOR	-	保留	保留
INFIX	-	非保留	-
INHERIT	非保留	-	-
INHERITS	非保留	-	-
INITIAL	非保留	-	-
INITIALIZE	-	保留	-
INITIALLY	保留	保留	保留
INITRANS	非保留	-	-
INLINE	非保留	-	-
INNER	保留(可以是函数或类型)	保留	保留
INOUT	非保留(不能是函数或类型)	保留	-
INPUT	非保留	保留	保留
INSENSITIVE	非保留	非保留	保留
INSERT	非保留	保留	保留
INSTANCE	-	非保留	-
INSTANTIABLE	-	非保留	-
INSTEAD	非保留	-	-
INT	非保留(不能是函数或类型)	保留	保留
INTEGER	非保留(不能是函数或类型)	保留	保留
INTERNAL	保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
INTERSECT	保留	保留	保留
INTERVAL	非保留(不能是函数或类型)	保留	保留
INTO	保留	保留	保留
INVALID	非保留	-	-
INVOKER	非保留	非保留	-
IS	保留	保留	保留
ISNULL	非保留(不能是函数或类型)	-	-
ISOLATION	非保留	保留	保留
ITERATE	-	保留	-
JOIN	保留(可以是函数或类型)	保留	保留
K	-	非保留	-
KEEP	非保留	-	-
KEEPALIVE	非保留	-	-
KEY	非保留	保留	保留
KEY_MEMBER	-	非保留	-
KEY_TYPE	-	非保留	-
LABEL	非保留	-	-
LANGUAGE	非保留	保留	保留
LARGE	非保留	保留	-
LAST	非保留	保留	保留
LATERAL	-	保留	-
LC_COLLATE	非保留	-	-
LC_CTYPE	非保留	-	-
LEADING	保留	保留	保留
LEAKPROOF	非保留	-	-
LEAST	非保留(不能是函数或类型)	-	-
LEFT	保留(可以是函数或类型)	保留	保留

 版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
LENGTH	-	非保留	非保留
LESS	保留	保留	-
LEVEL	非保留	保留	保留
LIGHT	非保留	-	-
LIKE	保留(可以是函数或类型)	保留	保留
LIMIT	保留	保留	-
LIST	非保留	-	-
LISTEN	非保留	-	-
LOAD	非保留	-	-
LOCAL	非保留	保留	保留
LOCALTIME	保留	保留	-
LOCALTIMESTAMP	保留	保留	-
LOCATION	非保留	-	-
LOCATOR	-	保留	-
LOCK	非保留	-	-
LOG	非保留	-	-
LOGGING	非保留	-	-
LOGIN	非保留	-	-
LOOP	非保留	-	-
LOWER	-	非保留	保留
M	-	非保留	-
MAP	-	保留	-
MAPPING	非保留	-	-
MASKED	非保留	-	-
MATCH	非保留	保留	保留
MATCHED	非保留	-	-
MATERIALIZED	非保留	-	-
MAX	-	非保留	保留
MAXEXTENTS	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
MAXSIZE	非保留	-	-
MAXTRANS	非保留	-	-
MAXVALUE	保留	-	-
MERGE	非保留	-	-
MESSAGE_LENGTH	-	非保留	非保留
MESSAGE_OCTET_LENGTH	-	非保留	非保留
MESSAGE_TEXT	-	非保留	非保留
METAADDRESS	非保留	-	-
METHOD	-	非保留	-
MIN	-	非保留	保留
MINEXTENTS	非保留	-	-
MINUS	保留	-	-
MINUTE	非保留	保留	保留
MINVALUE	非保留	-	-
MOD	-	非保留	-
MODE	非保留	-	-
MODIFIES	-	保留	-
MODIFY	保留	保留	-
MODULE	-	保留	保留
MONTH	非保留	保留	保留
MORE	-	非保留	非保留
MOVE	非保留	-	-
MOVEMENT	非保留	-	-
MUMPS	-	非保留	非保留
NAME	非保留	非保留	非保留
NAMES	非保留	保留	保留
NATIONAL	非保留(不能是函数或类型)	保留	保留
NATURAL	保留(可以是函数或类型)	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
NBUCKETS	非保留	-	-
NCHAR	非保留(不能是函数或类型)	保留	保留
NCLOB	-	保留	-
NEW	-	保留	-
NEXT	非保留	保留	保留
NLSSORT	保留	-	-
NO	非保留	保留	保留
NOCOMPRESS	非保留	-	-
NOCYCLE	非保留	-	-
NODE	非保留	-	-
NOLOGGING	非保留	-	-
NOLOGIN	非保留	-	-
NOMAXVALUE	非保留	-	-
NOMINVALUE	非保留	-	-
NONE	非保留(不能是函数或类型)	保留	-
NOT	保留	保留	保留
NOTHING	非保留	-	-
NOTIFY	非保留	-	-
NOTNULL	保留(可以是函数或类型)	-	-
NOWAIT	非保留	-	-
NULL	保留	保留	保留
NULLABLE	-	非保留	非保留
NULLIF	非保留(不能是函数或类型)	非保留	保留
NULLS	非保留	-	-
NUMBER	非保留(不能是函数或类型)	非保留	非保留
NUMERIC	非保留(不能是函数或类型)	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
NUMSTR	非保留	-	-
NVARCHAR2	非保留(不能是函数或类型)	-	-
NVL	非保留(不能是函数或类型)	-	-
OBJECT	非保留	保留	-
OCTET_LENGTH	-	非保留	保留
OF	非保留	保留	保留
OFF	非保留	保留	-
OFFSET	保留	-	-
OIDS	非保留	-	-
OLD	-	保留	-
ON	保留	保留	保留
ONLY	保留	保留	保留
OPEN	-	保留	保留
OPERATION	-	保留	-
OPERATOR	非保留	-	-
OPTIMIZATION	非保留	-	-
OPTION	非保留	保留	保留
OPTIONS	非保留	非保留	-
OR	保留	保留	保留
ORDER	保留	保留	保留
ORDINALITY	非保留	保留	-
OUT	非保留(不能是函数或类型)	保留	-
OUTER	保留(可以是函数或类型)	保留	保留
OUTLINE	非保留	-	-
OUTPUT	-	保留	保留
OVER	非保留	-	-
OVERLAPS	保留(可以是函数或类型)	非保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
OVERLAY	非保留(不能是函数或类型)	非保留	-
OVERRIDING	-	非保留	-
OVERWRITE	非保留	-	-
OWNED	非保留	-	-
OWNER	非保留	-	-
PACKAGE	非保留	-	-
PAD	-	保留	保留
PARAMETER	-	保留	-
PARAMETERS	-	保留	-
PARAMETER_MODE	-	非保留	-
PARAMETER_NAME	-	非保留	-
PARAMETER_ORDINAL_POSITION	-	非保留	-
PARAMETER_SPECIFIC_CATALOG	-	非保留	-
PARAMETER_SPECIFIC_NAME	-	非保留	-
PARAMETER_SPECIFIC_SCHEMA	-	非保留	-
PARENT_NAME	非保留	-	-
PARSER	非保留	-	-
PARTIAL	非保留	保留	保留
PARTITION	非保留	-	-
PARTITIONS	非保留	-	-
PASCAL	-	非保留	非保留
PASSING	非保留	-	-
PASSWORD	非保留	-	-
PATH	-	保留	-
PCTFREE	非保留	-	-
PER	非保留	-	-
PERM	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
PERCENT	非保留	-	-
PERFORMANCE	保留	-	-
PLACING	保留	-	-
PLAN	保留	-	-
PLANS	非保留	-	-
PLI	-	非保留	非保留
POLICY	非保留	-	-
POOL	非保留	-	-
POSITION	非保留(不能是函数或类型)	非保留	保留
POSTFIX	-	保留	-
PRECEDING	非保留	-	-
PRECISION	非保留(不能是函数或类型)	保留	保留
PREDICATE	非保留	-	-
PREFERRED	非保留	-	-
PREFIX	非保留	保留	-
PREORDER	-	保留	-
PREPARE	非保留	保留	保留
PREPARED	非保留	-	-
PRESERVE	非保留	保留	保留
PRESERVE_BLANKS	非保留	-	-
PRIMARY	保留	保留	保留
PRIOR	非保留	保留	保留
PRIVATE	非保留	-	-
PRIVILEGE	非保留	-	-
PRIVILEGES	非保留	保留	保留
PROCEDURAL	非保留	-	-
PROCEDURE	保留	保留	保留
PROFILE	非保留	-	-
PUBLIC	-	保留	保留

 版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
PUBLICATION	非保留	-	-
PURGE	非保留	-	-
QUERY	非保留	-	-
QUOTE	非保留	-	-
RANGE	非保留	-	-
RAW	非保留	-	-
READ	非保留	保留	保留
READS	-	保留	-
REAL	非保留(不能是函数或类型)	保留	保留
REASSIGN	非保留	-	-
REBUILD	非保留	-	-
RECHECK	非保留	-	-
RECURSIVE	非保留	保留	-
RECYCLEBIN	保留(可以是函数或类型)	-	-
REDACTION	非保留	-	-
REF	非保留	保留	-
REFRESH	非保留	-	-
REFERENCES	保留	保留	保留
REFERENCING	-	保留	-
REINDEX	非保留	-	-
REJECT	保留	-	-
RELATIVE	非保留	保留	保留
RELEASE	非保留	-	-
RELOPTIONS	非保留	-	-
REMOTE	非保留	-	-
RENAME	非保留	-	-
REPEATABLE	非保留	非保留	非保留
REPLACE	非保留	-	-
REPLICA	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
RESET	非保留	-	-
RESIZE	非保留	-	-
RESOURCE	非保留	-	-
RESTART	非保留	-	-
RESTRICT	非保留	保留	保留
RESULT	-	保留	-
RETURN	非保留	保留	-
RETURNED_LENGTH	-	非保留	非保留
RETURNED_OCTET_LENGTH	-	非保留	非保留
RETURNED_SQLSTATE	-	非保留	非保留
RETURNING	保留	-	-
RETURNS	非保留	保留	-
REUSE	非保留	-	-
REVOKE	非保留	保留	保留
REWRITE	非保留	-	-
RIGHT	保留(可以是函数或类型)	保留	保留
ROLE	非保留	保留	-
ROLLBACK	非保留	保留	保留
ROLLUP	非保留	保留	-
ROUTINE	-	保留	-
ROUTINE_CATALOG	-	非保留	-
ROUTINE_NAME	-	非保留	-
ROUTINE_SCHEMA	-	非保留	-
ROW	非保留(不能是函数或类型)	保留	-
ROWS	非保留	保留	保留
ROW_COUNT	-	非保留	非保留
RULE	非保留	-	-
SAVEPOINT	非保留	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
SCALE	-	非保留	非保留
SCHEMA	非保留	保留	保留
SCHEMA_NAME	-	非保留	非保留
SCOPE	-	保留	-
SCROLL	非保留	保留	保留
SEARCH	非保留	保留	-
SECOND	非保留	保留	保留
SECTION	-	保留	保留
SECURITY	非保留	非保留	-
SELECT	保留	保留	保留
SELF	-	非保留	-
SENSITIVE	-	非保留	-
SEPARATOR	非保留	-	-
SEQUENCE	非保留	保留	-
SEQUENCES	非保留	-	-
SERIALIZABLE	非保留	非保留	非保留
SERVER	非保留	-	-
SERVER_NAME	-	非保留	非保留
SESSION	非保留	保留	保留
SESSION_USER	保留	保留	保留
SET	非保留	保留	保留
SETOF	非保留(不能是函数或类型)	-	-
SETS	非保留	保留	-
SHARE	非保留	-	-
SHIPPABLE	非保留	-	-
SHOW	非保留	-	-
SIMILAR	保留(可以是函数或类型)	非保留	-
SIMPLE	非保留	非保留	-
SIZE	非保留	保留	保留

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
SLOT	非保留	-	-
SMALLDATETIME	非保留(不能是函数或类型)	-	-
SMALLDATETIME_FORMAT	非保留	-	-
SMALLINT	非保留(不能是函数或类型)	保留	保留
SNAPSHOT	非保留	-	-
SOME	保留	保留	保留
SOURCE	非保留	非保留	-
SPACE	非保留	保留	保留
SPECIFIC	-	保留	-
SPECIFICTYPE	-	保留	-
SPECIFIC_NAME	-	非保留	-
SPILL	非保留	-	-
SPLIT	非保留	-	-
SQL	非保留	保留	保留
SQLCODE	-	-	保留
SQLERROR	-	-	保留
SQLEXCEPTION	-	保留	-
SQLSTATE	-	保留	保留
SQLWARNING	-	保留	-
STABLE	非保留	-	-
STANDALONE	非保留	-	-
START	非保留	保留	-
STATE	-	保留	-
STATEMENT	非保留	保留	-
STATEMENT_ID	非保留	-	-
STATIC	-	保留	-
STATISTICS	非保留	-	-
STDIN	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
STDOUT	非保留	-	-
STORAGE	非保留	-	-
STORE	非保留	-	-
STRICT	非保留	-	-
STRIP	非保留	-	-
STRUCTURE	-	保留	-
STYLE	-	非保留	-
SUBCLASS_ORIGIN	-	非保留	非保留
SUBLIST	-	非保留	-
SUBSCRIPTION	非保留	-	-
SUBSTRING	非保留(不能是函数或类型)	非保留	保留
SUM	-	非保留	保留
SYMMETRIC	保留	非保留	-
SYNONYM	非保留	-	-
SYS_REFCURSOR	非保留	-	-
SYSDATE	保留	-	-
SYSID	非保留	-	-
SYSTEM	非保留	非保留	-
SYSTEM_USER	-	保留	保留
TABLE	保留	保留	保留
TABLES	非保留	-	-
TABLESAMPLE	保留(可以是函数或类型)	-	-
TABLE_NAME	-	非保留	非保留
TEMP	非保留	-	-
TEMPLATE	非保留	-	-
TEMPORARY	非保留	保留	保留
TERMINATE	-	保留	-
TEXT	非保留	-	-
THAN	非保留	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
THEN	保留	保留	保留
TIME	非保留(不能是函数或类型)	保留	保留
TIME_FILL	非保留(不能是函数或类型)	-	-
TIME_FORMAT	非保留	-	-
TIMECAPSULE	保留(可以是函数或类型)	-	-
TIMESTAMP	非保留(不能是函数或类型)	保留	保留
TIMESTAMPADD	非保留(不能是函数或类型)	-	-
TIMESTAMPDIFF	非保留(不能是函数或类型)	-	-
TIMESTAMP_FORMAT	非保留	-	-
TIMEZONE_HOUR	-	保留	保留
TIMEZONE_MINUTE	-	保留	保留
TINYINT	非保留(不能是函数或类型)	-	-
TO	保留	保留	保留
TRAILING	保留	保留	保留
TRANSACTION	非保留	保留	保留
TRANSACTIONS_COMMITTED	-	非保留	-
TRANSACTIONS_ROLLED_BACK	-	非保留	-
TRANSACTION_ACTIVE	-	非保留	-
TRANSFORM	-	非保留	-
TRANSFORMS	-	非保留	-
TRANSLATE	-	非保留	保留
TRANSLATION	-	保留	保留
TREAT	非保留(不能是函数或类型)	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
TRIGGER	非保留	保留	-
TRIGGER_CATALOG	-	非保留	-
TRIGGER_NAME	-	非保留	-
TRIGGER_SCHEMA	-	非保留	-
TRIM	非保留(不能是函数或类型)	非保留	保留
TRUE	保留	保留	保留
TRUNCATE	非保留	-	-
TRUSTED	非保留	-	-
TRY_CAST	非保留	-	-
TSTAG	保留，该字段仅在IoT数仓中使用	-	-
TSTIME	保留，该字段仅在IoT数仓中使用	-	-
TSFIELD	保留，该字段仅在IoT数仓中使用	-	-
TYPE	非保留	非保留	非保留
TYPES	非保留	-	-
UESCAPE	-	-	-
UNBOUNDED	非保留	-	-
UNCOMMITTED	非保留	非保留	非保留
UNDER	-	保留	-
UNENCRYPTED	非保留	-	-
UNION	保留	保留	保留
UNIQUE	保留	保留	保留
UNKNOWN	非保留	保留	保留
UNLIMITED	非保留	-	-
UNLISTEN	非保留	-	-
UNLOCK	非保留	-	-
UNLOGGED	非保留	-	-
UNNAMED	-	非保留	非保留
UNNEST	-	保留	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
UNTIL	非保留	-	-
UNUSABLE	非保留	-	-
UPDATE	非保留	保留	保留
UPPER	-	非保留	保留
USAGE	-	保留	保留
USER	保留	保留	保留
USER_DEFINED_TYPE_CATALOG	-	非保留	-
USER_DEFINED_TYPE_NAME	-	非保留	-
USER_DEFINED_TYPE_SCHEMA	-	非保留	-
USING	保留	保留	保留
VACUUM	非保留	-	-
VALID	非保留	-	-
VALIDATE	非保留	-	-
VALIDATION	非保留	-	-
VALIDATOR	非保留	-	-
VALUE	非保留	保留	保留
VALUES	非保留(不能是函数或类型)	保留	保留
VARCHAR	非保留(不能是函数或类型)	保留	保留
VARCHAR2	非保留(不能是函数或类型)	-	-
VARIABLE	-	保留	-
VARIADIC	保留	-	-
VARYING	非保留	保留	保留
VCGROUP	非保留	-	-
VERBOSE	保留(可以是函数或类型)	-	-
VERIFY	非保留	-	-
VERSION	非保留	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
VIEW	非保留	保留	保留
VOLATILE	非保留	-	-
WARMUP	保留	-	-
WHEN	保留	保留	保留
WHENEVER	-	保留	保留
WHERE	保留	保留	保留
WHITESPACE	非保留	-	-
WINDOW	保留	-	-
WITH	保留	保留	保留
WITHIN	非保留	-	-
WITHOUT	非保留	保留	-
WORK	非保留	保留	保留
WORKLOAD	非保留	-	-
WRAPPER	非保留	-	-
WRITE	非保留	保留	保留
XML	非保留	-	-
XMLATTRIBUTES	非保留(不能是函数或类型)	-	-
XMLCONCAT	非保留(不能是函数或类型)	-	-
XMLELEMENT	非保留(不能是函数或类型)	-	-
XMLEXISTS	非保留(不能是函数或类型)	-	-
XMLFOREST	非保留(不能是函数或类型)	-	-
XMLNAMESPACES	非保留(不能是函数或类型)	-	-
XMLPARSE	非保留(不能是函数或类型)	-	-
XMLPI	非保留(不能是函数或类型)	-	-
XMLROOT	非保留(不能是函数或类型)	-	-

版权所有 © 华为云计算技术有限公司

关键字	DWS	SQL:1999	SQL:1992
XMLSERIALIZE	非保留(不能是函数或类型)	-	-
XMLTABLE	非保留(不能是函数或类型)	-	-
YEAR	非保留	保留	保留
YES	非保留	-	-
ZONE	非保留	保留	保留

21.5 数据类型

21.5.1 数值类型

数值类型也叫数字类型。由1、2、4或8字节的整数以及4或8字节的浮点数和可选精度小数组成。

对应的数字操作符和相关函数，请参见**数字操作函数和操作符**。

DWS支持的数值类型按精度可以分为：整数类型，任意精度型，浮点类型和序列整型。

整数类型

TINYINT、SMALLINT、INTEGER、BINARY_INTEGER和BIGINT类型存储整个数值（不带有小数部分），也就是整数。如果尝试存储超出范围以外的数值将会导致错误。

常用的类型是INTEGER，一般只有取值范围确定不超过SMALLINT的情况下，才会使用SMALLINT类型。而只有在INTEGER的范围不够的时候才使用BIGINT，因为前者相对快得多。

表 21-4 整数类型

名称	描述	存储空间	范围
TINYINT	微整数，别名为INT1。	1字节	0 ~ 255
SMALLINT	小范围整数，别名为INT2。	2字节	-32,768 ~ +32,767
INTEGER	常用的整数，别名为INT4。	4字节	-2,147,483,648 ~ +2,147,483,647
BINARY_INTEGER	常用的整数INTEGER的别名，为兼容Oracle类型。	4字节	-2,147,483,648 ~ +2,147,483,647

版权所有 © 华为云计算技术有限公司

名称	描述	存储空间	范围
BIGINT	大范围的整数，别名为INT8。	8字节	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

示例：

创建带有TINYINT、INTEGER、BIGINT类型数据的表。

```
CREATE TABLE int_type_t1
(
    a TINYINT,
    b TINYINT,
    c INTEGER,
    d BIGINT
);
```

插入数据。

```
INSERT INTO int_type_t1 VALUES(100, 10, 1000, 10000);
```

查看数据。

```
SELECT * FROM int_type_t1;
 a | b |  c   |   d
---+---+------+-------
 100 | 10 | 1000 | 10000
(1 row)
```

任意精度型

NUMBER类型能够用于存储对于精度位数没有限制的数字，并且可以用于执行精确计算。当要求高精确度时，推荐使用这种类型来存储货币总量和其他类型的数量值。与整数类型相比，任意精度类型需要更大的存储空间，其存储效率、运算效率以及压缩比效果都要差一些。

NUMBER类型数值的范围是小数点右边部分的小数位数。NUMBER类型数值的精度是指整个数值包含的所有数字，也就是小数点左右两边的所有数字。所以，可以说数值23.1234的精度为6，范围是4。可以认为整数的范围是0。

使用Numeric/Decimal进行列定义时，建议指定该列的精度p（总位数）以及范围s（小数位数）。

如果数值的精度或者范围大于列的数据类型所声明的精度和范围，那么系统将会试图对这个值进行四舍五入。如果不能对数值进行四舍五入的处理来满足数据类型的限制，则会报错。

表 21-5 任意精度型

名称	描述	存储空间	范围
NUMERIC[(p[,s])], DECIMAL[(p[,s])]	精度p取值范围为[1,1000]，标度s取值范围为[0,p]。	用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。	未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。

版权所有 © 华为云计算技术有限公司

名称	描述	存储空间	范围
NUMBER[(p[,s])]	NUMERIC类型的别名，为兼容Oracle数据类型。	用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。	未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。

示例：

创建带有DECIMAL数值类型的表。

```
CREATE TABLE decimal_type_t1 (DT_COL1 DECIMAL(10,4));
```

插入数据。

```
INSERT INTO decimal_type_t1 VALUES(123456.122331);
INSERT INTO decimal_type_t1 VALUES(123456.452399);
```

查看数据。

```
SELECT * FROM decimal_type_t1;
   dt_col1
-------------
 123456.1223
 123456.4524
(2 rows)
```

浮点类型

浮点类型属于非精确，可变精度的数值类型。实际上，这些类型通常是对于二进制浮点算术（分别是单精度和双精度）的IEEE标准754的具体实现，在一定范围内由特定的处理器，操作系统和编译器所支持。

表 21-6 浮点类型

名称	描述	存储空间	范围
REAL, FLOAT4	单精度浮点数，不精准。	4字节	6位十进制数字精度。
DOUBLE PRECISION, FLOAT8	双精度浮点数，不精准。	8字节	1E-307~1E+308， 15位十进制数字精度。
FLOAT[(p)]	浮点数，不精准。精度p取值范围为[1,53]。 **说明** p为精度，表示总位数。	4字节或8字节	根据精度p不同选择REAL或DOUBLE PRECISION作为内部表示。如不指定精度，内部用DOUBLE PRECISION表示。

版权所有 © 华为云计算技术有限公司

名称	描述	存储空间	范围
BINARY_DOUBLE	是DOUBLE PRECISION的别名，为兼容Oracle类型。	8字节	1E-307~1E+308， 15位十进制数字精度。
DEC[(p[,s])]	精度p取值范围为[1,1000]，标度s取值范围为[0,p]。 **说明** p为总位数，s为小数位位数。	用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。	未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。
INTEGER[(p[,s])]	精度p取值范围为[1,1000]，标度s取值范围为[0,p]。	用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。	未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。

示例：

创建带有浮点类型的表。

```
CREATE TABLE float_type_t2
(
    FT_COL1 INTEGER,
    FT_COL2 FLOAT4,
    FT_COL3 FLOAT8,
    FT_COL4 FLOAT(3),
    FT_COL5 BINARY_DOUBLE,
    FT_COL6 DECIMAL(10,4),
    FT_COL7 INTEGER(6,3)
) DISTRIBUTE BY HASH ( ft_col1);
```

插入数据。

```
INSERT INTO float_type_t2 VALUES(10,10.365456,123456.1234,10.3214, 321.321, 123.123654, 123.123654);
```

查看数据。

```
SELECT * FROM float_type_t2;
 ft_col1 | ft_col2 |   ft_col3   | ft_col4 | ft_col5 | ft_col6  | ft_col7
---------+---------+-------------+---------+---------+----------+---------
      10 | 10.3655 | 123456.1234 | 10.3214 | 321.321 | 123.1237 | 123.124
(1 row)
```

序列整型

SMALLSERIAL，SERIAL和BIGSERIAL类型不是真正的类型，只是为在表中设置唯一标识而存在的概念。因此，创建一个整数字段，并且把它的缺省数值安排为从一个序列发生器读取。应用了一个NOT NULL约束以确保NULL不会被插入。在大多数情况下用户可能还希望附加一个UNIQUE或PRIMARY KEY约束避免意外地插入重复的数值。最

 版权所有 © 华为云计算技术有限公司

后，将序列发生器从属于那个字段，这样当该字段或表被删除的时候也一并删除该序列。目前只支持在创建表时指定SERIAL列，不可以在已有的表中增加SERIAL列。另外临时表也不支持创建SERIAL列。因为SERIAL不是真正的类型，也不可以将表中存在的列类型转化为SERIAL。

表 21-7 序列整型

名称	描述	存储空间	范围
SMALLSERIAL	二字节序列整型。	2字节	1 ~ 32,767
SERIAL	四字节序列整型。	4字节	1 ~ 2,147,483,647
BIGSERIAL	八字节序列整型。	8字节	1 ~ 9,223,372,036,854,775,807

示例：

创建带有序列类型的表。

```
CREATE TABLE smallserial_type_tab(a SMALLSERIAL);
```

插入数据。

```
INSERT INTO smallserial_type_tab VALUES(default);
```

再次插入数据。

```
INSERT INTO smallserial_type_tab VALUES(default);
```

查看数据。

```
SELECT * FROM smallserial_type_tab;
 a
---
 1
 2
(2 rows)
```

插入NULL值会报错。

```
INSERT INTO smallserial_type_tab VALUES(NULL);
ERROR:  dn_6001_6002: null value in column "a" violates not-null constraint
```

21.5.2 货币类型

货币类型存储带有固定小数精度的货币金额。表21-8中显示的范围假设有两位小数。可以以任意格式输入，包括整型、浮点型或者典型的货币格式（如“$1,000.00”）。根据区域字符集，输出一般是最后一种形式。

表 21-8 货币类型

名字	存储容量	描述	范围
money	8 字节	货币金额	-92233720368547758.08 到 +92233720368547758.07

版权所有 © 华为云计算技术有限公司

numeric，int和bigint类型的值可以转化为money类型。如果从real和double precision类型转换到money类型，可以先转化为numeric类型，再转化为money类型，例如：

```
SELECT '12.34'::float8::numeric::money;
```

这种用法是不推荐使用的。浮点数不应该用来处理货币类型，因为小数点的位数可能会导致错误。

money类型的值可以转换为numeric类型而不丢失精度。转换为其他类型可能丢失精度，并且必须通过以下两步来完成：

```
SELECT '52093.89'::money::numeric::float8;
```

当一个money类型的值除以另一个money类型的值时，结果是double precision（也就是，一个纯数字，而不是money类型）；在运算过程中货币单位相互抵消。

21.5.3 布尔类型

表 21-9 布尔类型

名称	描述	存储空间	取值
BOOLEAN	布尔类型	1字节。	• true：真 • false：假 • null：未知（unknown）

“真”值的有效文本值是：

TRUE、't'、'true'、'y'、'yes'、'1'。

“假”值的有效文本值是：

FALSE、'f'、'false'、'n'、'no'、'0'。

使用TRUE和FALSE是比较规范的用法（也是SQL兼容的用法）。

示例

显示用字母t和f输出boolean值。

```
--创建表。
CREATE TABLE bool_type_t1
(
    BT_COL1 BOOLEAN,
    BT_COL2 TEXT
) DISTRIBUTE BY HASH(BT_COL2);

--插入数据。
INSERT INTO bool_type_t1 VALUES (TRUE, 'sic est');

INSERT INTO bool_type_t1 VALUES (FALSE, 'non est');

--查看数据。
SELECT * FROM bool_type_t1;
 bt_col1 | bt_col2
---------+---------
 t       | sic est
 f       | non est
(2 rows)
```


版权所有 © 华为云计算技术有限公司

```
SELECT * FROM bool_type_t1 WHERE bt_col1 = 't';
 bt_col1 | bt_col2
---------+---------
 t       | sic est
(1 row)

--删除表。
DROP TABLE bool_type_t1;
```

21.5.4 字符类型

DWS支持的字符类型请参见**表21-10**。字符串操作符和相关的内置函数请参见**字符串处理函数和操作符**。

表 21-10 字符类型

名称	描述	长度	存储空间
CHAR(n)或BPCHAR CHARACTER(n) NCHAR(n)	定长字符串，超出会截断，不足填充空格。	n是指字节长度，如不带精度n，默认精度为1。n小于10485761。	最大为10MB。
VARCHAR(n) CHARACTER VARYING(n)	变长字符串。	n是指字节长度，n小于10485761。	最大为10MB。
VARCHAR2(n)	变长字符串。是VARCHAR(n)类型的别名，为兼容Oracle类型。	n是指字节长度，n小于10485761。	最大为10MB。
NVARCHAR2(n)	变长字符串。	n是指字符长度，n小于10485761。	最大为10MB。
CLOB	变长字符串。文本大对象。是TEXT类型的别名，为兼容Oracle类型。	-	最大为1GB-8203B（即1073733621B）。
TEXT	变长字符串。	-	最大为1GB-8203B（即1073733621B）。

说明

- 除了每列的大小限制以外，每个元组的总大小也不可超过1GB-8023B（即1073733621B）。
- 对于字符串数据，建议使用变长字符串数据类型，并指定最大长度。请务必确保指定的最大长度大于需要存储的最大字符数，避免超出最大长度时出现字符截断现象。除非明确知道数据类型为固定长度字符串，否则，不建议使用CHAR(n)、NCHAR(n)、CHARACTER(n)等定长字符类型。在DWS中，定长字符类型的运算会存在额外的存储和内存开销。
- 一般情况下，如果含有中文字符，用NCHAR/NVARCHAR2，如果纯英文和数字，用CHAR/VARCHAR。

版权所有 © 华为云计算技术有限公司

在DWS里另外还有两种定长字符类型。在表21-11里显示。

其中，name类型只用在内部系统表中，作为存储标识符，该类型长度当前定为64字节（63可用字符加结束符）。不建议普通用户使用这种数据类型。name类型与其他数据类型进行对齐时（比如case when的多个分支中，其中一个分支返回name类型，其他类型返回text类型），可能会出现向name类型对齐，字符截断。如果不希望出现字符按照64位截断的情况，则需要将name类型强制类型转化为text类型。

类型"char"只用了一个字节的存储空间。它在系统内部主要用于系统表，主要作为简单化的枚举类型使用。

表 21-11 特殊字符类型

名称	描述	存储空间
name	用于对象名的内部类型。	64字节。
"char"	单字节内部类型。	1字节。

长度

如果把一个字段定义为char(n)或者varchar(n)， 代表该字段最大可容纳n个长度的数据。无论哪种类型，可设置的最大长度都不得超过10485760（即10MB）。

当数据长度超过指定的长度n时，会抛出错误"value too long"。也可通过指定数据类型，使超过长度的数据自动截断。

示例：

1. 创建表t1，指定其字段的字符类型。
```
CREATE TABLE t1 (a char(5),b varchar(5));
```
2. 向表t1插入数据时超过指定的字节长度报错。
```
INSERT INTO t1 VALUES('bookstore','123');
ERROR:  value too long for type character(5)
CONTEXT:  referenced column: a
```
3. 向表t1插入数据并明确超过指定字节长度后自动截断。
```
INSERT INTO t1 VALUES('bookstore'::char(5),'12345678'::varchar(5));
INSERT 0 1

SELECT a,b FROM t1;
   a   |   b
-------+-------
 books | 12345
(1 row)
```

定长与变长

所有字符类型根据长度是否固定可以分为定长字符串与变长字符串两大类。

- 对于定长字符串，长度必须确定，如果不指定长度，则默认长度1；如果数据长度不足，会在尾部自动填充空格，用于存储和显示；但这部分填充的数据是无意义的，实际使用中会被忽略，如比较、排序或类型转换。
- 对于变长字符串，若指定长度，则为最大可存储数据长度；如果不指定长度，则认为该字段支持任意长度。

示例：

 版权所有 © 华为云计算技术有限公司

1. 创建表t2，指定其字段的字符类型。
```
CREATE TABLE t2 (a char(5),b varchar(5));
```
2. 向表t2插入数据并查询字段a的字节长度。因建表时指定a的字符类型为char(5)且是定长字符串，长度不足，填充空格，所以查询的字节长度为5。
```
INSERT INTO t2 VALUES('abc','abc');
INSERT 0 1

SELECT a,lengthb(a),b FROM t2;
   a   | lengthb |  b
-------+---------+-----
 abc   |       5 | abc
(1 row)
```
3. 用函数转换后查询字段a的实际字节长度为3。
```
SELECT a = b from t2;
 ?column?
----------
 t
(1 row)

SELECT cast(a as text) as val,lengthb(val) FROM t2;
 val | lengthb
-----+---------
 abc |       3
(1 row)
```

字节数和字符数

VARCHAR2(n)和NVARCHAR2(n)中长度含义不同，需区别对待。

- VARCHAR2中n为字节长度。
- NVARCHAR2中n为字符长度。

  说明

  以UTF8编码的数据库为例，字母占1个字节，汉字占3个字节，VARCHAR2(6)可以存放6个字母或2个汉字，而NVARCHAR2(6)可以存放6个字母或6个汉字。

示例：

1. 创建表t3，指定其字段的字符类型。
```
CREATE TABLE t3 (a varchar2(6),b nvarchar2(6));
```
2. 向表t3插入数据，超出长度报错。（以UTF8字符编码的数据库中操作为例）
```
INSERT INTO t3 values('产品名','auto');
ERROR:  value too long for type character varying(6)
CONTEXT:  referenced column: a

INSERT INTO t3 values('auto','产品名abcd');
ERROR:  dn_6003_6004: value too long for type nvarchar2(6)
CONTEXT:  referenced column: b
```
3. 调整插入数据长度后向表t3插入数据成功。
```
INSERT INTO t3 values('产品','产品名abc');
INSERT 0 1

SELECT a,b from t3;
  a   |    b
------+----------
 产品 | 产品名abc
```

空串与 NULL

Oracle兼容模式下，不区分空串与NULL，执行语句查询或数据导入时会将空串处理为NULL。

版权所有 © 华为云计算技术有限公司

由于空串默认被处理为NULL，那就不能使用 = '' 作为查询条件，也不能用is ''。虽然不会有语法错误，但是不会有结果集返回。正确的用法是is null，不等于就是is not null 。

示例：

1. 创建表t4，指定其字段的字符类型。
```
CREATE TABLE t4 (a text);
```
2. 向表t4插入数据，插入值中包含空串和NULL。
```
INSERT INTO t4 VALUES('abc'),(''),(null);
INSERT 0 3
```
3. 查询表t4中是否存在空值。
```
SELECT a,a isnull FROM t4;
  a  | ?column?
-----+----------
     | t
     | t
 abc | f
(3 rows)

SELECT a,a isnull FROM t4 WHERE a is null;
 a | ?column?
---+----------
   | t
   | t
(2 rows)
```

TD与MySQL兼容模式下，区分空串与null。

- TD兼容模式下示例：
```
SELECT  '' is null , null is null;
 isnull |  isnull
--------+----------
 f      | t
(1 rows)
```
- MySQL兼容模式下示例：
```
SELECT  '' is null , null is null;
 isnull |  isnull
--------+----------
 f      | t
(1 rows)
```

21.5.5 二进制类型

DWS支持的二进制类型请参见**表21-12**。

 版权所有 © 华为云计算技术有限公司

表 21-12 二进制类型

名称	描述	存储空间
BLOB	二进制大对象 目前BLOB支持的外部存取接口仅为： ● DBMS_LOB.GETLENGTH ● DBMS_LOB.READ ● DBMS_LOB.WRITE ● DBMS_LOB.WRITEAPPEND ● DBMS_LOB.COPY ● DBMS_LOB.ERASE 这些接口详细说明请参见《数据仓库服务开发指南》中“DBMS_LOB”章节。 **说明** 列存不支持BLOB类型	最大为1G-8023B（即1073733621B）。
RAW	变长的十六进制类型 **说明** 列存不支持RAW类型	4字节加上实际的十六进制字符串。最大为1G-8023B（即1073733621B）。
BYTEA	变长的二进制字符串	4字节加上实际的二进制字符串。最大为1G-8023B（即1073733621B）。

说明

除了每列的大小限制以外，每个元组的总大小也不可超过1G-8203字节。

示例

```
--创建表。
CREATE TABLE blob_type_t1
(
    BT_COL1 INTEGER,
    BT_COL2 BLOB,
    BT_COL3 RAW,
    BT_COL4 BYTEA
) DISTRIBUTE BY REPLICATION;

--插入数据。
INSERT INTO blob_type_t1 VALUES(10,empty_blob(),
HEXTORAW('DEADBEEF'),E'\\xDEADBEEF');

--查询表中的数据。
SELECT * FROM blob_type_t1;
 bt_col1 | bt_col2 | bt_col3  |  bt_col4
---------+---------+----------+------------
      10 |         | DEADBEEF | \xdeadbeef
(1 row)

--删除表。
DROP TABLE blob_type_t1;
```


版权所有 © 华为云计算技术有限公司

21.5.6 日期/时间类型

DWS支持的日期/时间类型请参见**表21-13**。该类型的操作符和内置函数请参见**时间、日期处理函数和操作符**。

说明

如果其他的数据库时间格式和DWS的时间格式不一致，可通过修改配置参数DateStyle的值来保持一致。

表 21-13 日期/时间类型

名称	描述	存储空间
DATE	Oracle兼容模式下等价于timestamp(0)，记录日期和时间。 其他模式下，记录日期。	Oracle兼容模式下，占存储空间8字节 其他模式下，占存储空间4字节
TIME [(p)] [WITHOUT TIME ZONE]	只用于一日内时间。 p表示小数点后的精度，取值范围为0~6。	8字节
TIME [(p)] [WITH TIME ZONE]	只用于一日内时间，带时区。 p表示小数点后的精度，取值范围为0~6。	12字节
TIMESTAMP[(p)] [WITHOUT TIME ZONE]	日期和时间。 p表示小数点后的精度，取值范围为0~6。	8字节
TIMESTAMP[(p)] [WITH TIME ZONE]	日期和时间，带时区。TIMESTAMP的别名为TIMESTAMPTZ。 p表示小数点后的精度，取值范围为0~6。	8字节
SMALLDATETIME	日期和时间，不带时区。 精确到分钟，秒位大于等于30秒进一位。	8字节
INTERVAL DAY (l) TO SECOND (p)	时间间隔，X天X小时X分X秒。 • l：天数的精度，取值范围为0~6。为适配Oracle语法，未实现具体功能。 • p：秒数的精度，取值范围为0~6。小数末尾的零不显示。	16字节

版权所有 © 华为云计算技术有限公司

名称	描述	存储空间
INTERVAL [FIELDS] [(p)]	时间间隔。 • fields：可以是YEAR，MONTH，DAY，HOUR，MINUTE，SECOND，DAY TO HOUR，DAY TO MINUTE，DAY TO SECOND，HOUR TO MINUTE，HOUR TO SECOND，MINUTE TO SECOND。 • p：秒数的精度，取值范围为0~6，且fields为SECOND，DAY TO SECOND，HOUR TO SECOND或MINUTE TO SECOND时，参数p才有效。小数末尾的零不显示。 **说明** 9.1.0.200及以上集群版本的MySQL兼容性模式下，当fields为YEAR，MONTH，DAY，HOUR，MINUTE，SECOND时，入参可以为不含正负号的整数或浮点数，例如INTERVAL 1 YEAR, INTERVAL 2.5 MONTH等。	12字节
reltime	相对时间间隔。格式为： X years X mons X days XX:XX:XX。 • 采用儒略历计时，规定一年为365.25天，一个月为30天，计算输入值对应的相对时间间隔，输出采用POSTGRES格式。	4字节

示例：

```
--创建表。
CREATE TABLE date_type_tab(coll date);

--插入数据。
INSERT INTO date_type_tab VALUES (date '12-10-2010');

--查看数据。
SELECT * FROM date_type_tab;
        coll
---------------------
 2010-12-10 00:00:00
(1 row)

--删除表。
DROP TABLE date_type_tab;

--创建表。
CREATE TABLE time_type_tab (da time without time zone ,dai time with time zone,dfgh timestamp without time zone,dfga timestamp with time zone, vbg smalldatetime);

--插入数据。
INSERT INTO time_type_tab VALUES ('21:21:21','21:21:21 pst','2010-12-12','2013-12-11 pst','2003-04-12 04:05:06');
```


版权所有 © 华为云计算技术有限公司

```
--查看数据。
SELECT * FROM time_type_tab;
    da    |     dai     |        dfgh         |          dfga          |         vbg
----------+-------------+---------------------+------------------------+---------------------
 21:21:21 | 21:21:21-08 | 2010-12-12 00:00:00 | 2013-12-11 16:00:00+08 | 2003-04-12 04:05:00
(1 row)

--删除表。
DROP TABLE time_type_tab;

--创建表。
CREATE TABLE day_type_tab (a int,b INTERVAL DAY(3) TO SECOND (4));

--插入数据。
INSERT INTO day_type_tab VALUES (1, INTERVAL '3' DAY);

--查看数据。
SELECT * FROM day_type_tab;
 a |    b
---+---------
 1 | 3 days
(1 row)

--删除表。
DROP TABLE day_type_tab;

--创建表。
CREATE TABLE year_type_tab(a int, b interval year (6));

--插入数据。
INSERT INTO year_type_tab VALUES(1,interval '2' year);

--查看数据。
SELECT * FROM year_type_tab;
 a |     b
---+----------
 1 | 2 years
(1 row)

--删除表。
DROP TABLE year_type_tab;
```

日期输入

日期和时间的输入几乎可以是任何合理的格式，包括ISO-8601格式、SQL-兼容格式、传统POSTGRES格式或者其它的形式。系统支持按照日、月、年的顺序自定义日期输入。如果把DateStyle参数设置为MDY就按照“月-日-年”解析，设置为DMY就按照“日-月-年”解析，设置为YMD就按照“年-月-日”解析。

日期的文本输入需要加单引号包围，语法如下：

type [(p)] 'value'

可选的精度声明中的p是一个整数，表示在秒域中小数部分的位数。**表21-14**显示了date类型的输入方式。

表 21-14 日期输入方式

例子	描述
1999-01-08	ISO 8601格式（建议格式），任何方式下都是1999年1月8号。

版权所有 © 华为云计算技术有限公司

例子	描述
January 8, 1999	在任何datestyle输入模式下都无歧义。
1/8/1999	有歧义，在MDY模式下是一月八号，在DMY模式下是八月一号。
1/18/1999	MDY模式下是一月十八日，其它模式下被拒绝。
01/02/03	• MDY模式下的2003年1月2日。 • DMY模式下的2003年2月1日。 • YMD模式下的2001年2月3日。
1999-Jan-08	任何模式下都是1月8日。
Jan-08-1999	任何模式下都是1月8日。
08-Jan-1999	任何模式下都是1月8日。
99-Jan-08	YMD模式下是1月8日，否则错误。
08-Jan-99	一月八日，除了在YMD模式下是错误的之外。
Jan-08-99	一月八日，除了在YMD模式下是错误的之外。
19990108	ISO 8601；任何模式下都是1999年1月8日。
990108	ISO 8601；任何模式下都是1999年1月8日。
1999.008	年和年里的第几天。
J2451187	儒略日。
January 8, 99 BC	公元前99年。

示例：

```
--创建表。
CREATE TABLE date_type_tab(coll date);

--插入数据。
INSERT INTO date_type_tab VALUES (date '12-10-2010');

--查看数据。
SELECT * FROM date_type_tab;
        coll
---------------------
 2010-12-10 00:00:00
(1 row)

--查看日期格式。
SHOW datestyle;
 DateStyle
-----------
 ISO, MDY
(1 row)

--设置日期格式。
SET datestyle='YMD';
SET
```


版权所有 © 华为云计算技术有限公司

```
--插入数据。
INSERT INTO date_type_tab VALUES(date '2010-12-11');

--查看数据。
SELECT * FROM date_type_tab;
        coll
---------------------
 2010-12-10 00:00:00
 2010-12-11 00:00:00
(2 rows)

--删除表。
DROP TABLE date_type_tab;
```

时间

时间类型包括time [(p)] without time zone和time [(p)] with time zone。如果只写time等效于time without time zone。

如果在time without time zone类型的输入中声明了时区，则会忽略这个时区。

时间输入类型的详细信息请参见**表21-15**，时区输入类型的详细信息请参见**表21-16**。

表 21-15 时间输入

例子	描述
05:06.8	ISO 8601
4:05:06	ISO 8601
4:05	ISO 8601
40506	ISO 8601
4:05 AM	与04:05一样，AM不影响数值
4:05 PM	与16:05一样，输入小时数必须<= 12
04:05:06.789-8	ISO 8601
04:05:06-08:00	ISO 8601
04:05-08:00	ISO 8601
040506-08	ISO 8601
04:05:06 PST	缩写的时区

表 21-16 时区输入

例子	描述
PST	太平洋标准时间（Pacific Standard Time）
-8:00	ISO 8601与PST的偏移
-800	ISO 8601与PST的偏移

版权所有 © 华为云计算技术有限公司

例子	描述
-8	ISO 8601与PST的偏移

示例：

```
SELECT time '04:05:06';
   time
----------
 04:05:06
(1 row)

SELECT time '04:05:06 PST';
   time
----------
 04:05:06
(1 row)

SELECT time with time zone '04:05:06 PST';
   timetz
-------------
 04:05:06-08
(1 row)
```

特殊值

DWS支持几个特殊值，在读取的时候将被转换成普通的日期/时间值，请参考**表 21-17**。

表 21-17 特殊值

输入字符串	适用类型	描述
epoch	date，timestamp	1970-01-01 00:00:00+00 （Unix系统零时）
infinity	timestamp	比任何其他时间戳都晚
-infinity	timestamp	比任何其他时间戳都早
now	date，time，timestamp	当前事务的开始时间
today	date，timestamp	今日午夜
tomorrow	date，timestamp	明日午夜
yesterday	date，timestamp	昨日午夜
allballs	time	00:00:00.00 UTC

时间段输入

reltime的输入方式可以采用任何合法的时间段文本格式，包括数字形式（含负数和小数）及时间形式，其中时间形式的输入支持SQL标准格式、ISO-8601格式、POSTGRES格式等。另外，文本输入需要加单引号。

版权所有 © 华为云计算技术有限公司

时间段输入的详细信息请参考**表6 时间段输入**。

表 21-18 时间段输入

输入示例	输出结果	描述
60	2 mons	采用数字表示时间段，默认单位是day，可以是小数或负数。特别的，负数时间段，在语义上，可以理解为“早于多久”。
31.25	1 mons 1 days 06:00:00	
-365	-12 mons -5 days	
1 years 1 mons 8 days 12:00:00	1 years 1 mons 8 days 12:00:00	采用POSTGRES格式表示时间段，可以正负混用，不区分大小写，输出结果为将输入时间段计算并转换得到的简化POSTGRES格式时间段。
-13 months -10 hours	-1 years -25 days -04:00:00	
-2 YEARS +5 MONTHS 10 DAYS	-1 years -6 mons -25 days -06:00:00	
P-1.1Y10M	-3 mons -5 days -06:00:00	采用ISO-8601格式表示时间段，可以正负混用，不区分大小写，输出结果为将输入时间段计算并转换得到的简化POSTGRES格式时间段。
-12H	-12:00:00	

示例：

```
--创建表。
CREATE TABLE reltime_type_tab(col1 character(30), col2 reltime);

--插入数据。
INSERT INTO reltime_type_tab VALUES ('90', '90');
INSERT INTO reltime_type_tab VALUES ('-366', '-366');
INSERT INTO reltime_type_tab VALUES ('1975.25', '1975.25');
INSERT INTO reltime_type_tab VALUES ('-2 YEARS +5 MONTHS 10 DAYS', '-2 YEARS +5 MONTHS 10 DAYS');
INSERT INTO reltime_type_tab VALUES ('30 DAYS 12:00:00', '30 DAYS 12:00:00');
INSERT INTO reltime_type_tab VALUES ('P-1.1Y10M', 'P-1.1Y10M');

--查看数据。
SELECT * FROM reltime_type_tab;
             col1              |               col2
-------------------------------+-----------------------------------
 1975.25                       | 5 years 4 mons 29 days
 -2 YEARS +5 MONTHS 10 DAYS    | -1 years -6 mons -25 days -06:00:00
 P-1.1Y10M                     | -3 mons -5 days -06:00:00
 -366                          | -1 years -18:00:00
 90                            | 3 mons
 30 DAYS 12:00:00              | 1 mon 12:00:00
(6 rows)

--删除表。
DROP TABLE reltime_type_tab;
```

21.5.7 几何类型

DWS支持的几何类型请参见**表21-19**。最基本的类型：点，是其它类型的基础。

 版权所有 © 华为云计算技术有限公司

表 21-19 几何类型

名字	存储空间	说明	表现形式
point	16字节	平面中的点	(x,y)
lseg	32字节	（有限）线段	((x1,y1),(x2,y2))
box	32字节	矩形	((x1,y1),(x2,y2))
path	16+16n字节	闭合路径（与多边形类似）	((x1,y1),...)
path	16+16n字节	开放路径	[(x1,y1),...]
polygon	40+16n字节	多边形（与闭合路径相似）	((x1,y1),...)
circle	24 字节	圆	<(x,y),r>（圆心和半径）

DWS提供了一系列的函数和操作符用来进行各种几何计算，如拉伸、转换、旋转、计算相交等。详细信息请参考**几何函数和操作符**。

点

点是几何类型的基本二维构造单位。用下面语法描述point的数值：

```
( x , y )
x , y
```

x和y是用浮点数表示的点的坐标。

点输出使用第一种语法。

线段

线段（lseg）是用一对点来代表的。用下面的语法描述lseg的数值：

```
[ ( x1 , y1 ) , ( x2 , y2 ) ]
( ( x1 , y1 ) , ( x2 , y2 ) )
( x1 , y1 ) , ( x2 , y2 )
x1 , y1   ,   x2 , y2
```

(x1,y1)和(x2,y2)表示线段的端点。

线段输出使用第一种语法。

矩形

矩形是用一对对角点来表示的。用下面的语法描述box的值：

```
( ( x1 , y1 ) , ( x2 , y2 ) )
( x1 , y1 ) , ( x2 , y2 )
x1 , y1   ,   x2 , y2
```

(x1,y1)和(x2,y2)表示矩形的一对对角点。

矩形的输出使用第二种语法。

任何两个对角都可以出现在输入中，但按照那样的顺序，右上角和左下角的值会被重新排序以存储。

 版权所有 © 华为云计算技术有限公司

路径

路径由一系列连接的点组成。路径可能是开放的，也就是认为列表中第一个点和最后一个点没有连接，也可能是闭合的，这时认为第一个和最后一个点连接起来。

用下面的语法描述path的数值：

```
[ ( x1 , y1 ) , ... , ( xn , yn ) ]
( ( x1 , y1 ) , ... , ( xn , yn ) )
( x1 , y1 ) , ... , ( xn , yn )
( x1 , y1   , ... ,  xn , yn )
x1 , y1   , ... ,  xn , yn
```

点表示组成路径的线段的端点。方括弧（ [] ）表明一个开放的路径，圆括弧（ () ）表明一个闭合的路径。当最外层的括号被省略，如在第三至第五语法，会假定一个封闭的路径。

路径的输出使用第一种或第二种语法输出。

多边形

多边形由一系列点代表（多边形的顶点）。多边形可以认为与闭合路径一样，但是存储方式不一样而且有自己的一套支持函数。

用下面的语法描述polygon的数值：

```
( ( x1 , y1 ) , ... , ( xn , yn ) )
( x1 , y1 ) , ... , ( xn , yn )
( x1 , y1   , ... ,  xn , yn )
x1 , y1   , ... ,  xn , yn
```

点表示多边形的端点。

多边形输出使用第一种语法。

圆

圆由一个圆心和半径标识。用下面的语法描述circle的数值：

```
< ( x , y ) , r >
( ( x , y ) , r )
( x , y ) , r
x , y   , r
```

(x,y)表示圆心，r表示半径。

圆的输出用第一种格式。

21.5.8 数组类型

数组是一组数据的集合。数组类型允许在单个数据库字段中存储多个值。数组类型通常用于存储和处理具有相似属性的数据。

语法格式

```
ARRAY [ param ]
```

或

```
'{ param }'
```

其中参数param说明如下：

版权所有 © 华为云计算技术有限公司

- param ： 数组包含的值，允许出现零个或多个，多个值之间用逗号分隔，没有值可填写为NULL。
- 以'{ param }' 这种格式作为数组常量时，其中的字符串类型的元素不能再以单引号开始和结束，需要使用双引号。两个连续单引号转换为一个单引号。
- 以第一个元素的数据类型作为数组的数据类型，要求数组中所有元素的类型相同，或者能够相互转换。

数组类型的定义

一个数组数据类型可以通过在数组元素的数据类型名称后面加上方括号（ [] ）来命名。

例如，创建表books，其中表示书本价格的列price的类型为一维integer类型数组，表示书本标签的列tag的类型为二维text类型数组。

```
CREATE TABLE books (id SERIAL PRIMARY KEY, title VARCHAR(100), price_by_quarter int[], tags TEXT[][]);
```

CREATE TABLE语法可以指定数组的大小，例如：

```
CREATE TABLE test ( a int[3]);
```

当前的数据库实现会忽略语句中数组的大小限制，即其行为与未指定长度的数组相同。同时，也不会强制所声明的维度数。一个特定元素类型的数组全部被当作是相同的类型，而忽略其大小或维度数。

也可以使用关键词ARRAY来定义一维数组。表books中的列price使用ARRAY定义并指定数组大小，如下所示：

```
price_by_quarter int ARRAY[4]
```

使用ARRAY定义，不指定数组尺寸：

```
price_by_quarter int ARRAY
```

数组值输入

输入数组值时要把一个数组值写成一个文字常数，将元素值用花括号包围并用逗号分隔。一个数组常量的一般格式如下：

```
'{ val1 delim val2 delim ... }'
```

其中，delim是类型的定界符，每个val可以是数组元素类型的一个常量或子数组。

一个数组常量的例子如下：

```
'{{1,2,3},{4,5,6},{7,8,9}}'
```

该常量是一个二维的，3乘3数组，它由3个整数子数组构成。

向表books插入数据并查询表books：

```
INSERT INTO books
  VALUES (1, 'One Hundred years of Solitude','{25,25,25,25}','{{"fiction"}, {"adventure"}}'),
         (2, 'Robinson Crusoe', '{30,32,32,32}', '{{"adventure"}, {"fiction"}}'),
         (3, 'Gone with the Wind', '{27,27,29,28}', '{{"romance"}, {"fantasy"}}');

SELECT * FROM books;
 id |            title            | price_by_quarter |          tags
----+-----------------------------+------------------+-------------------------
  1 | One Hundred years of Solitude | {25,25,25,25}    | {{fiction},{adventure}}
  2 | Robinson Crusoe               | {30,32,32,32}    | {{adventure},{fiction}}
  3 | Gone with the Wind            | {27,27,29,28}    | {{romance},{fantasy}}
(3 rows)
```

 版权所有 © 华为云计算技术有限公司

> **注意**
>
> 插入多维数组数据时，多维数组的每一维都必须有相匹配的长度。

使用ARRAY关键字插入数据：

```
INSERT INTO books
 VALUES (1, 'One Hundred years of Solitude',ARRAY[25,25,25,25],ARRAY['fiction', 'adventure']),
        (2, 'Robinson Crusoe', ARRAY[30,32,32,32], ARRAY['adventure', 'fiction']),
        (3, 'Gone with the Wind', ARRAY[27,27,29,28], ARRAY['romance', 'fantasy']);
```

访问数组

访问数组元素

查询在第二季度价格发生变化的书籍的名称：

```
SELECT title FROM books WHERE price_by_quarter[1] <> price_by_quarter[2];
      title
-----------------
 Robinson Crusoe
(1 row)
```

查询检索所有书籍第三季度的价格：

```
SELECT price_by_quarter[3] FROM books;
 price_by_quarter
------------------
               29
               25
               32
(3 rows)
```

访问数组切片

可以访问一个数组的任意矩形切片或者子数组。一个数组切片可以通过在一个或多个数组维度上指定[下界:上界]来定义。

查询Gone with the Wind的第二个标签：

```
SELECT tags[2:2] FROM books WHERE title = 'Gone with the Wind';
   tags
-----------
 {fantasy}
(1 row)
```

使用函数访问数组

使用array_dims函数获得数组值的当前维度：

```
SELECT array_dims(tags) FROM books WHERE title = 'Robinson Crusoe';
 array_dims
------------
 [1:2]
(1 row)
```

也可以使用array_upper和array_lower来获得数组维度，它们将分别返回一个指定数组的上界和下界：

```
SELECT array_upper(tags, 1) FROM books WHERE title = 'Robinson Crusoe';
 array_upper
-------------
           2
(1 row)
```

array_length函数将返回一个指定数组维度的长度：

版权所有 © 华为云计算技术有限公司

```
SELECT array_length(tags, 1) FROM books WHERE title = 'Robinson Crusoe';
 array_length
--------------
            2
(1 row)
```

修改数组

更新数组

更新整个数组数据：

```
UPDATE books SET price_by_quarter = '{30,30,30,30}'
    WHERE title = 'Robinson Crusoe';
```

使用ARRAY表达式语法更新整个数组数据：

```
UPDATE books SET price_by_quarter = ARRAY[30,30,30,30]
    WHERE title = 'Robinson Crusoe';
```

更新数组中的一个元素：

```
UPDATE books SET price_by_quarter[4] = 35
    WHERE title = 'Robinson Crusoe';
```

更新数组中的一个切片元素：

```
UPDATE books SET price_by_quarter[1:2] = '{27,27}'
    WHERE title = 'Robinson Crusoe';
```

一个已存储的数组值可以被通过对其还不存在的元素赋值来扩大大小。任何位于已存在的元素和新元素之间的位置都将被空值填充。例如，如果数组myarray目前有4个元素，使用UPDATE对myarray[6]赋值后它将有6个元素，其中myarray[5]为空值。目前，采用这种方式扩大数组只允许使用在一维数组上。

构建新数组

新的数组值也可以通过串接操作符“||”构建。串接操作符允许把一个单独的元素加入到一个一维数组的开头或末尾。也可接受两个N维数组，或者一个N维数组和一个N+1维数组。

```
SELECT ARRAY[1,2] || ARRAY[3,4];
 ?column?
-----------
 {1,2,3,4}
(1 row)

SELECT ARRAY[5,6] || ARRAY[[1,2],[3,4]];
     ?column?
---------------------
 {{5,6},{1,2},{3,4}}
(1 row)
```

使用函数array_prepend、array_append或array_cat构建数组。

```
SELECT array_prepend(1, ARRAY[2,3]);
 array_prepend
---------------
 {1,2,3}
(1 row)

SELECT array_append(ARRAY[1,2], 3);
 array_append
--------------
 {1,2,3}
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
SELECT array_cat(ARRAY[1,2], ARRAY[3,4]);
 array_cat
-----------
 {1,2,3,4}
(1 row)

SELECT array_cat(ARRAY[[1,2],[3,4]], ARRAY[5,6]);

     array_cat
---------------------
 {{1,2},{3,4},{5,6}}
(1 row)

SELECT array_cat(ARRAY[5,6], ARRAY[[1,2],[3,4]]);
     array_cat
---------------------
 {{5,6},{1,2},{3,4}}
(1 row)
```

21.5.9 枚举类型

枚举（enum）类型是由一个静态、值的有序集合构成的数据类型。它们等效于很多编程语言所支持的enum类型。枚举类型可以是一周中的日期，或者一个数据的状态值集合。

枚举类型的声明

枚举类型可以使用CREATE TYPE命令创建，例如：

```
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
```

枚举类型被创建后，可以在表和函数定义中使用：

```
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
CREATE TABLE person (name text, current_mood mood);
INSERT INTO person VALUES ('Moe', 'happy');
SELECT * FROM person WHERE current_mood = 'happy';
 name | current_mood
------+--------------
 Moe  | happy
(1 row)
```

排序

枚举类型值的排序是该类型被创建时所列出的值的顺序。

```
INSERT INTO person VALUES ('Larry', 'sad');
INSERT INTO person VALUES ('Curly', 'ok');

SELECT * FROM person WHERE current_mood > 'sad';
 name  | current_mood
-------+--------------
 Moe   | happy
 Curly | ok
(2 rows)

SELECT * FROM person WHERE current_mood > 'sad' ORDER BY current_mood;
 name  | current_mood
-------+--------------
 Curly | ok
 Moe   | happy
(2 rows)

SELECT name FROM person WHERE current_mood = (SELECT MIN(current_mood) FROM person);
 name
-------
```


版权所有 © 华为云计算技术有限公司

```
 Larry
(1 row)
```

枚举类型的安全性

每一种枚举数据类型都是独立的并且不能和其他枚举类型相比较。

```
CREATE TYPE happiness AS ENUM ('happy', 'very happy', 'ecstatic');
CREATE TABLE holidays (num_weeks integer, happiness happiness);
INSERT INTO holidays(num_weeks,happiness) VALUES (4, 'happy');
INSERT INTO holidays(num_weeks,happiness) VALUES (6, 'very happy');
INSERT INTO holidays(num_weeks,happiness) VALUES (8, 'ecstatic');

INSERT INTO holidays(num_weeks,happiness) VALUES (2, 'sad');
ERROR:  invalid input value for enum happiness: "sad"

SELECT person.name, holidays.num_weeks FROM person, holidays
      WHERE person.current_mood = holidays.happiness;
ERROR:  operator does not exist: mood = happiness
```

如果需要作比较，可以使用自定义的操作符或者在查询中加上显式类型：

```
 SELECT person.name, holidays.num_weeks FROM person, holidays
      WHERE person.current_mood::text = holidays.happiness::text;
 name | num_weeks
------+-----------
 Moe  |         4
(1 row)
```

注意事项

- 枚举标签是大小写敏感的，因此'happy'与'HAPPY'是不同的。标签中的空格也是有意义的。
- 尽管枚举类型的主要目的是用于值的静态集合，但也有方法在现有枚举类型中增加新值和重命名值（见**ALTER TYPE**）。不能从枚举类型中去除现有的值，也不能更改这些值的排序顺序，除非删除并且重建枚举类型。
- 从内部枚举值到文本标签的翻译保存在系统表PG_ENUM中。

21.5.10 网络地址类型

DWS提供用于存储IPv4、IPv6、MAC地址的数据类型。

网络地址类型提供输入错误检查和特殊的操作和功能（请参见**网络地址函数和操作符**），比纯文本类型更适合存储IPv4、IPv6、MAC地址的数据类型。

表 21-20 网络地址类型

名字	存储空间	描述
cidr	7或19字节	IPv4或IPv6网络
inet	7或19字节	IPv4或IPv6主机和网络
macaddr	6字节	MAC地址

在对inet或cidr数据类型进行排序的时候，IPv4地址总是排在IPv6地址前面，包括那些封装或者是映射在IPv6地址里的IPv4地址，例如::10.2.3.4或::ffff:10.4.3.2。

版权所有 © 华为云计算技术有限公司

cidr

cidr（无类别域间路由，Classless Inter-Domain Routing）类型，保存一个IPv4或IPv6网络地址。声明网络格式为address/y，address表示IPv4或者IPv6地址，y表示子网掩码的二进制位数。如果省略y，则掩码部分使用已有类别的网络编号系统进行计算，但要求输入的数据已经包括了确定掩码所需的所有字节。

- 示例一：CIDR格式换算为IP地址网段

  例如，10.0.0.0/8换算为32位二进制地址：00001010.00000000.00000000.00000000。其中/8表示8位网络ID，即32位二进制地址中前8位是固定不变的，对应网段为：00001010.00000000.00000000.00000000~00001010.11111111.11111111.11111111。则换算为十进制后，10.0.0.0/8表示：子网掩码为255.0.0.0，对应网段为10.0.0.0~10.255.255.255。

- 示例二：IP地址网段换算为CIDR格式

  例如，192.168.0.0~192.168.31.255，后两段IP换算为二进制地址：00000000.00000000~00011111.11111111，可以得出前19位（8*2+3）是固定不变的，则换算为CIDR格式后，表示为：192.168.0.0/19。

表 21-21 cidr 类型输入举例

cidr输入	cidr输出	abbrev（cidr）
192.168.100.128/25	192.168.100.128/25	192.168.100.128/25
192.168/24	192.168.0.0/24	192.168.0/24
192.168/25	192.168.0.0/25	192.168.0.0/25
192.168.1	192.168.1.0/24	192.168.1/24
192.168	192.168.0.0/24	192.168.0/24
10.1.2	10.1.2.0/24	10.1.2/24
10.1	10.1.0.0/16	10.1/16
10	10.0.0.0/8	10/8
10.1.2.3/32	10.1.2.3/32	10.1.2.3/32
2001:4f8:3:ba::/64	2001:4f8:3:ba::/64	2001:4f8:3:ba::/64
2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128	2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128	2001:4f8:3:ba:2e0:81ff:fe22:d1f1

inet

inet类型在一个数据区域内保存主机的IPv4或IPv6地址，以及一个可选子网。主机地址中网络地址的位数表示子网（“子网掩码”）。如果子网掩码是32并且地址是IPv4，则这个值不表示任何子网，只表示一台主机。在IPv6里，地址长度是128位，因此128位表示唯一的主机地址。

 版权所有 © 华为云计算技术有限公司

该类型的输入格式是address/y，address表示IPv4或者IPv6地址，y是子网掩码的二进制位数。如果省略/y，则子网掩码对IPv4是32，对IPv6是128，所以该值表示只有一台主机。如果该值表示只有一台主机，/y将不会显示。

inet和cidr类型之间的基本区别是inet接受子网掩码，而cidr不接受。

macaddr

macaddr类型存储MAC地址，也就是以太网卡硬件地址（尽管MAC地址还用于其它用途）。可以接受下列格式：

```
'08:00:2b:01:02:03'
'08-00-2b-01-02-03'
'08002b:010203'
'08002b-010203'
'0800.2b01.0203'
'08002b010203'
```

这些示例都表示同一个地址。对于数据位a到f，大小写都行。输出时都是以第一种形式展示。

21.5.11 位串类型

位串就是一串1和0的字符串。它们可以用于存储位掩码。

DWS支持两种位串类型：bit(n)和bit varying(n)，其中n是一个正整数。

bit类型的数据必须准确匹配长度n，如果存储短或者长的数据都会报错。bit varying类型的数据是最长为n的变长类型，超过n的类型会被拒绝。一个没有长度的bit等效于bit(1)，没有长度的bit varying表示没有长度限制。

说明

如果显式地把一个位串值转换成bit(n)，则此位串右边的内容将被截断或者在右边补齐零，直到刚好n位，而且不会抛出任何错误。类似地，如果显式地把一个位串数值转换成bit varying(n)，如果它超过了n位，则它的右边将被截断。

位串类型使用示例：

1. 创建示例表bit_type_t1：

```
CREATE TABLE bit_type_t1
(
    BT_COL1 INTEGER,
    BT_COL2 BIT(3),
    BT_COL3 BIT VARYING(5)
) DISTRIBUTE BY REPLICATION;
```

2. 插入数据：

```
INSERT INTO bit_type_t1 VALUES(1, B'101', B'00');
```

插入数据的长度不符合类型的标准会报错。

```
INSERT INTO bit_type_t1 VALUES(2, B'10', B'101');
ERROR:  bit string length 2 does not match type bit(3)
CONTEXT:  referenced column: bt_col2
```

3. 将不符合类型长度的数据进行转换：

```
INSERT INTO bit_type_t1 VALUES(2, B'10'::bit(3), B'101');
```

4. 查看数据：

```
SELECT * FROM bit_type_t1;
 bt_col1 | bt_col2 | bt_col3
---------+---------+---------
       1 | 101     | 00
       2 | 100     | 101
(2 rows)
```


版权所有 © 华为云计算技术有限公司

21.5.12 文本搜索类型

DWS提供了tsvector和tsquery两种数据类型用于支持全文检索。tsvector类型表示为文本搜索优化的文件格式，tsquery类型表示文本查询。

tsvector

tsvector类型表示一个检索单元，通常是一个数据库表中的一行文本字段或者这些字段的组合。

tsvector类型的值是唯一分词的分类列表，把一句话的词格式化为不同的词条，在进行分词处理的时候tsvector会按照一定的顺序录入，并自动去掉分词中重复的词条。

to_tsvector函数通常用于解析和标准化文档字符串。

通过tsvector把一个字符串按照空格进行分词，分词的顺序是按照字母和长短排序的，请看以下例子：

```
SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector;
                      tsvector
-------------------------------------------------------
 'a' 'and' 'ate' 'cat' 'fat' 'mat' 'on' 'rat' 'sat'
(1 row)
```

如果词条中包含空格或标点符号，可以用引号包围：

```
SELECT $$the lexeme '    ' contains spaces$$::tsvector;
                 tsvector
-------------------------------------------
 '    ' 'contains' 'lexeme' 'spaces' 'the'
(1 row)
```

使用常规的单引号引起来的字符串，字符串中嵌入的单引号(')和反斜杠(\)必须双写进行转义：

```
SELECT $$the lexeme 'Joe''s' contains a quote$$::tsvector;
                    tsvector
-------------------------------------------------
 'Joe''s' 'a' 'contains' 'lexeme' 'quote' 'the'
(1 row)
```

词条位置常量也可以放到词汇中：

```
SELECT 'a:1 fat:2 cat:3 sat:4 on:5 a:6 mat:7 and:8 ate:9 a:10 fat:11 rat:12'::tsvector;
                                  tsvector
-------------------------------------------------------------------------------
 'a':1,6,10 'and':8 'ate':9 'cat':3 'fat':2,11 'mat':7 'on':5 'rat':12 'sat':4
(1 row)
```

位置常量通常表示文档中源字的位置。位置信息可以用于进行排名。位置常量的范围是1到16383，最大值默认是16383。相同词的重复位会被忽略掉。

拥有位置的词汇可以进一步地被标注一个权重，它可以是A，B，C或D。 D是默认权重，因此输出中不会显示：

```
SELECT 'a:1A fat:2B,4C cat:5D'::tsvector;
          tsvector
----------------------------
 'a':1A 'cat':5 'fat':2B,4C
(1 row)
```

权重通常被用来反映文档结构，如：将标题标记成与正文词不同。文本检索排序函数可以为不同的权重标记分配不同的优先级。

tsvector类型标准用法示例如下：

版权所有 © 华为云计算技术有限公司

```
SELECT 'The Fat Rats'::tsvector;
      tsvector
--------------------
 'Fat' 'Rats' 'The'
(1 row)
```

但是对于英文全文检索应用来说，上面的单词会被认为非规范化的，所以需要通过to_tsvector函数对这些单词进行规范化处理：

```
SELECT to_tsvector('english', 'The Fat Rats');
   to_tsvector
-----------------
 'fat':2 'rat':3
(1 row)
```

tsquery

tsquery类型表示一个检索条件，存储用于检索的词汇，并且使用布尔操作符&（AND），|（OR）和!（NOT）将这些词汇进行组合，圆括号用来强调操作符的分组。如果没有圆括号，（NOT）的优先级最高，其次是&（AND），最后是|（OR）。to_tsquery函数及plainto_tsquery函数会将单词转换为tsquery类型前进行规范化处理。

```
SELECT 'fat & rat'::tsquery;
    tsquery
---------------
 'fat' & 'rat'
(1 row)

SELECT 'fat & (rat | cat)'::tsquery;
          tsquery
-----------------------------
 'fat' & ( 'rat' | 'cat' )
(1 row)

SELECT 'fat & rat & ! cat'::tsquery;
        tsquery
------------------------
 'fat' & 'rat' & !'cat'
(1 row)
```

tsquery中的词汇可以用一个或多个权重字母来标记，这些权重字母限制词汇只能与匹配权重的tsvector词汇进行匹配。

```
SELECT 'fat:ab & cat'::tsquery;
     tsquery
------------------
 'fat':AB & 'cat'
(1 row)
```

同样，tsquery中的词汇可以用*标记来指定前缀匹配。例如：这个查询可以匹配tsvector中以“super”开始的任意单词。

```
SELECT 'super:*'::tsquery;
  tsquery
-----------
 'super':*
(1 row)
```

需注意，前缀匹配会首先被文本搜索分词器处理。例如：postgres中提取的词干是postgr，匹配到了postgraduate，也就意味着下面的结果为真：

```
SELECT to_tsvector( 'postgraduate' ) @@ to_tsquery( 'postgres:*' ) AS RESULT;
 result
----------
```


版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
SELECT to_tsquery('postgres:*');
 to_tsquery
------------
 'postgr':*
(1 row)
```

to_tsquery函数会将单词转换为tsquery类型前进行规范化处理。'Fat:ab & Cats'规范化转为tsquery类型结果如下：

```
SELECT to_tsquery('Fat:ab & Cats');
    to_tsquery
------------------
 'fat':AB & 'cat'
(1 row)
```

21.5.13 UUID 类型

UUID：通用唯一识别码（Universally Unique Identifier）是用于计算机体系中以识别信息的一个128位标识符。

UUID的作用是让分布式系统中的所有元素都能有唯一的辨识信息，而不需要通过中央控制端来做辨识信息的指定。很多应用场景需要一个ID，仅用来标识一个对象。常见的例子有数据库表的ID字段。另一个例子是前端的各种UI库，因为它们通常需要动态创建各种UI元素，这些元素需要唯一的ID，这时就需要使用UUID。

DWS支持的UUID函数及UUID应用示例，可参考**UUID函数**。

UUID 格式

UUID由开放软件基金会标准化，作为分布式计算环境的一部分，在互联网工程任务组（IETF）公布的RFC 4122标准中对UUID进行了标准化。标准的UUID由36个字符组成，其中包括32个16进制数字和4个连字符‘-’，形式为8-4-4-4-12，标准的UUID示例如下：

```
a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
```

除了标准型的UUID，DWS同样支持以其他方式输入：大写字母和数字、由花括号包围的标准格式、省略部分或所有连字符、在任意一组四位数字之后加一个连字符。示例：

```
A0EEBC99-9C0B-4EF8-BB6D-6BB9BD380A11
{a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11}
a0eebc999c0b4ef8bb6d6bb9bd380a11
a0ee-bc99-9c0b-4ef8-bb6d-6bb9-bd38-0a11
```

UUID 组成

UUID采用时间信息和空间信息确保全局唯一，通常由时间戳信息、时钟序列和节点ID（MAC地址）组成。而DWS分布式集群中多个节点可能部署在同一个机器上，其MAC地址相同，UUID存在冲突的风险。因此DWS将最后48位为的MAC地址替换为生成UUID的CN或DN的序号和当前的线程ID，确保UUID在分布式集群内部做到全局唯一。

21.5.14 JSON 类型

JSON数据类型可以用来存储JSON（JavaScript Object Notation）数据。

可以是单独的一个标量，也可以是一个数组，也可以是一个键值对象，其中数组和对象可以统称容器(container)：

版权所有 © 华为云计算技术有限公司

1. 标量(scalar)：单一的数字、bool、string、null都可以叫做标量。
2. 数组(array)：[]结构，里面存放的元素可以是任意类型的JSON，并且不要求数组内所有元素都是同一类型。
3. 对象(object)：{}结构，存储key:value的键值对，其键只能是用“ ”包裹起来的字符串，值可以是任意类型的JSON，对于重复的键，按最后一个键值对为准。

DWS支持使用：json数据类型和jsonb数据类型存储JSON数据。其中：

- json对输入的字符串进行完整复制，使用时再去解析，所以它会保留输入的空格，重复键以及顺序等。
- jsonb解析输入后保存的二进制，它在解析时会删除语义无关的细节和重复的键，对键值也会进行排序，使用时不用再次解析。

因此可以发现，两者其实都是JSON，它们接受相同的字符串作为输入。它们实际的主要差别是效率。json数据类型存储输入文本的精确复制，处理函数必须在每个执行上重新解析； 而jsonb数据以分解的二进制格式存储， 这使得它由于添加了转换机制而在输入上稍微慢些，但是在处理上明显更快， 因为不需要重新解析。同时由于jsonb类型存在解析后的格式归一化等操作，同等的语义下只会有一种格式，因此可以更好更强大的支持很多其他额外的操作，比如按照一定的规则进行大小比较等。jsonb也支持索引，这也是一个明显的优势。

输入格式

json和jsonb输入必须是一个符合JSON数据格式的字符串，此字符串用单引号''声明。

null (null-json)：仅null，全小写。

```
SELECT 'null'::json;   -- suc
SELECT 'NULL'::jsonb;  -- err
```

数字 (num-json)：正负整数、小数、0，支持科学计数法。

```
SELECT '1'::json;
SELECT '-1.5'::json;
SELECT '-1.5e-5'::jsonb, '-1.5e+2'::jsonb;
SELECT '001'::json, '+15'::json, 'NaN'::json;  -- 不支持多余的前导0，正数的+号，以及NaN和infinity。
```

布尔(bool-json)：仅true、false，全小写。

```
SELECT 'true'::json;
SELECT 'false'::jsonb;
```

字符串(str-json)：必须是加双引号的字符串。

```
SELECT '"a"'::json;
SELECT '"abc"'::jsonb;
```

数组(array-json)：使用中括号[]包裹，满足数组书写条件。数组内元素类型可以是任意合法的JSON，且不要求类型一致。

```
SELECT '[1, 2, "foo", null]'::json;
SELECT '[]'::json;
SELECT '[1, 2, "foo", null, [[]], {}]'::jsonb;
```

对象(object-json)：使用大括号{}包裹，键必须是满足JSON字符串规则的字符串，值可以是任意合法的JSON。

```
SELECT '{}'::json;
SELECT '{"a": 1, "b": {"a": 2,  "b": null}}'::json;
SELECT '{"foo": [true, "bar"], "tags": {"a": 1, "b": null}}'::jsonb;
```

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- 'null'::json和null::json是两个不同的概念，类似于字符串str=“”和str=null。
- 对于数字，当使用科学计数法的时候，jsonb类型会将其展开，而json会精准复制输入。

jsonb 高级特性

json和jsonb的主要差异在于存储方式上的不同，jsonb存储的是解析后的二进制，能够体现JSON的层次结构，更便于直接访问等，因此jsonb较json具有很多高级特性。

格式归一化

- 对于输入的object-json字符串，解析成jsonb二进制后，会天然的丢弃语义上无关紧要的细节，比如空格：

```
SELECT '    [1, " a ", {"a"    :1     }]   '::jsonb;
    jsonb
----------------------
[1, " a ", {"a": 1}]
(1 row)
```

- 对于object-json，会删除重复的键值，只保留最后一个出现的，例如：

```
SELECT '{"a" : 1, "a" : 2}'::jsonb;
jsonb
----------
{"a": 2}
(1 row)
```

- 对于object-json，键值会重新进行排序，排序规则：长度长的在后、长度相等则ascii码大的在后，例如：

```
SELECT '{"aa" : 1, "b" : 2, "a" : 3}'::jsonb;
       jsonb
---------------------------
{"a": 3, "b": 2, "aa": 1}
(1 row)
```

大小比较

由于经过了格式归一化，保证了同一种语义下的jsonb只会有一种存在形式，因此按照指定的规则，可以比较大小。

1. 首先比较类型：object-jsonb > array-jsonb > bool-jsonb > num-jsonb > str-jsonb > null-jsonb
2. 同类型则比较内容：
   - str-json类型：依据text比较的方法，使用数据库默认排序规则进行比较，返回值正数代表大于，负数代表小于，0表示相等。
   - num-json类型：数值比较
   - bool-json类型：true > false
   - array-jsonb类型：长度长的 > 长度短的，长度相等则依次比较每个元素。
   - object-jsonb类型：长度长的 > 长度短的，长度相等则依次比较每个键值对，先比较键，再比较值。

⚠ 注意

object-jsonb类型内比较时使用的是格式整理后的最终结果进行比较，因此相对于直接的输入未必会很直观。

版权所有 © 华为云计算技术有限公司

创建索引

行存jsonb类型支持创建B-Tree索引和GIN索引，列存jsonb类型仅支持创建B-Tree索引。

如果整个JSONB列使用B-Tree索引，则能使用的运算符是=、<、<=、>、>=。

示例：创建表test，并插入数据。

```
CREATE TABLE test(id bigserial, data JSONB, PRIMARY KEY (id));
INSERT INTO test(data) VALUES('{"name":"Jack", "age":10, "nick_name":["Jacky","baobao"], "phone_list":["1111","2222"]}'::jsonb);
```

- 创建B-Tree索引。
  ```
  CREATE INDEX idx_test_data_age ON test USING btree(((data->>'age')::int));
  ```
  使用B-Tree索引查询age>1。
  ```
  SELECT * FROM test WHERE (data->>'age')::int>1;
  ```
- 创建GIN索引。
  ```
  CREATE INDEX idx_test_data ON test USING gin (data);
  ```
  使用GIN索引查询顶层关键词是否存在。
  ```
  SELECT * FROM test WHERE data ? 'id';
  SELECT * FROM test WHERE data ?| array['id','name'];
  ```
- 使用GIN索引查询非顶层关键词是否存在。
  ```
  CREATE INDEX idx_test_data_nick_name ON test USING gin((data->'nick_name'));
  SELECT * FROM test WHERE data->'nick_name' ? 'Jacky';
  ```
- 使用@>查询JSON中是否包含子json对象。
  ```
  SELECT * FROM test WHERE data @> '{"age":10, "nick_name":["Jacky"]}';
  ```

包含存在

查询一个JSON之中是否包含某些元素，或者某些元素是否存在于某个JSON中是jsonb的一个重要能力。

- 简单的标量/原始值只包含相同的值。
  ```
  SELECT '"foo"'::jsonb @> '"foo"'::jsonb;
  ```
- 左侧数组包含了右侧字符串。
  ```
  SELECT '[1, "aa", 3]'::jsonb ? 'aa';
  ```
- 左侧数组包含了右侧的数组所有元素，顺序、重复不重要。
  ```
  SELECT '[1, 2, 3]'::jsonb @> '[1, 3, 1]'::jsonb;
  ```
- 左侧object-json包含了右侧object-json的所有键值。
  ```
  SELECT '{"product": "PostgreSQL", "version": 9.4, "jsonb":true}'::jsonb @> '{"version":9.4}'::jsonb;
  ```
- 左侧数组并没有包含右侧的数组所有元素，因为左侧数组的三个元素为1、2、[1,3]，右侧的为1、3。
  ```
  SELECT '[1, 2, [1, 3]]'::jsonb @> '[1, 3]'::jsonb; -- false
  ```
- 相似的这种也没有包含。
  ```
  SELECT '{"foo": {"bar": "baz"}}'::jsonb @> '{"bar": "baz"}'::jsonb; -- false
  ```

21.5.15 RoaringBitmap 类型

DWS自8.1.3集群版本开始，支持RoaringBitmap数据类型，用于存储位图数据集。

RoaringBitmap数据类型支持行存，列存表。

版权所有 © 华为云计算技术有限公司

表 21-22 RoaringBitmap 类型

名字	存储容量	描述	范围
RoaringBitmap	32 字节	存储位图数据集	-2,147,483,648~2,147,483,647

示例：创建带有RoaringBitmap数据类型的表。

```
CREATE TABLE r_row (a int ,b text, c roaringbitmap);
CREATE TABLE r_col (a int ,b text, c roaringbitmap) with (orientation=column);
```

21.5.16 HLL 数据类型

HLL（HyperLoglog）是统计数据集中唯一值个数的高效近似算法。它有着计算速度快，节省空间的特点，不需要直接存储集合本身，而是存储一种名为HLL的数据结构。每当有新数据加入进行统计时，只需要把数据经过哈希计算并插入到HLL中，最后根据HLL就可以得到结果。

HLL与其他算法的比较请参见**表21-23**。

表 21-23 HLL 与其他算法比较

项目	Sort算法	Hash算法	HLL
时间复杂度	O(nlogn)	O(n)	O(n)
空间复杂度	O(n)	O(n)	1280 bytes
误差率	0	0	≈2%
所需存储空间	原始数据大小	原始数据大小	1280 bytes

HLL在计算速度和所占存储空间上都占优势。在时间复杂度上，Sort算法需要排序至少O(nlogn)的时间，虽说Hash算法和HLL一样扫描一次全表O(n)的时间就可以得出结果，但是存储空间上， Sort算法和Hash算法都需要先把原始数据存起来再进行统计，会导致存储空间消耗巨大，而对HLL来说不需要存原始数据，只需要维护HLL数据结构，故占用空间始终是1280bytes常数级别。

版权所有 © 华为云计算技术有限公司

须知

- 当前默认规格下可计算最大distinct值的数量为1.6e+12个，误差率最大仅2.3%。用户应注意如果计算结果超过当前规格下distinct最大值会导致计算结果误差率变大，或导致计算结果失败并报错。
- 用户在首次使用该特性时，应该对业务的distinct value做评估，选取适当的配置参数并做验证，以确保精度符合要求：
  - 当前默认参数下，可以计算的distinct value值为1.6e+12，如果计算得到的distinct value值为NaN，需要调整log2m和regwidth来容纳更多的distinct value。
  - 虽然hash算法存在极低的hash collision概率，但是建议用户在首次使用时，选取2-3个hash seed验证，如果得到的distinct value相差不大，则可以从该组seed中任选一个作为hash seed。

HLL中主要的数据结构，请参见**表21-24**。

表 21-24 HyperLogLog 中主要数据结构

数据类型	功能描述
hll	大小为确定的1280 bytes，可直接计算得到distinct值。

HLL 的应用场景

- 使用hll数据类型场景
  a. 创建带有hll类型的表并向表中插入空的hll。
  ```
  CREATE TABLE helloworld (id integer, set hll);
  INSERT INTO helloworld(id, set) VALUES (1, hll_empty());
  ```
  b. 把整数经过哈希计算加入到hll中。
  ```
  UPDATE helloworld SET set = hll_add(set, hll_hash_integer(12345)) WHERE id = 1;
  ```
  c. 把字符串经过哈希计算加入到hll中。
  ```
  UPDATE helloworld SET set = hll_add(set, hll_hash_text('hello world')) WHERE id = 1;
  ```
  d. 得到hll中的distinct值。
  ```
  SELECT hll_cardinality(set) FROM helloworld WHERE id = 1;
   hll_cardinality
  -----------------
                2
  (1 row)
  ```
- 使用hll进行网站访客统计场景
  a. 创建原始数据表facts，记录用户访问网站时间。
  ```
  CREATE TABLE facts (
          date            date,
          user_id         integer
  );
  ```
  b. 插入用户访问过网站的数据。
  ```
  INSERT INTO facts VALUES ('2019-02-20', generate_series(1,100));
  INSERT INTO facts VALUES ('2019-02-21', generate_series(1,200));
  INSERT INTO facts VALUES ('2019-02-22', generate_series(1,300));
  INSERT INTO facts VALUES ('2019-02-23', generate_series(1,400));
  INSERT INTO facts VALUES ('2019-02-24', generate_series(1,500));
  INSERT INTO facts VALUES ('2019-02-25', generate_series(1,600));
  ```

 版权所有 © 华为云计算技术有限公司

```
INSERT INTO facts VALUES ('2019-02-26', generate_series(1,700));
INSERT INTO facts VALUES ('2019-02-27', generate_series(1,800));
```

c. 创建表并指定列为hll。根据日期把数据分组，并把数据插入到hll中。

```
CREATE TABLE daily_uniques (
    date            date UNIQUE,
    users           hll
);

INSERT INTO daily_uniques(date, users)
    SELECT date, hll_add_agg(hll_hash_integer(user_id))
    FROM facts
    GROUP BY 1;
```

d. 计算每一天访问网站不同用户数量。

```
SELECT date, hll_cardinality(users) FROM daily_uniques ORDER BY date;
       date         | hll_cardinality
--------------------+------------------
 2019-02-20 00:00:00 |              100
 2019-02-21 00:00:00 | 203.813355588808
 2019-02-22 00:00:00 | 308.048239950384
 2019-02-23 00:00:00 | 410.529188080374
 2019-02-24 00:00:00 | 513.263875705319
 2019-02-25 00:00:00 | 609.271181107416
 2019-02-26 00:00:00 | 702.941844662509
 2019-02-27 00:00:00 | 792.249946595237
(8 rows)
```

e. 计算在2019.02.20到2019.02.26一周中有多少不同用户访问过网站。

```
SELECT hll_cardinality(hll_union_agg(users)) FROM daily_uniques WHERE date >=
'2019-02-20'::date AND date <= '2019-02-26'::date;
 hll_cardinality
------------------
 702.941844662509
(1 row)
```

f. 计算昨天访问过网站而今天没访问网站的用户数量。

```
SELECT date, (#hll_union_agg(users) OVER two_days) - #users AS lost_uniques FROM
daily_uniques WINDOW two_days AS (ORDER BY date ASC ROWS 1
PRECEDING);
       date         | lost_uniques
--------------------+--------------
 2019-02-20 00:00:00 |            0
 2019-02-21 00:00:00 |            0
 2019-02-22 00:00:00 |            0
 2019-02-23 00:00:00 |            0
 2019-02-24 00:00:00 |            0
 2019-02-25 00:00:00 |            0
 2019-02-26 00:00:00 |            0
 2019-02-27 00:00:00 |            0
(8 rows)
```

- 插入数据不满足hll数据结构要求时报错场景

  当用户给hll类型的字段插入数据的时候，必须保证插入的数据满足hll数据结构要求，如果解析后不满足就会报错。

  例如： 插入数据'E\\1234'时，该数据不满足hll数据结构，不能解析成功因此失败报错。

```
CREATE TABLE test(id integer, set hll);
INSERT INTO test VALUES(1, 'E\\1234');
ERROR:  invalid input syntax for integer: "E\\1234"
```

21.5.17 对象标识符类型

DWS在内部使用对象标识符（OID）作为各种系统表的主键。系统不会给用户创建的表增加一个OID系统字段，OID类型代表一个对象标识符。

目前OID类型用一个四字节的无符号整数实现。因此不建议在创建的表中使用OID字段做主键。

版权所有 © 华为云计算技术有限公司

表 21-25 对象标识符类型

名称	引用	描述	示例
OID	-	数字化的对象标识符。	564182
CID	-	命令标识符。它是系统字段cmin和cmax的数据类型。命令标识符是32位的量。	-
XID	-	事务标识符。它是系统字段xmin和xmax的数据类型。事务标识符也是32位的量。	-
TID	-	行标识符。它是系统表字段ctid的数据类型。行ID是一对数值（块号，块内的行索引），它标识该行在其所在表内的物理位置。	-
REGCONFIG	pg_ts_config	文本搜索配置。	english
REGDICTIONARY	pg_ts_dict	文本搜索字典。	simple
REGOPER	pg_operator	操作符名。	+
REGOPERATOR	pg_operator	带参数类型的操作符。	*(integer,integer)或-(NONE,integer)
REGPROC	pg_proc	函数名字。	sum
REGPROCEDURE	pg_proc	带参数类型的函数。	sum(int4)
REGCLASS	pg_class	关系名。	pg_type
REGTYPE	pg_type	数据类型名。	integer

OID类型：主要作为数据库系统表中字段使用。

示例：

```
SELECT oid FROM pg_class WHERE relname = 'pg_type';
 oid
------
 1247
(1 row)
```

OID别名类型REGCLASS：主要用于对象OID值的简化查找。

示例：

```
SELECT attrelid,attname,atttypid,attstattarget FROM pg_attribute WHERE attrelid = 'pg_type'::REGCLASS;
 attrelid |    attname     | atttypid | attstattarget
----------+----------------+----------+---------------
     1247 | xc_node_id     |       23 |             0
```


版权所有 © 华为云计算技术有限公司

```
  1247 | tableoid       |       26 |           0
  1247 | cmax           |       29 |           0
  1247 | xmax           |       28 |           0
  1247 | cmin           |       29 |           0
  1247 | xmin           |       28 |           0
  1247 | oid            |       26 |           0
  1247 | ctid           |       27 |           0
  1247 | typname        |       19 |          -1
  1247 | typnamespace   |       26 |          -1
  1247 | typowner       |       26 |          -1
  1247 | typlen         |       21 |          -1
  1247 | typbyval       |       16 |          -1
  1247 | typtype        |       18 |          -1
  1247 | typcategory    |       18 |          -1
  1247 | typispreferred |       16 |          -1
  1247 | typisdefined   |       16 |          -1
  1247 | typdelim       |       18 |          -1
  1247 | typrelid       |       26 |          -1
  1247 | typelem        |       26 |          -1
  1247 | typarray       |       26 |          -1
  1247 | typinput       |       24 |          -1
  1247 | typoutput      |       24 |          -1
  1247 | typreceive     |       24 |          -1
  1247 | typsend        |       24 |          -1
  1247 | typmodin       |       24 |          -1
  1247 | typmodout      |       24 |          -1
  1247 | typanalyze     |       24 |          -1
  1247 | typalign       |       18 |          -1
  1247 | typstorage     |       18 |          -1
  1247 | typnotnull     |       16 |          -1
  1247 | typbasetype    |       26 |          -1
  1247 | typtypmod      |       23 |          -1
  1247 | typndims       |       23 |          -1
  1247 | typcollation   |       26 |          -1
  1247 | typdefaultbin  |      194 |          -1
  1247 | typdefault     |       25 |          -1
  1247 | typacl         |     1034 |          -1
(38 rows)
```

21.5.18 伪类型

DWS数据类型中包含一系列特殊用途的类型，这些类型按照类别被称为伪类型。伪类型不能作为字段的数据类型，但是可以用于声明函数的参数或者结果类型。

当一个函数不仅仅是简单地接受并返回某种SQL数据类型，伪类型能起到很大的作用。表21-26列出了所有的伪类型。

表 21-26 伪类型

名字	描述
any	表示函数接受任何输入数据类型。
anyelement	表示函数接受任何数据类型。
anyarray	表示函数接受任意数组数据类型。
anynonarray	表示函数接受任意非数组数据类型。
anyenum	表示函数接受任意枚举数据类型。
anyrange	表示函数接受任意范围数据类型。
cstring	表示函数接受或者返回一个空结尾的C字符串。

版权所有 © 华为云计算技术有限公司

名字	描述
internal	表示函数接受或者返回一种服务器内部的数据类型。
language_handler	声明一个过程语言调用句柄返回language_handler。
fdw_handler	声明一个外部数据封装器返回fdw_handler。
record	标识函数返回一个未声明的行类型。
trigger	声明一个触发器函数返回trigger。
void	表示函数不返回数值。
opaque	一个已经过时的类型，以前用于所有上面这些用途。

声明用C编写的函数（不管是内置的还是动态装载的）都可以接受或者返回任何这样的伪数据类型。当伪类型作为参数类型使用时，用户需要保证函数的正常运行。

用过程语言编写的函数只能使用实现语言允许的伪类型。目前，过程语言不允许使用伪类型作为参数类型的，只允许使用void和record作为结果类型。一些多态的函数还支持使用anyelement，anyarray，anynonarray anyenum和anyrange类型。

伪类型internal用于声明只能在数据库系统内部调用的函数，这些函数不能直接在SQL查询里调用。如果某函数至少有一个internal类型的参数，则不能从SQL里调用。建议不要创建任何声明返回internal的函数，除非该函数至少有一个internal类型的参数。

示例：

创建或替换函数showall()。

```
CREATE OR REPLACE FUNCTION showall() RETURNS SETOF record
AS $$ SELECT count(*) from tpcds.store_sales where ss_customer_sk = 9692; $$
LANGUAGE SQL;
```

调用函数showall()。

```
SELECT showall();
 showall
---------
 (35)
(1 row)
```

删除函数。

```
DROP FUNCTION showall();
```

21.5.19 范围类型

范围类型是表示某些元素类型（称为范围的子类型）的值范围的数据类型。例如，时间戳范围可用于表示保留会议室的时间范围。在这种情况下，数据类型为tsrange（“时间戳范围”的缩写），时间戳是子类型。子类型必须具有总体的顺序，以便很好地定义元素值是在值的范围内、之前还是之后。

范围类型可以表达单一范围值中的多个元素值，并且可以很清晰地表达诸如范围重叠等概念。用于计划安排的时间和日期范围，也可以用于价格范围或仪器的测量范围等。

版权所有 © 华为云计算技术有限公司

内置范围类型

DWS带有下列内置范围类型：

- int4range — integer的范围
- int8range — bigint的范围
- numrange — numeric的范围
- tsrange — 不带时区的timestamp的范围
- tstzrange — 带时区的timestamp的范围
- daterange — date的范围

说明

DWS暂不支持用户自定义范围类型。

创建表reservation并插入数据，其中during字段类型为tsrange：

```
CREATE TABLE reservation (room int, during tsrange);
INSERT INTO reservation VALUES (1108, '[2010-01-01 14:30, 2010-01-01 15:30)');
```

查询范围是否包含：

```
SELECT int4range(10, 20) @> 3;
 ?column?
----------
 f
(1 row)
```

查询范围是否重叠：

```
SELECT numrange(11.1, 22.2) && numrange(20.0, 30.0);
 ?column?
----------
 t
(1 row)
```

抽取范围的上界：

```
SELECT upper(int8range(15, 25));
 upper
-------
    25
(1 row)
```

计算范围的交集：

```
SELECT int4range(10, 20) * int4range(15, 25);
 ?column?
----------
 [15,20)
(1 row)
```

查询范围是否为空：

```
SELECT isempty(numrange(1, 5));
 isempty
---------
 f
(1 row)
```

 版权所有 © 华为云计算技术有限公司

包含和排除边界

每一个非空范围都有两个界限：下界和上界。这些值之间的所有点都被包括在范围内。包含界限意味着边界点本身也被包括在范围内，而排除边界意味着边界点不被包括在范围内。

在范围的文本形式中，包含下界用“[”表示，排除下界用“(”表示。同样，包含上界用“]”表示，排除上界用“)”表示。

函数**lower_inc(anyrange)**和**lower_inc(anyrange)**分别测试一个范围值的上下界。

无限（无界）范围

范围的下界可以省略，这意味着所有小于上界的值都包括在范围中，例如(,3]。同样，范围的上界被省略，则所有大于下界的值都包括在范围中。如果下界和上界都被省略，则该元素类型的所有值都被认为在范围内。如果缺失的边界指定为包含则自动将包含转换为排除，例如[,]将转换为(,)。可以将这些缺失的值视为+/-无穷大，但它们是特殊的范围类型值，并且被视为超出任何范围元素类型的+/-无穷大值。

具有“无穷大”概念的元素类型可以将其作为显式边界值。例如，在时间戳范围，[today,infinity)不包括特殊的timestamp值infinity，尽管[today,infinity]包括它，就好比 [today,)和[today,]。

函数**lower_inf(anyrange)**和**upper_inf(anyrange)**分别测试一个范围的无限上下界。

范围的输入/输出

范围值的输入必须遵循下列模式之一：

```
(lower-bound,upper-bound)
(lower-bound,upper-bound]
[lower-bound,upper-bound)
[lower-bound,upper-bound]
empty
```

圆括号或方括号指示上下界是否为排除的或者包含的。最后一个模式是empty，它表示一个空范围（一个不包含点的范围）。

lower-bound可以作为子类型的合法输入的一个字符串，或者是空，表示没有下界。同样，upper-bound可以是作为子类型的合法输入的一个字符串，或者是空，表示没有上界。

每个界限值可以使用双引号引用。如果界限值包含圆括号、方括号、逗号、双引号或反斜线时，则必须使用双引号引用，否则这些符号会被认作范围语法的一部分。要想把双引号或反斜线放在被引用的界限值中，需要在双引号或反斜线前面加一个反斜线（在双引号引用的界限值中的一对双引号表示一个双引号字符，这与SQL字符串中的单引号规则类似）。此外，可以避免引用并且使用反斜线转义来保护所有数据字符，否则它们会被当做返回语法的一部分。什么都不写则表示一个无限界限，因此，要表示空字符串的界限值，可以写成""。

范围值前后允许有空格，但是圆括号或方括号之间的任何空格会被当做上下界值的一部分（取决于元素类型，它可能是有意义的也可能是无意义的）。

示例

查询包括3，不包括7，并且包括3和7之间的所有点：

```
SELECT '[3,7)'::int4range;
 int4range
```

 版权所有 © 华为云计算技术有限公司

```
-----------
 [3,7)
(1 row)
```

查询既不包括3也不包括 7，但是包括之间的所有点：

```
SELECT '(3,7)'::int4range;
 int4range
-----------
 [4,7)
(1 row)
```

查询只包括单独一个点4：

```
SELECT '[4,4]'::int4range;
 int4range
-----------
 [4,5)
(1 row)
```

查询不包括点（并且将被标准化为 '空'）：

```
SELECT '[4,4)'::int4range;
 int4range
-----------
 empty
(1 row)
```

构造范围

每一种范围类型都有一个与其同名的构造器函数。使用构造器函数比写一个范围文字常数更方便，因为它避免了对界限值的额外引用。构造器函数接受两个或三个参数。两个参数的形式以标准的形式构造一个范围（包含下界，排除上界），而三个参数的形式按照第三个参数指定的界限形式构造一个范围。第三个参数必须是下列字符串之一：“()”、“(]”、“[)”或者“[]”。例如：

完整形式是：下界、上界以及指示界限包含性/排除性的文本参数：

```
SELECT numrange(1.0, 14.0, '(]');
  numrange
------------
 (1.0,14.0]
(1 row)
```

如果第三个参数被忽略，则假定为 '[)'：

```
SELECT numrange(1.0, 14.0);
  numrange
------------
 [1.0,14.0)
(1 row)
```

尽管这里指定了'(]'，单返回结果时该值将被转换成标准形式，因为int8range是一种**离散范围类型**：

```
SELECT int8range(1, 14, '(]');
 int8range
-----------
 [2,15)
(1 row)
```

界限使用NULL导致范围是无界的：

```
SELECT numrange(NULL, 2.2);
 numrange
----------
 (,2.2)
(1 row)
```


版权所有 © 华为云计算技术有限公司

离散范围类型

离散范围是指其元素类型具有定义明确的“步长”的范围，如integer或date。在这些类型中，当两个元素之间没有有效值时，它们可以被说成是相邻。

离散范围类型的每个元素值都有一个明确的“下一个”或“上一个”值。这样就可以通过选择下一个或上一个元素值，在范围界限的包含和排除表达之间转换。例如，在整数范围类型中，[4,8]和(3,9)表示相同的值集合，但对于超过numeric的范围，情况并非如此。

离散范围类型应具有识别元素类型所需步长的规范函数。规范化函数负责将范围类型的等价值转换为具有相同的表示，特别是与包含或者排除界限一致。如果未指定规范化函数，则具有不同格式的范围将始终被视为不相等，即使它们实际上是表达相同的一组值。

内置范围类型int4range、int8range和daterange都使用规范形式，该形式包括下界并且排除上界，也就是[)。但是，用户定义的范围类型可以使用其他约定。

自定义范围类型

用户可以定义范围类型。例如，要创建一个subtype float8的范围类型：

```
CREATE TYPE floatrange AS RANGE (
        subtype = float8,
        subtype_diff = float8mi
    );

SELECT '[1.234, 5.678]'::floatrange;
```

索引

可以为范围类型的表列创建GiST索引。例如：

```
CREATE TABLE reservation (room int, during tsrange);
CREATE INDEX reservation_idx ON reservation USING gist (during);
```

GiST索引可以加速涉及以下范围操作符的查询： =、 &&、 <@、 @>、 <<、 >>、 -|-、 &<以及 &> 。详见**范围操作符**。

此外，也可以在范围类型的表列上创建B-tree索引。对于这些索引类型，有用的范围操作就是等值。范围类型的B-tree支持主要是为了允许在查询内部进行排序，而不是创建真正的索引。

21.5.20 复合类型

复合类型表示行或记录的结构，它本质上就是字段名及其数据类型的列表。DWS允许支持将表的列声明为复合类型。复合类型本质上和表的行类型相同，但是如果只想定义一种类型，使用**CREATE TYPE**可避免创建一个实际的表。单独的复合类型也是很有用的，例如可以作为函数的参数或者返回类型。

复合类型的声明

DWS支持用户使用**CREATE TYPE**定义复合类型：

```
CREATE TYPE complex AS (
        r       double precision,
        i       double precision
    );
```

 版权所有 © 华为云计算技术有限公司

```
CREATE TYPE inventory_item AS (
    name            text,
    supplier_id     integer,
    price           numeric
);
```

定义复合类型之后，可用来创建表或函数：

```
CREATE TABLE on_hand (
    item      inventory_item,
    count     integer
);

INSERT INTO on_hand VALUES (ROW('fuzzy dice', 42, 1.99), 1000);
CREATE FUNCTION price_extension(inventory_item, integer) RETURNS numeric
    AS 'SELECT $1.price * $2' LANGUAGE SQL;

SELECT price_extension(item, 10) FROM on_hand;
```

构造复合值

要把复合值写作文字常量，可以将字段值括在圆括号中，并用逗号分隔。可以在任何字段值加上双引号，如果字段值包含逗号或括号则必须这样做。复合常量的一般格式如下：

```
'( val1 , val2 , ... )'
```

上文中的'("fuzzy dice",42,1.99)'便属于inventory_item类型的一个合法值。

要让一个字段为NULL，在列表中对应位置上空出即可。如果需要一个字段为空字符串，使用引号即可。例如下列示例，第一个字段是非NULL空字符串，第三个是NULL：

```
'("",42,)'
```

ROW表达式也能被用来构建组合值。例如：

```
ROW('fuzzy dice', 42, 1.99)
ROW('', 42, NULL)
```

访问组合类型

要访问复合列的字段，可以写成一个点和字段的名称，就像从表名中选择字段一样。为了避免混淆，必须使用括号进行区分。例如，尝试从示例表on_hand中选择一些子域：

```
SELECT item.name FROM on_hand WHERE item.price > 9.99;
ERROR:  missing FROM-clause entry for table "item"
```

这样写是会与从表中选择字段混淆的，因为名称item会被当成是一个表名，而不是on_hand的一个列名。必须写成这样：

```
SELECT (item).name FROM on_hand WHERE (item).price > 9.99;
```

或者还需要使用表名（例如在一个多表查询中）：

```
SELECT (on_hand.item).name FROM on_hand WHERE (on_hand.item).price > 9.99;
```

现在加上括号的对象就被正确地解释为对item列的引用，然后可以从中选出子域。

只要从一个组合值中选择一个字段，相似的语法问题就适用。例如，要从返回组合值的函数的结果中选取一个字段，需要这样写：

```
SELECT (my_func(...)).field FROM ...
```


版权所有 © 华为云计算技术有限公司

如果没有额外的圆括号，这将生成一个语法错误。

21.5.21 列存表支持的数据类型

列存表支持的数据类型如表21-27所示。其他未列举的数据类型，暂不支持。

表 21-27 列存表支持的数据类型

类别	数据类型	描述	长度
数值类型	smallint	小范围整数，别名为INT2。	2
	integer	常用的整数，别名为INT4。	4
	bigint	大范围的整数，别名为INT8。	8
	decimal	任意精度型。	可变长度
	numeric	任意精度型。	可变长度
	real	单精度浮点数。	4
	double precision	双精度浮点数。	8
	smallserial	二字节序列整型。	2
	serial	四字节序列整型。	4
	bigserial	八字节序列整型。	8
货币类型	money	货币金额。	8
字符类型	character varying(n), varchar(n)	变长字符串。	可变长度
	character(n), char(n)	定长字符串。	n
	character、char	单字节内部类型。	1
	text	变长字符串。	可变长度
	nvarchar2	变长字符串。	可变长度
	clob	文本大对象。	可变长度
日期/时间类型	timestamp with time zone	日期和时间，带时区。	8
	timestamp without time zone	日期和时间。	8

版权所有 © 华为云计算技术有限公司

类别	数据类型	描述	长度
	date	Oracle兼容模式下记录日期和时间；其他兼容模式下，记录日期。	Oracle兼容模式下，占存储空间8字节；其他兼容模式下，占存储空间4字节。
	time without time zone	只用于一日内时间。	8
	time with time zone	只用于一日内时间，带时区。	12
	interval	时间间隔。	16
JSON类型	json	-	-
	jsonb	-	-
RoaringBitmap	-	高效位图，支持int类型数据集位图计算。	可变长度

21.5.22 XML 类型

XML数据类型可以被用来存储XML（eXtensible Markup Language）数据。XML数据也可以存储为text类型，但是XML数据类型的优势在于会检查每个存储的值是不是结构良好的XML值。XML可以存储由XML标准定义的格式良好的文档，以及由XML标准中定义的“内容”片段，内容片段中可以有多个顶级元素或字符节点。

XML类型相关的支持函数请参见**XML函数**。

设置 XML 参数

语法格式如下：

```
SET XML OPTION { DOCUMENT | CONTENT };
SET xmloption TO { DOCUMENT | CONTENT };
```

当一个字符串值在没有通过XMLPARSE或XMLSERIALIZE函数与XML类型进行转换时，由XML OPTION会话配置参数来决定选择DOCUMENT还是CONTENT。

默认为CONTENT，表示所有形式的xml数据都被允许。

示例：

```
SET XML OPTION DOCUMENT;
SET
SET xmloption TO DOCUMENT;
SET
```

设置二进制数据的编码格式

语法格式：

```
SET xmlbinary TO { base64 | hex};
```


版权所有 © 华为云计算技术有限公司

示例：

```
SET xmlbinary TO base64;
SET

SELECT xmlelement(name foo, bytea 'bar');
xmlelement
----------------
<foo>YmFy</foo>
(1 row)

SET xmlbinary TO hex;
SET

SELECT xmlelement(name foo, bytea 'bar');
xmlelement
-------------------
<foo>626172</foo>
(1 row)
```

访问 XML 值

XML数据类型比较特殊，它不提供任何比较操作符，这是因为对于XML数据没有通用的比较算法，所以无法通过比较一个XML值和一个搜索值来检索数据行。XML数据通常应该伴随一个ID值用于检索数据。另一种比较XML值的方案是将XML值转换成字符串，但字符串的比较并不能解决常见的XML值比较场景。

21.6 常量与宏

DWS支持的常量和宏请参见**表21-28**。

表 21-28 常量和宏

参数	描述	相关SQL语句示例
CURRENT_CATALOG	当前数据库	SELECT CURRENT_CATALOG;
CURRENT_ROLE	当前角色	SELECT CURRENT_ROLE;
CURRENT_SCHEMA	当前数据库模式	SELECT CURRENT_SCHEMA;
CURRENT_USER	当前用户	SELECT CURRENT_USER;
LOCALTIMESTAMP	当前会话时间（无时区）	SELECT LOCALTIMESTAMP;
NULL	空值	-
SESSION_USER	当前系统用户	SELECT SESSION_USER;
SYSDATE	当前系统日期	SELECT SYSDATE; 或 SELECT now()::DATE;

版权所有 © 华为云计算技术有限公司

参数	描述	相关SQL语句示例
USER	当前用户，与CURRENT_USER一致。	SELECT USER;

21.7 函数和操作符

21.7.1 字符串处理函数和操作符

21.7.1.1 字符串处理函数和操作符总体介绍

DWS提供的字符处理函数和操作符主要用于字符串处理，例如获取字符串长度、数据截取、拼接、转化、比较等操作。

在了解字符串处理函数前，您可以先简单了解**位**、**字节**、**字符**的相关概念：

- **位（bit）：** 计算机中最小的信息单位，表示二进制中的0或1，只有两种状态（对应电路的通 / 断、磁介质的正 / 反向磁化）。
- **字节（byte）：** 1字节=8位，是计算机存储和传输数据的基本单位。
- **字符：** 人类可读的文本符号，例如字母、数字、标点符号或汉字，而**数据库字符编码**决定了字符如何转换为字节。数据库字符编码在创建数据库语法中定义，详情请参见**CREATE DATABASE**。

例如在DWS的**UTF-8编码**中：一个英文字符占1个字节，而一个中文字符占3个字节，若字段定义为VARCHAR(100)，理论上最大占用100字符 * 3字节/字符 = 300字节。而在**GBK编码**中，一个中文字符占2个字节。

字符串处理函数按使用场景不同可分为以下几类：

- **获取字符串长度**：查询某个字符串的位数、字节长度等，例如查询“World”的位数。
- **截取字符串**：从一长串的字符串中截取某个子字符串，例如从“HelloWorld”中截取“Hello”。
- **拼接字符串**：将两串单独的字符串进行拼接，例如拼接“DATA”和“BASE”，得到“DATABASE”。
- **替换和填充字符串**：将字符串中的某个字符或子字符串替换成别的字符串，或者将多余空格删除，例如将“Database ”后面的空格清空。
- **删除指定字符**：从字符串中删除掉指定的字符，例如删除“DataxBasexx”后面多余的“xx”。
- **分割字符串**：将字符串按需要分割成几块，并用分隔符分割，例如将“hello world”分成“hello， world”。
- **转换字符串**：将字符串以某种规则进行转换，例如字母大小写转换。
- **比较字符串**：将两个字符串进行比较。
- **获取某一个字符或子字符串的位置**：例如，获取子字符串“bcd”在主字符串“abcdabcdabcd”中，从第2个字符开始，第2次匹配的位置。
- **判断某个字符串是否符合某种匹配模式(字符串相关的模式匹配函数)**：例如，判断“a”是否在“[ac]”的正则表达式里面。

版权所有 © 华为云计算技术有限公司

21.7.1.2 获取字符串长度

DWS支持以下函数计算字符串的长度，包括获取**位数**、**字节数**和**字符数**。关于位数、字节数和字符数的区别可参见**字符串处理函数和操作符总体介绍**。

表 21-29 获取字符串长度常用函数

分类	功能	对应函数	简单示例	使用区别
获取位数	获取字符串的位数。	**bit_length(string)**	SELECT bit_length('world'); bit_length ------------ 40 (1 row)	-
获取字节数	获取所有字符串类型的字节数。	**octet_length(string)**	SELECT octet_length('world'); octet_length ------------ 5 (1 row)	octet_length(string)与lengthb(string)在功能上完全一致，在处理MySQL/DWS等跨平台数据库开发时，**优先使用octet_length(string)**，**lengthb(string)**是为了兼容Oracle而设定的一个映射函数，**仅在Oracle迁移业务使用**。 lengthb(text/bpchar)与lengthb(string)一致，只是特指入参为text或bpchar类型的情况。
	获取所有字符串类型的字节数，仅为兼容Oracle。	**lengthb(string)**	SELECT lengthb('abc'::text); lengthb ------------ 3 (1 row)	
	获取text、bpchar类型的字节数，仅为兼容Oracle。	**lengthb(text/bpchar)**	SELECT lengthb('abc'::CHAR(5)); lengthb ------------ 5 (1 row)	
获取字符数	获取字符串的字符数。	**char_length(string)或character_length(string)** **length(string)**	SELECT length('database'); length -------- 8 (1 row)	length(string)功能与char_length(string)完全等价，写法更简洁。
	获取指定编码格式的字符串的字符数。	**length(string bytea, encoding name)**	-	-

版权所有 © 华为云计算技术有限公司

bit_length(string)

描述：字符串的位数。返回结果为字节数乘以8，得到总位数。字符串的位数取决于数据库编码方式，例如GBK字符编码下，一个中文为2个字节，UTF8和SQL_ASCII字符编码下，一个中文字为3个字节。

返回值类型：integer

示例：

```
SELECT bit_length('world');
 bit_length
------------
         40
(1 row)
```

GBK字符编码下，示例：

```
SELECT bit_length('我们');
 bit_length
------------
         32
(1 row)
```

UTF8和SQL_ASCII字符编码下，示例：

```
SELECT bit_length('我们');
 bit_length
------------
         48
(1 row)
```

octet_length(string)

描述：获取字符串中的字节数。该函数返回字符串的字节数，对于处理多字节字符（如 UTF-8 编码的汉字、特殊符号等）非常重要，它能准确获取字符串的存储大小。

返回值类型：integer

示例：

```
SELECT octet_length('data');
 octet_length
--------------
            4
(1 row)

SELECT octet_length('数据库');
 octet_length
--------------
            9
(1 row)
```

lengthb(string)

描述：获取字符串的字节长度。**功能与octet_length(string)没有任何差别**，只是为兼容Oracle而存在，如果是从Oracle迁移过来的场景，请使用lengthb(string)，但**如果业务需要跨数据库运行（如同时兼容DWS和MySQL），请使用octet_length(string)**，因为MySQL没有lengthb函数。

返回结果与字符集有关，同样的中文字符，在GBK与UTF8中，返回的字节数不同。

返回值类型：integer

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT lengthb('hello');
 lengthb
---------
       5
(1 row)
```

lengthb(text/bpchar)

描述：获取指定字符串的字节数，入参支持text、bpchar（等同于CHAR(n)），其中:

- lengthb(text)表示计算实际字符的字节长度（无填充空格），适合绝大多数业务场景。
- lengthb(bpchar)表示计算固定长度字符串的字节长度（包含填充空格），适合严格定长场景（如固定长度的编码、证件号），不推荐日常使用（空格填充易引发隐式问题）。

返回值类型：integer

说明

- 若字符串中存在换行符，如字符串由一个换行符和一个空格组成，在DWS中LENGTH和LENGTHB的值为2。
- 此函数获取的为指定字符串的字节数，对于多字节字符编码，一个字符可能占用多个字节（例如，UTF-8编码中，中文字符通常占用3个字节）。

示例：

```
SELECT lengthb('hello');
 lengthb
---------
       5
(1 row)
```

入参为text类型和bpchar类型的示例对比如下：

```
-- 案例1: 使用 text 类型
SELECT lengthb('abc'::text);    -- 'abc'占用3字节（UTF-8），返回3
-- 案例2: 使用 bpchar (CHAR(5)) 类型
SELECT lengthb('abc'::CHAR(5)); -- 存储为 'abc  '， 返回5（每个空格占1字节）
```

```
utf8=> SELECT lengthb('abc'::text);
 lengthb
---------
       3
(1 row)

utf8=> SELECT lengthb('abc'::CHAR(5));
 lengthb
---------
       5
(1 row)
```

UTF8编码中，示例：

```
SELECT lengthb('数据库');
 lengthb
---------
       9
(1 row)
```


版权所有 © 华为云计算技术有限公司

char_length(string)或 character_length(string)

描述：获取字符串中的字符个数。

返回值类型：integer

> **说明**
>
> 对于多字节字符编码，一个字符可能占用多个字节（例如，UTF-8编码中，中文字符通常占用3个字节）。

示例：

```
SELECT char_length('hello');
 char_length
-------------
           5
(1 row)

SELECT char_length('数据库');
 char_length
-------------
           9
(1 row)
```

length(string)

描述：获取字符串的字符长度（字符数量），length(string)功能与**char_length(string)**完全等价，写法更简洁。

返回值类型：integer

示例：

```
SELECT length('database');
 length
--------
      8
(1 row)
```

length(string bytea, encoding name)

描述：获取指定编码格式的字符串的字符数。在指定的编码格式中，string必须是有效的。

返回值类型：integer

示例：

```
SELECT length('database', 'UTF8');
 length
--------
      8
(1 row)
```

21.7.1.3 截取字符串

在数据仓库多维分析场景中，**字符串截取**是支撑业务分析的关键技术能力。

典型场景包括：

版权所有 © 华为云计算技术有限公司

- **提取核心信息**：对复合订单号进行智能分割，例如将"EC2023-BJ-0129A"拆解出核心序列号"0129A"，在18位身份证编码中使用substring()函数同时捕获行政区划代码、出生年月信息等。
- **脱敏数据**：用户隐私数据处理，例如通过right()函数精准提取手机号末4位实现数据脱敏。
- **精准提取信息**：支持**按正则表达式**进行匹配提取，从而精准提取有效信息，例如通过regexp_matches函数匹配出多个以b字母开头的字符串。
- **清除冗余数据**：例如清除某个订单号信息前后的多余空格或空字符。

DWS提供多种截取字符串的函数供用户选择。本章节中函数所使用的示例大部分以英文字符串示例为主，如果示例中存在中文字符串，则一般指UTF-8字符编码下的场景示例，字符编码在创建数据库时指定，参见**CREATE DATABASE**。

表 21-30 截取字符串常用函数

分类	功能	对应函数	简单示例	使用区别
去除两端字符	去除字符串两端指定的字符或空格。	**btrim(string text [, characters text])**	SELECT btrim(' sting '); btrim ------- sting (1 row)	-
提取子串，单位字节	提取子串，从第几个字节开始提取，一共提取几个字节。	**substrb(string,from,count)**	SELECT substrb('ABCDEF',2,2); substrb --------- BC (1 row)	四个函数的功能很相似，区别是提取剩余完整部分，还是只提取中间几个字节，入参是text类型，还是所有类字符串。
	提取子串，从第几个字节开始提取，一共提取几个字节。功能同上，区别是强调入参类型是text。	**substrb(text,int,int)**	-	
	提取子串，从第几个字节开始提取，提取剩余完整部分，当from是负数时，可以表示截取后边n个字节。	**substrb(string,from)**	SELECT substrb('ABCDEF',-2); substrb --------- EF (1 row)	
	提取子串，从第几个字节开始提取，提取剩余完整部分。功能同上，区别是强调入参类型是text。	**substrb(text,int)**	SELECT substrb('string',2); substrb --------- tring (1 row)	

版权所有 © 华为云计算技术有限公司

分类	功能	对应函数	简单示例	使用区别
提取子串，单位字符	提取子串，从第几个字符开始提取，一共提取几个字符。	**substr(string,from,count)**	SELECT substr('database',0,4); substr -------- data (1 row)	是提取剩余完整部分，还是只提取中间几个字符，**substr(string,from,count)**与**substring(string [from int] [for int])**高度相似，后者是标准SQL，前者更简洁。
	提取子串，从第几个字符开始提取，提取剩余完整部分。	**substr(string,from)**	SELECT substr('database',5); substr -------- base (1 row)	
	提取子串，从第几个字符开始提取，一共提取几个字符。与**substr(string,from,count)**功能一致。	**substring(string [from int] [for int])**	-	
	提取前几位字符。	**left(str text, n int)**	-	-
	提取后几位字符。	**right(str text, n int)**	-	-
按正则表达式提取子串	提取子串，匹配某种表达式，符合了才截取成功。	**substring(string from pattern)**	单击左侧链接跳转查看。	regexp_substr函数与substring函数类似，substring函数是SQL标准，regexp_substr属于扩展函数，正则能力更强。 regexp_matches返回类型为文本数组，适合提取分组信息的场景。
	提取子串，匹配某种表达式，符合了才截取成功，支持自定义转义符。	**substring(string from pattern for escape)**		
	功能同上类似，但是正则能力强。	**regexp_substr(text,text)**		
	功能同上类似，但是正则能力强。	**regexp_substr(source_char, pattern)**		
	提取子串，但是返回类型是文本数组，尤其适合需要多条件匹配或提取分组信息的场景。	**regexp_matches(string text, pattern text [, flags text])**		

版权所有 © 华为云计算技术有限公司

btrim(string text [, characters text])

描述：去除字符串（ string ）两端指定的字符。仅处理两端字符，中间字符即使匹配也不会被移除。

返回值类型：text

参数说明：

- string，必选，目标字符串。
- characters，可选，要在目标字符串两端移除的字符。characters参数缺省时，默认去除所有的空白字符。

示例：

```
SELECT btrim('sring' , 'ing');
 btrim
-------
 sr
(1 row)

SELECT btrim('   sting  ');  --不指定characters，默认去除多余空格
 btrim
-------
 sting
(1 row)

SELECT btrim('xxdataxxbasexx','x');   --注意只去除两端的x，中间有x也不会去除。
 btrim
-------
 dataxxbase
(1 row)
```

substrb(string,from,count)

描述：从字符串中截取指定长度的子字符串。from表示截取的起始位置，count表示截取的子字符串长度。

该函数和**substr(string,from,count)**函数功能一致，但是计算单位为字节。

- from为0时，按1处理。
- from为正数时，截取从from开始的count个字节。
- from为负数时，截取从倒数第n个开始的count个字节，n为from的绝对值。
- count小于1时，返回null。

返回值类型：bytea

示例：

```
SELECT substrb('ABCDEF',2,2);
 substrb
---------
 BC
(1 row)
```

substrb(text,int,int)

描述：从字符串中提取子字符串，第一个int表示从第几个字节位置开始提取，第二个int表示提取几个字节。

返回值类型：text

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT substrb('string',2,3);    ---从第2个字节“t”开始，提取3个字节。
 substrb
---------
 tri
(1 row)

SELECT substrb('学习数据库知识',7,9);    --UTF8编码下，一个中文占3个字节，从第7个字节位置开始，即跳过2个中文字，从“数”开始，截取9个字节（3个中文）
 substrb
---------
 数据库
(1 row)

SELECT substrb('学习数据库知识',7,9);    --GBK编码下，一个中文占2个字节，从第7个字节位置开始，即跳过3个中文字，从“据”开始，截取9个字节（4个中文）
 substrb
---------
 据库知识
(1 row)
```

substrb(string,from)

描述：从字符串中截取指定的子字符串，from表示截取的起始位置。该函数和**substr(string,from)**函数功能一致，但是计算单位为字节。

- from为0时，按1处理。
- from为正数时，截取从from到末尾的所有字节。
- from为负数时，截取字符串的后n个字节，n为from的绝对值。

返回值类型：bytea

示例：

```
SELECT substrb('ABCDEF',-2);
 substrb
---------
 EF
(1 row)
```

substrb(text,int)

描述：从字符串中提取子字符串，int表示从第几个字节开始提取，提取后面剩余的完整部分。

返回值类型：text

示例：

```
SELECT substrb('string',2);
 substrb
---------
 tring
(1 row)

SELECT substrb('学习数据库知识',7);    --UTF8编码下，一个中文占3个字节，从第7个字节位置开始，即跳过2个中文字，从“数”开始截取后面完整部分
 substrb
---------
 数据库知识
(1 row)
```


版权所有 © 华为云计算技术有限公司

substr(string,from,count)

描述：从字符串中截取指定长度的子字符串。

from表示截取的起始位置，count表示截取的子字符串长度。

- from为0时，按1处理。
- from为正数时，截取从from开始的count个字符。
- from为负数时，截取从倒数第n个开始的count个字符，n为from的绝对值。
- count小于1时，返回null。

返回值类型：varchar

示例：

from为0时，截取“database”从第1个字符开始的4个字符。

```
SELECT substr('database',0,4);
 substr
--------
 data
(1 row)
```

from为正数时，截取“database”从第5个字符开始的4个字符。

```
SELECT substr('database',5,4);
 substr
--------
 base
(1 row)
```

from为负数时，截取“database”从倒数第4个字符开始的3个字符。

```
SELECT substr('database',-4,3);
 substr
--------
 bas
(1 row)
```

substr(string,from)

描述：从字符串中截取指定的子字符串。

from表示截取的起始位置。

- from为0时，按1处理。
- from为正数时，截取从from到末尾的所有字符。
- from为负数时，截取字符串的后n个字符，n为from的绝对值。

返回值类型：varchar

示例：

from为0时，截取“database”从第1个字符至末尾的所有字符。

```
SELECT substr('database',0);
  substr
----------
 database
(1 row)
```

from为正数时，截取“database”从第5个字符至末尾的所有字符。

 版权所有 © 华为云计算技术有限公司

```
SELECT substr('database',5);
 substr
--------
 base
(1 row)
```

from为负数时，截取“database”从倒数第4个字符开始至末尾的所有字符。

```
SELECT substr('database',-4);
 substr
--------
 base
(1 row)
```

substring(string [from int] [for int])

描述：截取子字符串，from int表示从第几个字符开始截取，for int表示截取几个字符。与**substr(string,from,count)**功能一致。

返回值类型：text

示例：

从字符串“Thomas”的第2个字符“h”开始连续截取3个字符，即返回“hom”。

```
SELECT substring('Thomas' from 2 for 3);
 substring
-----------
 hom
(1 row)
```

left(str text, n int)

描述：返回字符串的前n个字符。

- ORA和TD兼容模式下，当n是负数时，返回除最后|n|个字符以外的所有字符。
- MySQL兼容模式下，当n是负数时，返回空串。

返回值类型：text

示例：

```
 SELECT left('database', 4);
 left
------
 data
(1 row)
```

right(str text, n int)

描述：返回字符串中的后n个字符。

- ORA和TD兼容模式下，当n是负数时，返回除前|n|个字符以外的所有字符。
- MySQL兼容模式下，当n是负数时，返回空串。

返回值类型：text

示例：

```
SELECT right('abcde', 2);
 right
-------
 de
```

 版权所有 © 华为云计算技术有限公司

```
(1 row)

SELECT right('abcde', -2);
 right
-------
 cde
(1 row)
```

substring(string from *pattern*)

描述：截取匹配POSIX正则表达式（也称为模式）的子字符串，其中string表示原始字符串，pattern表示待匹配的模式。如果该字符串没有匹配上模式则返回空值，否则返回文本中匹配到模式的那一部分。

返回值类型：text

示例：

```
SELECT substring('Thomas' from '...$');    ---匹配末尾3个连续字符
 substring
-----------
 mas
(1 row)

SELECT substring('Thomas' from '^...');     ---匹配开头3个连续字符
 substring
-----------
 Tho
(1 row)

SELECT substring('Tom' from '....$');      ---匹配末尾4个连续字符，Tom仅3个字符，无法匹配条件，结果为空
 substring
-----------

(1 row)

SELECT substring('order_20251120_12345' from '_([0-9]+)_');      ---获取两个“_”之间的订单号
 substring
-----------
 20251120
(1 row)


SELECT substring('tel:13812345678,status:valid' from 'tel:([0-9]{11})');     ---提取手机号信息
 substring
-----------
 13812345678
(1 row)


SELECT substring('foobar' from 'o(.)b');     --匹配o(.)b规则， o表示匹配第一个字母o， (.)表示匹配任意一个字符，并用括号捕获它， b表示匹配到字母b。 'foobar' 中，'oob' 是匹配的子串，括号位置取的是子串的中间字符，所以结果是第二个o。
 substring
--------
 o
(1 row)

SELECT substring('foobar' from '(o(.)b)');  --嵌套捕获规则，外层第一个捕获组，匹配到oob，内层第二个捕获组，匹配到第3个o，函数结果仅返回第一个捕获组，因此结果为oob
 substring
--------
 oob
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

如果POSIX正则表达式模式包含任何圆括号，那么将返回匹配第一对子表达式（对应第一个左圆括号的）的文本。如果想在表达式里使用圆括号而又不想导致这个例外，那么可以在整个表达式外边加上一对圆括号。

substring(string from *pattern* for *escape*)

描述：截取匹配SQL正则表达式的子字符串，其中string表示原始字符串，pattern表示匹配的模式，escape表示转义符，通过for进行指定。如果没有匹配到模式则返回空值。

为了标识在成功的时候应该返回的模式部分，模式必须包含转义字符的两次出现，并且每个转义字符后面要跟上双引号（"）。如果匹配这两个标记之间的模式的文本将被返回。

返回值类型：text

示例：

```
SELECT substring('Thomas' from '%#"o_a#"_' for '#');
 substring
-----------
 oma
(1 row)
```

示例解读：

1. **%#"o_a#"**_ 为匹配的模式，**#**为转义符。
2. **%**表示匹配任意数量的任意前缀的通配符，该例子匹配到前缀**Tho**。
3. #"为一个标记，该语法的目的是要匹配两个#"之间包含的文本，即**o_a**。
4. 最后一个_表示匹配到后缀字符**s**。

综上，整个匹配模式体**%#"o_a#"**_最后展开来是 % o_a _，含义变为匹配「任意前缀」+ o +「单个任意字符」+ a +「单个任意字符」。

字符串Thomas如果需要匹配上这个模式的话，即只截取两个#'之间的文本，即o_a，其中_可以表示通配任何字符，所以结果是**oma**。

拓展示例：

已知快递单号FX**5892**20251124，结构为FX + 4位单号 + 8位时间戳，实际系统只需要捕捉FX后面的4位，则可以通过以下写法获取。

```
SELECT substring('FX589220251124' from 'FX#"____#"%' for '#');
 substring
-----------
 5892
(1 row)

SELECT substring('FX564320241024' from 'FX#"____#"%' for '#');
 substring
-----------
 5643
(1 row)
```


版权所有 © 华为云计算技术有限公司

regexp_substr(text,text)

描述：正则表达式的抽取子串函数。第一个text表示待处理的原始字符串，第二个text为定义的正则表达式。与substr功能相似，正则表达式出现多个并列的括号时，也全部处理。

返回值类型：text

示例：

```
SELECT regexp_substr('str','[ac]');   ---从“str”字符串中，匹配“ac”中的任意一个字符，结果为空
 regexp_substr
---------------

(1 row)
```

示例：从混杂文本中提取第一个手机号。

```
SELECT regexp_substr('我的手机号是13812345678，备用号13987654321', '\d{11}');  -- \d{11}表示找11位连续数字
 regexp_substr
---------------
13812345678
(1 row)
```

示例：从快递单号中取前面10位订单号，10位后面的其它数字为时间戳，可以忽略。

```
SELECT regexp_substr('FX58920235822025112404', '\d{10}');   --\d{10} 表示找10位连续数字，匹配过程，从左到右扫描字符串，取第一个符合“10位连续数字”的子串。
 regexp_substr
---------------
 5892023582
(1 row)
```

regexp_substr(source_char, pattern)

描述：正则表达式的抽取子串函数，source_char表示原始字符串，pattern表示定义的正则表达式。

返回值类型：varchar

示例：

```
--匹配【一个逗号】+【至少1个非逗号字符】+【一个逗号】的连续子串
SELECT regexp_substr('500 Hello World, Redwood Shores, CA', ',[^,]+,') "REGEXPR_SUBSTR";
  REGEXPR_SUBSTR
-------------------
 , Redwood Shores,
(1 row)
```

regexp_matches(string text, pattern text [, flags text])

描述：返回string中所有匹配POSIX正则表达式的子字符串，返回结果是**文本数组**。如果pattern不匹配，该函数不返回行。这个函数在处理复杂文本解析时非常灵活，尤其适合需要多条件匹配或提取分组信息的场景。

- string为原始字符串。
- pattern为匹配模式。pattern可以是一个包含多个带圆括号的子表达式，例如**(bar)(beque)**，匹配成功后返回对应的含有多个元素的文本数组，例如**{bar,beque}**；如果pattern不包含带有圆括号的子表达式，则返回一个单一元素的文本数组，例如**{barbeque}**。

版权所有 © 华为云计算技术有限公司

- flags为可选参数，取值为i表示大小写不敏感，g表示替换每一个匹配的子字符串而不仅仅是第一个，即全局匹配。

须知

如果提供了最后一个参数flags，但参数值是空字符串（''），且数据库SQL兼容模式设置为ORA的情况下，会导致返回结果为空集。这是因为ORA兼容模式将''作为NULL处理，避免此类行为的方式有如下几种：

- 将数据库SQL兼容模式改为TD。
- 不提供最后一个参数，或最后一个参数不为空字符串。

返回值类型：多行文本数组 setof text[]

```
--匹配规则包含两个捕获分组，第一个(bar),匹配字面bar， 第二个(beque)，匹配字面beque，结果为包含2个元素的文本数组
SELECT regexp_matches('foobarbequebaz', '(bar)(beque)');
 regexp_matches
----------------
 {bar,beque}
(1 row)


--不带括号的匹配规则，结果为只包含1个元素的文本数组
SELECT regexp_matches('foobarbequebaz', 'barbeque');
 regexp_matches
----------------
 {barbeque}
(1 row)

--匹配b开头+若干非b字符，匹配结果分前后两组，即匹配到bar、beque ，但是不带g，所以只匹配了第一行
SELECT regexp_matches('foobarbequebazilbarfbonk', '(b[^b]+)(b[^b]+)');
    regexp_matches
--------------
 {bar,beque}
(1 rows)


--相比于上面的示例，带了g，则继续匹配，然后显示到第二行文本数组中
SELECT regexp_matches('foobarbequebazilbarfbonk', '(b[^b]+)(b[^b]+)', 'g');
    regexp_matches
--------------
 {bar,beque}
 {bazil,barf}
(2 rows)

--从文本中提取所有11位手机号，全局匹配，返回所有行
SELECT regexp_matches('13812345678,13987654321', '\d{11}', 'g');
 regexp_matches
----------------
 {13812345678}
 {13987654321}
(2 rows)

--忽略大小写匹配,加参数i
SELECT regexp_matches('fooBarBeQuebaz', 'barbeque', 'i');
 regexp_matches
----------------
 {BarBeQue}
(1 row)


--不带i，则区分大小写，匹配为空
SELECT regexp_matches('fooBarBeQuebaz', 'barbeque');
```

 版权所有 © 华为云计算技术有限公司

```
 regexp_matches
----------------
(0 rows)
```

⚠ 注意

如果没有子查询，当regexp_matches函数没有匹配上时，不会输出表中的数据，这通常不是所需的返回结果，应避免这种写法，建议可使用**regexp_substr(text,text)**函数来实现相同的功能。

```
SELECT * FROM tab;
 c1  | c2
-----+-----
 dws | dws
(1 row)

SELECT c1, regexp_matches(c2, '(bar)(beque)') FROM tab;
 c1 | regexp_matches
----+----------------
(0 rows)

SELECT c1, c2, (SELECT regexp_matches(c2, '(bar)(beque)')) FROM tab;
 c1  | c2  | regexp_matches
-----+-----+----------------
 dws | dws |
(1 row)
```

21.7.1.4 拼接字符串

rawcat(raw,raw)

描述：字符串拼接函数。

返回值类型：raw

示例：

```
SELECT rawcat('ab','cd');
 rawcat
--------
 ABCD
(1 row)
```

repeat(string text, number int)

描述：按照设置的次数将字符串重复输出。如果指定的重复次数为零或负数，函数返回空字符串。

返回值类型：text

示例：

```
SELECT repeat('abc', 4);
    repeat
--------------
 abcabcabcabc
(1 row)
```


版权所有 © 华为云计算技术有限公司

string || string

描述：连接字符串。

返回值类型：text

示例：

```
SELECT 'DA'||'TABASE' AS RESULT;
 result
----------
 DATABASE
(1 row)
```

string || non-string 或 non-string || string

描述：连接字符串和非字符串。

返回值类型：text

示例：

```
SELECT 'Value: '||123 AS RESULT;
   result
-------------
 Value: 123
(1 row)
```

concat(str1,str2)

描述：将字符串str1和str2连接并返回。

- ORA和TD兼容模式下，返回结果为所有非NULL字符串的连接。
- MySQL兼容模式下，入参中存在NULL时，返回结果为NULL。

返回值类型：varchar

示例：

```
SELECT concat('Hello', ' World!');
    concat
--------------
 Hello World!
(1 row)
```

concat_ws(sep text, str"any" [, str"any" [, ...]])

描述：以第一个参数为分隔符，连接第二个以后的所有参数。

返回值类型：text

示例：

```
SELECT concat_ws(',', 'ABCDE', 2, NULL, 22);
 concat_ws
------------
 ABCDE,2,22
(1 row)
```


版权所有 © 华为云计算技术有限公司

21.7.1.5 替换和填充字符串

lpad(string text, length int [, fill text])

描述：通过填充字符fill（缺省时为空白），把string填充为length长度。如果string已经比length长则将其尾部截断。

返回值类型：text

示例：

```
SELECT lpad('data', 8, 'xy');
  lpad
----------
 xyxydata
(1 row)
```

overlay(string placing string FROM int [for int])

描述：用指定的字符串替换原字符串中从指定位置开始的字符串部分，它可以灵活地处理字符串的替换操作。

参数说明：

FROM int，表示从第一个string的第几个字符开始替换。

for int，可选参数，表示替换第一个string的字符数目。如果不指定，默认替换从from开始的所有字符。

返回值类型：text

示例：从位置3开始，用“xyz'”替换原字符串中的部分，结果为“abxyzf”。

```
SELECT overlay('abcdef' placing 'xyz' FROM 3);
 overlay
---------
 abxyzf
(1 row)
```

在字符串“hello'”中，从第2个字符开始，连续替换3个字符，用“world”来覆盖这些位置，结果为“hworldo”。

```
SELECT overlay('hello' placing 'world' from 2 for 3 );
 overlay
---------
 hworldo
(1 row)
```

replace(string text, from text, to text)

描述：将原字符串中所有匹配from参数的部分替换为to参数指定的内容。

说明

replace函数大小写敏感，需确保from text和to text的大小写与原字符串匹配。

返回值类型：text

示例：

```
SELECT replace('databasedata', 'data', 'xxx');
  replace
------------
```


版权所有 © 华为云计算技术有限公司

```
 xxxbasexxx
(1 row)
```

rpad(string text, length int [, fill text])

描述：使用填充字符fill，把string填充到length长度。如果string已经比length长则将其从尾部截断。

参数说明：

- string，要处理的原字符串。
- length，目标字符串的长度。
- fill（可选参数）：用于填充的字符或字符串。缺省时为单个空格' '。

返回值类型：text

示例：

使用填充字符“xy”将长度为4的原字符串“data”填充到目标长度10。

```
SELECT rpad('data', 10, 'xy');
    rpad
------------
 dataxyxyxy
(1 row)
```

使用字符“xy”填充原字符串“database”时，因string（database）比length（4）长，则将其从尾部截断，返回字符串data。

```
SELECT rpad('database',4,'xy');
 rpad
------
 data
(1 row)
```

rtrim(string text [, characters text])

描述：去除字符串右侧（末尾）的指定字符。

参数说明：

- string，要处理的原字符串。
- characters（可选参数），指定要从字符串右侧去除的字符集。缺省时，去除所有空白字符（例如，空格等）。

返回值类型：text

示例：

未指定characters参数时默认去除字符串“Database”右侧的空白字符。

```
SELECT rtrim('Database    ');
  rtrim
----------
 Database
(1 row)
```

去除字符串中的指定字符“x”。

```
SELECT rtrim('trimxxxx', 'x');
 rtrim
-------
```


版权所有 © 华为云计算技术有限公司

```
 trim
(1 row)
```

rpad(string varchar, length int [, fill varchar])

描述：在字符串右侧填充指定字符，使字符串达到指定长度。如果string已经比length长则将其从尾部截断。

参数说明：

- string，要处理的原字符串。
- length，目标字符串的长度。length参数在DWS中表示字符长度。一个汉字长度计算为一个字符。
- fill（可选参数）：用于填充的字符或字符串。缺省时为单个空格' '。

返回值类型：varchar

示例：

```
SELECT rpad('data',10,'xyz');
 rpad
------------
 dataxyzxyz
(1 row)
SELECT rpad('hi',5,'abcdefg');
 rpad
-------
 hiabc
(1 row)
```

replace(string varchar, search_string varchar, replacement_string varchar)

描述：用于在字符串中查找特定子字符串（search_string）并将其替换为另一个字符串（replacement_string）。

参数说明：

- string，要处理的原字符串。
- search_string，需要被替换的子字符串。
- replacement_string：用于替换search_string的新字符串。

返回值类型：varchar

示例：

将字符串“ababab and mabab”中的所有子字符串“ab”替换为“cd”。

```
SELECT replace('ababab and mabab','ab','cd');
    replace
------------------
 cdcdcd and mcdcd
(1 row)
```

lpad(string varchar, length int[, repeat_string varchar])

描述：在字符串的左侧填充指定字符，使得字符串达到指定的长度。如果原字符串的长度已经大于目标长度（length参数），则从右侧将目标字符串截断为长度为length的字符串。

参数说明：

 版权所有 © 华为云计算技术有限公司

- string，需要填充的原字符串。
- length，目标字符串的长度。
- repeat_string，可选参数，用于指定填充的字符。缺省时使用空格填充。

返回值类型：varchar

示例：

原字符串“base”长度为4，目标长度为8，使用字符串“*”填充4个字符，填充后为“****base”。

```
SELECT lpad('base',8,'*');
   lpad
----------
 ****base
(1 row)
```

原字符串“database”长度为8，目标长度为6，不需要填充，从右侧将字符串“database”截断为长度为6的字符串，返回“databa”。

```
SELECT lpad('database',6,'data');
  lpad
--------
 databa
(1 row)
```

regexp_replace(string, pattern, replacement [,flags])

描述：替换匹配POSIX正则表达式的子字符串。 如果没有匹配pattern，那么返回不加修改的string串。 如果有匹配，则返回的string串里面的匹配子串将被replacement串替换掉。

replacement串可以包含\n， 其中\n是1到9， 表明string串里匹配模式里第n个圆括号子表达式的子串应该被插入， 并且它可以包含\&表示应该插入匹配整个模式的子串。

可选的flags参数包含零个或多个改变函数行为的单字母标记，见下表。

表 21-31 flags 参数的可选项

选项	描述
g	表示替换每一个匹配的子字符串而不仅仅是第一个（默认仅替换第一个匹配的子字符串）
B	默认情况下使用Henry Spencer正则库及其正则语法。指定B选项后，表示优先选用boost regex正则库及其正则语法。 以下两种情况在指定了B选项时，也会自动转换为选择Henry Spencer正则库及其正则语法： ● flags同时指定了p、q、w、x中的任意个字符。 ● string或pattern参数中含有多字节字符。
b	表示按照BRE(POSIX Basic Regular Expression)匹配模式的规则进行匹配。
c	大小写敏感匹配

 版权所有 © 华为云计算技术有限公司

选项	描述
e	表示按照ERE(POSIX Extended Regular Expression)匹配模式的规则进行匹配。当b和e都未指定时，如果选用的是Henry Spencer正则库，则按照ARE(Advanced Regular Expression，类似于Perl Regular Expression)匹配模式的规则进行匹配；如果选用的是boost regex正则库，则按照Perl Regular Expression匹配模式的规则进行匹配。
i	大小写不敏感匹配
m	换行敏感匹配，与选项n同义。
n	换行敏感匹配。此选项生效时，换行符影响元字符(.、^、$和[^)的匹配。
p	部分换行敏感匹配，此选项生效时，换行符影响元字符(.和[^)的匹配。部分是相对选项n而言。
q	重置正则表达式为加双引号的文本字符串，所有都是普通字符。
s	非换行敏感匹配。
t	紧凑语法（缺省）。该选项生效时，所有字符都很重要。
w	反部分换行敏感匹配，此选项生效时，换行符影响元字符(^和$)的匹配。部分是相对选项n而言。
x	扩展语法。与紧凑语法相对，在扩展的语法中，正则表达式中的空白字符被忽略。空白字符包括空格、水平制表符、新行、和任何属于space字符表的字符。

返回值类型：varchar

示例：

```
SELECT regexp_replace('Thomas', '.[mN]a.', 'M');
 regexp_replace
----------------
 ThM
(1 row)
SELECT regexp_replace('foobarbaz','b(..)', E'X\\1Y', 'g') AS RESULT;
   result
-------------
 fooXarYXazY
(1 row)
```

SPACE(n int)

描述：返回由n个空格组成的字符串。若参数含有NULL值，返回NULL。

返回值类型：text

示例：

```
SELECT SPACE(2) as result;
 result
--------

(1 row)
```


版权所有 © 华为云计算技术有限公司

21.7.1.6 删除指定字符

trim([leading |trailing |both] [characters] from string)

描述：删除字符串首部、尾部或者两端的特定字符。

参数说明：

- leading，仅删除字符串开头的空格或指定字符。
- trailing，仅删除字符串结尾的空格或指定字符。
- both，删除字符串开头和结尾的空格或指定字符（默认行为）。
- characters：可选参数，指定要去除的特定字符。如果不指定，默认去除空格。
- string：要处理的原字符串。

返回值类型：varchar

示例：

删除字符串“xTomxx”开头和结尾的字符“x”。

```
SELECT trim(BOTH 'x' FROM 'xTomxx');
 btrim
-------
 Tom
(1 row)

SELECT trim('x' FROM 'xTomxx');
 btrim
-------
 Tom
(1 row)
```

删除字符串“xTomxx”开头的字符“x”。

```
SELECT trim(LEADING 'x' FROM 'xTomxx');
 ltrim
-------
 Tomxx
(1 row)
```

删除字符串“xTomxx”结尾的字符“x”。

```
SELECT trim(TRAILING 'x' FROM 'xTomxx');
 rtrim
-------
 xTom
(1 row)
```

rtrim(string [, characters])

描述：从字符串尾部删除指定的字符。

参数说明：

- string：要处理的原字符串。
- characters（可选参数），指定要从字符串尾部删除的字符集。缺省时，删除所有空白字符（例如，空格等）。

返回值类型：varchar

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT rtrim('DataxBasexx','x');
   rtrim
-----------
 DataxBase
(1 row)

SELECT rtrim('    Data   ');
  rtrim
---------
    Data
(1 row)
```

ltrim(string [, characters])

描述：从字符串的开头删除指定的字符串。

参数说明：

- string：要处理的原字符串。
- characters（可选参数），指定要从字符串开头删除的字符集。缺省时，删除所有空白字符（例如，空格等）。

返回值类型：varchar

示例：

```
SELECT ltrim('xxDataxBase','x');
   ltrim
-----------
 DataxBase
(1 row)

SELECT ltrim('    Data   ');
 ltrim
--------
 Data
(1 row)
```

21.7.1.7 分割字符串

regexp_split_to_array(string text, pattern text [, flags text])

描述：用POSIX正则表达式作为分隔符，分隔string。和regexp_split_to_table相同，不过regexp_split_to_array会把它的结果以一个text数组的形式返回。

返回值类型：text[]

示例：

```
SELECT regexp_split_to_array('hello world', E'\\s+');
 regexp_split_to_array
-----------------------
 {hello,world}
(1 row)
```

regexp_split_to_table(string text, pattern text [, flags text])

描述：用POSIX正则表达式作为分隔符，分隔string。如果没有与pattern的匹配，该函数返回string。如果有至少有一个匹配，对每一个匹配它都返回从上一个匹配的末尾

版权所有 © 华为云计算技术有限公司

（或者串的开头）到这次匹配开头之间的文本。当没有更多匹配时，它返回从上一次匹配的末尾到串末尾之间的文本。

flags参数包含零个或多个改变函数行为的单字母标记。i表示进行大小写无关的匹配，g表示替换每一个匹配的子字符串而不仅仅是第一个。

返回值类型：setof text

示例：

```
SELECT regexp_split_to_table('hello world', E'\\s+');
 regexp_split_to_table
-----------------------
 hello
 world
(2 rows)
```

⚠ 注意

如果没有子查询，当regexp_split_to_table函数没有匹配上时，不会输出表中的数据。这通常不是所需的返回结果，应避免这种写法。

```
SELECT * FROM tab;
 c1  | c2
-----+-----
 dws |
(1 row)

SELECT c1, regexp_split_to_table(c2, E'\\s+') FROM tab;
 c1 | regexp_split_to_table
----+-----------------------
(0 rows)

SELECT c1, (select regexp_split_to_table(c2, E'\\s+')) FROM tab;
 c1  | regexp_split_to_table
-----+-----------------------
 dws |
(1 row)
```

split_part(string text, delimiter text, field int)

描述：将原字符串根据指定的分隔符（delimiter参数）分割成若干个部分，然后提取指定位置（field参数）的部分。该函数适用于从复合字符串中提取特定字段。

参数说明：

- string，要分割的原字符串。
- delimiter，用于分割字符串的分隔符。
- field，分割后要获取的部分，从1开始计数。

返回值类型：text

示例：

将字符串“abc~@~def~@~ghi”按照“~@~”分隔符进行分割，得到3个部分（abc，def，ghi），提取第2部分内容，即“def”。

```
SELECT split_part('abc~@~def~@~ghi', '~@~', 2);
 split_part
------------
```

 版权所有 © 华为云计算技术有限公司

```
 def
(1 row)
```

21.7.1.8 转换字符串

upper(string)

描述：把字符串转化为大写。

返回值类型：varchar

示例：

```
SELECT upper('database');
  upper
----------
 DATABASE
(1 row)
```

ucase(string)

描述：把字符串转化为大写。

返回值类型：varchar

示例：

```
SELECT ucase('database');
  ucase
----------
 DATABASE
(1 row)
```

lower(string)

描述：把字符串转化为小写。

返回值类型：varchar

示例：

```
SELECT lower('DATABASE');
  lower
----------
 database
(1 row)
```

lcase(string)

描述：把字符串转化为小写。

返回值类型：varchar

示例：

```
SELECT lcase('DATABASE');
  lcase
----------
 database
(1 row)
```


版权所有 © 华为云计算技术有限公司

initcap(string)

描述：将字符串中的每个单词的首字母转化为大写，其他字母转化为小写。

返回值类型：text

示例：

```
SELECT initcap('hi THOMAS');
  initcap
-----------
 Hi Thomas
(1 row)
```

quote_ident(string text)

描述：返回适用于SQL语句的标识符形式（使用适当的引号进行界定）。只有在必要的时候才会添加引号（字符串包含非标识符字符或者会转换大小写的字符）。返回值中嵌入的引号都写了两次。

返回值类型：text

示例：

```
SELECT quote_ident('hello world');
 quote_ident
--------------
 "hello world"
(1 row)
```

quote_literal(string text)

描述：返回适用于在SQL语句里当作文本使用的形式（使用适当的引号进行界定）。

返回值类型：text

示例：

```
SELECT quote_literal('hello');
 quote_literal
---------------
 'hello'
(1 row)
```

如果出现如下写法，text文本将进行转义。

```
SELECT quote_literal(E'O\'hello');
 quote_literal
---------------
 'O''hello'
(1 row)
```

如果出现如下写法，反斜杠会写入两次。

```
SELECT quote_literal('O\hello');
 quote_literal
---------------
 E'O\\hello'
(1 row)
```

如果参数为NULL，返回空。如果参数可能为null，通常使用函数quote_nullable更适用。

```
SELECT quote_literal(NULL);
 quote_literal
```

 版权所有 © 华为云计算技术有限公司

```
---------------

(1 row)
```

quote_literal(value anyelement)

描述：将给定的值强制转换为text，加上引号作为文本。

返回值类型：text

示例：

```
SELECT quote_literal(42.5);
 quote_literal
---------------
 '42.5'
(1 row)
```

如果出现如下写法，定值将进行转义。

```
SELECT quote_literal(E'O\'42.5');
 quote_literal
---------------
 'O''42.5'
(1 row)
```

如果出现如下写法，反斜杠会写入两次。

```
SELECT quote_literal('O\42.5');
 quote_literal
---------------
 E'O\\42.5'
(1 row)
```

quote_nullable(string text)

描述：返回适用于在SQL语句里当作字符串使用的形式（使用适当的引号进行界定）。

返回值类型：text

示例：

```
SELECT quote_nullable('hello');
 quote_nullable
----------------
 'hello'
(1 row)
```

如果出现如下写法，text文本将进行转义。

```
SELECT quote_nullable(E'O\'hello');
 quote_nullable
----------------
 'O''hello'
(1 row)
```

如果出现如下写法，反斜杠会写入两次。

```
SELECT quote_nullable('O\hello');
 quote_nullable
----------------
 E'O\\hello'
(1 row)
```

如果参数为NULL，返回NULL。

 版权所有 © 华为云计算技术有限公司

```
SELECT quote_nullable(NULL);
 quote_nullable
----------------
 NULL
(1 row)
```

quote_nullable(value anyelement)

描述：将给定的参数值转化为text，加上引号作为文本。

返回值类型：text

示例：

```
SELECT quote_nullable(42.5);
 quote_nullable
----------------
 '42.5'
(1 row)
```

如果出现如下写法，定值将进行转义。

```
SELECT quote_nullable(E'O\'42.5');
 quote_nullable
----------------
 'O''42.5'
(1 row)
```

如果出现如下写法，反斜杠会写入两次。

```
SELECT quote_nullable('O\42.5');
 quote_nullable
----------------
 E'O\\42.5'
(1 row)
```

如果参数为NULL，返回NULL。

```
SELECT quote_nullable(NULL);
 quote_nullable
----------------
 NULL
(1 row)
```

BIN(n bigint)

描述：将bigint类型从十进制转换成二进制，并以字符串的形式返回结果值。若参数含有NULL值，则返回NULL。

返回值类型：text

示例：

```
SELECT BIN(16) as result;
 result
--------
 10000
(1 row)
```

HEX(n)

描述：n可以是int类型也可以是字符串。返回n的十六进制字符串。若参数含有NULL值，返回NULL。

返回值类型：text

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT HEX(255) as result;
 result
--------
   FF
(1 row)
SELECT HEX('abc') as result;
 result
--------
 616263
(1 row)
```

UNHEX(n)

描述：执行HEX(n)的反向操作，n可以是int类型也可以是字符串，将参数中的每一对十六进制数字理解为一个数字，并将其转化为该数字代表的字符。若参数含有NULL值，返回NULL。该函数仅8.2.0及以上集群版本支持。

返回值类型：bytea

示例：

```
SELECT UNHEX('abc') as result;
 result
--------
 \x0abc
(1 row)
```

reverse(str)

描述：返回倒序排列的字符串。

返回值类型：text

示例：

```
SELECT reverse('abcde');
 reverse
---------
 edcba
(1 row)
```

to_hex(number int or bigint)

描述：将数字转换成十六进制的表现形式。

返回值类型：text

示例：

```
SELECT to_hex(2147483647);
  to_hex
----------
 7fffffff
(1 row)
```

translate(string text, from text, to text)

描述：根据from和to参数定义的字符映射关系，将原字符串中的每个字符替换为目标字符。如果from比to长，删掉在from中出现的额外的字符。

参数说明：

 版权所有 © 华为云计算技术有限公司

- string：要处理的原字符串。
- from：指定要替换的字符。
- to：指定替换后的字符。

返回值类型：text

示例：

根据from参数（143）和to参数（ax）的映射关系将原字符“12345”中的1替换为a，4替换为x， 3无对应替换字符所以删除。返回替换后的结果“a2x5”。

```
SELECT translate('12345', '143', 'ax');
 translate
-----------
 a2x5
(1 row)
```

ascii(string)

描述：返回字符串中第一个字符的ASCII码。

返回值类型：integer

示例：

返回字符串“xyz”中第一个字符“x”的ASCII码。

```
SELECT ascii('xyz');
 ascii
-------
   120
(1 row)
```

convert(string bytea, src_encoding name, dest_encoding name)

描述：以dest_encoding指定的目标编码方式转化字符串bytea。src_encoding指定源编码方式，在该编码下，string必须是合法的。

返回值类型：bytea

示例：

```
SELECT convert('text_in_utf8', 'UTF8', 'GBK');
          convert
----------------------------
 \x746578745f696e5f75746638
(1 row)
```

 版权所有 © 华为云计算技术有限公司

说明

如果源编码格式到目标编码格式的转化规则不存在，则字符串不进行任何转换直接返回，如GBK和LATIN1之间的转换规则是不存在的，具体转换规则可以通过查看系统表pg_conversion获得。

示例：

```
show server_encoding;
 server_encoding
-----------------
 LATIN1
(1 row)

SELECT convert_from('some text', 'GBK');
 convert_from
--------------
 some text
(1 row)

db_latin1=# SELECT convert_to('some text', 'GBK');
      convert_to
----------------------
 \x736f6d652074657874
(1 row)

db_latin1=# SELECT convert('some text', 'GBK', 'LATIN1');
       convert
----------------------
 \x736f6d652074657874
(1 row)
```

convert_from(string bytea, src_encoding name)

描述：以数据库的编码方式转化字符串bytea。

src_encoding指定源编码方式，在该编码下，string必须是合法的。

返回值类型：text

示例：

```
SELECT convert_from('text_in_utf8', 'UTF8');
 convert_from
--------------
 text_in_utf8
(1 row)
SELECT convert_from('\x6461746162617365','gbk');
 convert_from
--------------
 database
(1 row)
```

convert_to(string text, dest_encoding name)

描述：将字符串转化为dest_encoding的编码格式。

返回值类型：bytea

示例：

```
SELECT convert_to('some text', 'UTF8');
      convert_to
----------------------
 \x736f6d652074657874
(1 row)
SELECT convert_to('database', 'gbk');
    convert_to
```


版权所有 © 华为云计算技术有限公司

```
--------------------
 \x6461746162617365
(1 row)
```

chr(integer)

描述：将整数转换为对应的ASCII码的字符。

返回值类型：varchar

示例：

```
SELECT chr(65);
 chr
-----
 A
(1 row)
```

format(formatstr text [, str"any" [, ...]])

描述：格式化字符串。

返回值类型：text

示例：

```
SELECT format('Hello %s, %1$s', 'World');
       format
--------------------
 Hello World, World
(1 row)
```

md5(string)

描述：将字符串使用MD5加密，并以16进制数作为返回值。

说明

MD5的安全性较低，不建议使用。

返回值类型：text

示例：

```
SELECT md5('data');
               md5
----------------------------------
 8d777f385d3dfec8815d20f7496026dc
(1 row)
```

decode(string text, format text)

描述：将二进制数据从文本数据中解码。

返回值类型：bytea

示例：

```
SELECT decode('ZGF0YWJhc2U=', 'base64');
    decode
---------------
 \x6461746162617365
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
SELECT convert_from('\x6461746162617365','utf-8');
 convert_from
--------------
 database
(1 row)
```

encode(data bytea, format text)

描述：将二进制数据编码为文本数据。

返回值类型：text

示例：

```
SELECT encode('database', 'base64');
  encode
----------
 ZGF0YWJhc2U=
(1 row)
```

CONV(n, fromBase, toBase)

描述： 将给定的数值或者字符串转换成目标进制，并按照字符串的形式输出结果。若参数含有NULL值，返回NULL。进制表示范围为[-36, -2]&[2, 36]。

返回值类型：text

示例：

```
SELECT CONV(-1, 10, 16) as result;
      result
------------------
 FFFFFFFFFFFFFFFF
(1 row)
```

21.7.1.9 比较字符串

STRCMP(text, text)

描述：比较两个字符串大小，若所有的字符串均相同，则返回0，若根据当前分类次序，第一个字符串小于第二个，则返回-1，其它情况返回1。若参数含有NULL值，返回NULL。

返回值类型：text

示例：

```
SELECT STRCMP('AA', 'AA'), STRCMP('AA', 'AB'), STRCMP('AA', 'A');
STRCMP  |  STRCMP  |  STRCMP
--------------------------------
   0  |   -1   |    1
(1 row)
```

21.7.1.10 获取某一个字符或子字符串的位置

instr(text,text,int,int)

描述：获取子字符串在目标字符串中的位置。如果没有获取到匹配项，则返回0。

 版权所有 © 华为云计算技术有限公司

- 第一个text：必选，目标字符串。
- 第二个text，必选，待匹配的子字符串。
- 第一个int，表示开始匹配起始位置，若未配置该参数，默认从头开始匹配。
- 第二个int，表示匹配的次数，若未配置该参数，默认是第一次出现的位置。

返回值类型：integer

示例：

获取子字符串“bcd”在主字符串“abcdabcdabcd”中，从第2个字符开始，第2次匹配的位置。

```
SELECT instr( 'abcdabcdabcd', 'bcd', 2, 2 );
 instr
-------
     6
(1 row)

SELECT instr( 'abcdabcdabcd', 'lmk', 2);
 instr
-------
     0
(1 row)
```

position(substring in string)

描述：返回指定子字符串在字符串中的位置。若string中没有substring，则返回0。

返回值类型：integer

示例：

```
SELECT position('ing' in 'string');
 position
----------
        4
(1 row)

SELECT position('ing' in 'strin');
 position
----------
       0
(1 row)
```

strpos(string, substring)

描述：获取指定的子字符串（ substring ）在目标字符串（ string ）中首次出现的位置。子字符串不存在于目标字符串中时，返回0。

返回值类型：integer

示例：

```
SELECT strpos('source', 'rc');
 strpos
--------
      4
(1 row)

SELECT strpos('source', 'data');
 strpos
--------
      0
(1 row)
```

 版权所有 © 华为云计算技术有限公司

instr(string,substring[,position,occurrence])

描述：从字符串string的position（缺省时为1）所指的位置开始查找并返回第occurrence（缺省时为1）次出现子串substring的位置的值。

- 当position为0时，返回0。
- 当position为负数时，从字符串倒数第n个字符往前逆向搜索。n为position的绝对值。

本函数以字符为计算单位，如一个汉字为一个字符。

返回值类型：integer

示例：

```
SELECT instr('corporate floor','or', 3);
 instr
-------
     5
(1 row)
SELECT instr('corporate floor','or',-3,2);
 instr
-------
     2
(1 row)
```

locate(substring,string[,position])

描述：从字符串string的position（缺省时为1）所指的位置开始查找并返回第一次出现子串substring位置的值。以字符为计算单位。当string中不存在substring时，返回0。

返回值类型：integer

示例：

```
SELECT locate('ball','football');
 locate
--------
     5
(1 row)
SELECT locate('er','soccerplayer','6');
 locate
--------
    11
(1 row)
```

21.7.1.11 判断某个字符串是否符合某种匹配模式(字符串相关的模式匹配函数)

什么是模式匹配函数

模式匹配函数（LIKE）是一种字符串比较函数，用于判断**目标字符串**是否符合**指定的模式规则**，返回布尔值（true/false，示例中用 t/f 表示）。

它与**操作符**=的区别如下：

- =：严格匹配整个字符串（字符数、内容完全一致）。
- **LIKE**：模糊匹配（通过通配符忽略部分字符差异，只关注「关键模式」）。

适用场景：这类函数主要用于**模糊查询**，例如：

版权所有 © 华为云计算技术有限公司

- 查找姓名中包含“张”的用户：name LIKE '%张%';
- 查找手机号以“138”开头的用户：phone LIKE '138%';
- 查找邮箱后缀为@abc.com且用户名长度为5位的用户：email LIKE '_____@abc.com'（5 个下划线）。

regexp_like(text,text,text)

描述：正则表达式的模式匹配函数。

返回值类型：bool

示例：

```
SELECT regexp_like('str','[ac]');
 regexp_like
-------------
 f
(1 row)
```

string [NOT] LIKE pattern [ESCAPE escape-character]

描述：模式匹配函数。

如果pattern不包含百分号或者下划线，该模式只代表它本身，这时候LIKE的行为就像等号操作符。在pattern里的下划线（_）匹配任何单个字符；而一个百分号（%）匹配零或多个任何字符。

要匹配下划线或者百分号本身，在pattern里相应的字符必须前导逃逸字符。缺省的逃逸字符是反斜杠，但是用户可以用ESCAPE子句指定一个。要匹配逃逸字符本身，写两个逃逸字符。

返回值类型：boolean

示例：

```
SELECT 'AA_BBCC' LIKE '%A@_B%' ESCAPE '@' AS RESULT;
 result
--------
 t
(1 row)
SELECT 'AA_BBCC' LIKE '%A@_B%' AS RESULT;
 result
--------
 f
(1 row)
SELECT 'AA@_BBCC' LIKE '%A@_B%' AS RESULT;
 result
--------
 t
(1 row)
```

REGEXP_LIKE(source_string, pattern [, match_parameter])

描述：正则表达式的模式匹配函数。

source_string为源字符串，pattern为正则表达式匹配模式。 match_parameter为匹配选项，可取值为：

- 'i'：大小写不敏感。

 版权所有 © 华为云计算技术有限公司

- 'c'：大小写敏感。
- 'n'：允许正则表达式元字符“.”匹配换行符。
- 'm'：将source_string视为多行。

若忽略match_parameter选项，默认为大小写敏感，“.”不匹配换行符，source_string视为单行。

返回值类型：boolean

示例：

```
SELECT regexp_like('ABC', '[A-Z]');
 regexp_like
-------------
 t
(1 row)
SELECT regexp_like('ABC', '[D-Z]');
 regexp_like
-------------
 f
(1 row)
SELECT regexp_like('abc', '[A-Z]','i');
 regexp_like
-------------
 t
(1 row)
SELECT regexp_like('abc', '[A-Z]');
 regexp_like
-------------
 f
(1 row)
```

21.7.2 二进制字符串函数和操作符

SQL定义了一些二进制字符串函数，这些函数使用关键字而不是逗号来分隔参数。另外，DWS提供了函数调用所使用的常用语法。

octet_length(string)

描述：二进制字符串中的字节数。

返回值类型：integer

示例：

```
SELECT octet_length(E'jo\\000se'::bytea) AS RESULT;
 result
--------
      5
(1 row)
```

overlay(string placing string from int [for int])

描述：替换子串。

返回值类型：bytea

示例：

```
SELECT overlay(E'Th\\000omas'::bytea placing E'\\002\\003'::bytea from 2 for 3) AS RESULT;
     result
----------------
```


版权所有 © 华为云计算技术有限公司

```
 \x5402036d6173
(1 row)
```

position(substring in string)

描述：特定子字符串的位置。

返回值类型：integer

示例：

```
SELECT position(E'\\000om'::bytea in E'Th\\000omas'::bytea) AS RESULT;
 result
--------
      3
(1 row)
```

substring(string [from int] [for int])

描述：截取子串。

返回值类型：bytea

示例：

```
SELECT substring(E'Th\\000omas'::bytea from 2 for 3) AS RESULT;
  result
----------
 \x68006f
(1 row)
```

截取时间，获取小时数：

```
SELECT substring('2022-07-18 24:38:15',12,2)AS RESULT;
 result
------------
 24
(1 row)
```

trim([both] bytes from string)

描述：从string的开头和结尾删除只包含bytes中字节的最长字符串。

返回值类型：bytea

示例：

```
SELECT trim(E'\\000'::bytea from E'\\000Tom\\000'::bytea) AS RESULT;
  result
----------
 \x546f6d
(1 row)
```

substring_index(string, delim, count)

描述：按照区分大小写匹配查找delimiter分隔符，返回string字符串中第count次出现delim分隔符之前的子串。若count为负数，则从末尾向前查找delim分隔符。若参数含有NULL值，返回NULL。该函数仅8.2.0及以上集群版本支持。

返回值类型：text

示例：按照区分大小写匹配查找delimiter分隔符".wWw."，返回string字符串"www.wWw.cloud.wWw.com"中第2次出现delimiter分隔符之前的子串"www.wWw.cloud"。

 版权所有 © 华为云计算技术有限公司

```
SELECT SUBSTRING_INDEX('www.wWw.cloud.wWw.com', '.wWw.', 2) AS RESULT;
    result
---------------
 www.wWw.cloud
(1 row)
```

btrim(string bytea,bytes bytea)

描述：从string的开头和结尾删除只包含bytes中字节的最长的字符串。

返回值类型：bytea

示例：

```
SELECT btrim(E'\\000trim\\000'::bytea, E'\\000'::bytea) AS RESULT;
  result
------------
 \x7472696d
(1 row)
```

get_bit(string, offset)

描述：从字符串中抽取位。

返回值类型：integer

示例：

```
SELECT get_bit(E'Th\\000omas'::bytea, 45) AS RESULT;
 result
--------
      1
(1 row)
```

get_byte(string, offset)

描述：从字符串中抽取字节。

返回值类型：integer

示例：

```
SELECT get_byte(E'Th\\000omas'::bytea, 4) AS RESULT;
 result
--------
    109
(1 row)
```

set_bit(string,offset, newvalue)

描述：设置字符串中的位。

返回值类型：bytea

示例：

```
SELECT set_bit(E'Th\\000omas'::bytea, 45, 0) AS RESULT;
      result
------------------
 \x5468006f6d4173
(1 row)
```

 版权所有 © 华为云计算技术有限公司

set_byte(string,offset, newvalue)

描述：设置字符串中的字节。

返回值类型：bytea

示例：

```
SELECT set_byte(E'Th\\000omas'::bytea, 4, 64) AS RESULT;
      result
------------------
 \x5468006f406173
(1 row)
```

21.7.3 位串函数和操作符

除了常用的比较操作符之外，还可以使用以下的操作符。&，|和#的位串操作数必须等长。在位移的时候，保留原始的位串长度（并以0填充）。

||

描述：位串之间进行连接。

示例：

```
SELECT B'10001' || B'011' AS RESULT;
  result
----------
 10001011
(1 row)
```

&

描述：位串之间进行“与”操作。

示例：

```
SELECT B'10001' & B'01101' AS RESULT;
 result
--------
 00001
(1 row)
```

|

描述：位串之间进行“或”操作。

示例：

```
SELECT B'10001' | B'01101' AS RESULT;
 result
--------
 11101
(1 row)
```

#

描述：位串之间如果不一致进行“或”操作。如果两个位串中对应位置都为1或者0，则该位置返回为0。

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT B'10001' # B'01101' AS RESULT;
 result
--------
 11100
(1 row)
```

~

描述：位串之间进行“非”操作。

示例：

```
SELECT ~B'10001'AS RESULT;
 result
----------
 01110
(1 row)
```

<<

描述：位串进行左移操作。

示例：

```
SELECT B'10001' << 3 AS RESULT;
 result
----------
 01000
(1 row)
```

>>

描述：位串进行右移操作。

示例：

```
SELECT B'10001' >> 2 AS RESULT;
 result
----------
 00100
(1 row)
```

下列SQL标准函数除了可以用于字符串之外，也可以用于位串：length，bit_length，octet_length，position，substring，overlay。

下列的函数用于位串和二进制字符串：get_bit，set_bit。当用于位串时，函数位数从字符串的第一位（最左边）作为0位。

另外，在整数和bit之间可以相互转换。示例：

```
SELECT 44::bit(10) AS RESULT;
   result
------------
 0000101100
(1 row)

SELECT 44::bit(3) AS RESULT;
 result
--------
 100
(1 row)

SELECT cast(-44 as bit(12)) AS RESULT;
    result
--------------
 111111010100
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
SELECT '1110'::bit(4)::integer AS RESULT;
 result
--------
     14
(1 row)
```

说明

只是转换为“bit”的意思是转换成bit(1)，因此只会转换成整数的最低位。

21.7.4 数字操作函数和操作符

21.7.4.1 数字操作符

+

描述：加

示例：

```
SELECT 2+3 AS RESULT;
 result
--------
      5
(1 row)
```

-

描述：减

示例：

```
SELECT 2-3 AS RESULT;
 result
--------
     -1
(1 row)
```

*****

描述：乘

示例：

```
SELECT 2*3 AS RESULT;
 result
--------
      6
(1 row)
```

/

描述：除（除法操作符不会取整）

示例：

```
SELECT 4/2 AS RESULT;
 result
--------
```


版权所有 © 华为云计算技术有限公司

```
     2
(1 row)
SELECT 4/3 AS RESULT;
      result
------------------
 1.33333333333333
(1 row)
```

+/-

描述：正/负

示例：

```
SELECT -2 AS RESULT;
 result
--------
     -2
(1 row)
```

%

描述：模（求余）

示例：

```
SELECT 5%4 AS RESULT;
 result
--------
      1
(1 row)
```

@

描述：绝对值

示例：

```
SELECT @ -5.0 AS RESULT;
 result
--------
    5.0
(1 row)
```

^

描述：幂（指数运算）

MySQL兼容模式下，作用为异或，参见**位串函数和操作符**章节的操作符“**#**”。

示例：

```
SELECT 2.0^3.0 AS RESULT;
       result
--------------------
 8.0000000000000000
(1 row)
```

|/

描述：平方根

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT |/ 25.0 AS RESULT;
 result
--------
      5
(1 row)
```

||/

描述：立方根

示例：

```
SELECT ||/ 27.0 AS RESULT;
 result
--------
      3
(1 row)
```

!

描述：阶乘

示例：

```
SELECT 5! AS RESULT;
 result
--------
    120
(1 row)
```

!!

描述：阶乘（前缀操作符）

示例：

```
SELECT !!5 AS RESULT;
 result
--------
    120
(1 row)
```

&

描述：二进制AND

示例：

```
SELECT 91&15  AS RESULT;
 result
--------
     11
(1 row)
```

|

描述：二进制OR

示例：

```
SELECT 32|3  AS RESULT;
 result
--------
```


版权所有 © 华为云计算技术有限公司

```
    35
(1 row)
```

#

描述：二进制XOR

示例：

```
SELECT 17#5  AS RESULT;
 result
--------
    20
(1 row)
```

~

描述：二进制NOT

示例：

```
SELECT ~1 AS RESULT;
 result
--------
    -2
(1 row)
```

<<

描述：二进制左移

示例：

```
SELECT 1<<4 AS RESULT;
 result
--------
    16
(1 row)
```

>>

描述：二进制右移

示例：

```
SELECT 8>>2 AS RESULT;
 result
--------
     2
(1 row)
```

21.7.4.2 数字操作函数

abs(x)

描述：绝对值。

返回值类型：和输入相同。

示例：

```
SELECT abs(-17.4);
 abs
```

 版权所有 © 华为云计算技术有限公司

```
------
 17.4
(1 row)
```

acos(x)

描述：反余弦。

返回值类型：double precision

示例：

```
SELECT acos(-1);
       acos
------------------
 3.14159265358979
(1 row)
```

asin(x)

描述：反正弦。

返回值类型：double precision

示例：

```
SELECT asin(0.5);
       asin
------------------
 0.523598775598299
(1 row)
```

atan(x)

描述：反正切。

返回值类型：double precision

示例：

```
SELECT atan(1);
       atan
------------------
 0.785398163397448
(1 row)
```

atan2(y, x)

描述：y/x的反正切。

返回值类型：double precision

示例：

```
SELECT atan2(2, 1);
      atan2
------------------
 1.10714871779409
(1 row)
```

bitand(integer, integer)

描述：计算两个数字与运算(&)的结果。

 版权所有 © 华为云计算技术有限公司

返回值类型：bigint

示例：

```
SELECT bitand(127, 63);
 bitand
--------
     63
(1 row)
```

cbrt(double precision)

描述：立方根。

返回值类型：double precision

示例：

```
SELECT cbrt(27.0);
 cbrt
------
    3
(1 row)
```

ceil(double precision or numeric)

描述：不小于参数的最小的整数。

返回值类型：与输入相同。

示例：

```
SELECT ceil(-42.8);
 ceil
------
  -42
(1 row)
```

ceiling(double precision or numeric)

描述：不小于参数的最小整数（ceil的别名）。

返回值类型：与输入相同。

示例：

```
SELECT ceiling(-95.3);
 ceiling
---------
     -95
(1 row)
```

cos(x)

描述：余弦。

返回值类型：double precision

示例：

```
SELECT cos(-3.1415927);
        cos
--------------------
```

 版权所有 © 华为云计算技术有限公司

```
 -0.999999999999999
(1 row)
```

cot(x)

描述：余切。

返回值类型：double precision

示例：

```
SELECT cot(1);
       cot
------------------
 0.642092615934331
(1 row)
```

degrees(double precision)

描述：把弧度转为角度。

返回值类型：double precision

示例：

```
SELECT degrees(0.5);
     degrees
------------------
 28.6478897565412
(1 row)
```

div(y numeric, x numeric)

描述：y除以x的商的整数部分。

返回值类型：numeric

示例：

```
SELECT div(9,4);
 div
-----
   2
(1 row)
```

exp(double precision or numeric)

描述：自然指数。

返回值类型：与输入相同。

示例：

```
SELECT exp(1.0);
        exp
--------------------
 2.7182818284590452
(1 row)
```

floor(double precision or numeric)

描述：不大于参数的最大整数。

 版权所有 © 华为云计算技术有限公司

返回值类型：与输入相同。

示例：

```
SELECT floor(-42.8);
 floor
-------
   -43
(1 row)
```

radians(double precision)

描述：把角度转为弧度。

返回值类型：double precision

示例：

```
SELECT radians(45.0);
     radians
-------------------
 0.785398163397448
(1 row)
```

random()

描述：0.0到1.0之间的随机数。

返回值类型：double precision

示例：

```
SELECT random();
      random
-------------------
 0.824823560658842
(1 row)
```

rand()

描述：0.0到1.0之间的随机数。此函数为Mysql兼容性函数。该函数仅8.2.0及以上集群版本支持。

返回值类型：double precision

示例：

```
SELECT rand();
      rand
-------------------
 0.824823560658842
(1 row)
```

ln(double precision or numeric)

描述：自然对数。

返回值类型：与输入相同。

示例：

```
SELECT ln(2.0);
        ln
--------------------
```


版权所有 © 华为云计算技术有限公司

```
 0.6931471805599453
(1 row)
```

log(double precision or numeric)

描述：以10为底的对数。

- ORA和TD兼容模式下，表现为以10为底的对数。
- MySQL兼容模式下，表现为自然对数。

返回值类型：与输入相同。

示例：

```
-- ORA兼容模式
SELECT log(100.0);
        log
--------------------
 2.0000000000000000
(1 row)
-- TD兼容模式
SELECT log(100.0);
        log
--------------------
 2.0000000000000000
(1 row)
-- MySQL兼容模式
SELECT log(100.0);
        log
--------------------
 4.6051701859880914
(1 row)
```

log(b numeric, x numeric)

描述：以b为底的对数。

返回值类型：numeric

示例：

```
SELECT log(2.0, 64.0);
        log
--------------------
 6.0000000000000000
(1 row)
```

mod(x,y)

描述：x/y的余数（模）。如果x是0，则返回0。如果y是0，则返回x。

返回值类型：与参数类型相同。

示例：

```
SELECT mod(9,4);
 mod
-----
   1
(1 row)
SELECT mod(9,0);
 mod
-----
   9
(1 row)
```

 版权所有 © 华为云计算技术有限公司

pi()

描述：“π”常量。

返回值类型：double precision

示例：

```
SELECT pi();
       pi
------------------
 3.14159265358979
(1 row)
```

power(a double precision, b double precision)

描述：a的b次幂。

返回值类型：double precision

示例：

```
SELECT power(9.0, 3.0);
       power
----------------------
 729.0000000000000000
(1 row)
```

round(double precision or numeric)

描述：离输入参数最近的整数。

返回值类型：与输入相同。

示例：

```
SELECT round(42.4);
 round
-------
    42
(1 row)

SELECT round(42.6);
 round
-------
    43
(1 row)
```

说明

当调用round函数时，数值类型将舍入到零，而（在大多数计算机上）实数和双精度型，以最接近的偶数为结果。

round(v numeric, s int)

描述：保留小数点后s位，s后一位进行四舍五入。

返回值类型：numeric

示例：

```
SELECT round(42.4382, 2);
 round
-------
```

 版权所有 © 华为云计算技术有限公司

```
 42.44
(1 row)
```

setseed(double precision)

描述：为随后的random()调用设置种子(-1.0到1.0之间，包含边界值)。

返回值类型：void

示例：

```
SELECT setseed(0.54823);
 setseed
---------

(1 row)
```

sign(double precision or numeric)

描述：输出此参数的符号。

返回值类型：-1表示负数，0表示0，1表示正数。

示例：

```
SELECT sign(-8.4);
 sign
------
   -1
(1 row)
```

sin(x)

描述：正弦。

返回值类型：double precision

示例：

```
SELECT sin(1.57079);
       sin
------------------
 0.999999999979986
(1 row)
```

sqrt(double precision or numeric)

描述：平方根。

返回值类型：与输入相同。

示例：

```
SELECT sqrt(2.0);
       sqrt
------------------
 1.414213562373095
(1 row)
```

tan(x)

描述：正切。

 版权所有 © 华为云计算技术有限公司

返回值类型：double precision

示例：

```
SELECT tan(20);
       tan
------------------
 2.23716094422474
(1 row)
```

trunc(double precision or numeric)

描述：截断（取整数部分）。

返回值类型：与输入相同。

示例：

```
SELECT trunc(42.8);
 trunc
-------
    42
(1 row)
```

trunc(v numeric, s int)

描述：截断为s位小数。

返回值类型：numeric

示例：

```
SELECT trunc(42.4382, 2);
 trunc
-------
 42.43
(1 row)
```

truncate(v numeric, s int)

描述：将v截断为s位小数。v除了可以是任意精度型，还可以是整型和浮点型，返回值与入参v类型相同。s为负数时截断整数部分。该函数仅8.2.0及以上集群版本支持。

返回值类型：numeric

示例：

```
SELECT truncate(42.4382, 2);
 truncate
-------
 42.4300
(1 row)
```

width_bucket(operand numeric, b1 numeric, b2 numeric, count int)

描述：设定分组范围的最小值、最大值和分组个数，构建指定个数的大小相同的分组，返回指定字段值落入的分组编号。b1为分组范围的最小值，b2为分组范围的最大值，count为分组的个数。

返回值类型：integer

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT width_bucket(5.35, 0.024, 10.06, 5);
 width_bucket
--------------
            3
(1 row)
```

width_bucket(operand double precision, b1 double precision, b2 double precision, count int)

描述：设定分组范围的最小值、最大值和分组个数，构建指定个数的大小相同的分组，返回指定字段值落入的分组编号。b1为分组范围的最小值，b2为分组范围的最大值，count为分组的个数。

返回值类型：integer

示例：

```
SELECT width_bucket(5.35, 0.024, 10.06, 5);
 width_bucket
--------------
            3
(1 row)
```

21.7.5 时间、日期处理函数和操作符

21.7.5.1 时间/日期操作符

须知

用户在使用时间和日期操作符时，对应的操作数请使用明确的类型前缀修饰，以确保数据库在解析操作数的时候能够与用户预期一致，不会产生用户非预期的结果。

比如下面示例没有明确数据类型就会出现异常错误。

```
SELECT date '2001-10-01' - '7' AS RESULT;
```

表 21-32 时间和日期操作符

操作符	用法示例	示例结果
+	date + integer → timestamp 给日期加上天数。 `SELECT date '2001-09-28' + integer '7' AS RESULT;`	date类型参数与integer参数相加，获取时间间隔为7天后的时间： `2001-10-05 00:00:00`

版权所有 © 华为云计算技术有限公司

操作符	用法示例	示例结果
+	date + interval → timestamp 为日期添加时间间隔。 SELECT date '2001-09-28' + interval '1 hour' AS RESULT; SELECT date '2021-02-28' + interval '1 month' AS RESULT; SELECT date '2021-01-31' + interval '1 month' AS RESULT;	date类型参数与interval参数相加。 获取时间间隔为1小时后的时间： 2001-09-28 01:00:00 获取时间间隔为1个月的时间： 2021-02-28 00:00:00 date函数对于日期相加减超过月份的日期范围，会对齐到对应月份最后一天，不超过则不处理。 2021-03-28 00:00:00
+	date + time → timestamp 在日期中添加一天中的时间。 SELECT date '2001-09-28' + time '03:00' AS RESULT;	date类型参数与time类型参数相加，获取具体的日期和时间结果： 2001-09-28 03:00:00
+	interval + interval → interval 添加时间间隔。 SELECT interval '1 day' + interval '1 hour' AS RESULT;	interval参数相加，获取两个时间间隔之和： 1 day 01:00:00
+	timestamp + interval → timestamp 在时间戳中添加一个时间间隔。 SELECT timestamp '2001-09-28 01:00' + interval '23 hours' AS RESULT;	timestamp时间类型参数与interval参数相加，获取间隔23小时后的时间： 2001-09-29 00:00:00
+	time + interval → time 为时间添加时间间隔。 SELECT time '01:00' + interval '3 hours' AS RESULT;	time类型参数与interval参数相加，获取间隔时间为3小时后的时间： 04:00:00
-	date - date → integer 减去日期，生成日期时间差。 SELECT date '2001-10-01' - date '2001-09-28' AS RESULT;	date类型参数相减，获取两个日期的时间差： 3 days
-	date - integer → timestamp 从日期中减去天数。 SELECT date '2001-10-01' - integer '7' AS RESULT;	date类型参数与integer参数相减，获取两者的时间差： 2001-09-24 00:00:00
-	date - interval → timestamp 从日期中减去时间间隔。 SELECT date '2001-09-28' - interval '1 hour' AS RESULT;	date类型参数与interval参数相减，获取两者的日期、时间差： 2001-09-27 23:00:00
-	time - time → interval 减去时间。 SELECT time '05:00' - time '03:00' AS RESULT;	time类型参数相减，获取两参数的时间差： 02:00:00

版权所有 © 华为云计算技术有限公司

操作符	用法示例	示例结果
-	time - interval → time 从时间中减去时间间隔。 SELECT time '05:00' - interval '2 hours' AS RESULT;	time类型参数与interval相减，获取两参数的时间差： 03:00:00
-	timestamp - interval → timestamp 从时间戳中减去时间间隔。 SELECT timestamp '2001-09-28 23:00' - interval '23 hours' AS RESULT;	timestamp类型参数与interval相减，获取两者的日期时间差： 2001-09-28 00:00:00
-	interval - interval → interval 减去时间间隔。 SELECT interval '1 day' - interval '1 hour' AS RESULT;	interval参数相减，获取两者的时间差： 23:00:00
-	timestamp - timestamp → interval 减去时间戳。 SELECT timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00' AS RESULT;	timestamp类型参数相减，获取两者的日期时间差： 1 day 15:00:00
-	select now() - interval → timestamp with time zone 获取当前日期的前一天： select now() - interval '1 day' AS RESULT;	获取当前日期的前一天： 2025-07-29 10:54:29.116501+08
*	interval * double precision → interval 将时间间隔乘以数量： SELECT 900 * interval '1 second' AS RESULT; SELECT 21 * interval '1 day' AS RESULT; SELECT double precision '3.5' * interval '1 hour' AS RESULT;	间隔1秒乘以900。 00:15:00 间隔1天乘以21天。 21 days 间隔1小时乘以3.5。 03:30:00
/	interval / double precision → interval SELECT interval '1 hour' / double precision '1.5' AS RESULT;	用时间间隔除以数量，获取一段时间中的某一段： 00:40:00

21.7.5.2 时间/日期函数

age(timestamp, timestamp)

描述：将两个参数相减，并以年、月、日作为返回值。若相减值为负，则函数返回亦为负。

返回值类型：interval

示例：

```
SELECT age(timestamp '2001-04-10', timestamp '1957-06-13');
          age
--------------------------
 43 years 9 mons 27 days
(1 row)
```


版权所有 © 华为云计算技术有限公司

age(timestamp)

描述：当前时间和参数相减。

返回值类型：interval

示例：

```
SELECT age(timestamp '1957-06-13');
          age
-------------------------
 60 years 2 mons 18 days
(1 row)
```

adddate(date, interval | int)

描述：返回给定日期时间加上指定单位的时间间隔的结果。默认单位(即第二个参数为整型时)为天数。

返回值类型：timestamp

示例：

当入参为text类型时：

```
select adddate('2020-11-13', 10);
  adddate
------------
 2020-11-23
(1 row)

select adddate('2020-11-13', interval '1' month);
  adddate
------------
 2020-12-13
(1 row)

select adddate('2020-11-13 12:15:16', interval '1' month);
       adddate
---------------------
 2020-12-13 12:15:16
(1 row)


select adddate('2020-11-13', interval '1' minute);
       adddate
---------------------
 2020-11-13 00:01:00
(1 row)
```

当入参为date类型时：

```
select adddate(current_date, 10);
  adddate
------------
 2021-09-24
(1 row)

select adddate(date '2020-11-13', interval '1' month);
       adddate
---------------------
 2020-12-13 00:00:00
(1 row)
```


版权所有 © 华为云计算技术有限公司

addtime(timestamp | time | text, interval | text)

描述：返回给定日期/时间加上指定时间间隔的结果。该函数仅8.2.0及以上集群版本支持。

返回值类型：与第一个入参类型相同。

示例：

```
SELECT ADDTIME('2020-11-13 01:01:01', '23:59:59');
       addtime
---------------------
 2020-11-14 01:01:00
(1 row)
```

addtime(time, interval)

```
SELECT ADDTIME('20:1:1'::TIME, '20:59:59+8');
  addtime
-----------
 233:01:00
(1 row)

SELECT ADDTIME('20:1:1'::TIME, '8 20:59:59');
  addtime
-----------
 233:01:00
(1 row)
```

addtime(text, interval)

```
SELECT ADDTIME('20:1:1', '20:59:59+8');
  addtime
-----------
 233:01:00
(1 row)
SELECT ADDTIME('20:1:1', '8 20:59:59');
  addtime
-----------
 233:01:00
(1 row)
```

subdate(date, interval | int)

描述：返回给定日期时间减去指定单位的时间间隔的结果; 默认单位(即第二个参数为整型时)为天数。

返回值类型：timestamp

示例：

当入参为text类型时：

```
select subdate('2020-11-13', 10);
  subdate
------------
 2020-11-03
(1 row)

select subdate('2020-11-13', interval '2' month);
  subdate
------------
 2020-09-13
(1 row)

select subdate('2020-11-13 12:15:16', interval '1' month);
       subdate
```


版权所有 © 华为云计算技术有限公司

```
---------------------
 2020-10-13 12:15:16
(1 row)

select subdate('2020-11-13', interval '2' minute);
       subdate
---------------------
 2020-11-12 23:58:00
(1 row)
```

当入参为date类型时：

```
select subdate(current_date, 10);
  subdate
------------
 2021-09-05
(1 row)

select subdate(current_date, interval '1' month);
       subdate
---------------------
 2021-08-15 00:00:00
(1 row)
```

subtime(timestamp | time | text, interval | text)

描述：返回给定日期/时间减去指定时间间隔的结果。该函数仅8.2.0及以上集群版本支持。

返回值类型：与第一个入参类型相同。

示例：

```
select subtime('2020-11-13 01:01:01', '23:59:59');
       addtime
---------------------
 2020-11-12 01:01:02
(1 row)
```

date_add(date, interval)

描述：返回给定日期时间加上指定单位的时间间隔的结果。等效于**adddate(date, interval | int)**。

返回值类型：timestamp

date_sub(date, interval)

描述：返回给定日期时间减去指定单位的时间间隔的结果，等效于**subdate(date, interval | int)**。

返回值类型：timestamp

timestampadd(field, numeric, timestamp)

描述：将以单位field的整数时间间隔（秒数可以带小数）添加到日期时间表达式中。若数值为负，则表示从给定的时间日期时间表达式中减去对应的时间间隔。field支持的参数为year，month，quarter，day，week，hour，minute，second和microsecond。

版权所有 © 华为云计算技术有限公司

说明

当函数传入的时间表达式格式为yyyy-mon-day+{value}的格式时，其中{value}字段将被解析为时区表达式，表示当前时间的时区，如'2023-01-31+1'将被解析为东一区时区的2023年01月31日。DWS这种表达式传入格式的解析与MySQL解析规则不一致，MySQL数据库仅当开启宽松解析规则才能支持此类格式，且解析结果为系统时区的2023年01月31日1时。

返回值类型：timestamp

示例：

```
select timestampadd(year, 1, timestamp '2020-2-29');
    timestampadd
---------------------
 2021-02-28 00:00:00
(1 row)

select timestampadd(second, 2.354156, timestamp '2020-11-13');
        timestampadd
----------------------------
 2020-11-13 00:00:02.354156
(1 row)
```

timestampdiff(field, timestamp1, timestamp2)

描述：将两个日期参数相减(timestamp2 - timestamp1)，并以单位field作为返回值。若相减值为负，则函数返回值为负。field支持的参数为year、month、quarter、day、week、hour、minute、second和microsecond。

返回值类型：bigint

示例：

```
SELECT timestampdiff(day, timestamp '2001-02-01', timestamp '2003-05-01 12:05:55');
 timestampdiff
---------------
     819
(1 row)
```

timediff(timestamp | time | text, timestamp | time | text)

描述：将两个日期参数相减。若相减值为负，则函数返回值为负。两个入参类型要求一致。该函数仅8.2.0及以上集群版本支持。

返回值类型：time

示例：

```
SELECT timediff('2022-7-5 1:1:1', '2021-7-5 1:1:1');
   timediff
---------------
  8760:00:00
(1 row)
```

clock_timestamp()

描述：实时时钟的当前时间戳。

返回值类型：timestamp with time zone

示例：

```
SELECT clock_timestamp();
       clock_timestamp
```

 版权所有 © 华为云计算技术有限公司

```
-------------------------------
 2017-09-01 16:57:36.636205+08
(1 row)
```

current_date

描述：当前日期。

返回值类型：date

示例：

```
SELECT current_date;
    date
------------
 2017-09-01
(1 row)
```

curdate()

描述：当前日期。此函数为Mysql兼容性函数。该函数仅8.2.0及以上集群版本支持。

返回值类型：date

示例：

```
SELECT curdate();
   curdate
------------
 2022-09-19
(1 row)
```

current_time

描述：当前时间。

返回值类型：time with time zone

示例：

```
SELECT current_time;
       timetz
--------------------
 16:58:07.086215+08
(1 row)
```

curtime([fsp])

描述：当前时间。fsp为选填参数，参数类型为整型，表示指定返回的小数秒精度。该函数仅8.2.0及以上集群版本支持。

返回值类型：time with time zone

示例：

```
SELECT curtime();
       timetz
--------------------
 16:58:07.086215+08
(1 row)
SELECT curtime(2);
       timetz
--------------------
```


版权所有 © 华为云计算技术有限公司

```
 16:58:07.08+08
(1 row)
```

current_timestamp

描述：当前日期及时间。

返回值类型：timestamp with time zone

示例：

```
SELECT current_timestamp;
       pg_systimestamp
-------------------------------
 2017-09-01 16:58:19.22173+08
(1 row)
```

convert_tz(timestamp, from_tz, to_tz)

描述：将时间日期值timestamp从*from_tz* (text)给出的时区转到*to_tz*(text)给出的时区，返回转换后的时区时间。该函数仅8.2.0及以上集群版本支持。

返回值类型：timestamp without time zone

示例：

```
SELECT convert_tz('2018-12-25 13:25:00', '+02:00', '+08:00');
    convert_tz
--------------------
 2018-12-25 19:25:00
(1 row)
SELECT convert_tz('2000-02-28 23:00:00', 'GMT', 'MET');
    convert_tz
--------------------
 2000-02-29 00:00:00
(1 row)
```

datediff(date1, date2)

描述：返回给定日期之间相差的天数值。

返回值类型：bigint

示例：

```
select datediff(date '2020-11-13', date '2012-10-16');
 datediff
----------
     2950
(1 row)
```

date_part(text, timestamp)

描述：获取参数text指定的精度。

等效于extract(field from timestamp)。

返回值类型：double precision

示例：

```
SELECT date_part('hour', timestamp '2001-02-16 20:38:40');
 date_part
-----------
```

 版权所有 © 华为云计算技术有限公司

```
      20
(1 row)
```

date_part(text, interval)

描述：获取参数text指定的精度。如果大于12，则取与12的模。

等效于extract(field from timestamp)。

返回值类型：double precision

示例：

```
SELECT date_part('month', interval '2 years 3 months');
 date_part
-----------
         3
(1 row)
```

date_trunc(text, timestamp with time zone | timestamp without time zone | interval)

描述：将时间日期数据截断到指定的（text）精度。其中，text支持microseconds、milliseconds、second、minute、hour、day、week、month、quarter、year、decade、century、millennium。

返回值类型：timestamp with time zone、timestamp without time zone、interval

示例：

将输入的时间截断到小时。

```
SELECT date_trunc('hour', timestamp '2025-02-16 20:38:40');
     date_trunc
---------------------
 2025-02-16 20:00:00
(1 row)
```

将输入的时间截断到天。

```
SELECT date_trunc('day', timestamptz'2025-02-16 20:38:40+08');
       date_trunc
-------------------------
 2025-02-16 00:00:00-05
(1 row)
```

将输入的时间截断到月。

```
SELECT date_trunc('month', timestamp '2025-02-16 20:38:40');
     date_trunc
---------------------
 2025-02-01 00:00:00
(1 row)
```

将输入时间戳截断至一个季度的第一天。

```
SELECT DATE_TRUNC('quarter', TIMESTAMP '20250430 04:05:06.789');
     date_trunc
---------------------
 2025-04-01 00:00:00
(1 row)
```

截取当前月的1号12点。

```
SELECT date_trunc('month',now()) +interval '12h';
       ?column?
```

 版权所有 © 华为云计算技术有限公司

```
------------------------
 2025-09-01 12:00:00-04
(1 row)
```

截取当日的9点。

```
SELECT date_trunc('day',now()) + interval '9h';
       ?column?
------------------------
 2025-09-22 09:00:00-04
(1 row)
```

截取今年的第一天。

```
SELECT date_trunc('year',CURRENT_DATE);
      date_trunc
------------------------
 2025-01-01 00:00:00-05
(1 row)
```

trunc(timestamp)

描述：默认按天截取。

返回值类型：timestamp

示例：

```
SELECT trunc(timestamp  '2001-02-16
20:38:40');
                trunc
---------------------
2001-02-16 00:00:00
(1 row)
```

extract(field from timestamp)

描述：获取field指定精度的值。field的有效值参见**EXTRACT**。

返回值类型：double precision

示例：

```
SELECT extract(hour from timestamp '2001-02-16 20:38:40');
 date_part
-----------
       20
(1 row)
```

extract(field from interval)

描述：获取field指定精度的值。如果大于12，则取与12的模。field的有效值参见**EXTRACT**。

返回值类型：double precision

示例：

```
SELECT extract(month from interval '2 years 3 months');
 date_part
-----------
        3
(1 row)
```

 版权所有 © 华为云计算技术有限公司

day(date)

描述：获取日期时间date所处月份中的天数，与dayofmonth函数相同。

取值范围：1~31

返回值类型：integer

示例：

```
select day('2020-06-28');
 day
-----
  28
(1 row)
```

dayofmonth(date)

描述：获取日期时间date所处月份中的天数。

取值范围：1~31

返回值类型：integer

示例：

```
select dayofmonth('2020-06-28');
 dayofmonth
------------
         28
(1 row)
```

dayofweek(date)

描述：返回给定日期date对应的星期索引，星期日作为一周的开始日。

取值范围：1~7

返回值类型：integer

示例：

```
select dayofweek('2020-11-22');
 dayofweek
-----------
         1
(1 row)
```

dayofyear(date)

描述：返回给定日期date在本年中的天数。

取值范围：1~366

返回值类型：integer

示例：

```
select dayofyear('2020-02-29');
 dayofyear
-----------
        60
(1 row)
```

 版权所有 © 华为云计算技术有限公司

hour(timestamp with time zone)

描述：获取时间中的小时值。

返回值类型：integer

示例：

```
SELECT hour(timestamptz '2018-12-13 12:11:15+06');
 hour
------
   14
(1 row)
```

isfinite(date)

描述：测试是否为有限日期。

返回值类型：boolean

示例：

```
SELECT isfinite(date '2001-02-16');
 isfinite
----------
 t
(1 row)
SELECT isfinite(date 'infinity');
 isfinite
----------
 f
(1 row)
```

isfinite(timestamp)

描述：测试判断是否为有限时间。

返回值类型：boolean

示例：

```
SELECT isfinite(timestamp '2001-02-16 21:28:30');
 isfinite
----------
 t
(1 row)
SELECT isfinite(timestamp 'infinity');
 isfinite
----------
 f
(1 row)
```

isfinite(interval)

描述：测试是否为有限区间。

返回值类型：boolean

示例：

```
SELECT isfinite(interval '4 hours');
 isfinite
----------
 t
(1 row)
```


版权所有 © 华为云计算技术有限公司

justify_days(interval)

描述：将时间间隔以30天为单位，表示为月。

返回值类型：interval

示例：

```
SELECT justify_days(interval '35 days');
 justify_days
--------------
 1 mon 5 days
(1 row)
```

justify_hours(interval)

描述：将时间间隔以24小时为单位，表示为天。

返回值类型：interval

示例：

```
SELECT JUSTIFY_HOURS(INTERVAL '27 HOURS');
 justify_hours
----------------
 1 day 03:00:00
(1 row)
```

justify_interval(interval)

描述：结合justify_days和justify_hours，调整interval。

返回值类型：interval

示例：

```
SELECT JUSTIFY_INTERVAL(INTERVAL '1 MON -1 HOUR');
 justify_interval
------------------
 29 days 23:00:00
(1 row)
```

localtime

描述：当前时间。

返回值类型：time

示例：

```
SELECT localtime AS RESULT;
     result
----------------
 16:05:55.664681
(1 row)
```

localtimestamp

描述：当前日期及时间。

返回值类型：timestamp

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT localtimestamp;
         timestamp
----------------------------
 2017-09-01 17:03:30.781902
(1 row)
```

makedate(year, dayofyear)

描述：根据给定的年份和一年中的天数返回相对应的日期值。

返回值类型：date

示例：

```
select makedate(2020, 60);
  makedate
------------
 2020-02-29
(1 row)
```

maketime(hour, minute, second)

描述：根据所给的小时，分钟和秒数返回time类型的值。由于DWS中的time类型的取值范围为00:00:00到24:00:00, 故不支持hour大于24时和hour小于0时的场景。

返回值类型：time

示例：

```
select maketime(12, 15, 30.12);
  maketime
-------------
 12:15:30.12
(1 row)
```

microsecond(timestamp with time zone)

描述：获取时间中的微秒值。

返回值类型：integer

示例：

```
SELECT microsecond(timestamptz '2018-12-13 12:11:15.123634+06');
 microsecond
-------------
      123634
(1 row)
```

minute(timestamp with time zone)

描述：获取时间中的分钟值。

返回值类型：integer

示例：

```
SELECT minute(timestamptz '2018-12-13 12:11:15+06');
 minute
--------
     11
(1 row)
```


版权所有 © 华为云计算技术有限公司

month(date)

描述：返回给定日期时间的月份。

返回值类型：integer

示例：

```
select month('2020-11-30');
 month
-------
    11
(1 row)
```

now([fsp])

描述：当前事务开始的日期及时间，参数确定微秒输出精度，缺省时为6。

返回值类型：timestamp with time zone

示例：

```
SELECT now();
              now
-------------------------------
 2017-09-01 17:03:42.549426+08
(1 row)
SELECT now(3);
             now
-----------------------------
 2021-09-08 10:59:00.427+08
(1 row)
```

numtodsinterval(num, interval_unit)

描述：将数字转换为interval类型。num为numeric类型数字，interval_unit为固定格式字符串（ 'DAY' | 'HOUR' | 'MINUTE' | 'SECOND' ）。

可以通过设置参数IntervalStyle为oracle，兼容该函数在Oracle中的interval输出格式。

示例：

```
SELECT numtodsinterval(100, 'HOUR');
 numtodsinterval
-----------------
 100:00:00
(1 row)

SET intervalstyle = oracle;
SET
SELECT numtodsinterval(100, 'HOUR');
        numtodsinterval
-------------------------------
 +000000004 04:00:00.000000000
(1 row)
```

pg_sleep(seconds)

描述：是服务器线程延迟时间，以秒计。

返回值类型：void

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT pg_sleep(10);
 pg_sleep
----------

(1 row)
```

period_add(P, N)

描述：返回给定时期加上N个月后的日期。

返回值类型：integer

示例：

```
select period_add(200801, 2);
 period_add
------------
     200803
(1 row)
```

period_diff(P1, P2)

描述：返回给定日期之间的月数差值。

返回值类型：integer

```
select period_diff(200802, 200703);
 period_diff
-------------
          11
(1 row)
```

quarter(date)

描述：获取日期date所属的季度。

返回值类型：integer

示例：

```
SELECT quarter(date '2018-12-13');
 quarter
---------
       4
(1 row)
```

second(timestamp with time zone)

描述：获取时间的秒数值。

返回值类型：integer

示例：

```
SELECT second(timestamptz '2018-12-13 12:11:15+06');
 second
--------
     15
(1 row)
```

statement_timestamp()

描述：当前日期及时间。

 版权所有 © 华为云计算技术有限公司

返回值类型：timestamp with time zone

示例：

```
SELECT statement_timestamp();
      statement_timestamp
-------------------------------
 2017-09-01 17:04:39.119267+08
(1 row)
```

sysdate

描述：当前日期及时间。

返回值类型：timestamp

示例：

```
SELECT sysdate;
       sysdate
---------------------
 2017-09-01 17:04:49
(1 row)
```

timeofday()

描述：当前日期及时间（像clock_timestamp，但是返回时为text。）

返回值类型：text

示例：

```
SELECT timeofday();
              timeofday
--------------------------------------
 Fri Sep 01 17:05:01.167506 2017 CST
(1 row)
```

transaction_timestamp()

描述：当前日期及时间，与current_timestamp等效。

返回值类型：timestamp with time zone

示例：

```
SELECT transaction_timestamp();
     transaction_timestamp
-------------------------------
 2017-09-01 17:05:13.534454+08
(1 row)
```

from_unixtime(unix_timestamp[,format])

描述：格式串缺省时，将unix时间戳转换为日期时间类型输出。格式串指定时，将unix时间戳转换为指定格式的字符串输出。

返回值类型：timestamp（格式串缺省）/ text（格式串指定）

示例：

```
SELECT from_unixtime(875996580);
    from_unixtime
---------------------
```

 版权所有 © 华为云计算技术有限公司

```
 1997-10-04 20:23:00
(1 row)
SELECT from_unixtime(875996580, '%Y %D %M %h:%i:%s');
       from_unixtime
---------------------------
 1997 5th October 04:23:00
(1 row)
```

unix_timestamp([timestamp with time zone])

描述：获取从'1970-01-01 00:00:00'UTC到入参时间经历的秒数。无入参时，指定为当前时间。

返回值类型：bigint（无入参）/numeric（有入参）

示例：

```
SELECT unix_timestamp();
 unix_timestamp
----------------
     1693906219
(1 row)
SELECT unix_timestamp('2018-09-08 12:11:13+06');
 unix_timestamp
----------------
     1536387073.000000
(1 row)
```

add_months(d,n)

描述：用于计算时间点d再加上n个月的时间。

返回值类型：timestamp

示例：

```
SELECT add_months(to_date('2017-5-29', 'yyyy-mm-dd'), 11) FROM dual;
     add_months
---------------------
 2018-04-29 00:00:00
(1 row)
```

last_day(d)

描述：用于计算时间点d本月最后一天的时间。

- ORA和TD兼容模式下，返回值类型为timestamp。
- MySQL兼容模式下，返回值类型为date。

示例：

```
select last_day(to_date('2017-01-01', 'YYYY-MM-DD')) AS cal_result;
     cal_result
---------------------
 2017-01-31 00:00:00
(1 row)
```

next_day(x,y)

描述：用于计算x时间开始的下一个星期y的时间。

- ORA和TD兼容模式下，返回值类型为timestamp。

 版权所有 © 华为云计算技术有限公司

- MySQL兼容模式下，返回值类型为date。

示例：

```
select next_day(timestamp '2017-05-25 00:00:00','Sunday')AS cal_result;
     cal_result
---------------------
 2017-05-28 00:00:00
(1 row)
```

from_days(days)

描述：根据给定的天数，返回相对应的日期值。

返回值类型：date

示例：

```
select from_days(730669);
 from_days
------------
 2000-07-03
(1 row)
```

to_days(timestamp)

描述：返回自0年开始到入参日期的天数。

返回值类型：integer

示例：

```
SELECT to_days(timestamp '2008-10-07');
 to_days
---------
  733687
(1 row)
```

21.7.5.3 EXTRACT

EXTRACT(*field* **FROM** *source***)**

extract函数从日期或时间的数值里抽取子域，比如年、小时等。source必须是一个timestamp、time或interval类型的值表达式（类型为date的表达式转换为timestamp，因此也可以用）。field是一个标识符或者字符串，它指定从源数据中抽取的域。extract函数返回类型为double precision的数值。field的取值范围如下所示。

century

世纪。

第一个世纪从0001-01-01 00:00:00 AD开始。这个定义适用于所有使用阳历的国家。没有0世纪，直接从公元前1世纪到公元1世纪。

示例：

```
SELECT EXTRACT(CENTURY FROM TIMESTAMP '2000-12-16 12:21:13');
 date_part
-----------
        20
(1 row)
```

 版权所有 © 华为云计算技术有限公司

day

- 如果source为timestamp，表示月份里的日期（1-31）。

```
SELECT EXTRACT(DAY FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
        16
(1 row)
```

- 如果source为interval，表示天数。

```
SELECT EXTRACT(DAY FROM INTERVAL '40 days 1 minute');
 date_part
-----------
        40
(1 row)
```

decade

年份除以10。

```
SELECT EXTRACT(DECADE FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
       200
(1 row)
```

dow

每周的星期几，星期天（0）到星期六（6）。

```
SELECT EXTRACT(DOW FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
         5
(1 row)
```

doy

一年的第几天（1~365/366）。

```
SELECT EXTRACT(DOY FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
        47
(1 row)
```

epoch

- 如果source为timestamp with time zone，表示自1970-01-01 00:00:00-00 UTC以来的秒数（结果可能是负数）；

  如果source为date和timestamp，表示自1970-01-01 00:00:00-00当地时间以来的秒数；

  如果source为interval，表示时间间隔的总秒数。

```
SELECT EXTRACT(EPOCH FROM TIMESTAMP WITH TIME ZONE '2001-02-16 20:38:40.12-08');
  date_part
--------------
 982384720.12
(1 row)
SELECT EXTRACT(EPOCH FROM INTERVAL '5 days 3 hours');
 date_part
-----------
    442800
(1 row)
```


版权所有 © 华为云计算技术有限公司

- 将epoch值转换为时间戳的方法。

```
SELECT TIMESTAMP WITH TIME ZONE 'epoch' + 982384720.12 * INTERVAL '1 second' AS RESULT;
            result
----------------------------
 2001-02-17 12:38:40.12+08
(1 row)
```

hour

小时域（0-23）。

```
SELECT EXTRACT(HOUR FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
        20
(1 row)
```

isodow

一周的第几天（1-7）。

星期一为1，星期天为7。

说明

除了星期天外，都与dow相同。

```
SELECT EXTRACT(ISODOW FROM TIMESTAMP '2001-02-18 20:38:40');
 date_part
-----------
         7
(1 row)
```

isoyear

日期中的ISO 8601标准年（不适用于间隔）。

每个带有星期一开始的周中包含1月4日的ISO年，所以在年初的1月或12月下旬的ISO年可能会不同于阳历的年。详细信息请参见**week**描述。

```
SELECT EXTRACT(ISOYEAR FROM DATE '2006-01-01');
 date_part
-----------
      2005
(1 row)
SELECT EXTRACT(ISOYEAR FROM DATE '2006-01-02');
 date_part
-----------
      2006
(1 row)
```

microseconds

秒域（包括小数部分）乘以1,000,000。

```
SELECT EXTRACT(MICROSECONDS FROM TIME '17:12:28.5');
 date_part
-----------
  28500000
(1 row)
```

millennium

千年。

 版权所有 © 华为云计算技术有限公司

20世纪（19xx年）里面的年份在第二个千年里。第三个千年从2001年1月1日零时开始。

```
SELECT EXTRACT(MILLENNIUM FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
         3
(1 row)
```

milliseconds

秒域（包括小数部分）乘以1000。请注意它包括完整的秒。

```
SELECT EXTRACT(MILLISECONDS FROM TIME '17:12:28.5');
 date_part
-----------
     28500
(1 row)
```

minute

分钟域（0-59）。

```
SELECT EXTRACT(MINUTE FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
        38
(1 row)
```

month

如果source为timestamp，表示一年里的月份数（1-12）。

```
SELECT EXTRACT(MONTH FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
         2
(1 row)
```

如果source为interval，表示月的数目，然后对12取模（0-11）。

```
SELECT EXTRACT(MONTH FROM INTERVAL '2 years 13 months');
 date_part
-----------
         1
(1 row)
```

quarter

该天所在的该年的季度（1-4）。

```
SELECT EXTRACT(QUARTER FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
         1
(1 row)
```

second

秒域，包括小数部分（0-59）。

```
SELECT EXTRACT(SECOND FROM TIME '17:12:28.5');
 date_part
-----------
```

 版权所有 © 华为云计算技术有限公司

```
    28.5
(1 row)
```

timezone

与UTC的时区偏移量，单位为秒。正数对应UTC东边的时区，负数对应UTC西边的时区。

```
SELECT EXTRACT(timezone FROM TIMETZ '17:12:28');
 date_part
-----------
  28800
(1 row)
```

timezone_hour

时区偏移量的小时部分。

```
SELECT EXTRACT(timezone_hour FROM TIMETZ '17:12:28');
 date_part
-----------
      8
(1 row)
```

timezone_minute

时区偏移量的分钟部分。

```
SELECT EXTRACT(timezone_minute FROM TIMETZ '17:12:28');
 date_part
-----------
       0
(1 row)
```

week

该天在所在的年份里是第几周。ISO 8601定义一年的第一周包含该年的一月四日（ISO-8601 的周从星期一开始）。换句话说，一年的第一个星期四在第一周。

在ISO定义里，一月的头几天可能是前一年的第52或者第53周，十二月的后几天可能是下一年第一周。比如，2005-01-01是2004年的第53周，而2006-01-01是2005年的第52周，2012-12-31是2013年的第一周。建议isoyear字段和week一起使用，以得到一致的结果。

```
SELECT EXTRACT(WEEK FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
       7
(1 row)
```

year

年份域。

```
SELECT EXTRACT(YEAR FROM TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
    2001
(1 row)
```

 版权所有 © 华为云计算技术有限公司

21.7.5.4 date_part

date_part

date_part函数是在传统的Ingres函数的基础上制作的（该函数等效于SQL标准函数extract）：

date_part('*field*', *source*)

这里的field参数必须是一个字符串，而不是一个名字。有效的field与extract一样，详细信息请参见**EXTRACT**。

示例：

```
SELECT date_part('day', TIMESTAMP '2001-02-16 20:38:40');
 date_part
-----------
        16
(1 row)
SELECT date_part('hour', INTERVAL '4 hours 3 minutes');
 date_part
-----------
         4
(1 row)
```

21.7.5.5 date_format

date_format(timestamp, fmt)

date_format函数将日期参数按照fmt指定的格式转换为字符串。

示例：

```
SELECT date_format('2009-10-04 22:23:00', '%M %D %W');
    date_format
--------------------
 October 4th Sunday
(1 row)
SELECT date_format('2021-02-20 08:30:45', '%Y-%m-%d %H:%i:%S');
     date_format
---------------------
 2021-02-20 08:30:45
(1 row)
SELECT date_format('2021-02-20 18:10:15', '%r-%T');
     date_format
----------------------
 06:10:15 PM-18:10:15
(1 row)
```

表21-33显示了可以用于将日期参数格式化输出的格式类型，这些格式类型适用于函数date_format、time_format、str_to_date、str_to_time和from_unixtime。

表 21-33 date_format 支持的输出格式

格式	说明	取值
%a	缩写星期名	Sun...Sat
%b	缩写月份名	Jan...Dec
%c	月份	0...12

版权所有 © 华为云计算技术有限公司

格式	说明	取值
%D	带英文后缀的月份日期	0th, 1st, 2nd, 3rd, …
%d	一个月里的日，2位	00...31
%e	一个月里的日	0...31
%f	微秒	000000...999999
%H	小时，24小时制	00...23
%h	小时，12小时制	01...12
%I	小时，12小时制，同%h	01...12
%i	分钟	00...59
%j	一年里的日	001...366
%k	小时，24小时制，同%H	0...23
%l	小时，12小时制，同%h	1...12
%M	月份名	January...December
%m	月份，两位	00...12
%p	上下午	AM PM
%r	时间，12小时制	hh::mm::ss AM/PM
%S	秒	00...59
%s	秒，同%S	00...59
%T	时间，24小时制	hh::mm::ss
%U	周 (00-53) 星期日是一周的第一天	00...53
%u	周 (00-53) 星期一是一周的第一天	00...53
%V	周 (01-53) 星期日是一周的第一天，与%X搭配使用	01...53
%v	周 (01-53) 星期一是一周的第一天，与%x搭配使用	01...53
%W	星期名	Sunday...Saturday
%w	一周的日，周日为0	0...6
%X	年份，其中的星期日是周的第一天，4 位，与%V搭配使用	-
%x	年份，其中的星期一是周的第一天，4 位，与%v搭配使用	-

版权所有 © 华为云计算技术有限公司

格式	说明	取值
%Y	年份，4位	-
%y	年份，2位	-
%%	字符'%'	字符'%'
%*x*	'*x*'，上述未列出的任意字符	字符'*x*'

须知

date_format支持的输出格式中，%U、%u、%V、%v、%X、%x暂不支持。

21.7.5.6 time_format

time_format(time, fmt)

描述：time_format函数将日期参数按照fmt指定的格式转换为字符串。与date_format函数类似，但格式字符串只能包含小时、分钟、秒和微秒的格式说明符，如果包含其他说明符则会返回NULL值或0。

返回值类型：text

示例：

```
SELECT time_format('2009-10-04 22:23:00', '%M %D %W');
   time_format
--------------------

(1 row)
SELECT time_format('2021-02-20 08:30:45', '%Y-%m-%d %H:%i:%S');
    time_format
----------------------
 0000-00-00 08:30:45
(1 row)
SELECT time_format('2021-02-20 18:10:15', '%r-%T');
    time_format
-----------------------
 06:10:15 PM-18:10:15
(1 row)
```

须知

time_format仅支持时间相关的格式输出（%f、%H、%h、%I、%i、%k、%l、%p、%r、%S、%s、%T），不支持日期相关格式，其他情况处理为普通字符。

str_to_date(str, format)

描述：将日期/时间格式的字符串（str），按照所提供的显示格式（format）转换为日期类型的值。

返回值类型：timestamp

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT str_to_date('01,5,2021','%d,%m,%Y');
    str_to_date
---------------------
 2021-05-01 00:00:00
(1 row)
SELECT str_to_date('01,5,2021,09,30,17','%d,%m,%Y,%h,%i,%s');
    str_to_date
---------------------
 2021-05-01 09:30:17
(1 row)
```

适用于str_to_date的格式化输入的格式类型参考**表21-33**。这里仅支持“日期”格式、“日期+时间”格式的输入转换，对于仅“时间”格式的输入场景请使用str_to_time。

str_to_time(str, format)

描述：将时间格式的字符串（str），按照所提供的显示格式（format）转换为时间类型的值。

返回值类型：time

示例：

```
SELECT str_to_time('09:30:17','%h:%i:%s');
 str_to_time
-------------
 09:30:17
(1 row)
```

适用于str_to_time的格式化输入的格式类型参考**表21-33**，这里仅支持“时间”格式的输入转换，对于“日期”格式、“日期+时间”格式的输入场景请使用str_to_date。

week(date[, mode])

描述：根据模式返回指定日期时间所处年份中对应的周数，默认模式为0。

返回值类型：integer

表 21-34 week 函数中 mode 模式的工作原理

模式	一周的第一天	周数范围	第一周的判断规则
0	星期日	0-53	元旦后的第一个星期日所在周
1	星期一	0-53	元旦后有四天或者更多天所在周
2	星期日	1-53	元旦后的第一个星期日所在周
3	星期一	1-53	元旦后有四天或者更多天所在周
4	星期日	0-53	元旦后有四天或者更多天所在周
5	星期一	0-53	元旦后的第一个星期一所在周
6	星期日	1-53	元旦后有四天或者更多天所在周
7	星期一	1-53	元旦后的第一个星期一所在周

版权所有 © 华为云计算技术有限公司

示例：

```
select week('2018-01-01');
 week
------
    0
(1 row)

select week('2018-01-01', 0);
 week
------
    0
(1 row)

select week('2020-12-31', 1);
 week
------
   53
(1 row)

select week('2020-12-31', 5);
 week
------
   52
(1 row)
```

weekday(date)

描述：返回给定日期date对应的星期索引，星期一作为一周的开始日。

取值范围：0~6

返回值类型：integer

示例：

```
select weekday('2020-11-06');
 weekday
---------
       4
(1 row)
```

weekofyear(date)

描述：返回给定日期date所在周在本年中对应的周数，取值范围为[1, 53]，等价于week(date, 3)。

返回值类型：integer

示例：

```
select weekofyear('2020-12-30');
 weekofyear
------------
         53
(1 row)
```

year(date)

描述：获取时间日期date所处的年份

返回值类型：integer

示例：

 版权所有 © 华为云计算技术有限公司

```
select year('2020-11-13');
 year
------
 2020
(1 row)
```

yearweek(date[, mode])

描述：返回给定日期date在本年中对应的年份和周数，周数范围为[1, 53]。

返回值类型：integer

示例：

```
select yearweek('2019-12-31');
 yearweek
----------
   201952
(1 row)

select yearweek('2019-1-1');
 yearweek
----------
   201852
(1 row)
```

21.7.6 SEQUENCE 函数

序列函数为用户从序列对象中获取后续的序列值提供了简单的多用户安全的方法。

nextval(regclass)

递增序列并返回新值。

说明

- 为了避免从同一个序列获取值的并发事务被阻塞， nextval操作不会回滚；也就是说，一旦获取了一个值，它就会被认为已被使用，并且不会再次返回。 即使该操作处于事务中，当事务随后中断，或者调用查询最终没有使用该值，也会出现这种情况。这种情况将在指定值的顺序中留下未使用的“空洞”。 因此，DWS序列对象不能用于获取“无间隙”序列。
- 如果nextval被下推到DN上时，各个DN会自动连接GTM，请求next values值，例如（INSERT INTO t1 SELECT xxx，t1某一列需要调用nextval函数），由于GTM上有最大连接数为8192的限制，而这类下推语句会导致消耗过多的GTM连接数，因此对于这类语句的并发数目限制为7000（其它语句需要占用部分连接）/集群DN数目。

返回类型：bigint

nextval函数有两种调用方式（其中第二种调用方式兼容Oracle的语法，目前不支持Sequence命名中有特殊字符"."的情况），如下：

示例1：

```
SELECT nextval('seqDemo');
 nextval
---------
       2
(1 row)
```

示例2：

```
SELECT seqDemo.nextval;
 nextval
---------
```

 版权所有 © 华为云计算技术有限公司

```
     2
(1 row)
```

currval(regclass)

返回当前会话里最近一次nextval返回的指定的sequence的数值。如果当前会话还没有调用过指定的sequence的nextval，那么调用currval将会报错。需要注意的是，这个函数在默认情况下是不支持的，需要通过设置enable_beta_features为true之后，才能使用这个函数。同时在设置enable_beta_features为true之后，nextval()函数将不支持下推。

返回类型：bigint

currval函数有两种调用方式（其中第二种调用方式兼容Oracle的语法，目前不支持Sequence命名中有特殊字符"."的情况），如下：

示例1：

```
SELECT currval('seq1');
 currval
---------
     2
(1 row)
```

示例2：

```
SELECT seq1.currval seq1;
 currval
---------
     2
(1 row)
```

lastval()

描述：返回当前会话里最近一次nextval返回的数值。这个函数等效于currval，只是它不用序列名为参数，它抓取当前会话里面最近一次nextval使用的序列。如果当前会话还没有调用过nextval，那么调用lastval将会报错。

需要注意的是，这个函数在默认情况下是不支持的，需要通过设置enable_beta_features或者lastval_supported为true之后，才能使用这个函数。同时这种情况下，nextval()函数将不支持下推。

返回类型：bigint

示例：

```
SELECT lastval();
 lastval
---------
     2
(1 row)
```

setval(regclass, bigint)

描述：设置序列的当前数值。

返回类型：bigint

示例：

```
SELECT setval('seqDemo',1);
 setval
--------
```

 版权所有 © 华为云计算技术有限公司

```
    1
(1 row)
```

setval(regclass, bigint, boolean)

描述：设置序列的当前数值以及is_called标志。

返回类型：bigint

示例：

```
SELECT setval('seqDemo',1,true);
 setval
--------
    1
(1 row)
```

说明

Setval后当前会话及GTM上会立刻生效，但如果其他会话有缓存的序列值，只能等到缓存值用尽才能感知Setval的作用。所以为了避免序列值冲突，setval要谨慎使用。

因为序列是非事务的，setval造成的改变不会由于事务的回滚而撤销。

21.7.7 数组函数和操作符

21.7.7.1 数组操作符

数组比较是使用默认的B-Tree比较函数对所有元素逐一进行比较的。多维数组的元素按照行顺序进行访问。如果两个数组的内容相同但维数不等，决定排序顺序的首要因素是维数。

=

描述：两个数组是否相等

示例：

```
SELECT ARRAY[1.1,2.1,3.1]::int[] = ARRAY[1,2,3] AS RESULT;
 result
--------
 t
(1 row)
```

<>

描述：两个数组是否不相等

示例：

```
SELECT ARRAY[1,2,3] <> ARRAY[1,2,4] AS RESULT;
 result
--------
 t
(1 row)
```

<

描述：一个数组是否小于另一个数组

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT ARRAY[1,2,3] < ARRAY[1,2,4] AS RESULT;
 result
--------
 t
(1 row)
```

>

描述：一个数组是否大于另一个数组

示例：

```
SELECT ARRAY[1,4,3] > ARRAY[1,2,4] AS RESULT;
 result
--------
 t
(1 row)
```

<=

描述：一个数组是否小于或等于另一个数组

示例：

```
SELECT ARRAY[1,2,3] <= ARRAY[1,2,3] AS RESULT;
 result
--------
 t
(1 row)
```

>=

描述：一个数组是否大于或等于另一个数组

示例：

```
SELECT ARRAY[1,4,3] >= ARRAY[1,4,3] AS RESULT;
 result
--------
 t
(1 row)
```

@>

描述：一个数组是否包含另一个数组

示例：

```
SELECT ARRAY[1,4,3] @> ARRAY[3,1] AS RESULT;
 result
--------
 t
(1 row)
```

<@

描述：一个数组是否被包含于另一个数组

示例：

```
SELECT ARRAY[2,7] <@ ARRAY[1,7,4,2,6] AS RESULT;
 result
--------
```

 版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
```

&&

描述：一个数组是否和另一个数组重叠（有共同元素）

示例：

```
SELECT ARRAY[1,4,3] && ARRAY[2,1] AS RESULT;
 result
--------
 t
(1 row)
```

||

描述：数组与数组进行连接

示例：

```
SELECT ARRAY[1,2,3] || ARRAY[4,5,6] AS RESULT;
    result
---------------
 {1,2,3,4,5,6}
(1 row)
SELECT ARRAY[1,2,3] || ARRAY[[4,5,6],[7,8,9]] AS RESULT;
          result
---------------------------
 {{1,2,3},{4,5,6},{7,8,9}}
(1 row)
```

||

描述：元素与数组进行连接

示例：

```
SELECT 3 || ARRAY[4,5,6] AS RESULT;
  result
-----------
 {3,4,5,6}
(1 row)
```

||

描述：数组与元素进行连接

示例：

```
SELECT ARRAY[4,5,6] || 7 AS RESULT;
  result
-----------
 {4,5,6,7}
(1 row)
```

21.7.7.2 数组函数

array_append(anyarray, anyelement)

描述：向数组末尾添加元素，只支持一维数组。

返回类型：anyarray

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT array_append(ARRAY[1,2], 3) AS RESULT;
 result
---------
 {1,2,3}
(1 row)
```

array_prepend(anyelement, anyarray)

描述：向数组开头添加元素，只支持一维数组。

返回类型：anyarray

示例：

```
SELECT array_prepend(1, ARRAY[2,3]) AS RESULT;
 result
---------
 {1,2,3}
(1 row)
```

array_cat(anyarray, anyarray)

描述：连接两个数组，支持多维数组。

返回类型：anyarray

示例：

```
SELECT array_cat(ARRAY[1,2,3], ARRAY[4,5]) AS RESULT;
   result
-------------
 {1,2,3,4,5}
(1 row)

SELECT array_cat(ARRAY[[1,2],[4,5]], ARRAY[6,7]) AS RESULT;
       result
---------------------
 {{1,2},{4,5},{6,7}}
(1 row)
```

array_ndims(anyarray)

描述：返回数组的维数。

返回类型：int

示例：

```
SELECT array_ndims(ARRAY[[1,2,3], [4,5,6]]) AS RESULT;
 result
--------
      2
(1 row)
```

array_dims(anyarray)

描述：返回数组维数的文本表示。

返回类型：text

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT array_dims(ARRAY[[1,2,3], [4,5,6]]) AS RESULT;
  result
------------
 [1:2][1:3]
(1 row)
```

array_length(anyarray, int)

描述：返回数组维度的长度。

返回类型：int

示例：

```
SELECT array_length(array[1,2,3], 1) AS RESULT;
 result
--------
      3
(1 row)
```

array_lower(anyarray, int)

描述：返回数组维数的下界。

返回类型：int

示例：

```
SELECT array_lower('[0:2]={1,2,3}'::int[], 1) AS RESULT;
 result
--------
      0
(1 row)
```

array_upper(anyarray, int)

描述：返回数组维数的上界。

返回类型：int

示例：

```
SELECT array_upper(ARRAY[1,8,3,7], 1) AS RESULT;
 result
--------
      4
(1 row)
```

array_to_string(anyarray, text [, text])

描述：使用第一个text作为数组的新分隔符，使用第二个text替换数组值为null的值。

返回类型：text

示例：

```
SELECT array_to_string(ARRAY[1, 2, 3, NULL, 5], ',', '*') AS RESULT;
  result
-----------
 1,2,3,*,5
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

在string_to_array中，如果省略null字符串参数或为NULL，将字符串中没有输入内容的子串替换为NULL。

在array_to_string中，如果省略null字符串参数或为NULL，运算中将跳过在数组中的任何null元素，并且不会在输出字符串中出现。

string_to_array(text, text [, text])

描述：使用第二个text指定分隔符，使用第三个可选的text作为NULL值替换模板，如果分隔后的子串与第三个可选的text完全匹配，则将其替换为NULL。

返回类型：text[]

示例：

```
SELECT string_to_array('xx~^~yy~^~zz', '~^~', 'yy') AS RESULT;
   result
--------------
 {xx,NULL,zz}
(1 row)
SELECT string_to_array('xx~^~yy~^~zz', '~^~', 'y') AS RESULT;
  result
------------
 {xx,yy,zz}
(1 row)
```

说明

在string_to_array中，如果分隔符参数是NULL，输入字符串中的每个字符将在结果数组中变成一个独立的元素。如果分隔符是一个空白字符串，则整个输入的字符串将变为一个元素的数组。否则输入字符串将在每个分隔字符串处分开。

unnest(anyarray)

描述：扩大一个数组为一组行。

返回类型：setof anyelement

示例：

```
SELECT unnest(ARRAY[1,2]) AS RESULT;
 result
--------
      1
      2
(2 rows)
```

unnest函数配合string_to_array数组使用。数组转列，先将字符串按逗号分割成数组，然后再把数组转成列：

```
SELECT unnest(string_to_array('a,b,c,d',',')) AS RESULT;
 result
--------
 a
 b
 c
 d
(4 rows)
```


版权所有 © 华为云计算技术有限公司

interval(N, N1, N2, N3 ...)

描述：从输入的整数数组中，查找返回最后一个小于等于目标参数n的数组索引。若n为NULL，返回-1。interval函数不支持interval(N, N1)的场景。该函数仅8.2.0及以上集群版本支持。

返回类型：int

示例：

```
SELECT INTERVAL(10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 11) AS RESULT;
 result
--------
     11
(1 row)
```

split(string, delim)

描述：将字符串string按分隔符delimiter进行分隔，并返回数组。该函数仅8.2.0及以上集群版本支持。

返回类型：text[]

示例：

```
SELECT SPLIT('a-b-c-d-e', '-') AS RESULT;
   result
-------------
 {a,b,c,d,e}
(1 row)
SELECT SPLIT('a-b-c-d-e', '-')[4] AS RESULT;
 result
--------
 d
(1 row)
```

21.7.8 逻辑操作符

常用的逻辑操作符有AND、OR和NOT，其运算结果有三个值，分别为TRUE、FALSE和NULL，其中NULL代表未知。运算优先级顺序为：NOT>AND>OR。

运算规则请参见表21-35，表中的a和b代表逻辑表达式。

表 21-35 运算规则表

a	b	a AND b的结果	a OR b的结果	NOT a的结果
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	NULL	NULL	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	NULL	FALSE	NULL	TRUE
NULL	NULL	NULL	NULL	NULL

版权所有 © 华为云计算技术有限公司

说明

操作符AND和OR具有交换性，即交换左右两个操作数，不影响其结果。

21.7.9 比较操作符

比较操作符可用于所有相关的数据类型，并返回布尔类型数值。

所有比较操作符都是双目操作符，被比较的两个数据类型必须是相同的数据类型或者是可以进行隐式转换的类型。例如1<2<3这样的表达式为非法的，因为布尔值和3之间不能做比较。

DWS提供的比较操作符请参见**表21-36**。

表 21-36 比较操作符

操作符	描述
<	小于
>	大于
<=	小于或等于
>=	大于或等于
=	等于
<> 或 !=	不等于

21.7.10 模式匹配操作符

数据库提供了三种实现模式匹配的方法：SQL LIKE操作符、SIMILAR TO操作符和POSIX-风格的正则表达式。除了这些基本的操作符外，还有一些函数可用于提取或替换匹配子串并在匹配位置分离一个串。

LIKE

判断字符串是否能匹配上LIKE后的模式字符串。如果字符串与提供的模式匹配，则LIKE表达式返回为真（NOT LIKE表达式返回假），否则返回为假（NOT LIKE表达式返回真）。

- 匹配规则
  a. 此操作符只有在它的模式匹配整个串的时候才能成功。如果要匹配在串内任何位置的序列，该模式必须以百分号开头和结尾。
  b. 下划线（_）代表（匹配）任何单个字符； 百分号（%）代表任意串的通配符。
  c. 要匹配文本里的下划线（_）或者百分号（%），在提供的模式里相应字符必须前导逃逸字符。逃逸字符的作用是禁用元字符的特殊含义，缺省的逃逸字符是反斜线，也可以用ESCAPE子句指定一个不同的逃逸字符。
  d. 要匹配逃逸字符本身，需写两个逃逸字符。例如要写一个包含反斜线的模式常量，那就要在SQL语句里写两个反斜线。

版权所有 © 华为云计算技术有限公司

说明

参数standard_conforming_strings设置为off时，在文串常量中写的任何反斜线都需要被双写。因此写一个匹配单个反斜线的模式实际上要在语句里写四个反斜线。可通过用ESCAPE选择一个不同的逃逸字符来避免这种情况，这样反斜线就不再是LIKE的特殊字符了。但仍然是字符文本分析器的特殊字符，所以还是需要两个反斜线。也可通过写ESCAPE ''的方式不选择逃逸字符，这样可以有效地禁用逃逸机制，但是没有办法关闭下划线和百分号在模式中的特殊含义。

e. 关键字ILIKE可以用于替换LIKE，区别是LIKE大小写敏感，ILIKE大小写不敏感。

f. 操作符~~等效于LIKE，操作符~~*等效于ILIKE。

- 示例

```
SELECT 'abc' LIKE 'abc' AS RESULT;
 result
-----------
 t
(1 row)
SELECT 'abc' LIKE 'a%' AS RESULT;
 result
-----------
 t
(1 row)
SELECT 'abc' LIKE '_b_' AS RESULT;
 result
-----------
 t
(1 row)
SELECT 'abc' LIKE 'c' AS RESULT;
 result
-----------
 f
(1 row)
```

SIMILAR TO

SIMILAR TO操作符根据自己的模式判断是否匹配给定串而返回真或者假。它和LIKE非常类似，只不过它使用SQL标准定义的正则表达式理解模式。

- 匹配规则

a. 和LIKE一样，SIMILAR TO操作符只有在它的模式匹配整个串的时候才返回真。如果要匹配在串内任何位置的序列，该模式必须以百分号开头和结尾。

b. 下划线（_）代表（匹配）任何单个字符； 百分号（%）代表任意串的通配符。

c. SIMILAR TO也支持下面这些从POSIX正则表达式借用的模式匹配元字符。

表 21-37 模式匹配元字符

元字符	含义
|	表示选择（两个候选之一）。
*	表示重复前面的项零次或更多次。
+	表示重复前面的项一次或更多次。
?	表示重复前面的项零次或一次。
{m}	表示重复前面的项刚好m次。

 版权所有 © 华为云计算技术有限公司

元字符	含义
{m,}	表示重复前面的项m次或更多次。
{m,n}	表示重复前面的项至少m次并且不超过n次。
()	把多个项组合成一个逻辑项。
[...]	声明一个字符类，就像POSIX正则表达式一样。

d. 前导逃逸字符可以禁止所有这些元字符的特殊含义。逃逸字符的使用规则和LIKE一样。

- 注意事项

  如果SIMILAR TO正则表达式重复匹配字符数量非常庞大，由于受递归大小限制，执行语句会失败并报错invalid regular expression: regular expression is too complex，可尝试调大GUC参数max_stack_depth。

- 正则表达式函数

  支持使用函数**substring(string from pattern for escape)**截取匹配SQL正则表达式的子字符串。

- 示例

```
SELECT 'abc' SIMILAR TO 'abc' AS RESULT;
 result
------------
 t
(1 row)
SELECT 'abc' SIMILAR TO 'a' AS RESULT;
 result
------------
 f
(1 row)
SELECT 'abc' SIMILAR TO '%(b|d)%' AS RESULT;
 result
------------
 t
(1 row)
SELECT 'abc' SIMILAR TO '(b|c)%'  AS RESULT;
 result
------------
 f
(1 row)
```

POSIX 正则表达式

正则表达式是一个字符序列，它是定义一个串集合（一个正则集）的缩写。 如果一个串是正则表达式描述的正则集中的一员时， 那么就说这个串匹配该正则表达式。POSIX正则表达式提供了比LIKE和SIMILAR TO操作符更强大的含义。**表21-38**列出了所有可用于POSIX正则表达式模式匹配的操作符。

表 21-38 正则表达式匹配操作符

操作符	描述	例子
~	匹配正则表达式，大小写敏感	'thomas' ~ '.*thomas.*'

 版权所有 © 华为云计算技术有限公司

操作符	描述	例子
~*	匹配正则表达式，大小写不敏感	'thomas' ~* '.*Thomas.*'
!~	不匹配正则表达式，大小写敏感	'thomas' !~ '.*Thomas.*'
!~*	不匹配正则表达式，大小写不敏感	'thomas' !~* '.*vadim.*'

- 匹配规则
  a. 与LIKE不同，正则表达式允许匹配串里的任何位置，除非该正则表达式显式地挂接在串的开头或者结尾。
  b. 除了上文提到的元字符外， POSIX正则表达式还支持下表的模式匹配元字符。

  表 21-39 模式匹配元字符

元字符	含义
^	表示串开头的匹配。
$	表示串末尾的匹配。
.	匹配任意单个字符。

- 正则表达式函数
  POSIX正则表达式支持下面函数。
  - **substring(string from pattern)**函数提供了抽取一个匹配POSIX正则表达式模式的子串的方法。
  - **regexp_replace(string, pattern, replacement [,flags])**函数提供了将匹配POSIX正则表达式模式的子串替换为新文本的功能。
  - **regexp_matches(string text, pattern text [, flags text])**函数返回一个文本数组，该数组由匹配一个POSIX正则表达式模式得到的所有被捕获子串构成。
  - **regexp_split_to_table(string text, pattern text [, flags text])**函数把一个POSIX正则表达式模式当作一个定界符来分离一个串。
  - **regexp_split_to_array(string text, pattern text [, flags text])**和regexp_split_to_table类似，是一个正则表达式分离函数，不过它的结果以一个text数组的形式返回。

    说明

    正则表达式分离函数会忽略零长度的匹配，这种匹配发生在串的开头或结尾或者正好发生在前一个匹配之后。这和正则表达式匹配的严格定义是相悖的，后者由regexp_matches实现，但是通常前者是实际中最常用的行为。

- 示例

```
 SELECT 'abc' ~ 'Abc' AS RESULT;
result
```

 版权所有 © 华为云计算技术有限公司

```
--------
 f
(1 row)
SELECT 'abc' ~* 'Abc' AS RESULT;
 result
--------
 t
(1 row)
SELECT 'abc' !~ 'Abc' AS RESULT;
 result
--------
 t
(1 row)
SELECT 'abc'!~* 'Abc' AS RESULT;
 result
--------
 f
(1 row)
SELECT 'abc' ~ '^a' AS RESULT;
 result
--------
 t
(1 row)
SELECT 'abc' ~ '(b|d)'AS RESULT;
 result
--------
 t
(1 row)
SELECT 'abc' ~ '^(b|c)'AS RESULT;
 result
--------
 f
(1 row)
```

虽然大部分的正则表达式搜索都能很快地执行，但是正则表达式仍可能被人为地控制，通过任意长的时间和任意量的内存进行处理。不建议接受非安全来源的正则表达式搜索模式，如果必须这样做，建议加上语句超时限制。使用SIMILAR TO模式的搜索具有同样的安全性危险， 因为SIMILAR TO提供了很多和POSIX-风格正则表达式相同的能力。LIKE搜索比其他两种选项简单得多，因此在接受非安全来源的搜索模式时要更安全些。

21.7.11 聚集函数

sum(expression)

描述：所有输入行的expression总和。

返回类型：

通常情况下输入数据类型和输出数据类型是相同的，但以下情况会发生类型转换：

- 对于SMALLINT或INT输入，输出类型为BIGINT。
- 对于BIGINT输入，输出类型为NUMBER 。
- 对于浮点数输入，输出类型为DOUBLE PRECISION。

示例：

```
SELECT SUM(ss_ext_tax) FROM tpcds.STORE_SALES;
  sum
--------------
 213267594.69
(1 row)
```


版权所有 © 华为云计算技术有限公司

max(expression)

描述：所有输入行中expression的最大值。

参数类型：任意数组、数值、字符串、日期/时间类型。

返回类型：与参数数据类型相同。

示例：

```
SELECT MAX(inv_quantity_on_hand) FROM tpcds.inventory;
   max
---------
 1000000
(1 row)
```

min(expression)

描述：所有输入行中expression的最小值。

参数类型：任意数组、数值、字符串、日期/时间类型。

返回类型：与参数数据类型相同。

示例：

```
SELECT MIN(inv_quantity_on_hand) FROM tpcds.inventory;
 min
-----
   0
(1 row)
```

avg(expression)

描述：所有输入值的均值（算术平均）。

返回类型：

- 对于任何整数类型输入，结果都是NUMBER类型。
- 对于任何浮点输入，结果都是DOUBLE PRECISION类型。
- 其他，和输入数据类型相同。

示例：

```
SELECT AVG(inv_quantity_on_hand) FROM tpcds.inventory;
        avg
----------------------
 500.0387129084044604
(1 row)
```

median(expression)

描述：所有输入值的中位数值。当前只支持数值类型和interval类型。其中空值不参与计算。

返回类型：

- 对于任何整型数据输入，结果都是NUMERIC类型。否则与输入数据类型相同。
- Teradata兼容模式下，如果输入为整型，则返回的数据精度只有整数位。

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT MEDIAN(inv_quantity_on_hand) FROM tpcds.inventory;
 median
--------
    500
(1 row)
```

percentile_cont(const) within group(order by expression)

描述：返回一个对应于目标列排序中指定分位数的值，如有必要就在相邻的输入项之间插入值。其中空值不参与计算。

输入：const为在0-1之间的数值，expression当前只支持数值类型和interval类型。

返回类型：

- 对于任何整型数据输入，结果都是NUMERIC类型。否则与输入数据类型相同。
- Teradata兼容模式下，如果输入为整型，则返回的数据精度只有整数位。

示例：

```
SELECT percentile_cont(0.3) within group(order by x) FROM (SELECT generate_series(1,5) AS x) AS t;
percentile_cont
-----------------
2.2
(1 row)
SELECT percentile_cont(0.3) within group(order by x desc) FROM (SELECT generate_series(1,5) AS x) AS t;
percentile_cont
-----------------
3.8
(1 row)
```

percentile_disc(const) within group(order by expression)

描述：返回第一个在排序中位置等于或者超过指定分数的输入值。

输入：const为在0-1之间的数值，expression当前只支持数值类型和interval类型。其中空值不参与计算。

返回类型：对于任何整型数据输入，结果都是NUMERIC类型。否则，与输入数据类型相同。

示例：

```
SELECT percentile_disc(0.3) within group(order by x) FROM (SELECT generate_series(1,5) AS x) AS t;
percentile_disc
-----------------
2
(1 row)
SELECT percentile_disc(0.3) within group(order by x desc) FROM (SELECT generate_series(1,5) AS x) AS t;
percentile_disc
-----------------
4
(1 row)
```

count(expression)

描述：返回表中满足expression不为NULL的行数。

返回类型：BIGINT

示例：

```
SELECT COUNT(inv_quantity_on_hand) FROM tpcds.inventory;
  count
```


版权所有 © 华为云计算技术有限公司

```
----------
 11158087
(1 row)
```

count(*)

描述：返回表中的记录行数。

返回类型：BIGINT

示例：

```
SELECT COUNT(*) FROM tpcds.inventory;
  count
----------
 11745000
(1 row)
```

array_agg(expression)

描述：将所有输入值（包括空）连接成一个数组。函数入参不支持数组形式。

返回类型：参数类型的数组。

示例：

创建表employeeinfo，并插入数据：

```
CREATE TABLE employeeinfo (empno smallint, ename varchar(20), job varchar(20), hiredate date,deptno
smallint);
INSERT INTO employeeinfo VALUES (7155, 'JACK', 'SALESMAN', '2018-12-01', 30);
INSERT INTO employeeinfo VALUES (7003, 'TOM', 'FINANCE', '2016-06-15', 20);
INSERT INTO employeeinfo VALUES (7357, 'MAX', 'SALESMAN', '2020-10-01', 30);

SELECT * FROM employeeinfo;
 empno | ename |   job    |       hiredate       | deptno
-------+-------+----------+----------------------+--------
  7155 | JACK  | SALESMAN | 2018-12-01 00:00:00 |     30
  7357 | MAX   | SALESMAN | 2020-10-01 00:00:00 |     30
  7003 | TOM   | FINANCE  | 2016-06-15 00:00:00 |     20
(3 rows)
```

查询部门编号为30的所有员工姓名：

```
SELECT array_agg(ename) FROM employeeinfo where deptno = 30;
 array_agg
------------
 {JACK,MAX}
(1 row)
```

查询属于同一个部门的所有员工：

```
SELECT deptno, array_agg(ename) FROM employeeinfo group by deptno;
 deptno | array_agg
--------+------------
     30 | {JACK,MAX}
     20 | {TOM}
(2 rows)

SELECT distinct array_agg(ename) OVER (PARTITION BY deptno) FROM employeeinfo;
 array_agg
------------
 {TOM}
 {JACK,MAX}
(2 rows)
```

查询所有的部门编号且去重：

 版权所有 © 华为云计算技术有限公司

```
SELECT array_agg(distinct deptno) FROM employeeinfo group by deptno;
 array_agg
-----------
 {20}
 {30}
(2 rows)
```

查询所有的部门编号去重后按降序排列：

```
SELECT array_agg(distinct deptno order by deptno desc) FROM employeeinfo;
 array_agg
-----------
 {30,20}
(1 row)
```

string_agg(expression, delimiter)

描述：将输入值连接成为一个字符串，用分隔符分开。

返回类型：和参数数据类型相同。

示例：

基于创建的表employeeinfo，查询属于同一个部门的所有员工：

```
SELECT deptno, string_agg(ename,',') FROM employeeinfo group by deptno;
 deptno | string_agg
--------+------------
     30 | JACK,MAX
     20 | TOM
(2 rows)
```

查询工号小于7156的员工：

```
SELECT string_agg(ename,',') FROM employeeinfo where empno < 7156;
 string_agg
------------
 TOM,JACK
(1 row)
```

listagg(expression [, delimiter]) WITHIN GROUP(ORDER BY order-list)

描述：将聚集列数据按WITHIN GROUP指定的排序方式排列，并用delimiter指定的分隔符拼接成一个字符串。

- expression：必选。指定聚集列名或基于列的有效表达式，不支持DISTINCT关键字和VARIADIC参数。
- delimiter：可选。指定分隔符，可以是字符串常数或基于分组列的确定性表达式，缺省时表示分隔符为空。
- order-list：必选。指定分组内的排序方式。

返回类型：text

说明

listagg是兼容Oracle 11g2的列转行聚集函数，可以指定OVER子句用作窗口函数。为了避免与函数本身WITHIN GROUP子句的ORDER BY造成二义性，listagg用作窗口函数时，OVER子句不支持ORDER BY的窗口排序或窗口框架。

示例：

聚集列是文本字符集类型：

 版权所有 © 华为云计算技术有限公司

```
SELECT deptno, listagg(ename, ',') WITHIN GROUP(ORDER BY ename) AS employees FROM emp GROUP
BY deptno;
 deptno |              employees
--------+--------------------------------------
     10 | CLARK,KING,MILLER
     20 | ADAMS,FORD,JONES,SCOTT,SMITH
     30 | ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD
(3 rows)
```

聚集列是整型：

```
SELECT deptno, listagg(mgrno, ',') WITHIN GROUP(ORDER BY mgrno NULLS FIRST) AS mgrnos FROM emp
GROUP BY deptno;
 deptno |            mgrnos
--------+-------------------------------
     10 | 7782,7839
     20 | 7566,7566,7788,7839,7902
     30 | 7698,7698,7698,7698,7698,7839
(3 rows)
```

聚集列是浮点类型：

```
SELECT job, listagg(bonus, '($); ') WITHIN GROUP(ORDER BY bonus DESC) || '($)' AS bonus FROM emp
GROUP BY job;
   job     |                      bonus
-----------+-------------------------------------------------
 CLERK     | 10234.21($); 2000.80($); 1100.00($); 1000.22($)
 PRESIDENT | 23011.88($)
 ANALYST   | 2002.12($); 1001.01($)
 MANAGER   | 10000.01($); 2399.50($); 999.10($)
 SALESMAN  | 1000.01($); 899.00($); 99.99($); 9.00($)
(5 rows)
```

聚集列是时间类型：

```
SELECT deptno, listagg(hiredate, ', ') WITHIN GROUP(ORDER BY hiredate DESC) AS hiredates FROM emp
GROUP BY deptno;
 deptno |                                                hiredates
--------
+-----------------------------------------------------------------------------------------------------------------------------
--
     10 | 1982-01-23 00:00:00, 1981-11-17 00:00:00, 1981-06-09 00:00:00
     20 | 2001-04-02 00:00:00, 1999-12-17 00:00:00, 1987-05-23 00:00:00, 1987-04-19 00:00:00, 1981-12-03
00:00:00
     30 | 2015-02-20 00:00:00, 2010-02-22 00:00:00, 1997-09-28 00:00:00, 1981-12-03 00:00:00, 1981-09-08
00:00:00, 1981-05-01 00:00:00
(3 rows)
```

聚集列是时间间隔类型：

```
SELECT deptno, listagg(vacationTime, '; ') WITHIN GROUP(ORDER BY vacationTime DESC) AS vacationTime
FROM emp GROUP BY deptno;
 deptno |                                  vacationtime
--------+-------------------------------------------------------------------------------
     10 | 1 year 30 days; 40 days; 10 days
     20 | 70 days; 36 days; 9 days; 5 days
     30 | 1 year 1 mon; 2 mons 10 days; 30 days; 12 days 12:00:00; 4 days 06:00:00; 24:00:00
(3 rows)
```

分隔符缺省时，默认为空：

```
SELECT deptno, listagg(job) WITHIN GROUP(ORDER BY job) AS jobs FROM emp GROUP BY deptno;
 deptno |                    jobs
--------+-----------------------------------------------
     10 | CLERKMANAGERPRESIDENT
     20 | ANALYSTANALYSTCLERKCLERKMANAGER
     30 | CLERKMANAGERSALESMANSALESMANSALESMANSALESMAN
(3 rows)
```

 版权所有 © 华为云计算技术有限公司

listagg作为窗口函数时，OVER子句不支持ORDER BY的窗口排序，listagg列为对应分组的有序聚集：

```
SELECT deptno, mgrno, bonus, listagg(ename,'; ') WITHIN GROUP(ORDER BY hiredate) OVER(PARTITION
BY deptno) AS employees FROM emp;
 deptno | mgrno |  bonus   |                 employees
--------+-------+----------+-------------------------------------------
     10 |  7839 | 10000.01 | CLARK; KING; MILLER
     10 |       | 23011.88 | CLARK; KING; MILLER
     10 |  7782 | 10234.21 | CLARK; KING; MILLER
     20 |  7566 |  2002.12 | FORD; SCOTT; ADAMS; SMITH; JONES
     20 |  7566 |  1001.01 | FORD; SCOTT; ADAMS; SMITH; JONES
     20 |  7788 |  1100.00 | FORD; SCOTT; ADAMS; SMITH; JONES
     20 |  7902 |  2000.80 | FORD; SCOTT; ADAMS; SMITH; JONES
     20 |  7839 |   999.10 | FORD; SCOTT; ADAMS; SMITH; JONES
     30 |  7839 |  2399.50 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
     30 |  7698 |     9.00 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
     30 |  7698 |  1000.22 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
     30 |  7698 |    99.99 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
     30 |  7698 |  1000.01 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
     30 |  7698 |   899.00 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN
(14 rows)
```

group_concat(expression [ORDER BY {col_name | expr} [ASC | DESC]] [SEPARATOR str_val])

描述：将列数据使用指定的str_val分隔符，按照ORDER BY子句指定的排序方式拼接成字符串，ORDER BY子句必须指定排序方式，不支持ORDER BY 1的写法。

- expression：必选，指定列名或基于列的有效表达式，不支持DISTINCT关键字和VARIADIC参数。
- str_val：可选，指定的分隔符，可以是字符串常数或基于分组列的确定性表达式。缺省时表示分隔符为逗号。

返回类型：text

说明

group_concat函数仅8.1.2及以上版本支持。

示例：

默认分隔符为逗号：

```
SELECT group_concat(sname) FROM group_concat_test;
              group_concat
-------------------------------------------
 ADAMS,FORD,JONES,KING,MILLER,SCOTT,SMITH
(1 row)
```

group_concat函数支持自定义分隔符：

```
SELECT group_concat(sname separator ';') from group_concat_test;
              group_concat
-------------------------------------------
 ADAMS;FORD;JONES;KING;MILLER;SCOTT;SMITH
(1 row)
```

group_concat函数支持ORDER BY子句，将列数据进行有序拼接：

```
SELECT group_concat(sname order by snumber separator ';') FROM group_concat_test;
              group_concat
-------------------------------------------
 MILLER;FORD;SCOTT;SMITH;KING;JONES;ADAMS
(1 row)
```


版权所有 © 华为云计算技术有限公司

covar_pop(Y, X)

描述：总体协方差。

返回类型：double precision

示例：

```
 SELECT COVAR_POP(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
    covar_pop
------------------
 829.749627587403
(1 row)
```

covar_samp(Y, X)

描述：样本协方差。

返回类型：double precision

示例：

```
SELECT COVAR_SAMP(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
    covar_samp
------------------
 830.052235037289
(1 row)
```

stddev_pop(expression)

描述：总体标准差。

返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。

示例：

```
SELECT STDDEV_POP(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
    stddev_pop
------------------
 289.224294957556
(1 row)
```

stddev_samp(expression)

描述：样本标准差。

返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。

示例：

```
SELECT STDDEV_SAMP(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
   stddev_samp
------------------
 289.224359757315
(1 row)
```

var_pop(expression)

描述：总体方差（总体标准差的平方）。

返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT VAR_POP(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
      var_pop
--------------------
 83650.692793695475
(1 row)
```

var_samp(expression)

描述：样本方差（样本标准差的平方）。

返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。

示例：

```
SELECT VAR_SAMP(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
      var_samp
--------------------
 83650.730277028768
(1 row)
```

bit_and(expression)

描述：所有非NULL输入值的按位与（AND），如果全部输入值皆为NULL，那么结果也为NULL 。

返回类型：和参数数据类型相同。

示例：

```
SELECT BIT_AND(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
 bit_and
---------
       0
(1 row)
```

bit_or(expression)

描述：所有非NULL输入值的按位或（OR），如果全部输入值皆为NULL，那么结果也为NULL。

返回类型：和参数数据类型相同

示例：

```
SELECT BIT_OR(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
 bit_or
--------
   1023
(1 row)
```

bool_and(expression)

描述：如果所有输入值都是真，则为真，否则为假。

返回类型：bool

示例：

```
SELECT bool_and(100 <2500);
 bool_and
```

 版权所有 © 华为云计算技术有限公司

```
----------
 t
(1 row)
```

bool_or(expression)

描述：如果所有输入值只要有一个为真，则为真，否则为假。

返回类型：bool

示例：

```
SELECT bool_or(100 <2500);
 bool_or
----------
 t
(1 row)
```

corr(Y, X)

描述：相关系数。

返回类型：double precision

示例：

```
SELECT CORR(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
       corr
-------------------
 0.0381383624904186
(1 row)
```

every(expression)

描述：等效于bool_and。

返回类型：bool

示例：

```
SELECT every(100 <2500);
 every
-------
 t
(1 row)
```

rank(expression)

描述：根据expression对不同组内的元组进行跳跃排序。

返回类型：bigint

示例：

```
SELECT d_moy, d_fy_week_seq, rank() OVER(PARTITION BY d_moy ORDER BY d_fy_week_seq) FROM
tpcds.date_dim WHERE d_moy < 4 AND d_fy_week_seq < 7 ORDER BY 1,2;
  d_moy | d_fy_week_seq | rank
-------+---------------+------
     1 |             1 |    1
     1 |             1 |    1
     1 |             1 |    1
     1 |             1 |    1
     1 |             1 |    1
     1 |             1 |    1
```


版权所有 © 华为云计算技术有限公司

```
   1 |          1 |   1
   1 |          2 |   8
   1 |          2 |   8
   1 |          2 |   8
   1 |          2 |   8
   1 |          2 |   8
   1 |          2 |   8
   1 |          2 |   8
   1 |          3 |  15
   1 |          3 |  15
   1 |          3 |  15
   1 |          3 |  15
   1 |          3 |  15
   1 |          3 |  15
   1 |          3 |  15
   1 |          4 |  22
   1 |          4 |  22
   1 |          4 |  22
   1 |          4 |  22
   1 |          4 |  22
   1 |          4 |  22
   1 |          4 |  22
   1 |          5 |  29
   1 |          5 |  29
   2 |          5 |   1
   2 |          5 |   1
   2 |          5 |   1
   2 |          5 |   1
   2 |          5 |   1
   2 |          6 |   6
   2 |          6 |   6
   2 |          6 |   6
   2 |          6 |   6
   2 |          6 |   6
   2 |          6 |   6
   2 |          6 |   6
(42 rows)
```

regr_avgx(Y, X)

描述：自变量的平均值 (sum(X)/N)。

返回类型：double precision

示例：

```
SELECT REGR_AVGX(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
   regr_avgx
------------------
 578.606576740795
(1 row)
```

regr_avgy(Y, X)

描述：因变量的平均值 (sum(Y)/N)。

返回类型：double precision

示例：

```
SELECT REGR_AVGY(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
   regr_avgy
------------------
 50.0136711629602
(1 row)
```


版权所有 © 华为云计算技术有限公司

regr_count(Y, X)

描述：两个表达式都不为NULL的输入行数。

返回类型：bigint

示例：

```
SELECT REGR_COUNT(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
 regr_count
------------
       2743
(1 row)
```

regr_intercept(Y, X)

描述：根据所有输入的点(X, Y)按照最小二乘法拟合成一个线性方程，然后返回该直线的Y轴截距。

返回类型：double precision

示例：

```
SELECT REGR_INTERCEPT(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
  regr_intercept
------------------
 49.2040847848607
(1 row)
```

regr_r2(Y, X)

描述：相关系数的平方。

返回类型：double precision

示例：

```
SELECT REGR_R2(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
      regr_r2
--------------------
 0.00145453469345058
(1 row)
```

regr_slope(Y, X)

描述：根据所有输入的点(X, Y)按照最小二乘法拟合成一个线性方程， 然后返回该直线的斜率。

返回类型：double precision

示例：

```
SELECT REGR_SLOPE(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
     regr_slope
--------------------
 0.00139920009665259
(1 row)
```

regr_sxx(Y, X)

描述：sum(X^2) - sum(X)^2/N （自变量的“平方和”）。

返回类型：double precision

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT REGR_SXX(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
    regr_sxx
------------------
 1626645991.46135
(1 row)
```

regr_sxy(Y, X)

描述：sum(X*Y) - sum(X) * sum(Y)/N （自变量和因变量的“乘方积”）。

返回类型：double precision

示例：

```
SELECT REGR_SXY(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
    regr_sxy
------------------
 2276003.22847225
(1 row)
```

regr_syy(Y, X)

描述：sum(Y^2) - sum(Y)^2/N（因变量的“平方和”）。

返回类型：double precision

示例：

```
SELECT REGR_SYY(sr_fee, sr_net_loss) FROM tpcds.store_returns WHERE sr_customer_sk < 1000;
   regr_syy
-----------------
 2189417.6547314
(1 row)
```

stddev(expression)

描述：stddev_samp的别名。

返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。

示例：

```
SELECT STDDEV(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
     stddev
------------------
 289.224359757315
(1 row)
```

variance(expression)

描述：var_samp的别名。

返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。

示例：

```
SELECT VARIANCE(inv_quantity_on_hand) FROM tpcds.inventory WHERE inv_warehouse_sk = 1;
     variance
--------------------
 83650.730277028768
(1 row)
```

 版权所有 © 华为云计算技术有限公司

checksum(expression)

描述：返回所有输入值的CHECKSUM值。使用该函数可以用来验证DWS数据库（不支持DWS之外的其他数据库）的备份恢复或者数据迁移操作前后表中的数据是否相同。在备份恢复或者数据迁移操作前后都需要用户通过手工执行SQL命令的方式获取执行结果，通过对比获取的执行结果判断操作前后表中的数据是否相同。

说明

- 对于大表，CHECKSUM函数可能会需要很长时间。
- 如果某两表的CHECKSUM值不同，则表明两表的内容是不同的。由于CHECKSUM函数中使用散列函数不能保证无冲突，因此两个不同内容的表可能会得到相同的CHECKSUM值，存在这种情况的可能性较小。对于列进行的CHECKSUM也存在相同的情况。
- 对于时间类型timestamp, timestamptz和smalldatetime，计算CHECKSUM值时请确保时区设置一致。

- 若计算某列的CHECKSUM值，且该列类型可以默认转为TEXT类型，则expression为列名。
- 若计算某列的CHECKSUM值，且该列类型不能默认转为TEXT类型，则expression为列名::TEXT。
- 若计算所有列的CHECKSUM值，则expression为表名::TEXT。

可以默认转换为TEXT类型的类型包括：char, name, int8, int2, int1, int4, raw, pg_node_tree, float4, float8, bpchar, varchar, nvarchar2, date, timestamp, timestamptz, numeric, smalldatetime，其他类型需要强制转换为TEXT。

返回类型：numeric

示例：

表中可以默认转为TEXT类型的某列的CHECKSUM值：

```
SELECT CHECKSUM(inv_quantity_on_hand) FROM tpcds.inventory;
    checksum
--------------------
 24417258945265247
(1 row)
```

表中不能默认转为TEXT类型的某列的CHECKSUM值（注意此时CHECKSUM参数是列名::TEXT）：

```
SELECT CHECKSUM(inv_quantity_on_hand::TEXT) FROM tpcds.inventory;
    checksum
--------------------
 24417258945265247
(1 row)
```

表中所有列的CHECKSUM值。注意此时CHECKSUM参数是表名::TEXT，且表名前不加Schema：

```
SELECT CHECKSUM(inventory::TEXT) FROM tpcds.inventory;
    checksum
--------------------
 25223696246875800
(1 row)
```

approx_count_distinct(col_name)

描述：使用HyperLogLog++ (HLL++) 算法进行基数（某一列去重后的行数）的估算。该函数仅8.3.0及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

入参说明：col_name指需要估算基数的列。

说明

可通过GUC参数approx_count_distinct_precision调整误差率。

- 参数取值范围为[10,20]，默认值为17，理论误差率为千分之三。
- 该参数表示HyperLogLog++ (HLL++)算法中分桶个数，参数越大时，分桶数则越大，理论误差率则越小。
- 该参数取值越大，相应的计算时间和内存资源开销越大，但依然远小于精确的count distinct语句对应的开销。推荐在估算基数较大时使用该函数替换count distinct。

示例：

```
CREATE TABLE employeeinfo (empno smallint, ename varchar(20), job varchar(20), hiredate date,deptno smallint) WITH (ORIENTATION = COLUMN);
INSERT INTO employeeinfo VALUES (7155, 'JACK', 'SALESMAN', '2018-12-01', 30);
INSERT INTO employeeinfo VALUES (7003, 'TOM', 'FINANCE', '2016-06-15', 20);
INSERT INTO employeeinfo VALUES (7357, 'MAX', 'SALESMAN', '2020-10-01', 30);

SELECT APPROX_COUNT_DISTINCT(empno) from employeeinfo;
 approx_count_distinct
-----------------------
                     3
(1 row)

SELECT COUNT(DISTINCT empno) FROM employeeinfo GROUP BY ename;
 count
-------
     1
     1
     1
(3 rows)
```

UNIQ(col_name)

描述：计算非重复值个数的聚合函数，与COUNT DISTINCT类似，即计算某一列去重后的行数，结果返回一个去重值。该函数仅8.3.0及以上集群版本支持。

入参说明：col_name指需要计算去重后行数的列。支持SMALLINT、INTEGER、BIGINT、REAL、DOUBLE PRECISION、TEXT、VARCHAR、TIMESTAMP、TIMESTAMPTZ、DATE、TIMETZ、UUID类型。

说明

- DWS目前仅列存表支持UNIQ函数。
- 使用UNIQ函数时，SQL需要包含GROUP BY，为了取得更好性能所选GROUP BY字段分布需要尽量均匀。
- 建议当SQL中需要去重的列大于等于3列时，使用UNIQ函数去重，以取得比COUNT DISTINCT更好的效果。
- 一般情况下UNIQ函数内存消耗低于COUNT DISTINCT，如果使用COUNT DISTINCT时遇到内存超限，可以使用UNIQ函数进行替换。

示例：

```
CREATE TABLE employeeinfo (empno smallint, ename varchar(20), job varchar(20), hiredate date,deptno smallint) WITH (ORIENTATION = COLUMN);
INSERT INTO employeeinfo VALUES (7155, 'JACK', 'SALESMAN', '2018-12-01', 30);
INSERT INTO employeeinfo VALUES (7003, 'TOM', 'FINANCE', '2016-06-15', 20);
INSERT INTO employeeinfo VALUES (7357, 'MAX', 'SALESMAN', '2020-10-01', 30);

SELECT UNIQ(deptno) FROM employeeinfo GROUP BY ename;
 uniq
```


版权所有 © 华为云计算技术有限公司

```
------
   1
   1
   1
(3 rows)
```

21.7.12 窗口函数

普通的聚集函数只能用来计算一行内的结果，或者把所有行聚集成一行结果。而窗口函数可以跨行计算，并且把结果填到每一行中。

通过查询筛选出的行的某些部分，窗口调用函数实现了类似于**聚集函数**的功能，所以聚集函数也可以作为窗口函数使用。 窗口函数可以扫描所有的行，并同时将原始数据和聚集分析结果同时显示出来。

注意事项

- 列存表目前只支持窗口函数rank(expression)和row_number(expression)，以及聚集函数的sum，count，avg，min和max，而行存表没有限制。
- 单个查询中可以包含一个或多个窗口函数表达式。
- 窗口函数仅能出现在输出列中。如果需要使用窗口函数的值进行条件过滤，需要将窗口函数嵌套在子查询中，在外层使用窗口函数表达式的别名进行条件过滤。例如：
  ```
  SELECT classid, id, score FROM(SELECT *, avg(score) OVER(PARTITION BY classid) as avg_score FROM score) WHERE score >= avg_score;
  ```
- 窗口函数所在查询块中支持使用GROUP BY表达式进行分组去重，但要求窗口函数中的PARTITION BY子句中必须是GROUP BY表达式的子集，以保证窗口函数在GROUP BY列去重后的结果上进行窗口运算，同时ORDER BY子句的表达式也需要是GROUP BY表达式的子集，或聚集运算的聚集函数。例如：
  ```
  SELECT classid,rank() OVER(PARTITION BY classid ORDER BY sum(score)) as avg_score FROM score GROUP BY classid, id;
  ```

语法格式

窗口函数需要特殊的关键字OVER语句来指定窗口触发窗口函数。OVER语句用于对数据进行分组，并对组内元素进行排序。窗口函数用于给组内的值生成序号：

```
function_name ([expression [, expression ... ]]) OVER ( window_definition )
function_name ([expression [, expression ... ]]) OVER window_name
function_name ( * ) OVER ( window_definition )
function_name ( * ) OVER window_name
```

其中window_definition子句option为：

```
[ existing_window_name ]
[ PARTITION BY expression [, ...] ]
[ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]
[ frame_clause ]
```

PARTITION BY选项指定了将具有相同PARTITION BY表达式值的行分为一组。

ORDER BY选项用于控制窗口函数处理行的顺序。ORDER BY后面必须跟字段名，若ORDER BY后面跟数字，该数字会被按照常量处理，对目标列没有起到排序的作用。

frame_clause子句option为：

```
[ RANGE | ROWS ] frame_start
[ RANGE | ROWS ] BETWEEN frame_start AND frame_end
```


版权所有 © 华为云计算技术有限公司

当需要指定一个窗口对分组内所有行结果进行计算时，我们需要指定窗口区间开始的行和结束的行。窗口区间支持RANGE、ROWS两种模式，ROWS以物理单位（行）指定窗口，RANGE将窗口指定为逻辑偏移量。

RANGE、ROWS中可以使用BETWEEN frame_start AND frame_end指定边界可取值。如果仅指定frame_start，则frame_end默认为CURRENT ROW。

frame_start和frame_end取值为：

- CURRENT ROW，当前行。
- N PRECEDING，当前行向前第n行。
- UNBOUNDED PRECEDING，当前PARTITION的第1行。
- N FOLLOWING，当前行向后第n行。
- UNBOUNDED FOLLOWING，当前PARTITION的最后1行。

需要注意，frame_start不能为UNBOUNDED FOLLOWING，frame_end不能为UNBOUNDED PRECEDING，并且frame_end选项不能比上面取值中出现的frame_start选项早。例如RANGE BETWEEN CURRENT ROW AND N PRECEDING是不被允许的。

RANK()

描述：RANK函数为各组内值生成跳跃排序序号，其中相同的值具有相同序号，但相同值占用多个编号。

返回值类型：BIGINT

示例：

给定表score(id, classid, score)，每行表示学生id，所在班级id以及考试成绩。

使用RANK函数对学生成绩进行排序：

```
CREATE TABLE score(id int,classid int,score int);
INSERT INTO score VALUES(1,1,95),(2,2,95),(3,2,85),(4,1,70),(5,2,88),(6,1,70);

SELECT id, classid, score,RANK() OVER(ORDER BY score DESC) FROM score;
 id | classid | score | rank
----+---------+-------+------
  1 |       1 |    95 |    1
  2 |       2 |    95 |    1
  6 |       1 |    70 |    5
  4 |       1 |    70 |    5
  5 |       2 |    88 |    3
  3 |       2 |    85 |    4
(6 rows)
```

ROW_NUMBER()

描述：ROW_NUMBER函数为各组内值生成连续排序序号，其中相同的值其序号也不相同。

返回值类型：BIGINT

示例：

```
SELECT id, classid, score,ROW_NUMBER() OVER(ORDER BY score DESC) FROM score ORDER BY score DESC;
 id | classid | score | row_number
----+---------+-------+------------
  1 |       1 |    95 |          1
```


版权所有 © 华为云计算技术有限公司

```
 2 |       2 |    95 |          2
 5 |       2 |    88 |          3
 3 |       2 |    85 |          4
 6 |       1 |    70 |          5
 4 |       1 |    70 |          6
(6 rows)
```

DENSE_RANK()

描述：DENSE_RANK函数为各组内值生成连续排序序号，其中相同的值具有相同序号，相同值只占用一个编号。

返回值类型：BIGINT

示例：

```
SELECT id, classid, score,DENSE_RANK() OVER(ORDER BY score DESC) FROM score;
 id | classid | score | dense_rank
----+---------+-------+------------
 1 |       1 |    95 |          1
 2 |       2 |    95 |          1
 5 |       2 |    88 |          2
 3 |       2 |    85 |          3
 6 |       1 |    70 |          4
 4 |       1 |    70 |          4
(6 rows)
```

PERCENT_RANK()

描述：PERCENT_RANK函数为各组内对应值生成相对序号，即根据公式 (rank - 1) / (total rows - 1)计算所得的值。其中rank为该值依据RANK函数所生成的对应序号，totalrows为该分组内的总元素个数。

返回值类型：DOUBLE PRECISION

示例：

```
SELECT id, classid, score,PERCENT_RANK() OVER(ORDER BY score DESC) FROM score;
 id | classid | score | percent_rank
----+---------+-------+--------------
 1 |       1 |    95 |            0
 2 |       2 |    95 |            0
 3 |       2 |    85 |           .6
 4 |       1 |    70 |           .8
 5 |       2 |    88 |           .4
 6 |       1 |    70 |           .8
(6 rows)
```

CUME_DIST()

描述：CUME_DIST函数为各组内对应值生成累积分布序号。即根据公式(小于等于当前值的数据行数)/(该分组总行数totalrows)计算所得的相对序号。

返回值类型：DOUBLE PRECISION

示例：

```
SELECT id,classid,score,CUME_DIST() OVER(ORDER BY score DESC) FROM score;
 id | classid | score |     cume_dist
----+---------+-------+-------------------
 1 |       1 |    95 | .333333333333333
 2 |       2 |    95 | .333333333333333
 5 |       2 |    88 |               .5
 3 |       2 |    85 | .666666666666667
```


版权所有 © 华为云计算技术有限公司

```
 4 |      1 |   70 |             1
 6 |      1 |   70 |             1
(6 rows)
```

NTILE(num_buckets integer)

描述：NTILE函数根据num_buckets integer将有序的数据集合平均分配到num_buckets所指定数量的桶中，并将桶号分配给每一行。分配时应尽量做到平均分配。

返回值类型：INTEGER

示例：

```
SELECT id,classid,score,NTILE(3) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | ntile
----+---------+-------+-------
 1 |      1 |   95 |     1
 2 |      2 |   95 |     1
 5 |      2 |   88 |     2
 3 |      2 |   85 |     2
 4 |      1 |   70 |     3
 6 |      1 |   70 |     3
(6 rows)
```

LAG(value any [, offset integer [, default any]])

描述：LAG函数为各组内对应值生成滞后值。即当前值对应的行数往前偏移offset位后所得行的value值作为序号。若经过偏移后行数不存在，则对应结果取为default值。若无指定，在默认情况下，offset取为1，default值取为NULL。

返回值类型：与参数数据类型相同。

示例：

```
SELECT id,classid,score,LAG(id,3) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | lag
----+---------+-------+-----
 1 |      1 |   95 |
 2 |      2 |   95 |
 5 |      2 |   88 |
 3 |      2 |   85 |   1
 4 |      1 |   70 |   2
 6 |      1 |   70 |   5
(6 rows)
```

LEAD(value any [, offset integer [, default any]])

描述：LEAD函数为各组内对应值生成提前值。即当前值对应的行数向后偏移offset位后所得行的value值作为序号。若经过向后偏移后行数超过当前组内的总行数，则对应结果取为default值。若无指定，在默认情况下，offset取为1，default值取为NULL。

返回值类型：与参数数据类型相同。

示例：

```
SELECT id,classid,score,LEAD(id,3) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | lead
----+---------+-------+------
 1 |      1 |   95 |    3
 2 |      2 |   95 |    4
 5 |      2 |   88 |    6
 3 |      2 |   85 |
 4 |      1 |   70 |
```


版权所有 © 华为云计算技术有限公司

```
 6 |      1 |   70 |
(6 rows)
```

FIRST_VALUE(value any)

描述：FIRST_VALUE函数取各组内的第一个值作为返回结果。

返回值类型：与参数数据类型相同。

示例：

```
SELECT id,classid,score,FIRST_VALUE(id) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | first_value
----+---------+-------+-------------
 1 |      1 |   95 |           1
 2 |      2 |   95 |           1
 5 |      2 |   88 |           1
 3 |      2 |   85 |           1
 4 |      1 |   70 |           1
 6 |      1 |   70 |           1
(6 rows)
```

LAST_VALUE(value any)

描述：LAST_VALUE函数取各组内的最后一个值作为返回结果。

返回值类型：与参数数据类型相同。

LAST_VALUE函数支持IGNORE NULLS语法，该语法返回非NULL窗口中的最后一个值，如果所有值都为NULL，则返回NULL，具体格式为：

```
LAST_VALUE (expression) [IGNORE NULLS] OVER (window_definition)
```

当前IGNORE NULLS仅支持ROWS between CURRENT ROW and UNBOUNDED FOLLOWING和ROWS BETWEEN UNBOUNDED PRECEDING and CURRENT ROW两种窗口区间。

示例：

```
SELECT id,classid,score,LAST_VALUE(id) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | last_value
----+---------+-------+------------
 1 |      1 |   95 |          2
 2 |      2 |   95 |          2
 5 |      2 |   88 |          5
 3 |      2 |   85 |          3
 4 |      1 |   70 |          6
 6 |      1 |   70 |          6
(6 rows)
```

NTH_VALUE(value any, nth integer)

描述：NTH_VALUE函数返回该组内的第nth行作为结果。若该行不存在，则默认返回NULL。

返回值类型：与参数数据类型相同。

示例：

```
SELECT id,classid,score,NTH_VALUE(id,3) OVER(ORDER BY score DESC) FROM score;
 id | classid | score | nth_value
----+---------+-------+-----------
 1 |      1 |   95 |
 2 |      2 |   95 |
```


版权所有 © 华为云计算技术有限公司

```
 5 |      2 |   88 |        5
 3 |      2 |   85 |        5
 4 |      1 |   70 |        5
 6 |      1 |   70 |        5
(6 rows)
```

21.7.13 类型转换函数

cast(x as y)

描述：类型转换函数，将x转换成y指定的类型。

示例：

```
SELECT cast('22-oct-1997' as timestamp);
     timestamp
---------------------
 1997-10-22 00:00:00
(1 row)
```

cast(x, y)

描述：类型转换函数，将x转换成y指定的类型。该函数仅8.2.0及以上集群版本支持。

示例：

```
SELECT cast('22-oct-1997', timestamp);
     timestamp
---------------------
 1997-10-22 00:00:00
(1 row)
```

try_cast(x as type)

描述：将x转换成给定的type类型值，若转换失败且当前类型转换为DWS允许的转换，则返回NULL，否则报错。该函数仅8.2.0及以上集群版本支持。

示例：

```
SELECT try_cast('a' as int4);
     int4
---------------


(1 row)
SELECT try_cast('22-oct-1997' as timestamp);
     timestamp
---------------------
 1997-10-22 00:00:00
(1 row)
```

hextoraw(string)

描述：将一个十六进制构成的字符串转换为二进制。

返回值类型：raw

示例：

```
SELECT hextoraw('7D');
 hextoraw
----------
 7D
(1 row)
```


版权所有 © 华为云计算技术有限公司

numtoday(numeric)

描述：将数字类型的值转换为指定格式的时间戳。

返回值类型：timestamp

示例：

```
SELECT numtoday(2);
 numtoday
----------
 2 days
(1 row)
```

pg_systimestamp()

描述：获取系统时间戳。

返回值类型：timestamp with time zone

示例：

```
SELECT pg_systimestamp();
        pg_systimestamp
-------------------------------
 2015-10-14 11:21:28.317367+08
(1 row)
```

rawtohex(string)

描述：将一个二进制构成的字符串转换为十六进制的字符串。

结果为输入字符的ASCII码，以十六进制表示。

返回值类型：varchar

示例：

```
SELECT rawtohex('1234567');
    rawtohex
----------------
 31323334353637
(1 row)
```

to_char (datetime/interval [, fmt])

描述：将一个DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE或者TIMESTAMP WITH LOCAL TIME ZONE类型的DATETIME或者INTERVAL值按照fmt指定的格式转换为VARCHAR类型。

- 可选参数fmt可以为以下几类：日期、时间、星期、季度和世纪。每类都可以有不同的模板，模板之间可以合理组合，常见的模板有：HH、MI、SS、YYYY、MM、DD，可参考**表21-41**。
- 模板可以有修饰词，常用的修饰词是FM，可以用来抑制前导的零或尾随的空白。

返回值类型：varchar

示例：

```
SELECT to_char(current_timestamp,'HH12:MI:SS');
 to_char
----------
```

 版权所有 © 华为云计算技术有限公司

```
 10:19:26
(1 row)
SELECT to_char(current_timestamp,'FMHH12:FMMI:FMSS');
 to_char
----------
 10:19:46
(1 row)
```

to_char(double precision, text)

描述：将双精度类型的值转换为指定格式的字符串。

返回值类型：text

示例：

```
SELECT to_char(125.8::real, '999D99');
 to_char
---------
  125.80
(1 row)
```

to_char (integer/number[, fmt])

描述：将一个整型或者浮点类型的值转换为指定格式的字符串。

- 可选参数fmt可以为以下几类：十进制字符、“分组”符、正负号和货币符号，每类都可以有不同的模板，模板之间可以合理组合，常见的模板有：9、0、,（千分隔符）、.（小数点），可参考**表21-40**。
- 模板可以有类似FM的修饰词，但FM不抑制由模板0指定而输出的0。
- 要将整型类型的值转换成对应16进制值的字符串，使用模板“x”或“X”。

返回值类型：varchar

示例：

```
SELECT to_char(1485,'9,999');
 to_char
---------
  1,485
(1 row)
SELECT to_char( 1148.5,'9,999.999');
  to_char
------------
  1,148.500
(1 row)
SELECT to_char(148.5,'990999.909');
   to_char
-------------
    0148.500
(1 row)
SELECT to_char(123,'XXX');
 to_char
---------
   7B
(1 row)
```

to_char(interval, text)

描述：将时间间隔类型的值转换为指定格式的字符串。

返回值类型：text

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT to_char(interval '15h 2m 12s', 'HH24:MI:SS');
 to_char
----------
 15:02:12
(1 row)
```

to_char(int, text)

描述：将整数类型的值转换为指定格式的字符串。

返回值类型：text

示例：

```
SELECT to_char(125, '999');
 to_char
---------
  125
(1 row)
```

to_char(numeric, text)

描述：将数字类型的值转换为指定格式的字符串。

返回值类型：text

示例：

```
SELECT to_char(-125.8, '999D99S');
 to_char
---------
 125.80-
(1 row)
```

to_char (string)

描述：将CHAR、VARCHAR、VARCHAR2、CLOB类型转换为VARCHAR类型。

如使用该函数对CLOB类型进行转换，且待转换CLOB类型的值超出目标类型的范围，则返回错误。

返回值类型：varchar

示例：

```
SELECT to_char('01110');
 to_char
---------
 01110
(1 row)
```

to_char(timestamp, text)

描述：将时间戳类型的值转换为指定格式的字符串。

返回值类型：text

示例：

```
SELECT to_char(current_timestamp, 'HH12:MI:SS');
 to_char
----------
 10:55:59
(1 row)
```

 版权所有 © 华为云计算技术有限公司

to_clob(char/nchar/varchar/nvarchar/varchar2/nvarchar2/text/raw)

描述：将RAW类型或者文本字符集类型CHAR、NCHAR、VARCHAR、VARCHAR2、NVARCHAR2、TEXT转成CLOB类型。

返回值类型：clob

示例：

```
SELECT to_clob('ABCDEF'::RAW(10));
 to_clob
---------
 ABCDEF
(1 row)
SELECT to_clob('hello111'::CHAR(15));
 to_clob
----------
 hello111
(1 row)
SELECT to_clob('gauss123'::NCHAR(10));
 to_clob
----------
 gauss123
(1 row)
SELECT to_clob('gauss234'::VARCHAR(10));
 to_clob
----------
 gauss234
(1 row)
SELECT to_clob('gauss345'::VARCHAR2(10));
 to_clob
----------
 gauss345
(1 row)
SELECT to_clob('gauss456'::NVARCHAR2(10));
 to_clob
----------
 gauss456
(1 row)
SELECT to_clob('World222!'::TEXT);
  to_clob
-----------
 World222!
(1 row)
```

to_date(text)

描述：将文本类型的值转换为指定格式的时间戳。

返回值类型：timestamp

示例：

```
SELECT to_date('2015-08-14');
       to_date
---------------------
 2015-08-14 00:00:00
(1 row)
```

to_date(text, text)

描述：将字符串类型的值转换为指定格式的日期。

返回值类型：timestamp

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT to_date('05 Dec 2000', 'DD Mon YYYY');
      to_date
---------------------
 2000-12-05 00:00:00
(1 row)
```

to_date(string, fmt)

描述：将字符串string按fmt指定格式转化为DATE类型的值。fmt格式可参考**表21-41**。

该函数不能直接支持CLOB类型，但是CLOB类型的参数能够通过隐式转换实现。

返回值类型：date

示例：

```
SELECT TO_DATE('05 Dec 2010','DD Mon YYYY');
      to_date
---------------------
 2010-12-05 00:00:00
(1 row)
```

to_number (expr [, fmt])

描述：将expr按指定格式转换为一个NUMBER类型的值。

类型转换格式请参考**表21-40**。

转换十六进制字符串为十进制数字时，最多支持16个字节的十六进制字符串转换为无符号数。

转换十六进制字符串为十进制数字时，格式字符串中不允许出现除'x'或'X'以外的其他字符，否则报错。

返回值类型：number

示例：

```
SELECT to_number('12,454.8-', '99G999D9S');
 to_number
-----------
  -12454.8
(1 row)
```

to_number(text, text)

描述：将字符串类型的值转换为指定格式的数字。

返回值类型：numeric

示例：

```
SELECT to_number('12,454.8-', '99G999D9S');
 to_number
-----------
  -12454.8
(1 row)
```

to_timestamp(double precision)

描述：把UNIX纪元转换成时间戳。

 版权所有 © 华为云计算技术有限公司

返回值类型：timestamp with time zone

示例：

```
SELECT to_timestamp(1284352323);
      to_timestamp
------------------------
 2010-09-13 12:32:03+08
(1 row)
```

to_timestamp(string [,fmt])

描述：将字符串string按fmt指定的格式转换成时间戳类型的值。不指定fmt时，按参数nls_timestamp_format所指定的格式转换。fmt格式可参考**表21-41**。

DWS的to_timestamp中，

- 如果输入的年份YYYY=0，系统报错。
- 如果输入的年份YYYY<0，在fmt中指定SYYYY，则正确输出公元前绝对值n的年份。

fmt中出现的字符必须与日期/时间格式化的模式相匹配，否则报错。

返回值类型：timestamp without time zone

示例：

```
SHOW nls_timestamp_format;
    nls_timestamp_format
-----------------------------
 DD-Mon-YYYY HH:MI:SS.FF AM
(1 row)

SELECT to_timestamp('12-sep-2014');
    to_timestamp
---------------------
 2014-09-12 00:00:00
(1 row)
SELECT to_timestamp('12-Sep-10 14:10:10.123000','DD-Mon-YY HH24:MI:SS.FF');
      to_timestamp
-------------------------
 2010-09-12 14:10:10.123
(1 row)
SELECT to_timestamp('-1','SYYYY');
      to_timestamp
------------------------
 0001-01-01 00:00:00 BC
(1 row)
SELECT to_timestamp('98','RR');
    to_timestamp
---------------------
 1998-01-01 00:00:00
(1 row)
SELECT to_timestamp('01','RR');
    to_timestamp
---------------------
 2001-01-01 00:00:00
(1 row)
```

to_timestamp(text, text)

描述：将字符串类型的值转换为指定格式的时间戳。

返回值类型：timestamp

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT to_timestamp('05 Dec 2000', 'DD Mon YYYY');
   to_timestamp
---------------------
 2000-12-05 00:00:00
(1 row)
```

表21-40显示了可以用于格式化数值的模板模式，适用于函数to_number。

表 21-40 数值格式化的模板模式

模式	描述
9	带有指定数值位数的值
0	带前导零的值
.（句点）	小数点
,（逗号）	分组（千）分隔符
PR	尖括号内负值
S	带符号的数值（使用区域设置）
L	货币符号（使用区域设置）
D	小数点（使用区域设置）
G	分组分隔符（使用区域设置）
MI	在指明的位置的负号（如果数字 < 0）
PL	在指明的位置的正号（如果数字 > 0）
SG	在指明的位置的正/负号
RN	罗马数字（输入在 1 和 3999 之间）
TH或th	序数后缀
V	移动指定位（小数）

表21-41显示了可以用于格式化日期和时间值的模板，这些模式适用于函数to_date、to_timestamp、to_char和参数nls_timestamp_format。

表 21-41 用于日期/时间格式化的模式

类别	模式	描述
小时	HH	一天的小时数（01-12）
	HH12	一天的小时数（01-12）
	HH24	一天的小时数（00-23）
分钟	MI	分钟（00-59）

版权所有 © 华为云计算技术有限公司

类别	模式	描述
秒	SS	秒（00-59）
	FF	微秒（000000-999999）
	SSSSS	午夜后的秒（0-86399）
上、下午	AM或A.M.	上午标识
	PM或P.M.	下午标识
年	Y,YYY	带逗号的年（4和更多位）
	SYYYY	公元前四位年
	YYYY	年（4和更多位）
	YYY	年的后三位
	YY	年的后两位
	Y	年的最后一位
	IYYY	ISO年（4位或更多位）
	IYY	ISO年的最后三位
	IY	ISO年的最后两位
	I	ISO年的最后一位
	RR	年的后两位（可在21世纪存储20世纪的年份） 规则如下： ● 输入的两位年份在00~49之间： 当前年份的后两位在00~49之间，返回值年份的前两位和当前年份的前两位相同； 当前年份的后两位在50~99之间，返回值年份的前两位是当前年份的前两位加1。 ● 输入的两位年份在50~99之间： 当前年份的后两位在00~49之间，返回值年份的前两位是当前年份的前两位减1； 当前年份的后两位在50~99之间，返回值年份的前两位和当前年份的前两位相同。
	RRRR	可接收4位年或两位年。若是两位，则和RR的返回值相同，若是四位，则和YYYY相同。
	● BC或B.C. ● AD或A.D.	纪元标识。BC（公元前），AD（公元后）。
月	MONTH	全长大写月份名（空白填充为9字符）
	MON	大写缩写月份名（3字符）
	MM	月份数（01-12）

版权所有 © 华为云计算技术有限公司

类别	模式	描述
	RM	罗马数字的月份（I-XII； I=JAN）（大写）
天	DAY	全长大写日期名（空白填充为9字符）
	DY	缩写大写日期名（3字符）
	DDD	一年里的日（001-366）
	DD	一个月里的日（01-31）
	D	一周里的日（1-7； 周日是 1）
周	W	一个月里的周数（1-5）（第一周从该月第一天开始）
	WW	一年里的周数（1-53）（第一周从该年的第一天开始）
	IW	ISO一年里的周数（第一个星期四在第一周里）
世纪	CC	世纪（2位）（21 世纪从 2001-01-01 开始）
儒略日	J	儒略日（自公元前 4712 年 1 月 1 日来的天数）
季度	Q	季度

21.7.14 JSON/JSONB 函数和操作符

21.7.14.1 JSON/JSONB 操作符

表 21-42 json 和 jsonb 通用操作符

操作符	左操作数类型	右操作数类型	返回类型	描述	示例
->	Array-json(b)	int	json(b)	获得array-json元素。下标不存在返回空。	SELECT '[{"a":"foo"}, {"b":"bar"}, {"c":"baz"}]'::json->2; ?column? ---------- {"c":"baz"} (1 row)
->	object-json(b)	text	json(b)	通过键获得值。不存在则返回空。	SELECT '{"a":{"b":"foo"}}'::json->'a'; ?column? ---------- {"b":"foo"} (1 row)

版权所有 © 华为云计算技术有限公司

操作符	左操作数类型	右操作数类型	返回类型	描述	示例
->>	Array-json(b)	int	text	获得array-json元素。下标不存在返回空。	SELECT '[{"a":"foo"}, {"b":"bar"}, {"c":"baz"}]'::json->>2; ?column? ---------- {"c":"baz"} (1 row)
->>	object-json(b)	text	text	通过键获得值。不存在则返回空	SELECT '{"a":{"b":"foo"}}'::json->>'a'; ?column? ---------- {"b":"foo"} (1 row)
#>	container-json(b)	text[]	json	获取在指定路径的JSON对象，路径不存在则返回空。 **说明** DWS对象标识符支持以符号"#"结尾，为避免a#>b解析过程出现歧义，因此操作符"#>"前后需要增加空格，否则解析报错。	SELECT '{"a":{"b":{"c":1}}}'::json #> '{a, b}'; ?column? ---------- {"c":1} (1 row)
#>>	container-json(b)	text[]	text	获取在指定路径的JSON对象，路径不存在则返回空。	SELECT '{"a":{"b":{"c":1}}}'::json #>> '{a, b}'; ?column? ---------- {"c":1} (1 row)

表 21-43 jsonb 支持的操作符

操作符	右操作类型	返回类型	描述	示例
=	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_eq	SELECT '{"a":{"b":{"c":1}}}'::jsonb = '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- t (1 row)
<>	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_ne	SELECT '{"a":{"b":{"c":1}}}'::jsonb <> '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- f (1 row)

版权所有 © 华为云计算技术有限公司

操作符	右操作类型	返回类型	描述	示例
<	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_lt	SELECT '{"a":{"b":{"c":2}}}'::jsonb < '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- f (1 row)
>	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_gt。	SELECT '{"a":{"b":{"c":2}}}'::jsonb > '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- t (1 row)
<=	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_le。	SELECT '{"a":{"b":{"c":2}}}'::jsonb <= '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- f (1 row)
>=	jsonb	bool	判断两个jsonb的大小关系。同函数jsonb_ge。	SELECT '{"a":{"b":{"c":2}}}'::jsonb >= '{"a":{"b":{"c":1}}}'::jsonb; ?column? ---------- t (1 row)
?	text	bool	键/元素的字符串是否存在JSON值的顶层	SELECT '{"a":1, "b":2}'::jsonb ? 'b'; ?column? ---------- t (1 row)
?|	text[]	bool	这些数组字符串中的任何一个是否作为顶层键存在。	SELECT '{"a":1, "b":2, "c":3, "d":4}'::jsonb ?| '{a, b, e}'::text[]; ?column? ---------- t (1 row)
?&	text[]	bool	是否所有这些数组字符串都作为顶层键存在。	SELECT '{"a":1, "b":2, "c":3, "d":4}'::jsonb ?& '{a, b, c}'::text[]; ?column? ---------- t (1 row)
<@	jsonb	bool	左边的JSON的所有项是否全部存在于右边JSON的顶层。	SELECT '{"b":3}'::jsonb <@ '{"a":{"b":{"c":2}}, "b":3}'::jsonb; ?column? ---------- t (1 row)
@>	jsonb	bool	左边的JSON的顶层是否包含右边JSON的顶层所有项。	SELECT '{"a":{"b":{"c":2}}, "b":3}'::jsonb @> '{"b":3}'::jsonb; ?column? ---------- t (1 row)

版权所有 © 华为云计算技术有限公司

操作符	右操作类型	返回类型	描述	示例
||	jsonb	jsonb	两个jsonb对象合并成一个。	SELECT '{"a":1, "b":2}'::jsonb || '{"c":3, "d":4}'::jsonb; ?column? ---------- {"a": 1, "b": 2, "c": 3, "d": 4} (1 row)
-	text	jsonb	删除jsonb对象删除指定的键值对。	SELECT '{"a":1, "b":2}'::jsonb - 'a'; ?column? ---------- {"b": 2} (1 row)
-	text	jsonb	删除jsonb对象删除指定的键值对。	SELECT '{"a":1, "b":2, "c":3, "d":4}'::jsonb - '{a, b}'::text[]; ?column? ---------- {"c": 3, "d": 4} (1 row)
-	int	jsonb	删除jsonb数组中下标对应的元素。	SELECT '["a", "b", "c"]'::jsonb - 2; ?column? ---------- ["a", "b"] (1 row)
#-	text[]	jsonb	删除jsonb对象中路径对应的键值对。	SELECT '{"a":{"b":{"c":{"d":1}}}, "e":2, "f":3}'::jsonb #- '{a, b}'::text[]; ?column? ---------- {"a": {}, "e": 2, "f": 3} (1 row)

21.7.14.2 JSON/JSONB 函数

JSON/JSONB函数表示可以用于JSON类型（请参考**JSON类型**）数据的函数。

说明

- 除函数array_to_json和row_to_json外，其余有关JSON/JSONB函数和操作符仅8.1.2及以上集群版本支持。
- JSON/JSONB函数只提供基础功能，如有其他复杂或特殊性能要求场景，请联系华为工程师进行评估。

array_to_json(anyarray [, pretty_bool])

描述：返回JSON类型的数组。一个多维数组成为一个JSON数组的数组。如果pretty_bool设置为true，将在一维元素之间添加换行符。

返回类型：json

示例：

```
SELECT array_to_json('{{1,5},{99,100}}'::int[]);
array_to_json
------------------
[[1,5],[99,100]]
(1 row)
```


版权所有 © 华为云计算技术有限公司

row_to_json(record [, pretty_bool])

描述：返回JSON类型的行。如果pretty_bool设置为true，将在第一级元素之间添加换行符。

返回类型：json

示例：

```
SELECT row_to_json(row(1,'foo'));
    row_to_json
---------------------
 {"f1":1,"f2":"foo"}
(1 row)
```

json_agg(any)

描述：将值聚集为json数组。

返回类型：array-json

示例：

```
SELECT * FROM classes;
 name | score
 -----+-------
 A    |     2
 A    |     3
 D    |     5
 D    |
(4 rows)
SELECT name, json_agg(score) score FROM classes group by name order by name;
 name |      score
 -----+------------------
 A    | [2, 3]
 D    | [5, null]
      | [null]
(3 rows)
```

json_object_agg(any, any)

描述：将值聚集为json对象。

返回类型：json

示例：

```
SELECT * FROM classes;
 name | score
 -----+-------
 A    |     2
 A    |     3
 D    |     5
 D    |
(4 rows)

SELECT json_object_agg(name, score) FROM classes group by name order by name;
    json_object_agg
-------------------------
 { "A" : 2, "A" : 3 }
 { "D" : 5, "D" : null }
(2 rows)
```


版权所有 © 华为云计算技术有限公司

json_build_array(VARIADIC "any")

描述：从可变参数列表中构建一个可能是异构类型的JSON数组。

返回类型：json

示例：

```
SELECT json_build_array(1,2,'3',4,5);
 json_build_array
------------------
 [1, 2, "3", 4, 5]
(1 row)
```

json_build_object(VARIADIC "any")

描述：从可变参数列表中构建JSON对象。参数列表由交替的键和值组成。其入参必须为偶数个，两两一组组成键值对。注意键不可为null。

返回类型：json

示例：

```
SELECT json_build_object('foo',1,'bar',2);
   json_build_object
------------------------
 {"foo" : 1, "bar" : 2}
(1 row)
```

json_object(text[])、json_object(text[], text[])

描述：从文本数组中构建JSON对象。

这是个重载函数，当入参为一个文本数组的时候，其数组长度必须为偶数，成员被当做交替出现的键/值对。两个文本数组的时候，第一个数组被视为键，第二个被视为值，两个数组长度必须相等。键不可为null。

返回类型：json

示例：

```
SELECT json_object('{a, 1, b, "def", c, 3.5}');
              json_object
---------------------------------------
 {"a" : "1", "b" : "def", "c" : "3.5"}
(1 row)

SELECT json_object('{{a, 1},{b, "def"},{c, 3.5}}');
              json_object
---------------------------------------
 {"a" : "1", "b" : "def", "c" : "3.5"}
(1 row)

SELECT json_object('{a,b,"a b c"}', '{a,1,1}');
               json_object
 ---------------------------------------
 {"a" : "a", "b" : "1", "a b c" : "1"}
(1 row)
```

to_json(anyelement)

描述：把参数转换为json。

返回类型：json

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT to_json('Fred said "Hi."'::text);
       to_json
---------------------
 "Fred said \"Hi.\""
(1 row)
——将列存表json_tbl_2转换为json
postgres=# SELECT * FROM json_tbl_2;
  a |  b
 ---+-----
  1 | aaa
  1 | bbb
  2 | ccc
  2 | ddd
 (4 rows)
postgres=# SELECT to_json(t.*) FROM json_tbl_2 t;
      to_json
--------------------
 {"a":1,"b":"bbb"}
 {"a":2,"b":"ddd"}
 {"a":1,"b":"aaa"}
 {"a":2,"b":"ccc"}
(4 rows)
```

json_strip_nulls(json)

描述：所有具有空值的对象字段被忽略，其他值保持不变。

返回类型：json

示例：

```
SELECT json_strip_nulls('[{"f1":1,"f2":null},2,null,3]');
  json_strip_nulls
---------------------
 [{"f1":1},2,null,3]
(1 row)
```

json_object_field(json, text)

描述：同操作符->, 返回对象中指定键对应的值。

返回类型：json

示例：

```
SELECT json_object_field('{"a": {"b":"foo"}}','a');
 json_object_field
--------------------
 {"b":"foo"}
(1 row)
```

json_object_field_text(object-json, text)

描述：同操作符->>, 返回对象中指定键对应的值。

返回类型：text

示例：

```
SELECT json_object_field_text('{"a": {"b":"foo"}}','a');
 json_object_field_text
-------------------------
 {"b":"foo"}
(1 row)
```


版权所有 © 华为云计算技术有限公司

json_array_element(array-json, integer)

描述：同操作符->, 返回数组中指定下标的元素。

返回类型：json

示例：

```
SELECT json_array_element('[1,true,[1,[2,3]],null]',2);
 json_array_element
--------------------
 [1,[2,3]]
(1 row)
```

json_array_element_text(array-json, integer)

描述：同操作符->>, 返回数组中指定下标的元素。

返回类型：text

示例：

```
SELECT json_array_element_text('[1,true,[1,[2,3]],null]',2);
 json_array_element_text
-------------------------
 [1,[2,3]]
(1 row)
```

json_extract_path(json, VARIADIC text[])

描述：同操作符#>，返回$2所指路径的JSON值。

返回类型：json

示例：

```
SELECT json_extract_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');
 json_extract_path
--------------------
 "stringy"
(1 row)
```

json_extract_path_text(json, VARIADIC text[])

描述：同操作符#>>，返回$2所指路径的text值。

返回类型：text

示例：

```
SELECT json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');
 json_extract_path_text
------------------------
 stringy
(1 row)
```

json_array_elements(array-json)

描述：拆分数组，每一个元素返回一行。

返回类型：json

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT json_array_elements('[1,true,[1,[2,3]],null]');
 json_array_elements
---------------------
 1
 true
 [1,[2,3]]
 null
(4 rows)
```

json_array_elements_text(array-json)

描述：拆分数组，每一个元素返回一行。

返回类型：text

示例：

```
SELECT * FROM json_array_elements_text('[1,true,[1,[2,3]],null]');
   value
-----------
 1
 true
 [1,[2,3]]

(4 rows)
```

json_array_length(array-json)

描述：返回数组长度。

返回类型：integer

示例：

```
SELECT json_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4,null]');
 json_array_length
--------------------
         6
(1 row)
```

json_object_keys(object-json)

描述：返回对象中顶层的所有键。

返回类型：text

示例：

```
SELECT json_object_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}, "f1":"abcd"}');
 json_object_keys
------------------
 f1
 f2
 f1
(3 rows)
```

json_each(object-json)

描述：将对象的每个键值对拆分转换成一行两列。

返回类型：setof(key text, value json)

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM json_each('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');
 key |  value
-----+----------
 f1  | [1,2,3]
 f2  | {"f3":1}
 f4  | null
(3 rows)
```

json_each_text(object-json)

描述：将对象的每个键值对拆分转换成一行两列。

返回类型：setof(key text, value text)

示例：

```
SELECT * FROM json_each_text('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');
 key |  value
-----+----------
 f1  | [1,2,3]
 f2  | {"f3":1}
 f4  |
(3 rows)
```

json_populate_record(anyelement, object-json [, bool])

描述：$1必须是一个复合类型的参数。将会把object-json里的每个对键值进行拆分，以键当做列名，与$1中的列名进行匹配查找，并填充到$1的格式中。

说明

JSON/JSONB函数中入参为复合类型时，可以使用CREATE TYPE或CREATE TABLE定义复合类型，例如：

```
CREATE TYPE jpop AS (a text, b INT, c timestamp);
CREATE TABLE jpop2(a text, b INT, c timestamp);
```

返回类型：anyelement

示例：

```
CREATE TYPE jpop AS (a text, b INT, c timestamp);
SELECT * FROM json_populate_record(null::jpop,'{"a":"blurfl","x":43.2}');
   a    | b | c
--------+---+---
 blurfl |   |
(1 row)
```

json_populate_recordset(anyelement, array-json [, bool])

描述：参考函数json_populate_record、jsonb_populate_record，对$2数组的每一个元素进行上述参数函数的操作，因此这也要求$2数组的每个元素都是object-json类型。

返回类型：setof anyelement

示例：

```
CREATE TYPE jpop AS (a text, b INT, c timestamp);
SELECT * FROM json_populate_recordset(null::jpop, '[{"a":1,"b":2},{"a":3,"b":4}]');
 a | b | c
---+---+---
 1 | 2 |
 3 | 4 |
(2 rows)
```


版权所有 © 华为云计算技术有限公司

json_to_record(object-json)

描述：正如所有返回record的函数一样，调用者必须用一个AS子句显式地定义记录的结构。会将object-json的键值对进行拆分重组，把键当做列名，去匹配填充AS显示指定的记录的结构。

返回类型：record

示例：

```
SELECT * FROM json_to_record('{"a":1,"b":"foo","c":"bar"}'::json) as x(a int, b text, d text);
 a |  b  | d
---+-----+---
 1 | foo |
(1 row)
```

json_to_recordset(array-json)

描述：参考函数json_to_record，对数组内的每个元素，执行上述函数的操作，因此这要求数组内的每个元素都得是object-json。

返回类型：setof record

示例：

```
SELECT * FROM json_to_recordset('[{"a":1,"b":{"d":"foo"},"c":true},{"a":2,"c":false,"b":{"d":"bar"}}]') AS x(a
INT, b json, c BOOLEAN);
 a |      b      | c
---+-------------+---
 1 | {"d":"foo"} | t
 2 | {"d":"bar"} | f
(2 rows)

SELECT * FROM json_to_recordset('[{"a":1,"b":"foo","d":false},{"a":2,"b":"bar","c":true}]') AS x(a INT, b text, c
BOOLEAN);
 a |  b  | c
---+-----+---
 1 | foo |
 2 | bar | t
(2 rows)
```

json_typeof(json)

描述：检测json类型。

返回类型：text

示例：

```
SELECT value, json_typeof(value) from (values (json '123.4'), (json '"foo"'), (json 'true'), (json 'null'), (json
'[1, 2, 3]'), (json '{"x":"foo", "y":123}'), (NULL::json)) as data(value);
        value          | json_typeof
-----------------------+-------------
 123.4                 | number
 "foo"                 | string
 true                  | boolean
 null                  | null
 [1, 2, 3]             | array
 {"x":"foo", "y":123} | object
                       |
(7 rows)
```


版权所有 © 华为云计算技术有限公司

jsonb_object(text[])

描述：从一个文本数组构造一个object-jsonb。这是个重载函数，当入参为一个文本数组的时候，其数组长度必须为偶数，成员被当做交替出现的键/值对。

返回类型：jsonb

示例：

```
SELECT jsonb_object('{a,1,b,2,3,NULL,"d e f","a b c"}');
                    jsonb_object
-----------------------------------------------------
 {"3": null, "a": "1", "b": "2", "d e f": "a b c"}
(1 row)
```

jsonb_object(text[], text[])

描述：两个文本数组的时候，第一个数组认为是键，第二个认为是值，两个数组长度必须相等。键不可为null。

返回类型：jsonb

示例：

```
SELECT jsonb_object('{a,b,"a b c"}', '{a,1,1}');
            jsonb_object
-------------------------------------
 {"a": "a", "b": "1", "a b c": "1"}
(1 row)
```

to_jsonb(anyment)

描述：将其他类型转换成对应的jsonb类型。

返回类型：jsonb

示例：

```
SELECT to_jsonb(1.1);
 to_jsonb
----------
 1.1
(1 row)
```

jsonb_agg

描述：将jsonb对象聚合成jsonb数组。

返回类型：jsonb

示例：

```
SELECT * FROM json_tbl_2;
 a |  b
---+-----
 1 | aaa
 1 | bbb
 2 | ccc
 2 | ddd
(4 rows)

SELECT a, jsonb_agg(b) FROM json_tbl_2 GROUP BY a ORDER BY a;
 a |   jsonb_agg
---+----------------
```

 版权所有 © 华为云计算技术有限公司

```
 1 | ["aaa", "bbb"]
 2 | ["ccc", "ddd"]
(2 rows)
```

jsonb_object_agg

描述：将键/值对聚集成一个JSON对象。

返回类型：jsonb

示例：

```
SELECT * FROM json_tbl_3;
 a |  b  | c
---+-----+----
 1 | aaa | 10
 1 | bbb | 20
 2 | ccc | 30
 2 | ddd | 40
(4 rows)
SELECT a, jsonb_object_agg(b, c) FROM json_tbl_3 GROUP BY a ORDER BY a;
 a |     jsonb_object_agg
---+--------------------------
 1 | {"aaa": 10, "bbb": 20}
 2 | {"ccc": 30, "ddd": 40}
(2 rows)
```

jsonb_build_array([VARIADIC "any"])

描述：从一个可变参数列表构造一个可能包含异质类型的JSON数组。

返回类型：jsonb

示例：

```
SELECT jsonb_build_array('a',1,'b',1.2,'c',true,'d',null,'e',json '{"x": 3, "y": [1,2,3]}','');
                                jsonb_build_array
-------------------------------------------------------------------------------------
 ["a", 1, "b", 1.2, "c", true, "d", null, "e", {"x": 3, "y": [1, 2, 3]}, null]
(1 row)
```

jsonb_build_object([VARIADIC “any”])

描述：从一个可变参数列表构造出一个JSON对象，其入参必须为偶数个，两两一组组成键值对。注意键不可为null。

返回类型：jsonb

示例：

```
SELECT jsonb_build_object(1,2);
 jsonb_build_object
--------------------
 {"1": 2}
(1 row)
```

jsonb_strip_nulls(jsonb)

描述：所有具有空值的对象字段均被省略。其他空值保持不变。

返回类型：jsonb

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT jsonb_strip_nulls('[{"f1":1,"f2":null},2,null,3]');
    jsonb_strip_nulls
-------------------------
 [{"f1": 1}, 2, null, 3]
(1 row)
```

jsonb_object_field(jsonb, text)

描述：同操作符->, 返回对象中指定键对应的值。

返回类型：jsonb

示例：

```
SELECT jsonb_object_field('{"a": {"b":"foo"}}','a');
 jsonb_object_field
--------------------
 {"b": "foo"}
(1 row)
```

jsonb_object_field_text(jsonb, text)

描述：同操作符->>, 返回对象中指定键对应的值。

返回类型：text

示例：

```
SELECT jsonb_object_field_text('{"a": {"b":"foo"}}','a');
 jsonb_object_field_text
-------------------------
 {"b": "foo"}
(1 row)
```

jsonb_array_element(array-jsonb, integer)

描述：同操作符->, 返回数组中指定下标的元素。

返回类型：jsonb

示例：

```
SELECT jsonb_array_element('[1,true,[1,[2,3]],null]',2);
 jsonb_array_element
---------------------
 [1, [2, 3]]
(1 row)
```

jsonb_array_element_text(array-jsonb, integer)

描述：同操作符->>, 返回数组中指定下标的元素。

返回类型：text

示例：

```
SELECT jsonb_array_element_text('[1,true,[1,[2,3]],null]',2);
 jsonb_array_element_text
--------------------------
 [1, [2, 3]]
(1 row)
```


版权所有 © 华为云计算技术有限公司

jsonb_extract_path((jsonb, VARIADIC text[])

描述：等价于操作符#>，返回$2所指路径的值。

返回类型：jsonb

示例：

```
SELECT jsonb_extract_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');
 jsonb_extract_path
--------------------
 "stringy"
(1 row)
```

jsonb_extract_path_text((jsonb, VARIADIC text[])

描述：等价于操作符#>>，返回$2所指路径的值。

返回类型：text

示例：

```
SELECT jsonb_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');
 jsonb_extract_path_text
-------------------------
 stringy
(1 row)
```

jsonb_extract((jsonb, VARIADIC text[])

描述：输入任意的object-jsonb类型、array-jsonb类型，返回$2所指路径的值。该函数仅9.1.0及以上集群版本支持。

返回类型：SETOF jsonb

示例：

```
SELECT jsonb_extract('{"f2":{"f3":1},"f4":[{"f5":99},{"f6":"stringy"}]}','f2','f3');
 jsonb_extract
---------------
 1
(1 row)

SELECT jsonb_extract('{"f2":{"f3":1},"f4":[{"f5":99},{"f5":"stringy"}]}', 'f4','f5');
 jsonb_extract
---------------
 99
 "stringy"
(2 rows)
```

jsonb_extract_text((jsonb, VARIADIC text[])

描述：输入任意的object-jsonb类型、array-jsonb类型，返回$2所指路径的值。该函数仅9.1.0及以上集群版本支持。

返回类型：SETOF text

示例：

```
SELECT jsonb_extract_text('{"f2":{"f3":1},"f4":[{"f5":99},{"f6":"stringy"}]}','f2','f3');
 jsonb_extract_text
--------------------
 1
(1 row)
```


版权所有 © 华为云计算技术有限公司

```
SELECT jsonb_extract_text('{"f2":{"f3":1},"f4":[{"f5":99},{"f5":"stringy"}]}', 'f4','f5');
 jsonb_extract_text
--------------------
 99
 stringy
(2 rows)
```

jsonb_array_elements(array-jsonb)

描述：拆分数组，每一个元素返回一行。

返回类型：jsonb

示例：

```
SELECT jsonb_array_elements('[1,true,[1,[2,3]],null]');
 jsonb_array_elements
----------------------
 1
 true
 [1, [2, 3]]
 null
(4 rows)
```

jsonb_array_elements_text(array-jsonb)

描述：拆分数组，每一个元素返回一行。

返回类型：text

示例：

```
SELECT * FROM jsonb_array_elements_text('[1,true,[1,[2,3]],null]');
    value
-------------
 1
 true
 [1, [2, 3]]

(4 rows)
```

jsonb_array_length(array-jsonb)

描述：返回数组长度。

返回类型：integer

示例：

```
SELECT jsonb_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4,null]');
 jsonb_array_length
--------------------
          6
(1 row)
```

jsonb_object_keys(object-jsonb)

描述：返回对象中顶层的所有键。

返回类型：SETOF text

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT jsonb_object_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}, "f1":"abcd"}');
 jsonb_object_keys
-------------------
 f1
 f2
(2 rows)
```

jsonb_each(object-jsonb)

描述：将对象的每个键值对拆分转换成一行两列。

返回类型：setof(key text, value jsonb)

示例：

```
SELECT * FROM jsonb_each('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');
 key |   value
-----+-----------
 f1  | [1, 2, 3]
 f2  | {"f3": 1}
 f4  | null
(3 rows)
```

jsonb_each_text(object-jsonb)

描述：将对象的每个键值对拆分转换成一行两列。

返回类型：setof(key text, value text)

示例：

```
SELECT * FROM jsonb_each_text('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');
 key |   value
-----+-----------
 f1  | [1, 2, 3]
 f2  | {"f3": 1}
 f4  |
(3 rows)
```

jsonb_populate_record(anyelement, object-jsonb [, bool])

描述：$1必须是一个复合类型的参数。将会把object-json里的每个对键值进行拆分，以键当做列名，与$1中的列名进行匹配查找，并填充到$1的格式中。

返回类型：anyelement

示例：

```
SELECT * FROM jsonb_populate_record(null::jpop,'{"a":"blurfl","x":43.2}');
   a    | b | c
--------+---+---
 blurfl |   |
(1 row)
```

jsonb_populate_recordset(anyelement, array-jsonb [, bool])

描述：参考上述函数json_populate_record、jsonb_populate_record，对$2数组的每一个元素进行上述参数函数的操作，因此这也要求$2数组的每个元素都是object-json类型。

返回类型：setof anyelement

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM jsonb_populate_recordset(null::jpop, '[{"a":1,"b":2},{"a":3,"b":4}]');
 a | b | c
---+---+---
 1 | 2 |
 3 | 4 |
(2 rows)
```

jsonb_to_record(object-json)

描述：正如所有返回record的函数一样，调用者必须用一个AS子句显式地定义记录的结构。会将object-json的键值对进行拆分重组，把键当做列名，去匹配填充AS显示指定的记录的结构。

返回类型：record

示例：

```
SELECT * FROM jsonb_to_record('{"a":1,"b":"foo","c":"bar"}'::jsonb) as x(a int, b text, d text);
 a |  b  | d
---+-----+---
 1 | foo |
(1 row)
```

jsonb_to_recordset(array-json)

描述：参考函数jsonb_to_record，对数组内个每个元素，执行上述函数的操作，因此这要求数组内的每个元素都得是object-jsonb。

返回类型：setof record

示例：

```
SELECT * FROM jsonb_to_recordset('[{"a":1,"b":"foo","d":false},{"a":2,"b":"bar","c":true}]') AS x(a INT, b text,
c boolean);
 a |  b  | c
---+-----+---
 1 | foo |
 2 | bar | t
(2 rows)
```

jsonb_typeof(jsonb)

描述：检测jsonb类型。

返回类型：text

示例：

```
SELECT jsonb_typeof(to_jsonb(1.1));
 jsonb_typeof
--------------
 number
(1 row)
```

jsonb_ne(jsonb, jsonb)

描述：同操作符<>，比较两个值的大小。

返回类型：bool

示例：

```
SELECT jsonb_ne('{"a":1, "b":2}'::jsonb, '{"a":1, "b":3}'::jsonb);
 jsonb_ne
```

 版权所有 © 华为云计算技术有限公司

```
----------
 t
(1 row)
```

jsonb_lt(jsonb, jsonb)

描述：同操作符<，比较两个值的大小。

返回类型：bool

示例：

```
SELECT jsonb_lt('{"a":1, "b":2}'::jsonb, '{"a":1, "b":3}'::jsonb);
 jsonb_lt
----------
 t
(1 row)
```

jsonb_gt(jsonb, jsonb)

描述：同操作符>，比较两个值的大小。

返回类型：bool

示例：

```
SELECT jsonb_gt('{"a":1, "b":2}'::jsonb, '{"a":1, "b":3}'::jsonb);
 jsonb_gt
----------
 f
(1 row)
```

jsonb_le(jsonb, jsonb)

描述：同操作符<=，比较两个值的大小。

返回类型：bool

示例：

```
SELECT jsonb_le('["a", "b"]', '{"a":1, "b":2}');
 jsonb_le
----------
 t
(1 row)
```

jsonb_ge(jsonb, jsonb)

描述：同操作符>=，比较两个值的大小。

返回类型：bool

示例：

```
SELECT jsonb_ge('["a", "b"]', '{"a":1, "b":2}');
 jsonb_ge
----------
 f
(1 row)
```

jsonb_eq(jsonb, jsonb)

描述：同操作符=，比较两个值的大小

版权所有 © 华为云计算技术有限公司

返回类型：bool

示例：

```
SELECT jsonb_eq('["a", "b"]', '{"a":1, "b":2}');
 jsonb_eq
----------
 f
(1 row)
```

jsonb_cmp(jsonb, jsonb)

描述：比较大小，正数表示大于，负数表示小于，0表示相等。

返回类型：integer

示例：

```
SELECT jsonb_cmp('["a", "b"]', '{"a":1, "b":2}');
 jsonb_cmp
-----------
 -1
(1 row)
```

jsonb_exists(jsonb, text)

描述：同操作符?，字符串$2是否存在$1的顶层以key\elem\scalar的形式存在。

返回类型：bool

示例：

```
SELECT jsonb_exists('["1",2,3]', '1');
 jsonb_exists
--------------
 t
(1 row)
```

jsonb_exists_any(jsonb, text[])

描述：同操作符?|，字符串数组$2里面是否存在的元素，在$1的顶层以key\elem\scalar的形式存在。

返回类型：

示例：

```
SELECT jsonb_exists_any('["1","2",3]', '{1, 2, 4}');
 jsonb_exists_any
------------------
 t
(1 row)
```

jsonb_exists_all(jsonb, text[])

描述：同操作符?&，字符串数组$2里面是否所有的元素，都在$1的顶层以key\elem\scalar的形式存在。

返回类型：

bool

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT jsonb_exists_all('["1","2",3]', '{1, 2}');
 jsonb_exists_all
------------------
 t
(1 row)
```

jsonb_contained(jsonb, jsonb)

描述：同操作符<@, 判断$1中的所有元素是否在$2的顶层存在。

返回类型：bool

示例：

```
SELECT jsonb_contained('[1,2,3]', '[1,2,3,4]');
 jsonb_contained
-----------------
 t
(1 row)
```

jsonb_contains(jsonb, jsonb)

描述：同操作符@>, 判断$1中的顶层所有元素是否包含在$2的所有元素。

返回类型：bool

示例：

```
SELECT jsonb_contains('{"a":1, "b":2, "c":3}'::jsonb, '{"a":1}');
 jsonb_contains
-----------------
 t
(1 row)
```

jsonb_concat(jsonb, jsonb)

描述：连接两个jsonb对象为一个jsonb。

返回类型：jsonb

示例：

```
SELECT jsonb_concat('{"a":1, "b":2}'::jsonb, '{"c":3, "d":4}'::jsonb);
          jsonb_concat
---------------------------------
 {"a": 1, "b": 2, "c": 3, "d": 4}
(1 row)
```

jsonb_delete(jsonb, text)

描述：删除jsonb中的key值对应的键值对。

返回类型：jsonb

示例：

```
SELECT jsonb_delete('{"a":1, "b":2}'::jsonb, 'a');
 jsonb_delete
--------------
 {"b": 2}
(1 row)
```


版权所有 © 华为云计算技术有限公司

jsonb_delete_idx(jsonb, text)

描述：删除数组下标对应的元素。

返回类型：jsonb

示例：

```
SELECT jsonb_delete_idx('[0,1,2,3,4]'::jsonb, 2);
 jsonb_delete_idx
------------------
 [0, 1, 3, 4]
(1 row)
```

jsonb_delete_array(jsonb, VARIADIC text[])

描述：删除jsonb数组中的多个元素。

返回类型：jsonb

示例：

```
SELECT jsonb_delete_array('["a", "b", "c"]'::jsonb , 'a', 'b');
 jsonb_delete_array
--------------------
 ["c"]
(1 row)
```

jsonb_delete_path(jsonb, text[])

描述：删除jsonb数组中指定路径的元素。

返回类型：jsonb

示例：

```
SELECT jsonb_delete_path('{"a":{"b":{"c":1, "d":2}}, "e":3}'::jsonb , array['a', 'b']);
 jsonb_delete_path
-------------------
 {"a": {}, "e": 3}
(1 row)
```

jsonb_set(target jsonb, path text[], new_value jsonb [, create_missing boolean])

描述：返回target，用path指定的部分被new_value替换，或者如果create_missing为true（默认值为true）且path指定的项不存在，则添加new_value。与面向路径的运算符一样，path中出现的负整数从JSON数组的末尾开始计数。

返回类型：jsonb

示例：

```
SELECT jsonb_set('[{"f1":1,"f2":null},2,null,3]', '{0,f1}','[2,3,4]', false);
                jsonb_set
----------------------------------------------
 [{"f1": [2, 3, 4], "f2": null}, 2, null, 3]
(1 row)
```

jsonb_pretty(jsonb)

描述：以缩进的JSON文本形式返回。

 版权所有 © 华为云计算技术有限公司

返回类型：jsonb

示例：

```
SELECT jsonb_pretty('{"a":{"b":{"c":1, "d":2}}, "e":3}'::jsonb);
   jsonb_pretty
---------------------
{                  +
    "a": {         +
        "b": {     +
            "c": 1,+
            "d": 2 +
        }          +
    },             +
    "e": 3         +
}
(1 row)
```

jsonb_insert(target jsonb, path text[], new_value jsonb [, insert_after boolean])

描述：返回被插入了new_value的target。如果path指定的target部分位于JSONB数组中，则new_value将被插入到目标之前（insert_after为false，默认值）或者之后（insert_after为true）。如果path指定的target部分位于一个JSONB对象内，则仅当target不存在时才插入new_value。与面向路径的运算符一样，path中出现的负整数表示从JSON数组的末尾开始计数。

返回类型：jsonb

示例：

```
SELECT jsonb_insert('{"a": [0,1,2]}', '{a, 1}', '"new_value"');
        jsonb_insert
--------------------------------
 {"a": [0, "new_value", 1, 2]}
(1 row)
```

ts_headline([config regconfig,] document jsonb, query tsquery [, options text])

描述：高亮jsonb搜索结果。

返回类型：jsonb

示例：

```
SELECT ts_headline('english',
'[{"id":9928,"user_id":4562,"user_name":"9LOHR4","create_time":"2021-06-22T16:28:16.504518+08:00"},
{"id":9959,"user_id":5524,"user_name":"YID07D","create_time":"2021-06-22T16:28:16.557228+08:00"},
{"id":9962,"user_id":7991,"user_name":"7C6QOM","create_time":"2021-06-22T16:28:16.56234+08:00"}]'::jsonb,
 to_tsquery('english', '9LOHR4'), 'StartSel = <, StopSel = >');

ts_headline

--------------------------------
```

json_to_tsvector(config regconfig,] json, jsonb)

描述：将json格式转换为用于支持全文检索的文件格式tsvector。

返回类型：jsonb

示例：

```
SELECT json_to_tsvector('{"a":1, "b":2, "c":3}'::json, to_jsonb('key'::text));
 json_to_tsvector
------------------
 'b':2 'c':4
(1 row)
```

21.7.15 安全函数

gs_password_deadline()

描述：显示当前账户距离密码过期的时间。密码过期后提示用户修改密码。与GUC参数password_effect_time相关。

返回值类型：interval

示例：

```
SELECT gs_password_deadline();
  gs_password_deadline
--------------------------
 83 days 17:44:32.196094
(1 row)
```

gs_password_expiration()

描述：显示当前账户距离密码过期的时间。密码过期后用户无法登录数据库。与创建用户的DDL语句PASSWORD EXPIRATION period相关，函数返回值大于等于-1，如果创建用户时未指定PASSWORD EXPIRATION period，该函数的缺省值为-1，表示没有过期限制。

返回值类型：interval

示例：

```
SELECT gs_password_expiration();
  gs_password_expiration
--------------------------
 29 days 23:59:49.731482
(1 row)
```

login_audit_messages(flag boolean)

描述：查看登录用户的登录信息。

返回值类型：元组

示例：

- 查看上一次登录认证通过的日期、时间和IP等信息：

```
SELECT * FROM login_audit_messages(true);
  username  | database |       logintime        |     type      | result |  client_conninfo   |
session_id
------------+----------+------------------------+---------------+--------+--------------------
+----------------------------------------
    dbadmin    | gaussdb | 2017-06-02 15:28:34+08 | login_success | ok     | gsql@[local]    |
```


版权所有 © 华为云计算技术有限公司

```
1663745305.140192538154752.coordinator1
(1 row)
```

- 查看上一次登录认证失败的日期、时间和IP等信息：

```
SELECT * FROM login_audit_messages(false) ORDER BY logintime desc limit 1;
 username  | database |       logintime        |     type     | result |     client_conninfo     | session_id
-----------+----------+------------------------+--------------+--------+-------------------------+------------
(0 rows)
```

- 查看自从最后一次认证通过以来失败的尝试次数、日期和时间：

```
SELECT * FROM login_audit_messages(false);
 username  | database |       logintime        |     type     | result |     client_conninfo     | session_id
-----------+----------+------------------------+--------------+--------+-------------------------+------------
(0 rows)
```

login_audit_messages_pid(flag boolean)

描述：查看登录用户的登录信息。与login_audit_messages的区别在于结果基于当前backendid向前查找。所以不会因为同一用户的后续登录，而影响本次登录的查询结果。也就是查询不到该用户后续登录的信息。

返回值类型：元组

示例：

- 查看上一次登录认证通过的日期、时间和IP等信息：

```
SELECT * FROM login_audit_messages_pid(true);
 username  | database |       logintime        |     type      | result |  client_conninfo | backendid
|              session_id
-----------+----------+------------------------+---------------+--------+-------------------
+-----------------------------------------
    dbadmin    | postgres | 2017-06-02 15:28:34+08 | login_success | ok     | gsql@[local] |
140311900702464 | 1663745305.140192538154752.coordinator1
(1 row)
```

- 查看上一次登录认证失败的日期、时间和IP等信息：

```
SELECT * FROM login_audit_messages_pid(false) ORDER BY logintime desc limit 1;
 username  | database |       logintime        |     type     | result |     client_conninfo   | backendid |
session_id
-----------+----------+------------------------+--------------+--------+-------------------------+------------
(0 rows)
```

- 查看自从最后一次认证通过以来失败的尝试次数、日期和时间：

```
SELECT * FROM login_audit_messages_pid(false);
 username  | database |       logintime        |     type     | result |     client_conninfo    | backendid |
session_id
-----------+----------+------------------------+--------------+--------+-------------------------+------------
(0 rows)
```

pg_query_audit()

描述：查看当前CN节点审计日志。

返回值类型：record

函数返回字段如下：

表 21-44 pg_query_audit()函数返回字段

名称	类型	描述
begintime	timestamp with time zone	操作的执行开始时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
endtime	timestamp with time zone	操作的执行结束时间。
operation_type	text	操作类型，具体类型见表21-45。
audit_type	text	审计类型，具体类型见表21-46。
result	text	操作结果。
username	text	执行操作的用户名。
database	text	数据库名称。
client_conninfo	text	客户端连接信息，即gsql，jdbc或odbc。
object_name	text	操作对象名称。
object_details	text	记录语句中涉及的列及其使用类型。
command_text	text	操作的执行命令。
detail_info	text	执行操作详细信息。
transaction_xid	text	事务ID。
query_id	text	查询ID。
node_name	text	节点名称。
session_id	text	会话ID。
local_port	text	本地端口。
remote_port	text	远端端口。
result_rows	text	语句执行返回的记录数。
error_code	text	语句执行错误时的错误码。

表 21-45 operation_type 操作类型项

操作类型	描述
audit_switch	表示对用户打开和关闭审计日志操作场景进行审计。
login_logout	表示对用户登录和登出操作场景进行审计。
system	表示对系统的启停、实例切换操作场景进行审计。
sql_parse	表示对SQL语句解析场景进行审计。
user_lock	表示对用户锁定和解锁操作的场景进行审计。
grant_revoke	表示对用户权限授予和回收操作场景进行审计。

版权所有 © 华为云计算技术有限公司

操作类型	描述
violation	表示对用户访问存在越权的场景进行审计。
ddl	表示对DDL操作场景进行审计，因为DDL操作会根据操作对象进行更细粒度控制，仍然沿用审计开关audit_system_object，即由audit_system_object控制对哪些对象的DDL操作进行审计（此处不配置ddl，只要配置了audit_system_object，审计也会生效）。
dml	表示对DML操作场景进行审计。
select	表示对SELECT操作场景进行审计。
internal_event	表示对内部事件操作场景进行审计。
user_func	表示对用户自定义函数、存储过程、匿名块操作场景进行审计。 **说明** 如果自定义函数、存储过程中有fetch语句，则审计fetch语句时，其中common_text字段记录的为其对应的CURSOR内容。
special_func	表示对特殊函数调用操作场景进行审计，特殊函数包括：pg_terminate_backend和pg_cancel_backend。
copy	表示对COPY操作场景进行审计。
set	表示对SET操作场景进行审计。
transaction	表示对事务操作场景进行审计。
vacuum	表示对VACUUM操作场景进行审计。
analyze	表示对ANALYZE操作场景进行审计。
cursor	表示对游标操作的场景进行审计。
anonymous_block	表示对匿名块操作场景进行审计。
explain	表示对EXPLAIN操作场景进行审计。
show	表示对SHOW操作场景进行审计。
lock_table	表示对锁表操作场景进行审计。
comment	表示对COMMENT操作场景进行审计。
preparestmt	表示对PREPARE、EXECUTE、DEALLOCATE操作场景进行审计。
cluster	表示对CLUSTER操作场景进行审计。
constraints	表示对CONSTRAINTS操作场景进行审计。
checkpoint	表示对CHECKPOINT操作场景进行审计。
barrier	表示对BARRIER操作场景进行审计。
cleanconn	表示对CLEAN CONNECTION操作场景进行审计。

版权所有 © 华为云计算技术有限公司

操作类型	描述
seclabel	表示对安全标签操作进行审计。
notify	表示对通知操作进行审计。
load	表示对加载操作进行审计。
discard	表示对清理当前会话所有的全局临时表信息场景进行审计。

表 21-46 audit_type 审计类型项

审计类型	描述
audit_open/audit_close	表示审计类型为打开和关闭审计日志操作。
user_login/user_logout	表示审计类型为用户登录/退出成功的操作和用户。
system_start/system_stop/ system_recover/system_switch	表示审计类型为系统的启停、实例切换操作。
sql_wait/sql_parse	表示审计类型为SQL语句解析。
lock_user/unlock_user	表示审计类型为用户锁定和解锁成功的操作。
grant_role/revoke__role	表示审计类型为用户权限授予和回收的操作。
user_violation	表示审计类型为用户访问存在越权的操作。
ddl_数据库对象	表示审计类型为DDL操作，因为DDL操作由会根据操作对象进行更细粒度控制，仍然沿用审计开关audit_system_object，即由audit_system_object控制对哪些对象的DDL操作进行审计（此处不配置ddl，只要配置了audit_system_object，审计也会生效）。 例如：ddl_sequence表示审计类型为序列相关操作。
dml_action_insert/ dml_action_delete/ dml_action_update/ dml_action_merge/ dml_action_select	表示审计类型为INSERT、DELETE、UPDATE、MERGE等DML操作。
internal_event	表示审计类型为内部事件。
user_func	表示审计类型为用户自定义函数、存储过程、匿名块操作。 **说明** 如果自定义函数、存储过程中有fetch语句，则审计fetch语句时，其中common_text字段记录的为其对应的CURSOR内容。

版权所有 © 华为云计算技术有限公司

审计类型	描述
special_func	表示审计类型为特殊函数调用操作，特殊函数包括：pg_terminate_backend和pg_cancel_backend。
copy_to/copy_from	表示审计类型为COPY相关操作。
set_parameter	表示审计类型为SET操作。
trans_begin/trans_commit/ trans_prepare/ trans_rollback_to/ trans_release/trans_savepoint/ trans_commit_prepare/ trans_rollback_prepare/ trans_rollback	表示审计类型为事务相关操作。
vacuum/vacuum_full/ vacuum_merge	表示审计类型为VACUUM相关操作。
analyze/analyze_verify	表示审计类型为ANALYZE相关操作。
cursor_declare/cursor_move/ cursor_fetch/cursor_close	表示审计类型为游标相关操作。
codeblock_execute	表示审计类型为匿名块。
explain	表示审计类型为EXPLAIN操作。
show	表示审计类型为SHOW操作。
lock_table	表示审计类型为锁表操作。
comment	表示审计类型为COMMENT操作。
prepare/execute/deallocate	表示审计类型为PREPARE、EXECUTE或DEALLOCATE操作。
cluster	表示审计类型为CLUSTER操作。
constraints	表示审计类型为CONSTRAINTS操作。
checkpoint	表示审计类型为CHECKPOINT操作。
barrier	表示审计类型为BARRIER操作。
cleanconn	表示审计类型为CLEAN CONNECTION操作。
seclabel	表示审计类型为安全标签操作。
notify	表示审计类型为通知操作。
load	表示审计类型为加载操作。
discard	表示审计类型为DISCARD操作。

 版权所有 © 华为云计算技术有限公司

pgxc_query_audit()

描述：查看所有CN节点审计日志。

返回值类型：record

函数返回字段同pg_query_audit函数。

pg_query_audit_details()

描述：查看审计日志并将审计日志中的object_name和object_details字段由json格式解析出来。该函数仅8.2.1.100及以上集群版本支持。

返回值类型：record

函数返回字段如下：

表 21-47 pg_query_audit()函数返回字段

名称	类型	描述
begintime	timestamp with time zone	操作的执行开始时间。
endtime	timestamp with time zone	操作的执行结束时间。
operation_type	text	操作类型，具体类型见**表21-45**。
audit_type	text	审计类型，具体类型见**表21-46**。
result	text	操作结果。
username	text	执行操作的用户名。
database	text	数据库名称。
client_conninfo	text	客户端连接信息，即gsql，jdbc或odbc。
transaction_xid	text	事务ID。
query_id	text	查询ID。
node_name	text	节点名称。
session_id	text	会话ID。
local_port	text	本地端口。
remote_port	text	远端端口。
object_name	text	表名、函数名、视图名。
column_name	text	列名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
type_of_use	text	对象的使用类型： 1：仅涉及（在实际使用中，此标志位暂不存在） 2：执行过程中访问（语句中出现的列以及在解析过程中访问到的列） 4：条件中发现（条件类型语句中，不包括条件表达式及函数） 8：inner join中发现 16：outer join中发现 32：聚合节点中发现（包括distinct、group by、聚集函数） 64：full join中发现 该列数值为叠加显示的数值。
use_type	text	type_of_use解析出的具体类型： 1：Reference only 2：Access 4：Conditional 8：Inner join 16：Outer join 32：Sum 64：Full join
command_text	text	操作的执行命令。

示例：

1. 查询审计语句中所有对象的列及其在语句中使用的类型：

```
SET audit_object_details = on;
SELECT object_name,object_details,result_rows,error_code,command_text FROM
pg_query_audit('2023-05-12-03 8:00:00','2023-05-12 22:55:00') where command_text like '%student%';
```

查询结果如下：

```
                        object_name                         |
object_details                                | result_rows | error_code
|                                                    command_text
-------------------------------------------------------------------
+-----------------------------------------------------------------------------------------+-------------+------------
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------
 student
|                                                                | 0           |           | CREATE TABLE
student(stuNo int, stuName TEXT);
 studentscore
|                                                                | 0           |           | CREATE TABLE
studentscore(stuNo int, stuscore int);
 ["public.student_view01","public.studentscore","public.student"]
|                                                                | 0           |           | CREATE OR
REPLACE VIEW student_view01 AS SELECT * FROM student t1 where t1.stuNo in (select stuNo from
studentscore t2 where t1.stuNo = t2.stuNo);
 ["public.student_view01","public.student","public.studentscore"] | {"public.student":[{"stuno":"6"},
```

 版权所有 © 华为云计算技术有限公司

```
{"stuname":"2"}],"public.studentscore":[{"stuno":"6"}]} | 0           |           | SELECT * FROM
student_view01
```

2. 查询结果object_details中显示部分语句执行过程中，涉及到的列及其使用类型，使用json格式进行记录。

   使用pg_query_audit_details函数对object_name和object_details列进行解析：

```
SELECT database,object_name,column_name,type_of_use,use_type FROM
pg_query_audit_details('2021-02-03 8:00:00','2024-02-03 22:55:00','current') where command_text like
'%student%';
```

   查询结果如下：

```
 database |      object_name      | column_name | type_of_use |      use_type
----------+-----------------------+-------------+-------------+--------------------
 gaussdb  | student               |             | 2           | Access
 gaussdb  |                       |             | 2           | Access
 gaussdb  | studentscore          |             | 2           | Access
 gaussdb  |                       |             | 2           | Access
 gaussdb  | public.student_view01 |             | 2           | Access
 gaussdb  | public.studentscore   |             | 2           | Access
 gaussdb  | public.student        |             | 2           | Access
 gaussdb  |                       |             | 2           | Access
 gaussdb  | public.student        | stuno       | 6           | Access,Conditional
 gaussdb  | public.student        | stuname     | 2           | Access
 gaussdb  | public.studentscore   | stuno       | 6           | Access,Conditional
 gaussdb  | public.student_view01 |             | 2           | Access
 gaussdb  | public.student        |             | 2           | Access
 gaussdb  | public.studentscore   |             | 2           | Access
 gaussdb  |                       |             | 2           | Access
(15 rows)
```

pg_delete_audit()

描述：删除指定时间段的审计日志。8.1.3及以上集群版本中该函数已废弃。 返回值类型：void

说明

基于数据库安全考虑，不提供删除指定时间段的审计日志的函数接口，调用该函数将直接报“ERROR: For security purposes, it is not allowed to manually delete audit logs”。

create_audit_csv_foregion_table()

描述：创建读审计日志的分区外表。用户调用该函数生成审计日志外表pgxc_audit_logs，通过读取pgxc_audit_logs可以读取放在obs上的审计日志信息。该函数仅8.3.0及以上集群支持。

入参：

- obs_server：text类型，obs server名称
- file_path：text类型，obs文件路径

返回值类型：record

示例：

```
--创建obs server
CREATE SERVER obs_server FOREIGN DATA WRAPPER DFS_FDW OPTIONS (
  address 'obs.example.xxx:xxx',
  encrypt 'true',
  access_key 'xxxxxxxxx',
  secret_access_key 'xxxxxxxxxxxxxx',
  type 'obs' );
--调用该函数生成审计日志外表
SELECT * FROM pg_catalog.create_audit_csv_foregion_table('obs_server','/obs-audit/test/');
```


版权所有 © 华为云计算技术有限公司

```
 create_audit_csv_foregion_table
---------------------------------
 t
(1 row)
--读取放在obs上的审计日志信息
SELECT * FROM pgxc_audit_logs where year=2023 and month=7 and date=24;
 year | month | date |          begintime          |          endtime            | operation_type |    audit_type     | result
| username | database | client_conninfo |          object_name          | object_details |command_text|
detail_info | transaction_xid |      query_id      | node_name |             session_id              | local_port |
remote_port | result_rows | error_code
------+-------+------+-----------------------------+-----------------------------+----------------+-------------------
+--------+----------+----------+-----------------+---------------------------------+----------------
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+------------+----------------+------------+-------------------
+-----------+-------------------------------------+------------+------------+------------+------------
 2023 | 7     | 24   | 2023-07-24 09:56:40.727+08 | 2023-07-24 09:56:42.215+08 | dml            |
dml_action_select | ok     | dbadmin  | gaussdb  | gsql@[local]    | public.pgxc_audit_logs         |               |
select * f
rom pgxc_audit_logs where year=2023 and date=24 and month=7;
```

21.7.16 条件表达式函数

coalesce(expr1, expr2, ..., exprn)

描述：返回参数列表中第一个非NULL的参数值。

COALESCE(expr1, expr2) 等价于CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END。

示例：

```
SELECT coalesce(NULL,'hello');
 coalesce
----------
 hello
(1 row)
```

说明

- 如果表达式列表中的所有表达式都等于NULL，则本函数返回NULL。
- 它常用于在显示数据时用缺省值替换NULL。
- 和CASE表达式一样，COALESCE不会计算不需要用来判断结果的参数；即在第一个非空参数右边的参数不会被计算。

decode(base_expr, compare1, value1, Compare2,value2, … default)

描述：把base_expr与后面的每个compare(n) 进行比较，如果匹配返回相应的value(n)。如果没有发生匹配，则返回default。

示例：

```
SELECT decode('A','A',1,'B',2,0);
 case
------
 1
(1 row)
```

if(bool_expr, expr1, expr2)

描述：当bool_expr为true时，返回expr1，否则返回expr2。

版权所有 © 华为云计算技术有限公司

if(bool_expr, expr1, expr2) 等价于CASE WHEN bool_expr = true THEN expr1 ELSE expr2 END。

示例：

```
SELECT if(1 < 2, 'yes', 'no');
 if
-----
 yes
(1 row)
```

说明

参数expr1和expr2可以为任意类型，返回结果类型规则请参考**UNION，CASE和相关构造**。

ifnull(expr1, expr2)

描述：当expr1不为NULL时，返回expr1，否则返回expr2。

ifnull(expr1, expr2) 逻辑上等价于CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END。

示例：

```
SELECT ifnull(NULL,'hello');
 ifnull
--------
 hello
(1 row)
```

说明

参数expr1和expr2可以为任意类型，返回结果类型规则请参考**UNION，CASE和相关构造**。

isnull(expr)

描述：当expr为NULL时，返回true，否则返回false。

isnull(expr) 逻辑上等价于expr IS NULL。

示例：

```
SELECT isnull(NULL), isnull('abc');
 isnull | isnull
--------+--------
 t      | f
(1 row)
```

nullif(expr1, expr2)

描述：当且仅当expr1和expr2相等时，NULLIF才返回NULL，否则它返回expr1。

nullif(expr1, expr2) 逻辑上等价于CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END。

示例：

```
SELECT nullif('hello','world');
 nullif
--------
 hello
(1 row)
```

备注：

版权所有 © 华为云计算技术有限公司

如果两个参数的数据类型不同，则：

- 两种数据类型之间存在隐式转换，则以其中优先级较高的数据类型为基准将另一个参数隐式转换成该类型，转换成功则进行计算，转换失败则返回错误。如：

```
SELECT nullif('1234'::VARCHAR,123::INT4);
 nullif
--------
   1234
(1 row)
SELECT nullif('1234'::VARCHAR,'2012-12-24'::DATE);
ERROR:  invalid input syntax for type timestamp: "1234"
```

- 两种数据类型之间不存在隐式转换，则返回错误 。如：

```
SELECT nullif(TRUE::BOOLEAN,'2012-12-24'::DATE);
ERROR:  operator does not exist: boolean = timestamp without time zone
LINE 1: SELECT nullif(TRUE::BOOLEAN,'2012-12-24'::DATE) FROM DUAL;
^
HINT:  No operator matches the given name and argument type(s). You might need to add explicit type casts.
```

nvl(expr1 , expr2)

描述：如果expr1为NULL则返回expr2。如果expr1非NULL，则返回expr1。

示例：

```
SELECT nvl('hello','world');
  nvl
-------
 hello
(1 row)
```

说明

参数expr1和expr2可以为任意类型，当NVL的两个参数不属于同类型时，看第二个参数是否可以向第一个参数进行隐式转换，如果可以则返回第一个参数类型。如果第二个参数不能向第一个参数进行隐式转换而第一个参数可以向第二个参数进行隐式转换，则返回第二个参数的类型。如果两个参数之间不存在隐式类型转换并且也不属于同一类型则报错。

sys_context('namespace' , 'parameter')

描述：获取并返回指定namespace下参数parameter的值。

返回值类型：text

示例：

```
SELECT sys_context('USERENV', 'CURRENT_SCHEMA');
 sys_context
-------------
 public
(1 row)
```

根据当前所在的实际schema而变化。

说明

目前仅支持SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA') 和SYS_CONTEXT('USERENV', 'CURRENT_USER')两种格式。

greatest(expr1 [, ...])

描述：获取并返回参数列表中值最大的表达式的值。

版权所有 © 华为云计算技术有限公司

- ORA和TD兼容模式下，返回结果为所有非null参数的最大值。
- MySQL兼容模式下，入参中存在null时，返回结果为null。

示例：

```
SELECT greatest(1*2,2-3,4-1);
 greatest
----------
        3
(1 row)
SELECT greatest('ABC', 'BCD', 'CDE');
 greatest
----------
 CDE
(1 row)
```

least(expr1 [, ...])

描述：获取并返回参数列表中值最小的表达式的值。

- ORA和TD兼容模式下，返回结果为所有非null参数的最小值。
- MySQL兼容模式下，入参中存在null时，返回结果为null。

示例：

```
SELECT least(1*2,2-3,4-1);
 least
-------
    -1
(1 row)
SELECT least('ABC','BCD','CDE');
 least
--------
 ABC
(1 row)
```

EMPTY_BLOB()

描述：使用EMPTY_BLOB在INSERT或UPDATE语句中初始化一个BLOB变量，取值为NULL。

返回值类型：BLOB

示例：

```
--新建表
CREATE TABLE blob_tb(b blob,id int) DISTRIBUTE BY REPLICATION;
--插入数据
INSERT INTO blob_tb VALUES (empty_blob(),1);
--删除表
 DROP TABLE blob_tb;
```

说明

使用DBMS.GETLENGTH求得的长度为0。

21.7.17 范围函数和操作符

版权所有 © 华为云计算技术有限公司

21.7.17.1 范围操作符

=

描述：等于

示例：

```
SELECT int4range(1,5) = '[1,4]'::int4range AS RESULT;
 result
--------
 t
(1 row)
```

<>

描述：不等于

示例：

```
SELECT numrange(1.1,2.2) <> numrange(1.1,2.3) AS RESULT;
 result
--------
 t
(1 row)
```

<

描述：小于

示例：

```
SELECT int4range(1,10) < int4range(2,3) AS RESULT;
 result
--------
 t
(1 row)
```

>

描述：大于

示例：

```
SELECT int4range(1,10) > int4range(1,5) AS RESULT;
 result
--------
 t
(1 row)
```

<=

描述：小于或等于

示例：

```
SELECT numrange(1.1,2.2) <= numrange(1.1,2.2) AS RESULT;
 result
--------
 t
(1 row)
```


版权所有 © 华为云计算技术有限公司

>=

描述：大于或等于

示例：

```
SELECT numrange(1.1,2.2) >= numrange(1.1,2.0) AS RESULT;
 result
--------
 t
(1 row)
```

@>

描述：包含范围、包含元素

示例：

```
SELECT int4range(2,4) @> int4range(2,3) AS RESULT;
 result
--------
 t
(1 row)
SELECT '[2011-01-01,2011-03-01)'::tsrange @> '2011-01-10'::timestamp AS RESULT;
 result
--------
 t
(1 row)
```

<@

描述：范围包含于、元素包含于

示例：

```
SELECT int4range(2,4) <@ int4range(1,7) AS RESULT;
 result
--------
 t
(1 row)
SELECT 42 <@ int4range(1,7) AS RESULT;
 result
--------
 f
(1 row)
```

&&

描述：重叠（有共同点）

示例：

```
SELECT int8range(3,7) && int8range(4,12) AS RESULT;
 result
--------
 t
(1 row)
```

<<

描述：范围值是否比另一个范围值的最小值还小（没有交集）

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT int8range(1,10) << int8range(100,110) AS RESULT;
 result
--------
 t
(1 row)
```

>>

描述：范围值是否比另一个范围值的最大值还大（没有交集）

示例：

```
SELECT int8range(50,60) >> int8range(20,30) AS RESULT;
 result
--------
 t
(1 row)
```

&<

描述：范围值的最大值是否不超过另一个范围值的最大值。

示例：

```
SELECT int8range(1,20) &< int8range(18,20) AS RESULT;
 result
--------
 t
(1 row)
```

&>

描述：范围值的最小值是否不小于另一个范围值的最小值。

示例：

```
SELECT int8range(7,20) &> int8range(5,10) AS RESULT;
 result
--------
 t
(1 row)
```

-|-

描述：相邻

示例：

```
SELECT numrange(1.1,2.2) -|- numrange(2.2,3.3) AS RESULT;
 result
--------
 t
(1 row)
```

+

描述：并集

示例：

```
SELECT numrange(5,15) + numrange(10,20) AS RESULT;
 result
--------
```

 版权所有 © 华为云计算技术有限公司

```
 [5,20)
(1 row)
```

*****

描述：交集

示例：

```
SELECT int8range(5,15) * int8range(10,20) AS RESULT;
 result
---------
 [10,15)
(1 row)
```

-

描述：差集

示例：

```
SELECT int8range(5,15) - int8range(10,20) AS RESULT;
 result
--------
 [5,10)
(1 row
```

说明

- 简单的比较操作符<，>，<=和>=先比较下界，只有下界相等时才比较上界。
- <<、>>和-|-操作符当包含空范围时也会返回false；也就是，不认为空范围在其他范围之前或之后。
- 并集和差集操作符的执行结果无法包含两个不相交的子范围。

21.7.17.2 范围函数

lower(anyrange)

描述：范围的下界

返回类型：范围元素类型

示例：

```
SELECT lower(numrange(1.1,2.2)) AS RESULT;
 result
--------
    1.1
(1 row)
```

upper(anyrange)

描述：范围的上界

返回类型：范围元素类型

示例：

```
SELECT upper(numrange(1.1,2.2)) AS RESULT;
 result
--------
```


版权所有 © 华为云计算技术有限公司

```
   2.2
(1 row)
```

isempty(anyrange)

描述：范围是否为空

返回类型：boolean

示例：

```
SELECT isempty(numrange(1.1,2.2)) AS RESULT;
 result
--------
 f
(1 row)
```

lower_inc(anyrange)

描述：是否包含下界

返回类型：boolean

示例：

```
SELECT lower_inc(numrange(1.1,2.2)) AS RESULT;
 result
--------
 t
(1 row)
```

upper_inc(anyrange)

描述：是否包含上界

返回类型：boolean

示例：

```
SELECT upper_inc(numrange(1.1,2.2)) AS RESULT;
 result
--------
 f
(1 row)
```

lower_inf(anyrange)

描述：下界是否为无穷

返回类型：boolean

示例：

```
SELECT lower_inf('(,)'::daterange) AS RESULT;
 result
--------
 t
(1 row)
```

upper_inf(anyrange)

描述：上界是否为无穷

 版权所有 © 华为云计算技术有限公司

返回类型：boolean

示例：

```
SELECT upper_inf('(,)'::daterange) AS RESULT;
 result
--------
 t
(1 row)
```

说明

如果范围是空或者需要的界限是无穷的，lower和upper函数将返回null。lower_inc、upper_inc、lower_inf和upper_inf函数均对空范围返回false。

21.7.18 数据脱敏函数

数据脱敏函数提供一系列不同脱敏形式的函数接口，可以覆盖常见的脱敏场景。通常结合数据脱敏语法，与脱敏列绑定使用，不推荐直接作用在查询语句上。

mask_none(column_name)

描述：不作任何脱敏处理，仅内部测试用。

返回值类型：与入参column_name数据类型相同

mask_full(column_name)

描述：全脱敏成固定值。脱敏列的数据类型不同，脱敏的固定值不同。

返回值类型：与入参column_name数据类型相同。

mask_partial(column_name, mask_digital, mask_from[, mask_to])

描述：针对数值类型数据，将第mask_from到mask_to位的数字部分脱敏成mask_digital对应的数字。其中，参数mask_to允许缺省，缺省时即脱敏到数据结束位置。参数mask_digital只能取[0,9]区间内的数字。

返回值类型：与入参column_name数据类型相同。

mask_partial(column_name [, input_format, output_format], mask_char, mask_from[, mask_to])

描述：针对字符类型数据，对照指定的输入输出格式，将第mask_from到mask_to位的数字部分脱敏成mask_char指定的字符。

参数说明：

- input_format

  输入格式是由V和F组成的字符序列，与脱敏列数据长度相同。V对应位置的字符可能会被脱敏，F对象位置的字符会被忽略跳过，V字符序列标识脱敏范围。输入输出格式参数适用于定长数据，比如，银行卡号、身份证号、手机号等。

- output_format

  输出格式是由V和其他任意字符组成的字符序列，与脱敏列数据长度相同。V字符位置与input_format的V位置对应，其他字符位置与input_format的F位置对应，且不会脱敏，通常为数据分隔符。

版权所有 © 华为云计算技术有限公司

input_format和output_format可以缺省或指定为空串""，此时，无输入输出格式要求，原始字符序列范围即为脱敏范围。

- mask_char

  脱敏字符，仅允许长度为1的任意字符。场景的脱敏字符包括"*"，"#"等。

- mask_from

  脱敏范围的起始位置，要求大于0。

- mask_to

  脱敏范围的结束位置，允许缺省。缺省时，即脱敏到原始数据结束位置。

返回值类型：与入参column_name数据类型相同。

mask_partial(column_name, mask_field1, mask_value1, mask_field2, mask_value2, mask_field3, mask_value3)

描述：按指定三个时间域做部分脱敏，仅适用于日期或时间类型数据。若mask_value取-1，即此mask_field不脱敏。其中，mask_field可以取"month"、"day"、"year"、"hour"、"minute"、"second"六个时间域之一。各域的取值范围需满足实际时间单位的取值范围。

返回值类型：与入参column_name数据类型相同。

说明

脱敏函数可以覆盖常见敏感信息的脱敏场景，推荐用户优先使用脱敏函数创建脱敏策略。

数据脱敏函数的使用方法，请参考《 数据仓库服务开发指南 》中“DWS数据库安全管理 > DWS敏感数据管理 > DWS数据脱敏”中的示例。

自定义脱敏函数

支持用户使用PL/PGSQL语言自定义脱敏函数。

自定义脱敏函数需要严格遵循如下要求：

- 返回值与脱敏列类型一致。
- 函数必须可下推。
- 参数列表除脱敏格式外，只能包含一个脱敏列。
- 函数仅实现针对特定数据类型的格式化改写功能，不能涉及与其他表对象的复杂关联操作。

不满足前两项中任一项时，创建脱敏策略会报错。不满足后两项中任一项时，可成功创建脱敏策略，但查询执行结果可能会出现不可预知的问题。

如果绑定的自定义脱敏函数被删除，查询会因为脱敏策略失效而执行报错。

21.7.19 位图函数和操作符

21.7.19.1 Roaring Bitmap 操作符

DWS自8.1.3集群版本开始，支持高效的位图处理操作符，可用于用户画像，精准营销等场景，极大的提高了查询性能。

版权所有 © 华为云计算技术有限公司

=

描述：比较两个roaringbitmap是否相等。

返回值类型：bool

示例：

```
SELECT rb_build('{1,2,3}') = rb_build('{1,2,3}');
?column?
----------
t
(1 row)
SELECT rb_build('{2,3}') = rb_build('{1,2,3}');
?column?
----------
f
(1 row)
```

<>

描述：比较两个roaringbitmap是否不相等。

返回值类型：bool

示例：

```
SELECT rb_build('{1,2,3}') <> rb_build('{1,2,3}');
?column?
----------
f
(1 row)
SELECT rb_build('{2,3}') <> rb_build('{1,2,3}');
?column?
----------
t
(1 row)
```

&

描述：计算两个roaringbitmap交集以后的结果。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3}') & rb_build('{1,2,3}'));
rb_to_array
-------------
{2,3}
(1 row)
```

|

描述：计算两个roaringbitmap并集后的结果。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3}') | rb_build('{1,2,3}'));
rb_to_array
-------------
{1,2,3}
(1 row)
```

 版权所有 © 华为云计算技术有限公司

|

描述：计算一个roaringbitmap增加一个ID后的结果。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3}') | 4);
rb_to_array
-------------
{2,3,4}
(1 row)
```

#

描述：两个roaringbitmap进行异或后的结果。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3}') # rb_build('{1,2,3}'));
rb_to_array
-------------
{1}
(1 row)
```

-

描述：计算属于第一个roaringbitmap，但是不属于第二个roaringbitmap的结果集。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3,4}') - rb_build('{1,2,3}'));
rb_to_array
-------------
{4}
(1 row)
```

-

描述：从roaringbitmap中去掉一个指定ID后的结果集。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_build('{2,3,4}') - 3);
rb_to_array
-------------
{2,4}
(1 row)
```

@>

描述：操作符前面的roaringbitmap是否包含后面的roaringbitmap。

返回值类型：bool

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT rb_build('{2,3,4}') @> rb_build('{2,3}');
?column?
----------
t
(1 row)
SELECT rb_build('{2,3,4}') @> 4;
?column?
----------
t
(1 row)
```

<@

描述：操作符前面的roaringbitmap是否被后面的roaringbitmap包含。

返回值类型：bool

示例：

```
SELECT  4 <@ rb_build('{2,3,4}');
?column?
----------
t
(1 row)
SELECT rb_build('{2,3,4}') <@ rb_build('{2,3}');
?column?
----------
f
(1 row)
```

&&

描述：两个roaringbitmap如果有交集返回true，否则返回false。

返回值类型：bool

示例：

```
SELECT rb_build('{2,3,4}') && rb_build('{2,3}');
?column?
----------
t
(1 row)
SELECT rb_build('{2,3,4}') && rb_build('{7,8,9}');
?column?
----------
f
(1 row)
```

21.7.19.2 位图函数

DWS自8.1.3集群版本开始，支持高效的位图处理函数和操作符，可用于用户画像，精准营销等场景，极大的提高了查询性能。

rb_build(array)

描述：将int数组转成roaringbitmap类型。

返回值类型：roaringbitmap

示例：

```
SELECT rb_build('{1,2,3}');
rb_build
```

 版权所有 © 华为云计算技术有限公司

```
-----------------------------------------------
\x3a30000001000000000002001000000010002000300
(1 row)
CREATE TABLE r_row (a int, b text, c roaringbitmap);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE

INSERT INTO r_row values (1, 'a', rb_build('{1,2,3}'));
INSERT 0 1

SELECT * FROM r_row;
 a | b |                        c
---+---+-------------------------------------------------
 1 | a | \x3a30000001000000000002001000000010002000300
(1 row)

INSERT INTO r_row values (2, 'b', rb_build('{}'));
INSERT 0 1

SELECT * FROM r_row;
 a | b |                        c
---+---+-------------------------------------------------
 2 | b | \x3a30000000000000
 1 | a | \x3a30000001000000000002001000000010002000300
(2 rows)
```

rb_iterate(roaringbitmap)

描述：把roaringbitmap数据转成int，按照多行输出。

返回值类型：record类型(多行int值)

示例：

```
SELECT rb_iterate(c) FROM r_row;
rb_iterate
------------
1
2
3
(3 rows)
```

rb_to_array(roaringbitmap)

描述：rb_build的逆向操作，把roaringBitmap转成int数组。

返回值类型: array

示例：

```
SELECT rb_to_array(c) FROM r_row;
rb_to_array
-------------
{1,2,3}
(1 row)
SELECT rb_to_array('\x3a30000001000000000002001000000010002000300');
rb_to_array
-------------
{1,2,3}
(1 row)
```

rb_and(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap的交集。

 版权所有 © 华为云计算技术有限公司

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_and(rb_build('{1,2,3}'), rb_build('{2,3,4}')));
rb_to_array
-------------
{2,3}
(1 row)
```

rb_or(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap的并集。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_or(rb_build('{1,2,3}'), rb_build('{2,3,4}')));
rb_to_array
-------------
{1,2,3,4}
(1 row)
```

rb_xor(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap的异或。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_xor(rb_build('{1,2,3}'), rb_build('{2,3,4}')));
rb_to_array
-------------
{1,4}
(1 row)
```

rb_andnot(roaringbitmap, roaringbitmap)

描述：在第一个roaringbitmap集合中，但是不在第二个roaringbitmap中的集合。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_andnot(rb_build('{1,2,3}'), rb_build('{2,3,4}')));
rb_to_array
-------------
{1}
(1 row)
```

rb_cardinality(roaringbitmap)

描述：计算一个roaringbitmap的基数。

返回值类型：int

示例：

```
SELECT rb_cardinality(rb_build('{1,2,3}'));
rb_cardinality
----------------
```

 版权所有 © 华为云计算技术有限公司

```
3
(1 row)
```

rb_and_cardinality(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap的交集的基数。

返回值类型：int

示例：

```
SELECT rb_and_cardinality(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_and_cardinality
--------------------
2
(1 row)
```

rb_or_cardinality(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap的并集的基数。

返回值类型：int

示例：

```
SELECT rb_or_cardinality(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_or_cardinality
-------------------
4
(1 row)
```

rb_xor_cardinality(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap异或以后的基数。

返回值类型：int

示例：

```
SELECT rb_xor_cardinality(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_xor_cardinality
--------------------
2
(1 row)
```

rb_andnot_cardinality(roaringbitmap, roaringbitmap)

描述：计算两个roaringbitmap按照andnot计算结果以后的基数。

返回值类型：int

示例：

```
SELECT rb_andnot_cardinality(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_andnot_cardinality
-----------------------
1
(1 row)
```

rb_is_empty(roaringbitmap)

描述：判断一个roaringbitmap是否为空。

 版权所有 © 华为云计算技术有限公司

返回值类型：bool

示例：

```
SELECT rb_is_empty(rb_build('{1,2,3}'));
rb_is_empty
-------------
f
(1 row)
```

rb_equals(roaringbitmap, roaringbitmap)

描述：判断两个roaringbitmap是否相等。

返回值类型：bool

示例：

```
SELECT rb_equals(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_equals
-----------
f
(1 row)
```

rb_intersect(roaringbitmap, roaringbitmap)

描述：判断两个roaringbitmap是否相交。

返回值类型：bool

示例：

```
SELECT rb_intersect(rb_build('{1,2,3}'), rb_build('{2,3,4}'));
rb_intersect
--------------
t
(1 row)
```

rb_min(roaringbitmap)

描述：返回roaringbitmap中的最小值。

返回值类型：int

示例：

```
SELECT rb_min(rb_build('{1,2,3}'));
rb_min
--------
1
(1 row)
```

rb_max(roaringbitmap)

描述：返回roaringbitmap中的最大值。

返回值类型：int

示例：

```
SELECT rb_max(rb_build('{1,2,3}'));
rb_max
--------
```

 版权所有 © 华为云计算技术有限公司

```
3
(1 row)
```

rb_add(roaringbitmap, int)

描述：在roaringbitmap中增加一个元素。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_add(rb_build('{1,3}'), 2));
rb_to_array
-------------
{1,2,3}
(1 row)
```

rb_added(int, roaringbitmap)

描述：在roaringbitmap中增加一个元素。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_added(2, rb_build('{1,3}')));
rb_to_array
-------------
{1,2,3}
(1 row)
```

rb_contain(roaringbitmap,int)

描述：判断roaringbitmap是否包含指定的元素。

返回值类型：bool

示例：

```
SELECT rb_contain(rb_build('{1,3}'), 2);
rb_contain
------------
f
(1 row)
```

rb_containedby(int,roaringbitmap)

描述：判断给定的元素是否被给定的roaringbitmap包含。

示例：

```
SELECT rb_containedby(2,rb_build('{1,3}'));
rb_containedby
----------------
f
(1 row)
```

rb_contain_rb(roaringbitmap,roaringbitmap)

描述：判断第一个roaringbitmap是否包含第二个roaringbitmap。

返回值类型：bool

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT rb_contain_rb(rb_build('{1,3}'), rb_build('{2,3}'));
rb_contain_rb
---------------
f
(1 row)
```

rb_containedby_rb(roaringbitmap,roaringbitmap)

描述：判断给定的第二个roaringbitmap是否包含第一个roaringbitmap。

返回值类型：bool

示例：

```
SELECT rb_containedby_rb(rb_build('{1,3}'), rb_build('{2,3}'));
rb_containedby_rb
---------------
f
(1 row)
```

rb_remove(roaringbitmap,int)

描述：从roaringbitmap中移除指定的元素。

返回值类型：roaringbitmap

示例：

```
SELECT rb_to_array(rb_remove(rb_build('{1,3}'),1));
rb_to_array
-------------
{3}
(1 row)
```

rb_clear(roaringbitmap,int,int)

描述：从roaringbitmap中清除指定范围内的元素。

返回值类型：roaringbitmap

示例：

```
SELECT
rb_to_array(rb_clear(rb_build('{1,2,3}'),1,2));

rb_to_array

-------------

{2,3}
                                                                (1 row)
```

rb_flip(roaringbitmap,int,int)

描述：反转指定范围的元素。

示例：

```
SELECT rb_to_array(rb_flip(rb_build('{1,2,3,7,9}'), 1,10));
rb_to_array
---------------
{4,5,6,8,10}
(1 row)
```

 版权所有 © 华为云计算技术有限公司

rb_rank(roaringbitmap,int)

描述：返回小于指定值的集合的基数。

返回值类型：int

示例：

```
SELECT rb_rank(rb_build('{1,10,100}'),99);
rb_rank
---------
2
(1 row)
```

21.7.19.3 Roaring Bitmap 聚合函数

DWS自8.1.3集群版本开始，支持高效的位图聚合函数，可用于用户画像，精准营销等场景，极大的提高了查询性能。

rb_build_agg(int)

描述：将分组内的int值聚合成一个roaringbitmap值。

返回值类型：roaringbitmap

示例：

```
CREATE TABLE t1 (a int ,b int);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
INSERT INTO t1 SELECT generate_series(1,10),generate_series(1,20,2);
INSERT 0 10
SELECT rb_iterate(rb_build_agg(b)) FROM t1;
rb_iterate
------------
1
3
5
7
9
11
13
15
17
19
(10 rows)
```

rb_and_agg(roaringbitmap)

描述：将分组内的roaringbitmap数据按照交的操作聚合成一个roaringbitmap集合。

示例：

```
CREATE TABLE r1(a int ,b roaringbitmap);
INSERT INTO r1 SELECT a, rb_build_agg(b) FROM t1 GROUP BY a;
INSERT INTO t1 SELECT generate_series(1,10),generate_series(1,20,4);
INSERT INTO r1 SELECT a, rb_build_agg(b) FROM t1 GROUP BY a;
SELECT a, rb_to_array(rb_and_agg(b)) FROM r1 GROUP BY a ORDER BY a;
 a  | rb_to_array
----+-------------
 1  | {1}
 2  | {3}
 3  | {5}
 4  | {7}
```


版权所有 © 华为云计算技术有限公司

```
 5  | {9}
 6  | {11}
 7  | {13}
 8  | {15}
 9  | {17}
10  | {19}
(10 rows)
```

rb_or_agg(roaringbitmap)

描述：将分组内的roaringbitmap按照并的逻辑组合成一个roaringbitmap。

示例：

```
SELECT a, rb_to_array(rb_or_agg(b)) FROM r1 GROUP BY a ORDER BY a;
 a  | rb_to_array
----+-------------
 1  | {1}
 2  | {3,5}
 3  | {5,9}
 4  | {7,13}
 5  | {9,17}
 6  | {1,11}
 7  | {5,13}
 8  | {9,15}
 9  | {13,17}
 10 | {17,19}
(10 rows)
```

rb_xor_agg(roaringbitmap)

描述：将分组内的roaringbitmap按照异或的逻辑组合成一个roaringbitmap。

示例：

```
SELECT a, rb_to_array(rb_xor_agg(b)) FROM r1 GROUP BY a ORDER BY a;
a   | rb_to_array
----+-------------
 1  | {}
 2  | {5}
 3  | {9}
 4  | {13}
 5  | {17}
 6  | {1}
 7  | {5}
 8  | {9}
 9  | {13}
 10 | {17}
(10 rows)
```

rb_and_cardinality_agg(roaringbitmap)

描述：分组内的roaringbitmap按照交集计算后的基数。

返回值类型：int

示例：

```
SELECT a, rb_and_cardinality_agg(b) FROM r1 GROUP BY a ORDER BY 1;
 a  | rb_and_cardinality_agg
----+------------------------
 1  |                      1
 2  |                      1
 3  |                      1
 4  |                      1
 5  |                      1
```


版权所有 © 华为云计算技术有限公司

```
 6 |                  1
 7 |                  1
 8 |                  1
 9 |                  1
 10 |                 1
(10 rows)
```

rb_or_cardinality_agg(roaringbitmap)

描述：将分组内的roaringbitmap按照并集计算后的基数。

示例：

```
SELECT a, rb_or_cardinality_agg(b) FROM r1 GROUP BY a ORDER BY 1;
 a  | rb_or_cardinality_agg
----+-----------------------
 1  |                     1
 2  |                     2
 3  |                     2
 4  |                     2
 5  |                     2
 6  |                     2
 7  |                     2
 8  |                     2
 9  |                     2
 10 |                     2
(10 rows)
```

rb_xor_cardinality_agg(roaringbitmap)

描述：将分组内的roaringbitmap按照异或的逻辑合并后的基数。

示例：

```
SELECT a, rb_xor_cardinality_agg(b) FROM r1 GROUP BY a ORDER BY 1;
 a  | rb_xor_cardinality_agg
----+------------------------
 1  |                      0
 2  |                      1
 3  |                      1
 4  |                      1
 5  |                      1
 6  |                      1
 7  |                      1
 8  |                      1
 9  |                      1
 10 |                      1
(10 rows)
```

21.7.19.4 使用场景

背景

目前在互联网、教育、游戏等行业都有实时精准营销的需求。通过系统生成用户画像，在营销时通过条件组合筛选用户，快速提取目标群体。例如：

- 在电商行业中，商家在进行营销活动前，需要根据活动的目的，圈选一批满足特定特征的目标用户群体进行广告推送。
- 在教育行业中，需要根据学生不同的特征，推送有针对性的练习题目，帮助学生查漏补缺。
- 在搜索、视频、门户网站中，根据用户关注的热点，推送不同的内容。

这些业务场景都有一些共同的特点：

版权所有 © 华为云计算技术有限公司

- 数据量庞大，运算量极大。
- 用户规模庞大，标签多，字段多，占用存储空间也多。
- 圈选的特征条件多样化，很难找到固定索引，如果每个字段一个索引，存储空间又会暴增。
- 性能要求高，因为实时营销要求秒级响应。
- 数据更新时效要求高，用户画像几乎要求实时更新。

针对上述业务场景特点，DWS的roaringbitmap可以高效生成、压缩、解析位图数据，支持最常见的位图聚合操作（与、或、非、异或），满足用户在亿级以上、千万级标签的大数据量下实时精准营销、快速圈选用户的需求。

roaringbitmap 使用示例

假设有一张用户浏览网页的流水信息表userinfo，表中的字段如下：

```
CREATE TABLE userinfo
(userid int,
age int,
gender text,
salary int,
hobby  text
)WITH (orientation=column);
```

userinfo表中的数据会随着用户信息的变化不断增长，比如用户有多个hobby属性，那么就有多条记录。

如果用户需要筛选出所有“收入大于10000元的男性，年龄大于30岁，爱好钓鱼”的群体，向这些目标群体推送特定的消息。

传统的方法是直接在原表上执行查询，语句如下：

```
SELECT distinct userid FROM userinfo WHERE salary > 10000 AND age > 30 AND gender ='m' AND hobby ='fishing';
```

当userinfo表的数据量不大的时候，可以通过在salary，age，gender，hobby列上建立索引来满足需求。但是如果userinfo表的数据量非常大，同时一张表的标签数非常多的时候，上述语句就不能满足诉求，因为如下原因：

- 需要创建的索引会非常多。
- count (distinct)的性能比较差。

这种场景下使用roaringbitmap就会有比较好的效果。

1. 新建一张Roaringbitmap表：
   ```
   CREATE TABLE userinfoset
   ( age int,
   gender text,
   salary int,
   hobby  text,
   userset roaringbitmap,
   PRIMARY KEY(age,gender,salary,hobby)
   )WITH (orientation=column);
   ```
2. 所有userinfo表中的数据要通过标签列聚合到userinfoset表中。可以采用对全量数据进行聚合的方法（如下命令所示）。也可以采用只对增量数据进行聚合的方法。只对增量数据进行聚合即对含有相同的标签的用户集合放到表的一条记录中，通常可以通过upsert来实现。考虑到其中频繁的update操作可能产生大量的脏数据，因此对增量数据进行聚合的方法，建议将userinfoset表创建为行存表。
   ```
   INSERT INTO userinfoset
   SELECT age, gender, salary, hobby, rb_build_agg(userid)
   ```

 版权所有 © 华为云计算技术有限公司

```
FROM
userinfo
GROUP BY age, gender, salary, hobby;
```

3. 直接查询userinfoset表获得用户筛选信息。

```
SELECT rb_iterate(rb_or_agg(userset)) FROM userinfoset WHERE salary > 10000 AND age > 30 AND
gender ='m' AND hobby ='fishing';
```

数据进行聚合后的userinfoset的数据量相比源表小了很多，基表scan的性能会快很多，同时基于Roaringbitmap的优势，计算rb_or_agg和rb_iterate的性能也很好，相比传统的方法，性能明显提升。

21.7.20 UUID 函数

UUID函数表示可以用于生成UUID类型（请参考**UUID类型**）数据的函数。

uuid_generate_v1()

描述：生成一个UUID类型的序列号。

返回类型：UUID

示例：

```
SELECT uuid_generate_v1();
          uuid_generate_v1
--------------------------------------
 c71ceaca-a175-11e9-a920-797ff7000001
(1 row)
```

说明

uuid_generate_v1函数根据时间信息、集群节点编号和生成该序列的线程号生成UUID，该UUID在单个集群内是全局唯一的，但在多个集群间的时间信息、集群节点编号、线程号和时钟序列仍然存在同时相等的可能性，因此多个集群间生成的UUID仍然存在极低概率的重复风险。

uuid()

描述：生成一个UUID类型的序列号。此函数为MySQL兼容性函数，仅8.2.0及以上集群版本支持。

返回类型：UUID

示例：

```
SELECT uuid();
             uuid
-----------------------------------
 6327dc96-f0e7-0100-f2f2-6c9ff700fffe
(1 row)
```

说明

uuid函数内部生成原理同**uuid_generate_v1()**函数，即根据时间信息、集群节点编号和生成该序列的线程号生成UUID，该UUID在单个集群内是全局唯一的，但在多个集群间的时间信息、集群节点编号、线程号和时钟序列仍然存在同时相等的可能性，因此多个集群间生成的UUID仍然存在极低概率的重复风险。

sys_guid()

描述：生成Oracle的GUID序列号，类似UUID。此函数为Oracle兼容性函数。

返回类型：text

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT sys_guid();
              sys_guid
-----------------------------------
 4EBD3C74A17A11E9A1BF797FF7000001
(1 row)
```

说明

sys_guid函数内部生成原理同uuid_generate_v1函数。

UUID 函数应用示例

UUID全局唯一的特点，可以作为数据表生成主键，也可以作为数据表的分布列，uuid_generate_v1()作为数据表分布列的默认值时，通过Hash分布可以将数据均匀分布到各个DN上，防止数据倾斜。

说明

UUID的显著优点就是全局唯一，不需要中心节点，单个节点独立生成。但是也存在缺点，UUID较INT占用更多的存储空间，索引效率低，生成的ID随机，没有递增的特性，所以辨识困难。因此，在应用中，要根据实际情况选择UUID还是Sequence作为数据表主键。

示例如下：

1. INT类型作为分布列。

   创建示例哈希表mytable01，int类型作为分布列，插入数据后，查询数据存在数据倾斜。

   ```
   CREATE TABLE mytable01(a INT, b INT) DISTRIBUTE BY hash(a);
   CREATE TABLE
   INSERT INTO mytable01 VALUES(1, 10);
   INSERT 0 1
   INSERT INTO mytable01 VALUES(1, 10);
   INSERT 0 1
   INSERT INTO mytable01 VALUES(1, 10);
   INSERT 0 1
   INSERT INTO mytable01 VALUES(1, 10);
   INSERT 0 1
   INSERT INTO mytable01 VALUES(1, 10);
   INSERT 0 1
   SELECT * FROM mytable01;
    a | b
   ---+----
    1 | 10
    1 | 10
    1 | 10
    1 | 10
    1 | 10
   (5 rows)

   SELECT table_skewness('mytable01');
             table_skewness
   ----------------------------------------
    ("dn_6003_6004         ",5,100.000%)
    ("dn_6001_6002         ",0,0.000%)
    ("dn_6005_6006         ",0,0.000%)
   (3 rows)
   ```

2. UUID类型作为分布列。

   创建示例哈希表mytable02，UUID类型作为分布列，插入数据后，查询数据分布正常。

   ```
   CREATE TABLE mytable02 (id UUID default uuid_generate_v1(), a INT, b INT) DISTRIBUTE BY hash(id);
   CREATE TABLE
   ```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO mytable02(a, b) VALUES(1, 10);
INSERT 0 1
INSERT INTO mytable02(a, b) VALUES(1, 10);
INSERT 0 1
INSERT INTO mytable02(a, b) VALUES(1, 10);
INSERT 0 1
INSERT INTO mytable02(a, b) VALUES(1, 10);
INSERT 0 1
INSERT INTO mytable02(a, b) VALUES(1, 10);
INSERT 0 1

SELECT * FROM mytable02;
                  id                  | a | b
--------------------------------------+---+----
 63e45c14-cc74-0e00-e9aa-0a2c3fa0fffe | 1 | 10
 63e45c1f-4d18-0700-e9ab-0a2c3fa0fffe | 1 | 10
 63e45c26-f859-0b00-e9ad-0a2c3fa0fffe | 1 | 10
 63e45c23-9e5d-0300-e9ac-0a2c3fa0fffe | 1 | 10
 63e45c2a-5825-0600-e9ae-0a2c3fa0fffe | 1 | 10
(5 rows)

SELECT table_skewness('mytable02');
           table_skewness
-------------------------------------
 ("dn_6001_6002         ",3,60.000%)
 ("dn_6003_6004         ",2,40.000%)
 ("dn_6005_6006         ",0,0.000%)
(3 rows)
```

21.7.21 文本检索函数和操作符

21.7.21.1 文本检索操作符

@@

描述：tsvector类型的词汇与tsquery类型的词汇是否匹配

示例：

```
SELECT to_tsvector('fat cats ate rats') @@ to_tsquery('cat & rat') AS RESULT;
 result
--------
 t
(1 row)
```

@@@

描述：@@的同义词

示例：

```
SELECT to_tsvector('fat cats ate rats') @@@ to_tsquery('cat & rat') AS RESULT;
 result
--------
 t
(1 row)
```

&&

描述：将两个tsquery类型的词汇进行“与”操作

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT 'fat | rat'::tsquery && 'cat'::tsquery AS RESULT;
          result
---------------------------
 ( 'fat' | 'rat' ) & 'cat'
(1 row)
```

||

描述：将两个tsquery类型的词汇进行“或”操作

示例：

```
SELECT 'fat | rat'::tsquery || 'cat'::tsquery AS RESULT;
          result
---------------------------
 ( 'fat' | 'rat' ) | 'cat'
(1 row)
SELECT 'a:1 b:2'::tsvector || 'c:1 d:2 b:3'::tsvector AS RESULT;
          result
----------------------------
 'a':1 'b':2,5 'c':3 'd':4
(1 row)
```

!!

描述：tsquery类型词汇的非关系

示例：

```
SELECT !! 'cat'::tsquery AS RESULT;
 result
--------
 !'cat'
(1 row)
```

@>

描述：一个tsquery类型的词汇是否包含另一个tsquery类型的词汇

示例：

```
SELECT 'cat'::tsquery @> 'cat & rat'::tsquery AS RESULT;
 result
--------
 f
(1 row)
```

<@

描述：一个tsquery类型的词汇是否被包含另一个tsquery类型的词汇

示例：

```
SELECT 'cat'::tsquery <@ 'cat & rat'::tsquery AS RESULT;
 result
--------
 t
(1 row)
```

除了上述的操作符，还为tsvector类型和tsquery类型的数据定义了普通的B-tree比较操作符（=，<等）。

 版权所有 © 华为云计算技术有限公司

21.7.21.2 文本检索函数

get_current_ts_config()

描述：获取文本检索的默认配置。

返回类型：regconfig

示例：

```
SELECT get_current_ts_config();
 get_current_ts_config
-----------------------
 english
(1 row)
```

length(tsvector)

描述：tsvector类型词汇的单词数。

返回类型：integer

示例：

```
SELECT length('fat:2,4 cat:3 rat:5A'::tsvector);
 length
--------
      3
(1 row)
```

numnode(tsquery)

描述：tsquery类型的单词加上操作符的数量。

返回类型：integer

示例：

```
SELECT numnode('(fat & rat) | cat'::tsquery);
 numnode
---------
       5
(1 row)
```

plainto_tsquery([config regconfig ,] query text)

描述：产生tsquery类型的词汇，并忽略标点。

返回类型：tsquery

示例：

```
SELECT plainto_tsquery('english', 'The Fat Rats');
 plainto_tsquery
-----------------
 'fat' & 'rat'
(1 row)
```

querytree(query tsquery)

描述：获取tsquery类型的词汇可加索引的部分。

返回类型：text

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT querytree('foo & ! bar'::tsquery);
 querytree
-----------
 'foo'
(1 row)
```

setweight(tsvector, "char")

描述：给tsvector类型的每个元素分配权值。

返回类型：tsvector

示例：

```
SELECT setweight('fat:2,4 cat:3 rat:5B'::tsvector, 'A');
           setweight
-------------------------------
 'cat':3A 'fat':2A,4A 'rat':5A
(1 row)
```

strip(tsvector)

描述：删除tsvector类型单词中的position和权值。

返回类型：tsvector

示例：

```
SELECT strip('fat:2,4 cat:3 rat:5A'::tsvector);
       strip
--------------------
 'cat' 'fat' 'rat'
(1 row)
```

to_tsquery([config regconfig ,] query text)

描述：标准化单词，并转换为tsquery类型。

返回类型：tsquery

示例：

```
SELECT to_tsquery('english', 'The & Fat & Rats');
  to_tsquery
---------------
 'fat' & 'rat'
(1 row)
```

to_tsvector([config regconfig ,] document text)

描述：去除文件信息，并转换为tsvector类型。

返回类型：tsvector

示例：

```
SELECT to_tsvector('english', 'The Fat Rats');
   to_tsvector
-----------------
 'fat':2 'rat':3
(1 row)
```


版权所有 © 华为云计算技术有限公司

ts_headline([config regconfig,] document text, query tsquery [, options text])

描述：高亮显示查询的匹配项。

返回类型：text

示例：

```
SELECT ts_headline('x y z', 'z'::tsquery);
 ts_headline
--------------
 x y <b>z</b>
(1 row)
```

ts_rank([weights float4[],] vector tsvector, query tsquery [, normalization integer])

描述：文档查询排名。

返回类型：float4

示例：

```
SELECT ts_rank('hello world'::tsvector, 'world'::tsquery);
 ts_rank
----------
 .0607927
(1 row)
```

ts_rank_cd([weights float4[],] vector tsvector, query tsquery [, normalization integer])

描述：排序文件查询使用覆盖密度。

返回类型：float4

示例：

```
SELECT ts_rank_cd('hello world'::tsvector, 'world'::tsquery);
 ts_rank_cd
------------
          0
(1 row)
```

ts_rewrite(query tsquery, target tsquery, substitute tsquery)

描述：替换目标tsquery类型的单词。

返回类型：tsquery

示例：

```
SELECT ts_rewrite('a & b'::tsquery, 'a'::tsquery, 'foo|bar'::tsquery);
       ts_rewrite
-------------------------
 'b' & ( 'foo' | 'bar' )
(1 row)
```

ts_rewrite(query tsquery, select text)

描述：使用SELECT命令的结果替代目标中tsquery类型的单词。

 版权所有 © 华为云计算技术有限公司

返回类型：tsquery

示例：

```
SELECT ts_rewrite('world'::tsquery, 'select ''world''::tsquery, ''hello''::tsquery');
 ts_rewrite
------------
 'hello'
(1 row)
```

21.7.21.3 文本检索调试函数

ts_debug([config regconfig,] document text, OUT alias text, OUT description text, OUT token text, OUT dictionaries regdictionary[], OUT dictionary regdictionary, OUT lexemes text[])

描述：测试一个配置。

返回类型：setof record

示例：

```
SELECT ts_debug('english', 'The Brightest supernovaes');
                                   ts_debug
-------------------------------------------------------------------------------
 (asciiword,"Word, all ASCII",The,{english_stem},english_stem,{})
 (blank,"Space symbols"," ",{},,)
 (asciiword,"Word, all ASCII",Brightest,{english_stem},english_stem,{brightest})
 (blank,"Space symbols"," ",{},,)
 (asciiword,"Word, all ASCII",supernovaes,{english_stem},english_stem,{supernova})
(5 rows)
```

ts_lexize(dict regdictionary, token text)

描述：测试一个数据字典。

返回类型：text[]

示例：

```
SELECT ts_lexize('english_stem', 'stars');
 ts_lexize
-----------
 {star}
(1 row)
```

ts_parse(parser_name text, document text, OUT tokid integer, OUT token text)

描述：测试一个解析。

返回类型：setof record

示例：

```
SELECT ts_parse('default', 'foo - bar');
 ts_parse
-----------
 (1,foo)
 (12," ")
 (12,"- ")
 (1,bar)
(4 rows)
```


版权所有 © 华为云计算技术有限公司

ts_parse(parser_oid oid, document text, OUT tokid integer, OUT token text)

描述：测试一个解析。

返回类型：setof record

示例：

```
SELECT ts_parse(3722, 'foo - bar');
 ts_parse
-----------
 (1,foo)
 (12," ")
 (12,"- ")
 (1,bar)
(4 rows)
```

ts_token_type(parser_name text, OUT tokid integer, OUT alias text, OUT description text)

描述：获取分析器定义的记号类型。

返回类型：setof record

示例：

```
SELECT ts_token_type('default');
                        ts_token_type
---------------------------------------------------------------
 (1,asciiword,"Word, all ASCII")
 (2,word,"Word, all letters")
 (3,numword,"Word, letters and digits")
 (4,email,"Email address")
 (5,url,URL)
 (6,host,Host)
 (7,sfloat,"Scientific notation")
 (8,version,"Version number")
 (9,hword_numpart,"Hyphenated word part, letters and digits")
 (10,hword_part,"Hyphenated word part, all letters")
 (11,hword_asciipart,"Hyphenated word part, all ASCII")
 (12,blank,"Space symbols")
 (13,tag,"XML tag")
 (14,protocol,"Protocol head")
 (15,numhword,"Hyphenated word, letters and digits")
 (16,asciihword,"Hyphenated word, all ASCII")
 (17,hword,"Hyphenated word, all letters")
 (18,url_path,"URL path")
 (19,file,"File or path name")
 (20,float,"Decimal notation")
 (21,int,"Signed integer")
 (22,uint,"Unsigned integer")
 (23,entity,"XML entity")
(23 rows)
```

ts_token_type(parser_oid oid, OUT tokid integer, OUT alias text, OUT description text)

描述：获取分析器定义的记号类型。

返回类型：setof record

示例：

```
SELECT ts_token_type(3722);
                        ts_token_type
```


版权所有 © 华为云计算技术有限公司

```
----------------------------------------------------------------
 (1,asciiword,"Word, all ASCII")
 (2,word,"Word, all letters")
 (3,numword,"Word, letters and digits")
 (4,email,"Email address")
 (5,url,URL)
 (6,host,Host)
 (7,sfloat,"Scientific notation")
 (8,version,"Version number")
 (9,hword_numpart,"Hyphenated word part, letters and digits")
 (10,hword_part,"Hyphenated word part, all letters")
 (11,hword_asciipart,"Hyphenated word part, all ASCII")
 (12,blank,"Space symbols")
 (13,tag,"XML tag")
 (14,protocol,"Protocol head")
 (15,numhword,"Hyphenated word, letters and digits")
 (16,asciihword,"Hyphenated word, all ASCII")
 (17,hword,"Hyphenated word, all letters")
 (18,url_path,"URL path")
 (19,file,"File or path name")
 (20,float,"Decimal notation")
 (21,int,"Signed integer")
 (22,uint,"Unsigned integer")
 (23,entity,"XML entity")
(23 rows)
```

ts_stat(sqlquery text, [weights text,] OUT word text, OUT ndoc integer, OUT nentry integer)

描述：获取tsvector列的统计数据。

返回类型：setof record

示例：

```
SELECT ts_stat('select ''hello world''::tsvector');
  ts_stat
-------------
 (world,1,1)
 (hello,1,1)
(2 rows)
```

21.7.22 HLL 函数和操作符

21.7.22.1 HLL 操作符

HLL类型支持如下操作符：

版权所有 © 华为云计算技术有限公司

表 21-48 HLL 操作符

HLL操作符	描述	返回值类型	示例
=	比较hll或hll_hashval的值是否相等。	bool	• hll SELECT (hll_empty() || hll_hash_integer(1)) = (hll_empty() || hll_hash_integer(1)); ?column? ---------- t (1 row) • hll_hashval SELECT hll_hash_integer(1) = hll_hash_integer(1); ?column? ---------- t (1 row)
<> or !=	比较hll或hll_hashval是否不相等。	bool	• hll SELECT (hll_empty() || hll_hash_integer(1)) <> (hll_empty() || hll_hash_integer(2)); ?column? ---------- t (1 row) • hll_hashval SELECT hll_hash_integer(1) <> hll_hash_integer(2); ?column? ---------- t (1 row)
||	可代表hll_add，hll_union，hll_add_rev三个函数的功能。	hll	• hll_add SELECT hll_empty() || hll_hash_integer(1); ?column? -------------------------- \x128b7f8895a3f5af28cafe (1 row) • hll_add_rev SELECT hll_hash_integer(1) || hll_empty(); ?column? -------------------------- \x128b7f8895a3f5af28cafe (1 row) • hll_union SELECT (hll_empty() || hll_hash_integer(1)) || (hll_empty() || hll_hash_integer(2)); ?column? -- \x128b7f8895a3f5af28cafeda0ce907e4355b60 (1 row)
#	计算出hll的distinct值，同hll_cardinality函数。	integer	SELECT #(hll_empty() || hll_hash_integer(1)); ?column? ---------- 1 (1 row)

版权所有 © 华为云计算技术有限公司

21.7.22.2 哈希函数

hll_hash_boolean(bool)

描述：对bool类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_boolean(FALSE);
 hll_hash_boolean
--------------------
 5048724184180415669
(1 row)
```

hll_hash_boolean(bool, int32)

描述：设置hash seed（即改变哈希策略）并对bool类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_boolean(FALSE, 10);
 hll_hash_boolean
--------------------
 391264977436098630
(1 row)
```

hll_hash_smallint(smallint)

描述：对smallint类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_smallint(100::smallint);
 hll_hash_smallint
--------------------
 4631102026669432776
(1 row)
```

说明

数值大小相同的参数使用不同数据类型的哈希函数计算，最后结果会不一样，因为不同类型哈希函数会选取不同的哈希计算策略。

hll_hash_smallint(smallint, int32)

描述：设置hash seed（即改变哈希策略）同时对smallint类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_smallint(100::smallint, 10);
 hll_hash_smallint
--------------------
 8349353095166695771
(1 row)
```


版权所有 © 华为云计算技术有限公司

hll_hash_integer(integer)

描述：对integer类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_integer(0);
  hll_hash_integer
----------------------
 -3485513579396041028
(1 row)
```

hll_hash_integer(integer, int32)

描述：对integer类型数据计算哈希值，并设置hashseed（即改变哈希策略）。

返回值类型：hll_hashval

示例：

```
 SELECT hll_hash_integer(0, 10);
  hll_hash_integer
--------------------
 183371090322255134
(1 row)
```

hll_hash_bigint(bigint)

描述：对bigint类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_bigint(100::bigint);
  hll_hash_bigint
---------------------
 8349353095166695771
(1 row)
```

hll_hash_bigint(bigint, int32)

描述：对bigint类型数据计算哈希值，并设置hashseed（即改变哈希策略）。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_bigint(100::bigint, 10);
  hll_hash_bigint
---------------------
 4631120266694327276
(1 row)
```

hll_hash_bytea(bytea)

描述：对bytea类型数据计算哈希值。

返回值类型：hll_hashval

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT hll_hash_bytea(E'\\x');
 hll_hash_bytea
----------------
 0
(1 row)
```

hll_hash_bytea(bytea, int32)

描述：对bytea类型数据计算哈希值，并设置hashseed（即改变哈希策略）。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_bytea(E'\\x', 10);
   hll_hash_bytea
---------------------
 6574525721897061910
(1 row)
```

hll_hash_text(text)

描述：对text类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_text('AB');
    hll_hash_text
---------------------
 5365230931951287672
(1 row)
```

hll_hash_text(text, int32)

描述：对text类型数据计算哈希值，并设置hashseed（即改变哈希策略）。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_text('AB', 10);
    hll_hash_text
---------------------
 7680762839921155903
(1 row)
```

hll_hash_any(anytype)

描述：对任意类型数据计算哈希值。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_any(1);
     hll_hash_any
----------------------
 -8604791237420463362
(1 row)

SELECT hll_hash_any('08:00:2b:01:02:03'::macaddr);
     hll_hash_any
```

 版权所有 © 华为云计算技术有限公司

```
----------------------
 -4883882473551067169
(1 row)
```

hll_hash_any(anytype, int32)

描述：对任意类型数据计算哈希值，并设置hashseed（即改变哈希策略）。

返回值类型：hll_hashval

示例：

```
SELECT hll_hash_any(1, 10);
    hll_hash_any
----------------------
 -1478847531811254870
(1 row)
```

hll_hashval_eq(hll_hashval, hll_hashval)

描述：比较两个hll_hashval类型数据是否相等。

返回值类型：bool

示例：

```
SELECT hll_hashval_eq(hll_hash_integer(1), hll_hash_integer(1));
 hll_hashval_eq
----------------
 t
(1 row)
```

hll_hashval_ne(hll_hashval, hll_hashval)

描述：比较两个hll_hashval类型数据是否不相等。

返回值类型：bool

示例：

```
SELECT hll_hashval_ne(hll_hash_integer(1), hll_hash_integer(1));
 hll_hashval_ne
----------------
 f
(1 row)
```

21.7.22.3 精度函数

HLL（HyperLogLog）主要存在三种模式Explicit，Sparse，Full。当数据规模比较小的时候会使用Explicit模式和Sparse模式， 这两种模式在计算结果上基本上没有误差。随着distinct值越来越多，就会转换成Full模式，但结果也会存在一定误差。下列函数用于查看HLL中精度参数。

hll_schema_version(hll)

描述：查看当前hll中的schema version。

返回值类型：integer

示例：

```
SELECT hll_schema_version(hll_empty());
 hll_schema_version
```

 版权所有 © 华为云计算技术有限公司

```
--------------------
         1
(1 row)
```

hll_type(hll)

描述：查看当前hll的类型。

返回值类型：integer

示例：

```
SELECT hll_type(hll_empty());
 hll_type
----------
       1
(1 row)
```

hll_log2m(hll)

描述：查看当前hll的log2m数值，此值会影响最后hll计算distinct误差率，误差率计算公式为：

$$\pm 1.04/\sqrt{2 \wedge log2m}$$

返回值类型：integer

示例：

```
SELECT hll_log2m(hll_empty());
 hll_log2m
-----------
       11
(1 row)
```

hll_regwidth(hll)

描述：查看hll数据结构中桶的位数大小。

返回值类型：integer

示例：

```
SELECT hll_regwidth(hll_empty());
 hll_regwidth
--------------
          5
(1 row)
```

hll_expthresh(hll)

描述：得到当前hll中expthresh大小，hll通常会由Explicit模式到Sparse模式再到Full模式，这个过程称为promotion hierarchy策略。可以通过调整expthresh值的大小改变策略，比如expthresh为0的时候就会跳过Explicit模式而直接进入Sparse模式。当显式指定expthresh的取值为1-7之间时，该函数得到的是 $2^{expthresh}$。

返回值类型：record

示例：

```
SELECT hll_expthresh(hll_empty());
 hll_expthresh
```

 版权所有 © 华为云计算技术有限公司

```
---------------
 (-1,160)
(1 row)

SELECT hll_expthresh(hll_empty(11,5,3));
 hll_expthresh
---------------
 (8,8)
(1 row)
```

hll_sparseon(hll)

描述：是否启用sparse模式，0是关闭，1是开启。

返回值类型：integer

示例：

```
SELECT hll_sparseon(hll_empty());
 hll_sparseon
--------------
            1
(1 row)
```

21.7.22.4 聚合函数

hll_add_agg(hll_hashval)

描述：把哈希后的数据按照分组放到hll中。

返回值类型：hll

示例：

1. 准备数据。
   ```
   CREATE TABLE t_id(id int);
   INSERT INTO t_id VALUES(generate_series(1,500));
   CREATE TABLE t_data(a int, c text);
   INSERT INTO t_data SELECT mod(id,2), id FROM t_id;
   ```
2. 创建表并指定列为hll。
   ```
   CREATE TABLE t_a_c_hll(a int, c hll);
   ```
3. 根据a列group by对数据分组，把各组数据加到hll中。
   ```
   INSERT INTO t_a_c_hll SELECT a, hll_add_agg(hll_hash_text(c)) FROM t_data GROUP BY a;
   ```
4. 得到每组数据中hll的Distinct值。
   ```
   SELECT a, #c as cardinality FROM t_a_c_hll order by a;
    a |   cardinality
   ---+------------------
    0 | 250.741759091658
    1 | 250.741759091658
   (2 rows)
   ```

hll_add_agg(hll_hashval, int32 log2m)

描述：把哈希后的数据按照分组放到hll中。并指定参数log2m，取值范围为10~16。

返回值类型：hll

示例：

```
SELECT hll_cardinality(hll_add_agg(hll_hash_text(c), 10)) FROM t_data;
 hll_cardinality
------------------
```


版权所有 © 华为云计算技术有限公司

```
 503.932348927339
(1 row)
```

hll_add_agg(hll_hashval, int32 log2m, int32 regwidth)

描述：把哈希后的数据按照分组放到hll中。依次指定参数log2m，regwidth。regwidth取值范围为1~5。

返回值类型：hll

示例：

```
SELECT hll_cardinality(hll_add_agg(hll_hash_text(c), NULL, 1)) FROM t_data;
 hll_cardinality
------------------
 496.628982624022
(1 row)
```

hll_add_agg(hll_hashval, int32 log2m, int32 regwidth, int64 expthresh)

描述：把哈希后的数据按照分组放到hll中，依次指定参数log2m、regwidth、expthresh。expthresh的取值范围是-1~7之间的整数，该参数可以用来设置从Explicit模式到Sparse模式的阈值大小。-1表示自动模式，0表示跳过Explicit模式，取1~7表示在基数到达 $2^{expthresh}$时切换模式。

返回值类型：hll

示例：

```
 SELECT hll_cardinality(hll_add_agg(hll_hash_text(c), NULL, 1, 4)) FROM t_data;
 hll_cardinality
------------------
 496.628982624022
(1 row)
```

hll_add_agg(hll_hashval, int32 log2m, int32 regwidth, int64 expthresh, int32 sparseon)

描述：把哈希后的数据按照分组放到hll中，依次指定参数log2m、regwidth、expthresh、sparseon。sparseon取值范围为0或者1。

返回值类型：hll

示例：

```
 SELECT hll_cardinality(hll_add_agg(hll_hash_text(c), NULL, 1, 4, 0)) FROM t_data;
 hll_cardinality
------------------
 496.628982624022
(1 row)
```

hll_union_agg(hll)

描述：将多个hll类型数据union成一个hll。

返回值类型：hll

示例：

将各组中的hll数据union成一个hll，并计算distinct值。

```
SELECT #hll_union_agg(c) as cardinality FROM t_a_c_hll;
  cardinality
```

 版权所有 © 华为云计算技术有限公司

```
------------------
 496.628982624022
(1 row)
```

说明

当两个或者多个hll数据结构执行union时，必须要保证其中每一个hll里面的精度参数一样，否则不能进行union。同样的约束也适用于函数hll_union(hll,hll)。

21.7.22.5 功能函数

hll_print(hll)

描述：打印hll的一些debug参数信息。

返回值类型：cstring

示例：

```
SELECT hll_print(hll_empty());
                          hll_print
------------------------------------------------------------
 EMPTY, nregs=2048, nbits=5, expthresh=-1(160), sparseon=1
(1 row)
```

hll_empty()

描述：创建一个空的hll。

返回值类型：hll

示例：

```
SELECT hll_empty();
 hll_empty
-----------
 \x118b7f
(1 row)
```

hll_empty(int32 log2m)

描述：创建空的hll并指定参数log2m，取值范围是10到16。

返回值类型: hll

示例：

```
SELECT hll_empty(10);
 hll_empty
-----------
 \x118a7f
(1 row)
```

hll_empty(int32 log2m, int32 regwidth)

描述：创建空的hll并依次指定参数log2m、regwidth。regwidth取值范围是1到5。

返回值类型: hll

示例：

```
SELECT hll_empty(10, 4);
 hll_empty
```

 版权所有 © 华为云计算技术有限公司

```
-----------
 \x116a7f
(1 row)
```

hll_empty(int32 log2m, int32 regwidth, int64 expthresh)

描述：创建空的hll并依次指定参数log2m、regwidth、expthresh。expthresh取值范围是-1到7之间的整数。该参数可以用来设置从Explicit模式到Sparse模式的阈值大小。-1表示自动模式，0表示跳过Explicit模式，取1-7表示在基数到达$2^{expthresh}$时切换模式。

返回值类型：hll

示例：

```
SELECT hll_empty(10, 4, 7);
 hll_empty
-----------
 \x116a48
(1 row)
```

hll_empty(int32 log2m, int32 regwidth, int64 expthresh, int32 sparseon)

描述：创建空的hll并依次指定参数log2m、regwidth、expthresh、sparseon。sparseon取0或者1。

返回值类型：hll

示例：

```
SELECT hll_empty(10,4,7,0);
 hll_empty
-----------
 \x116a08
(1 row)
```

hll_add(hll, hll_hashval)

描述：把hll_hashval加入到hll中。

返回值类型：hll

示例：

```
SELECT hll_add(hll_empty(), hll_hash_integer(1));
        hll_add
--------------------------
 \x128b7f8895a3f5af28cafe
(1 row)
```

hll_add_rev(hll_hashval, hll)

描述：把hll_hashval加入到hll中，和hll_add功能一样，只是参数位置进行了交换。

返回值类型：hll

示例：

```
SELECT hll_add_rev(hll_hash_integer(1), hll_empty());
      hll_add_rev
--------------------------
 \x128b7f8895a3f5af28cafe
(1 row)
```

 版权所有 © 华为云计算技术有限公司

hll_eq(hll, hll)

描述：比较两个hll是否相等。

返回值类型：bool

示例：

```
SELECT hll_eq(hll_add(hll_empty(), hll_hash_integer(1)), hll_add(hll_empty(), hll_hash_integer(2)));
 hll_eq
--------
 f
(1 row)
```

hll_ne(hll, hll)

描述：比较两个hll是否不相等。

返回值类型：bool

示例：

```
SELECT hll_ne(hll_add(hll_empty(), hll_hash_integer(1)), hll_add(hll_empty(), hll_hash_integer(2)));
 hll_ne
--------
 t
(1 row)
```

hll_cardinality(hll)

描述：计算hll的distinct值。

返回值类型：integer

示例：

```
SELECT hll_cardinality(hll_empty() || hll_hash_integer(1));
 hll_cardinality
-----------------
               1
(1 row)
```

hll_union(hll, hll)

描述：把两个hll数据结构union成一个。

返回值类型：hll

示例：

```
SELECT hll_union(hll_add(hll_empty(), hll_hash_integer(1)), hll_add(hll_empty(), hll_hash_integer(2)));
                hll_union
-----------------------------------------
 \x128b7f8895a3f5af28cafeda0ce907e4355b60
(1 row)
```

21.7.22.6 内置函数

HLL（HyperLogLog）有一系列内置函数用于内部对数据进行处理，一般情况下不建议用户使用。

版权所有 © 华为云计算技术有限公司

表 21-49 内置函数

函数名称	功能描述
hll_in	以string格式接收hll数据。
hll_out	以string格式发送hll数据。
hll_recv	以bytea格式接收hll数据。
hll_send	以bytea格式发送hll数据。
hll_trans_in	以string格式接收hll_trans_type数据。
hll_trans_out	以string格式发送hll_trans_type数据。
hll_trans_recv	以bytea形式接收hll_trans_type数据。
hll_trans_send	以bytea形式发送hll_trans_type数据。
hll_typmod_in	接收typmod类型数据。
hll_typmod_out	发送typmod类型数据。
hll_hashval_in	接收hll_hashval类型数据。
hll_hashval_out	发送hll_hashval类型数据。
hll_add_trans0	类似于hll_add所提供的功能，通常在分布式聚合运算的第一阶段DN上使用。
hll_union_trans	类似hll_union所提供的功能，在分布式聚合运算的第一阶段DN上使用。
hll_union_collect	类似于hll_union所提供的功能，在分布式聚合运算第二阶段CN上使用，汇总各个DN上的结果。
hll_pack	在分布式聚合运算第三阶段CN上使用，把自定义hll_trans_type类型最后转换成hll类型。
hll	用于hll类型转换成hll类型，根据输入参数会设定指定参数。
hll_hashval	用于bigint类型转换成hll_hashval类型。
hll_hashval_int4	用于int4类型转换成hll_hashval类型。

21.7.23 返回集合的函数

21.7.23.1 序列号生成函数

generate_series()函数根据指定的开始值(start)、结束值(stop)和步长(step)返回一个基于系列的集合。

generate_series()函数的入参中，当step是正数且start大于stop，则返回零行。相反，当step是负数且start小于stop，则返回零行。如果任何输入为NULL也会返回零行。如果step为零则会报错。

版权所有 © 华为云计算技术有限公司

generate_series(start, stop)

描述：生成一个数值序列，从start到stop，步长默认为1。

参数类型：int、bigint、numeric

返回值类型：setof int、setof bigint、setof numeric（与参数类型相同）

示例：

```
SELECT * FROM generate_series(2,4);
 generate_series
-----------------
               2
               3
               4
(3 rows)

SELECT * FROM generate_series(4,3);
 generate_series
-----------------
(0 rows)

SELECT * FROM generate_series(1,NULL);
 generate_series
-----------------
(0 rows)
```

generate_series(start, stop, step)

描述：生成一个数值序列，从start到stop，步长为step。

参数类型：int、bigint、numeric

返回值类型：setof int、setof bigint、setof numeric（与参数类型相同）

示例：

```
SELECT * FROM generate_series(5,1,-2);
 generate_series
-----------------
               5
               3
               1
(3 rows)

SELECT * FROM generate_series(4,6,-5);
 generate_series
-----------------
(0 rows)

SELECT * FROM generate_series(4,3,0);
ERROR:  step size cannot equal zero
```

generate_series(start, stop, step interval)

描述：生成一个数值序列，从start到stop，步长为step。

参数类型：timestamp或timestamp with time zone

返回值类型：setof timestamp或setof timestamp with time zone（与参数类型相同）

示例：

 版权所有 © 华为云计算技术有限公司

```
--这个示例应用于date-plus-integer操作符。
SELECT current_date + s.a AS dates FROM generate_series(0,14,7) AS s(a);
   dates
------------
 2017-06-02
 2017-06-09
 2017-06-16
(3 rows)

SELECT * FROM generate_series('2008-03-01 00:00'::timestamp, '2008-03-04 12:00', '10 hours');
   generate_series
---------------------
 2008-03-01 00:00:00
 2008-03-01 10:00:00
 2008-03-01 20:00:00
 2008-03-02 06:00:00
 2008-03-02 16:00:00
 2008-03-03 02:00:00
 2008-03-03 12:00:00
 2008-03-03 22:00:00
 2008-03-04 08:00:00
(9 rows)
```

21.7.23.2 下标生成函数

generate_subscripts(array anyarray, dim int)

描述：生成一系列包括给定数组的下标。

返回值类型：setof int

示例：

```
SELECT generate_subscripts('{NULL,1,NULL,2}'::int[], 1) AS s;
 s
---
 1
 2
 3
 4
(4 rows)
```

generate_subscripts(array anyarray, dim int, reverse boolean)

描述：生成一系列包括给定数组的下标。当reverse为真时，该系列则以相反的顺序返回。

返回值类型：setof int

示例：

```
SELECT generate_subscripts('{NULL,1,NULL,2}'::int[], 1,TRUE) AS s;
 s
---
 4
 3
 2
 1
(4 rows)
```

generate_subscripts是为给定数组的指定维度生成有效下标集的函数。如果数组中没有所请求的维度或者任何输入NULL数组，均返回零行（但是会给数组元素为空的返回有效下标）。示例：

```
--unnest一个2D数组。
CREATE OR REPLACE FUNCTION unnest2(anyarray)
```

 版权所有 © 华为云计算技术有限公司

```
RETURNS SETOF anyelement AS $$
SELECT $1[i][j]
   FROM generate_subscripts($1,1) g1(i),
        generate_subscripts($1,2) g2(j);
$$ LANGUAGE sql IMMUTABLE;

SELECT * FROM unnest2(ARRAY[[1,2],[3,4]]);
 unnest2
---------
       1
       2
       3
       4
(4 rows)
```

21.7.24 几何函数和操作符

21.7.24.1 几何操作符

+

描述：平移，即从第一个参数的每个点的坐标中加上第二个point的坐标。

示例：

```
SELECT box '((0,0),(1,1))' + point '(2.0,0)' AS RESULT;
   result
-------------
 (3,1),(2,0)
(1 row)
```

-

描述：平移，从第一个参数的每个点的坐标中减去第二个point的坐标。

示例：

```
SELECT box '((0,0),(1,1))' - point '(2.0,0)' AS RESULT;
    result
---------------
 (-1,1),(-2,0)
(1 row)
```

*****

描述：伸展/旋转。

示例：

```
SELECT box '((0,0),(1,1))' * point '(2.0,0)' AS RESULT;
   result
-------------
 (2,2),(0,0)
(1 row)
```

/

描述：收缩/旋转。

示例：

```
SELECT box '((0,0),(2,2))' / point '(2.0,0)' AS RESULT;
   result
```


版权所有 © 华为云计算技术有限公司

```
-------------
 (1,1),(0,0)
(1 row)
```

#

描述：两个图形交点或者交面。

示例：

```
SELECT box'((1,-1),(-1,1))' # box'((1,1),(-1,-1))' AS RESULT;
 result
--------
(1,1),(-1,-1)
(1 row)
```

#

描述：图形的路径数目或多边形顶点数。

示例：

```
SELECT # path'((1,0),(0,1),(-1,0))' AS RESULT;
 result
--------
      3
(1 row)
```

@-@

描述：图形的长度或者周长。

示例：

```
SELECT @-@ path '((0,0),(1,0))' AS RESULT;
 result
--------
      2
(1 row)
```

@@

描述：图形的中心。

示例：

```
SELECT @@ circle '((0,0),10)' AS RESULT;
 result
--------
 (0,0)
(1 row)
```

##

描述：第一个图形相对第二个图形的最近点。

示例：

```
SELECT point '(0,0)' ## box '((2,0),(0,2))' AS RESULT;
 result
--------
 (0,0)
(1 row)
```


版权所有 © 华为云计算技术有限公司

<->

描述：两个图形之间的距离。

示例：

```
SELECT circle '((0,0),1)' <-> circle '((5,0),1)' AS RESULT;
 result
--------
      3
(1 row)
```

&&

描述：两个图形是否重叠（有一个共同点就为真）。

示例：

```
SELECT box '((0,0),(1,1))' && box '((0,0),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

<<

描述：图形是否全部在另一个图形的左边（没有相同的横坐标）。

示例：

```
SELECT circle '((0,0),1)' << circle '((5,0),1)' AS RESULT;
 result
--------
 t
(1 row)
```

>>

描述：图形是否全部在另一个图形的右边（没有相同的横坐标）。

示例：

```
SELECT circle '((5,0),1)' >> circle '((0,0),1)' AS RESULT;
 result
--------
 t
(1 row)
```

&<

描述：图形的最右边是否不超过在另一个图形的最右边。

示例：

```
SELECT box '((0,0),(1,1))' &< box '((0,0),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

&>

描述：图形的最左边是否不超过在另一个图形的最左边。

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT box '((0,0),(3,3))' &> box '((0,0),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

<<|

描述：图形是否全部在另一个图形的下边（没有相同的纵坐标）。

示例：

```
SELECT box '((0,0),(3,3))' <<| box '((3,4),(5,5))' AS RESULT;
 result
--------
 t
(1 row)
```

|>>

描述：图形是否全部在另一个图形的上边（没有相同的纵坐标）。

示例：

```
SELECT box '((3,4),(5,5))' |>> box '((0,0),(3,3))' AS RESULT;
 result
--------
 t
(1 row)
```

&<|

描述：图形的最上边是否不超过另一个图形的最上边。

示例：

```
SELECT box '((0,0),(1,1))' &<| box '((0,0),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

|&>

描述：图形的最下边是否不超过另一个图形的最下边。

示例：

```
SELECT box '((0,0),(3,3))' |&> box '((0,0),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

<^

描述：图形是否低于另一个图形（允许两个图形有接触）。

示例：

```
SELECT box '((0,0),(-3,-3))' <^ box '((0,0),(2,2))' AS RESULT;
 result
```

 版权所有 © 华为云计算技术有限公司

```
--------
 t
(1 row)
```

>^

描述：图形是否高于另一个图形（允许两个图形有接触）。

示例：

```
SELECT box '((0,0),(2,2))' >^ box '((0,0),(-3,-3))'  AS RESULT;
 result
--------
 t
(1 row)
```

?#

描述：两个图形是否相交。

示例：

```
SELECT lseg '((-1,0),(1,0))' ?# box '((-2,-2),(2,2))' AS RESULT;
 result
--------
 t
(1 row)
```

?-

描述：图形是否处于水平位置。

示例：

```
SELECT ?- lseg '((-1,0),(1,0))' AS RESULT;
 result
--------
 t
(1 row)
```

?-

描述：图形是否水平对齐。

示例：

```
SELECT point '(1,0)' ?- point '(0,0)' AS RESULT;
 result
--------
 t
(1 row)
```

?|

描述：图形是否处于竖直位置。

示例：

```
SELECT ?| lseg '((-1,0),(1,0))' AS RESULT;
 result
--------
 f
(1 row)
```

 版权所有 © 华为云计算技术有限公司

?|

描述：图形是否竖直对齐。

示例：

```
SELECT point '(0,1)' ?| point '(0,0)' AS RESULT;
 result
--------
 t
(1 row)
```

?-|

描述：两条线是否垂直。

示例：

```
SELECT lseg '((0,0),(0,1))' ?-| lseg '((0,0),(1,0))' AS RESULT;
 result
--------
 t
(1 row)
```

?||

描述：两条线是否平行。

示例：

```
SELECT lseg '((-1,0),(1,0))' ?|| lseg '((-1,2),(1,2))' AS RESULT;
 result
--------
 t
(1 row)
```

@>

描述：图形是否包含另一个图形。

示例：

```
SELECT circle '((0,0),2)' @> point '(1,1)' AS RESULT;
 result
--------
 t
(1 row)
```

<@

描述：图形是否被包含于另一个图形。

示例：

```
SELECT point '(1,1)' <@ circle '((0,0),2)' AS RESULT;
 result
--------
 t
(1 row)
```

~=

描述：两个图形是否相同?

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT polygon '((0,0),(1,1))' ~= polygon '((1,1),(0,0))' AS RESULT;
 result
--------
 t
(1 row)
```

21.7.24.2 几何函数

area(object)

描述：计算图形的面积。

返回类型：double precision

示例：

```
SELECT area(box '((0,0),(1,1))') AS RESULT;
 result
--------
      1
(1 row)
```

center(object)

描述：计算图形的中心。

返回类型：point

示例：

```
SELECT center(box '((0,0),(1,2))') AS RESULT;
 result
---------
 (0.5,1)
(1 row)
```

diameter(circle)

描述：计算圆的直径。

返回类型：double precision

示例：

```
SELECT diameter(circle '((0,0),2.0)') AS RESULT;
 result
--------
      4
(1 row)
```

height(box)

描述：矩形的竖直高度。

返回类型：double precision

示例：

```
SELECT height(box '((0,0),(1,1))') AS RESULT;
 result
--------
```


版权所有 © 华为云计算技术有限公司

```
   1
(1 row)
```

isclosed(path)

描述：图形是否为闭合路径。

返回类型：boolean

示例：

```
SELECT isclosed(path '((0,0),(1,1),(2,0))') AS RESULT;
 result
--------
 t
(1 row)
```

isopen(path)

描述：图形是否为开放路径。

返回类型：boolean

示例：

```
SELECT isopen(path '[(0,0),(1,1),(2,0)]') AS RESULT;
 result
--------
 t
(1 row)
```

length(object)

描述：计算图形的长度。

返回类型：double precision

示例：

```
SELECT length(path '((-1,0),(1,0))') AS RESULT;
 result
--------
   4
(1 row)
```

npoints(path)

描述：计算路径的顶点数。

返回类型：int

示例：

```
SELECT npoints(path '[(0,0),(1,1),(2,0)]') AS RESULT;
 result
--------
   3
(1 row)
```

npoints(polygon)

描述：计算多边形的顶点数。

 版权所有 © 华为云计算技术有限公司

返回类型：int

示例：

```
SELECT npoints(polygon '((1,1),(0,0))') AS RESULT;
 result
--------
      2
(1 row)
```

pclose(path)

描述：把路径转换为闭合路径。

返回类型：path

示例：

```
SELECT pclose(path '[(0,0),(1,1),(2,0)]') AS RESULT;
       result
---------------------
 ((0,0),(1,1),(2,0))
(1 row)
```

popen(path)

描述：把路径转换为开放路径。

返回类型：path

示例：

```
SELECT popen(path '((0,0),(1,1),(2,0))') AS RESULT;
       result
---------------------
 [(0,0),(1,1),(2,0)]
(1 row)
```

radius(circle)

描述：计算圆的半径。

返回类型：double precision

示例：

```
SELECT radius(circle '((0,0),2.0)') AS RESULT;
 result
--------
      2
(1 row)
```

width(box)

描述：计算矩形的水平尺寸。

返回类型：double precision

示例：

```
SELECT width(box '((0,0),(1,1))') AS RESULT;
 result
--------
```

 版权所有 © 华为云计算技术有限公司

```
     1
(1 row)
```

21.7.24.3 几何类型转换函数

box(circle)

描述：将圆转换成矩形。

返回类型：box

示例：

```
SELECT box(circle '((0,0),2.0)') AS RESULT;
                                result
-----------------------------------------------------------------------------
 (1.41421356237309,1.41421356237309),(-1.41421356237309,-1.41421356237309)
(1 row)
```

box(point, point)

描述：将点转换成矩形。

返回类型：box

示例：

```
SELECT box(point '(0,0)', point '(1,1)') AS RESULT;
  result
-------------
 (1,1),(0,0)
(1 row)
```

box(polygon)

描述：将多边形转换成矩形。

返回类型：box

示例：

```
SELECT box(polygon '((0,0),(1,1),(2,0))') AS RESULT;
  result
-------------
 (2,1),(0,0)
(1 row)
```

circle(box)

描述：矩形转换成圆。

返回类型：circle

示例：

```
SELECT circle(box '((0,0),(1,1))') AS RESULT;
            result
--------------------------------
 <(0.5,0.5),0.707106781186548>
(1 row)
```


版权所有 © 华为云计算技术有限公司

circle(point, double precision)

描述：将圆心和半径转换成圆。

返回类型：circle

示例：

```
SELECT circle(point '(0,0)', 2.0) AS RESULT;
  result
----------
 <(0,0),2>
(1 row)
```

circle(polygon)

描述：将多边形转换成圆。

返回类型：circle

示例：

```
SELECT circle(polygon '((0,0),(1,1),(2,0))') AS RESULT;
                  result
-----------------------------------------------
 <(1,0.333333333333333),0.924950591148529>
(1 row)
```

lseg(box)

描述：矩形对角线转化成线段。

返回类型：lseg

示例：

```
SELECT lseg(box '((-1,0),(1,0))') AS RESULT;
    result
----------------
 [(1,0),(-1,0)]
(1 row)
```

lseg(point, point)

描述：点转换成线段。

返回类型：lseg

示例：

```
SELECT lseg(point '(-1,0)', point '(1,0)') AS RESULT;
    result
----------------
 [(-1,0),(1,0)]
(1 row)
```

path(polygon)

描述：多边形转换成路径。

返回类型：path

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT path(polygon '((0,0),(1,1),(2,0))') AS RESULT;
      result
--------------------
 ((0,0),(1,1),(2,0))
(1 row)
```

point(double precision, double precision)

描述：结点。

返回类型：point

示例：

```
SELECT point(23.4, -44.5) AS RESULT;
   result
--------------
 (23.4,-44.5)
(1 row)
```

point(box)

描述：矩形的中心。

返回类型：point

示例：

```
SELECT point(box '((-1,0),(1,0))') AS RESULT;
 result
--------
 (0,0)
(1 row)
```

point(circle)

描述：圆心。

返回类型：point

示例：

```
SELECT point(circle '((0,0),2.0)') AS RESULT;
 result
--------
 (0,0)
(1 row)
```

point(lseg)

描述：线段的中心。

返回类型：point

示例：

```
SELECT point(lseg '((-1,0),(1,0))') AS RESULT;
 result
--------
 (0,0)
(1 row)
```

 版权所有 © 华为云计算技术有限公司

point(polygon)

描述：多边形的中心。

返回类型：point

示例：

```
SELECT point(polygon '((0,0),(1,1),(2,0))') AS RESULT;
       result
----------------------
 (1,0.333333333333333)
(1 row)
```

polygon(box)

描述：矩形转换成4点多边形。

返回类型：polygon

示例：

```
SELECT polygon(box '((0,0),(1,1))') AS RESULT;
         result
----------------------------
 ((0,0),(0,1),(1,1),(1,0))
(1 row)
```

polygon(circle)

描述：圆转换成12点多边形。

返回类型：polygon

示例：

```
SELECT polygon(circle '((0,0),2.0)') AS RESULT;

result

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 ((-2,0),(-1.73205080756888,1),(-1,1.73205080756888),(-1.22464679914735e-16,2),(1,1.73205080756888),
(1.73205080756888,1),(2,2.44929359829471e-16),(1.73205080756888,-0.999999999999999),
(1,-1.73205080756888),(3.67394039744206e-16,-2),(-0.999999999999999,-1.73205080756888),
(-1.73205080756888,-1))
(1 row)
```

polygon(npts, circle)

描述：圆转换成npts点多边形。

返回类型：polygon

示例：

```
SELECT polygon(12, circle '((0,0),2.0)') AS RESULT;

result

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 ((-2,0),(-1.73205080756888,1),(-1,1.73205080756888),(-1.22464679914735e-16,2),(1,1.73205080756888),
```

 版权所有 © 华为云计算技术有限公司

```
(1.73205080756888,1),(2,2.44929359829471e-16),(1.73205080756888,-0.999999999999999),
(1,-1.73205080756888),(3.67394039744206e-16,-2),(-0.999999999999999,-1.73205080756888),
(-1.73205080756888,-1))
(1 row)
```

polygon(path)

描述：路径转换成多边形。

返回类型：polygon

示例：

```
SELECT polygon(path '((0,0),(1,1),(2,0))') AS RESULT;
      result
---------------------
 ((0,0),(1,1),(2,0))
(1 row)
```

21.7.25 网络地址函数和操作符

21.7.25.1 cidr 和 inet 操作符

操作符<<，<<=，>>，>>=对子网包含进行测试。它们只考虑两个地址的网络部分（忽略任何主机部分），然后判断其中一个网络是等于另外一个网络，还是另外一个网络的子网。

<

描述：小于

示例：

```
SELECT inet '192.168.1.5' < inet '192.168.1.6' AS RESULT;
 result
--------
 t
(1 row)
```

<=

描述：小于或等于

示例：

```
SELECT inet '192.168.1.5' <= inet '192.168.1.5' AS RESULT;
 result
--------
 t
(1 row)
```

=

描述：等于

示例：

```
SELECT inet '192.168.1.5' = inet '192.168.1.5' AS RESULT;
 result
--------
```

 版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
```

>=

描述：大于或等于

示例：

```
SELECT inet '192.168.1.5' >= inet '192.168.1.5' AS RESULT;
 result
--------
 t
(1 row)
```

>

描述：大于

示例：

```
SELECT inet '192.168.1.5' > inet '192.168.1.4' AS RESULT;
 result
--------
 t
(1 row)
```

<>

描述：不等于

示例：

```
SELECT inet '192.168.1.5' <> inet '192.168.1.4' AS RESULT;
 result
--------
 t
(1 row)
```

<<

描述：包含于

示例：

```
SELECT inet '192.168.1.5' << inet '192.168.1/24' AS RESULT;
 result
--------
 t
(1 row)
```

<<=

描述：包含于或等于

示例：

```
SELECT inet '192.168.1/24' <<= inet '192.168.1/24' AS RESULT;
 result
--------
 t
(1 row)
```

 版权所有 © 华为云计算技术有限公司

>>

描述：包含

示例：

```
SELECT inet '192.168.1/24' >> inet '192.168.1.5' AS RESULT;
 result
--------
 t
(1 row)
```

>>=

描述：包含或等于

示例：

```
SELECT inet '192.168.1/24' >>= inet '192.168.1/24' AS RESULT;
 result
--------
 t
(1 row)
```

~

描述：位非

示例：

```
SELECT ~ inet '192.168.1.6' AS RESULT;
    result
---------------
 63.87.254.249
(1 row)
```

&

描述：两个网络地址的每一位都进行“与”操作。

示例：

```
SELECT inet '192.168.1.6' & inet '10.0.0.0' AS RESULT;
 result
---------
 0.0.0.0
(1 row)
```

|

描述：两个网络地址的每一位都进行“或”操作。

示例：

```
SELECT inet '192.168.1.6' | inet '10.0.0.0' AS RESULT;
   result
-------------
 202.168.1.6
(1 row)
```

+

描述：加

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT inet '192.168.1.6' + 25 AS RESULT;
    result
--------------
 192.168.1.31
(1 row)
```

-

描述：减

示例：

```
SELECT inet '192.168.1.43' - 36 AS RESULT;
   result
-------------
 192.168.1.7
(1 row)
SELECT inet '192.168.1.43' - inet '192.168.1.19' AS RESULT;
 result
--------
     24
(1 row)
```

21.7.25.2 网络地址函数

函数abbrev，host，text主要是为了提供可选的显示格式。

任何cidr值都能以显式或者隐式的方式转换为inet值，因此能够操作inet值的函数也同样能够操作cidr值。inet值也可以转换为cidr值，此时inet子网掩码右侧的所有位都将转换为零，以创建一个有效的cidr值。另外，用户还可以使用常规的类型转换语法将一个文本字符串转换为inet或cidr值。例如：inet(expression)或colname::cidr。

abbrev(inet)

描述：缩写显示格式文本。

返回类型：text

示例：

```
SELECT abbrev(inet '10.1.0.0/16') AS RESULT;
   result
-------------
 10.1.0.0/16
(1 row)
```

abbrev(cidr)

描述：缩写显示格式文本。

返回类型：text

示例：

```
SELECT abbrev(cidr '10.1.0.0/16') AS RESULT;
 result
---------
 10.1/16
(1 row)
```


版权所有 © 华为云计算技术有限公司

broadcast(inet)

描述：网络广播地址。

返回类型：inet

示例：

```
SELECT broadcast('192.168.1.5/24') AS RESULT;
      result
------------------
 192.168.1.255/24
(1 row)
```

family(inet)

描述：抽取地址族，4为IPv4，6为IPv6。

返回类型：int

示例：

```
SELECT family('::1') AS RESULT;
 result
--------
      6
(1 row)
```

host(inet)

描述：将主机地址类型抽出为文本。

返回类型：text

示例：

```
SELECT host('192.168.1.5/24') AS RESULT;
   result
-------------
 192.168.1.5
(1 row)
```

hostmask(inet)

描述：为网络构造主机掩码。

返回类型：inet

示例：

```
SELECT hostmask('192.168.23.20/30') AS RESULT;
 result
---------
 0.0.0.3
(1 row)
```

masklen(inet)

描述：抽取子网掩码长度。

返回类型：int

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT masklen('192.168.1.5/24') AS RESULT;
 result
--------
     24
(1 row)
```

netmask(inet)

描述：为网络构造子网掩码。

返回类型：inet

示例：

```
SELECT netmask('192.168.1.5/24') AS RESULT;
    result
---------------
 255.255.255.0
(1 row)
```

network(inet)

描述：抽取地址的网络部分。

返回类型：cidr

示例：

```
SELECT network('192.168.1.5/24') AS RESULT;
     result
-----------------
 192.168.1.0/24
(1 row)
```

set_masklen(inet, int)

描述：为inet数值设置子网掩码长度。

返回类型：inet

示例：

```
SELECT set_masklen('192.168.1.5/24', 16) AS RESULT;
     result
-----------------
 192.168.1.5/16
(1 row)
```

set_masklen(cidr, int)

描述：为cidr数值设置子网掩码长度。

返回类型：cidr

示例：

```
SELECT set_masklen('192.168.1.0/24'::cidr, 16) AS RESULT;
     result
-----------------
 192.168.0.0/16
(1 row)
```

 版权所有 © 华为云计算技术有限公司

text(inet)

描述：把IP地址和掩码长度抽取为文本。

返回类型：text

示例：

```
SELECT text(inet '192.168.1.5') AS RESULT;
     result
----------------
 192.168.1.5/32
(1 row)
```

trunc(macaddr)

函数trunc(macaddr)返回一个MAC地址，该地址的最后三个字节设置为零。

描述：把后三个字节置为零。

返回类型：macaddr

示例：

```
SELECT trunc(macaddr '12:34:56:78:90:ab') AS RESULT;
      result
-------------------
 12:34:56:00:00:00
(1 row)
```

macaddr类型还支持标准关系操作符（>，<=等）用于词法排序，和按位运算符（~，&和|）非，与和或。

21.7.26 系统信息函数

21.7.26.1 会话信息函数

current_catalog

描述：当前数据库的名字（在标准SQL中称“catalog”），与current_database()同义。

返回值类型：name

示例：

```
SELECT current_catalog;
 current_database
------------------
 gaussdb
(1 row)
```

current_database()

描述：当前数据库的名字。

返回值类型：name

示例：

```
SELECT current_database();
 current_database
```


版权所有 © 华为云计算技术有限公司

```
------------------
 gaussdb
(1 row)
```

pg_client_encoding()

描述：用于获取当前客户端的字符编码方式。当遇到字符显示问题时，通过检查客户端编码可快速定位问题。

返回值类型：name

示例：

```
SELECT pg_client_encoding();
 pg_client_encoding
--------------------
 UTF8
(1 row)
```

current_query()

描述：由客户端提交的当前执行语句（可能包含多个声明）。

返回值类型：text

示例：

```
SELECT current_query();
      current_query
--------------------------
 SELECT current_query();
(1 row)
```

current_schema[()]

描述：current_schema返回在搜索路径中第一个顺位有效的模式名。（如果搜索路径为空则返回NULL，没有有效的模式名也返回NULL）。如果创建表或者其他命名对象时没有声明目标模式，则将使用这些对象的模式。

返回值类型：name

示例：

```
SELECT current_schema();
 current_schema
----------------
 public
(1 row)
```

current_schemas(boolean)

描述：current_schemas(boolean)返回搜索路径中所有模式名字的数组。布尔选项决定像pg_catalog这样隐含包含的系统模式是否包含在返回的搜索路径中。

返回值类型：name[]

示例：

```
SELECT current_schemas(true);
   current_schemas
---------------------
 {pg_catalog,public}
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

搜索路径可以通过运行时设置更改。命令是：

```
SET search_path TO schema [, schema, ...]
```

current_user

描述：当前执行环境下的用户名。current_user是用于权限检查的用户标识。通常用来表示会话用户，但是可以通过**SET ROLE**改变这个设置。在函数执行的过程中随着属性SECURITY DEFINER的改变，其值也会改变。

返回值类型：name

示例：

```
SELECT current_user;
 current_user
--------------
 dbadmin
(1 row)
```

inet_client_addr()

描述：显示当前连接的客户端IP信息。

说明

- 此函数只有在远程连接模式下有效。
- 如果是通过本地连接，使用此接口显示为空。

返回值类型：inet

示例：

```
SELECT inet_client_addr();
 inet_client_addr
------------------
 10.10.0.50
(1 row)
```

inet_client_port()

描述：显示当前连接的客户端的端口号。

说明

此函数只有在远程连接模式下有效。

返回值类型：integer

示例：

```
SELECT inet_client_port();
 inet_client_port
------------------
            33143
(1 row)
```

inet_server_addr()

描述：显示当前服务器的IP信息。

 版权所有 © 华为云计算技术有限公司

说明

- 此函数只有在远程连接模式下有效。
- 如果是通过本地连接，使用此接口显示为空。

返回值类型：inet

示例：

```
SELECT inet_server_addr();
 inet_server_addr
------------------
 10.10.0.13
(1 row)
```

inet_server_port()

描述：显示当前服务器的端口。如果是通过Unix-domain socket连接的，则所有这些函数都返回NULL。

说明

此函数只有在远程连接模式下有效。

返回值类型：integer

示例：

```
SELECT inet_server_port();
 inet_server_port
------------------
 8000
(1 row)
```

pg_backend_pid()

描述：当前会话连接的服务进程的进程ID。

返回值类型：integer

示例：

```
SELECT pg_backend_pid();
 pg_backend_pid
-----------------
 140229352617744
(1 row)
```

pg_conf_load_time()

描述：配置加载时间。pg_conf_load_time返回最后加载服务器配置文件的时间戳。

返回值类型：timestamp with time zone

示例：

```
SELECT pg_conf_load_time();
      pg_conf_load_time
-------------------------------
 2017-09-01 16:05:23.89868+08
(1 row)
```

 版权所有 © 华为云计算技术有限公司

pg_my_temp_schema()

描述：pg_my_temp_schema返回当前会话中临时模式的OID，如果不存在（没有创建临时表）的话则返回0。如果给定的OID是其它会话中临时模式的OID，pg_is_other_temp_schema则返回true。

返回值类型：oid

示例：

```
SELECT pg_my_temp_schema();
 pg_my_temp_schema
-------------------
                 0
(1 row)
```

pg_is_other_temp_schema(oid)

描述：是否为另一个会话的临时模式。

返回值类型：boolean

示例：

```
SELECT pg_is_other_temp_schema(25356);
 pg_is_other_temp_schema
-------------------------
 f
(1 row)
```

pg_postmaster_start_time()

描述：数据库实例启动时间。pg_postmaster_start_time返回数据库实例启动时的timestamp with time zone。

返回值类型：timestamp with time zone

示例：

```
SELECT pg_postmaster_start_time();
   pg_postmaster_start_time
-------------------------------
 2017-08-30 16:02:54.99854+08
(1 row)
```

pg_trigger_depth()

描述：触发器的嵌套层次。

返回值类型：integer

示例：

```
SELECT pg_trigger_depth();
 pg_trigger_depth
------------------
                0
(1 row)
```

pgxc_version()

描述：Postgres-XC版本信息。

 版权所有 © 华为云计算技术有限公司

返回值类型：text

示例：

```
SELECT pgxc_version();
                                                pgxc_version
-----------------------------------------------------------------------------------------------------------
 Postgres-XC 1.1 on x86_64-unknown-linux-gnu, based on PostgreSQL 9.2.4, compiled by g++ (GCC) 5.4.0, 64-bit
(1 row)
```

session_user

描述：会话用户名。session_user通常是连接当前数据库的初始用户，不过系统管理员可以用**SET SESSION AUTHORIZATION**修改这个设置。

返回值类型：name

示例：

```
SELECT session_user;
 session_user
--------------
 dbadmin
(1 row)
```

user

描述：等价于current_user。

返回值类型：name

示例：

```
SELECT user;
 current_user
--------------
 dbadmin
(1 row)
```

version()

描述：版本信息。version返回一个描述服务器版本信息的字符串。

返回值类型：text

示例：

```
SELECT version();
                                                
```

has_any_column_privilege(table, privilege)

描述：当前用户是否有访问表任何列的权限。

返回类型：boolean

备注：has_any_column_privilege检查用户是否以特定方式访问表的任何列。其参数可能与has_table_privilege类似，除了访问权限类型必须是SELECT、INSERT、UPDATE或REFERENCES的一些组合。

说明

拥有表的表级别权限则隐含的拥有该表每列的列级权限，因此如果与has_table_privilege参数相同，has_any_column_privilege总是返回true。但是如果授予至少一列的列级权限也返回成功。

has_column_privilege(user, table, column, privilege)

描述：指定用户是否有访问列的权限。

返回类型：boolean

has_column_privilege(table, column, privilege)

描述：当前用户是否有访问列的权限。

返回类型：boolean

备注：has_column_privilege检查用户是否以特定方式访问一列。其参数类似于has_table_privilege，可以通过列名或属性号添加列。想要的访问权限类型必须是SELECT、INSERT、UPDATE或REFERENCES的一些组合。

说明

拥有表的表级别权限则隐含的拥有该表每列的列级权限。

has_database_privilege(user, database, privilege)

描述：指定用户是否有访问数据库的权限。

返回类型：boolean

has_database_privilege(database, privilege)

描述：当前用户是否有访问数据库的权限。

返回类型：boolean

备注：has_database_privilege检查用户是否能以在特定方式访问数据库。其参数类似has_table_privilege。访问权限类型必须是CREATE、CONNECT、TEMPORARY或TEMP（等价于TEMPORARY）的一些组合。

has_foreign_data_wrapper_privilege(user, fdw, privilege)

描述：指定用户是否有访问外部数据封装器的权限。

参数fdw，表示外部数据封装器，取值范围为外部数据封装器名字或ID。

返回类型：boolean

版权所有 © 华为云计算技术有限公司

has_foreign_data_wrapper_privilege(fdw, privilege)

描述：当前用户是否有访问外部数据封装器的权限。

返回类型：boolean

备注：has_foreign_data_wrapper_privilege检查用户是否能以特定方式访问外部数据封装器。其参数类似has_table_privilege。访问权限类型必须是USAGE。

has_function_privilege(user, function, privilege)

描述：指定用户是否有访问函数的权限。

返回类型：boolean

has_function_privilege(function, privilege)

描述：当前用户是否有访问函数的权限。

返回类型：boolean

备注：has_function_privilege检查一个用户是否能以指定方式访问函数。其参数类似has_table_privilege。使用文本字符而不是OID声明一个函数时，允许输入的类型和regprocedure数据类型一样（请参考**对象标识符类型**）。访问权限类型必须是EXECUTE。

has_language_privilege(user, language, privilege)

描述：指定用户是否有访问语言的权限。

返回类型：boolean

has_language_privilege(language, privilege)

描述：当前用户是否有访问语言的权限。

返回类型：boolean

备注：has_language_privilege检查用户是否能以特定方式访问过程语言。其参数类似has_table_privilege。访问权限类型必须是USAGE。

has_schema_privilege(user, schema, privilege)

描述：指定用户是否有访问模式的权限。

返回类型：boolean

has_schema_privilege(schema, privilege)

描述：当前用户是否有访问模式的权限。

返回类型：boolean

备注：has_schema_privilege检查用户是否能以特定方式访问模式。其参数类似has_table_privilege。访问权限类型必须是CREATE或USAGE的一些组合。

 版权所有 © 华为云计算技术有限公司

has_server_privilege(user, server, privilege)

描述：指定用户是否有访问外部服务的权限。

返回类型：boolean

has_server_privilege(server, privilege)

描述：当前用户是否有访问外部服务的权限。

返回类型：boolean

备注：has_server_privilege检查用户是否能以指定方式访问外部服务器。其参数类似has_table_privilege。访问权限类型必须是USAGE。

has_table_privilege(user, table, privilege)

描述：指定用户是否有访问表的权限。

返回类型：boolean

has_table_privilege(table, privilege)

描述：当前用户是否有访问表的权限。

返回类型：boolean

备注：has_table_privilege检查用户是否以特定方式访问表。用户可以通过名字或OID（pg_authid.oid）来指定，public表明PUBLIC伪角色，如果缺省该参数，则使用current_user。该表可以通过名字或者OID声明。如果用名字声明，则在必要时可以用模式进行修饰。如果使用文本字符串来声明权限类型，该文本字符串必须是SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES或TRIGGER之一的值。可以给权限类型添加WITH GRANT OPTION，用来测试权限是否拥有授权选项。也可以用逗号分隔列出的多个权限类型，如果拥有任何所列出的权限，则结果便为true。

示例：

```
SELECT has_table_privilege('tpcds.web_site', 'select');
 has_table_privilege
---------------------
 t
(1 row)

SELECT has_table_privilege('dbadmin', 'tpcds.web_site', 'select,INSERT WITH GRANT OPTION ');
 has_table_privilege
---------------------
 t
(1 row)
```

pg_has_role(user, role, privilege)

描述：指定用户是否有角色的权限。

返回类型：boolean

pg_has_role(role, privilege)

描述：当前用户是否有角色的权限。

返回类型：boolean

版权所有 © 华为云计算技术有限公司

备注：pg_has_role检查用户是否能以特定方式访问角色。其参数类似has_table_privilege，除了public不能用做用户名。访问权限类型必须是MEMBER或USAGE的一些组合。 MEMBER表示的是角色中的直接或间接成员关系（也就是SET ROLE的权限），而USAGE表示无需通过SET ROLE也直接拥有角色的使用权限。

21.7.26.3 模式可见性查询函数

每个函数对数据库对象执行可见性检查。对于函数和操作符，如果在前面的搜索路径中没有相同的对象名称和参数的数据类型，则此对象是可见的。对于操作符类，则要同时考虑名字和相关索引的访问方法。

所有这些函数都需要使用OID来标识要需要检查的对象。如果用户想通过名字测试对象，则可使用OID别名类型（regclass、regtype、regprocedure、regoperator、regconfig或regdictionary）。

比如，如果一个表所在的模式在搜索路径中，并且在前面的搜索路径中没有同名的表，那么这个表是可见的。它等效于表可以不带明确模式修饰进行引用。比如，要列出所有可见表的名字：

```
SELECT relname FROM pg_class WHERE pg_table_is_visible(oid);
```

pg_collation_is_visible(collation_oid)

描述：排序是否在搜索路径中可见。

返回类型：boolean

pg_conversion_is_visible(conversion_oid)

描述：转换是否在搜索路径中可见。

返回类型：boolean

pg_function_is_visible(function_oid)

描述：函数是否在搜索路径中可见。

返回类型：boolean

pg_opclass_is_visible(opclass_oid)

描述：操作符类是否在搜索路径中可见。

返回类型：boolean

pg_operator_is_visible(operator_oid)

描述：操作符是否在搜索路径中可见。

返回类型：boolean

pg_opfamily_is_visible(opclass_oid)

描述：操作符族是否在搜索路径中可见。

返回类型：boolean

 版权所有 © 华为云计算技术有限公司

pg_table_is_visible(table_oid)

描述：表是否在搜索路径中可见。

返回类型：boolean

pg_ts_config_is_visible(config_oid)

描述：文本检索配置是否在搜索路径中可见。

返回类型：boolean

pg_ts_dict_is_visible(dict_oid)

描述：文本检索词典是否在搜索路径中可见。

返回类型：boolean

pg_ts_parser_is_visible(parser_oid)

描述：文本搜索解析是否在搜索路径中可见。

返回类型：boolean

pg_ts_template_is_visible(template_oid)

描述：文本检索模板是否在搜索路径中可见。

返回类型：boolean

pg_type_is_visible(type_oid)

描述：类型（或域）是否在搜索路径中可见。

返回类型：boolean

21.7.26.4 系统表信息函数

format_type(type_oid, typemod)

描述：获取数据类型的SQL名称。

返回类型：text

备注：

format_type通过数据类型的类型OID以及可能的类型修饰词，返回其SQL名称。如果不知道具体的修饰词，则在类型修饰词的位置传入NULL。类型修饰词一般只对有长度限制的数据类型有意义。format_type所返回的SQL名称中包含数据类型的长度值，其大小是：实际存储长度len - sizeof(int32)，单位字节。数据存储时需要32位的空间来存储用户对数据类型的自定义长度信息，即实际存储长度要比用户定义长度多4个字节。在下例中，format_type返回的SQL名称为“character varying(6)”，6表示varchar类型的长度值是6字节，因此该类型的实际存储长度为10字节。

```
SELECT format_type((SELECT oid FROM pg_type WHERE typname='varchar'), 10);
    format_type
----------------------
```

 版权所有 © 华为云计算技术有限公司

```
 character varying(6)
(1 row)
```

pg_check_authid(role_oid)

描述：检查是否存在给定oid的角色名。

返回类型：bool

pg_describe_object(catalog_id, object_id, object_sub_id)

描述：获取数据库对象的描述。

返回类型：text

备注：pg_describe_object返回由目录OID，对象OID和一个（或许0个）子对象ID指定的数据库对象的描述。这有助于确认存储在pg_depend系统表中对象的身份。

pg_get_constraintdef(constraint_oid)

描述：获取约束的定义。

返回类型：text

pg_get_constraintdef(constraint_oid, pretty_bool)

描述：获取约束的定义。

返回类型：text

备注：pg_get_constraintdef和pg_get_indexdef分别从约束或索引上使用创建命令进行重构。

pg_get_expr(pg_node_tree, relation_oid)

描述：反编译表达式的内部形式，假设其中的任何Vars都引用第二个参数指定的关系。

返回类型：text

pg_get_expr(pg_node_tree, relation_oid, pretty_bool)

描述：反编译表达式的内部形式，假设其中的任何Vars都引用第二个参数指定的关系。

返回类型：text

备注：pg_get_expr反编译一个独立表达式的内部形式，比如一个字段的缺省值。在检查系统表的内容的时候很有用。如果表达式可能包含关键字，则指定它们引用相关的OID作为第二个参数；如果没有关键字，设置为零即可。

pg_get_indexdef(index_oid)

描述：获取索引的CREATE INDEX命令。

返回类型：text

index_oid为索引的OID，可以通过PG_STATIO_ALL_INDEXES系统视图查询。

 版权所有 © 华为云计算技术有限公司

示例：查询索引ds_ship_mode_t1_index1的OID及其创建命令。

```
SELECT indexrelid FROM PG_STATIO_ALL_INDEXES WHERE indexrelname = 'ds_ship_mode_t1_index1';
 indexrelid
------------
     136035
(1 row)
SELECT * FROM pg_get_indexdef(136035);
                                        pg_get_indexdef
-------------------------------------------------------------------------------------------------------------------
 CREATE INDEX ds_ship_mode_t1_index1 ON tpcds.ship_mode_t1 USING psort (sm_ship_mode_sk)
TABLESPACE pg_default
(1 row)
```

pg_get_indexdef(index_oid, column_no, pretty_bool)

描述：获取索引的CREATE INDEX命令，或者如果column_no不为零，则只获取一个索引字段的定义。

返回类型：text

```
SELECT * FROM pg_get_indexdef(136035,0,false);
                                        pg_get_indexdef
-------------------------------------------------------------------------------------------------------------------
 CREATE INDEX ds_ship_mode_t1_index1 ON tpcds.ship_mode_t1 USING psort (sm_ship_mode_sk)
TABLESPACE pg_default
(1 row)
SELECT * FROM pg_get_indexdef(136035,1,false);
 pg_get_indexdef
-----------------
 sm_ship_mode_sk
(1 row)
```

pg_get_keywords()

描述：获取SQL关键字和类别列表。

返回类型：setof record

备注：pg_get_keywords返回一组关于描述服务器识别SQL关键字的记录。word列包含关键字。catcode列包含一个分类代码：U表示通用的，C表示列名，T表示类型或函数名，或R表示保留。catdesc列包含了一个可能本地化描述分类的字符串。

pg_get_ruledef(rule_oid)

描述：获取规则的CREATE RULE命令。

返回类型：text

pg_get_ruledef(rule_oid, pretty_bool)

描述：获取规则的CREATE RULE命令。

返回类型：text

pg_get_userbyid(role_oid)

描述：获取给定OID的角色名。

返回类型：name

备注：pg_get_userbyid通过角色的OID抽取对应的用户名。

版权所有 © 华为云计算技术有限公司

pg_get_viewdef(viewname text [, pretty bool [, fullflag bool]])

描述：为视图获取底层的SELECT命令。

返回类型：text

备注：

- pg_get_viewdef重构定义视图的SELECT查询。pretty bool参数为true时，显示格式“适合打印”，且该格式易读。pretty bool参数缺省值为false，显示格式不易读。如果用于转储，那么尽可能使用缺省格式。pretty bool参数只对有效视图生效。
- fullflag bool参数为true时，显示视图的完整定义。其缺省值为false。

pg_get_viewdef(viewoid oid [, pretty bool [, fullflag bool]])

描述：为视图获取底层的SELECT命令。

返回类型：text

pg_get_viewdef(view_oid, wrap_column_int)

描述：为视图获取底层的SELECT命令；行字段被换到指定的列数，打印是隐含的。

返回类型：text

pg_get_tabledef(table_oid)

描述：根据表的OID获取表定义。

返回类型：text

示例：先通过系统表pg_class获取表customer_t2的OID，再使用此函数查询表customer_t2的定义，可获取创建表customer_t2时的表字段，表的存储方式（行存或列存）及表的分布方式等信息。

```
select oid from pg_class where relname ='customer_t2';
  oid
-------
 17353
(1 row)

select * from pg_get_tabledef(17353);
              pg_get_tabledef
--------------------------------------------
 SET search_path = dbadmin;                +
 CREATE  TABLE customer_t2 (               +
        state_id character(2),             +
        state_name character varying(40), +
        area_id numeric                    +
 )                                         +
 WITH (orientation=column, compression=low)+
 DISTRIBUTE BY HASH(state_id)              +
 TO GROUP group_version1;
(1 row)
```

pg_get_tabledef(table_name)

描述：根据表名获取表定义。

返回类型：text

 版权所有 © 华为云计算技术有限公司

备注：pg_get_tabledef重构出表定义的CREATE语句，包含了表定义本身、索引信息、comments信息。对于表对象依赖的group、schema、tablespace、server等信息，需要用户自己去创建，表定义里不会有这些对象的创建语句。

pg_get_tabledef(table_name/table_oid, forCreate boolean)

描述：根据表名或者表的OID获取表定义。该函数仅9.1.0.100及以上集群版本支持。

返回类型：text

- forCreate参数为true时获得的表定义中不带TO GROUP
- forCreate参数为false时获得的表定义中含有TO GROUP。

pg_get_tabledef(table_name/table_oid, forCreate boolean,withColComm boolean)

描述：根据表名或者表的OID获取表定义。该函数仅9.1.0.200及以上集群版本支持。

返回类型：text

- withColComment参数为true时，列字段的comment紧随对应字段显示。
- withColComment参数为false时，列字段的comment显示在表定义内容的末尾。

示例：

```
SELECT pg_get_tabledef('person',false,true);
                pg_get_tabledef
-----------------------------------------------------
 SET search_path = public;                          +
 CREATE  TABLE person (                              +
        id integer COMMENT '学号',                    +
        name character varying(25) COMMENT '姓名',+
        address text COMMENT '地址'                   +
 )                                                  +
 WITH (orientation=row, compression=no)              +
 DISTRIBUTE BY ROUNDROBIN                            +
 TO GROUP group_version1;
(1 row)

SELECT pg_get_tabledef('person',false,false);
              pg_get_tabledef
-----------------------------------------------
 SET search_path = public;                    +
 CREATE  TABLE person (                        +
        id integer,                           +
        name character varying(25),           +
        address text                          +
 )                                           +
 WITH (orientation=row, compression=no)       +
 DISTRIBUTE BY ROUNDROBIN                     +
 TO GROUP group_version1;                     +
 COMMENT ON COLUMN person.id IS '学号';       +
 COMMENT ON COLUMN person.name IS '姓名';     +
 COMMENT ON COLUMN person.address IS '地址';
(1 row)
```

pg_get_stat_expressions(oid)

描述：获取扩展统计信息对象中的表达式。仅8.2.1.200及以上集群版本支持。

返回类型：text []

 版权所有 © 华为云计算技术有限公司

pg_options_to_table(reloptions)

描述：获取存储选项名称/值对的集合。

返回类型：setof record

备注：pg_options_to_table当通过pg_class.reloptions或pg_attribute.attoptions时返回存储选项名字/值对（option_name/option_value）的集合。

示例：

```
CREATE TABLE customer_test
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
)
WITH (ORIENTATION = COLUMN,COMPRESSION=middle);
SELECT pg_options_to_table(reloptions) FROM pg_class WHERE relname='customer_test';
pg_options_to_table
----------------------
 (orientation,column)
 (compression,middle)
 (bucketnums,16384)
 (colversion,2.0)
 (enable_delta,false)
(5 rows)
```

pg_typeof(any)

描述：获取任何值的数据类型。

返回类型：regtype

备注：

pg_typeof返回传递给他的值的数据类型OID。这可能有助于故障排除或动态构造SQL查询。声明此函数返回regtype，这是一个OID别名类型（请参考**对象标识符类型**）；这意味着它是一个为了比较而显示类型名字的OID。

示例：

```
SELECT pg_typeof(33);
 pg_typeof
-----------
 integer
(1 row)

SELECT typlen FROM pg_type WHERE oid = pg_typeof(33);
 typlen
--------
      4
(1 row)
```

collation for (any)

描述：获取参数的排序。

返回类型：text

备注：

表达式collation for返回传递给他的值的排序。示例：

```
SELECT collation for (description) FROM pg_description LIMIT 1;
 pg_collation_for
```


版权所有 © 华为云计算技术有限公司

```
------------------
 "default"
(1 row)
```

值可能是引号括起来的并且模式限制的。如果没有为参数表达式排序，则返回一个null值。如果参数不是排序的类型，则抛出一个错误。

getdistributekey(table_name)

描述：获取一个hash表的分布列。

返回类型：text

示例：

```
SELECT getdistributekey('item');
 getdistributekey
------------------
 i_item_sk
(1 row)
```

21.7.26.5 系统函数信息函数

pv_builtin_functions()

描述：查询系统内置函数的信息。

返回类型：record

pg_get_functiondef(func_oid)

描述：获取函数的定义。

返回类型：text

func_oid为函数的OID，可以通过PG_PROC系统表查询。

示例：查询函数justify_days的OID及其函数定义。

```
SELECT oid FROM pg_proc WHERE proname ='justify_days';
 oid
------
 1295
(1 row)

SELECT * FROM pg_get_functiondef(1295);
 headerlines |                          definition
-------------+-----------------------------------------------------------------
           4 | CREATE OR REPLACE FUNCTION pg_catalog.justify_days(interval)+
             |  RETURNS interval                                           +
             |  LANGUAGE internal                                            +
             |  IMMUTABLE STRICT NOT FENCED NOT SHIPPABLE                  +
             | AS $function$interval_justify_days$function$                +
             |
(1 row)
```

备注：pg_get_functiondef返回函数查询结果为存储过程原始文本格式，自带转义符'\'，便于后续应用的正确解析。

pg_get_function_arguments(func_oid)

描述：获取函数定义的参数列表（带默认值）。

版权所有 © 华为云计算技术有限公司

返回类型：text

备注：pg_get_function_arguments返回一个函数的参数列表，需要在CREATE FUNCTION中使用这种格式。

pg_get_function_identity_arguments(func_oid)

描述：获取参数列表来确定一个函数 (不带默认值)。

返回类型：text

备注：pg_get_function_identity_arguments返回需要的参数列表用来标识函数，这种形式需要在ALTER FUNCTION中使用，并且这种形式省略了默认值。

pg_get_function_result(func_oid)

描述：获取函数的RETURNS子句。

返回类型：text

备注：pg_get_function_result为函数返回适当的RETURNS子句。

21.7.26.6 注释信息函数

col_description(table_oid, column_number)

描述：获取一个表字段的注释。

返回类型：text

备注：col_description返回一个表中字段的注释，通过表OID和字段号来声明。

示例：查询pg_class系统表获取表OID，查询INFORMATION_SCHEMA.COLUMNS系统视图获取column_number。

```
SELECT COLUMN_NAME,ORDINAL_POSITION FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME = 't' AND COLUMN_NAME = 'a';
column_name | ordinal_position
-------------+------------------
a           |                1
(1 row)

SELECT oid FROM pg_class WHERE relname='t';
  oid
-------
 44020
(1 row)

SELECT col_description(44020,1);
    col_description
-----------------------
 This is a test table.
(1 row)
```

obj_description(object_oid, catalog_name)

描述：获取一个数据库对象的注释。

返回类型：text

版权所有 © 华为云计算技术有限公司

备注：带有两个参数的obj_description返回一个数据库对象的注释，该对象是通过其OID和其所属的系统表名字声明。比如，obj_description(123456,'pg_class')将返回OID为123456的表的注释。只带一个参数的obj_description只要求对象OID。

obj_description不能用于表字段，因为字段没有自己的OID。

obj_description(object_oid)

描述：获取一个数据库对象的注释。

返回类型：text

shobj_description(object_oid, catalog_name)

描述：获取一个共享数据库对象的注释。

返回类型：text

备注：shobj_description和obj_description功能相似，不同之处仅在于shobj_description用于共享对象。一些系统表是通用于集群中所有数据库的全局表，因此这些表的注释也是全局存储的。

21.7.26.7 事务 ID 和快照

以下的函数在输出形式中提供服务器事务信息。这些函数的主要用途是为了确定在两个快照之间有哪个事务提交。

pgxc_is_committed(transaction_id)

描述：如果提交或忽略给定的XID（gxid）。NULL表示的状态是未知的（运行，准备，冻结等）。

返回类型：bool

txid_current()

描述：获取当前事务ID。

返回类型：bigint

txid_current_snapshot()

描述：获取当前快照。

返回类型：txid_snapshot

txid_snapshot_xip(txid_snapshot)

描述：在快照中获取正在进行的事务ID。

返回类型：setof bigint

txid_snapshot_xmax(txid_snapshot)

描述：获取快照的xmax。

版权所有 © 华为云计算技术有限公司

返回类型：bigint

txid_snapshot_xmin(txid_snapshot)

描述：获取快照的xmin。

返回类型：bigint

txid_visible_in_snapshot(bigint, txid_snapshot)

描述：在快照中事务ID是否可见（不使用子事务ID）。

返回类型：boolean

内部事务ID类型（xid）是32位，每40亿事务一次循环。这些函数使用的数据类型txid_snapshot，存储在特定时刻事务ID可见性的信息。其组件描述在**表21-50**。

表 21-50 快照组件

名字	描述
xmin	最早的事务ID（txid）仍然活动。所有较早事务将是已经提交可见的，或者是直接回滚。
xmax	作为尚未分配的txid。所有大于或等于此txids的都是尚未开始的快照时间，因此不可见。
xip_list	当前快照中活动的txids。这个列表只包含在xmin和xmax之间活动的txids；有可能活动的txids高于xmax。 介于大于等于xmin、小于xmax，并且不在这个列表中的txid，在这个时间快照已经完成的，因此按照提交状态查看他是可见还是回滚。这个列表不包含子事务的txids。

txid_snapshot的文本表示为：xmin:xmax:xip_list。

示例：10:20:10,14,15意思为：xmin=10, xmax=20, xip_list=10, 14, 15。

21.7.26.8 计算子集群函数

pv_compute_pool_workload()

描述：返回计算子集群当前的负载状态。

返回类型：void

示例：

```
SELECT * from pv_compute_pool_workload();
nodename  | rpinuse | maxrp | nodestate
----------+---------+-------+-----------
datanode1 |       0 |  1000 | normal
datanode2 |       0 |  1000 | normal
(2 rows)
```


版权所有 © 华为云计算技术有限公司

21.7.26.9 锁信息函数

pgxc_get_lock_conflicts()

描述：返回集群中有冲突的锁信息。当某个锁正在等待另一个锁，或正被另一个锁等待，则认为该锁是冲突的。

返回类型：setof record

示例：

```
SELECT * from pgxc_get_lock_conflicts();
                                        pgxc_get_lock_conflicts

-------------------------------------------------------------------------------------------------------------------
 (relation,cn_5001,gaussdb,u1,test,,,,,omm,648406,,,"<insufficient
privilege>",140651661784832,ExclusiveLock,f,2024-10-29 17:24:13.425672+08,)
(1 row
```

21.7.27 系统管理函数

21.7.27.1 配置设置函数

配置设置函数是用于查询及修改运行时配置参数的函数。

current_setting(setting_name)

描述：当前的设置值。

返回值类型：text

备注：current_setting用于以查询形式获取setting_name的当前值。和SQL语句SHOW是等效的。比如：

```
SELECT current_setting('datestyle');

 current_setting
-----------------
 ISO, MDY
(1 row)
```

read_global_var(global_setting_name)

描述：用于读取全局变量当前的设置值。

返回值类型：text

备注：read_global_var用于以查询形式获取global_setting_name的当前值。和SQL语句SHOW是等效的。

示例：

```
SET my.var = 1000;
SELECT read_global_var('my.var');
 read_global_var
-----------------
 1000
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

global_setting_name表示类似于my.var类型的变量，其特征为名称中存在小数点且小数点左右均不为空。这种类型的变量叫做全局变量。

全局变量只能通过SET命令设置，不支持gs_guc、ALTER DATABASE dbname SET paraname TO value、ALTER USER username SET paraname TO value等设置方法，对所有用户可见。

set_config(setting_name, new_value, is_local)

描述：设置参数并返回新值。

返回值类型：text

备注：set_config将参数setting_name设置为new_value，如果is_local为true，则新值将只应用于当前事务。如果希望新值应用于当前会话，可以使用false，和SQL语句SET是等效的。比如：

```
SELECT set_config('log_statement_stats', 'off', false);

 set_config
------------
 off
(1 row)
```

注意

如果GUC参数behavior_compat_options设置配置项为DISABLE_SET_GLOBAL_VAR_ON_DATANODE，则不可以通过该函数直接在DN上设置全局变量。

21.7.27.2 通用文件访问函数

通用文件访问函数提供了对数据库服务器上的文件的本地访问接口。只有数据库集群目录和log_directory目录里面的文件可以访问。使用相对路径访问集群目录里面的文件，以及匹配log_directory配置而设置的路径访问日志文件。只有数据库系统管理员才能使用这些函数。

pg_ls_dir(dirname text)

描述：列出目录中的文件。

返回值类型：setof text

备注：pg_ls_dir返回指定目录里面的除了特殊项“.”和“..”之外所有名字。

示例：

```
SELECT pg_ls_dir('./');
     pg_ls_dir
----------------------
 .postgresql.conf.swp
 postgresql.conf
 pg_tblspc
 PG_VERSION
 pg_ident.conf
 core
 server.crt
 pg_serial
```


版权所有 © 华为云计算技术有限公司

```
 pg_twophase
 postgresql.conf.lock
 pg_stat_tmp
 pg_notify
 pg_subtrans
 pg_ctl.lock
 pg_xlog
 pg_clog
 base
 pg_snapshots
 postmaster.opts
 postmaster.pid
 server.key.rand
 server.key.cipher
 pg_multixact
 pg_errorinfo
 server.key
 pg_hba.conf
 pg_replslot
 .pg_hba.conf.swp
 cacert.pem
 pg_hba.conf.lock
 global
 gaussdb.state
(32 rows)
```

pg_read_file(filename text, offset bigint, length bigint)

描述：返回一个文本文件的内容。

返回值类型：text

备注：pg_read_file返回一个文本文件的一部分，从offset开始，最多返回length字节（如果先达到文件结尾，则小于这个数值）。如果offset是负数，则它是相对于文件结尾回退的长度。如果省略了offset和length，则返回整个文件。

示例：

```
SELECT pg_read_file('postmaster.pid',0,100);
             pg_read_file
----------------------------------------
 53078                                 +
/srv/BigData/hadoop/data1/coordinator+
1500022474                             +
8000                                  +
/var/run/dws                           +
localhost                             +
 2
(1 row)
```

pg_read_binary_file(filename text [, offset bigint, length bigint,missing_ok boolean])

描述：返回一个二进制文件的内容。

返回值类型：bytea

备注：pg_read_binary_file的功能与pg_read_file类似，除了结果的返回值为bytea类型不一致，相应地不会执行编码检查。与convert_from函数结合，这个函数可以用来读取用指定编码的一个文件。

```
SELECT convert_from(pg_read_binary_file('filename'), 'UTF8');
```

 版权所有 © 华为云计算技术有限公司

pg_stat_file(filename text)

描述：返回一个文本文件的状态信息。

返回值类型：record

备注：pg_stat_file返回一条记录，其中包含：文件大小、最后访问时间戳、最后更改时间戳、最后文件状态修改时间戳以及标识传入参数是否为目录的boolean值。典型的用法：

```
SELECT * FROM pg_stat_file('filename');
SELECT (pg_stat_file('filename')).modification;
```

示例：

```
SELECT * FROM pg_stat_file('postmaster.pid');

 size |         access         |      modification      |         change
| creation | isdir
------+------------------------+------------------------+------------------------
+----------+-------
  117 | 2017-06-05 11:06:34+08 | 2017-06-01 17:18:08+08 | 2017-06-01 17:18:08+08
|          | f
(1 row)
SELECT (pg_stat_file('postmaster.pid')).modification;
      modification
------------------------
 2017-06-01 17:18:08+08
(1 row)
```

21.7.27.3 服务器信号函数

服务器信号函数向其他服务器进程发送控制信号。只有系统管理员才能使用这些函数。

pg_cancel_backend(pid int)

描述：取消一个后端的当前查询。

返回值类型：boolean

备注：pg_cancel_backend向由pid标识的后端进程发送一个查询取消（SIGINT）信号。一个活动的后端进程的PID可以从pg_stat_activity视图的pid字段找到，或者在服务器上用ps列出数据库进程。

示例：

```
SELECT pid FROM pg_stat_activity WHERE stmt_type ='RESET';
       pid
-----------------
 281471222065200
(1 row)

SELECT pg_cancel_backend(281471222065200);
 pg_cancel_backend
-------------------
 t
(1 row)
```

pg_reload_conf()

描述：服务器进程重新装载配置文件。

返回值类型：boolean

版权所有 © 华为云计算技术有限公司

备注：pg_reload_conf给服务器发送一个SIGHUP信号，导致所有服务器进程重新装载配置文件。

示例：

```
SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
```

pg_rotate_logfile()

描述：滚动服务器的日志文件。

返回值类型：boolean

备注：pg_rotate_logfile通知日志文件管理器立即切换到一个新的输出文件。该函数仅在内置日志收集器运行时有效。

示例：

```
SELECT pg_rotate_logfile();
 pg_rotate_logfile
-------------------
 t
(1 row)
```

pg_terminate_backend(pid int)

描述：终止一个后台线程。

返回值类型：boolean

备注：如果成功，函数返回true，否则返回false。

示例：

```
SELECT pid FROM pg_stat_activity;
       pid
-----------------
 140657876268816
 140433774061312
 140433587902208
 140433656592128
 140433723717376
 140433637189376
 140433552770816
 140433481983744
 140433349310208
(9 rows)

SELECT pg_terminate_backend(140657876268816);
 pg_terminate_backend
----------------------
 t
(1 row)
```

pg_wlm_jump_queue(pid int)

描述：调整任务到CN队列的最前端。

返回值类型：boolean

备注：如果成功，函数返回true，否则返回false。

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT pid FROM pg_stat_activity WHERE stmt_type ='RESET';
      pid
-----------------
 281471222065200
(1 row)

SELECT pg_wlm_jump_queue(281471222065200);
 pg_wlm_jump_queue
--------------------
 t
(1 row)
```

gs_wlm_switch_cgroup(pid int, cgroup text)

描述：调整作业的优先级到新控制组。

返回值类型：boolean

备注：如果成功，函数返回true，否则返回false。

pg_cancel_query(queryId int)

描述：取消一个后端的当前查询。该函数8.1.2及以上版本支持。

返回值类型：boolean

备注：pg_cancel_query向由query_id标识的后端进程发送一个查询取消（SIGINT）信号。一个活动的后端进程的query_id可以从pg_stat_activity视图的query_id字段找到。

示例：

```
SELECT query_id FROM pgxc_stat_activity WHERE stmt_type ='RESET';
 query_id
----------
       0
       0
(2 rows)

SELECT pg_cancel_query(0);
 pg_cancel_query
-----------------
 f
(1 row)
```

pgxc_cancel_query(queryId int)

描述：取消当前集群下正在执行的查询。该函数8.1.2及以上版本支持。

返回值类型：boolean

备注：如果所有节点的查询均已取消，函数返回true，否则返回false。

示例：

```
SELECT query_id FROM pgxc_stat_activity WHERE stmt_type ='RESET';
 query_id
----------
       0
       0
(2 rows)

SELECT pgxc_cancel_query(0);
 pgxc_cancel_query
```

 版权所有 © 华为云计算技术有限公司

```
--------------------
 f
(1 row)
```

pg_terminate_query(queryId int)

描述：终止一个后端的当前查询。该函数8.1.2及以上版本支持。

返回值类型：boolean

示例：

```
SELECT query_id FROM pgxc_stat_activity WHERE stmt_type ='RESET';
 query_id
----------
        0
        0
(2 rows)

SELECT pg_terminate_query(0);
 pg_terminate_query
--------------------
 f
(1 row)
```

pgxc_terminate_query(queryId int)

描述：终止当前集群下正在执行的查询。该函数8.1.2及以上版本支持。

返回值类型：boolean

示例：

```
SELECT query_id FROM pgxc_stat_activity;
    query_id
-----------------
 72339069014638631
(1 rows)

SELECT pgxc_terminate_query(72339069014638631);
 pgxc_terminate_query
----------------------
 t
(1 row)
```

21.7.27.4 快照同步函数

快照同步函数是导出当前快照的标识符。

create_wdr_snapshot()

描述：创建性能数据快照。

返回值类型：text

版权所有 © 华为云计算技术有限公司

说明

- 该函数只有数据库管理员SYSADMIN才可以执行，非管理员执行会提示无权限。
- 该函数只能在CN上执行，在DN上执行会返回：“WDR snapshot can only be created on coordinator.”。
- 执行该函数前需确认enable_wdr_snapshot参数处于开启状态。如果enable_wdr_snapshot为off，执行该函数会返回：“WDR snapshot request can't be executed, because GUC parameter 'enable_wdr_snapshot' is off.”。
- 如果执行该函数时，快照线程由于节点重启等原因尚未启动，会提示错误：“WDR snapshot request can not be accepted, please retry later.”。
- 如果执行该函数失败，会提示：“Cannot respond to WDR snapshot request.”。
- 如果执行成功，会返回：“WDR snapshot request has been submitted.”。该提示表明创建快照请求已发送至后台快照线程，但不代表创建快照成功。

kill_snapshot(scope cstring)

描述：中止后台快照线程。该函数向后台快照线程发送中止信号并等待线程结束。

输入参数scope：表示操作范围。该参数取值范围为local和global。

- local表示中止当前CN上的快照线程。
- global表示不仅会中止当前CN上的快照线程，还会向集群中所有其他CN发送中止快照线程的请求，即中止集群中所有CN上的快照线程。
- 如果输入其他值，则报错“Scope is invalid, use "local" or "global".”。
- 输入参数可为空，表示默认取值为local。

返回值类型：无

说明

- 该函数只有数据库管理员SYSADMIN才有权执行，非管理员执行会提示无权限。
- 该函数只能在CN上执行，在DN上执行会提示：“kill_snapshot can only be executed on coordinator.”；
- 执行该函数会向后台快照线程发送中止信号并等待其结束。如果100s内快照线程仍未中止则会报错：“Kill snapshot thread failed”；

pg_export_snapshot()

描述：保存当前的快照并返回它的标识符。

返回值类型：text

备注：函数pg_export_snapshot保存当前的快照并返回一个文本字符串标识此快照。这个字符串必须传递给想要导入快照的客户端。可用在set transaction snapshot snapshot_id时导入snapshot，但是应用的前提是该事务设置了REPEATABLE READ隔离级别。该函数的输出不可用做set transaction snapshot的输入。

示例：

```
SELECT pg_export_snapshot();
 pg_export_snapshot
--------------------
 00000000000A7128-1
(1 row)
```

 版权所有 © 华为云计算技术有限公司

21.7.27.5 咨询锁函数

咨询锁函数用于管理咨询锁（Advisory Lock），此类函数目前仅内部使用。

pg_advisory_lock(key bigint)

描述：获取会话级别的排它咨询锁。

返回值类型：void

备注：pg_advisory_lock锁定应用程序定义的资源，该资源可以用一个64位或两个不重叠的32位键值标识。如果已经有另外的会话锁定了该资源，则该函数将阻塞到该资源可用为止。这个锁是排它的。多个锁定请求将会被压入栈中，因此，如果同一个资源被锁定了三次，它必须被解锁三次以将资源释放给其他会话使用。

pg_advisory_lock(key1 int, key2 int)

描述：获取会话级别的排它咨询锁。

返回值类型：void

pg_advisory_lock_shared(key bigint)

描述：获取会话级别的共享咨询锁。

返回值类型：void

pg_advisory_lock_shared(key1 int, key2 int)

描述：获取会话级别的共享咨询锁。

返回值类型：void

备注：pg_advisory_lock_shared类似于pg_advisory_lock，不同之处仅在于共享锁会话可以和其他请求共享锁的会话共享资源，但排它锁除外。

pg_advisory_unlock(key bigint)

描述：释放会话级别的排它咨询锁。

返回值类型：boolean

pg_advisory_unlock(key1 int, key2 int)

描述：释放会话级别的排它咨询锁。

返回值类型：boolean

备注：pg_advisory_unlock释放先前取得的排它咨询锁。如果释放成功则返回true。如果实际上并未持有指定的锁，将返回false并在服务器中产生一条SQL警告信息。

pg_advisory_unlock_shared(key bigint)

描述：释放会话级别的共享咨询锁。

返回值类型：boolean

版权所有 © 华为云计算技术有限公司

pg_advisory_unlock_shared(key1 int, key2 int)

描述：释放会话级别的共享咨询锁。

返回值类型：boolean

备注：pg_advisory_unlock_shared类似于pg_advisory_unlock，不同之处在于该函数释放的是共享咨询锁。

pg_advisory_unlock_all()

描述：释放当前会话持有的所有咨询锁。

返回值类型：void

备注：pg_advisory_unlock_all将会释放当前会话持有的所有咨询锁，该函数在会话结束的时候被隐含调用，即使客户端异常地断开连接也是一样。

pg_advisory_xact_lock(key bigint)

描述：获取事务级别的排它咨询锁。

返回值类型：void

pg_advisory_xact_lock(key1 int, key2 int)

描述：获取事务级别的排它咨询锁。

返回值类型：void

备注：pg_advisory_xact_lock类似于pg_advisory_lock，不同之处在于锁是自动在当前事务结束时释放，而且不能被显式的释放。

pg_advisory_xact_lock_shared(key bigint)

描述：获取事务级别的共享咨询锁。

返回值类型：void

pg_advisory_xact_lock_shared(key1 int, key2 int)

描述：获取事务级别的共享咨询锁。

返回值类型：void

备注：pg_advisory_xact_lock_shared类似于pg_advisory_lock_shared，不同之处在于锁是在当前事务结束时自动释放，而且不能被显式的释放。

pg_try_advisory_lock(key bigint)

描述：尝试获取会话级排它咨询锁。

返回值类型：boolean

备注：pg_try_advisory_lock类似于pg_advisory_lock，不同之处在于该函数不会阻塞以等待资源的释放。它要么立即获得锁并返回true，要么返回false表示目前不能锁定。

 版权所有 © 华为云计算技术有限公司

pg_try_advisory_lock(key1 int, key2 int)

描述：尝试获取会话级排它咨询锁。

返回值类型：boolean

pg_try_advisory_lock_shared(key bigint)

描述：尝试获取会话级共享咨询锁。

返回值类型：boolean

pg_try_advisory_lock_shared(key1 int, key2 int)

描述：尝试获取会话级共享咨询锁。

返回值类型：boolean

备注：pg_try_advisory_lock_shared类似于pg_try_advisory_lock，不同之处在于该函数尝试获得共享锁而不是排它锁。

pg_try_advisory_xact_lock(key bigint)

描述：尝试获取事务级别的排它咨询锁。

返回值类型：boolean

pg_try_advisory_xact_lock(key1 int, key2 int)

描述：尝试获取事务级别的排它咨询锁。

返回值类型：boolean

备注：pg_try_advisory_xact_lock类似于pg_try_advisory_lock，不同之处在于如果得到锁，在当前事务的结束时自动释放，而且不能被显式的释放。

pg_try_advisory_xact_lock_shared(key bigint)

描述：尝试获取事务级别的共享咨询锁。

返回值类型：boolean

pg_try_advisory_xact_lock_shared(key1 int, key2 int)

描述：尝试获取事务级别的共享咨询锁。

返回值类型：boolean

备注：pg_try_advisory_xact_lock_shared类似于pg_try_advisory_lock_shared，不同之处在于如果得到锁，在当前事务结束时自动释放，而且不能被显式的释放。

21.7.27.6 复制函数

复制函数是系统为实现高可用在各个实例间进行日志同步或数据同步所提供的统计或操作方法。

版权所有 © 华为云计算技术有限公司

说明

除统计查询外的复制函数为内部调用函数，不建议用户直接使用。

pg_create_logical_replication_slot('slot_name', 'plugin_name')

描述：创建逻辑复制槽。该函数在9.1.0.200及之后版本已废弃，为兼容历史版本功能保留该函数。

参数说明：

- slot_name

  流复制槽名称。

  取值范围：字符串，不支持除字母，数字，以及（ _?-. ）以外的字符。

- plugin_name

  插件名称。

  取值范围：字符串，当前只支持“mppdb_decoding”。

返回值类型：name, text

备注：第一个返回值表示slot_name，第二个返回值表示该逻辑复制槽解码的起始LSN位置。

pg_create_physical_replication_slot ('slot_name', isDummyStandby)

描述：创建物理复制槽。

参数说明：

- slot_name

  流复制槽名称。

  取值范围：字符串，不支持除字母，数字，以及（ _?-. ）以外的字符。

- isDummyStandby

  复制槽是否为从备。

  取值范围：布尔值，支持true或false。

返回值类型：name, text

备注：第一个返回值表示slot_name，第二个返回值表示该物理复制槽解码的起始LSN位置。

pg_get_replication_slots()

描述：显示当前DN上所有的复制槽信息。

返回值类型：record

函数返回信息如下：

 版权所有 © 华为云计算技术有限公司

表 21-51 pg_get_replication_slots()字段

名称	类型	描述
slot_name	text	复制槽的名称
plugin	name	逻辑复制槽对应的输出插件名
slot_type	text	复制槽的类型
datoid	oid	复制槽的数据库OID
active	boolean	复制槽是否为激活状态
xmin	xid	复制槽事务标识
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识
restart_lsn	text	复制槽的Xlog文件信息
dummy_standby	boolean	复制槽是否为从备

示例：

```
SELECT * FROM pg_get_replication_slots();
    slot_name     | plugin | slot_type | datoid | active | xmin | catalog_xmin |     restart_lsn     | dummy_standby
------------------+--------+-----------+--------+--------+------+--------------+---------------------+---------------
 gs_roach_common |        | physical  |      0 | f      |      |    602861775 | FFFFFFFF/FFFFFFFF | f
(1 row)
```

pg_drop_replication_slot('slot_name')

描述：删除流复制槽。

参数说明：

slot_name，流复制槽名称。取值范围：字符串，不支持除字母，数字，以及（ _?-. ）以外的字符。

返回值类型：void

pg_logical_slot_peek_changes('slot_name', 'LSN', upto_nchanges, 'options_name', 'options_value')

描述：解码并不推进流复制槽（下次解码可以再次获取本次解出的数据）。该函数在9.1.0.200及之后版本已废弃，为兼容历史版本功能保留该函数。

参数说明：

- slot_name

  流复制槽名称。

  取值范围：字符串，不支持除字母，数字，以及（ _?-. ）以外的字符。

- LSN

  日志的LSN，表示只解码小于等于此LSN的日志。

  取值范围：字符串（ LSN，格式为xlogid/xrecoff ），如'1/2AAFC60'。为NULL时表示不对解码截止的日志位置做限制。

版权所有 © 华为云计算技术有限公司

- upto_nchanges

  解码条数（包含begin和commit）。假设一共有三条事务，分别包含3、5、7条记录，如果upto_nchanges为4，那么会解码出前两个事务共8条记录。解码完第二条事务时发现解码条数记录大于等于upto_nchanges，会停止解码。

  取值范围：非负整数。

  说明

  LSN和upto_nchanges中任一参数达到限制，解码都会结束。

- options：此项为可选参数。
  - include-xids

    解码出的data列是否包含xid信息。

    取值范围：0或1，默认值为1。

    - 0：设为0时，解码出的data列不包含xid信息。
    - 1：设为1时，解码出的data列包含xid信息。
  - skip-empty-xacts

    解码时是否忽略空事务信息。

    取值范围：0或1，默认值为0。

    - 0：设为0时，解码时不忽略空事务信息。
    - 1：设为1时，解码时会忽略空事务信息。
  - include-timestamp

    解码信息是否包含commit时间戳。

    取值范围：0或1，默认值为0。

    - 0：设为0时，解码信息不包含commit时间戳。
    - 1：设为1时，解码信息包含commit时间戳。

返回值类型：text, uint, text

备注：函数返回解码结果，每一条解码结果包含三列，对应上述返回值类型，分别表示LSN位置、xid和解码内容。

pg_logical_slot_get_changes('slot_name', 'LSN', upto_nchanges, 'options_name', 'options_value')

描述：解码并推进流复制槽。该函数在9.1.0.200及之后版本已废弃，为兼容历史版本功能保留该函数。

参数说明： 与pg_logical_slot_peek_changes一致，详细内容请参见**pg_logical_slot_peek_changes('slot_name', 'LSN', upto_nchanges, 'options_name', 'options_value')**。

pg_replication_slot_advance ('slot_name', 'LSN')

描述：直接推进流复制槽到指定LSN，不输出解码结果。

参数说明：

 版权所有 © 华为云计算技术有限公司

- slot_name
  流复制槽名称。
  取值范围：字符串，不支持除字母，数字，以及（ _?-. ）以外的字符。
- LSN
  推进到的日志LSN位置，下次解码时只会输出提交位置比该LSN大的事务结果。如果输入的LSN比当前流复制槽记录的推进位置还要小，则直接返回；如果输入的LSN比当前最新物理日志LSN还要大，则推进到当前最新物理日志LSN。
  取值范围：字符串（ LSN，格式为xlogid/xrecoff ）。

返回值类型：name, text

备注：返回值分别对应slot_name和实际推进至的LSN。

pg_stat_get_data_senders()

描述：显示当前DN上所有的数据页复制发送线程的统计信息。

返回值类型：record

函数返回信息如下：

表 21-52 pg_stat_get_data_senders()字段

名称	类型	描述
pid	bigint	线程pid
sender_pid	integer	当前sender的pid
local_role	text	本地的角色
peer_role	text	对端的角色
state	text	当前sender的复制状态
catchup_start	timestamp with time zone	catchup启动的时间
catchup_end	timestamp with time zone	catchup结束的时间
queue_size	text	数据队列大小
queue_lower_tail	text	数据队列尾1位置
queue_header	text	数据队列头位置
queue_upper_tail	text	数据队列尾2位置
send_position	text	发送端发送的位置
receive_position	text	接收端接收的位置
catchup_type	text	catchup方式为全量还是增量
catchup_bcm_filename	text	catchup当前执行的bcm文件

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
catchup_bcm_finished	integer	catchup已操作完成的bcm文件数量
catchup_bcm_total	integer	catchup总共需要操作的bcm文件数量
catchup_percent	text	catchup已经操作完成的百分比
catchup_remaining_time	text	catchup预估剩余时间

pg_stat_get_wal_senders()

描述：显示当前DN上所有的WAL复制发送线程的统计信息。

返回值类型：record

函数返回信息如下：

表 21-53 pg_stat_get_wal_senders()字段

名称	类型	描述
pid	bigint	线程pid
sender_pid	integer	当前sender的pid
local_role	text	本地的角色
peer_role	text	对端的角色
peer_state	text	对端的状态
state	text	当前sender的复制状态
catchup_start	timestamp with time zone	catchup启动的时间
catchup_end	timestamp with time zone	catchup结束的时间
sender_sent_location	text	发送端发送的LSN位置
sender_write_location	text	发送端write的LSN位置
sender_flush_location	text	发送端flush的LSN位置
sender_replay_location	text	发送端replay的LSN位置
receiver_received_location	text	接收端received的LSN位置

版权所有 © 华为云计算技术有限公司

名称	类型	描述
receiver_write_location	text	接收端write的LSN位置
receiver_flush_location	text	接收端flush的LSN位置
receiver_replay_location	text	接收端replay的LSN位置
sync_percent	text	同步百分比
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步者）
sync_priority	integer	同步复制的优先级（0表示异步）
sync_most_available	text	在备机同步失败时，是否阻塞主机
channel	text	WALSender的信道信息

pg_stat_get_wal_receiver()

描述：显示当前DN上所有的WAL复制接收线程的统计信息。

返回值类型：record

函数返回信息如下：

表 21-54 pg_stat_get_wal_receiver()

名称	类型	描述
receiver_pid	integer	当前receiver的pid
local_role	text	本地的角色
peer_role	text	远端的角色
peer_state	text	远端的状态
state	text	当前receiver的复制状态
sender_sent_location	text	发送端发送的LSN位置
sender_write_location	text	发送端write的LSN位置
sender_flush_location	text	发送端flush的LSN位置
sender_replay_location	text	发送端replay的LSN位置

版权所有 © 华为云计算技术有限公司

名称	类型	描述
receiver_received_location	text	接收端received的LSN位置
receiver_write_location	text	接收端write的LSN位置
receiver_flush_location	text	接收端flush的LSN位置
receiver_replay_location	text	接收端replay的LSN位置
sync_percent	text	同步百分比
channel	text	WALReceiver的信道信息

pg_stat_get_stream_replications()

描述：显示当前DN上所有的复制统计信息。

返回值类型：record

函数返回信息如下：

表 21-55 pg_stat_get_stream_replications()

名称	类型	描述
local_role	text	本地的角色
static_connections	integer	连接统计
db_state	text	数据库状态
detail_information	text	详细信息

示例：

```
SELECT * FROM pg_stat_get_stream_replications();
 local_role | static_connections | db_state | detail_information
------------+--------------------+----------+--------------------
 Normal     |                  0 | Normal   | Normal
(1 row)
```

pg_stat_xlog_space()

描述：显示当前DN上Xlog空间使用信息。

返回值类型：record

函数返回信息如下：

版权所有 © 华为云计算技术有限公司

表 21-56 pg_stat_xlog_space()

名称	类型	描述
xlog_files	bigint	pg_xlog目录下，去除backup、archive_status等子目录，所有识别为xlog文件的数目。
xlog_size	bigint	pg_xlog目录下，去除backup、archive_status等子目录，所有识别为xlog文件的文件大小之和，单位为MB。
other_size	bigint	pg_xlog目录下backup、archive_status等子目录文件的大小之和，单位为MB。

示例：

```
SELECT * FROM pg_stat_xlog_space();
 xlog_files | xlog_size | other_size
------------+-----------+------------
         79 |      1264 |          0
(1 row)
```

pgxc_stat_xlog_space()

描述：显示所有主DN上Xlog空间使用信息。

返回值类型：record

函数返回信息如下：

表 21-57 pgxc_stat_xlog_space()

名称	类型	描述
node_name	name	节点名称
xlog_files	bigint	pg_xlog目录下，去除backup、archive_status等子目录，所有识别为xlog文件的数目。
xlog_size	bigint	pg_xlog目录下，去除backup、archive_status等子目录，所有识别为xlog文件的文件大小之和，单位为MB。
other_size	bigint	pg_xlog目录下backup、archive_status等子目录文件的大小之和，单位为MB。

示例：

```
SELECT * FROM  pgxc_stat_xlog_space();
  node_name   | xlog_files | xlog_size | other_size
--------------+------------+-----------+------------
dn_6001_6002 |         73 |      1168 |          0
dn_6003_6004 |         73 |      1168 |          0
```


版权所有 © 华为云计算技术有限公司

```
dn_6005_6006 |        73 |     1168 |         0
cn_5003      |        79 |     1264 |         0
cn_5001      |        72 |     1152 |         0
cn_5002      |        73 |     1168 |         0
(6 rows)
```

21.7.27.7 资源管理函数

资源管理模块相关函数介绍。

gs_switch_respool(query_id bigint, resource_pool_name name)

描述：作业切换资源池函数，9.1.0及以上集群版本支持。

入参要求resource_pool_name必须小于64字符。调用该函数后，如果作业处于排队阶段，作业切到新资源池会进行判断是否需要启动运行；如果作业处于运行阶段，作业切到新资源池可以启动原资源池的排队作业，同时受到新资源池的CPU管控限制。

该函数同时支持静态管控和动态管控模式下的资源池切换，也支持短查询切换资源池。

返回值类型：bool

⚠ 注意

- 该函数仅支持管理员或者管理员权限用户使用，禁止普通用户使用，该函数的使用前提是集群的CPU管控功能正常，即Cgroup功能正常。可使用pgxc_cgroup_reload_conf()函数查询cgroup是否正常，如果返回结果有非success内容，则需要进行修复。
- 该函数切换资源池后，会按照新资源池的所属异常规则进行管控。
- 当作业处于全局并发排队，即waiting in global queue时，切换资源池后并不会立即生效，此时还未进入资源池排队队列。
- 作业CPU管控切换，支持CN和DN，比如：资源池A占用40个核，资源池B占用5个核，将作业从资源池A切换到资源池B后，作业会在资源池B占用的5个核上执行，此时作业运行会更慢，但是隔离了该作业，不会再影响资源池A原有的作业运行。即实现语句的降级处理。
- 切换资源池时，用户可能会使用如下类似语句进行批量切换，在使用此类语句操作时，需要确保修改的是同一类状态的语句，如切换的都是排队作业或者切换的都是运行作业。如果切换不同类的语句，建议用户分两次进行，如果混合起来一起切换会导致语句运行并发数超限，作业异常唤醒。
  ```
  SELECT gs_switch_respool（queryid,'xxx'）FROM xxx where xxx;
  ```
- 在切换语句资源池时，有以下约束：
  - 切换资源池A运行的作业到资源池B，资源池A的排队队首作业会被唤醒，如果资源池B并发作业数达到上限，作业A1不会直接运行。
  - 切换资源池A排队的作业A2到资源池B，如果资源池B并发作业数达到上限，作业A2会继续排队，如果未达到上限，则会被唤醒，直接运行。
  - 资源池切换时仅支持同车道切换，不支持不同车道切换，如资源池A的快车道切换后会到资源池B的快车道中，资源池A的慢车道切换后会到资源池B的慢车道中，无论资源池B是否开启了短查询加速功能，如资源池B未开启则作业会直接运行。
- 该函数仅支持在所有节点正常的情况下使用，如果节点故障，该函数会执行失败。

版权所有 © 华为云计算技术有限公司

示例：

查询ID为74590868828368101的语句在执行过程中，管理员将其切换至poolg1资源池，切换成功：

```
CALL gs_switch_respool(74590868828368101,'poolg1');
 gs_switch_respool
-------------------
 t
(1 row)
```

gs_increase_except_num(unique_sql_id int8)

描述：作业异常信息记录函数。入参要求必须大于0。调用该函数后会将作业异常次数+1，同时将作业最新异常时间更新为当前时间。

返回值类型：bool

示例：

```
SELECT gs_increase_except_num(111);
gs_increase_except_num
------------------------
t
(1 row)
```

gs_increase_except_num(unique_sql_id int8，except_num int4)

描述：作业异常信息记录函数。入参要求必须大于0。调用该函数后会将作业异常次数加except_num，同时将作业最新异常时间更新为当前时间。

返回值类型：bool

示例：

```
SELECT gs_increase_except_num(111, 4);
gs_increase_except_num
-----------------------
t
(1 row)
```

gs_increase_except_num(unique_sql_id int8，except_num int4, except_time int8)

描述：作业异常信息记录函数。入参要求必须大于0。调用该函数后会将作业异常次数加except_num，同时将作业最新异常时间更新为except_time，except_time为时间戳，该函数主要用于内部调用。

返回值类型：bool

示例：

```
SELECT gs_increase_except_num(111, 4, 714623414421256);
gs_increase_except_num
-----------------------
t
(1 row)
```

gs_append_blocklist(unique_sql_id int8)

描述：用于新增作业黑名单。即在系统表GS_BLOCKLIST_QUERY中新增一个黑名单，并更新GS_BLOCKLIST_QUERY中黑名单信息。

版权所有 © 华为云计算技术有限公司

返回值类型：bool

示例：

```
SELECT gs_append_blocklist(111);
gs_append_blocklist
--------------------
t
(1 row)
```

gs_remove_blocklist(unique_sql_id int8)

描述：用于从作业黑名单中移除一个作业。即从系统表GS_BLOCKLIST_QUERY中移除一个黑名单，并更新GS_BLOCKLIST_QUERY中黑名单信息。

返回值类型：bool

示例：

```
SELECT gs_remove_blocklist(111);
gs_append_blocklist
--------------------
t
(1 row)
```

gs_increase_sql_except_num(sql_hash text)

描述：作业异常信息记录函数。调用该函数后会将作业异常次数+1，同时将作业最新异常时间更新为当前时间。该函数仅9.1.0.200及以上集群版本支持，主要用于内部调用。

返回值类型：bool

示例：

```
SELECT gs_increase_sql_except_num('sql_7bf6381d2a5a456a1936027198ad8b12');
gs_increase_sql_except_num
-------------------------
t
(1 row)
```

gs_increase_sql_except_num(sql_hash text，except_num int4)

描述：作业异常信息记录函数。调用该函数后会将作业异常次数加except_num，同时将作业最新异常时间更新为当前时间。该函数仅9.1.0.200及以上集群版本支持，主要用于内部调用。

返回值类型：bool

示例：

```
SELECT gs_increase_sql_except_num('sql_7bf6381d2a5a456a1936027198ad8b12', 4);
gs_increase_sql_except_num
-------------------------
t
(1 row)
```


版权所有 © 华为云计算技术有限公司

gs_increase_sql_except_num(sql_hash text，except_num int4, except_time int8)

描述：作业异常信息记录函数。调用该函数后会将作业异常次数加except_num，同时将作业最新异常时间更新为except_time。其中，except_time为时间戳。该函数仅9.1.0.200及以上集群版本支持，主要用于内部调用。

返回值类型：bool

示例：

```
SELECT gs_increase_sql_except_num('sql_7bf6381d2a5a456a1936027198ad8b12', 4, 714623414421256);
gs_increase_sql_except_num
----------------------
t
(1 row)
```

gs_append_blocklist(sql_hash text)

描述：用于新增作业黑名单。即在系统表GS_BLOCKLIST_SQL中新增一个黑名单，并更新GS_BLOCKLIST_SQL中黑名单信息。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：bool

示例：

```
SELECT gs_append_blocklist('sql_7bf6381d2a5a456a1936027198ad8b12');
gs_append_blocklist
--------------------
t
(1 row)
```

gs_remove_blocklist(sql_hash text)

描述：用于从作业黑名单中移除一个作业。即从系统表GS_BLOCKLIST_SQL中移除一个黑名单，并更新GS_BLOCKLIST_SQL中黑名单信息。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：bool

示例：

```
SELECT gs_remove_blocklist('sql_7bf6381d2a5a456a1936027198ad8b12');
gs_remove_blocklist
--------------------
t
(1 row)
```

gs_wlm_rebuild_except_rule_hash()

描述：用于重建当前节点的异常规则内存hash表，异常规则的生效阈值由监控线程实时从内存hash获取，hash异常时可以使用该函数进行rebuild恢复。

返回值类型：bool

示例：

```
SELECT gs_wlm_rebuild_except_rule_hash();
gs_wlm_rebuild_except_rule_hash
--------------------
t
(1 row)
```

 版权所有 © 华为云计算技术有限公司

gs_wlm_readjust_user_space(oid)

描述：用户永久存储空间校准函数。入参为用户oid，入参为0的情况下，会校准所有用户的永久存储空间。

返回值类型：text

示例：

```
SELECT gs_wlm_readjust_user_space(0);
gs_wlm_readjust_user_space
----------------------------
Exec Success
(1 row)
```

pgxc_wlm_readjust_schema_space()

描述：Schema永久存储空间校准函数。

返回值类型：text

示例：

```
SELECT pgxc_wlm_readjust_schema_space();
pgxc_wlm_readjust_schema_space
--------------------------------
Exec Success
(1 row)
```

pgxc_wlm_readjust_relfilenode_size_table()

描述：空间统计校准函数，不重建PG_RELFILENODE_SIZE系统表，重新校准用户和schema空间。

说明

由于校准函数执行时与其余业务之间的事务隔离性，校准函数对其它的正在执行的业务不可见，导致校准函数会漏掉此类业务的空间变化，为避免校准之后出现空间差误，建议用户使用空间校准函数时，应选择空间大小稳定无变化的时候执行校准。

返回值类型：text

示例：

```
SELECT pgxc_wlm_readjust_schema_space();
pgxc_wlm_readjust_relfilenode_size_table
-----------------------------------------
Exec Success
(1 row)
```

pgxc_wlm_readjust_relfilenode_size_table(integer)

描述：空间统计校准函数。

说明

由于校准函数执行时与其余业务之间的事务隔离性，校准函数对其它的正在执行的业务不可见，导致校准函数会漏掉此类业务的空间变化，为避免校准之后出现空间差误，建议用户使用空间校准函数时，应选择空间大小稳定无变化的时候执行校准。

参数：取值范围为0~4，不同的入参值代表不同的校准粒度。

版权所有 © 华为云计算技术有限公司

- 入参为0(默认入参为0)：不重建PG_RELFILENODE_SIZE系统表，重新校准用户和schema空间。
- 入参为1：重建PG_RELFILENODE_SIZE系统表，并且重新校准用户和schema空间。
- 入参为2：重建PG_RELFILENODE_SIZE系统表。
- 入参为3：重新校准schema空间。
- 入参为4：重新校准用户空间。

返回值类型：text

示例：

```
SELECT * FROM pgxc_wlm_readjust_relfilenode_size_table(1);
   result
--------------
 Exec success
(1 row)
```

pgxc_wlm_get_schema_space(cstring)

描述：在CN上查询某个逻辑集群下各实例的Schema空间信息，入参为逻辑集群名称。

返回值类型：record

函数返回字段如下：

名称	类型	描述
schemaname	text	模式名
schemaid	oid	模式OID
databasename	text	数据库名
databaseid	oid	数据库OID
nodename	text	实例名称
nodegroup	text	节点组名称
usedspace	bigint	已使用的空间大小
permspace	bigint	空间上限值

示例：

```
SELECT * FROM pgxc_wlm_get_schema_space('group1');
     schemaname      | schemaid | databasename | databaseid |    nodename    |  nodegroup   | usedspace |
permspace
--------------------+----------+--------------+------------+----------------+--------------+-----------+-----------
 pg_catalog         |       11 | test1        |      16384 | datanode1      | installation |   9469952 |        -1
 public             |     2200 | postgres     |      15253 | datanode1      | installation |  25280512 |        -1
 pg_toast           |       99 | test1        |      16384 | datanode1      | installation |   1859584 |        -1
 cstore             |      100 | test1        |      16384 | datanode1      | installation |         0 |        -1
 data_redis         |    18106 | postgres     |      15253 | datanode1      | installation |    655360 |        -1
 data_redis         |    18116 | test1        |      16384 | datanode1      | installation |         0 |        -1
 public             |     2200 | test1        |      16384 | datanode1      | installation |     16384 |        -1
```


版权所有 © 华为云计算技术有限公司

```
dbms_om            |     3987 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_job           |     3988 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_om            |     3987 | test1        |      16384 | datanode1     | installation |         0 |        -1
dbms_job           |     3988 | test1        |      16384 | datanode1     | installation |         0 |        -1
sys                |    11693 | postgres     |      15253 | datanode1     | installation |         0 |        -1
sys                |    11693 | test1        |      16384 | datanode1     | installation |         0 |        -1
utl_file           |    14644 | postgres     |      15253 | datanode1     | installation |         0 |        -1
utl_raw            |    14669 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_sql           |    14674 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_output        |    14662 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_random        |    14666 | postgres     |      15253 | datanode1     | installation |         0 |        -1
dbms_lob           |    14701 | postgres     |      15253 | datanode1     | installation |         0 |        -1
information_schema |    14300 | postgres     |      15253 | datanode1     | installation |    294912 |        -1
information_schema |    14300 | test1        |      16384 | datanode1     | installation |    294912 |        -1
utl_file           |    14644 | test1        |      16384 | datanode1     | installation |         0 |        -1
dbms_output        |    14662 | test1        |      16384 | datanode1     | installation |         0 |        -1
dbms_random        |    14666 | test1        |      16384 | datanode1     | installation |         0 |        -1
utl_raw            |    14669 | test1        |      16384 | datanode1     | installation |         0 |        -1
dbms_sql           |    14674 | test1        |      16384 | datanode1     | installation |         0 |        -1
dbms_lob           |    14701 | test1        |      16384 | datanode1     | installation |         0 |        -1
pg_catalog         |       11 | postgres     |      15253 | datanode1     | installation |  13049856 |        -1
redisuser          |    16387 | postgres     |      15253 | datanode1     | installation |    630784 |        -1
pg_toast           |       99 | postgres     |      15253 | datanode1     | installation |   3080192 |        -1
cstore             |      100 | postgres     |      15253 | datanode1     | installation |   2408448 |        -1
pg_catalog         |       11 | test1        |      16384 | datanode2     | installation |   9469952 |        -1
public             |     2200 | postgres     |      15253 | datanode2     | installation |  25214976 |        -1
pg_toast           |       99 | test1        |      16384 | datanode2     | installation |   1859584 |        -1
cstore             |      100 | test1        |      16384 | datanode2     | installation |         0 |        -1
data_redis         |    18106 | postgres     |      15253 | datanode2     | installation |    655360 |        -1
data_redis         |    18116 | test1        |      16384 | datanode2     | installation |         0 |        -1
public             |     2200 | test1        |      16384 | datanode2     | installation |     16384 |        -1
dbms_om            |     3987 | postgres     |      15253 | datanode2     | installation |         0 |        -1
dbms_job           |     3988 | postgres     |      15253 | datanode2     | installation |         0 |        -1
dbms_om            |     3987 | test1        |      16384 | datanode2     | installation |         0 |        -1
dbms_job           |     3988 | test1        |      16384 | datanode2     | installation |         0 |        -1
```

pgxc_wlm_analyze_schema_space(cstring)

描述：在CN上查询某个逻辑集群下集群整体的Schema空间信息，入参为逻辑集群名称。

返回值类型：record

函数返回字段如下：

名称	类型	描述
schemaname	text	模式名
databasename	text	数据库名
nodegroup	text	节点组名称
total_value	bigint	该模式的集群空间总值
avg_value	bigint	该模式的各实例空间平均值
skew_percent	integer	倾斜率
extend_info	text	提供信息包括：单实例空间最大值、单实例空间最小值以及最大最小空间所在的实例名

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT * FROM pgxc_wlm_analyze_schema_space('group1');
     schemaname      | databasename |  nodegroup    | total_value | avg_value | skew_percent |
extend_info
--------------------+--------------+--------------+-------------+-----------+--------------
+-----------------------------------------------
 pg_catalog          | test1        | installation |    56819712 |   9469952 |            0 | min:9469952
datanode1,max:9469952 datanode1
 public              | postgres     | installation |   150495232 |  25082538 |            0 | min:24903680
datanode6,max:25280512 datanode1
 pg_toast            | test1        | installation |    11157504 |   1859584 |            0 | min:1859584
datanode1,max:1859584 datanode1
 cstore              | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 data_redis          | postgres     | installation |     1966080 |    327680 |           50 | min:0
datanode4,max:655360 datanode1
 data_redis          | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 public              | test1        | installation |       98304 |     16384 |            0 | min:16384
datanode1,max:16384 datanode1
 dbms_om             | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 dbms_job            | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 dbms_om             | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_job            | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 sys                 | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 sys                 | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 utl_file            | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 utl_raw             | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_sql            | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_output         | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 dbms_random         | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 dbms_lob            | postgres     | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 information_schema | postgres     | installation |     1769472 |    294912 |            0 | min:294912
datanode1,max:294912 datanode1
 information_schema | test1        | installation |     1769472 |    294912 |            0 | min:294912
datanode1,max:294912 datanode1
 utl_file            | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_output         | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 dbms_random         | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0
datanode1
 utl_raw             | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_sql            | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 dbms_lob            | test1        | installation |           0 |         0 |            0 | min:0 datanode1,max:0 datanode1
 pg_catalog          | postgres     | installation |    75431936 |  12571989 |            3 | min:12124160
datanode4,max:13049856 datanode1
 redisuser           | postgres     | installation |     1884160 |    314026 |           50 | min:16384
datanode4,max:630784 datanode1
 pg_toast            | postgres     | installation |    17154048 |   2859008 |            7 | min:2637824
datanode4,max:3080192 datanode1
 cstore              | postgres     | installation |    15294464 |   2549077 |            5 | min:2408448
datanode1,max:2703360 datanode6
(31 rows)
```

gs_wlm_set_queryband_action(cstring,cstring,int4)

描述：设置query_band关联行为和次序。

返回值类型：boolean

函数入参字段如下：

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
qband	cstring	query_band键值对，长度上限为63个字符。
action	cstring	query_band关联行为。
order	int4	query_band搜索次序，缺省参数，默认为-1。

示例：

```
SELECT * FROM gs_wlm_set_queryband_action('a=1','respool=p1');
 gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)

SELECT * FROM gs_wlm_set_queryband_action('a=3','respool=p1;priority=rush',1);
 gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

gs_wlm_set_queryband_order(cstring,int4)

描述：设置query_band次序。

返回值类型：boolean

函数入参字段如下：

名称	类型	描述
qband	cstring	query_band键值对。
order	int4	query_band搜索次序，缺省参数，默认为-1。

示例：

```
SELECT * FROM gs_wlm_set_queryband_order('a=1',2);
 gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

gs_wlm_get_queryband_action(cstring)

描述：查询query_band关联行为和次序。

返回值类型：record

函数返回字段如下：

名称	类型	描述
qband	cstring	query_band键值对。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
respool_id	Oid	query band关联资源池OID。
respool	text	query band关联资源池名。
priority	text	query band关联队列内优先级。
qborder	int4	query_band搜索次序。

示例：

```
SELECT * FROM gs_wlm_get_queryband_action('a=1');
qband | respool_id | respool | priority | qborder
-------+------------+---------+----------+---------
 a=1   |      16388 | p1      | Medium   |      -1
(1 row)
```

gs_cgroup_reload_conf()

描述：在当前实例上进行cgroup配置文件在线加载。

返回值类型：record

函数返回字段如下：

名称	类型	描述
node_name	text	实例名称。
node_host	text	实例所在节点的IP地址。
result	text	cgroup在线加载是否成功。

示例：

```
SELECT * FROM gs_cgroup_reload_conf();
 node_name |    node_host    | result
-----------+-----------------+---------
 cn_5001   | 192.168.178.35 | success
```

pgxc_cgroup_reload_conf()

描述：在系统所有实例上进行cgroup配置文件在线加载。

返回值类型：record

函数返回字段如下：

名称	类型	描述
node_name	text	实例名称。
node_host	text	实例所在节点的IP地址。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
result	text	cgroup在线加载是否成功。

示例：

```
SELECT * FROM pgxc_cgroup_reload_conf();
  node_name   |    node_host    | result
--------------+-----------------+---------
 dn_6025_6026 | 192.168.178.177 | success
 dn_6049_6050 | 192.168.179.79  | success
 dn_6051_6052 | 192.168.179.79  | success
 dn_6055_6056 | 192.168.179.79  | success
 dn_6067_6068 | 192.168.181.57  | success
 dn_6023_6024 | 192.168.178.39  | success
 dn_6009_6010 | 192.168.181.21  | success
 dn_6011_6012 | 192.168.181.21  | success
 dn_6015_6016 | 192.168.181.21  | success
 dn_6029_6030 | 192.168.178.177 | success
 dn_6031_6032 | 192.168.178.177 | success
 dn_6045_6046 | 192.168.179.45  | success
 cn_5001      | 192.168.178.35  | success
 cn_5003      | 192.168.178.39  | success
 dn_6061_6062 | 192.168.181.179 | success
 cn_5006      | 192.168.179.45  | success
 cn_5004      | 192.168.178.177 | success
 cn_5002      | 192.168.181.21  | success
 cn_5005      | 192.168.178.187 | success
 dn_6019_6020 | 192.168.178.39  | success
 dn_6007_6008 | 192.168.178.35  | success
 dn_6071_6072 | 192.168.181.57  | success
 dn_6003_6004 | 192.168.178.35  | success
 dn_6013_6014 | 192.168.181.21  | success
 dn_6035_6036 | 192.168.178.187 | success
 dn_6037_6038 | 192.168.178.187 | success
 dn_6001_6002 | 192.168.178.35  | success
 dn_6063_6064 | 192.168.181.179 | success
 dn_6005_6006 | 192.168.178.35  | success
 dn_6057_6058 | 192.168.181.179 | success
 dn_6069_6070 | 192.168.181.57  | success
 dn_6027_6028 | 192.168.178.177 | success
 dn_6059_6060 | 192.168.181.179 | success
 dn_6041_6042 | 192.168.179.45  | success
 dn_6043_6044 | 192.168.179.45  | success
 dn_6047_6048 | 192.168.179.45  | success
 dn_6033_6034 | 192.168.178.187 | success
 dn_6065_6066 | 192.168.181.57  | success
 dn_6021_6022 | 192.168.178.39  | success
 dn_6017_6018 | 192.168.178.39  | success
 dn_6039_6040 | 192.168.178.187 | success
 dn_6053_6054 | 192.168.179.79  | success
(42 rows)
```

pgxc_cgroup_reload_conf(text)

描述：在某个节点上进行cgroup配置文件在线加载。入参为节点的IP地址。

返回值类型：record

函数返回字段如下：

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
node_name	text	实例名称。
node_host	text	实例所在节点的IP地址。
result	text	cgroup在线加载是否成功。

示例：

```
SELECT * FROM pgxc_cgroup_reload_conf('192.168.178.35');
  node_name   |   node_host    | result
--------------+----------------+---------
 cn_5001      | 192.168.178.35 | success
 dn_6007_6008 | 192.168.178.35 | success
 dn_6003_6004 | 192.168.178.35 | success
 dn_6001_6002 | 192.168.178.35 | success
 dn_6005_6006 | 192.168.178.35 | success
(5 rows)
```

gs_wlm_node_recover(boolean isForce)

描述：动态资源管理模式下，对CCN管控计数和作业信息进行更新恢复。该函数仅支持管理员执行，通常用于CN实例故障重启后的实例恢复，由集群管理组件（CM）调用，不建议用户直接调用。具体功能如下：

- CN执行：通知CCN清理该CN执行的作业信息和作业对应的管控计数信息。
- CCN执行：重置管控计数信息，并从CN上获取最新的慢车道作业信息。

返回值类型：bool

gs_wlm_node_clean(cstring nodename)

描述：动态资源管理模式下，对CCN上入参指定的CN执行的作业信息和对应的管控计数信息进行清理。该函数仅支持管理员执行，通常用于CN实例故障重启后的实例恢复，由集群管理组件（CM）调用，正常情况下不建议用户直接调用。

返回值类型：bool

pg_stat_get_wlm_node_resource_info(int4)

描述：显示所有DN资源的汇总信息。

返回值类型：record

函数返回字段如下：

名称	类型	描述
min_mem_util	integer	DN最小内存使用率。
max_mem_util	integer	DN最大内存使用率。
min_cpu_util	integer	DN最小CPU使用率。
max_cpu_util	integer	DN最大CPU使用率。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_io_util	integer	DN最小IO使用率。
max_io_util	integer	DN最大IO使用率。
phy_usemem_rate	integer	物理节点最大内存使用率。

pg_stat_get_workload_struct_info()

描述：定位CCN排队问题的负载管理函数。此函数为内部函数，如需使用请联系技术支持工程师。

返回值类型：record

pgxc_query_resource_info(query_id bigint)

描述：显示指定query_id对应的语句在所有DN上的资源监控信息。该函数仅9.1.0.100及以上集群版本支持。

返回值类型：record

函数返回字段如下：

名称	类型	描述
node_name	text	实例名称，只包含DN。
user_id	oid	用户ID。
queryid	bigint	语句执行使用的内部query id。
used_mem	int	语句在当前DN使用的内存大小，单位MB。
cpu_time	bigint	语句在当前DN上的CPU时间，单位ms。
used_cpu	double	语句在当前DN使用的CPU个数。
spill_size	bigint	若发生下盘，语句在当前DN下盘数据量，默认为0。单位MB。
read_bytes	bigint	语句在当前DN使用的逻辑读字节数，单位KB。
write_bytes	bigint	语句在当前DN使用的逻辑写字节数，单位KB。
read_count	bigint	语句在当前DN使用的逻辑读次数。
write_count	bigint	语句在当前DN使用的逻辑写次数。
read_speed	int	语句在当前DN使用的逻辑读速率，单位KB/s。
write_speed	int	语句在当前DN使用的逻辑写速率，单位KB/s。
curr_iops	int	语句在当前DN上的每秒IO数值（列存单位是次/s，行存单位是万次/s）。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
send_pkg	bigint	语句在当前DN上的通信包发送总量，单位packages。
recv_pkg	bigint	语句在当前DN上的通信包接收总量，单位packages
send_bytes	bigint	语句在当前DN上的通信流发送数据总量，单位Byte。
recv_bytes	bigint	语句在当前DN上的通信流接收数据总量，单位Byte。
send_speed	int	语句在当前DN使用的网络发送速率，单位KB/s。
recv_speed	int	语句在当前DN使用的网络接收速率，单位KB/s。

21.7.27.8 内存管理函数

内存管理函数仅9.1.0及以上集群版本支持。

pg_shared_chunk_detail(contextname char(64))

描述：查询指定共享内存下内存上下文申请的所有chunk信息。

参数contextname，表示内存上下文名称。

使用该函数需先使用pg_shared_chunk_dump(contextname char(64))函数将相关信息打印成文件。

返回值类型：record

示例：

```
SELECT * FROM pg_shared_chunk_detail('pgstat file hash table');
       parent        | level |  file_name   | line_number | chunk_size | requested_number | total_size
---------------------+-------+--------------+-------------+------------+------------------+------------
 pgstat file context |     2 | dynahash.cpp |         158 |       2048 |                1 |       2048
 pgstat file context |     2 | dynahash.cpp |         158 |        256 |                1 |        256
 pgstat file context |     2 | dynahash.cpp |         158 |       4096 |                9 |      36864
 pgstat file context |     2 | dynahash.cpp |         158 |       8192 |                4 |      32768
(4 rows)
```

pv_session_chunk_detail(tid int, contextname char(64))

描述：查询指定线程创建的某个内存上下文申请的所有chunk信息。

参数tid，表示线程ID；参数contextname，表示内存上下文名称。

使用该函数需使用pv_session_chunk_dump(tid int, contextname char(64))函数将相关信息打印成文件。

返回值类型：record

示例：

```
SELECT * FROM pv_session_chunk_detail(140178810990936, 'Timezones');
      parent      | level |   file_name   | line_number | chunk_size | requested_number | total_size
------------------+-------+---------------+-------------+------------+------------------+------------
```


版权所有 © 华为云计算技术有限公司

```
 TopMemoryContext |     1 | dynahash.cpp  |        158 |       1280 |             2 |       2560
 TopMemoryContext |     1 | dynahash.cpp  |        158 |        160 |             1 |        160
 TopMemoryContext |     1 | dynahash.cpp  |        158 |       2560 |             2 |       5120
 TopMemoryContext |     1 | localtime.cpp |       1965 |        128 |             1 |        128
 TopMemoryContext |     1 | localtime.cpp |       1965 |        448 |             1 |        448
(5 rows)
```

pg_shared_chunk_dump(contextname char(64))

描述：将指定共享内存下内存上下文申请的所有chunk信息打印成文件。

参数contextname，表示内存上下文名称。

返回值类型：boolean

示例：

```
SELECT * FROM pg_shared_chunk_dump('pgstat file hash table');
 pg_shared_chunk_dump
----------------------
         t
(1 row)
```

pv_session_chunk_dump(tid int, contextname char(64))

描述：将指定线程创建的某个内存上下文申请的所有chunk信息打印成文件。

参数tid，表示线程ID；参数contextname，表示内存上下文名称。

返回值类型：boolean

示例：

```
SELECT * FROM pv_session_chunk_dump(140472797325280, 'Timezones');
 pv_session_chunk_dump
-----------------------
 t
(1 row)
```

21.7.27.9 其它函数

pgxc_pool_check()

描述：检查连接池中缓存的连接数据是否与pgxc_node一致。8.3.0及以上集群版本中该参数已废弃。

返回值类型：boolean

pgxc_pool_reload()

描述：更新连接池中缓存的连接信息。

返回值类型：boolean

pgxc_lock_for_backup()

描述：备份前给集群加锁，备份是为了在新增节点上执行恢复操作。

返回值类型：boolean

版权所有 © 华为云计算技术有限公司

说明

pgxc_lock_for_backup是在使用gs_dump或gs_dumpall工具备份集群前，用来给集群加锁的。当给集群加锁后，不允许有改变系统结构的操作。该函数不影响DML语句。

pg_pool_validate(clear boolean, co_node_name cstring)

描述：清理CN上无效的后台线程（这些后台线程持有无效的pooler连接，这里无效的pooler连接指的是连接到当前DN备实例的连接）。

返回值类型：record

pg_nodes_memory()

描述：查看所有节点的内存占用。

返回值类型：record

table_skewness(text)

描述：查看表数据在所有节点的占比。

参数：表示待查询表的表名，为text类型。

返回值类型：record

table_skewness(table_name text, column_name text[, row_num text])

描述：查看表里某列数据，按hash分布规则，在各节点的占比情况。结果以数据节点上的数据量排序。

参数：table_name为表名；column_name为列名；row_num表示查看当前列所有数据，可缺省，默认为0。非0时，表示抽取指定条数的数据查看占比情况（每次采样结果可能不相同，只保证采集row_num条数据进行占比计算）。

返回值类型：record

示例：

根据tx表中的a列，按hash分布，则在1，2，0编号的DN上分布的数据量分别为7条，2条，1条。

```
SELECT * FROM table_skewness('tx','a');
 seqnum | num |  ratio
--------+-----+----------
 1      | 7   | 70.000%
 2      | 2   | 20.000%
 0      | 1   | 10.000%
(3 row)
```

table_data_skewness(data_row record, locatorType "char")

描述：计算指定表中列拼接出的record，对应的桶分布索引。

参数：data_row表示指定表中列拼接出的record，locatorType表示分布规则，当前建议指定'H'，按hash分布计算。

返回值类型：smallint

示例：

 版权所有 © 华为云计算技术有限公司

计算tx表中a列拼接的record，按照hash分布规则对应的桶分布索引。

```
select a, table_data_skewness(row(a), 'H') from tx;
 a | table_data_skewness
---+---------------------
 3 |                   0
 6 |                   2
 7 |                   2
 4 |                   1
 5 |                   1
(5 rows)
```

table_distribution(schemaname text, tablename text)

描述：查看指定表在各个节点上占用的存储空间。

参数：表示待查询表的模式名和表名，均为text类型。

返回值类型：record

说明

- 使用本函数查询指定表存储分布信息，需要具备指定表的SELECT权限。
- table_distribution性能比table_skewness更优，尤其是在大集群大数据量场景下，请优先考虑使用table_distribution函数。
- 当使用table_distribution并希望直观的看到空间占比时，可使用dnsize/(sum(dnsize) over ())的方式查看出具体的占比情况。

table_distribution(regclass)

描述：查看指定表在各个节点上占用的存储空间。

参数：表示待查询表的表名或OID，表名可以有模式名限定。为regclass类型。

返回值类型：record

说明

- 使用本函数查询指定表存储分布信息，需要具备指定表的SELECT权限。
- table_distribution性能比table_skewness更优，尤其是在大集群大数据量场景下，请优先考虑使用table_distribution函数。
- 当使用table_distribution并希望直观的看到空间占比时，可使用dnsize/(sum(dnsize) over ())的方式查看出具体的占比情况。

table_distribution()

描述：查看当前库中所有表在各节点的存储空间分布情况。

返回值类型：record

说明

- 使用本函数涉及全库表信息查询，需要具备管理员权限或预置角色gs_role_read_all_stats权限。
- 当前基于table_distribution()函数，DWS提供视图PGXC_GET_TABLE_SKEWNESS进行数据倾斜查询，建议在数据库中表数量（小于10000）较少的场景直接使用。

版权所有 © 华为云计算技术有限公司

gs_table_distribution(schemaname text, tablename text)

描述：快速查看指定表在各个节点上占用的存储空间。

返回值类型：record

表 21-58 gs_table_distribution(schemaname text, tablename text)函数返回字段

名称	类型	描述
schemaname	name	模式名称。
tablename	name	表名。
relkind	character	类型。 ● i：索引 ● r：表
relpersistence	character	类型。 ● t：本地临时表 ● g：全局临时表 ● u：unlogged表 ● p：普通表
nodename	name	节点名称。
dnsize	bigint	表在该节点上的存储空间大小，单位：字节。
sessionid	bigint	对于全局临时表显示每个会话的session id。

说明

- 使用本函数查询指定表存储分布信息，需要具备指定表的SELECT权限。
- 该函数基于PG_RELFILENODE_SIZE系统表上的物理文件存储空间记录，需确保GUC参数use_workload_manager和enable_perm_space必须开启。
- 性能上，单表查询时，gs_table_distribution函数低于table_distribution函数；在全库表查询时，gs_table_distribution函数大幅度优于table_distribution函数；在大集群大数据量场景下，如果进行全库表查询，建议优先使用gs_table_distribution函数。
- 全局临时表可以显示每个会话在存储空间分布情况。通过sessionid列区分不同会话。

gs_table_distribution()

描述：快速查看当前库中所有表在各节点的存储空间分布情况。

返回值类型：record

版权所有 © 华为云计算技术有限公司

表 21-59 gs_table_distribution()函数返回字段

名称	类型	描述
schemaname	name	模式名称。
tablename	name	表名。
relkind	character	类型，i: 索引，r: 表。
relpersistence	character	类型。 ● t：本地临时表 ● g：全局临时表 ● u：unlogged表 ● p：普通表
nodename	name	节点名称。
dnsize	bigint	表在该节点上的存储空间大小，单位：字节。
sessionid	bigint	对于全局临时表显示每个会话的session id。

说明

- 使用本函数查询指定表存储分布信息，需要具备指定表的SELECT权限。
- 该函数基于PG_RELFILENODE_SIZE系统表上的物理文件存储空间记录，需确保GUC参数use_workload_manager和enable_perm_space必须开启。
- 性能上，单表查询时，gs_table_distribution函数低于table_distribution函数；在全库表查询时，gs_table_distribution函数大幅度优于table_distribution函数；在大集群大数据量场景下，如果进行全库表查询，建议优先使用gs_table_distribution函数。
- 基于gs_table_distribution()函数，DWS自8.2.1集群版本提供了视图PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS进行数据倾斜查询，建议在数据库中表数量（小于10000）较少的场景直接使用。

check_foreign_key_constraint(schema.table)

描述：检测外键表中的数据是否满足外键约束。该参数仅9.1.0.200及以上集群版本支持。

返回值类型：text

示例：

```
set info_constraint_options = 'foreign_key';

CREATE TABLE ptt1 (a int,b int,e varchar(50),f varchar(50),PRIMARY KEY (a,b)) DISTRIBUTE BY HASH(a,b);
CREATE TABLE tt2 (a int , b int, c int, d int) DISTRIBUTE BY HASH(a,b);

select check_foreign_key_constraint('table_constraints.tt2');
check_foreign_key_constraint
----------------------------------------
tt2 table has no foreign key constraint

ALTER TABLE tt2 ADD CONSTRAINT fk_tt1_a FOREIGN KEY (a,b) REFERENCES ptt1(a,b);
```

 版权所有 © 华为云计算技术有限公司

```
select check_foreign_key_constraint('table_constraints.tt2');
check_foreign_key_constraint
----------------------------------------
tt2 satisfied all Foreign key constraints
```

plan_seed()

描述：获取前一次查询语句的seed值（内部使用）。

返回值类型：integer

pg_stat_get_env()

描述：提供获取当前节点的环境变量信息。

返回值类型：record

pg_stat_get_thread()

描述：提供当前节点下所有线程的状态信息。

返回值类型：record

pgxc_get_os_threads()

描述：提供当前集群中所有正常节点下的线程状态信息。

返回值类型：record

pg_stat_get_sql_count()

描述：提供当前节点中所有用户执行的SELECT/UPDATE/INSERT/DELETE/MERGE INTO语句的计数和响应时间的统计结果以及DDL、DML、DCL语句的计数结果。

返回值类型：record

pgxc_get_sql_count()

描述：提供当前集群所有节点中所有用户执行的SELECT/UPDATE/INSERT/DELETE/MERGE INTO语句的计数和响应时间的统计结果以及DDL、DML、DCL语句的计数结果。

返回值类型：record

pgxc_get_workload_sql_count()

描述：提供当前集群所有CN上所有Workload控制组内执行的SELECT/UPDATE/INSERT/DELETE语句的计数统计结果以及DDL、DML、DCL类型语句的计数统计结果。

返回值类型：record

pgxc_get_workload_sql_elapse_time()

描述：提供当前集群所有CN上所有Workload控制组内执行的SELECT/UPDATE/INSERT/DELETE语句的响应时间统计结果。

 版权所有 © 华为云计算技术有限公司

返回值类型：record

get_instr_unique_sql()

描述：提供当前节点中收集的Unique SQL的信息。如果是CN节点，将返回该CN上收集的Unique SQL的完整信息，即会收集其他CN和DN上对应Unique SQL的信息并进行汇总展示；如果是DN节点，将返回本DN节点上的Unique SQL信息。详见视图GS_INSTR_UNIQUE_SQL。

返回值类型：record

reset_instr_unique_sql(cstring, cstring, INT8)

描述：清理已收集的Unique SQL信息。输入参数含义如下：

- GLOBAL/LOCAL：清理范围为所有节点或当前节点。
- ALL/BY_USERID/BY_CNID/BY_GUC：ALL表示清理所有，BY_USERID/BY_CNID表示按照USERID或CNID进行清理，BY_GUC表示清理操作是由GUC参数instr_unique_sql_count设置值变小引起的。
- 第三个参数值对应第二个参数设置，ALL/BY_GUC的情况下该值无意义。

返回值类型：bool

pgxc_get_instr_unique_sql()

描述：提供集群中所有CN上收集的Unique SQL的完整信息。该函数只能在CN上执行。

返回值类型：record

get_instr_unique_sql_remote_cns()

描述：提供集群中除正在执行此函数的CN之外的所有CN上收集的Unique SQL的完整信息。该函数只能在CN上执行。

返回值类型：record

pgxc_get_node_env()

描述：提供获取集群中所有节点的环境变量信息。

返回值类型：record

gs_switch_relfilenode()

描述：交换两个表或分区的元信息（重分布工具内部使用，用户直接使用会有错误信息提示）。

返回值类型：integer

copy_error_log_create()

描述：创建COPY FROM容错机制所需要的错误表（public.pgxc_copy_error_log）。

返回值类型：boolean

 版权所有 © 华为云计算技术有限公司

说明

- 此函数会尝试创建public.pgxc_copy_error_log表，表的详细信息请参见**表21-60**。
- 在relname列上创建B-Tree索引，并REVOKE ALL on public.pgxc_copy_error_log FROM public对错误表进行权限控制（与COPY语句权限一致）。
- 由于尝试创建的public.pgxc_copy_error_log定义是一张行存表，因此集群上必须支持行存表的创建才能够正常运行此函数，并使用后续的COPY容错功能。需要特别注意的是，enable_hadoop_env这个GUC参数开启后会禁止在集群内创建行存表（DWS默认为off）。
- 此函数自身权限为Sysadmin及以上（与错误表、COPY权限一致）。
- 若创建前public.pgxc_copy_error_log表已存在或者copy_error_log_relname_idx索引已存在，则此函数会报错回滚。

表 21-60 错误表 public.pgxc_copy_error_log 信息

列名称	类型	描述
relname	varchar	表名称。以模式名.表名形式显示。
begintime	timestamp with time zone	出现数据格式错误的时间。
filename	varchar	出现数据格式错误的数据源文件名。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。为了防止字段长度过大，限制字段的长度不超过1024 byte。
detail	text	详细错误信息。
columnname	varchar	在数据源文件中，出现数据格式错误的字段名称。仅8.2.1.100及以上版本支持。
errcode	varchar	出错信息对应的错误码，使用sqlstate错误码。仅8.2.1.100及以上版本支持。
queryid	bigint	执行copy的sql语句id，用于唯一标识一条sql语句。仅8.2.1.100及以上版本支持。

pv_compute_pool_workload()

描述：提供云上加速集群当前负载信息。

返回值类型：record

pg_stat_get_status(tid, num_node_display)

描述：查询当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况，其返回结果的详细含义参见PG_THREAD_WAIT_STATUS视图。输入参数含义如下：

版权所有 © 华为云计算技术有限公司

- tid：表示线程ID，bigint类型。如果为NULL，则返回所有工作线程和辅助线程的等待情况；否则只返回指定ID线程的等待情况。
- num_node_display：integer类型。对于等待状态为“wait node”的记录，指定其wait_status列中显示的被等待节点的最大数量。
  - 如果为空或者小于等于0，则只显示一个被等待节点。
  - 如果大于20，则最多只显示20个节点。
  - 如果大于0且小于等于20，则显示数量为num_node_display和实际被等待节点数量的最小者。例如查询“SELECT * from pg_stat_get_status(NULL, 10)”，如果实际被等待节点数量大于10，则只随机显示其中10个节点名称，如果实际被等待节点数量小于等于10，则显示全部被等待节点名称。当实际被等待节点数量大于显示数量时，被显示的节点名称为随机挑选。

返回值类型：record

pgxc_get_thread_wait_status(num_node_display)

描述：查询集群各个节点上所有SQL语句产生的线程之间的调用层次关系，以及各个线程的阻塞等待状态。其返回结果的详细含义同PGXC_THREAD_WAIT_STATUS视图。输入参数num_node_display的类型和含义与上述函数pg_stat_get_status相同。

返回值类型：record

pgxc_os_run_info()

描述：查询集群中各节点上操作系统运行的状态信息。函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PV_OS_RUN_INFO”章节。

返回值类型：record

get_instr_wait_event()

描述：查询当前实例上各类等待状态和事件的统计信息。函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>GS_WAIT_EVENTS”。如果GUC参数enable_track_wait_event为off，则返回0行。

返回值类型：record

pgxc_wait_events()

描述：查询集群中各节点上各类等待状态和事件的统计信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PGXC_WAIT_EVENTS”视图。如果GUC参数enable_track_wait_event为off，则返回0行。

返回值类型：record

pgxc_stat_bgwriter()

描述：查询集群中各节点上后端写进程活动的统计信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PG_STAT_BGWRITER”视图。

返回值类型：record

 版权所有 © 华为云计算技术有限公司

pgxc_stat_replication()

描述：查询集群中各节点上日志同步的状态信息，如发起端发送日志位置，收端接收日志位置等，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PG_STAT_REPLICATION”视图。

返回值类型：record

pgxc_replication_slots()

描述：查询集群中各DN上复制的状态信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PG_REPLICATION_SLOTS”视图。

返回值类型：record

pgxc_settings()

描述：查询集群中各节点上运行时参数的相关信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PG_SETTINGS”视图。

返回值类型：record

pgxc_instance_time()

描述：查询集群中各节点的运行时间统计信息及各执行阶段所消耗时间，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PV_INSTANCE_TIME”视图。

返回值类型：record

pg_stat_get_redo_stat()

描述：查询当前节点上的XLOG重做统计信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PV_REDO_STAT”视图。

返回值类型：record

pgxc_redo_stat()

描述：查询集群中各节点上的XLOG重做统计信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>PV_REDO_STAT”视图。

返回值类型：record

get_local_rel_iostat()

描述：查询当前实例上磁盘读写的统计信息。函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>GS_REL_IOSTAT”视图。

返回值类型：record

pgxc_rel_iostat()

描述：查询集群中各节点上磁盘读写的统计信息，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>GS_REL_IOSTAT”视图。

版权所有 © 华为云计算技术有限公司

返回值类型：record

get_node_stat_reset_time()

描述：返回当前实例统计信息被重置的时间。

返回值类型：timestamptz

pgxc_node_stat_reset_time()

描述：查询集群中各节点统计信息被重置的时间，函数返回结果信息详见《开发指南》中“系统表和系统视图>系统视图>GS_NODE_STAT_RESET_TIME”视图。

返回值类型：record

说明

实例正常运行过程中，内存中的各类统计数值会逐渐累加，以下情况会导致内存中的统计数值被重置为0：

- 实例重启或集群发生了切换；
- 数据库Database被删除（drop）；
- 用户执行了重置操作，如执行pgstat_recv_resetcounter函数会将数据库中的统计计数器清零，执行reset_instr_unique_sql函数会将Unique SQL数据清零。

如果发生了以上事件，DWS会记录统计信息被重置的时间，可通过get_node_stat_reset_time函数查询。

pgxc_parallel_query(text, text)

描述：在指定类型的数据实例上执行指定的SQL查询语句，并把查询语句的结果返回给当前CN。该函数8.1.2及以上版本支持。

函数有两个参数：

第一个参数：SQL语句在哪些实例上执行。当前支持的有效入参是 'dn'，'datanode'，'cn'， 'coordinator'， 'all'， 其它值会导致函数执行报错。其中 'dn'，'datanode'表示在所有DN上执行，'cn'， 'coordinator'表示在所有CN上执行， all表示在所有的CN和所有的DN上执行。

第二个参数：需要发往远程节点执行的SQL语句，函数内部会对SQL语句中查询的对象进行校验，不支持用户表、分布式表和自定义的多结果集函数。

返回值类型：record

 版权所有 © 华为云计算技术有限公司

说明

- 该函数功能仅为便于开发人员高效的收集集群内实例执行信息或者状态的视图，不建议用户直接使用。
- 该函数为多结果集函数，返回的数据类型为record，所以需要在函数调用后面增加AS语句显示的指出输出的列名和数据类型，如下所示：
  ```
  SELECT * FROM pgxc_parallel_query('all', 'select node_name, db_name, thread_name, query_id, tid, lwtid, ptid, tlevel, smpid, wait_status, wait_event from pg_thread_wait_status') AS (node_name text, db_name text, thread_name text, query_id bigint, tid bigint, lwtid integer , ptid integer, tlevel integer , smpid integer, wait_status text, wait_event text);
  ```
- 函数第二个参数指定的SQL语句输出结果的数据类型必须跟AS后面指明的数据类型一致，否则执行时可能会因为类型不匹配而报错。
- 函数第二个参数指定的SQL语句中不能触发跨节点的查询动作，否则会触发执行报错。
- 函数第二个参数指定的SQL语句只能是SELECT/UPDATE/DELETE/INSERT语句中的一种，且
  - 不支持语句中有returning语句。
  - 调用函数用户需具有SQL语句中对象相应的操作权限。
  - 如果是INSERT语句，不支持INSERT OVERWRITE、UPSERT、和INSERT INTO。
  - 对于UPDATE/DELETE/INSERT语句，只允许初始化用户在就地升级模式下或者管理员用户在重分布模式下执行；要求语句在每个实例上修改的记录数相同，否则执行会报错；函数会输出一条一列bigint类型的数值，此数值表示语句在每个实例上操作的记录数。
    ```
    SELECT * FROM pgxc_parallel_query('cn', 'UPDATE pg_partition SET relpages = 0') AS (updated bigint);
    ```

create_wdr_snapshot()

描述：创建性能数据快照。

返回值类型：text

说明

- 该函数只有数据库管理员SYSADMIN才可以执行，非管理员执行会提示无权限。
- 该函数只能在CN上执行，在DN上执行会返回：“WDR snapshot can only be created on coordinator.”。
- 执行该函数前需确认enable_wdr_snapshot参数处于开启状态。如果enable_wdr_snapshot为off，执行该函数会返回：“WDR snapshot request can't be executed, because GUC parameter 'enable_wdr_snapshot' is off.”。
- 如果执行该函数时，快照线程由于节点重启等原因尚未启动，会提示错误：“WDR snapshot request can not be accepted, please retry later.”。
- 如果执行该函数失败，会提示：“Cannot respond to WDR snapshot request.”。
- 如果执行成功，会返回：“WDR snapshot request has been submitted.”。该提示表明创建快照请求已发送至后台快照线程，但不代表创建快照成功。

kill_snapshot(scope cstring)

描述：中止后台快照线程。该函数向后台快照线程发送中止信号并等待线程结束。

输入参数scope：表示操作范围。该参数取值范围为local和global。

- local表示中止当前CN上的快照线程。
- global表示不仅会中止当前CN上的快照线程，还会向集群中所有其他CN发送中止快照线程的请求，即中止集群中所有CN上的快照线程。

 版权所有 © 华为云计算技术有限公司

- 如果输入其他值，则报错“Scope is invalid, use "local" or "global".”。
- 输入参数可为空，表示默认取值为local。

返回值类型：无

说明

- 该函数只有数据库管理员SYSADMIN才有权执行，非管理员执行会提示无权限。
- 该函数只能在CN上执行，在DN上执行会提示：“kill_snapshot can only be executed on coordinator.”；
- 执行该函数会向后台快照线程发送中止信号并等待其结束。如果100s内快照线程仍未中止则会报错：“Kill snapshot thread failed”；

generate_wdr_report(begin_snap_id bigint, end_snap_id bigint, report_type cstring, report_scope cstring, node_name cstring)

描述：创建负荷分析报告。

输入参数含义如下：

- begin_snap_id，end_snap_id：生成报告的起止快照ID，bigint型，要求begin_snap_id < end_snap_id，并且起止快照的时间没有交集。判断快照时间是否有交集可通过查询dbms_om.snapshot表格select s1.end_ts < s2.start_ts from (select * from dbms_om.snapshot where snapshot_id=end_snap_id) as s2;，如果返回true则没有交集，反之有交集。
- report_type：报告类型，cstring型，包括“summary”，“detail”，“all”三种类型。
- report_scope：报告范围，cstring型，包括“cluster”和“node”两种。
- node_name：节点名称，cstring型，如果report_scope="node"，该参数必须是pg_catalog. pgxc_node表格中node_name字段里的CN或DN节点名称。

返回值类型：text

说明

- 该函数只有数据库管理员SYSADMIN才有权执行，非管理员执行会提示无权限。
- 该函数只能在CN上执行，在DN上执行会提示错误：“WDR report can only be created on coordinator.”。
- 如果生成报告成功，会返回：“Report %s has been generated”。
- 生成报告的两个快照期间不能发生统计信息重置事件，否则会提示错误：“Instance reset time is different”。引起统计信息重置的事件参见**pgxc_node_stat_reset_time**函数。

wdr_xdb_query(db_name text, snapshot_id bigint, view_name text)

描述：查询指定数据库下的指定视图。有的视图在不同数据库中查询结果不同，例如global_table_stat视图用于查询表格的统计信息，由于不同数据库下表格不同，在不同数据库中查询该视图得到的结果也不同。wdr_xdb_query函数可以在当前连接中访问db_name指定的数据库，并在该数据库中查询view_name指定的视图。输入参数含义如下：

- db_name：指定的数据库名称，text型。
- snapshot_id：快照ID，bigint型，参见《数据仓库服务开发指南》中“性能视图 快照”章节。

版权所有 © 华为云计算技术有限公司

- view_name：指定视图名称，text型。视图名称必须在如下白名单中：
  - global_table_stat
  - global_table_change_stat
  - global_column_table_io_stat
  - global_row_table_io_stat

  返回值类型：record，其第一列为snapshot_id bigint，第二列为db_name text，其他列的名称、类型和顺序与view_name指定的视图相同。

  示例：

  ```
  select snapshot_id, db_name, schemaname, relname, distribute_mode,
  seq_scan ,seq_tuple_read ,index_scan ,index_tuple_read ,tuple_inserted
  ,tuple_updated ,tuple_deleted ,tuple_hot_updated ,live_tuples ,dead_tuples from
  wdr_xdb_query('postgres'::text, 1, 'global_table_stat'::text) as i(snapshot_id bigint, db_name text,
  schemaname name, relname name, distribute_mode char, seq_scan bigint, seq_tuple_read
  bigint, index_scan bigint, index_tuple_read bigint, tuple_inserted bigint, tuple_updated bigint,
  tuple_deleted bigint, tuple_hot_updated bigint, live_tuples bigint, dead_tuples bigint);
  ```

  说明

  - 该函数仅8.1.2及以上版本支持。
  - 该函数只有数据库管理员SYSADMIN才有权执行，非管理员执行会提示无权限。
  - 该函数只能查询白名单中的视图，如果查询其他视图，会报错：“Input view name is invalid.”。

vac_fileclear_relation(oid)

描述：用于强制清理指定列存表中被VACUUM重写的文件，完成空间回收。

参数：列存表oid。

返回值类型：integer

说明

- 使用本函数前需要设置参数colvacuum_threshold_scale_factor，并确保VACUUM对指定列存表的文件完成重写后，才会清理文件并回收空间。
- 本函数将对指定的列存表施加排他锁。
- 延迟DDL开启后，此函数无法使用。

vac_fileclear_all_relation()

描述：用于强制清理所有列存表中被VACUUM重写的文件，完成空间回收。

返回值类型：record

说明

延迟DDL开启后，此函数无法使用。

get_col_file_info(table_name)

描述：查询指定列存表的空CU文件数和总CU文件数，该函数仅8.2.0及以上集群版本支持。

参数：列存表名。

 版权所有 © 华为云计算技术有限公司

返回值类型：int

返回值字段：

- total_file_num int：总CU文件数，范围为-1~INT_MAX，返回-1时表示表类型不支持等原因的失败，0~INT_MAX为总文件数。
- empty_file_num int：空CU文件数，范围为-1~INT_MAX，返回-1时表示表类型不支持等原因的失败，0~INT_MAX为空文件数。

示例：

```
call get_col_file_info('t4');
 total_file_num | empty_file_num
----------------+----------------
             10 |              7
(1 row)
```

get_all_col_file_info()

描述：查询所有列存表的空CU文件数和总CU文件数，该函数仅8.2.0及以上集群版本支持。

返回值类型：record

返回值字段：

- space_name text：列存表所属schema
- table_name text：列存表名
- total_file_num int：总CU文件数，范围为-1~INT_MAX，返回-1时表示表类型不支持等原因的失败，0~INT_MAX为总文件数。
- empty_file_num int：空CU文件数，范围为-1~INT_MAX，返回-1时表示表类型不支持等原因的失败，0~INT_MAX为空文件数。

示例：

```
call get_all_col_file_info();
 space_name | table_name | total_file_num | empty_file_num
------------+------------+----------------+----------------
 public     | t4         |             10 |              7
 public     | t2         |              1 |              1
 public     | t1         |              3 |              0
(3 rows)
```

pg_get_bucket_epoch(relid)

描述：查询v3表在当前节点上的epoch值。对于分区表，需要输入父表oid，将打印所有子分区的epoch值。该函数只支持在写VW上执行。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值字段：

- relfilenode oid：列存非分区表的relfilenode。对于分区表，此值为0。
- partfilenode oid：列存分区表的relfilenode。对于非分区表，此值为0。
- epoch xid：当前节点上该表对应的epoch值。

示例：

 版权所有 © 华为云计算技术有限公司

```
select * from pg_get_bucket_epoch(2147483732);
 relfilenode | partfilenode |  epoch
-------------+--------------+--------
  2137383742 |            0 | 1000001
(1 row)
```

pg_get_bucket_epoch(tablespaceid, databaseid, relfilenode)

描述：查询v3表在当前节点上的epoch值。输入v3表所在的表空间oid，数据库oid和表对应的relfilenode。该函数在写VW或读VW上均可执行。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：xid

返回值字段：

epoch xid：当前节点上该表对应的epoch值。

示例：

```
select * from pg_get_bucket_epoch(2147483729, 16792, 2147483732);
 epoch
-------
1000001
(1 row)
```

pgxc_get_bucket_epoch(tablespaceid, databaseid, relfilenode)

描述：查询v3表在所有dn节点上的epoch值。输入v3表所在的表空间oid，数据库oid和表对应的relfilenode。该函数在写VW或读VW上均可执行。该函数仅9.1.0.200及以上集群版本支持。

返回值类型：record

返回值字段：

- nodename text：节点名称。
- epoch xid：当前节点上该表对应的epoch值。

示例：

```
select * from pgxc_get_bucket_epoch(2147483729, 16792, 2147483732);
  nodename |  epoch
-----------+---------
 datanode3 | 1000001
 datanode1 |       0
 datanode2 |       0
(1 row)
```

current_temp_schema()

描述：返回当前会话的临时schema。如果当前会话没有临时schema，则返回空。该函数仅8.2.1.220及以上集群版本支持。

返回值类型：name

示例：

```
select current_temp_schema();
 current_temp_schema
---------------------

(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
create temp table tmp1(a int,b int) distribute by hash(a);
CREATE TABLE
select current_temp_schema();
          current_temp_schema
----------------------------------------
 pg_temp_coordinator1_2_2_140718823485088
(1 row)
```

get_volatile_pg_class()

描述：获取当前会话中所有volatile临时表相关的pg_class元数据。该函数仅8.2.0及以上集群版本支持。

返回值类型：record

返回值字段：

- oid：返回volatile临时表的oid。
- 其他字段：同pg_class系统表中的基础字段(不包括系统隐藏字段)。

get_volatile_pg_class(relname text)

描述：获取当前会话中指定volatile临时表相关的pg_class元数据。该函数仅8.2.0及以上集群版本支持。

参数：当前会话中volatile临时表的表名。

返回值类型：record

返回值字段：

- oid：返回volatile临时表的oid。
- 其他字段：同pg_class系统表中的基础字段(不包括系统隐藏字段)。

示例：

```
SELECT * FROM get_volatile_pg_class('tx1');
  oid  | relname | relnamespace | reltype | reloftype | relowner | relam | relfilenode | reltablespace | relpages |
reltuples | relallvisible | reltoastrelid | reltoastidxid | reldeltarelid |
 reldeltaidx | relcudescrelid | relcudescidx | relhasindex | relisshared | relpersistence | relkind | relnatts |
relchecks | relhasoids | relhaspkey | relhasrules | relhastriggers | relhassub
class | relcmprs | relhasclusterkey | relrowmovement | parttype | relfrozenxid | relacl |
reloptions             | relreplident | relfrozenxid64
-------+---------+--------------+---------+-----------+----------+-------+-------------+---------------+----------
+-----------+---------------+---------------+---------------+---------------+
-------------+----------------+--------------+-------------+-------------+-----------------+---------+----------
+-----------+------------+------------+-------------+----------------+----------
------+----------+------------------+---------------+----------+--------------+--------
+-----------------------------------+--------------+---------------
 16772 | tx1     |       16770 |   16774 |        0 |      10 |     0 |      16772 |         1665 |        0 |         0
|           0 |        16775 |           0 |           0 |
         0 |           0 |          0 | f          | f          | v            | r       |       2 |        0 | f         | f         |
f          | f            | f
     | 1       | f             | f            | n       | 11815       |       | {orientation=row,compression=no} |
d            |         11815
(1 row)
```

get_volatile_pg_attribute()

描述：获取当前会话中所有volatile临时表相关的pg_attribute元数据。该函数仅8.2.0及以上集群版本支持。

返回值类型：record

 版权所有 © 华为云计算技术有限公司

返回值字段：

- oid：返回该列信息的oid。
- 其他字段：同pg_attribute系统表中的基础字段（不包括系统隐藏字段）。

get_volatile_pg_attribute(relname text, attrname text)

描述：获取当前会话中指定volatile临时表相关的pg_attribute元数据。该函数仅8.2.0及以上集群版本支持。

参数：

- relname：表名(必须在当前会话中)。
- attrname：列名。

返回值类型：record

返回值字段：

- oid：返回该列信息的oid。
- 其他字段：同pg_attribute系统表中的基础字段（不包括系统隐藏字段）。

示例：

```
SELECT * FROM get_volatile_pg_attribute('tx1', 'b');
 attrelid | attname | atttypid | attstattarget | attlen | attnum | attndims | attcacheoff | atttypmod | attbyval |
attstorage | attalign | attnotnull | atthasdef | attisdropped | attislocal |
 attcmprmode | attinhcount | attcollation | attacl | attoptions | attfdwoptions | attinitdefval | attkvtype
----------+---------+----------+---------------+--------+--------+----------+-------------+-----------+----------
+------------+----------+------------+-----------+--------------+------------+
-------------+-------------+--------------+--------+------------+---------------+---------------+-----------
   16772 | b       |       25 |            -1 |     -1 |      2 |        0 |          -1 |        -1 | f        | x          | i        |
f          | f         | f           | t          |
 127         |          0 |         100 |       |           |             | 0
(1 row)
```

pg_get_publication_tables(pubname text)

描述：根据发布的名称，返回对应发布要发布的表的relid列表。该函数仅8.2.0.100及以上集群版本支持。

参数：pubname

返回值类型：set of oid

示例：

```
SELECT * FROM pg_get_publication_tables('mypub');
 relid
-------
 16757
 16776
(2 rows)
```

pg_relation_is_publishable(relname regclass)

描述：检查一个表是否可以发布。该函数仅8.2.0.100及以上集群版本支持。

参数：relname

返回值类型：boolean

 版权所有 © 华为云计算技术有限公司

示例：

```
SELECT * FROM pg_relation_is_publishable('t1');
 pg_relation_is_publishable
----------------------------
 t
(1 row)
```

get_col_cu_info(schema_name text, table_name text, row_count int8, dirty_percent int8)

描述：查询某张列存表的CU信息，每个分区单独统计。该函数仅8.2.0.100及以上集群版本支持。

参数：模式名（必选），表名（必选），小CU的行数阈值（可选，默认200，取值范围为1~60000），脏CU的删除百分比阈值（可选，默认70，取值范围1~100）。

返回值类型：record

返回值字段：

node_name：DN节点名。

part_name：分区名，普通表此列为空。

zero_size_cu_count：cuSize=0，且行数小于等于row_count的CU数。

small_cu_count：cuSize=ALIGNOF_CUSIZE(8192)，且行数小于等于row_count的CU数。

dirty_cu_count: 因delete，deadtupe占比超过dirty_percent的CU数。

total_cu_count：总CU数。

small_cu_size：8kCU的总大小。

total_cu_size：总CU大小。

示例：

```
 SELECT * FROM get_col_cu_info('public','hs_part');
 node_name | part_name | zero_size_cu_count | small_cu_count | dirty_cu_count | total_cu_count |
small_cu_size | total_cu_size
-----------+-----------+--------------------+----------------+----------------+----------------+---------------
+---------------
 dn_1      | p1        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
 dn_1      | p2        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
 dn_1      | p3        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
(3 rows)

 SELECT * FROM get_col_cu_info('public','hs_part', 200, 90);
 node_name | part_name | zero_size_cu_count | small_cu_count | dirty_cu_count | total_cu_count |
small_cu_size | total_cu_size
-----------+-----------+--------------------+----------------+----------------+----------------+---------------
+---------------
 dn_1      | p1        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
 dn_1      | p2        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
 dn_1      | p3        |                  3 |              0 |              0 |              3 | 0 bytes       | 0 bytes
(3 rows)
```


版权所有 © 华为云计算技术有限公司

get_col_file_vacuum_info(schema_name text, table_name text, force_get_rewritten_file_num bool)

描述：查询某张列存表的文件vacuum相关信息，每个分区单独统计。该函数仅8.2.0.100及以上集群版本支持。

参数：模式名（必选），表名（必选），是否强制获取准确的可清理文件数（必选，默认为false）

返回值类型：record

返回值字段：

node_name：DN节点名。

part_name：分区名，普通表此列为空。

total_file_num：总CU文件数。

rewritable_file_num：可重写，但还未重写的文件数。

rewritten_file_num：已重写未清理文件数，此值来源于内存态ClearCache，若由于重启等原因ClearCache丢失，可通过设置force_get_rewritten_file_num=true，消耗额外的时间，强制获取准确的可清理文件数。

empty_file_num：已清理文件数。

示例：

```
SELECT * FROM get_col_file_vacuum_info('public','pa',false);
 node_name | part_name | total_file_num | rewritable_file_num | rewritten_file_num | empty_file_num
-----------+-----------+----------------+---------------------+--------------------+----------------
 datanode1 | pa1       |              1 |                   0 |                  0 |              0
 datanode1 | pa2       |              1 |                   0 |                  0 |              0
 datanode2 | pa1       |              1 |                   0 |                  0 |              0
 datanode2 | pa2       |              1 |                   0 |                  0 |              0
 datanode3 | pa1       |              1 |                   0 |                  0 |              0
 datanode3 | pa2       |              1 |                   0 |                  0 |              0
(6 rows)
```

get_col_file_vacuum_info(schema_name text, table_name text, colvacuum_threshold_scale_factor int)

描述：查询某张列存表的文件vacuum相关信息，每个分区单独统计。该函数仅8.2.0.100及以上集群版本支持。

参数：模式名（必选），表名（必选），colvacuum_threshold_scale_factor（必选，0-100为dead tuple的比例值）

返回值类型：record

返回值：

node_name：DN节点名。

part_name：分区名，普通表此列为空。

total_file_num：总CU文件数。

rewritable_file_num：可重写，但还未重写的文件数。

版权所有 © 华为云计算技术有限公司

rewritten_file_num：已重写未清理文件数，此值来源于内存态ClearCache，若由于重启等原因ClearCache丢失，可通过设置force_get_rewritten_file_num=true，消耗额外的时间，强制获取准确的可清理文件数。

empty_file_num：已清理文件数。

示例：

```
SELECT * FROM get_col_file_vacuum_info('public','pa',10);
 node_name | part_name | total_file_num | rewritable_file_num | rewritten_file_num | empty_file_num
-----------+-----------+----------------+---------------------+--------------------+----------------
 datanode1 | pa1       |              1 |                   0 |                  0 |              0
 datanode1 | pa2       |              1 |                   0 |                  0 |              0
 datanode2 | pa1       |              1 |                   0 |                  0 |              0
 datanode2 | pa2       |              1 |                   0 |                  0 |              0
 datanode3 | pa1       |              1 |                   0 |                  0 |              0
 datanode3 | pa2       |              1 |                   0 |                  0 |              0
(6 rows)
```

get_all_col_cu_info(row_count int8)

描述：查询全库所有列存表的CU信息。该函数仅8.2.0.100及以上集群版本支持。

参数：小CU的行数阈值（可选，默认值200，取值范围为1~60000。）

返回值类型：record

返回值字段：

node_name：DN节点名。

schema_name：模式名。

table_name：表名。

zero_size_cu_count：cuSize=0，且行数小于等于row_count的CU数。

small_cu_count：cuSize=ALIGNOF_CUSIZE(8192)，且行数小于等于row_count的CU数。

total_cu_count：总CU数。

small_cu_size：8kCU的总大小。

total_cu_size：总CU大小。

示例：

```
SELECT * FROM get_all_col_cu_info(200);
 node_name | schema_name |      table_name      | zero_size_cu_count | small_cu_count | total_cu_count | small_cu_size | total_cu_size
-----------+-------------+----------------------+--------------------+----------------+----------------+---------------+---------------
 datanode1 | public      | udi_48076            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
 datanode1 | public      | udi_48077            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
 datanode2 | public      | udi_48076            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
 datanode2 | public      | udi_48077            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
 datanode3 | public      | udi_48076            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
 datanode3 | public      | udi_48077            |                  5 |              1 |              6 | 8192 bytes    | 8192 bytes
(6 rows)
```

get_all_col_file_vacuum_info(force_get_rewritten_file_num bool)

描述：查询全库所有列存表的文件vacuum相关信息。该函数仅8.2.0.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

参数：是否强制获取准确的可清理文件数（必选，true或false）

返回值类型：record

返回值字段：

node_name：DN节点名。

schema_name：模式名。

table_name：表名。

total_file_num：总CU文件数。

rewritable_file_num：可重写，但还未重写的文件数。

rewritten_file_num：已重写未清理文件数，此值来源于内存态ClearCache，若由于重启等原因ClearCache丢失，可通过设置force_get_rewritten_file_num=true，消耗额外的时间，强制获取准确的可清理文件数。

empty_file_num：已清理文件数。

示例：

```
SELECT * FROM get_all_col_file_vacuum_info(false);
 node_name | schema_name |      table_name      | total_file_num | rewritable_file_num | rewritten_file_num
| empty_file_num
-----------+-------------+----------------------+----------------+---------------------+--------------------
+----------------
 datanode1 | public      | udi_57373            |              2 |                   0 |                   0 |              1
 datanode1 | public      | udi_57374            |              2 |                   0 |                   0 |              1
 datanode2 | public      | udi_57373            |              2 |                   0 |                   0 |              1
 datanode2 | public      | udi_57374            |              2 |                   0 |                   0 |              1
 datanode3 | public      | udi_57373            |              2 |                   0 |                   0 |              1
 datanode3 | public      | udi_57374            |              2 |                   0 |                   0 |              1
```

show_tsc_info()

描述：查询当前数据库节点已获取的TSC（TimeStamp-Counter）信息。该函数仅8.2.1及以上集群版本支持。

返回值类型：record

表 21-61 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tsc_freq_error_info	text	获取TSC频率的错误信息。

示例：

```
SELECT * FROM show_tsc_info();
  node_name   | tsc_mult | tsc_shift | tsc_frequency | tsc_use_frequency | tsc_ready |
tsc_scalar_error_info     | tsc_freq_error_info
--------------+----------+-----------+---------------+-------------------+-----------+-----------------------------
+---------------------
 coordinator1 |          |           |          2400 | t                 | t         | TSC scalar is not initialized |
```

get_tsc_info()

描述：重新获取当前数据库节点的TSC（TimeStamp-Counter）信息。该函数仅8.2.1及以上集群版本支持。

返回值类型：record

表 21-62 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

示例：

```
SELECT * FROM get_tsc_info();
  node_name   | tsc_mult | tsc_shift | tsc_frequency | tsc_use_frequency | tsc_ready |
tsc_scalar_error_info     | tsc_freq_error_info
--------------+----------+-----------+---------------+-------------------+-----------+-----------------------------
+---------------------
 coordinator1 |          |           |          2400 | t                 | t         | TSC scalar is not initialized |
```


版权所有 © 华为云计算技术有限公司

test_tsc_info(time float8, loops int)

描述：测试当前节点通过TSC（TimeStamp-Counter）换算时间的准确性。该函数仅8.2.1及以上集群版本支持。

输入参数含义如下：

- time：测试的时间差（单位s），测试时长需要小于或者等于60s
- loops：测试次数，范围值[1，10]

返回值类型：record

返回值字段：

- id：循环次数。
- real_time_diff：通过gettimeofday获取的时间差（单位us）。
- est_time_scalar：通过TSC换算信息转换得到的时间差（单位us）。
- est_time_frequency：通过TSC频率转换得到的时间差（单位us）。

示例：

```
SELECT * FROM test_tsc_info(0.01,10);
 id | real_time_diff | est_time_scalar | est_time_frequency
----+----------------+-----------------+--------------------
  1 |          10057 |                 |            10056.9
  2 |          10057 |                 |   10057.4816666667
  3 |          10056 |                 |   10055.2841666667
  4 |          10054 |                 |   10054.4908333333
  5 |          10055 |                 |         10054.2875
  6 |          10055 |                 |   10054.7483333333
  7 |          10055 |                 |         10054.4725
  8 |          10054 |                 |   10054.0766666667
  9 |          10058 |                 |   10058.1016666667
 10 |          10057 |                 |   10056.3733333333
(10 rows)
```

pgxc_recyclebin_purge

描述：遍历并清理回收站中的对象，该函数仅9.1.0.200及以上集群版本支持。

返回值类型：void

示例：

```
SELECT * FROM pgxc_recyclebin_purge();
 pgxc_recyclebin_purge
-----------------------

(1 row)
```

21.7.28 备份恢复控制函数

21.7.28.1 备份控制函数

备份控制函数可帮助进行在线备份。

pg_create_restore_point(name text)

描述：为执行恢复创建一个命名点。（需要管理员角色）

版权所有 © 华为云计算技术有限公司

返回值类型：text

备注：pg_create_restore_point创建了一个可以用作恢复目的、有命名的事务日志记录，并返回相应的事务日志位置。在恢复过程中，recovery_target_name可以通过这个名称定位对应的日志恢复点，并从此处开始执行恢复操作。避免使用相同的名称创建多个恢复点，因为恢复操作将在第一个匹配（恢复目标）的名称上停止。

pg_current_xlog_location()

描述：获取当前事务日志的写入位置。

返回值类型：text

备注：pg_current_xlog_location使用与前面那些函数相同的格式显示当前事务日志的写入位置。如果是只读操作，不需要系统管理员权限。

示例：

```
 SELECT pg_current_xlog_location();
 pg_current_xlog_location
--------------------------
 0/4310D5D8
(1 row)
```

pg_current_xlog_insert_location()

描述：获取当前事务日志的插入位置。

返回值类型：text

备注：pg_current_xlog_insert_location显示当前事务日志的插入位置。插入点是事务日志在某个瞬间的“逻辑终点”，而实际的写入位置则是从服务器内部缓冲区写出时的终点。写入位置是可以从服务器外部检测到的终点，如果要归档部分完成事务日志文件，则该操作即可实现。插入点主要用于服务器调试目的。如果是只读操作，不需要系统管理员权限。

示例：

```
 SELECT pg_current_xlog_insert_location();
 pg_current_xlog_insert_location
---------------------------------
 0/4316AD50
(1 row)
```

pg_start_backup(label text [, fast boolean])

描述：开始执行在线备份。（需要管理员角色或复制的角色）

返回值类型：text

备注：pg_start_backup接受一个用户定义的备份标签（通常这是备份转储文件存放地点的名字）。这个函数向数据库集群的数据目录写入一个备份标签文件，然后以文本方式返回备份的事务日志起始位置。

```
SELECT pg_start_backup('label_goes_here');
 pg_start_backup
-----------------
 0/3000020
(1 row)
```

 版权所有 © 华为云计算技术有限公司

pg_stop_backup()

描述：完成执行在线备份。（需要管理员角色或复制的角色）

返回值类型：text

备注：pg_stop_backup删除pg_start_backup创建的标签文件，并且在事务日志归档区里创建一个备份历史文件。这个历史文件包含给予pg_start_backup的标签、备份的事务日志起始与终止位置、备份的起始和终止时间。返回值是备份的事务日志终止位置。计算出中止位置后，当前事务日志的插入点将自动前进到下一个事务日志文件，这样，结束的事务日志文件可以被立即归档从而完成备份。

示例：

```
 SELECT pg_stop_backup();
 pg_stop_backup
----------------
 0/4401E610
(1 row)
```

pg_switch_xlog()

描述：切换到一个新的事务日志文件。（需要管理员角色）

返回值类型：text

备注：pg_switch_xlog移动到下一个事务日志文件，以允许将当前日志文件归档（假定使用连续归档）。返回值是刚完成的事务日志文件的事务日志结束位置+1。如果从最后一次事务日志切换以来没有活动的事务日志，则pg_switch_xlog什么事也不做，直接返回当前事务日志文件的开始位置。

示例:

```
SELECT pg_switch_xlog();
 pg_switch_xlog
----------------
 0/45017978
(1 row)
```

pg_xlogfile_name(location text)

描述：将事务日志的位置字符串转换为文件名。

返回值类型：text

备注：pg_xlogfile_name仅抽取事务日志文件名称。如果给定的事务日志位置恰好位于事务日志文件的交界上，这两个函数都返回前一个事务日志文件的名字。这对于管理事务日志归档来说是非常有利的，因为前一个文件是当前最后一个需要归档的文件。

pg_xlogfile_name_offset(location text)

描述：将事务日志的位置字符串转换为文件名并返回在文件中的字节偏移量。

返回值类型：text,integer

备注：可以使用pg_xlogfile_name_offset从前述函数的返回结果中抽取相应的事务日志文件名称和字节偏移量。例如：

```
SELECT * FROM pg_xlogfile_name_offset(pg_stop_backup());
NOTICE:  pg_stop_backup cleanup done, waiting for required WAL segments to be archived
```

 版权所有 © 华为云计算技术有限公司

```
NOTICE:  pg_stop_backup complete, all required WAL segments have been archived
        file_name         | file_offset
--------------------------+-------------
000000010000000000000003 |         272
(1 row)
```

pg_xlog_location_diff(location text, location text)

描述：计算两个事务日志位置之间在字节上的区别。

返回值类型：numeric

pg_cbm_tracked_location()

描述：用于查询cbm解析到的lsn位置。

返回值类型：text

pg_cbm_get_merged_file(startLSNArg text, endLSNArg text)

描述：用于将指定lsn范围之内的cbm文件合并成一个cbm文件，并返回合并完的cbm文件名。

返回值类型：text

pg_cbm_get_changed_block(startLSNArg text, endLSNArg text)

描述：用于将指定lsn范围之内的cbm文件合并成一个表，并返回表的各行记录。

返回值类型：record

备注：pg_cbm_get_changed_block返回的表字段包含：合并起始的lsn，合并截止的lsn， 表空间oid，库oid，表的relfilenode，表的fork number，表是否被删除，表是否被创建，表是否被截断，表被截断后的页面数，有多少页被修改以及被修改的页号的列表。

pg_cbm_recycle_file(slotName name, targetLSNArg text)

描述：删除不再使用的cbm文件，并返回删除后的第一条lsn。其中，slotName为空字符串时，将直接使用targetLSNArg作为回收点；在备份和容灾时，因为并行因素，需要指定相应的slotName，首先记录该任务的targetLSNArg到该slot中，然后遍历所有的backup slots，找到最小的lsn作为回收点。

返回值类型：text

pg_cbm_force_track(targetLSNArg text,timeOut int)

描述：强制执行一次cbm追踪到指定的xlog位置，并返回实际追踪结束点的xlog位置。

返回值类型：text

pg_enable_delay_ddl_recycle()

描述：开启延迟DDL功能，并返回开启点的xlog位置。

返回值类型：text

示例：

 版权所有 © 华为云计算技术有限公司

```
SELECT pg_enable_delay_ddl_recycle();
 pg_enable_delay_ddl_recycle
-----------------------------
 00000000/4607B250
(1 row)
```

pg_disable_delay_ddl_recycle(barrierLSNArg text, isForce bool)

描述：关闭延迟DDL功能，并返回本次延迟DDL生效的xlog范围。

返回值类型：record

pg_enable_delay_xlog_recycle()

描述：开启延迟xlog回收功能。

返回值类型：void

示例：

```
SELECT pg_enable_delay_xlog_recycle();
 pg_enable_delay_xlog_recycle
------------------------------

(1 row)
```

pg_disable_delay_xlog_recycle()

描述：关闭延迟xlog回收功能。

返回值类型：void

示例：

```
SELECT pg_disable_delay_xlog_recycle();
 pg_disable_delay_xlog_recycle
-------------------------------

(1 row)
```

pgxc_get_senders_catchup_time()

描述：显示所有DN上当前活跃的主备发送线程的追赶信息。

返回值类型：record

函数返回信息如下：

表 21-63 pgxc_get_senders_catchup_time()字段

名称	类型	描述
node_name	text	节点名称。
lwpid	integer	当前sender的lwpid。
local_role	text	本地的角色。
peer_role	text	对端的角色。
state	text	当前sender的复制状态。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
sender	text	当前sender的类型。
catchup_start	timestamp with time zone	catchup启动的时间。
catchup_end	timestamp with time zone	catchup结束的时间。
catchup_type	text	catchup方式为全量还是增量。
catchup_bcm_filename	text	catchup当前执行到的bcm文件
catchup_bcm_finished	integer	catchup已经操作完成的bcm文件数量。
catchup_bcm_total	integer	catchup总共需要操作的bcm文件数量。
catchup_percent	text	catchup已经操作完成的百分比。
catchup_remaining_time	text	catchup预估剩余时间。

pgxc_lock_for_backup()

描述：备份前给集群加锁，备份是为了在新增节点上执行恢复操作。

返回值类型：boolean

说明

pgxc_lock_for_backup是在使用gs_dump或gs_dumpall工具备份集群前，用来给集群加锁的。当给集群加锁后，不允许有改变系统结构的操作。该函数不影响DML语句。

示例：

```
SELECT pgxc_lock_for_backup();
 pgxc_lock_for_backup
----------------------
 t
(1 row)
```

21.7.28.2 恢复控制函数

恢复控制函数提供了当前备机状态的信息。此类函数可在恢复期间或数据库正常运行时执行。

pg_is_in_recovery()

描述：如果恢复仍然在进行中则返回true。

返回值类型：bool

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 f
(1 row)
```

pg_last_xlog_receive_location()

描述：获取最后接收事务日志的位置并通过流媒体复制同步到磁盘。当流复制仍在进行，事务日志将持续递增。如果恢复已完成，则这个值将保持最后接收事务记录的状态并在恢复期间同步到磁盘。如果不能用流复制，或还没有开始，该函数返回NULL。

返回值类型：text

示例：

```
SELECT pg_last_xlog_receive_location();
 pg_last_xlog_receive_location
-------------------------------

(1 row)
```

pg_last_xlog_replay_location()

描述：获取最后一个事务日志在恢复时重放的位置。当恢复仍在进行，事务日志将持续递增。如果已经完成恢复，则将保持在恢复期间最后接收WAL记录的值。如果未进行恢复但服务器正常启动时，则这个函数返回NULL。

返回值类型：text

示例：

```
SELECT pg_last_xlog_replay_location();
 pg_last_xlog_replay_location
------------------------------
 0/2B16530
(1 row)
```

pg_last_xact_replay_timestamp()

描述：获取最后一个事务在恢复时重放的时间戳。这是为在主节点上生成事务提交或终止WAL记录的时间。如果在恢复时没有事务重放，则这个函数返回NULL。否则，如果恢复仍在进行，则事务日志将持续递增。如果恢复已经完成，则将保持在恢复期间最后接收WAL记录的值。如果服务器无需恢复就已正常启动，则这个函数返回NULL。

返回值类型：timestamp with time zone

恢复控制函数控制恢复的进程。这些函数可能只在恢复时被执行。

示例：

```
SELECT pg_last_xact_replay_timestamp();
 pg_last_xact_replay_timestamp
-------------------------------
 2023-01-04 07:03:08.098024+00
(1 row)
```

pg_is_xlog_replay_paused()

描述：如果恢复暂停则返回true。

返回值类型：bool

 版权所有 © 华为云计算技术有限公司

pg_xlog_replay_pause()

描述：立即暂停恢复。

返回值类型：void

pg_xlog_replay_resume()

描述：如果恢复处于暂停状态，则重新启动。

当恢复暂停时，没有发生数据库更改。如果是在热备里，所有新的查询将看到一致的数据库快照，并且不会有进一步的查询冲突产生，直到恢复继续。

如果不能使用流复制，则暂停状态将无限的延续。当流复制正在进行时，将连续接收WAL记录，最终将填满可用磁盘空间，这个进度取决于暂停的持续时间，WAL生成的速度和可用的磁盘空间。

返回值类型：void

pg_xlog_replay_completion()

描述：显示当前DN上xlog redo的进度信息。

返回值类型：record

示例：

```
SELECT * FROM pg_xlog_replay_completion();
 replay_start | replay_current | replay_end | replay_percent
--------------+----------------+------------+----------------
 0/2ACAB80    | 0/2B16530      | 0/4F62B090 | 0%
(1 row)
```

函数返回信息如下：

表 21-64 pg_xlog_replay_completion()字段

名称	类型	描述
replay_start	integer	Xlog Redo的起始LSN
replay_current	integer	Xlog Redo的当前replay的LSN
replay_end	integer	最大的需要Xlog Redo的LSN
replay_percent	integer	Xlog Redo的当前完成的百分比

pg_data_sync_from_dummy_completion()

描述：显示当前DN上Failover过程中数据页文件同步的进度信息。

返回值类型：record

示例：

```
SELECT * FROM pg_data_sync_from_dummy_completion();
 start_index | current_index | total_index | sync_percent
-------------+---------------+-------------+--------------
```

 版权所有 © 华为云计算技术有限公司

```
        0 |           0 |           0 | 100%
(1 row)
```

函数返回信息如下：

表 21-65 pg_data_sync_from_dummy_completion()字段

名称	类型	描述
start_index	integer	数据页文件同步的起始编号
current_index	integer	数据页文件同步的当前编号
total_index	integer	数据页文件同步的最大编号
sync_percent	integer	数据页文件当前完成的百分比

gs_roach_stop_backup(backupid text)

描述：停止一个内部备份工具GaussRoach开启的备份，返回当前日志的插入位置。与pg_stop_backup系统函数类似，但更轻量。

返回值类型：text

gs_roach_enable_delay_ddl_recycle(backupid name)

描述：开启延迟DDL功能，并返回开启点的日志位置。与pg_enable_delay_ddl_recycle系统函数类似，但更轻量。且通过传入不同的backupid，可支持并发打开延迟DDL。

返回值类型：text

gs_roach_disable_delay_ddl_recycle(backupid text)

描述：关闭延迟DDL功能，并返回本次延迟DDL生效的日志范围，并删除该范围内被用户删除的列存表物理文件。与pg_enable_delay_ddl_recycle系统函数类似，但更轻量。且通过传入不同的backupid，可支持并发关闭延迟DDL功能。

返回值类型：record

gs_roach_switch_xlog(request_ckpt bool)

描述：切换当前使用的日志段文件，返回切段日志的位置。且如果request_ckpt为true，则触发一个全量检查点。

返回值类型：text

pg_resume_bkp_flag(backupid name)

描述：用于备份恢复获取delay xlog标志，并返回start_backup_flag boolean，to_delay boolean，ddl_delay_recycle_ptr text，rewind_time text。

版权所有 © 华为云计算技术有限公司

返回值类型：record

pg_update_reloptions_in_newcluster

描述：用于V3表集群级全量恢复到新集群时，将obs-store-path-list参数指定文件中存储的路径写入pg_class表的reloptions字段中。该函数仅9.1.0及以上集群版本支持。

返回值类型：bool

21.7.29 数据库对象函数

21.7.29.1 数据库对象尺寸函数

数据库对象尺寸函数用于计算数据库对象使用的实际磁盘空间，其计算结果都返回以字节为单位的大小。

pg_column_size(any)

描述：存储一个指定的数值需要的字节数（可能压缩过）。

返回值类型：integer

备注：pg_column_size显示用于存储某个独立数据值的空间。

```
SELECT pg_column_size(1);
 pg_column_size
----------------
              4
(1 row)
```

pg_database_size(oid)

描述：指定OID代表的数据库使用的磁盘空间。

返回值类型：bigint

pg_database_size(name)

描述：指定名称的数据库使用的磁盘空间。

返回值类型：bigint

备注：pg_database_size接收一个数据库的OID或者名字，然后返回该对象使用的全部磁盘空间。

示例：

```
SELECT pg_database_size('gaussdb');
 pg_database_size
------------------
         51590112
(1 row)
```

pg_relation_size(oid)

描述：指定OID代表的表或者索引所使用的磁盘空间。

返回值类型：bigint

版权所有 © 华为云计算技术有限公司

get_db_source_datasize()

描述：估算当前数据库非压缩态的数据总容量。

返回值类型：bigint

备注：（1）调用该函数前需要做analyze；（2）通过估算列存的压缩率计算非压缩态的数据总容量。

示例：

```
analyze;
ANALYZE
SELECT get_db_source_datasize();
 get_db_source_datasize
------------------------
            35384925667
(1 row)
```

pg_relation_size(text)

描述：指定名称的表或者索引使用的磁盘空间。表名字可以用模式名修饰。

返回值类型：bigint

pg_relation_size(relation regclass, fork text)

描述：指定表或索引的指定分叉树（'main'，'fsm'或'vm'）使用的磁盘空间。

返回值类型：bigint

pg_relation_size(relation regclass)

描述：pg_relation_size(..., 'main')的简写。

返回值类型：bigint

备注：pg_relation_size返回指定OID或名称的表、索引、压缩表的字节大小。

pg_partition_size(oid,oid)

描述：指定OID代表的分区使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。

返回值类型：bigint

pg_partition_size(text, text)

描述：指定名称的分区使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。

返回值类型：bigint

pg_partition_indexes_size(oid,oid)

描述：指定OID代表的分区的索引使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。

返回值类型：bigint

版权所有 © 华为云计算技术有限公司

pg_partition_indexes_size(text,text)

描述：指定名称的分区的索引使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。

返回值类型：bigint

pg_indexes_size(regclass)

描述：附加到指定表的索引使用的总磁盘空间。

返回值类型：bigint

pg_size_pretty(bigint)

描述：把字节计算的尺寸转换成一个易读的尺寸。

返回值类型：text

pg_size_pretty(numeric)

描述：把用数值表示的字节计算的尺寸转换成一个易读的尺寸。

返回值类型：text

备注：pg_size_pretty用于把其他函数的结果格式化成一种易读的格式，可以根据情况使用KB 、MB 、GB 、TB。

pg_table_size(regclass)

描述：指定的表使用的磁盘空间，不计索引（但是包含TOAST，自由空间映射和可见性映射）。

返回值类型：bigint

pg_total_relation_size(oid)

描述：指定OID代表的表使用的磁盘空间，包括索引和压缩数据。

返回值类型：bigint

pg_total_relation_size(regclass)

描述：指定的表使用的总磁盘空间，包括所有的索引和TOAST数据。

返回值类型：bigint

pg_total_relation_size(text)

描述：指定名称的表所使用的全部磁盘空间，包括索引和压缩数据。表名字可以用模式名修饰。

返回值类型：bigint

备注：pg_total_relation_size可指定表或压缩表的OID或者名称，然后返回以字节计的数据和所有相关的索引和压缩表的尺寸。

 版权所有 © 华为云计算技术有限公司

pg_obs_file_size(regclass)

描述：指定列存V3版本表OID或者表名字，获取其在OBS上存储的CU文件大小、文件名、bucket号。该函数仅9.1.0及以上集群版本支持。

参数：入参可以是表的OID或者表名。

返回值类型：record

表 21-66 返回字段

名称	类型	描述
bucketid	integer	CU文件存储的bucket号。
filename	text	CU文件名。
size	bigint	CU文件大小，单位为字节。

示例：

```
SELECT * FROM pg_obs_file_size('t3_col_part_dif_partition');
 bucketid |      filename      | size
----------+--------------------+------
        9 | 23488119324673.0 |  512
       22 | 23488102760449.0 |  512
       21 | 23488119521281.0 |  512
       16 | 23488102662145.0 |  512
(4 rows)
```

pg_obs_file_size(text, text)

描述：指定列存V3版本表(主表名,分区表)的某个分区，返回其在OBS上存储的CU文件大小、文件名、bucket号。该函数仅9.1.0及以上集群版本支持。

参数：入参可以是表的OID或者表名，分区名则直接使用分区名。

返回值类型：record

名称	类型	描述
bucketid	integer	CU文件存储的bucket号
filename	text	CU文件名
size	bigint	CU文件大小，单位为字节

21.7.29.2 数据库对象位置函数

pg_relation_filenode(relation regclass)

描述：指定关系的文件节点数。

返回值类型：oid

版权所有 © 华为云计算技术有限公司

备注：pg_relation_filenode接收一个表、索引、序列或压缩表的OID或者名字，并且返回当前分配给它的“filenode”数。文件节点是关系使用的文件名字的基本组件。对大多数表来说，结果和pg_class.relfilenode相同，但对确定的系统目录来说，relfilenode为0而且这个函数必须用来获取正确的值。如果传递一个没有存储的关系，比如一个视图，那么这个函数返回NULL。

pg_relation_filepath(relation regclass)

描述：指定关系的文件路径名。

返回值类型：text

备注：pg_relation_filepath类似于pg_relation_filenode，但是它返回关系的整个文件路径名（相对于数据库集群的数据目录PGDATA）。

21.7.29.3 分区管理函数

proc_add_partition (relname regclass, boundaries_interval interval)

描述：用于给开启自动创建分区功能的表添加分区。

返回值类型：void

备注：该函数运行时，会在现有分区boundary的基础上，创建多个时间范围为boundaries_interval的新分区，直到new_part_boundary – now_time >= 29 * boundaries_interval，之后再额外多创建一个分区，保证该函数运行时，一定会创建一个新分区 。

示例：

```
call proc_add_partition('my_schema.my_table', interval '1 day');
 proc_add_partition
--------------------

(1 row)
```

proc_drop_partition (relname regclass, older_than interval)

描述：用于给开启自动删除分区功能的表删除分区。

返回值类型：void

备注：该函数运行时，遍历表所有分区，并删除其中boundary小于(now_time - older_than)的分区；如果所有分区都满足删除条件，则保留一个分区，并truncate该表。

示例：

```
call proc_drop_partition('my_schema.my_table', interval '7 day');
 proc_drop_partition
--------------------

(1 row)
```

pg_partition_management_time(table_name, time)

描述：用来修改自增自减分区调度任务的执行时间，默认为创建自增分区表的时间。该函数仅8.3.0及以上版本支持。

版权所有 © 华为云计算技术有限公司

返回值：SUCCESS，任务时间修改成功。

参数说明：

- table_name为自增分区表的表名，类型为name，
- time为数据切换任务调度时间，类型为Time。

示例：

```
call pg_partition_management_time('partition_management_time','3:00');
 pg_partition_management_time
------------------------------
 SUCCESS
(1 row)
```

21.7.29.4 排序规则版本函数

pg_collation_actual_version (oid)

描述：返回当前安装在操作系统中的该排序规则对象的实际版本，目前仅对case_insensitive有效。

返回值类型：text

示例：

```
SELECT oid FROM pg_collation WHERE collname ='case_insensitive';
 oid
------
 3300
(1 row)

SELECT pg_collation_actual_version(3300);
 pg_collation_actual_version
------------------------------
 153.14
(1 row)
```

21.7.29.5 冷热表用户函数

pg_obs_cold_refresh_time(table_name, time)

描述：用来修改冷热表的冷数据切换至OBS上的时间，默认为每日0点。

table_name为冷热表的表名，类型为name，time为数据切换任务调度时间，类型为Time。

返回值：SUCCESS，任务时间修改成功。

示例：

```
SELECT * FROM pg_obs_cold_refresh_time('lifecycle_table', '06:30:00');
pg_obs_cold_refresh_time
--------------------------
 SUCCESS
(1 row)
```

pg_refresh_storage()

描述：切换所有冷热表，将符合冷热切换规则的数据由热数据切换至冷数据（OBS中）。

版权所有 © 华为云计算技术有限公司

返回值类型：int

返回值字段：

- success_count int：切换成功的表个数
- failed_count int：切换失败的表个数

示例：

```
SELECT * FROM pg_refresh_storage();
 success_count | failed_count
---------------+--------------
             1 |            0
(1 row)
```

pg_lifecycle_table_data_distribute(table_name)

描述：查看某个冷热表的数据分布情况。

table_name为表名，不可缺省。

返回值：record

示例：根据节点数量形成多条记录，如下示例为只有一个dn节点时w1表数据分布情况。

```
SELECT * FROM pg_catalog.pg_lifecycle_table_data_distribute('w1');
 schemaname | tablename | nodename | hotpartition | coldpartition | switchablepartition | hotdatasize |
colddatasize | switchabledatasize
------------+-----------+----------+--------------+---------------+---------------------+-------------+--------------
+--------------------
 public     | w1        | dn_1     | p2           | p1            |                     | 80 KB       | 0 bytes      | 0 bytes
(1 row)
```

pg_lifecycle_node_data_distribute()

描述：查看所有冷热表数据分布情况。

返回值：record

示例：数据库中当前存在两个冷热表，其数据分布情况如下。

```
SELECT * FROM pg_catalog.pg_lifecycle_node_data_distribute();
 schemaname | tablename | nodename | hotpartition | coldpartition | switchablepartition | hotdatasize |
colddatasize | switchabledatasize
------------+-----------+----------+--------------+---------------+---------------------+-------------+--------------
+--------------------
 public     | w1        | dn_1     | p2           | p1            |                     |       81920 |            0 |                  0
 public     | w2        | dn_1     | p2           | p1            |                     |       81920 |            0 |                  0
(2 rows)
```

refresh_hot_storage(relname text)

描述：用于将指定冷热表的所有分区数据全部刷到OBS上。返回的数值为执行该函数切换分区的个数。该函数仅8.2.1.100及以上版本支持。

参数：

relname：表名(指定冷热表的表名，使用其他非冷热表的表名，并不会报错，返回值为0)

返回值类型：integer

版权所有 © 华为云计算技术有限公司

```
SELECT refresh_hot_storage('multi_temper_table');
 refresh_hot_storage
---------------------
           4
(1 row)
```

refresh_hot_storage(relname text，partname text)

描述：用于将指定冷热表的分区数据刷到OBS上。返回的数值为执行该函数切换分区的个数。该函数仅8.2.1.100及以上版本支持。

参数：

- relname：表名（指定冷热表的表名，使用其他非冷热表的表名，并不会报错，返回值为0）。
- partname：分区名（指定冷热表下的分区名）。

返回值类型：integer

```
SELECT refresh_hot_storage('multi_temper_table','p1');
 refresh_hot_storage
---------------------
           1
(1 row)
```

reload_cold_partition(relname text)

描述：用于将指定冷热表的所有冷分区转换成热分区。返回的数值为执行该函数切换分区的个数。该函数仅8.3.0及以上版本支持。

参数：

relname：表名（指定冷热表的表名，使用其他非冷热表的表名，并不会报错，返回值为0）。

返回值类型：integer

```
SELECT reload_cold_partition('multi_temper_table');
 reload_cold_partition
---------------------
           4
(1 row)
```

reload_cold_partition(relname text，partname text)

描述：用于将指定冷热表的特定冷分区转换成热分区。返回的数值为执行该函数切换分区的个数。该函数仅8.3.0及以上版本支持。

参数：

- relname：表名（指定冷热表的表名，使用其他非冷热表的表名，并不会报错，返回值为0）。
- partname：分区名（指定冷热表下的分区名）。

返回值类型：integer

```
SELECT reload_cold_partition('multi_temper_table','p1');
 reload_cold_partition
---------------------
           1
(1 row)
```


版权所有 © 华为云计算技术有限公司

21.7.30 残留文件管理函数

21.7.30.1 获取残留文件列表函数

pg_get_residualfiles()

描述：用于获取当前节点的所有残留文件记录。该函数为实例级函数，与当前所在的数据库无关，可以在任意实例上运行。

返回值类型：record

返回信息如下：

表 21-67 pg_get_residualfiles()返回字段

名称	类型	描述
isverified	bool	是否已经验证。
isdeleted	bool	是否已经被删除。
dbname	text	所属数据库名称。
residualfile	text	数据文件路径。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

```
SELECT * FROM pg_get_residualfiles();
 isverified | isdeleted | dbname |   residualfile    |         filepath          | notes
------------+-----------+--------+-------------------+---------------------------+-------
 f          | f         | db2    | base/49155/114691 | pgrf_20200908160211441546 |
 f          | f         | db2    | base/49155/114694 | pgrf_20200908160211441546 |
 f          | f         | db2    | base/49155/114696 | pgrf_20200908160211441546 |
(3 rows)
```

pgxc_get_residualfiles()

描述：pg_get_residualfiles()的CN统一查询函数。该函数为集群级函数，与当前所在的数据库无关，在CN实例上运行。

返回值类型：record

返回信息如下：

表 21-68 pgxc_get_residualfiles()返回字段

名称	类型	描述
nodename	text	节点名称。
isverified	bool	是否已经验证。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
isdeleted	bool	是否已经被删除。
dbname	text	所属数据库名称。
residualfile	text	数据文件路径。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

```
SELECT * FROM pgxc_get_residualfiles();
   nodename   | isverified | isdeleted |  dbname  |   residualfile    |         filepath          | notes
--------------+------------+-----------+----------+-------------------+---------------------------+-------
 cn_5001      | f          | f         | postgres | base/15092/32803  | pgrf_20200910170129360401 |
 dn_6001_6002 | f          | f         | db2      | base/49155/114691 | pgrf_20200908160211441546 |
 dn_6001_6002 | f          | f         | db2      | base/49155/114694 | pgrf_20200908160211441546 |
 dn_6001_6002 | f          | f         | db2      | base/49155/114696 | pgrf_20200908160211441546 |
(4 rows)
```

21.7.30.2 扫描残留文件函数

扫描残留文件函数仅8.3.0及以上集群版本支持，本章节所有函数仅限sysadmin角色调用。

pg_scan_residualfiles()

描述：用于扫描当前节点所连接数据库的所有残留文件记录。该函数为库级函数，与当前所在的数据库相关，可以在任意实例上运行。

- 连接到CN执行时，扫描当前CN节点所在数据库的本地残留文件和OBS全部残留文件。
- 连接到DN执行时，扫描当前DN节点所在数据库的本地残留文件。

返回类型：record

返回信息如下：

表 21-69 返回字段

名称	类型	描述
pgscrf	text	记录残留文件信息的元文件本地路径。

示例：

```
SELECT * FROM pg_scan_residualfiles();
                                pgscrf
-----------------------------------------------------------------------
 pg_residualfiles/pgscrf_meta_1663_16323_20231027143716005244
 pg_residualfiles/pgscrf_meta_2147484281_16323_20231027143716005713
(2 rows)
```


版权所有 © 华为云计算技术有限公司

说明

该函数的使用约束如下：

- 只能扫描当前登录的数据库中是否存在残留文件，当前只扫描默认表空间，自定义表空间和OBS表空间。
- 该函数为重载操作，不支持单节点并发使用，也不建议在业务繁忙和资源负载较高时使用。
- 只支持冷热表热分区，不支持冷分区。
- 禁止在升级观察期内使用。
- 无法对已删除数据库的残留文件进行扫描。
- 如果集群元数据信息存在不一致情况，该函数可能会错误扫描到有效的数据文件。

pgxc_scan_residualfiles(query_flag)

描述：pg_scan_residualfiles()的CN统一执行函数。该函数为集群级函数，与当前所在的数据库相关，在CN实例上运行。

参数说明：query_flag。参数类型int，表示执行范围。1表示CN，2表示主DN，4表示备DN，通过或运算可以得到查询并集，如1|2=3表示CN+主DN，1|2|4=7表示CN+主DN+备DN，默认值是7。

返回类型：record

返回信息如下：

表 21-70 返回字段

名称	类型	描述
nodename	name	节点名称。
instance_id	text	实例名称。
pgscrf	text	记录残留文件信息的元文件本地路径。

示例：

```
SELECT * FROM pgxc_scan_residualfiles();
  node_name   | instance_id |                                pgscrf
--------------+-------------+-----------------------------------------------------------------------
 dn_6001_6002 | dn_6001     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144103839354
 dn_6001_6002 | dn_6001     | pg_residualfiles/pgscrf_meta_2147484281_16323_20231027144103839826
 cn_5001      | cn_5001     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144103946217
 dn_6007_6008 | dn_6008     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144104171311
(4 rows)

SELECT * FROM pgxc_scan_residualfiles(1);
 node_name | instance_id |                            pgscrf
-----------+-------------+-----------------------------------------------------------------
 cn_5001   | cn_5001     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144421861628
(1 row)

SELECT * FROM pgxc_scan_residualfiles(2);
  node_name   | instance_id |                                pgscrf
--------------+-------------+-----------------------------------------------------------------------
 dn_6001_6002 | dn_6001     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144424210395
 dn_6001_6002 | dn_6001     | pg_residualfiles/pgscrf_meta_2147484281_16323_20231027144424210855
(2 rows)
```


版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pgxc_scan_residualfiles(4);
  node_name   | instance_id |                          pgscrf
--------------+-------------+-------------------------------------------------------------------
 dn_6007_6008 | dn_6008     | pg_residualfiles/pgscrf_meta_1663_16323_20231027144427492060
(1 row)
```

说明

该函数的使用约束如下：

- 只能扫描当前登录的数据库中是否存在残留文件，当前只扫描默认表空间，自定义表空间和OBS表空间。
- 该函数为重载操作，不支持单节点并发使用，也不建议在业务繁忙和资源负载较高时使用。
- 只支持冷热表热分区，不支持冷分区。
- 禁止在升级观察期内使用。
- 无法对已删除数据库的残留文件进行扫描。
- 如果集群元数据信息存在不一致情况，该函数可能会错误扫描到有效的数据文件。

pg_get_scan_residualfiles()

描述：用于获取当前节点所有扫描到的残留文件记录。该函数为实例级函数，与当前所在的数据库无关，可以在任意实例上运行。

返回类型：record

返回信息如下：

表 21-71 返回字段

名称	类型	描述
handled	bool	残留文件是否已经被处理，被移动或者被更改。
dbname	text	数据库名称。
residualfile	text	残留文件路径。
size	bigint	文件大小，单位为byte。OBS路径的残留文件该项为0。
inode	bigint	残留文件stat信息中的Inode。OBS路径的残留文件该项为0。
atime	timestamptz	残留文件stat信息中的Access time。OBS路径的残留文件该项为空。
mtime	timestamptz	残留文件stat信息中的Modify time。OBS路径的残留文件该项为空。
ctime	timestamptz	残留文件stat信息中的Chang time。OBS路径的残留文件该项为空。
filepath	text	记录残留文件信息的元文件本地路径，对应pgscrf_meta文件。
notes	text	注释。

版权所有 © 华为云计算技术有限公司

pgxc_get_scan_residualfiles(query_flag)

描述：pg_get_scan_residualfiles()的CN统一执行函数。该函数为集群级函数，与当前所在的数据库无关，在CN实例上运行。

参数query_flag，参数类型int，用于表示执行范围。1表示CN，2表示主DN，4表示备DN，通过或运算可以得到查询并集，如1|2=3表示CN+主DN，1|2|4=7表示CN+主DN+备DN。默认值是7。

返回类型：record

返回信息如下：

表 21-72 返回字段

名称	类型	描述
nodename	name	节点名称。
instance_id	text	实例名称。
handled	bool	残留文件是否已经被处理，被移动或者被更改。
dbname	text	数据库名称。
residualfile	text	残留文件路径。
size	bigint	文件大小，单位为byte。OBS路径的残留文件该项为0。
inode	bigint	残留文件stat信息中的Inode。OBS路径的残留文件该项为0。
atime	timestamptz	残留文件stat信息中的Access time。OBS路径的残留文件该项为空。
mtime	timestamptz	残留文件stat信息中的Modify time。OBS路径的残留文件该项为空。
ctime	timestamptz	残留文件stat信息中的Chang time。OBS路径的残留文件该项为空。
filepath	text	记录残留文件信息的元文件本地路径，对应pgscrf_meta文件。
notes	text	注释。

pg_archive_scan_residualfiles()

描述：用于归档当前节点所连接数据库上所有扫描到的残留文件记录。该函数为实例级函数，与当前所在的数据库相关，可以在任意实例上运行。

返回类型：record

返回信息如下：

 版权所有 © 华为云计算技术有限公司

表 21-73 返回字段

名称	类型	描述
archive	text	归档后的本地文件夹路径。OBS路径的残留文件归档在对应OBS数据库目录下。
count	bigint	归档的残留文件数量。
size	bigint	归档的残留文件总大小，单位为byte。

示例：

```
SELECT * FROM pg_archive_scan_residualfiles();
                              archive                               | count | size
--------------------------------------------------------------------+-------+-------
 pg_residualfiles/archive/pgscrf_archive_20231027144613791801 |     4 | 81920
(1 row)
```

说明

- 此函数只能归档当前登录的数据库中记录过的残留文件，归档时会记录中的残留文件进行相应的校验，校验结果分为以下三种情况：
  - 校验通过：校验通过后会对残留文件进行归档，并标记为已处理。
  - 校验失败：校验失败会跳过归档，并标记为已处理。
  - 延迟校验：延迟校验会跳过归档，通常表示当前未满足校验时机，延迟校验的情况通常和事务完成情况及备机REDO进度有关。
- 实际归档目录与对应表空间在同一个文件系统下，删除表空间也会删除对应的残留文件归档。
- 延迟DDL开启后，此函数无法使用。
- 此函数为重载操作，不支持单节点并发使用，也不建议在业务繁忙和资源负载较高时使用。

pgxc_archive_scan_residualfiles(query_flag)

描述：pg_archive_scan_residualfiles()的CN统一执行函数。该函数为集群级函数，与当前所在的数据库相关，在CN实例上运行。

参数query_flag，参数类型int，用于表示执行范围。1表示CN，2表示主DN，4表示备DN，通过或运算可以得到查询并集，如1|2=3表示CN+主DN，1|2|4=7表示CN+主DN+备DN。默认值是7。

返回类型：record

返回信息如下：

表 21-74 返回字段

名称	类型	描述
nodename	name	节点名称。
instance_id	text	实例名称。
archive	text	归档后的本地文件夹路径。OBS路径的残留文件归档在对应OBS数据库目录下。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
count	bigint	归档的残留文件数量。
size	bigint	归档的残留文件总大小，单位为byte。

示例：

```
SELECT * FROM pgxc_archive_scan_residualfiles();
  node_name   | instance_id |                          archive                          | count | size
--------------+-------------+-----------------------------------------------------------+-------+-------
 cn_5001      | cn_5001     | pg_residualfiles/archive/pgscrf_archive_20231027145050896440 |     1 | 40960
 dn_6007_6008 | dn_6008     | pg_residualfiles/archive/pgscrf_archive_20231027145051018138 |     1 | 24576
(2 rows)
```

说明

- 此函数只能归档当前登录的数据库中记录过的残留文件，归档时会记录中的残留文件进行相应的校验，校验结果分为以下三种情况：
  - 校验通过：校验通过后会对残留文件进行归档，并标记为已处理。
  - 校验失败：校验失败会跳过归档，并标记为已处理。
  - 延迟校验：延迟校验会跳过归档，通常表示当前未满足校验时机，延迟校验的情况通常和事务完成情况及备机REDO进度有关。
- 实际归档目录与对应表空间在同一个文件系统下，删除表空间也会删除对应的残留文件归档。
- 延迟DDL开启后，此函数无法使用。
- 此函数为重载操作，不支持单节点并发使用，也不建议在业务繁忙和资源负载较高时使用。

pg_rm_scan_residualfiles_archive()

描述：用于删除当前节点所有归档的残留文件记录。该函数为实例级函数，与当前所在的数据库无关，可以在任意实例上运行。

返回类型：record

返回信息如下：

表 21-75 返回字段

名称	类型	描述
count	bigint	归档中已删除的残留文件数量。本地路径的残留文件统计所删除的文件数，OBS路径的残留文件统计所删除的表目录数量。
size	bigint	归档中已删除的残留文件总大小，单位为byte。OBS路径的残留文件该项均为0。

示例：

```
SELECT * FROM pg_rm_scan_residualfiles_archive();
 count | size
-------+-------
```


版权所有 © 华为云计算技术有限公司

```
    4 | 81920
(1 row)
```

pgxc_rm_scan_residualfiles_archive(query_flag)

描述：pg_rm_scan_residualfiles_archive()的CN统一执行函数。该函数为集群级函数，与当前所在的数据库无关，在CN实例上运行。

参数query_flag，参数类型int，用于表示执行范围。1表示CN，2表示主DN，4表示备DN，通过或运算可以得到查询并集，如1|2=3表示CN+主DN，1|2|4=7表示CN+主DN+备DN。默认值是7。

返回类型：record

返回信息如下：

表 21-76 返回字段

名称	类型	描述
nodename	name	节点名称。
instance_id	text	实例名称。
count	bigint	归档中已删除的残留文件数量。本地路径的残留文件统计所删除的文件数，OBS路径的残留文件统计所删除的表目录数量。
size	bigint	归档中已删除的残留文件总大小，单位为byte。OBS路径的残留文件该项均为0。

示例：

```
SELECT * FROM pgxc_rm_scan_residualfiles_archive();
  node_name   | instance_id | count | size
--------------+-------------+-------+-------
 dn_6001_6002 | dn_6001     |     4 | 81920
 cn_5001      | cn_5001     |     1 | 40960
 dn_6007_6008 | dn_6008     |     1 | 24576
 coordinator1 | coordinator1|     1 |     0
(4 rows)
```

21.7.30.3 验证残留文件函数

pg_verify_residualfiles(filepath)

描述：用于验证参数指定文件中记录的文件是否为残留文件。该函数为实例级函数，与当前所在的数据库相关，可以在任意实例上运行。

参数类型：text

返回值类型：bool

函数返回字段如下：

版权所有 © 华为云计算技术有限公司

表 21-77 pg_verify_residualfiles(filepath)返回字段

名称	类型	描述
isverified	bool	是否完成验证。

示例：

```
SELECT * FROM pg_verify_residualfiles('pgrf_20200908160211441546');
 isverified
------------
 t
(1 row)
```

说明

本函数只能验证记录的文件在当前登录的数据库中是否是残留文件。如果记录的文件不属于当前登录的数据库，则不会进行校验行为。

pg_verify_residualfiles()

描述：用于验证当前实例上所有残留文件列表中记录的文件是否为残留文件。该函数为实例级函数，与当前所在的数据库相关，可以在任意实例上运行。

参数类型：无

返回值类型：record

函数返回字段如下：

表 21-78 pg_verify_residualfiles()返回字段

名称	类型	描述
result	bool	是否完成验证。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

```
SELECT * FROM pg_verify_residualfiles();
 result |          filepath           | notes
--------+-----------------------------+-------
 t      | pgrf_20200908160211441546 |
(1 row)
```

说明

本函数只能验证记录的文件在当前登录的数据库中是否是残留文件。如果记录的文件不属于当前登录的数据库，则不会进行校验行为。

pgxc_verify_residualfiles()

描述：pg_verify_residualfiles()的CN统一查询函数。该函数为集群级函数，与当前所在的数据库相关，在CN实例上运行。

版权所有 © 华为云计算技术有限公司

参数类型：无

返回值类型：record

函数返回字段如下：

表 21-79 pgxc_verify_residualfiles()返回字段

名称	类型	描述
nodename	text	节点名称。
result	bool	是否完成验证。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

```
SELECT * FROM pgxc_verify_residualfiles();
   nodename   | result |          filepath           | notes
--------------+--------+-----------------------------+-------
 cn_5001      | t      | pgrf_20200910170129360401 |
 dn_6001_6002 | t      | pgrf_20200908160211441546 |
(2 rows)
```

说明

本函数只能验证记录的文件在当前登录的数据库中是否是残留文件。如果记录的文件不属于当前登录的数据库，则不会进行校验行为。

pg_is_residualfiles(residualfile)

描述：用于查询当前库中指定的relfilenode是否为残留文件。该函数为实例级函数，与当前所在的数据库相关，可以在任意实例上运行。

参数类型：text

返回值类型：bool

函数返回字段如下：

表 21-80 pg_is_residualfiles(residualfile)返回字段

名称	类型	描述
result	bool	是否是残留文件。

示例：

```
SELECT * FROM pg_is_residualfiles('base/49155/114691');
 result
--------
 t
(1 row)
```

 版权所有 © 华为云计算技术有限公司

说明

本函数只能验证记录的文件在当前登录的database中是否为残留文件。如果记录的文件不属于当前登录的数据库，则会被检测为是残留文件。

例如：针对gaussdb数据库中的非残留文件base/15092/14790，如果在gaussdb库中查询，则认为是非残留文件；在其他数据库中查询，则认为是残留文件。

```
SELECT * FROM pg_is_residualfiles('base/15092/14790');
result
--------
f
(1 row)

\c db2
db2=# SELECT * FROM pg_is_residualfiles('base/15092/14790');
result
--------
t
(1 row)
```

21.7.30.4 删除残留文件函数

pg_rm_residualfiles(filepath)

描述：用于删除当前实例中指定残留文件列表中的文件。该函数为实例级函数，与当前所在的数据库无关，可以在任意实例上运行。

参数类型：text

返回值类型：record

函数返回字段如下：

表 21-81 pg_rm_residualfiles(filepath)返回字段

名称	类型	描述
result	bool	是否已经完成删除。

示例：

```
SELECT * FROM pg_rm_residualfiles('pgrf_20200908160211441599');
 result
--------
 t
(1 row)
```

说明

- 残留文件只有在调用pg_verify_residualfiles()进行verify后才能被真正删除。
- 删除动作不区分数据库，指定文件中所有已经verify的文件都会被删除。
- 如果指定文件中记录的所有文件都已经被删除，指定文件会被移除并备份到$PGDATA/pg_residualfile/backup目录下。

pg_rm_residualfiles()

描述：用于删除当前实例中所有的残留文件列表中的文件。该函数为实例级函数，与当前所在的数据库无关，可以在任意实例上运行。

版权所有 © 华为云计算技术有限公司

参数类型：无

返回值类型：record

函数返回字段如下：

表 21-82 pg_rm_residualfiles()返回字段

名称	类型	描述
result	bool	是否已经完成删除。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

```
SELECT * FROM pg_rm_residualfiles();
 result |          filepath           | notes
--------+-----------------------------+-------
 t      | pgrf_20200908160211441546 |
(1 row)
```

说明

- 残留文件只有在调用pg_verify_residualfiles()进行验证后才能被真正删除。
- 删除动作不区分数据库，指定文件中所有已经验证的文件都会被删除。
- 如果指定文件中记录的所有文件都已经被删除，指定文件会被移除并备份到$PGDATA/pg_residualfile/backup目录下。

pgxc_rm_residualfiles()

描述：pgxc_rm_residualfiles的CN统一查询函数。该函数为集群级函数，与当前所在的数据库无关，在CN实例上运行。

参数类型：无

返回值类型：record

函数返回字段如下：

表 21-83 pgxc_rm_residualfiles()返回字段

名称	类型	描述
nodename	text	节点名。
result	bool	是否已经完成删除。
filepath	text	残留文件记录路径。
notes	text	注释。

示例：

版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pgxc_rm_residualfiles();
   nodename   | result |         filepath          | notes
--------------+--------+---------------------------+-------
 cn_5001      | t      | pgrf_20200910170129360401 |
 dn_6001_6002 | t      | pgrf_20200908160211441546 |
(2 rows)
```

pgxc_clear_disk_cache()

描述：用于完全清理磁盘缓存文件。该函数仅9.0.2及以上集群版本支持。

返回值类型：void

示例：

```
SELECT pgxc_clear_disk_cache();
 pgxc_clear_disk_cache
-----------------------

(1 row)
```

21.7.30.5 残留文件管理函数应用

使用步骤

步骤1 调用pgxc_get_residualfiles()函数，获取存在残留文件的数据库名称。

步骤2 分别进入确认有残留文件的数据库，调用pgxc_verify_residualfiles()函数，对当前数据库中记录的残留文件进行验证。

步骤3 调用pgxc_rm_residualfiles()函数，删除所有已经验证过的残留文件。

----结束

说明

pgxc类残留文件管理函数只对CN和当前主DN进行操作，不会验证和清理备DN上的残留文件。所以主DN完成清理后，应在备DN上及时执行残留文件清理操作或对备机进行build，防止主备切换后由于增量build导致备机残留文件被重新复制回主DN，导致未成功清理的假象。

使用示例

以当前两个用户自建的数据库db1、db2为例：

```
postgres=# \l
                                  List of databases
   Name    | Owner  | Encoding  | Collate | Ctype | Access privileges
-----------+--------+-----------+---------+-------+-------------------
 db1       | fpi810 | SQL_ASCII | C       | C     |
 db2       | fpi810 | SQL_ASCII | C       | C     |
 postgres  | fpi810 | SQL_ASCII | C       | C     |
 template0 | fpi810 | SQL_ASCII | C       | C     | =c/fpi810        +
           |        |           |         |       | fpi810=CTc/fpi810
 template1 | fpi810 | SQL_ASCII | C       | C     | =c/fpi810        +
           |        |           |         |       | fpi810=CTc/fpi810
(5 rows)
```

1. 在CN上获取集群的所有残留文件记录：

```
db1=# SELECT * FROM pgxc_get_residualfiles() order by 4, 6; -- order by不是必须的
```


版权所有 © 华为云计算技术有限公司

```
db1=# select * from pgxc_get_residualfiles() order by 4, 6;
   nodename    | isverified | isdeleted | dbname |   residualfile   |         filepath          | notes
---------------+------------+-----------+--------+------------------+---------------------------+-------
 dn_6001_6002 | f          | f         | db1    | base/16390/16395 | pgrf_20200921153355979438 |
 dn_6001_6002 | f          | f         | db1    | base/16390/24592 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24579 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24582 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24584 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24585 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24588 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24589 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24591 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24576 | pgrf_20200921153612088342 |
(10 rows)
```

当前集群中：

- dn_6001_6002 节点（当前的主节点实例）的db1和db2数据库中都存在残留文件记录。
- 残留文件在residualfile列展示。
- filepath列为记录残留文件的记录文件，保存在实例数据目录下pg_residualfiles目录中。

2. 调用pgxc_verify_residualfiles()函数对db1库进行验证：

```
db1=# SELECT * FROM pgxc_verify_residualfiles();
```

```
postgres=# \c db1
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "db1" as user "fpi810".
db1=# select * from pgxc_verify_residualfiles();
   nodename    | result |         filepath          | notes
---------------+--------+---------------------------+-------
 dn_6001_6002 | t      | pgrf_20200921153355979438 |
 dn_6001_6002 | t      | pgrf_20200921153612088342 |
(2 rows)
```

因为**verify类函数都是database级别**，所以当前在db1库中调用verify函数时，只对属于db1的残留文件进行验证。

可以再次调用get函数查看是否验证完成：

```
db1=# SELECT * FROM pgxc_get_residualfiles() order by 4, 6;
```

```
db1=# select * from pgxc_get_residualfiles() order by 4, 6;
   nodename    | isverified | isdeleted | dbname |   residualfile   |         filepath          | notes
---------------+------------+-----------+--------+------------------+---------------------------+-------
 dn_6001_6002 | t          | f         | db1    | base/16390/16395 | pgrf_20200921153355979438 |
 dn_6001_6002 | t          | f         | db1    | base/16390/24592 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24579 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24582 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24584 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24585 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24588 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24589 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24591 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24576 | pgrf_20200921153612088342 |
(10 rows)
```

如上图所示，已确认db1数据库中的残留文件都已经验证，db2数据库中的残留文件都未进行验证。

3. 调用 pgxc_rm_residualfiles()函数删除残留文件。

```
db1=# SELECT * FROM pgxc_rm_residualfiles();
```

```
db1=# select * from pgxc_rm_residualfiles();
   nodename    | result |         filepath          | notes
---------------+--------+---------------------------+-------
 dn_6001_6002 | t      | pgrf_20200921153355979438 |
 dn_6001_6002 | t      | pgrf_20200921153612088342 |
(2 rows)
```

4. 再次调用pgxc_get_residualfiles()函数检查删除后的结果。

版权所有 © 华为云计算技术有限公司

```
db1=# select * from pgxc_get_residualfiles() order by 4, 6;
   nodename   | isverified | isdeleted | dbname |   residualfile   |         filepath          | notes
--------------+------------+-----------+--------+------------------+---------------------------+-------
 dn_6001_6002 | t          | t         | db1    | base/16390/24592 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24579 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24582 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24584 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24576 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24588 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24589 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24591 | pgrf_20200921153612088342 |
 dn_6001_6002 | f          | f         | db2    | base/16391/24585 | pgrf_20200921153612088342 |
(9 rows)
```

结果显示db1数据库中的残留文件已经被删除（isdeleted标记为t），db2中的残留文件都未被删除。

同时可以看到查询出9条结果，与之前查询出的结果相比，缺少一条以9438结尾的残留文件记录文件。这是因为以9438结尾的残留文件记录文件中只有一条残留文件记录，这条记录在步骤3中被删除，当记录文件中的所有残留文件都被删除后，记录文件本身也会被删除，并备份到pg_residualfiles/backup目录中：

```
[fpi810@host-192-168-244-162 pg_residualfiles]$ ls
backup  pendingdeletesfile  pgrf_20200921153612088342
[fpi810@host-192-168-244-162 pg_residualfiles]$ ls backup/
pgrf_20200921153355979438_bak
```

5. 如果需要删除db2数据库中的文件，需要在db2中调用verify函数后再调用rm函数。

   a. 进入db2数据库，并调用验证函数：

   ```
   db1=# \c db2
   Non-SSL connection (SSL connection is recommended when requiring high-security)
   You are now connected to database "db2" as user "fpi810".
   db2=# select * from pgxc_verify_residualfiles();
      nodename   | result |         filepath          | notes
   --------------+--------+---------------------------+-------
    dn_6001_6002 | t      | pgrf_20200921153612088342 |
   (1 row)
   ```

   此时可以查询验证的结果：

   ```
   db2=# select * from pgxc_get_residualfiles() order by 4, 6;
      nodename   | isverified | isdeleted | dbname |   residualfile   |         filepath          | notes
   --------------+------------+-----------+--------+------------------+---------------------------+-------
    dn_6001_6002 | t          | t         | db1    | base/16390/24592 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24579 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24582 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24584 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24576 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24588 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24589 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24591 | pgrf_20200921153612088342 |
    dn_6001_6002 | t          | f         | db2    | base/16391/24585 | pgrf_20200921153612088342 |
   (9 rows)
   ```

   b. 调用删除函数：

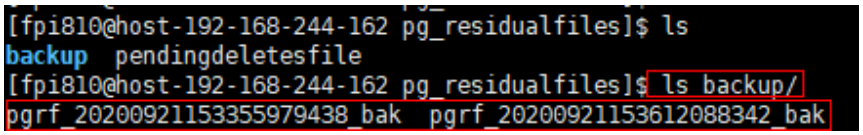
   ```
   db2=# select * from pgxc_rm_residualfiles();
      nodename   | result |         filepath          | notes
   --------------+--------+---------------------------+-------
    dn_6001_6002 | t      | pgrf_20200921153612088342 |
   (1 row)
   ```

   c. 再查询删除的结果：

   ```
   db2=# select * from pgxc_get_residualfiles() order by 4, 6;
    nodename | isverified | isdeleted | dbname | residualfile | filepath | notes
   ----------+------------+-----------+--------+--------------+----------+-------
   (0 rows)
   ```


版权所有 © 华为云计算技术有限公司

此时因为 8342 结尾的记录文件中残留文件已经全部删除，所以整个记录文件也被删除并备份到backup目录下，所以查询到0条记录。



21.7.31 统计信息函数

统计信息函数根据访问对象分为两种类型：针对某个数据库进行访问的函数，以数据库中每个表或索引的OID作为参数，标识需要报告的数据库；针对某个服务器进行访问的函数，以一个服务器进程号为参数，其范围从1到当前活跃服务器的数目。

对于全局临时表，函数入参是表在pg_class中的OID，函数返回当前会话数据的统计信息，所以同一张全局临时表在不同会话查询下列统计信息函数得到的结果可能不同。

pg_stat_get_db_numbackends(oid)

描述：查询当前实例上指定数据库活跃的服务器线程数目。

返回值类型：integer

pg_stat_get_db_total_numbackends(oid)

描述：在CN上执行该函数，返回集群中所有CN上指定数据库活跃的服务器线程总数。在DN上执行该函数，返回当前实例上指定数据库活跃的服务器线程数目。

返回值类型：integer

pg_stat_get_db_xact_commit(oid)

描述：返回当前实例上指定数据库中已提交事务的数量。

返回值类型：bigint

pg_stat_get_db_total_xact_commit(oid)

描述：在CN上执行该函数，返回集群中所有CN上指定数据库中已提交事务的总数。在DN上执行该函数，返回当前实例上指定数据库中已提交事务的数量。

返回值类型：bigint

pg_stat_get_db_xact_rollback(oid)

描述：返回当前实例上指定数据库中回滚事务的数量。

返回值类型：bigint

pg_stat_get_db_total_xact_rollback(oid)

描述：在CN上执行该函数，返回集群中所有CN上指定数据库中回滚事务的总数。在DN上执行该函数，返回当前实例上指定数据库中回滚事务的数量。

返回值类型：bigint

版权所有 © 华为云计算技术有限公司

pg_stat_get_db_blocks_fetched(oid)

描述：返回当前实例上指定数据库中磁盘块抓取请求的数量。

返回值类型：bigint

pg_stat_get_db_total_blocks_fetched(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库中磁盘块抓取请求的总数。在DN上执行该函数，返回当前实例上指定数据库中磁盘块抓取请求的数量。

返回值类型：bigint

pg_stat_get_db_blocks_hit(oid)

描述：返回当前实例上指定数据库在缓冲区中找到的请求磁盘块的数量。

返回值类型：bigint

pg_stat_get_db_total_blocks_hit(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库在缓冲区中找到的请求磁盘块的总数。在DN上执行该函数，返回当前实例上指定数据库在缓冲区中找到的请求磁盘块的数量。

返回值类型：bigint

pg_stat_get_db_tuples_returned(oid)

描述：返回当前实例上指定数据库返回的元组数量。

返回值类型：bigint

pg_stat_get_db_total_tuples_returned(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库返回的元组总数。在DN上执行该函数，返回当前实例上指定数据库返回的元组数量。

返回值类型：bigint

pg_stat_get_db_tuples_fetched(oid)

描述：返回当前实例上指定数据库中读取的元组数量。

返回值类型：bigint

pg_stat_get_db_total_tuples_fetched(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库读取的元组总数。在DN上执行该函数，返回当前实例上指定数据库读取的元组数量。

返回值类型：bigint

pg_stat_get_db_tuples_inserted(oid)

描述：返回当前实例上指定数据库中插入的元组数量。

版权所有 © 华为云计算技术有限公司

返回值类型：bigint

pg_stat_get_db_total_tuples_inserted(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库插入的元组总数。在DN上执行该函数，返回当前实例上指定数据库插入的元组数量。

返回值类型：bigint

pg_stat_get_db_tuples_updated(oid)

描述：返回当前实例上指定数据库中更新的元组数量。

返回值类型：bigint

pg_stat_get_db_total_tuples_updated(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库更新的元组总数。在DN上执行该函数，返回当前实例上指定数据库更新的元组数量。

返回值类型：bigint

pg_stat_get_db_tuples_deleted(oid)

描述：返回当前实例上指定数据库中删除的元组数量。

返回值类型：bigint

pg_stat_get_db_total_tuples_deleted(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库删除的元组总数。在DN上执行该函数，返回当前实例上指定数据库删除的元组数量。

返回值类型：bigint

pg_stat_get_db_conflict_lock(oid)

描述：在CN上执行该函数，返回集群中所有CN和DN上指定数据库锁冲突的总数。在DN上执行该函数，返回当前实例上指定数据库中锁冲突数量。

返回值类型：bigint

pg_stat_get_db_deadlocks(oid)

描述：返回当前实例上指定数据库中死锁的数量。

返回值类型：bigint

pg_stat_get_db_total_deadlocks(oid)

描述：在CN上执行该函数，返回集群中所有CN和DN上指定数据库死锁的总数。在DN上执行该函数，返回当前实例上指定数据库中死锁的数量。

返回值类型：bigint

版权所有 © 华为云计算技术有限公司

pg_stat_get_db_conflict_all(oid)

描述：返回当前实例上指定数据库中发生冲突恢复的次数。

返回值类型：bigint

pg_stat_get_db_total_conflict_all(oid)

描述：在CN上执行该函数，返回集群中所有CN和DN上指定数据库发生冲突恢复的总次数。在DN上执行该函数，返回当前实例上指定数据库中发生冲突恢复的次数。

返回值类型：bigint

pg_stat_get_db_temp_files(oid)

描述：返回当前实例上指定数据库中创建临时文件的个数。

返回值类型：bigint

pg_stat_get_db_total_temp_files(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库中创建临时文件的总个数。在DN上执行该函数，返回当前实例上指定数据库中创建临时文件的个数。

返回值类型：bigint

pg_stat_get_db_temp_bytes(oid)

描述：返回当前实例上指定数据库中创建临时文件的字节数。

返回值类型：bigint

pg_stat_get_db_total_temp_bytes(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库中创建临时文件的总字节数。在DN上执行该函数，返回当前实例上指定数据库中创建临时文件的字节数。

返回值类型：bigint

pg_stat_get_db_blk_read_time(oid)

描述：返回当前实例上指定数据库中读数据块所用的时间。

返回值类型：double

pg_stat_get_db_total_blk_read_time(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库中读数据块所用的总时间。在DN上执行该函数，返回当前实例上指定数据库中读数据块所用的时间。

返回值类型：double

pg_stat_get_db_blk_write_time(oid)

描述：返回当前实例上指定数据库中写数据块所用的时间。

返回值类型：double

 版权所有 © 华为云计算技术有限公司

pg_stat_get_db_total_blk_write_time(oid)

描述：在CN上执行该函数，返回集群中所有DN上指定数据库中写数据块所用的总时间。在DN上执行该函数，返回当前实例上指定数据库中写数据块所用的时间。

返回值类型：double

pg_stat_get_numscans(oid)

描述：如果参数是一个表，则顺序扫描读取的行数目。

如果参数是一个索引，则返回索引行的数目。

返回值类型：bigint

pg_stat_get_tuple()

描述：在CN和DN上均可以执行，该函数仅8.1.3及以上集群版本支持。

函数无参时，查询CN上所有系统表的统计信息及表在每个CN上的脏页率，查询DN上所有系统表和用户表的统计信息和表在每个DN上的脏页率；

函数带入参时，入参是schema和表名，带入参的函数执行时查询单张表的统计信息和脏页率。

说明

该函数的统计信息依赖于ANALYZE，为获取最准确的信息请先对表进行ANALYZE。

返回值类型：record

函数返回字段如下：

名称	类型	描述
nodename	text	节点名。
tableid	oid	表的OID。
partid	oid	分区表的分区OID。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次手动analyze时间。
last_autoanalyze	timestamp with time zone	最后一次autoanalyze时间。
vacuum_count	bigint	vacuum次数。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
autoanalyze_count	bigint	自动分析的次数。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	HOT更新的行数。
n_tup_change	bigint	analyze之后改变的行数。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
dirty_rate	bigint	单节点的脏页率（单CN或单DN级）。
last_data_changed	timestamp with time zone	记录表最后一次数据变化的时间。该字段在8.2.0及以上版本固定为空，建议通过pg_stat_get_last_data_changed_time(oid)查询。

pg_stat_get_tuples_returned(oid)

描述：如果参数是一个表，则顺序扫描读取的行数目。

如果参数是一个索引，则返回的索引行的数目。

返回值类型：bigint

pg_stat_get_tuples_fetched(oid)

描述：如果参数是一个表，则位图扫描抓取的行数目。

如果参数是一个索引，则用简单索引扫描抓取的行数目。

返回值类型：bigint

pg_stat_get_tuples_inserted(oid)

描述：插入表中行的数量。

返回值类型：bigint

pg_stat_get_local_tuples_inserted(oid)

描述：当前节点上插入表中行的数量，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

 版权所有 © 华为云计算技术有限公司

pg_stat_get_tuples_updated(oid)

描述：在表中已更新行的数量。

返回值类型：bigint

pg_stat_get_local_tuples_updated(oid)

描述：当前节点上在表中已更新行的数量，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_tuples_deleted(oid)

描述：从表中删除行的数量。

返回值类型：bigint

pg_stat_get_local_tuples_deleted(oid)

描述：当前节点上从表中删除行的数量，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_tuples_changed(oid)

描述：在CN上查询该函数，返回该表自上一次analyze或autoanalyze之后插入、更新、删除行的总数量。在DN上查询该函数，返回该表在当前节点上自上一次analyze或autoanalyze之后插入、更新、删除行的总数量。

返回值类型：bigint

pg_stat_get_local_tuples_changed(oid)

描述：该表在当前节点上自从上一次analyze或autoanalyze之后插入、更新、删除行的数量。

返回值类型：bigint

pg_stat_get_tuples_hot_updated(oid)

描述：热更新的行数表。

返回值类型：bigint

pg_stat_get_local_tuples_hot_updated(oid)

描述：当前节点上热更新的行数表，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_live_tuples(oid)

描述：表格的活元组数。

返回值类型：bigint

 版权所有 © 华为云计算技术有限公司

pg_stat_get_local_live_tuples(oid)

描述：当前节点上表格的活元组数，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_dead_tuples(oid)

描述：表格的死元组数。

返回值类型：bigint

pg_stat_get_local_dead_tuples(oid)

描述：当前节点上表格的死元组数，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_blocks_fetched(oid)

描述：表或者索引的磁盘块抓取请求的数量。

返回值类型：bigint

pg_stat_get_blocks_hit(oid)

描述：在缓冲区中找到的表或者索引的磁盘块请求数目。

返回值类型：bigint

pg_stat_get_partition_tuples_inserted(oid)

描述：插入相应表分区中行的数量。

返回值类型：bigint

pg_stat_get_partition_tuples_updated(oid)

描述：在相应表分区中已更新行的数量。

返回值类型：bigint

pg_stat_get_partition_tuples_deleted(oid)

描述：从相应表分区中删除行的数量。

返回值类型：bigint

pg_stat_get_partition_tuples_changed(oid)

描述：该表分区上一次analyze或autoanalyze之后插入、更新、删除行的总数量。

返回值类型：bigint

pg_stat_get_partition_live_tuples(oid)

描述：活行数表分区。

 版权所有 © 华为云计算技术有限公司

返回值类型：bigint

pg_stat_get_partition_dead_tuples(oid)

描述：死行数表分区。

返回值类型：bigint

pg_stat_get_xact_tuples_inserted(oid)

描述：表相关的活跃子事务中插入的tuple数。

返回值类型：bigint

pg_stat_get_xact_tuples_deleted(oid)

描述：表相关的活跃子事务中删除的tuple数。

返回值类型：bigint

pg_stat_get_xact_tuples_hot_updated(oid)

描述：表相关的活跃子事务中热更新的tuple数。

返回值类型：bigint

pg_stat_get_xact_tuples_updated(oid)

描述：表相关的活跃子事务中更新的tuple数。

返回值类型：bigint

pg_stat_get_xact_partition_tuples_inserted(oid)

描述：表分区相关的活跃子事务中插入的tuple数。

返回值类型：bigint

pg_stat_get_xact_partition_tuples_deleted(oid)

描述：表分区相关的活跃子事务中删除的tuple数。

返回值类型：bigint

pg_stat_get_xact_partition_tuples_hot_updated(oid)

描述：表分区相关的活跃子事务中热更新的tuple数。

返回值类型：bigint

pg_stat_get_xact_partition_tuples_updated(oid)

描述：表分区相关的活跃子事务中更新的tuple数。

返回值类型：bigint

 版权所有 © 华为云计算技术有限公司

pg_stat_get_last_vacuum_time(oid)

描述：用户在该表上最后一次手动启动清理或者autovacuum线程启动清理的时间。

返回值类型：timestamptz

pg_stat_get_last_autovacuum_time(oid)

描述：autovacuum守护线程在该表上最后一次启动清理的时间。

返回值类型：timestamptz

pg_stat_get_local_last_autovacuum_time(oid)

描述：当前节点的autovacuum守护线程在该表上最后一次启动清理的时间，该函数仅8.1.2及以上版本支持。

返回值类型：timestamptz

pg_stat_get_vacuum_count(oid)

描述：用户在该表上手动启动清理的次数。

返回值类型：bigint

pg_stat_get_autovacuum_count(oid)

描述：autovacuum守护线程在该表上启动清理的次数。

返回值类型：bigint

pg_stat_get_local_autovacuum_count(oid)

描述：当前节点上的autovacuum守护线程在该表上启动清理的次数，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_last_analyze_time(oid)

描述：用户在该表上最后一次手动启动分析或者autovacuum线程启动分析的时间。

返回值类型：timestamptz

pg_stat_get_last_autoanalyze_time(oid)

描述：autovacuum守护线程在该表上最后一次启动分析的时间。

返回值类型：timestamptz

pg_stat_get_local_last_autoanalyze_time(oid)

描述：当前节点的autovacuum守护线程在该表上最后一次启动分析的时间，该函数仅8.1.2及以上版本支持。

返回值类型：timestamptz

 版权所有 © 华为云计算技术有限公司

pg_stat_get_analyze_count(oid)

描述：用户在该表上手动启动分析的次数。

返回值类型：bigint

pg_stat_get_autoanalyze_count(oid)

描述：autovacuum守护线程在该表上启动分析的次数。

返回值类型：bigint

pg_stat_get_local_autoanalyze_count(oid)

描述：当前节点的autovacuum守护线程在该表上启动分析的次数，该函数仅8.1.2及以上版本支持。

返回值类型：bigint

pg_stat_get_local_analyze_status(oid)

描述：指定表在当前节点上的是否需要analyze的状态，仅在CN端有意义。该函数仅8.1.2及以上版本支持。

- 如果该表的修改行数超过analyze的阈值（根据autovacuum_analyze_threshold + autovacuum_analyze_scale_factor * reltuples计算，其中reltuples是pg_class中记录的表的估算行数），则返回“Analyze needed”。
- 如果该表的修改行数不超过analyze的阈值，则返回“Analyze not needed”。

返回值类型：text

pg_stat_get_runtime_relstats(oid)

描述：返回轻量化autoanalyze生成的内存表级统计信息。

返回值类型：setof record

pg_stat_get_runtime_attstats(oid)

描述：返回轻量化autoanalyze生成的内存表列级统计信息。

返回值类型：setof record

pg_stat_get_runtime_partstats(oid)

描述：返回轻量化autoanalyze生成的内存分区级统计信息。仅8.2.1.200及以上集群版本支持。

返回值类型：setof record

pg_stat_get_runtime_part_attstats(oid)

描述：返回轻量化autoanalyze生成的内存分区列级统计信息。仅8.2.1.200及以上集群版本支持。

返回值类型：setof record

 版权所有 © 华为云计算技术有限公司

pg_stat_get_runtime_exprstats(oid)

描述：返回轻量化autoanalyze生成的内存表达式统计信息。仅8.2.1.200及以上集群版本支持。

返回值类型：setof record

pg_stat_get_predicate_columns(oid)

描述：返回表收集到的谓词列信息。仅9.1.0.100及以上集群版本支持。

返回值类型：setof smallint

pg_stat_delete_predicate_columns(oid)

描述：清空表的谓词列信息。仅9.1.0.100及以上集群版本支持。

返回值类型：void

pg_total_autovac_tuples(bool)

描述：返回total autovac相关的tuple记录，如nodename,nspname,relname以及各类tuple的IUD信息。

返回值类型：setof record

pg_autovac_status(oid)

描述：返回和autovac状态相关的参数信息，如nodename,nspname,relname,analyze,vacuum设置，analyze/vacuum阈值, analyze/vacuum tuple数等。

返回值类型：setof record

pg_autovac_timeout(oid)

描述：返回某个表做autovac连续超时的次数，表信息非法或node信息异常返回NULL。

返回值类型：bigint

pg_autovac_coordinator(oid)

描述：返回对某个表做autovac的coordinator名字，表信息非法或node信息异常返回NULL。

返回值类型：text

pg_stat_get_all(regclass)

描述：返回表在当前CN上pg_stat_object中对应的tuple记录，包括从databaseid到autovac_vacuum_count共29个字段。入参为表的oid。8.2.1.210及以上集群版本支持。

返回值类型：setof record

示例：查询当前节点public模式下t1表的最近一次访问时间。

 版权所有 © 华为云计算技术有限公司

```
SELECT data_access_timestamp FROM pg_stat_get_all('public.t1'::regclass);
data_access_timestamp
-------------------------------
2024-06-27 16:16:26.643132+08
(1 row)
```

pgxc_stat_get_all(relnamespace text, relname text)

描述：返回表在所有CN上pg_stat_object中对应的tuple记录，包括从databaseid到autovac_vacuum_count共29个字段。8.2.1.210及以上集群版本支持。

返回值类型：setof record

示例：查询所有CN上public模式下t1表的最近一次访问时间。

```
SELECT relid,data_access_timestamp FROM pgxc_stat_get_all('public','t1');
 relid      | data_access_timestamp
------------+-------------------------------
2147516676 | 2000-01-01 08:00:00+08
2147516674 | 2000-01-01 08:00:00+08
2147516655 | 2024-06-27 16:16:26.643132+08
(3 rows)
```

pgxc_stat_get_last_data_access_timestamp(relnamespace text, relname text)

描述：返回指定表在所有节点中最近一次的访问时间。9.1.0.100及以上集群版本支持。

返回值类型：timestamptz

示例：查询所有节点上public模式下t1表的最近一次访问时间。

```
SELECT * FROM pgxc_stat_get_last_data_access_timestamp('public','t1');
pgxc_stat_get_last_data_access_timestamp
-----------------------------------------
2024-06-27 16:16:26.643132+08
(1 row)
```

pgxc_stat_flush_object_data()

描述：手动执行持久化，每个节点将本地所有库中的所有表的统计信息写入pg_stat_object系统表。统计信息包括pg_stat_object中从databaseid到autovac_vacuum_count共29个字段。只有系统管理员可以执行此函数。9.1.0.100及以上集群版本支持。

返回值类型：boolean

示例：系统管理员用户执行持久化。

```
SELECT pgxc_stat_flush_object_data();
pgxc_stat_flush_object_data
-----------------------------
t
(1 row)
```

说明

此函数不支持并发执行。

版权所有 © 华为云计算技术有限公司

pgxc_get_wlm_session_info_bytime(text, timestamp without time zone, timestamp without time zone, int)

描述：PGXC_WLM_SESSION_INFO视图在统计数据量很大的场景中性能较差，建议使用该函数进行筛选查询。入参分别为：筛选时间列（'start_time', 'finish_time'），起始区间时间，结束区间时间，每个CN返回的最大数量。返回值为GS_WLM_SESSION_HISTORY。

返回值类型：setof record

pgxc_get_wlm_current_instance_info(text, int default null)

描述：在CN节点上查询集群各节点当前的资源使用情况，读取内存中还未存到GS_WLM_INSTANCE_HISTORY系统表的数据。入参分别为：节点名称(可以输入ALL、C、D、实例名称)，每个节点返回的最大数量。返回值为GS_WLM_INSTANCE_HISTORY。

返回值类型：setof record

pgxc_get_wlm_history_instance_info(text, TIMESTAMP, TIMESTAMP, int default null)

描述：在CN节点上查询集群各节点历史资源使用情况，读取GS_WLM_INSTANCE_HISTORY系统表的数据。入参分别为：节点名称(可以输入ALL、C、D、实例名称)，起始区间时间，结束区间时间，每个实例返回的最大数量。返回值为GS_WLM_INSTANCE_HISTORY。

返回值类型：setof record

pg_stat_get_last_data_changed_time(oid)

描述：insert/update/delete, exchange/truncate/drop partition在该表上最后一次操作的时间，PG_STAT_ALL_TABLES视图last_data_changed列的数据是通过该函数求值，在表数量很大的场景中，通过视图获取表数据最后修改时间的性能较差，建议直接使用该函数获取表数据的最后修改时间。

返回值类型：timestamptz

pg_stat_set_last_data_changed_time(oid)

描述：手动设置该表上最后一次insert/update/delete, exchange/truncate/drop partition操作的时间。

返回值类型：void

pg_stat_set_last_data_changed_num(oid)

描述：设置该表上当前节点的历史累计修改计数。

返回值类型：void

pv_session_time()

描述：统计当前节点各会话线程的运行时间信息及各执行阶段所消耗时间。

版权所有 © 华为云计算技术有限公司

返回值类型：record

pv_instance_time()

描述：统计当前节点的运行时间信息及各执行阶段所消耗时间。

返回值类型：record

pg_stat_get_activity(integer)

描述：返回一个关于带有特殊PID的后台进程的记录信息，当参数为NULL时，则返回每个活动的后台进程的记录。返回结果是PG_STAT_ACTIVITY视图中的一个子集，不包含connection_info列。

返回值类型：setof record

pg_stat_get_activity_with_conninfo(integer)

描述：返回一个关于带有特殊PID的后台进程的记录信息，当参数为NULL时，则返回每个活动的后台进程的记录。返回结果是PG_STAT_ACTIVITY视图中的一个子集。

返回值类型：setof record

pg_user_iostat(text)

描述：该函数8.1.2版本中已废弃，当前版本查询无效。8.2.0及以上集群版本中可使用PGXC_TOTAL_USER_RESOURCE_INFO视图查询实例上用户实时资源消耗信息。

返回值类型：record

名称	类型	描述
userid	oid	用户ID。
min_curr_iops	int4	当前该用户io在各DN中的最小值。
max_curr_iops	int4	当前该用户io在各DN中的最大值。
min_peak_iops	int4	该用户io峰值中，各DN的最小值。
max_peak_iops	int4	该用户io峰值中，各DN的最大值。
io_limits	int4	用户指定的资源池所设置的io_limits。
io_priority	text	该用户所设io_priority。

pg_stat_get_function_calls(oid)

描述：函数已被调用次数。

返回值类型：bigint

 版权所有 © 华为云计算技术有限公司

pg_stat_get_function_total_time(oid)

描述：该函数花费的总挂钟时间，以微秒为单位。包括花费在此函数调用上的时间。

返回值类型：double precision

pg_stat_get_function_self_time(oid)

描述：在当前事务中仅花费在此函数上的时间。不包括花费在调用函数上的时间。

返回值类型：double precision

pg_stat_get_backend_idset()

描述：设置当前活动的服务器进程数（从1到活动服务器进程的数量）。

返回值类型：setofinteger

pg_stat_get_backend_pid(integer)

描述：给定的服务器线程的线程ID。

返回值类型：bigint

```
SELECT pg_stat_get_backend_pid(1);
 pg_stat_get_backend_pid
-------------------------
        139706243217168
(1 row)
```

pg_stat_get_backend_dbid(integer)

描述：连接到给定服务器进程的数据库ID。

返回值类型：oid

pg_stat_get_backend_userid(integer)

描述：给定的服务器进程的用户ID。

返回值类型：oid

pg_stat_get_backend_activity(integer)

描述：给定服务器进程的当前活动查询，仅在调用者是系统管理员或被查询会话的用户，并且打开track_activities的时候才能获得结果。

返回值类型：text

pg_stat_get_backend_waiting(integer)

描述：如果给定服务器进程在等待某个锁，并且调用者是系统管理员或被查询会话的用户，并且打开track_activities的时候才返回真。

返回值类型：boolean

 版权所有 © 华为云计算技术有限公司

pg_stat_get_backend_activity_start(integer)

描述：给定服务器进程当前正在执行的查询的起始时间，仅在调用者是系统管理员或被查询会话的用户，并且打开track_activities的时候才能获得结果。

返回值类型：timestampwithtimezone

pg_stat_get_backend_xact_start(integer)

描述：给定服务器进程当前正在执行的事务的开始时间，但只有当前用户是系统管理员或被查询会话的用户，并且打开track_activities的时候才能获得结果。

返回值类型：timestampwithtimezone

pg_stat_get_backend_start(integer)

描述：给定服务器进程启动的时间，如果当前用户不是系统管理员或被查询的后端的用户，则返回NULL。

返回值类型：timestampwithtimezone

pg_stat_get_backend_client_addr(integer)

描述：连接到给定客户端后端的IP地址。

如果是通过Unix域套接字连接的则返回NULL；如果当前用户不是系统管理员或被查询会话的用户，也返回NULL。

返回值类型：inet

备注：该函数中IP地址作为入参时，需写为不带圆点的格式，例如，192.168.100.128写为192168100128。

pg_stat_get_backend_client_port(integer)

描述：连接到给定客户端后端的TCP端口。

如果是通过Unix域套接字连接的则返回-1；如果当前用户不是系统管理员或被查询会话的用户，也返回NULL。

返回值类型：integer

pg_stat_get_bgwriter_timed_checkpoints()

描述：后台写进程开启定时检查点的时间（因为checkpoint_timeout时间已经过期了）。

返回值类型：bigint

pg_stat_get_bgwriter_requested_checkpoints()

描述：后台写进程开启基于后端请求的检查点的时间，因为已经超过了checkpoint_segments或因为已经执行了CHECKPOINT。

返回值类型：bigint

 版权所有 © 华为云计算技术有限公司

pg_stat_get_bgwriter_buf_written_checkpoints()

描述：在检查点期间后台写进程写入的缓冲区数目。

返回值类型：bigint

pg_stat_get_bgwriter_buf_written_clean()

描述：为日常清理脏块，后台写进程写入的缓冲区数目。

返回值类型：bigint

pg_stat_get_bgwriter_maxwritten_clean()

描述：后台写进程停止清理扫描的时间，因为已经写入了更多的缓冲区（相比bgwriter_lru_maxpages参数声明的缓冲区数）。

返回值类型：bigint

pg_stat_get_buf_written_backend()

描述：后端进程写入的缓冲区数，因为它们需要分配一个新的缓冲区。

返回值类型：bigint

pg_stat_get_buf_alloc()

描述：分配的总缓冲区数。

返回值类型：bigint

pg_stat_clear_snapshot()

描述：清理当前的统计快照。

返回值类型：void

pg_stat_reset()

描述：为当前数据库重置统计计数器为0（需要系统管理员权限）。

返回值类型：void

pg_stat_reset_shared(text)

描述：重置shared cluster每个节点当前数据统计计数器为0（需要系统管理员权限）。

返回值类型：void

pg_stat_reset_single_table_counters(oid)

描述：为当前数据库中的一个表或索引重置统计为0（需要系统管理员权限）。

返回值类型：void

 版权所有 © 华为云计算技术有限公司

pg_stat_reset_single_function_counters(oid)

描述：为当前数据库中的一个函数重置统计为0（需要系统管理员权限）。

返回值类型：void

pg_stat_session_cu(int, int, int)

描述：获取当前节点所运行session的CU命中统计信息。

返回值类型：record

gs_get_stat_session_cu(text, int, int, int)

描述：获取集群所有运行session的CU命中统计信息。

返回值类型：record

gs_get_stat_db_cu(text, text, bigint, bigint, bigint)

描述：获取集群一个数据库的CU命中统计信息。

返回值类型：record

pg_stat_get_cu_mem_hit(oid)

描述：获取当前节点当前数据库中一个列存表的CU内存命中次数。

返回值类型：bigint

pg_stat_get_cu_hdd_sync(oid)

描述：获取当前节点当前数据库中一个列存表从磁盘同步读取CU次数。

返回值类型：bigint

pg_stat_get_cu_hdd_asyn(oid)

描述：获取当前节点当前数据库中一个列存表从磁盘异步读取CU次数。

返回值类型：bigint

pg_stat_get_db_cu_mem_hit(oid)

描述：获取当前节点一个数据库CU内存命中次数。

返回值类型：bigint

pg_stat_get_db_cu_hdd_sync(oid)

描述：获取当前节点一个数据库从磁盘同步读取CU次数。

返回值类型：bigint

pg_stat_get_db_cu_hdd_asyn(oid)

描述：获取当前节点一个数据库从磁盘异步读取CU次数。

 版权所有 © 华为云计算技术有限公司

返回值类型：bigint

pgxc_fenced_udf_process()

描述：查看UDF Master和Work进程数。

返回值类型：record

pgxc_terminate_all_fenced_udf_process()

描述：Kill所有的UDF Work进程。

返回值类型：bool

gs_all_nodegroup_control_group_info(text)

描述：提供了所有逻辑集群的控制组信息。该函数在调用的时候需要指定要查询逻辑集群的名称。例如要查询'installation'逻辑集群的控制组信息：

```
SELECT * FROM gs_all_nodegroup_control_group_info('installation');
```

返回值类型：record

函数返回字段如下：

名称	类型	描述
name	text	控制组的名称。
type	text	控制组的类型。
gid	bigint	控制组ID。
classgid	bigint	Workload所属Class的控制组ID。
class	text	Class控制组。
workload	text	Workload控制组。
shares	bigint	控制组分配的CPU资源配额。
limits	bigint	控制组分配的CPU资源限额。
wdlevel	bigint	Workload控制组层级。
cpucores	text	控制组使用的CPU核的信息。

gs_get_nodegroup_tablecount(name)

描述：得到一个逻辑集群中所有数据库包含的用户表数目。

返回值类型：integer

pgxc_max_datanode_size(name)

描述：得到一个逻辑集群的所有DN节点中数据库文件占用磁盘空间的最大值，单位为字节。

版权所有 © 华为云计算技术有限公司

返回值类型：bigint

gs_check_logic_cluster_consistency()

描述：检查当前系统中所有逻辑集群是否存在系统信息不一致的情况，如果返回空记录，表示不存在不一致情况；否则，逻辑集群中CN和DN上的NodeGroup信息存在不一致。该函数应该在非扩缩容重分布时调用。

返回值类型：record

gs_check_tables_distribution()

描述： 检查当前系统中用户表的分布是否存在不一致，如果返回空记录，表示不存在不一致。该函数应该在非扩缩容重分布时调用。

返回值类型：record

pg_stat_bad_block(text, int, int, int, int, int, timestamp with time zone, timestamp with time zone)

描述：获取当前节点自启动后，读取出现Page/CU的损坏信息。

返回值类型：record

pgxc_stat_bad_block(text, int, int, int, int, int, timestamp with time zone, timestamp with time zone)

描述：获取集群所有节点自启动后，读取出现Page/CU的损坏信息。

返回值类型：record

pg_stat_bad_block_clear()

描述：清理节点记录的读取出现的Page/CU损坏信息（需要系统管理员权限）。

返回值类型：void

pgxc_stat_bad_block_clear()

描述：清理集群所有节点记录的读取出现的Page/CU损坏信息（需要系统管理员权限）。

返回值类型：void

gs_respool_exception_info(pool text)

描述：查看某个资源池关联的查询规则信息。

返回值类型：record

gs_control_group_info(pool text)

描述：查看资源池关联的控制组信息

返回值类型：record

 版权所有 © 华为云计算技术有限公司

返回信息如下：

属性	属性值	描述
name	class_a:workload_a1	class和workload名称。
class	class_a	Class控制组名称。
workload	workload_a1	Workload控制组名称。
type	DEFWD	控制组类型（Top、CLASS、BAKWD、DEFWD、TSWD）。
gid	87	控制组ID。
shares	30	占父节点CPU资源的百分比。
limits	0	占父节点CPU核数的百分比。
rate	0	Timeshare中的分配比例。
cpucores	0-3	CPU核心数。

gs_wlm_user_resource_info(name text)

描述：查询具体某个用户的资源限额和资源使用情况。

返回值类型：record

pgxc_stat_single_table(schema, tablename)

描述：在CN上执行，入参为schema和表名。查询单张表在全库中的统计信息及该表在每个DN上的脏页率。

该函数仅8.1.3及以上集群版本支持。

说明

该函数的统计信息依赖于ANALYZE，为获取该表最准确的信息请先对表进行ANALYZE。

返回值类型：record

返回值字段与函数**pg_stat_get_tuple()**相同。

```
SELECT * FROM pgxc_stat_single_table('public','t1');
 nodename  | tableid | partid |      last_vacuum       |    last_autovacuum     |        last_analyze         |
last_autoanalyze     | vacuum_count | autovacuum_count | analyze_count | autoanalyze_count | n_tup_ins | n_
tup_upd | n_tup_del | n_tup_hot_upd | n_tup_change | n_live_tup | n_dead_tup | dirty_rate |
last_data_changed
-----------+---------+--------+------------------------+------------------------+-----------------------------
+------------------------+--------------+------------------+---------------+-------------------+-----------+---
--------+-----------+---------------+--------------+------------+------------+------------+------------------
 datanode1 | 1270075 |        | 2000-01-01 08:00:00+08 | 2000-01-01 08:00:00+08 | 2023-01-09
09:38:43.220876+08 | 2000-01-01 08:00:00+08 |            0 |                0 |             1 |                 0 |         0 |
       0 |         0 |             0 |            0 |          0 |          0 |          0 |
(1 row)
```

 版权所有 © 华为云计算技术有限公司

pgxc_stat_wal_write()

描述：在所有CN上执行，记录各个DN上与CN交互的线程信息，并统计wal日志和数据页导入量及速率。该函数仅8.2.0及以上集群版本支持。

返回值类型：bigint

返回值字段：

- node_name：实例名。
- application_name：应用名。
- query_start：当前正在执行语句的起始时间。
- datapage_write：当前query导入的数据页总量，单位为bytes。
- datapage_write_speed：数据页导入速率，单位为bytes/s。
- wal_write：当前query导入的wal日志总量，单位为bytes。
- wal_write_speed：wal导入速率，单位为bytes/s。
- total_datapage_write：当前线程产生数据页总量，单位为bytes。
- total_wal_write：wal总量，单位为bytes。

```
SELECT * FROM pgxc_stat_wal_write();
 node_name | datid |       pid       | lwtid | usesysid |    application_name    | state  |          query_start
|         backend_start         |     query_id      | datapage_write | datapage_write_speed | wal_write |
wal_write_speed | total_datapage_write | total_wal_write
-----------+-------+-----------------+-------+----------+------------------------+--------+-------------------------------
+-------------------------------+-------------------+----------------+----------------------+-----------+-----------------
+----------------------+-----------------
 datanode1 | 15979 | 140328473442048 | 12408 |       10 | CalculateSpaceInfo     | active | 2022-11-15
10:39:45.00219+08  | 2022-11-15 10:39:45.000918+08 |                0 |              0 |                   0 |     70697
|              0 |                   0 |              0
 datanode1 |     0 | 140328075503360 | 12510 |       10 | WalSender to Secondary | idle
|                           | 2022-11-15 10:39:46.708557+08 |                0 |              0 |                     |       210
|              0 |                   0 |              0
 datanode1 | 15979 | 140327896741632 | 13612 |       10 | coordinator1           | active | 2022-11-15
10:46:32.832548+08 | 2022-11-15 10:40:20.117516+08 | 72620543991349940 |      586579968 |
17425000 |  11743056 |          348000 |          5337505792 |       107245825
 datanode1 | 15979 | 140327583217408 | 13614 |       10 | coordinator1           | active | 2022-11-15
10:46:32.832548+08 |                               | 72620543991349940 |              0 |                   0 |         0
|              0 |                   0 |              0
 datanode1 | 15979 | 140327485175552 | 27914 |       10 | coordinator1           | active | 2022-11-15
10:47:06.493584+08 | 2022-11-15 10:47:06.489062+08 | 72620543991350020 |              0 |                   0
|         0 |              0 |                   0 |           8675
(5 rows)
```

说明

行存不带索引批量导入时，数据页copy导入会产生logical newpage相关的xlog日志，在xlog量大于默认值时同样会触发流控。

pg_stat_wal_write()

描述：在CN或者DN上执行，记录当前实例的线程信息，并统计wal日志和数据页导入量及速率，这里以CN为例。

该函数仅8.2.0及以上集群版本支持。

返回值类型：bigint

返回值字段：

- node_name：实例名。

版权所有 © 华为云计算技术有限公司

- application_name：应用名。
- query_start：当前正在执行语句的起始时间。
- datapage_write：当前query导入的数据页总量，单位为bytes。
- datapage_write_speed：数据页导入速率，单位为bytes/s。
- wal_write：当前query导入的wal日志总量，单位为bytes。
- wal_write_speed：wal导入速率，单位为bytes/s。
- total_datapage_write：当前线程产生数据页总量，单位为bytes。
- total_wal_write：wal总量，单位为bytes。

```
SELECT * FROM pg_stat_wal_write();
  node_name   | datid |       pid       | lwtid | usesysid |   application_name   | state  |          query_start
|         backend_start         |     query_id      | datapage_write | datapage_write_speed | wal_write |
wal_write_speed | total_datapage_write | total_wal_write
--------------+-------+-----------------+-------+----------+----------------------+--------+-------------------------------
+-------------------------------+-------------------+----------------+----------------------+-----------+-----------------
+----------------------+-----------------
 coordinator1 | 15979 | 140234153498368 | 25353 |       10 | JobScheduler       | active |                               |
2022-11-15 11:57:54.396347+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 | 15979 | 140234132027136 | 25354 |       10 | StatCollector      | idle   |
|                               |                 0 |              0 |                      |         0 |             0 |                  0
|             0
 coordinator1 | 15979 | 140234090084096 | 25356 |       10 | WDRSnapshot        | idle   | 2022-11-15
11:57:54.388948+08 | 2022-11-15 11:57:54.403055+08 |                 0 |              0 |                  0 |         0
|             0 |                 0 |              0
 coordinator1 | 15979 | 140234027169536 | 25359 |       10 | workload           | active | 2022-11-15
11:57:54.404836+08 | 2022-11-15 11:57:54.389259+08 | 72620543991404472 |              0 |                  0 |
1722112 |              0 |                 0 |              0
 coordinator1 | 15979 | 140234004621056 | 25360 |       10 | CalculateSpaceInfo | active | 2022-11-15
11:57:54.405677+08 | 2022-11-15 11:57:54.389847+08 | 72620543991349266 |              0 |                  0
|    191326 |              0 |                 0 |              0
 coordinator1 | 15979 | 140233987839744 | 25361 |       10 | WorkloadMonitor    | active | 2022-11-15
11:57:54.406263+08 | 2022-11-15 11:57:54.390373+08 | 72620543991404418 |              0 |                  0 |
1183741 |              0 |                 0 |              0
 coordinator1 | 15979 | 140233971058432 | 25362 |       10 | WLMArbiter         | active | 2022-11-15
11:57:54.406367+08 | 2022-11-15 11:57:54.390947+08 |                 0 |              0 |                  0 |         0
|             0 |                 0 |              0
 coordinator1 | 15979 | 140233228666624 | 14415 |       10 | gsql               | active | 2022-11-15
21:19:32.200305+08 | 2022-11-15 21:09:38.916931+08 | 72620543991404422 |              0 |                  0
|     30872 |           1000 |                 0 |              0
 coordinator1 | 15979 | 140233172035328 | 15516 |       10 | gsql               | active | 2022-11-15
21:19:49.7877+08   | 2022-11-15 21:10:13.447312+08 | 72620543991404485 |              0 |                  0
|         0 |              0 |                 0 |              0
 coordinator1 |     0 | 140234245269248 | 25348 |        0 | Background writer  | idle   |                               |
2022-11-15 11:57:54.383651+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234291410688 | 25347 |        0 | CheckPointer       | idle   |                               |
2022-11-15 11:57:54.383231+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234228487936 | 25349 |        0 | Wal Writer         | idle   |                               |
2022-11-15 11:57:54.384069+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234068584192 | 25357 |        0 | TwoPhase Cleaner   | idle   |                               |
2022-11-15 11:57:54.388332+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234048657152 | 25358 |        0 | LWLock Monitor     | idle   |                               |
2022-11-15 11:57:54.389239+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234194925312 | 25351 |        0 | CBM Writer         | idle   |                               |
2022-11-15 11:57:54.38506+08  |                 0 |              0 |                      |         0 |             0
|                    0 |             0
 coordinator1 |     0 | 140234211706624 | 25350 |        0 | Tick service       | idle   |                               |
2022-11-15 11:57:54.384136+08 |                 0 |              0 |                      |         0 |             0
|                    0 |             0
(16 rows)
```


版权所有 © 华为云计算技术有限公司

说明

行存不带索引批量导入时，数据页copy导入会产生logical newpage相关的xlog日志，在xlog量大于默认值时同样会触发流控。

gs_stack()

描述：获取CN或DN进程的堆栈信息，函数入参tid均需要通过视图进行获取，不支持lwtid。该函数仅8.2.1及以上集群版本支持。

gs_stack()提供了比gdb、gstack等命令更灵活的、更快速的堆栈获得手段。仅建议管理员用户在问题分析定位过程中使用，不建议作为日常的监控工具使用。

返回值字段：

- tid：线程id。
- lwtid：轻量级线程id。
- stack：tid或lwtid线程对应的堆栈信息。

gs_stack()函数有四种使用方式：

- 方式一：gs_stack(tid)函数入参值为0，表示打印gsql所连接的CN或DN进程内所有线程的堆栈。

  说明

  - 可通过gsql连接CN或DN执行。
  - 获取CN或DN进程内所有线程的运行堆栈，运行时间长，对正常业务运行有影响，不建议频繁使用。

```
SELECT * FROM gs_stack(0);
      tid       | lwtid |
stack
-----------------+-------
+-----------------------------------------------------------------------------------------------------------------------------
---
 140186894757888 | 95275 | __poll +
0x2d                                                                                                                        +
            |     | ServerLoop() +
0x457                                                                                                                         +
            |     | PmStartupThreads() +
0x151                                                                                                                       +
            |     | PostmasterMain(int, char**) +
0x22a                                                                                                                     +
            |     | main +
0x250                                                                                                                       +
            |     | __libc_start_main +
0xf5                                                                                                                      +
            |     |
0xb8d2d7                                                                                                                        +
            |     |
 140185903825152 | 95316 | __poll +
0x2d                                                                                                                        +
            |     | GsWaiter::wait(void*, long) +
0x11a                                                                                                                     +
            |     | GsWaiter::wait(void*, int, int, long) +
0x7e                                                                                                                    +
            |     | SysLoggerMain() +
0x109                                                                                                                     +
            |     | SubPostmasterMain(tag_gs_thread_args*) +
0xc9f                                                                                                                   +
            |     | MainStarterThreadFunc(void*) +
0x6a                                                                                                                  +
            |     | ThreadStarterFunc(void*) +
```


版权所有 © 华为云计算技术有限公司

```
0x66                                                          +
          |      | start_thread +
0xc5                                                              +
          |      | clone +
0x6d                                                                  +
          |      |
 140185903825528 | 95317 | __poll +
0x2d                                                                  +
          |      | GsWaiter::wait(void*, long) +
0x11a                                                         +
          |      | AlarmCheckerMain() +
0x21b                                                             +
          |      | SubPostmasterMain(tag_gs_thread_args*) +
0xcc6                                                         +
          |      | MainStarterThreadFunc(void*) +
0x6a                                                      +
          |      | ThreadStarterFunc(void*) +
0x66                                                          +
          |      | start_thread +
0xc5                                                              +
          |      | clone +
0x6d                                                                  +
          |      |
 140185903825904 | 95320 | __poll +
0x2d                                                                  +
          |      | GsWaiter::wait(void*, long) +
0x11a                                                         +
          |      | ReaperBackendMain() +
0x1ee                                                             +
          |      | SubPostmasterMain(tag_gs_thread_args*) +
0xd6f                                                         +
          |      | MainStarterThreadFunc(void*) +
0x6a                                                      +
          |      | ThreadStarterFunc(void*) +
0x66                                                          +
          |      | start_thread +
0xc5                                                              +
          |      | clone +
0x6d                                                                  +
          |      |
 140185903826656 | 95324 | __poll +
0x2d                                                                  +
          |      | GsWaiter::wait(void*, long) +
0x11a                                                         +
          |      | CheckpointerMain() +
0x8bb                                                             +
          |      | AuxiliaryThreadUnderPm(AuxProcType) +
0x210                                                         +
          |      | SubPostmasterMain(tag_gs_thread_args*) +
0x7e0                                                         +
          |      | MainStarterThreadFunc(void*) +
0x6a                                                      +
          |      | ThreadStarterFunc(void*) +
0x66                                                          +
          |      | start_thread +
0xc5                                                              +
          |      | clone +
0x6d                                                                  +
          |      |
 140185903827032 | 95325 | __poll +
0x2d                                                                  +
          |      | GsWaiter::wait(void*, long) +
0x11a                                                         +
          |      | BackgroundWriterMain() +
0x541                                                             +
          |      | AuxiliaryThreadUnderPm(AuxProcType) +
0x210                                                         +
          |      | SubPostmasterMain(tag_gs_thread_args*) +
0x7e0                                                         +
```


版权所有 © 华为云计算技术有限公司

```
             |      | MainStarterThreadFunc(void*) +
0x6a                                                          +
             |      | ThreadStarterFunc(void*) +
0x66                                                            +
```

- 方式二：gs_stack(tid)函数入参值为指定的tid，表示打印gsql所连接的CN或DN进程内tid线程的堆栈。

  说明

  - 可通过gsql连接CN或DN执行。
  - tid必须为所连CN或DN进程上存在的线程tid，否则会报错“invalid thread id”。
  - tid需通过其他视图获取，不支持lwtid。
  - 获取指定CN或DN进程内指定线程的运行堆栈，运行时间短，对正常业务运行无影响，推荐使用。

```
SELECT * FROM gs_stack(140185903826656);
      tid       | lwtid |                      stack
----------------+-------+-------------------------------------------------
 140185903826656 | 95275 | __poll + 0x2d                                  +
             |      | GsWaiter::wait(void*, long) + 0x11a           +
             |      | CheckpointerMain() + 0x8bb                   +
             |      | AuxiliaryThreadUnderPm(AuxProcType) + 0x210   +
             |      | SubPostmasterMain(tag_gs_thread_args*) + 0x7e0+
             |      | MainStarterThreadFunc(void*) + 0x6a           +
             |      | ThreadStarterFunc(void*) + 0x66              +
             |      | start_thread + 0xc5                         +
             |      | clone + 0x6d                               +
             |      |
(1 row)
```

- 方式三：gs_stack('nodename', 0)函数第一个参数值为nodename，第二个参数值为0，表示打印由nodename指定的进程内所有线程的堆栈。

  说明

  - 底层实现使用了“execute direct on”，因此必须使用gsql连接CN上执行。
  - 第一个参数nodename需用单引号包括。
  - 获取指定CN或DN进程内所有线程的运行堆栈，运行时间长，对正常业务运行有影响，不建议频繁使用。

```
SELECT * FROM gs_stack('datanode2', 0);
      tid       | lwtid |
stack
----------------+-------
+-----------------------------------------------------------------------------------------------------------------------------------------------
---
 140634541242112 | 95442 | do_futex_wait +
0x5f                                                                  +
             |      | __new_sem_wait_slow +
0x57                                                                +
             |      | sem_timedwait +
0x35                                                                 +
             |      | tagBinarySemaphore::timed_wait(int) +
0x4f                                                               +
             |      | AuxiliaryThreadMain(void*) +
0x6d                                                                +
             |      | LibcommThreadRoutine(void*) +
0x41                                                               +
             |      | start_thread +
0xc5                                                                +
             |      | clone +
0x6d                                                                    +
             |      |
 140634549634816 | 95438 | epoll_wait +
0x33                                                                  +
             |      | LibcommEpollWait(int, int) +
```


版权所有 © 华为云计算技术有限公司

```
0x38                                                    +
          |     | RecvDataThreadMain(void*) +
0x8e                                                     +
          |     | LibcommThreadRoutine(void*) +
0x41                                                    +
          |     | start_thread +
0xc5                                                          +
          |     | clone +
0x6d                                                              +
          |     |
 140634558027520 | 95432 | epoll_wait +
0x33                                                            +
          |     | LibcommEpollWait(int, int) +
0x38                                                     +
          |     | RecvDataThreadMain(void*) +
0x8e                                                     +
          |     | LibcommThreadRoutine(void*) +
0x41                                                    +
          |     | start_thread +
0xc5                                                          +
          |     | clone +
0x6d                                                              +
          |     |
 140634566420224 | 95430 | epoll_wait +
0x33                                                            +
          |     | LibcommEpollWait(int, int) +
0x38                                                     +
          |     | RecvDataThreadMain(void*) +
0x8e                                                     +
          |     | LibcommThreadRoutine(void*) +
0x41                                                    +
          |     | start_thread +
0xc5                                                          +
          |     | clone +
0x6d                                                              +
          |     |
 140634589476608 | 95426 | epoll_wait +
0x33                                                            +
          |     | LibcommEpollWait(int, int) +
0x38                                                     +
          |     | RecvDataThreadMain(void*) +
0x8e                                                     +
          |     | LibcommThreadRoutine(void*) +
0x41                                                    +
          |     | start_thread +
0xc5                                                          +
          |     | clone +
0x6d                                                              +
          |     |
 140634597869312 | 95424 | do_futex_wait +
0x5f                                                        +
          |     | __new_sem_wait_slow +
0x57                                                     +
          |     | sem_timedwait +
0x35                                                        +
          |     | tagBinarySemaphore::timed_wait(int) +
0x4f                                                    +
          |     | SendDataWait() +
0x58                                                        +
          |     | SendDataThreadMain(void*) +
0xe4                                                     +
          |     | LibcommThreadRoutine(void*) +
0x41                                                    +
          |     | start_thread +
0xc5                                                          +
          |     | clone +
0x6d                                                              +
          |     |
 140634606262016 | 95387 | do_futex_wait +
```


版权所有 © 华为云计算技术有限公司

```
0x5f                                                                +
            |      | __new_sem_wait_slow +
0x57                                                              +
            |      | sem_timedwait +
0x35                                                               +
            |      | tagBinarySemaphore::timed_wait(int) +
0x4f                                                             +
            |      | SendDataWait() +
0x58                                                               +
            |      | SendDataThreadMain(void*) +
0xe4                                                           +
```

- 方式四：gs_stack('nodename', 0)函数第一个参数值为nodename，第二个参数值为tid，表示打印由nodename指定的进程内tid线程的堆栈。

  说明

  - 底层实现使用了“execute direct on”，因此必须gsql连接CN上执行。
  - 第一个参数nodename需用单引号包括。
  - 第二个参数必须是nodename指定进程内存在的线程，否则会报错“invalid thread id”。
  - tid需通过其他视图获取，不支持lwtid。
  - 仅获取指定CN或DN进程内指定线程的运行堆栈，运行时间短，对正常业务运行无影响，推荐使用。

```
SELECT * FROM gs_stack('datanode2', 140634549634816);
      tid       | lwtid |               stack
----------------+-------+-----------------------------------
 140634549634816 | 95438 | epoll_wait + 0x33               +
            |      | LibcommEpollWait(int, int) + 0x38 +
            |      | RecvDataThreadMain(void*) + 0x8e  +
            |      | LibcommThreadRoutine(void*) + 0x41+
            |      | start_thread + 0xc5              +
            |      | clone + 0x6d                    +
            |      |
(1 row)
```

pgxc_get_xlog_stats()

描述：在CN上执行，统计各个DN节点自开机到当前不同xlog类型数量。该函数仅8.3.0及以上集群版本支持。

返回值类型：record

返回信息如下：

名称	类型	描述
node_name	text	节点名称。
rmgr_name	text	xlog主类型。
record_info	text	xlog info信息。
record_count	bigint	数量信息。
count_pct	text	数量占比。
record_size	bigint	record大小， 单位byte。
record_pct	text	record大小占比。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
fpi_size	bigint	fpi(full page image)大小，单位byte。
fpi_pct	text	fpi大小占比。
combined_size	bigint	xlog总大小，单位byte。
combined_pct	text	xlog总大小占比。

说明

- 该函数只统计DN节点最近一次启动以来，不同类型的xlog数量与大小。
- 该函数统计的xlog的大小未考虑到其写盘时的对齐问题，与实际xlog段文件大小存在部分误差。

```
SELECT * FROM pgxc_get_xlog_stats();
 node_name |  rmgr_name  |          record_info          | record_count | count_pct | record_size | record_pct |
fpi_size | fpi_pct | combined_size | combined
_pct
-----------+-------------+-------------------------------+--------------+-----------+-------------+------------+----------
+---------+---------------+---------
-----
 datanode1 | XLOG        | checkpoint: shutdown          |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | checkpoint: online            |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | xlog no-op                    |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | nextOid                       |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | xlog switch                   |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | backup end                    |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | parameter change              |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | restore point                 |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | full page writes              |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | xlog recovery end             |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | XLOG        | page hint                     |            1 | 0.12%     |          56 | 0.11%      |     8192 |
17.92%  |          8248 | 8.71%
 datanode1 | XLOG        | xlog fpi                      |            0 | 0.00%     |           0 | 0.00%      |        0 | 0.00%
|             0 | 0.00%
 datanode1 | Transaction | commit                        |            3 | 0.35%     |         246 | 0.50%      |        0 |
0.00%   |           246 | 0.26%
 datanode1 | Transaction | prepare                       |            1 | 0.12%     |         469 | 0.96%      |        0 |
0.00%   |           469 | 0.50%
 datanode1 | Transaction | abort                         |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | Transaction | commit prepared               |            1 | 0.12%     |         106 | 0.22%      |        0 |
0.00%   |           106 | 0.11%
 datanode1 | Transaction | abort prepared                |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | Transaction | xid assignment xtop           |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | Transaction | commit compact                |            0 | 0.00%     |           0 | 0.00%      |        0 |
0.00%   |             0 | 0.00%
 datanode1 | Storage     | file create                   |            0 | 0.00%     |           0 | 0.00%      |        0 | 0.00%
|             0 | 0.00%
```


版权所有 © 华为云计算技术有限公司

```
 datanode1 | Storage     | file truncate               |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Storage     | vacuum clear file           |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | CLOG        | zero page                   |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | CLOG        | truncate before             |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Database    | create db                   |          1 | 0.12%     |         50 | 0.10%      |       0 |
0.00%   |          50 | 0.05%
 datanode1 | Database    | drop db                     |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Tablespace  | create tablespace           |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Tablespace  | drop tablespace             |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Tablespace  | relative create             |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | MultiXact   | zero offsets page           |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | MultiXact   | zero members page           |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | MultiXact   | create multixact            |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | RelMap      | update relmap               |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Standby     | AccessExclusive locks       |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Standby     | xlog running xacts          |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Standby     | release AccessExclusive locks |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Standby     | xlog standby csn            |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | freeze                      |          0 | 0.00%     |          0 | 0.00%      |       0 | 0.00%
|           0 | 0.00%
 datanode1 | Heap2       | clean                       |          0 | 0.00%     |          0 | 0.00%      |       0 | 0.00%
|           0 | 0.00%
 datanode1 | Heap2       | page upgrade                |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | cleanup info                |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | visible                     |          0 | 0.00%     |          0 | 0.00%      |       0 | 0.00%
|           0 | 0.00%
 datanode1 | Heap2       | multi-insert                |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | bcm                         |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | logical newpage rel         |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | no repair page              |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap2       | multi-insert(init) base xid |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap        | insert                      |        280 | 32.67%    |      15964 | 32.63%     |   14314 |
31.31%  |       30278 | 31.99%
 datanode1 | Heap        | delete                      |          0 | 0.00%     |          0 | 0.00%      |       0 | 0.00%
|           0 | 0.00%
 datanode1 | Heap        | update                      |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap        | base shift                  |          0 | 0.00%     |          0 | 0.00%      |       0 | 0.00%
|           0 | 0.00%
 datanode1 | Heap        | hot update                  |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap        | new page                    |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap        | lock type                   |          0 | 0.00%     |          0 | 0.00%      |       0 |
0.00%   |           0 | 0.00%
 datanode1 | Heap        | inplace                     |          2 | 0.23%     |        116 | 0.24%      |    8164 |
```


版权所有 © 华为云计算技术有限公司

```
17.86%  |          8280 | 8.75%
 datanode1 | Heap         | init page                  |          2 | 0.23%     |        130 | 0.27%      |       92 |
0.20%   |          222 | 0.23%
 datanode1 | Heap         | insert(init) base xid      |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | insert leaf                |        562 | 65.58%    |      31484 | 64.35%     |    13352 |
29.21%  |        44836 | 47.37%
 datanode1 | Btree        | insert upper               |          1 | 0.12%     |         72 | 0.15%      |      148 |
0.32%   |          220 | 0.23%
 datanode1 | Btree        | insert meta                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | split left                 |          1 | 0.12%     |         78 | 0.16%      |      744 | 1.63%
|          822 | 0.87%
 datanode1 | Btree        | split right                |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Btree        | split left root            |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Btree        | split right root           |          1 | 0.12%     |         78 | 0.16%      |      672 |
1.47%   |          750 | 0.79%
 datanode1 | Btree        | delete                     |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Btree        | delete page                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | delete page meta           |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | lev                        |          1 | 0.12%     |         78 | 0.16%      |       40 | 0.09%
|          118 | 0.12%
 datanode1 | Btree        | delete page half           |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | vacuum                     |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Btree        | reuse page                 |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Create index               |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Create ptree               |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Insert item                |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Gin          | Page split                 |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Gin          | Vacuum page                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Delete page                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Update metapage            |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Insert new list page       |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Delete list pages          |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gin          | Vacuum data leaf page      |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gist         | page update                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gist         | page split                 |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | Gist         | Create index               |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Gist         | page delete                |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | Sequence     | log                        |          0 | 0.00%     |          0 | 0.00%      |        0 | 0.00%
|            0 | 0.00%
 datanode1 | SPGist       | create index               |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | SPGist       | add leaf to page           |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
 datanode1 | SPGist       | move leafs                 |          0 | 0.00%     |          0 | 0.00%      |        0 |
0.00%   |            0 | 0.00%
```

 版权所有 © 华为云计算技术有限公司

```
 datanode1 | SPGist      | add node                 |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | SPGist      | prefix off               |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
 datanode1 | SPGist      | picksplit                |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
 datanode1 | SPGist      | vacuum leaf              |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | SPGist      | vacuum root              |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | SPGist      | newest XID               |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | Slot        | create slot              |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
 datanode1 | Slot        | advance slot             |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | Slot        | drop slot                |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
 datanode1 | Slot        | check slot               |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
 datanode1 | Heap3       | new cid                  |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | Heap3       | heap rewrite             |          0 | 0.00%    |          0 | 0.00%      |       0 |
0.00%   |            0 | 0.00%
 datanode1 | Barrier     | barrier                  |          0 | 0.00%    |          0 | 0.00%      |       0 | 0.00%
|            0 | 0.00%
(102 rows)
```

pgxc_get_wal_speed()

描述：在CN上执行，获取每一个DN实例的wal生成速率以及备DN receive、write、flush、redo速率。该函数仅8.3.0及以上集群版本支持。

返回值类型：record

返回信息如下：

名称	类型	描述
node_name	text	节点名称。
pid	bigint	线程PID号。
local_role	text	当前节点role。
peer_role	text	接收方role。
peer_state	text	接收方状态。
state	text	传输状态。
generate_speed	bigint	主DNwal生成速率，单位为byte/s。
receive_speed	bigint	备DN接收速率，单位为byte/s。
write_speed	bigint	备DN写速率，单位为byte/s。
flush_speed	bigint	备DN刷盘速率，单位为byte/s。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
redo_speed	bigint	备DN redo速率，单位为byte/s。

```
SELECT * FROM pgxc_get_wal_speed();
node_name |       pid       | local_role | peer_role | peer_state |   state    | generate_speed | receive_speed |
write_speed | flush_speed | apply_speed
-----------+-----------------+------------+-----------+------------+-----------+----------------+---------------
+-------------+-------------+-------------
 datanode1 | 140240389260824 | Primary    | Secondary | Normal     | Streaming |              0 |             0
|           0 |           0 |           0
 datanode1 | 140240389262192 | Primary    | Standby   | Normal     | Streaming |              0 |             0
|           0 |           0 |           0
(2 rows)
```

pg_xlog_display_one_lsn(start_lsn)

描述：可以在CN或者DN上执行，根据起始LSN号完整解析当前位置的xlog。该函数仅8.3.0及以上集群版本支持。

入参：start_lsn，表示起始LSN。需要保证输入的起始LSN号是一条xlog的开始处。

返回值类型：record

返回信息如下：

名称	类型	描述
node_name	text	当前实例名。
start_lsn	text	设定的起始LSN。
end_lsn	text	设定的终止LSN。
startlsn	text	xlog起始LSN。
endlsn	text	xlog终止LSN。
prelsn	text	前一条xlog起始LSN。
xid	xid	xlog事务ID号。
datalen	int4	xlog数据长度，单位为byte。
totallen	int4	xlog长度，单位为byte。
type	text	xlog类型。
desc	text	xlog内容。
blkref	text	xlog关联的relfilenode。

示例：

```
SELECT * FROM pg_xlog_display_one_lsn('0/101CA00');
node_name | startlsn  |  endlsn   |  prelsn   | xlog_tid | datalen | totallen |  xlogtype   |
```


版权所有 © 华为云计算技术有限公司

```
xlogdescribe
         | blkref
-----------+-----------+-----------+-----------+----------+---------+----------+-------------
+-------------------------------------------------------------------------
----------+--------
datanode1 | 0/101CA00 | 0/101CA78 | 0/101C3F8 |        5 |      80 |      114 | Transaction | commit:
2023-10-19 22:21:38.617092+08; csn:0; inval msgs: catcache 11 ca
tcache 10 |
(1 row)
```

pg_xlogdump (tablename)

描述：在CN或者DN上执行，根据表名解析xlog文件并过滤。该函数仅8.3.0及以上集群版本支持。

返回值类型：record

返回信息如下：

名称	类型	描述
node_name	text	节点名称。
start_lsn	text	设定的起始LSN。
end_lsn	text	设定的终止LSN。
startlsn	text	xlog起始LSN。
endlsn	text	xlog终止LSN。
prelsn	text	前一条xlog起始LSN。
xid	xid	xlog事务ID号。
datalen	int4	xlog数据长度，单位为byte。
totallen	int4	xlog长度，单位为byte。
type	text	xlog类型。
desc	text	xlog内容。
blkref	text	xlog关联的relfilenode。

说明

- 由于不同xlog类型xlogdescribe字段的长度不一致，pg_xlogdump()函数会对该字段进行裁剪，仅保留前64个字节。
- 可以找到目标xlog后结合pg_xlog_display_one_lsn()函数查看完整xlog内容。

示例：

```
SELECT * FROM pg_xlogdump(pg_class);
 node_name | startlsn  |  endlsn   |  prelsn   | xlog_tid | datalen | totallen | xlogtype |  xlogdescribe
|                    blkref
-----------+-----------+-----------+-----------+----------+---------+----------+----------+----------------
+-------------------------------------------------
```


版权所有 © 华为云计算技术有限公司

```
 datanode1 | 0/2DFC660 | 0/2DFE1D0 | 0/2DFA610 |   0 |   2 |  7000 | Heap  | inplace: off 17 |
blkrel #0: rel 1663/16324/15920, fork main, blk 2
 datanode1 | 0/2E02270 | 0/2E02D88 | 0/2E00220 |   0 |   2 |  2840 | Heap  | inplace: off 2  | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E06E60 | 0/2E06F68 | 0/2E06E28 |   0 |   2 |   264 | Heap  | inplace: off 10 | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E0FC08 | 0/2E0FD10 | 0/2E0EBB8 |   0 |   2 |   264 | Heap  | inplace: off 13 | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E12960 | 0/2E12A68 | 0/2E11960 |   0 |   2 |   264 | Heap  | inplace: off 15 | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E12C88 | 0/2E12D90 | 0/2E12B78 |   0 |   2 |   264 | Heap  | inplace: off 17 | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E13FF8 | 0/2E14118 | 0/2E13110 |   0 |   2 |   264 | Heap  | inplace: off 18 | blkrel
#0: rel 1663/16324/15920, fork main, blk 0
 datanode1 | 0/2E169F8 | 0/2E185E0 | 0/2E169B0 |   0 |   2 |  7120 | Heap  | inplace: off 1  | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
 datanode1 | 0/2E188B8 | 0/2E189D0 | 0/2E185E0 |   0 |   2 |   280 | Heap  | inplace: off 2  | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
 datanode1 | 0/2E18BF8 | 0/2E18CE0 | 0/2E189D0 |   0 |   2 |   232 | Heap  | inplace: off 28 | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
 datanode1 | 0/2E18E80 | 0/2E18F88 | 0/2E18DC0 |   0 |   2 |   264 | Heap  | inplace: off 4  | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
 datanode1 | 0/2E19160 | 0/2E19268 | 0/2E19088 |   0 |   2 |   264 | Heap  | inplace: off 5  | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
 datanode1 | 0/2E19460 | 0/2E1B1D0 | 0/2E19388 |   0 |   2 |  7508 | Heap  | inplace: off 16 | blkrel
#0: rel 1663/16324/15920, fork main, blk 6
 datanode1 | 0/2E1B1D0 | 0/2E1B2E8 | 0/2E19460 |   0 |   2 |   280 | Heap  | inplace: off 18 | blkrel
#0: rel 1663/16324/15920, fork main, blk 6
 datanode1 | 0/2E1B2E8 | 0/2E1B3D0 | 0/2E1B1D0 |   0 |   2 |   232 | Heap  | inplace: off 30 | blkrel
#0: rel 1663/16324/15920, fork main, blk 1
(15 rows)
```

pg_xlogdump (xid)

描述：可以在CN或者DN上执行，根据事务id号解析xlog文件并过滤。该函数仅8.3.0及以上集群版本支持。

返回值类型：record

返回信息如下：

名称	类型	描述
node_name	text	节点名称。
start_lsn	text	设定的起始LSN。
end_lsn	text	设定的终止LSN。
startlsn	text	xlog起始LSN。
endlsn	text	xlog终止LSN。
prelsn	text	前一条xlog起始LSN。
xid	xid	xlog事务ID号。
datalen	int4	xlog数据长度，单位为byte。
totallen	int4	xlog长度，单位为byte。
type	text	xlog类型。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
desc	text	xlog内容。
blkref	text	xlog关联的relfilenode。

说明

- 由于不同xlog类型xlogdescribe字段的长度不一致，pg_xlogdump()函数会对该字段进行裁剪，仅保留前64个字节。
- 可以找到目标xlog后结合pg_xlog_display_one_lsn()函数查看完整xlog内容。

示例：

```
SELECT * FROM pg_xlogdump(1);
 node_name | startlsn  |  endlsn   |  prelsn   | xlog_tid | datalen | totallen | xlogtype |  xlogdescribe
|                 blkref
-----------+-----------+-----------+-----------+----------+---------+----------+----------+----------------
+-----------------------------------------------
 datanode1 | 0/10000A0 | 0/10020F0 | 0/1000028 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/1255, fork fsm, blk 2
 datanode1 | 0/10020F0 | 0/1004140 | 0/10000A0 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/1247, fork fsm, blk 2
 datanode1 | 0/1004140 | 0/1004168 | 0/10020F0 |        1 |       4 |       38 | XLOG     | nextOid: 18192 |
 datanode1 | 0/1004168 | 0/10061B8 | 0/1004140 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2616, fork fsm, blk 2
 datanode1 | 0/10061B8 | 0/1008208 | 0/1004168 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2602, fork fsm, blk 2
 datanode1 | 0/1008208 | 0/100A258 | 0/10061B8 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2603, fork fsm, blk 2
 datanode1 | 0/100A258 | 0/100C2A8 | 0/1008208 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2609, fork fsm, blk 2
 datanode1 | 0/100C2A8 | 0/100E2F8 | 0/100A258 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2605, fork fsm, blk 2
 datanode1 | 0/100E2F8 | 0/1010348 | 0/100C2A8 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/1259, fork fsm, blk 2
 datanode1 | 0/1010348 | 0/1012398 | 0/100E2F8 |        1 |       0 |     8248 | XLOG     | page hint      | blkrel
#0: rel 1663/1/2610, fork fsm, blk 2
(10 rows)
```

pg_xlogdump (start_lsn, end_lsn)

描述：可以在CN或者DN上执行，根据起始和终止lsn解析xlog文件。该函数仅8.3.0及以上集群版本支持。

入参：start_lsn，表示设定的起始LSN号；end_lsn，表示设定的终止LSN号。无需保证起始LSN号是一条xlog的起始位置，会从起始LSN号后第一条有效的xlog开始解析。

返回值类型：record

返回信息如下：

字段名称	类型	含义
node_name	text	节点名称。
start_lsn	text	设定的起始LSN。
end_lsn	text	设定的终止LSN。

版权所有 © 华为云计算技术有限公司

字段名称	类型	含义
startlsn	text	xlog起始LSN。
endlsn	text	xlog终止LSN。
prelsn	text	前一条xlog起始LSN。
xid	xid	xlog事务ID号。
datalen	int4	xlog数据长度，单位为byte。
totallen	int4	xlog长度，单位为byte。
type	text	xlog类型。
desc	text	xlog内容。
blkref	text	xlog关联的relfilenode。

说明

- 由于不同xlog类型xlogdescribe字段的长度不一致，pg_xlogdump()函数会对该字段进行裁剪，仅保留前64个字节。
- 可以找到目标xlog后结合pg_xlog_display_one_lsn()函数查看完整xlog内容。

示例：

```
SELECT * FROM pg_xlogdump('0/101EF70','0/101F590');
 node_name | startlsn  |  endlsn   |  prelsn   | xlog_tid | datalen | totallen | xlogtype |     xlogdescribe
|                 blkref
-----------+-----------+-----------+-----------+----------+---------+----------+----------+-----------------------
+-----------------------------------------------
 datanode1 | 0/101EF70 | 0/101EFC0 | 0/101EF18 |        6 |       2 |       80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101EFC0 | 0/101F018 | 0/101EF70 |        6 |       3 |       88 | Heap     | insert: off 2 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
 datanode1 | 0/101F018 | 0/101F068 | 0/101EFC0 |        6 |       2 |       80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F068 | 0/101F0B8 | 0/101F018 |        6 |       2 |       80 | Btree    | insert leaf: off 2    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101F0B8 | 0/101F110 | 0/101F068 |        6 |       3 |       88 | Heap     | insert: off 3 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
 datanode1 | 0/101F110 | 0/101F160 | 0/101F0B8 |        6 |       2 |       80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F160 | 0/101F1B0 | 0/101F110 |        6 |       2 |       80 | Btree    | insert leaf: off 3    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101F1B0 | 0/101F208 | 0/101F160 |        6 |       3 |       88 | Heap     | insert: off 4 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
 datanode1 | 0/101F208 | 0/101F258 | 0/101F1B0 |        6 |       2 |       80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F258 | 0/101F2A8 | 0/101F208 |        6 |       2 |       80 | Btree    | insert leaf: off 4    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101F2A8 | 0/101F300 | 0/101F258 |        6 |       3 |       88 | Heap     | insert: off 5 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
 datanode1 | 0/101F300 | 0/101F350 | 0/101F2A8 |        6 |       2 |       80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F350 | 0/101F3A0 | 0/101F300 |        6 |       2 |       80 | Btree    | insert leaf: off 5    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101F3A0 | 0/101F3F8 | 0/101F350 |        6 |       3 |       88 | Heap     | insert: off 6 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
```


版权所有 © 华为云计算技术有限公司

```
 datanode1 | 0/101F3F8 | 0/101F448 | 0/101F3A0 |      6 |      2 |     80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F448 | 0/101F498 | 0/101F3F8 |      6 |      2 |     80 | Btree    | insert leaf: off 5    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
 datanode1 | 0/101F498 | 0/101F4F0 | 0/101F448 |      6 |      3 |     88 | Heap     | insert: off 7 flags 0 |
blkrel #0: rel 1663/1/2608, fork main, blk 0
 datanode1 | 0/101F4F0 | 0/101F540 | 0/101F498 |      6 |      2 |     80 | Btree    | insert leaf: off 1    |
blkrel #0: rel 1663/1/2673, fork main, blk 1
 datanode1 | 0/101F540 | 0/101F590 | 0/101F4F0 |      6 |      2 |     80 | Btree    | insert leaf: off 6    |
blkrel #0: rel 1663/1/2674, fork main, blk 1
(19 rows)
```

21.7.32 触发器函数

pg_get_triggerdef(oid)

描述：获取触发器的定义信息。

参数：待查触发器的OID。

返回值类型：text

示例：

```
SELECT pg_get_triggerdef(oid) FROM pg_trigger;
                                                pg_get_triggerdef
-----------------------------------------------------------------------------------------------------------------------
 CREATE TRIGGER insert_trigger BEFORE INSERT ON test_trigger_src_tbl FOR EACH ROW EXECUTE
PROCEDURE tri_insert_func()
(1 row)
```

pg_get_triggerdef(oid, boolean)

描述：获取触发器的定义信息。

参数：待查触发器的OID及是否以pretty方式展示。

返回值类型：text

说明

仅在创建trigger时指定WHEN条件的情况下，布尔类型参数才生效。

示例：

```
SELECT pg_get_triggerdef(oid,true) FROM pg_trigger;
                                                pg_get_triggerdef
-----------------------------------------------------------------------------------------------------------------------
 CREATE TRIGGER insert_trigger BEFORE INSERT ON test_trigger_src_tbl FOR EACH ROW EXECUTE
PROCEDURE tri_insert_func()
(1 row)

SELECT pg_get_triggerdef(oid,false) FROM pg_trigger;
                                                pg_get_triggerdef
-----------------------------------------------------------------------------------------------------------------------
 CREATE TRIGGER insert_trigger BEFORE INSERT ON test_trigger_src_tbl FOR EACH ROW EXECUTE
PROCEDURE tri_insert_func()
(1 row)
```

21.7.33 XML 函数

 版权所有 © 华为云计算技术有限公司

21.7.33.1 产生 XML 内容

本节函数和类函数的表达式可以用来从SQL数据产生XML内容。适用于将查询结果格式化成XML文档以便于在客户端应用中处理。

XMLPARSE ({ DOCUMENT | CONTENT } *value*)

描述：从字符数据中生成一个XML类型的值。

返回值类型：xml

示例：

```
SELECT xmlparse(document '<foo>bar</foo>');
   xmlparse
----------------
 <foo>bar</foo>
(1 row)
```

XMLSERIALIZE ({ DOCUMENT | CONTENT } *value* AS *type*

描述：从XML类型的值生成一个字符串。

返回值类型：type，可以是character，character varying或text（或其别名）

示例：

```
SELECT xmlserialize(content 'good' AS CHAR(10));
 xmlserialize
--------------
 good
(1 row)
```

xmlcomment(text)

描述：用于创建一个XML值，它含有一个以指定文本作为内容的XML注释。该文本中不能包含“--”或以一个“-”结尾，这样的文本才是有效的XML注释。当参数为空时，结果也为空。

返回值类型：xml

示例：

```
SELECT xmlcomment('hello');
  xmlcomment
--------------
 <!--hello-->
(1 row)
```

xmlconcat(xml[, ...])

描述：用于将XML值组成的列表串接成一个单独的值。其中空值会被忽略，当所有参数都为空时，结果都为空。

返回值类型：xml

示例：

```
SELECT xmlconcat('<abc/>', '<bar>foo</bar>');
      xmlconcat
----------------------
 <abc/><bar>foo</bar>
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明：XML声明如果存在，结果如下：如果所有参数值都使用相同的XML版本声明，则在结果中使用版本，否则不用版本。如果所有参数值都有standalone属性，且值都为yes， 则结果中standalone值为yes，如果至少有一个是no，则结果中standalone值为no。否则结果中将不带standalone属性。

示例：

```
SELECT xmlconcat('<?xml version="1.1"?><foo/>', '<?xml version="1.1" standalone="no"?><bar/>');
           xmlconcat
-----------------------------------
 <?xml version="1.1"?><foo/><bar/>
(1 row)
```

xmlelement(name name [, xmlattributes(value [AS attname] [, ...])] [, content, ...])

描述：使用给定名称、属性和内容产生一个XML元素。

返回值类型：xml

示例：

```
SELECT xmlelement(name foo);
 xmlelement
------------
 <foo/>
(1 row)

SELECT xmlelement(name foo, xmlattributes('xyz' as bar));
   xmlelement
------------------
 <foo bar="xyz"/>
(1 row)

SELECT xmlelement(name foo, xmlattributes(current_date as bar), 'cont', 'ent');
            xmlelement
--------------------------------------
 <foo bar="2023-08-16">content</foo>
(1 row)
```

没有合法XML名称的元素和属性名会被转义，转义的方式是将不合法的字符用序列_xHHHH_替换，其中HHHH是该字符以16进制表达的Unicode代码点。例如：

```
SELECT xmlelement(name "foo$bar", xmlattributes('xyz' as "a&b"));

           xmlelement
-----------------------------------
 <foo_x0024_bar a_x0026_b="xyz"/>
```

如果属性值是一个列引用，则不需要指定显式的属性名，这种情况下默认将把列名用作该属性的名字。在其他情况下，必须给该属性一个显式的命名。因此这个例子是合法的：

```
CREATE TABLE test (a xml, b xml);
SELECT xmlelement(name test, xmlattributes(a, b)) FROM test;
```

但这些就不合法：

```
SELECT xmlelement(name test, xmlattributes('constant'), a, b) FROM test;
SELECT xmlelement(name test, xmlattributes(func(a, b))) FROM test;
```

元素内容（如果指定）将被根据其数据类型进行格式化。如果内容本身是类型xml，则会构建复杂的XML文档。例如：

```
SELECT xmlelement(name foo, xmlattributes('xyz' as bar),xmlelement(name
abc),xmlcomment('test'),xmlelement(name xyz));
```

 版权所有 © 华为云计算技术有限公司

```
                xmlelement
-----------------------------------------------
 <foo bar="xyz"><abc/><!--test--><xyz/></foo>
```

其他类型的内容将被格式化为合法的XML字符数据。这意味着特别的字符<、>以及&将会被转换成实体。二进制数据（数据类型bytea）将被表示为base64或者十六进制编码，具体取决于配置参数xmlbinary。

xmlforest(content [AS name] [, ...])

描述：使用给定名称和内容产生一个元素的XML森林（序列）。

返回值类型：xml

示例：

```
SELECT xmlforest('abc' AS foo, 123 AS bar);
         xmlforest
------------------------------
 <foo>abc</foo><bar>123</bar>
(1 row)

SELECT xmlforest(table_name, column_name) FROM ALL_TAB_COLUMNS WHERE schema = 'pg_catalog';
                                            xmlforest
-----------------------------------------------------------------------------------------------------------------------------
 <table_name>pg_authid</table_name><column_name>rolsuper</column_name>
 <table_name>pg_authid</table_name><column_name>rolinherit</column_name>
 <table_name>pg_authid</table_name><column_name>rolcreaterole</column_name>
```

xmlpi(name target [, content])

描述：创建一个XML处理指令。内容不能包含字符序列?>

返回值类型：xml

示例：

```
SELECT xmlpi(name php, 'echo "hello world";');
            xmlpi
------------------------------
 <?php echo "hello world";?>
(1 row)
```

xmlroot(xml, version text | no value [, standalone yes|no|no value])

描述：修改一个XML值的根节点的属性。如果指定了一个版本，它会替换根节点的版本声明中的值；如果指定了一个standalone值，它会替换根节点standalone中的值。

返回值类型：xml

示例：

```
SELECT xmlroot(xmlparse(document '<?xml version="1.0" standalone="no"?><content>abc</content>'),
version '1.1', standalone yes);
                           xmlroot
----------------------------------------------------------------
 <?xml version="1.1" standalone="yes"?><content>abc</content>
(1 row)
```

xmlagg(xml)

描述：函数xmlagg是一个聚集函数，将输入值串接起来。

返回值类型：xml

 版权所有 © 华为云计算技术有限公司

示例：

```
CREATE TABLE test (y int, x xml);
INSERT INTO test VALUES (1, '<foo>abc</foo>');
INSERT INTO test VALUES (2, '<bar/>');

SELECT xmlagg(x) FROM test;
       xmlagg
----------------------
 <bar/><foo>abc</foo>
(1 row)
```

为聚集调用增加一个ORDER BY子句即可决定串接的顺序，例如：

```
SELECT xmlagg(x ORDER BY y DESC) FROM test;
       xmlagg
----------------------
 <bar/><foo>abc</foo>
(1 row)
```

21.7.33.2 XML 谓词

本节的函数用户检查xml值的属性。

xml IS DOCUMENT

描述：如果参数XML值是一个正确的XML文档，则IS DOCUMENT返回真；如果非正确XML文档，则返回假；参数为空时返回空。

返回值类型：bool

```
SELECT '<abc/>' is document;
 ?column?
----------
 t
(1 row)
```

xml IS NOT DOCUMENT

描述：如果参数XML值不是一个正确的XML文档，则IS NOT DOCUMENT返回真；如果是正确XML文档，则返回假；参数为空时返回空。

返回值类型：bool

```
SELECT 'abc' is document;
 ?column?
----------
 f
(1 row)
```

XMLEXISTS(text PASSING [BY REF] xml [BY REF])

描述：如果第一个参数中的XPath表达式返回任何节点，则函数XMLEXISTS返回真，否则返回假（如果哪一个参数为空，则结果就为空）。BY REF子句没有作用，用于保持SQL兼容性。

返回值类型：bool

示例：

```
SELECT xmlexists('//town[text() = ''TScity'']' PASSING BY REF '<towns><town>TScity</town><town>TOcity</
town></towns>');
 xmlexists
```


版权所有 © 华为云计算技术有限公司

```
-----------
 t
(1 row)
```

xml_is_well_formed(text)

描述：检查text字符串是不是格式良好的XML值，返回布尔结果。如果xmloption参数设置为DOCUMENT则检查文档，如果设置为CONTENT则检查内容。

返回值类型：bool

示例：

```
SET xmloption TO DOCUMENT;
SET

SELECT xml_is_well_formed('<>');
 xml_is_well_formed
--------------------
 f
(1 row)

SELECT xml_is_well_formed('<abc/>');
 xml_is_well_formed
--------------------
 t
(1 row)

SET xmloption TO CONTENT;
SELECT xml_is_well_formed('abc');
 xml_is_well_formed
--------------------
 t
(1 row)
```

xml_is_well_formed_document(text)

描述：检查text字符串是不是格式良好的文档，返回布尔结果。

返回值类型：bool

示例：

```
SELECT xml_is_well_formed_document('<test:foo xmlns:test="http://test.com/test">bar</test:foo>');
 xml_is_well_formed
--------------------
 t
(1 row)

SELECT xml_is_well_formed_document('<test:foo xmlns:test="http://test.com/test">bar</my:foo>');
 xml_is_well_formed_document
-----------------------------
 f
(1 row)
```

xml_is_well_formed_content(text)

描述：检查text字符串是不是格式良好的内容，返回布尔结果。

返回值类型：bool

示例：

```
SELECT xml_is_well_formed_content('content');
 xml_is_well_formed_content
```

 版权所有 © 华为云计算技术有限公司

```
----------------------------
 t
(1 row)

SELECT xml_is_well_formed_content('<content');
 xml_is_well_formed_content
----------------------------
 f
(1 row)
```

21.7.33.3 处理 XML

为了处理数据类型XML的值，DWS提供了函数xpath和xpath_exists计算XPath表达式以及XMLTABLE表函数。

xpath(xpath, xml [, nsarray])

描述：它返回一个XML值的数组对应xpath表达式所产生的节点集。如果xpath表达式返回一个标量值而不是节点集，将返回一个单个元素的数组。

该函数的第二个参数xml必须是一个完整的XML文档，它必须有一个根节点元素。

第三个参数为可选参数，是一个命名空间的数组映射。这个数组应该是一个二维text数组，第二个维的长度等于2（它应该是一个数组的数组，其中每一个正好包含2个元素）。每个数组项的第一个元素是命名空间名称的别名，第二个元素是命名空间URI。这个数组中提供的别名不必与XML文档本身中使用的别名相同。换句话说，在XML文档和xpath函数上下文中，别名都是本地的。

返回值类型：xml值的数组

示例：

```
SELECT xpath('/my:a/text()', '<my:a xmlns:my="http://example.com">test</my:a>', ARRAY[ARRAY['my',
'http://example.com']]);
 xpath
--------
 {test}
(1 row)
```

要处理默认（匿名）命名空间：

```
SELECT xpath('//mydefns:b/text()', '<a xmlns="http://example.com"><b>test</b></
a>',ARRAY[ARRAY['mydefns', 'http://example.com']]);
 xpath
--------
 {test}
(1 row)
```

xpath_exists(xpath, xml [, nsarray])

描述：函数xpath_exists是xpath函数的一种特殊形式。这个函数不是返回满足XPath的XML值，它返回一个布尔值表示查询是否被满足。这个函数等价于标准的XMLEXISTS谓词，不过它还提供了对一个名字空间映射参数的支持。

返回值：bool

示例：

```
SELECT xpath_exists('/my:a/text()', '<my:a xmlns:my="http://example.com">test</my:a>',
ARRAY[ARRAY['my', 'http://example.com']]);
 xpath_exists
--------------
```


版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
```

xmltable

描述：xmltable函数根据输入的XML数据、XPath路径表达式、列的定义信息生成一个表。xmltable在语法上类似于一个函数，但它只能以表的形式出现在查询的FROM子句里。

返回值：setof record

语法：

```
XMLTABLE ( [ XMLNAMESPACES ( namespace_uri AS namespace_name [,  ...] ), ]
            row_expression PASSING [ BY  { REF | VALUE } ]
document_expression [ BY  { REF | VALUE } ]
COLUMNS name  { type  [ PATH column_expression  ] [ DEFAULT default_expression ] [ NOT NULL |
NULL ] | FOR ORDINALITY }
[, ...]
)
```

参数说明：

- 可选的XMLNAMESPACES子句是一个逗号分隔的命名空间定义列表，其中namespace_uri都是text类型表达式，namespace_name都是简单标识符。XMLNAMESPACES指定了在文档中使用的XML命名空间及它们的别名。当前不支持默认的命名空间声明。
- 必选的row_expression参数是一个XPath 1.0表达式，该表达式根据提供的XML文档document_expression进行计算获取XML节点序列，该序列是xmltable转换成输出行的顺序。如果document_expression值为NULL，或row_expression产生空的节点集，结果将不返回任何行。
- document_expression参数用于输入XML文档，输入的文档必须是符合XML格式的文档，不支持XML片段数据或格式错误的XML文档。BY REF和BY VALUE子句不起作用，仅用于实现SQL标准兼容性。
- COLUMNS子句指定输出表中字段列表定义，列名和列的数据类型是必选的，路径、默认值和是否为空子句是可选的。
  - 列的column_expression是一个XPath 1.0表达式，用于从row_expression计算出当前行中提取出列的值。如果没有给出column_expression，那么字段名将用作隐式路径。
  - 列可以被标记为NOT NULL，如果NOT NULL列的column_expression不返回任何数据，并且没有DEFAULT子句或者default_expression的计算结果为NULL，就会报错。
  - 标记为FOR ORDINALITY的字段将从1开始填充行号，其顺序为从row_expression结果集中检索到的节点顺序，最多只有一列可以标记为FOR ORDINALITY。

  须知

  XPath 1.0没有为节点指定顺序，因此返回结果的顺序取决于数据获取顺序。

示例：

```
SELECT * FROM XMLTABLE('/ROWS/ROW'
PASSING '<ROWS><ROW id="1"><CITY_ID>1002a</CITY_ID><CITY_NAME>snowcity</CITY_NAME></
ROW><ROW id="2"><CITY_ID>1003b</CITY_ID><CITY_NAME>icecity</CITY_NAME></ROW><ROW
```

 版权所有 © 华为云计算技术有限公司

```
id="3"><CITY_ID>1004c</CITY_ID><CITY_NAME>windcity</CITY_NAME></ROW></ROWS>'
COLUMNS id INT PATH '@id',
ordinality FOR ORDINALITY,
CITY_id TEXT PATH 'CITY_ID',CITY_name TEXT PATH 'CITY_NAME' NOT NULL);
 id | ordinality | city_id | city_name
----+------------+---------+-----------
  1 |          1 | 1002a   | snowcity
  2 |          2 | 1003b   | icecity
  3 |          3 | 1004c   | windcity
(3 rows)
```

21.7.33.4 将表映射到 XML

table_to_xml(tbl regclass, nulls boolean, tableforest boolean, targetns text)

描述：把表的内容映射成XML值。

返回值类型：xml

table_to_xmlschema(tbl regclass, nulls boolean, tableforest boolean, targetns text)

描述：把关系表的模式映射成XML模式文档。

返回值类型：xml

table_to_xml_and_xmlschema(tbl regclass, nulls boolean, tableforest boolean, targetns text)

描述：把关系表映射成XML值和模式文档。

返回值类型：xml

query_to_xml(query text, nulls boolean, tableforest boolean, targetns text)

描述：把SQL查询的内容映射成XML值。

返回值类型：xml

query_to_xmlschema(query text, nulls boolean, tableforest boolean, targetns text)

描述：把SQL查询映射成XML模式文档。

返回值类型：xml

query_to_xml_and_xmlschema(query text, nulls boolean, tableforest boolean, targetns text)

描述：把SQL查询映射成XML值和模式文档。

返回值类型：xml

版权所有 © 华为云计算技术有限公司

cursor_to_xml(cursor refcursor, count int, nulls boolean, tableforest boolean, targetns text)

描述：把游标查询映射成XML值。

返回值类型：xml

cursor_to_xmlschema(cursor refcursor, nulls boolean, tableforest boolean, targetns text)

描述：把游标查询映射成XML模式文档。

返回值类型：xml

schema_to_xml(schema name, nulls boolean, tableforest boolean, targetns text)

描述：把模式中的表映射成XML值。

返回值类型：xml

schema_to_xmlschema(schema name, nulls boolean, tableforest boolean, targetns text)

描述：把模式中的表映射成XML模式文档

返回值类型：xml

schema_to_xml_and_xmlschema(schema name, nulls boolean, tableforest boolean, targetns text)

描述：把模式中的表映射成XML值和模式文档。

返回值类型：xml

database_to_xml(nulls boolean, tableforest boolean, targetns text)

描述：把数据库的表映射成XML值。

返回值类型：xml

database_to_xmlschema(nulls boolean, tableforest boolean, targetns text)

描述：把数据库的表映射成XML模式文档

返回值类型：xml

database_to_xml_and_xmlschema(nulls boolean, tableforest boolean, targetns text)

描述：把数据库的表映射成XML值和模式文档。

返回值类型：xml

 版权所有 © 华为云计算技术有限公司

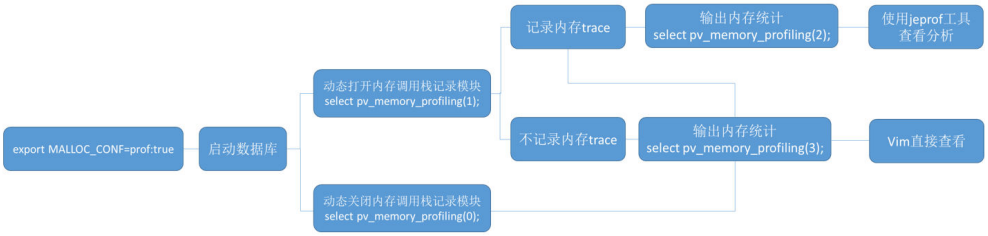
说明

将表映射到XML有关函数的参数说明如下：

- tbl：表名。
- nulls：在输出中是否包含空值，若为true，列中的空值表示为：<columnname xsi:nil="true"/>，若为false，包含空值的列会从输出中省略。
- tableforest：若为true，则输出xml片段，false，输出xml文档。
- targetns：指定想要结果的XML命名空间。若不指定，应传递一个空字符串。
- query：SQL查询语句。
- cursor：游标名。
- count：从游标中获取的数据量。
- schema：模式名

21.7.34 调用栈记录函数

DWS通过pv_memory_profiling(type int)和环境变量MALLOC_CONF，控制gaussdb进程中malloc等内存分配调用栈记录模块开启关闭、以及进程级内存调用栈输出等，使用流程如下图所示：



MALLOC_CONF

环境变量MALLOC_CONF用于控制监控模块是否开启，位于${BIGDATA_HOME}/mppdb/.mppdbgs_profile文件内，默认开启。需注意：

- 修改此环境变量变化，需要重启数据库。
- 如果在集群中启用了om_monitor，完成环境变量设置后，先重启om_monitor进程后，然后重启数据库，使得开关生效。
- 该环境变量可以设置在集群所有服务器中，也可以仅设置在需要开启模块的个别服务器中，对GaussDB进程而言，各进程是根据各自MALLOC_CONF环境变量，控制模块是否打开。

MALLOC_CONF开启和关闭命令：

- 开启监控模块：
  ```
  export MALLOC_CONF=prof:true
  ```
- 关闭监控模块：
  ```
  export MALLOC_CONF=prof:false
  ```

pv_memory_profiling (type int)

参数说明：控制gaussdb进程中malloc等内存分配函数的backtrace记录及输出。

取值范围：正整数，0~3

版权所有 © 华为云计算技术有限公司

表 21-84 pv_memory_profiling 取值含义

pv_memory_profiling 取值	含义
0	关闭内存trace，不记录malloc等调用栈信息。
1	开启内存trace，记录malloc等调用栈信息。
2	输出malloc等调用栈的trace日志。 ● 输出路径为：/proc/pid/cwd指向的目录，pid为gaussdb进程ID。 ● 输出日志命名规则为：jeprof.<pid>.*.heap，其中pid表示GaussDB进程ID，*表示trace日志输出的序号，序号唯一。如：jeprof.195473.0.u0.heap
3	输出内存统计信息。 ● 输出路径为：/proc/pid/cwd指向的目录，pid为gaussdb进程ID。 ● 命名规则为节点名称+进程id+时间+“heap_stats”+.out。该文件可用vim直接打开。

返回值类型：boolean

备注：

- 如果成功，函数返回true，否则返回false，返回相关提示信息。
- 如果失败提示信息为“Memory profiling failed, check if $MALLOC_CONF contain 'prof:true'.”，说明未设置MALLOC_CONF=prof:true的情况下，使用了本模块，需要设置环境变量即可解决。
- 如果失败提示信息为“Type %d is not supported. The valid range is 0-3.”，说明用户输入参数错误，正确数值为0,1,2,3。
- 如果失败提示信息为“Memory profiling failed, input type is %d, failed number is %d.”请联系技术支持工程师提供技术支持。

输出内存调用栈信息

操作步骤：

步骤1 输出内存调用栈信息，在GaussDB进程所在目录输出trace文件：

```
SELECT * FROM pv_memory_profiling(2);
```

步骤2 使用jemalloc中提供的jeprof工具，解析日志信息：

方式1：以text格式输出。

```
jeprof --text --show_bytes $GAUSSHOME/bin/gaussdb trace文件1 >prof.txt
```

方式2：以pdf格式输出。

```
jeprof --pdf --show_bytes $GAUSSHOME/bin/gaussdb trace文件1 > prof.pdf
```


版权所有 © 华为云计算技术有限公司

说明

- 解析内存调用栈信息，需要依靠GaussDB源码进行分析，需要将trace文件返回给研发工程师，进行分析。
- 分析trace文件，需要使用jeprof工具，该工具由Jemalloc生成。在常规使用中，需要依赖perl环境，如果需要生成pdf调用图，需要安装与操作系统匹配的GraphViz工具。

----结束

示例

系统管理员用户登录，设置环境变量，启动数据库：

```
export MALLOC_CONF=prof:true
```

在数据库运行期间，关闭内存trace记录功能：

```
SELECT pv_memory_profiling(0);
pv_memory_profiling
----------------------------
t
(1 row)
```

在数据库运行期间，打开内存trace记录功能：

```
SELECT pv_memory_profiling(1);
pv_memory_profiling
----------------------------
t
(1 row)
```

输出内存trace记录功能：

```
SELECT pv_memory_profiling(2);
pv_memory_profiling
----------------------------
t
```

(1 row)

在GaussDB进程所在目录输出trace文件，以text格式或PDF格式输出：

```
jeprof --text --show_bytes $GAUSSHOME/bin/gaussdb trace文件1 >prof.txt
jeprof --pdf --show_bytes $GAUSSHOME/bin/gaussdb trace文件1 > prof.pdf
```

输出内存统计信息。执行以下命令，在GaussDB进程所在目录输出内存统计文件，可直接读取：

```
SELECT pv_memory_profiling(3);
pv_memory_profiling
----------------------------
t
(1 row)
```

21.7.35 Hudi 系统函数

Hudi系统函数仅9.1.0.100及以上集群版本支持。

pg_show_custom_settings()

描述：查询Hudi外表参数设置详情。

返回值类型：setof record

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT * FROM pg_show_custom_settings();
                        name                         |      setting       | unit |      category      |        short_desc        |
extra_desc | context | vartype | source  | min_val | max_val |
 enumvals | boot_val | reset_val | sourcefile | sourceline
-----------------------------------------------------+--------------------+------+--------------------
+--------------------------+------------+---------+---------+---------+---------+---------+
----------+----------+-----------+------------+------------
 hoodie.public.hudi_mor_ft.consume.ending.timestamp | 20230404172329544 |      | Customized Options |
GUC placeholder variable |            | user    | string  | session |         |         |
          |          |           |            |
 hoodie.public.hudi_mor_ft.consume.mode             | incremental       |      | Customized Options | GUC
placeholder variable |            | user    | string  | session |         |         |
          |          |           |            |
 hoodie.public.hudi_mor_ft.consume.start.timestamp  | 20230404172329543 |      | Customized Options |
GUC placeholder variable |            | user    | string  | session |         |         |
          |          |           |            |
(3 rows)
```

hudi_get_options(regclass)

描述：查询Hudi外表的属性信息（hoodie.properties）。以key-value键值对表示。

返回值类型：setof record

示例：

```
select * from hudi_get_options('public.hudi_mor_ft');
                        key                         |
                                                
```

hudi_get_max_commit(regclass)

描述：获取当前Hudi外表最新commit的时间戳和数据写入时间。

返回值类型：record

示例：

```
SELECT * FROM hudi_get_max_commit('public.hudi_mor_ft');
   max_commit   |        write_time
----------------+------------------------
 20221207141822 | 2022-12-07 14:18:30+08
(1 row)
```

hudi_get_commit(regclass, cstring, int)

描述：获取当前Hudi外表从指定commit开始，到第N个commit的时间戳和对应commit数据写入时间。当第N个commit不存在，则返回最新的commit和对应数据写入时间。

返回值类型：record

示例：

```
SELECT * FROM hudi_get_commit('public.hudi_mor_ft', '20230329174744657', 3);
    end_commit     |        write_time
-------------------+------------------------
 20230329174808908 | 2023-08-31 15:43:08+08
(1 row)
```

hudi_sync_task_submit(regclass, regclass)

描述：提交Hudi自动同步任务。第一个入参为同步目标表，第二个入参为HUDI外表。提交任务成功会返回任务的task-id。

返回值类型：text

说明

- 同步目标表必须包含主键，且主键需要与hudi recordkey一致。
- 如果hudi表包含precombine字段，那么同步目标表也必须包含与之对应的字段。
- 如果同步目标表只包含主键（除了主键外没有其他字段），则无法正常提交同步任务。
- 用户需要有同步目标表的insert和update权限、HUDI外表的select权限，否则无法正常提交同步任务。

示例：

```
SELECT hudi_sync_task_submit('public.hudi_sync_i','public.hudi_mor_ft');
       hudi_sync_task_submit
--------------------------------------
 6465efe2-3ea1-0b00-dde5-b57dfb30fffe
(1 row)
```

hudi_sync_task_submit(regclass, regclass, interval)

描述：功能与hudi_sync_task_submit(regclass, regclass)一致。不同之处在于，用户可以额外指定一个interval类型入参，用于指定任务的调度周期，取值范围为5秒至24小时。提交任务成功会返回任务的task-id。

返回值类型：text

 版权所有 © 华为云计算技术有限公司

说明

- 同步目标表必须包含主键，且主键需要与hudi recordkey一致。
- 如果hudi表包含precombine字段，那么同步目标表也必须包含与之对应的字段。
- 如果同步目标表只包含主键（除了主键外没有其他字段），则无法正常提交同步任务。
- 用户需要有同步目标表的insert和update权限、HUDI外表的select权限，否则无法正常提交同步任务。

示例：

```
SELECT hudi_sync_task_submit('public.hudi_sync_i','public.hudi_mor_ft','1 hour');
       hudi_sync_task_submit
--------------------------------------
 6465efe2-3ea1-0b00-dde5-b57dfb30fffe
(1 row)
```

hudi_sync_task_submit(regclass, regclass, text, text)

描述：功能与hudi_sync_task_submit(regclass, regclass)一致，不同之处在于，用户可以额外指定两个text入参，表示用户期望哪些字段被同步。字段之间使用','分隔，支持引号和转义字符的解析。两个text参数的字段数量和顺序应当一致，表示同步字段之间的对应关系。提交任务成功会返回任务的task-id。

返回值类型：text

说明

- 同步目标表必须包含主键，且主键需要与hudi recordkey一致。
- 如果hudi表包含precombine字段，那么同步目标表也必须包含与之对应的字段。
- 如果同步目标表只包含主键（除了主键外没有其他字段），则无法正常提交同步任务。
- 用户需要有同步目标表的insert和update权限、HUDI外表的select权限，否则无法正常提交同步任务。

示例：

```
SELECT hudi_sync_task_submit('public.hudi_sync_i','public.hudi_mor_ft','_hoodie_commit_time, col_bigint,
col_text', '_hoodie_commit_time, col_bigint, col_text');
       hudi_sync_task_submit
--------------------------------------
 646610bc-cdd1-0d00-d07d-b57e89a0fffe
(1 row)
```

hudi_sync_task_submit(regclass, regclass, text, text, interval)

描述：功能与hudi_sync_task_submit(regclass, regclass, text, text)一致，不同之处在于，用户可以额外指定一个interval类型入参，用于指定任务的调度周期，取值范围为5秒至24小时。

返回值类型：text

说明

- 同步目标表必须包含主键，且主键需要与hudi recordkey一致。
- 如果hudi表包含precombine字段，那么同步目标表也必须包含与之对应的字段。
- 如果同步目标表只包含主键（除了主键外没有其他字段），则无法正常提交同步任务。
- 用户需要有同步目标表的insert和update权限、HUDI外表的select权限，否则无法正常提交同步任务。

版权所有 © 华为云计算技术有限公司

示例：

```
SELECT hudi_sync_task_submit('public.hudi_sync_i','public.hudi_mor_ft','_hoodie_commit_time, col_bigint,
col_text', '_hoodie_commit_time, col_bigint, col_text', '10 minute 30second');
       hudi_sync_task_submit
--------------------------------------
 646610bc-cdd1-0d00-d07d-b57e89a0fffe
(1 row)
```

hudi_show_sync_state()

描述：获取Hudi自动同步任务的同步状态。

返回值类型：setof record

示例：

```
SELECT * FROM hudi_show_sync_state();
    target_tbl     |    source_ftbl     |                    payload_type                    | precombine_key |
latest_commit
--------------------+--------------------+----------------------------------------------------+----------------
+-------------------
 public.hudi_sync_i | public.hudi_mor_ft | org.apache.hudi.common.model.OverwriteWithLatestAvroPayload |
col_int        | 20230511114021573
(1 row)
```

hudi_sync(regclass, regclass)

描述：存储过程，Hudi自动同步任务调用入口。使用pg_catalog.hudi_sync_task_submit(regclass, regclass)提交的任务会执行该存储过程。执行成功会提示同步的行数和时间戳。

返回值类型：text

示例：

```
CALL hudi_sync('public.hudi_sync_i', 'public.hudi_mor_ft');
NOTICE:  execute full sync
CONTEXT:  PL/pgSQL function hudi_sync(regclass,regclass) line 11 at RETURN
            hudi_sync
--------------------------------------
 sync 1 rows up to 20230511114021573.
(1 row)
```

hudi_sync_custom(regclass, regclass, text)

描述：存储过程，Hudi自动同步任务调用入口，支持用户自定义目标表和数据源表的字段同步对应关系。使用pg_catalog.hudi_sync_task_submit(regclass, regclass, text, text)提交的任务会执行该存储过程。其中text为json风格字符串，表示两张表字段间的同步对应关系。执行成功会提示同步的行数和时间戳。

返回值类型：text

示例：

```
CALL hudi_sync_custom('public.hudi_sync_i', 'public.hudi_mor_ft', '{"_hoodie_commit_time" :
"_hoodie_commit_time", "col_bigint" : "col_bigint", "col_text" : "col_text"}');
NOTICE:  execute full sync
CONTEXT:  PL/pgSQL function hudi_sync_custom(regclass,regclass,text) line 14 at RETURN
          hudi_sync_custom
--------------------------------------
 sync 1 rows up to 20230511114021573.
(1 row)
```


版权所有 © 华为云计算技术有限公司

hudi_set_sync_commit(regclass, regclass, text)

描述：设置Hudi自动同步任务首次同步的起点时间戳，避免在已经同步了部分数据的情况下，重新同步已有数据。第一个参数为同步目标表，第二个参数为hudi外表，第三个参数为用户期望的同步起点。该函数需要在提交同步任务之前使用。

返回值类型：text

示例：

```
select hudi_set_sync_commit('public.hudi_sync_i', 'public.hudi_mor_ft', '20230511114021573');
NOTICE:  set sync commit successfully, the next synchronization will start from 20230511114021573
CONTEXT:  referenced column: hudi_set_sync_commit
 hudi_set_sync_commit
----------------------
 20230511114021573
(1 row)
```

说明

用户需要有同步目标表的insert和update权限、HUDI外表的select权限，否则无法正常设置同步进度。

hudi_set_sync_commit(text, text)

描述：设置Hudi自动同步任务下一次同步的起点时间戳，可以用于重复同步历史数据或者跳过某些数据。第一个参数为任务id，第二个参数为用户期望的下一次同步起点。该函数需要在提交同步任务后使用，并且使用前需要暂停任务。

返回值类型：text

示例：

```
select hudi_set_sync_commit('6524c8e3-aae9-0000-5a14-be8ec000fffe', '20230511114021573');
NOTICE:  set sync commit successfully, the next synchronization will start from 20230511114021573
CONTEXT:  referenced column: hudi_set_sync_commit
 hudi_set_sync_commit
----------------------
 20230511114021573
(1 row)
```

说明

- 只有拥有目标任务权限的用户才能成功调用该函数。
- 调用该函数前，目标任务需要处于暂停状态，且至少成功执行过一次。

pg_task_show(text)

描述：查询当前自动调度任务信息，对于hudi同步任务，入参应该是'SQLonHudi'。

返回值类型：setof record

示例：

```
SELECT * FROM pg_task_show('SQLonHudi');
              task_id                |
what                                                
```

```
 64d257e9-1e9b-0d00-3ce3-7e61b5e0fffe | call pg_catalog.hudi_sync_custom('public.hudi_read_target',
'public.hudi_read101', '{"_hoodie_commit_seqno" : "_hoodie_commit_seqno", "id" : "id", "ts" : "ts",
"long_field" : "ts"}'); | SQLonHudi   |     10 | f         | '00:00:10' |           | 2023-08-08 22:58:15.846903+08
|          |          |         5 | 2023-08-08 22:58:15.846903+08 | 2023-08-08 22:58:24.846903+08 |
|           0
(1 row)
```

说明

last_log和failure_times字段用于记录上一次任务的状态。

- last_log于任务结束时刷新，若任务成功，则清空内容；若任务失败则记录任务的失败日志。
- failure_times于时间窗结束时刷新，若任务成功，置为0，失败则累加1，本轮未拉起则不变，可用于推断第一次失败出现的时间。

pg_task_remove(text)

描述：删除某个自动调度任务，入参为该任务的task-id，函数返回被删除任务的个数。

返回值类型：integer

示例：

```
SELECT pg_task_remove('64661705-8ada-0100-d07f-b57e89a0fffe');
 pg_task_remove
----------------
              1
(1 row)
```

pg_task_pause(text)

描述：暂停某个自动调度任务，入参为该任务的task-id，函数返回被暂停任务的个数。

返回值类型：integer

示例：

```
SELECT pg_task_pause('64661705-8ada-0100-d07f-b57e89a0fffe');
 pg_task_pause
---------------
             1
(1 row)
```

pg_task_resume(text)

描述：恢复某个自动调度任务执行，入参为该任务的task-id。函数返回被恢复执行任务的个数。

返回值类型：integer

示例：

```
SELECT pg_task_resume('64661705-8ada-0100-d07f-b57e89a0fffe');
 pg_task_resume
----------------
              1
(1 row)
```

 版权所有 © 华为云计算技术有限公司

pg_task_reset_interval(text, interval)

描述：修改某个同步任务的调度周期，第一个入参为任务的task_id，第二个入用于指定任务的调度周期，取值范围为5秒至24小时。函数返回被修改周期的任务的个数。

返回值类型：integer

示例：

```
select pg_task_reset_interval('64bfd69c-a016-0000-120e-1e802978fffe', '10 hours 30 minutes');
pg_task_reset_interval
------------------------
1
(1 row)
```

21.7.36 漏斗函数和留存函数

漏斗和留存相关函数仅8.3.0及以上集群版本支持。

背景信息

漏斗函数与**留存函数**都是常见的用户行为分析工具，广泛应用于产品分析和数据分析领域，尤其是在产品经理、数据科学家和市场营销人员的工作中。这些函数的目标是帮助分析用户的行为路径、理解用户的流失和转化问题，以及评估产品的长期吸引力和用户忠诚度。

- 漏斗分析：一种用于分析用户在执行某一目标过程中所经历的各个步骤的分析方法。通常，漏斗分析可以帮助产品团队或营销人员了解用户在使用产品或服务时的行为轨迹，以及每个步骤中的转化率和流失情况。漏斗分析的关键在于定义一个“漏斗”，即用户从进入某个流程（如访问网站、注册账户、选购商品等）到完成最终目标（如完成支付、完成注册等）的整个路径。
- 留存分析：留存分析关注的是在某一特定时间段后，仍然活跃使用产品的用户比例。具体来说，留存分析计算的是从某一时间点（例如，用户注册、首次下单、首次使用等）开始，经过一段时间后，仍然继续使用产品的用户数量。常见的留存分析函数通常涉及的是“日留存率”、“周留存率”、“月留存率”等，它们帮助分析用户在首次接触后是否会继续使用该产品。

漏斗函数 windowfunnel

函数描述：windowfunnel函数用于在滑动的时间窗口中搜索事件列表并计算条件匹配的事件列表的最大长度。

DWS根据用户定义的事件列表，从第一个事件开始匹配，依次做有序最长的匹配，返回匹配的最大长度。一旦匹配失败，结束整个匹配。具体介绍如下：

假设在窗口足够大的条件下：

- 条件事件为c1，c2，c3，而用户数据为c1，c2，c3，c4，最终匹配到c1，c2，c3，函数返回值为3。
- 条件事件为c1，c2，c3，而用户数据为c4，c3，c2，c1，最终匹配到c1，函数返回值为1。
- 条件事件为c1，c2，c3，而用户数据为c4，c3，最终没有匹配到事件，函数返回值为0。

函数语法

```
windowFunnel(window, mode, timestamp, cond1, cond2, ..., condN)
```


版权所有 © 华为云计算技术有限公司

入参说明

- window：bigint类型。滑动的时间窗口的大小，指从第一个事件开始，往后推移的时间窗口大小，单位为秒。
- mode：text类型。目前仅支持default模式，其他模式报错处理。Default模式是指在同一个窗口期内，从第一个事件开始匹配，尽量匹配多的事件。
- timestamp：事件发生的时间范围，支持timestamp without time zone、timestamp with time zone、date、int、bigint类型。
- cond：变长boolean数组。指当前Tuple的数据满足事件的哪个步骤。DWS只支持1~32个condition，不在此范围报错处理。

返回值

level：int类型。条件匹配的事件列表的最大长度。

留存函数 retention

函数描述：retention函数可以将一组条件作为参数，分析事件是否满足该条件。

函数语法

```
retention(cond1, cond2, ..., cond32);
```

入参说明

cond：变长boolean数组，最大长度32，用来表示事件是否满足特定条件。DWS只支持1~32个condition，不在此范围报错处理。

返回值

retention condition：tinyint数组类型。返回结果的表达式，与入参cond长度一致，若第cond1和condi条件满足，则返回值第i个值为1，否则为0。

留存扩展函数

DWS留存扩展函数支持：range_retention_count和range_retention_sum函数。

- **range_retention_count**

  函数描述：记录每个用户的留存情况，该函数返回数组，可以作为range_retention_sum函数的入参。

  函数语法

  ```
  range_retention_count(is_first, is_active, dt, retention_interval, retention_granularity, output_format)
  ```

  入参说明

  - is_first：bool类型，是否符合初始行为，true表示符合， false表示不符合 。
  - is_active：bool类型，是否符合留存行为，true表示符合， false表示不符合 。
  - dt：date类型，发生行为的日期。
  - retention_interval：数组类型，表示留存间隔，最多支持15个留存间隔。例如ARRAY[1,3,5,7,15,30]。
  - retention_granularity：text类型，表示留存分析粒度，执行日（day）、周（week）、月（month）三种。
  - output_format：text类型，表示输出格式，支持normal（默认）和expand（可取得每日留存明细）两种。

版权所有 © 华为云计算技术有限公司

返回值：用户留存情况BIGINT数组。

- **range_retention_sum**

  函数描述：汇总计算所有用户每天(周/月)的留存率。

  函数语法

  ```
  range_retention_sum(range_retention_count_result)
  ```

  入参说明：range_retention_count函数返回值。

  返回值：用户留存统计情况text数组。

示例

创建表funnel_test：

```
CREATE TABLE IF NOT EXISTS funnel_test
(
    user_id INT ,
    event_type TEXT,
    event_time TIMESTAMP,
    event_timez TIMESTAMP WITH TIME ZONE,
    event_time_int BIGINT
);
```

插入数据：

```
INSERT INTO funnel_test VALUES
(1,'Browse','2021-01-31 11:00:00', '2021-01-31 11:00:00+08', 10),
(1,'Click','2021-01-31 11:10:00', '2021-01-31 11:10:00+07', 20),
(1,'Order','2021-01-31 11:20:00', '2021-01-31 11:20:00+06', 30),
(1,'Pay','2021-01-31 11:30:00', '2021-01-31 11:30:00+05', 40),
(2,'Order','2021-01-31 11:00:00', '2021-01-31 11:00:00+08', 11),
(2,'Pay','2021-01-31 11:10:00', '2021-01-31 11:10:00+08', 12),
(1,'Browse','2021-01-31 11:00:00', '2021-01-31 11:00:00+01', 50),
(3,'Browse','2021-01-31 11:20:00', '2021-01-31 11:20:00-04', 30),
(3,'Click','2021-01-31 12:00:00', '2021-01-31 12:00:00-04', 80),
(4,'Browse','2021-01-31 11:50:00', '2021-01-31 11:50:00-01', 1000),
(4,'Pay','2021-01-31 12:00:00', '2021-01-31 12:00:00-02', 900),
(4,'Order','2021-01-31 12:00:00', '2021-01-31 12:00:00-03', 1001),
(4,'Click','2021-01-31 12:00:00', '2021-01-31 12:00:00-04', 1001),
(5,'Browse','2021-01-31 11:50:00', '2021-01-31 11:50:00+08', NULL),
(5,'Click','2021-01-31 12:00:00', '2021-01-31 12:00:00+08', 776),
(5,'Order','2021-01-31 11:10:00', '2021-01-31 11:10:00+08', 999),
(6,'Browse','2021-01-31 11:50:00', '2021-01-31 11:50:00+01', -1),
(6,'Click','2021-01-31 12:00:00', '2021-01-31 12:00:00+02', -2),
(6,'Order','2021-01-31 12:10:00', '2021-01-31 12:00:00+03', -3);
```

计算每个用户的漏斗情况。返回结果如下，其中level=0表示用户在窗口期内匹配最大事件深度为0，level=1表示用户在窗口期内匹配最大事件深度为1：

```
SELECT
    user_id,
  windowFunnel(
    0, 'default', event_timez,
    event_type = 'Browse', event_type = 'Click', event_type = 'Order', event_type = 'Pay'
  ) AS level
FROM funnel_test
GROUP BY user_id
ORDER by user_id;

 user_id | level
---------+-------
       1 |     1
       2 |     0
       3 |     1
       4 |     1
       5 |     1
```


版权所有 © 华为云计算技术有限公司

```
      6 |     1
(6 rows)
```

计算每个用户的漏斗情况，指定滑动的时间窗口的大小为NULL，返回报错：

```
SELECT
    user_id,
  windowFunnel(
    NULL, 'default', event_time,
    event_type = 'Browse', event_type = 'Click', event_type = 'Order', event_type = 'Pay'
  ) AS level
FROM funnel_test
GROUP BY user_id
ORDER by user_id;
ERROR:  Invalid parameter : window length or mode is null.
```

计算每个用户的漏斗情况，指定多个条件：

```
SELECT
    user_id,
  windowFunnel(
    40, 'default', date(event_time),
    true, true, false, true
  ) AS level
FROM funnel_test
GROUP BY user_id
ORDER by user_id;
 user_id | level
---------+-------
       1 |     2
       2 |     2
       3 |     2
       4 |     2
       5 |     2
       6 |     2
(6 rows)
```

分析用户的留存情况：

```
SELECT
    user_id,
  retention(
    event_type = 'Browse', event_type = 'Click', event_type = 'Order', event_type = 'Pay'
    ) AS r
FROM funnel_test
GROUP BY user_id
ORDER BY user_id ASC;
 user_id |      r
---------+-----------
       1 | {1,1,1,1}
       2 | {0,0,0,0}
       3 | {1,1,0,0}
       4 | {1,1,1,1}
       5 | {1,1,1,0}
       6 | {1,1,1,0}
(6 rows)
```

分析用户的留存情况，指定第一个时间为false：

```
SELECT
    user_id,
  retention(
    false, event_type = 'Browse', event_type = 'Click', event_type = 'Order', event_type = 'Pay'
    ) AS r
FROM funnel_test
GROUP BY user_id
ORDER BY user_id ASC;
 user_id |      r
---------+-------------
       1 | {0,0,0,0,0}
```


版权所有 © 华为云计算技术有限公司

```
    2 | {0,0,0,0,0}
    3 | {0,0,0,0,0}
    4 | {0,0,0,0,0}
    5 | {0,0,0,0,0}
    6 | {0,0,0,0,0}
(6 rows)
```

分析用户的留存情况总和：

```
SELECT sum(r[1]), sum(r[2]), sum(r[3]), sum(r[4])
FROM
(
    SELECT
    retention(event_type = 'Browse', event_type = 'Click', event_type = 'Order', event_type = 'Pay') AS r
    FROM funnel_test
    GROUP BY user_id
);
 sum | sum | sum | sum
-----+-----+-----+-----
   5 |   5 |   4 |   2
(1 row)
```

创建表retention_test：

```
CREATE TABLE retention_test(
uid INT,
event TEXT,
event_time TIMESTAMP
);
```

插入数据：

```
INSERT INTO retention_test VALUES
(1, 'pay', '2024-05-01'),
(1, 'login', '2024-05-01'),
(1, 'pay', '2024-05-02'),
(1, 'login', '2024-05-02'),
(2, 'login', '2024-05-01'),
(3, 'login', '2024-05-02'),
(3, 'pay', '2024-05-03'),
(3, 'pay', '2024-05-04');
```

统计每个用户在1，2天后的付费留存率：

```
WITH retention_count_info AS (
    SELECT
        uid,
        range_retention_count(event = 'login', event = 'pay',
                              DATE(event_time), array[1, 2], 'day') AS info
    FROM
        retention_test
    GROUP BY
        uid
), retention_sum AS (
         SELECT regexp_split_to_array(unnest(range_retention_sum(info)), ',') AS s
         FROM retention_count_info
     ) SELECT to_date(s[1]::int) AS login_date,
              s[3]::numeric / s[2]::numeric AS retention_rate1,
              s[4]::numeric / s[2]::numeric AS retention_rate2
FROM retention_sum
ORDER BY login_date;
     login_date      |    retention_rate1    |    retention_rate2
---------------------+-----------------------+-----------------------
 2024-05-01 00:00:00 | .50000000000000000000 | 0.00000000000000000000
 2024-05-02 00:00:00 | .50000000000000000000 |  .50000000000000000000
(2 rows)
```

21.7.37 EXTERNAL SCHEMA 系统函数

EXTERNAL SCHEMA相关系统函数仅8.3.0及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

pg_get_external_schema_table_options(text, text)

描述：获取external schema表的option。

入参: 第一个入参为external schema名称，第二个入参为表名称。

返回值类型：setof record

示例：

```
SELECT * FROM pg_get_external_schema_table_options('ex_lf', 'test_lf');
 option_name |             option_value
-------------+--------------------------------------
 encoding    | utf8
 format      | parquet
 foldername  | /***/***/***
(3 rows)
```

pg_get_external_schema_table_col(text, text)

描述：获取external schema表的列信息。

入参: 第一个入参为external schema名称，第二个入参为表名称。

返回值类型：setof record

示例：

```
SELECT * FROM pg_get_external_schema_table_col('ex_lf', 'test_lf');
        col_name        |   col_type    | part_col
------------------------+---------------+----------
 field_smallint         | smallint      | f
 field_int              | int           | f
 field_integer          | int           | f
 field_bigint           | bigint        | f
 field_float            | float         | f
 field_double           | double        | f
 field_double_precision | double        | f
 field_decimal          | decimal(10,0) | f
 field_numeric          | decimal(10,0) | f
 field_timestamp        | timestamp     | f
 field_date             | date          | f
 field_varchar          | varchar(5)    | f
 field_char             | char(5)       | f
 field_boolean          | boolean       | f
 field_string           | string        | f
(15 rows)
```

21.7.38 存算分离函数

21.7.38.1 数据共享函数

pgxc_group_add_subscription(src_vw_name, target_vw_name)

描述：存算分离架构下，逻辑集群（Virtual Warehouse，以下简称VW）之间建立KV订阅关系，建立了订阅关系之后，消费者VW的KVcache会定期的从生产者的obs cudesc快照目录增量同步sst文件到本地，后续就可以支持在消费者VM中跨VW查询生产者VW中的表。 创建的订阅关系可以在pgxc_group_subscription表中查询到记录。该函数仅9.0.3及以上集群版本支持。

返回值类型：void

版权所有 © 华为云计算技术有限公司

返回信息如下：

表 21-85 pgxc_group_add_subscription(src_vw_name, target_vw_name)返回字段

名称	类型	描述
src_vw_name	text	生产者VW名称，通常作为写入数据的VW。
target_vw_name	text	消费者VW名称，通常作为读取数据的VW。

示例：

```
SELECT pgxc_group_add_subscription('write_group', 'single_read_group');
pgxc_group_add_subscription
------------------

(1 row)
```

pgxc_group_drop_subscription(src_vw_name, target_vw_name)

描述：存算分离架构下，删除VW之间已经建立的KV订阅关系，删除操作是删除pgxc_group_subscription表的记录， 删除KV订阅关系后，消费者VW无法继续跨VW查询生产者VW中的表。该函数仅9.0.3及以上集群版本支持。

返回值类型：void

返回信息如下：

表 21-86 pgxc_group_drop_subscription(src_vw_name, target_vw_name)返回字段

名称	类型	描述
src_vw_name	text	生产者VW名称，通常作为写入数据的VW。
target_vw_name	text	消费者VW名称，通常作为读取数据的VW。

示例：

```
SELECT  pgxc_group_drop_subscription('write_group', 'single_read_group');
pgxc_group_drop_subscription
------------------------------
(1 row)
```

21.7.38.2 获取 tablespace 垃圾桶文件信息函数

pgxc_tablespace_trash_info()

描述：存算分离架构下获取filesystem为OBS的tablespace在垃圾桶目录下的文件统计信息，包括文件数量、文件总大小以及对应垃圾桶目录是否开启了自动清理生命周期策略。该函数仅9.1.0.222及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

返回值类型：record

函数返回字段如下：

名称	类型	描述
spcname	text	tablespace名称。
file_count	int	垃圾桶目录下tablespace相关文件数量。
total_size	int	垃圾桶目录下tablespace相关文件总大小。
lifecycle	text	是否为垃圾桶目录添加了过期生命周期策略。该字段当前版本中暂未生效。 ● active：已添加。 ● inactive：未添加。

示例：

```
select * from pgxc_tablespace_trash_info();
spcname   | file_count | total_size  | lifecycle
----------+------------+-------------+-----------
cu_obs_tbs|      0     |      0      | inactive
(1 row)
```

21.7.39 时序函数

DWS支持的时序计算函数参见**表21-87**。

表 21-87 时序计算函数列表

时序计算函数	函数描述
delta(field numeric)	计算按照时间排序后两行之间的差值。
spread(field numeric)	计算某段时间内最大和最小值的差值。
mode() within group (order by value anyelement)	对给定的列，返回出现频率最高的值，如果多个值频率相同，返回这些值中最小的那个值。
value_of_percentile(column float, percentile float, compression float)	计算百分位。是percentile_cont的近似算法。
percentile_of_value(column float, percentilevalue float, compression float)	给定百分位，计算对应的值。是value_of_percentile的逆运算。
first(column1, column2)	通过比较分组内column2列的值，找到其中的最小值，输出对应column1列的值。
last(column1, column2)	通过比较分组内column2列的值，找到其中的最大值，输出对应column1列的值。

版权所有 © 华为云计算技术有限公司

delta(field numeric)

描述：计算按照时间排序后两行之间的差值。该函数通常用于时序场景计算按照时间排序后相邻两行差值，用于流量，速度等指标监控。

参数说明：field，需要计算的列。

⚠ 注意

delta是一个窗口函数，需要与over窗口函数使用。且over中rows语句不会改变delta函数结果，比如，delta(value) over(order by time rows 1 preceding) 和 delta(value) over(order by time rows 3 preceding) 返回的结果是一致的。

示例：

```
SELECT
    delta(value) over (rows 1 preceding)
FROM
    (VALUES ('2019-07-12 00:00:00'::timestamptz, 1),
            ('2019-07-12 00:01:00'::timestamptz, 3),
            ('2019-07-12 00:02:00'::timestamptz, 7))
    v(time,value) order by 1;
 delta
-------
    0
    2
    4
(3 rows)
```

spread(field numeric)

描述：计算每个指标在分组内的增量，通常按照时间排序后计算。

参数说明：field，需要计算的列。

⚠ 注意

每个分组内如果少于2个元组，返回结果为0，不要和over窗口函数混用。

示例：

```
SELECT
    time, SPREAD(value)
 FROM
    (VALUES ('2019-07-12 00:00:00'::timestamptz, 1), ('2019-07-12 00:00:00'::timestamptz, 3),
            ('2019-07-12 00:01:00'::timestamptz, 3), ('2019-07-12 00:01:00'::timestamptz, 6),
            ('2019-07-12 00:02:00'::timestamptz, 7), ('2019-07-12 00:02:00'::timestamptz, 9))
    v(time,value) group by time order by time;
         time          | spread
-----------------------+--------
 2019-07-12 00:00:00+08 |      2
 2019-07-12 00:01:00+08 |      3
 2019-07-12 00:02:00+08 |      2
(3 rows)
```

mode() within group (order by value anyelement)

描述：对给定的列，返回出现频率最高的值，如果多个值频率相同，返回这些值中最小的那个值。

版权所有 © 华为云计算技术有限公司

参数说明：value，查询列。

注意

- 该函数需要与within group一起使用，无within group语句，会报错。该函数参数放在group的order by后面。
- 不能和over子句一起使用。

示例：

```
SELECT
    mode() within group (order by value)
 FROM
    (VALUES ('2019-07-12 00:00:00'::timestamptz, 1),
            ('2019-07-12 00:01:00'::timestamptz, 1),
            ('2019-07-12 00:02:00'::timestamptz, 7))
    v(time,value);
 mode
------
    1
(1 row)
```

value_of_percentile(column float, percentile float, compression float)

描述：对于给定的列按照从小到大的顺序返回百分位的近似值。是percentile_cont的近似结果，但是性能比percentile_cont好。

表 21-88 参数说明

参数名	类型	描述
column	float	要计算百分位的列。
percentile	float	指定的百分位的值，取值范围[0，1]。
compression	float	指定的压缩系数，取值范围为[0，500]，默认值是300，这个值取值越大，函数计算过程占用内存越大，同时结果的精度也相对高。如果指定的值不在取值范围内，会按照300求解。

示例：

```
SELECT value_of_percentile(values, 0.8, 0) from TABLE;
```

percentile_of_value(column float, percentilevalue float, compression float)

描述：对于给定的列按照从小到大的顺序返回百分位。是value_of_percentile的逆过程。

表 21-89 参数说明

参数名	类型	描述
column	float	要计算百分位的列。

版权所有 © 华为云计算技术有限公司

参数名	类型	描述
percentilevalue	float	指定该值，用于计算它所在的百分位。
compression	float	指定的压缩系数，取值范围为[0，500]，默认值是300，这个值取值越大，函数计算过程占用内存越大，同时结果的精度也相对高。如果指定的值不在取值范围内，会按照300求解。

示例：

```
SELECT percentile_of_value(values, 80, 0) from TABLE;
```

first(column1, column2)

描述：聚合函数。通过比较分组内column2列的值，找到其中的最小值，输出对应column1列的值。

表 21-90 参数说明

参数名	类型	描述
column1	bigint/text/double/numeric	最终的输出列。
column2	timestamp/timestamptz/numeric	比较列。

示例：样例数据参考**time_fill()**表达式中的表定义和数据。

求按照scope_name分组，每个分组内按照时间排序最靠前的idle的值：

```
select first(idle, time_string) from dcs_cpu group by scope_name;
 first
-------
     1
     3
(2 rows)
```

last(column1, column2)

描述：聚合函数。通过比较分组内column2列的值，找到其中的最大值，输出对应column1列的值。

表 21-91 参数说明

参数名	类型	描述
column1	bigint/text/double/numeric	最终的输出列。

版权所有 © 华为云计算技术有限公司

参数名	类型	描述
column2	timestamp/timestamptz/ numeric	比较列。

示例：样例数据参考**time_fill()**表达式中的表定义和数据。

求按照scope_name分组，每个分组内按照时间排序最靠后的idle的值：

```
select last(idle, time_string) from dcs_cpu group by scope_name;
 last
------
    3
    2
(2 rows)
```

时序填充函数

DWS支持的时序填充函数参见**表21-92**。

表 21-92 时序填充函数列表

时序填充表达式	描述
time_fill(interval, time_column, start_time, end_time), **fill_last**(agg_function(agg_column))	按照时间列排序后，补充缺失的时间数据信息（聚合以后的结果） 用按照时间排序后的**前值填充后值**。
time_fill(interval, time_column, start_time, end_time), **fill_first**(agg_function(agg_column))	按照时间列排序后，补充缺失的时间数据信息（聚合以后的结果） 用按照时间排序后的**后值填充前值**。
time_fill(interval, time_column, start_time, end_time), **fill_avg**(agg_function(agg_column))	按照时间列排序后，补充缺失的时间数据信息（聚合以后的结果） 用按照时间排序后**前后两者的值填充当前值**。

版权所有 © 华为云计算技术有限公司

⚠ 注意

- time_fill函数需要作为聚合函数使用，group by需要引用自身计算结果，不支持与自身嵌套使用，不支持单条查询内多次调用，不支持作为下层计算节点使用，不支持与within group使用。
- time_fill在使用时，必须与fill_avg、fill_first或者fill_last配合使用；或者与Agg函数组合使用。
- time_fill必须出现在group by中，并且group by中只能出现这一列。
- time_fill不支持出现在处理select后面的其他位置，比如where后面或其他关联条件中。
- start时间戳的值必须小于finish时间戳的值， 且两者间距需要大于window_width的值。
- 所有参数不支持空值，start和finish需为确定值。

表 21-93 时序填充表达式的参数说明

参数名	类型	描述
interval	时间间隔类型：INTERVAL，最小单位是1秒。	按照时间分组的时间间隔。
time_column	时间类型，timestamp/timestamptz。	按照指定列做时间分组。
start_time	时间类型，timestamp/timestamptz。	分组的起始时间。
end_time	时间类型，timestamp/timestamptz。	分组的结束时间。
agg_function(agg_column)	指定Agg函数对指定列做聚合。比如max(col)。	对Agg的结果按照指定的填充方法填充。

time_fill(interval, time_column, start_time, end_time), fill_last(agg_function(agg_column))

描述：按照时间列排序后，用前值补充后值。

创建示例表dcs_cpu，并且插入数据。

```
create table dcs_cpu(
idle real,
vcpu_num int,
node text,
scope_name text,
server_ip text,
iowait numeric,
time_string timestamp
)with (orientation=column) distribute by hash(node);
insert  into dcs_cpu VALUES(1.0,1,'node_a','scope_a','1.1.1.1',1.0,'2019-07-12 00:10:10');
insert  into dcs_cpu VALUES(2.0,2,'node_b','scope_a','1.1.1.2',2.0,'2019-07-12 00:12:10');
insert  into dcs_cpu VALUES(3.0,3,'node_c','scope_b','1.1.1.3',3.0,'2019-07-12 00:13:10');
```

以1min为单位，求分组内的均值。用前一时间段的值，填充后一时间段的值。

版权所有 © 华为云计算技术有限公司

```
select time_fill(interval '1 min',time_string,'2019-07-12 00:09:00','2019-07-12 00:14:00'), fill_last(avg(idle))
from dcs_cpu group by time_fill order by time_fill;
     time_fill      | fill_last
--------------------+-----------
 2019-07-12 00:09:00 |
 2019-07-12 00:10:00 |         1
 2019-07-12 00:11:00 |         1
 2019-07-12 00:12:00 |         2
 2019-07-12 00:13:00 |         3
 2019-07-12 00:14:00 |         3
(6 rows)
```

time_fill(interval, time_column, start_time, end_time), fill_first(agg_function(agg_column))

描述：按照时间列排序后，用后值补充前值。

示例：

以1min为单位，求分组内的均值。用后一时间段的值，填充前一时间段的值（样例数据参考**time_fill(interval, time_column, start_time, end_time), fill_last(agg_function(agg_column))**中的表定义和数据）。

```
select time_fill(interval '1 min',time_string,'2019-07-12 00:09:00','2019-07-12 00:14:00'), fill_first(avg(idle))
from dcs_cpu group by time_fill order by time_fill;
     time_fill      | fill_first
--------------------+------------
 2019-07-12 00:09:00 |          1
 2019-07-12 00:10:00 |          1
 2019-07-12 00:11:00 |          2
 2019-07-12 00:12:00 |          2
 2019-07-12 00:13:00 |          3
 2019-07-12 00:14:00 |
(6 rows)
```

time_fill(interval, time_column, start_time, end_time), fill_avg(agg_function(agg_column))

描述：按照时间排序后，前后两者的值填充当前值。

示例：

以1min为单位，求分组内的平均值，用前后两个时间的加权平均值，填充当前值（样例数据参考**time_fill(interval, time_column, start_time, end_time), fill_last(agg_function(agg_column))**中的表定义和数据）。

```
select time_fill(interval '1 min',time_string,'2019-07-12 00:09:00','2019-07-12 00:14:00'), fill_avg(avg(idle))
from dcs_cpu group by time_fill order by time_fill;
     time_fill      | fill_avg
--------------------+----------
 2019-07-12 00:09:00 |        1
 2019-07-12 00:10:00 |        1
 2019-07-12 00:11:00 |      1.5
 2019-07-12 00:12:00 |        2
 2019-07-12 00:13:00 |        3
 2019-07-12 00:14:00 |        3
(6 rows)
```

21.8 表达式

版权所有 © 华为云计算技术有限公司

21.8.1 简单表达式

逻辑表达式

逻辑表达式的操作符和运算规则，请参见**逻辑操作符**。

比较表达式

常用的比较操作符，请参见**比较操作符**。

除比较操作符外，还可以使用以下句式结构：

- BETWEEN操作符
  a BETWEEN x AND y等效于a >= x AND a <= y
  a NOT BETWEEN x AND y等效于a < x OR a > y
- 检查一个值是不是null，可使用：
  expression IS NULL
  expression IS NOT NULL
  或者与之等价的句式结构，但不是标准的：
  expression ISNULL
  expression NOTNULL

须知

不要写expression=NULL或expression<>(!=)NULL，因为NULL代表一个未知的值，不能通过该表达式判断两个未知值是否相等。

示例

```
SELECT 2 BETWEEN 1 AND 3 AS RESULT;
 result
----------
 t
(1 row)

SELECT 2 >= 1 AND 2 <= 3 AS RESULT;
 result
----------
 t
(1 row)

SELECT 2 NOT BETWEEN 1 AND 3 AS RESULT;
 result
----------
 f
(1 row)

SELECT 2 < 1 OR 2 > 3 AS RESULT;
 result
----------
 f
(1 row)

SELECT 2+2 IS NULL AS RESULT;
 result
```


版权所有 © 华为云计算技术有限公司

```
----------
 f
(1 row)

SELECT 2+2 IS NOT NULL AS RESULT;
 result
----------
 t
(1 row)

SELECT 2+2 ISNULL AS RESULT;
 result
----------
 f
(1 row)

SELECT 2+2 NOTNULL AS RESULT;
 result
----------
 t
(1 row)

SELECT 2+2 IS DISTINCT FROM NULL AS RESULT;
 result
----------
 t
(1 row)

SELECT 2+2 IS NOT DISTINCT FROM NULL AS RESULT;
 result
----------
 f
(1 row)
```

21.8.2 条件表达式

在执行SQL语句时，可通过条件表达式筛选出符合条件的数据。

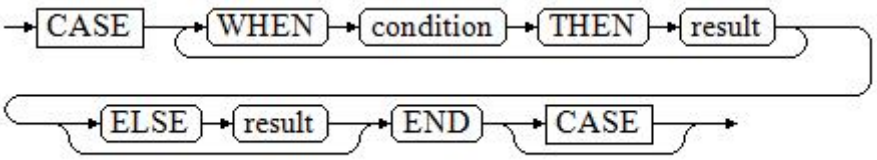
条件表达式主要有以下几种：

- CASE

  CASE表达式是条件表达式，类似于其他编程语言中的CASE语句。

  CASE表达式的语法图请参考图21-3。

  图 21-3 case::=

  CASE子句可以用于合法的表达式中。condition是一个返回BOOLEAN数据类型的表达式：

  - 如果结果为真，CASE表达式的结果就是符合该条件所对应的result。
  - 如果结果为假，则以相同方式处理随后的WHEN或ELSE子句。
  - 如果各WHEN condition都不为真，表达式的结果就是在ELSE子句执行的result。如果省略了ELSE子句且没有匹配的条件，结果为NULL。

版权所有 © 华为云计算技术有限公司

⚠ 注意

- case when嵌套使用时不建议超过3层，解析器在解析时需要将所有涉及字段递归展开，递归过深可能导致内存占用过大。
- 如果case when很复杂可以直接使用子查询的方式替换，参考**子查询表达式**。

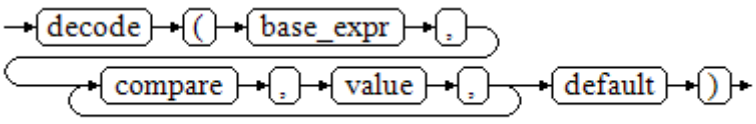
示例：

```
CREATE TABLE tpcds.case_when_t1(CW_COL1 INT)  DISTRIBUTE BY HASH (CW_COL1);

INSERT INTO tpcds.case_when_t1 VALUES (1), (2), (3);

SELECT * FROM tpcds.case_when_t1;
 cw_col1
---------
       3
       1
       2
(3 rows)

SELECT CW_COL1, CASE WHEN CW_COL1=1 THEN 'one' WHEN CW_COL1=2 THEN 'two' ELSE 'other'
END FROM tpcds.case_when_t1;
 cw_col1 | case
---------+-------
       3 | other
       1 | one
       2 | two
(3 rows)

DROP TABLE tpcds.case_when_t1;
```

- DECODE

  DECODE的语法图请参见**图21-4**。

  图 21-4 decode::=

  将表达式base_expr与后面的每个compare(n) 进行比较，如果匹配返回相应的value(n)。如果没有发生匹配，则返回default。

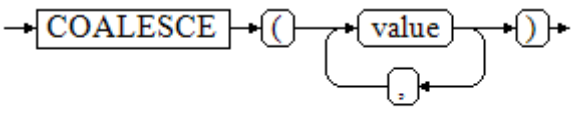
  示例请参见**条件表达式函数**。

```
SELECT DECODE('A','A',1,'B',2,0);
 case
------
    1
(1 row)
```

- COALESCE

  COALESCE的语法图请参见**图21-5**。

  图 21-5 coalesce::=

 版权所有 © 华为云计算技术有限公司

COALESCE返回它的第一个非NULL的参数值。如果参数都为NULL，则返回NULL。它常用于在显示数据时用缺省值替换NULL。和CASE表达式一样，COALESCE只计算用来判断结果的参数，即在第一个非空参数右边的参数不会被计算。

示例：

```
CREATE TABLE tpcds.c_tabl(description varchar(10), short_description varchar(10), last_value
varchar(10))
DISTRIBUTE BY HASH (last_value);

INSERT INTO tpcds.c_tabl VALUES('abc', 'efg', '123');
INSERT INTO tpcds.c_tabl VALUES(NULL, 'efg', '123');

INSERT INTO tpcds.c_tabl VALUES(NULL, NULL, '123');

SELECT description, short_description, last_value, COALESCE(description, short_description, last_value)
FROM tpcds.c_tabl ORDER BY 1, 2, 3, 4;
 description | short_description | last_value | coalesce
-------------+-------------------+------------+----------
 abc         | efg               | 123        | abc
             | efg               | 123        | efg
             |                   | 123        | 123
(3 rows)

DROP TABLE tpcds.c_tabl;
```

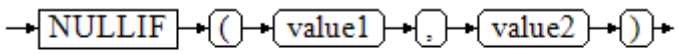
如果description不为NULL，则返回description的值，否则计算下一个参数short_description；如果short_description不为NULL，则返回short_description的值，否则计算下一个参数last_value；如果last_value不为NULL，则返回last_value的值，否则返回（none）。

```
SELECT COALESCE(NULL,'Hello World');
   coalesce
---------------
 Hello World
(1 row)
```

- NULLIF

  NULLIF的语法图请参见**图21-6**。

  图 21-6 nullif::=

  只有当value1和value2相等时，NULLIF才返回NULL。否则它返回value1。

  示例：

```
CREATE TABLE tpcds.null_if_t1 (
    NI_VALUE1 VARCHAR(10),
    NI_VALUE2 VARCHAR(10)
)  DISTRIBUTE BY HASH (NI_VALUE1);

INSERT INTO tpcds.null_if_t1 VALUES('abc', 'abc');
INSERT INTO tpcds.null_if_t1 VALUES('abc', 'efg');

SELECT NI_VALUE1, NI_VALUE2, NULLIF(NI_VALUE1, NI_VALUE2) FROM tpcds.null_if_t1 ORDER BY 1,
2, 3;

 ni_value1 | ni_value2 | nullif
-----------+-----------+--------
 abc       | abc       |
 abc       | efg       | abc
```


版权所有 © 华为云计算技术有限公司

```
(2 rows)
DROP TABLE tpcds.null_if_t1;
```

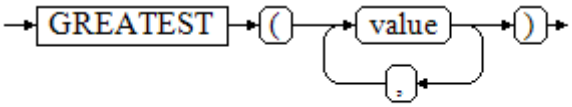
如果value1等于value2则返回NULL，否则返回value1。

```
SELECT NULLIF('Hello','Hello World');
 nullif
--------
 Hello
(1 row)
```

- GREATEST（最大值），LEAST（最小值）

  GREATEST的语法图请参见图21-7。

  图 21-7 greatest::=

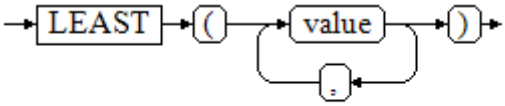
  

  从一个任意数字表达式的列表里选取最大的数值。

```
SELECT greatest(9000,155555,2.01);
 greatest
----------
   155555
(1 row)
```

  LEAST的语法图请参见图21-8。

  图 21-8 least::=

  从一个任意数字表达式的列表里选取最小的数值。

  以上的数字表达式必须都可以转换成一个普通的数据类型，该数据类型将是结果类型。

  列表中的NULL值将被忽略。只有所有表达式的结果都是NULL的时候，结果才是NULL。

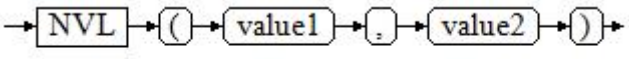
```
SELECT least(9000,2);
 least
-------
     2
(1 row)
```

  示例请参见**条件表达式函数**。

- NVL

  NVL的语法图请参见图21-9。

  图 21-9 nvl::=

 版权所有 © 华为云计算技术有限公司

如果value1为NULL则返回value2，如果value1非NULL，则返回value1。

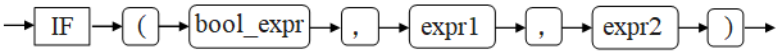
示例：

```
SELECT nvl(null,1);
nvl
-----
 1
(1 row)
SELECT nvl ('Hello World' ,1);
      nvl
---------------
 Hello World
(1 row)
```

- IF

  IF的语法图请参见**图21-10**。

  图 21-10 if::=

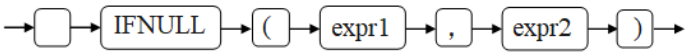
  当bool_expr为true时，返回expr1，否则返回expr2。

  示例请参见**条件表达式函数**。

- IFNULL

  IFNULL的语法图请参见**图21-11**。

  图 21-11 ifnull::=

  当expr1不为NULL时，返回expr1，否则返回expr2。

  示例请参见**条件表达式函数**。

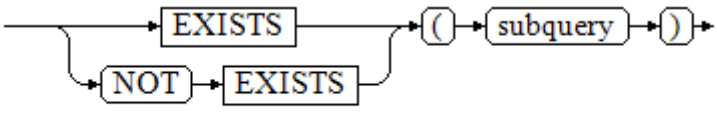
21.8.3 子查询表达式

子查询表达式主要有以下几种：

- EXISTS/NOT EXISTS

  EXISTS/NOT EXISTS的语法图请参见**图21-12**。

  图 21-12 EXISTS/NOT EXISTS::=

版权所有 © 华为云计算技术有限公司

EXISTS的参数是一个任意的SELECT语句，或者说子查询。系统对子查询进行运算以判断它是否返回行。如果它至少返回一行，则EXISTS结果就为"真"；如果子查询没有返回任何行， EXISTS的结果是"假"。

这个子查询通常只是运行到能判断它是否可以生成至少一行为止，而不是等到全部结束。

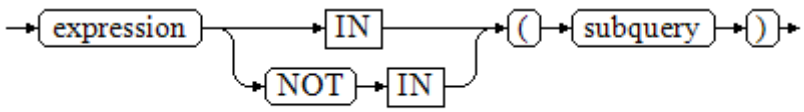
示例：

```
SELECT sr_reason_sk,sr_customer_sk FROM tpcds.store_returns WHERE EXISTS (SELECT d_dom FROM tpcds.date_dim WHERE d_dom = store_returns.sr_reason_sk and sr_customer_sk <10);
sr_reason_sk | sr_customer_sk
--------------+----------------
          13 |              2
          22 |              5
          17 |              7
          25 |              7
           3 |              7
          31 |              5
           7 |              7
          14 |              6
          20 |              4
           5 |              6
          10 |              3
           1 |              5
          15 |              2
           4 |              1
          26 |              3
(15 rows)
```

- IN/NOT IN

  IN/NOT IN的语法请参见**图21-13**。

  图 21-13 IN/NOT IN::=

  右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式对子查询结果的每一行进行一次计算和比较。如果找到任何相等的子查询行，则IN结果为"真"。如果没有找到任何相等行，则结果为"假"（包括子查询没有返回任何行的情况）。

  表达式或子查询行里的NULL遵照SQL处理布尔值和NULL组合时的规则。如果两个行对应的字段都相等且非空，则这两行相等；如果任意对应字段不等且非空，则这两行不等；否则结果是未知（NULL）。如果每一行的结果都是不等或NULL，并且至少有一个NULL，则IN的结果是NULL。

  示例：

  ```
   SELECT sr_reason_sk,sr_customer_sk  FROM tpcds.store_returns WHERE sr_customer_sk IN (SELECT d_dom FROM tpcds.date_dim WHERE d_dom < 10);
  sr_reason_sk | sr_customer_sk
  --------------+----------------
            10 |              3
            26 |              3
            22 |              5
            31 |              5
             1 |              5
            32 |              5
            32 |              5
             4 |              1
  ```

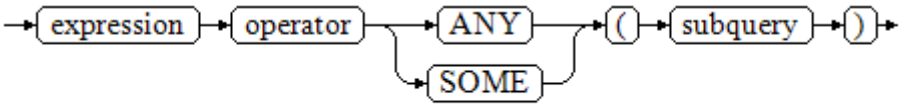

版权所有 © 华为云计算技术有限公司

```
        15 |            2
        13 |            2
        33 |            4
        20 |            4
        33 |            8
         5 |            6
        14 |            6
        17 |            7
         3 |            7
        25 |            7
         7 |            7
(19 rows)
```

- ANY/SOME

  ANY/SOME的语法图请参见**图21-14**。

  图 21-14 any/some::=

  右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较，其结果必须是布尔值。如果至少获得一个真值，则ANY结果为“真”。如果全部获得假值，则结果是“假”（包括子查询没有返回任何行的情况）。SOME是ANY的同义词。IN与ANY可以等效替换 。

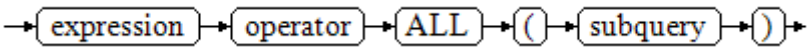
  示例：

```
SELECT sr_reason_sk,sr_customer_sk  FROM tpcds.store_returns WHERE sr_customer_sk < ANY
(SELECT d_dom FROM tpcds.date_dim WHERE d_dom < 10);
sr_reason_sk | sr_customer_sk
--------------+----------------
        26 |            3
        17 |            7
        32 |            5
        32 |            5
        13 |            2
        31 |            5
        25 |            7
         5 |            6
         7 |            7
        10 |            3
         1 |            5
        14 |            6
         4 |            1
         3 |            7
        22 |            5
        33 |            4
        20 |            4
        33 |            8
        15 |            2
(19 rows)
```

- ALL

  ALL的语法请参见**图21-15**。

  图 21-15 all::=

 版权所有 © 华为云计算技术有限公司

右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较，其结果必须是布尔值。如果全部获得真值，ALL结果为"真"（包括子查询没有返回任何行的情况）。如果至少获得一个假值，则结果是"假"。

示例：

```
SELECT sr_reason_sk,sr_customer_sk  FROM tpcds.store_returns WHERE sr_customer_sk < all(SELECT d_dom FROM tpcds.date_dim WHERE d_dom < 10);
 sr_reason_sk | sr_customer_sk
--------------+----------------
(0 rows)
```

21.8.4 数组表达式

IN

expression **IN** *(value [, ...])*

右侧括号中的是一个表达式列表。左侧表达式的结果与表达式列表的内容进行比较。如果列表中的内容符合左侧表达式的结果，则IN的结果为true。如果没有相符的结果，则IN的结果为false。

示例如下：

```
SELECT 8000+500 IN (10000, 9000) AS RESULT;
 result
----------
 f
(1 row)
```

说明

如果表达式结果为null，或者表达式列表不符合表达式的条件且右侧表达式列表返回结果至少一处为空，则IN的返回结果为null，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

NOT IN

expression **NOT IN** *(value [, ...])*

右侧括号中的是一个表达式列表。左侧表达式的结果与表达式列表的内容进行比较。如果在列表中的内容没有符合左侧表达式结果的内容，则NOT IN的结果为true。如果有符合的内容，则NOT IN的结果为false。

示例如下：

```
SELECT 8000+500 NOT IN (10000, 9000) AS RESULT;
 result
----------
 t
(1 row)
```

说明

如果查询语句返回结果为空，或者表达式列表不符合表达式的条件且右侧表达式列表返回结果至少一处为空，则NOT IN的返回结果为null，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

提示：在所有情况下X NOT IN Y等价于NOT(X IN Y)。

版权所有 © 华为云计算技术有限公司

ANY/SOME (array)

expression operator **ANY** *(array expression)*

expression operator **SOME** *(array expression)*

```
SELECT 8000+500 < SOME (array[10000,9000]) AS RESULT;
 result
----------
 t
(1 row)
SELECT 8000+500 < ANY (array[10000,9000]) AS RESULT;
 result
----------
 t
(1 row)
```

右侧括号中的是一个数组表达式，它必须产生一个数组值。左侧表达式的结果使用操作符对数组表达式的每一行结果都进行计算和比较，比较结果必须是布尔值。

- 如果对比结果至少获取一个真值，则ANY的结果为true。
- 如果对比结果没有真值，则ANY的结果为false。

说明

如果结果没有真值，并且数组表达式生成至少一个值为null，则ANY的值为NULL，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

SOME是ANY的同义词。

ALL (array)

expression operator **ALL** *(array expression)*

右侧括号中的是一个数组表达式，它必须产生一个数组值。左侧表达式的结果使用操作符对数组表达式的每一行结果都进行计算和比较，比较结果必须是布尔值。

- 如果所有的比较结果都为真值（包括数组不含任何元素的情况），则ALL的结果为true。
- 如果结果有任一结果是false，则ALL的结果为false。

如果数组表达式产生一个NULL数组，则ALL的结果为NULL。如果左边表达式的值为NULL，则ALL的结果通常也为NULL(某些不严格的比较操作符可能得到不同的结果)。另外，如果右边的数组表达式中包含null元素并且比较结果没有假值，则ALL的结果将是NULL(某些不严格的比较操作符可能得到不同的结果)， 而不是真。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

```
SELECT 8000+500 < ALL (array[10000,9000]) AS RESULT;
 result
----------
 t
(1 row)
```

21.8.5 行表达式

语法：

row_constructor operator row_constructor

两边都是一个行构造器，两行值必须具有相同数目的字段，每一行都进行比较，行比较允许使用=，<>，<，<=，>=等操作符，或其中一个相似的语义符。

版权所有 © 华为云计算技术有限公司

=<>和别的操作符使用略有不同。如果两行值的所有字段都是非空并且相等，则认为两行是相等的；如果两行值的任意字段为非空并且不相等，则认为两行是不相等的；否则比较结果是未知的（null）。

对于<，<=，>，> =的情况下，行中元素从左到右依次比较，直到遇到一对不相等的元素或者一对为空的元素。如果这对元素中存在至少一个null值，则比较结果是未知的（null），否则这对元素的比较结果为最终的结果。

示例：

```
SELECT ROW(1,2,NULL) < ROW(1,3,0) AS RESULT;
 result
----------
 t
(1 row)
```

21.9 类型转换

21.9.1 概述

背景信息

在SQL语言中，每个数据都与一个决定其行为和用法的数据类型相关。DWS提供一个可扩展的数据类型系统，该系统比其它SQL实现更具通用性和灵活性。因而，DWS中大多数类型转换是由通用规则来管理的，这种做法允许使用混合类型的表达式。

DWS扫描/分析器只将词法元素分解成五个基本种类：整数、浮点数、字符串、标识符和关键字。大多数非数字类型首先表现为字符串。SQL语言的定义允许将常量字符串声明为具体的类型。例，下面查询：

```
SELECT text 'Origin' AS "label", point '(0,0)' AS "value";
 label  | value
--------+-------
 Origin | (0,0)
(1 row)
```

示例中有两个文本常量，类型分别为text和point。如果没有为字符串文本声明类型，则该文本首先被定义成一个unknown类型。

在DWS分析器里，有四种基本的SQL结构需要独立的类型转换规则：

- 函数调用

  多数SQL类型系统是建筑在一套丰富的函数上的。函数调用可以有一个或多个参数。因为SQL允许函数重载，所以不能通过函数名直接找到要调用的函数，分析器必须根据函数提供的参数类型选择正确的函数。

- 操作符

  SQL允许在表达式上使用前缀或后缀（单目）操作符，也允许表达式内部使用双目操作符（两个参数）。像函数一样，操作符也可以被重载，因此操作符的选择也和函数一样取决于参数类型。

- 值存储

  INSERT和UPDATE语句将表达式结果存入表中。语句中的表达式类型必须和目标字段的类型一致或者可以转换为一致。

- UNION，CASE和相关构造

  因为联合SELECT语句中的所有查询结果必须在一列里显示出来，所以每个SELECT子句中的元素类型必须相互匹配并转换成一个统一类型。类似地，一个CASE构造

 版权所有 © 华为云计算技术有限公司

的结果表达式必须转换成统一的类型，这样整个case表达式会有一个统一的输出类型。同样的要求也存在于ARRAY构造以及GREATEST和LEAST函数中。

系统表PG_CAST存储了有关数据类型之间的转换关系以及如何执行这些转换的信息。详细信息请参见《数据仓库服务开发指南》中“PG_CAST”章节。

语义分析阶段会决定表达式的返回值类型并选择适当的转换行为。数据类型的基本类型分类，包括：boolean、numeric、string、bitstring、datetime、timespan、geometric和network。每种类型都有一种或多种首选类型用于解决类型选择的问题。根据首选类型和可用的隐含转换，就可能保证有歧义的表达式（那些有多个候选解析方案的）得到有效的方式解决。

所有类型转换规则都是建立在下面几个基本原则上的：

- 隐含转换绝不能有奇怪的或不可预见的输出。
- 如果一个查询不需要隐含的类型转换，分析器和执行器不应该进行更多的额外操作。这就是说，任何一个类型匹配、格式清晰的查询不应该在分析器里耗费更多的时间，也不应该向查询中引入任何不必要的隐含类型转换调用。
- 另外，如果一个查询在调用某个函数时需要进行隐式转换，当用户定义了一个有正确参数的函数后，解释器应该选择使用新函数。

TD 兼容模式下，空串转换为数值类型的处理

- TD数据库不同于Oracle，Oracle将空串当做NULL进行处理，TD在将空串转换为数值类型的时候，默认将空串转换为0进行处理，因此查询空串会查询到数值为0的数据。同样地，在TD兼容模式下，字符串转换数值的过程中，也会将空串默认转换为相应数值类型的0值进行处理。除此之外，' - '、' + '、' '这些字符串也都会在TD兼容模式下默认转换为0进行处理，但是小数点字符串' . '会报错。例如：

```
create table t1(no int,col varchar);
insert into t1 values(1,'');
insert into t1 values(2,null);
select * from t1 where col is null;
 no | col
----+-----
  2 |
(1 row)
select * from t1 where col='';
 no | col
----+-----
  1 |
(1 row)
```

- MySQL兼容模式下对空串转换为数值类型的处理和TD兼容模式相同。

21.9.2 操作符

操作符类型解析

1. 从系统表pg_operator中选出要考虑的操作符。如果可以找到一个参数类型以及参数个数都一致的操作符，那么这个操作符就是最终使用的操作符。如果找到了多个备选的操作符，将从中选择一个最合适的。
2. 寻找最优匹配。
   a. 丢弃输入类型不匹配以及无法隐式转换成匹配的候选操作符。unknown文本在这种情况下可以转换成任何类型。如果只剩下一个候选操作符，则使用，否则继续下一步。

 版权所有 © 华为云计算技术有限公司

b. 查看所有候选操作符，并保留输入类型最匹配的操作符。此时，域被看作和其基本类型相同。如果没有完全匹配的操作符，则保留所有候选。如果只剩下一个候选操作符，则使用，否则继续下一步。

c. 查看所有候选操作符，保留需要类型转换时接受（属于输入数据类型的类型范畴的）首选类型位置最多的操作符。如果没有接受首选类型的操作符，则保留所有候选。如果只剩下一个候选操作符，则使用，否则继续下一步。

d. 如果有任何输入参数是unknown类型，请检查其余候选操作符对应参数位置的类型范畴。在每一个能够接受string类型范畴的位置使用string类型（这种偏向字符串的做法合理，因为unknown文本跟字符串相似）。另外，如果所有剩下的候选操作符都接受相同的类型范畴，则选择该类型范畴，否则会报错（因为在没有更多线索的条件下无法作出正确的选择）。现在丢弃不接受选定类型范畴的候选操作符。此外，如果有任意候选操作符接受该范畴中的首选类型，则丢弃该参数接受非首选类型的候选操作符。如果没有一个操作符能被保留，则保留所有候选。如果只剩下一个候选操作符，则使用，否则继续下一步。

e. 如果同时有unknown和已知参数，并且所有已知参数都是相同的类型，那么假设unknown参数也属于该类型，并检查哪些候选操作符在unknown参数位置接受该类型。如果只有一个操作符符合，则使用。否则，报错。

示例

示例1：阶乘操作符类型解析。在系统表中里只有一个阶乘操作符（后缀!），它以bigint作为参数。扫描器给下面查询表达式的参数赋予bigint的初始类型：

```
SELECT 40 ! AS "40 factorial";

                  40 factorial
-----------------------------------------------------
 815915283247897734345611269596115894272000000000
(1 row)
```

分析器对参数做类型转换，查询等效于：

```
SELECT CAST(40 AS bigint) ! AS "40 factorial";
```

示例2：字符串连接操作符类型分析。一种字符串风格的语法既可以用于字符串也可以用于复杂的扩展类型。未声明类型的字符串将被所有可能的候选操作符匹配。有一个未声明的参数的例子：

```
SELECT text 'abc' || 'def' AS "text and unknown";
 text and unknown
------------------
 abcdef
(1 row)
```

本例中分析器寻找两个参数都是text的操作符。确实有这样的操作符，两个参数都是text类型。

下面是连接两个未声明类型的值：

```
SELECT 'abc' || 'def' AS "unspecified";
 unspecified
-------------
 abcdef
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

因为查询中没有声明任何类型，所以本例中对类型没有任何初始提示。因此，分析器查找所有候选操作符，发现既存在接受字符串类型范畴的操作符也存在接受位串类型范畴的操作符。因为字符串类型范畴是首选，所以选择字符串类型范畴的首选类型text作为解析未知类型文本的声明类型。

示例3：绝对值和取反操作符类型分析。DWS操作符表里面有几条记录对应于前缀操作符@，它们都用于为各种数值类型实现绝对值操作。其中之一用于float8类型，它是数值类型范畴中的首选类型。因此，在面对unknown输入的时候， DWS会使用该类型：

```
SELECT @ '-4.5' AS "abs";
 abs
-----
 4.5
(1 row)
```

此处，系统在应用选定的操作符之前隐式的转换unknown类型的文字为float8类型。

示例4：数组包含操作符类型分析。以操作符两侧带有一个已知类型和一个未知类型的情况为例：

```
SELECT array[1,2] <@ '{1,2,3}' as "is subset";
 is subset
-----------
 t
(1 row)
```

说明

DWS的pg_operator表中有多条记录对应于中缀操作符<@，但是只有“数组包含(anyarray <@ anyarray) ”和“范围包含(anyelement <@ anyrange)”可在左侧接受一个整数数组。因为多态伪类型(参阅**伪类型**)不被认为是首选操作符类型，因此解析器无法在此基础上解决歧义。不过，**寻找最优匹配**的最后一条解析规则又指出，假定未知类型的文字和另一个已知的输入类型相同，也就是只有两个运算符之一可以匹配，所以选择数组包含。（如果选择了范围包含，就会报错，因为字符串格式不正确不能作为范围。）

21.9.3 函数

函数类型解析

1. 从系统表pg_proc中选择所有可能被选到的函数。如果使用了一个不带模式修饰的函数名字，那么认为该函数是那些在当前搜索路径中的函数。如果给出一个带修饰的函数名，那么只考虑指定模式中的函数。

   如果搜索路径中找到了多个不同参数类型的函数。将从中选择一个合适的函数。
2. 查找和输入参数类型完全匹配的函数。如果找到一个，则用之。如果输入的实参类型都是unknown类型，则不会找到匹配的函数。
3. 如果未找到完全匹配，请查看该函数是否为一个特殊的类型转换函数。
4. 寻找最优匹配。

   a. 抛弃那些输入类型不匹配并且也不能隐式转换成匹配的候选函数。unknown文本在这种情况下可以转换成任何东西。如果只剩下一个候选项，则用之，否则继续下一步。

   b. 遍历所有候选函数，保留那些输入类型匹配最准确的。此时，域被看作和它们的基本类型相同。如果没有一个函数能准确匹配，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。

版权所有 © 华为云计算技术有限公司

c. 遍历所有候选函数，保留那些需要类型转换时接受首选类型位置最多的函数。如果没有接受首选类型的函数，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。

d. 如果有任何输入参数是unknown类型，检查剩余的候选函数对应参数位置的类型范畴。在每一个能够接受字符串类型范畴的位置使用string类型（这种对字符串的偏爱是合适的，因为unknown文本确实像字符串）。另外，如果所有剩下的候选函数都接受相同的类型范畴，则选择该类型范畴，否则抛出一个错误（因为在没有更多线索的条件下无法作出正确的选择）。现在抛弃不接受选定的类型范畴的候选函数，然后，如果任意候选函数在那个范畴接受一个首选类型，则抛弃那些在该参数位置接受非首选类型的候选函数。如果没有一个候选符合这些测试则保留所有候选。如果只有一个候选函数符合，则使用它；否则，继续下一步。

e. 如果同时有unknown和已知类型的参数，并且所有已知类型的参数有相同的类型，假设unknown参数也是这种类型，检查哪个候选函数可以在unknown参数位置接受这种类型。如果正好一个候选符合，那么使用它。否则，产生一个错误。

示例

示例1：圆整函数参数类型解析。只有一个round函数有两个参数（第一个是numeric，第二个是integer）。所以下面的查询自动把第一个类型为integer的参数转换成numeric类型。

```
SELECT round(4, 4);
 round
--------
 4.0000
(1 row)
```

实际上它被分析器转换成：

```
SELECT round(CAST (4 AS numeric), 4);
```

因为带小数点的数值常量初始时被赋予numeric类型，因此下面的查询将不需要类型转换，并且可能会略微高效一些：

```
SELECT round(4.0, 4);
```

示例2：子字符串函数类型解析。有好几个substr函数，其中一个接受text和integer类型。如果用一个未声明类型的字符串常量调用它，系统将选择接受string类型范畴的首选类型（也就是text类型）的候选函数。

```
SELECT substr('1234', 3);
 substr
--------
    34
(1 row)
```

如果该字符串声明为varchar类型，就像从表中取出来的数据一样，分析器将试着将其转换成text类型：

```
SELECT substr(varchar '1234', 3);
 substr
--------
    34
(1 row)
```

被分析器转换后实际上变成：

```
SELECT substr(CAST (varchar '1234' AS text), 3);
```

 版权所有 © 华为云计算技术有限公司

说明

分析器从pg_cast表中了解到text和varchar是二进制兼容的，意思是说一个可以传递给接受另一个的函数而不需要做任何物理转换。因此，在这种情况下，实际上没有做任何类型转换。

而且，如果以integer为参数调用函数，分析器将试图将其转换成text类型：

```
SELECT substr(1234, 3);
substr
--------
 34
(1 row)
```

被分析器转换后实际上变成：

```
SELECT substr(CAST (1234 AS text), 3);
 substr
--------
    34
(1 row)
```

21.9.4 值存储

值存储数据类型解析

1. 查找与目标字段准确的匹配。
2. 试着将表达式直接转换成目标类型。如果已知这两种类型之间存在一个已登记的转换函数，那么直接调用该转换函数即可。如果表达式是一个未知类型文本，该文本字符串的内容将交给目标类型的输入转换过程。
3. 检查目标类型是否有长度转换。长度转换是一个从某类型到自身的转换。如果在pg_cast表里面找到一个，那么在存储到目标字段之前先在表达式上应用。这样的转换函数总是接受一个额外的类型为integer的参数，它接收目标字段的atttypmod值（实际上是其声明长度，atttypmod的解释随不同的数据类型而不同），并且它可能接受一个boolean类型的第三个参数，表示转换是显式的还是隐式的。转换函数负责施加那些长度相关的语义，比如长度检查或者截断。

示例

character存储类型转换。对一个目标列定义为character(20)的语句，下面的语句显示存储值的长度正确：

```
CREATE TABLE x1
(
    customer_sk             integer,
    customer_id             char(20),
    first_name              char(6),
    last_name               char(8)
)
with (orientation = column,compression=middle)
distribute by hash (last_name);

INSERT INTO x1(customer_sk, customer_id, first_name) VALUES (3769, 'abcdef', 'Grace');

SELECT customer_id, octet_length(customer_id) FROM x1;
     customer_id       | octet_length
-----------------------+--------------
 abcdef                |           20
(1 row)
DROP TABLE x1;
```


版权所有 © 华为云计算技术有限公司

说明

这里真正发生的事情是两个unknown文本缺省解析成text，这样就允许||操作符解析成text连接。然后操作符的text结果转换成bpchar("空白填充的字符型"， character类型内部名称)以匹配目标字段类型。不过，从text到bpchar的转换是二进制兼容的，这样的转换是隐含的并且实际上不做任何函数调用。最后，在系统表里找到长度转换函数bpchar(bpchar, integer, boolean)并且应用于该操作符的结果和存储的字段长。这个类型相关的函数执行所需的长度检查和额外的空白填充。

21.9.5 UNION，CASE 和相关构造

SQL UNION构造把不相同的数据类型进行匹配输出为统一的数据类型结果集。因为SELECT UNION语句中的所有查询结果必须在一列里显示出来，所以每个SELECT子句中的元素类型必须相互匹配并转换成一个统一的数据类型。类似地，一个CASE构造的结果表达式必须转换成统一的类型，这样整个case表达式会有一个统一的输出类型。同样的要求也存在于ARRAY构造以及GREATEST和LEAST函数中。

UNION，CASE 和相关构造解析

- 如果所有输入都是相同的数据类型，不包括unknown类型（即输入的字符串文本未声明类型，该文本首先被定义成一个未知类型），那么解析成所输入的相同数据类型。
- 如果所有输入都是unknown类型则解析成text类型（字符串类型范畴的首选类型）。否则，忽略unknown输入。
- 如果输入不属于同一个类型范畴，查询失败（unknown类型除外）。
- 如果输入类型是同一个类型范畴，则选择该类型范畴的首选类型（union操作会选择第一个分支的类型作为所选类型的情况除外）。

  说明

  系统表pg_type中typcategory表示数据类型范畴， typispreferred表示是否是typcategory分类中的首选类型。
- 把所有输入转换为所选的类型（对于字符串保持原有长度）。如果从给定的输入到所选的类型没有隐式转换则失败。
- 若输入中含json、txid_snapshot、sys_refcursor或几何类型，则不能进行union。

对于 CASE、COALESCE、IF 和 IFNULL，在 TD 兼容模式下的处理

- 如果所有输入都是相同的数据类型，不包括unknown类型，那么解析成所输入的相同数据类型。
- 如果所有输入都是unknown类型则解析成text类型。
- 如果输入字符串（包括unknown，unknown当text来处理）和数字类型，那么解析成字符串类型，如果是其他不同的类型范畴，则报错。
- 如果输入类型是同一个类型范畴，则选择该类型的优先级较高的类型。
- 把所有输入转换为所选的类型。如果从给定的输入到所选的类型没有隐式转换则失败。

对于 CASE、COALESCE、IF 和 IFNULL，在 MySQL 兼容模式下的处理

- 如果所有输入都是相同的类型，不包括unknown类型，那么解析成所输入的相同数据类型。

版权所有 © 华为云计算技术有限公司

- 如果所有输入都是unknown类型则解析成text类型。
- 如果输入是unknown类型和某一非unknown类型，则解析成该非unknown类型。
- 如果存在多种非unknown类型，将enum类型当做text类型，再进行比较。
- 如果输入类型是同一个类型范畴，则选择该类型的优先级较高的类型。如果是不同的类型范畴，则解析成text类型。
- 把所有输入转换为所选的类型。如果从给定的输入到所选的类型没有隐式转换则失败。

示例

示例1：Union中的未知类型解析。示例中未知类型文本'b'将被解析成text类型。

```
SELECT text 'a' AS "text" UNION SELECT 'b';
 text
------
 a
 b
(2 rows)
```

示例2：简单Union中的类型解析。文本1.2是numeric类型，且integer类型值1可以隐式地转换为numeric，因此使用numeric类型。

```
SELECT 1.2 AS "numeric" UNION SELECT 1;
 numeric
---------
       1
     1.2
(2 rows)
```

示例3：转换Union中的类型解析。示例中由于类型real不能被隐式地转换为integer，而integer可以隐式地转换成real类型，那么联合结果类型被系统决定为real。

```
SELECT 1 AS "real" UNION SELECT CAST('2.2' AS REAL);
 real
------
    1
  2.2
(2 rows)
```

示例4：COALESCE函数输入int和varchar类型，ORA模式下会报错，TD模式下解析为varchar类型，MySQL模式下解析为text类型。

```
--指定兼容模式创建数据库ora_db、td_db、mysql_db。
CREATE DATABASE ora_db dbcompatibility = 'ORA';
CREATE DATABASE td_db dbcompatibility = 'TD';
CREATE DATABASE mysql_db dbcompatibility = 'MySQL';

--切换数据库为ora_db。
\c ora_db

--创建表t1。
ora_db=# CREATE TABLE t1(a int, b varchar(10));

--查看coalesce参数输入int和varchar类型的查询语句的执行计划。
ora_db=# EXPLAIN SELECT coalesce(a, b) FROM t1;
ERROR:  COALESCE types integer and character varying cannot be matched
CONTEXT:  referenced column: coalesce

--删除表。
ora_db=# DROP TABLE t1;

--切换数据库为td_db。
ora_db=# \c td_db
```

 版权所有 © 华为云计算技术有限公司

```
--创建表t2。
td_db=# CREATE TABLE t2(a int, b varchar(10));

--查看coalesce参数输入int和varchar类型的查询语句的执行计划。
td_db=# EXPLAIN VERBOSE select coalesce(a, b) from t2;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                    Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (COALESCE((t2.a)::character varying, t2.b))
        Node/s: All datanodes
        Remote query: SELECT COALESCE(a::character varying, b) AS "coalesce" FROM public.t2
(10 rows)

--删除表。
td_db=# DROP TABLE t2;

--切换数据库为mysql_db。
td_db=# \c mysql_db

--创建表t3。
mysql_db=# CREATE TABLE t3(a int, b varchar(10));

--查看coalesce参数输入int和varchar类型的查询语句的执行计划。
mysql_db=# EXPLAIN VERBOSE select coalesce(a, b) from t3;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                    Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (COALESCE((t3.a)::text, (t3.b)::text))
        Node/s: All datanodes
        Remote query: SELECT COALESCE(a::text, b::text) AS "coalesce" FROM public.t3
(10 rows)

--删除表。
mysql_db=# DROP TABLE t3;

--切换数据库为gaussdb
mysql_db=# \c gaussdb

--删除数据库。
DROP DATABASE ora_db;
DROP DATABASE td_db;
DROP DATABASE mysql_db;
```

21.10 全文检索

21.10.1 介绍

21.10.1.1 全文检索概述

文本搜索操作符在数据库中已存在多年。DWS为文本数据类型提供~、~*、LIKE和ILIKE操作符，但这些操作符缺乏现代信息系统所要求的许多必要属性，不过这一问题可以通过使用索引及词典进行解决。

 版权所有 © 华为云计算技术有限公司

文本检索缺乏信息系统所要求的必要属性：

- 没有语义支持，即使是英语。

  由于要识别派生词并不是那么容易，因此正则表达式也不能满足要求。如，satisfies和satisfy，当使用正则表达式寻找satisfy时，并不会查询到包含satisfies的文档。用户可以使用OR搜索多种派生形式，但过程非常繁琐。并且有些词会有上千的派生词，因此容易出错。
- 没有对搜索结果的分类（排序）。当搜索出成千的文档时，查找效率很低。
- 由于没有索引的支持，每一次的搜索需要遍历所有的文档，整体搜索比较缓慢。

使用全文索引可以对文档进行预处理，并且可以使后续的搜索更快速。预处理过程包括：

- 将文档解析成token。

  为每个文档标记不同类别的token是非常有必要的，例如：数字、文字、复合词、电子邮件地址，这样就可以做不同的处理。原则上token的类别依赖于具体的应用，但对于大多数的应用来说，可以使用一组预定义的token类。
- 将token转换为词素。

  词素像token一样是一个字符串，但它已经标准化处理，这样同一个词的不同形式是一样的。例如，标准化通常包括：将大写字母折成小写字母、删除后缀（如英语中的s或者es）。这将允许通过搜索找到同一个词的不同形式，不需要繁琐地输入所有可能的变形样式。同时，这一步通常会删除停用词。这些停用词通常因为太常见而对搜索无用。（总之，token是文档文本的原片段，而词素被认为是有用的索引和搜索词。）DWS使用词典执行这一步，且提供了各种标准的词典。
- 保存搜索优化后的预处理文档。

  比如，每个文档可以呈现为标准化词素的有序组合。伴随词素，通常还需要存储词素位置信息以用于邻近排序。因此文档包含的查询词越密集其排序越高。

词典能够对token如何标准化做到细粒度控制。使用合适的词典，可以定义不被索引的停用词。

数据类型tsvector用于存储预处理文档，tsquery用于存储查询条件，详细内容可参见**文本搜索类型**。为数据类型tsvector提供的函数和操作符可参见**文本检索函数和操作符**，其中最重要的是匹配运算符@@，请参见**基本文本匹配**。

21.10.1.2 文档概念

文档是全文搜索系统的搜索单元，例如：杂志上的一篇文章或电子邮件消息。文本搜索引擎必须能够解析文档，而且可以存储父文档的关联词素（关键词）。后续，这些关联词素用来搜索包含查询词的文档。

在DWS中，文档通常是一个数据库表中一行的文本字段，或者这些字段的可能组合（级联）。文档可能存储在多个表中或者需动态获取。换句话说，一个文档由被索引化的不同部分构成，因此无法存储为一个整体。比如：

```
SELECT d_dow || '-' || d_dom || '-' || d_fy_week_seq  AS identify_serials FROM tpcds.date_dim WHERE
d_fy_week_seq = 1;
identify_serials
------------------
 5-6-1
 0-8-1
 2-3-1
 3-4-1
 4-5-1
 1-2-1
 6-7-1
(7 rows)
```


版权所有 © 华为云计算技术有限公司

须知

实际上，在这些示例查询中，应该使用coalesce防止一个独立的NULL属性导致整个文档的NULL结果。

另外一种可能是：文档在文件系统中作为简单的文本文件存储。在这种情况下，数据库可以用于存储全文索引并且执行搜索，同时可以使用一些唯一标识从文件系统中检索文档。然而，从数据库外部检索文件需要拥有系统管理员权限或者特殊函数支持。因此，还是将所有数据保存在数据库中比较方便。同时，将所有数据保存在数据库中可以方便地访问文档元数据以便于索引和显示。

为了实现文本搜索目的，必须将每个文档减少至预处理后的tsvector格式。搜索和相关性排序都是在tsvector形式的文档上执行的。原始文档只有在被选中要呈现给用户时才会被当检索。因此，常将tsvector说成文档，但是很显然其实它只是完整文档的一种紧凑表示。

21.10.1.3 基本文本匹配

DWS的全文检索基于匹配算子@@，当一个tsvector(document)匹配到一个tsquery(query)时，则返回true。其中，tsvector(document)和tsquery(query)两种数据类型可以任意排序。

```
SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector @@ 'cat & rat'::tsquery AS RESULT;
 result
----------
 t
(1 row)
SELECT 'fat & cow'::tsquery @@ 'a fat cat sat on a mat and ate a fat rat'::tsvector AS RESULT;
 result
----------
 f
(1 row)
```

正如上面例子表明，tsquery不仅是文本，且比tsvector包含的要多。tsquery包含已经标注化为词条的搜索词，同时可能是使用AND、OR、或NOT操作符连接的多个术语。详细请参见**文本搜索类型**。函数to_tsquery和plainto_tsquery对于将用户书写文本转换成适合的tsquery是非常有用的，比如将文本中的词标准化。类似地，to_tsvector用于解析和标准化文档字符串。因此，实际中文本搜索匹配看起来更像这样：

```
SELECT to_tsvector('fat cats ate fat rats') @@ to_tsquery('fat & rat') AS RESULT;
result
----------
 t
(1 row)
```

需要注意的是，下面这种方式是不可行的：

```
SELECT 'fat cats ate fat rats'::tsvector @@ to_tsquery('fat & rat')AS RESULT;
result
----------
 f
(1 row)
```

由于tsvector没有对rats进行标准化，所以rats不匹配rat。

@@操作符也支持text输入，允许一个文本字符串的显示转换为tsvector或者在简单情况下忽略tsquery。可用形式是：

```
tsvector @@ tsquery
tsquery  @@ tsvector
text @@ tsquery
text @@ text
```


版权所有 © 华为云计算技术有限公司

形式text @@ tsquery等价于to_tsvector(text) @@ tsquery，而text @@ text等价于to_tsvector(text) @@ plainto_tsquery(text)。

21.10.1.4 分词器

全文检索功能还可以做更多事情：忽略索引某个词（停用词），处理同义词和使用复杂解析，例如，不仅基于空格的解析。这些功能通过文本搜索分词器控制。DWS支持多语言的预定义的分词器，并且可以创建分词器（gsql的\dF命令显示了所有可用分词器）。

在安装期间选择一个合适的分词器，并且在postgresql.conf中相应的设置default_text_search_config。如果为了整个集群使用同一个文本搜索分词器可以使用postgresql.conf中的值。如果需要在集群中使用不同分词器，可以使用ALTER DATABASE ... SET在任一数据库进行配置。用户也可以在每个会话中设置default_text_search_config。

每个依赖于分词器的文本搜索函数有一个可选的配置参数，用来明确声明所使用的分词器。仅当忽略这个参数的时候，才使用default_text_search_config。

为了更方便的建立自定义文本搜索分词器，可以通过简单的数据库对象建立分词器。DWS文本搜索功能提供了四种类型与分词器相关的数据库对象：

- 文本搜索解析器将文档分解为token，并且分类每个token（例如：词和数字）。
- 文本搜索词典将token转换成规范格式并且丢弃停用词。
- 文本搜索模板提供潜在的词典功能：一个词典指定一个模板，并且为模板设置参数。
- 文本搜索分词器选择一个解析器，并且使用一系列词典规范化语法分析器产生的token。

21.10.2 表和索引

21.10.2.1 搜索表

在不使用索引的情况下也可以进行全文检索。

- 一个简单查询：将body字段中包含science的每一行打印出来。

```
DROP SCHEMA IF EXISTS tsearch CASCADE;

CREATE SCHEMA tsearch;

CREATE TABLE tsearch.pgweb(id int, body text, title text, last_mod_date date);

INSERT INTO tsearch.pgweb VALUES(1, 'Philology is the study of words, especially the history and development of the words in a particular language or group of languages.', 'Philology', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(2, 'Mathematics is the science that deals with the logic of shape, quantity and arrangement.', 'Mathematics', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(3, 'Computer science is the study of processes that interact with data and that can be represented as data in the form of programs.', 'Computer science', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(4, 'Chemistry is the scientific discipline involved with elements and compounds composed of atoms, molecules and ions.', 'Chemistry', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(5, 'Geography is a field of science devoted to the study of the lands, features, inhabitants, and phenomena of the Earth and planets.', 'Geography', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(6, 'History is a subject studied in schools, colleges, and
```

 版权所有 © 华为云计算技术有限公司

```
universities that deals with events that have happened in the past.', 'History', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(7, 'Medical science is the science of dealing with the
maintenance of health and the prevention and treatment of disease.', 'Medical science', '2010-1-1');

INSERT INTO tsearch.pgweb VALUES(8, 'Physics is one of the most fundamental scientific disciplines,
and its main goal is to understand how the universe behaves.', 'Physics', '2010-1-1');


SELECT id, body, title FROM tsearch.pgweb WHERE to_tsvector('english', body) @@
to_tsquery('english', 'science');
 id |                                                 body                                                 |  title
----
+----------------------------------------------------------------------------------------------------------------------------------
----+---------

 2 | Mathematics is the science that deals with the logic of shape, quantity and
arrangement.                                        | Mathematics
 3 | Computer science is the study of processes that interact with data and that can be represented as
data in the form of programs. | Computer science
 5 | Geography is a field of science devoted to the study of the lands, features, inhabitants, and
phenomena of the Earth and planets.   | Geography
 7 | Medical science is the science of dealing with the maintenance of health and the prevention and
treatment of disease.           | Medical science
(4 rows)
```

像science这样的相关词也会被找到，因为这些词都被处理成了相同标准的词条。

上面的查询指定english配置来解析和规范化字符串。也可以省略此配置，通过default_text_search_config进行配置设置：

```
SHOW default_text_search_config;
 default_text_search_config
----------------------------
 pg_catalog.english
(1 row)

SELECT id, body, title FROM tsearch.pgweb WHERE to_tsvector(body) @@ to_tsquery('science');
 id |                                                 body                                                 |  title
----
+----------------------------------------------------------------------------------------------------------------------------------
----+---------

 2 | Mathematics is the science that deals with the logic of shape, quantity and
arrangement.                                        | Mathematics
 3 | Computer science is the study of processes that interact with data and that can be represented as
data in the form of programs. | Computer science
 5 | Geography is a field of science devoted to the study of the lands, features, inhabitants, and
phenomena of the Earth and planets.   | Geography
 7 | Medical science is the science of dealing with the maintenance of health and the prevention and
treatment of disease.           | Medical science

(4 rows)
```

- 一个复杂查询：检索出在title或者body字段中包含treatment和science的最近10篇文档：

```
SELECT title FROM tsearch.pgweb WHERE to_tsvector(title || ' ' || body) @@ to_tsquery('treatment &
science') ORDER BY last_mod_date DESC LIMIT 10;
 title
--------

Medical science

(1 rows)
```

为了清晰，举例中没有调用coalesce函数在两个字段中查找包含NULL的行。

以上例子均在没有索引的情况下进行查询。对于大多数应用程序来说，这个方法很慢。因此除了偶尔的特定搜索，文本搜索在实际使用中通常需要创建索引。

 版权所有 © 华为云计算技术有限公司

21.10.2.2 创建索引

为了加速文本搜索，可以创建GIN索引。

```
CREATE INDEX pgweb_idx_1 ON tsearch.pgweb USING gin(to_tsvector('english', body));
```

to_tsvector()函数有两个版本，只输一个参数的版本和输两个参数的版本。

只输一个参数时，系统默认采用default_text_search_config所指定的分词器。

创建索引时必须使用to_tsvector的两参数版本，否则索引内容可能不一致。只有指定了分词器名称的全文检索函数才可以在索引表达式中使用。因为索引的内容不受default_text_search_config的影响。由于default_text_search_config的值可以随时调整，从而导致不同条目生成的tsvector采用了不同的分词器，并且无法区分究竟使用了哪个分词器。正确地转储和恢复这样的索引也是不支持的。

在上述创建索引中to_tsvector使用了两个参数，只有当查询时也使用了两个参数，且参数值与索引中相同时，才会使用该索引。例如，WHERE to_tsvector('english', body) @@ 'a & b' 可以使用索引，但WHERE to_tsvector(body) @@ 'a & b'不能使用索引。这确保索引各条目是使用相同的分词器创建的。

索引中的分词器名称由另一列指定时可以建立更复杂的表达式索引。例如：

```
CREATE INDEX pgweb_idx_2 ON tsearch.pgweb USING gin(to_tsvector('zhparser', body));
```

说明

本示例中zhparser仅支持UTF8/GBK的数据库编码格式，在Encoding为SQL_ASCII下会报错。

其中body是pgweb表中的一列。当对索引的各条目使用了哪个分词器进行记录时，允许在同一索引中存在混合分词器。在某些场景下这将是有用的。例如，文档集合中包含不同语言的文档时。再次强调，打算使用索引的查询必须措辞匹配，例如，WHERE to_tsvector(config_name, body) @@ 'a & b'与索引中的to_tsvector措辞匹配。

索引甚至可以连接列：

```
CREATE INDEX pgweb_idx_3 ON tsearch.pgweb USING gin(to_tsvector('english', title || ' ' || body));
```

另一个方法是创建一个单独的tsvector列控制to_tsvector的输出。下面的例子是title和body的连接， 当其它是NULL的时候，使用coalesce确保一个字段仍然会被索引：

```
ALTER TABLE tsearch.pgweb ADD COLUMN textsearchable_index_col tsvector;
UPDATE tsearch.pgweb SET textsearchable_index_col = to_tsvector('english', coalesce(title,'') || ' ' ||
coalesce(body,''));
```

然后为加速搜索创建一个GIN索引：

```
CREATE INDEX textsearch_idx_4 ON tsearch.pgweb USING gin(textsearchable_index_col);
```

现在，就可以执行一个快速全文搜索了：

```
SELECT title
FROM tsearch.pgweb
WHERE textsearchable_index_col @@ to_tsquery('science & Computer')
ORDER BY last_mod_date DESC
LIMIT 10;

 title
--------
 Computer science

(1 rows)
```

相比于一个表达式索引，单独列方法的一个优势是：它没有必要在查询时明确指定分词器以便能使用索引。正如上面例子所示，查询可以依赖于

版权所有 © 华为云计算技术有限公司

default_text_search_config。另一个优势是搜索比较快速，因为它没有必要重新利用to_tsvector调用来验证索引匹配。表达式索引方法更容易建立，且它需要较少的磁盘空间，因为tsvector形式没有明确存储。

21.10.2.3 索引使用约束

下面是一个使用索引的示例：

```
create table table1 (c_int int,c_bigint bigint,c_varchar varchar,c_text text) with(orientation=row);

create text search configuration ts_conf_1(parser=POUND);
create text search configuration ts_conf_2(parser=POUND) with(split_flag='%');

set default_text_search_config='ts_conf_1';
create index idx1 on table1 using gin(to_tsvector(c_text));

set default_text_search_config='ts_conf_2';
create index idx2 on table1 using gin(to_tsvector(c_text));

select c_varchar,to_tsvector(c_varchar) from table1 where to_tsvector(c_text) @@ plainto_tsquery('#@……&**') and to_tsvector(c_text) @@ plainto_tsquery('company') and c_varchar is not null order by 1 desc limit 3;
```

该例子的关键点是表table1的同一个列c_text上建立了两个GIN索引：idx1和idx2，但这两个索引是在不同default_text_search_config的设置下建立的。该例子和同一张表的同一个列上建立普通索引的不同之处在于：

- GIN索引使用了不同的parser（即分隔符不同），那么idx1和idx2的索引数据是不同的；
- 在同一张表的同一个列上建立的多个普通索引的索引数据是相同的；

因此当执行同一个查询时，使用idx1和idx2查询出的结果是不同的。

使用约束

通过上面的例子，索引使用满足如下条件时：

- 在同一个表的同一个列上建立了多个GIN索引；
- 这些GIN索引使用了不同的parser（即分隔符不同）；
- 在查询中使用了该列，且执行计划中使用索引进行扫描；

  为了避免使用不同GIN索引导致查询结果不同的问题，需要保证在物理表的一列上只有一个GIN索引可用。

21.10.3 控制文本搜索

21.10.3.1 解析文档

DWS中提供了to_tsvector函数把文档处理成tsvector数据类型。

to_tsvector([*config* **regconfig**,] *document* **text**) **returns tsvector**

to_tsvector将文本文档解析为token，再将token简化到词素，并返回一个tsvector。其中tsvector中列出了词素及它们在文档中的位置。文档是根据指定的或默认的文本搜索分词器进行处理的。这里有一个简单的例子：

```
SELECT to_tsvector('english', 'a fat  cat sat on a mat - it ate a fat rats');
                  to_tsvector
```


版权所有 © 华为云计算技术有限公司

```
-----------------------------------------------------
 'ate':9 'cat':3 'fat':2,11 'mat':7 'rat':12 'sat':4
```

通过以上例子可发现结果tsvector不包含词a、on或者it，rats变成rat，并且忽略标点符号-。

to_tsvector函数内部调用一个解析器，将文档的文本分解成token并给每个token指定一个类型。对于每个token，有一系列词典可供查询。词典系列因token类型的不同而不同。识别token的第一本词典将发出一个或多个标准词素来表示token。例如：

- rats变成rat因为词典认为词rats是rat的复数形式。
- 有些词被作为停用词（请参考**停用词**），这样它们就会被忽略，因为它们出现得太过频繁以致于搜索中没有用处。比如例子中的a、on和it。
- 如果没有词典识别token，那么它也被忽略。在这个例子中，符号“-”被忽略，因为词典没有给它分配token类型（空间符号），即空间符号永远不会被索引。

语法解析器、词典和要索引的token类型由选定的文本搜索分词器决定。可以在同一个数据库中有多种不同的分词器，以及提供各种语言的预定义分词器。在以上例子中，使用缺省分词器english。

函数setweight可以给tsvector的记录加权重，权重是字母A、B、C、D之一。这通常用于标记来自文档不同部分的记录，比如标题、正文。之后，这些信息可以用于排序搜索结果。

因为to_tsvector(NULL)会返回空，当字段可能是空的时候，建议使用coalesce。以下是推荐的为结构化文档创建tsvector的方法：

```
CREATE TABLE tsearch.tt (id int, title text, keyword text, abstract text, body text, ti tsvector);

INSERT INTO tsearch.tt(id, title, keyword, abstract, body) VALUES (1, 'book', 'literature', 'Ancient
poetry','Tang poem Song jambic verse');

UPDATE tsearch.tt SET ti =
    setweight(to_tsvector(coalesce(title,'')), 'A')    ||
    setweight(to_tsvector(coalesce(keyword,'')), 'B')  ||
    setweight(to_tsvector(coalesce(abstract,'')), 'C') ||
    setweight(to_tsvector(coalesce(body,'')), 'D');
DROP TABLE tsearch.tt;
```

上例使用setweight标记已完成的tsvector中的每个词的来源，并且使用tsvector连接操作符||合并标记过的tsvector值，**处理tsvector**一节详细介绍了这些操作。

21.10.3.2 解析查询

DWS提供了函数to_tsquery和plainto_tsquery将查询转换为tsquery数据类型，to_tsquery提供比plainto_tsquery更多的功能，但对其输入要求更严格。

```
to_tsquery([ config regconfig, ] querytext text) returns tsquery
```

to_tsquery从querytext中创建一个tsquery，querytext必须由布尔运算符& (AND)，| (OR)和! (NOT)分割的单个token组成。这些运算符可以用圆括弧分组。换句话说，to_tsquery输入必须遵循tsquery输入的通用规则，具体请参见**文本搜索类型**。不同的是基本tsquery以token表面值作为输入，而to_tsquery使用指定或默认分词器将每个token标准化成词素，并依据分词器丢弃属于停用词的token。例如：

```
SELECT to_tsquery('english', 'The & Fat & Rats');
  to_tsquery
---------------
 'fat' & 'rat'
(1 row)
```


版权所有 © 华为云计算技术有限公司

像在基本tsquery中的输入一样，weight(s)可以附加到每个词素来限制它只匹配那些有相同weight(s)的tsvector词素。比如：

```
SELECT to_tsquery('english', 'Fat | Rats:AB');
    to_tsquery
------------------
 'fat' | 'rat':AB
(1 row)
```

同时，*也可以附加到词素来指定前缀匹配：

```
SELECT to_tsquery('supern:*A & star:A*B');
        to_tsquery
--------------------------
 'supern':*A & 'star':*AB
(1 row)
```

这样的词素将匹配tsquery中指定字符串和权重的项。

```
plainto_tsquery([ config regconfig, ] querytext text) returns tsquery
```

plainto_tsquery将未格式化的文本querytext变换为tsquery。类似于to_tsvector，文本被解析并且标准化，然后在存在的词之间插入&(AND)布尔算子。

比如：

```
SELECT plainto_tsquery('english', 'The Fat Rats');
 plainto_tsquery
-----------------
 'fat' & 'rat'
(1 row)
```

请注意，plainto_tsquery无法识别布尔运算符、权重标签，或在其输入中的前缀匹配标签：

```
SELECT plainto_tsquery('english', 'The Fat & Rats:C');
   plainto_tsquery
---------------------
 'fat' & 'rat' & 'c'
(1 row)
```

在这里，所有输入的标点符号作为空格符号丢弃。

21.10.3.3 排序查询结果

排序试图针对特定查询衡量文档的相关度，从而将众多的匹配文档中相关度最高的文档排在最前。DWS提供了两个预置的排序函数。函数考虑了词法，距离，和结构信息；也就是，考虑查询词在文档中出现的频率、紧密程度、以及出现的地方在文档中的重要性。然而，相关性的概念是模糊的，并且是跟应用强相关的。不同的应用程序可能需要额外的信息来排序，比如，文档的修改时间，内置的排序函数等。也可以开发自己的排序函数或者采用附加因素组合这些排序函数的结果来满足特定需求。

两个预置的排序函数：

```
ts_rank([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ]) returns float4
```

基于词素匹配率对vector进行排序：

```
ts_rank_cd([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ]) returns float4
```

该函数需要位置信息的输入。因此它不能在"剥离"tsvector值的情况下运行—它将总是返回零。

对于这两个函数，可选的weights参数提供给词加权重的能力，词的权重大小取决于所加的权值。权重阵列指定在排序时为每类词汇加多大的权重。

版权所有 © 华为云计算技术有限公司

```
{D-weight, C-weight, B-weight, A-weight}
```

如果没有提供weights，则使用缺省值：{0.1, 0.2, 0.4, 1.0}

通常的权重是用来标记文档特殊领域的词，如标题或最初的摘要，所以相对于文章主体中的词它们有着更高或更低的重要性。

由于较长的文档有更多的机会包含查询词，因此有必要考虑文档的大小。例如，包含有5个搜索词的一百字文档比包含有5个搜索词的一千字文档相关性更高。两个预置的排序函数都采用了一个整型的标准化选项来定义文档长度是否影响排序及如何影响。这个整型选项控制多个行为，所以它是一个屏蔽字：可以使用|指定一个或多个行为（例如，2|4）。

- 0（缺省）表示：跟长度大小没有关系
- 1 表示：排名（rank）除以(文档长度的对数+1)
- 2表示：排名除以文档的长度
- 4表示：排名除以两个扩展词间的调和平均距离。只能使用ts_rank_cd实现
- 8表示：排名除以文档中单独词的数量
- 16表示：排名除以单独词数量的对数+1
- 32表示：排名除以排名本身+1

当指定多个标志位时，会按照所列的顺序依次进行转换。

需要特别注意的是，排序函数不使用任何全局信息，所以不可能产生一个某些情况下需要的1%或100%的理想标准值。标准化选项32 (rank/(rank+1))可用于所有规模的从零到一之间的排序。需要注意的是：这只是一个表面变化，并不会影响搜索结果的排序。

下面是一个例子，仅选择排名前十的匹配：

由于SQL_ASCII的数据库编码格式不支持中文字符，请在Encoding为UTF8/GBK的数据库中执行以下示例：

```
SELECT id, title, ts_rank_cd(to_tsvector(body), query) AS rank
FROM tsearch.pgweb, to_tsquery('science') query
WHERE query @@ to_tsvector(body)
ORDER BY rank DESC
LIMIT 10;
 id |  title  | rank
----+---------+------
 11 | Philology |  .2
  2 | Mathematics |  .1
 12 | Geography |  .1
 13 | Computer science |  .1
(4 rows)
```

这是使用标准化排序的相同例子：

```
SELECT id, title, ts_rank_cd(to_tsvector(body), query, 32 /* rank/(rank+1) */ ) AS rank
FROM tsearch.pgweb, to_tsquery('science') query
WHERE  query @@ to_tsvector(body)
ORDER BY rank DESC
LIMIT 10;
 id |  title  |   rank
----+---------+----------
 11 | Philology | .166667
  2 | Mathematics | .0909091
 12 | Geography | .0909091
 13 | Computer science | .0909091
(4 rows)
```

下面是使用中文分词法排序查询的例子：

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE tsearch.ts_zhparser(id int, body text);
INSERT INTO tsearch.ts_zhparser VALUES(1, '排序');
INSERT INTO tsearch.ts_zhparser VALUES(2, '排序查询');
INSERT INTO tsearch.ts_zhparser VALUES(3, '查询排序');
--精确匹配
SELECT id, body, ts_rank_cd(to_tsvector('zhparser',body), query) AS rank FROM tsearch.ts_zhparser,
to_tsquery('排序') query WHERE query @@ to_tsvector(body);
 id | body | rank
----+------+------
  1 | 排序 |   .1
(1 row)

--模糊匹配
SELECT id, body, ts_rank_cd(to_tsvector('zhparser',body), query) AS rank FROM tsearch.ts_zhparser,
to_tsquery('排序') query WHERE query @@ to_tsvector('zhparser',body);
 id |   body   | rank
----+----------+------
  3 | 查询排序 |   .1
  1 | 排序     |   .1
  2 | 排序查询 |   .1
(3 rows)
```

排序要遍历每个匹配的tsvector，因此资源消耗多，可能会因为I/O限制导致排序慢。可是这是很难避免的，因为实际查询中通常会有大量的匹配。

21.10.3.4 高亮搜索结果

搜索结果的理想显示是：列出每篇文档中与搜索相关的部分，并标识为什么与查询相关。搜索引擎能够显示标识了搜索词的文档片段。DWS提供了函数ts_headline支持这部分功能。

```
ts_headline([ config regconfig, ] document text, query tsquery [, options text ]) returns text
```

ts_headline的输入是带有查询条件的文档，其返回文档中的摘录，在摘录中查询词是高亮显示的。用来解析文档的分词器由config参数指定。如果省略config，则使用default_text_search_config的值所指定的分词器。

指定options字符串时，需由一个或多个option=value对组成，且必须用逗号分隔。options可以是下面的选项：

- StartSel，StopSel：分隔文档中出现的查询词，以区别于其他摘录词。当包含有空格或逗号时，必须用双引号将字符串引起来。
- MaxWords，MinWords：定义摘录的最长和最短值。
- ShortWord：在摘录的开始和结束会丢弃此长度或更短的词。默认值3会消除常见的英语冠词。
- HighlightAll：布尔标志。如果为真，整个文档将作为摘录。忽略前面三个参数的值。
- MaxFragments：要显示的文本摘录或片段的最大数量。默认值0表示选择非片段的摘录生成方法。 大于0的值表示选择基于片段的摘录生成。此方法查找带有尽可能多查询词的文本片段，并显示查询词周围的上下文片段。因此，查询词临近每个片段的中间，且查询词两边都有词。每个片段至多有MaxWords，并且长度为ShortWord或更短的词在每一个片段开始和结束被丢弃。如果在文档中没有找到所有的查询词，则文档中开头将显示MinWords单片段。
- FragmentDelimiter：当有一个以上的片段时，通过该字符串分隔这些片段。

不声明选项时，采用下面的缺省值：

```
StartSel=<b>, StopSel=</b>,
MaxWords=35, MinWords=15, ShortWord=3, HighlightAll=FALSE,
MaxFragments=0, FragmentDelimiter=" ... "
```

 版权所有 © 华为云计算技术有限公司

比如：

```
SELECT ts_headline('english',
'The most common type of search
is to find all documents containing given query terms
and return them in order of their similarity to the
query.',
to_tsquery('english', 'query & similarity'));
                        ts_headline
------------------------------------------------------------
 containing given <b>query</b> terms
 and return them in order of their <b>similarity</b> to the
 <b>query</b>.
(1 row)

SELECT ts_headline('english',
'The most common type of search
is to find all documents containing given query terms
and return them in order of their similarity to the
query.',
to_tsquery('english', 'query & similarity'),
'StartSel = <, StopSel = >');
                        ts_headline
-------------------------------------------------------
 containing given <query> terms
 and return them in order of their <similarity> to the
 <query>.
(1 row)
```

ts_headline使用原始文档，而不是tsvector摘录，因此使用起来会慢，应慎重使用。

21.10.4 附加功能

21.10.4.1 处理 tsvector

DWS提供了用来操作tsvector类型的函数和操作符。

- tsvector || tsvector

  tsvector连接操作符返回一个新的tsvector类型，它综合了两个tsvector中词素和位置信息，并保留词素的位置信息和权重标签。右侧的tsvector的起始位置位于左侧tsvector的最后位置，因此，新生成的tsvector几乎等同于将两个原始文档字符串连接后进行to_tsvector操作。（这个等价是不准确的，因为任何从左边tsvector中删除的停用词都不会影响结果，但是，在使用文本连接时，则会影响词素在右侧tsvector中的位置。）

  相较于对文本进行连接后再执行to_tsvector操作，使用tsvector类型进行连接操作的优势在于，可以对文档的不同部分使用不同配置进行解析。因为setweight函数会对给定的tsvector中的语素进行统一设置，如果想要对文档的不同部分设置不同的权重，需要在连接之前对文本进行解析和权重设置。

- setweight(vector tsvector, weight "char") returns tsvector

  setweight返回一个输入tsvector的副本，其中每一个位置都使用给定的权重做了标记。权值可以为A、B、C或D（D是tsvector副本的默认权重，并且不在输出中呈现）。当对tsvector进行连接操作时，这些权重标签将会被保留，文档不同部分以不同的权重进行排序。

  须知

  权重标签作用于位置，而不是词素。如果传入的tsvector已经被剥离了位置信息，那么setweight函数将什么都不做。

版权所有 © 华为云计算技术有限公司

- length(vector tsvector) returns integer

  返回vector中的词素的数量。

- strip(vector tsvector) returns tsvector

  返回一个tsvector类型，其中包含输入的tsvector的同义词，但不包含任何位置和权重信息。虽然在相关性排序中，这里返回的tsvector要比未拆分的tsvector的作用小很多，但它通常都比未拆分的tsvector小的多。

21.10.4.2 处理查询

DWS提供了函数和操作符用来操作tsquery类型的查询。

- tsquery && tsquery

  返回两个给定查询tsquery的与结果。

- tsquery || tsquery

  返回两个给定查询tsquery的或结果。

- !! tsquery

  返回给定查询tsquery的非结果。

- numnode(query tsquery) returns integer

  返回tsquery中的节点数目（词素加操作符），这个函数在检查查询是否有效（返回值大于0），或者只包含停用词（返回值等于0）时，是有用的。例如：

```
SELECT numnode(plainto_tsquery('the any'));
NOTICE:  text-search query contains only stop words or doesn't contain lexemes, ignored
CONTEXT:  referenced column: numnode
 numnode
---------
       0

SELECT numnode('foo & bar'::tsquery);
 numnode
---------
       3
```

- querytree(query tsquery) returns text

  返回可用于索引搜索的tsquery部分，该函数对于检测非索引查询是有用的（例如只包含停用词或否定项）。例如:

```
SELECT querytree(to_tsquery('!defined'));
 querytree
-----------
 T
(1 row)
```

21.10.4.3 查询重写

ts_rewrite函数族可以从tsquery中搜索一个特定的目标子查询，并在该子查询每次出现的地方都替换为另一个子查询。 实际上这只是通过字符串替换而得到的一个特定tsquery版本。目标子查询和替换查询组合起来可以被认为是一个重写规则。一组类似的重写规则可以为搜索提供强大的帮助。例如，可以使用同义词扩大搜索范围（例如，new york, big apple, nyc, gotham）或限制搜索范围在用户直接感兴趣的热点话题上。

- ts_rewrite (query tsquery, target tsquery, substitute tsquery) returns tsquery

  ts_rewrite的这种形式只适用于一个单一的重写规则：任何出现目标子查询的地方都被无条件替换。例如：

 版权所有 © 华为云计算技术有限公司

```
SELECT ts_rewrite('a & b'::tsquery, 'a'::tsquery, 'c'::tsquery);
 ts_rewrite
------------
 'b' & 'c'
```

- ts_rewrite (query tsquery, select text) returns tsquery

  ts_rewrite的这种形式接受一个起始查询和SQL查询命令。 这里的查询命令是文本字符串形式，必须产生两个tsquery列。查询结果的每一行，第一个字段的值（目标子查询） 都会被第二个字段（替代子查询）替换。

  说明

  当多个规则需要重写时，重写顺序非常重要； 因此在实践中需要使用ORDER BY将源查询按照某些字段进行排序。

  例如：举一个现实生活中天文学上的例子。将使用表驱动的重写规则扩大supernovae的查询范围：

```
CREATE TABLE tsearch.aliases (id int, t tsquery, s tsquery);

INSERT INTO tsearch.aliases VALUES(1, to_tsquery('supernovae'), to_tsquery('supernovae|sn'));

SELECT ts_rewrite(to_tsquery('supernovae & crab'), 'SELECT t, s FROM tsearch.aliases');

          ts_rewrite
---------------------------------
 'crab' & ( 'supernova' | 'sn' )
```

  可以通过更新表修改重写规则：

```
UPDATE tsearch.aliases
SET s = to_tsquery('supernovae|sn & !nebulae')
WHERE t = to_tsquery('supernovae');

SELECT ts_rewrite(to_tsquery('supernovae & crab'), 'SELECT t, s FROM tsearch.aliases');

                ts_rewrite
-----------------------------------------------
 'crab' & ( 'supernova' | 'sn' & !'nebula' )
```

  需要重写的规则越多，重写操作就越慢。因为它要检查每一个可能匹配的规则。为了过滤明显的非候选规则，可以使用tsquery类型的操作符来实现。在下面的例子中， 只选择那些可能与原始查询匹配的规则：

```
SELECT ts_rewrite('a & b'::tsquery, 'SELECT t,s FROM tsearch.aliases WHERE ''a & b''::tsquery @> t');

 ts_rewrite
------------
 'b' & 'a'
(1 row)
DROP TABLE ts_rewrite;
```

21.10.4.4 收集文献统计

函数ts_stat可用于检查配置和查找候选停用词。

```
ts_stat(sqlquery text, [ weights text, ]
    OUT word text, OUT ndoc integer,
    OUT nentry integer) returns setof record
```

sqlquery是一个包含SQL查询语句的文本，该SQL查询将返回一个tsvector。ts_stat执行SQL查询语句并返回一个包含tsvector中每一个不同的语素（词）的统计信息。返回信息包括：

- word text：词素。
- ndoc integer：词素在文档（tsvector）中的编号。

版权所有 © 华为云计算技术有限公司

- nentry integer：词素出现的频率。

如果设置了权重条件，只有标记了对应权重的词素才会统计频率。例如，在一个文档集中检索使用频率最高的十个单词：

```
SELECT * FROM ts_stat('SELECT to_tsvector(''english'', sr_reason_sk) FROM tpcds.store_returns WHERE sr_customer_sk < 10') ORDER BY nentry DESC, ndoc DESC, word LIMIT 10;;
  word | ndoc | nentry
------+------+--------
 32   |    2 |      2
 33   |    2 |      2
 1    |    1 |      1
 10   |    1 |      1
 13   |    1 |      1
 14   |    1 |      1
 15   |    1 |      1
 17   |    1 |      1
 20   |    1 |      1
 22   |    1 |      1
(10 rows)
```

同样的情况，但是只计算权重为A或者B的单词使用频率：

```
SELECT * FROM ts_stat('SELECT to_tsvector(''english'', sr_reason_sk) FROM tpcds.store_returns WHERE sr_customer_sk < 10', 'a') ORDER BY nentry DESC, ndoc DESC, word LIMIT 10;
 word | ndoc | nentry
------+------+--------
(0 rows)
```

21.10.5 解析器

文本搜索解析器负责将原文档文本分解为多个token，并标识每个token的类型。这里的类型集由解析器本身定义。注意，解析器并不修改文本，它只是确定合理的单词边界。由于这一限制，人们更需要定制词典，而不是为每个应用程序定制解析器。

目前DWS提供了四个内置的解析器，分别为pg_catalog.default/pg_catalog.ngram/pg_catalog.zhparser/pg_catalog.pound，其中pg_catalog.default适用于英文分词场景，pg_catalog.ngram/pg_catalog.zhparser/pg_catalog.pound是为了支持中文全文检索功能新增的三种解析器，适用于中文及中英混合分词场景。

内置解析器pg_catalog.default，它能识别23种token类型，显示在**表21-94**中。

表 21-94 默认解析器类型

别名	描述	示例
asciiword	Word, all ASCII letters	elephant
word	Word, all letters	mañana
numword	Word, letters and digits	beta1
asciihword	Hyphenated word, all ASCII	up-to-date
hword	Hyphenated word, all letters	lógico-matemática
numhword	Hyphenated word, letters and digits	postgresql-beta1
hword_asciipart	Hyphenated word part, all ASCII	postgresql in the context postgresql-beta1

版权所有 © 华为云计算技术有限公司

别名	描述	示例
hword_part	Hyphenated word part, all letters	lógico or matemática in the context lógico-matemática
hword_numpart	Hyphenated word part, letters and digits	beta1 in the context postgresql-beta1
email	Email address	foo@example.com
protocol	Protocol head	http://
url	URL	example.com/stuff/index.html
host	Host	example.com
url_path	URL path	/stuff/index.html, in the context of a URL
file	File or path name	/usr/local/foo.txt, if not within a URL
sfloat	Scientific notation	-1.23E+56
float	Decimal notation	-1.234
int	Signed integer	-1234
uint	Unsigned integer	1234
version	Version number	8.3.0
tag	XML tag	<a href="dictionaries.html">
entity	XML entity	&
blank	Space symbols	(any whitespace or punctuation not otherwise recognized)

注意：对于解析器来说，一个“字母”的概念是由数据库的语言区域设置，即lc_ctype设置决定的。只包含基本ASCII字母的词被报告为一个单独的token类型，因为这类词有时需要被区分出来。大多数欧洲语言中，对token类型word和asciiword的处理方法是类似的。

email不支持某些由RFC 5322定义的有效电子邮件字符。具体来说，可用于email用户名的非字母数字字符仅包含句号、破折号和下划线。

解析器可能对同一内容进行重叠token。例如，包含连字符的单词将作为一个整体进行报告，其组件也会分别被报告：

```
SELECT alias, description, token FROM ts_debug('english','foo-bar-beta1');
      alias      |              description               |     token
-----------------+----------------------------------------+---------------
 numhword        | Hyphenated word, letters and digits    | foo-bar-beta1
 hword_asciipart | Hyphenated word part, all ASCII        | foo
 blank           | Space symbols                          | -
 hword_asciipart | Hyphenated word part, all ASCII        | bar
```


版权所有 © 华为云计算技术有限公司

```
blank           | Space symbols                          | -
hword_numpart   | Hyphenated word part, letters and digits | beta1
```

这种行为是有必要的，因为它支持搜索整个复合词和各组件。这里是另一个例子：

```
SELECT alias, description, token FROM ts_debug('english','http://example.com/stuff/index.html');
  alias   |  description  |            token
----------+---------------+------------------------------
 protocol | Protocol head | http://
 url      | URL           | example.com/stuff/index.html
 host     | Host          | example.com
 url_path | URL path      | /stuff/index.html
```

N-gram是一种机械分词方法，适用于无语义中文分词场景。N-gram分词法可以保证分词的完备性，但是为了照顾所有可能，把很多不必要的词也加入到索引中，导致索引项增加。N-gram支持中文编码包括GBK、UTF-8。内置6种token类型，如表21-95所示。

表 21-95 token 类型

Alias	Description
zh_words	chinese words
en_word	english word
numeric	numeric data
alnum	alnum string
grapsymbol	graphic symbol
multisymbol	multiple symbol

Zhparser是基于词典的语义分词方法，底层调用SCWS(https://github.com/hightman/scws)分词算法，适用于有语义的中文分词场景。SCWS是一套基于词频词典的机械式中文分词引擎，可以将一整段的中文文本正确地切分成词。支持GBK、UTF-8两种中文编码格式。内置26种token类型如表21-96所示：

表 21-96 token 类型

Alias	Description
A	形容词
B	区别词
C	连词
D	副词
E	叹词
F	方位词
G	语素
H	前接成分

版权所有 © 华为云计算技术有限公司

Alias	Description
I	成语
J	简称略语
K	后接成分
L	习用语
M	数词
N	名词
O	拟声词
P	介词
Q	量词
R	代词
S	处所词
T	时间词
U	助词
V	动词
W	标点符号
X	未知词
Y	语气词
Z	状态词

Pound是一种固定格式分词方法，适用于无语意但待解析文本以固定分隔符分割开来的中英文分词场景。支持中文编码包括GBK、UTF8，支持英文编码包括ASCII。内置6种token类型，如**表4 token类型**所示；支持5种分隔符，如**表21-98**所示，在用户不进行自定义设置的情况下分隔符默认为“#”。Pound限制单个token长度不能超过256个字符。

表 21-97 token 类型

Alias	Description
zh_words	chinese words
en_word	english word
numeric	numeric data
alnum	alnum string
grapsymbol	graphic symbol

版权所有 © 华为云计算技术有限公司

Alias	Description
multisymbol	multiple symbol

表 21-98 分隔符类型

分隔符	描述
@	Special character
#	Special character
$	Special character
%	Special character
/	Special character

21.10.6 词典

21.10.6.1 词典概述

词典用于定义停用词（stop words），即全文检索时不搜索哪些词。

词典还可以用于对同一词的不同形式进行规范化，这样同一个词的不同派生形式都可以进行匹配。规范化后的词称为词位（lexeme）。

除了提高检索质量外，词的规范化和删除停用词可以减少文档tsvector格式的大小，从而提高性能。词的规范化和删除停用词并不总是具有语言学意义，用户可以根据应用环境在词典定义文件中自定义规范化和删除规则。

一个词典是一个程序，接收标记（token）作为输入，并返回：

- 如果token在词典中已知，返回对应lexeme数组（注意，一个标记可能对应多个lexeme）。
- 一个lexeme。一个新token会代替输入token被传递给后继词典（当前词典可被称为过滤词典）。
- 如果token在词典中已知，但它是一个停用词，返回空数组。
- 如果词典不能识别输入的token，返回NULL。

DWS提供了多种语言的预定义词典，同时提供了五种预定义的词典模板，分别是Simple，Synonym，Thesaurus，Ispell，和Snowball，可用于创建自定义参数的新词典。

在使用全文检索时，建议用户：

- 可以在文本搜索配置中定义一个解析器，以及一组用于处理该解析器的输出标记词典。对于解析器返回的每个标记类型，可以在配置中指定不同的词典列表进行处理。当解析器输出一种类型的标记后，在对应列表的每个词典中会查阅该标记，直到某个词典识别它。如果它被识别为一个停用词， 或者没有任何词典识别，该token将被丢弃，即不被索引或检索到。通常情况下，第一个返回非空结果

版权所有 © 华为云计算技术有限公司

的词典决定了最终结果，后继词典将不会继续处理。但是一个过滤类型的词典可以依据规则替换输入token，然后将替换后的token传递给后继词典进行处理。

- 配置词典列表的一般规则是，第一个位置放置一个应用范围最小的、最具体化定义的词典，其次是更一般化定义的词典， 最后是一个普适定义的词典，比如Snowball词干词典或Simple词典。在下面例子中，对于一个针对天文学的文本搜索配置astro_en，可以定义标记类型asciiword（ASCII词）对应的词典列表为：天文术语的Synonym同义词词典， Ispell英语词典和Snowball英语词干词典。

  ```
  ALTER TEXT SEARCH CONFIGURATION astro_en
    ADD MAPPING FOR asciiword WITH astro_syn, english_ispell, english_stem;
  ```

  过滤类型的词典可以放置在词典列表中除去末尾的任何地方，放置在末尾时是无效的。使用这些词典对标记进行部分规范化，可以有效简化后继词典的处理。

21.10.6.2 停用词

停用词是很常见的词，几乎出现在每一个文档中，并且没有区分值。因此，在全文搜索的语境下可忽视它们。停用词处理逻辑和词典类型相关。例如，Ispell词典会先对标记进行规范化，然后再查看停用词表，而Snowball词典会最先检查输入标记是否为停用词。

例如，每个英文文本包含像a和the的单词，因此没必要将它们存储在索引中。然而，停用词影响tsvector中的位置，同时位置也会影响相关度：

```
SELECT to_tsvector('english','in the list of stop words');
        to_tsvector
----------------------------
 'list':3 'stop':5 'word':6
```

位置1、2、4是停用词，所以不显示。为包含和不包含停用词的文档计算出的排序是完全不同的：

```
SELECT ts_rank_cd (to_tsvector('english','in the list of stop words'), to_tsquery('list & stop'));
 ts_rank_cd
------------
        .05

SELECT ts_rank_cd (to_tsvector('english','list stop words'), to_tsquery('list & stop'));
 ts_rank_cd
------------
         .1
```

21.10.6.3 Simple 词典

Simple词典首先将输入标记转换为小写字母，然后检查停用词表。如果识别为停用词则返回空数组，即表示该标记会被丢弃。否则，输入标记的小写形式作为规范化后的lexeme返回。此外，Simple词典可通过设置参数Accept为false（默认值true），将非停用词报告为未识别，传递给后继词典继续处理。

注意事项

- 大多数词典的功能依赖于词典定义文件，词典定义文件名仅支持小写字母、数字、下划线组合。
- 临时模式pg_temp下不允许创建词典。
- 词典定义文件的字符集编码必须为UTF-8格式。实际应用时，如果与数据库的字符编码格式不一致，在读入词典定义文件时会进行编码转换。
- 通常情况下，每个session仅读取词典定义文件一次，当且仅当在第一次使用该词典时。需要修改词典文件时，可通过ALTER TEXT SEARCH DICTIONARY命令进行词典定义文件的更新和重新加载。

 版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 创建Simple词典。

```
CREATE TEXT SEARCH DICTIONARY public.simple_dict (
    TEMPLATE = pg_catalog.simple,
    STOPWORDS = english
);
```

其中，停用词表文件全名为english.stop。关于创建simple词典的语法和更多参数，请参见**CREATE TEXT SEARCH DICTIONARY**。

步骤2 使用Simple词典。

```
SELECT ts_lexize('public.simple_dict','Yes');
 ts_lexize
-----------
 {yes}
(1 row)

SELECT ts_lexize('public.simple_dict','The');
 ts_lexize
-----------
 {}
(1 row)
```

步骤3 设置参数ACCEPT=false，使Simple词典返回NULL，而不是返回非停用词的小写形式。

```
ALTER TEXT SEARCH DICTIONARY public.simple_dict ( Accept = false );
SELECT ts_lexize('public.simple_dict','Yes');
 ts_lexize
-----------

(1 row)

SELECT ts_lexize('public.simple_dict','The');
 ts_lexize
-----------
 {}
(1 row)
```

----结束

21.10.6.4 Synonym 词典

Synonym词典用于定义、识别token的同义词并转化，不支持词组（词组形式的同义词可用Thesaurus词典定义，详细请参见**Thesaurus词典**）。

示例

- Synonym词典可用于解决语言学相关问题，例如，为避免使单词"Paris"变成"pari"，可在Synonym词典文件中定义一行"Paris paris"，并将该词典放置在预定义的english_stem词典之前。

```
SELECT * FROM ts_debug('english', 'Paris');
   alias   |   description   | token |  dictionaries  |  dictionary  | lexemes
-----------+-----------------+-------+----------------+--------------+---------
 asciiword | Word, all ASCII | Paris | {english_stem} | english_stem | {pari}
(1 row)

CREATE TEXT SEARCH DICTIONARY my_synonym (
    TEMPLATE = synonym,
    SYNONYMS = my_synonyms,
    FILEPATH =  'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=xx-xx-xx'
);
```


版权所有 © 华为云计算技术有限公司

```
ALTER TEXT SEARCH CONFIGURATION english
    ALTER MAPPING FOR asciiword
    WITH my_synonym, english_stem;

SELECT * FROM ts_debug('english', 'Paris');
   alias   |   description   | token |       dictionaries        | dictionary | lexemes
-----------+-----------------+-------+---------------------------+------------+---------
 asciiword | Word, all ASCII | Paris | {my_synonym,english_stem} | my_synonym | {paris}
(1 row)

SELECT * FROM ts_debug('english', 'paris');
   alias   |   description   | token |       dictionaries        | dictionary | lexemes
-----------+-----------------+-------+---------------------------+------------+---------
 asciiword | Word, all ASCII | Paris | {my_synonym,english_stem} | my_synonym | {paris}
(1 row)

ALTER TEXT SEARCH DICTIONARY my_synonym ( CASESENSITIVE=true);

SELECT * FROM ts_debug('english', 'Paris');
   alias   |   description   | token |       dictionaries        | dictionary | lexemes
-----------+-----------------+-------+---------------------------+------------+---------
 asciiword | Word, all ASCII | Paris | {my_synonym,english_stem} | my_synonym | {paris}
(1 row)

SELECT * FROM ts_debug('english', 'paris');
   alias   |   description   | token |       dictionaries        | dictionary | lexemes
-----------+-----------------+-------+---------------------------+------------+---------
 asciiword | Word, all ASCII | Paris | {my_synonym,english_stem} | my_synonym | {pari}
(1 row)
```

其中，同义词词典文件全名为my_synonyms.syn，所在目录为 'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=*xx-xx-xx*'。关于创建词典的语法和更多参数，请参见**CREATE TEXT SEARCH DICTIONARY**。

- 星号（*）可用于词典文件中的同义词结尾，表示该同义词是一个前缀。在to_tsvector()中该星号将被忽略，但在to_tsquery()中会匹配该前缀并对应输出结果（参照**处理查询**一节）。

  假设词典文件synonym_sample.syn内容如下：

  ```
  postgres        pgsql
  postgresql      pgsql
  postgre pgsql
  gogle   googl
  indices index*
  ```

  创建并使用词典：

  ```
  CREATE TEXT SEARCH DICTIONARY syn (
      TEMPLATE = synonym,
      SYNONYMS = synonym_sample
  );

  SELECT ts_lexize('syn','indices');
   ts_lexize
  -----------
   {index}
  (1 row)

  CREATE TEXT SEARCH CONFIGURATION tst (copy=simple);

  ALTER TEXT SEARCH CONFIGURATION tst ALTER MAPPING FOR asciiword WITH syn;

  SELECT to_tsvector('tst','indices');
   to_tsvector
  -------------
   'index':1
  (1 row)

  SELECT to_tsquery('tst','indices');
   to_tsquery
  ```


版权所有 © 华为云计算技术有限公司

```
------------
 'index':*
(1 row)

SELECT 'indexes are very useful'::tsvector;
            tsvector
---------------------------------
 'are' 'indexes' 'useful' 'very'
(1 row)

SELECT 'indexes are very useful'::tsvector @@ to_tsquery('tst','indices');
 ?column?
----------
 t
(1 row)
```

21.10.6.5 Thesaurus 词典

Thesaurus词典，也叫做分类词典（缩写为TZ），是一组定义了词以及词组间关系的集合，包括广义词（BT）、狭义词（NT）、首选词、非首选词、相关词等。根据词典文件中的定义，TZ词典用一个指定的短语替换对应匹配的所有短语，并且可选择保留原始短语进行索引。TZ词典实际上是Synonym词典的一个扩展，增加了短语支持。

注意事项

- 由于TZ词典需要识别短语，所以在处理过程中必须保存当前状态并与解析器进行交互，以决定是否处理下一个token或是结束当前识别。此外，TZ词典配置时需谨慎，如果设置TZ词典仅处理asciiword类型的token，则类似one 7的分类词典定义将不会生效，因为uint类型的token不会传给TZ词典处理。
- 在索引期间要用到分类词典，因此分类词典参数中的任何变化都要求重新索引。对于其他大多数类型的词典来说，类似添加或删除停用词这种修改并不需要强制重新索引。

操作步骤

步骤1 创建一个名为thesaurus_astro的TZ词典。

以一个简单的天文学词典thesaurus_astro为例，其中定义了两组天文短语及其同义词如下：

```
supernovae stars : sn
crab nebulae : crab
```

执行如下语句创建TZ词典：

```
CREATE TEXT SEARCH DICTIONARY thesaurus_astro (
    TEMPLATE = thesaurus,
    DictFile = thesaurus_astro,
    Dictionary = pg_catalog.english_stem,
    FILEPATH = 'obs://bucket_name/path accesskey=ak secretkey=sk region=rg'
);
```

其中，词典定义文件全名为thesaurus_astro.ths，所在目录为 "obs://bucket_name/path accesskey=ak secretkey=sk region=rg"。子词典pg_catalog.english_stem是预定义的Snowball类型的英语词干词典，用于规范化输入词，子词典自身相关配置（例如停用词等）不在此处显示。关于创建词典的语法和更多参数，请参见**CREATE TEXT SEARCH DICTIONARY**。

步骤2 创建词典后，将其绑定到对应文本搜索配置中需要处理的token类型上：

版权所有 © 华为云计算技术有限公司

```
ALTER TEXT SEARCH CONFIGURATION english
    ALTER MAPPING FOR asciiword, asciihword, hword_asciipart
    WITH thesaurus_astro, english_stem;
```

步骤3 使用TZ词典。

- 测试TZ词典。

  ts_lexize函数对于测试TZ词典作用不大，因为该函数是按照单个token处理输入。可以使用plainto_tsquery、to_tsvector、to_tsquery函数测试TZ词典，这些函数能够将输入分解成多个token（ to_tsquery函数需要将输入加上引号 ）。

  ```
  SELECT plainto_tsquery('english','supernova star');
   plainto_tsquery
  -----------------
   'sn'
  (1 row)

  SELECT to_tsvector('english','supernova star');
   to_tsvector
  -------------
   'sn':1
  (1 row)

  SELECT to_tsquery('english','''supernova star''');
   to_tsquery
  ------------
   'sn'
  (1 row)
  ```

  其中，supernova star匹配了词典thesaurus_astro定义中的supernovae stars，这是因为在thesaurus_astro词典定义中指定了Snowball类型的子词典english_stem，该词典移除了e和s。

- 如果同时需要索引原始短语，只要将其同时放置在词典定义文件中对应定义的右侧即可，如下：

  ```
  supernovae stars : sn supernovae stars

  ALTER TEXT SEARCH DICTIONARY thesaurus_astro (
      DictFile = thesaurus_astro,
      FILEPATH = 'file:///home/dicts/');

  SELECT plainto_tsquery('english','supernova star');
        plainto_tsquery
  ----------------------------
   'sn' & 'supernova' & 'star'
  (1 row)
  ```

----结束

21.10.6.6 Ispell 词典

Ispell词典模板支持词法词典，它可以把一个词的各种语言学形式规范化成相同的词位。比如，一个Ispell英语词典可以匹配搜索词bank的词尾变化和词形变化，如banking、banked、banks、banks'和bank's等。

DWS不提供任何预定义的Ispell类型词典或词典文件。dict文件和affix文件支持多种开源词典格式，包括Ispell、MySpell和Hunspell等。

操作步骤

步骤1 获取词典定义文件和词缀文件。

用户可以使用开源词典，直接获取的开源词典后缀名可能为.aff和.dic，此时需要将扩展名改为.affix和.dict。此外，对于某些词典文件，还需要使用下面的命令把字符转换成UTF-8编码，比如挪威语词典：

 版权所有 © 华为云计算技术有限公司

```
iconv -f ISO_8859-1 -t UTF-8 -o nn_no.affix nn_NO.aff
iconv -f ISO_8859-1 -t UTF-8 -o nn_no.dict nn_NO.dic
```

步骤2 创建Ispell词典。

```
CREATE TEXT SEARCH DICTIONARY norwegian_ispell (
    TEMPLATE = ispell,
    DictFile = nn_no,
    AffFile = nn_no,
    FilePath =  'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=xx-xx-xx'
);
```

其中，词典文件全名为nn_no.dict和nn_no.affix，所在目录为 'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=*xx-xx-xx*'。关于创建词典的语法和更多参数，请参见**CREATE TEXT SEARCH DICTIONARY**。

步骤3 使用Ispell词典进行复合词拆分。

```
SELECT ts_lexize('norwegian_ispell', 'sjokoladefabrikk');
      ts_lexize
----------------------
 {sjokolade,fabrikk}
(1 row)
```

MySpell不支持复合词，Hunspell对复合词有较好的支持。DWS仅支持Hunspell中基本的复合词操作。通常情况下，Ispell词典能够识别的词是一个有限集合，其后应该配置一个更广义的词典，例如一个可以识别所有词的Snowball词典。

----结束

21.10.6.7 Snowball 词典

Snowball词典模板支持词干分析词典，基于Martin Porter的Snowball项目，内置有许多语言的词干分析算法。DWS中预定义有多种语言的Snowball词典，可通过系统表PG_TS_DICT查看预定义的词干分析词典以及支持的语言词干分析算法。

无论是否可以简化，Snowball词典将标识所有输入为已识别，因此它应当被放置在词典列表的最后。把Snowball词典放在任何其他词典前面会导致后继词典失效，因为输入token不会通过Snowball词典进入到下一个词典。

关于Snowball词典的语法，请参见**CREATE TEXT SEARCH DICTIONARY**。

21.10.7 配置示例

文本搜索配置（Text Search Configuration），指定了将文档转换成tsvector过程中所必需的组件：

- 解析器，用于把文本分解成标记token；
- 词典列表，用于将每个token转换成词位lexeme。

每次to_tsvector或to_tsquery函数调用时，都需要指定一个文本搜索配置来指定具体的处理过程。GUC参数default_text_search_config指定了默认的文本搜索配置，当文本搜索函数中没有显式指定文本搜索配置参数时，将会使用该默认值进行处理。

DWS中预定义有一些可用的文本搜索配置，用户也可创建自定义的文本搜索配置。此外，为了便于管理文本搜索对象，还提供有多个gsql元命令，可以显示有关文本搜索对象的信息，详细请参见《数据仓库服务工具指南》中“元命令参考”章节。

版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 创建一个文本搜索配置ts_conf，复制预定义的文本搜索配置english。

```
CREATE TEXT SEARCH CONFIGURATION ts_conf ( COPY = pg_catalog.english );
CREATE TEXT SEARCH CONFIGURATION
```

步骤2 创建Synonym词典。

假设同义词词典定义文件pg_dict.syn内容如下：

```
postgres    pg
pgsql       pg
postgresql  pg
```

执行如下语句创建Synonym词典：

```
CREATE TEXT SEARCH DICTIONARY pg_dict (
    TEMPLATE = synonym,
    SYNONYMS = pg_dict,
    FILEPATH =  'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=xx-xx-xx'
);
```

步骤3 创建一个Ispell词典english_ispell（词典定义文件来自开源词典）。

```
CREATE TEXT SEARCH DICTIONARY english_ispell (
   TEMPLATE = ispell,
   DictFile = english,
   AffFile = english,
   StopWords = english,
   FILEPATH =  'obs://bucket01/obs.example.com accesskey=xxxxx secretkey=xxxxx region=xx-xx-xx'
);
```

步骤4 设置文本搜索配置ts_conf，修改某些类型的token对应的词典列表。关于token类型的详细信息，请参见**解析器**。

```
ALTER TEXT SEARCH CONFIGURATION ts_conf
   ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                     word, hword, hword_part
   WITH pg_dict, english_ispell, english_stem;
```

步骤5 在文本搜索配置中，选择设置不索引或搜索某些token类型。

```
ALTER TEXT SEARCH CONFIGURATION ts_conf
   DROP MAPPING FOR email, url, url_path, sfloat, float;
```

步骤6 使用文本检索调测函数ts_debug()对所创建的词典配置ts_conf进行测试。

```
SELECT * FROM ts_debug('ts_conf', '
PostgreSQL, the highly scalable, SQL compliant, open source object-relational
database management system, is now undergoing beta testing of the next
version of our software.
');
```

步骤7 可以设置当前session使用ts_conf作为默认的文本搜索配置。此设置仅在当前session有效。

```
\dF+ ts_conf
     Text search configuration "public.ts_conf"
Parser: "pg_catalog.default"
     Token      |            Dictionaries
-----------------+------------------------------------
 asciihword      | pg_dict,english_ispell,english_stem
 asciiword       | pg_dict,english_ispell,english_stem
 file            | simple
 host            | simple
 hword           | pg_dict,english_ispell,english_stem
 hword_asciipart | pg_dict,english_ispell,english_stem
 hword_numpart   | simple
 hword_part      | pg_dict,english_ispell,english_stem
 int             | simple
 numhword        | simple
 numword         | simple
```

 版权所有 © 华为云计算技术有限公司

```
 uint            | simple
 version         | simple
 word            | pg_dict,english_ispell,english_stem

SET default_text_search_config = 'public.ts_conf';
SET
SHOW default_text_search_config;
 default_text_search_config
----------------------------
 public.ts_conf
(1 row)
```

----结束

21.10.8 测试和调试文本搜索

21.10.8.1 分词器测试

函数ts_debug允许简单测试文本搜索分词器。

```
ts_debug([ config regconfig, ] document text,
        OUT alias text,
        OUT description text,
        OUT token text,
        OUT dictionaries regdictionary[],
        OUT dictionary regdictionary,
        OUT lexemes text[])
        returns setof record
```

ts_debug显示document的每个token信息，token是由解析器生成，由指定的词典进行处理。如果忽略对应参数，则使用config指定的分词器或者default_text_search_config指定的分词器。

ts_debug为文本解析器标识的每个token返回一行记录。记录中的列分别是：

- alias：text类型，token的别名。
- description：text类型，token的描述。
- token：text类型，token的文本内容。
- dictionaries：regdictionary数组类型，是分词器为token选定的词典。
- dictionary：regdictionary类型，用来识别token的词典。如果为空，则不做识别。
- lexemes：text数组类型，词典识别token时生成的词素。如果为空，则不生成词素。空数组（{}）意味着token将被识别成停用词。

一个简单的例子：

```
SELECT * FROM ts_debug('english','a fat  cat sat on a mat - it ate a fat rats');
   alias   |   description   | token |  dictionaries  |  dictionary  | lexemes
-----------+-----------------+-------+----------------+--------------+---------
 asciiword | Word, all ASCII | a     | {english_stem} | english_stem | {}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | fat   | {english_stem} | english_stem | {fat}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | cat   | {english_stem} | english_stem | {cat}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | sat   | {english_stem} | english_stem | {sat}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | on    | {english_stem} | english_stem | {}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | a     | {english_stem} | english_stem | {}
 blank     | Space symbols   |       | {}             |              |
```


版权所有 © 华为云计算技术有限公司

```
 asciiword | Word, all ASCII | mat   | {english_stem} | english_stem | {mat}
 blank     | Space symbols   |       | {}             |              |
 blank     | Space symbols   | -     | {}             |              |
 asciiword | Word, all ASCII | it    | {english_stem} | english_stem | {}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | ate   | {english_stem} | english_stem | {ate}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | a     | {english_stem} | english_stem | {}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | fat   | {english_stem} | english_stem | {fat}
 blank     | Space symbols   |       | {}             |              |
 asciiword | Word, all ASCII | rats  | {english_stem} | english_stem | {rat}
(24 rows)
```

21.10.8.2 解析器测试

函数ts_parse可以直接测试文本搜索解析器。

```
ts_parse(parser_name text, document text,
        OUT tokid integer, OUT token text) returns setof record
```

ts_parse解析指定的document并返回一系列的记录，一条记录代表一个解析生成的token。每条记录包括标识token类型的tokid，及token文本。比如：

```
SELECT * FROM ts_parse('default', '123 - a number');
 tokid | token
-------+--------
    22 | 123
    12 |
    12 | -
     1 | a
    12 |
     1 | number
(6 rows)
ts_token_type(parser_name text, OUT tokid integer,
              OUT alias text, OUT description text) returns setof record
```

ts_token_type返回一个表，这个表描述了指定解析器可以识别的每种token类型。对于每个token类型，表中给出了整数类型的tokid--用于解析器标记对应的token类型；alias——命名分词器命令中的token类型；及简单描述。比如：

```
SELECT * FROM ts_token_type('default');
 tokid |      alias      |               description
-------+-----------------+------------------------------------------
     1 | asciiword       | Word, all ASCII
     2 | word            | Word, all letters
     3 | numword         | Word, letters and digits
     4 | email           | Email address
     5 | url             | URL
     6 | host            | Host
     7 | sfloat          | Scientific notation
     8 | version         | Version number
     9 | hword_numpart   | Hyphenated word part, letters and digits
    10 | hword_part      | Hyphenated word part, all letters
    11 | hword_asciipart | Hyphenated word part, all ASCII
    12 | blank           | Space symbols
    13 | tag             | XML tag
    14 | protocol        | Protocol head
    15 | numhword        | Hyphenated word, letters and digits
    16 | asciihword      | Hyphenated word, all ASCII
    17 | hword           | Hyphenated word, all letters
    18 | url_path        | URL path
    19 | file            | File or path name
    20 | float           | Decimal notation
    21 | int             | Signed integer
    22 | uint            | Unsigned integer
    23 | entity          | XML entity
(23 rows)
```


版权所有 © 华为云计算技术有限公司

21.10.8.3 词典测试

函数ts_lexize用于进行词典测试。

ts_lexize(dict regdictionary, token text) returns text[]如果输入的token可以被词典识别，那么ts_lexize返回词素的数组；如果token可以被词典识别但它是一个停用词，则返回空数组；如果是一个不可识别的词则返回NULL。

比如：

```
SELECT ts_lexize('english_stem', 'stars');
 ts_lexize
-----------
 {star}

SELECT ts_lexize('english_stem', 'a');
 ts_lexize
-----------
 {}
```

须知

ts_lexize函数支持单一token，不支持文本。

21.10.9 限制约束

DWS的全文检索功能当前限制约束是：

- 每个分词长度必须小于2K字节。
- tsvector结构（分词+位置）的长度必须小于1兆字节。
- tsvector的位置值必须大于0，小于等于16,383。
- 每个分词在文档中位置数必须小于256，若超过将舍弃后面的位置信息。
- tsquery中的关键字及对应运算符最大支持到32,768。

21.11 系统操作

DWS通过SQL语句执行不同的系统操作，比如：设置变量，显示执行计划和垃圾收集等操作。

设置变量

设置会话或事务中需要使用的各种参数，请参考**SET**。

显示执行计划

显示DWS为SQL语句规划的执行计划，请参考**EXPLAIN**。

事务日志检查点

预写式日志（WAL）缺省时在事务日志中每隔一段时间放置一个检查点。CHECKPOINT强制立即进行检查，而不是等到下一次调度时的检查点。请参考**CHECKPOINT**。

版权所有 © 华为云计算技术有限公司

垃圾收集

进行垃圾收集以及可选择的对数据库进行分析。请参考**VACUUM**。

收集统计信息

收集与数据库中表内容相关的统计信息。请参考**ANALYZE | ANALYSE**。

设置当前事务的约束检查模式

设置当前事务里的约束检查的特性。请参考**SET CONSTRAINTS**。

21.12 事务管理

DWS支持数据库事务ACID属性，提供了事务的读已提交隔离级别和可重复读隔离级别。

警告

- 客户端显式开启事务或手动关闭Autocommit场景，业务最后必须手动执行Commit提交事务。
- 针对执行时长超过30分钟的语句，建议进行优化。
- **尽量避免**出现执行时长超过2小时的业务，避免长事务、长持锁等影响。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

事务的概念

- **事务**指一个操作，由多个步骤组成，要么全部成功，要么全部失败。
- **数据库事务**（TRANSACTION）是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成（通常由事务开始与事务结束之间执行的全部数据库操作组成），这些操作要么全部执行，要么全部不执行，是一个不可分割的执行单位。

事务的作用

数据库事务的目的主要是：

- 为数据库操作序列提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。
- 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。

事务的执行过程

当事务被提交给数据库管理系统（DBMS）后，DBMS需要确保该事务中的所有操作都成功完成，并且其结果被永久保存在数据库中。如果事务中有操作没有成功完成，则事务中的所有操作都需要回滚，回到事务执行前的状态；同时，该事务对数据库或者其他事务的执行无影响，所有的事务互不干扰和影响，好像在独立的环境中运行。

版权所有 © 华为云计算技术有限公司

事务的属性

事务具有以下四个标准属性，通常根据首字母缩写为ACID。

- Atomicity（原子性）：事务中的全部操作在数据库中是不可分割的，整个事务中的所有操作要么全部完成，要么全部失败，对于一个事务来说，不能只执行其中的一部分操作。

  比如： A给B转账，A扣除500元 ，B增加500元。整个事务的操作要么全部成功，要么全部失败，不能出现A扣钱，但是B不增加的情况。如果原子性不能保证，就会出现一致性问题。

- Consistency（一致性）：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的数据必须完全符合所有的预设规则，这包含数据的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

  比如：A给B转账，A扣除500元 ，B增加500元，扣除的钱-500与增加的钱+500，相加应该为0。如从A账户转账500元到B账户，不管操作成功与否，A和B的存款总额是不变的。

- Isolation（隔离性）：一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

  数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（serializable）。

- Durability（持久性）：一旦事务提交，则其所做的修改就会永久保存到数据库中。即使系统故障，已经提交的修改数据也不会丢失。

表 21-99 ACID 用途

ACID	属性	用途
Atomicity	原子性	并发控制，故障恢复。
Consistency	一致性	SQL的完整性约束（主键约束、外键约束）。
Isolation	隔离性	并发控制。
Durability	持久性	故障恢复。

常用的并发控制技术有基于锁的并发控制和基于时间戳的并发控制，DWS数据库针对DDL语句采用两阶段锁技术，而针对DML语句则采用多版本控制技术（Multi-Version Concurrency Control，MVCC）。DWS数据库的故障恢复采用WAL日志的方式来实现，目前主要支持Redo日志，通过Redo日志和MVCC可以保证事务读写的一致性。

Redo日志（重做日志）是数据库系统中用于保证数据持久性和故障恢复的核心机制之一，是数据库中用于故障恢复的关键日志文件，它记录了数据库事务对数据所做的“修改操作本身”（而非修改前后的数据），目的是在数据库发生故障（如宕机、崩溃）后，能够重新执行这些操作，将数据恢复到故障发生前的一致状态。通过Redo日志恢复过程可参见Redo恢复过程。

Redo日志的恢复过程：数据库崩溃重启后，通过读取预写日志记录(WAL日志)，从最近一个检查点(Checkpoint)（类似“存档点”）开始，扫描所有已提交但未持久化的

 版权所有 © 华为云计算技术有限公司

操作日志，重新执行这些操作（即“前滚（Redo）”），将内存丢失的数据变更重新写入磁盘；同时，对于未提交的事务，则通过逆向操作回滚其修改（即“后滚（Undo）”），最终确保数据恢复到崩溃前一致且完整的状态，既保证已提交操作的持久性，又维持未提交操作的原子性。这一机制是数据库实现高可靠性的核心基础。

隔离级别

Isolation（隔离性）可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离级别，决定多个事务并发操作同一个对象时的处理方式。

DWS的事务隔离级别，由GUC参数transaction_isolation或**SET TRANSACTION**语法设置，支持以下隔离级别，默认为读已提交（read committed）。

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。
- read uncommitted：读未提交隔离级别，DWS不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
- serializable：事务可序列化，DWS不支持serializable，如果设置了serializable，实际上使用的是repeatable read。

事务控制语法

- 启动事务

  DWS通过START TRANSACTION和BEGIN语法启动事务，请参考**START TRANSACTION**和**BEGIN**。

- 设置事务

  DWS通过SET TRANSACTION或者SET LOCAL TRANSACTION语法设置事务，请参考**SET TRANSACTION**。

- 提交事务

  DWS通过COMMIT或者END可完成提交事务的功能，即提交事务的所有操作，请参考**COMMIT | END**。

- 回滚事务

  回滚是在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销。请参考**ROLLBACK**。

  说明

  数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，如果其中有一个语句失败，那么整个请求都将会被回滚。

- 其他事务操作
  - SAVEPOINT用于在当前事务里建立一个新的保存点。即在一个事务中标记一个位置并且允许做部分回滚。用户可以回滚在一个保存点之后执行的命令但保留该保存点之前执行的命令。请参考**SAVEPOINT**。
  - ROLLBACK TO SAVEPOINT回滚事务到一个保存点。隐含地删除所有在该保存点之后建立的保存点。请参考**ROLLBACK TO SAVEPOINT**。
  - RELEASE SAVEPOINT删除一个事务内的保存点。请参考**RELEASE SAVEPOINT**。

版权所有 © 华为云计算技术有限公司

事务场景示例

某顾客在商店使用电子支付购买100元的物品，当中至少包括两个操作：1. 该顾客的账户减少100元。2. 商店账户（商户）增加100元。支持事务的数据库管理系统就是要确保以上两个操作（整个“事务”）都能完成，或一起取消。

1. 创建样例数据：

   创建一个简单的用户金额表并向表中插入数据（假设商户和顾客的账户上各有500元）。

   ```
   CREATE TABLE customer_info (
      NAME VARCHAR(32) PRIMARY KEY,
      MONEY INTEGER
   );
   INSERT INTO customer_info (name, money) VALUES ('buyer', 500), ('shop', 500);
   ```

   查看表数据显示商户和顾客各有500元。

   ```
   SELECT * FROM customer_info;
   ```

   ```
    name  | money
   -------+-------
    buyer |   500
    shop  |   500
   (2 rows)
   ```

2. 普通操作（正常模式）。

   模拟正常购买过程，顾客先扣款100元，商户再增加款额100元。

   ```
   UPDATE customer_info SET money = money-100 WHERE name IN (SELECT name FROM
   customer_info WHERE name = 'buyer');
   UPDATE customer_info SET money = money+100 WHERE name IN (SELECT name FROM
   customer_info WHERE name = 'shop');

   SELECT * FROM customer_info;
   ```

   ```
    name  | money
   -------+-------
    buyer |   400
    shop  |   600
   (2 rows)
   ```

3. 恢复初始值。

   ```
   UPDATE customer_info SET money=500;
   select * from customer_info;
   ```

   ```
    name  | money
   -------+-------
    shop  |   500
    buyer |   500
   (2 rows)
   ```

4. 普通操作（异常模式）。

   模拟购买过程出现状况，顾客发生扣款100元，商户没有增加款额。

   a. 顾客先扣款100元。

      ```
      UPDATE customer_info SET money = money-100 WHERE name IN (SELECT name FROM
      customer_info WHERE name = 'buyer');
      ```


版权所有 © 华为云计算技术有限公司

b. 商户发现支付有问题，终止了后续交易。商户增加款操作直接报错，终止执行下面的语句。（仅商户觉得支付有问题）

```
UPDATE customer_info SET money = money+100 WHERE name IN (SELECT name FROM customer_info WHERE name = 'shop');
```

c. 查询结果发现：消费者已经扣款，但商户没增加款额，这里顾客的金额少了100元。

```
SELECT * FROM customer_info;
```

```
 name  | money
-------+-------
 buyer |   400
 shop  |   500
(2 rows)
```

因此，如果没有事务，一旦SQL语句中间出现异常，整个账户系统的收支就不平衡了。

使用数据库事务，模拟出现异常操作时，进行事务回滚。

1. 恢复初始值：

```
UPDATE customer_info SET money=500;
```

2. 开启事务后，顾客先扣款100元。

```
BEGIN TRANSACTION;
UPDATE customer_info SET money = money-100 WHERE name IN (SELECT name FROM customer_info WHERE name = 'buyer');
```

3. 商户增加款额操作直接报错，终止执行下面的语句。

```
UPDATE customer_info SET money = money+100 WHERE name IN (SELECT name FROM customer_info WHERE name = 'shop');
```

4. 回滚事务，在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销。

```
ERROR:  syntax error at or near "shop"
LINE 1: ...e IN (SELECT name FROM customer_info WHERE name = ''shop'');
```

```
END TRANSACTION;
ROLLBACK
```

5. 查询显示顾客和商户的账户金额仍旧完整一致。即数据库在事务在执行过程中发生错误，会被恢复（Rollback）到事务开始前的状态，数据库的完整性没有被破坏。

```
SELECT * FROM customer_info;
```

```
 name  | money
-------+-------
 buyer |   500
 shop  |   500
(2 rows)
```

两阶段事务

DWS属于分布式share-nothing架构，表的数据分布在不同的节点上。客户端的一条或多条语句可能会同时修改多个节点上的数据，这种情况下，会产生分布式事务。DWS采用两阶段提交事务来保证分布式事务中数据的一致性和事务的原子性。顾名思义，两阶段提交就是将事务提交划分为两个阶段，通常针对的是包含写操作的事务。当写操作将数据写入不同的节点时，需要满足事务的原子性要求，要么全部提交，要么全部回滚。

版权所有 © 华为云计算技术有限公司

不支持两阶段的场景如下：

- 不支持显示的两阶段提交语法PREPARE TRANSACTION。
```
BEGIN;
PREPARE TRANSACTION 'p1';
```
```
ERROR: Explicit prepare transaction is not supported.
```
- 不支持在两阶段事务中修改系统表的文件映射关系。
```
REINDEX TABLE pg_class;
```
```
ERROR: cannot PREPARE a transaction that modified relation mapping.
```
- 不支持在跨节点的事务中提交导出事务快照。
```
BEGIN;
CREATE TABLE t1(a int);
SELECT pg_export_snapshot();
END;
```
```
ERROR: cannot PREPARE a transaction that has exported snapshots.
```

21.13 DDL 语法

21.13.1 DDL 语法一览表

DDL（Data Definition Language数据定义语言），用于定义或修改数据库中的对象。如：表、索引、视图等。

说明

DWS不支持CN不完整时进行DDL操作。例如：集群中有1个CN故障时执行新建数据库、表等操作都会失败。

定义数据库

数据库是组织、存储和管理数据的仓库，而数据库定义主要包括：创建数据库、修改数据库属性，以及删除数据库。所涉及的SQL语句，请参考下表。

表 21-100 数据库定义相关 SQL

功能	相关SQL
创建数据库	CREATE DATABASE
修改数据库属性	ALTER DATABASE
删除数据库	DROP DATABASE

定义模式

模式是一组数据库对象的集合，主要用于控制对数据库对象的访问。所涉及的SQL语句，请参考下表。

版权所有 © 华为云计算技术有限公司

表 21-101 模式定义相关 SQL

功能	相关SQL
创建模式	CREATE SCHEMA
修改模式属性	ALTER SCHEMA
删除模式	DROP SCHEMA

定义表

表是数据库中的一种特殊数据结构，用于存储数据对象以及对象之间的关系。所涉及的SQL语句，请参考下表。

表 21-102 表定义相关 SQL

功能	相关SQL
创建表	CREATE TABLE
修改表属性	ALTER TABLE
修改表名	RENAME TABLE
删除表	DROP TABLE
删除表的所有数据	TRUNCATE

定义分区表

分区表是一种逻辑表，数据是由普通表存储的，主要用于提升查询性能。所涉及的SQL语句，请参考下表。

表 21-103 分区表定义相关 SQL

功能	相关SQL
创建分区表	CREATE TABLE PARTITION
修改分区	ALTER TABLE PARTITION
修改分区表属性	ALTER TABLE PARTITION
删除分区	ALTER TABLE PARTITION
删除分区表	DROP TABLE

定义索引

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。所涉及的SQL语句，请参考下表。

版权所有 © 华为云计算技术有限公司

表 21-104 索引定义相关 SQL

功能	相关SQL
创建索引	CREATE INDEX
修改索引属性	ALTER INDEX
删除索引	DROP INDEX
重建索引	REINDEX

定义角色

角色是用来管理权限的，从数据库安全的角度考虑，可以把所有的管理和操作权限划分到不同的角色上。所涉及的SQL语句，请参考下表。

表 21-105 角色定义相关 SQL

功能	相关SQL
创建角色	CREATE ROLE
修改角色属性	ALTER ROLE
删除角色	DROP ROLE

定义用户

用户是用来登录数据库的，通过对用户赋予不同的权限，可以方便地管理用户对数据库的访问及操作。所涉及的SQL语句，请参考下表。

表 21-106 用户定义相关 SQL

功能	相关SQL
创建用户	CREATE USER
修改用户属性	ALTER USER
删除用户	DROP USER

定义脱敏策略

数据脱敏策略是指对某些敏感信息通过脱敏规则进行数据的变形，实现敏感隐私数据的可靠保护。用户可以在指定表对象创建脱敏策略，并限定策略生效范围，也可以新增、修改、删除脱敏列信息。所涉及的SQL语句，请参考下表。

版权所有 © 华为云计算技术有限公司

表 21-107 脱敏策略定义相关 SQL

功能	相关SQL
对表创建数据脱敏策略	CREATE REDACTION POLICY
修改应用在指定表的脱敏策略	ALTER REDACTION POLICY
删除应用在指定表的脱敏策略	DROP REDACTION POLICY

定义行级访问控制

行级访问控制策略控制数据库表中行级数据可见性。不同用户执行相同的SQL查询操作，读取到的结果不同。所涉及的SQL语句，请参考下表。

表 21-108 行级访问控制定义相关 SQL

功能	相关SQL
创建行访问控制策略	CREATE ROW LEVEL SECURITY POLICY
修改已存在的行访问控制策略	ALTER ROW LEVEL SECURITY POLICY
删除表上某个行访问控制策略	DROP ROW LEVEL SECURITY POLICY

定义存储过程

存储过程是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。所涉及的SQL语句，请参考下表。

表 21-109 存储过程定义相关 SQL

功能	相关SQL
创建存储过程	CREATE PROCEDURE
删除存储过程	DROP PROCEDURE

定义函数

在DWS中，它和存储过程类似，也是一组SQL语句集，使用上没有差别。所涉及的SQL语句，请参考下表。

版权所有 © 华为云计算技术有限公司

表 21-110 函数定义相关 SQL

功能	相关SQL
创建函数	CREATE FUNCTION
修改函数属性	ALTER FUNCTION
删除函数	DROP FUNCTION

定义视图

视图是从一个或几个基本表中导出的虚表，可用于控制用户对数据访问，请参考下表。

表 21-111 视图定义相关 SQL

功能	相关SQL
创建视图	CREATE VIEW
删除视图	DROP VIEW

定义游标

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化，请参考下表。

表 21-112 游标定义相关 SQL

功能	相关SQL
创建游标	CURSOR
移动游标	MOVE
从游标中提取数据	FETCH
关闭游标	CLOSE

操作会话

用户与数据库之间建立的连接称为会话，请参考下表。

表 21-113 会话相关 SQL

功能	相关SQL
修改会话	ALTER SESSION

版权所有 © 华为云计算技术有限公司

功能	相关SQL
结束会话	ALTER SYSTEM KILL SESSION

定义资源池

资源池是负载管理模块使用的系统表，主要用于指定资源管理相关的属性，如控制组。所涉及的SQL语句，请参考下表。

表 21-114 资源池定义相关 SQL

功能	相关SQL
创建资源池	CREATE RESOURCE POOL
修改资源池属性	ALTER RESOURCE POOL
删除资源池	DROP RESOURCE POOL

定义同义词

同义词是兼容Oracle的一种特殊的数据库对象，用于存储与另一个数据库对象名间的映射关系，目前仅支持使用同义词关联以下数据库对象：表、视图、函数和存储过程。所涉及的SQL语句，请参考下表。

表 21-115 同义词定义相关 SQL

功能	相关SQL
创建同义词	CREATE SYNONYM
修改同义词	ALTER SYNONYM
删除同义词	DROP SYNONYM

定义文本搜索配置

文本搜索配置指定了文本搜索解析器，该文本搜索解析器可以将字符串划分为标记，外加一些词典（可被用来决定哪些标记是搜索感兴趣的）。所涉及的SQL语句，请参考下表。

表 21-116 文本搜索配置相关 SQL

功能	相关SQL
创建文本搜索配置	CREATE TEXT SEARCH CONFIGURATION

版权所有 © 华为云计算技术有限公司

功能	相关SQL
修改文本搜索配置	ALTER TEXT SEARCH CONFIGURATION
删除文本搜索配置	DROP TEXT SEARCH CONFIGURATION

定义全文检索词典

词典是在全文检索时识别特定词并进行处理。词典的创建依赖于预定义模板（在系统表PG_TS_TEMPLATE中定义），支持创建五种类型的词典，分别是Simple、Ispell、Synonym、Thesaurus、以及Snowball，每种类型的词典可以完成不同的任务。所涉及的SQL语句，请参考下表。

表 21-117 全文检索词典相关 SQL

功能	相关SQL
创建全文检索词典	CREATE TEXT SEARCH DICTIONARY
修改全文检索词典	ALTER TEXT SEARCH DICTIONARY
删除全文检索词典	DROP TEXT SEARCH DICTIONARY

定义查询过滤规则

对于提前识别到的慢SQL语句，用户可以通过创建查询过滤器规则来进行拦截。所涉及的SQL语句，请参考下表。

表 21-118 查询过滤规则相关 SQL

功能	相关SQL
创建查询过滤规则	CREATE BLOCK RULE
修改查询过滤规则	ALTER BLOCK RULE
删除查询过滤规则	DROP BLOCK RULE

21.13.2 数据库&表空间&模式&数据类型

21.13.2.1 CREATE DATABASE

功能描述

创建新的数据库。

版权所有 © 华为云计算技术有限公司

DWS采用从模板数据库复制的方式来创建新的数据库。DWS系统提供了两个模板数据库（template0和template1）以及一个默认的用户数据库（gaussdb）。默认情况下，新的数据库基于template1创建。template1不允许修改字符编码，因此若要变更编码，用户可以通过指定**template**参数选择template0模板。

注意事项

- 只有拥有CREATEDB权限的用户才可以创建新数据库，系统管理员默认拥有此权限。
- CREATE DATABASE不能在事务块中执行。
- 在创建数据库过程中，若出现类似“could not initialize database directory”的错误提示，可能是由于文件系统上数据目录的权限不足或磁盘满等原因引起。

语法格式

```
CREATE DATABASE database_name
   [ [ WITH ] { [ OWNER [=] user_name ] |
            [ TEMPLATE [=] template ] |
            [ ENCODING [=] encoding ] |
            [ LC_COLLATE [=] lc_collate ] |
            [ LC_CTYPE [=] lc_ctype ] |
            [ DBCOMPATIBILITY [=] compatibility_type ] |
            [ DBCOMPATIBILITY_BEHAVIOR [=] opt_compat_behavior] |

            [ CONNECTION LIMIT [=] connlimit ]}[...] ];
```

参数说明

表 21-119 CREATE DATABASE 参数说明

参数	描述	取值范围
database_name	要创建的数据库名称。	字符串，需符合**标识符命名规范**。
user_name	指定新数据库的所有者。缺省时，新数据库的所有者是当前用户。	已存在的用户名。
template	创建新数据库时指定的模板名称，即使用哪个模板创建新数据库。不指定时，系统默认使用template1作为模板。	template0和template1。 • template0，如果需要指定字符集编码（encoding），需指定为template0。 • template1，缺省值。模板template1不允许指定字符集编码（encoding）。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
encoding	指定数据库使用的字符集编码，可以是字符串（如'SQL_ASCII'）、整数编号。不指定时，默认使用模板数据库的编码。 模板数据库template0和template1的编码默认与操作系统环境相关。	常用取值为GBK、UTF8、Latin1、SQL_ASCII。
lc_collate	指定新数据库使用的字符集。例如，通过lc_collate = 'zh_CN.gbk'设定该参数。 该参数的使用会影响到对字符串的排序顺序（例如使用ORDER BY的查询中，以及在文本列上使用索引的顺序）。默认使用模板数据库的排序顺序。有关其他限制请参考**字符集编码的说明**。	有效的排序类型。
lc_ctype	指定新数据库使用的字符分类。例如，通过lc_ctype = 'zh_CN.gbk'设定该参数。 该参数的使用会影响到字符的分类，如大写、小写和数字。默认是使用模板数据库的字符分类。	有效的字符分类。
compatibility_type	指定兼容的数据库的类型。若不指定该参数，默认为ORA。	取值为ORA、TD或MySQL。 分别表示兼容Oracle、Teradata或MySQL数据库。
opt_compat_behavior	指定数据库的兼容性行为。该参数仅9.1.0及以上集群版本支持。 若不指定该参数，默认为NO_BEHAVIOR，表示无特殊行为。	取值为td_rtrim或pg_char。 • td_rtrim，表示TD兼容模式下变长字符串去尾空。 • pg_char，表示PG兼容模式下varchar类型转为nvarchar2类型。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
connlimit	指定数据库的并发连接数。默认值为-1，表示没有限制。 ● 系统管理员不受此参数的限制。 ● 为保证集群正常使用，connection limit的最小值是集群中CN的数目。在集群做ANALYZE时，其他CN节点会连接当前做ANALYZE的CN节点来同步元数据。例如集群中有3个CN节点，那么connection limit应该设置为>=3。	大于等于-1的整数。

字符集编码的说明

- 可使用“show server_encoding;”命令查看当前数据库的字符集编码。
- 为了适应全球化的需求，使数据库编码能够存储与表示绝大多数的字符，建议创建数据库时使用UTF8编码。
- 指定新的数据库字符集编码必须与所选择的语言环境中（LC_COLLATE和LC_CTYPE）的设置兼容。
- 若语言环境设置为C（或POSIX），则允许所有的编码类型，但是对于其他的语言环境设置，字符集编码必须和区域设置相同。
- 编码和区域设置必须匹配模板数据库，除了将template0当作模板。 因为其他数据库可能会包含不匹配指定编码的数据，或者可能包含排序顺序受LC_COLLATE和LC_CTYPE影响的索引。复制这些数据会导致在新数据库中的索引失效。template0是不包含任何会受到影响的数据或者索引。
- 支持的有效编码类型与当前所处的环境有关。若出现“invalid locale name”的字可通过locale -a命令检查环境所支持的字符集编码。
- 当指定的字符集编码为GBK时，部分中文生僻字无法直接作为对象名。这是因为GBK第二个字节的编码范围在0x40-0x7E之间时，字节编码与ASCII字符@A-Z[\]^_`a-z{|}重叠。其中@[\]^_'{|}是数据库中的操作符，直接作为对象名时，会语法报错。例如“俦”字，GBK16进制编码为0x8240，第二个字节为0x40，与ASCII“@”符号编码相同，因此无法直接作为对象名使用。如果确实要使用，可以在创建和访问对象时，通过增加双引号来规避这个问题。
- 当前版本GBK字符集支持了欧元符'€'，十六进制表示为'0x80'，用户可以在GBK库中操作欧元符，也使得DWS的GBK字符集可以兼容CP936字符集。需注意GBK字符集约等于CP936字符集，但是GBK字符集中不包含欧元符的定义。

示例

创建一个GBK编码的数据库music（本地环境的编码格式必须也为GBK）。

```
CREATE DATABASE music ENCODING 'GBK' template = template0;
```

创建数据库music2，并指定所有者为jim。

```
CREATE USER jim PASSWORD "{Password}";
CREATE DATABASE music2 OWNER jim;
```

用模板template0创建数据库music3，并指定所有者为jim。

版权所有 © 华为云计算技术有限公司

```
CREATE DATABASE music3 OWNER jim TEMPLATE template0;
```

创建兼容ORA格式的数据库，连接限制为10。

```
CREATE DATABASE ora_compatible_db DBCOMPATIBILITY 'ORA' CONNECTION LIMIT = 10;
```

相关链接

ALTER DATABASE，**DROP DATABASE**

常见问题

参见《数据仓库服务用户指南》的“常见问题 > 数据库使用 > 如何查看和设置DWS数据库的字符集编码格式”章节。

21.13.2.2 ALTER DATABASE

功能描述

修改数据库的属性，包括它的名称、所有者、连接数限制、对象隔离属性等。

注意事项

- 只有拥有数据库所有者权限的用户才能执行ALTER DATABASE命令，系统管理员默认拥有此权限。如果是非系统管理员，针对所要修改属性的不同，对其还有以下权限约束：
  - 修改数据库名称，必须拥有CREATEDB权限。
  - 修改数据库所有者，当前用户必须是该数据库的所有者，且拥有CREATEDB权限，并确保该用户是新所有者角色的成员。
  - 修改数据库默认表空间，该用户必须是该数据库的所有者或系统管理员，且拥有新表空间的CREATE权限。该语法会从物理上将一个数据库原来缺省表空间上的表和索引移至新的表空间。注意不在缺省表空间的表和索引不受此影响。
  - 修改某个按数据库设置的相关参数，只有数据库所有者或者系统管理员可以改变这些设置。
  - 修改某个数据库对象隔离属性，只有数据库所有者或者系统管理员可以执行此操作。
- 不能重命名当前使用的数据库，如果需要重新命名，必须连接至其他数据库上。
- 不支持修改现有数据库的兼容模式，只能在创建数据库时指定兼容模式，详情请参见**CREATE DATABASE**。

语法格式

- 修改数据库的最大连接数。

```
ALTER DATABASE database_name
  [ [ WITH ] CONNECTION LIMIT connlimit ];
```

- 修改数据库名称。

```
ALTER DATABASE database_name
  RENAME TO new_name;
```

说明

若该数据库中有OBS冷热表，则不支持修改数据库名。

版权所有 © 华为云计算技术有限公司

- 修改数据库所属者。
```
ALTER DATABASE database_name
    OWNER TO new_owner;
```
- 修改数据库指定会话参数值。
```
ALTER DATABASE database_name
    SET configuration_parameter { { TO | = } { value | DEFAULT } | FROM CURRENT };
```
- 数据库配置参数重置。
```
ALTER DATABASE database_name RESET
    { configuration_parameter | ALL };
```
- 修改数据库对象隔离属性。
```
ALTER DATABASE database_name [ WITH ] { ENABLE | DISABLE } PRIVATE OBJECT;
```

说明

- 修改数据库的对象隔离属性时须连接至该数据库，否则无法更改。
- 新创建的数据库，对象隔离属性默认是关闭的。当开启数据库对象隔离属性后，普通用户只能查看有权访问的对象（表、函数、视图、字段等）。对象隔离特性对管理员用户不生效，当开启对象隔离特性后，管理员也可以查看到全量的数据库对象。

参数说明

表 21-120 ALTER DATABASE 参数说明

参数	描述	取值范围
database_name	需要修改属性的数据库名称。	字符串，需符合**标识符命名规范**。
connlimit	数据库可以接收的最大并发连接数（管理员用户连接除外）。	整数，建议填写1~50。 -1（缺省）表示没有限制。
new_name	数据库的新名称。	字符串，需符合**标识符命名规范**。
new_owner	数据库的新所有者。	字符串，有效的用户名。
configuration_parameter value	将指定的数据库会话参数值设置为给定值。	字符串： • DEFAULT，表示在新的会话中使用系统的缺省设置。 • RESET，表示在新的会话中使用系统的缺省设置。 • OFF，表示关闭设置。
FROM CURRENT	根据当前会话连接的数据库设置该参数的值。	-
RESET configuration_parameter	重置指定的数据库会话参数值。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
RESET ALL	重置全部的数据库会话参数值。 修改的数据库会话参数值，将在下一次会话中生效。	-

示例

创建数据库testdb和用户user1。

```
CREATE DATABASE testdb ENCODING 'UTF8' template = template0;
DROP USER IF EXISTS user1;
CREATE USER user1 PASSWORD '{Password}';
```

设置数据库testdb的连接数为10。

```
ALTER DATABASE testdb CONNECTION LIMIT= 10;
```

将testdb名称改为testdb_1。

```
ALTER DATABASE testdb RENAME TO testdb_1;
```

将数据库testdb_1的所属者修改为user1。

```
ALTER DATABASE testdb_1 OWNER TO user1;
```

关闭在数据库testdb_1上缺省的索引扫描。

```
ALTER DATABASE testdb_1 SET enable_indexscan TO off;
```

重置enable_indexscan参数。

```
ALTER DATABASE testdb_1 RESET enable_indexscan;
```

相关链接

CREATE DATABASE，**DROP DATABASE**

21.13.2.3 DROP DATABASE

功能描述

删除数据库。

注意事项

- 只有数据库所有者有权限执行DROP DATABASE命令，系统管理员默认拥有此权限。
- 不能对系统默认安装的三个数据库（gaussdb、TEMPLATE0和TEMPLATE1）执行删除操作，系统做了保护。如果想查看当前服务中有哪几个数据库，可以用gsql的\l命令查看。
- 如果有用户正在与要删除的数据库连接，则删除操作失败。如果要查看当前存在哪些数据库连接，可以通过视图v$session查看。
- 不能在事务块中执行DROP DATABASE命令。

版权所有 © 华为云计算技术有限公司

- 如果执行DROP DATABASE失败，事务回滚，需要再次执行一次DROP DATABASE IF EXISTS。
- DROP DATABASE若提示database is being accessed by other users类错误，可能原因为CLEAN CONNECTION过程存在线程无法及时响应信号，出现连接清理不完全的情况，需要再次执行CLEAN CONNECTION。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
DROP DATABASE [ IF EXISTS ] database_name ;
```

参数说明

表 21-121 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的数据库不存在时返回一个提示而不是错误。	-
database_name	需要删除的数据库名称。	已存在的数据库名称。

示例

删除名称为music的数据库。

```
DROP DATABASE music;
```

相关链接

CREATE DATABASE，**ALTER DATABASE**

常见问题

当执行DROP DATABASE语法时出现以下类似报错，表示待删除数据库被其他用户连接。

```
database "mysql_compatible_db" is being accessed by other users Detail: There are 2 other sessions using the database.
```

可执行以下语法，先断开所有用户对该数据库建立的连接。

```
CLEAN CONNECTION TO ALL FORCE FOR DATABASE mysql_compatible_db;
```


版权所有 © 华为云计算技术有限公司

再重新执行DROP DATABASE，可删除成功（如删除失败，可以重复执行以上CLEAN CONNECTION再重试）。

```
DROP DATABASE mysql_compatible_db;
```

21.13.2.4 CREATE SCHEMA

功能描述

创建模式。

访问命名对象时可以使用模式名作为前缀进行访问，如果无模式名前缀，则访问当前模式下的命名对象。创建命名对象时也可用模式名作为前缀修饰。

另外，CREATE SCHEMA可以包括在新模式中创建对象的子命令，这些子命令和那些在创建完模式后发出的命令没有任何区别。如果使用了AUTHORIZATION子句，则所有创建的对象都将被该用户所拥有。

注意事项

- 只要用户对当前数据库有CREATE权限，就可以创建模式。
- 系统管理员在普通用户同名Schema下创建的对象，所有者为Schema的同名用户（非系统管理员）。

语法格式

- 根据指定的名字创建模式：

```
CREATE SCHEMA schema_name
    [ AUTHORIZATION user_name ] [ WITH PERM SPACE 'space_limit'] [ schema_element [ ... ] ];
CREATE SCHEMA IF NOT EXISTS schema_name
    [ AUTHORIZATION user_name ] [ WITH PERM SPACE 'space_limit'] ;
```

- 根据用户名创建模式：

```
CREATE SCHEMA AUTHORIZATION user_name [ WITH PERM SPACE 'space_limit'] [ schema_element [ ... ] ];
CREATE SCHEMA IF NOT EXISTS AUTHORIZATION user_name [ WITH PERM SPACE 'space_limit'] ;
```

参数说明

表 21-122 CREATE SCHEMA 参数说明

参数	描述	取值范围
schema_name	模式名字。 **须知** • 模式名不能和当前数据库里其他的模式重名。 • 模式的名字不可以“pg_”开头。	字符串，需符合标识符的命名规范，参见**标识符命名规范**。
AUTHORIZATION user_name	指定模式的所有者。当不指定schema_name时，把user_name当作模式名，此时user_name只能是角色名。	已存在的用户名/角色名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
WITH PERM SPACE 'space_limit'	指定模式的永久表存储空间上限。当不指定space_limit时，则不限制。	字符串格式为正整数+单位，单位当前支持K/M/G/T/P。解析后的数值以K为单位，且范围不能够超过64比特表示的有符号整数，即1KB~9007199254740991KB。
schema_element	在模式里创建对象的SQL语句。目前仅支持CREATE TABLE、CREATE VIEW、CREATE INDEX、CREATE PARTITION、GRANT子句。 子命令所创建的对象都被AUTHORIZATION子句指定的用户所拥有。	-
IF NOT EXISTS	指定IF NOT EXISTS时，若不存在同名SCHEMA，则可以成功创建SCHEMA。若已存在同名SCHEMA，创建时不会报错，仅会提示该SCHEMA已存在，且不执行任何操作。当使用该参数时，不能包括Schema_element子命令。 IF NOT EXISTS参数仅9.1.0及以上版本支持。	如果当前搜索路径上的模式中存在同名对象时，需要明确指定引用对象所在的模式。可以通过命令SHOW SEARCH_PATH来查看当前搜索路径上的模式。

示例

创建一个角色role1。

```
CREATE ROLE role1 IDENTIFIED BY '{password}';
```

为用户role1创建一个同名schema，子命令创建的表films和winners的拥有者为role1。

```
CREATE SCHEMA AUTHORIZATION role1
    CREATE TABLE films (title text, release date, awards text[])
    CREATE VIEW winners AS
    SELECT title, release FROM films WHERE awards IS NOT NULL;
```

相关链接

ALTER SCHEMA，**DROP SCHEMA**

21.13.2.5 ALTER SCHEMA

功能描述

修改模式属性。

版权所有 © 华为云计算技术有限公司

注意事项

- 只有模式的所有者或者被授予了模式ALTER权限的用户，有权限执行ALTER SCHEMA命令，系统管理员默认拥有此权限。
- 非系统管理员要修改模式的所有者或模式的永久表存储空间上限，该用户还必须是新的所有角色的直接或间接成员， 并且该成员必须在此数据库上有CREATE权限。

语法格式

- 修改模式的名称。
```
ALTER SCHEMA schema_name
    RENAME TO new_name;
```
- 修改模式的所有者。
```
ALTER SCHEMA schema_name
    OWNER TO new_owner;
```
- 修改模式的永久表存储空间限制。
```
ALTER SCHEMA schema_name
    WITH PERM SPACE 'space_limit';
```

参数说明

表 21-123 ALTER SCHEMA 参数说明

参数	描述	取值范围
schema_name	需要修改的模式的名称。	已存在的模式名。
new_name	模式的新名称。	字符串，需符合**标识符命名规范**。
new_owner	模式的新所有者。	已存在的用户名/角色名。
WITH PERM SPACE 'space_limit'	模式的永久表存储空间上限。	字符串格式为正整数+单位，单位当前支持K/M/G/T/P。解析后的数值以K为单位，且范围不能够超过64比特表示的有符号整数，即1KB~9007199254740991KB。

示例

创建示例模式schema_test和用户user_a：

```
CREATE SCHEMA schema_test;
CREATE USER user_a PASSWORD '{Password}';
```

将当前模式schema_test更名为schema_test1：

```
ALTER SCHEMA schema_test RENAME TO schema_test1;
```

将schema_test1的所有者修改为user_a：

```
ALTER SCHEMA schema_test1 OWNER TO user_a;
```


版权所有 © 华为云计算技术有限公司

相关链接

CREATE SCHEMA，DROP SCHEMA

21.13.2.6 DROP SCHEMA

功能描述

从数据库中删除模式。

注意事项

- 只有模式的所有者或者被授予了模式DROP权限的用户有权限执行DROP SCHEMA命令，系统管理员默认拥有此权限。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
DROP SCHEMA [ IF EXISTS ] schema_name [, ...] [ CASCADE | RESTRICT ];
```

参数说明

表 21-124 DROP SCHEMA 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的模式不存在，发出一个notice而不是抛出一个错误。	-
schema_name	要删除的模式名。	已存在的模式名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：自动删除包含在模式中的对象。 • RESTRICT：如果模式包含任何对象，则删除失败（缺省行为）。

版权所有 © 华为云计算技术有限公司

须知

- 不要随意删除pg_temp或pg_toast_temp开头的模式，这些模式是系统内部使用的，如果删除，可能导致无法预知的结果。
- 无法删除当前模式。如果要删除当前模式，须切换到其他模式下。

示例

删除模式ds_new。

```
DROP SCHEMA ds_new;
```

相关链接

ALTER SCHEMA，**CREATE SCHEMA**

21.13.2.7 CREATE EXTERNAL SCHEMA

功能描述

创建EXTERNAL SCHEMA，即外部模式。该语法仅8.3.0及以上版本支持。

创建EXTERNAL SCHEMA来访问Lakeformation、DLI、Hive Metastore的表。用户可以使用外部模式名作为前缀进行访问，若无模式名前缀，则访问当前模式下的命名对象。

注意事项

- 只要拥有当前数据库CREATE权限的用户，就可以创建外部SCHEMA。
- 创建命名对象时不可用EXTERNAL SCHEMA作为前缀修饰，即不支持在EXTERNAL SCHEMA下创建对象。
- CREATE EXTERNAL SCHEMA不支持在新模式中创建对象的子命令。

语法格式

- 根据指定的名称创建EXTERNAL SCHEMA。

```
CREATE EXTERNAL SCHEMA schema_name
    WITH SOURCE source_name
        DATABASE 'database_name'
        SERVER server_name
        [ CATALOG 'catalog_name' ]
        [ OPTIONS ( { option_name ' value ' } [, ...] ) ]
        [METAADDRESS 'address']
        [CONFIGURATION 'confpath'];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-125 CREATE EXTERNAL SCHEMA 参数说明

参数	描述	取值范围
schema_name	外部模式名称。 • 模式名不能和当前数据库里其他的模式重名。 • 模式的名称不可以“pg_”开头。	字符串，需符合**标识符命名规范**。
SOURCE	外部元数据存储引擎的类型。	当前仅支持dli，lakeformation，hive。
DATABASE	指定外部SCHEMA所对应的要访问的数据库。	-
SERVER	外部SCHEMA关联的外部服务器名称，关联后可访问到外部数据。	支持关联type类型为lf、obs、dli或hdfs的Server。详情可参见**CREATE SERVER**。
CATALOG	Lakeformation中对应的要访问的catalog，当Server的type为lf时需要指定catalog，否则不需要指定。	-
OPTIONS	用于指定外表数据的各类参数，当前支持的参数类型如右列所示。该参数仅8.3.0及以上版本支持。	dli_project_id：DLI服务对应的项目编号，可在管理控制台上获取项目ID，该参数仅支持type类型为dli的Server。
METAADDRESS	表示Hive Metastore通讯接口。该参数仅9.1.0及以上版本支持。	-
CONFIGURATION	表示Hive Metastore相关配置文件存放路径。该参数仅9.1.0及以上版本支持。	如果当前搜索路径上的模式中存在同名对象时，需要明确指定引用对象所在的模式。可以通过命令SHOW SEARCH_PATH来查看当前搜索路径上的模式。

示例：通过 external schema 读取 Lakeformation 表

通过external schema读取Lakeformation表的方式如下：

1. 创建lf_server，对应的foreign data wrapper为DFS_FDW。

版权所有 © 华为云计算技术有限公司

说明

如何创建lf_server请参见《数据仓库服务用户指南》中“管理LakeFormation数据源”章节。

2. 创建external schema。

   SOURCE填写lakeformation，表关联的DLI服务器为lf_server，DATABASE为对应的Lakeformation数据库，

   CATALOG为对应要访问的Lakeformation的CATALOG，根据实际替换。

```
CREATE EXTERNAL SCHEMA ex_lf
   WITH SOURCE lakeformation
        DATABASE 'demo'
        SERVER lf_server
        CATALOG 'hive';
```

3. 角色授权。

   a. 查询当前用户。

```
SELECT current_user;
```

   b. Lakeformation管控面创建与当前同名角色并对要访问的表授权。

4. 通过external schema查询Lakeformation表的数据，test_lf为要访问的Lakeformation表。

```
SELECT COUNT(*) FROM ex_dli.test_lf;
 count
-------
    20
(1 row)
```

通过 external schema 读取 Hive Metastore 表。

通过external schema读取Hive Metastore表。仅9.1.0及以上版本支持。

1. 创建obs/hdfs server，对应的foreign data wrapper为DFS_FDW。

```
CREATE SERVER hdfs_server
FOREIGN DATA WRAPPER HDFS_FDW
OPTIONS (
address '***.***.***.***:9000',
type'HDFS');

CREATE SERVER obs_server
FOREIGN DATA WRAPPER dfs_fdw
OPTIONS (
    address 'obs.example.com' ,
    ACCESS_KEY 'access_key_value_to_be_replaced',
    SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
    encrypt 'on',
    type 'obs' );
```

2. 创建external schema，SOURCE为hive，表关联的服务器为obs/hdfs server。其中DATABASE为对应的HMS数据库，METAADDRESS表示Hive Metastore通讯接口，CONFIGURATION是一个绝对路径，表示Hive Metastore相关配置文件存放路径，根据实际替换。

   手工创建external schema对接Hive Metastore时，用户需先准备好相关配置文件，具体如下：

   a. 从集群管理页面或hive安装目录获取hive-site.xml、hivemetastore-site.xml这两个配置文件，上传至CONFIGURATION目录。

   b. 新增elk-site.xml配置文件并添加配置项（或在原有的elk-site.xml配置文件中新增），需要根据实际替换用户名、域名、keytab文件名等信息。

```
    <property>
        <name>hive.kerberos.principal</name>
        <value>user/hadoop.hadoop.com@HADOOP.COM</value> //连接Hadoop的用户@域名
```


版权所有 © 华为云计算技术有限公司

```
    </property>
    <property>
        <name>hive.kerberos.keytab</name>
        <value>{配置目录}/user.keytab</value> //连接Hadoop用户的keytab文件的绝对路径
    </property>
```

```
CREATE EXTERNAL SCHEMA ex_hms
  WITH SOURCE source_type
  DATABASE 'db_name'
  SERVER srv_name
  METAADDRESS 'address'
  CONFIGURATION 'confpath';
```

3. 通过external schema查询HMS表的数据，test_hms为要访问的HMS表，根据实际替换。

```
SELECT COUNT(*) FROM ex_hms.test_hms;
 count
-------
    20
(1 row)
```

相关链接

ALTER EXTERNAL SCHEMA

21.13.2.8 ALTER EXTERNAL SCHEMA

功能描述

修改EXTERNAL SCHEMA。该语法仅8.3.0及以上版本支持。

语法格式

根据指定的名称修改EXTERNAL SCHEMA。

```
ALTER EXTERNAL SCHEMA schema_name
  WITH [ SOURCE source_name ]
      [ DATABASE 'database_name' ]
      [ SERVER server_name ]
      [ CATALOG 'catalog_name' ]
      [ OPTIONS ( { option_name ' value ' } [, ...] ) ]
      [ METAADDRESS 'address']
      [ CONFIGURATION 'confpath'];
```

参数说明

表 21-126 ALTER EXTERNAL SCHEMA 参数说明

参数	描述	取值范围
schema_name	需要修改的外部模式的名称。	已有的模式名。
SOURCE	外部元数据存储引擎的类型。	当前source_type仅支持dli，lakeformation，hive。
DATABASE	指定外部SCHEMA所对应的要访问的数据库。	-
SERVER	通过external schema关联foreign server以达到访问外部数据的目的。	已存在的foreign server且type类型为lf、obs、dli或hdfs。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CATALOG	Lakeformation中对应的要访问的catalog。	-
OPTIONS	指定外表数据的各类参数。该参数仅8.3.0及以上版本支持。	dli_project_id，DLI服务对应的项目编号，可在管理控制台上获取项目ID，该参数仅支持server类型为DLI时设置。
METAADDRESS	Hive Metastore通讯接口。该参数仅9.1.0及以上版本支持。	-
CONFIGURATION	Hive Metastore相关配置文件存放路径。该参数仅9.1.0及以上版本支持。 如果当前搜索路径上的模式中存在同名对象时，需要明确指定引用对象所在的模式。可以通过命令SHOW SEARCH_PATH来查看当前搜索路径上的模式。	-

示例

修改名称为ex1的EXTERNAL SCHEMA对应的数据库和FOREIGN SERVER。

```
ALTER EXTERNAL SCHEMA ex1
    WITH DATABASE 'demo'
        SERVER my_server;
```

21.13.2.9 CREATE TYPE

功能描述

在当前数据库中定义新的数据类型。定义数据类型的用户将成为该数据类型的所有者。定义类型只适用于行存表。

有四种形式的CREATE TYPE，分别为：复合类型、基本类型、shell类型和枚举类型。

- 复合类型

  复合类型由一个属性名和数据类型的列表指定。如果属性的数据类型是可排序的，也可以指定该属性的排序规则。复合类型本质上和表的行类型相同，但是如果只想定义一种类型，使用CREATE TYPE避免了创建一个实际的表。单独的复合类型也是很有用的，例如可以作为函数的参数或者返回类型。

  为了能够创建复合类型，必须拥有在其所有属性类型上的USAGE特权。

- 基本类型

  用户可以自定义一种新的基本类型（标量类型）。通常来说这些函数必须是用C或者另外一种底层语言所编写。

- shell类型

版权所有 © 华为云计算技术有限公司

shell类型是一种用于后面要定义的类型的占位符，通过发出一个只带类型名参数的CREATE TYPE命令来创建该类型。在创建基本类型时，需要shell类型作为一种向前引用。

- 枚举类型

  由若干个标签构成的列表，每一个标签值都是一个非空字符串，且字符串长度必须不超过64个字节。

注意事项

- 如果给定一个模式名，那么该类型将被创建在指定的模式中，否则它会被创建在当前模式中。类型名称必须与同一个模式中任何现有的类型或者域有所区别（因为表具有相关的数据类型，类型名称也必须与同一个模式中任何现有表的名字不同）。
- 自定义基本类型时，参数可以以任意顺序出现，input_function和output_function为必选参数，其它为可选参数。
- category和preferred参数可以被用来帮助控制在混淆的情况下应用哪一种隐式造型。每一种数据类型都属于一个用单个ASCII字符命名的分类，并且每一种类型可以是其所属分类中的“首选”。当有助于解决重载函数或操作符时，解析器将优先造型到首选类型（但是只能从同类的其他类型造型）。对于没有隐式转换到或来自任意其他类型的类型，让这些设置保持默认即可。不过，对于有隐式转换的相关类型的组，把它们都标记为属于同一个类别并且选择一种或两种“最常用”的类型作为该类别的首选通常是很有用的。在把一种用户定义的类型增加到一个现有的内建类别（例如，数字或者字符串类型）中时，category参数特别有用。不过，也可以创建新的全部是用户定义类型的类别。对这样的类别，可选择除大写字母之外的任何ASCII字符。
- 如果类型支持修饰符（附加在类型声明上的可选约束，例如，char(5)或numeric(30,2)），则需要可选的type_modifier_input_function和type_modifier_output_function。DWS允许用户定义的类型有一个或者多个简单常量或者标识符作为修饰符。不过，为了存储在系统目录中，该信息必须能被打包到一个非负整数值中。所声明的修饰符会被以cstring数组的形式传递给type_modifier_input_function。type_modifier_input_function必须检查该值的合法性（如果值错误就抛出一个错误），如果值正确，要返回一个非负integer值，该值将被存储在“typmod”列中。如果类型没有type_modifier_input_function则类型修饰符将被拒绝。type_modifier_output_function把内部的整数typmod值转换回正确的形式用于用户显示。type_modifier_output_function必须返回一个cstring值，该值就是追加到类型名称后的字符串。例如，numeric的函数可能会返回(30,2)。如果默认的显示格式就是只把存储的typmod整数值放在圆括号内，则允许省略type_modifier_output_function。
- 输入和输出函数能被声明为具有新类型的结果或参数是因为：必须在创建新类型之前创建这两个函数。而新类型应该首先被定义为一种shell type，它是一种占位符类型，除了名称和拥有者之外它没有其他属性。这可以通过不带额外参数的命令CREATE TYPE name做到。然后用C写的I/O函数可以被定义为引用这种shell type。最后，用带有完整定义的CREATE TYPE把该shell type替换为一个完全的、合法的类型定义，之后新类型就可以正常使用了。

语法格式

```
CREATE TYPE name AS
    ( [ attribute_name data_type [ COLLATE collation ] [, ... ] ] )

CREATE TYPE name (
    INPUT = input_function,
    OUTPUT = output_function
```

 版权所有 © 华为云计算技术有限公司

```
    [ , RECEIVE = receive_function ]
    [ , SEND = send_function ]
    [ , TYPMOD_IN = type_modifier_input_function ]
    [ , TYPMOD_OUT = type_modifier_output_function ]
    [ , ANALYZE = analyze_function ]
    [ , INTERNALLENGTH = { internallength | VARIABLE } ]
    [ , PASSEDBYVALUE ]
    [ , ALIGNMENT = alignment ]
    [ , STORAGE = storage ]
    [ , LIKE = like_type ]
    [ , CATEGORY = category ]
    [ , PREFERRED = preferred ]
    [ , DEFAULT = default ]
    [ , ELEMENT = element ]
    [ , DELIMITER = delimiter ]
    [ , COLLATABLE = collatable ]
)

CREATE TYPE name

CREATE TYPE name AS ENUM
    ( [ 'label' [, ... ] ] )
```

参数说明

表 21-127 CREATE TYPE 参数说明

参数	描述	取值范围
name	要创建的类型的名称，可以用模式修饰。	需符合**标识符命名规范**。
attribute_name	复合类型的一个属性（列）的名称。	-
data_type	要成为复合类型的一个列的现有数据类型的名称。	-
collation	与复合类型的列关联的现有排序规则的名称。	-
input_function	将数据从类型的外部文本形式转换为内部形式的函数名。 输入函数可以被声明为有一个cstring类型的参数，或者有三个类型分别为cstring、 oid、integer的参数。 ● cstring参数是以C字符串存在的输入文本。 ● oid参数是该类型自身的OID（对于数组类型则是其元素类型的OID）。 ● integer参数是目标列的typmod（如果知道，不知道则将传递 -1）。 输入函数必须返回一个该数据类型本身的值。通常，一个输入函数应该被声明为STRICT。 如果不是这样，在读到一个NULL输入值时，调用输入函数时第一个参数会是NULL。在这种情况下，该函数必须仍然返回NULL，除非调用函数发生了错误（这种情况主要是想支持域输入函数，域输入函数可能需要拒绝NULL输入）。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
output_function	将数据从类型的内部形式转换为外部文本形式的函数名。 输出函数必须被声明为有一个新数据类型的参数。输出函数必须返回类型cstring。对于NULL值不会调用输出函数。	-
receive_function（可选参数）	将数据从类型的外部二进制形式转换成内部形式的函数名。 如果没有该函数，该类型不能参与到二进制输入中。二进制表达转换成内部形式代价更低，然而却更容易移植（例如，标准的整数数据类型使用网络字节序作为外部二进制表达，而内部表达是机器本地的字节序）。receive_function应该执行足够的检查以确保该值是有效的。 接收函数可以被声明为有一个internal类型的参数，或者有三个类型分别为internal、oid、integer的参数。 ● internal参数是一个指向StringInfo缓冲区的指针，其中保存着接收到的字节串。 ● oid和integer参数和文本输入函数的相同。 接收函数必须返回一个该数据类型本身的值。通常，一个接收函数应该被声明为STRICT。如果不是这样，在读到一个NULL输入值时调用接收函数时第一个参数会是NULL。在这种情况下，该函数必须仍然返回NULL，除非接收函数发生了错误（这种情况主要是想支持域接收函数，域接收函数可能需要拒绝NULL输入）。	-
send_function（可选参数）	将数据从类型的内部形式转换为外部二进制形式的函数名。 如果没有该函数，该类型将不能参与到二进制输出中。发送函数必须被声明为有一个新数据类型的参数。发送函数必须返回类型bytea。对于NULL值不会调用发送函数。	-
type_modifier_input_function（可选参数）参数。	将类型的修饰符数组转换为内部形式的函数名。	-
type_modifier_output_function（可选参数）	将类型的修饰符的内部形式转换为外部文本形式的函数名。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
analyze_function（可选参数）	为该数据类型执行统计分析的函数名的可选参数。 默认情况下，如果该类型有一个默认的B-Tree操作符类，ANALYZE将尝试用类型的“equals”和“less-than”操作符来收集统计信息。这种行为对于非标量类型并不合适，因此可以通过指定一个自定义分析函数来覆盖这种行为。分析函数必须被声明为有一个类型为internal的参数，并且返回一个boolean结果。	-
internallength（可选参数）	一个数字常量，用于指定新类型的内部表达的字节长度。默认为变长。 虽然只有I/O函数和其他为该类型创建的函数才知道新类型的内部表达的细节， 但是内部表达的一些属性必须被向DWS声明。其中最重要的是internallength。基本数据类型可以是定长的（这种情况下internallength是一个正整数）或者是变长的（把internallength设置为VARIABLE，在内部通过把typlen设置为-1表示）。所有变长类型的内部表达都必须以一个4字节整数开始，internallength定义了总长度。	-
PASSEDBYVALUE（可选参数）	表示此数据类型的值需要被传值而不是传引用。传值的类型必须是定长的，并且它们的内部表达不能超过Datum类型（某些机器上是4字节，其他机器上是8字节）的尺寸。	-
alignment（可选参数）	该参数指定数据类型的存储对齐需求。如果被指定，必须是char、int2、int4或者double。默认是int4。 允许的值等同于以1、2、4或8字节边界对齐。要注意变长类型的alignment参数必须至少为4，因为它们需要包含一个int4作为它们的第一个组成部分。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
storage（可选参数）	该数据类型的存储策略。 如果被指定，必须是plain、external、extended或者main。 默认是plain。 ● plain指定该类型的数据将总是被存储在线内并且不会被压缩。（对定长类型只允许plain） ● extended指定系统将首先尝试压缩一个长的数据值，并且将在数据仍然太长的情况下把值移出主表行。 ● external允许值被移出主表， 但是系统将不会尝试对它进行压缩。 ● main允许压缩，但是不鼓励把值移出主表（如果没有其他办法让行的大小变得合适，具有这种存储策略的数据项仍将被移出主表，但比起extended以及external项来，这种存储策略的数据项会被优先考虑保留在主表中）。 除plain之外所有的storage值都暗示该数据类型的函数能处理被TOAST过的值。指定的值仅决定一种可TOAST数据类型的列的默认TOAST存储策略，用户可以使用ALTER TABLE SET STORAGE为列选取其他策略。	-
like_type（可选参数）	与新类型具有相同表达的现有数据类型的名称。会从这个类型中复制internallength、passedbyvalue、 alignment以及storage的值（ 除非在这个CREATE TYPE命令的其他地方用显式说明覆盖）。 当新类型的底层实现是以一种现有的类型为参考时，用这种方式指定表达特别有用。	-
category（可选参数）	此类型的分类码（单个ASCII字符）。 默认为'U'，表示用户定义类型。为了创建自定义分类，也可以选择其他ASCII字符。	-
preferred（可选参数）	如果此类型是其类型分类中的优先类型则为TRUE，否则为FALSE。默认为FALSE。在现有类型分类中创建新的优先类型要非常谨慎， 因为这可能会导致很大的改变。	-
default（可选参数）	数据类型的默认值。如果被省略，默认值是空。 如果用户希望该数据类型的列被默认为某种非空值，可以指定一个默认值。默认值可以用DEFAULT关键词指定（这样一个默认值可以被附加到一个特定列的显式DEFAULT子句覆盖）。	-
element（可选参数）	被创建的类型是一个数组，element指定了数组元素的类型。例如，要定义一个4字节整数的数组（int4）， 应指定ELEMENT = int4。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
delimiter（可选参数）	指定此类型组成的数组中分隔值的定界符。 可以把delimiter设置为一个特定字符，默认的定界符是逗号（,）。注意定界符是与数组元素类型相关的，而不是数组类型本身相关。	-
collatable（可选参数）	如果此类型的操作可以使用排序规则信息，则为TRUE。默认为FALSE。 如果collatable为TRUE，此类型的列定义和表达式可能通过使用COLLATE子句携带有排序规则信息。在该类型上操作的函数的实现负责真正利用这些信息，仅把类型标记为可排序的并不会让它们自动地去使用这类信息。	-
label（可选参数）	与枚举类型的一个值相关的文本标签，其值为长度不超过64个字符的非空字符串。	-

示例

示例一：创建一种复合类型，建表并插入数据以及查询。

```
CREATE TYPE compfoo AS (f1 int, f2 text);
CREATE TABLE t1_compfoo(a int, b compfoo);
CREATE TABLE t2_compfoo(a int, b compfoo);
INSERT INTO t1_compfoo values(1,(1,'demo'));
INSERT INTO t2_compfoo select * from t1_compfoo;
SELECT (b).f1 FROM t1_compfoo;
SELECT * FROM t1_compfoo t1 join t2_compfoo t2 on (t1.b).f1=(t1.b).f1;
```

示例二：创建一个枚举类型，并在表定义中使用它。

```
CREATE TYPE bugstatus AS ENUM ('create', 'modify', 'closed');
CREATE TABLE customer (name text,current_bugstatus bugstatus);
INSERT INTO customer VALUES ('type','create');
SELECT * FROM customer WHERE current_bugstatus = 'create';
```

示例三：编译.so文件，并创建shell类型。

```
CREATE TYPE complex;
```

这个语句的作用是为要定义的类型创建了一个占位符，这样允许在定义其I/O函数时引用该类型。现在可以定义I/O函数，需要注意的是在创建函数时function必须声明为NOT FENCED模式：

```
CREATE FUNCTION
complex_in(cstring)
   RETURNS complex
   AS 'filename'
   LANGUAGE C IMMUTABLE STRICT not fenced;

CREATE FUNCTION
complex_out(complex)
   RETURNS cstring
   AS 'filename'
   LANGUAGE C IMMUTABLE STRICT not fenced;

CREATE FUNCTION
complex_recv(internal)
   RETURNS complex
```

 版权所有 © 华为云计算技术有限公司

```
    AS 'filename'
    LANGUAGE C IMMUTABLE STRICT not fenced;

CREATE FUNCTION
complex_send(complex)
    RETURNS bytea
    AS 'filename'
    LANGUAGE C IMMUTABLE STRICT not fenced;
```

提供该数据类型的完整定义。

```
CREATE TYPE complex (
internallength = 16,
input = complex_in,
output = complex_out,
receive = complex_recv,
send = complex_send,
alignment = double
);
```

input、output、receive及send函数对应的C函数定义如下：

```
--定义结构体Complex如下：
typedef struct Complex {
    double      x;
    double      y;
} Complex;

--定义input函数：
PG_FUNCTION_INFO_V1(complex_in);

Datum
complex_in(PG_FUNCTION_ARGS)
{
    char       *str = PG_GETARG_CSTRING(0);
    double      x,
                y;
    Complex    *result;

    if (sscanf(str, " ( %lf , %lf )", &x, &y) != 2)
        ereport(ERROR,
                (errcode(ERRCODE_INVALID_TEXT_REPRESENTATION),
                 errmsg("invalid input syntax for complex: \"%s\"",
                        str)));

    result = (Complex *) palloc(sizeof(Complex));
    result->x = x;
    result->y = y;
    PG_RETURN_POINTER(result);
}

--定义output函数：
PG_FUNCTION_INFO_V1(complex_out);

Datum
complex_out(PG_FUNCTION_ARGS)
{
        Complex    *complex = (Complex *) PG_GETARG_POINTER(0);
        char       *result;

        result = (char *) palloc(100);
        snprintf(result, 100, "(%g,%g)", complex->x, complex->y);
        PG_RETURN_CSTRING(result);
}

--定义receive函数：
PG_FUNCTION_INFO_V1(complex_recv);

Datum
complex_recv(PG_FUNCTION_ARGS)
```

 版权所有 © 华为云计算技术有限公司

```
{
    StringInfo  buf = (StringInfo) PG_GETARG_POINTER(0);
    Complex    *result;

    result = (Complex *) palloc(sizeof(Complex));
    result->x = pq_getmsgfloat8(buf);
    result->y = pq_getmsgfloat8(buf);
    PG_RETURN_POINTER(result);
}

--定义send函数:
PG_FUNCTION_INFO_V1(complex_send);

Datum
complex_send(PG_FUNCTION_ARGS)
{
    Complex    *complex = (Complex *) PG_GETARG_POINTER(0);
    StringInfoData buf;

    pq_begintypsend(&buf);
    pq_sendfloat8(&buf, complex->x);
    pq_sendfloat8(&buf, complex->y);
    PG_RETURN_BYTEA_P(pq_endtypsend(&buf));
}
```

相关链接

ALTER TYPE，**DROP TYPE**

21.13.2.10 ALTER TYPE

功能描述

修改一个类型的定义。

注意事项

- 要使用ALTER TYPE，必须是该类型的所有者。
- 要修改类型的模式，还必须在新模式上拥有CREATE权限。
- 要修改所有者，必须是新的所有角色的直接或间接成员， 并且该角色必须在此类型的模式上有CREATE权限。 这些限制强制要求修改所有者不能执行任何通过删除和重建该类型无法实现的操作。
- 系统管理员拥有以任何方式修改任意类型的所有权。
- 要增加一个属性或是修改一个属性的类型，也必须有该类型的USAGE权限。

语法格式

- 修改类型。

```
ALTER TYPE name action [, ... ];
ALTER TYPE name OWNER TO { new_owner | CURRENT_USER | SESSION_USER };
ALTER TYPE name RENAME ATTRIBUTE attribute_name TO new_attribute_name [ CASCADE |
RESTRICT ];
ALTER TYPE name RENAME TO new_name;
ALTER TYPE name SET SCHEMA new_schema;
ALTER TYPE name ADD VALUE [ IF NOT EXISTS ] new_enum_value [ { BEFORE | AFTER }
neighbor_enum_value ];
ALTER TYPE name RENAME VALUE existing_enum_value TO new_enum_value;

where action is one of:
    ADD ATTRIBUTE attribute_name data_type [ COLLATE collation ] [ CASCADE | RESTRICT ];
```


版权所有 © 华为云计算技术有限公司

```
    DROP ATTRIBUTE [ IF EXISTS ] attribute_name [ CASCADE | RESTRICT ];
    ALTER ATTRIBUTE attribute_name [ SET DATA ] TYPE data_type [ COLLATE collation ] [ CASCADE |
RESTRICT ];
```

- 给复合类型增加新的属性。
  ```
  ALTER TYPE name ADD ATTRIBUTE attribute_name data_type [ COLLATE collation ] [ CASCADE |
  RESTRICT ];
  ```
- 从复合类型删除一个属性。
  ```
  ALTER TYPE name DROP ATTRIBUTE [ IF EXISTS ] attribute_name [ CASCADE | RESTRICT ];
  ```
- 改变一种复合类型中某个属性的类型。
  ```
  ALTER TYPE name ALTER ATTRIBUTE attribute_name [ SET DATA ] TYPE data_type [ COLLATE
  collation ] [ CASCADE | RESTRICT ];
  ```

说明

ADD ATTRIBUTE、DROP ATTRIBUTE和ALTER ATTRIBUTE选项可以组合成一个列表同时执行。例如，在一条命令中同时增加多个属性或是更改多个属性的类型。

- 改变类型的所有者。
  ```
  ALTER TYPE name OWNER TO { new_owner | CURRENT_USER | SESSION_USER };
  ```
- 改变类型的名称或是一个复合类型中的一个属性的名称。
  ```
  ALTER TYPE name RENAME TO new_name;
  ALTER TYPE name RENAME ATTRIBUTE attribute_name TO new_attribute_name [ CASCADE |
  RESTRICT ];
  ```
- 将类型移至一个新的模式中。
  ```
  ALTER TYPE name SET SCHEMA new_schema;
  ```
- 为枚举类型增加一个新值。
  ```
  ALTER TYPE name ADD VALUE [ IF NOT EXISTS ] new_enum_value [ { BEFORE | AFTER }
  neighbor_enum_value ];
  ```
- 重命名枚举类型的一个标签值。
  ```
  ALTER TYPE name RENAME VALUE existing_enum_value TO new_enum_value;
  ```

参数说明

表 21-128 ALTER TYPE 参数说明

参数	描述	取值范围
name	需要修改的现有类型的名称，可以用模式修饰。	有效的类型名称。
new_name	类型的新名称。	字符串。
new_owner	类型新所有者的用户名。	有效的用户名。
new_schema	类型的新模式。	有效的模式名。
attribute_name	要增加、更改或删除的属性的名称。	-
new_attribute_name	要重命名的属性的新名称。	-
data_type	要新增属性的数据类型，或是要更改的属性的新类型名。	-
new_enum_value	枚举类型新增加的标签值。	字符串。非空且长度不超过64个字节。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
neighbor_enum_value	一个已有枚举标签值，新值应该被增加在紧接着该枚举值之前或者之后的位置上。	-
existing_enum_value	现有的要重命名的枚举值。	字符串。非空且长度不超过64个字节。
CASCADE	自动级联更新需更新类型以及相关联的记录和继承它们的子表。	-
RESTRICT	如果需级联更新类型是已更新类型的关联记录，则拒绝更新。RESTRICT为缺省选项。	-

示例

创建示例复合类型test，枚举类型testdata和用户user_t。

```
CREATE TYPE test AS (col1 int, col text);
CREATE TYPE testdata AS ENUM ('create', 'modify', 'closed');
CREATE USER user_t PASSWORD '{Password}';
```

重命名数据类型test为test1。

```
ALTER TYPE test RENAME TO test1;
```

修改用户定义类型test1的所有者为user_t。

```
ALTER TYPE test1 OWNER TO user_t;
```

把用户定义类型test1的模式改为user_t。

```
ALTER TYPE test1 SET SCHEMA user_t;
```

给用户定义类型test1添加一个新的属性col3。

```
ALTER TYPE user_t.test1 ADD ATTRIBUTE col3 int;
```

给枚举类型testdata添加一个标签值。

```
ALTER TYPE testdata ADD VALUE IF NOT EXISTS 'regress' BEFORE 'closed';
```

重命名枚举类型testdata的一个标签值。

```
ALTER TYPE testdata RENAME VALUE 'create' TO 'new';
```

相关链接

CREATE TYPE，**DROP TYPE**

21.13.2.11 DROP TYPE

功能描述

删除一个用户定义的数据类型。只有类型所有者才有删除权限。

版权所有 © 华为云计算技术有限公司

语法格式

```
DROP TYPE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

参数说明

表 21-129 DROP TYPE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的类型不存在，那么发出一个notice而不是抛出一个错误。	-
name	要删除的类型名（可以有模式修饰）。	已存在的类型名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖该类型的对象（比如字段、函数、操作符等）。 • RESTRICT：如果有依赖对象，则拒绝删除该类型（缺省行为）。

示例

删除compfoo类型。

```
DROP TYPE compfoo cascade;
```

相关链接

ALTER TYPE，**CREATE TYPE**

21.13.3 用户和角色

21.13.3.1 CREATE USER

功能描述

创建新的数据库用户。

注意事项

- 通过CREATE USER创建的用户，默认具有LOGIN权限。
- 通过CREATE USER创建用户的同时系统会在执行该命令的数据库中，为该用户创建一个同名的SCHEMA；其他数据库中，则不自动创建同名的SCHEMA；用户可使用CREATE SCHEMA命令，分别在其他数据库中，为该用户创建同名SCHEMA。
- 系统管理员在普通用户同名SCHEMA下创建的对象，所有者为SCHEMA的同名用户（非系统管理员）。

版权所有 © 华为云计算技术有限公司

- 除系统管理员之外，其他用户即使被授权了SCHEMA的所有权限也无法在普通用户的同名SCHEMA下创建对象，除非把同名SCHEMA相关的角色权限赋予其他用户。具体操作可参考《故障排除》中的“赋予用户schema的all权限后建表仍然报”章节。

⚠ 注意

- **尽量避免**所有业务使用同一个数据库用户运行，请按业务模块规划不同数据库用户。
- **不建议**使用系统管理员用户跑业务，不同模块业务请通过多用户和权限进行访问控制。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
CREATE USER user_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD |
IDENTIFIED BY } { 'password' | DISABLE };
```

其中option子句用于设置权限及属性等信息。

```
{SYSADMIN | NOSYSADMIN}
    | {AUDITADMIN | NOAUDITADMIN}
    | {CREATEDB | NOCREATEDB}
    | {USEFT | NOUSEFT}
    | {CREATEROLE | NOCREATEROLE}
    | {INHERIT | NOINHERIT}
    | {LOGIN | NOLOGIN}
    | {REPLICATION | NOREPLICATION}
    | {INDEPENDENT | NOINDEPENDENT}
    | {VCADMIN | NOVCADMIN}
    | CONNECTION LIMIT connlimit
    | VALID BEGIN 'timestamp'
    | VALID UNTIL 'timestamp'
    | RESOURCE POOL 'respool'
    | USER GROUP 'groupuser'
    | PERM SPACE 'spacelimit'
    | TEMP SPACE 'tmpspacelimit'
    | SPILL SPACE 'spillspacelimit'
    | NODE GROUP logic_cluster_name
    | IN ROLE role_name [, ...]
    | IN GROUP role_name [, ...]
    | ROLE role_name [, ...]
    | ADMIN role_name [, ...]
    | USER role_name [, ...]
    | SYSID uid
    | DEFAULT TABLESPACE tablespace_name
    | PROFILE DEFAULT
    | PROFILE profile_name
    | PGUSER
    | AUTHINFO 'authinfo'
    | PASSWORD EXPIRATION period
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-130 CREATE USER 参数说明

参数	描述	取值范围或默认值
user_name	要创建的用户名。	字符串，需符合标识符的命名规范且最多为63个字符。参见**标识符命名规范**。
password	登录密码。 密码规则如下： • 密码默认不少于8个字符。 • 不能与用户名及用户名倒序相同。 • 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（~!@#$%^&*()-_=+|[{}];:,<.>/?）四类字符中的三类字符。使用范围外的字符会收到告警，但依然允许创建。	字符串。
DISABLE	默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：IAM认证、kerberos或ldap认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。	-
ENCRYPTED | UNENCRYPTED	控制密码存储在系统表里的密码是否加密。（如果没有指定，那么缺省的行为由配置参数password_encryption_type控制。）按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在DWS中禁止使用。因为系统无法对指定的加密密码字符串进行解密，所以如果目前的密码字符串已经使用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的密码。	-
SYSADMIN | NOSYSADMIN	决定新用户是否为“系统管理员”，具有SYSADMIN属性的用户拥有系统最高权限。	缺省为NOSYSADMIN。
AUDITADMIN | NOAUDITADMIN	定义用户是否有审计管理属性。	缺省为NOAUDITADMIN。
CREATEDB | NOCREATEDB	控制新用户是否能创建数据库。 新用户没有创建数据库的权限。	缺省为NOCREATEDB。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或默认值
USEFT | NOUSEFT	决定一个新用户是否能操作外表，包括：新建外表、删除外表、修改外表、读写外表。 缺省为NOUSEFT，表示新建用户默认不能操作外表。 新用户没有操作外表的权限。	-
CREATEROLE | NOCREATEROLE	控制用户是否可以创建新角色或用户（也就是执行CREATE ROLE和CREATE USER）。拥有CREATEROLE权限的用户也可以修改和删除其他用户或角色。	缺省为NOCREATEROLE。
INHERIT | NOINHERIT	控制用户是否可以“继承”它所在组的用户的权限。不推荐使用。	-
LOGIN | NOLOGIN	控制用户是否可以登录数据库。具有LOGIN属性的用户才可以登录数据库。	缺省为LOGIN。
REPLICATION | NOREPLICATION	控制用户是否允许流复制或设置系统为备份模式。REPLICATION属性是特定的用户，仅用于复制。	缺省为NOREPLICATION。
INDEPENDENT | NOINDEPENDENT	定义私有、独立的用户。具有INDEPENDENT属性的用户，管理员对其进行的控制、访问的权限被分离，具体规则如下： ● 未经INDEPENDENT用户授权，管理员无权对其表对象进行增、删、查、改、拷贝、授权操作。 ● 未经INDEPENDENT用户授权，管理员无权修改INDEPENDENT用户的继承关系。 ● 管理员无权修改INDEPENDENT用户的表对象的属主。 ● 管理员无权去除INDEPENDENT用户的INDEPENDENT属性。 ● 管理员无权修改INDEPENDENT用户的数据库密码，INDEPENDENT用户需管理好自身密码，密码丢失无法重置。 ● 管理员属性用户不允许定义修改为INDEPENDENT属性。	-
VCADMIN | NOVCADMIN	定义逻辑集群管理员用户。具有逻辑集群管理员属性的用户，和普通用户相比，有如下额外权限： ● 在所关联逻辑集群中创建、修改和删除资源池的权限。 ● 将所关联的逻辑集群的访问权限授予其他用户或角色，或回收其他用户或角色对关联逻辑集群的访问权限。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或默认值
CONNECTION LIMIT	定义用户在单个CN上可以使用的并发连接数量。 为保证集群正常使用，connection limit的最小值是集群中CN的数目。在集群执行ANALYZE时，其他CN节点会连接当前执行ANALYZE的CN节点来同步元数据。例如集群中有3个CN节点，那么connection limit应该设置为大于等于3。	整数，大于等于-1。 缺省值为-1，表示没有限制。
VALID BEGIN	设置用户生效的时间戳。如果省略了该子句，用户无有效开始时间限制。	-
VALID UNTIL	设置用户失效的时间戳，即过了这个时间后，账户不可使用。如果省略了该子句，用户无有效结束时间限制。	-
RESOURCE POOL	设置用户使用的resource pool名称，该名称属于系统表：pg_resource_pool	-
USER GROUP 'groupuser'	创建用户的子用户。	-
PERM SPACE	设置用户永久表存储空间限额。 space_limit，表示永久表存储空间上限。	space_limit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。
TEMP SPACE	设置用户临时表存储空间限额。 tmpspacelimit，表示临时表存储空间限额。	tmpspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。
SPILL SPACE	设置用户算子落盘空间限额。 spillspacelimit，表示算子落盘空间限额。	spillspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。
NODE GROUP	设置用户关联的逻辑集群名称。如果需要关联的逻辑集群名称包含大写字符或特殊字符，指定逻辑集群名称时需要加双引号。	-
IN ROLE	新用户立即拥有IN ROLE子句中列出的一个或多个现有用户拥有的权限。不推荐使用。	-
IN GROUP	IN GROUP是IN ROLE过时的拼法。不推荐使用。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或默认值
ROLE	ROLE子句列出一个或多个现有的用户，它们将自动添加为这个新用户的成员，拥有新用户所有的权限。	-
ADMIN	ADMIN子句类似ROLE子句，不同的是ADMIN后的用户可以把新用户的权限赋给其他用户。	-
USER	USER子句是ROLE子句过时的拼法。	-
SYSID	SYSID子句将被忽略，无实际意义。	-
DEFAULT TABLESPACE	DEFAULT TABLESPACE子句将被忽略，无实际意义。	-
PROFILE	PROFILE子句将被忽略，无实际意义。	-
PGUSER	该属性用于兼容开源Postgres的连接通讯，开源的Postgres客户端接口（推荐使用Postgres 9.2.19版本的相关客户端接口）可以使用具有该属性的数据库用户连接数据库。 该属性只用于兼容连接过程，而由于本产品与Postgres的内核差异导致的不兼容，不在此属性控制范围内。	-
AUTHINFO 'authinfo'	该属性用于指定用户认证类型，authinfo为类型说明字符串，大小写敏感。 当前仅支持LDAP类型，对应的类型说明字符串为'ldap'。LDAP属于外部认证，故需要同时指定PASSWORD DISABLE。 ● authinfo可以加上LDAP认证的额外信息，比如LDAP认证中的fulluser，fulluser等同于ldapprefix+username+ldapsuffix。当authinfo为'ldap'，表示用户认证类型为LDAP，此时ldapprefix和ldapsuffix的信息由pg_hba.conf中匹配的记录提供。 ● ALTER ROLE时不允许用户切换认证类型，仅允许LDAP用户修改LDAP属性。	-
PASSWORD EXPIRATION period	声明该用户的登录密码过期天数，登录密码过期之前用户需要及时修改密码。登录密码过期后用户无法登录，需要请管理员设置新的登录密码后登录。	整数，-1~999。 ● -1（缺省值），表示没有过期限制。 ● 0，表示用户登录密码立即过期。

版权所有 © 华为云计算技术有限公司

示例

创建用户new_u1。

```
CREATE USER new_u1 PASSWORD '{password}';
```

下面语句与上面的等价。

```
CREATE USER new_u1 IDENTIFIED BY '{password}';
```

如果创建有“创建数据库”权限的用户，则需要指定CREATEDB关键字。

```
CREATE USER new_u1 CREATEDB PASSWORD '{password}';
```

相关链接

ALTER USER，**CREATE ROLE**，**DROP USER**

21.13.3.2 ALTER USER

功能描述

修改数据库用户的属性。

注意事项

ALTER USER修改的会话参数只针对指定的用户，且在下一次会话中有效。

语法格式

- 修改用户的权限等信息。

```
ALTER USER user_name [ [ WITH ] option [ ... ] ];
```

其中option子句为。

```
{ CREATEDB | NOCREATEDB }
    | { CREATEROLE | NOCREATEROLE }
    | { INHERIT | NOINHERIT }
    | { AUDITADMIN | NOAUDITADMIN }
    | { SYSADMIN | NOSYSADMIN }
    | { USEFT | NOUSEFT }
    | { LOGIN | NOLOGIN }
    | { REPLICATION | NOREPLICATION }
    | {INDEPENDENT | NOINDEPENDENT}
    | {VCADMIN | NOVCADMIN}
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD { 'password' | DISABLE }
    | [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY { 'password' [ REPLACE 'old_password' ] |
DISABLE }
    | VALID BEGIN 'timestamp'
    | VALID UNTIL 'timestamp'
    | RESOURCE POOL 'respool'
    | USER GROUP 'groupuser'
    | PERM SPACE 'spacelimit'
    | TEMP SPACE 'tmpspacelimit'
    | SPILL SPACE 'spillspacelimit'
    | NODE GROUP logic_cluster_name
    | ACCOUNT { LOCK | UNLOCK }
    | PGUSER
    | AUTHINFO 'authinfo'
    | PASSWORD EXPIRATION period
```

- 修改用户名。

```
ALTER USER user_name
    RENAME TO new_name;
```


版权所有 © 华为云计算技术有限公司

- 修改与用户关联的指定会话参数值。
```
ALTER USER user_name
    SET configuration_parameter { { TO | = } { value | DEFAULT } | FROM CURRENT };
```
- 重置与用户关联的指定会话参数值。
```
ALTER USER user_name
    RESET { configuration_parameter | ALL };
```

参数说明

表 21-131 ALTER USER 参数说明

参数	描述	取值范围及默认值
user_name	需要修改的用户名。	有效的用户名。
new_name	用户的新名称。	字符串，需符合**标识符命名规范**。
CREATEDB | NOCREATEDB	控制用户是否能创建数据库。 新用户没有创建数据库的权限。	缺省为NOCREATEDB。
CREATEROLE | NOCREATEROLE	控制用户是否可以创建新用户或角色（也就是执行CREATE ROLE和CREATE USER）。拥有CREATEROLE权限的用户也可以修改和删除其他用户。	缺省为NOCREATEROLE。
INHERIT | NOINHERIT	控制用户是否可以“继承”它所在组的用户的权限。不推荐使用。	-
AUDITADMIN | NOAUDITADMIN	定义用户是否有审计管理属性。	缺省为NOAUDITADMIN。
SYSADMIN | NOSYSADMIN	控制用户是否为“系统管理员”，具有SYSADMIN属性的用户拥有系统最高权限。	缺省为NOSYSADMIN。
USEFT | NOUSEFT	控制用户是否有权限操作外表，包括：新建外表、删除外表、修改外表、读写外表。 缺省为NOUSEFT，表示新建用户默认不能操作外表。 新用户没有操作外表的权限。	-
LOGIN | NOLOGIN	控制用户是否可以登录数据库。具有LOGIN属性的用户才可以登录数据库。	缺省为LOGIN。
REPLICATION | NOREPLICATION	控制用户是否允许流复制或设置系统为备份模式。REPLICATION属性是特定的用户，仅用于复制。	缺省为NOREPLICATION。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围及默认值
INDEPENDENT | NOINDEPENDENT	定义私有、独立的用户。具有INDEPENDENT属性的用户，管理员对其进行的控制、访问的权限被分离，具体规则如下： ● 未经INDEPENDENT用户授权，管理员无权对其表对象进行增、删、查、改、拷贝、授权操作。 ● 未经INDEPENDENT用户授权，管理员无权修改INDEPENDENT用户的继承关系。 ● 管理员无权修改INDEPENDENT用户的表对象的属主。 ● 管理员无权去除INDEPENDENT用户的INDEPENDENT属性。 ● 管理员无权修改INDEPENDENT用户的数据库密码，INDEPENDENT用户需管理好自身密码，密码丢失无法重置。 ● 管理员属性用户不允许定义修改为INDEPENDENT属性。	-
VCADMIN | NOVCADMIN	定义逻辑集群管理员用户。具有逻辑集群管理员属性的用户，和普通用户相比，有如下额外权限： ● 在所关联逻辑集群中创建、修改和删除资源池的权限。 ● 将所关联的逻辑集群的访问权限授予其他用户或角色，或回收其他用户或角色对关联逻辑集群的访问权限。	-
CONNECTION LIMIT	定义用户在单个CN上可以使用的并发连接数量。 为保证集群正常使用，connection limit的最小值是集群中CN的数目。在集群执行ANALYZE时，其他CN节点会连接当前执行ANALYZE的CN节点来同步元数据。例如集群中有3个CN节点，那么connection limit应该设置为大于等于3。	取值为整数，大于等于-1。 缺省值为-1，表示没有限制。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围及默认值
ENCRYPTED | UNENCRYPTED	控制密码存储在系统表里的密码是否加密。（如果没有指定，那么缺省的行为由配置参数password_encryption_type控制。）按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在DWS中禁止使用。因为系统无法对指定的加密密码字符串进行解密，所以如果目前的密码字符串已经使用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的密码。 ● password，登录密码。 ● DISABLE，默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：IAM认证、kerberos或ldap认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。	密码规则： ● 密码默认不少于8个字符。 ● 不能与用户名及用户名倒序相同。 ● 至少包含大写字母（A-Z）、小写字母（a-z）、数字（0-9）、非字母数字字符（~!@#$ %^&*()-_=+| [{}];:,<.>/?）四类字符中的三类字符。使用范围外的字符会收到告警，但依然允许创建。
VALID BEGIN	设置用户生效的时间戳。如果省略了该子句，用户无有效开始时间限制。	-
VALID UNTIL	设置用户失效的时间戳，即过了这个时间后，账户不可使用。如果省略了该子句，用户无有效结束时间限制。	-
RESOURCE POOL	设置用户使用的资源池名称，该名称属于系统表：pg_resource_pool	-
USER GROUP 'groupuser'	创建用户的子用户。	-
PERM SPACE	设置用户永久表存储空间限额。 space_limit，表示永久表存储空间上限。	space_limit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。
TEMP SPACE	设置用户临时表存储空间限额。 tmpspacelimit，表示临时表存储空间限额。	tmpspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围及默认值
SPILL SPACE	设置用户算子落盘空间限额。 spillspacelimit，表示算子落盘空间限额。	spillspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。 0表示不限制。
NODE GROUP	设置用户关联的逻辑集群名称。如果需要关联的逻辑集群名称包含大写字符或特殊字符，指定逻辑集群名称时需要加双引号。	-
ACCOUNT LOCK | ACCOUNT UNLOCK	锁定或解锁用户账号： • ACCOUNT LOCK：锁定账户，禁止登录数据库。 • ACCOUNT UNLOCK：解锁账户，允许登录数据库。	-
PGUSER	该属性用于兼容开源Postgres的连接通讯，开源的Postgres客户端接口（推荐使用Postgres 9.2.19版本的相关客户端接口）可以使用具有该属性的数据库用户连接数据库。 当前版本不允许修改用户的PGUSER属性。更多说明内容请参考**PGUSER参数建议**。	-
AUTHINFO 'authinfo'	该属性用于指定用户认证类型，authinfo为类型说明字符串，大小写敏感。 当前仅支持LDAP类型，对应的类型说明字符串为'ldap'。LDAP属于外部认证，故需要同时指定PASSWORD DISABLE。 • authinfo可以加上LDAP认证的额外信息，比如LDAP认证中的fulluser，fulluser等同于ldapprefix+username+ldapsuffix。当authinfo为'ldap'，表示用户认证类型为LDAP，此时ldapprefix和ldapsuffix的信息由pg_hba.conf中匹配的记录提供。 • ALTER ROLE时不允许用户切换认证类型，仅允许LDAP用户修改LDAP属性。	-
PASSWORD EXPIRATION period	设置用户的登录密码过期天数，登录密码过期之前用户需要及时修改密码。登录密码过期后用户无法登录，需要管理员设置新的登录密码后登录。	整数，-1~999。 • -1，表示没有过期限制，缺省值。 • 0，表示用户登录密码立即过期。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围及默认值
SET configuration_parameter	设置用户的会话参数。ALTER USER中修改的会话参数只针对指定的用户，且在下一次该用户启动的会话中有效。 • DEFAULT，表示清除configuration_parameter参数的值，configuration_parameter参数的值将继承本角色新产生的SESSION的默认值。 • FROM CURRENT，取当前会话中的值设置为configuration_parameter参数的值。	configuration_parameter和value的取值请参见**SET**。
RESET configuration_parameter/ALL	清除configuration_parameter参数的值。与SET configuration_parameter TO DEFAULT的效果相同。 ALL表示清除所有参数的值。	-

PGUSER 参数建议

PGUSER属性只用于兼容连接过程，而由于本产品与Postgres的内核差异导致的不兼容，不在此属性控制范围内。

由于具有PGUSER属性的用户的认证方式与其他用户不同，开源客户端的报错信息可能导致数据库用户PGUSER属性被枚举，建议使用本产品自有的客户端。例如：

```
#normaluser是不具有PGUSER属性的用户，psql是Postgres的客户端工具
pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U normaluser
psql: authentication method 10 not supported

#pguser用户是具有PGUSER属性的用户
pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U pguser
Password for user pguser:
```

示例

创建示例用户u1。

```
DROP USER IF EXISTS u1 CASCADE;
CREATE USER u1 PASSWORD '{Password}';
```

修改用户u1的登录密码。

```
ALTER USER u1 IDENTIFIED BY '{new_Password}' REPLACE '{Password}';
```

为用户u1追加CREATEROLE权限。

```
ALTER USER u1 CREATEROLE;
```

设置用户u1的最大数据库连接数为10。

```
ALTER USER u1 with  CONNECTION LIMIT 10;
```

将与用户u1关联的会话参数enable_seqscan值设置为on，设置成功后，在下一次启动的会话中生效。

```
ALTER USER u1 SET enable_seqscan TO on;
```


版权所有 © 华为云计算技术有限公司

重置u1的enable_seqscan参数。

```
ALTER USER u1 RESET enable_seqscan;
```

锁定u1账户。

```
ALTER USER u1 ACCOUNT LOCK;
```

相关链接

CREATE ROLE，**CREATE USER**，**DROP USER**

21.13.3.3 DROP USER

功能描述

删除用户，同时会删除同名的schema。

注意事项

- 须使用CASCADE级联删除依赖用户的对象（除数据库外）。当删除用户的级联对象时，如果级联对象处于锁定状态，则此级联对象无法被删除，直到对象被解锁或锁定级联对象的进程被终止。
- 在数据库中删除用户时，如果依赖用户的对象在其他数据库中或者依赖用户的对象是其他数据库，请用户先手动删除其他数据库中的依赖对象或直接删除依赖数据库，再删除用户。即drop user不支持跨数据库进行级联删除。
- 在多租户场景下，删除组用户时，业务用户也会同时被删除，如果指定CASCADE级联删除，那么删除业务用户时同时也指定CASCADE。如果在删除某个用户失败时，会报错，同时其他用户也无法成功删除。
- 如果用户A创建的GDS外表指定的错误表在用户B的schema下，则无法通过drop user指定CASCADE关键字直接删除用户B。
- DROP USER若提示role is being used by other users错误，可能原因为CLEAN CONNECTION过程存在线程无法及时响应信号，出现连接清理不完全的情况，需要再次执行CLEAN CONNECTION。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
DROP USER [ IF EXISTS ] user_name [, ...] [ CASCADE | RESTRICT ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-132 DROP USER 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的用户不存在，发出一个notice而不是抛出一个错误。	-
user_name	待删除的用户名。	已存在的用户名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。 **说明** 在DWS中，存在一个配置参数enable_kill_query，此参数在配置文件postgresql.conf中。此参数影响级联删除用户对象的行为： • 当参数enable_kill_query为on ，且使用CASCADE模式删除用户时，会自动kill锁定用户级联对象的进程，并删除用户。 • 当参数enable_kill_query为off，且使用CASCADE模式删除用户时，会等待锁定级联对象的进程结束之后再删除用户。	• CASCADE：级联删除依赖用户的表等对象。级联删除用户的时候，会删除掉owner是这个用户的对象，并清理掉其他对象对这个用户的授权信息。 • RESTRICT：如果用户还有任何依赖的对象，则拒绝删除该用户（缺省行为）。

示例

删除用户jim。

```
DROP USER jim CASCADE;
```

相关链接

ALTER USER，**CREATE USER**

21.13.3.4 CREATE ROLE

功能描述

创建角色。

角色是拥有数据库对象和权限的实体。在不同的环境中角色可以认为是一个用户，一个组或者兼顾两者。

注意事项

- 在数据库中添加一个新角色，角色无登录权限。
- 创建角色的用户必须具备CREATE ROLE的权限或者是系统管理员。

版权所有 © 华为云计算技术有限公司

语法格式

```
CREATE ROLE role_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD |
IDENTIFIED BY } { 'password' | DISABLE };
```

其中角色信息设置子句option语法为：

```
{SYSADMIN | NOSYSADMIN}
    | {AUDITADMIN | NOAUDITADMIN}
    | {CREATEDB | NOCREATEDB}
    | {USEFT | NOUSEFT}
    | {CREATEROLE | NOCREATEROLE}
    | {INHERIT | NOINHERIT}
    | {LOGIN | NOLOGIN}
    | {REPLICATION | NOREPLICATION}
    | {INDEPENDENT | NOINDEPENDENT}
    | {VCADMIN | NOVCADMIN}
    | CONNECTION LIMIT connlimit
    | VALID BEGIN 'timestamp'
    | VALID UNTIL 'timestamp'
    | RESOURCE POOL 'respool'
    | USER GROUP 'groupuser'
    | PERM SPACE 'spacelimit'
    | TEMP SPACE 'tmpspacelimit'
    | SPILL SPACE 'spillspacelimit'
    | NODE GROUP logic_cluster_name
    | IN ROLE role_name [, ...]
    | IN GROUP role_name [, ...]
    | ROLE role_name [, ...]
    | ADMIN role_name [, ...]
    | USER role_name [, ...]
    | SYSID uid
    | DEFAULT TABLESPACE tablespace_name
    | PROFILE DEFAULT
    | PROFILE profile_name
    | PGUSER
    | AUTHINFO 'authinfo'
    | PASSWORD EXPIRATION period
```

参数说明

表 21-133 CREATE ROLE 参数说明

参数	描述	取值范围
role_name	角色名称。	字符串，需符合标识符的命名规范，且最多为63个字符。参见**标识符命名规范**。
password	登录密码。	密码规则如下： • 密码默认不少于8个字符。 • 不能与用户名及用户名倒序相同。 • 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（~!@#$%^&*()-_=+|[{}];:,<.>/?）四类字符中的三类字符。使用范围外的字符会收到告警，但依然允许创建。 取值为字符串。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
DISABLE	默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。	禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：IAM认证、kerberos、ldap认证或oneaccess认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。
ENCRYPTED | UNENCRYPTED	控制密码存储在系统表里的密码是否加密。	如果没有指定，那么缺省的行为由配置参数password_encryption_type控制。按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在DWS中禁止使用。因为系统无法对指定的加密密码字符串进行解密，所以如果目前的密码字符串已经使用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的密码。
SYSADMIN | NOSYSADMIN	决定一个新角色是否为“系统管理员”，具有SYSADMIN属性的角色拥有系统最高权限。	默认为NOSYSADMIN。
AUDITADMIN | NOAUDITADMIN	定义角色是否有审计管理属性。	默认为NOAUDITADMIN。
CREATEDB | NOCREATEDB	决定一个新角色是否能创建数据库。新角色没有创建数据库的权限。	默认为NOCREATEDB。
USEFT | NOUSEFT	决定一个新角色是否能操作外表，包括：新建外表、删除外表、修改外表、读写外表。 新角色没有操作外表的权限。	缺省为NOUSEFT。 **说明** 8.2.0及以上集群版本中，若GUC参数security_enable_options的foreign_table_options选项打开，即表示允许使用外表操作功能，不需要再赋予用户useft权限。
CREATEROLE | NOCREATEROLE	决定一个角色是否可以创建新角色（也就是执行CREATE ROLE和CREATE USER）。一个拥有CREATEROLE权限的角色也可以修改和删除其他角色。	默认为NOCREATEROLE。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
INHERIT | NOINHERIT	这些子句决定一个角色是否“继承”它所在组的角色的权限。不推荐使用。	-
LOGIN | NOLOGIN	具有LOGIN属性的角色才可以登录数据库。一个拥有LOGIN属性的角色可以认为是一个用户。	默认为NOLOGIN。
REPLICATION | NOREPLICATION	定义角色是否允许流复制或设置系统为备份模式。REPLICATION属性是特定的角色，仅用于复制。	默认为NOREPLICATION。
INDEPENDENT | NOINDEPENDENT	定义私有、独立的角色。具有INDEPENDENT属性的角色，管理员对其进行的控制、访问的权限被分离。	具体规则如下： ● 未经INDEPENDENT角色授权，管理员无权对其表对象进行增、删、查、改、拷贝、授权操作。 ● 未经INDEPENDENT角色授权，管理员无权修改INDEPENDENT角色的继承关系。 ● 管理员无权修改INDEPENDENT角色的表对象的属主。 ● 管理员无权去除INDEPENDENT角色的INDEPENDENT属性。 ● 管理员无权修改INDEPENDENT角色的数据库密码，INDEPENDENT角色需管理好自身密码，密码丢失无法重置。 ● 管理员属性用户不允许定义修改为INDEPENDENT属性。
VCADMIN | NOVCADMIN	定义逻辑集群管理员角色。具有逻辑集群管理员属性的角色，和普通用户相比，有如下额外权限： ● 在所关联逻辑集群中创建、修改和删除资源池的权限。 ● 将所关联的逻辑集群的访问权限授予其他用户或角色，或回收其他用户或角色对关联逻辑集群的访问权限。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CONNECTION LIMIT	声明该角色在单个CN上可以使用的并发连接数量。	整数，>=-1。缺省值为-1，表示没有限制。 **须知** 为保证集群正常使用，connection limit的最小值是集群中CN的数目。在集群做ANALYZE时，其他CN节点会连接当前做ANALYZE的CN节点来同步元数据。例如集群中有3个CN节点，那么connection limit应该设置为>=3。
VALID BEGIN	设置角色生效的时间戳。如果省略了该子句，角色无有效开始时间限制。	-
VALID UNTIL	设置角色失效的时间戳。如果省略了该子句，角色无有效结束时间限制。	-
RESOURCE POOL	设置角色使用的resource pool名字，该名字属于系统表：pg_resource_pool。	-
USER GROUP 'groupuser'	创建一个user的子用户。	-
PERM SPACE	设置用户永久表存储空间限额。	space_limit：永久表存储空间上限。取值范围：字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
TEMP SPACE	设置用户临时表存储空间限额。	tmpspacelimit：临时表存储空间限额。取值范围：字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
SPILL SPACE	设置用户算子落盘空间限额。	spillspacelimit：算子落盘空间限额。取值范围：字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
NODE GROUP	设置用户关联的逻辑集群名称。如果需要关联的逻辑集群名称包含大写字符或特殊字符，指定逻辑集群名称时需要加双引号。	-
IN ROLE	新角色立即拥有IN ROLE子句中列出的一个或多个现有角色拥有的权限。不推荐使用。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
IN GROUP	IN GROUP是IN ROLE过时的拼法。不推荐使用。	-
ROLE	ROLE子句列出一个或多个现有的角色，它们将自动添加为这个新角色的成员，拥有新角色所有的权限。	-
ADMIN	ADMIN子句类似ROLE子句，不同的是ADMIN后的角色可以把新角色的权限赋给其他角色。	-
USER	USER子句是ROLE子句过时的拼法。	-
SYSID	SYSID子句将被忽略，无实际意义。	-
DEFAULT TABLESPACE	DEFAULT TABLESPACE子句将被忽略，无实际意义。	-
PROFILE	PROFILE子句将被忽略，无实际意义。	-
PGUSER	该属性用于兼容开源Postgres的连接通讯，开源的Postgres客户端接口（推荐使用Postgres 9.2.19版本的相关客户端接口）可以使用具有该属性的数据库用户连接数据库。	该属性只用于兼容连接过程，而由于本产品与Postgres的内核差异导致的不兼容，不在此属性控制范围内。 由于具有PGUSER属性的用户的认证方式与其他用户不同，开源客户端的报错信息可能导致数据库用户PGUSER属性被枚举，建议使用本产品自有的客户端。例如： #normaluser是不具有PGUSER属性的用户，psql是Postgres的客户端工具 pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U normaluser psql: authentication method 10 not supported #pguser用户是具有PGUSER属性的用户 pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U pguser Password for user pguser:

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
AUTHINFO 'authinfo'	该属性用于指定角色认证类型，authinfo为类型说明字符串，大小写敏感。当前仅支持LDAP和OneAccess类型，对应的类型说明字符串为“ldap”和“oneaccess”。LDAP和OneAccess属于外部认证，故需要同时指定PASSWORD DISABLE。	• authinfo可以加上LDAP认证的额外信息，比如LDAP认证中的fulluser，fulluser等同于ldapprefix+username +ldapsuffix。当authinfo为“ldap”，表示角色认证类型为LDAP，此时ldapprefix和ldapsuffix的信息由pg_hba.conf中匹配的记录提供。 • 在支持OneAccess认证方式时，authinfo必须有oneaccessClientId和domain信息，满足'oneaccessClientId=xxxx,domain=xxxx' 格式。oneaccessClientId是oneaccess用户的ID，最大长度为256字符；domain是oneaccess的访问域名，最大长度为64字符。 • ALTER ROLE时不允许用户切换认证类型，仅允许LDAP用户修改LDAP属性或OneAccess用户修改OneAccess属性。 • OneAccess是通过内部代码逻辑识别的，无需配置pg_hba.conf，认证信息由authinfo给出。
PASSWORD EXPIRATION period	声明该角色的登录密码过期天数，登录密码过期之前用户需要及时修改密码。登录密码过期后用户无法登录，需要请管理员设置新的登录密码后登录。	整数，-1~999。缺省值为-1，表示没有过期限制；0表示用户登录密码立即过期。

示例

创建一个角色，名为manager。

```
CREATE ROLE manager IDENTIFIED BY '{password}';
```

创建一个角色，从2015年1月1日开始生效，到2026年1月1日失效。

```
CREATE ROLE miriam WITH LOGIN PASSWORD '{password}' VALID BEGIN '2015-01-01' VALID UNTIL '2026-01-01';
```

创建一个角色，认证类型是LDAP，LDAP认证的其他信息由pg_hba.conf提供。

 版权所有 © 华为云计算技术有限公司

```
CREATE ROLE role1 WITH LOGIN AUTHINFO 'ldap' PASSWORD DISABLE;
```

创建一个角色，认证类型是LDAP，LDAP认证的fulluser信息在创建时指定，此时ldap大小写敏感，需要用单引号包含。

```
CREATE ROLE role2 WITH LOGIN AUTHINFO 'ldapcn=role2,cn=user,dc=lework,dc=com' PASSWORD DISABLE;
```

创建一个角色，认证类型是OneAccess，OneAccess认证的authinfo信息在创建时指定，此时oneaccess大小写敏感，需要用单引号包括。

```
CREATE ROLE role3 WITH LOGIN AUTHINFO
'oneaccessClientId=AbCd123,domain=example.exampleoneaccess.com' PASSWORD DISABLE;
```

创建一个角色，登录密码有效期是30天。

```
CREATE ROLE role4 WITH LOGIN PASSWORD '{password}' PASSWORD EXPIRATION 30;
```

相关链接

SET ROLE，**ALTER ROLE**，**DROP ROLE**，**GRANT**，**REVOKE**

21.13.3.5 ALTER ROLE

功能描述

修改角色属性。

注意事项

无。

语法格式

- 修改角色的权限。

  ```
  ALTER ROLE role_name [ [ WITH ] option [ ... ] ];
  ```

  其中权限项子句option为。

  ```
  {CREATEDB | NOCREATEDB}
    | {CREATEROLE | NOCREATEROLE}
    | {INHERIT | NOINHERIT}
    | {AUDITADMIN | NOAUDITADMIN}
    | {SYSADMIN | NOSYSADMIN}
    | {USEFT | NOUSEFT}
    | {LOGIN | NOLOGIN}
    | {REPLICATION | NOREPLICATION}
    | {INDEPENDENT | NOINDEPENDENT}
    | {VCADMIN | NOVCADMIN}
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
    | [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY 'password' [ REPLACE 'old_password' ]
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD { 'password' | DISABLE }
    | [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY { 'password' [ REPLACE 'old_password' ] |
  DISABLE }
    | VALID BEGIN 'timestamp'
    | VALID UNTIL 'timestamp'
    | RESOURCE POOL 'respool'
    | USER GROUP 'groupuser'
    | PERM SPACE 'spacelimit'
    | TEMP SPACE 'tmpspacelimit'
    | SPILL SPACE 'spillspacelimit'
    | NODE GROUP logic_cluster_name
    | ACCOUNT { LOCK | UNLOCK }
    | PGUSER
  ```


版权所有 © 华为云计算技术有限公司

```
  | AUTHINFO 'authinfo'
  | PASSWORD EXPIRATION period
```

- 修改角色的名称。

```
ALTER ROLE role_name
  RENAME TO new_name;
```

- 设置角色的配置参数。

```
ALTER ROLE role_name [ IN DATABASE database_name ]
  SET configuration_parameter {{ TO | = } { value | DEFAULT } | FROM CURRENT};
```

- 重置角色的配置参数。

```
ALTER ROLE role_name
  [ IN DATABASE database_name ] RESET {configuration_parameter|ALL};
```

参数说明

表 21-134 ALTER ROLE 参数说明

参数	描述	取值范围
role_name	需要修改的角色名。	已存在的角色名。
new_name	角色的新名称。	字符串，需符合**标识符命名规范**。
CREATEDB | NOCREATEDB	决定一个新角色是否能创建数据库。缺省为NOCREATEDB。 新角色没有创建数据库的权限。	-
CREATEROLE | NOCREATEROLE	决定一个角色是否可以创建新角色（也就是执行CREATE ROLE和CREATE USER）。 缺省为NOCREATEROLE。 拥有CREATEROLE权限的角色也可以修改和删除其他角色。	-
INHERIT | NOINHERIT	决定一个角色是否“继承”它所在组的角色的权限。不推荐使用。	-
AUDITADMIN | NOAUDITADMIN	定义角色是否有审计管理属性。 缺省为NOAUDITADMIN。	-
SYSADMIN | NOSYSADMIN	决定一个新角色是否为“系统管理员”，具有SYSADMIN属性的角色拥有系统最高权限。 缺省为NOSYSADMIN。	-
USEFT | NOUSEFT	决定一个新角色是否能操作外表，包括：新建外表、删除外表、修改外表、读写外表。 新角色没有操作外表的权限。 缺省为NOUSEFT，表示新建角色默认不能操作外表。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
LOGIN | NOLOGIN	定义用户是否可以登录数据库。具有LOGIN属性的角色才可以登录数据库。一个拥有LOGIN属性的角色可以认为是一个用户。 缺省为NOLOGIN。	-
REPLICATION | NOREPLICATION	定义角色是否允许流复制或设置系统为备份模式。REPLICATION属性是特定的角色，仅用于复制。 缺省为NOREPLICATION。	-
INDEPENDENT | NOINDEPENDENT	定义私有、独立的角色。具有INDEPENDENT属性的角色，管理员对其进行的控制、访问的权限被分离，具体规则如下： ● 未经INDEPENDENT角色授权，管理员无权对其表对象进行增、删、查、改、拷贝、授权操作。 ● 未经INDEPENDENT角色授权，管理员无权修改INDEPENDENT角色的继承关系。 ● 管理员无权修改INDEPENDENT角色的表对象的属主。 ● 管理员无权去除INDEPENDENT角色的INDEPENDENT属性。 ● 管理员无权修改INDEPENDENT角色的数据库密码，INDEPENDENT角色需管理好自身密码，密码丢失无法重置。 ● 管理员属性用户不允许定义修改为INDEPENDENT属性。	-
VCADMIN | NOVCADMIN	定义逻辑集群管理员角色。具有逻辑集群管理员属性的角色，和普通用户相比，有如下额外权限： ● 在所关联逻辑集群中创建、修改和删除资源池的权限。 ● 将所关联的逻辑集群的访问权限授予其他用户或角色，或回收其他用户或角色对关联逻辑集群的访问权限。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CONNECTION LIMIT	定义角色在单个CN上可以使用的并发连接数量。 为保证集群正常使用，connection limit的最小值是集群中CN的数目。在集群执行ANALYZE时，其他CN节点会连接当前执行ANALYZE的CN节点来同步元数据。例如集群中有3个CN节点，那么connection limit应该设置为大于等于3。	整数，大于等于-1。 缺省值为-1，表示没有限制。
ENCRYPTED | UNENCRYPTED	控制密码存储在系统表里的密码是否加密。（如果没有指定，那么缺省的行为由配置参数password_encryption_type控制。）按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在DWS中禁止使用。因为系统无法对指定的加密密码字符串进行解密，所以如果目前的密码字符串已经使用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的密码。 ● password 登录密码。 ● DISABLE 默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：IAM认证、kerberos或ldap认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。	密码规则： ● 密码默认不少于8个字符。 ● 不能与用户名及用户名倒序相同。 ● 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（~!@#$%^&*()-_=+|[{}];:,<.>/?）四类字符中的三类字符。使用范围外的字符会收到告警，但依然允许创建。
VALID BEGIN	设置角色生效的时间戳。如果省略了该子句，角色无有效开始时间限制。	-
VALID UNTIL	设置角色失效的时间戳。如果省略了该子句，角色无有效结束时间限制。	-
RESOURCE POOL	设置角色使用的resource pool名称，该名称属于系统表：pg_resource_pool	-
USER GROUP 'groupuser'	创建一个user的子用户。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PERM SPACE	设置用户永久表存储空间限额。 space_limit表示永久表存储空间上限。	space_limit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
TEMP SPACE	设置用户临时表存储空间限额。 tmpspacelimit表示临时表存储空间限额。	tmpspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
SPILL SPACE	设置用户算子落盘空间限额。 spillspacelimit：算子落盘空间限额。	spillspacelimit，字符串格式为正整数+单位，单位当前支持K/M/G/T/P。0表示不限制。
NODE GROUP	设置用户关联的逻辑集群名称。如果需要关联的逻辑集群名称包含大写字符或特殊字符，指定逻辑集群名称时需要加双引号。	-
ACCOUNT LOCK | ACCOUNT UNLOCK	• ACCOUNT LOCK：锁定账户，禁止登录数据库。 • ACCOUNT UNLOCK：解锁账户，允许登录数据库。	-
PGUSER	该属性用于兼容开源Postgres的连接通讯，开源的Postgres客户端接口（推荐使用Postgres 9.2.19版本的相关客户端接口）可以使用具有该属性的数据库用户连接数据库。 当前版本不允许修改角色的PGUSER属性。更多说明内容请参考**PGUSER参数建议**。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
AUTHINFO 'authinfo'	该属性用于指定角色认证类型，authinfo为类型说明字符串，大小写敏感。当前仅支持LDAP类型，对应的类型说明字符串为'ldap'。LDAP属于外部认证，故需要同时指定PASSWORD DISABLE。 ● authinfo可以加上LDAP认证的额外信息，比如LDAP认证中的fulluser，fulluser等同于ldapprefix+username+ldapsuffix。当authinfo为'ldap'，表示角色认证类型为LDAP，此时ldapprefix和ldapsuffix的信息由pg_hba.conf中匹配的记录提供。 ● ALTER ROLE时不允许用户切换认证类型，仅允许LDAP用户修改LDAP属性。	-
PASSWORD EXPIRATION period	声明该角色的登录密码过期天数，登录密码过期之前用户需要及时修改密码。登录密码过期后用户无法登录，需要请管理员设置新的登录密码后登录。	整数，-1~999。 ● -1，表示没有过期限制，缺省值。 ● 0，表示用户登录密码立即过期。
IN DATABASE database_name	表示修改角色在指定数据库上的参数。	-
SET configuration_parameter	设置角色的参数。ALTER ROLE中修改的会话参数只针对指定的角色，且在下一次该角色启动的会话中有效。 ● DEFAULT，表示清除configuration_parameter参数的值，configuration_parameter参数的值将继承本角色新产生的SESSION的默认值。 ● FROM CURRENT，表示取当前会话中的值设置为configuration_parameter参数的值。	configuration_parameter和value的取值请参见**SET**。
RESET configuration_parameter/ALL	清除configuration_parameter参数的值。与SET configuration_parameter TO DEFAULT的效果相同。 ALL表示清除所有参数的值。	-

版权所有 © 华为云计算技术有限公司

PGUSER 参数建议

PGUSER属性只用于兼容连接过程，而由于本产品与Postgres的内核差异导致的不兼容，不在此属性控制范围内。

由于具有PGUSER属性的用户的认证方式与其他用户不同，开源客户端的报错信息可能导致数据库用户PGUSER属性被枚举，建议使用本产品自有的客户端。例如：

```
#normaluser是不具有PGUSER属性的用户，psql是Postgres的客户端工具
pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U normaluser
psql: authentication method 10 not supported

#pguser用户是具有PGUSER属性的用户
pg@dws04:~> psql -d postgres -p 8000 -h 10.11.12.13 -U pguser
Password for user pguser:
```

示例

创建示例角色r1，r2和r3。

```
CREATE ROLE r1 IDENTIFIED BY '{Password}';
CREATE ROLE r2 WITH LOGIN AUTHINFO 'ldapcn=r2,cn=user,dc=lework,dc=com' PASSWORD DISABLE;
CREATE ROLE r3 WITH LOGIN PASSWORD '{Password}' PASSWORD EXPIRATION 30;
```

修改角色r1的登录权限：

```
ALTER ROLE r1 login;
```

修改角色r1的密码：

```
ALTER ROLE r1 IDENTIFIED BY '{new_Password}' REPLACE '{Password}';
```

修改角色manager为系统管理员：

```
ALTER ROLE r1 SYSADMIN;
```

修改LDAP认证角色的fulluser信息：

```
ALTER ROLE r2 WITH LOGIN AUTHINFO 'ldapcn=role2,cn=user2,dc=func,dc=com' PASSWORD DISABLE;
```

修改角色的登录密码有效期为90天：

```
ALTER ROLE r3 PASSWORD EXPIRATION 90;
```

相关链接

CREATE ROLE，**DROP ROLE**，**SET**

21.13.3.6 SET ROLE

功能描述

设置当前会话的当前用户标识符。

注意事项

- 当前会话的用户必须是指定的rolename角色的成员，但系统管理员可以选择任何角色。
- 使用这条命令，它可能会增加一个用户的权限，也可能会限制一个用户的权限。如果会话用户的角色有INHERITS属性，则它自动拥有它能SET ROLE变成的角色的所有权限；在这种情况下，SET ROLE实际上是删除了所有直接赋予会话用户的权

版权所有 © 华为云计算技术有限公司

限，以及它的所属角色的权限，只剩下指定角色的权限。另一方面，如果会话用户的角色有NOINHERITS属性，SET ROLE删除直接赋予会话用户的权限，而获取指定角色的权限。

语法格式

- 设置当前会话的当前用户标识符。
  ```
  SET [ SESSION | LOCAL ] ROLE role_name PASSWORD '{password}';
  ```
- 重置当前用户标识为当前会话用户标识符。
  ```
  RESET ROLE;
  ```

参数说明

表 21-135 SET ROLE 参数说明

参数	描述	取值范围
SESSION	声明这个命令只对当前会话起作用，此参数为缺省值。	字符串，需符合**标识符命名规范**。
LOCALE	声明该命令只在当前事务中有效。	-
role_name	角色名。	字符串，需符合**标识符命名规范**。
password	角色的密码。要求符合密码的命名规则。	-
RESET ROLE	用于重置当前用户标识。	-

示例

创建一个角色，名为manager。

```
CREATE ROLE paul IDENTIFIED BY '{Password}';
```

设置当前用户为paul。

```
SET ROLE paul PASSWORD '{password}';
```

查看当前会话用户，当前用户。

```
SELECT SESSION_USER, CURRENT_USER;
```

重置当前用户。

```
RESET role;
```

21.13.3.7 DROP ROLE

功能描述

删除指定的角色。

版权所有 © 华为云计算技术有限公司

注意事项

DROP ROLE若提示role is being used by other users错误，可能原因为CLEAN CONNECTION过程存在线程无法及时响应信号，出现连接清理不完全的情况，需要再次执行CLEAN CONNECTION。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
DROP ROLE [ IF EXISTS ] role_name [, ...];
```

参数说明

表 21-136 DROP ROLE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的角色不存在，则发出一个notice而不是抛出一个错误。	-
role_name	要删除的角色名称。	已存在的角色。

示例

删除角色manager。

```
DROP ROLE manager;
```

相关链接

CREATE ROLE，**ALTER ROLE**，**SET ROLE**

21.13.3.8 CREATE GROUP

功能描述

创建一个新用户组。

注意事项

CREATE GROUP是CREATE ROLE的别名，非SQL标准语法，不推荐使用，建议用户直接使用CREATE ROLE替代。

版权所有 © 华为云计算技术有限公司

语法格式

```
CREATE GROUP group_name [ [ WITH ] option [ ... ] ]
    [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' | DISABLE };
```

其中可选项action子句语法为：

```
where option can be:
{SYSADMIN | NOSYSADMIN}
    | {AUDITADMIN | NOAUDITADMIN}
    | {CREATEDB | NOCREATEDB}
    | {USEFT | NOUSEFT}
    | {CREATEROLE | NOCREATEROLE}
    | {INHERIT | NOINHERIT}
    | {LOGIN | NOLOGIN}
    | {REPLICATION | NOREPLICATION}
    | {INDEPENDENT | NOINDEPENDENT}
    | {VCADMIN | NOVCADMIN}
    | CONNECTION LIMIT connlimit
    | VALID BEGIN 'timestamp'
    | VALID UNTIL 'timestamp'
    | RESOURCE POOL 'respool'
    | USER GROUP 'groupuser'
    | PERM SPACE 'spacelimit'
    | NODE GROUP logic_group_name
    | IN ROLE role_name [, ...]
    | IN GROUP role_name [, ...]
    | ROLE role_name [, ...]
    | ADMIN role_name [, ...]
    | USER role_name [, ...]
    | SYSID uid
    | DEFAULT TABLESPACE tablespace_name
    | PROFILE DEFAULT
    | PROFILE profile_name
    | PGUSER
```

参数说明

请参考CREATE ROLE的**参数说明**。

相关链接

ALTER GROUP，**DROP GROUP**，**CREATE ROLE**

21.13.3.9 ALTER GROUP

功能描述

修改用户组的属性。

注意事项

ALTER GROUP是ALTER ROLE的别名，非SQL标准语法，不推荐使用，建议用户直接使用ALTER ROLE替代。

语法格式

- 向用户组中添加用户。

  ```
  ALTER GROUP group_name
      ADD USER user_name [, ... ];
  ```

- 从用户组中删除用户。

版权所有 © 华为云计算技术有限公司

```
ALTER GROUP group_name
    DROP USER user_name [, ... ];
```

- 修改用户组的名称。

```
ALTER GROUP group_name
    RENAME TO new_name;
```

参数说明

更多参数信息，请参考ALTER ROLE的**参数说明**。

表 21-137 ALTER GROUP 参数说明

参数	描述	取值范围
group_name	需要修改的用户组名称。	字符串，已存在的组名称。
user_name	要添加到组或从组中删除的用户。	已存在的用户名。
new_name	组的新名称。	字符串，需符合**标识符命名规范**。

相关链接

CREATE GROUP，**DROP GROUP**，**ALTER ROLE**

21.13.3.10 DROP GROUP

功能描述

删除用户组。

DROP GROUP是DROP ROLE的别名。

注意事项

DROP GROUP是集群管理工具封装的内部接口，用来实现集群管理。该接口不建议用户直接使用，以免对集群状态造成影响。

语法格式

```
DROP GROUP [ IF EXISTS ] group_name [, ...];
```

参数说明

请参见DROP ROLE的**参数说明**。

相关链接

CREATE GROUP，**ALTER GROUP**，**DROP ROLE**

版权所有 © 华为云计算技术有限公司

21.13.3.11 DROP OWNED

功能描述

删除一个数据库角色所拥有的数据库对象。

注意事项

所有该角色在当前数据库里和共享对象（数据库，表空间）上的所有对象上的权限都将被撤销。

语法格式

```
DROP OWNED BY name [, ...] [ CASCADE | RESTRICT ];
```

参数说明

表 21-138 DROP OWNED 参数说明

参数	描述	取值范围
name	将要删除所拥有对象并且撤销其权限的角色名。	已存在的角色名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除所有依赖于被删除对象的对象。 • RESTRICT：缺省值，拒绝删除那些有任何依赖对象存在的对象。

示例

删除名称为u1的角色拥有的所有数据库对象。

```
DROP OWNED BY u1;
```

21.13.4 表和外部数据源

21.13.4.1 CREATE TABLE

功能描述

在当前数据库中创建一个新的空白表。

该表由命令执行者所有，但系统管理员在普通用户同名schema下创建的表，表的所有者为schema的同名用户（非系统管理员）。

版权所有 © 华为云计算技术有限公司

注意事项

表 21-139 CREATE TABLE 注意事项

约束项	详细内容
创建列存和HDFS分区表的数量	建议不超过1000个。
列存表支持的数据类型	请参考**列存表支持的数据类型**。
主键与唯一约束要求	表中的主键约束和唯一约束必须包含分布列。
分布列修改限制	不支持修改已有表的分布列数据类型。
行存REPLICATION表限制	行存REPLICATION分布表不支持将系统列设置为主键。
列存表约束支持范围	PARTIAL CLUSTER KEY、主键和唯一表级约束，不支持外键表级约束。
列存表的字段约束	只支持NULL、NOT NULL和DEFAULT常量值。
冷热表	• 冷热表仅支持列存分区表，依赖于可用的OBS服务。 • 冷热表仅支持默认表空间为default_obs_tbs，如需新增obs表空间可联系技术支持。
列存表支持delta表	列存表支持delta表，由表级参数enable_delta控制是否开启，由参数deltarow_threshold控制进入delta表的阈值。不推荐使用列存带Delta表，否则会出现由于来不及merge而导致的磁盘膨胀以及性能劣化等问题。列存delta表由于上述问题在9.1.0.200后续版本不再支持新创建，统一使用HStore表替代，HStore相关资料参考《数据仓库服务开发指南》“实时数仓”章节。
HSTORE表	• 创建HSTORE表时，必须确保数据库GUC参数设置满足以下条件： – autovacuum设置为on。 – autovacuum_max_workers_hstore取值大于0。 – autovacuum_max_workers取值大于autovacuum_max_workers_hstore的取值。
V3表	• 创建表后，不支持通过ALTER TABLE语法将非V3表切换为V3表（即colversion为2.0不支持切为3.0）。 • V3表（即colversion=3.0，存算分离表，以下简称V3表）不支持设置delta表和列存二级分区。 • V3表不支持设置为Hstore表、冷热表、时序表。 • V3表不支持创建全局临时表和临时表，创建的临时表会自动转化为colversion=2.0的临时表。

版权所有 © 华为云计算技术有限公司

约束项	详细内容
其他	如果在建表过程中数据库系统发生故障，系统恢复后可能无法自动清除之前已创建的、大小为0的磁盘文件。此种情况出现概率小，不影响数据库系统的正常运行。

警告

- 不建议创建普通表时指定自定义TABLESPACE。
- 创建行存表时应避免指定COMPRESS压缩属性。
- 创建HASH分布的表对象时，要确保数据分布均匀（10G以上数据量的表，倾斜率控制在10%以内）。
- 创建REPLICATION分布的表对象，要确保表数据量控制在100万行以内。
- 针对存在时间字段的大表（数据量5000万行以上），必须设计成分区表，根据查询特征合理设计分区间隔。
- 针对有大批量数据增删改的表，索引个数建议控制在3个以内，最多不超过5个。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
CREATE [ [ GLOBAL | LOCAL | VOLATILE ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ]
table_name
    { ({ column_name data_type [ compress_mode ] [ COLLATE collation ] [ column_constraint [ ... ] ]
        | table_constraint
        | LIKE source_table [ like_option [...] ] }
        [, ... ])|
        LIKE source_table [ like_option [...] ] }
    [ WITH ( {storage_parameter = value} [, ... ] ) ]
    [ ON COMMIT { PRESERVE ROWS | DELETE ROWS } ]
    [ COMPRESS | NOCOMPRESS ]
    [ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
    [ TO { GROUP groupname | NODE ( nodename [, ... ] ) } ]
    [ COMMENT [=] 'text' ];
```

- 其中列约束column_constraint为：

  ```
  [ CONSTRAINT constraint_name ]
  { NOT NULL |
    NULL |
    CHECK ( expression ) |
    DEFAULT default_expr |
    ON UPDATE on_update_expr |
    COMMENT 'text' |
    UNIQUE [ NULLS [NOT] DISTINCT | NULLS IGNORE ] index_parameters |
    PRIMARY KEY index_parameters |
    REFERENCES reftable [ ( refcolumn ) ] }
  [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
  ```

- 其中列的压缩可选项compress_mode为：

  ```
  { DELTA | PREFIX | DICTIONARY | NUMSTR | NOCOMPRESS }
  ```

- 其中表约束table_constraint为：

  ```
  [ CONSTRAINT constraint_name ]
  { CHECK ( expression ) |
    UNIQUE [ NULLS [NOT] DISTINCT | NULLS IGNORE ] ( column_name [, ... ] ) index_parameters |
  ```


版权所有 © 华为云计算技术有限公司

```
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |
  PARTIAL CLUSTER KEY ( column_name [, ... ] ) }
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

- 其中like选项like_option为：

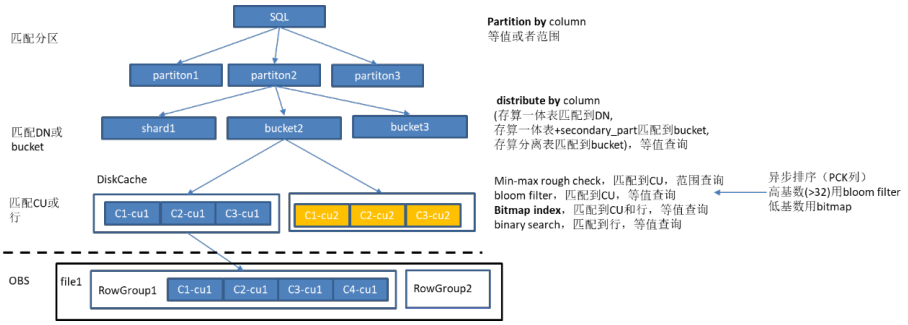
```
{ INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES | STORAGE | COMMENTS |
PARTITION | RELOPTIONS | DISTRIBUTION | DROPCOLUMNS | ALL }
```

- 其中索引参数index_parameters为：

```
[ WITH ( {storage_parameter = value} [, ... ] ) ]
```

表设计参考

DWS兼容PostgreSQL生态，行存及其B-Tree索引和PostgreSQL类似，列存及其索引为自研。建表时，选择合适的存储方式、分布列、分区键、索引，能够使SQL在执行时快速命中数据，减少IO消耗。下图是一个SQL从发起到获取数据的执行流程，通过下图，可以理解每个技术手段的作用，用于指导性能调优。



1. SQL执行时，分区表会通过Partition Column分区裁剪，定位到所在分区。
2. Hash分布表会通过Distribute Column快速定位到数据所在的数据分片（存算一体定位到DN，存算分离定位到bucket）。
3. 行存通过B-Tree快速定位到数据所在的Page；列存通过min-max索引快速定位到可能存在数据的CU数据块，PCK(Partial Cluster Key)列上的min-max索引过滤效果最好。
4. 系统自动为列存的所有列维护min-max索引，无需用户定义。min-max索引是粗过滤，满足min-max条件的CU数据块不一定真正存在满足filter条件的数据行，如果定义了bitmap column，可以通过bitmap index快速定位到CU内符合条件的数据所在的行号；对于有序的CU，也会通过二分查找快速定位数据的行号。
5. 列存也支持B-Tree和GIN索引，通过B-Tree/GIN索引也可以快速定位到满足条件的数据所在的CU及行号，但是，索引的维护代价比较大，除非对点查有极高的性能诉求，推荐使用bitmap index替代B-Tree/GIN。

DWS已有的优化手段可参考**表21-140**。

 版权所有 © 华为云计算技术有限公司

表 21-140 优化手段

编号	优化手段	使用建议	SQL示例	建表后是否可修改
1	字符串类型	1. 字符串类型会比定长类型慢，如果能使用定长类型时不建议使用字符串类型。 2. 尽量指定长度在16以内，性能会显著提高。	-	是，已有数据会重写。
2	Numeric类型	Numeric类型要求都指定精度，性能会显著提高，不建议使用无精度的numeric。	-	是，已有数据会重写。
3	Partition by Column	1. 需用户定义，适用分区表，通过分区键进行剪枝，支持partition wise join，适用等值和范围查询。 2. 分区数目不建议超过1000，分区列不建议超过两列。	SELECT * FROM t1 WHERE t1.c1='p1';	否，如需修改请重新建表。
4	secondary_part_column	1. 需用户定义，只适用于列存表，适用等值查询。 2. 可在最常用的一个等值过滤列上指定二级分区。	SELECT * FROM t1 WHERE t1.c1=’p1’;	否，如需修改请重新建表。
5	Distribute by Column	需用户定义，适用于频繁进行group by或者多表join的join字段，通过local join减少数据shuffle，适用等值查询。	SELECT * FROM t1 join t2 on t1.c3 = t2.c1;	否，如需修改请重新建表。
6	Bitmap column	需用户定义，根据CU内的重复值，自适应创建bitmap index（基数<=32），或者bloom filter（基数>32），适用varchar/text类型列的等值查询，对于where条件涉及的列，建议创建。	SELECT * FROM t1 WHERE t1.c4=’hello’;	可以修改，已有数据不重写，只影响新数据。
7	Min-max索引	1. min-max索引无需用户定义，适用等值和范围查询。 2. min-max的过滤效果取决于数据的有序性，指定PCK的列，过滤效果会更好。	SELECT * FROM t1 WHERE c3 > 100 and c3 < 200;	PCK可以修改，已有数据不重写，只影响新数据。

版权所有 © 华为云计算技术有限公司

编号	优化手段	使用建议	SQL示例	建表后是否可修改
8	主键（B-Tree索引）	1. UPSERT入库强依赖主键，需用户定义，适用等值和范围查询，不建议超过5列。 2. 满足业务前提下，能使用定长类型的列更好，定义时尽可能将distinct值多的列放前面。	SELECT * FROM t1 WHERE c3 > 100 and c3 < 200;	可以修改，重建索引。
9	GIN索引	1. 需用户定义，适用不固定的多条件等值查询，Distinct值超过100W的列不建议使用。 2. 过滤后数据量小于1000的适用，过滤后数据量依然较大的，不推荐使用。	SELECT * FROM t1 WHERE c1 = 100 and c3 = 200 and c2 = 105;	可以修改，重建索引。
10	Orientation=column/row	指定表为行存或者列存，行存不支持压缩，适合点查询和频繁更新场景；列存支持压缩，适合分析场景。	-	否，如需修改请重新建表。

版权所有 © 华为云计算技术有限公司

参数说明

表 21-141 CREATE TABLE 参数说明

参数	描述	取值范围或示例
UNLOGGED	如果指定UNLOGGED字段，则创建的表为非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动截断，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。 **须知** 1. UNLOGGED表无主备机制，在系统故障或异常断点等情况下，会有数据丢失风险，不可用来存储基础数据。 2. 9.1.0版本，UNLOGGED表默认存储在pg_unlogged表空间下，且不可迁移或指定到其他表空间。 3. 从低版本升级到9.1.0版本后，旧版本创建的UNLOGGED表还存储在原表空间下。 4. 由于实例异常重启时会对UNLOGGED表进行重置，所以可能影响实例RTO。9.1.0版本提供UNLOGGED表迁移脚本switch_unlogged_tablespace.py，配合GUC参数enable_unlogged_tablespace_compat，可以优化RTO。	使用场景：非日志表不能保证数据的安全性，用户应该在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。 故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。 示例： `CREATE UNLOGGED TABLE table_unlog ( col1 VARCHAR(50) , col2 TEXT );`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
GLOBAL | LOCAL | VOLATILE	创建临时表时在TEMP或TEMPORARY前指定GLOBAL、LOCAL、VOLATILE关键字，以创建不同属性的临时表。仅8.2.1.220及以上集群版本支持全局临时表。 • 指定关键字LOCAL，则创建LOCAL临时表（本地临时表）。 • 指定关键字GLOBAL，则临时表的属性取决于GUC参数**enable_global_temp_table**的设置，该参数默认值为ON。 当enable_global_temp_table=on时，表示创建GLOBAL临时表（全局临时表）； 当enable_global_temp_table=off时，表示创建LOCAL临时表（本地临时表），等同于指定关键字LOCAL。 • 指定关键字VOLATILE，则创建VOLATILE临时表。 当**default_temptable_type**=local时，不指定关键字创建的临时表为LOCAL类型临时表，当default_temptable_type=volatile时，不指定关键字创建的临时表为VOLATILE类型临时表。	示例： CREATE **LOCAL** TEMP TABLE temp_test (col1 INT, col2 DECIMAL(10,2));
TEMPORARY | TEMP	指定TEMP或TEMPORARY关键字，则创建的表为临时表。 临时表只在当前会话可见，本会话结束后会自动删除。因此，在除当前会话连接的CN以外的其他CN故障时，仍然可以在当前会话上创建和使用临时表。 由于临时表只在当前会话创建，对于涉及对临时表操作的DDL语句，会产生DDL失败的报错。因此，建议DDL语句中不要对临时表进行操作。TEMP和TEMPORARY等价。	该参数其他使用约束请参见**TEMPORARY | TEMP参数使用说明**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
IF NOT EXISTS	指定IF NOT EXISTS时，若不存在同名表，则可以成功创建表。若已存在同名表，创建时不会报错，仅会提示该表已存在并跳过创建。	示例： 使用IF NOT EXISTS避免表重复创建错误。 CREATE TABLE **IF NOT EXISTS** test_t1(col1 INT,col2 VARCHAR(30));
table_name	要创建的表名。	• 表名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。 • 支持中文表名，支持的中文字数与数据库字符集编码有关，请参见《数据仓库服务开发指南》中“DWS数据库对象命名规则”章节。 • 应避免使用**关键字**作为表名。 • 使用双引号括起来的表名可以包含空格和特殊字符，但不建议在表名中使用这些字符，因为这样可能会使表名难以引用和使用，而且不同的数据库兼容模式下可能对于这种表名的处理方式也有所不同。
column_name	新表中要创建的字段名。	字段名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。
data_type	字段的数据类型。	在兼容Teradata或MySQL语法的数据库中，字段数据类型指定为DATE时同样返回为DATE类型，否则返回TIMESTAMP类型。
compress_mode	表字段的压缩选项，当前仅对行存表有效。该选项指定表字段优先使用的压缩算法。	DELTA、PREFIX、DICTIONARY、NUMSTR或NOCOMPRESS。
COLLATE collation	COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。	如果没有指定，则使用默认的排序规则。
LIKE source_table [like_option ...]	LIKE子句声明一个表，新表自动从这个表中继承所有字段名及其数据类型和非空约束。	LIKE相关使用约束参见**LIKE参数使用说明**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
WITH	这个子句为表或索引指定一个可选的存储参数。 **说明** 使用任意精度类型Numeric定义列时，建议指定精度p以及刻度s。在不指定精度和刻度时，会按输入的显示出来。	WITH包含的子参数较多，按场景分类如下，开发者仅需关注特定场景下的参数即可： • 存储类相关参数：参见**表21-142**。 • 分区表相关参数：参见**表21-143**。 • delta表相关参数：参见**表21-144**。 • Hstore表相关参数：参见**表21-145** • 性能相关参数：参见**表21-146**。
ON COMMIT { PRESERVE ROWS | DELETE ROWS }	ON COMMIT选项决定在事务中执行创建临时表操作，当事务提交时，此临时表的后续操作。全局临时表只支持PRESERVE ROWS选项。	• PRESERVE ROWS（缺省值）：提交时不对临时表做任何操作，临时表及其表数据保持不变。 • DELETE ROWS：提交时删除临时表中数据。
COMPRESS | NOCOMPRESS	创建新表时，需要在CREATE TABLE语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。	缺省值：NOCOMPRESS，即不对元组数据进行压缩。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
DISTRIBUTE BY	指定表如何在节点之间分布或者复制。 • REPLICATION：表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。 • ROUNDROBIN：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。（ROUNDROBIN仅8.1.2及以上版本支持） • HASH（column_name）：对指定的列进行Hash，通过映射，把数据分布到指定DN。 其他详细使用约束参见**DISTRIBUTE BY参数使用说明**。	示例： 指定表的数据分布方式为HASH。 `CREATE TABLE test_t2(col1 INT,col2 VARCHAR(30))DISTRIBUTE BY HASH(col1);`
TO { GROUP groupname | NODE (nodename [, ...]) }	TO GROUP指定创建表所在的Node Group，目前不支持hdfs表使用。TO NODE主要供内部扩容工具使用，一般用户不应该使用。	在逻辑集群模式下，如果不指定TO GROUP，表默认会创建在逻辑集群用户关联的节点组中；如果用户没有管理逻辑集群（例如管理员用户或其他普通用户），表默认会创建在第一个逻辑集群中（pgxc_group中oid最小的逻辑集群是第一个逻辑集群）。 如果TO GROUP指定的节点组是复制表节点组，表将创建在所有CN和DN节点上，但复制表数据将只分布在复制表节点组包含的DN节点上。 存算分离3.0版本支持只读逻辑集群，如果未绑定到只读逻辑集群的用户建表语句中TO GROUP指定的是只读逻辑集群，则创建表会报错；但绑定到只读逻辑集群的用户创建表会遵循未绑定逻辑集群用户的建表规则，将表创建到由GUC参数default_storage_nodegroup指定的逻辑集群上。如果default_storage_nodegroup是installation, 表会创建在第一个逻辑集群中。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
COMMENT [=] 'text'	COMMENT子句可在创建表时指定表注释。	示例： 添加表注释。 CREATE TABLE table_user(user_id INT,username VARCHAR(50)) **COMMENT** '*user Information*';
COMMENT 'text'	COMMENT子句可以指定列的注释。	示例： 添加指定列的注释。 CREATE TABLE table_users(user_id INT **COMMENT** '*employee id*',username VARCHAR(50));
CONSTRAINT constraint_name	列约束或表约束的名字。可选的约束子句用于声明约束，新行或者更新的行必须满足这些约束才能成功插入或更新。 定义约束有两种方法： ● 列约束：作为一个列定义的一部分，仅影响该列。 ● 表约束：不和某个列绑在一起，可以作用于多个列。	示例： 创建表table_c_products，定义了主键约束：表的主键由C_CUSTKEY和C_NAME组成。 CREATE TABLE table_c_products (C_CUSTKEY BIGINT, C_NAME VARCHAR(25) , C_ADDRESS VARCHAR(40), C_NATIONKEY INT, C_PHONE CHAR(15) , C_ACCTBAL DECIMAL(15,2), **CONSTRAINT** C_CUSTKEY_KEY PRIMARY KEY(C_CUSTKEY,C_NAME)) DISTRIBUTE BY HASH(C_CUSTKEY,C_NAME);
NOT NULL	字段值不允许为NULL。	-
NULL	字段值允许为NULL ，这是缺省值。 这个子句只是为和非标准SQL数据库兼容。不建议使用。	-
CHECK (expression)	CHECK约束声明一个布尔表达式，每次要插入的新行或者要更新的行的新值必须使表达式结果为真或未知才能成功，否则会抛出一个异常并且不会修改数据库。 声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。	expression表达式中，如果存在“<>NULL”或“!=NULL”，这种写法是无效的，需要写成“is NOT NULL”。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
DEFAULT default_expr	DEFAULT子句给字段指定缺省值。	该数值可以是任何不含变量的表达式(不允许使用子查询和对本表中的其他字段的交叉引用)。缺省表达式的数据类型必须和字段类型匹配。 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。
ON UPDATE on_update_expr	ON UPDATE子句给字段指定时间戳函数。ON UPDATE子句支持指定的字段类型必须为timestamp类型或者timestamptz类型。 当执行含有update操作的SQL语句时，自动更新此列为时间戳函数所代表的时间。	on_update_expr时间戳函数仅支持CURRENT_TIMESTAMP, CURRENT_TIME, CURRENT_DATE, LOCALTIME, LOCALTIMESTAMP。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
UNIQUE [NULLS [NOT] DISTINCT | NULLS IGNORE] index_parameters UNIQUE [NULLS [NOT] DISTINCT | NULLS IGNORE] (column_name [, ...]) index_parameters	UNIQUE约束表示表里的一个字段或多个字段的组合必须在全表范围内唯一。 其中[NULLS [NOT] DISTINCT | NULLS IGNORE]字段用来指定Unique唯一索引中索引列NULL值的处理方式。	默认值：该参数默认缺省，即NULL值可重复插入。 在对插入的新数据和表中原始数据进行列的等值比较时，对于NULL值有以下三种处理方式： • NULLS DISTINCT：NULL值互不相等，即NULL值可重复插入。 • NULLS NOT DISTINCT：NULL值相等。若索引列全为NULL，则NULL值不可重复插入；部分索引列为NULL，只有非NULL值不相等，才可成功插入数据。 • NULLS IGNORE：在等值比较时跳过NULL值。若索引列全为NULL，则NULL值可重复插入；部分索引列为NULL，只有非NULL值不相等，才可成功插入数据。 三种处理方式具体的行为如表21-148所示。 **说明** 如果没有声明DISTRIBUTE BY REPLICATION，则唯一约束的列集合中必须包含分布列。
PRIMARY KEY index_parameters PRIMARY KEY (column_name [, ...]) index_parameters	主键约束声明表中的一个或者多个字段只能包含唯一的非NULL值。 一个表只能声明一个主键。 **注意** 如果没有声明DISTRIBUTE BY REPLICATION，则主键约束的列集合中必须包含分布列。	示例： 创建表products，定义product_no为整数类型，并将其设为主键。 `CREATE TABLE products (product_no integer PRIMARY KEY, name text, price numeric ) DISTRIBUTE BY HASH(product_no);`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
REFERENCES reftable [(refcolumn)]	外键约束要求新表中一列或多列构成的组应该只包含、匹配被参考表中被参考字段值。被参考列应该是被参考表中的唯一字段或主键。 由于DWS不会进行外键约束检查，使用外键约束时需通过函数。check_foreign_key_constraint来检测外键表中的数据是否满足外键约束。	-
DEFERRABLE | NOT DEFERRABLE	这两个关键字设置该约束是否可推迟。一个不可推迟的约束将在每条命令之后马上检查。可推迟约束可以推迟到事务结尾使用SET CONSTRAINTS命令检查。	缺省是NOT DEFERRABLE。目前，只有行存的UNIQUE约束和主键约束可以接受这个子句。所有其他约束类型都是不可推迟的。
PARTIAL CLUSTER KEY	局部聚簇存储，列存表导入数据时按照指定的列(单列或多列)，进行局部排序。	示例： 创建表products_new，并在product_no列上定义了局部聚簇。 `CREATE TABLE products_new (  product_no integer,  name text,  price numeric,  PARTIAL CLUSTER KEY(product_no) ) WITH (ORIENTATION = COLUMN);`
INITIALLY IMMEDIATE | INITIALLY DEFERRED	如果约束是可推迟的，则这个子句声明检查约束的缺省时间。	● 如果约束是INITIALLY IMMEDIATE（缺省），则在每条语句执行之后就立即检查它。 ● 如果约束是INITIALLY DEFERRED ，则只有在事务结尾才检查它。 ● 约束检查的时间可以用SET CONSTRAINTS命令修改。

版权所有 © 华为云计算技术有限公司

表 21-142 WITH 参数说明(存储相关)

参数	描述	取值范围
ORIENTATION	指定表数据的存储方式，即行存方式、列存方式，该参数设置成功后就不再支持修改。 • 行存储适合于OLTP业务，此类型的表上交互事务比较多，一次交互会涉及表中的多个列，用行存查询效率较高。 • 列存储适合于数据仓库业务，此类型的表上会做大量的汇聚计算，且涉及的列操作较少。	• ROW，表示表的数据将以行式存储。 • COLUMN，表示表的数据将以列式存储。 默认值：ROW，即行存方式。 **说明** 如需创建列存表，需显式设置orientation属性为column，存算分离版本如需创建本地表（数据全部存储在EVS盘），需显式指定colversion=2.0
COMPRESSION	指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比也越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。	该参数仅列存表支持，有效值为：LOW、MIDDLE或HIGH。默认值为LOW。 DWS内部提供的压缩算法参见**表21-147**。 **说明** • 暂不支持行存表压缩功能。 • ORC格式迁移到DWS列存表，如果使用low级别压缩，单副本大小大约是ORC的1.5~2倍；如果使用high级别压缩，单副本大小基本与ORC持平，在新建DWS集群规模时，需考虑该转换关系。 • 列存middle压缩固定使用字典压缩，对于数据特征不适合字典压缩的数据，使用middle压缩的结果可能比low压缩更大。
COMPRESSLEVEL	指定表数据同一压缩级别下的不同压缩水平，它决定了同一压缩级别下表数据的压缩比以及压缩时间。 对同一压缩级别进行了更加详细的划分，为用户选择压缩比和压缩时间提供了更多的空间。总体来讲，此值越大，表示同一压缩级别下压缩比越大，压缩时间越长；反之亦然。该参数只对列存表有效。	0~3 默认值：0
MAX_BATCHROW	指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。	10000~60000 默认值：60,000

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PARTIAL_CLUSTER_ROWS	指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。	600000~2147483647 默认值：4,200,000

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
COLVERSION	指定列存存储格式的版本，支持不同存储格式版本之间的切换。	• 1.0：列存表的每列以一个单独的文件进行存储，文件名以relfilenode.C1.0、relfilenode.C2.0、relfilenode.C3.0等命名。 • 2.0：列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名。 • 3.0：列存表的每列合并存储在一个文件中，文件存储在OBS文件系统上，文件名以C1_fileid.0命名。 默认值：存算一体版本默认值2.0，存算分离版本默认值3.0。 需注意，OBS冷热表仅支持COLVERSION为2.0格式。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围

		说明
		● 8.1.0集群版本该参数默认值为1.0，8.1.1及以上集群版本该参数默认值为2.0，若集群版本由8.1.0升级至8.1.1或以上版本，该参数默认值也会由1.0变为2.0。
		● 在建列存表时，选择COLVERSION=2.0，相比于1.0存储格式，在以下场景中性能有明显提升：
		  ● 创建列存宽表场景下，建表时间显著减少。
		  ● roach备份数据场景下，备份时间显著减少。
		  ● build、catch up耗时显著减少。
		● 存算分离版本兼容列存所有版本，建表时需显式指定colversion=1.0/2.0/3.0。当colversion=3.0时建立表为存算分离表，不显式指定colversion时默认创建3.0版本列存表。需注意，创建存算分离表时指定colversion为3.0的同时需要将orientation属性设置为column。
		● V3存算分离表不支持使用ALTER TABLE切换colversion，比如从2.0升级到3.0。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
cache_policy（该参数仅9.0.2及以上的存算分离版本支持）	控制了表或者分区表（磁盘）缓存的方式，如果在缓存策略中指定相应参数范围，则会进行热缓存，否则进行冷缓存。热缓存相比冷缓存占用的空间更大，技术上使用更加复杂的替换策略。	• ALL： 对整个表进行热缓存。 • NONE：对整个表进行冷缓存。 • HPN：N分区表中前N个分区会被热缓存，其余分区进行冷缓存。 • HPL：P1, P2, ... 分区表中在缓存策略中被指定名称的分区会被热缓存，其余分区进行冷缓存。 默认值：ALL **说明** • 对于外表和非分区内表只支持ALL和NONE两种缓存策略。 • 仅range和list分区内表支持HPN和HPL缓存策略。
enable_disaster_cstore	指定了列存表是否开启细粒度容灾功能。该参数仅适用于COLVERSION为2.0的普通列存表，不支持hstore表、行存表、临时表、unlog表、外表、时序表、冷热表，物化视图以及开启delta表功能，并且不能和enable_hstore、enable_delta同时打开。该参数仅8.2.0.100及以上集群版本支持。	-
fine_disaster_table_role	该参数8.2.1版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。 指定了细粒度容灾表的表角色为主表还是备表。使用该参数时，必须要保证已经开启了enable_disaster_cstore参数。	• primary，表示细粒度容灾主表。 • standby，表示细粒度容灾备表。
on_commit_preserve_rows	与 **ON COMMIT { PRESERVE ROWS | DELETE ROWS } 类似**。两者不能同时指定。该参数仅用于全局临时表。	默认值：true。

版权所有 © 华为云计算技术有限公司

表 21-143 WITH 参数说明(分区表相关)

参数	描述	取值范围
TTL	设置分区表定时删除分区task任务。默认不创建删除分区task任务。	1 hour ~ 100 years
PERIOD	设置分区表定时创建分区task任务。如果设置TTL，PERIOD不能大于TTL。	1 hour ~ 100 years 默认值：1 day
time_format	自增减分区支持INT4/INT8/VARCHAR/TEXT列，存储内容为时间时使用。time_format选项仅当分区键为INT4/INT8/VARCHAR/TEXT时，同时指定period才生效。该参数仅9.1.0.200及以上集群版本支持。	1. time_format写法遵从postgresql规范，例如：time_format = 'yyyymmdd'。 2. 分区键的类型**VARCHAR/TEXT**对time_format的规格有如下限制： ● 精度支持到秒级。 ● 输入内容不包括字母类型，例如month，am，pm等。 ● 时间必须从大到小排列，例如年月日小时分钟秒。 **INT4/INT8**对time_format的规格有如下限制： ● 精度支持到小时级。 ● 输入内容仅包括Y,M,D,HH24。 ● 时间必须从大到小排列，例如年月日小时。 3. ALTER相关限制 ● 不支持set。 ● 当period被reset时（表示已经关闭自动分区，会报出提示），此时可以reset此选项。

版权所有 © 华为云计算技术有限公司

表 21-144 WITH 参数说明(delta 表相关)

参数	描述	取值范围
enable_delta	指定了列存表是否开启delta表。该参数只对列存表有效。V3表暂不支持该参数，即COLVERSION为3.0时不支持设置enable_delta为on。 不推荐使用列存带Delta表，否则会出现由于来不及merge而导致的磁盘膨胀以及性能劣化等问题。	默认值：off
DELTAROW_THRESHOLD	指定列存表导入时小于多少行的数据进入delta表，只在表级参数enable_delta开启时生效。该参数只对列存表有效。	0～60000 默认值：6000

表 21-145 WITH 参数说明(Hstore 表相关)

参数	描述	取值范围
enable_hstore	指定了是否创建为HStore表（基于列存表实现）。该参数只对列存表有效。该参数仅8.2.0.100及以上集群版本支持。V3表暂不支持该参数，即COLVERSION为3.0时不支持设置enable_delta为on。	默认值：off **说明** 打开该参数时必须设置以下GUC参数用于保证HStore表的清理，推荐值如下： autovacuum=on，autovacuum_max_workers=6，autovacuum_max_workers_hstore=3。
enable_hstore_opt	enable_hstore_opt表级参数打开时会默认同时打开enable_hstore表级参数，该参数仅8.3.0及以上集群版本支持。	默认值：false

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
secondary_part_column	指定列存表二级分区列的列名，仅能指定一列作为二级分区列且只适用于HStore列存表。该参数仅8.3.0及以上集群版本支持。	• 被指定为二级分区的列不能被删除或者修改。 • 只能在建表时指定二级分区列，建表后不支持修改。 • 不建议指定分布列作为二级分区列。 • 指定二级分区列后，该表将会在DN上按照二级分区列进行逻辑上的HASH分区存储，从而有效提升该列等值查询的性能。
secondary_part_num	指定列存表二级分区的数量，仅适用于HStore列存表。该参数仅8.3.0及以上集群版本支持。	1~32 默认值：8 **须知** • 只有secondary_part_column被指定时，该参数才可以指定。 • 只能在建表时指定二级分区的数量，建表后不支持修改。 • 不建议用户随意设置该默认值，否则会影响导入和查询的性能。
bitmap_columns	bitmap index只适用于新版hstore（hstore_opt表），只有开启表级参数enable_hstore_opt且开启bitmap_columns="指定列"，才能生成bitmap index映射关系。该参数仅8.3.0及以上集群版本支持。	-

版权所有 © 华为云计算技术有限公司

表 21-146 WITH 参数说明(其他性能相关参数)

参数	描述	取值范围
FILLFACTOR	一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是合适的选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数对于列存表没有意义。	10~100
analyze_mode	控制表级自动analyze的方式。	● frozen：禁止所有形式的analyze（例外：无统计信息时，还能触发动态采样）。 ● backend：仅允许autovacuum轮询触发的analyze。 ● runtime：仅允许优化器触发的runtime analyze。 ● all：backend和runtime两种自动analyze都允许触发。 默认值：all
incremental_analyze	控制分区表是否启用增量analyze模式。该参数只对分区表有效，复制表不支持设置。该参数仅9.1.0.100及以上集群版本支持。	默认值：false

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
enable_turbo_store	用于控制是否创建为turbo表（基于列存表实现）。该参数只对列存表有效。该选项仅9.1.0.100及以上集群版本支持。 **说明** • 3.0普通列存表不支持设置turbo存储格式，仅3.0 hstore_opt表支持turbo存储格式。 • 2.0版本列存表无限制。 • turbo表当前不支持时序表、delta表以及轻量化update。	默认值：off **说明** turbo表对部分numeric和varchar数据类型做了整数格式存储优化，以提升对于numeric和varchar数据类型的处理速度。 • 为获取最佳性能，推荐定义numeric类型时指定precision和scale(p,s)。具体来说，当p小于等于9时，使用4字节整数存储；p小于等于18时，使用8字节整数存储；p小于等于37时，使用16字节整数存储；p大于37或未指定时，则依然使用变长格式存储，其占用较多存储空间，且性能较差。 • 如果定义varchar类型时指定其最大字节长度n小于等于2字节，使用2字节整数存储；n小于等于4字节时，使用4字节整数存储；依次类推，最大当n小于等于16字节时，使用16字节整数存储。因此对于varchar类型的GROUP BY列或者HashJoin关联列，如果可使用整数存储字符串，建议定义时指定其最大字节长度n小于等于16，可以成倍提升计算性能。
SKIP_FPI_HINT	顺序扫描过程中，若需要写FPW(full page writes)日志时，该参数控制是否跳过设置HintBits操作。	默认值：false **说明** 设置SKIP_FPI_HINT=true时，在对某表执行checkpoint操作后，若对该表进行顺序扫描，将不再产生Xlog。适用于查询次数较少的中间表，有效减少Xlog的大小，提升查询性能。
enable_column_autovacuum_garbage	用于控制是否开启列存表AUTOVACUUM重写CU的逻辑。该参数仅8.2.1.100及以上集群版本支持。 当前列存轻量化UPDATE与后台列存AUTOVACUUM并发会小概率报错， 可以通过设置该表级参数为off来避免。	默认值：true

版权所有 © 华为云计算技术有限公司

表 21-147 列存压缩算法

COMPRESSION	NUMERIC	STRING	INT
LOW	delta压缩+RLE压缩	lz4压缩	delta压缩（RLE可选）
MIDDLE	delta压缩+RLE压缩+lz4压缩	dict压缩或lz4压缩	delta压缩或lz4压缩（RLE可选）
HIGH	delta压缩+RLE压缩+zlib压缩	dict压缩或zlib压缩	delta压缩或zlib压缩（RLE可选）

表 21-148 唯一索引中索引列 NULL 值的处理方式

字段控制	索引列全为NULL	部分索引列为NULL
NULLS DISTINCT	可重复插入	可重复插入
NULLS NOT DISTINCT	不可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功
NULLS IGNORE	可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功

TEMPORARY | TEMP 参数使用说明

- LOCAL/VOLATILE临时表通过每个会话独立的以pg_temp开头的schema来保证只对当前会话可见，因此，不建议用户在日常操作中手动删除以pg_temp，pg_toast_temp开头的schema。
- 如果建表时不指定TEMPORARY/TEMP关键字，而指定表的schema为当前会话的pg_temp_开头的schema，则此表会被创建为临时表。
- LOCAL临时表的所有相关元数据同普通表类似，都存储在系统表内，而VOLATILE临时表会将除schema外的相关表结构元数据直接存储在内存中。所以相对本地临时表而言，VOLATILE临时表有更多约束：
  - 当前CN或DN重启之后，对应实例上的内存数据丢失，相关volatile临时表会失效。
  - VOLATILE临时表当前不支持ALTER/GRANT等修改表结构相关操作。
  - VOLATILE临时表和LOCAL临时表共用一种临时schema，所以在同一session中，VOLATILE临时表和LOCAL临时表不能存在同名表。
  - VOLATILE临时表信息不存储在系统表内，所以无法通过对系统表执行DML语句查询到VOLATILE相关元数据。
  - VOLATILE临时表仅支持普通的行存、列存表，不支持delta表、时序表、冷热表。
  - 不支持基于VOLATILE临时表创建视图。
  - 不支持创建临时表时指定tablespace(VOLATILE临时表默认tablespace均为pg_volatile)。

版权所有 © 华为云计算技术有限公司

  - 创建VOLATILE临时表时不支持指定约束：CHECK约束、UNIQUE约束、主键约束、触发器约束、EXCLUDE约束、PARTIAL CLUSTER约束。
- GLOBAL临时表的所有相关元数据同普通表类似，都存储在系统表内。
  - GLOBAL临时表与LOCAL临时表不同的是，会话退出时元数据不会删除，但会话的数据会删除。不同会话的数据独立，但共享同一份GLOBAL临时表的元数据。
  - 全局临时表的schema与普通表类似，不是以pg_temp开头的schema，但与LOCAL/VOLATILE临时表不同，所以可以与LOCAL/VOLATILE临时表同名。
  - 全局临时表仅支持普通的行存、列存表，不支持delta表、时序表、冷热表。
  - 不支持操作其他逻辑集群的全局临时表。

LIKE 参数使用说明

新表与源表之间在创建动作完毕之后是完全无关的。在源表做的任何修改都不会传播到新表中，并且也不可能在扫描源表的时候包含新表的数据。

被复制的列和约束并不使用相同的名字进行融合。如果明确的指定了相同的名字或者在另外一个LIKE子句中，将会报错。

- 源表上的字段缺省表达式或者ON UPDATE表达式只有在指定INCLUDING DEFAULTS时，才会复制到新表中。缺省是不包含缺省表达式的，即新表中的所有字段的缺省值都是NULL。
- 源表上的CHECK约束仅在指定INCLUDING CONSTRAINTS时，会复制到新表中，而其他类型的约束永远不会复制到新表中。非空约束总是复制到新表中。此规则同时适用于表约束和列约束。
- 如果指定了INCLUDING INDEXES，则源表上的索引也将在新表上创建，默认不建立索引。
- 如果指定了INCLUDING STORAGE，则复制列的STORAGE设置会复制到新表中，默认情况下不包含STORAGE设置。
- 如果指定了INCLUDING COMMENTS，则源表列、约束和索引的注释会复制到新表中。默认情况下，不复制源表的注释。
- 如果指定了INCLUDING PARTITION，则源表的分区定义会复制到新表中，同时新表将不能再使用PARTITION BY子句。默认情况下，不拷贝源表的分区定义。
- 如果指定了INCLUDING RELOPTIONS，则源表的存储参数（即源表的WITH子句）会复制到新表中。默认情况下，不复制源表的存储参数。

  说明

  WITH中的'PERIOD'，'TTL'为partition相关参数，LIKE INCLUDING RELOPTIONS不会复制到新表中，若要复制需INCLUDING PARTITION。
- 如果指定了INCLUDING DISTRIBUTION，则源表的分布信息会复制到新表中，包括分布类型和分布列，同时新表将不能再使用DISTRIBUTE BY子句。默认情况下，不拷贝源表的分布信息。
- 如果指定了INCLUDING DROPCOLUMNS，则源表被删除的列信息会被复制到新表中。默认情况下，不复制源表的删除列信息。
- INCLUDING ALL包含了INCLUDING DEFAULTS、INCLUDING CONSTRAINTS、INCLUDING INDEXES、INCLUDING STORAGE、INCLUDING COMMENTS、INCLUDING PARTITION、INCLUDING RELOPTIONS、INCLUDING DISTRIBUTION和INCLUDING DROPCOLUMNS的内容。

 版权所有 © 华为云计算技术有限公司

- 如果指定了EXCLUDING，则表示不包括指定的参数。
- 如果是OBS冷热表，INCLUDING PARTITION后新表所有分区均为本地热分区。
- 对源表同时执行ALTER TABLE...ADD PARTITION与CREATE TABLE语句，偶发可能会出现不同节点的分区数量不一致的情况，从而导致CREATE TABLE语句报错。出现此情况时，建议重新执行CREATE TABLE语句，即可解决。

须知

- 如果源表包含serial、bigserial、smallserial类型，或者源表字段的默认值是sequence，且sequence属于源表（通过CREATE SEQUENCE ... OWNED BY创建），这些Sequence不会关联到新表中，新表中会重新创建属于自己的sequence。这和之前版本的处理逻辑不同。如果用户希望源表和新表共享Sequence，需要首先创建一个共享的Sequence（避免使用OWNED BY），并配置为源表字段默认值，这样创建的新表会和源表共享该Sequence。
- 不建议将其他表私有的Sequence配置为源表字段的默认值，尤其是其他表只分布在特定的NodeGroup上，这可能导致CREATE TABLE ... LIKE执行失败。另外，如果源表配置其他表私有的Sequence，当该表删除时Sequence也会连带删除，这样源表的Sequence将不可用。如果用户希望多个表共享Sequence，建议创建共享的Sequence。

DISTRIBUTE BY 参数使用说明

指定表如何在节点之间分布或者复制。

取值范围：

- REPLICATION：表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。
- ROUNDROBIN：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。（ROUNDROBIN仅8.1.2及以上版本支持）
- HASH（column_name）：对指定的列进行Hash，通过映射，把数据分布到指定DN。

  说明

  - 当指定DISTRIBUTE BY HASH (column_name)参数时，创建主键和唯一索引必须包含“ column_name”列。
  - 当被参照表指定DISTRIBUTE BY HASH (column_name)参数时，参照表的外键必须包含“ column_name”列。
  - 如果TO GROUP指定为复制表节点组（8.1.2及以上版本支持），DISTRIBUTE BY必须指定为REPLICATION。如果没有指定DISTRIBUTE BY，创建的表会自动设置为复制表。

默认值：由GUC参数default_distribution_mode控制。

- 当default_distribution_mode=roundrobin时，DISTRIBUTE BY的默认值按如下规则选取：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，则选取ROUNDROBIN分布。
- 当default_distribution_mode=hash时，DISTRIBUTE BY的默认值按如下规则选取：

 版权所有 © 华为云计算技术有限公司

a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
b. 若建表时不包含主键/唯一约束，但存在数据类型支持作分布列的列，则选取HASH分布，分布列为第一个数据类型支持作分布列的列。
c. 若建表时不包含主键/唯一约束，也不存在数据类型支持作分布列的列，选取ROUNDROBIN分布。

以下数据类型支持作为分布列：

- INTEGER TYPES：TINYINT，SMALLINT，INT，BIGINT，NUMERIC/DECIMAL
- CHARACTER TYPES：CHAR，BPCHAR，VARCHAR，VARCHAR2，NVARCHAR2，TEXT
- DATE/TIME TYPES：DATE，TIME，TIMETZ，TIMESTAMP，TIMESTAMPTZ，INTERVAL，SMALLDATETIME

说明

在建表时，选择分布列和分区键可对SQL查询性能产生重大影响。因此，需要根据一定策略选择合适的分布列和分区键。

- 选择合适的分布列

  对于采用散列（Hash）方式的数据分布表，一个合适的分布列应将一个表内的数据，均匀分散存储在多个DN内，避免出现数据倾斜现象（即多个DN内数据分布不均）。请按照如下原则判定合适的分布列：

  1. 判断是否已发生数据倾斜现象。

     连接数据库，执行如下语句，查看各DN内元组数目。命令中的斜体部分tablename，请填入待分析的表名。

     SELECT a.count,b.node_name FROM (SELECT count(*) AS count,xc_node_id FROM *tablename* GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count DESC;

     如果各DN内元组数目相差较大（如相差数倍、数十倍），则表明已发生数据倾斜现象，请按照下面原则调整分布列。

  2. 重新选择分布列，可通过ALTER TABLE语句调整分布列，选择原则如下：

     分布列的列值应比较离散，以便数据能够均匀分布到各个DN。例如，考虑选择表的主键为分布列，如在人员信息表中选择身份证号码为分布列。

     在满足上面原则的情况下，考虑选择查询中的连接条件为分布列，以便Join任务能够下推到DN中执行，且减少DN之间的通信数据量。

  3. 如果找不到一个合适的分布列，使数据能够均匀分布到各个DN，那么可以考虑使用REPLICATION或ROUNDROBIN的数据分布方式。由于REPLICATION的数据分布方式会在每个DN中存放完整的数据，因此在表较大且找不到合适的分布列时，推荐使用ROUNDROBIN的数据分布方式。（ROUNDROBIN分布方式8.1.2及以上版本支持）

- 选择合适的分区键

  数据分区功能，可根据表的一列或者多列，将要插入表的记录分为若干个范围（这些范围在不同的分区里没有重叠）。然后为每个范围创建一个分区，用来存储相应的数据。

  调整分区键，使每次查询结果尽可能存储在相同或者最少的分区内（称为“分区剪枝”），通过获取连续I/O大幅度提升查询性能。

  实际业务中，经常将时间作为查询对象的过滤条件，因此，可考虑选择时间列为分区键，键值范围可根据总数据量、一次查询数据量调整。

示例：创建存算一体表

创建表test_t4。

```
CREATE TABLE IF NOT EXISTS test_t4 (  ---当数据库中没有同名表（test_t4）时创建表
   col_id INT PRIMARY KEY, ---定义一个整数类型的列，并将其设为主键
```

 版权所有 © 华为云计算技术有限公司

```
    col_number INT NOT NULL, ---定义一个整数类型的列，并设置为非空
    col_date DATE NOT NULL，---定义一个日期类型的列，并设置为非空
    col_price NUMERIC(10,2)，--- 定义一个数值类型的列，允许的最大整数位数为10，小数位数为2
    col_status TEXT    ---定义一个文本类型的列
)
WITH (ORIENTATION = COLUMN,COLVERSION=2.0,COMPRESSION=middle) ---表的存储方式为列存，表的压缩级别为MIDDLE
DISTRIBUTE BY HASH (col_id)                    ---使用哈希分布
COMMENT ' order information for test_table4';   ---表注释
```

示例：创建存算分离 V3 表

创建名为test_t5的存算分离V3表（仅存算分离3.0版本支持）。

```
CREATE TABLE  test_t5
(
id integer not null,
data integer,
age integer
)
WITH (ORIENTATION =COLUMN, COLVERSION =3.0)
DISTRIBUTE BY ROUNDROBIN;
```

创建表的时候指定缓存策略（仅存算分离3.0版本集群支持）。

```
CREATE TABLE sports
(
    N_NATIONKEY  INT NOT NULL
  , N_NAME       CHAR(25) NOT NULL
  , N_REGIONKEY  INT NOT NULL
  , N_COMMENT    VARCHAR(152)
) WITH (orientation = column, colversion = 3.0, cache_policy = 'HPL: Balls, Basketball')
tablespace cu_obs_tbs
DISTRIBUTE BY ROUNDROBIN
partition by list(N_NAME)
(
  partition Balls values ('Basketball', 'football', 'badminton'),
  partition Athletics values ('High jump', 'long jump', 'javelin'),
  partition Water_Sports values ('Surfing', 'diving', 'swimming'),
  partition Shooting values ('air guns', 'Rifles', 'archery'),
  partition rest values (DEFAULT)
);
```

示例：创建表时定义约束

为表定义主键表约束，可以在表的一列或多列上定义主键表约束。

```
CREATE TABLE CUSTOMER
(
    C_CUSTKEY     BIGINT       ,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT          ,
    C_PHONE       CHAR(15)     ,
    C_ACCTBAL     DECIMAL(15,2)   ,
    CONSTRAINT C_CUSTKEY_KEY PRIMARY KEY(C_CUSTKEY,C_NAME)
)
DISTRIBUTE BY HASH(C_CUSTKEY,C_NAME);
```

为表定义唯一列约束。

```
CREATE TABLE CUSTOMER
(
    C_CUSTKEY     BIGINT NOT NULL CONSTRAINT C_CUSTKEY_pk PRIMARY KEY,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT          ,
    C_PHONE       CHAR(15)     ,
```

 版权所有 © 华为云计算技术有限公司

```
    C_ACCTBAL     DECIMAL(15,2)
)
DISTRIBUTE BY HASH(C_CUSTKEY);
```

定义CHECK表约束。

```
CREATE TABLE CUSTOMER
(
    C_CUSTKEY     BIGINT NOT NULL CONSTRAINT C_CUSTKEY_pk PRIMARY KEY,
    C_NAME        VARCHAR(25)      ,
    C_ADDRESS     VARCHAR(40)      ,
    C_NATIONKEY   INT              ,
    CONSTRAINT C_CUSTKEY_KEY2 CHECK(C_CUSTKEY > 0 AND C_NAME <> '')
)
DISTRIBUTE BY HASH(C_CUSTKEY);
```

定义CHECK列约束。

```
CREATE TABLE CUSTOMER
(
    C_CUSTKEY     BIGINT NOT NULL CONSTRAINT C_CUSTKEY_pk PRIMARY KEY  ,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT NOT NULL  CHECK (C_NATIONKEY > 0)
)
DISTRIBUTE BY HASH(C_CUSTKEY);
```

使用DEFAULT为列col_STATE声明默认值为“GA”。

```
CREATE TABLE test_t5
(
    col_SK INTEGER NOT NULL,
    col_ID CHAR(16) NOT NULL,
    colE_NAME VARCHAR(20) UNIQUE DEFERRABLE,
    col_WAREHOUSE INTEGER,
    col_COUNTY VARCHAR(30),
    col_STATE CHAR(2) DEFAULT 'GA',
    col_ZIP CHAR(10)
);
```

示例：创建列存表并指定压缩方式

创建列存表customer_address并指定该表的数据压缩级别为高压缩。

```
CREATE TABLE customer_address
(
    ca_address_sk       INTEGER                 NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)              ,
    ca_street_type      CHARACTER(15)                      ,
    ca_suite_number     CHARACTER(10)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=HIGH,COLVERSION=2.0)
DISTRIBUTE BY HASH (ca_address_sk);
```

示例：使用 CREATE TABLE LIKE 语句创建表

使用CREATE TABLE LIKE方式快速复制现有表test_t6结构创建表test_t7。

```
CREATE TABLE IF NOT EXISTS test_t6
(
    col_id INT PRIMARY KEY,
    col_number INT NOT NULL,
    col_date DATE NOT NULL,
    col_price NUMERIC(10,2),
    col_status TEXT
)
```

 版权所有 © 华为云计算技术有限公司

```
WITH (ORIENTATION = COLUMN, COMPRESSION=middle)
DISTRIBUTE BY HASH (col_id);

CREATE TABLE test_t7 (LIKE test_t6 INCLUDING ALL);
```

相关链接

ALTER TABLE，**12.101-RENAME TABLE**，**DROP TABLE**

21.13.4.2 CREATE TABLE PARTITION

功能描述

创建分区表。逻辑上的一张表根据某种方案分成几张物理块进行存储，这张逻辑上的表称之为分区表，物理块称之为分区。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。

常见的分区策略都是按照某一列或者某几列定义一些数据分布范围，然后每个分区承载一个范围的数据，这些列称之为**分区键**。

目前DWS行存表、列存表支持的分区类型包括**范围分区**和**列表分区**。

- **范围分区（Range Partitioning）**

  根据表的一列或者多列，将要插入表的记录分为若干个范围，这些范围在不同的分区里没有重叠。为每个范围创建一个分区，用来存储相应的数据。

  - **核心特征**：数据按连续区间分布，常用于有序且按范围检索的场景。
  - **适用场景**：分区键具有线性增长或连续区间特征，数据查询常围绕 “某一范围” 展开，且新数据会自然落入新的区间。例如按时间范围进行分区的订单数据，按数值范围分区的用户等级数据等。
  - **范围分区策略**是指记录插入分区的方式，根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息，这是最常用的分区策略。目前范围分区仅支持范围分区策略。

- **列表分区（List Partitioning）**

  根据表的一列，将要插入表的记录通过每一个分区中出现的键值划分到对应的分区中，这些键值在不同的分区里没有重叠。为每组键值创建一个分区，用来存储相应的数据。**列表分区仅8.1.3及以上集群版本支持**。

  - **核心特征**：数据按离散的枚举值划分，常用于固定分类或高频过滤的维度。
  - **适用场景**：分区键值是固定且离散的类别，数据查询常围绕 “某一类别” 展开，键值不会无规律新增。例如按省份分区的用户信息表，每个省份对应一个分区。
  - **列表分区策略**是根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息。

版权所有 © 华为云计算技术有限公司

范围分区和列表分区的核心差异

表 21-149 范围分区和列表分区

维度	范围分区（Range）	列表分区（List）
数据分布	连续区间（时间、数值）	离散枚举（分类、状态）
查询优化	范围过滤（BETWEEN, >, <）	精确匹配（=, IN）
管理场景	定期增删区间（如按月份滚动）	固定分类维护（如地区、类型）
典型用例	日志、时间序列、数值归档	地域划分、状态隔离、多租户数据

分区的优势

- 某些类型的查询性能可以得到极大提升，特别是表中访问率较高的行位于一个单独分区或少数几个分区上的情况下。分区可以减少数据的搜索空间，提高数据访问效率。
- 当查询或更新一个分区的大部分记录时，连续扫描对应分区而不是访问整个表可以获得巨大的性能提升。
- 如果需要大量加载或者删除的记录位于单独的分区上，则可以通过直接读取或删除那个分区以获得巨大的性能提升，同时还可以避免由于大量DELETE导致的VACUUM超载（仅范围分区）。
- 通常情况下，单个DN数据量**超过1000W行**的场景，建议将表规划为分区表，建立完分区后，保持单DN上，单个分区的数据量控制在100W~1000W之间。

注意事项

- 单表的分区数最大为32767个，所有分区的边界值个数最大为32767个。建议不超过1000个。
- 有限地支持唯一约束和主键约束，即唯一约束和主键约束的约束键必须包含所有分区键。

语法格式

```
CREATE TABLE [ IF NOT EXISTS ] partition_table_name
( [
   { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
   | table_constraint
   | LIKE source_table [ like_option [...] ] }[, ... ]
] )
   [ WITH ( {storage_parameter = value} [, ... ] ) ]
   [ COMPRESS | NOCOMPRESS ]
   [ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { [ HASH ] ( column_name ) } } ]
   [ TO { GROUP groupname | NODE ( nodename [, ... ] ) } ]
   PARTITION BY {
      {VALUES (partition_key)} |
      {RANGE (partition_key) ( partition_less_than_item [, ... ] )} |
      {RANGE (partition_key) ( partition_start_end_item [, ... ] )} |
      {LIST (partition_key) (list_partition_item [, ...])}
   } [ { ENABLE | DISABLE } ROW MOVEMENT ];
```

 版权所有 © 华为云计算技术有限公司

CREATE TABLE PARTITION 主句参数说明

表 21-150 CREATE TABLE PARTITION 主句参数说明

参数	描述	取值范围或示例
IF NOT EXISTS	如果已经存在相同名称的表，不会抛出一个错误，而会发出一个通知，告知表关系已存在。	-
partition_table_name	分区表的名称。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
column_name	新表中要创建的字段名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
data_type	字段的数据类型。	-

COLLATE collation	COLLATE子句指定列的排序规则，仅支持char、varchar、text、nchar、nvarchar这些数据类型的排序规则，排序规则主要影响查询结果的排列顺序，例如SELECT ... ORDER BY后展示的优先级。如果没有指定，则使用默认的排序规则C（即严格区分大小写）。
	常见的排序规则有：
	• C：基于二进制编码的字节顺序进行排序，完全区分大小写、重音和特殊字符（严格按ASCII 码值比较）。
	• case_insensitive：不区分大小写，8.1.3.x及以上版本支持。
	• en_US.utf8：美式英语UTF8编码，不区分大小写。
	DWS支持的排序规则可通过系统表 PG_COLLATION查询：
	`SELECT * FROM pg_collation;`
	查看当前数据库默认的排序规则可使用如下语句：
	`SELECT pg_database.datcollate AS current_collation FROM pg_catalog.pg_database WHERE pg_database.datname = pg_catalog.current_database();`
		例如，对某列的取值采用不区分大小写排序规则。
		`CREATE TABLE t1 (a text collate case_insensitive);`
		`INSERT INTO t1 values('a'),('A'),('b'),('B');`
		`SELECT a, a='a' FROM t1;`
		查询时结果如下，表示A和a等价。
		`a | ?column?`
		`---+----------`
		`b | f`
		`A | t`
		`a | t`
		`B | f`
		`(4 rows)`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
CONSTRAINT constraint_name	列约束或表约束的名字。可选的约束子句用于声明约束，新行或者更新的行必须满足这些约束才能成功插入或更新。 定义约束有两种方法： • 列约束：作为一个列定义的一部分，仅影响该列。 • 表约束：不和某个列绑在一起，可以作用于多个列。	支持的约束参见**表21-151**。
LIKE source_table [like_option ...]	LIKE子句声明一个表，新表自动从声明的表中继承所有字段名及其数据类型和非空约束。	使用约束参见**LIKE参数使用说明**。
WITH	这个子句为表或索引指定一个可选的存储参数。	支持的WITH参数参见**表21-152**。
COMPRESS / NOCOMPRESS	创建一个新表时，需要在创建表语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。	缺省值为NOCOMPRESS，即不对元组数据进行压缩。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
DISTRIBUTE BY	指定表如何在节点之间分布或者复制。	• REPLICATION：表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。 • ROUNDROBIN：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。（ROUNDROBIN仅8.1.2及以上版本支持） • HASH (column_name)：对指定的列进行Hash，通过映射，把数据分布到指定DN。 其他详细使用约束参见**DISTRIBUTE BY参数使用说明**。 **注意** • 当指定DISTRIBUTE BY HASH (column_name)参数时，创建主键和唯一索引必须包含“column_name”列。 • 当被参照表指定DISTRIBUTE BY HASH (column_name)参数时，参照表的外键必须包含“column_name”列。
TO { GROUP groupname | NODE (nodename [, ...]) }	TO GROUP指定创建表所在的Node Group，目前不支持hdfs表使用。TO NODE主要供内部扩容工具使用，一般用户不建议使用。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
{ ENABLE | DISABLE } ROW MOVEMENT	行迁移开关，当更新表中某行分区键值时，如果新值属于其他分区，则会根据开关给出报错信息或者将该行数据迁移到对应的分区。 • ENABLE：打开行迁移功能，适用于频繁更新分区键的场景。 • DISABLE：关闭行迁移功能，适用于严格禁止数据跨分区更新的场景，更新分区键会报错。 默认为DISABLE，关闭。 **使用建议**： • **启用条件**：当业务需要通过更新操作改变分区键值，且性能影响可接受时使用。 • **性能考量**：ENABLE ROW MOVEMENT会增加更新操作的开销，特别是在高并发场景下需谨慎使用。 • **替代方案**：若需频繁调整分区归属，可考虑先删除原行再插入新行，或使用分区交换（**Exchange Partition**）操作。 **注意** 开启ROW MOVEMENT则允许跨分区更新，但此时如果有SELECT FOR UPDATE查询该分区表并发执行，存在查询结果瞬时不一致的可能性，需要谨慎使用。	举例，某分区表的日期列（分区键），按季度进行分区，分为分区p_2023q1和p_2023q2，某一行的“2023-02-15”原属于第一季度的分区p_2023q1，当更新该值为“2023-05-15”后，对应的这行数据，会根据ROW MOVEMENT开关是否打开而决定是否行数据迁移： • ENABLE：打开行迁移功能，则该数据迁移到新分区p_2023q2。 • DISABLE：关闭行迁移功能，则系统会报错。 详细示例参见**示例：开启和关闭行迁移功能ROW MOVEMENT**

列约束 CONSTRAINT constraint_name 参数说明

```
{ NOT NULL |
  NULL |
  CHECK ( expression ) |
  DEFAULT default_expr |
  UNIQUE [ NULLS [NOT] DISTINCT | NULLS IGNORE ] index_parameters |
  PRIMARY KEY index_parameters }
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

表 21-151 CONSTRAINT constraint_name 参数说明

参数	描述	取值范围或示例
NOT NULL	字段值不允许为NULL。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
NULL	字段值允许NULL ，这是默认值，该子句只是为和非标准SQL数据库兼容。不建议使用。	-

CHECK (condition) [NO INHERIT]	CHECK约束声明一个布尔表达式，每次要插入的新行或者要更新的行的新值必须使表达式结果为真或未知才能成功，否则会报错并且执行失败。
	声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。
	用NO INHERIT标记的约束将不会传递到子表中去。
		单个字段约束:
		价格必须为正数：
		`CREATE TABLE products ( product_id INT, price DECIMAL(10,2) CHECK (price > 0), discount DECIMAL(3,2) );`
		插入一条价格为负数的数据：
		`INSERT INTO products (product_id, price, discount) VALUES (2, -10.00, 1.1);`
		违反约束，则报错，插入不成功。
		`ERROR: dn_6003_6004: new row for relation "products" violates check constraint "products_price_check"`
		表约束（约束多个字段）：
		创建一个库存表，分两个字段，可用量和预定量，且可用量必须大于等于预定量：
		`CREATE TABLE inventory ( item_id INT PRIMARY KEY, quantity_on_hand INT, quantity_reserved INT, CHECK (quantity_on_hand >= quantity_reserved));`

DEFAULT default_expr	DEFAULT子句给字段指定缺省值。该数值可以是任何不含变量的表达式(不允许使用子查询和对本表中的其他字段的交叉引用)。缺省表达式的数据类型必须和字段类型匹配。 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
UNIQUE [NULLS [NOT] DISTINCT | NULLS IGNORE] index_parameters UNIQUE [NULLS [NOT] DISTINCT | NULLS IGNORE] (column_name [, ...]) index_parameters	UNIQUE约束表示表里的一个字段或多个字段的组合必须在全表范围内唯一。 其中[NULLS [NOT] DISTINCT | NULLS IGNORE]字段用来指定Unique唯一索引中索引列NULL值的处理方式。	默认取值：该参数默认缺省，即NULL值可重复插入。 在对插入的新数据和表中原始数据进行列的等值比较时，对于NULL值有以下三种处理方式： • NULLS DISTINCT：将每个NULL值视为不同的值，即允许NULL值重复插入。 • NULLS NOT DISTINCT：将所有NULL值视为相同的值，即索引中只保留一个NULL值条目，不允许NULL值重复插入，只有非NULL值且与原索引列中已有数据不相等的数据可插入。 • NULLS IGNORE：在等值比较时跳过NULL值。若索引列全为NULL，则NULL值可重复插入；部分索引列为NULL，只有非NULL值不相等，才可成功插入数据。 三种处理方式具体的行为如**表21-154**所示。 更多使用方法可以参见**关于NULLS DISTINCT、NULLS NOT DISTINCT、NULLS IGNORE使用上的场景差异**。 **注意** 如果没有声明DISTRIBUTE BY REPLICATION，则唯一约束的列集合中必须包含分布列。
PRIMARY KEY index_parameters PRIMARY KEY (column_name [, ...]) index_parameters	主键约束声明表中的一个或者多个字段只能包含唯一的非NULL值。 一个表只能声明一个主键。 **注意** 如果没有声明DISTRIBUTE BY REPLICATION，则主键约束的列集合中必须包含分布列。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
DEFERRABLE | NOT DEFERRABLE	设置该约束是否可推迟。一个不可推迟的约束将在每条命令之后马上检查。可推迟约束可以推迟到事务结尾使用SET CONSTRAINTS命令检查。缺省是NOT DEFERRABLE。目前，行存的UNIQUE约束和主键约束可以接受这个子句。所有其他约束类型都是不可推迟的。	-
INITIALLY IMMEDIATE | INITIALLY DEFERRED	如果约束是可推迟的，则这个子句声明检查约束的缺省时间。 • 如果约束是INITIALLY IMMEDIATE（缺省），则在每条语句执行之后就立即检查它； • 如果约束是INITIALLY DEFERRED ，则只有在事务结尾才检查它。 约束检查的时间可以用SET CONSTRAINTS命令修改。	-

表约束 table_constraint 参数说明

表约束table_constraint，参数解释同列约束相似不再赘述，参见**列约束CONSTRAINT constraint_name参数说明**。

```
{ CHECK ( expression ) |
  UNIQUE [ NULLS [NOT] DISTINCT | NULLS IGNORE ] ( column_name [, ... ] ) index_parameters |
  PRIMARY KEY ( column_name [, ... ] ) index_parameters}
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

LIKE 参数使用说明

like选项like_option：

```
{ INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES | STORAGE | COMMENTS |
RELOPTIONS | DISTRIBUTION | ALL }
```

新表与原来的表之间在创建动作完毕之后是完全无关的。在源表做的任何修改都不会传播到新表中，并且也不可能在扫描源表的时候包含新表的数据。

字段缺省表达式只有在声明了INCLUDING DEFAULTS之后才会包含进来。缺省是不包含缺省表达式的，即新表中所有字段的缺省值都是NULL。

非空约束将总是复制到新表中，CHECK约束则仅在指定了INCLUDING CONSTRAINTS的时候才复制，而其他类型的约束则永远也不会被复制。此规则同时适用于表约束和列约束。

被复制的列和约束并不使用相同的名字进行融合。如果明确的指定了相同的名字或者在另外一个LIKE子句中，将会报错。

- 如果指定了INCLUDING INDEXES，则源表上的索引也将在新表上创建，默认不建立索引。

版权所有 © 华为云计算技术有限公司

- 如果指定了INCLUDING STORAGE，则拷贝列的STORAGE设置也将被拷贝，默认情况下不包含STORAGE设置。
- 如果指定了INCLUDING COMMENTS，则源表列、约束和索引的注释也会被拷贝过来。默认情况下，不拷贝源表的注释。
- 如果指定了INCLUDING RELOPTIONS，则源表的存储参数（即源表的WITH子句）也将拷贝至新表。默认情况下，不拷贝源表的存储参数。
- 如果指定了INCLUDING DISTRIBUTION，则新表将拷贝源表的分布信息，包括分布类型和分布列，同时新表将不能再使用DISTRIBUTE BY子句。默认情况下，不拷贝源表的分布信息。
- INCLUDING ALL是INCLUDING DEFAULTS INCLUDING CONSTRAINTS INCLUDING INDEXES INCLUDING STORAGE INCLUDING COMMENTS INCLUDING RELOPTIONS INCLUDING DISTRIBUTION的简写形式。

WITH 子句参数说明

表 21-152 WITH 参数说明

参数	描述	取值范围
FILLFACTOR	一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是优良选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数对于列存表没有意义。	10~100
ORIENTATION	决定了表的数据的存储方式。 **注意** orientation不支持修改。	● COLUMN：表的数据将以列式存储。 ● ROW（缺省值）：表的数据将以行式存储。 ● ORC：表的数据将以ORC格式存储（仅HDFS表）。
COMPRESSION	该参数仅列存表支持，有效值为：LOW、MIDDLE或HIGH。默认值为LOW。 **说明** 暂不支持行存表压缩功能。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
MAX_BATCHROW	指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。	10000~60000 默认值：60000
PARTIAL_CLUSTER_ROWS	指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。	其有效值为大于等于10万。此值是MAX_BATCHROW的倍数。
enable_delta	指定了在列存表是否开启delta表。该参数只对列存表有效。不推荐使用列存带Delta表，否则会出现由于来不及merge而导致的磁盘膨胀以及性能劣化等问题。	默认值：off
DELTAROW_THRESHOLD	预留参数。该参数只对列存表有效。	0～60000 默认值：6000
COLD_TABLESPACE	指定冷分区保存的OBS表空间，仅冷热表支持。该参数仅支持列存分区表，且该参数不支持修改，需与STORAGE_POLICY同时使用。在指定STORAGE_POLICY时，可不设置该参数，默认为default_obs_tbs。	有效的OBS TABLESPACE名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
STORAGE_POLICY	指定冷热分区切换规则，仅冷热表支持。该参数需与COLD_TABLESPACE同时使用。	取值范围："冷热切换策略名称:冷热切换的阈值"，目前冷热切换的策略名称只支持LMT和HPN，LMT指按分区的最后更新时间切换，HPN指保留热分区的个数切换。 ● LMT：[day]：表示切换[day]时间前修改的热分区数据为冷分区，将该数据迁至OBS表空间中。其中[day]为整型，范围[0, 36500]，单位为天。 ● HPN:[hot_partition_num]：表示保留[hot_partition_num]个有数据的分区为热分区。保留规则为查找出有数据的分区的最大的Sequence ID，大于Sequence ID的无数据分区为热分区，并按这个Sequence ID从大到小保留[hot_partition_num]个分区为热分区；分区Sequence ID小于保留的最小热分区的Sequence ID的分区为冷分区，在冷热切换时，需要将数据迁移至OBS表空间中。其中[hot_partition_num]为整型，范围为[0,1600]。 **注意** ● 对于LIST分区，建议谨慎使用HPN策略，否则可能出现新增分区不是热分区的情况。
ENABLE_MULTI_TEMP_DISKCACHE	冷热表性能优化参数，仅支持冷热表设置，该参数仅9.1.0及以上版本支持。	on或off 默认值：on/true

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PERIOD	指定分区管理中自动创建分区的周期，并开启自动创建分区功能。仅支持行存、列存范围分区表、时序表以及冷热表；分区键唯一并且类型仅支持TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE；不支持存在maxvalue分区；(nowTime - boundaryTime) / PERIOD需要小于分区个数上限，其中nowTime为当前时间，boundaryTime为现有分区中最早的分区边界时间；不支持在小型机、加速集群上使用。	1 hour ~ 100 years **注意** • 在兼容Teradata或MySQL的数据库中，分区键类型为DATE时，PERIOD不能小于1 day。 • 建分区表时，如果设置了PERIOD，则可以只指定分区键不指定分区。建表时将创建两个默认分区，这两个默认分区的分区时间范围均为PERIOD。其中，第一个默认分区的边界时间是大于当前时间的第一个整时/整天/整周/整月/整年的时间，具体选择哪种整点时间取决于PERIOD的最大单位；第二个默认分区的边界时间是第一个分区边界时间加PERIOD。假设当前时间是2022-02-17 16:32:45，各种情况的第一个默认分区的分区边界选择如**表21-155**。 有关默认分区的更多内容，请参见**示例：创建不指定分区的自动分区管理分区表**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
TTL	指定分区管理中分区过期的时间，并开启自动删除分区功能。不支持单独设置，必须要提前或同时设置PERIOD，并且要大于或等于PERIOD。 **TTL（Time To Live）**指明该表的数据保存周期，超过TTL周期的数据将被清理，具体做法是以period周期，自动调用**proc_drop_partition (relname regclass, older_than interval)**函数。PERIOD和TTL的值为Interval类型，例如：“1 hour”，“1 day”，“1 week”，“1 month”，“1 year”，“1 month 2 day 3 hour”。 PERIOD指明按照时间划分的周期对数据进行分区，分区的大小可能对查询性能有影响，同时每隔周期时间会创建一个新的周期大小的分区，具体做法是以period周期，自动调用**proc_add_partition (relname regclass, boundaries_interval interval)**函数。	1 hour ~ 100 years

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
COLVERSION	指定列存存储格式的版本，支持不同存储格式版本之间的切换，但分区表不支持存储格式版本切换。	• 1.0：列存表的每列以一个单独的文件进行存储，文件名以relfilenode.C1.0、relfilenode.C2.0、relfilenode.C3.0等命名。 • 2.0：列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名。 默认值：2.0 需注意，OBS冷热表仅支持colversion 2.0格式。 **注意** 在建列存表时选择COLVERSION=2.0，相比于1.0存储格式，在以下场景中性能有明显提升： 1. 创建列存宽表场景下，建表时间显著减少。 2. roach备份数据场景下，备份时间显著减少。 3. build、catch up耗时显著减少。
SKIP_FPI_HINT	顺序扫描过程中，若需要写FPW(full page writes)日志时，该参数控制是否跳过设置HintBits操作。	默认值：false **注意** 设置SKIP_FPI_HINT=true时，在对某表执行checkpoint操作后，若对该表进行顺序扫描，将不再产生Xlog。适用于查询次数较少的中间表，有效减少Xlog的大小，提升查询性能。

PARTITION BY 子句参数说明

- partition_less_than_item：
  ```
  PARTITION partition_name VALUES LESS THAN ( { partition_value | MAXVALUE } )
  ```
- partition_start_end_item：
  ```
  PARTITION partition_name {
        {START(partition_value) END (partition_value) EVERY (interval_value)} |
        {START(partition_value) END ({partition_value | MAXVALUE})} |
        {START(partition_value)} |
        {END({partition_value | MAXVALUE})}
  }
  ```
- list_partition_item：
  ```
  PARTITION partition_name VALUES ( { (partition_value) [, ...] | DEFAULT } )
  ```


版权所有 © 华为云计算技术有限公司

表 21-153 PARTITION BY 子句参数说明

参数	描述	取值范围
PARTITION BY RANGE(partition_key)	指定范围分区策略语法，partition_key为分区键的名称。	• 对于从句是VALUES LESS THAN的语法格式： **注意** 对于从句是VALUE LESS THAN的语法格式，范围分区策略的分区键最多支持4列，且分区键只能是列名。当存在多个分区键时，一个列名只能出现一次，且相邻的两个分区键要使用逗号隔开。 该情形下，分区键支持的数据类型为：SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE PRECISION、CHARACTER VARYING(n)、VARCHAR(n)、CHARACTER(n)、CHAR(n)、CHARACTER、CHAR、TEXT、NVARCHAR2、NAME、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。 • 对于从句是START END的语法格式： **注意** 对于从句是START END的语法格式，范围分区策略的分区键仅支持1列。 该情形下，分区键支持的数据类型为：SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE PRECISION、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)]

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
		[WITH TIME ZONE]、DATE。
PARTITION BY LIST (partition_key,[...])	指定列表分区策略语法，partition_key为分区键的名称。	列表分区策略分区键支持的数据类型为：TINYINT、SMALLINT、INTEGER、BIGINT、NUMERIC/DECIMAL、TEXT、NVARCHAR2、VARCHAR(n)、CHAR、BPCHAR、TIME、TIME WITH TIMEZONE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、DATE、INTERVAL、SMALLDATETIME。 **注意** 列表分区策略的分区键最多支持4列。
partition_less_than_item	范围分区策略下分区（简称为范围分区）的定义语法。partition_name为范围分区的名称。partition_value为范围分区的上边界，取值依赖于partition_key的类型。MAXVALUE表示分区的上边界，它通常用于设置最后一个范围分区的上边界。 `PARTITION partition_name VALUES LESS THAN ( { partition_value | MAXVALUE } )` **注意** • 每个分区都需要指定一个上边界。 • 分区上边界的类型应当和分区键的类型一致。 • 分区列表是按照分区上边界升序排列的，值较小的分区位于值较大的分区之前。 • 如果分区键由多个字段组成，比较大小时，先比较第一个字段，当第一个字段相等时比较第二个字段，以此类推。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
partition_start_end_item	使用起始值以及间隔值定义范围分区的语法。 PARTITION partition_name {START (partition_value) END (partition_value) EVERY (interval_value)} | {START (partition_value) END (partition_value| MAXVALUE)} | {START(partition_value)} | {END (partition_value| MAXVALUE)}	各参数含义如下： • partition_name：范围分区的名称或名称前缀，除以下情形外（假定其中的partition_name是p1），均为分区的名称。 – 若该定义是START+END+EVERY从句，则语义上定义的分区的名称依次为p1_1, p1_2, ...。例如对于定义“PARTITION p1 START(1) END(4) EVERY(1)”，则生成的分区是：[1, 2), [2, 3) 和 [3, 4)，名称依次为p1_1, p1_2和p1_3，即此处的p1是名称前缀。 – 若该定义是第一个分区定义，且该定义有START值，则范围（MINVALUE, START）将自动作为第一个实际分区，其名称为p1_0，然后该定义语义描述的分区名称依次为p1_1, p1_2, ...。例如对于完整定义“PARTITION p1 START(1), PARTITION p2 START(2)”，则生成的分区是：(MINVALUE, 1), [1, 2) 和 [2, MAXVALUE)，其名称依次为p1_0, p1_1和p2，即此处p1是名称前缀，p2是分区名称。这里MINVALUE表示最小值。 • partition_value：范围分区的端点值（起始或终点），取值依赖于partition_key的类型，不可是MAXVALUE。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
		● interval_value：对[START，END) 表示的范围进行切分，interval_value是指定切分后每个分区的宽度，不可是MAXVALUE；如果（ END-START ）值不能整除以EVERY值，则仅最后一个分区的宽度小于EVERY值。 ● MAXVALUE：表示最大值，它通常用于设置最后一个范围分区的上边界。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
		注意 1. 在创建分区表若第一个分区定义含START值，则范围（MINVALUE，START）将自动作为实际的第一个分区。 2. START END语法需要遵循以下限制： ● 每个partition_start_end_item中的START值（如果有的话，下同）必须小于其END值； ● 相邻的两个partition_start_end_item，第一个的END值必须等于第二个的START值； ● 每个partition_start_end_item中的EVERY值必须是正向递增的，且必须小于（END-START）值； ● 每个分区包含起始值，不包含终点值，即形如：[起始值，终点值)，起始值是MINVALUE时则不包含； ● 一个partition_start_end_item创建的每个分区所属的TABLESPACE一样； ● partition_name作为分区名称前缀时，其长度不要超过57字节，超过时自动截断； ● 在创建、修改分区表时请注意分区表的分区总数不可超过最大限制（32767）； 3. 在创建分区表时START END与LESS THAN语法不可混合使用。 4. 即使创建分区表时使用START END语法，备份（gs_dump）出的SQL语句也是VALUES LESS THAN语法格式。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
list_partition_item	列表分区策略下分区（简称为列表分区）的定义语法。partition_name为分区的名称。partition_value为列表分区边界的一个枚举值，取值依赖于partition_key的类型。DEFAULT表示默认分区的边界。 PARTITION partition_name VALUES ({ (partition_value) [, ...] | DEFAULT })	对于列表分区表，存在以下约定和约束： • 边界值为DEFAULT的分区，称之为默认分区。 • 每个列表分区表只能有一个DEFAULT分区。 • 分区表的所有分区数不超过32767个，所有分区的边界值个数不大于32767个。 • 不管分区键的个数，DEFAULT分区的边界只能是一个DEFAULT。 • 如果分区键由多个字段组成，每个partition_value需要包含所有分区键的值，当分区键只有一列时，partition_value两侧的括号可以省略，参见**示例：创建多个分区键的list分区表**。 • 如果分区键由多个字段组成，比较大小时，先逐个字段比较大小，任何一个字段值不一样即可认为是不一样的键值。 • 边界中不同的partition_value值不能重复。 • 数据插入时，如果数据的分区键值能匹配任何非DEFAULT分区的边界，那么数据会写入对应的分区；否则数据会写入DEFAULT分区。

表 21-154 唯一索引中索引列 NULL 值的处理方式

字段控制	索引列全为NULL	部分索引列为NULL
NULLS DISTINCT	可重复插入	可重复插入
NULLS NOT DISTINCT	不可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功

版权所有 © 华为云计算技术有限公司

字段控制	索引列全为NULL	部分索引列为NULL
NULLS IGNORE	可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功

表 21-155 分区边界选择

period	period最大单位	第一个默认分区的分区边界
1hour	Hour	2022-02-17 17:00:00
1day	Day	2022-02-18 00:00:00
1month	Month	2022-03-01 00:00:00
13month	Year	2023-01-01 00:00:00

DISTRIBUTE BY 参数使用说明

默认值：由GUC参数default_distribution_mode控制。

- 当default_distribution_mode=roundrobin时，DISTRIBUTE BY的默认值按如下规则选取：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，则选取ROUNDROBIN分布。
- 当default_distribution_mode=hash时，DISTRIBUTE BY的默认值按如下规则选取：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，但存在数据类型支持作分布列的列，则选取HASH分布，分布列为第一个数据类型支持作分布列的列。
  c. 若建表时不包含主键/唯一约束，也不存在数据类型支持作分布列的列，选取ROUNDROBIN分布。

以下数据类型支持作为分布列：

- INTEGER TYPES：TINYINT，SMALLINT，INT，BIGINT，NUMERIC/DECIMAL
- CHARACTER TYPES：CHAR，BPCHAR，VARCHAR，VARCHAR2，NVARCHAR2，TEXT
- DATE/TIME TYPES：DATE，TIME，TIMETZ，TIMESTAMP，TIMESTAMPTZ，INTERVAL，SMALLDATETIME

示例：使用 LESS THAN 语法创建 range 分区表

Range分区表customer_address含有4个分区，分区键为integer类型。分区的范围分别为：ca_address_sk<2450815，2450815<= ca_address_sk< 2451179，2451179<= ca_address_sk< 2451544，2451544<=ca_address_sk。

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE customer_address
(
    ca_address_sk       INTEGER                 NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)              ,
    ca_street_type      CHARACTER(15)                      ,
    ca_suite_number     CHARACTER(10)
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE(ca_address_sk)
(
        PARTITION P1 VALUES LESS THAN(2450815),
        PARTITION P2 VALUES LESS THAN(2451179),
        PARTITION P3 VALUES LESS THAN(2451544),
        PARTITION P4 VALUES LESS THAN(MAXVALUE)
);
```

查看分区表信息。

```
SELECT relname, boundaries FROM pg_partition p where p.parentid='customer_address'::regclass ORDER BY
1;
     relname      | boundaries
------------------+------------
 customer_address |
 p1               | {2450815}
 p2               | {2451179}
 p3               | {2451544}
 p4               | {NULL}
(5 rows)
```

查询分区P1的行数。

```
SELECT count(*) FROM customer_address PARTITION (P1);
SELECT count(*) FROM customer_address PARTITION FOR (2450815);
```

示例：开启和关闭行迁移功能 ROW MOVEMENT

在更新某行的分区键为新值且新值属于其他分区时，如果打开ROW MOVEMENT，则该行数据对应迁移到新的分区中。

频繁更新分区键时，ENABLE ROW MOVEMENT 可能导致额外I/O开销，需评估业务需求。

创建按日期范围分区的表，并启用行迁移。

```
DROP TABLE IF EXISTS sale_data;
CREATE TABLE  sales_data (
    sale_id  INT,
    product_name VARCHAR(100),
    sale_date DATE,
    amount DECIMAL(10, 2)
)
PARTITION BY RANGE (sale_date) (
    PARTITION p_2023q1 VALUES LESS THAN ('2023-04-01'),
    PARTITION p_2023q2 VALUES LESS THAN ('2023-07-01'),
    PARTITION p_2023q3 VALUES LESS THAN ('2023-10-01'),
    PARTITION p_2023q4 VALUES LESS THAN ('2024-01-01')
)
ENABLE ROW MOVEMENT;
```

插入测试数据。

```
INSERT INTO sales_data (sale_id, product_name, sale_date, amount)
VALUES
(1, 'Product A', '2023-02-15', 1000.00),  -- 属于 p_2023q1
(2, 'Product B', '2023-05-20', 1500.00);  -- 属于 p_2023q2
```


版权所有 © 华为云计算技术有限公司

查询各分区数据分布。

```
SELECT tableoid::regclass AS partition, * FROM sales_data ORDER BY sale_id;
```

```
 partition  | sale_id | product_name |      sale_date      | amount
------------+---------+--------------+---------------------+---------
 2147617615 |       1 | Product A    | 2023-02-15 00:00:00 | 1000.00
 2147617616 |       2 | Product B    | 2023-05-20 00:00:00 | 1500.00
(2 rows)
```

尝试更新分区键值（从 p_2023q1 移动到 p_2023q2）。

```
UPDATE sales_data SET sale_date = '2023-05-01'  WHERE sale_id = 1;
```

再次查询各分区数据分布，确认行已移动。

```
SELECT tableoid::regclass AS partition, *  FROM sales_data  ORDER BY sale_id;
```

```
 partition  | sale_id | product_name |      sale_date      | amount
------------+---------+--------------+---------------------+---------
 2147617616 |       1 | Product A    | 2023-05-01 00:00:00 | 1000.00
 2147617616 |       2 | Product B    | 2023-05-20 00:00:00 | 1500.00
(2 rows)
```

关闭行迁移功能。

```
ALTER TABLE sales_data DISABLE ROW MOVEMENT;
```

再次修改分区键（从 p_2023q2 移动回到 p_2023q2）。

```
UPDATE sales_data SET sale_date = '2023-02-15'  WHERE sale_id = 1;
```

回显提示，行迁移关闭，不允许更新分区键。

```
gaussdb=> UPDATE sales_data SET sale_date = '2023-02-15'  WHERE sale_id = 1;
ERROR:  dn_6003_6004: fail to update partitioned table "sales_data"
DETAIL:  disable row movement
```

示例：使用 START END 语法创建列存 range 分区表

```
CREATE TABLE customer_address_SE
(
    ca_address_sk       INTEGER               NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)               ,
    ca_street_type      CHARACTER(15)                     ,
    ca_suite_number     CHARACTER(10)
)
WITH (ORIENTATION = COLUMN)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE(ca_address_sk)
(
    PARTITION p1 START(1) END(1000) EVERY(200),
    PARTITION p2 END(2000),
    PARTITION p3 START(2000) END(5000)
);
```

查看分区表信息。

```
SELECT relname, boundaries FROM pg_partition p where p.parentid='customer_address_SE'::regclass ORDER BY 1;
```

 版权所有 © 华为云计算技术有限公司

```
    relname      | boundaries
---------------------+------------
 customer_address_se |
 p1_0                | {1}
 p1_1                | {201}
 p1_2                | {401}
 p1_3                | {601}
 p1_4                | {801}
 p1_5                | {1000}
 p2                 | {2000}
 p3                 | {5000}
(9 rows)
```

示例：START END 语法创建、修改 Range 分区表

创建临时schema。

```
CREATE SCHEMA tpcds;
SET CURRENT_SCHEMA TO tpcds;
```

创建分区表，分区键是integer类型。

```
CREATE TABLE tpcds.startend_pt (c1 INT, c2 INT)
DISTRIBUTE BY HASH (c1)
PARTITION BY RANGE (c2) (
    PARTITION p1 START(1) END(1000) EVERY(200),
    PARTITION p2 END(2000),
    PARTITION p3 START(2000) END(2500),
    PARTITION p4 START(2500),
    PARTITION p5 START(3000) END(5000) EVERY(1000)
)
ENABLE ROW MOVEMENT;
```

查看分区表信息。

```
SELECT relname, boundaries FROM pg_partition p where p.parentid='tpcds.startend_pt'::regclass ORDER BY
1;
   relname   | boundaries
-------------+------------
 p1_0        | {1}
 p1_1        | {201}
 p1_2        | {401}
 p1_3        | {601}
 p1_4        | {801}
 p1_5        | {1000}
 p2          | {2000}
 p3          | {2500}
 p4          | {3000}
 p5_1        | {4000}
 p5_2        | {5000}
 tpcds.startend_pt |
(12 rows)
```

导入数据，查看分区数据量。

```
INSERT INTO tpcds.startend_pt VALUES (GENERATE_SERIES(0, 4999), GENERATE_SERIES(0, 4999));
SELECT COUNT(*) FROM tpcds.startend_pt PARTITION FOR (0);
 count
-------
     1
(1 row)

SELECT COUNT(*) FROM tpcds.startend_pt PARTITION (p3);
 count
-------
   500
(1 row)
```

查看分区表信息。

 版权所有 © 华为云计算技术有限公司

```
SELECT relname, boundaries FROM pg_partition p where p.parentid='tpcds.startend_pt'::regclass ORDER BY
1;
   relname   | boundaries
-------------+------------
 p1_0        | {1}
 p1_1        | {201}
 p1_2        | {401}
 p1_3        | {601}
 p1_4        | {801}
 p1_5        | {1000}
 p2          | {2000}
 p3          | {2500}
 p4          | {3000}
 p5_1        | {4000}
 p6_1        | {5300}
 p6_2        | {5600}
 p6_3        | {5900}
 p71         | {6000}
 q1_1        | {4250}
 q1_2        | {4500}
 q1_3        | {4750}
 q1_4        | {5000}
 tpcds.startend_pt |
(19 rows)
```

示例：创建一个分区键的 list 分区表

```
CREATE TABLE data_list
(
    id int,
    time int,
    salary decimal(12,2)
)
PARTITION BY LIST (time)
(
        PARTITION P1 VALUES (202209),
        PARTITION P2 VALUES (202210,202208),
        PARTITION P3 VALUES (202211),
        PARTITION P4 VALUES (202212),
        PARTITION P5 VALUES (202301)
);
```

示例：创建多个分区键的 list 分区表

分区表有两个分区键：period，city。

```
CREATE TABLE sales_info
(
sale_time  timestamptz,
period     int,
city       text,
price      numeric(10,2),
remark     varchar2(100)
)
DISTRIBUTE BY HASH(sale_time)
PARTITION BY LIST (period, city)
(
PARTITION north_2022 VALUES (('202201', 'north1'), ('202202', 'north2')),
PARTITION south_2022 VALUES (('202201', 'south1'), ('202202', 'south2'), ('202203', 'south2')),
PARTITION rest VALUES (DEFAULT)
);
```

示例：创建不指定分区的自动分区管理分区表

指定分区管理中自动创建分区的周期PERIOD为1 day，分区键为time。

```
CREATE TABLE time_part
 (
```

 版权所有 © 华为云计算技术有限公司

```
    id integer,
    time timestamp
) with (PERIOD='1 day')
partition by range(time);
```

建表时将创建两个默认分区，第一个默认分区的边界时间是大于当前时间的第一个整天的时间，即2022-12-13 00:00:00；第二个默认分区的边界时间是第一个分区边界时间加PERIOD，即2022-12-13 00:00:00+1day=2022-12-14 00:00:00。

```
SELECT now();
              now
-------------------------------
 2022-12-12 20:41:21.603172+08
(1 row)

SELECT relname, boundaries FROM pg_partition p where p.parentid='time_part'::regclass ORDER BY 1;
    relname     |       boundaries
----------------+-------------------------
 default_part_1 | {"2022-12-13 00:00:00"}
 default_part_2 | {"2022-12-14 00:00:00"}
 time_part      |
(3 rows)
```

示例：创建指定分区的自动分区管理分区表

```
CREATE TABLE CPU(
    id integer,
    idle numeric,
    IO numeric,
    scope text,
    IP text,
    time timestamp
) with (TTL='7 days',PERIOD='1 day')
partition by range(time)
(
    PARTITION P1 VALUES LESS THAN('2022-01-05 16:32:45'),
    PARTITION P2 VALUES LESS THAN('2022-01-06 16:56:12')
);
```

示例：按照月份创建分区表

创建分区表customer_address，含有13个分区，分区键为date类型。

```
CREATE TABLE customer_address
(
    ca_address_sk       integer           NOT NULL,
    ca_address_date       date            NOT NULL
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE (ca_address_date)
(
        PARTITION p202001 VALUES LESS THAN('20200101'),
        PARTITION p202002 VALUES LESS THAN('20200201'),
        PARTITION p202003 VALUES LESS THAN('20200301'),
        PARTITION p202004 VALUES LESS THAN('20200401'),
        PARTITION p202005 VALUES LESS THAN('20200501'),
        PARTITION p202006 VALUES LESS THAN('20200601'),
        PARTITION p202007 VALUES LESS THAN('20200701'),
        PARTITION p202008 VALUES LESS THAN('20200801'),
        PARTITION p202009 VALUES LESS THAN('20200901'),
        PARTITION p202010 VALUES LESS THAN('20201001'),
        PARTITION p202011 VALUES LESS THAN('20201101'),
        PARTITION p202012 VALUES LESS THAN('20201201'),
        PARTITION p202013 VALUES LESS THAN(MAXVALUE)
);
```

插入数据。

版权所有 © 华为云计算技术有限公司

```
INSERT INTO customer_address values('1','20200215');
INSERT INTO customer_address values('7','20200805');
INSERT INTO customer_address values('9','20201111');
INSERT INTO customer_address values('4','20201231');
```

查询分区。

```
SELECT * FROM customer_address PARTITION(p202009);
 ca_address_sk |    ca_address_date
---------------+---------------------
             7 | 2020-08-05 00:00:00
(1 row)
```

示例：使用 START END 语法一次创建含有多个分区的分区表

- 创建分区表day_part，每一天为一个分区，分区键为date类型。

  ```
  CREATE table day_part(id int,d_time date)
  DISTRIBUTE BY HASH (id)
  PARTITION BY RANGE (d_time)
  (PARTITION p1 START('2022-01-01') END('2022-01-31') EVERY(interval '1 day'));
  ALTER TABLE  day_part ADD PARTITION pmax VALUES LESS THAN (maxvalue);
  ```

- 创建分区表week_part，每7天为一个分区，分区键为date类型。

  ```
  CREATE table week_part(id int,w_time date)
  DISTRIBUTE BY HASH (id)
  PARTITION BY RANGE (w_time)
  (PARTITION p1 START('2021-01-01') END('2022-01-01') EVERY(interval '7 day'));
  ALTER TABLE  week_part ADD PARTITION pmax VALUES LESS THAN (maxvalue);
  ```

- 创建分区表month_part，每一个月为一个分区，分区键为date类型。

  ```
  CREATE table month_part(id int,m_time date)
  DISTRIBUTE BY HASH (id)
  PARTITION BY RANGE (m_time)
  (PARTITION p1 START('2021-01-01') END('2022-01-01') EVERY(interval '1 month'));
  ALTER TABLE  month_part ADD PARTITION pmax VALUES LESS THAN (maxvalue);
  ```

示例：创建冷热表

仅支持列存分区表，使用obs默认表空间，冷热切换规则设置LMT为30。

```
CREATE TABLE cold_hot_table
(
    W_WAREHOUSE_ID          CHAR(16)             NOT NULL,
    W_WAREHOUSE_NAME          VARCHAR(20)                ,
    W_STREET_NUMBER          CHAR(10)                   ,
    W_STREET_NAME           VARCHAR(60)                ,
    W_STREET_ID             CHAR(15)                   ,
    W_SUITE_NUMBER           CHAR(10)
)
WITH (ORIENTATION = COLUMN, storage_policy = 'LMT:30')
DISTRIBUTE BY HASH (W_WAREHOUSE_ID)
PARTITION BY RANGE(W_STREET_ID)
(
    PARTITION P1 VALUES LESS THAN(100000),
    PARTITION P2 VALUES LESS THAN(200000),
    PARTITION P3 VALUES LESS THAN(300000),
    PARTITION P4 VALUES LESS THAN(MAXVALUE)
)ENABLE ROW MOVEMENT;
```

相关链接

ALTER TABLE PARTITION，**DROP TABLE**

版权所有 © 华为云计算技术有限公司

21.13.4.3 CREATE TABLE AS

功能描述

根据查询结果创建表。适用于数据迁移或复制场景。

CREATE TABLE AS创建一个表，并使用SELECT查询结果填充该表。该表的字段与SELECT输出字段的名称及数据类型关联。不过用户可以通过明确地给出一个字段名称列表来覆盖SELECT输出字段的名称。

CREATE TABLE AS对源表进行一次查询，然后将数据写入新表中，而查询视图结果会根据源表的变化而有所改变。相比之下，每次做查询的时候，视图都重新计算定义它的SELECT语句。

注意事项

- 分区表不能采用此方式进行创建。
- 如果在建表过程中数据库系统发生故障，系统恢复后可能无法自动清除之前已创建的、大小非0的磁盘文件。此种情况出现概率小，不影响数据库系统的正常运行。
- UNLOGGED表无主备机制，在系统故障或异常断点等情况下，会有数据丢失风险，不可用来存储基础数据。
- UNLOGGED表不能保证数据的安全性，用户应该在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。

语法格式

```
CREATE [ [ GLOBAL | LOCAL | VOLATILE ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE table_name
    [ (column_name [, ...] ) ]
    [ WITH ( {storage_parameter = value} [, ... ] ) ]
    [ COMPRESS | NOCOMPRESS ]

    [ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { [HASH ] ( column_name ) } } ]

    [ COMMENT [=] 'text' ]
    AS query
    [ WITH [ NO ] DATA ];
```

参数说明

表 21-156 CREATE TABLE AS 参数说明

参数	描述	取值范围
GLOBAL | LOCAL | VOLATILE	指定临时表类型。	具体参考**参数说明**。
TEMPORARY | TEMP	指定TEMP或TEMPORARY关键字时创建的表为临时表。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
UNLOGGED	指定表为非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是，它也是不安全的，非日志表在冲突或异常关机后会被自动删截。非日志表中的内容也不会被复制到备用服务器中。在该类表中创建的索引也不会被自动记录。 当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。	自9.1.0版本，UNLOGGED表默认存储在pg_unlogged表空间下，且不可迁移或指定到其他表空间；从低版本升级到9.1.0版本后，旧版本创建的UNLOGGED表还存储在原表空间下。 由于实例异常重启时会对UNLOGGED表进行重置，所以可能影响实例RTO。自9.1.0版本提供UNLOGGED表迁移脚本switch_unlogged_tablespace.py，配合GUC参数enable_unlogged_tablespace_compat，可以优化RTO。
table_name	要创建的表名。	字符串，需符合**标识符命名规范**。
column_name	新表中要创建的字段名。	字符串，需符合**标识符命名规范**。
WITH (storage_parameter [= value] [, ...])	WITH子句设置表或其索引的存储选项。	参数的详细说明如**表21-157**所示。
COMPRESS | NOCOMPRESS	• 指定关键字COMPRESS，对表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。 • 指定关键字NOCOMPRESS则不对表进行压缩。 缺省值：NOCOMPRESS，即不对元组数据进行压缩	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
DISTRIBUTE BY	指定表如何在节点之间分布或者复制。 ● REPLICATION：表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。 ● ROUNDROBIN：表的每一行被依次发送给各个DN，在这种分布策略下可以保证数据分布不会存在倾斜，但是因为数据分布节点是随机的，导致这类表在计算时会更大概率的触发此表的重分布。各列倾斜都比较严重的大表推荐使用此种分布策略。（ROUNDROBIN仅8.1.2及以上版本支持） ● HASH (column_name)：对指定的列进行Hash，通过映射，把数据分布到指定DN。 默认值：由GUC参数default_distribution_mode控制。DISTRIBUTE BY的默认值选取规则请参考此表下方的须知内容。	以下数据类型支持作为分布列： ● INTEGER TYPES：TINYINT，SMALLINT，INT，BIGINT，NUMERIC/DECIMAL ● CHARACTER TYPES：CHAR，BPCHAR，VARCHAR，VARCHAR2，NVARCHAR2，TEXT ● DATE/TIME TYPES：DATE，TIME，TIMETZ，TIMESTAMP，TIMESTAMPTZ，INTERVAL，SMALLDATETIME
COMMENT [=] 'text'	COMMENT子句可以在创建表时指定表注释。	-
AS query	一个SELECT VALUES命令或者一个运行预备好的SELECT或VALUES查询的EXECUTE命令。	-
[WITH [NO] DATA]	创建表时，是否也插入查询到的数据。默认是要数据，选择“NO”参数时，则不要数据。	-

版权所有 © 华为云计算技术有限公司

须知

DISTRIBUTE BY参数说明：

- 当指定DISTRIBUTE BY HASH (column_name)参数时，创建主键和唯一索引必须包含“ column_name”列。
- 当被参照表指定DISTRIBUTE BY HASH (column_name)参数时，参照表的外键必须包含“ column_name”列。
- 当default_distribution_mode=roundrobin时，DISTRIBUTE BY的默认值按如下规则选取：
  1. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  2. 若建表时不包含主键/唯一约束，则选取ROUNDROBIN分布。
- 当default_distribution_mode=hash时，DISTRIBUTE BY的默认值按如下规则选取：
  1. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  2. 若建表时不包含主键/唯一约束，但存在数据类型支持作分布列的列，则选取HASH分布，分布列为第一个数据类型支持作分布列的列。
  3. 若建表时不包含主键/唯一约束，也不存在数据类型支持作分布列的列，选取ROUNDROBIN分布。

表 21-157 storage_parameter 参数说明

storage_parameter 参数	描述	取值范围
FILLFACTOR	一个表的填充因子（ fillfactor ）是一个介于10和100之间的百分数。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是合适的选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。 （ ）默认值为100，表示完全填充。 该参数只对行存表有效。	10~100

版权所有 © 华为云计算技术有限公司

storage_parameter 参数	描述	取值范围
ORIENTATION	指定表数据的存储方式，即行存方式、列存方式，该参数设置成功后就不再支持修改。 默认值：ROW，即行存方式。	• ROW，表示表的数据将以行式存储。 行存储适合于OLTP业务，此类型的表上交互事务比较多，一次交互会涉及表中的多个列，用行存查询效率较高。 • COLUMN，表示表的数据将以列式存储。 列存储适合于数据仓库业务，此类型的表上会做大量的汇聚计算，且涉及的列操作较少。
COMPRESSION	指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比也越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。 **说明** 暂不支持行存表压缩功能。	该参数仅列存表支持，有效值为：LOW、MIDDLE或HIGH。默认值为LOW。
MAX_BATCHROW	指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。 默认值：60000	10000~60000
PARTIAL_CLUSTER_ROWS	指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。 默认值：4,200,000	600000~2147483647
enable_delta	指定了在列存表是否开启delta表。该参数只对列存表有效。 不推荐使用列存带Delta表，否则会出现由于来不及merge而导致的磁盘膨胀以及性能劣化等问题。 默认值：off	on或off

版权所有 © 华为云计算技术有限公司

storage_parameter 参数	描述	取值范围
COLVERSION	指定列存存储格式的版本，支持不同存储格式版本之间的切换。 默认值：2.0 **说明** 在建列存表时选择COLVERSION=2.0，相比于1.0存储格式，在以下场景中性能有明显提升： 1. 创建列存宽表场景下，建表时间显著减少。 2. roach备份数据场景下，备份时间显著减少。 3. build、catch up耗时显著减少。	取值范围： ● 1.0：列存表的每列以一个单独的文件进行存储，文件名以relfilenode.C1.0、relfilenode.C2.0、relfilenode.C3.0等命名。 ● 2.0：列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名
SKIP_FPI_HINT	顺序扫描过程中，若需要写FPW（full page writes）日志时，该参数控制是否跳过设置HintBits操作。 默认值：false **说明** 设置SKIP_FPI_HINT=true时，在对某表执行checkpoint操作后，若对该表进行顺序扫描，将不再产生Xlog。适用于查询次数较少的中间表，有效减少Xlog的大小，提升查询性能。	true或false

示例

创建表customer。

```
DROP TABLE IF EXISTS customer;
CREATE TABLE customer
(
   C_CUSTKEY     BIGINT NOT NULL CONSTRAINT C_CUSTKEY_pk PRIMARY KEY  ,
   C_NAME        VARCHAR(25)  ,
   C_ADDRESS     VARCHAR(40)  ,
   C_NATIONKEY   INT NOT NULL  CHECK (C_NATIONKEY > 0)
)
DISTRIBUTE BY HASH(C_CUSTKEY);
```

创建表store_returns_t1并插入CUSTOMER表中C_CUSTKEY字段中大于4795的数值。

```
CREATE TABLE store_returns_t1 AS SELECT * FROM CUSTOMER WHERE C_CUSTKEY > 4795;
```

使用store_returns_t1复制一个新表store_returns_t2。

```
CREATE TABLE store_returns_t2 AS table store_returns_t1;
```


版权所有 © 华为云计算技术有限公司

相关链接

CREATE TABLE，SELECT

21.13.4.4 ALTER TABLE

功能描述

修改表，包括修改表的定义、重命名表、重命名表中指定的列、重命名表的约束、设置表的所属模式、添加/更新多个列、打开/关闭行访问控制开关。

只有表的所有者或者被授予了表ALTER权限的用户有权限执行ALTER TABLE命令，系统管理员默认拥有此权限。但要修改表的所有者或者修改表的模式，当前用户必须是该表的所有者或者系统管理员，且该用户是新的所有角色的直接或间接成员。

注意事项

表 21-158 ALTER TABLE 注意事项

约束项	详细内容
不支持修改表数据的存储方式	建表后不支持修改存储参数ORIENTATION（该参数指定表为行存或列存）。
列存表约束支持范围	列存表支持PARTIAL CLUSTER KEY，不支持外键表级约束。列存表自8.1.1集群版本开始建表时支持创建主键和唯一表级约束。
列存表约束修改限制	• 支持通过ALTER方式对已有列存表添加非空约束NOT NULL以及主键约束。该约束仅8.2.0及以上集群版本支持。 • 通过约束名删除PRIMARY KEY约束时，不会删除NOT NULL约束。如有需要，可手动删除NOT NULL约束。
列存表的字段约束	列存表支持的字段约束包括NULL、NOT NULL和DEFAULT常量值；对字段约束的修改，当前支持对DEFAULT值的修改（SET DEFAULT）、删除（DROP DEFAULT）和NOT NULL约束的删除。
列存表支持操作	• 添加字段ADD COLUMN，对于添加的字段和修改的字段类型要求是列存支持的**数据类型**。 • 修改字段的数据类型ALTER TYPE，ALTER TYPE的USING选项只支持常量表达式和涉及本字段的表达式，暂不支持涉及其他字段的表达式。 • 设置单个字段的收集目标SET STATISTICS。 • 支持更改表名称。 • 支持删除字段DROP COLUMN。
分布列修改限制	不支持修改已有表的分布列数据类型。
行存REPLICATION表限制	行存REPLICATION分布表不支持将系统列设置为主键。

版权所有 © 华为云计算技术有限公司

约束项	详细内容
不支持增加自增列	不支持增加自增列，或者增加DEFAULT值中包含nextval()表达式的列。
OBS冷热表修改约束	• OBS冷热表不支持ALTER RESET的cold_tablespace，storage_policy参数，不支持修改COLVERSION为1.0。 • 可将原列存COLVERSION为2.0表ALTER为OBS冷热表，需同时增加cold_tablespace和storage_policy参数。 • ALTER TABLE支持针对RELOPTIONS的storage_policy的选项更改。冷热切换策略发生更改后，不改变现有冷数据的冷热属性，只改变下一次执行冷热切换的规则，下一次执行冷热切换命令时将按新规则进行冷热切换。
V3表	V3表（COLVERSION为3.0）不支持修改COLVERSION，也不支持将非V3表切换为V3表（即COLVERSION为2.0不支持切为3.0）。
触发表重建的操作	对表执行以下ALTER TABLE操作时会触发表重建（表重建过程中会先把数据转储到一个新的数据文件中，重建完成之后会删除原始文件），当表比较大时，重建会消耗较多的磁盘空间。当磁盘空间不足时，要谨慎对待大表ALTER TABLE操作，防止触发集群只读。 • 修改列数据类型。 • 行存表增加列（包括oid列）。 • 列存表修改COLVERSION。 • 列存表增加列指定DEFAULT常量值，DEFAULT值中包含volatile函数以及DEFAULT值非NULL且不属于特定数据类型中的任何一个。
其他	• 修改列存表存储参数COLVERSION或者enable_delta时，不能与其他ALTER操作同时进行。 • 对unlogged表执行ALTER TABLE时，不允许指定表空间。对非unlogged表执行ALTER TABLE时，不允许指定表空间为pg_unlogged。 • 全局临时表不支持修改分布列。

⚠ 警告

- 避免在业务高峰期执行ALTER TABLE/ALTER TABLE PARTITION（增删改查、DROP PARTITION）、TRUNCATE操作，避免有长SQL阻塞ALTER、TRUNCATE操作或SQL业务被ALTER、TRUNCATE阻塞。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

版权所有 © 华为云计算技术有限公司

ALTER TABLE 语法格式

```
ALTER TABLE [ IF EXISTS ] { table_name [*] | ONLY table_name | ONLY ( table_name ) }
    action [, ... ];
```

action表示支持以下修改操作：

- 修改表已存在字段的数据类型，可同时修改空约束，default值和注释信息，只允许相同大类的类型转换（数值之间，字符串之间，时间之间等）。
  ```
  ALTER TABLE [ IF EXISTS ] table_name
      MODIFY ( { column_name data_type | [ CONSTRAINT constraint_name ] NOT NULL [ ENABLE ] |
          [ CONSTRAINT constraint_name ] NULL | DEFAULT default_expr | COMMENT 'text' } [, ...] );
  ```
- 重命名表。对名称的修改不会影响所存储的数据；不支持跨schema对表进行重命名操作；支持新表名前带有原表的schema名；如果new_table_name不指定schema，则表示new_table_name与原表处于同一schema中。
  ```
  ALTER TABLE [ IF EXISTS ] table_name
      RENAME TO new_table_name;
  ALTER TABLE [ IF EXISTS ] table_name
      RENAME TO schema.new_table_name;
  ```
- 重命名表中指定的列。
  ```
  ALTER TABLE [ IF EXISTS ] { table_name [*] | ONLY table_name | ONLY ( table_name )}
      RENAME [ COLUMN ] column_name TO new_column_name;
  ```
- 重命名表的约束。
  ```
  ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
      RENAME CONSTRAINT constraint_name TO new_constraint_name;
  ```
- 设置表的所属模式。
  ```
  ALTER TABLE [ IF EXISTS ] table_name
      SET SCHEMA new_schema;
  ```

  说明

  - SET SCHEMA操作不支持修改为系统内部模式，当前仅支持用户模式之间的修改。
  - 这种形式把表移动到另外一个模式。相关的索引、约束都跟着移动。目前序列不支持改变schema。 若该表拥有序列，需要将序列删除，重建，或者取消拥有关系， 才能将表schema更改成功。
  - 要修改一个表的模式，用户必须在新模式上拥有CREATE权限。要把该表添加为一个父表的新子表，用户必须同时又是父表的所有者。要修改所有者，用户还必须是新的所有角色的直接或间接成员，并且该成员必须在此表的模式上有CREATE权限。这些限制规定了该用户不能做出了重建和删除表之外的事情。不过，系统管理员可以以任何方式修改任意表的所有权限。
- 添加多个列。
  ```
  ALTER TABLE [ IF EXISTS ] table_name
      ADD ( { column_name data_type [ compress_mode ] [ COLLATE collation ] [ column_constraint
  [ ... ] ]} [, ...] );
  ```
- 更新多个列。
  ```
  ALTER TABLE [ IF EXISTS ] table_name
      MODIFY ( { column_name data_type | column_name [ CONSTRAINT constraint_name ] NOT NULL
  [ ENABLE ] | column_name [ CONSTRAINT constraint_name ] NULL } [, ...] );
  ```
- 删除指定表上的索引。
  ```
  ALTER TABLE table_name DROP INDEX index_name;
  ```

  说明

  - 除了RENAME和SET SCHEMA之外所有动作都可以捆绑在一个经过多次修改的列表中并行使用。比如，可以在一个命令里增加几个字段或修改几个字段的类型。对于大表，此种操作带来的效率提升更明显，原因在于只需要对该大表做一次处理。
  - 增加一个CHECK或NOT NULL约束将会扫描该表，以保证现有的行符合约束要求。
  - 用一个非空缺省值增加一个字段或者改变一个字段的现有类型会重写整个表。对于大表来说，这个操作可能会花很长时间，并且它还临时需要两倍的磁盘空间。

 版权所有 © 华为云计算技术有限公司

表操作相关的 action 子句

具体表操作action可以是以下子句之一：

```
column_clause
    | ADD table_constraint [ NOT VALID ]
    | ADD table_constraint_using_index
    | VALIDATE CONSTRAINT constraint_name
    | DROP CONSTRAINT [ IF EXISTS ]  constraint_name [ RESTRICT | CASCADE ]
    | CLUSTER ON index_name
    | SET WITHOUT CLUSTER
    | SET ( {storage_parameter = value} [, ... ] )
    | RESET ( storage_parameter [, ... ] )
    | OWNER TO new_owner
    | SET TABLESPACE new_tablespace
    | SET {COMPRESS|NOCOMPRESS}
    | DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } }
    | TO { GROUP groupname | NODE ( nodename [, ... ] ) }
    | ADD NODE ( nodename [, ... ] )
    | DELETE NODE ( nodename [, ... ] )
    | DISABLE TRIGGER [ trigger_name | ALL | USER ]
    | ENABLE TRIGGER [ trigger_name | ALL | USER ]
    | ENABLE REPLICA TRIGGER trigger_name
    | ENABLE ALWAYS TRIGGER trigger_name
    | DISABLE ROW LEVEL SECURITY
    | ENABLE ROW LEVEL SECURITY
    | FORCE ROW LEVEL SECURITY
    | NO FORCE ROW LEVEL SECURITY
    | REFRESH STORAGE
```

- **ADD table_constraint [NOT VALID]**

  给表增加一个新的约束。与NOT VALID选项组合时，这种约束仅在对外键和CHECK类约束有效。 如果约束条件增加了NOT VALID选项后，表中已有记录是否满足初始约束检查会被跳过。

- **ADD table_constraint_using_index**

  根据已有唯一索引为表增加主键约束或唯一约束。

- **VALIDATE CONSTRAINT constraint_name**

  验证一个外键或是一个使用NOT VALID选项创建的检查类约束，通过扫描全表来保证所有记录都符合约束条件。如果约束已标记为有效时，什么操作也不会发生。

- **DROP CONSTRAINT [IF EXISTS] constraint_name [RESTRICT | CASCADE]**

  删除一个表上的约束。

- **CLUSTER ON index_name**

  为将来的CLUSTER操作选择默认索引。实际上并没有重新盘簇化处理该表。

- **SET WITHOUT CLUSTER**

  从表中删除最新使用的CLUSTER索引。这样会影响将来那些没有声明索引的集群操作。

- **SET ({storage_parameter = value} [, ...])**

  修改表的一个或多个存储参数。

- **RESET (storage_parameter [, ...])**

  重置表的一个或多个存储参数。与SET一样，根据参数的不同可能需要重写表才能获得想要的效果。

- **OWNER TO new_owner**

  将表、序列、视图的属主改变成指定的用户。

 版权所有 © 华为云计算技术有限公司

- **SET {COMPRESS|NOCOMPRESS}**

  修改表的压缩特性。表压缩特性的改变只会影响后续批量插入的数据的存储方式，对已有数据的存储毫无影响。也就是说，表压缩特性的修改会导致该表中同时存在着已压缩和未压缩的数据。

- **DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH (column_name [,...]) } }**

  修改表的分布方式，在修改表分布信息的同时会将表数据在物理上按新分布方式重新分布，修改完成后建议对被修改表执行ANALYZE，以便收集全新的统计信息。

  说明

  - 本操作属于重大变更操作，涉及表分布信息的修改以及数据的物理重分布，修改过程中会阻塞业务，修改完成后原有业务的执行计划会发生变化，请按照正规变更流程进行。
  - 本操作属于资源密集操作，针对大表的分布方式修改，建议在计算和存储资源充裕情况下进行，保证整个集群和原表所在表空间有足够的剩余空间能存储一张与原表同等大小且按照新分布方式进行分布的表。

- **TO { GROUP groupname | NODE (nodename [, ...]) }**

  此语法仅在扩展模式（GUC参数enable_cluster_resize为on时）下可用。该模式谨慎打开，主要供内部扩容工具使用，一般用户不应使用该模式。

- **ADD NODE (nodename [, ...])**

  此语法主要供内部扩容工具使用，一般用户不建议使用。

- **DELETE NODE (nodename [, ...])**

  此语法主要供内部缩容工具使用，一般用户不建议使用。

- **DISABLE TRIGGER [trigger_name | ALL | USER]**

  禁用trigger_name所表示的单个触发器，或禁用所有触发器，或仅禁用用户触发器（此选项不包括内部生成的约束触发器，例如，可延迟唯一性和排除约束的约束触发器）。

  说明

  应谨慎使用此功能，因为如果不执行触发器，则无法保证原先期望的约束的完整性。

- **ENABLE TRIGGER [trigger_name | ALL | USER]**

  启用trigger_name所表示的单个触发器，或启用所有触发器，或仅启用用户触发器。

- **ENABLE REPLICA TRIGGER trigger_name**

  触发器触发机制受配置变量session_replication_role的影响，当复制角色为“origin”（默认值）或“local”时，将触发简单启用的触发器。

  配置为ENABLE REPLICA的触发器仅在会话处于“replica”模式时触发。

- **ENABLE ALWAYS TRIGGER trigger_name**

  无论当前复制模式如何，配置为ENABLE ALWAYS的触发器都将触发。

- **DISABLE/ENABLE ROW LEVEL SECURITY**

  开启或关闭表的行访问控制开关。

  当开启行访问控制开关时，如果未在该数据表定义相关行访问控制策略，数据表的行级访问将不受影响；如果关闭表的行访问控制开关，即使定义了行访问控制策略，数据表的行访问也不受影响。详细信息参见**CREATE ROW LEVEL SECURITY POLICY**章节。

 版权所有 © 华为云计算技术有限公司

说明

不支持对HDFS表、外表、临时表开启行访问控制开关。

- **NO FORCE/FORCE ROW LEVEL SECURITY**

  强制开启或关闭表的行访问控制开关。

  默认情况，表所有者不受行访问控制特性影响，但当强制开启表的行访问控制开关时，表的所有者（不包含系统管理员用户）会受影响。系统管理员可以绕过所有的行访问控制策略，不受影响。

- **REFRESH STORAGE**

  根据OBS冷热表storage_policy所定义的规则，将符合条件的本地热分区切换为存储在OBS上的冷分区。

  例如创建OBS冷热表时，设置storage_policy为 'LMT:10'，则在执行该操作时可将10日前无修改的分区切为冷存储，存至OBS中。

列相关的操作 column_clause 子句

其中列相关的操作column_clause可以是以下子句之一：

```
ADD [ COLUMN ] column_name data_type [ compress_mode ] [ COLLATE collation ] [ column_constraint [ ... ] ]
    | MODIFY [ COLUMN ] column_name data_type
    | MODIFY [ COLUMN ] column_name [ CONSTRAINT constraint_name ] NOT NULL [ ENABLE ]
    | MODIFY [ COLUMN ] column_name [ CONSTRAINT constraint_name ] NULL
    | MODIFY [ COLUMN ] column_name DEFAULT default_expr
    | MODIFY [ COLUMN ] column_name ON UPDATE on_update_expr
    | MODIFY [ COLUMN ] column_name COMMENT comment_text
    | DROP [ COLUMN ] [ IF EXISTS ] column_name [ RESTRICT | CASCADE ]
    | ALTER [ COLUMN ] column_name [ SET DATA ] TYPE data_type [ COLLATE collation ] [ USING expression ]
    | ALTER [ COLUMN ] column_name { SET DEFAULT expression | DROP DEFAULT }
    | ALTER [ COLUMN ] column_name { SET | DROP } NOT NULL
    | ALTER [ COLUMN ] column_name SET STATISTICS [PERCENT] integer
    | ADD STATISTICS (( column_1_name, column_2_name [, ...] ))
    | ADD { INDEX | UNIQUE [ INDEX ] } [ index_name ] ( { { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS LAST ] } [, ...] ) [ USING method ] [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ] [ COMMENT 'text' ] LOCAL [ ( { PARTITION index_partition_name } [, ...] ) ] [ WITH ( { storage_parameter = value } [, ...] ) ]
    | ADD { INDEX | UNIQUE [ INDEX ] } [ index_name ] ({ { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS { FIRST | LAST } ] }[, ...] ) [ USING method ] [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ] [ COMMENT 'text' ] [ WITH ( {storage_parameter = value} [, ... ] ) ] [ WHERE predicate ]
    | DROP { INDEX | KEY } index_name
    | CHANGE [ COLUMN ] old_column_name new_column_name data_type [ [ CONSTRAINT constraint_name ] NOT NULL [ ENABLE ] |
        [ CONSTRAINT constraint_name ] NULL | DEFAULT default_expr | COMMENT 'text' ]
    | DELETE STATISTICS (( column_1_name, column_2_name [, ...] ))
    | ALTER [ COLUMN ] column_name SET ( {attribute_option = value} [, ... ] )
    | ALTER [ COLUMN ] column_name RESET ( attribute_option [, ... ] )
    | ALTER [ COLUMN ] column_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
```

- **ADD [COLUMN] column_name data_type [compress_mode] [COLLATE collation] [column_constraint [...]]**

  向表中增加一个新的字段。用ADD COLUMN增加一个字段，所有表中现有行都初始化为该字段的缺省值（如果没有声明DEFAULT子句，值为NULL）。

- **ADD ({ column_name data_type [compress_mode] } [, ...])**

  向表中增加多列。

- **MODIFY [COLUMN] column_name data_type**

  修改表已存在字段的数据类型。需注意：不支持修改分布列数据类型。

版权所有 © 华为云计算技术有限公司

- **MODIFY [COLUMN] column_name [CONSTRAINT constraint_name] NOT NULL [ENABLE]**

  为表的某列添加not null约束，列存表暂不支持。

- **MODIFY [COLUMN] column_name [CONSTRAINT constraint_name] NULL**

  为表的某列移除not null约束。

- **MODIFY [COLUMN] column_name DEFAULT default_expr**

  修改表的default值。

- **MODIFY [COLUMN] column_name ON UPDATE on_update_expr**

  修改表中指定列的on update表达式，该列必须为timestamp类型或者timestamptz类型，当on_update_expr为NULL值时，则为删除ON UPDATE子句。

- **MODIFY [COLUMN] column_name COMMENT comment_text**

  修改表的注释信息。

- **DROP [COLUMN] [IF EXISTS] column_name [RESTRICT | CASCADE]**

  从表中删除一个字段，和这个字段相关的索引和表约束也会被自动删除。如果任何表之外的对象依赖于这个字段，必须声明CASCADE ，比如外键参考、视图等。

  DROP COLUMN命令并不是物理上把字段删除，而只是简单地把它标记为对SQL操作不可见。随后对该表的插入和更新将在该字段存储一个NULL。因此，删除一个字段是很快的，但是它不会立即释放表在磁盘上的空间，因为被删除了的字段占据的空间还没有回收。这些空间将在执行VACUUM时而得到回收。

- **ALTER [COLUMN] column_name [SET DATA] TYPE data_type [COLLATE collation] [USING expression]**

  改变表字段的数据类型，只允许相同大类的类型转换（数值之间，字符串之间，时间之间等）。该字段涉及的索引和简单的表约束将被自动地转换为使用新的字段类型，方法是重新分析最初提供的表达式。

  ALTER TYPE要求重写整个表的特性有时候是一个优点，因为重写的过程消除了表中没用的空间。比如，要想立刻回收被一个已经删除的字段占据的空间，最快的方法是：

  ```
  ALTER TABLE table ALTER COLUMN anycol TYPE anytype;
  ```

  这里的anycol是任何在表中还存在的字段，而anytype是和该字段的原类型一样的类型。这样的结果是在表上没有任何可见的语意的变化，但是这个命令强制重写，这样就删除了不再使用的数据。

  ALTER TYPE的USING选项实际上可以声明涉及该行旧值的任何表达式，即它可以引用除了正在被转换的字段之外其他的字段。这样，就可以用ALTER TYPE语法做非常普遍性的转换。因为这个灵活性，USING表达式并没有作用于该字段的缺省值（如果有的话），结果可能不是缺省表达式要求的常量表达式。这就意味着如果从旧类型到新类型没有隐含或者赋值转换的话，即使存在USING子句，ALTER TYPE也可能无法把缺省值转换成新的类型。在这种情况下，应该用DROP DEFAULT先删除缺省，执行ALTER TYPE，然后使用SET DEFAULT增加一个合适的新缺省值。类似的考虑也适用于涉及该字段的索引和约束。

- **ALTER [COLUMN] column_name { SET DEFAULT expression | DROP DEFAULT }**

  为一个字段设置或者删除缺省值。请注意缺省值只应用于随后的INSERT命令，它们不会修改表中已经存在的行。也可以为视图创建缺省，这个时候它们是在视图的ON INSERT规则应用之前插入到INSERT句中的。

 版权所有 © 华为云计算技术有限公司

- **ALTER [COLUMN] column_name { SET | DROP } NOT NULL**

  修改一个字段是否允许NULL值或者拒绝NULL值。如果表在字段中包含非NULL，则只能使用SET NOT NULL。

- **ALTER [COLUMN] column_name SET STATISTICS [PERCENT] integer**

  为随后的ANALYZE操作设置针对每个字段的统计收集目标。目标的范围可以在0到10000之内设置。设置为-1时表示重新恢复到使用系统缺省的统计目标。

- **ADD { INDEX | UNIQUE [INDEX] } [index_name] ({ { column_name | (expression) } [COLLATE collation] [opclass] [ASC | DESC] [NULLS LAST] } [, ...]) [USING method] [NULLS [NOT] DISTINCT | NULLS IGNORE] [COMMENT 'text'] LOCAL [({ PARTITION index_partition_name } [, ...])] [WITH ({ storage_parameter = value } [, ...])]**

  为表的分区表创建索引，具体参数可参考**CREATE INDEX**。

- **ADD { INDEX | UNIQUE [INDEX] } [index_name] ({ { column_name | (expression) } [COLLATE collation] [opclass] [ASC | DESC] [NULLS { FIRST | LAST }] }[, ...]) [USING method] [NULLS [NOT] DISTINCT | NULLS IGNORE] [COMMENT 'text'] [WITH ({storage_parameter = value} [, ...])] [WHERE predicate]**

  在表上创建索引，具体参数可参考**CREATE INDEX**。

- **DROP { INDEX | KEY } index_name**

  删除一个表上的索引。

- **CHANGE [COLUMN] old_column_name new_column_name data_type [[CONSTRAINT constraint_name] NOT NULL [ENABLE] |**

  [CONSTRAINT constraint_name] NULL | DEFAULT default_expr | COMMENT 'text']

  修改表中列信息，可将旧列名修改成新列名，以及修改列字段信息。

- **{ADD | DELETE} STATISTICS ((column_1_name, column_2_name [, ...]))**

  用于添加和删除多列统计信息声明（不实际进行多列统计信息收集），以便在后续进行全表或全库analyze时进行多列统计信息收集。每组多列统计信息最多支持32列。不支持添加/删除多列统计信息声明的表：系统表、外表。

- **ALTER [COLUMN] column_name SET ({attribute_option = value} [, ...])**

  ALTER [COLUMN] column_name RESET (attribute_option [, ...])

  设置/重置属性选项。

  属性选项定义的参数有：n_distinct、n_distinct_inherited和cstore_cu_sample_ratio。n_distinct设置并固定表的distinct值统计信息，n_distinct_inherited设置并固定继承表的distinct值统计信息，cstore_cu_sample_ratio设置对cstore列存表进行analyze时所选CU的比例。目前禁止SET/RESET n_distinct_inherited参数。

  - n_distinct

    手动设置该列的distinct值统计信息。

    取值范围：-1.0 ~ INT_MAX

    默认值：0，表示不设置。

  - n_distinct_inherited

    手动设置继承表的该列的distinct值统计信息。

    取值范围：-1.0 ~ INT_MAX

版权所有 © 华为云计算技术有限公司

默认值：0，表示不设置。

– cstore_cu_sample_ratio

设置列存表执行analyze，计算需要采样的CU个数时，需要扩大的倍数。

取值范围：1.0 ~ 10000.0

默认值：1.0

- **ALTER [COLUMN] column_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }**

  为一个字段设置存储模式。这个设置控制这个字段是内联保存还是保存在一个附属的表里，以及数据是否要压缩。仅支持对行存表的设置；对列存表没有意义，执行时报错。SET STORAGE本身并不改变表上的任何东西，只是设置将来的表操作时，建议使用的策略。

其他参数的选项

- 列约束column_constraint为：

```
[ CONSTRAINT constraint_name ]
   { NOT NULL |
     NULL |
     CHECK ( expression ) |
     DEFAULT default_expr  |
     UNIQUE [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ] index_parameters |
     PRIMARY KEY index_parameters }
   [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

- 列的压缩可选项compress_mode为：

```
[ DELTA | PREFIX | DICTIONARY | NUMSTR | NOCOMPRESS ]
```

- 根据已有唯一索引为表增加主键约束或唯一约束table_constraint_using_index为：

```
[ CONSTRAINT constraint_name ]
   { UNIQUE | PRIMARY KEY } USING INDEX index_name
   [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

- 增加外键约束REFERENCES为：

```
[ CONSTRAINT constraint_name ]
 FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ] }
```

- 表约束table_constraint为：

```
[ CONSTRAINT constraint_name ]
   { CHECK ( expression ) |
     UNIQUE [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ] ( column_name [, ... ] ) index_parameters |
     PRIMARY KEY ( column_name [, ... ] ) index_parameters }

   [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

- 索引参数index_parameters为：

```
[ WITH ( {storage_parameter = value} [, ... ] ) ]
   [ USING INDEX TABLESPACE tablespace_name ]
```

参数说明

表 21-159 ALTER TABLE 参数说明

参数	描述	取值范围
IF EXISTS	如果表名不存在时返回一个提示而不是错误。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ONLY	仅对指定的表名执行操作。 如果不使用ONLY关键字， 该表及其所有子表都将会被更改。 需注意：只允许在父表或子表中添加或删除列，或更改列的类型。 父表及其后代表必须始终具有相同的列和类型。	-
table_name	需要修改的表名称。 可以在表名称后面显示地增加*选项来指定包括子表，即表示所有后代表都被扫描，这是默认行为	有效的表名。
new_table_name	表的新名称。	字符串，需符合**标识符命名规范**。
column_name， column_1_name， column_2_name	现有的或新字段的名称。	有效的字段名。
new_column_name	指定列的新名称。	字符串，需符合**标识符命名规范**。
constraint_name	约束的名称。	有效的约束名。
new_constraint_name	表约束的新名称。	字符串，需符合**标识符命名规范**。
schema_name	表所在的模式名称。	有效的模式名
new_schema	修改后新的模式名称。	有效的模式名。
index_name	索引名称。	有效的索引名。
new_owner	表的新所有者的名称。	有效的用户名或角色名。
new_tablespace	表所属的新表空间的名称。	有效的表空间名。
data_type	新字段的类型，或者现有字段的新类型。	字符串，需符合**标识符命名规范**。
storage_parameter	表的存储参数。	参见**表21-160**。
compress_mode	表字段的压缩可选项，当前仅对行存表有效。该子句指定该字段优先使用的压缩算法	-
collation	字段排序规则名称。可选字段COLLATE指定了新字段的排序规则，如果省略，排序规则为新字段的默认类型。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
USING expression	USING子句声明如何从旧的字段值里计算新的字段值；如果省略，缺省从旧类型向新类型的赋值转换。如果从旧数据类型到新类型没有隐含或者赋值的转换，则必须提供一个USING子句。	-
NOT NULL | NULL	设置列是否允许空值。	-
integer	带符号的整数常值。当使用PERCENT时表示按照表数据的百分比收集统计信息。	0~100
attribute_option	属性选项。	-
PLAIN | EXTERNAL | EXTENDED | MAIN	字段存储模式。 ● PLAIN必须用于定长的数值（比如integer）并且是内联的、不压缩的。 ● MAIN用于内联、可压缩的数据。 ● EXTERNAL用于外部保存、不压缩的数据。使用EXTERNAL将令在text和bytea字段上的子字符串操作更快，但付出的代价是增加了存储空间。 ● EXTENDED用于外部的压缩数据，EXTENDED是大多数支持非PLAIN存储的数据的缺省。	-
CHECK (expression)	每次将要插入的新行或者将要被更新的行必须使表达式结果为真才能成功，否则会抛出一个异常并且不会修改数据库。 声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。 目前，CHECK表达式不能包含子查询也不能引用除当前行字段之外的变量。	-
DEFAULT default_expr	给字段指定缺省值。 缺省表达式的数据类型必须和字段类型匹配。 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。 default_expr中若使用后缀操作符（如!），需使用括号括起来。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
UNIQUE [NULLS [NOT] DISTINCT | NULLS IGNORE] index_parameters UNIQUE (column_name [, ...]) [NULLS [NOT] DISTINCT | NULLS IGNORE] index_parameters	UNIQUE约束表示表里的一个或多个字段的组合必须在全表范围内唯一。 其中[NULLS [NOT] DISTINCT | NULLS IGNORE]字段用来指定Unique唯一索引中索引列NULL值的处理方式。具体可参考**表21-161**。 默认取值：该参数默认取值为空，即NULL值可重复插入。	-
PRIMARY KEY index_parameters PRIMARY KEY (column_name [, ...]) index_parameters	主键约束表明表中的一个或者一些字段只能包含唯一（不重复）的非NULL值。	-
DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE	设置该约束是否可推迟，列存暂不支持。 • DEFERRABLE：可以推迟到事务结尾使用SET CONSTRAINTS命令检查。 • NOT DEFERRABLE：在每条命令之后马上检查。 • INITIALLY IMMEDIATE：那么每条语句之后就立即检查它。 • INITIALLY DEFERRED：只有在事务结尾才检查它。	-
WITH ({storage_parameter = value} [, ...])	为表或索引指定一个可选的存储参数。	-
COMPRESS| NOCOMPRESS	• NOCOMPRESS：如果指定关键字NOCOMPRESS则不会修改表的现有压缩特性。 • COMPRESS：如果指定COMPRESS关键字，则对该表进行批量插入元组时触发该特性。	-
CASCADE	级联删除依赖于被依赖字段或者约束的对象（比如引用该字段的视图）。	-
RESTRICT	如果字段或者约束还有任何依赖的对象，则拒绝删除该字段。这是缺省行为。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
cache_policy	表缓存策略，仅存算分离3.0版本支持。 默认值为ALL。	• ALL： 对整个表进行热缓存。 • NONE：对整个表进行冷缓存。 • HPN：N分区表中前N个分区会被热缓存，其余分区进行冷缓存。 • HPL：P1, P2, ... 分区表中在缓存策略中被指定名称的分区会被热缓存，其余分区进行冷缓存。 **说明** • 对于外表和非分区内表只支持ALL和NONE两种缓存策略。 • 仅range和list分区内表支持HPN和HPL缓存策略。

表 21-160 storage_parameter 参数说明

storage_parameter 参数	参数项	描述	取值范围
分区管理新增的两个选项	PERIOD	设置分区管理中自动创建分区的周期。 **说明** • 在建表时，如果没有设置该参数，可以通过set的方式添加该参数，并开启自动创建分区功能；如果之前已经设置该参数，则通过set的方式修改该参数。 • 用户可以通过reset该参数的方式关闭自动创建分区功能，但是在自动删除分区功能存在的情况下，不支持关闭自动创建分区功能。	interval类型。 PERIOD的范围要求以及开启该功能的约束请参考**WITH子句参数说明**中的PERIOD参数。

版权所有 © 华为云计算技术有限公司

storage_parameter 参数	参数项	描述	取值范围
	TTL	设置分区管理中自动删除分区的分区过期时间。 **说明** • 在建表时，如果没有设置该参数，可以通过set的方式添加该参数，并开启自动删除分区功能；如果之前已经设置该参数，则通过set的方式修改该参数。 • 用户可以通过reset该参数的方式关闭自动删除分区功能。	interval类型。 TTL的范围要求以及开启该功能的约束请参考**WITH子句参数说明**中的TTL参数。
turbo存储格式列存表新增的选项	enable_turbo_store	设置列存表是否为turbo存储格式。该选项仅9.1.0.100及以上集群版本支持。 **说明** • 3.0普通列存表不支持设置turbo存储格式，仅3.0 hstore_opt表支持turbo存储格式。 • 2.0版本列存表无限制。	-

唯一索引中索引列NULL值的处理方式的说明如下：

[NULLS [NOT] DISTINCT | NULLS IGNORE]字段用来指定Unique唯一索引中索引列NULL值的处理方式。

在对插入的新数据和表中原始数据进行列的等值比较时，对于NULL值有以下三种处理方式：

- NULLS DISTINCT：NULL值互不相等，即NULL值可重复插入。
- NULLS NOT DISTINCT：NULL值相等。若索引列全为NULL，则NULL值不可重复插入；部分索引列为NULL，只有非NULL值不相等，才可成功插入数据。
- NULLS IGNORE：在等值比较时跳过NULL值。若索引列全为NULL，则NULL值可重复插入；部分索引列为NULL，只有非NULL值不相等，才可成功插入数据。

表 21-161 唯一索引中索引列 NULL 值的处理方式

字段控制	索引列全为NULL	部分索引列为NULL
NULLS DISTINCT	可重复插入	可重复插入
NULLS NOT DISTINCT	不可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功

版权所有 © 华为云计算技术有限公司

字段控制	索引列全为NULL	部分索引列为NULL
NULLS IGNORE	可重复插入	非NULL值相等，不可插入；非NULL值不相等，则插入成功

表操作示例

创建示例表customer。

```
DROP TABLE IF EXISTS customer;
CREATE TABLE customer
(
   C_CUSTKEY     BIGINT       ,
   C_NAME        VARCHAR(25)  ,
   C_ADDRESS     VARCHAR(40)  ,
   C_NATIONKEY   INT          ,
   C_PHONE       CHAR(15)     ,
   C_ACCTBAL     DECIMAL(15,2)
)
DISTRIBUTE BY HASH(C_CUSTKEY);
```

创建示例列存表customer_address。

```
DROP TABLE IF EXISTS customer_address;
CREATE TABLE customer_address
(
   ca_address_sk       INTEGER                 NOT NULL   ,
   ca_address_id       CHARACTER(16)           NOT NULL   ,
   ca_street_number    CHARACTER(10)                      ,
   ca_street_name      CHARACTER varying(60)              ,
   ca_street_type      CHARACTER(15)                      ,
   ca_suite_number     CHARACTER(10)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=HIGH,COLVERSION=2.0)
DISTRIBUTE BY HASH (ca_address_sk);
```

- 修改表的模式。

  设置表customer的所属模式为myschema。

  ```
  ALTER TABLE customer SET SCHEMA myschema;
  ```

- 根据已有唯一索引为表增加主键约束或唯一约束。

  为表customer创建唯一索引customer_constraint1，然后根据已有唯一索引增加主键约束，并对前面创建的索引进行重命名。

  ```
  CREATE UNIQUE INDEX customer_constraint1 ON customer(C_CUSTKEY);
  ALTER TABLE customer ADD CONSTRAINT customer_constraint2 PRIMARY KEY USING INDEX
  customer_constraint1;
  ```

- 重命名表约束。

  ```
  ALTER TABLE customer RENAME CONSTRAINT customer_constraint2 TO customer_constraint;
  ```

- 删除表约束。

  ```
  ALTER TABLE customer DROP CONSTRAINT customer_constraint;
  ```

- 给表新增主键约束。

  ```
  ALTER TABLE CUSTOMER ADD PRIMARY KEY(C_CUSTKEY);
  ```

- 向列存表customer_address中添加局部聚簇列。

  ```
  ALTER TABLE customer_address ADD CONSTRAINT customer_address_cluster PARTIAL CLUSTER
  KEY(ca_address_sk);
  ```

- 删除列存表customer_address中的局部聚簇列。

  ```
  ALTER TABLE customer_address DROP CONSTRAINT customer_address_cluster;
  ```

- 修改列存表的存储格式。

  ```
  ALTER TABLE customer_address SET (COLVERSION = 1.0);
  ```


版权所有 © 华为云计算技术有限公司

- 修改表的分布方式。
  ```
  ALTER TABLE customer_address DISTRIBUTE BY REPLICATION;
  ```
- 重命名表。
  ```
  ALTER TABLE CUSTOMER RENAME TO CUSTOMER_t;
  ```
- 修改表缓存策略（仅存算分离3.0版本集群支持）。
  ```
  ALTER TABLE orders SET (cache_policy = 'NONE');
  ```
- 关闭表自动删除分区功能。创建表的示例可参考**示例：创建指定分区的自动分区管理分区表**。
  ```
  ALTER TABLE CPU RESET (ttl);
  ```

索引操作示例

- 给表增加一个索引。
  ```
  ALTER TABLE customer ADD INDEX customer_index(C_CUSTKEY);
  ```
- 删除表索引。
  ```
  ALTER TABLE customer DROP INDEX customer_index;
  ALTER TABLE customer DROP KEY customer_index;
  ```
- 给表中某一列添加索引。
  ```
  ALTER TABLE CUSTOMER ADD c_address_id varchar(20) CONSTRAINT ca_address_index CHECK
  (c_address_id > 0);
  ```

冷热表操作示例

- 单表冷热切换。
  ```
  ALTER TABLE cold_hot_table REFRESH STORAGE;
  ```
- 列存分区表修改为冷热表。
  ```
  CREATE table test_1(id int,d_time date)
  WITH(ORIENTATION=COLUMN)
  DISTRIBUTE BY HASH (id)
  PARTITION BY RANGE (d_time)
  (PARTITION p1 START('2022-01-01') END('2022-01-31') EVERY(interval '1 day'))

  ALTER TABLE test_1 SET (storage_policy = 'LMT:100');
  ```

列操作示例

创建示例表warehouse_t。

```
DROP TABLE IF EXISTS warehouse_t;
CREATE TABLE warehouse_t
(
   W_WAREHOUSE_SK           INTEGER               NOT NULL,
   W_WAREHOUSE_ID           CHAR(16)              NOT NULL,
   W_WAREHOUSE_NAME           VARCHAR(20)   UNIQUE DEFERRABLE,
   W_WAREHOUSE_SQ_FT         INTEGER                     ,
   W_COUNTY                VARCHAR(30)                 ,
   W_STATE                 CHAR(2)           DEFAULT 'GA',
   W_ZIP                   CHAR(10)
);
```

- 增加列。

  向表warehouse_t中增加单列。
  ```
  ALTER TABLE warehouse_t ADD W_GOODS_CATEGORY int;
  ```

  向表warehouse_t中增加多列。
  ```
  ALTER TABLE warehouse_t ADD (W_COMMENT VARCHAR(117) NOT NULL, W_COUNT int);
  ```
- 删除列。

  从表warehouse_t中删除一列。
  ```
  ALTER TABLE warehouse_t DROP COLUMN W_STATE;
  ```

 版权所有 © 华为云计算技术有限公司

- 重命名列。

  表warehouse_t的列名W_ZIP修改为new_W_ZIP。
  ```
  ALTER TABLE warehouse_t RENAME W_ZIP TO new_W_ZIP;
  ```
- 变更列名同时修改列字段信息。

  表warehouse_t的列名W_GOODS_CATEGORY修改为W_GOODS_CATEGORY2，同时修改W_GOODS_CATEGORY2字段数据类型为DECIMAL，字段约束为非空。
  ```
  ALTER TABLE warehouse_t CHANGE W_GOODS_CATEGORY W_GOODS_CATEGORY2 DECIMAL NOT NULL COMMENT 'W_GOODS_CATEGORY';
  ```
- 变更列的注释信息（COMMENT）。
  ```
  ALTER TABLE warehouse_t MODIFY W_GOODS_CATEGORY2 COMMENT 'W_GOODS_CATEGORY';
  ```
- 变更列的数据类型。

  修改表字段warehouse_t的数据类型，并将字段约束设置为非空。
  ```
  ALTER TABLE warehouse_t MODIFY W_WAREHOUSE_SQ_FT varchar(20) NOT NULL;
  ```
- 指定列添加非空约束（not null）。
  ```
  ALTER TABLE warehouse_t ALTER COLUMN W_COUNTY SET NOT NULL;
  ```
- 为表warehouse_t增加主键。
  ```
  ALTER TABLE warehouse_t ADD PRIMARY KEY(w_warehouse_name);
  ```
- 修改表中指定列的on update表达式。

  为warehouse_t表增加一个带有on update的timestamp列，在表执行UPDATE操作时自动更新该列的时间戳。
  ```
  ALTER TABLE warehouse_t ADD COLUMN C_TIME timestamp on update current_timestamp;
  ```

  将warehouse_t表中带有on update的timestamp列删除。
  ```
  ALTER TABLE warehouse_t MODIFY COLUMN C_TIME timestamp on update NULL;
  ```

相关链接

CREATE TABLE，12.101-RENAME TABLE，DROP TABLE

21.13.4.5 ALTER TABLE PARTITION

功能描述

本章节介绍修改表分区的语法功能，包括增删分区、切割分区、合成分区、从普通表数据迁移到分区表，以及修改分区属性等。

逻辑上的一张表根据某种方案分成几张物理块进行存储，这张逻辑上的表称之为**分区表**，物理块称之为**分区**。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。

常见的分区策略都是按照某一列或者某几列定义一些数据分布范围，然后每个分区承载一个范围的数据，这些列称之为**分区键**。

目前DWS行存表、列存表支持的分区类型包括**范围分区**和**列表分区**。

- **范围分区（Range Partitioning）**

  根据表的一列或者多列，将要插入表的记录分为若干个范围，这些范围在不同的分区里没有重叠。为每个范围创建一个分区，用来存储相应的数据。
  - **核心特征**：数据按连续区间分布，常用于有序且按范围检索的场景。
  - **适用场景**：分区键具有线性增长或连续区间特征，数据查询常围绕 “某一范围” 展开，且新数据会自然落入新的区间。例如按时间范围进行分区的订单数据，按数值范围分区的用户等级数据等。

版权所有 © 华为云计算技术有限公司

  - **范围分区策略**是指记录插入分区的方式，根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息，这是最常用的分区策略。目前范围分区仅支持范围分区策略。

- **列表分区（List Partitioning）**

  根据表的一列，将要插入表的记录通过每一个分区中出现的键值划分到对应的分区中，这些键值在不同的分区里没有重叠。为每组键值创建一个分区，用来存储相应的数据。**列表分区仅8.1.3及以上集群版本支持**。

  - **核心特征**：数据按离散的枚举值划分，常用于固定分类或高频过滤的维度。
  - **适用场景**：分区键值是固定且离散的类别，数据查询常围绕 “某一类别” 展开，键值不会无规律新增。例如按省份分区的用户信息表，每个省份对应一个分区。
  - **列表分区策略**是根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息。

注意事项

- 添加分区的名称不能与该分区表已有分区的名称相同。
- 对于范围分区表，要添加的分区的边界值要和分区表的分区键的类型一致，且要大于分区表的最后一个分区的上边界。
- 对于列表分区表，如果已经定义DEFAULT分区，则不能添加新分区。
- 若文档中未特殊注明，则表明范围分区表和列存分区的语法使用相同。
- 如果目标分区表中已有分区数达到了最大值（32767），则不能继续添加分区。
- 当分区表只有一个分区时，不能删除该分区。
- 删除分区（DROP PARTITION）时会连同分区内数据一起删除。
- 选择分区使用PARTITION FOR()，括号里指定值个数应该与定义分区时使用的列个数相同，并且一一对应。
- Value分区表不支持相应的Alter Partition操作。
- OBS冷热表对于move，exchange，merge，split操作，不支持指定分区表的表空间为OBS表空间；执行ALTER语法时，需保持分区数据冷热属性不变（即冷分区操作后为冷分区，热分区操作后为热分区），不支持将冷分区数据切至本地表空间；对于冷分区仅支持默认表空间；merge操作不支持将冷分区与热分区进行合并，exchange操作不支持冷分区交换。

⚠ 警告

- 避免在业务高峰期执行ALTER TABLE/ALTER TABLE PARTITION（增删改查、DROP PARTITION）、TRUNCATE操作，避免有长SQL阻塞ALTER、TRUNCATE操作或SQL业务被ALTER、TRUNCATE阻塞。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

修改表分区主语法：

 版权所有 © 华为云计算技术有限公司

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
    action [, ... ];
```

其中action统指如下分区维护子语法。当存在多个分区维护子句时，保证了分区的连续性，无论这些子句的排序如何，DWS总会先执行DROP PARTITION再执行ADD PARTITION操作，最后顺序执行其它分区维护操作。

- **MODIFY PARTITION子语法：设置分区索引是否可用**
- **REBUILD PARTITION子语法：重建分区索引**
- **EXCHANGE PARTITION（交换分区）子语法：把普通表的数据迁移到指定的分区**
- **ROW MOVEMENT子语法：设置分区表的行迁移开关**
- **MERGE PARTITIONS子语法：把多个分区合并成一个分区**
- **SPLIT PARTITION子语法：把一个分区切割成多个分区**
- **增加分区ADD PARTITION子语法：为指定的分区表添加一个或多个分区**
- **删除分区DROP PARTITION子语法：删除分区表中的指定分区或多个分区**
- **TRUNCATE PARTITION子语法：清理表分区的数据**

ALTER TABLE PARTITION 主句参数说明

表 21-162 ALTER TABLE PARTITION 主句参数说明

参数	描述	取值范围
table_name	需要修改的分区表的名称。	已存在的分区表名。
action [, ...]	action统指分区维护子语法。当存在多个分区维护子句时，保证了分区的连续性，无论这些子句的排序如何，DWS总会先执行DROP PARTITION再执行ADD PARTITION操作，最后顺序执行其它分区维护操作。	参见以下各个子语法使用。
partition_name	需要修改的分区名。	已存在的分区名。
partition_value	分区键值。 通过PARTITION FOR (partition_value [, ...])子句指定的这一组值，可以唯一确定一个分区。	需要进行重命名分区的分区键的取值范围。

MODIFY PARTITION 子语法：设置分区索引是否可用

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
MODIFY PARTITION partition_name { UNUSABLE LOCAL INDEXES | REBUILD UNUSABLE LOCAL INDEXES }
```


版权所有 © 华为云计算技术有限公司

表 21-163 modify_clause 子语法参数说明

参数	描述	取值范围
partition_name	需要修改的分区名。	已存在的分区名。
UNUSABLE LOCAL INDEXES	设置该分区上的所有索引不可用。	-
REBUILD UNUSABLE LOCAL INDEXES	重建该分区上的所有索引。	-

REBUILD PARTITION 子语法：重建分区索引

REBUILD PARTITION是DWS中用于重建分区表索引的操作，主要用于修复因异常操作（如分区数据大量变更、索引损坏等）导致的分区索引不可用状态，或优化分区索引性能。与重建整个表的索引相比，它仅针对指定分区操作，能显著减少资源消耗和执行时间。

典型使用场景:

- **分区索引失效后修复**：当分区执行了大量DML操作（如批量删除、更新）或异常中断后，分区索引可能变为 UNUSABLE 状态，需重建恢复可用性。
- **性能优化**：长期运行的分区表，索引可能产生碎片，重建可优化索引结构，提升查询效率。
- **分区数据迁移后**：通过EXCHANGE PARTITION交换分区数据后，索引可能需要重建以保持一致性。

该语法**仅8.3.0.100及以上集群版本**支持。

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
REBUILD PARTITION partition_name [ WITHOUT UNUSABLE ]
```

表 21-164 rebuild_clause 子语法参数说明

参数	描述	取值范围
WITHOUT UNUSABLE	重建该分区上的索引时，忽略UNUSABLE状态的索引。	-

EXCHANGE PARTITION（交换分区）子语法：把普通表的数据迁移到指定的分区

EXCHANGE PARTITION（交换分区，或称分区交换）是数据仓库中用于高效管理分区表的关键操作。其核心功能是将**分区表中的一个分区与普通表进行元数据层交换，而非物理移动数据**。这意味着数据块的所有权快速切换，避免大规模I/O操作，极大提升性能。

使用场景:

1. **数据批量加载：**将预处理好的数据先加载到临时表，再替换到目标分区，减少业务高峰期对生产表的写入压力。

版权所有 © 华为云计算技术有限公司

2. **历史数据归档：**将不再频繁访问的分区交换到独立表中，便于迁移到低成本存储。
3. **分区数据修复**：将问题分区的数据交换到临时表进行修正，再重新交换回分区表，降低修复期间的锁竞争。
4. **数据迁移：**跨表或跨库迁移特定分区数据时，直接交换避免逐条插入。

语法格式：

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
EXCHANGE PARTITION { ( partition_name ) | FOR ( partition_value [, ...] ) }
    WITH TABLE {[ ONLY ] ordinary_table_name | ordinary_table_name * | ONLY ( ordinary_table_name )}
    [ { WITH | WITHOUT } VALIDATION ] [ VERBOSE ]
```

表 21-165 exchange_clause 子语法参数说明

参数	描述	取值范围
partition_value	分区键值。 通过PARTITION FOR (partition_value [, ...])子句指定的这一组值，可以唯一确定一个分区。	需要进行重命名分区的分区键的取值范围。
ordinary_table_name	进行迁移的普通表的名称。	已存在的普通表名。
{ WITH | WITHOUT } VALIDATION	在进行数据迁移时，是否检查普通表中的数据满足指定分区的分区键范围。默认为WITH。 由于检查比较耗时，特别是当数据量很大的情况下更甚。所以在保证当前普通表中的数据满足分区的分区键范围时，可以加上WITHOUT来指明不进行检查。	● WITH：对于普通表中的数据要检查是否满足分区的分区键范围，如果有数据不满足，则报错。 ● WITHOUT：对于普通表中的数据不检查是否满足分区的分区键范围。
VERBOSE	在VALIDATION是WITH状态时，如果检查出普通表有不满足要交换分区的分区键范围的数据，那么把这些数据插入到正确的分区，如果路由不到任何分区，再报错。 **注意** 只有在VALIDATION是WITH状态时，才可以指定VERBOSE。	-

进行交换的普通表和分区表必须满足如下条件：

- 普通表和分区表的列数目相同，对应列的信息严格一致，包括：列名、列的数据类型、列约束、列的Collation信息、列的存储参数、列的压缩信息、已删除字段的数据类型等。
- 普通表和分区表的表压缩信息严格一致。

版权所有 © 华为云计算技术有限公司

- 普通表和分区表的分布列信息严格一致。
- 普通表和分区表的索引个数相同，且对应索引的信息严格一致。
- 普通表和分区表的表约束个数相同，且对应表约束的信息严格一致。
- 普通表不可以是临时表和unlogged表。
- 普通表和分区表应该在同一个逻辑集群或节点组（NodeGroup）中，如果不在同一个逻辑集群或节点组，将会采用将数据插入对方表内方式来实现交换分区，这样交换分区的时间与表数据量有关，对于数据量非常大的表和分区表，交换分区将会非常耗时。
- 在线扩容重分布场景中，如果普通表和分区表正在重分布，交换分区语句有可能中断正在重分布的普通表或分区表（取决于交换分区和重分布语句是否产生锁冲突），通常重分布的普通表或分区表被中断后会重试2次，但同一个表交换分区执行过于频繁可能导致普通表或分区表多次重试重分布都失败。如果普通表重分布过程被交换分区操作打断，在重试重分布时，数据已经被替换为原分区表中的数据，会重新进行全量重分布。
- 如果行存分区表中最后一个有效字段后的其他字段全部被删除，在不考虑这些删除字段的情况下，分区表与普通表字段信息一致时，分区表和普通表可以进行交换。
- 列存普通表和列存分区表的表级参数colversion必须一致：禁止colversion2.0与colversion1.0执行交换分区操作。

完成交换后，普通表和分区表的数据被置换，同时普通表和分区表的表空间信息被置换。此时，普通表和分区表的统计信息变得不可靠，需要对普通表和分区表重新执行analyze。

ROW MOVEMENT 子语法：设置分区表的行迁移开关

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
{ ENABLE | DISABLE } ROW MOVEMENT
```

 版权所有 © 华为云计算技术有限公司

表 21-166 row_clause 子语法参数说明

参数	描述	取值范围
ENABLE/DISABLE ROW MOVEMENT	行迁移开关，当更新表中某行分区键值时，如果新值属于其他分区，则会根据开关给出报错信息或者将该行数据迁移到对应的分区。 • ENABLE：打开行迁移功能，适用于频繁更新分区键的场景。 • DISABLE：关闭行迁移功能，适用于严格禁止数据跨分区更新的场景，更新分区键会报错。 默认为DISABLE，关闭。 **注意** 开启ROW MOVEMENT则允许跨分区更新，但此时如果有SELECT FOR UPDATE查询该分区表并发执行，存在查询结果瞬时不一致的可能性，需要谨慎使用。	举例，某分区表的日期列（分区键），按季度进行分区，分为分区p_2023q1和p_2023q2，某一行的“2023-02-15”原属于第一季度的分区p_2023q1，当更新该值为“2023-05-15”后，对应的这行数据，会根据ROW MOVEMENT开关是否打开而决定是否行数据迁移： • ENABLE：打开行迁移功能，则该数据迁移到新分区p_2023q2。 • DISABLE：关闭行迁移功能，则系统会报错。 详细示例参见**示例：开启和关闭行迁移功能ROW MOVEMENT**。

MERGE PARTITIONS 子语法：把多个分区合并成一个分区

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
MERGE PARTITIONS { partition_name } [, ...] INTO PARTITION partition_name
```

- INTO关键字前的分区称为源分区，INTO关键字后的分区称为目标分区。
- 源分区个数不能小于2个。
- 源分区名称不能重复。
- 源分区不能存在unusable的索引，否则执行会报错。
- 目标分区名只能跟最后一个源分区的名称相同，或者跟表的所有分区名都不相同。
- 目标分区的边界是所有源分区边界的并集。
- 对于范围分区表，所有的源分区必须是边界连续的分区。
- 对于列表分区，如果源分区中包含DEFAULT分区，那么目标分区的边界也是DEFAULT。

SPLIT PARTITION 子语法：把一个分区切割成多个分区

范围分区的split_clause语法如下：

 版权所有 © 华为云计算技术有限公司

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
SPLIT PARTITION { partition_name | FOR ( partition_value [, ...] ) } { split_point_clause |
no_split_point_clause }
```

- 指定切割点split_point_clause的语法为：
  ```
  AT ( partition_value ) INTO ( PARTITION partition_name , PARTITION partition_name  )
  ```

  注意

  切割点的大小要位于正在被切割分区的分区键范围内，指定切割点的方式只能把一个分区切割成两个新分区。

- 不指定切割点no_split_point_clause的语法为。
  ```
  INTO { ( partition_less_than_item [, ...] ) | ( partition_start_end_item [, ...] ) }
  ```

  注意

  - 不指定切割点的方式，partition_less_than_item指定的第一个新分区的分区键要大于正在被切割的分区的前一个分区（如果存在的话）的分区键，partition_less_than_item指定的最后一个分区的分区键要等于正在被切割分区的分区键大小。
  - 不指定切割点的方式，partition_start_end_item指定的第一个新分区的起始点（如果存在的话）必须等于正在被切割的分区的前一个分区（如果存在的话）的分区键，partition_start_end_item指定的最后一个分区的终止点（如果存在的话）必须等于正在被切割分区的分区键。
  - partition_less_than_item支持的分区键个数最多为4，而partition_start_end_item仅支持1个分区键，其支持的数据类型参见**表21-150**中的“PARTITION BY RANGE(partition_key)”参数。
  - 在同一语句中partition_less_than_item和partition_start_end_item两者不可同时使用；不同split语句之间没有限制。

- 分区项partition_less_than_item的语法为：
  ```
  PARTITION partition_name VALUES LESS THAN ( { partition_value | MAXVALUE }  [, ...] )
  ```

- 分区项partition_start_end_item的语法为，其约束参见**表21-150**中的“partition_start_end_item”参数。
  ```
  PARTITION partition_name {
        {START(partition_value) END (partition_value) EVERY (interval_value)} |
        {START(partition_value) END ({partition_value | MAXVALUE})} |
        {START(partition_value)} |
        {END({partition_value | MAXVALUE})}
  }
  ```

列表分区的split_clause语法如下：

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
SPLIT PARTITION { partition_name | FOR ( partition_value [, ...] ) } { split_values_clause |
split_no_values_clause }
```

- 指定切割点的split_values_clause的语法为：
  ```
  VALUES ( { (partition_value) [, ...] | DEFAULT } ) INTO ( PARTITION partition_name , PARTITION
  partition_name  )
  ```

 版权所有 © 华为云计算技术有限公司

注意

- 如果源分区不是**DEFAULT分区**，那么切割点所指定的边界是源分区边界的一个非空真子集；如果源分区是DEFAULT分区，那么切割点所指定的边界不能和其它非DEFAULT分区的边界存在重叠。
- 切割点的指定的边界是INTO关键字后面的第一个分区的边界，源分区边界与切割点的指定的边界的差集是第二个分区的边界。
- 当源分区是DEFAULT分区时，第二个分区的边界还是DEFAULT。

- 不指定切割点的split_no_values_clause的语法为：

```
INTO ( list_partition_item [, ....], PARTITION partition_name )
```

注意

- 此处的**list_partition_item**和创建列表分区表的时候指定分区的语法一样，除了此处的分区定义中边界值不能为DEFAULT。
- 除了最后一个分区，其他分区需要显式定义边界，定义的边界不能是DEFAULT，并且必须是源分区边界的非空真子集。最后一个分区的边界是源分区边界与其它分区边界的差集，且最后一个分区的边界为空(即差集不能为空集)。
- 如果源分区是DEFAULT分区，则最后一个分区的边界为DEFAULT。

增加分区 ADD PARTITION 子语法：为指定的分区表添加一个或多个分区

范围分区的add_clause语法如下：

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
ADD { partition_less_than_item... | partition_start_end_item }
```

注意

- 使用**partition_less_than_item**语法时，分区表必须是范围分区表，否则执行会报错。
- 此处**partition_less_than_item**和创建范围分区表的时候指定分区的语法一样。
- 当前分区表的最后一个分区的边界为MAXVALUE，不允许添加新的分区，否则执行会报错。

列表分区的add_clause语法如下：

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
ADD list_partition_item
```

注意

- 使用**list_partition_item**语法时，分区表必须是列表分区表，否则执行会报错。
- 此处的**list_partition_item**和创建列表分区表的时候指定分区的语法一样。
- 当前分区表存在DEFAULT分区时，不允许添加新的分区动作，否则执行会报错。

版权所有 © 华为云计算技术有限公司

删除分区 DROP PARTITION 子语法：删除分区表中的指定分区或多个分区

删除指定分区。

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
DROP PARTITION  { partition_name | FOR (  partition_value [, ...] )  }
```

删除多个分区。（8.1.3.100及以上集群版本支持。）

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
DROP PARTITION  { partition_name [, ... ] }
```

TRUNCATE PARTITION 子语法：清理表分区的数据

```
ALTER TABLE [ IF EXISTS ] { table_name  [*] | ONLY table_name | ONLY ( table_name  )}
TRUNCATE PARTITION { partition_name | FOR (  partition_value  [, ...] )  };
```

- partition_value为分区键值。支持指定多个分区键值，多个分区键值以英文逗号分隔。
- 使用PARTITION FOR子句时，partition_value所在的整个分区会被清空。

修改表分区名称语法

```
ALTER TABLE [ IF EXISTS ] { table_name [*] | ONLY table_name | ONLY ( table_name  )}
   RENAME PARTITION { partition_name | FOR ( partition_value [, ...] ) } TO partition_new_name;
```

表 21-167 修改分区名称参数说明

参数	描述	取值范围
partition_new_name	分区的新名称。	字符串，需符合**标识符命名规范**。

示例：创建范围分区表 customer_address

```
DROP TABLE IF EXISTS customer_address;
CREATE TABLE customer_address
(
    ca_address_sk       INTEGER                 NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)              ,
    ca_street_type      CHARACTER(15)                      ,
    ca_suite_number     CHARACTER(10)
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE(ca_address_sk)
(
        PARTITION P1 VALUES LESS THAN(100),
        PARTITION P2 VALUES LESS THAN(200),
        PARTITION P3 VALUES LESS THAN(300)
);
```

示例：创建 list 分区表

```
DROP TABLE IF EXISTS data_list;
CREATE TABLE data_list(
    id int,
    time int,
    salary decimal(12,2)
)PARTITION BY LIST (time)(
        PARTITION P1 VALUES (202209),
        PARTITION P2 VALUES (202210,202208),
```


版权所有 © 华为云计算技术有限公司

```
    PARTITION P3 VALUES (202211),
    PARTITION P4 VALUES (202212),
    PARTITION P5 VALUES (202301)
);
```

示例：使用 MODIFY PARTITION 子句设置分区索引是否可用

给分区表customer_address创建LOCAL索引student_grade_index，并指定分区的索引名称。

```
CREATE INDEX customer_address_index ON customer_address(ca_address_id) LOCAL
(
    PARTITION P1_index,
    PARTITION P2_index,
    PARTITION P3_index
);
```

重建分区表customer_address中分区P1上的所有索引。

```
ALTER TABLE customer_address MODIFY PARTITION P1 REBUILD UNUSABLE LOCAL INDEXES;
```

设置分区表customer_address的分区P3上的所有索引不可用。

```
ALTER TABLE customer_address MODIFY PARTITION P3 UNUSABLE LOCAL INDEXES;
```

示例：使用 REBUILD PARTITION 子语法重建插入过数据的分区索引

步骤1 创建分区表。

```
DROP TABLE IF EXISTS sales;
CREATE TABLE sales (
   id          INT,
   sale_date   DATE,
   amount      DECIMAL(10,2)
)
PARTITION BY RANGE (sale_date)
(
   PARTITION p202310 VALUES LESS THAN ('2023-11-01'),
   PARTITION p202311 VALUES LESS THAN ('2023-12-01')
);
```

步骤2 创建索引。

```
CREATE INDEX idx_sale_date ON sales (sale_date)  LOCAL;
```

步骤3 插入10月、11月分区数据。

```
INSERT INTO sales PARTITION (p202310)
VALUES
(1, '2023-10-05', 100.50),
(2, '2023-10-10', 200.75);
INSERT INTO sales PARTITION (p202311)
VALUES
(3, '2023-11-15', 300.00);
```

步骤4 重建特定分区p202310的索引。

```
ALTER TABLE sales  REBUILD PARTITION p202310  WITHOUT UNUSABLE;
```

----结束

示例：使用 ADD PARTITION 子句为指定的分区表添加一个或多个分区

为范围分区表customer_address增加分区。

```
ALTER TABLE customer_address ADD PARTITION P5 VALUES LESS THAN (500);
```


版权所有 © 华为云计算技术有限公司

为范围分区表customer_address增加分区: [500, 600), [600, 700)。

```
ALTER TABLE customer_address ADD PARTITION p6 START(500) END(700) EVERY(100);
```

为范围分区表customer_address增加MAXVALUE分区p7。

```
ALTER TABLE customer_address ADD PARTITION p7 END(MAXVALUE);
```

为列表分区表增加分区P6。

```
ALTER TABLE data_list ADD PARTITION P6 VALUES (202302,202303);
```

示例：使用 SPLIT PARTITION 子句将一个分区切割成多个分区

将范围分区表customer_address的P7分区以800为分割点切分。

```
ALTER TABLE customer_address SPLIT PARTITION P7 AT(800) INTO (PARTITION P6a,PARTITION P6b);
```

将范围分区表customer_address中400所在的分区分割成多个分区。

```
ALTER TABLE customer_address SPLIT PARTITION FOR(400) INTO(PARTITION p_part START(300) END(500)
EVERY(100));
```

将列表分区表data_list的分区P2分割成p2a和p2b两个分区。

```
ALTER TABLE data_list SPLIT PARTITION P2 VALUES(202210) INTO (PARTITION p2a,PARTITION p2b);
```

示例：使用 EXCHANGE PARTITION（交换分区）子句将普通表的数据迁移到指定分区

如下示例将演示一个普通表math_grade数据迁移到分区表student_grade中分区（math）的操作。

1. 创建分区表student_grade。

```
CREATE TABLE student_grade (
        stu_name     char(5),
        stu_no       integer,
        grade        integer,
        subject      varchar(30)
)
PARTITION BY LIST(subject)
(
        PARTITION gym VALUES('gymnastics'),
        PARTITION phys VALUES('physics'),
        PARTITION history VALUES('history'),
        PARTITION math VALUES('math')
);
```

2. 插入数据到分区表student_grade中。

```
INSERT INTO student_grade VALUES
        ('Ann', 20220101, 75, 'gymnastics'),
        ('Jeck', 20220103, 60, 'math'),
        ('Anna', 20220108, 56, 'history'),
        ('Jann', 20220107, 82, 'physics'),
        ('Molly', 20220104, 91, 'physics'),
        ('Sam', 20220105, 72, 'math');
```

3. 查询分区表student_grade的math分区记录。

```
SELECT * FROM student_grade PARTITION (math);
```

查询结果如下：

```
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jeck     | 20220103 |    60 | math
 Sam      | 20220105 |    72 | math
(2 rows)
```


版权所有 © 华为云计算技术有限公司

4. 创建一个与分区表student_grade定义匹配的普通表math_grade。

```
CREATE TABLE math_grade
(
    stu_name    char(5),
    stu_no      integer,
    grade       integer,
    subject     varchar(30)
);
```

5. 插入数据到表math_grade中。数据与分区表student_grade的math分区的分区规则一致。

```
INSERT INTO math_grade VALUES
    ('Ann', 20220101, 75, 'math'),
    ('Jeck', 20220103, 60, 'math'),
    ('Anna', 20220108, 56, 'math'),
    ('Jann', 20220107, 82, 'math');
```

6. 将普通表math_grade数据迁移到分区表student_grade的分区（math）。

```
ALTER TABLE student_grade EXCHANGE PARTITION (math) WITH TABLE math_grade;
```

7. 查询分区表student_grade，结果显示表math_grade中的数据已和分区表student_grade的分区math中的数据交换。

```
SELECT * FROM student_grade PARTITION (math);
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Anna     | 20220108 |    56 | math
 Jeck     | 20220103 |    60 | math
 Ann      | 20220101 |    75 | math
 Jann     | 20220107 |    82 | math
(4 rows)
```

8. 查询表math_grade，显示之前存储在分区表student_grade的分区math中的数据已交换到表math_grade中。

```
SELECT * FROM math_grade;
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jeck     | 20220103 |    60 | math
 Sam      | 20220105 |    72 | math
(2 rows)
```

示例：使用 TRUNCATE PARTITION 子语法清理表分区的数据

清空表customer_address分区p1：

```
ALTER TABLE customer_address TRUNCATE PARTITION p1;
```

示例：开启和关闭行迁移功能 ROW MOVEMENT

在更新某行的分区键为新值且新值属于其他分区时，如果打开ROW MOVEMENT，则该行数据对应迁移到新的分区中。

频繁更新分区键时，ENABLE ROW MOVEMENT 可能导致额外I/O开销，需评估业务需求。

创建按日期范围分区的表，并启用行迁移。

```
DROP TABLE IF EXISTS sale_data;
CREATE TABLE  sales_data (
   sale_id  INT,
   product_name VARCHAR(100),
   sale_date DATE,
   amount DECIMAL(10, 2)
)
PARTITION BY RANGE (sale_date) (
   PARTITION p_2023q1 VALUES LESS THAN ('2023-04-01'),
```

 版权所有 © 华为云计算技术有限公司

```
    PARTITION p_2023q2 VALUES LESS THAN ('2023-07-01'),
    PARTITION p_2023q3 VALUES LESS THAN ('2023-10-01'),
    PARTITION p_2023q4 VALUES LESS THAN ('2024-01-01')
)
ENABLE ROW MOVEMENT;
```

插入测试数据。

```
INSERT INTO sales_data (sale_id, product_name, sale_date, amount)
VALUES
(1, 'Product A', '2023-02-15', 1000.00),  -- 属于 p_2023q1
(2, 'Product B', '2023-05-20', 1500.00);  -- 属于 p_2023q2
```

查询各分区数据分布。

```
SELECT tableoid::regclass AS partition, * FROM sales_data ORDER BY sale_id;
```

```
 partition  | sale_id | product_name |      sale_date      | amount
------------+---------+--------------+---------------------+---------
 2147617615 |       1 | Product A    | 2023-02-15 00:00:00 | 1000.00
 2147617616 |       2 | Product B    | 2023-05-20 00:00:00 | 1500.00
(2 rows)
```

尝试更新分区键值（从 p_2023q1 移动到 p_2023q2）。

```
UPDATE sales_data SET sale_date = '2023-05-01'  WHERE sale_id = 1;
```

再次查询各分区数据分布，确认行已移动。

```
SELECT tableoid::regclass AS partition, *  FROM sales_data  ORDER BY sale_id;
```

```
 partition  | sale_id | product_name |      sale_date      | amount
------------+---------+--------------+---------------------+---------
 2147617616 |       1 | Product A    | 2023-05-01 00:00:00 | 1000.00
 2147617616 |       2 | Product B    | 2023-05-20 00:00:00 | 1500.00
(2 rows)
```

关闭行迁移功能。

```
ALTER TABLE sales_data DISABLE ROW MOVEMENT;
```

再次修改分区键（从 p_2023q2 移动回到 p_2023q2）。

```
UPDATE sales_data SET sale_date = '2023-02-15'  WHERE sale_id = 1;
```

回显提示，行迁移关闭，不允许更新分区键。

```
gaussdb=> UPDATE sales_data SET sale_date = '2023-02-15'  WHERE sale_id = 1;
ERROR:  dn_6003_6004: fail to update partitioned table "sales_data"
DETAIL:  disable row movement
```

示例：使用 MERGE PARTITIONS 子句把多个分区合并成一个分区

将范围分区表customer_address的P2，P3两个分区合并为一个分区。

```
ALTER TABLE customer_address MERGE PARTITIONS P2, P3 INTO PARTITION P_M;
```

示例：使用 DROP PARTITION 子句删除分区表中的指定分区

删除分区表customer_address的多个分区P6a，P6b。

 版权所有 © 华为云计算技术有限公司

```
ALTER TABLE customer_address DROP PARTITION P6a, P6b;
```

相关链接

CREATE TABLE PARTITION，**DROP TABLE**

21.13.4.6 RENAME TABLE

功能描述

重命名指定表。

RENAME TABLE命令与如下命令的作用相同：

```
ALTER TABLE table_name RENAME to new_table_name
```

注意事项

不支持跨schema对表进行重命名操作。

语法格式

```
RENAME TABLE
{[schema.]table_name TO new_table_name} [, ...];
```

参数说明

表 21-168 RENAME TABLE 参数说明

参数	描述	取值范围
schema	模式名称。	有效的模式名。
table_name	需要修改的表名称。	已存在的表名。
new_table_name	修改后新的表名称。	字符串，需符合**标识符命名规范**。

示例

创建列存表指定存储格式和压缩方式。

```
DROP TABLE IF EXISTS customer_address;
CREATE TABLE customer_address
(
    ca_address_sk       INTEGER                 NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)              ,
    ca_street_type      CHARACTER(15)                      ,
    ca_suite_number     CHARACTER(10)
)
WITH (ORIENTATION = COLUMN, COMPRESSION=HIGH,COLVERSION=2.0)
DISTRIBUTE BY HASH (ca_address_sk);
```

修改表名。

版权所有 © 华为云计算技术有限公司

```
RENAME TABLE customer_address TO new_customer_address;
```

相关链接

CREATE TABLE，**ALTER TABLE**，**DROP TABLE**

21.13.4.7 DROP TABLE

功能描述

删除指定的表。

注意事项

- DROP TABLE会强制删除指定的表，删除表后，依赖该表的索引会被删除，而使用到该表的函数和存储过程将无法执行。删除分区表，会同时删除分区表中的所有分区。
- 只有表的所有者、模式所有者或者被授予了表的DROP权限的用户才能执行DROP TABLE，系统管理员默认拥有该权限。要清空指定表中的行但是不删除该表定义，可以使用TRUNCATE或者DELETE。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
DROP TABLE [ IF EXISTS ]
   { [schema.]table_name } [, ...] [ CASCADE | RESTRICT ] ;
```

参数说明

表 21-169 DROP TABLE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的表不存在，则发出一个notice而不是抛出一个错误。	-
schema	要删除的表所属的模式名称。	-
table_name	要删除的表名称。	已存在的表名称。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖于表的对象（比如视图）。 • RESTRICT（缺省项）：如果存在依赖对象，则拒绝删除该表。这个是缺省。
PURGE	指定Purge参数时，无论回收站特性是否开启，直接删除，不放入回收站。该参数仅9.1.0.200及以上集群版本支持。	-
KEEP ttl HOUR	对于V3表，回收站特性开启时，设置TTL，物理文件会放入回收站中，后续可以从回收站中恢复，TTL到期之后，后台自动清理回收站中的对象。该参数仅9.1.0.200及以上集群版本支持。 **须知** 没有做合理的空间管控，建议在DROP和TRUNCATE频率低业务场景中使用。	-

示例

删除表warehouse_t1。

```
DROP TABLE tpcds.warehouse_t1;
```

相关链接

ALTER TABLE，**12.101-RENAME TABLE**，**CREATE TABLE**

21.13.4.8 CREATE FOREIGN TABLE (SQL on OBS or Hadoop)

功能描述

在当前数据库创建一个HDFS或OBS外表，用来访问存储在HDFS或者OBS分布式集群文件系统上的结构化数据，也可以通过HDFS或OBS外表导出ORC和PARQUET格式数据到HDFS或者OBS上。

外表（Foreign Table）是数据库或大数据平台中一种虚拟表结构，它不实际存储数据，而是通过元数据信息（如表结构、存储位置、文件格式等）将外部存储系统（如HDFS、OBS等）中的数据映射到数据库内部，使用户能够以标准SQL操作直接访问或操作外部数据。外表的本质是一种“数据代理”，将外部文件系统的数据逻辑映射成数据库可识别的表结构。

外表的核心价值是 “打破数据孤岛”： 通过元数据映射，让数据库用户以熟悉的方式访问外部系统数据，同时支持数据向外部系统的导出，实现跨系统数据的高效协同、分析与共享，尤其在大数据场景中能显著降低数据移动成本、提升协作效率。

版权所有 © 华为云计算技术有限公司

DWS通过外表访问外部数据源时，对于不同的数据源，访问性能有所不同：

- 数据存储在OBS：数据存储和计算分离，集群存储成本低，存储量不受限制，并且集群可以随时删除，但计算性能取决于OBS访问性能，相对HDFS有所下降，建议在数据计算不频繁场景下使用。
- 数据存储在HDFS：数据存储和计算不分离，集群成本较高，计算性能高，但存储量受磁盘空间限制，删除集群前需将数据导出保存，建议在数据计算频繁场景下使用。

说明

以下描述的读写外表，仅存算分离3.0版本支持。

注意事项

- 使用OBS外表访问OBS桶数据时，应确保DWS与OBS桶在同一个区域(Region)下。
- 仅系统管理员dbadmin角色或已授权了USEFT权限的普通用户可执行外表操作。
  ```
  ALTER USER user_name USEFT;
  ```
- HDFS外表与OBS外表分为只读外表、只写外表和读写外表，只读外表用于查询操作，只写外表可以将DWS中的数据导出到分布式文件系统中，读写外表可以查询和数据导出，其中**读写外表仅9.1.0.100及以上版本支持**。
- 格式支持：
  - **导入查询**支持ORC、TEXT、CSV、CARBONDATA、PARQUET和JSON格式。
  - **导出数据**中，OBS外表支持ORC、PARQUET（仅存算分离3.0版本支持）格式，HDFS外表支持ORC、PARQUET格式。
- 该方式使用外表前需要用户先手动创建外部服务器**外部服务器**，**外部服务器**是数据库中配置的一个虚拟对象，用于定义连接到外部数据源的参数（如地址、协议、认证信息等）。通过外部服务器，本地数据库可以直接查询、插入外部数据，就像操作本地表一样（语法类似，但可能受性能或功能限制）。具体请参见**CREATE SERVER**。
- 若手动创建外部服务器Server时指定foreign data wrapper为HDFS_FDW或者DFS_FDW，创建只读外表时需使用DISTRIBUTE BY子句指定分布方式。

表 21-170 OBS 外表支持读写格式说明

数据类型	DFS_FDW/HDFS_FDW		
-	READ ONLY	WRITE ONLY	READ WRITE
ORC	√	√	√
PARQUET	√	√	√
CARBONDATA	√	×	×
TEXT	√	×	×
CSV	√	×	×
JSON	√	×	×

版权所有 © 华为云计算技术有限公司

语法格式

创建外表。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER server_name
   OPTIONS ( { option_name ' value ' } [, ...] )
   [ {WRITE ONLY | READ ONLY | READ WRITE} ]
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}

   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

- 其中column_constraint为：
  ```
  [CONSTRAINT constraint_name]
  {PRIMARY KEY | UNIQUE}
  [NOT ENFORCED [ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION] | ENFORCED]
  ```
- 其中table_constraint为：
  ```
  [CONSTRAINT constraint_name]
  {PRIMARY KEY | UNIQUE} (column_name)
  [NOT ENFORCED [ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION] | ENFORCED]
  ```

参数说明

表 21-171 CREATE FOREIGN TABLE (SQL on OBS or Hadoop)参数说明

参数	描述	取值范围
IF NOT EXISTS	如果已经存在相同名称的表，系统不会报错，而会发出一个通知，告知表关系已存在。	-
table_name	外表的表名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
column_name	外表中的字段名。可以选择多个字段名，中间用“,”隔开。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。 JSON对象由嵌套或并列的name-value对组成，具有顺序无关性，当导入JSON格式数据时，需要通过字段名与name的自动对应来确定字段与value的对应关系。用户需要定义恰当的字段名，否则可能导致导入结果不符合预期。字段名与name的自动对应规则如下： • 无嵌套无数组的情况下，字段名应当与name一致，不区分大小写。 • 字段名使用‘_’字符拼接两个name，标识嵌套关系。 • 字段名使用‘#’字符加十进制非负整数‘n’标识数组的第n个元素（从0开始）。 例如，要导入JSON对象{"A" : "simple", "B" : {"C" : "nesting"}, "D" : ["array", 2, {"E" : "complicated"}]}中的每个元素，外表字段名应当分别定义为a、b、b_c、d、d#0、d#1、d#2、d#2_e，字段的定义顺序不会影响导入结果的正确性。
type_name	字段的数据类型。	-
constraint_name	外表的表约束名。	-
{ NULL | NOT NULL }	标识此列是否允许NULL值。 在创建表时，对于列的约束NULL/NOT NULL，并不能保证该表在HDFS系统中的数据为NULL或者NOT NULL，数据的一致性由用户保证。所以需要由用户判断该列是否一定不为空或者一定为空，在建表的时候选用NULL或NOT NULL。(优化器对列为NULL/NOT NULL做了优化处理，会产生更优的计划。)	-
SERVER server_name	外表的server名字。允许用户自定义名字。	字符串，要符合标识符的命名规范，并且这个server必须存在。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
OPTIONS	用于指定外表数据的各类参数。	不同场景下，需要指定的OPTIONS参数有所不同，说明如下。 • 数据格式参数：参见**表21-172**。 • 容错性参数：参见**表21-173**。 • 仅Server类型为DLI时需要指定的参数：参见**表21-174**。 • 仅format为hudi时需要指定的参数：参见**表 OPTIONS参数说明（仅format为hudi时设置）**。 • OBS外表，各数据源格式支持的OPTIONS参数有所不同，具体参见**表21-176**。 • HDFS外表，各数据源格式支持的OPTIONS参数有所不同，具体参见**表21-177**。
WRITE ONLY | READ ONLY | READ WRITE	指定创建的外表类型。	• WRITE ONLY指定创建HDFS/OBS的只写外表。 • READ ONLY指定创建HDFS/OBS的只读外表。 • READ WRITE指定创建HDFS/OBS的读写外表。 如果不指定创建的外表的类型，**默认为只读外表**。
DISTRIBUTE BY ROUNDROBIN	指定HDFS/OBS外表为ROUNDROBIN分布方式。表的每一行被轮番地发送给各个数据节点（DN），因此数据会被均匀地分布在各个DN中。	-
DISTRIBUTE BY REPLICATION	指定HDFS外表为REPLICATION分布方式。表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PARTITION BY (column_name) AUTOMAPPED	column_name指定分区列。对于分区表，AUTOMAPPED表示HDFS分区外表指定的分区列会和HDFS数据中的分区目录信息自动对应，前提是必须保证HDFS分区外表指定分区列的顺序和HDFS数据中分区目录定义的顺序一致，该功能只适用于只读外表，只写外表不支持。	• 只写分区外表最多支持四级分区（即最多指定四个列为分区列）。且至少需要保留一列为非分区列。 • HDFS只读外表支持text，csv，carbondata，orc，parquet格式分区表。 • HDFS只写外表支持orc，parquet格式分区表。 • HDFS读写外表支持orc，parquet格式分区表。 • OBS只读外表/只写外表/读写外表均支持orc，parquet格式分区表。 • 不支持浮点类型和布尔类型的列作为分区列。 • 分区字段长度限制可通过guc参数dfs_partition_directory_length调整。 • 分区目录名称由 "分区列名=分区列值" 组成。此名称内如果包含特殊字符还会经过转义，所以转义前推荐长度不要超过 (dfs_partition_directory_length + 1) / 3 ，以此保证其转义后总长度不会因为超过dfs_partition_directory_length而出现报错。 • 不推荐包含过长中文字符的列作为分区列。因为一个中文字符和一个英文字符所占的空间大小并不一致，用户不易计算最终的分区目录名称长度，更容易触发超过dfs_partition_directory_length长度限制的报错。
CONSTRAINT constraint_name	用于指定外表所建立的信息约束（Informational Constraint）的名字。	字符串，要符合标识符的命名规范。
PRIMARY KEY	主键约束，表示表里的一个或者一些字段只能包含唯一（不重复）的非NULL值。一个表只能声明一个主键。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
UNIQUE	唯一约束，表示表里的一个或者多个字段的组合必须在全表范围内唯一。对于唯一约束，NULL被认为是互相不等的。	-
NOT ENFORCED	指定所建立的约束为信息约束，即数据库不会强制验证该约束，这个选项与后续的ENABLE QUERY OPTIMIZATION搭配使用，主要有以下作用： • **仅为信息约束**：通过 PRIMARY KEY 或 UNIQUE 声明列的唯一性约束，帮助优化查询执行计划。 • **不强制验证**：数据库不会检查外部数据源中的数据是否实际满足约束条件（如是否存在重复值）。 • **性能优化**：允许查询优化器利用约束信息生成更高效的执行计划，但需用户自行确保数据符合约束，否则可能导致查询结果不准确。 • **风险**：若实际数据违反约束（如存在重复值），可能导致查询结果错误（如聚合结果异常、连接产生多余记录）。 更多信息参见**信息约束（Informational Constraint）**。	例如声明primary key，但是不强制验证。 CREATE FOREIGN TABLE hdfs_users (user_id INT PRIMARY KEY NOT ENFORCED, ...
ENFORCED	数据库强制验证约束，预留参数，目前DWS不支持设置为ENFORCED。	-
PRIMARY KEY (column_name)	指定所建立的信息约束位于column_name列上。	字符串，要符合标识符的命名规范，并且这个column_name必须存在。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ENABLE QUERY OPTIMIZATION	允许查询优化器利用信息约束生成更高效的执行计划，与前面的NOT ENFORCED搭配使用。	创建外表，为字段名p_partkey int增加唯一约束，但是不强制验证该约束，且允许优化器利用该信息约束生成更优执行计划： CREATE FOREIGN TABLE ft_part (p_partkey int UNIQUE NOT ENFORCED ENABLE QUERY OPTIMIZATION, ...
DISABLE QUERY OPTIMIZATION	禁止查询优化器利用信息约束生成高效的执行计划。	-

表 21-172 OPTIONS 数据格式参数说明

参数	描述	取值范围
header	指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV格式的文件中。	• true/on，则数据文件第一行会被识别为标题行，导出时会忽略此行。 • false/off，则数据文件中第一行会被识别为数据。 默认为false/off。
quote	CSV格式文件下的引号字符，缺省值为双引号。	quote参数不能和分隔符、null参数相同。 quote参数只能是单字节的字符。 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。
escape	CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。 在数据库中，逃逸字符（Escape Character）是一种特殊字符，用于转义其他字符，它的主要作用是让数据库系统将原本具有特殊含义的字符当作普通字符处理，让数据库识别并正确处理那些可能与SQL语法冲突的字符（如引号、特殊符号等）。常见的逃逸字符有反斜杠\或双引号''。 缺省值为双引号。当与quote值相同时，会被替换为'\0'。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
location	OBS外表参数，指定存储在OBS上的文件路径，多个桶的数据源数据之间使用分隔符‘|’进行分割，例如：LOCATION 'obs://bucket1/folder/ | obs://bucket2/'，数据库将会扫描指定路径文件夹下面的所有对象。 当访问DLI多版本表时，无需指定location参数。	-
format	外表中数据源文件的格式。	HDFS外表READ ONLY外表支持ORC、TEXT、JSON、CSV、PARQUET、HUDI文件格式，WRITE ONLY/READ WRITE外表支持ORC和PARQUET文件格式。 OBS外表READ ONLY外表支持ORC、TEXT、JSON、CSV、CARBONDATA、PARQUET、HUDI文件格式，WRITE ONLY外表支持ORC和PARQUET文件格式。 **注意** • 对于JSON格式数据，仅支持JSON对象（object，最外层由{}构造）导入，不支持JSON数组（array，最外层由[]构造）导入，但支持JSON对象内部数组的导入。 • Hudi即Apache Hudi，是一个事务性数据湖平台。当前版本支持将OBS外表的format参数指定为Hudi来访问相应的Hudi表数据集，并提供访问Hudi表增量/实时视图的方法。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
foldername	外表中数据源文件目录，即表数据目录在HDFS文件系统和OBS上对应的文件目录。此选项对WRITE ONLY和READ WRITE可写外表为必选项，对READ ONLY外表为可选项。 当访问DLI多版本表时，无需指定foldername参数。 ● OBS外表，格式为：'/OBS桶名/文件夹名/'。应确保OBS桶与DWS集群在同一个区域，不支持跨区访问OBS桶数据。 ● HDFS外表，格式为远端HDFS中的数据文件所在目录，如MRS集群，可登录MRS控制台，进入集群基本信息页面，单击“文件管理”，选择“HDFS文件列表”，找到数据存储的目录。	OBS外表，如下举例： OPTIONS (foldername '/obs-demo01/obs-dws/', format 'parquet', encoding 'utf8') HDFS外表：如下举例： OPTIONS (foldername '/user/hive/warehouse/demo.db/product_info_orc/', format 'orc', encoding 'utf8')
encoding	外表中数据源文件的编码格式名称，缺省为utf8。此选项为可选参数。	-
totalrows	可选参数，估计表的行数，仅OBS外表使用。由于OBS上文件可能很多，做analyze可能会很慢，通过此参数让用户设置一个预估的值，使优化器能通过这个值做大小表的估计。一般预估值和实际值相近时，查询效率较高。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
filenames	外表中数据源文件，以","间隔。	• 推荐通过使用foldername参数指定数据源的位置，对于只读外表filenames参数与foldername参数两者必有其一，而只写外表和读写外表只能通过foldername指定。 • foldername为绝对目录时，前后必须有'/', 多个路径用', '分隔。 • 查询分区表时，会先根据分区信息进行剪枝，然后查询满足条件的数据文件。由于剪枝操作会涉及多次扫描HDFS分区目录内容，不建议使用重复度非常小的列作为分区列，因为这可能导致分区目录非常的多，增加对HDFS的查询压力。 • OBS只读外表和读写外表不支持。
delimiter	指定数据文件行数据的字段分隔符，不指定则使用默认分隔符，TEXT格式的默认分隔符是水平制表符（tab）。	支持多字符分隔符，但分隔符不能超过10个字节。 • 分隔符不能是\r和\n。 • 分隔符不能和null参数相同。 • 分隔符不能包含“\”、“.”、数字和字母。 • 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。 • 分隔符推荐使用多字符（例如'$^&'）和不可见字符（例如0x07、0x08、0x1b等）。 • delimiter参数只在TEXT和CSV格式下有效。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
eol	指定导入数据文件换行符样式。	支持多字符换行符，但换行符不能超过10个字节。常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。 • eol参数只能用于TEXT格式的导入。 • eol参数不能和分隔符、null参数相同。 • eol参数不能包含：数字，字母和符号“.”。
null	用来指定数据文件中空值的表示。	在TEXT格式下缺省值是\N。 • null值不能是\r和\n，最大为100个字符。 • null值不能是分隔符。 • null参数只在TEXT和CSV格式下有效。
noescaping	TEXT格式下，不对'\'和后面的字符进行转义。	该参数只在TEXT格式下有效。true/on，false/off。缺省值为false/off。
date_format	导入对于DATE类型指定格式。此语法仅对READ ONLY的外表有效。	合法DATE格式。可参考**时间、日期处理函数和操作符**。 • 对于指定为ORACLE兼容类型的数据库，则DATE类型内建为TIMESTAMP类型。在导入的时候，若需指定格式，可以参考下面的timestamp_format参数。 • date_format参数只在TEXT和CSV格式下有效。
time_format	导入对于TIME类型指定格式。此语法仅对READ ONLY的外表有效。	该参数只在TEXT和CSV格式下有效。 取值范围：合法TIME格式，不支持时区。可参考**时间、日期处理函数和操作符**。
timestamp_format	导入对于TIMESTAMP类型指定格式。此语法仅对READ ONLY的外表有效。	该参数只在TEXT和CSV格式下有效。 取值范围：合法TIMESTAMP格式，不支持时区。可参考**时间、日期处理函数和操作符**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
smalldatetime_format	导入对于SMALLDATETIME类型指定格式。此语法仅对READ ONLY的外表有效。	该参数只在TEXT和CSV格式下有效。 取值范围：合法SMALLDATETIME格式。可参考**时间、日期处理函数和操作符**。
dataencoding	在数据库编码与数据表的数据编码不一致时，该参数用于指定导出数据表的数据编码。比如数据库编码为Latin-1，而导出的数据表中的数据为UTF-8编码。此选项为可选项，如果不指定该选项，默认采用数据库编码。此语法仅对HDFS的WRITE ONLY/READ WRITE外表有效。	该参数只对ORC格式的WRITE ONLY/READ WRITE的HDFS外表有效。 取值范围：该数据库编码支持转换的数据编码。
filesize	指定WRITE ONLY外表的文件大小。此选项为可选项，不指定该选项默认分布式文件系统配置中文件大小的配置值。此语法仅对WRITE ONLY/READ WRITE可写外表有效。	该参数只对ORC格式的WRITE ONLY/READ WRITE的HDFS外表有效。 取值范围：[1, 1024]的整数。
compression	指定文件的压缩方式，此选项为可选项，该参数仅8.2.0及以上集群版本支持。 对WRITE ONLY/READ WRITE的外表：指定ORC格式文件的压缩方式。 对READ ONLY的外表：指定TEXT、CSV或JSON格式文件的压缩方式。	• 当format为ORC时：zlib，snappy，lz4，缺省值为zlib。 • 当format为PARQUET时：zlib，snappy，lz4，lz4_hadoop，缺省值为snappy。 • 当format为TEXT、CSV或JSON时：gzip。 **注意** • 8.2.0以下版本不支持READ ONLY外表指定compression参数。 • lz4_hadoop压缩选项，该选项适配hive的lz4，当parquet外表设置该压缩格式时，可以与hive端相互读取。
version	指定ORC格式的版本号，此选项为可选项。此语法仅对WRITE ONLY/READ WRITE的外表有效。	取值范围：目前仅支持0.12。缺省值为0.12。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
julian_adjust	指定是否矫正数据中的儒略日，此选项为可选项，该参数仅8.3.0及以上集群版本支持。 spark2.X版本使用了julian日，3.0版本后使用Proleptic Gregorian公历，导致了parquet 1582年前的部分日期有差别，根据实际情况选择是否需要矫正。	取值范围：true，false。缺省值为true。
cache_policy	指定外表磁盘缓存(disk cache)策略。该参数仅存算分离3.0版本支持。	取值范围：外表只支持ALL和NONE两种缓存策略，ALL表示使用disk cache中的热缓存，NONE则使用冷缓存。热缓存相比冷缓存占用的空间更大，技术上使用更加复杂的替换算法。

表 21-173 OPTIONS 容错性参数说明

参数	描述	取值范围
fill_missing_fields	当数据加载时，若数据源文件中一行的最后一个字段缺失时的处理方式。	true/on，false/off。缺省值为false/off。 ● 参数为true/on，当数据加载时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 ● 参数为false/off，如果最后一个字段缺失会显示如下错误信息。 `missing data for column "tt"` **注意** ● TEXT格式下执行SELECT COUNT(*)不会去解析具体字段，因此不会对字段缺失情况报错。 ● fill_missing_fields参数只在TEXT和CSV格式下有效。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ignore_extra_data	若数据源文件比外表定义列数多，是否会忽略多出的列。该参数只在数据导入过程中使用。	true/on，false/off。缺省值为false/off。 ● 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。 ● 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。 extra data after last expected column **注意** ● 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。 ● TEXT格式下执行SELECT COUNT(*)不会去解析具体字段，因此不会对多余的情况报错。 ● ignore_extra_data参数只在TEXT和CSV格式下有效。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
checkencoding	是否检查字符编码。	取值范围：no、low、high 。缺省值为low。 TEXT格式下，导入非法字符容错规则如下： • 对于'\0'，容错后转换为空格； • 对于其他非法字符，容错后转换为问号； • 若checkencoding为low标识，导入时对于非法字符进行容错处理，则若NULL、DELIMITER设置为空格或问号则会通过如"illegal chars conversion may confuse null 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。 ORC格式下，导入非法字符容错规则如下： • checkencoding为no标识，导入时不检查非法字符。在用户确认无非法字符的情况下，此选项可以缩短导入时间。 • checkencoding为low标识，若导入时检查到某个字段中包含非法字符，则自动将当前列当前行的字段整体替换为同样长度的‘？’字符； • checkencoding为high标识，若导入时检查到某个字段中包含非法字符，则报错退出。
force_mapping	JSON格式下，外表列无法匹配到正确的name-value键值对时的处理方式。	true，false。缺省值为true。 • force_mapping为true，相应的列填null，该null与JSON定义中的null含义相同。 • force_mapping为false，查询报错，提示不存在这样的列。 **注意** 由于对JSON对象没有限制，但外表字段定义需要符合DWS的标识符规范（例如长度、字符等限制），因此这种导入方式可能导致异常：例如，字段无法正确标识JSON name、字段需重复定义等。建议使用容错性选项force_mapping或json操作符（可参考**JSON/JSONB函数和操作符**）来规避。 JSON格式下执行SELECT COUNT(*) 不会去解析具体字段，因此不会对字段缺失、格式错误等情况报错。

版权所有 © 华为云计算技术有限公司

表 21-174 OPTIONS 参数说明（仅 Server 类型为 DLI 时设置）

参数	描述	取值范围
dli_project_id	DLI服务对应的项目编号，可在管理控制台上获取项目ID，该参数仅支持server类型为DLI时设置。该参数仅8.1.1及以上集群版本支持。	-
dli_database_name	待访问的DLI多版本表所在的数据库名称，该参数仅支持server类型为DLI时设置。该参数仅8.1.1及以上集群版本支持。	-
dli_table_name	待访问的DLI多版本表的名称，该参数仅支持server类型为DLI时设置。该参数仅8.1.1及以上集群版本支持。	-

表 21-175 OPTIONS 参数说明（仅 format 为 hudi 时设置，以下参数 9.1.0.100 及以上集群版本支持。）

参数	描述	取值范围
auth_server	指定一个server，用于支持hudi外表数据使用KMS加密时的用户身份认证，仅支持format为hudi时指定。	取值范围：可用的hdfs server名。 MRS服务提供了hudi数据列级加密的能力，通过RangerKMS对用户以及keyId进行管理以及加解密权限控制。用户通过在创建外表时指定auth_server参数，绑定RangerKMS用户信息，进一步进行安全认证、获取密钥，从而实现对读取加密数据的支持。
kms_url	指定KMS服务的ip和端口。仅支持format为hudi、且指定了auth_server时指定。	取值范围：合法的ip:port风格字符串。 一般情况下，仅指定auth_server即可自动解析KMS服务的ip和端口。如果指定了该参数，则会强制使用地址访问KMS服务。
cow_improve	针对COPY_ON_WRITE表的优化参数，该参数打开时，DN会替代dws-bigdata组件直接从OBS上读取数据。 仅支持format为hudi时指定，且不能与auth_server同时指定。	取值范围：true/on，false/off。 缺省值为false/off。 **说明** MERGE_ON_READ表指定该参数无效。

版权所有 © 华为云计算技术有限公司

表 21-176 OBS 外表支持的 OPTIONS 说明

参数名称	OBS										
-	TEXT	CSV	JSON	ORC			CARBONDATA	PARQUET			HUDI
	READONLY	READONLY	READONLY	READONLY	WRITEONLY	READWRITE	READONLY	READONLY	WRITEONLY	READWRITE	READONLY
location	√	√	√	√	×	×	√	√	×	×	×
format	√	√	√	√	√	√	√	√	√	√	√
header	×	√	×	×	×	×	×	×	×	×	×
delimiter	√	√	×	×	×	×	×	×	×	×	×
quote	×	√	×	×	×	×	×	×	×	×	×
escape	×	√	×	×	×	×	×	×	×	×	×
null	√	√	×	×	×	×	×	×	×	×	×
noescaping	√	×	×	×	×	×	×	×	×	×	×
encoding	√	√	√	√	√	√	√	√	√	√	√
fill_missing_fields	√	√	×	×	×	×	×	×	×	×	×
ignore_extra_data	√	√	×	×	×	×	×	×	×	×	×
date_format	√	√	√	×	×	×	×	×	×	×	×

版权所有 © 华为云计算技术有限公司

参数名称	OBS										
time_format	√	√	√	×	×	×	×	×	×	×	×
timestamp_format	√	√	√	×	×	×	×	×	×	×	×
smalldatetime_format	√	√	√	×	×	×	×	×	×	×	×
chunksize	√	√	√	×	×	×	×	×	×	×	×
filenames	×	×	×	×	×	×	×	×	×	×	×
foldername	√	√	√	√	√	√	√	√	√	√	√
dataencoding	×	×	×	×	×	×	×	×	×	×	×
filesize	×	×	×	×	×	×	×	×	×	×	×
compression	√	√	√	×	√	√	×	×	√	√	×
version	×	×	×	×	√	√	×	×	×	×	×
checkencoding	√	√	√	√	×	√	√	√	√	√	√
totalrows	√	√	√	√	×	√	×	×	×	×	×

版权所有 © 华为云计算技术有限公司

参数名称	OBS										
force_mapping	×	×	√	×	×	×	×	×	×	×	×
auth_server	×	×	×	×	×	×	×	×	×	×	√
kms_url	×	×	×	×	×	×	×	×	×	×	√
cow_improve	×	×	×	×	×	×	×	×	×	×	√
julian_adjust	×	×	×	×	×	×	×	√	×	×	√

表 21-177 HDFS 外表支持的 OPTIONS 说明

参数名称	HDFS								
-	TEXT	CSV	JSON	ORC			PARQUET		
	READ ONLY	READ ONLY	READ ONLY	READ ONLY	WRITE ONLY	READ WRITE	READ ONLY	WRITE ONLY	READ WRITE
location	×	×	×	×	×	×	×	×	×
format	√	√	√	√	√	√	√	√	√
header	×	√	×	×	×	×	×	×	×
delimiter	√	√	×	×	×	×	×	×	×
quote	×	√	×	×	×	×	×	×	×
escape	×	√	×	×	×	×	×	×	×
null	√	√	×	×	×	×	×	×	×

版权所有 © 华为云计算技术有限公司

参数名称	HDFS								
noescaping	√	×	×	×	×	×	×	×	×
encoding	√	√	√	√	√	√	√	√	√
fill_missing_fields	√	√	×	×	×	×	×	×	×
ignore_extra_data	√	√	×	×	×	×	×	×	×
date_format	√	√	√	×	×	×	×	×	×
time_format	√	√	√	×	×	×	×	×	×
timestamp_format	√	√	√	×	×	×	×	×	×
smalldatetime_format	√	√	√	×	×	×	×	×	×
chunksize	√	√	√	×	×	×	×	×	×
filenames	√	√	√	√	×	×	√	×	×
foldername	√	√	√	√	√	√	√	√	√
dataencoding	×	×	×	×	√	√	×	×	×
filesize	×	×	×	×	√	√	×	×	×
compression	√	√	√	×	√	√	×	√	√
version	×	×	×	×	√	√	×	×	×
checkencoding	√	√	√	√	√	√	√	√	√

版权所有 © 华为云计算技术有限公司

参数名称	HDFS								
totalrows	×	×	×	×	×	×	×	×	×
force_mapping	×	×	√	×	×	×	×	×	×
julian_adjust	×	×	×	×	×	×	√	×	×

说明

chunksize表示在DN中每个HDFS读取线程的缓存大小，支持text，csv，json格式，默认大小为4MB。

信息约束（Informational Constraint）

在DWS中，数据的约束完全由使用者保证，数据源数据能够严格遵守某种信息约束条件，能够加速对已经具有这种约束特征数据的查询。目前外表不支持索引，所以采取使用Informational Constraint信息优化Plan，提高查询性能。

建立外表信息约束的约束条件：

- 只有用户保证表中的其中一列的非空值具有唯一性时才可以建立Informational Constraint，否则查询结果将与期望值不同。
- DWS的Informational Constraint只支持PRIMARY KEY和UNIQUE两种约束。
- DWS的Informational Constraint支持NOT ENFORCED属性，不支持ENFORCED属性。
- 一个表上的多列可以分别建立UNIQUE类型的Informational Constraint，但是PRIMARY KEY一个表中只能建立一个。
- 一个表的一列上可以建立多个Informational Constraint（由于一个列上有多个约束和一个的作用一致，所以不建议一个列上建立多个Informational Constraint），但是Primary Key类型只能建立一个。
- 不支持表级COMMENT。
- 不支持多列组合约束。
- ORC格式只写外表不支持同一个集群不同CN向同一外表并发导出。
- ORC格式只写外表的目录，只能用于DWS的单个外表的导出目录，不能用于多个外表，并且其他组件不能向此目录写入其他文件。

示例：创建 HDFS 外表

在HDFS通过HIVE导入TPC-H benchmark测试数据表part表及region表。part表的文件路径为/user/hive/warehouse/partition.db/part_4，region表的文件路径为/user/hive/warehouse/mppdb.db/region_orc11_64stripe/。

1. 创建HDFS_Server，对应的foreign data wrapper为HDFS_FDW或者DFS_FDW。

```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER HDFS_FDW OPTIONS
(address '10.10.0.100:25000,10.10.0.101:25000',
```

 版权所有 © 华为云计算技术有限公司

```
hdfscfgpath '/opt/hadoop_client/HDFS/hadoop/etc/hadoop',
type'HDFS');
```

说明

- 在可选项options里面写入了HDFS集群对应的NameNode的IP地址及端口号。具体端口号请在MRS-HDFS服务配置中搜索参数“dfs.namenode.rpc.port”查看。本示例假设端口号为25000。
- ‘10.10.0.100:25000,10.10.0.101:25000’中列出了两组NameNode的地址及端口号，分别表示HDFS的主NameNode及备NameNode，这里推荐使用该种主备方式填写。两组参量中间使用“,”进行分割。

2. 创建HDFS外表。表关联的HDFS server为hdfs_server，表ft_region对应的HDFS服务器上的文件格式为‘orc’，在HDFS文件系统上对应的文件目录为'/user/hive/warehouse/mppdb.db/region_orc11_64stripe/'。
   - 创建不包含分区列的HDFS外表：

```
CREATE FOREIGN TABLE ft_region
(
   R_REGIONKEY INT4,
   R_NAME TEXT,
   R_COMMENT TEXT
)
SERVER
   hdfs_server
OPTIONS
(
   FORMAT 'orc',
   encoding 'utf8',
   FOLDERNAME '/user/hive/warehouse/mppdb.db/region_orc11_64stripe/'
)
DISTRIBUTE BY
    roundrobin;
```

   - 创建包含分区列的HDFS外表：

```
CREATE FOREIGN TABLE ft_part
(
    p_partkey int,
    p_name text,
    p_mfgr text,
    p_brand text,
    p_type text,
    p_size int,
    p_container text,
    p_retailprice float8,
    p_comment text
)
SERVER
    hdfs_server
OPTIONS
(
    FORMAT 'orc',
    encoding 'utf8',
    FOLDERNAME '/user/hive/warehouse/partition.db/part_4'
)
DISTRIBUTE BY
    roundrobin
PARTITION BY
    (p_mfgr) AUTOMAPPED;
```

说明

DWS支持2种文件指定方式：通过关键字filenames指定和通过foldername指定。推荐通过使用foldername进行指定。关键字distribute指定了表ft_region的存储分布方式。

3. 查看创建的外表：

```
SELECT * FROM pg_foreign_table WHERE ftrelid='ft_region'::regclass;
 ftrelid | ftserver | ftwriteonly |                              ftoptions
```

 版权所有 © 华为云计算技术有限公司

```
---------+----------+-------------+------------------------------------------------------------------------------
  16510 |    16509 | f           | {format=orc,foldername=/user/hive/warehouse/mppdb.db/
region_orc11_64stripe/}
(1 row)

select * from pg_foreign_table where ftrelid='ft_part'::regclass;
 ftrelid | ftserver | ftwriteonly |                          ftoptions
---------+----------+-------------+-------------------------------------------------------------------
  16513 |    16509 | f           | {format=orc,foldername=/user/hive/warehouse/partition.db/part_4}
(1 row)
```

示例：创建 HDFS 只写外表

通过HDFS只写外表，将TPC-H benchmark测试数据表region中的数据导出至HDFS文件系统的/user/hive/warehouse/mppdb.db/region_orc/目录下。

1. 创建HDFS SERVER，对应的foreign data wrapper为HDFS_FDW或者DFS_FDW，同**示例一**。
2. 创建HDFS只写外表。

```
CREATE FOREIGN TABLE ft_wo_region
(
    R_REGIONKEY INT4,
    R_NAME TEXT,
    R_COMMENT TEXT
)
SERVER
    hdfs_server
OPTIONS
(
    FORMAT 'orc',
    encoding 'utf8',
    FOLDERNAME '/user/hive/warehouse/mppdb.db/region_orc/'
)
WRITE ONLY;
```

3. 通过只写外表向HDFS文件系统写入数据。

```
INSERT INTO ft_wo_region SELECT * FROM region;
```

示例：创建 HDFS 只写分区外表（仅存算分离 3.0 版本支持）

通过HDFS多级分区只写外表将TPC-H benchmark测试数据表region表中的数据导出HDFS文件系统的/user/hive/warehouse/mppdb.db/region_orc/目录下。

1. 创建HDFS外表，对应的foreign data wrapper为HDFS_FDW或者DFS_FDW，同**示例：创建HDFS外表**。
2. 创建HDFS多级分区只写外表。

```
CREATE FOREIGN TABLE ft_wo_region_partition
(
    R_REGIONKEY INT4,
    R_NAME TEXT,
    R_COMMENT TEXT
)
SERVER
    hdfs_server
OPTIONS
(
    FORMAT 'orc',
    encoding 'utf8',
    FOLDERNAME '/user/hive/warehouse/mppdb.db/region_orc/'
)
WRITE ONLY
PARTITION BY (R_REGIONKEY, R_NAME);
```

3. 通过只写外表向HDFS文件系统写入数据。

```
INSERT INTO ft_wo_region_partition SELECT * FROM region;
```


版权所有 © 华为云计算技术有限公司

示例：创建 OBS 只读外表（仅存算分离 3.0 版本支持）

通过partsupp外表从obs文件系统读取数据。

1. 创建obs server。

   须知

   - 认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
   - 了解更多创建Server语法请参见**CREATE SERVER**。

   ```
   CREATE SERVER obs_srv FOREIGN DATA WRAPPER dfs_fdw OPTIONS (
       address 'obs.example.com',
       type 'obs',
       access_key 'xxx',
       secret_access_key 'xxx'
   );
   ```

2. 创建obs外表partsupp，使用disk cache中的冷缓存。

   ```
   CREATE FOREIGN TABLE PARTSUPP
   (
     PS_PARTKEY     BIGINT NOT NULL,
     PS_SUPPKEY     BIGINT NOT NULL,
     PS_AVAILQTY    BIGINT NOT NULL,
     PS_SUPPLYCOST  DECIMAL(15,2)  NOT NULL,
     PS_COMMENT     VARCHAR(199) NOT NULL
   ) SERVER obs_srv options (
       encoding     'utf-8',
       format       'parquet',
       foldername   'xxx',
       cache_policy 'NONE'
   ) READ ONLY DISTRIBUTE BY ROUNDROBIN;
   ```

3. 通过外表partsupp从obs上读取数据。

   ```
   SELECT * FROM PARTSUPP;
   ```

示例：创建包含信息约束（指定 not enforced）的 HDFS 外表

关于包含信息约束（Informational Constraint）HDFS外表的相关操作。

- 创建含有信息约束（Informational Constraint）的HDFS外表。

  ```
  CREATE FOREIGN TABLE ft_region  (
   R_REGIONKEY  int,
   R_NAME TEXT,
   R_COMMENT TEXT
    , primary key (R_REGIONKEY) not enforced)
  SERVER hdfs_server
  OPTIONS(format 'orc',
      encoding 'utf8',
   foldername '/user/hive/warehouse/mppdb.db/region_orc11_64stripe')
  DISTRIBUTE BY roundrobin;
  ```

- 查看region表是否有信息约束索引：

  ```
  SELECT relname,relhasindex FROM pg_class WHERE oid='ft_region'::regclass;
         relname          | relhasindex
  ------------------------+-------------
         ft_region        | f
  (1 row)

  SELECT conname, contype, consoft, conopt, conindid, conkey FROM pg_constraint WHERE conname
  ='ft_region_pkey';
    conname       | contype | consoft | conopt | conindid | conkey
  ----------------+---------+---------+--------+----------+--------
  ```


版权所有 © 华为云计算技术有限公司

```
 ft_region_pkey | p       | t       | t     |       0 | {1}
(1 row)
```

- 删除信息约束：

```
ALTER FOREIGN TABLE ft_region DROP CONSTRAINT ft_region_pkey RESTRICT;

SELECT conname, contype, consoft, conindid, conkey FROM pg_constraint WHERE conname
='ft_region_pkey';
 conname | contype | consoft | conindid | conkey
---------+---------+---------+----------+--------
(0 rows)
```

- 添加一个唯一信息约束：

```
ALTER FOREIGN TABLE ft_region ADD CONSTRAINT constr_unique UNIQUE(R_REGIONKEY) NOT
ENFORCED;
```

删除唯一信息约束：

```
ALTER FOREIGN TABLE ft_region DROP CONSTRAINT constr_unique RESTRICT;

SELECT conname, contype, consoft, conindid, conkey FROM pg_constraint WHERE conname
='constr_unique';
 conname | contype | consoft | conindid | conkey
---------+---------+---------+----------+--------
(0 rows)
```

- 添加一个唯一信息约束：

```
ALTER FOREIGN TABLE ft_region ADD CONSTRAINT constr_unique UNIQUE(R_REGIONKEY) NOT
ENFORCED disable query optimization;

SELECT relname,relhasindex FROM pg_class WHERE oid='ft_region'::regclass;
       relname         | relhasindex
-----------------------+-------------
       ft_region       | f
(1 row)
```

删除唯一信息约束：

```
ALTER FOREIGN TABLE ft_region DROP CONSTRAINT constr_unique CASCADE;
```

示例：通过 OBS 外表读取 OBS 的 json 数据

通过外表读取OBS上的json数据。

1. OBS上有如下json文件，json对象中存在嵌套、数组，部分对象的某些字段缺失，部分对象name重复。

```
{"A" : "simple1", "B" : {"C" : "nesting1"}, "D" : ["array", 2, {"E" : "complicated"}]}
{"A" : "simple2", "D" : ["array", 2, {"E" : "complicated"}]}
{"A" : "simple3", "B" : {"C" : "nesting3"}, "D" : ["array", 2, {"E" : "complicated3"}]}
{"B" : {"C" : "nesting4"},"A" : "simple4",  "D" : ["array", 2, {"E" : "complicated4"}]}
{"A" : "simple5", "B" : {"C" : "nesting5"}, "D" : ["array", 2, {"E" : "complicated5"}]}
```

2. 创建obs_server，对应的foreign data wrapper为DFS_FDW。

```
CREATE SERVER obs_server FOREIGN DATA WRAPPER DFS_FDW OPTIONS (
  ADDRESS 'obs.example.com',
  ACCESS_KEY 'xxxxxxxxx',
  SECRET_ACCESS_KEY 'yyyyyyyyyyyyy',
  TYPE 'OBS'
);
```

说明

- ADDRESS是OBS的终端节点（Endpoint），请根据实际替换。也是使用region参数，通过指定regionCode在region_map文件中查找对应的域名。
- ACCESS_KEY和SECRET_ACCESS_KEY是云账号体系访问密钥。请根据实际替换。
- TYPE表示创建的Server为OBS Server。请保持OBS取值不变。

3. 创建OBS外表json_f ，定义字段名，以d#2_e为例，从命名可以看出该字段是数组d的第二个元素里嵌套的e对象。表关联的OBS服务器为obs_server。foldername为外表中数据源文件目录，即表数据目录在OBS上对应的文件目录。

 版权所有 © 华为云计算技术有限公司

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE FOREIGN TABLE json_f (
  a VARCHAR(10),
  b_c TEXT,
  d#1 INTEGER,
  d#2_e VARCHAR(30)
)SERVER obs_server OPTIONS (
    foldername '/xxx/xxx/',
    format 'json',
    encoding 'utf8',
    force_mapping 'true'
)distribute by roundrobin;
```

4. 查询外表json_f。由于容错性参数force_mapping默认打开，json对象缺失的字段会填NULL；json对象name重复的以最后一次出现的name为准。

```
SELECT * FROM json_f;
   a    |   b_c    | d#1 |    d#2_e
--------+----------+-----+--------------
 simple1 | nesting1 |   2 | complicated1
 simple2 |          |   2 | complicated2
 simple3 | nesting3 |   2 | complicated3
 simple4 | nesting4 |   2 | complicated4
 repeat  | nesting5 |   2 | complicated5
(5 rows)
```

示例：通过 DLI 外表访问 DLI 上的数据

通过外表读取DLI多版本外表。DLI多版本外表示例仅8.1.1及以上版本支持。

1. 创建dli_server，对应的foreign data wrapper为DFS_FDW。

```
CREATE SERVER dli_server FOREIGN DATA WRAPPER DFS_FDW OPTIONS (
  ADDRESS 'obs.example.com',
  ACCESS_KEY 'xxxxxxxxx',
  SECRET_ACCESS_KEY 'yyyyyyyyyyyyy',
  TYPE 'DLI',
  DLI_ADDRESS 'dli.example.com',
  DLI_ACCESS_KEY 'xxxxxxxxx',
  DLI_SECRET_ACCESS_KEY 'yyyyyyyyyyyyy'
);
```

说明

- ADDRESS是OBS的终端节点（Endpoint）。DLI_ADDRESS是DLI的终端节点（Endpoint），请根据实际替换。
- ACCESS_KEY和SECRET_ACCESS_KEY是云账号体系访问OBS服务的密钥。请根据实际替换。
- DLI_ACCESS_KEY和DLI_SECRET_ACCESS_KEY是云账号体系访问DLI服务的密钥。请根据实际替换。
- TYPE表示创建的Server为DLI Server。请保持DLI取值不变。

2. 创建访问DLI多版本的OBS外表customer_address，不包含分区列，表关联的DLI服务器为dli_server。其中project_id为xxxxxxxxxxxxxxx，dli上的database_name为database123，需要访问的table_name为table456，根据实际替换。

版权所有 © 华为云计算技术有限公司

须知

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE FOREIGN TABLE customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(36,33)                ,
    ca_location_type          char(20)
)
SERVER dli_server OPTIONS (
    FORMAT 'ORC',
    ENCODING 'utf8',
    DLI_PROJECT_ID 'xxxxxxxxxxxxxxx',
    DLI_DATABASE_NAME 'database123',
    DLI_TABLE_NAME 'table456'
)
DISTRIBUTE BY roundrobin;
```

3. 通过外表查询DLI多版本表的数据。

```
SELECT COUNT(*) FROM customer_address;
 count
-------
    20
(1 row)
```

相关链接

- 修改和删除外表语法请参见**ALTER FOREIGN TABLE (For HDFS or OBS)**和**DROP FOREIGN TABLE**。

21.13.4.9 CREATE FOREIGN TABLE (OBS 导入导出)

功能描述

在当前数据库创建一个OBS外表，用于OBS数据并行导入导出。该方式不需要用户再额外创建外部服务器server，使用数据库默认创建的gsmpp_server即可。

本语法功能仅支持OBS桶上的**TEXT和CSV格式**数据，如果需要处理**ORC、CARBONDATA、PARQUET**等格式，请参见**CREATE FOREIGN TABLE (SQL on OBS or Hadoop)**，且需要额外创建外部server。

外表（Foreign Table）是数据库或大数据平台中一种虚拟表结构，它不实际存储数据，而是通过元数据信息（如表结构、存储位置、文件格式等）将外部存储系统（如HDFS、OBS等）中的数据映射到数据库内部，使用户能够以标准SQL操作直接访问或操作外部数据。外表的本质是一种“数据代理”，将外部文件系统的数据逻辑映射成数据库可识别的表结构。

版权所有 © 华为云计算技术有限公司

注意事项

- 这种方式仅支持TEXT和CSV格式，并且需要额外指定OBS连接信息。
- 使用OBS外表访问OBS桶数据时，应确保DWS与OBS桶在同一个区域(Region)下。
- 创建的外表分为只读外表（READ ONLY）和只写外表（WRITE ONLY），缺省为只读外表。导入OBS数据时，请将外表设为READ ONLY；将数据导出到OBS时，请设为WRITE ONLY。
- 仅系统管理员dbadmin角色或已授权了USEFT权限的普通用户可执行外表操作。
  ```
  ALTER USER user_name USEFT;
  ```
- OBS外表不需要显式指定分布方式，默认支持ROUNDROBIN分布方式。
- 所创建外表只对**信息约束（Informational Constraint）**约束生效。
- 不支持OBS桶中带中文路径的场景。

表 21-178 OBS 外表支持读写格式说明

数据类型	DIST_FDW	
-	READ ONLY	WRITE ONLY
ORC	×	×
PARQUET	×	×
CARBONDATA	×	×
TEXT	√	√
CSV	√	√
JSON	×	×

语法格式

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( { column_name type_name [column_constraint ]
    | LIKE source_table | table_constraint [, ...]} [, ...] )
SERVER server_name
OPTIONS ( { option_name ' value ' } [, ...] )
[ { WRITE ONLY | READ ONLY }]
[ WITH error_table_name | LOG INTO error_table_name]
[PER NODE REJECT LIMIT 'value'] ;
```

- 其中column_constraint为：
  ```
  [CONSTRAINT constraint_name]
  {PRIMARY KEY | UNIQUE}
  [NOT ENFORCED [ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION] | ENFORCED]
  ```
- 其中table_constraint为：
  ```
  [CONSTRAINT constraint_name]
  {PRIMARY KEY | UNIQUE} (column_name)
  [NOT ENFORCED [ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION] | ENFORCED]
  ```

 版权所有 © 华为云计算技术有限公司

参数说明

表 21-179 CREATE FOREIGN TABLE (OBS 导入导出)参数说明

参数	描述	取值范围或示例
IF NOT EXISTS	如果已经存在相同名称的表，系统不会报错，而会发出一个通知，告知表关系已存在。	-
table_name	外表的表名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
column_name	外表中的字段名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
type_name	字段的数据类型。	-
SERVER server_name	外表的server名字。对于导入导出的OBS外表，可以使用初始数据库默认创建的gsmpp_server，也可以使用自定义的server。	• 如果使用自定义server，FOREIGN DATA WRAPPER应该为dist_fdw。 • 8.2.0及以上集群版本支持在自定义dist_fdw server中指定OBS访问相关参数：access_key、secret_access_key和security_token。当server中指定以上参数时，外表中不需要再重复指定。
OPTIONS	用于指定外表数据的各类参数。	• 数据格式参数，参见**表21-180**。 • 容错性参数，参见**表21-181**。 • 性能参数，参见**表21-182**。
READ ONLY	外表只读，该参数只供数据导入使用。	-
WRITE ONLY	外表只写。该参数只供数据导出使用。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
WITH error_table_name	数据导入过程中出现的数据格式错误信息将被写入error_table_name指定的错误信息表中，可以在并行导入结束后查询此错误信息表，获取详细的错误信息。此参数只在设置了reject_limit参数时有效。	字符串，需符合标识符的命名规范。 **注意** 如果为了兼容postgres开源接口，此语法建议用LOG INTO代替。该参数指定时错误表自动创建。
LOG INTO error_table_name	数据导入过程中出现的数据格式错误信息将被写入error_table_name指定的错误信息表中，可以在并行导入结束后查询此错误信息表，获取详细的错误信息。	字符串，需符合标识符的命名规范。 ● 若没有指定PER NODE REJECT LIMIT参数，则此参数不起作用。 ● 该参数指定时，错误表自动创建。
PER NODE REJECT LIMIT 'value'	指定本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量，如果有一个DN实例上的错误数量大于设定值，本次导入失败，报错退出。	整型，unlimited（无限），不指定该参数时，有错误信息立即返回。 **注意** 此语法指定的是单个节点的错误容忍度。 数据格式错误是指缺少或者多出字段值，数据类型错误或者编码错误。对于非数据格式错误，一旦发生就将导致整个数据扫描失败。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
NOT ENFORCED	指定所建立的约束为信息约束，即数据库不会强制验证该约束，这个选项与后续的ENABLE QUERY OPTIMIZATION搭配使用，主要有以下作用： • **仅为信息约束**：通过 PRIMARY KEY 或 UNIQUE 声明列的唯一性约束，帮助优化查询执行计划。 • **不强制验证**：数据库不会检查外部数据源中的数据是否实际满足约束条件（如是否存在重复值）。 • **性能优化**：允许查询优化器利用约束信息生成更高效的执行计划，但需用户自行确保数据符合约束，否则可能导致查询结果不准确。 • **风险**：若实际数据违反约束（如存在重复值），可能导致查询结果错误（如聚合结果异常、连接产生多余记录）。 更多信息参见**信息约束（Informational Constraint）**。	例如声明primary key，但是不强制验证。 CREATE FOREIGN TABLE hdfs_users (user_id INT PRIMARY KEY NOT ENFORCED, ...
ENFORCED	数据库强制验证约束，预留参数，目前DWS不支持设置为ENFORCED。	-
PRIMARY KEY (column_name)	指定所建立的信息约束位于column_name列上。	字符串，需符合标识符的命名规范，并且这个column_name必须存在。
ENABLE QUERY OPTIMIZATION	允许查询优化器利用信息约束生成更高效的执行计划，与前面的NOT ENFORCED搭配使用。	创建外表，为字段名p_partkey int增加唯一约束，但是不强制验证该约束，且允许优化器利用该信息约束生成更优执行计划： CREATE FOREIGN TABLE ft_part (p_partkey int UNIQUE NOT ENFORCED ENABLE QUERY OPTIMIZATION, ...

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
DISABLE QUERY OPTIMIZATION	禁止查询优化器利用信息约束生成高效的执行计划。	-

表 21-180 OPTIONS 数据格式参数说明

参数	描述	取值范围或示例
encrypt	数据传输过程中使用HTTPS，否则使用HTTP，默认off。	-
access_key	OBS访问协议对应的AK值（由用户从服务界面上用户信息里获取），创建外表时AK值不会加密保存到数据库的元数据表中。创建外表时不校验该正确性。	-
secret_access_key	OBS访问协议对应的SK值（由用户从服务界面上用户信息里获取），创建外表时SK值会加密保存到数据库的元数据表中。创建外表时不校验该正确性。	-
security_token	对应统一身份认证服务IAM中临时安全凭证的SecurityToken值，与临时AK、SK共同组成临时安全凭证，临时安全凭证有效期不超过24小时。该参数仅8.2.0及以上集群版本支持。 **注意** • 该参数仅8.2.0及以上集群版本支持。 • 使用该参数时，access_key和secret_access_key参数分别对应临时AK、SK。	-
chunksize	在DN中每个OBS读取线程的缓存大小，可指定范围8~512，默认大小为64，单位为MB。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
location	OBS外表的数据源位置，即OBS数据的存放目录，目前支持URL描述，格式为'obs://OBS桶名/文件夹名'，多个URL使用‘|’分割。应确保OBS桶与DWS集群在同一个区域，不支持跨区访问OBS桶数据。 使用该参数请参见**location参数说明**。	举例如下： OPTIONS (encoding 'utf8', location 'obs://<obs_bucket_name>/traffic-data/gcxx', format 'text', ...
region	可选参数，region参数指定regionCode，regionCode为云上的region信息。 若显式指定此参数，域名信息将会读取指定的region参数；若此参数不指定，则读取defaultRegion的值，即安装集群时指定的region。	TEXT、CSV格式的OBS导入导出外表格式参数使用说明如下： ● location参数必选，其中前缀gsobs、obs均支持，都识别为OBS的信息，若为gsobs时，其中包含obs url、bucket、prefix，若为obs时则表示bucket、prefix。 ● 多个桶的数据源数据之间使用分隔符‘|’进行分割，LOCATION 'obs://bucket1/folder/ | obs://bucket2/'，数据库将会扫描指定路径文件夹下面的所有对象。
format	外表中数据源文件的格式。	CSV或TEXT。缺省值为TEXT。DWS只支持CSV和TEXT格式。 ● CSV（逗号分隔文件格式）： – 格式的文件，可以有效处理数据列中的换行符，但对一些特殊字符处理有欠缺。 – 由记录组成，每条记录被分隔符分隔为字段，且每条记录都有同样的字段序列。 ● TEXT（文本格式）：由换行符区分每条记录，由分隔符区分每个字段。可以有效处理一些特殊字符，但无法正确处理数据列中的换行符。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
header	指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。	OBS导出数据时不支持该参数为true，使用缺省值false，不需要设置，表示导出的数据文件第一行不是标题行（即表头）。 在导入数据时，如果header选项为on，则数据文件中第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。 取值范围：true/on，false/off。缺省值为false/off。
delimiter	指定数据文件行数据的字段分隔符，不指定则使用默认分隔符，TEXT格式的默认分隔符是水平制表符（tab），CSV格式的默认分隔符为“,”。	支持多字符分隔符，但分隔符不能超过10个字节。 ● TEXT格式，分隔符不能是\r和\n。 ● 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。 ● TEXT格式数据的分隔符不能包含：字母、数字以及特殊字符“\”和“.”。 ● 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。 ● 分隔符推荐使用多字符（例如'$^&'）和不可见字符（例如0x07、0x08、0x1b等）。
quote	CSV格式文件下的引号字符，缺省值为双引号。	● quote参数不能和分隔符、null参数相同。 ● quote参数只能是单字节的字符。 ● 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。
escape	CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。	缺省值为双引号。当与quote值相同时，会被替换为'\0'。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
null	用来指定数据文件中空值的表示方式。	• 在TEXT格式下缺省值是\N。 • CSV格式下缺省值是一个没有引号的空字符串。 • null的值不能是\r和\n，最大为100个字符。 • null值不能和分隔符、quote参数相同。
noescaping	TEXT格式下，开启后不对'\'和后面的字符进行转义。	noescaping参数只在TEXT格式下有效。 true/on，false/off。缺省值为false/off。
encoding	指定数据文件的编码格式名称，即需要以何编码格式对数据文件进行解析和校验/输出文件为何种编码格式。缺省值为当前数据库的默认客户端编码格式，即client_encoding。	导入外表此处强烈建议指定为文件的编码格式，或根据文件的字符集在导入前对client_encoding进行设置。否则可能会导致不必要的解析、校验错误以及其导致的导入报错回滚，甚至非法数据入库。导出外表同样希望指定此选项，以避免导出采用默认字符集设置时与预期不符。 在创建外表时此选项未指定，会在客户端给出对应Warning信息。 **注意** • 目前OBS导入外表不支持解析带有多种字符集编码格式混合的文件。 • 目前OBS导出外表不支持写出带有多种字符集编码格式混合的文件。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
eol	指定导入导出数据文件换行符样式。	支持多字符换行符，但换行符不能超过10个字节。常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。 • eol参数只能用于TEXT格式的导入导出，不支持CSV格式。 • eol参数不能和分隔符、null参数相同。 • eol参数不能包含：数字，字母和符号“.”。
date_format	导入对于DATE类型指定格式。此语法仅对READ ONLY的外表有效。	合法DATE格式。可参考**时间、日期处理函数和操作符**。 **说明** 对于指定为ORACLE兼容类型的数据库，则DATE类型内建为TIMESTAMP类型。在导入的时候，若需指定格式，可以参考下面的timestamp_format参数。
time_format	导入对于TIME类型指定格式。此语法仅对READ ONLY的外表有效。	合法的TIME格式，不支持时区。
timestamp_format	导入对于TIMESTAMP类型指定格式。此语法仅对READ ONLY的外表有效。	合法的TIMESTAMP格式，不支持时区。
smalldatetime_format	导入对于SMALLDATETIME类型指定格式。此语法仅对READ ONLY的外表有效。	合法的SMALLDATETIME格式。
bom	标识CSV文件是否包含utf8 BOM字段。	仅在只读外表且文件编码为utf8时生效。 取值范围：true/on，false/off。 缺省值：false

版权所有 © 华为云计算技术有限公司

表 21-181 OPTIONS 容错性参数说明

参数	描述	取值范围
fill_missing_fields	当数据导入时，若数据源文件中一行的最后一个字段缺失的处理方式。	true/on，false/off。缺省值为false/off。 ● 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 ● 参数为false/off，如果最后一个字段缺失会显示如下错误信息。 `missing data for column "tt"`
ignore_extra_data	数据源文件中的字段比外表定义列数多时，是否忽略多出的列。该参数只在数据导入过程中使用。	true/on，false/off。缺省值为false/off。 ● 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。 ● 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。 `extra data after last expected column` **注意** 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。
reject_limit	指定本次数据导入允许出现的数据格式错误个数，当导入过程中出现的数据格式错误未达到限定值时，本次数据导入可以成功。	整型值、unlimited（无限制）。 不指定该参数时，有错误信息立即返回。 **注意** 此语法建议用PER NODE REJECT LIMIT 'value'替代。 数据格式错误是指缺少或者多出字段值，数据类型错误或者编码错误。对于非数据格式错误，一旦发生就将导致整个数据导入失败。
force_save_err	表示是否在导入报错退出后，保存错误信息至错误表中。	true/on，false/off。缺省值为false/off。 与reject_limit配合使用，若该参数开启： ● reject_limit不指定时，保留一条错误记录至错误表中。 ● reject_limit设置为N，保留N+1条错误记录至错误表中。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
obs_null_file	用于DWS和OBS之间空文件导入导出的处理。 **注意** • 该参数仅8.2.1及以上版本支持。 • obs_null_file为true/on时，导出时若导出目录只有“_SUCCESS”空文件时，可以重复导出，obs_null_file为false/off时不能。 • obs_null_file为true/on时，多桶导入，对于第一个文件不存在的路径进行报错。	true/on，false/off。缺省值为false/off。 当obs_null_file为true/on时： • DWS导出空表时，生成名字为“_SUCCESS”的空文件，表示导出成功；导出非空表时，生成本身的文件+“_SUCCESS”空文件。 • DWS导入文件时，如果文件不存在或者路径错误，会显示如下错误信息。 `No such file or directory: 'XXX'`
compatible_illegal_chars	导入非法字符容错参数。此语法仅对READ ONLY的外表有效。	true/on，false/off。缺省值为false/off。 • 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。 • 参数为false/off，导入时遇到非法字符进行报错，中断导入。 • 导入非法字符容错规则如下： 1. 对于'\0'，容错后转换为空格； 2. 对于其他非法字符，容错后转换为问号； 3. 若compatible_illegal_chars为true/on标识导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如“illegal chars conversion may confuse COPY escape 0x20”等报错信息提示用户修改可能引起混淆的参数以避免导入错误。 **注意** Windows平台下OBS若按照文本格式读取数据文件，遇到0x1A会作为EOF符号结束数据读入造成解析错误，这是Windows平台的实现约束。由于OBS不支持BINARY形式读取，可将相应数据文件交由Linux平台下的OBS读取。

版权所有 © 华为云计算技术有限公司

表 21-182 OPTIONS 性能参数说明

参数	描述	取值范围
file_split_threshold	TEXT格式导入性能优化参数，含义是指定文件逻辑分块的大小下限。当指定该参数时，会根据实际文件和DN状况，通过对大文件进行拆分来提高导入并发度，最终目的是让各个DN的任务均衡分布。因此适合文件数少于DN数或文件大小不均衡的导入场景。	0~2147483647，单位MB。默认为0，不会拆分文件。 ● 该参数仅8.2.0及以上版本支持。 ● 该参数仅支持TEXT格式READ ONLY外表。 ● 该参数的含义是指定文件逻辑分块的大小下限，因此文件不一定按照该参数切块，但切块大小一定不会小于file_split_threshold。 例如：当前文件大小为1024MB，DN数为4。当指定file_split_threshold小于256时，文件会被均匀的分成4块，每个DN分配一个256MB的文件导入任务；当指定file_split_threshold为500时，由于用户明确告知不希望分块小于500MB，因此最终文件被拆分成500MB和524MB，分配给2个DN。多文件时该参数仍适用。 ● 除非有明确的分块大小需求，一般建议设置较小的值，例如10，否则可能影响并发效果。

location 参数说明

- 对于只读外表（默认为只读）的URL末尾可以指定到对象路径的前缀或直接指定到对象全路径。指定方式为obs://bucket/prefix。（其中，prefix是指对象路径的前缀。）例如：obs://mybucket/tpch/nation/
- 对于obs://bucket/prefix格式，若显式指定region参数时，域名信息将会读取指定的region参数；若region参数不指定，则读取defaultRegion的值，即安装集群时指定的region。
- 对于可写外表，URL不需要指定文件名。外部表数据源位置只可指定一个，并且要预先创建好对应目录。
- 对于只读外表不能使用多个相同的URL地址。
- 向外表中插入数据需要指定location。
- location参数中前缀gsobs、obs均支持，都识别为OBS的信息，若为gsobs时，其中包含obs url、bucket、prefix，若为obs时则表示bucket、prefix。

在实际导入导出数据时，location参数使用建议如下：

版权所有 © 华为云计算技术有限公司

- 导入时“location”建议指定到具体文件名。如果仅指定到OBS桶或目录，则会导入其中的所有文本文件。当数据格式不正确时，则会报错。如果设置了容错，则容错表可能导入大量数据。
- 支持OBS单桶多文件导入，根据文件名前缀进行匹配，匹配到的文件都会被导入。

  例如，有以下两个数据文件，只要在“location”中指定前缀mybucket/input_data/product_info就能识别并导入这两个文件。

  ```
  mybucket/input_data/product_info.0
  mybucket/input_data/product_info.1
  ```
- 导入时如果指定到文件名，例如“1.csv”，那么在此文件的桶或目录存在此名称为前缀的其他文件，也会被导入。即“1.csv1”、“1.csv22”等，都会被自动导入。
- 导入时，“location”中如果使用obs方式，支持多个url，并且用‘|’分隔；如果使用gsobs方式，则不支持多个路径。
- 导出时“location”默认按目录处理。如果仅指定到自定义名称的文件，则导出时会以该文件为名称创建目录，然后再生成导出文件。文件名由DWS自动生成。
- 导出时“location”只支持一个路径。

示例

创建外表OBS_ft，用来以txt格式导入OBS上指定的对象数据到row_tbl表中：

⚠ 注意

认证用的AK和SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
DROP FOREIGN TABLE IF EXISTS OBS_ft;

CREATE FOREIGN TABLE OBS_ft
( a int, b int)
SERVER gsmpp_server
OPTIONS
(location 'obs://gaussdbcheck/obs_ddl/test_case_data/txt_obs_informatonal_test001',
format 'text',
encoding 'utf8',
chunksize '32',
encrypt 'on',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
delimiter E'\x08')
read only;

DROP TABLE row_tbl;

CREATE TABLE row_tbl( a int, b int);

INSERT INTO row_tbl select * from OBS_ft;
```

相关链接

ALTER FOREIGN TABLE (For HDFS or OBS)，**DROP FOREIGN TABLE**

版权所有 © 华为云计算技术有限公司

优化建议

- delimiter
  - 分隔符不能是\r和\n，分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。
  - 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。
  - 分隔符推荐使用多字符（例如'$^&'）和不可见字符（例如0x07、0x08、0x1b等）。
- quote
  - quote参数只能是单字节的字符，quote参数不能和分隔符、null参数相同。
  - 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。
- mode Normal
  - 支持所有文件格式(包括CSV、TEXT、FIXED)，数据导入需要在数据服务器上启动Gauss data service协助完成。
- mode Shared
  - 支持TEXT文集格式，此时导入无需Gauss data service的协助，但是需要将用户数据通过NFS或挂载到所有节点的相同路径下。
- mode Private
  - 适用于用户数据已经存放到Datanode节点的本地目录下，并且位于相同的路径。

21.13.4.10 CREATE FOREIGN TABLE (GDS 导入导出)

功能描述

在当前数据库创建一个GDS外表，用于数据并行导入导出。GDS外表分为只读外表和只写外表，分别用于数据并行导入和并行导出，缺省为只读外表。

GDS是一款运行在Linux操作系统的命令行工具，通过和外表机制的配合，实现数据的高速导入导出。GDS工具包需要安装在数据源文件所在的服务器上，数据源文件所在的服务器称为数据服务器，也叫GDS服务器。

注意事项

- 仅系统管理员dbadmin角色或已授权了USEFT权限的普通用户可执行外表操作。
  ```
  ALTER USER user_name USEFT;
  ```
- GDS外表不需要显式指定分布方式，默认支持ROUNDROBIN分布方式。
- 对于GDS外表指定任何约束（列约束、表约束等）均不生效。
- GDS导入导出支持的文件格式：TEXT、CSV和FIXED。

语法格式

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
   ( [ { column_name type_name POSITION(offset,length) | LIKE source_table } [, ...] ] )
   SERVER gsmpp_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   [ { WRITE ONLY | READ ONLY }]
   [ WITH error_table_name | LOG INTO error_table_name]
   [REMOTE LOG 'name']
```


版权所有 © 华为云计算技术有限公司

```
[PER NODE REJECT LIMIT 'value']
[ TO { GROUP groupname | NODE ( nodename [, ... ] ) } ];
```

参数说明

表 21-183 CREATE FOREIGN TABLE (GDS 导入导出)参数说明

参数	描述	取值范围
IF NOT EXISTS	如果已经存在相同名称的表，系统不会报错，而会发出一个通知，告知表关系已存在。	-
table_name	外表的表名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
column_name	外表中的字段名。	
type_name	字段的数据类型。	-
POSITION(offset,length)	在固定长度模式中，定义每一个字段在数据文件中的位置。 offset为该字段在数据源文件中的起始位置，length为该字段的长度。	offset取值不能小于0字节，单位为字节。 每条记录的长度不能大于1GB，文件中没有出现的列默认以空值代替。
SERVER gsmpp_server	外表的server名字。对于GDS外表，其server是初始数据库默认创建的，即gsmpp_server。	-
OPTIONS	用于指定外表数据的各类参数。	• 数据格式参数：参见**表21-184**。 • 容错性参数：参见**表21-185**。 • 性能参数：参见**表21-186**。
READ ONLY	只读外表，该参数只供数据导入使用。	-

版权所有 © 华为云计算技术有限公司

表 21-184 OPTIONS 数据格式参数说明

参数	描述	取值范围
location	外表的数据源位置，目前支持URL方式的描述。多个URL使用‘|’分隔。 gds目前可以支持导出的时候自动创建外表定义的目录。 如外表location定义“gsfs://192.168.0.91:5000/2019/09”执行导出任务的时候，如果gds数据目录下的子目录“2019/09”不存在则会自动的创建该子目录，不需要用户手动创建外表中指定的目录。	• 对于使用GDS从远端服务器并行导入时的只读外表（默认为只读）的URL末尾必须指定文件的匹配模式或者文件名。 例如: gsfs://192.168.0.90:5000/*或者file:///data/data.txt或者gsfs:// 192.168.0.90:5000/* | gsfs:// 192.168.0.91:5000/* • 对于使用GDS并行导出到远端服务器时的只写外表，URL不需要指定文件名。当导出数据文件存储位置为远端URL时，例如gsfs://192.168.0.90:5000/，则数据源位置可指定多个，此时：若导出数据文件存储位置数量小于等于数据节点数量时，使用此外部表执行导出任务，数据将被平均分配至各数据源位置；若导出数据存储位置数量大于数据节点数量时，执行导出任务，数据将被平均分配给此位置列表中从前端开始等于数据节点数量的数据源位置下，剩余数据源位置仍会创建数据文件，但文件中不会有任何数据。 • 对于使用GDS从远端服务器并行导入时的只读外表，URL个数应小于DN个数，且不能使用多个location相同的URL。 • 当使用gsfss协议，即当URL为“gsfss://”开头，进行加密导入导出时，并发数量不能超过10。 • gds导出时location指定的文件路径“gsfs://127.0.0.1:7789/2019/09/”中的2019/09子目录会在执行导出任务的时候自动创建。 • 设置file_type为“pipe”时，GDS会根据URL中最后一个字符是否为“/”来判断导入导出的目标是管道文件还是目录。如： – gsfs://192.168.0.90:5000/a/b，GDS会将b识别成一个管道文件。 – gsfs://192.168.0.90:5000/a/b/，

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
		GDS将b识别成一个目录，并在b目录下创建管道文件。
format	外表中数据源文件的格式。	取值范围：CSV、TEXT、FIXED，缺省值为TEXT。 ● CSV格式的文件，对一些转义序列按照普通字符串进行处理，因此可以有效处理数据列中的换行符。 ● TEXT格式的文件，可以有效处理一些转义序列，因此无法正确处理数据列中的换行符。 ● FIXED格式的文件，适用于每条数据的数据列都比较固定的数据，长度不足的列会添加空格补齐。 **说明** ● 转义序列指的是反斜杠开头的字符串，包括：\b（退格）、\f（换页）、\n（换行）、\r（回车）、\t（横向制表）、\v（纵向制表）、\数字（八进制编码）、\x数字（十六进制编码）。TEXT格式可以按照本身含义进行处理，其他格式只能按照普通字符串进行处理。 ● 定长格式（FIXED）定义如下。当为FIXED时，必须为每一列指定POSITION。 1. 每条记录的每个字段长度相同。 2. 长度不足的字段以空格填充，数字类型字段左对齐，字符字段右对齐。 3. 字段和字段之间没有分隔符。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
header	指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV，FIXED格式的文件中。 在导入数据时，如果header选项为on，则数据文件中第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。 在导出数据时，如果header选项为on，则需要指定**fileheader**。fileheader用来指定导出头文件的格式。如果header为off，则导出数据文件不包含标题行。	true/on，false/off。缺省值为false/off。
fileheader	指定导出数据要包含的标题行定义的文件，文件一般只包含一行用来描述每一列数据信息的字符串。 例如：在包含商品信息的数据前加标题行，定义文件如下 The information of products.\n	• 标题行定义文件仅在header为on或true的情况下有效，且需要提前写好备用。 • 在Remote导出模式下，定义文件必须放在GDS的工作目录（即启动gds时指定的-d路径）下。 • 定义文件只能包含一行标题信息，并以换行符结尾，多余的行将被丢弃（标题信息不能包含换行符）。 • 定义文件包括换行符在内长度不超过1M。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
out_filename_prefix	指定write only外表导出时，GDS端生成导出数据文件的文件名前缀。 file_type设置为pipe时，会生成“dbName_schemaName_foreignTableName.pipe”的管道文件。 如果out_filename_prefix和location中都指定了管道名，则以location中指定的管道文件名为准。	● 指定文件名前缀需合法，符合GDS部署物理环境使用的文件系统的约束，否则出现文件创建失败： – 指定的导出文件名前缀中不含有非法字符，其中非法字符包含但不限于'/', '?', '*', ':', '|', '\\', '<', '>', '@', '#', '$', '&', '(', ')', '+', '-'，允许的字符范围为[a-z]*[A-Z]*[0-9]*和'_' – 指定的导出文件名前缀中不可以是一些Windows和linux预留的特性字段，其中包括但不限于："con","aux","nul","prn","com0","com1","com2","com3","com4","com5","com6","com7","com8","com9","lpt0","lpt1","lpt2","lpt3","lpt4","lpt5","lpt6","lpt7","lpt8","lpt9" – 指定的导出文件名前缀，与GDS -d目录和“.dat”或者“.pipe”拼接为绝对路径后必须符合GDS所在部署文件系统的文件名长度要求。 – 指定的导出文件名前缀，需要可以被数据文件的最终接收方正确解析识别（包括但不限于GDS再次导入库中），对于造成文件名解析问题的指定选项，需要用户识别。 ● 在多文件同时导出的高并发导出场景下，请确认并发的导出任务不要使用同一个文件名前缀设定，否则从操作系统/文件系统层面可能会出现导出文件的覆盖和丢失。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
delimiter	指定数据文件行数据的字段分隔符，不指定则使用默认分隔符，TEXT格式的默认分隔符是水平制表符（tab），CSV格式的默认分隔符为“,”，FIXED格式没有分隔符。	支持多字符分隔符，但分隔符不能超过10个字节。 ● 分隔符不能是\r和\n。 ● 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。 ● TEXT格式数据的分隔符不能包含：\.abcdefghijklmnopqrstuvwxyz0123456789。 ● 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。 ● 分隔符推荐使用多字符（例如'$^&'）和不可见字符（例如0x07、0x08、0x1b等）。 ● 建议TEXT格式下多字符分隔符中的字符不要完全相同，例如不建议使用delimiter '---' 。
quote	用于设置将CSV格式数据源文件中的什么字符识别为引号字符。缺省值为双引号。	● quote参数不能和分隔符、null参数相同。 ● quote参数只能是单字节的字符。 ● 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。
escape	用来指定CSV格式的数据源文件中，什么字符为逃逸字符。逃逸字符只能指定为单字节字符。 缺省值和quote相同。	-
null	用来指定数据文件中空值的表示。	● 在TEXT格式下缺省值是\N。 ● CSV格式下缺省值是一个没有引号的空字符串。 ● null值不能是\r和\n，最大为100个字符。 ● null值不能和分隔符、quote参数相同。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
blank_number_str_to_nul	用来指定数据文件中数值类型（INT1OID、INT2OID、INT4OID、INT8OID、FLOAT4OID、FLOAT8OID）不定数量的空格转换为NULL值。该参数仅8.1.3、8.2.1.236、9.1.0及以上集群版本支持。 例如，设置该选项为true时，定义null ''，null ' '（一个空格）、null ' '（二个空格）时都返回NULL值。	true/false，默认值为false，表示不做转换。
noescaping	TEXT格式下，不对'\'和后面的字符进行转义。	noescaping参数只在TEXT格式下有效。true/on，false/off。缺省值为false/off。
encoding	指定数据文件的编码格式名称，即需要以何编码格式对数据文件进行解析和校验/输出文件为何种编码格式。缺省值为当前数据库的默认客户端编码格式，即client_encoding。 导入外表此处强烈建议指定为文件的编码格式，或根据文件的字符集在导入前对client_encoding进行设置。否则可能会导致不必要的解析、校验错误以及其导致的导入报错回滚，甚至非法数据入库。导出外表同样希望指定此选项，以避免导出采用默认字符集设置时与预期不符。 在创建外表时此选项未指定，会在客户端给出对应Warning信息。	• 目前GDS导入外表不支持解析带有多种字符集编码格式混合的文件，GDS导出外表不支持写出带有多种字符集编码格式混合的文件。 • 常用的编码格式有UTF8、GBK、GB18030，其中GB18030有两个版本，GB18030和GB18030_2022。GB18030_2022，用于支持汉字最新的国家标准**GB 18030-2022**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
dataencoding	用于指定数据在latin1数据库表中的实际编码，实现GDS导入导出时数据从dataencoding与encoding之间的转换。该参数仅8.2.0及以上集群版本支持。	GBK、GB18030、GB18030_2022、UTF8。 典型场景： 1. 将GB编码的文本数据导入到latin1库中并转换为UTF8编码，设置dataencoding为 'UTF8'，encoding为 'GB'。 2. 将latin1库中GB编码的数据导出为UTF8的数据文件，设置dataencoding为 'GB'，encoding为 'UTF8'。 **注意** ● 该参数仅支持在latin1的库中使用。 ● 使用该参数时，encoding参数的取值范围也被限定在GBK、GB18030、GB18030_2022和UTF8之内。
mode	指定数据导入过程中，数据导入策略。DWS只支持Normal策略。	取值范围： ● Normal（缺省值）：支持所有文件格式(包括CSV、TEXT、FIXED)，数据导入需要在数据服务器上启动Gauss data service协助完成。
eol	指定导入导出数据文件换行符样式。	取值范围：支持多字符换行符，但换行符不能超过10个字节。常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。 ● eol参数一般用于TEXT格式的导入导出，不支持CSV格式和FIXED格式导入。为了兼容原有eol参数，仍然支持导出CSV格式和FIXED格式时指定eol参数为0x0D或0x0D0A。 ● 导出FIXED格式时可指定eol参数为空串''，这样会将FIXED格式数据导出为一行，但是这种方式导出的数据不支持再通过同样的方式导入到DWS数据库中。 ● eol参数不能和分隔符、null参数相同。 ● eol参数不能包含：数字，字母和符号“.”。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
conflict_delimiter	此参数一般配合**compatible_illegal_chars**参数一起使用，当用户的数据文件中包含半个汉字字符的时候，并且这半个字符和分隔符会由于外表的编码和数据库编码不一致被编码成一个其他的汉字，导致分隔符被掩盖从而报错缺少字段。 如果用户不希望让这半个字符和分隔符编码成一个其他字符则可以使用此参数。	取值范围：true/on，false/off。缺省值为false/off。 ● 参数为true/on，允许这半个字符和分隔符编码成一个其他字符。 ● 参数为false/off，不允许这半个字符和分隔符编码成一个其他字符。 此参数默认关闭。由于场景较少出现，不建议打开。如果未能识别场景而打开此参数则会有入表信息错乱的风险。 例如：假设有一行GBK数据"3|+|臗+|20191212"要被导入到UTF8的数据库，用户自定义的字段分隔符为"|+|"。 这行GBK数据和十六进制对应关系如下 33 7C 2B 7C C4 7C 2B 7C 323031393132313 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ 3 | + | ** | + | 20191212 当导入UTF8数据库时，数据库会发生如下转换。 GBK -> UTF8 33->33 (3) 7C->7C (|) 2B->2B (+) 7C->7C(|) C47C-> E886A2(臗) 2B->2B(+) 7C->7C(|) 323031393132313-> 20191212 由于用户定义的字段分隔符号为"|+|"，因此对该行数据分隔后获取的数据是3，臗+|20191212, 实际上用户可能要获得的是3，？,20191212。 "臗" 这个字符其实是一个用户不期望的半个字符，和用户定义的分隔符“|+|”中的“|”产生了冲突，导致数据错乱或者导入失败。 如果用户想要忽略这种字符可以使用conflict_delimiter配合**compatible_illegal_chars**参数将“C4”这个半个GBK字符准换成“？”导入UTF8数据库。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
file_type	指定导入或者导出的文件类型。	取值范围：normal，pipe，interconn。缺省为normal。 ● 参数为normal，表示导入或者导出的文件类型为普通文件。 ● 参数为pipe，表示导入或者导出的文件类型为命名管道文件。 ● 参数为interconn，表示当前的GDS外表用于互联互通压缩导入导出，该参数必须配合gds_compress参数使用，仅8.2.0及以上集群版本支持。 **注意** 当file_type的参数为interconn时，gds需要升级到8.2.0及以上版本，否则会出现ERROR: un-support format报错。
auto_create_pipe	用于设置GDS进程是否自动创建命名管道文件。 设置auto_create_pipe参数时，file_type必须设置为pipe或者interconn，否则不能成功创建外表。	取值范围：true/on，false/off。缺省值为true/on。 ● 参数为true/on，表示允许GDS进程自动创建命名管道文件。 ● 参数为false/off，表示用户需手动创建命名管道文件。 ● auto_create_pipe设置为false，执行导入导出时，若未指定管道文件，会打开“数据库名_模式名_外表名.pipe”文件；若已指定管道文件，会打开location参数中指定的管道文件。该命名管道文件在pipe-timeout参数设置的时间内没有被其他程序写入或者以写的方式打开，则导入导出任务报错超时。若发现该文件不是管道文件则导入导出任务直接报错。 ● auto_create_pipe设置为true，执行导入导出时，若未指定管道文件，会打开“数据库名_模式名_外表名.pipe”文件；若该文件为普通文件类型，则导入导出任务报错。若为管道文件会自动删除该文件，并重新创建该命名管道文件。 ● 管道文件导出时location参数可以指定导出的管道文件，如：location‘gsfs://127.0.0.1:7789/aa.pipe'，当auto_create_pipe设置为true，GDS会自动在数据目录下创建“aa.pipe”管道文件。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
del_pipe	用于设置不落地导入/导出业务结束后是否自动删除管道文件。 设置del_pipe时，file_type必须设置为pipe或者interconn，否则不能成功创建外表。	取值范围：true/on，false/off。缺省值为true/on。 ● 参数为true/on，表示允许GDS进程自动删除命名管道文件。 ● 参数为false/off，表示GDS进程不会删除命名管道文件。
gds_compress	该参数用于降低用户使用GDS管道文件时集群之间互联互通的网络带宽，当前仅支持snappy算法。该参数仅8.2.0及以上集群版本支持。	取值范围：snappy。 当前仅支持snappy算法，表示GDS管道导入导出的数据是以snappy算法压缩和解压缩的。 ● 设置该参数时，file_type参数必须为interconn，format必须为CSV， 默认不压缩。当前仅支持设置为snappy，使用该功能时需要保证本地集群，远端集群，GDS的版本至少在同一版本。 ● 当file_type的参数为interconn时，gds需要升级到8.2.0及以上版本，否则会出现ERROR: un-support format报错。 ● 该参数会降低集群间的网络带宽，但是会增加集群的CPU负载，用户可以根据实际情况选择是否开启该参数。 ● 为了实现GDS互联互通外表迁移数据时，数据在网络层压缩传输，GDS外表必须支持该参数，用户不仅可以使用互联互通外表进行数据压缩迁移，也可以手动使用GDS外表实现数据在网络层压缩迁移，由于手动实现时需要用户设置参数较多，推荐使用互联互通外表的方案。 ● 导入和导出端的外表必须都指定该参数。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
preserve_blanks	用于定长导入时，控制每列数据拖尾的空白字符（包括空格，\t，\v，\f）是否保留。该参数仅8.2.0.100及以上集群版本支持。	取值范围：true/on，false/off。缺省值为false/off。 ● 参数为true/on，表示针对BYTEAOID、CHAROID、NAMEOID、TEXTOID、BPCHAROID、VARCHAROID、NVARCHAR2OID、CSTRINGOID对应类型的列数据，保留拖尾\t，不进行修剪。 ● 参数为false/off，表示针对所有数据类型的数据拖尾的空白字符（包括空格，\t，\v，\f）进行修剪。
fix	指定每一行定长格式数据的长度。按字节计算。此语法仅对READ ONLY的外表有效。	取值范围：N >= POSITION指定的总长度（总长度即为表定义最后一个字段的offset与length的和）AND N < 1GB
out_fix_alignment	定长导出中，指定BYTEAOID、CHAROID、NAMEOID、TEXTOID、BPCHAROID、VARCHAROID、NVARCHAR2OID、CSTRINGOID对应类型所在列的对齐方式。	取值范围：align_left、align_right 默认值：align_right **注意** 由于bytea数据类型要求十六进制格式（如"\XXXX"）或八进制格式（如"\XXX\XXX\XXX"），导入时需要左对齐（即列数据以两种格式开头，而非空格）。因此若导出文件需要重新以GDS外表入库且数据长度小于外表formatter指定长度，导出时需要指定左对齐，否则会在入库的过程中报错。
out_fix_num_alignment	定长导出中，指定数值类型TINYINT、SMALLINT、INTEGER、BINARY_INTEGER、BIGINT、NUMERIC[(p[,s])]、DECIMAL[(p[,s])]、NUMBER[(p[,s])]、REAL、FLOAT4、DOUBLE PRECISION、FLOAT8、FLOAT[(p)]、BINARY_DOUBLE、DEC[(p[,s])]、INTEGER[(p[,s])]、SMALLSERIAL、SERIAL、BIGSERIAL对应类型所在列的对齐方式。该选项仅9.1.0及以上集群版本支持。	取值范围：align_left、align_right 默认值：align_left

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
date_format	导入对于DATE类型指定格式。此语法仅对READ ONLY的外表有效。	取值范围：合法DATE格式。可参考**时间、日期处理函数和操作符**。 **注意** 对于指定为ORACLE兼容类型的数据库，则DATE类型内建为TIMESTAMP类型。在导入的时候，若需指定格式，可以参考下面的timestamp_format参数。
time_format	导入对于TIME类型指定格式。此语法仅对READ ONLY的外表有效。	取值范围：合法TIME格式，不支持时区。可参考**时间、日期处理函数和操作符**。
timestamp_format	导入对于TIMESTAMP类型指定格式。此语法仅对READ ONLY的外表有效。	取值范围：合法TIMESTAMP格式，不支持时区。可参考**时间、日期处理函数和操作符**。
smalldatetime_format	导入对于SMALLDATETIME类型指定格式。此语法仅对READ ONLY的外表有效。	取值范围：合法SMALLDATETIME格式。可参考**时间、日期处理函数和操作符**。

表 21-185 OPTIONS 容错性参数说明

参数	描述	取值范围
fill_missing_fields	当数据导入时，若数据源文件中一行的最后一个字段缺失的处理方式。	取值范围：true/on，false/off。缺省值为false/off。 • 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 • 参数为false/off，如果最后一个字段缺失会显示如下错误信息。 `missing data for column "tt"`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ignore_extra_data	若数据源文件比外表定义列数多，是否会忽略对多出的列。该参数只在数据导入过程中使用。	取值范围：true/on，false/off。缺省值为false/off。 ● 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。 ● 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。 `extra data after last expected column` **注意** 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。
reject_limit	指定本次数据导入允许出现的数据格式错误个数，当导入过程中出现的数据格式错误未达到限定值时，本次数据导入可以成功。 **说明** 此语法建议用PER NODE REJECT LIMIT 'value'替代。 数据格式错误是指缺少或者多出字段值，数据类型错误或者编码错误。对于非数据格式错误，一旦发生就将导致整个数据导入失败。	取值范围：正整型值、unlimited（无限制）。指定正整型参数时需要添加单引号。 不指定该参数时，有错误信息立即返回。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
compatible_illegal_chars	导入导出非法字符容错参数。此语法对READ ONLY和WRITE ONLY的外表有效。 导出容错仅8.1.3.331及以上版本集群支持。	取值范围：true/on，false/off。缺省值为false/off。 ● 参数为true/on，则导入导出时遇到非法字符进行容错处理，非法字符转换后入库或出库，不报错，不中断导入导出。 ● 参数为false/off，导入导出时遇到非法字符进行报错，中断导入。 ● 导入非法字符容错规则如下： – 对于'\0'，容错后转换为空格。 – 对于其他非法字符，容错后转换为问号。 – 如果compatible_illegal_chars为true/on标识导入时对于非法字符进行容错处理，如果NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数，以避免导入错误。 – 需注意：导出外表容错参数开启后，非法字符数据导出被容错为问号，再次导入回DWS库中就会与原数据不一致。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
replace_illegal_chars	导入非法字符容错参数，用于将'\0'替换为指定字符串。此语法仅对READ ONLY的外表有效。	指定用于替换'\0'的字符串有如下约束： ● 仅支持compatible_illegal_chars参数为true/on时设置。 ● 仅支持format参数为TEXT或CSV时设置。 ● 字符串长度不能超过10。 ● 字符串不能与delimiter相同或者包含delimiter，不能包含以下字符：数字，字母以及特殊字符“\”和“.”。
WITH error_table_name	数据导入过程中出现的数据格式错误信息将被写入error_table_name指定的错误信息表中，可以在并行导入结束后查询此错误信息表，获取详细的错误信息。此参数只在设置了reject_limit参数时有效。 **说明** 如果为了兼容postgres开源接口，此语法建议用LOG INTO代替。	字符串，要符合标识符的命名规范。
LOG INTO error_table_name	数据导入过程中出现的数据格式错误信息将被写入error_table_name指定的错误信息表中，可以在并行导入结束后查询此错误信息表，获取详细的错误信息。 **说明** 若没有指定PER NODE REJECT LIMIT参数，则此参数不起作用。	字符串，要符合标识符的命名规范。
REMOTE LOG 'name'	数据导入过程中出现的数据格式错误信息将被写到GDS端以文件方式保存。name为错误数据文件的文件名前缀。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PER NODE REJECT LIMIT 'value'	指定本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量，如果有一个DN实例上的错误数量大于设定值，本次导入失败，报错退出。 **注意** 此语法指定的是单个节点的错误容忍度。 数据格式错误是指缺少或者多出字段值，数据类型错误或者编码错误。对于非数据格式错误，一旦发生就将导致整个数据扫描失败。	整型值，unlimited（无限），不指定该参数时，有错误信息立即返回。

表 21-186 OPTIONS 性能参数说明

参数	描述	取值范围
file_sequence	用于多任务GDS外表并行导入，提升单个文件的导入性能。该参数仅供数据导入使用。 格式为file_sequence '文件被拆分的总数-当前分片' 。例如： file_sequence '3-1' 表示导入的文件在逻辑上被拆分成3份，当前外表导入的数据为第一个分片上的数据。 file_sequence '3-2' 表示导入的文件在逻辑上被拆分成3份，当前外表导入的数据为第二个分片上的数据。 file_sequence '3-3' 表示导入的文件在逻辑上被拆分成3份，当前外表导入的数据为第三个分片上的数据。	使用该参数需遵循以下约束： • 文件被拆分的总数小于等于8。 • 当前分片小于等于文件被拆分的总数。 • 导入的文件仅支持CSV和text格式。 使用CSV格式进行并行导入时，在某些场景中会因为CSV本身的规则和GDS拆分逻辑冲突而导致其中的某些分片导入失败，详情参见**使用file_sequence参数异常场景说明**。

示例

创建外表customer_ft，用来以TEXT格式导入GDS服务器10.10.123.234上的数据：

```
CREATE FOREIGN TABLE customer_ft
(
    c_customer_sk             integer               ,
    c_customer_id             char(16)              ,
    c_current_cdemo_sk        integer               ,
    c_current_hdemo_sk        integer               ,
    c_current_addr_sk         integer               ,
    c_first_shipto_date_sk    integer               ,
    c_first_sales_date_sk     integer               ,
    c_salutation              char(10)              ,
```


版权所有 © 华为云计算技术有限公司

```
    c_first_name               char(20)              ,
    c_last_name                char(30)              ,
    c_preferred_cust_flag      char(1)               ,
    c_birth_day                integer               ,
    c_birth_month              integer               ,
    c_birth_year               integer                     ,
    c_birth_country            varchar(20)                 ,
    c_login                    char(13)                    ,
    c_email_address            char(50)                    ,
    c_last_review_date         char(10)
)
    SERVER gsmpp_server
    OPTIONS
(
    location 'gsfs://10.10.123.234:5000/customer1*.dat',
    FORMAT 'TEXT' ,
    DELIMITER '|',
    encoding 'utf8',
    mode 'Normal')
READ ONLY;
```

创建外表foreign_HR_staffS_ft，用来以TEXT格式导入GDS服务器192.168.0.90和192.168.0.91上的数据，导入过程错误信息将记录到err_HR_staffS中。本次数据导入允许出现的数据格式错误个数为2。

```
CREATE FOREIGN TABLE foreign_HR_staffS_ft
(
  staff_ID       NUMBER(6) ,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID  NUMBER(4)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*', format
'TEXT', delimiter E'\x08',  null '',reject_limit '2') WITH err_HR_staffS_ft;
```

建立外表，用来以CSV格式导入input_data目录下存放在各个节点名文件下的所有文件。

```
CREATE FOREIGN TABLE foreign_HR_staffS_ft1
(
  staff_ID       NUMBER(6) ,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
) SERVER gsmpp_server OPTIONS (location 'file:///input_data/*', format 'csv', quote E'\x08', mode 'private',
delimiter ',') WITH err_HR_staffS_ft1;
```

建立外表，用来以CSV格式导出数据到output_data目录下。

```
CREATE FOREIGN TABLE foreign_HR_staffS_ft2
(
  staff_ID       NUMBER(6) ,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
```

 版权所有 © 华为云计算技术有限公司

```
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID  NUMBER(4)
) SERVER gsmpp_server OPTIONS (location 'file:///output_data/', format 'csv', quote E'\x08', delimiter '|',
header 'on') WRITE ONLY;
```

使用 file_sequence 参数异常场景说明

使用CSV格式进行并行导入时，在如下示例场景中会因为CSV本身的规则和GDS拆分逻辑冲突而导致其中的某些分片导入失败。

场景：csv文件中包含未转义的换行符，且该换行符被quote指定的字符所包含，并且该行数据处于逻辑分片的第一行。

示例：并行导入一个文件big.csv ，正确导入显示内容如下：

```
--id, username, address
10001,"customer1 name","Rose District"
10002,"customer2 name","
23 Road Rose
District NewCity"
10003,"customer3 name","NewCity"
```

文件被拆分成两份后，第一个分片显示内容如下：

```
10001,"customer1 name","Rose District"
10002,"customer2 name","
23
```

第二个分片显示内容如下：

```
Road Rose
District NewCity"
10003,"customer3 name","NewCity"
```

因为第二个分片第一行后面的换行符包含在一个双引号之间，导致GDS无法分辨该换行符是字段中的换行符还是行中的分隔符，因此第一个分片会成功导入两条数据，第二个分片导入失败。

相关链接

ALTER FOREIGN TABLE (GDS导入导出)，**DROP FOREIGN TABLE**

21.13.4.11 ALTER FOREIGN TABLE (For HDFS or OBS)

功能描述

对HDFS外表和OBS外表进行修改。

外表（Foreign Table）是数据库或大数据平台中一种虚拟表结构，它不实际存储数据，而是通过元数据信息（如表结构、存储位置、文件格式等）将外部存储系统（如HDFS、OBS等）中的数据映射到数据库内部，使用户能够以标准SQL操作直接访问或操作外部数据。外表的本质是一种“数据代理”，将外部文件系统的数据逻辑映射成数据库可识别的表结构。

外表的核心价值是 “打破数据孤岛”： 通过元数据映射，让数据库用户以熟悉的方式访问外部系统数据，同时支持数据向外部系统的导出，实现跨系统数据的高效协同、分析与共享，尤其在大数据场景中能显著降低数据移动成本、提升协作效率。

版权所有 © 华为云计算技术有限公司

注意事项

无。

语法格式

- 设置外表属性。
```
ALTER FOREIGN TABLE [ IF EXISTS ]  table_name
   OPTIONS ( {[ ADD | SET | DROP ] option ['value']} [, ... ]);
```
- 设置外表的所有者。
```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
   OWNER TO new_owner;
```
- 更新外表列。
```
ALTER FOREIGN TABLE [ IF EXISTS ] table_name
   MODIFY ( { column_name data_type | column_name [ CONSTRAINT constraint_name ] NOT NULL
[ ENABLE ] | column_name [ CONSTRAINT constraint_name ] NULL } [, ...] );
```
- 修改外表的列。
```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
   action [, ... ];
```
其中action语法为：
```
ALTER [ COLUMN ] column_name [ SET DATA ] TYPE data_type
  | ALTER [ COLUMN ] column_name { SET | DROP } NOT NULL
  | ALTER [ COLUMN ] column_name SET STATISTICS [PERCENT] integer
  | ALTER [ COLUMN ] column_name OPTIONS ( {[ ADD | SET | DROP ] option ['value'] } [, ... ])
  | MODIFY column_name data_type
  | MODIFY column_name [ CONSTRAINT constraint_name ] NOT NULL [ ENABLE ]
  | MODIFY column_name [ CONSTRAINT constraint_name ] NULL
```
参考**ALTER TABLE**。
- 增加外表信息约束（Informational Constraint）。
```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
   ADD [ CONSTRAINT constraint_name ]
   { PRIMARY KEY | UNIQUE } ( column_name )
   [ NOT ENFORCED [ ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION ] |
ENFORCED ];
```
对于增加外表信息约束（Informational Constraint）相关参数请参考**参数说明**。
- 删除外表的信息约束（Informational Constraint）。
```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
   DROP CONSTRAINT constraint_name ;
```
- 设置外表行级访问控制开关（9.1.0及以上集群版本支持）。
```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
      ENABLE | DISABLE | FORCE | NO FORCE ROW LEVEL SECURITY
```

参数说明

表 21-187 ALTER FOREIGN TABLE (For HDFS or OBS)参数说明

参数	描述	取值范围
IF EXISTS	如果指定的外表不存在时返回一个提示而不是错误。	-
tablename	需要修改的外表名称。	已存在的外表名。
new_owner	外表的新所有者。	字符串，有效的用户名。
data_type	已存在字段的新数据类型。	字符串。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
constraint_name	要添加/删除的约束的名称。	字符串。
column_name	已存在字段的名称。	字符串。

修改外表语法中其他参数，请参见ALTER TABLE的**参数说明**。

示例

1. 创建HDFS_Server，对应的foreign data wrapper为HDFS_FDW或者DFS_FDW。
```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER HDFS_FDW OPTIONS (address
'10.10.0.100:25000,10.10.0.101:25000',hdfscfgpath '/opt/hadoop_client/HDFS/hadoop/etc/
hadoop',type'HDFS');
```
2. 创建不包含分区列的HDFS外表：
```
DROP FOREIGN TABLE IF EXISTS ft_region;
CREATE FOREIGN TABLE ft_region
(
    R_REGIONKEY INT4,
    R_NAME TEXT,
    R_COMMENT TEXT
)
SERVER
    hdfs_server
OPTIONS
(
    FORMAT 'orc',
    encoding 'utf8',
    FOLDERNAME '/user/hive/warehouse/gauss.db/region_orc11_64stripe/'
)
DISTRIBUTE BY
     roundrobin;
```
3. 修改外表ft_region的r_name字段类型为text。
```
ALTER FOREIGN TABLE ft_region ALTER r_name TYPE TEXT;
```
4. 标记外表ft_region的r_name列为not null。
```
ALTER FOREIGN TABLE ft_region ALTER r_name SET NOT NULL;
```

相关链接

CREATE FOREIGN TABLE (SQL on OBS or Hadoop)，**DROP FOREIGN TABLE**

21.13.4.12 ALTER FOREIGN TABLE (GDS 导入导出)

功能描述

对GDS外表进行修改。

注意事项

无。

语法格式

- 设置外表属性。

版权所有 © 华为云计算技术有限公司

```
ALTER FOREIGN TABLE [ IF EXISTS ]  table_name
    OPTIONS ( {[ ADD | SET | DROP ] option ['value']}[, ... ]);
```

- 设置外表新的所有者。

```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
    OWNER TO new_owner;
```

- 设置外表行级访问控制开关（9.1.0及以上集群版本支持）。

```
ALTER FOREIGN TABLE [ IF EXISTS ] tablename
        ENABLE | DISABLE | FORCE | NO FORCE ROW LEVEL SECURITY
```

参数说明

表 21-188 ALTER FOREIGN TABLE (GDS 导入导出)参数说明

参数	描述	取值范围
IF EXISTS	如果指定的索引不存在时返回一个提示而不是错误。	-
table_name	需要修改的外表名称。	已存在的外表名。
option	需要修改的option名称。	请参见CREATE FOREIGN TABLE的**参数说明**。
value	option的新值。	-
new_owner	外表的新所有者。	字符串，有效的用户名。

示例

创建外表customer_ft，用来以TEXT格式导入GDS服务器10.10.123.234上的数据：

```
DROP FOREIGN TABLE IF EXISTS customer_ft;
CREATE FOREIGN TABLE customer_ft
(
    c_customer_sk             integer               ,
    c_customer_id             char(16)              ,
    c_current_cdemo_sk        integer               ,
    c_current_hdemo_sk        integer               ,
    c_current_addr_sk         integer               ,
    c_first_shipto_date_sk    integer               ,
    c_first_sales_date_sk     integer               ,
    c_salutation              char(10)              ,
    c_first_name              char(20)              ,
    c_last_name               char(30)              ,
    c_preferred_cust_flag     char(1)               ,
    c_birth_day               integer               ,
    c_birth_month             integer               ,
    c_birth_year              integer               ,
    c_birth_country           varchar(20)           ,
    c_login                   char(13)              ,
    c_email_address           char(50)              ,
    c_last_review_date        char(10)
)
    SERVER gsmpp_server
    OPTIONS
(
    location 'gsfs://10.10.123.234:5000/customer1*.dat',
    FORMAT 'TEXT' ,
    DELIMITER '|',
```


版权所有 © 华为云计算技术有限公司

```
    encoding 'utf8',
    mode 'Normal')READ ONLY;
```

修改外表customer_ft属性，删除mode选项：

```
ALTER FOREIGN TABLE customer_ft options(drop mode);
```

相关链接

CREATE FOREIGN TABLE (GDS导入导出)，DROP FOREIGN TABLE

21.13.4.13 DROP FOREIGN TABLE

功能描述

删除指定的外表。

注意事项

DROP FOREIGN TABLE会强制删除指定的外表，删除外表后，依赖该外表的索引会被删除，而使用到该外表的函数和存储过程将无法执行。

语法格式

```
DROP FOREIGN TABLE [ IF EXISTS ]
    table_name [, ...] [ CASCADE | RESTRICT ];
```

参数说明

表 21-189 DROP FOREIGN TABLE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的外表不存在时返回一个提示而不是错误。	-
table_name	要删除的外表名称。	已存在的表名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖于外表的对象（比如视图）。 • RESTRICT：如果存在依赖对象，则拒绝删除该外表（缺省行为）。

示例

删除名称为customer_ft的外表。

```
DROP FOREIGN TABLE customer_ft;
```


版权所有 © 华为云计算技术有限公司

相关链接

ALTER FOREIGN TABLE (GDS导入导出)，ALTER FOREIGN TABLE (For HDFS or OBS)，CREATE FOREIGN TABLE (GDS导入导出)，CREATE FOREIGN TABLE (SQL on OBS or Hadoop)

21.13.4.14 CREATE SERVER

功能描述

创建一个外部服务器。

外部服务器是存储HDFS集群信息、OBS服务器信息、DLI连接信息或其他同构集群信息的载体。

注意事项

默认只有系统管理员才可以创建外部服务器，否则需要对所使用的FOREIGN DATA WRAPPER授权，授权语法为：

```
GRANT USAGE ON FOREIGN DATA WRAPPER fdw_name TO username;
```

其中fdw_name为FOREIGN DATA WRAPPER的名字，username为创建SERVER的用户名。

语法格式

```
CREATE SERVER server_name
    FOREIGN DATA WRAPPER fdw_name
    OPTIONS ( { option_name ' value ' } [, ...] ) ;
```

参数说明

表 21-190 CREATE SERVER 参数说明

参数	描述	取值范围
server_name	要创建的外部服务器的名称。服务器名称在数据库中必须唯一。	字符串，长度必须小于等于63。需符合标识符命名规范。
FOREIGN DATA WRAPPER fdw_name	指定外部数据封装器的名字。	fdw_name是数据库初始化时系统创建的数据封装器。 • 对于HDFS集群，fdw_name的名字可以是hdfs_fdw或者dfs_fdw。 • 对于其他同构集群，fdw_name为gc_fdw。对于数据导入导出场景，GDS外表使用gsmpp_server，fdw_name为dist_fdw。
OPTIONS	用于指定外部服务器的各类参数。	支持的各类参数参见表21-191。

版权所有 © 华为云计算技术有限公司

表 21-191 OPTIONS 参数说明

参数	描述	取值范围
address	指定的OBS服务终端节点或HDFS集群的IP地址。	OBS：OBS服务的终端节点（Endpoint）。 HDFS：HDFS集群的元数据节点（NameNode）所在的IP地址以及端口，或者同构其他集群的CN的IP地址以及端口。 为保证HA（High Availability），HDFS NameNode经常采用主备模式。主备NameNode的地址都需要加入到address值中。DWS访问HDFS服务时，会动态查找当前处于active状态的主NameNode。 若HDFS为联邦模式时，可将Router的地址都加入到address值中，DWS访问HDFS服务时，会动态随机查找当前处于active状态的Router。 **说明** ● address option必须存在，若用于跨集群互联互通场景则只允许设置1个。 ● 当server类型为DLI时，address为DLI服务上数据所存储的OBS address。 ● 若HDFS为联邦模式时，即fed 'rbf'，address可设置为多组IP、port，对应为HDFS Router的address。
hdfscfgpath	该参数仅支持type为HDFS时设置。 用户通过配置hdfscfgpath参数来指定HDFS配置文件路径。DWS会根据配置文件路径下的HDFS配置文件指定的连接配置方式，以及安全模式，来访问HDFS集群。非安全模式连接HDFS集群时，不支持数据传输加密。 如果没有指定address选项，默认采用hdfscfgpath指定的配置文件中指定的address。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
fed	表示dfs_fdw连接的是HDFS为联邦模式。	取值rbf，表示HDFS为联邦rbf方式。 **说明** 该参数8.1.2及以上版本支持。
encrypt	是否对数据进行加密，该参数仅支持type为OBS时设置。默认值为off。	• on表示对数据进行加密。 • off表示不对数据进行加密。
access_key	OBS访问协议对应的AK值（OBS云服务界面由用户获取），创建外表时AK值会保存到数据库的元数据表中。	• FOREIGN DATA WRAPPER为dfs_fdw时，仅支持type为OBS时设置。 • 8.2.0及以上版本集群，支持FOREIGN DATA WRAPPER为dist_fdw时指定该参数。
secret_access_key	OBS访问协议对应的SK值（OBS云服务界面由用户获取），创建外表时SK值会加密保存到数据库的元数据表中。	• FOREIGN DATA WRAPPER为dfs_fdw时，仅支持type为OBS时设置。 • 8.2.0及以上集群版本，支持FOREIGN DATA WRAPPER为dist_fdw时指定该参数。
security_token	对应统一身份认证服务IAM中临时安全凭证的SecurityToken的值，与临时AK、SK共同组成临时安全凭证，临时安全凭证有效期不超过24小时。该参数仅8.2.0及以上集群版本支持。	• FOREIGN DATA WRAPPER为dfs_fdw时，仅支持type为OBS时设置。 • 8.2.0及以上集群版本，支持FOREIGN DATA WRAPPER为dist_fdw时指定该参数。 • 使用该参数时，access_key和secret_access_key参数分别对应临时AK、SK。
type	表示dfs_fdw连接的类型。	• OBS表示连接的是OBS。 • HDFS表示连接的是HDFS。 • DLI表示连接的是DLI。
dli_address	DLI服务的终端节点，即endpoint。该参数仅支持type为DLI时设置。	-
dli_access_key	DLI访问协议对应的AK值（DLI云服务界面由用户获取），创建外表时AK值会保存到数据库的元数据表中。该参数仅支持type为DLI时设置。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
dli_secret_access_key	DLI访问协议对应的SK值（DLI云服务界面由用户获取），创建外表时SK值会加密保存到数据库的元数据表中。该参数仅支持type为DLI时设置。	-
dbname	用于协同分析、跨集群互联互通，表示将要连接的远端集群的数据库名字。	-
username	用于协同分析、跨集群互联互通，表示将要连接的远端集群的用户名。	-
password	用于协同分析、跨集群互联互通，表示将要连接的远端集群的用户名密码。 **说明** 对于云下集群迁移到云上的场景，从云下集群导出的server配置中密码为密文，由于云上和云下集群加解密的密钥不同，如果直接在云上集群执行导出时的CREATE SERVER，会执行失败，报解密失败的错误。这种场景下需要将CREATE SERVER中的password手动修改成明文密码进行配置。	-
syncsrv	仅用于跨集群互联互通，表示数据同步过程中使用到的GDS服务，设置方式与GDS外表的location属性相同。	-

示例

建立一个hdfs_server，其中hdfs_fdw为数据库中存在的foreign data wrapper。（address、hdfscfgpath的值仅供参考，请根据实际配置情况进行修改。）

```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER HDFS_FDW OPTIONS
  (address '10.10.0.100:25000,10.10.0.101:25000',
   hdfscfgpath '/opt/hadoop_client/HDFS/hadoop/etc/hadoop',
   type 'HDFS'
) ;
```

建立一个obs_server，其中dfs_fdw为数据库中存在的foreign data wrapper。

```
CREATE SERVER obs_server FOREIGN DATA WRAPPER DFS_FDW OPTIONS (
 address 'obs.example.com',
  access_key 'xxxxxxxxx',
 secret_access_key 'yyyyyyyyyyyyy',
 type 'obs'
);
```

建立一个dli_server，其中dfs_fdw为数据库中存在的foreign data wrapper。

```
CREATE SERVER dli_server FOREIGN DATA WRAPPER DFS_FDW OPTIONS (
 address 'obs.example.com',
```


版权所有 © 华为云计算技术有限公司

```
  access_key 'xxxxxxxxx',
  secret_access_key 'yyyyyyyyyyyyy',
  type 'dli',
  dli_address 'dli.example.com',
  dli_access_key 'xxxxxxxxx',
  dli_secret_access_key 'yyyyyyyyyyyyy'
);
```

建立另外一个同构集群的server，其中gc_fdw为数据库中存在的foreign data wrapper。

```
CREATE SERVER server_remote FOREIGN DATA WRAPPER GC_FDW OPTIONS
   (address '10.10.0.100:25000,10.10.0.101:25000',
  dbname 'test',
  username 'test',
  password 'xxxxxxxx'
);
```

建立一个FOREIGN DATA WRAPPER为dist_fdw的server，用于obs上文本数据的导入导出。

```
CREATE SERVER import_server FOREIGN DATA WRAPPER DIST_FDW OPTIONS
(
  access_key 'ak_string',
  secret_access_key 'sk_string'
);
```

相关链接

ALTER SERVER，**DROP SERVER**

21.13.4.15 ALTER SERVER

功能描述

增加、修改和删除外部服务器的定义。

现有外部服务器（server）可以使用pg_foreign_server系统表查询。

注意事项

- 只有服务器的所有者有权限执行ALTER SERVER命令，系统管理员默认拥有此权限。
- 要更改服务器的所有者，必须是外部服务器的OWNER并且也是新的所有者角色的直接或者间接成员，且必须对外部服务器的外部数据封装器有USAGE权限。

语法格式

- 修改外部服务的参数。
  ```
  ALTER SERVER server_name [ VERSION 'new_version' ]
      [ OPTIONS ( {[ ADD | SET | DROP ] option ['value']} [, ... ] ) ];
  ```
- 修改外部服务的所有者。
  ```
  ALTER SERVER server_name
      OWNER TO new_owner;
  ```
- 修改外部服务的名称。
  ```
  ALTER SERVER server_name
      RENAME TO new_name;
  ```
- 刷新HDFS配置文件。
  ```
  ALTER SERVER server_name REFRESH OPTIONS;
  ```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-192 ALTER SERVER 参数说明

参数	描述	取值范围
server_name	需要修改的外部服务的名称。	已存在的外部服务器名。
new_version	新的外部服务器版本。	-
OPTIONS	更改服务器的选项。ADD、SET和DROP指定要执行的操作。OPTIONS选项未指定时默认为ADD操作。 对于HDFS Server目前只支持SET操作，ADD/DROP操作现有版本不支持。语法中SET和DROP操作语法依然保留，以便后续扩展使用。	option和value为对应操作的参数可参考**表21-193**。
new_owner	外部服务器的新所有者。	有效的用户名。
new_name	外部服务器的新名称。	字符串，需符合**标识符命名规范**。
REFRESH OPTIONS	刷新HDFS的配置文件缓存信息，在配置文件有变动时执行，若不执行可能会访问报错。 该参数用于更新hdfs server配置缓存，不能与其相关的HDFS外表访问并发。	-

表 21-193 修改外部服务器所支持的 **OPTIONS** 参数说明

参数	描述	取值范围
address	OBS服务的终端节点（Endpoint）。 HDFS集群的主备节点所在的IP地址以及端口。 **说明** • 对于HDFS server，address必须存在，所以ADD和DROP操作不被允许。 • address目前只支持点分十进制格式的IPv4格式，且address字符串中不能出现空格，多组address以逗号作为分隔符。ip和port之间使用“:”来区分。HDFS集群中ip、port组对推荐设置两组，分别对应HDFS NameNode主备节点的address。 • 当server类型为DLI时，address为DLI服务上数据所存储的OBS address。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
hdfscfgpath	HDFS集群的配置文件。 **说明** ● 若HDFS为安全模式时，hdfscfgpath是必选项，否则为可选项。 ● 若设置hdfscfgpath时，path仅能设置一个。	-
fed	表示dfs_fdw连接的是HDFS为联邦模式。 取值rbf，表示HDFS为联邦rbf方式。 **说明** 该参数8.1.2及以上版本支持；8.0.0基线版本下，仅8.0.0.10及以上版本支持。	-
encrypt	是否对数据进行加密，该参数仅支持在type为OBS时设置。默认值为off。	● on表示对数据进行加密。 ● off表示不对数据进行加密。
access_key	OBS访问协议对应的AK值（OBS云服务界面由用户获取），创建外表时AK值会加密保存到数据库的元数据表中。该参数仅支持type为OBS时设置。	-
secret_access_key	OBS访问协议对应的SK值（OBS云服务界面由用户获取），创建外表时SK值会加密保存到数据库的元数据表中。该参数仅支持type为OBS时设置。	-
dli_address	DLI服务的终端节点，即endpoint。该参数仅支持type为DLI时设置。	-
dli_access_key	DLI访问协议对应的AK值（DLI云服务界面由用户获取），创建外表时AK值会保存到数据库的元数据表中。该参数仅支持type为DLI时设置。	-
dli_secret_access_key	DLI访问协议对应的SK值（DLI云服务界面由用户获取），创建外表时SK值会加密保存到数据库的元数据表中。该参数仅支持type为DLI时设置。	-
region	此参数表示OBS服务的IP地址或者域名信息。该参数仅支持type为OBS时设置。	-
dbname	用于协同分析、跨集群互联互通，表示将要连接的远端集群的数据库名称。	-
username	用于协同分析、跨集群互联互通，表示将要连接的远端集群的用户名。	-
password	用于协同分析、跨集群互联互通，表示将要连接的远端集群的用户名密码。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
syncsrv	仅用于跨集群互联互通，表示数据同步过程中使用到的GDS服务，设置方式与GDS外表的location属性相同。该参数仅8.1.2及以上版本支持。	-

示例

创建外部服务器hdfs_server，其中hdfs_fdw为数据库中存在的foreign data wrapper。

```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER HDFS_FDW OPTIONS
  (address '10.10.0.100:25000,10.10.0.101:25000',
   hdfscfgpath '/opt/hadoop_client/HDFS/hadoop/etc/hadoop',
   type 'HDFS'
) ;
```

修改外部服务器hdfs_server的address选项。

```
ALTER SERVER hdfs_server OPTIONS ( SET address '10.10.0.110:25000,10.10.0.120:25000');
```

修改外部服务器hdfs_server的hdfscfgpath选项。

```
ALTER SERVER hdfs_server OPTIONS ( SET hdfscfgpath '/opt/bigdata/hadoop');
```

相关链接

CREATE SERVER，**DROP SERVER**

21.13.4.16 DROP SERVER

功能描述

删除现有的一个数据服务器。

注意事项

只有server的owner才可以删除。

语法格式

```
DROP SERVER [ IF EXISTS ] server_name [ {CASCADE | RESTRICT} ] ;
```

参数描述

表 21-194 DROP SERVER 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的服务器不存在，则发出一个notice而不是抛出一个错误。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
server_name	要删除的服务器名称。	已存在的服务器名称。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖于server的对象 。 • RESTRICT（缺省值）：如果存在依赖对象，则拒绝删除该server。

示例

删除hdfs_server。

```
DROP SERVER hdfs_server;
```

相关链接

ALTER SERVER，**CREATE SERVER**

21.13.5 视图和物化视图

21.13.5.1 CREATE VIEW

功能描述

创建一个视图。

视图与基本表不同，是一个虚拟的表。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。

注意事项

无。

语法格式

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW view_name [ ( column_name [, ...] ) ]
    [ WITH ( {view_option_name [= view_option_value]} [, ... ] ) ]
    AS query;
```


版权所有 © 华为云计算技术有限公司

说明

- 创建视图时使用WITH(security_barriers)可以创建一个相对安全的视图，避免攻击者利用低成本函数的RAISE语句打印出隐藏的基表数据。
- GUC参数view_independent打开时，支持普通视图删除列。需注意，如果存在列级约束，不支持该列的删除。
- 视图解耦功能下不支持CTE重名。例如：

```
CREATE TABLE t1(a1 INT, b1 INT);
CREATE TABLE t2(a2 INT, b2 INT, c2 INT);
CREATE OR REPLACE VIEW v1 AS WITH tmp AS (SELECT * FROM t2) ,tmp1 AS (SELECT b2,c2
FROM tmp WHERE b2 = (WITH RECURSIVE tmp(aa, bb) AS (SELECT a1,b1 FROM t1) SELECT bb
FROM tmp WHERE aa = c2)) SELECT c2 FROM tmp1;
```

参数说明

表 21-195 CREATE VIEW 参数说明

参数	描述	取值范围
OR REPLACE	如果已存在相同名称的视图，则重新定义。	-
TEMP | TEMPORARY	创建临时视图。	-
view_name	要创建的视图名称。可以用模式修饰。	字符串，需符合**标识符命名规范**。
column_name	可选的名字列表，用作视图的字段名。如果没有给出，字段名取自查询中的字段名。	字符串，需符合**标识符命名规范**。
WITH ({view_option_name [= view_option_value]} [, ...])	该子句指定视图的可选参数。 目前view_option_name支持的参数仅有security_barrier，当VIEW试图提供行级安全时，应使用该参数。	security_barrier为boolean类型： • TRUE • FALSE
query	为视图提供行和列的SELECT或VALUES语句。	-

示例

创建字段spcname为pg_default组成的视图。

```
CREATE VIEW myView AS
    SELECT * FROM pg_tablespace WHERE spcname = 'pg_default';
```

对已存在视图myView进行重新定义，创建字段spcname为pg_global组成的视图。

```
CREATE OR REPLACE VIEW myView AS
    SELECT * FROM pg_tablespace WHERE spcname = 'pg_global';
```

创建一个由age小于12的内容组成的视图。

```
DROP TABLE IF EXISTS customer;
```

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE customer
(id int, name varchar(20) , age int)
with (orientation = column,COMPRESSION=MIDDLE)
distribute by hash(id);

INSERT INTO customer VALUES (1,'lily',10),(2, 'lucy',12),(3,'lilei',15);

CREATE VIEW customer_details_view_v1 AS
SELECT * FROM customer
WHERE age < 12;
```

可更新的视图

当开启视图可更新参数（enable_view_update）后，系统允许对简单视图使用INSERT，UPDATE、DELETE和MERGE INTO语句进行更新。（MERGE INTO语句更新仅8.1.2及以上版本支持）

满足以下所有条件的视图可进行更新：

- 视图定义的FROM语句中只能有一个普通表，不能是系统表、外表、dfs表、delta表、toast表、错误表。
- 视图中包含可更新的列，这些列是对基础表可更新列的简单引用。
- 视图定义不能包含WITH、DISTINCT、GROUP BY、ORDER BY、FOR UPDATE、FOR SHARE、HAVING、TABLESAMPLE、LIMIT、OFFSET子句。
- 视图定义不能包含UNION、INTERSECT、EXCEPT集合操作。
- 视图定义的选择列表不能包含聚集函数、窗口函数、返回集合的函数。
- 对于INSERT、UPDATE和DELETE语句，视图上不能有触发时机为INSTEAD OF的触发器。对于MERGE INTO语句，视图和基础表上都不能有触发器。
- 视图定义不能包含子链接。
- 视图定义不能包含属性为VOLATILE的函数（函数值可以在一次表扫描内改变的函数）
- 视图定义不能对表的分布键所在列起别名，或将普通列起别名为分布键列名。
- 视图更新操作中包含RETURNING子句时，视图定义中的列只能来自于基础表。

如果可更新的视图定义包含WHERE条件，则该条件将会限制UPDATE和DELETE语句修改基础表上的行。如果UPDATE语句更改行后不再满足WHERE条件，更新后通过视图将无法查询到。类似地如果INSERT命令插入了不满足WHERE条件的数据，插入后通过视图将无法查询到。在视图上执行插入、更新或删除的用户必须在视图和表上具有相应的插入、更新或删除权限。

相关链接

ALTER VIEW，**DROP VIEW**

21.13.5.2 ALTER VIEW

功能描述

ALTER VIEW更改视图的各种辅助属性。（如果用户是更改视图的查询定义，要使用CREATE OR REPLACE VIEW。）

版权所有 © 华为云计算技术有限公司

注意事项

- 用户必须是视图的所有者才可以执行ALTER VIEW。
- 要修改视图的模式，用户必须要有新模式的CREATE权限。
- 要修改视图的所有者，用户必须是新所属角色的直接或者间接的成员，并且此角色必须有视图模式的CREATE权限。
- 管理员用户可以更改任何视图的所属关系。

语法格式

- 设置视图列的默认值。
```
ALTER VIEW [ IF EXISTS ] view_name
    ALTER [ COLUMN ] column_name SET DEFAULT expression;
```
- 取消列视图列的默认值。
```
ALTER VIEW [ IF EXISTS ] view_name
    ALTER [ COLUMN ] column_name DROP DEFAULT;
```
- 修改视图的所有者。
```
ALTER VIEW [ IF EXISTS ] view_name
    OWNER TO new_owner;
```
- 重命名视图，支持新视图名前带有原视图所在的schema名，不支持同时修改schema名。
```
ALTER VIEW [ IF EXISTS ] view_name
    RENAME TO new_name;
ALTER VIEW [ IF EXISTS ] view_name
    RENAME TO schema.new_name;
```
- 设置视图的所属模式。
```
ALTER VIEW [ IF EXISTS ] view_name
    SET SCHEMA new_schema;
```
- 设置视图的选项。
```
ALTER VIEW [ IF EXISTS ] view_name
    SET ( { view_option_name [ = view_option_value ] } [, ... ] );
```
- 重置视图的选项。
```
ALTER VIEW [ IF EXISTS ] view_name
    RESET ( view_option_name [, ... ] );
```
- 重建本视图及上层和下层依赖的无效视图。
```
ALTER VIEW [ IF EXISTS ] view_name
    REBUILD;
```
- 重建本视图及下层依赖的视图。
```
ALTER VIEW [ IF EXISTS ] ONLY view_name
    REBUILD;
```

参数说明

表 21-196 ALTER VIEW 参数说明

参数	描述	取值范围
IF EXISTS	如果视图不存在时返回一个提示而不是错误。	-
view_name	需要修改的视图的名称，可以用模式修饰。	有效的视图名。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
column_name	可选的名称列表，视图的字段名。如果没有给出，字段名取自查询中的字段名。	有效的字段名。
SET/DROP DEFAULT	设置或删除一个列的缺省值，该参数暂无实际意义。	-
new_owner	视图的新所有者。	有效的用户名。
new_name	视图的新名称。	字符串，需符合**标识符命名规范**。
new_schema	视图的新模式。	有效的模式名。
view_option_name [= view_option_value]	该子句为视图指定一个可选的参数。 目前view_option_name支持的参数仅有security_barrier，当VIEW试图提供行级安全时，应使用该参数。	security_barrier为boolean类型： • TRUE • FALSE
REBUILD	该子句用于视图解耦，可使用已保存的原始语句重新创建视图，恢复依赖关系。9.1.0.200及以上集群版本中，视图解耦功能无需打开view_independent开关，检测视图如果为无效状态，则可以进行重建。	REBUILD的详细使用说明请参见**REBUILD使用说明**。
ONLY	控制视图重建的范围，只重建视图及其所依赖的视图。此功能无需打开view_independent开关，检测视图如果为无效状态，则可以进行重建。	-

REBUILD 使用说明

- 重建视图会从当前视图开始，依次向后级联刷新与其关联的所有视图，如果其依赖的前向视图也为不可用状态，会触发自动重建。
- 不支持对有依赖关系的临时表及临时视图的解耦DROP，可以对没有依赖关系的临时视图进行REBUILD操作。
- 支持视图模式名称及视图名称的修改，REBUILD按照最新的名称重建，但是query部分保留原始定义。
- 基表字段类型仅支持大类（字符型、数字型、时间类型等）范围内的修改；当基表添加字段时，视图不会置为无效，且定义不变。
- 无效视图备份时以注释形式导出，恢复时需要自行手动处理。

版权所有 © 华为云计算技术有限公司

说明

以下场景会触发上层级联视图无效：

- DROP TABLE/VIEW
- RENAME TABLE/VIEW
- ALTER TABLE DROP COLUMN
- ALTER TABLE CHANGE/ALTER COLUMN TYPE
- ALTER TABLE CHANGE/ALTER COLUMN NAME
- ALTER TABLE/VIEW NAMESPACE
- ALTER TABLE/VIEW RENAME

示例

创建示例视图myview。

```
CREATE OR REPLACE VIEW myview AS
    SELECT * FROM pg_tablespace WHERE spcname = 'pg_default';
```

修改视图名称。

```
ALTER VIEW myview RENAME TO product_view;
```

修改视图所属schema。

```
ALTER VIEW product_view SET schema public;
```

重建视图。

```
ALTER VIEW public.product_view REBUILD;
```

重建依赖视图。

```
ALTER VIEW ONLY public.product_view REBUILD;
```

相关链接

CREATE VIEW，**DROP VIEW**

21.13.5.3 DROP VIEW

功能描述

数据库中强制删除已有的视图。

注意事项

只有视图的所有者有权限执行DROP VIEW的命令，系统管理员默认拥有此权限。

危险

- **DROP对象**操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除，操作需谨慎，操作前需考虑数据备份。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

版权所有 © 华为云计算技术有限公司

语法格式

```
DROP VIEW [ IF EXISTS ] view_name [, ...] [ CASCADE | RESTRICT ];
```

参数说明

表 21-197 DROP VIEW 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的视图不存在，则发出一个notice而不是抛出一个错误。	-
view_name	要删除的视图名字。	已存在的视图。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖此视图的对象（比如其他视图）。 • RESTRICT：如果有依赖对象存在，则拒绝删除此视图。此选项为缺省值。

示例

删除视图myView。

```
DROP VIEW myView;
```

删除视图customer_details_view_v2。

```
DROP VIEW public.customer_details_view_v2;
```

相关链接

ALTER VIEW，**CREATE VIEW**

21.13.5.4 CREATE MATERIALIZED VIEW

功能描述

创建新的物化视图。物化视图语法仅8.2.1.220及以上集群支持。

物化视图（Materialized View）是数据库中一种特殊类型的数据库对象。它将复杂计算的查询结果预计算并存储在数据库中，实现加速查询的目的。

适用于频繁对大量数据进行复杂的查询和分析场景，使用物化视图可以预先计算和存储该场景的查询结果，显著提高查询性能。

物化视图权限要求

1. 创建权限：创建物化视图时需要schema的CREATE权限和基表或列的SELECT权限。
2. 查询权限：查询物化视图需要物化视图的SELECT权限。

版权所有 © 华为云计算技术有限公司

3. 刷新权限：刷新需要物化视图的INSERT和基表或列的SELECT权限。
4. 其他权限：

   物化视图支持ANALYZE/VACUUM/ALTER/DROP等细粒度权限。

   物化视图支持with grant option的权限传递操作。
5. 安全管理：

   物化视图不支持更高级别的安全控制，如果基表存在行级访问控制、脱敏策略或owner为私有用户等限制SELECT权限的场景，则禁止创建物化视图；

   如果已存在物化视图，基表增加RLS、脱敏策略或修改owner为私有用户，则物化视图可以执行查询，但无法刷新。

注意事项

- 使用CREATE MATERIALIZED VIEW语句需确保enable_matview设置为on。
- 物化视图的基表可以是行存表、列存表、hstore表、冷热表（9.1.0.200及以上集群版本支持）、分区表或者指定某个分区、外表、其他物化视图。
- 不支持包含临时表（包括全局临时表、volatile临时表和普通临时表），不支持对自动分区表指定分区。
- 不支持物化视图定义中的CTE包含除SELECT语句以外的其他DML。
- 不支持基表上存在脱敏策略和行级访问控制或者基表属于私有用户。脱敏策略和行级访问控制可能会导致结果集错误，而私有用户可能导致刷新物化视图失败等问题。
- 物化视图不支持的存储类型：外表、时序表。
- 物化视图禁止INSERT/UPDATE/MERGE INTO/DELETE对数据进行修改。
- 物化视图执行一次将结果并保存，每次查询结果是一致的。BUILD IMMEDIATE或REFRESH后，物化视图可以查询出正确结果。
- 物化视图不能通过语法指定Node Group。支持物化视图的基表指定Node Group创建，物化视图会继承基表NodeGroup信息创建，需要多个基表的NodeGroup相同。

语法格式

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name [ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-198 CREATE MATERIALIZED VIEW 参数说明

参数	描述	取值范围/默认值
BUILD DEFERRED | IMMEDIATE	IMMEDIATE，表示创建物化视图时即包含最新数据。 DEFERRED，表示创建物化视图时需要等到第一次refresh时才会包含数据。	-
REFRESH	指定物化视图的刷新方式。 创建物化视图后，物化视图中的数据只反映创建时刻基表的状态。当基表中的数据发生变化时，需要通过刷新物化视图（REFRESH MATERIALIZED VIEW）更新物化视图中的数据。	• 刷新方式：DEFAULT默认刷新，RESTRICT 强制刷新。仅创建后台异步刷新的物化视图可以指定。 • 刷新方向：级联嵌套使用物化视图时的刷新方向，FORWARD向上刷新，BACKWARD向下刷新，ALL全部都刷。仅创建后台异步刷新的物化视图可以指定。具体参考**REFRESH MATERIALIZED VIEW**。 • 目前只支持COMPLETE全量刷新这一种刷新方式。执行物化视图定义的查询语句并更新物化视图。 • 刷新触发方式。 ON DEMAND：手动按需刷新。 START WITH (timestamptz) | EVERY (interval)：定时自动刷新。START WITH指定首次刷新时间，EVERY 指定刷新间隔，根据指定的时间定时自动刷新，支持MONTH、DAY、HOUR、MINUTE及SECOND。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围/默认值
ENABLE | DISABLE QUERY REWRITE	是否开启查询重写。 查询重写是指在对基表进行查询时， 如果基表上创建有物化视图，数据库系统自动判断是否可以使用物化视图中的预计算结果处理查询。 如果可以使用某个物化视图，会直接从该物化视图读取预计算结果，起到加速查询的作用。	默认关闭。 指定ENABLE QUERY REWRITE时，需要设置GUC参数**mv_rewrite_rule**才能启用物化视图查询重写功能。
WITH	创建物化视图指定的相关参数。	参见**表21-199**。
DISTRIBUTE BY	指定表如何在节点之间分布或者复制。	REPLICATION、ROUNDROBIN或HASH： ● REPLICATION：表的每一行存在所有数据节点（DN）中，即每个数据节点都有完整的表数据。 ● ROUNDROBIN：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。（ROUNDROBIN仅8.1.2及以上版本支持） ● HASH：对指定的列进行Hash，通过映射，把数据分布到指定DN。 默认值：由参数default_distribution_mode决定。 **说明** 当物化视图指定为hash分布时，可能存在数据倾斜。可使用普通表检查数据是否倾斜的方法对物化视图进行检查，详细内容请参考《数据仓库服务开发指南》的“查看数据倾斜状态”章节。若物化视图的数据存储存在倾斜时，可参考《数据仓库服务开发指南》的“数据倾斜调优”章节进行存储层的倾斜调优。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围/默认值
PARTITION BY	创建分区物化视图语法仅9.1.0.200及以上集群版本支持。 **注意** ● 物化视图的分区键，必须是被查询表中某个分区表的分区键，用于建立按分区失效和刷新的对应关系。 ● 物化视图的分区键，必须包含在查询的最外层SQL中，以便将查询数据按分区键刷入物化视图表。 ● 物化视图只支持range分区。 ● 被查询表可以是组合分区，如果被查询表有多个分区表且分区条件都一样，都会与物化视图自动建立映射关系。 ● 不支持外表的分区映射，因为无法感知外表的数据变化。 ● 不支持Outer join中会产生空值的列做分区键的情况。 ● 不支持基表分区的merge和split。	● 等比例关联，指定查询中某个分区表的分区键作为物化视图的分区键，并建立一比一的映射关系。 ● 上卷式关联，通过date_trunc函数对查询中某个分区表的分区键进行上卷，并建立上卷式的映射关系。要求数据类型必须是date_trunc函数可接受的日期类型。
AS query	基于query的结果创建物化视图。	-

版权所有 © 华为云计算技术有限公司

表 21-199 WITH 参数说明

参数	描述	取值范围
ORIENTATION	指定表数据的存储方式，即行存方式、列存方式，该参数设置成功后就不再支持修改。	ROW或COLUMN： ● ROW，表示表的数据将以行式存储。 行存储适合于OLTP业务，此类型的表上交互事务比较多，一次交互会涉及表中的多个列，用行存查询效率较高。 ● COLUMN，表示表的数据将以列式存储。 列存储适合于数据仓库业务，此类型的表上会做大量的汇聚计算，且涉及的列操作较少。 默认值：ROW
enable_foreign_table_query_rewrite	指定是否允许包含外表的物化视图进行查询重写，需要与ENABLE QUERY REWRITE一起使用。 外表数据有变化，物化视图无法感知。如果需要对包含外表的物化视图使用查询重写功能，需要指定此选项。	● on，允许包含外表的物化视图进行查询重写。 ● off，不允许包含外表的物化视图进行查询重写。 默认值：off
bitmap_columns	bitmap index只适用于hstore_opt表，只有开启表级参数enable_hstore_opt且开启bitmap_columns='指定列'。该参数仅9.1.0.200及以上集群版本支持。	-
secondary_part_num	指定列存表二级分区的数量，仅适用于HStore列存表。该参数仅9.1.0.200及以上集群版本支持。	1~32。 默认值：8
enable_hstore_opt	enable_hstore_opt表级参数打开时会默认同时打开enable_hstore表级参数。该参数仅9.1.0.200及以上集群版本支持。	true或者false。 默认值：false
enable_turbo_store	用于控制是否创建为turbo表（基于列存表实现）。该参数只对列存表有效。该参数仅9.1.0.200及以上集群版本支持。	on或者off。 默认值：off

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
mv_analyze_mode	控制物化视图自动analyze的方式。该参数仅9.1.0.200及以上集群版本支持。	● none，表示物化视图在刷新后不自动执行analyze。 ● light，表示物化视图在刷新后执行light analyze。 默认值：light
mv_pck_column	物化视图的局部聚簇存储，列存表导入数据时按照指定的列（单列或多列），进行局部排序。该参数仅9.1.0.200及以上集群版本支持。 开启mv_pck_column='指定列'。	-
mv_support_function_type	开启物化视图创建时查询语句中可以使用的函数属性。该参数仅9.1.0.200及以上集群版本支持。	● stable，表示支持在查询语句中使用类型为STABLE和IMMUTABLE的函数。 ● volatile，表示支持在查询语句中使用类型为VOLATILE，STABLE和IMMUTABLE的函数。 默认值：空
excluded_inactive_tables	指定的基表发生数据变化不会失效该物化视图。该参数仅9.1.0.200及以上集群版本支持。 设置方式为：excluded_inactive_tables='schemaName1.tableName1,schemaName2.tableName2'	默认值：空
force_rewrite_timeout	在刷新后的指定时间间隔内，可以查询重写，无论此物化视图数据是否新鲜。该参数仅9.1.0.200及以上集群版本支持。	单位为秒，默认值：0。
TABLESPACE tablespace_name	声明一个表空间。用于V3存储格式的创建，如果default_tablespace为空，将使用数据库的缺省表空间。该参数仅9.1.0.200及以上集群版本支持。	-

示例

创建基表，并向基表插入数据。

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE t1 (a int, b int) DISTRIBUTE BY HASH(a);
INSERT INTO t1 SELECT x,x FROM generate_series(1,10) x;
```

创建默认BUILD IMMEDIATE方式的物化视图：

```
CREATE MATERIALIZED VIEW mv1 AS SELECT * FROM t1;
```

创建指定列存方式的物化视图。

```
CREATE MATERIALIZED VIEW mv2 WITH(orientation = column) AS SELECT * FROM t1;
```

创建手动按需刷新的物化视图。

```
CREATE MATERIALIZED VIEW mv3 BUILD DEFERRED REFRESH ON DEMAND AS SELECT * FROM t1;
```

创建指定刷新时间的物化视图。

```
CREATE MATERIALIZED VIEW mv4 BUILD DEFERRED REFRESH START WITH(trunc(sysdate)) EVERY
(interval '1 day') AS SELECT * FROM t1;
```

创建带有bitmap index的物化视图。

```
CREATE MATERIALIZED VIEW mv1
with (ORIENTATION = COLUMN, enable_hstore=true, enable_hstore_opt=on, bitmap_columns='col1')  AS
SELECT * FROM base_table;
```

创建物化视图并指定列存表二级分区的数量。

```
CREATE MATERIALIZED VIEW mv
WITH (ORIENTATION=COLUMN, ENABLE_HSTORE=ON, enable_hstore_opt=on, mv_pck_column='c3',
secondary_part_column = 'c2', secondary_part_num = 8)  AS SELECT * FROM base_table;
```

创建物化视图并指定pck列进行排序。

```
CREATE MATERIALIZED VIEW mv
WITH (ORIENTATION=COLUMN, ENABLE_HSTORE=ON, enable_hstore_opt=on, mv_pck_column='col3')  AS
SELECT * FROM base_table;
```

创建物化视图并指定其analyze的方式。

```
CREATE MATERIALIZED VIEW mv1 enable query rewrite
with(excluded_inactive_tables='matview_basic."T1",matview_basic."a=b"',mv_analyze_mode='none') as
SELECT * FROM base_table;
```

创建V3物化视图：

```
CREATE MATERIALIZED VIEW mv1
with (orientation=column, enable_hstore=true, compression=low, enable_hstore_opt=on, COLVERSION =
3.0) TABLESPACE cu_obs_tbs distribute by hash(scope_name)
AS SELECT * FROM dicttbl_low;
```

创建包含外表的物化视图进行查询重写。

```
CREATE MATERIALIZED VIEW mv1 with (enable_foreign_table_query_rewrite = true) as SELECT * FROM
base_table;
```

创建物化视图指定查询语句中可以使用volatile函数。

```
CREATE MATERIALIZED VIEW mv_date with(mv_support_function_type = 'volatile') as select to_date(a)
from t_date;
```

相关链接

ALTER MATERIALIZED VIEW，**DROP MATERIALIZED VIEW**，**REFRESH MATERIALIZED VIEW**

 版权所有 © 华为云计算技术有限公司

21.13.5.5 ALTER MATERIALIZED VIEW

功能描述

修改物化视图属性。物化视图语法仅8.2.1.220及以上集群支持。

注意事项

使用ALTER MATERIALIZED VIEW语句需确保enable_matview设置为on。

语法格式

- 设置是否对物化视图启动查询重写。
```
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    [ ENABLE | DISABLE ] QUERY REWRITE;
```
- 修改物化视图的刷新方式。目前仅支持COMPLETE全量刷新这一种刷新方式。执行物化视图定义的查询语句并更新物化视图。
```
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] |
[ CASCADE ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY
(interval) ] ];
```
  - REFRESH ON DEMAND，表示手动按需刷新。
  - START WITH，指定首次刷新时间。
  - EVERY指定刷新间隔，根据指定的时间定时刷新，支持MONTH、DAY、HOUR、MINUTE及SECOND。
- 修改后台异步刷新物化视图的刷新方式，仅针对后台异步刷新的物化视图。该语法仅9.1.0.200及以上集群版本支持。具体内容参考**REFRESH MATERIALIZED VIEW**。
```
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    REFRESH [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] |
[ CASCADE ALL ] ;
```
- 修改物化视图的所有者。
```
ALTER MATERIALIZED VIEW { materialized_view_name }
    OWNER TO new_owner;
```
- 设置物化视图的表属性。该语法仅9.1.0.200及以上集群版本支持。
```
ALTER MATERIALIZED VIEW { materialized_view_name }
    SET ( {storage_parameter = value} [, ... ] )
    | RESET ( storage_parameter [, ... ] )
```
- 重命名物化视图。不支持跨schema对物化视图进行重命名操作。重命名物化视图时原物化视图名materialized_view_name可指定模式名，new_materialized_view_name不能指定模式名。
```
ALTER MATERIALIZED VIEW [ IF EXISTS ] materialized_view_name
    RENAME TO new_materialized_view_name;
```

参数说明

表 21-200 ALTER MATERIALIZED VIEW 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的物化视图不存在时，返回一个提示而不是错误。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
materialized_view_name	需要修改的物化视图的名称。	已存在的物化视图名。
ENABLE | DISABLE QUERY REWRITE	是否开启查询重写。	默认关闭。 指定ENABLE QUERY REWRITE时，需要设置GUC参数**mv_rewrite_rule**才能启用物化视图查询重写功能。
REFRESH	修改物化视图的刷新方式。 当基表中的数据发生变化时，需要通过刷新物化视图（REFRESH MATERIALIZED VIEW）更新物化视图中的数据。	• 刷新方式：DEFAULT默认刷新，RESTRICT 强制刷新。仅创建后台异步刷新的物化视图可以指定。 • 刷新方向：级联嵌套使用物化视图时的刷新方向，FORWARD向上刷新，BACKWARD向下刷新，ALL 全部都刷。仅创建后台异步刷新的物化视图可以指定。具体参考**REFRESH MATERIALIZED VIEW**。 • 目前只支持COMPLETE全量刷新这一种刷新方式。执行物化视图定义的查询语句并更新物化视图。 • 刷新触发方式。 ON DEMAND：手动按需刷新。 START WITH (timestamptz) | EVERY (interval)：定时刷新。START WITH指定首次刷新时间，EVERY 指定刷新间隔，根据指定的时间定时刷新，支持MONTH、DAY、HOUR、MINUTE及SECOND。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
storage_parameter	设置物化视图的表属性。 支持设置参数： mv_pck_column， bitmap_columns， enable_foreign_table_query_rewrite， excluded_inactive_tables， force_rewrite_timeout， mv_analyze_mode等。	更多支持设置的参数可参考**参数说明**。
new_owner	物化视图的新所有者。	已存在的用户名。
new_materialized_view_name	新的物化视图名称。	字符串，需符合**标识符命名规范**。

示例

启用物化视图的查询重写。

```
ALTER MATERIALIZED VIEW mv1 ENABLE QUERY REWRITE;
NOTICE:  REFRESH MATERIALIZED VIEW should be executed to enable query rewrite.
ALTER MATERIALIZED VIEW
```

修改物化视图的表属性。

```
ALTER MATERIALIZED VIEW mv1 SET (force_rewrite_timeout=100);
ALTER MATERIALIZED VIEW mv1 SET (mv_pck_column='col1');
ALTER MATERIALIZED VIEW mv1 SET (enable_foreign_table_query_rewrite = true);
```

修改物化视图的刷新时间。

```
ALTER MATERIALIZED VIEW mv1 REFRESH START WITH('2025-01-01 15:15:15'::timestamptz) EVERY
(interval '60 s');
```

相关链接

CREATE MATERIALIZED VIEW，**DROP MATERIALIZED VIEW**，**REFRESH MATERIALIZED VIEW**

21.13.5.6 REFRESH MATERIALIZED VIEW

功能描述

刷新物化视图。该语法仅8.2.1.220及以上集群支持。

刷新方式由**CREATE MATERIALIZED VIEW**语法中的REFRESH参数指定。目前支持全量刷新或者定时刷新。

注意事项

刷新操作会阻塞基表的DML操作。

版权所有 © 华为云计算技术有限公司

语法格式

```
REFRESH MATERIALIZED VIEW
{[schema.]materialized_view_name} [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE ALL ] ]
```

参数说明

表 21-201 REFRESH MATERIALIZED VIEW 参数说明

参数	描述	取值范围
materialized_view_name	需要刷新的物化视图的名称。	-
DEFAULT	采用默认的刷新方式。即刷新单个物化视图，若基表未发生变化，不进行刷新操作。	-
RESTRICT	采用强制刷新。不关注基表是否变化，一定会执行刷新操作。	-
CASCADE FORWARD	级联向上刷新。先刷新当前物化视图，再依次刷新依赖此物化视图的嵌套物化视图。刷新方式为默认刷新方式。	-
CASCADE BACKWARD	级联向下刷新。先刷新当前物化视图所依赖的最底层物化视图，再依次刷新底层物化视图依赖的嵌套物化视图，最后刷新当前物化视图。刷新方式为默认刷新方式。	-
CASCADE ALL	级联刷新全部。先执行级联向下刷新，然后执行级联向上刷新。	-

示例

刷新物化视图。

```
REFRESH MATERIALIZED VIEW mv1;
```

级联向下刷新物化视图。

```
CREATE TABLE t1(a int,b int);
CREATE MATERIALIZED VIEW mv1 AS SELECT * FROM t1 where a > 0;
CREATE MATERIALIZED VIEW mv2 AS SELECT * FROM mv1;
INSERT INTO t1 values(1,1),(2,2);
REFRESH MATERIALIZED VIEW mv2 cascade backward; -- 刷新完mv2, mv1数据也进行了更新

SELECT * FROM mv1 ORDER BY 1,2;
 a | b
---+---
 1 | 1
 2 | 2
(2 rows)

SELECT * FROM mv2 ORDER BY 1,2;
 a | b
---+---
```


版权所有 © 华为云计算技术有限公司

```
 1 | 1
 2 | 2
(2 rows)
```

相关链接

CREATE MATERIALIZED VIEW，**ALTER MATERIALIZED VIEW**，**DROP MATERIALIZED VIEW**

21.13.5.7 DROP MATERIALIZED VIEW

功能描述

删除物化视图。物化视图语法仅8.2.1.220及以上集群支持。

注意事项

无

语法格式

```
DROP MATERIALIZED VIEW [ IF EXISTS ]
{[schema.]materialized_view_name} [, ...] [ CASCADE | RESTRICT ];
```

参数说明

表 21-202 DROP MATERIALIZED VIEW 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的物化视图不存在，则发出一个notice而不是抛出一个错误。	-
materialized_view_name	需要删除的物化视图名称。	已存在的物化视图名称。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：自动删除包含在模式中的对象。 • RESTRICT：如果模式包含任何对象，则删除失败（缺省行为）。

示例

删除物化视图。

```
DROP MATERIALIZED VIEW mv1;
```

相关链接

ALTER MATERIALIZED VIEW，**CREATE MATERIALIZED VIEW**，**REFRESH MATERIALIZED VIEW**

版权所有 © 华为云计算技术有限公司

21.13.6 索引&序列&文件目录

21.13.6.1 CREATE INDEX

功能描述

在指定的表上创建索引。**索引（Index）**是一种通过优化数据检索效率的数据结构，其核心作用类似于书籍的目录——通过预先组织关键信息，帮助数据库快速定位所需数据，减少查询时需要扫描的数据量。

索引的核心作用是：

1. **加速查询速度：**

   在处理大量数据的数据库应用中，用户经常遇到查询速度慢的问题，尤其是在没有索引的情况下，数据库查询需要扫描整个表（全表扫描），效率极低。如何提高查询效率，减少数据处理时间？通过建立索引，可以显著提升查询速度。有索引时，可通过索引直接定位到符合条件的数据位置，大幅减少数据扫描量。例如：查询where id=100时，若id列有索引，能直接找到该记录，无需遍历所有行。

2. **优化排序和分组操作：**

   索引本身是有序的（如 Btree索引），当查询涉及ORDER BY（排序）或GROUP BY（分组）时，可直接利用索引的有序性，避免额外的排序操作，提升效率。

3. **约束数据完整性**：

   某些索引（如主键索引、唯一索引）可强制列的值唯一（如主键不能重复），辅助维护数据的完整性。

DWS 索引支持的类型

DWS支持创建的索引类型包括：

- **B-Tree**：B-Tree索引使用一种类似于B+树的结构来存储数据的键值，通过这种结构能够快速的查找索引。B-Tree适合范围查询、排序、等值查询。
- **GIN**：GIN索引是倒排索引，可以处理包含多个键的值（比如数组），适用场景：多值元素、模糊匹配、全文检索。
- **GiST**：GiST索引适用于几何和地理等多维数据类型和集合数据类型，适用场景：多维数据、空间搜索、复杂范围。
- **PSORT**：PSORT索引是针对列存表进行局部排序索引。

其中，

- **行存表**支持：**B-Tree**（行存表缺省值）、**GIN**、**GiST**。
- **列存表**支持：**PSORT**（列存表缺省值）、**B-Tree**、**GIN**。

从**分区表**的角度分析，又分为本地索引（LOCAL INDEX），全局索引（GLOBAL INDEX）。

- **本地索引**（LOCAL INDEX）：也叫分区索引，即在分区表的每个区上单独创建索引，本地索引与分区表的分区一一对应，具有与其表相同的分区数和相同的分区范围。当表分区发生变化时，只需维护对应的索引分区。
- **全局索引**（GLOBAL INDEX）：全局索引是跨分区的全局索引，索引结构独立于基表的分区方式，覆盖整个表的所有分区数据。目前DWS不支持全局索引。

版权所有 © 华为云计算技术有限公司

DWS 支持的索引方式

表 21-203 索引方式

索引方式	描述
唯一索引	可用于约束索引属性值的唯一性，或者属性组合值的唯一性。如果一个表声明了唯一约束或者主键，则DWS自动在组成主键或唯一约束的字段上创建唯一索引（可能是多字段索引），以实现这些约束。目前，DWS只有B-Tree可以创建唯一索引。
多字段索引（或复合索引、联合索引）	一个索引可以定义在表中的多个属性上。目前，DWS中的B-Tree支持多字段索引，且最多可在32个字段上创建索引。
部分索引	建立在一个表的子集上的索引，这种索引方式只包含满足条件表达式的元组。
表达式索引	索引建立在一个函数或者从表中一个或多个属性计算出来的表达式上。表达式索引只有在查询时使用与创建时相同的表达式才会起作用。

创建索引的原则

索引可以用来提高数据库查询性能，但是不恰当的使用将导致数据库性能下降。建议仅在匹配如下某条原则时创建索引：

- 经常执行查询的字段。
- 在连接条件上创建索引，对于存在多字段连接的查询，建议在这些字段上建立组合索引。例如，select * from t1 join t2 on t1.a=t2.a and t1.b=t2.b，可以在t1表上的a，b字段上建立组合索引。
- where子句的过滤条件字段上（尤其是范围条件）。
- 在经常出现在order by、group by和distinct后的字段。
- 对于点查询场景，推荐建立B-Tree索引。

在分区表上创建索引与在普通表上创建索引的语法不太相同，使用时请注意，如分区表上不支持并行创建索引、不支持创建部分索引、不支持NULL FIRST特性。

注意事项

- 索引自身也占用存储空间、消耗计算资源，创建过多的索引将对数据库性能造成负面影响（尤其影响数据导入的性能，建议在数据导入后再建索引）。因此，仅在必要时创建索引。
- 索引定义里的所有函数和操作符都必须是immutable类型的，即结果只依赖于其输入参数，而不受任何外部的影响（如另外一个表的内容或者当前时间），该限制可以确保该索引的行为是定义良好的。在一个索引上或WHERE语句中使用用户定义函数，请将其标记为immutable类型函数。
- 在分区表上创建索引时，索引项中必须包含分布列和所有分区键。
- DWS在分区表上创建索引时只支持本地索引（LOCAL INDEX），不支持全局索引（GLOBAL INDEX）。

版权所有 © 华为云计算技术有限公司

- 列存表和HDFS表支持B-Tree索引，不支持创建表达式索引、部分索引。
- 列存表支持通过B-Tree索引建立唯一索引。
- 列存表和HDFS表支持的PSORT索引不支持创建表达式索引、部分索引和唯一索引。
- 列存表支持的GIN索引支持创建表达式索引，但表达式不能包含空分词、空列和多列，不支持创建部分索引和唯一索引。
- 列存索引不支持OR查询过滤条件及inlist场景。
- roundrobin表不支持创建主键/唯一索引。
- 对表执行CREATE INDEX或REINDEX操作时会触发索引重建（索引重建过程中会先把数据转储到一个新的数据文件中，重建完成之后会删除原始文件），当表比较大时，重建会消耗较多的磁盘空间。当磁盘空间不足时，要谨慎对待大表CREATE INDEX或REINDEX操作，防止触发集群只读。

警告

- 针对有大批量数据增删改的表，索引个数建议控制在3个以内，最多不超过5个。
- 避免在业务高峰期对大表执行CREATE INDEX和REINDEX操作。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

- 在表上创建索引。

```
CREATE [ UNIQUE ] INDEX [ [IF NOT EXISTS] [ schema_name. ] index_name ] ON table_name
[ USING method ]
    ({ { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS
{ FIRST | LAST } ] }[, ...] )
    [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ]
    [ COMMENT 'text' ]
    [ WITH ( {storage_parameter = value} [, ... ] ) ]

    [ WHERE predicate ];
```

- 在分区表上创建索引。

```
CREATE [ UNIQUE ] INDEX [ [IF NOT EXISTS] [ schema_name. ] index_name ] ON table_name
[ USING method ]
    ( {{ column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS
LAST ] }[, ...] )
    [ NULLS [ NOT ] DISTINCT | NULLS IGNORE ]
    [ COMMENT 'text' ]
    LOCAL [ ( { PARTITION index_partition_name  } [, ...] ) ]
    [ WITH ( { storage_parameter = value } [, ...] ) ]
    ;
```

 版权所有 © 华为云计算技术有限公司

参数说明

表 21-204 CREATE INDEX 参数说明

参数	描述	取值范围
UNIQUE	创建唯一性索引，每次添加数据时检测表中是否有重复值。如果插入或更新的值会引起重复的记录时，系统会报错。 目前只有行存表B-Tree索引和列存表的B-Tree索引支持唯一索引。	-
IF NOT EXISTS	指定IF NOT EXISTS时，若不存在同名索引，则可以成功创建索引。若已存在同名索引，创建时不会报错，仅会提示该索引已存在，且不执行任何操作。当使用IF NOT EXISTS时，需要指定索引名。该参数仅9.1.0及以上集群版本支持。	-
schema_name	要创建的索引所在的模式名。指定的模式名需与表所在的模式相同。	-
index_name	要创建的索引名，索引的模式与表相同。	字符串，需符合**标识符命名规范**。
table_name	需要为其创建索引的表的名字，可以用模式修饰。	已存在的表名。
USING method	指定创建索引的方法。 • 行存表支持的索引类型：btree（行存表缺省值）、gin、gist。 • 列存表支持的索引类型：psort（列存表缺省值）、btree、gin。	创建索引的方法支持： • **btree**：B-Tree索引使用一种类似于B+树的结构来存储数据的键值，通过这种结构能够快速的查找索引。B-Tree适合支持比较查询以及查询范围。 • **gin**：GIN索引是倒排索引，可以处理包含多个键的值（比如数组）。 • **gist**：GiST索引适用于几何和地理等多维数据类型和集合数据类型。 • **psort**：PSORT索引。针对列存表进行局部排序索引。
column_name	表中需要创建索引的列的名字（字段名）。	如果索引方式支持多字段索引，可以声明多个字段。最多可以声明32个字段。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
expression	创建一个基于该表的一个或多个字段的表达式索引，通常必须写在圆括号中。	如果表达式有函数调用的形式，圆括号可以省略。 表达式索引可用于获取对基本数据的某种变形的快速访问。比如，一个在upper(col)上的函数索引将允许WHERE upper(col) = 'JIM'子句使用索引。 在创建表达式索引时，如果表达式中包含IS NULL子句，则这种索引是无效的。此时，建议用户尝试创建一个部分索引。
COLLATE collation	COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。如果没有指定，则使用默认的排序规则。	-
opclass	操作符类的名字。	对于索引的每一列可以指定一个操作符类，操作符类标识了索引那一列的使用的操作符。例如一个B-Tree索引在一个四字节整数上可以使用int4_ops；这个操作符类包括四字节整数的比较函数。 实际上对于列上的数据类型默认的操作符类是足够用的。操作符类主要用于一些有多种排序的数据。例如，用户想按照绝对值或者实数部分排序一个复数。能通过定义两个操作符类然后在建立索引时选择合适的类。
ASC	指定按升序排序（默认）。本选项仅行存支持。	-
DESC	指定按降序排序。本选项仅行存支持。	-
NULLS FIRST	指定空值在排序中排在非空值之前，当指定DESC排序时，本选项为默认的。	-
NULLS LAST	指定空值在排序中排在非空值之后，未指定DESC排序时，本选项为默认的。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
NULLS DISTINCT NULLS NOT DISTINCT NULLS IGNORE	指定Unique唯一索引中索引列NULL值的处理方式。	默认取值：该参数默认取值为空，即NULL值可重复插入。 在对插入的新数据和表中原始数据进行列的等值比较时，对于NULL值有以下三种处理方式： • NULLS DISTINCT：将每个NULL值视为不同的值，即允许NULL值重复插入。 • NULLS NOT DISTINCT：将所有NULL值视为相同的值，即索引中只保留一个NULL值条目，不允许NULL值重复插入，只有非NULL值且与原索引列中已有数据不相等的数据可插入。 • NULLS IGNORE：在等值比较时跳过NULL值。若索引列全为NULL，则NULL值可重复插入；部分索引列为NULL，只有非NULL值且与原索引列中已有数据不相等的数据可插入。 三种处理方式具体的行为参见**表21-206**所示。
COMMENT 'text'	指定索引的注释信息。	-
WITH ({storage_parameter = value} [, ...])	指定索引方法的存储参数。包括：**FILLFACTOR**、**INVISIBLE**、**FASTUPDATE**、**GIN_PENDING_LIST_LIMIT**。	取值范围和详细用法参见**表21-205**。
WHERE predicate	创建一个部分索引。 部分索引是一个只包含表的一部分记录的索引，通常是该表中比其他部分数据更有用的部分。例如，有一个表，表里包含已记账和未记账的订单，未记账的订单只占表的一小部分而且这部分是最常用的部分，此时就可以通过只在未记账部分创建一个索引来改善性能。另外一个可能的用途是使用带有UNIQUE的WHERE强制一个表的某个子集的唯一性。	predicate表达式只能引用表的字段，它可以使用所有字段，而不仅是被索引的字段。目前，子查询和聚集表达式不能出现在WHERE子句里。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
LOCAL	本地索引（分区索引），即在分区表的每个区上单独创建索引，本地索引与分区表的分区一一对应，具有与其表相同的分区数和相同的分区范围。当表分区发生变化时，只需维护对应的索引分区。	-
PARTITION index_partition_name	索引分区的名称。	字符串，需符合标识符的命名规范。

版权所有 © 华为云计算技术有限公司

表 21-205 WITH ({storage_parameter = value} [, ...])指定索引方法的存储参数

存储参数	描述	示例
FILLFACTOR	**GIN和PSORT索引不支持该参数**，表示一个索引的填充因子（fillfactor），取值范围是一个介于10和100之间的百分比值，用于指定索引页初始填充的百分比，剩余的空间可以用于后续的插入和更新操作，从而减少页面分裂和提高性能。 该参数的作用是控制索引页的填充程度，从而影响数据库的性能和存储空间的利用效率，通过调整FILLFACTOR的值，可以减少后续更新或插入操作导致的页分裂，优化索引的存储和性能。 **FILLFACTOR对索引性能的影响**： ● **性能**：当索引页填满后，新的数据插入会导致页面分裂，这会消耗更多的I/O资源和时间。较低的 fillfactor可以减少页面分裂，提高插入和更新性能，但会增加存储空间的使用。较高的fillfactor可以提高存储空间的利用率，但可能会导致更多的页面分裂，降低插入和更新性能。 ● **存储空间**：较低的fillfactor会增加每个数据页面的空闲空间，导致更多的磁盘空间使用。较高的fillfactor会更紧凑地存储数据，减少磁盘空间使用。 **如何选择合适的FILLFACTOR值**： ● 对于更新频繁的表，选择较小fillfactor值。例如60~80，预留空间，减少页分裂。 ● 对于更新较少的表，选择较大fillfactor值。例如90~100，以提高存储空间利用率。	在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定B-Tree索引，并指定该索引的填充因子**FILLFACTOR**为70%。 这意味着每个数据页面在初次填充时只会填满70%。剩余的30%空间留给将来的插入和更新操作。 CREATE INDEX ds_ship_mode_t1_index6 ON tpcds.ship_mode_t1 USING btree(SM_SHIP_MODE_SK) WITH (**FILLFACTOR** = 70);

版权所有 © 华为云计算技术有限公司

存储参数	描述	示例
INVISIBLE	控制优化器是否生成索引扫描相关计划。 **适用场景**：维护期间临时禁用索引，或测试索引对性能的影响。 **取值范围**： ● ON表示不生成索引扫描相关计划。 ● OFF表示生成索引扫描相关计划。 **默认值**：OFF	在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定B-Tree索引，并设置**优化器不生成该索引扫描相关计划** CREATE INDEX ds_ship_mode_t1_index7 ON tpcds.ship_mode_t1 USING btree(SM_SHIP_MODE_SK) WITH (**INVISIBLE** = ON);
FASTUPDATE	**仅GIN索引支持该参数**，GIN索引是否使用快速更新，当启用后，数据变更会暂存到待处理列表（Pending List）。 **取值范围**： ● ON，表示使用快速更新。 ● OFF，表示关闭快速更新。 **默认值**：ON	在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定GIN索引，并打开快速更新参数**FASTUPDATE**，设置该索引Pending List容量的最大值为8MB。 CREATE INDEX ds_ship_mode_t1_index8 ON tpcds.ship_mode_t1 USING gin(SM_SHIP_MODE_SK) WITH (FASTUPDATE = ON, GIN_PENDING_LIST_LIMIT = 8192);
GIN_PENDING_LIST_LIMIT	**仅GIN索引支持该参数**，当GIN索引启用fastupdate时，设置该索引Pending List容量的最大值。 **Pending List**是GIN索引特有的数据结构，用于临时存储索引更新操作。当GIN索引启用FASTUPDATE参数时，新的索引项不会直接写入主索引结构，而是先存入Pending List，待满足特定条件时再批量合并到主索引中。 **取值范围**：64~INT_MAX，单位KB。 **默认值**：由GUC参数gin_pending_list_limit的值决定（默认为4MB）。	

表 21-206 唯一索引中索引列 NULL 值的处理方式

字段控制	索引列全为NULL	部分索引列为NULL
NULLS DISTINCT	可重复插入。	可重复插入。
NULLS NOT DISTINCT	不可重复插入。	非NULL值相等，不可插入；非NULL值不相等，则插入成功。

版权所有 © 华为云计算技术有限公司

字段控制	索引列全为NULL	部分索引列为NULL
NULLS IGNORE	可重复插入。	非NULL值相等，不可插入；非NULL值不相等，则插入成功。

示例

- 创建示例表tpcds.ship_mode_t1。

```
CREATE SCHEMA tpcds;
DROP TABLE IF EXISTS tpcds.ship_mode_t1;
CREATE TABLE tpcds.ship_mode_t1
(
    SM_SHIP_MODE_SK           INTEGER               NOT NULL,
    SM_SHIP_MODE_ID           CHAR(16)              NOT NULL,
    SM_TYPE                   CHAR(30)                      ,
    SM_CODE                   CHAR(10)                      ,
    SM_CARRIER                CHAR(20)                      ,
    SM_CONTRACT               CHAR(20)
)
DISTRIBUTE BY HASH(SM_SHIP_MODE_SK);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建唯一索引。

```
CREATE UNIQUE INDEX ds_ship_mode_t1_index1 ON tpcds.ship_mode_t1(SM_SHIP_MODE_SK);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建UNIQUE唯一索引，并指定NULL值的处理方式。

```
CREATE UNIQUE INDEX ds_ship_mode_t1_index5 ON tpcds.ship_mode_t1(SM_SHIP_MODE_SK)
NULLS NOT DISTINCT;
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建索引时添加索引的注释。

```
CREATE INDEX ds_ship_mode_t1_index_comment ON tpcds.ship_mode_t1(SM_SHIP_MODE_SK)
COMMENT 'index';
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定B-Tree索引。

```
CREATE INDEX ds_ship_mode_t1_index4 ON tpcds.ship_mode_t1 USING btree(SM_SHIP_MODE_SK);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定B-Tree索引，并指定该索引的填充因子**FILLFACTOR**为70%。

```
CREATE INDEX ds_ship_mode_t1_index6 ON tpcds.ship_mode_t1 USING btree(SM_SHIP_MODE_SK)
WITH (FILLFACTOR = 70);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定B-Tree索引，并设置**优化器不生成该索引扫描相关计划**(默认为OFF，表示生成，ON表示不生成)。

```
CREATE INDEX ds_ship_mode_t1_index7 ON tpcds.ship_mode_t1 USING btree(SM_SHIP_MODE_SK)
WITH (INVISIBLE = ON);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建指定GIN索引，并打开快速更新参数**FASTUPDATE**，设置该索引Pending List容量的最大值为8MB。

```
CREATE INDEX ds_ship_mode_t1_index8 ON tpcds.ship_mode_t1 USING gin(SM_SHIP_MODE_SK)
WITH (FASTUPDATE = ON, GIN_PENDING_LIST_LIMIT = 8192);
```

在表tpcds.ship_mode_t1上SM_CODE字段上创建表达式索引。

```
CREATE INDEX ds_ship_mode_t1_index2 ON tpcds.ship_mode_t1(SUBSTR(SM_CODE,1 ,4));
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建SM_SHIP_MODE_SK大于10的部分索引。

```
CREATE UNIQUE INDEX ds_ship_mode_t1_index3 ON tpcds.ship_mode_t1(SM_SHIP_MODE_SK)
WHERE SM_SHIP_MODE_SK>10;
```

 版权所有 © 华为云计算技术有限公司

- 创建示例表tpcds.customer_address_p1。

```
CREATE TABLE tpcds.customer_address_p1
(
    CA_ADDRESS_SK             INTEGER               NOT NULL,
    CA_ADDRESS_ID             CHAR(16)              NOT NULL,
    CA_STREET_NUMBER          CHAR(10)                      ,
    CA_STREET_NAME            VARCHAR(60)                   ,
    CA_STREET_TYPE            CHAR(15)                      ,
    CA_SUITE_NUMBER           CHAR(10)                      ,
    CA_CITY                   VARCHAR(60)                   ,
    CA_COUNTY                 VARCHAR(30)                   ,
    CA_STATE                  CHAR(2)                       ,
    CA_ZIP                    CHAR(10)                      ,
    CA_COUNTRY                VARCHAR(20)                   ,
    CA_GMT_OFFSET             DECIMAL(5,2)                  ,
    CA_LOCATION_TYPE          CHAR(20)
)
DISTRIBUTE BY HASH(CA_ADDRESS_SK)
PARTITION BY RANGE(CA_ADDRESS_SK)
(
  PARTITION p1 VALUES LESS THAN (3000),
  PARTITION p2 VALUES LESS THAN (5000) ,
  PARTITION p3 VALUES LESS THAN (MAXVALUE)
)
ENABLE ROW MOVEMENT;
```

创建分区表索引ds_customer_address_p1_index1，不指定索引分区的名字。

```
CREATE INDEX ds_customer_address_p1_index1 ON tpcds.customer_address_p1(CA_ADDRESS_SK)
LOCAL;
```

创建分区表索引ds_customer_address_p1_index2，并指定索引分区的名字。

```
CREATE INDEX ds_customer_address_p1_index2 ON tpcds.customer_address_p1(CA_ADDRESS_SK)
LOCAL
(
    PARTITION CA_ADDRESS_SK_index1,
    PARTITION CA_ADDRESS_SK_index2,
    PARTITION CA_ADDRESS_SK_index3
)
;
```

创建分区表索引ds_customer_address_p1_index_comment，并添加索引注释。

```
CREATE INDEX ds_customer_address_p1_index_comment ON
tpcds.customer_address_p1(CA_ADDRESS_SK) COMMENT 'index' LOCAL
(
    PARTITION CA_ADDRESS_SK_index1,
    PARTITION CA_ADDRESS_SK_index2,
    PARTITION CA_ADDRESS_SK_index3
)
;
```

关于 NULLS DISTINCT、NULLS NOT DISTINCT、NULLS IGNORE 使用上的场景差异

以下举例说明这3个参数的使用差异，便于理解。

示例

假设我们有一张表users，包含id和email（可能为 NULL，表示未提供邮箱）。

场景 1：默认 NULLS DISTINCT（NULL可重复插入）

```
-- 创建表
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id INT,
  email TEXT
```

 版权所有 © 华为云计算技术有限公司

```
)
DISTRIBUTE BY HASH(email);
-- 创建唯一索引（默认 NULLS DISTINCT）
CREATE UNIQUE INDEX idx_email_unique ON users (email);
-- 插入数据
INSERT INTO users (id, email) VALUES (1, NULL); -- 成功
INSERT INTO users (id, email) VALUES (2, NULL); -- 成功（NULL 被视为不同）
INSERT INTO users (id, email) VALUES (3, 'a@example.com'); -- 成功
INSERT INTO users (id, email) VALUES (4, 'a@example.com'); -- 失败（非 NULL 重复）
```

```
error_code: DWS.S0010010
error_msg: Constraint Violation. STATE: 23505, message: ERROR: dn_6005_6006: duplicate key value violates unique constraint "idx_email_unique"
Detail: Key (email)=(a@example.com) already exists.
```

场景 2：NULLS NOT DISTINCT（NULL不可重复插入）

```
-- 重新创建唯一索引，指定NULLS NOT DISTINCT
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id INT,
  email TEXT
)
DISTRIBUTE BY HASH(email);
CREATE UNIQUE INDEX idx_email_unique ON users (email) NULLS NOT DISTINCT;
-- 插入数据
INSERT INTO users (id, email) VALUES (5, NULL); -- 成功（第一个 NULL）
INSERT INTO users (id, email) VALUES (6, NULL); -- 失败（第二个 NULL，违反唯一约束）
INSERT INTO users (id, email) VALUES (7, 'b@example.com'); -- 成功
INSERT INTO users (id, email) VALUES (8, 'b@example.com'); -- 失败（非 NULL 重复）
```

场景 3：NULLS IGNORE（忽略NULL值）

```
-- 创建索引，忽略NULL值
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id INT,
  email TEXT
)
DISTRIBUTE BY HASH(email);
CREATE UNIQUE INDEX idx_email_ignore ON users (email) NULLS IGNORE;
-- 插入数据
INSERT INTO users (id, email) VALUES (9, NULL); -- 成功（第一个 NULL）
INSERT INTO users (id, email) VALUES (10, NULL); -- 成功（NULL值被忽略）
INSERT INTO users (id, email) VALUES (11, 'c@example.com'); -- 成功
INSERT INTO users (id, email) VALUES (12, 'c@example.com'); -- 失败（非 NULL 重复）
```

相关链接

- 修改和删除索引语法参见**ALTER INDEX**和**DROP INDEX**。
- 更多索引案例请参见《数据仓库服务开发指南》的“创建和管理索引”章节。

21.13.6.2 ALTER INDEX

功能描述

ALTER INDEX用于修改索引的定义。

注意事项

只有索引的所有者有权限执行此命令，系统管理员默认拥有此权限。

版权所有 © 华为云计算技术有限公司

语法格式

- 重命名表索引的名称，支持新的索引名称前带有原索引所在的schema名，但是不支持同时修改schema名。
```
ALTER INDEX [ IF EXISTS ] index_name
    RENAME TO new_name;
ALTER INDEX [ IF EXISTS ] index_name
    RENAME TO schema.new_name;
```
- 修改表索引的存储参数。
```
ALTER INDEX [ IF EXISTS ] index_name
    SET ( {storage_parameter = value} [, ... ] );
```
- 修改索引的状态标记。
```
ALTER INDEX [ IF EXISTS ] index_name
    SET ( {invisible = value} [, ... ] );
```
- 重置表索引的存储参数。
```
ALTER INDEX [ IF EXISTS ] index_name
    RESET ( storage_parameter [, ... ] ) ;
```
- 设置表索引或索引分区不可用。**列存表不支持该语法**。
```
ALTER INDEX [ IF EXISTS ] index_name
    [ MODIFY PARTITION index_partition_name ] UNUSABLE;
```
- 重建表索引或索引分区。
```
ALTER INDEX index_name
    REBUILD [ PARTITION index_partition_name ] [WITHOUT UNUSABLE];
```
- 重命名索引分区。PG_OBJECT系统表记录索引的最后修改时间时，**暂不支持对该语法的记录**。
```
ALTER INDEX [ IF EXISTS ] index_name
    RENAME PARTITION index_partition_name TO new_index_partition_name;
```
- 添加，修改索引的注释。
```
ALTER INDEX [ IF EXISTS ] index_name
    COMMENT 'text';
```
- 删除索引的注释。
```
ALTER INDEX [ IF EXISTS ] index_name
    COMMENT '';
ALTER INDEX [ IF EXISTS ] index_name
    COMMENT NULL;
```

参数说明

表 21-207 ALTER INDEX 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的索引不存在时，返回一个提示而不是错误。	-
RENAME TO	修改索引的名称，对存储的数据本身没有任何影响。	-
SET ({ STORAGE_PARAMETER = value } [, ...])	修改索引的一个或多个索引方法特定的存储参数。 此命令不会立即修改索引内容，根据参数的不同，可能需要使用REINDEX重建索引来获得所需的效果。	取值范围和详细用法参见**表21-208**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
RESET ({ storage_parameter } [, ...])	重置索引的一个或多个索引方法特定的存储参数为默认值。与SET一样，可能需要使用REINDEX来完全更新索引。	-
[MODIFY PARTITION index_partition_name] UNUSABLE	用于设置表或者索引分区上的索引不可用。	-
REBUILD [PARTITION index_partition_name]	用于重建表或者索引分区上的索引。	-
WITHOUT UNUSABLE	重建表或者索引分区上的索引时不重建状态为UNUSABLE的索引。	-
RENAME PARTITION	用于重命名索引分区。	-
COMMENT comment_text	用于添加，修改或删除索引的注释。	-
index_name	需要修改的索引名。	已存在的索引名。
new_name	新的索引名。	字符串，需符合**标识符命名规范**。
storage_parameter	索引方法特定的参数名。	-
invisible	控制优化器是否生成索引扫描相关计划。默认值：OFF。	ON或OFF。 • ON表示不生成索引扫描相关计划。 • OFF表示生成索引扫描相关计划。
value	索引方法特定的存储参数的新值。根据参数的不同，这可能是一个数字或单词。	-
new_index_partition_name	新索引分区名。	字符串，需符合**标识符命名规范**。
index_partition_name	索引分区名。	字符串，需符合**标识符命名规范**。
comment_text	索引的注释信息。	-

版权所有 © 华为云计算技术有限公司

表 21-208 WITH ({storage_parameter = value} [, ...])指定索引方法的存储参数

存储参数	描述	示例
FILLFACTOR	**GIN和PSORT索引不支持该参数**，表示一个索引的填充因子（ fillfactor ），取值范围是一个介于10和100之间的百分比值，用于指定索引页初始填充的百分比，剩余的空间可以用于后续的插入和更新操作，从而减少页面分裂和提高性能。 该参数的作用是控制索引页的填充程度，从而影响数据库的性能和存储空间的利用效率，通过调整FILLFACTOR的值，可以减少后续更新或插入操作导致的页分裂，优化索引的存储和性能。 **FILLFACTOR对索引性能的影响**： ● **性能**：当索引页填满后，新的数据插入会导致页面分裂，这会消耗更多的I/O资源和时间。较低的 fillfactor可以减少页面分裂，提高插入和更新性能，但会增加存储空间的使用。较高的fillfactor可以提高存储空间的利用率，但可能会导致更多的页面分裂，降低插入和更新性能。 ● **存储空间**：较低的fillfactor会增加每个数据页面的空闲空间，导致更多的磁盘空间使用。较高的fillfactor会更紧凑地存储数据，减少磁盘空间使用。 **如何选择合适的FILLFACTOR值**： ● 对于更新频繁的表，选择较小fillfactor值如60~80，预留空间，减少页分裂。 ● 对于更新较少的表，选择较大fillfactor值如90~100，以提高存储空间利用率。	修改索引的填充因子**FILLFACTOR**为60%。 `ALTER INDEX tpcds.ds_ship_mode_t1_index7 SET (**FILLFACTOR** = 60);`

 版权所有 © 华为云计算技术有限公司

存储参数	描述	示例
INVISIBLE	控制优化器是否生成索引扫描相关计划。 **适用场景**：维护期间临时禁用索引，或测试索引对性能的影响。 **取值范围**： ● ON表示不生成索引扫描相关计划。 ● OFF表示生成索引扫描相关计划。 **默认值**：OFF	设置**优化器不生成索引的扫描相关计划。** ALTER INDEX tpcds.ds_ship_mode_t1_index9 SET (INVISIBLE = ON);
FASTUPDATE	**仅GIN索引支持该参数**，GIN索引是否使用快速更新，当启用后，数据变更会暂存到待处理列表（Pending List）。 **取值范围**：ON，OFF **默认值**：ON	打开快速更新参数**FASTUPDATE**，设置该索引Pending List容量的最大值为8MB。 ALTER INDEX tpcds.ds_ship_mode_t1_index10 SET (FASTUPDATE = ON, GIN_PENDING_LIST_LIMIT = 8192);
GIN_PENDING_LIST_LIMIT	**仅GIN索引支持该参数**，当GIN索引启用fastupdate时，设置该索引Pending List容量的最大值。 **Pending List**是GIN索引特有的数据结构，用于临时存储索引更新操作。当GIN索引启用FASTUPDATE参数时，新的索引项不会直接写入主索引结构，而是先存入Pending List，待满足特定条件时再批量合并到主索引中。 **取值范围**：64~INT_MAX，单位KB。 **默认值**：由GUC参数gin_pending_list_limit的值决定（默认为4MB）。	

示例

创建示例表tpcds.ship_mode_t1。

```
DROP TABLE IF EXISTS tpcds.ship_mode_t1;
CREATE TABLE tpcds.ship_mode_t1
(
    SM_SHIP_MODE_SK          INTEGER               NOT NULL,
    SM_SHIP_MODE_ID          CHAR(16)              NOT NULL,
    SM_TYPE                  CHAR(30)                      ,
    SM_CODE                  CHAR(10)                      ,
    SM_CARRIER               CHAR(20)                      ,
    SM_CONTRACT              CHAR(20)
)
DISTRIBUTE BY HASH(SM_SHIP_MODE_SK);
```

在表tpcds.ship_mode_t1上的SM_SHIP_MODE_SK字段上创建唯一索引。

版权所有 © 华为云计算技术有限公司

```
DROP INDEX IF EXISTS ds_ship_mode_t1_index1;
CREATE UNIQUE INDEX ds_ship_mode_t1_index1 ON tpcds.ship_mode_t1(SM_SHIP_MODE_SK);
```

在表tpcds.ship_mode_t1上SM_CODE字段上创建表达式索引。

```
DROP INDEX IF EXISTS ds_ship_mode_t1_index2;
CREATE INDEX ds_ship_mode_t1_index2 ON tpcds.ship_mode_t1(SUBSTR(SM_CODE,1 ,4));
```

重命名现有的索引ds_ship_mode_t1_index1为ds_ship_mode_t1_index5。

```
ALTER INDEX tpcds.ds_ship_mode_t1_index1 RENAME TO ds_ship_mode_t1_index5;
```

设置索引ds_ship_mode_t1_index2不可用。

```
ALTER INDEX tpcds.ds_ship_mode_t1_index2 UNUSABLE;
```

重建索引ds_ship_mode_t1_index2。

```
ALTER INDEX tpcds.ds_ship_mode_t1_index2 REBUILD;
```

相关链接

CREATE INDEX，**DROP INDEX**，**REINDEX**

21.13.6.3 REINDEX

功能描述

为表中的数据重建索引。

在以下几种情况下需要使用REINDEX重建索引：

- 索引崩溃，并且不再包含有效的数据。
- 索引变得“臃肿”，包含大量的空页或接近空页。
- 为索引更改了存储参数（例如填充因子），并且希望这个更改完全生效。
  使用CONCURRENTLY选项创建索引失败，留下了一个“非法”索引。

注意事项

REINDEX DATABASE和SYSTEM这种形式的重建索引不能在事务块中执行。

存算分离架构下，不支持REINDEX全库索引。

⚠ 警告

- 避免在业务高峰期执行对大表执行CREATE INDEX和REINDEX操作。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

- 重建全库索引或者系统表的索引。
  ```
  REINDEX { DATABASE | SYSTEM } name [ FORCE ];
  ```
- 重建普通索引或者普通表上的索引。
  ```
  REINDEX { INDEX |  TABLE  } name [ FORCE | WITHOUT UNUSABLE ]
  ```


版权所有 © 华为云计算技术有限公司

- 重建索引分区或者分区上的索引。

```
REINDEX { | TABLE} name
  PARTITION partition_name [ FORCE | WITHOUT UNUSABLE ];
```

参数说明

表 21-209 REINDEX 参数说明

参数	描述	取值范围
INDEX	重新建立指定的索引。	-
TABLE	重新建立指定表的所有索引，如果表有从属的“TOAST”表，则这个表也会重建索引。	-
DATABASE	重建当前数据库里的所有索引。	-
SYSTEM	在当前数据库上重建所有系统表上的索引。不会处理在用户表上的索引。	-
name	需要重建索引的索引、表、数据库的名称。表和索引可以有模式修饰。 **说明** REINDEX DATABASE和SYSTEM只能重建当前数据库的索引，所以name必须和当前数据库名称相同。	-
FORCE	无效选项，会被忽略。	-
WITHOUT UNUSABLE	指定该参数时，在REINDEX时会忽略对象上的UNUSABLE索引或者UNUSABLE索引分区。该参数仅9.1.0.100及以上版本支持。	-
partition_name	需要重建索引的分区的名字或者索引分区的名字。	• 如果前面是REINDEX INDEX，则这里应该指定索引分区的名字。 • 如果前面是REINDEX TABLE，则这里应该指定分区的名字。

须知

REINDEX DATABASE和SYSTEM这种形式的重建索引不能在事务块中执行。

示例

重建一个单独索引。

```
REINDEX INDEX tpcds.tpcds_customer_index1;
```

重建表tpcds.customer_t1上的所有索引。

版权所有 © 华为云计算技术有限公司

```
REINDEX TABLE tpcds.customer_t1;
```

重建分区表customer_address分区P1的分区索引。

```
DROP TABLE IF EXISTS customer_address;
CREATE TABLE customer_address
(
    ca_address_sk       INTEGER                 NOT NULL   ,
    ca_address_id       CHARACTER(16)           NOT NULL   ,
    ca_street_number    CHARACTER(10)                      ,
    ca_street_name      CHARACTER varying(60)              ,
    ca_street_type      CHARACTER(15)                      ,
    ca_suite_number     CHARACTER(10)
)
DISTRIBUTE BY HASH (ca_address_sk)
PARTITION BY RANGE(ca_address_sk)
(
        PARTITION P1 VALUES LESS THAN(2450815),
        PARTITION P2 VALUES LESS THAN(2451179),
        PARTITION P3 VALUES LESS THAN(2451544),
        PARTITION P4 VALUES LESS THAN(MAXVALUE)
);

CREATE INDEX customer_address_index on customer_address(CA_ADDRESS_SK) LOCAL;

REINDEX TABLE customer_address PARTITION P1;
```

21.13.6.4 DROP INDEX

功能描述

删除索引。

注意事项

只有索引的所有者有权限执行DROP INDEX命令，系统管理员默认拥有此权限。

语法格式

```
DROP INDEX [ CONCURRENTLY ][ IF EXISTS ]
    index_name [, ...] [ CASCADE | RESTRICT ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-210 DROP INDEX 参数说明

参数	描述	取值范围
CONCURRENTLY	删除索引而不锁定索引表上的并发选择、插入、更新和删除。 普通的DROP INDEX在表上获取排他锁，从而阻止其他访问，直到可以完成索引删除为止。使用此选项，命令将一直等到冲突的事务完成。 **说明** 使用此选项时需要注意：只能指定一个索引名称，并且不支持CASCADE选项（因此，不能以这种方式删除支持UNIQUE或PRIMARY KEY约束的索引）。可以在事务块内执行常规的DROP INDEX命令，但不能以DROP INDEX CONCURRENTLY方式执行。	-
IF EXISTS	如果指定的索引不存在，则发出一个notice而不是抛出一个错误。	-
index_name	要删除的索引名。	已存在的索引。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：表示允许级联删除依赖于该索引的对象。 • RESTRICT（缺省值）：表示有依赖与此索引的对象存在，则该索引无法被删除。

示例

删除现有的索引ds_ship_mode_t1_index2。

```
DROP INDEX tpcds.ds_ship_mode_t1_index2;
```

相关链接

ALTER INDEX，**CREATE INDEX**

21.13.6.5 CREATE SEQUENCE

功能描述

向当前数据库里增加一个新的序列。

序列是一个数据库对象，主要用于生成一系列唯一的、递增或递减的数值，通常用作表中的主键或唯一标识符。

版权所有 © 华为云计算技术有限公司

注意事项

- 创建序列的用户为该序列的所有者。
- 如果指定了模式名称，则会在指定模式中创建序列，否则会在当前模式中创建。序列名必须和同一个模式中的其他序列、表、索引、视图或外表的名字不同。
- 创建序列后，在表中使用序列的nextval()函数和generate_series(1,N)函数对表插入数据，请保证nextval的可调用次数大于等于N+1次，否则会因为generate_series()函数会调用N+1次而导致报错。
- 不支持在template1数据库中创建SEQUENCE。
- 建议控制SEQUENCE的使用数量（含SERIAL类型），因分布式场景下SEQUENCE中序列值的产生依赖GTM全局管理，频繁的与GTM交互会严重影响性能。建议引用公共的SEQUENCE对象，或者针对需要唯一值的场景使用UUID进行替换。同时为尽量减少与GTM的交互，针对手动创建的SEQUENCE，建议指定CACHE值不小于10000。

语法格式

```
CREATE SEQUENCE [ IF NOT EXISTS ] name [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE | NOMINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE | NOMAXVALUE]
    [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE | NOCYCLE ]
    [ OWNED BY { table_name.column_name | NONE } ];
```

参数说明

表 21-211 CREATE SEQUENCE 参数说明

参数	描述	取值范围
IF NOT EXISTS	指定IF NOT EXISTS时，若不存在同名SEQUENCE，则可以成功创建SEQUENCE。若已存在同名SEQUENCE，创建时不会报错，仅会提示该SEQUENCE已存在，且不执行任何操作。 该参数仅9.1.0及以上版本支持。	-
name	将要创建的序列名称。	字符串，需符合**标识符命名规范**。
increment	指定序列的步长。正值将生成一个递增的序列，负值将生成一个递减的序列。 默认值为1。	-
MINVALUE minvalue | NO MINVALUE| NOMINVALUE	确定序列可以生成的最小值。如果没有指定minvalue或者指定了NO MINVALUE，则递增序列的缺省值为1，递减序列的缺省值为-2^{63}-1。 NOMINVALUE等价于NO MINVALUE。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
MAXVALUE maxvalue | NO MAXVALUE| NOMAXVALUE	确定序列可以生成的最大值。如果没有声明maxvalue或者声明了NO MAXVALUE，则递增序列的缺省值为2^{63}-1，递减序列的缺省值为-1。 NOMAXVALUE等价于NO MAXVALUE。	-
start	指定序列的起始值。 默认值：对于递增序列为minvalue，递减序列为maxvalue。	-
cache	指定数据库预先缓存（预分配）多少个序列号到内存中，从而加快访问速度。一个缓存周期内，CN不再向GTM索取序列号，而是使用本地预先申请的序列号。 9.1.0.100及以上集群版本中缺省值由GUC参数default_sequence_cache控制。 ● 新安装的9.1.0.100及以上集群版本中，默认值为20。 ● 从低版本升级到9.1.0.100及以上集群版本，默认值为1。表示一次只能生成一个值，即没有缓存。	● 不建议同时定义cache和maxvalue或minvalue。因为定义cache后不能保证序列的连续性，可能会产生空洞，造成序列号段浪费。 ● 建议cache值不要设置过大，否则会出现缓存序列号时（每个cache周期的第一个nextval）耗时过长的情况；同时建议cache值小于100000000。实际使用时应根据业务设置合理的cache值，既能保证快速访问，又不会浪费序列号。
CYCLE | NOCYCLE	当序列达到maxvalue或者minvalue时，允许序列可循环重新开始。 默认值为NO CYCLE，表示序列达到极大或极小值后将无法生成新值。NOCYCLE的作用等价于NO CYCLE。 ● 如果指定序列为CYCLE，则不能保证序列的唯一性。 ● 如果指定了NO CYCLE，则在序列达到其最大值后任何对nextval的调用都将返回错误。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
OWNED BY	将序列与特定表的指定字段相关联。这样，如果该字段或其所在表被删除，该序列也将被自动删除。关联的表和序列的所有者必须是同一个用户，并且在同一个模式中。 缺省值为OWNED BY NONE，表示不存在这种关联。	• 通过指定OWNED BY，仅建立了表的对应列和SEQUENCE之间关联关系，并不会在插入数据时在该列上产生自增序列。 • 通过指定OWNED BY创建的SEQUENCE不建议用于其他表，如果希望多个表共享SEQUENCE，该SEQUENCE不应该从属于特定表。

示例

创建一个名为serial的递增序列，从101开始。

```
CREATE SEQUENCE serial
 START 101
 CACHE 20;
```

从序列中选出下一个数字。

```
SELECT nextval('serial');
 nextval
 ---------
     101
```

从序列中选出下一个数字。

```
SELECT nextval('serial');
 nextval
 ---------
     102
```

创建与表关联的序列。

```
CREATE TABLE customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
    ca_location_type          char(20)
) ;

CREATE SEQUENCE serial1
```


版权所有 © 华为云计算技术有限公司

```
 START 101
 CACHE 20
OWNED BY customer_address.ca_address_sk;
```

使用serial创建主键自增序列表serial_table。

```
CREATE TABLE serial_table(a int, b serial);
INSERT INTO serial_table (a) VALUES (1),(2),(3);
SELECT * FROM serial_table ORDER BY b;
 a | b
---+---
 1 | 1
 2 | 2
 3 | 3
(3 rows)
```

相关链接

ALTER SEQUENCE，**DROP SEQUENCE**

21.13.6.6 ALTER SEQUENCE

功能描述

修改序列定义。

注意事项

- 使用ALTER SEQUENCE的用户必须是该序列的所有者。
- 用户要修改序列的所有者，必须是新角色的直接或者间接成员，并且所有者角色必须有序列所在模式上的CREATE权限。
- 当前版本仅支持修改拥有者、归属列和最大值。若要修改其他参数，可以删除重建，并用Setval函数恢复当前值。
- ALTER SEQUENCE MAXVALUE不支持在事务、函数和存储过程中使用。
- 修改序列的最大值后，会清空该序列在所有会话的cache。
- ALTER SEQUENCE会阻塞nextval、setval、currval和lastval的调用。

语法格式

修改序列最大值或归属列。

```
ALTER SEQUENCE [ IF EXISTS ] name
    [ MAXVALUE maxvalue | NO MAXVALUE | NOMAXVALUE ]
    [ CACHE cache ]
    [ OWNED BY { table_name.column_name | NONE } ] ;
```

修改序列的所有者。

```
ALTER SEQUENCE [ IF EXISTS ] name OWNER TO new_owner;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-212 ALTER SEQUENCE 参数说明

参数	描述	取值范围
IF EXISTS	如果序列不存在时返回一个提示而不是错误。	-
name	需要修改的序列的名称。	已存在的序列名称。
MAXVALUE maxvalue | NO MAXVALUE	序列所能达到的最大值。如果指定了NO MAXVALUE，则递增序列的缺省值为2^{63}-1，递减序列的缺省值为-1。NOMAXVALUE等价于NO MAXVALUE。	-
CACHE cache	指定要预先分配多少序列号并存储在内存中，以加快访问速度。该参数仅8.2.1.100及以上集群版本支持。	• cachevalue最小值为 1（一次只能生成一个值，即 NOCACHE）。 • 如果未指定，则保留旧缓存值。
OWNED BY	将序列和表的指定字段进行关联。这样，在删除指定字段或其所在表的时候会自动删除已关联的序列。 如果序列已经和表有关联后，使用OWNED BY参数后新的关联关系会覆盖旧的关联。关联的表和序列的所有者必须是同一个用户，并且在同一个模式中。使用OWNED BY NONE将删除任何已经存在的关联。	-
new_owner	序列新所有者的用户名。	已存在的用户名。

示例

创建示例序列seq_test和示例表t1。

```
CREATE SEQUENCE seq_test START 101;
DROP TABLE IF EXISTS t1;
CREATE TABLE t1(c1 bigint default nextval('seq_test'));
```

将序列serial的最大值修改为200。

```
ALTER SEQUENCE seq_test MAXVALUE 200;
```

将序列seq_test的归属列变为t1.c1。

```
ALTER SEQUENCE seq_test OWNED BY t1.c1;
```

相关链接

CREATE SEQUENCE，**DROP SEQUENCE**

版权所有 © 华为云计算技术有限公司

21.13.6.7 DROP SEQUENCE

功能描述

从当前数据库里删除序列。

注意事项

只有序列的所有者或者系统管理员才能删除。

语法格式

```
DROP SEQUENCE [ IF EXISTS ] {[schema.]sequence_name} [ , ... ] [ CASCADE | RESTRICT ];
```

参数说明

表 21-213 DROP SEQUENCE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的序列不存在，则发出一个notice而不是抛出一个错误。	-
name	要删除的序列名称。	已存在的序列名称。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖序列的对象。 • RESTRICT：如果存在任何依赖的对象，则拒绝删除序列（缺省行为）。

示例

删除序列serial。

```
DROP SEQUENCE serial;
```

相关链接

ALTER SEQUENCE，**CREATE SEQUENCE**

21.13.6.8 CREATE TEXT SEARCH CONFIGURATION

功能描述

创建新的文本搜索配置。文本搜索配置指定了文本搜索解析器，该文本搜索解析器可以将字符串划分为标记，外加一些词典（可被用来决定哪些标记是搜索感兴趣的）。

注意事项

- 如果仅指定了解析器，则新的文本搜索配置初始没有从标记类型到词典的映射，因此会忽略所有的单词。后面必须使用ALTER TEXT SEARCH CONFIGURATION

版权所有 © 华为云计算技术有限公司

命令创建映射使配置生效。如果指定了COPY选项，则会自动拷贝指定的文本搜索配置的解析器、映射、配置选项等信息。

- 如果指定了模式名称，则会在指定模式中创建文本搜索配置，否则会在当前模式中创建。
- 定义文本搜索配置的用户成为其所有者。
- PARSER和COPY选项是互斥的，因为复制现有配置时，其解析器配置也会被复制。

语法格式

```
CREATE TEXT SEARCH CONFIGURATION name
    ( PARSER = parser_name | COPY = source_config )
    [ WITH ( {configuration_option = value} [, ...] )];
```

参数说明

表 21-214 CREATE TEXT SEARCH CONFIGURATION 参数说明

参数	描述	取值范围
name	要创建的文本搜索配置的名称，可以用模式修饰。	字符串，需符合**标识符命名规范**。
parser_name	用于该配置的文本搜索解析器的名称。	-
source_config	要复制的现有文本搜索配置的名称。	-
configuration_option	文本搜索配置的配置参数，主要是针对parser_name执行的解析器，或者source_config隐含的解析器而言的。	目前共支持default、ngram、zhparser三种类型的解析器。 • default类型的解析器没有对应的configuration_option。 • ngram、zhparser类型解析器对应的configuration_option如**表21-215**所示。

表 21-215 ngram、zhparser 类型解析器对应的配置参数

解析器	配置参数	参数描述	取值范围
ngram	gram_size	分词长度。	正整数，1~4。 默认值：2。

版权所有 © 华为云计算技术有限公司

解析器	配置参数	参数描述	取值范围
	punctuation_ignore	是否忽略标点符号。	● true（默认值）：忽略标点符号。 ● false：不忽略标点符号。
	grapsymbol_ignore	是否忽略图形化字符。	● true：忽略图形化字符。 ● false（默认值）：不忽略图形化字符。
zhparser	punctuation_ignore	分词结果是否忽略所有的标点等特殊符号（不会忽略\r和\n）。	● true（默认值）：忽略所有的标点等特殊符号。 ● false：不忽略所有的标点等特殊符号。
	seg_with_duality	是否将闲散文字自动以二字分词法聚合。	● true：将闲散文字自动以二字分词法聚合。 ● false（默认值）：不将闲散文字自动以二字分词法聚合。
	multi_short	分词执行时是否执行针对长词复合切分。	● true（默认值）：执行针对长词复合切分。 ● false：不执行针对长词复合切分。
	multi_duality	设定是否将长词内的文字自动以二字分词法聚合。	● true：将长词内的文字自动以二字分词法聚合。 ● false（默认值）：不将长词内的文字自动以二字分词法聚合。
	multi_zmain	是否将重要单字单独显示。	● true：将重要单字单独显示。 ● false（默认值）：不将重要单字单独显示。
	multi_zall	是否将全部单字单独显示。	● true：将全部单字单独显示。 ● false（默认值）：不将全部单字单独显示。

示例

创建名为ngram1的文本搜索配置，使用ngram解析器。

```
CREATE TEXT SEARCH CONFIGURATION ngram1 (parser=ngram) WITH (gram_size = 2, grapsymbol_ignore = false);
```

创建名为ngram2的文本搜索配置，复制现有配置ngram1的所有参数。

 版权所有 © 华为云计算技术有限公司

```
CREATE TEXT SEARCH CONFIGURATION ngram2 (copy=ngram1);
```

创建名为english_1的文本搜索配置，使用默认文本解析器。

```
CREATE TEXT SEARCH CONFIGURATION english_1 (parser=default);
```

相关链接

ALTER TEXT SEARCH CONFIGURATION，DROP TEXT SEARCH CONFIGURATION

21.13.6.9 ALTER TEXT SEARCH CONFIGURATION

功能描述

修改文本搜索配置的定义。

用户可以将映射从标记类型调整为字典，更改配置的名称或者所有者，或者修改搜索配置的配置参数。

注意事项

- 当一个搜索配置已经被引用（如被用来创建索引），则不允许用户修改此文本搜索配置。
- 要使用ALTER TEXT SEARCH CONFIGURATION，用户必须是配置的所有者。

语法格式

- 增加文本搜索配置标记类型映射。
```
ALTER TEXT SEARCH CONFIGURATION name
    ADD MAPPING FOR token_type [, ... ] WITH dictionary_name [, ... ];
```
- 修改文本搜索配置字典。
```
ALTER TEXT SEARCH CONFIGURATION name
    ALTER MAPPING FOR token_type [, ... ] REPLACE old_dictionary WITH new_dictionary;
```
- 修改文本搜索配置标记类型。
```
ALTER TEXT SEARCH CONFIGURATION name
    ALTER MAPPING FOR token_type [, ... ] WITH dictionary_name [, ... ];
```
- 更改文本搜索配置字典。
```
ALTER TEXT SEARCH CONFIGURATION name
    ALTER MAPPING REPLACE old_dictionary WITH new_dictionary;
```
- 删除文本搜索配置标记类型映射。
```
ALTER TEXT SEARCH CONFIGURATION name
    DROP MAPPING [ IF EXISTS ] FOR token_type [, ... ];
```
- 重命名文本搜索配置所有者。
```
ALTER TEXT SEARCH CONFIGURATION name OWNER TO new_owner;
```
- 重命名文本搜索配置名称。
```
ALTER TEXT SEARCH CONFIGURATION name RENAME TO new_name;
```
- 重命名文本搜索配置命名空间。
```
ALTER TEXT SEARCH CONFIGURATION name SET SCHEMA new_schema;
```
- 修改文本搜索配置属性。
```
ALTER TEXT SEARCH CONFIGURATION name SET ( { configuration_option = value } [, ...] );
```
- 重置文本搜索配置属性。
```
ALTER TEXT SEARCH CONFIGURATION name RESET ( {configuration_option} [, ...] );
```


版权所有 © 华为云计算技术有限公司

说明

- ADD MAPPING FOR选项为文本搜索配置增加标记类型映射；如果ADD MAPPING FOR后面任何一个标记类型的映射已经存在于此文本搜索配置中，那么系统将会报错。
- ALTER MAPPING FOR选项会首先清除已有的标记类型映射，然后添加指定的标记类型映射。
- ALTER MAPPING REPLACE ... WITH ... 与ALTER MAPPING FOR ... REPLACE ... WITH ...选项会直接使用new_dictionary替换old_dictionary。需要注意的是，只有pg_ts_config_map系统表中存在maptokentype与old_dictionary对应关系的元组时，才能更新成功，否则不会成功，也不会有任何提示信息返回。
- DROP MAPPING FOR选项会删除当前文本搜索配置中指定的标记类型映射。 如果没有指定IF EXISTS选项，当DROP MAPPING FOR选项指定的标记类型映射在文本搜索配置中不存在时，数据库会报错。

参数说明

表 21-216 ALTER TEXT SEARCH CONFIGURATION 参数说明

参数	描述	取值范围
name	需要修改的文本搜索配置的名称，可以用模式修饰。	-
token_type	与配置的语法解析器关联的标记类型的名称。	详细信息参见**解析器**。
dictionary_name	文本搜索字典名称。 如果有多个字典，则它们会按指定的顺序搜索。	-
old_dictionary	映射中要替换的文本搜索字典名称。	-
new_dictionary	替换old_dictionary的文本搜索字典的名称。	-
new_owner	文本搜索配置的新所有者。	有效的用户名。
new_name	文本搜索配置的新名称。	字符串，需符合**标识符命名规范**。
new_schema	文本搜索配置的新模式名。	有效的模式名。
configuration_option	文本搜索配置项。	详细信息参见**CREATE TEXT SEARCH CONFIGURATION**。
value	文本搜索配置项的值。	-

示例

创建文本搜索配置ngram1。

```
DROP TEXT SEARCH CONFIGURATION IF EXISTS ngram1;
CREATE TEXT SEARCH CONFIGURATION ngram1 (parser=ngram) WITH (gram_size = 2, grapsymbol_ignore = false);
```

 版权所有 © 华为云计算技术有限公司

给文本搜索类型ngram1添加类型映射。

```
ALTER TEXT SEARCH CONFIGURATION ngram1 ADD MAPPING FOR multisymbol WITH simple;
```

修改文本搜索配置的所有者。

```
CREATE USER joe password '{Password}';
ALTER TEXT SEARCH CONFIGURATION ngram1 OWNER TO joe;
```

修改文本搜索配置的schema。

```
ALTER TEXT SEARCH CONFIGURATION ngram1 SET SCHEMA joe;
```

重命名文本搜索配置。

```
ALTER TEXT SEARCH CONFIGURATION joe.ngram1 RENAME TO ngram_1;
```

删除类型映射：

```
ALTER TEXT SEARCH CONFIGURATION joe.ngram_1 DROP MAPPING IF EXISTS FOR multisymbol;
```

创建文本搜索配置：

```
DROP TEXT SEARCH CONFIGURATION IF EXISTS english_1;
CREATE TEXT SEARCH CONFIGURATION english_1 (parser=default);
```

增加文本搜索配置标记类型映射。

```
ALTER TEXT SEARCH CONFIGURATION english_1 ADD MAPPING FOR word WITH simple,english_stem;
```

增加文本搜索配置标记类型映射。

```
ALTER TEXT SEARCH CONFIGURATION english_1 ADD MAPPING FOR email WITH english_stem,
french_stem;
```

修改文本搜索配置标记类型映射。

```
ALTER TEXT SEARCH CONFIGURATION english_1 ALTER MAPPING REPLACE french_stem with german_stem;
```

查询文本搜索配置相关信息。

```
SELECT b.cfgname,a.maptokentype,a.mapseqno,a.mapdict,c.dictname FROM pg_ts_config_map
a,pg_ts_config b, pg_ts_dict c WHERE a.mapcfg=b.oid AND a.mapdict=c.oid AND b.cfgname='english_1'
ORDER BY 1,2,3,4,5;
```

```
  cfgname  | maptokentype | mapseqno | mapdict |   dictname
-----------+--------------+----------+---------+--------------
 english_1 |            2 |        1 |    3765 | simple
 english_1 |            2 |        2 |   14505 | english_stem
 english_1 |            4 |        1 |   14505 | english_stem
 english_1 |            4 |        2 |   14511 | german_stem
(4 rows)
```

相关链接

CREATE TEXT SEARCH CONFIGURATION，**DROP TEXT SEARCH CONFIGURATION**

21.13.6.10 DROP TEXT SEARCH CONFIGURATION

功能描述

删除已有文本搜索配置。

版权所有 © 华为云计算技术有限公司

注意事项

要执行DROP TEXT SEARCH CONFIGURATION命令，用户必须是该文本搜索配置的所有者。

语法格式

```
DROP TEXT SEARCH CONFIGURATION [ IF EXISTS ] name [ CASCADE | RESTRICT ];
```

参数说明

表 21-217 DROP TEXT SEARCH CONFIGURATION 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的文本搜索配置不存在，那么发出一个notice而不是抛出一个错误。	-
name	要删除的文本搜索配置名称（可有模式修饰）。	已存在的文本搜索配置名称。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖文本搜索配置的对象。 • RESTRICT：如果有任何对象依赖文本搜索配置则拒绝删除它。这是默认情况。此选项为缺省值。

示例

删除文本搜索配置ngram1。

```
DROP TEXT SEARCH CONFIGURATION ngram1;
```

相关链接

ALTER TEXT SEARCH CONFIGURATION，**CREATE TEXT SEARCH CONFIGURATION**

21.13.6.11 CREATE TEXT SEARCH DICTIONARY

功能描述

创建一个新的全文检索词典。全文检索词典指定在全文检索时识别特定词并处理的方法。

词典的创建依赖于预定义模板（在系统表PG_TS_TEMPLATE中定义），支持创建五种类型的词典，分别是Simple、Ispell、Synonym、Thesaurus、以及Snowball，每种类型的词典可以完成不同的任务。

版权所有 © 华为云计算技术有限公司

注意事项

- 具有SYSADMIN权限的用户可以执行创建词典操作，创建该词典的用户自动成为其所有者。
- 临时模式（pg_temp）下不允许创建词典。
- 创建或修改词典之后，任何对于用户自定义的词典定义文件的修改，将不会影响到数据库中的词典。如果需要在数据库中使用这些修改，需使用ALTER语句更新对应词典的定义文件。
- 不支持修改词典的TEMPLATE参数值。
- 不支持仅修改FILEPATH参数而不修改对应的词典定义文件参数。
- 词典定义文件的文件名仅支持小写字母、数据、下划线混合。
- 预定义词典文件位于$GAUSSHOME/share/postgresql/tsearch_data目录下。

语法格式

```
CREATE TEXT SEARCH DICTIONARY name (
    TEMPLATE = template
    [, option = value [, ... ]]
);
```

参数说明

表 21-218 CREATE TEXT SEARCH DICTIONARY 参数说明

参数	描述	取值范围
name	要创建的全文检索词典的名称，可以用模式修饰。	字符串，需符合**标识符命名规范**。
template	模板名。	系统表PG_TS_TEMPLATE中定义的模板：Simple/Synonym/Thesaurus/Ispell/Snowball。
option	全文检索词典特定选项的参数名称。与template对应，不同的词典类型具有不同的参数列表，且与指定顺序无关。	详细参数说明参见**表21-219**。
value	参数值。如果不是简单的标识符或数字，则参数值必须加单引号（标识符和数字同样可以加上单引号）。	-

版权所有 © 华为云计算技术有限公司

表 21-219 Simple、Ispell、Synonym、Thesaurus、以及 Snowball 的 option 参数

option	option参数选项的名称	描述
Simple词典对应的option	STOPWORDS	停用词表文件名，默认后缀名为stop。例如STOPWORDS = french，则实际文件名为french.stop。停用词文件格式为一组word列表，每行定义一个停用词。词典处理时，文件中的空行和空格会被忽略，并将stopword词组转换为小写形式。
	ACCEPT	是否将非停用词设置为已识别。默认值为true。 当Simple词典设置参数ACCEPT=true时，将不会传递任何token给后继词典，此时建议将其放置在词典列表的最后。反之，当ACCEPT=false时，建议将该Simple词典放置在列表中的至少一个词典之前。

 版权所有 © 华为云计算技术有限公司

option	option参数选项的名称	描述
	FILEPATH	停用词文件所在目录。停用词文件可以存放于本地，也可以存放于对象存储服务OBS服务器。如果存在本地，目录格式为'file://absolute_path'。如果存放于OBS服务器，目录格式为'obs://bucket/path accesskey=ak secretkey=sk region=region_name'。注意目录要用单引号包含。默认值为预定义词典文件所在目录。FILEPATH参数必须和STOPWORDS参数同时指定，不允许单独指定。 使用OBS服务器上的停用词文件创建字典的过程如下： 1. 将停用词文件上传到OBS服务器。例如将french.stop文件上传到OBS服务器obsv3.sa-fb-1.externaldemo.com上名为gaussdb的桶中，其URL为https://gaussdb.obsv3.sa-fb-1.externaldemo.com/french.stop。上传文件及查询URL的方式请参考OBS服务的用户手册。 2. 修改$GAUSSHOME/etc/region_map文件，在其中加入一行"region_name": "obs domain"。region_name可以为任意由大小写字母、数字、斜杠（/）或下划线组成的字符串。obs domain为OBS服务器的域名。 示例，将region_name设为rg，region_map包含的内容如："rg": "obsv3.sa-fb-1.externaldemo.com"。 **须知** region_name和obs domain都用双引号，冒号的左边没有空格，右边有1个空格。 3. 执行CREATE TEXT SEARCH DICTIONARY命令创建字典。命令如下： `CREATE TEXT SEARCH DICTIONARY french_dict ( TEMPLATE = pg_catalog.simple, STOPWORDS = french, FILEPATH = 'obs://gaussdb accesskey=xxx secretkey=yyy region=rg' );` 由于french.stop文件放在gaussdb桶的根目录下，因此path为空。
Synonym词典对应的option	SYNONYM	同义词词典的定义文件名，默认后缀名为syn。 文件格式为一组同义词列表，每行格式为"token synonym"，即token和其对应的synonym，中间以空格相连。
	CASESENSITIVE	设置是否大小写敏感，默认值为false，此时词典文件中的token和synonym均会转为小写形式处理。如果设置为true，则不会进行小写转换。

版权所有 © 华为云计算技术有限公司

option	option参数选项的名称	描述
	FILEPATH	同义词词典文件所在目录。目录可以指定为本地目录和OBS目录两种形式。默认值为预定义词典文件所在目录。其中目录格式、以及使用OBS服务器上的文件创建Synonym字典的过程与Simple字典相同。
Thesaurus词典对应的option	DICTFILE	词典定义文件名，默认后缀名为ths。 文件格式为一组同义词列表，每行格式为"sample words : indexed words"，中间冒号（:）作为短语和其替换词间的分隔符。TZ词典处理时，如果有多个匹配的sample words，将选择最长匹配输出。
	DICTIONARY	用于词规范化的子词典名，必须且仅能定义一个。该词典必须是已经存在的，在检查短语匹配之前使用，用于识别和规范输入文本。 如果子词典无法识别输入词，将会报错。此时，需要移除该词或者更新子词典使其识别。此外，可在indexed words的开头放上一个星号（*）来跳过在其上应用子词典，但是所有sample words必须可以被子词典识别。 如果词典文件定义的sample words中，含有子词典中定义的停用词，需要用问号（？）替代停用词。假设a和the是子词典中所定义的停用词，如下： `? one ? two : swsw` 上述同义词组定义会匹配"a one the two"以及"the one a two"，这两个短语均会被swsw替代输出。
	FILEPATH	词典定义文件所在目录。目录可以指定为本地目录和OBS目录两种形式。默认值为预定义词典文件所在目录。其中目录格式、以及使用OBS服务器上的文件创建Synonym字典的过程与Simple字典相同。
Ispell词典	DICTFILE	词典定义文件名，默认后缀名为dict。
	AFFFILE	词缀文件名，默认后缀名为affix。
	STOPWORDS	停用词文件名，默认后缀名为stop，文件格式要求与Simple类型词典的停用词文件相同。
	FILEPATH	词典文件所在目录。可以指定为本地目录和OBS目录两种形式。默认值为预定义词典文件所在目录。其中目录格式、以及使用OBS服务器上的文件创建Synonym字典的过程与Simple字典相同。
Snowball词典	LANGUAGE	语言名，标识使用哪种语言的词干分析算法。算法按照对应语言中的拼写规则，缩减输入词的常见变体形式为一个基础词或词干。
	STOPWORDS	停用词表文件名，默认后缀名为stop，文件格式要求与Simple类型词典的停用词文件相同。

版权所有 © 华为云计算技术有限公司

option	option参数选项的名称	描述
	FILEPATH	词典定义文件所在目录。可以指定为本地目录或者OBS目录。默认值为预定义词典文件所在目录。FILEPATH参数必须和STOPWORDS参数同时指定，不允许单独指定。其中目录格式、以及用OBS服务器上的文件创建Snowball字典的过程与Simple字典相同。

示例

创建一个Ispell词典类型的词典english_ispell（词典定义文件来自开源词典）。

```
CREATE TEXT SEARCH DICTIONARY english_ispell (
    TEMPLATE = ispell,
    DictFile = english,
    AffFile = english,
    StopWords = english,
    FilePath = 'obs://bucket_name/path accesskey=ak secretkey=sk region=rg'
);
```

请参见**配置示例**一节的示例。

相关链接

ALTER TEXT SEARCH DICTIONARY，**CREATE TEXT SEARCH DICTIONARY**

21.13.6.12 ALTER TEXT SEARCH DICTIONARY

功能描述

修改全文检索词典的相关定义，包括参数、名称、所有者以及模式等。

注意事项

- 预定义词典不支持ALTER操作。
- 只有词典的所有者可以执行ALTER操作，系统管理员默认拥有此权限。
- 创建或修改词典之后，任何对于filepath路径下用户自定义的词典定义文件的修改，将不会影响到数据库中的词典。如果需要在数据库中使用这些修改，需使用ALTER TEXT SEARCH DICTIONARY语句更新对应词典的定义文件。

语法格式

- 修改词典定义。
  ```
  ALTER TEXT SEARCH DICTIONARY name ( option = value | option [, ...] );
  ```
- 重命名词典。
  ```
  ALTER TEXT SEARCH DICTIONARY name RENAME TO new_name;
  ```
- 设置词典的所属模式。
  ```
  ALTER TEXT SEARCH DICTIONARY name SET SCHEMA new_schema;
  ```
- 修改词典的所属者。
  ```
  ALTER TEXT SEARCH DICTIONARY name OWNER TO new_owner;
  ```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-220 ALTER TEXT SEARCH DICTIONARY 参数说明

参数	描述	取值范围
name	需要修改的全文检索词典的名称，可以用模式修饰。	有效的全文检索词典名。
option	要修改的全文检索词典特定选项的参数名称。与template对应，不同的词典类型具有不同的参数列表，且与指定顺序无关。 • 不支持修改词典的TEMPLATE参数值。 • 不支持仅修改FILEPATH参数而不修改对应的词典定义文件参数。 • 词典定义文件的文件名仅支持小写字母、数据、下划线混合。	详细参数说明请见**表21-218**中option参数。
value	要修改的参数值。如果省略等号（=）和value，则表示删除该option的先前设置，使用默认值。	对应option定义。
new_name	全文检索词典的新名称。	字符串，需符合**标识符命名规范**。
new_owner	全文检索词典的新所有者。	有效的用户名。
new_schema	全文检索词典的新模式。	有效的模式名。

示例

创建全文检索词典my_dict。

```
CREATE TEXT SEARCH DICTIONARY my_dict (
    TEMPLATE = snowball,
    Language = english,
    StopWords = english
);
```

修改全文检索词典my_dict的Language参数，并删除停用词定义。

```
ALTER TEXT SEARCH DICTIONARY my_dict (Language = dutch,StopWords);
```

更新全文检索词典my_dict定义，实际不更改任何内容。

```
ALTER TEXT SEARCH DICTIONARY my_dict (dummy);
```

相关链接

CREATE TEXT SEARCH DICTIONARY，**DROP TEXT SEARCH DICTIONARY**

版权所有 © 华为云计算技术有限公司

21.13.6.13 DROP TEXT SEARCH DICTIONARY

功能描述

删除全文检索词典。

注意事项

- 预定义词典不支持DROP操作。
- 只有词典的所有者可以执行DROP操作，系统管理员默认拥有此权限。
- 谨慎执行DROP...CASCADE操作，该操作将级联删除使用该词典的文本搜索配置（TEXT SEARCH CONFIGURATION）。

语法格式

```
DROP TEXT SEARCH DICTIONARY [ IF EXISTS ] name [ CASCADE | RESTRICT ]
```

参数说明

表 21-221 DROP TEXT TEXT SEARCH DICTIONARY 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的全文检索词典不存在，那么发出一个Notice而不是报错。	-
name	要删除的词典名称（可指定模式名，否则默认在当前模式下）。	已存在的词典名。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：自动删除依赖于该词典的对象，并依次删除依赖于这些对象的所有对象。如果存在任何一个使用该词典的文本搜索配置，此DROP命令将不会成功。可添加CASCADE以删除引用该词典的所有文本搜索配置以及词典。 • RESTRICT：如果任何对象依赖词典，则拒绝删除该词典。此选项是缺省值。

示例

删除词典english。

```
DROP TEXT SEARCH DICTIONARY english;
```


版权所有 © 华为云计算技术有限公司

相关链接

ALTER TEXT SEARCH DICTIONARY，CREATE TEXT SEARCH DICTIONARY

21.13.6.14 CLUSTER

功能描述

根据索引对表进行聚簇排序。

- CLUSTER基于索引名指定的索引来聚簇由表名指定的表。索引名指定的索引必须已经定义在指定表上。
- 当对一个表聚簇后，该表将基于索引信息进行物理排序。聚簇是一次性操作：当表被更新之后， 更改的内容不会被聚簇。也就是说，系统不会试图按照索引顺序对新的存储内容及更新记录进行重新聚簇。
- 在对表聚簇时，DWS会记录在哪个索引上建立了聚簇。 形式CLUSTER table_name会使用前面所用的同一个索引对表重新聚簇。用户也可以用CLUSTER或ALTER TABLE的SET WITHOUT CLUSTER形式来设置索引用于后续的聚簇操作或清除任何之前的设置。
- 不带任何参数的CLUSTER会将当前用户所拥有的数据库中的已执行过聚簇的所有表重新聚簇，如果是系统管理员调用，则是所有已被聚簇过的表。
- 在对表进行聚簇时，会在其上请求ACCESS EXCLUSIVE锁。这样避免了在CLUSTER完成之前对此表执行其它的操作（包括读写）。

注意事项

- CLUSTER不允许在事务中执行。
- HDFS表不能进行CLUSTER操作。
- 全局临时表不能进行CLUSTER操作。
- 对表执行CLUSTER操作时会触发表重建（表重建过程中会先把数据转储到一个新的数据文件中，重建完成之后会删除原始文件），当表比较大时，重建会消耗较多的磁盘空间。当磁盘空间不足时，要谨慎对待大表CLUSTER操作，防止触发集群只读。
- 存算分离架构下，对CLUSTER全库的操作会有提示不支持。
- 只有行存B-Tree索引支持CLUSTER操作。
- 如果用户只是随机访问表中的行，那么表中数据的实际存储顺序是无关紧要的。但是， 如果对某些数据的访问多于其它数据，而且有一个索引将这些数据分组，那么使用CLUSTER会有所帮助。如果从一个表中请求一定索引范围的值， 或者是一个索引值对应多行，CLUSTER也会有所帮助，因为如果索引标识出第一匹配行所在的存储页，所有其它行也可能已经在同一个存储页里了，因此可以节省磁盘访问的时间并加快查询速度。
- 在聚簇过程中，系统先创建一个按照索引顺序建立的表的临时副本，同时也创建了表上每个索引的临时副本。因此，磁盘上需要有足够的可用空间， 至少等于表大小和索引大小的总和。
- 由于CLUSTER会记忆聚集信息，因此可以在第一次手动对要聚簇的表进行聚簇，然后设置一个定期维护脚本，这样就可以周期地自动对表进行聚簇操作。
- 因为优化器会记录有关表排序的统计信息，所以建议在新的聚簇表上运行ANALYZE。否则，优化器可能会选择错误的查询规划。

版权所有 © 华为云计算技术有限公司

语法格式

- 对指定表进行聚簇排序。
  ```
  CLUSTER [ VERBOSE ] table_name [ USING index_name ];
  ```
- 对指定分区进行聚簇排序。
  ```
  CLUSTER [ VERBOSE ] table_name PARTITION ( partition_name ) [ USING index_name ];
  ```
- 对已执行过聚簇的表重新进行聚簇。
  ```
  CLUSTER [ VERBOSE ];
  ```

参数说明

表 21-222 CLUSTER 参数说明

参数	描述	取值范围
VERBOSE	启用显示进度信息。	-
table_name	表名称。	已存在的表名称。
index_name	索引名称。	已存在的索引名称。
partition_name	分区名称。	已存在的分区名称。

示例

创建一个分区表。

```
CREATE TABLE tpcds.inventory_p1
(
   INV_DATE_SK             INTEGER              NOT NULL,
   INV_ITEM_SK             INTEGER              NOT NULL,
   INV_WAREHOUSE_SK        INTEGER              NOT NULL,
   INV_QUANTITY_ON_HAND    INTEGER
)
DISTRIBUTE BY HASH(INV_ITEM_SK)
PARTITION BY RANGE(INV_DATE_SK)
(
      PARTITION P1 VALUES LESS THAN(2451179),
      PARTITION P2 VALUES LESS THAN(2451544),
      PARTITION P3 VALUES LESS THAN(2451910),
      PARTITION P4 VALUES LESS THAN(2452275),
      PARTITION P5 VALUES LESS THAN(2452640),
      PARTITION P6 VALUES LESS THAN(2453005),
      PARTITION P7 VALUES LESS THAN(MAXVALUE)
);
```

创建索引ds_inventory_p1_index1。

```
CREATE INDEX ds_inventory_p1_index1 ON tpcds.inventory_p1 (INV_ITEM_SK) LOCAL;
```

对表tpcds.inventory_p1进行聚簇。

```
CLUSTER tpcds.inventory_p1 USING ds_inventory_p1_index1;
```

对分区p3进行聚簇。

```
CLUSTER tpcds.inventory_p1 PARTITION (p3) USING ds_inventory_p1_index1;
```

对数据库中可以进行聚集的表进聚簇。

```
CLUSTER;
```

 版权所有 © 华为云计算技术有限公司

21.13.7 存储过程&函数&触发器&游标

21.13.7.1 CREATE FUNCTION

功能描述

FUNCTION（函数）是一段可重用的代码块，它接受输入参数，执行特定操作，并返回一个值或结果集。CREATE FUNCTION语句用于在数据库中创建**用户自定义函数（UDF）**，实现DWS当前内置的**系统函数**中还未支持的其他计算功能，覆盖用户特定的复杂计算场景。

函数的主要特点包括：

- 支持输入参数，可根据不同条件返回结果。
- 需声明返回值的数据类型，执行结果返回单一值（标量函数）或表（表值函数）。
- 将复杂逻辑封装为单一函数，可以在SQL语句中像内置函数一样调用，提高代码的复用性。

DWS 函数易变属性的三种行为

SQL中函数的行为特性通常分为三类，包括**immutable**（不可变函数）、**stable**（稳定函数）、**volatile**（易变函数），三者关系如**表21-223**所示：

表 21-223 函数的易变属性

易变属性	定义	典型场景
immutable	immutable函数的结果仅由输入参数决定，与任何外部状态（如数据库当前数据、时间、事务上下文等）无关。无论何时、何地调用，只要输入参数相同，返回结果永远一致。且函数不会修改数据库中的数据。	数学计算、哈希、字符串操作，例如ABS(x)，POWER(a, b)，MD5(str)。
stable	stable函数的结果在单次查询执行期间保持稳定。即在同一个查询的多次调用中，只要输入参数相同，结果一致。但结果可能依赖数据库的当前状态（如表数据、会话变量等），因此在不同查询或事务中可能返回不同结果。	基于当前数据的过滤、聚合，例如部分聚合函数：MAX(col), MIN(col)（在单次扫描中结果稳定）。
volatile	volatile函数是最“不稳定”的类型。其结果可能在每次调用时变化，或可能修改数据库状态（如写入数据）。优化器无法预测其行为，必须每次调用都重新计算。	随机数、时间戳、DML操作，例如随机数函数：RANDOM()（每次调用返回不同值）。

查看函数属于哪种属性，可以查看PG_PROC系统表的provolatile字段（i表示immutable，s表示stable，v表示volatile）。

版权所有 © 华为云计算技术有限公司

函数是否下推到 DN（数据节点）执行（SHIPPABLE 和 NOT SHIPPABLE 介绍）

CREATE FUNCTION语法中，通过SHIPPABLE（下推）和NOT SHIPPABLE（不下推）来控制该函数是否下推到数据节点（DN）执行。

- 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。
- 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。

  用户在定义函数的SHIPPABLE属性时也需特别慎重，SHIPPABLE意味着整个函数会下推到DN上执行，如果设置不当，会导致结果错误等严重问题。

  与定义IMMUTABLE属性一样，SHIPPABLE属性的定义也有诸多约束，简单来说就是函数内不能有不可下推的因素，函数下推到单DN执行后，函数内部的计算逻辑仅依赖当前DN的数据集合。

不能定义为IMMUTABLE的情况，举例如下：

1. 如果自定义函数中引用了表、视图等对象，那么该函数就不能定义为IMMUTABLE，因为当表的数据发生变化的时候，函数的返回值可能发生变化。
2. 如果自定义函数中引用了STABLE/VOLATILE类型的函数，那么该函数不能定义为IMMUTABLE。
3. 如果自定义函数中有不下推的因素，则该函数不能定义成IMMUTABLE，因为IMMUTABLE意味着要下推到DN执行，与函数内部的不下推因素相互冲突。典型场景例如，包含不下推的函数、语法等。
4. 如果自定义函数中含有聚合运算，但聚合运算的运算需要生成STREAM计划才能完成计算的（部分结果在DN计算，部分结果在CN计算，例如listagg函数等）。

不能定义为SHIPPABLE的情况，举例如下：

1. 如果函数内部引用了表，并且表为HASH分布，那么该函数通常不能定义为SHIPPABLE。
2. 函数内部有不可下推的因素，函数，语法等，那么该函数不能定义为SHIPPABLE，可参考“语句下推调优”章节。
3. 函数内部的计算过程可能需要跨DN数据，这种情况该函数通常不能定义为SHIPPABLE，例如一些聚合运算等。

⚠ 注意

用户在使用DWS时，应该正确指定函数属性，错误指定函数属性不仅会导致查询语句执行效率低，而且可能会导致结果集不稳定的情况。

注意事项

- 如果创建函数时参数或返回值带有精度，不进行精度检测。
- 创建函数时，函数定义中对表对象的操作建议都显式指定模式（例如hr.table1，明确指定是hr模式下面的表），否则可能会导致函数执行异常。
- 在创建函数时，函数内部通过SET语句设置current_schema和search_path无效。函数执行完的search_path和current_schema与函数执行前的search_path和current_schema保持一致。

版权所有 © 华为云计算技术有限公司

- 如果函数参数中带有出参，SELECT调用函数必须缺省出参，CALL调用函数适配Oracle必须指定出参，对于调用重载的带有PACKAGE属性的函数，CALL调用函数可以缺省出参，具体信息参见**CALL**的示例。
- 兼容PostgreSQL风格的函数或者带有PACKAGE属性的函数支持重载。在指定REPLACE的时候，如果参数个数、类型、返回值有变化，不会替换原有函数，而是会建立新的函数。
- SELECT调用可以指定不同参数来进行同名函数调用。由于语法CALL适配自Oracle，因此不支持调用不带有PACKAGE属性的同名函数。
- 在创建function时，不能在avg函数外面嵌套其他agg函数，或者其他系统函数。
- 在非逻辑集群模式下，暂不支持将返回值、参数以及变量设置为建在非系统默认安装Node Group的表，sql function内部语句暂不支持对建在非系统默认安装Node Group的表操作。
- 在逻辑集群模式下，如果函数返回值和参数是用户表类型，所有涉及表都必须在同一逻辑集群内；如果函数内部涉及对多个逻辑集群表操作，函数定义时不能为IMMUTABLE和SHIPPABLE类型，以避免函数被下推到DN执行。
- 在逻辑集群模式下，函数参数、返回值不能用%type引用表字段类型，否则会导致函数创建失败。
- 新创建的函数默认会给PUBLIC授予执行权限（详见**GRANT**）。用户可以选择收回PUBLIC默认执行权限，然后根据需要将执行权限授予其他用户，为了避免出现新函数能被所有人访问的时间窗口，应在一个事务中创建函数并且设置函数执行权限。
- 在多个CN的集群中，不能将函数的入参或出参设置为临时表类型，因为在非当前连接CN上执行创建函数过程中，无法单独根据表名获取到正确的临时schema，进而无法获取到准确的表类型。

语法格式

- 兼容PostgreSQL风格的创建自定义函数语法。

```
CREATE [ OR REPLACE ] FUNCTION function_name
   ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ]} [, ...] ] )
   [ RETURNS rettype [ DETERMINISTIC ] | RETURNS TABLE ( { column_name column_type }
[, ...] )]
   LANGUAGE lang_name
   [
     {IMMUTABLE | STABLE | VOLATILE }
      | {SHIPPABLE | NOT SHIPPABLE}
      | WINDOW
      | [ NOT ] LEAKPROOF
      | {CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
      | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER |
AUTHID CURRENT_USER}
      | {FENCED | NOT FENCED}
      | {PACKAGE}

      | COST execution_cost
      | ROWS result_rows
      | SET configuration_parameter { {TO | =} value | FROM CURRENT }}
    ][...]
   {
      AS 'definition'
      | AS 'obj_file', 'link_symbol'
   }
```

- Oracle风格的创建自定义函数的语法。

```
CREATE [ OR REPLACE ] FUNCTION function_name
   ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ] } [, ...] ] )
   RETURN rettype [ DETERMINISTIC ]
   [
```

 版权所有 © 华为云计算技术有限公司

```
      {IMMUTABLE  | STABLE  | VOLATILE }
      | {SHIPPABLE | NOT SHIPPABLE}
      | {PACKAGE}
      | {FENCED | NOT FENCED}
      | [ NOT  ] LEAKPROOF
      | {CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
      | {[ EXTERNAL  ] SECURITY INVOKER  | [ EXTERNAL  ] SECURITY DEFINER |
AUTHID DEFINER | AUTHID CURRENT_USER
}
      | COST execution_cost
      | ROWS result_rows
      | SET configuration_parameter { {TO | =} value  | FROM CURRENT

  ][...]

  {
    IS  | AS
} plsql_body
/
```

参数说明

表 21-224 CREATE FUNCTION 参数说明

参数	描述	取值范围
function_name	要创建的函数名称。 **注意** 如果创建的函数名与系统函数同名，建议指定schema。调用自定义函数时需指定schema，否则系统会优先调用系统函数。	字符串，需符合标识符的命名规范。参见**标识符命名规范**。
argname	函数参数的名称。	字符串，需符合标识符的命名规范。
argmode	函数参数的模式。	IN，OUT，INOUT或VARIADIC。 • IN：输入参数，缺省值是IN。 • OUT：输出参数，只有OUT模式的参数后面能跟VARIADIC。 • INOUT：既是输入又是输出参数。 • OUT VARIADIC: 表示输出参数，且声明为数组类型。 **注意** OUT和INOUT模式的参数不能用在RETURNS TABLE的函数定义中。
argtype	函数参数的数据类型。	常用的数据类型参见**数据类型**。
expression	函数参数的默认表达式。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
rettype	函数返回值的数据类型。	如果存在OUT或IN OUT参数，可以省略RETURNS子句。如果该子句存在，则必须和输出参数所表示的结果类型一致。如果有多个输出参数，则为RECORD，否则与单个输出参数的类型相同。 SETOF修饰词表示该函数将返回一个集合，而不是单独一项。
DETERMINISTIC	为适配Oracle SQL语法，未实现功能，不推荐使用。	-
column_name	字段名称。	-
column_type	字段类型。	-
definition	一个定义函数的字符串常量，含义取决于语言。它可以是一个内部函数名字、一个指向某个目标文件的路径、一个SQL查询、一个过程语言文本。	-
LANGUAGE lang_name	用于实现函数的语言的名字。可以是SQL，internal，或者是用户定义的过程语言名字。为了保证向下兼容，该名字可以用单引号（包围）。若采用单引号，则引号内必须为大写。	-
WINDOW	表示该函数是窗口函数，替换函数定义时不能改变WINDOW属性。 **注意** 自定义窗口函数只支持LANGUAGE是internal，并且引用的内部函数必须是窗口函数。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
IMMUTABLE	表示该函数在给出同样的参数值时总是返回同样的结果。	如果函数的入参是常量，会在优化器阶段计算该函数的值。益处是可以尽早获取表达式的值，从而能更准确的进行代价估算，生成的执行计划也更优。 用户自定义的IMMUTABLE的函数是会被自动下推到DN执行的，但是这样可能有潜在的风险，即如果用户错误定义了函数的IMMUTABLE属性，但是函数执行的过程并不是IMMUTABLE的，那么可能会导致结果错误等严重问题。因此，用户在指定函数的属性为IMMUTABLE的时候，要特别慎重。 举例如下： 1. 如果自定义函数中引用了表，视图等对象，那么该函数就不能定义为IMMUTABLE，因为当表的数据发生变化的时候，函数的返回值可能发生变化。 2. 如果自定义函数中引用了STABLE/VOLATILE类型的函数，那么该函数不能定义为IMMUTABLE。 3. 如果自定义函数中有不下推的因素，则该函数不能定义成IMMUTABLE，因为IMMUTABLE意味着要下推到DN执行，与函数内部的不下推因素相互冲突。典型场景例如，包含不下推的函数、语法等。 4. 如果自定义函数中含有聚合运算，但聚合运算的运算需要生成STREAM计划才能完成计算的（部分结果在DN计算，部分结果在CN计算，例如listagg函数等）。 同时，为了防止这种情况下可能出现严重问题，数据库内部可以通过设置behavior_compat_options=‘check_function_conflicts’来开启对函数定义冲突的检查，目前可以识别出上述1和2场景。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
STABLE	表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。	-
VOLATILE	表示该函数值可以在一次表扫描内改变，因此不会做任何优化。	-
SHIPPABLE NOT SHIPPABLE	表示该函数是否可以下推到DN上执行。	• 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。 • 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。 用户在定义函数的SHIPPABLE属性时也需特别慎重，SHIPPABLE意味着整个函数会下推到DN上执行，如果设置不当，会导致结果错误等严重问题。 与定义IMMUTABLE属性一样，SHIPPABLE属性的定义也有诸多约束，简单来说就是函数内不能有不可下推的因素，函数下推到单DN执行后，函数内部的计算逻辑仅依赖当前DN的数据集合。 举例如下： 1. 如果函数内部引用了表，并且表为HASH分布，那么该函数通常不能定义为SHIPPABLE。 2. 函数内部有不可下推的因素，函数，语法等，那么该函数不能定义为SHIPPABLE，可参考语句下推调优。 3. 函数内部的计算过程可能需要跨DN数据，这种情况该函数通常不能定义为SHIPPABLE，例如一些聚合运算等。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PACKAGE	表示该函数是否支持重载。PostgreSQL风格的函数本身就支持重载，此参数主要是针对Oracle风格的函数。	• 不允许package函数和非package函数重载或者替换。 • package函数不支持VARIADIC类型的参数。 • 不允许修改函数的package属性。
LEAKPROOF	指出该函数的参数只包括返回值。LEAKPROOF只能由系统管理员设置。	-
CALLED ON NULL INPUT	表明该函数的某些参数是NULL的时候可以按照正常的方式调用。该参数可以省略。	-
RETURNS NULL ON NULL INPUT STRICT	STRICT用于指定如果函数的某个参数是NULL，此函数总是返回NULL。如果声明了这个参数，当有NULL值参数时该函数不会被执行；而只是自动返回一个NULL结果。 RETURNS NULL ON NULL INPUT和STRICT的功能相同。	-
EXTERNAL	目的是和SQL兼容，是可选的，这个特性适合于所有函数，而不仅是外部函数。	-
SECURITY INVOKER AUTHID CURRENT_USER	表明该函数将带着调用它的用户的权限执行。该参数可以省略。 SECURITY INVOKER和AUTHID CURRENT_USER的功能相同。	-
SECURITY DEFINER AUTHID DEFINER	声明该函数将以创建它的用户的权限执行。 AUTHID DEFINER和SECURITY DEFINER的功能相同。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FENCED NOT FENCED	该函数只对用户定义的C函数生效，声明函数是在保护模式还是非保护模式下执行。如果函数声明为NOT FENCED模式，则函数的执行在CN或者DN进程中进行。如果函数声明为FENCED模式，则函数在新fork的进程执行，这样函数的异常不会影响CN或者DN进程。	FENCED/NOT FENCED模式的选择： ● 正在开发或者调试的Function使用FENCED模式。开发测试完成，使用NOT FENCED模式执行，减少fork进程以及通信的开销。 ● 复杂的操作系统操作，例：打开文件，信号处理，线程处理等操作，使用FENCED模式。否则可能影响DWS数据库的执行。 ● 默认值为FENCED。
COST execution_cost	用来估计函数的执行成本。execution_cost以cpu_operator_cost为单位。	正数。
ROWS result_rows	估计函数返回的行数。用于函数返回的是一个集合。	正数，默认值是1000行。
configuration_parameter	该函数配置的参数值。	● **value** 把指定的数据库会话参数值设置为给定的值。如果value是DEFAULT或者RESET，则在新的会话中使用系统的缺省设置。OFF关闭设置。 取值范围：字符串 – DEFAULT – OFF – RESET 指定默认值。 ● **from current** 取当前会话中的值设置为configuration_parameter的值。
plsql_body	PL/SQL存储过程体。 当在函数中创建用户时，日志中会记录密码的明文。因此不建议用户在函数中创建用户。	-

版权所有 © 华为云计算技术有限公司

示例：创建一个求两个整数求和的 SQL 函数

语法如下：

```
CREATE  OR REPLACE FUNCTION func_add_sql(
integer,                --第一个整数参数，未命名，使用位置参数$1引用
integer                 --第二个整数参数，未命名，使用位置参数$2引用
)
RETURNS integer             --指定函数返回类型为整数
    AS 'select $1 + $2;'    --函数体：使用位置参数$1和$2相加
    LANGUAGE SQL            --使用SQL语言实现该函数
    IMMUTABLE               --函数类型为IMMUTABLE，表示相同输入总是返回相同结果
    RETURNS NULL ON NULL INPUT;       --等同于STRICT，任何参数为NULL时，函数返回NULL
```

执行函数，返回两个入参的求和值：

```
SELECT func_add_sql(8,3);
```

func_add_sql
11

执行函数，输入为Null，返回为Null。

```
SELECT func_add_sql(Null,3);
```

func_add_sql
(Null)

示例：创建一个处理整数自增的 PL/pgSQL 语言函数

语法如下：

```
CREATE OR REPLACE FUNCTION func_increment_plsql(
i integer                           --输入参数，整数类型
) RETURNS integer                       --函数返回值为整数类型
AS $$
       BEGIN                        --表示函数体开始
              RETURN i + 1;           --返回输入参数加1的结果
       END;
$$ LANGUAGE plpgsql;                    --指定函数使用PL/pgSQL语言编写
```

执行函数，返回整数自增结果。

```
SELECT func_increment_plsql(5);
```

func_increment_plsql
6

入参由一个表numbers的第一列指定，使用函数计算后，将计算结果返回到第二列：

```
DROP TABLE IF EXISTS numbers;
CREATE TABLE numbers (value integer);
INSERT INTO numbers VALUES (1), (2), (3), (4), (5);
SELECT value, func_increment_plsql(value) AS incremented_value FROM numbers ORDER BY 1;
```

value	incremented_value
1	2
2	3
3	4
4	5
5	6

版权所有 © 华为云计算技术有限公司

示例：创建一个返回 RECORD 类型的 PL/pgSQL 语言函数

```
CREATE OR REPLACE FUNCTION compute(
i int,                          --输入参数，整数类型
out result_1 bigint,            --输出结果1，大范围整数类型
out result_2 bigint             --输出结果2，大范围整数类型
)
RETURNS SETOF RECORD            --结果返回一组记录
AS $$
BEGIN
    result_1 = i + 1;           --计算第一个结果值，输入值+1
    result_2 = i * 10;          --计算第二个结果值，输入值乘以10
RETURN next;                    --返回当前计算结果并继续（允许返回多行）
END;
$$LANGUAGE plpgsql;             --指定函数使用PL/pgSQL语言编写
```

单次调用函数，返回结果：

```
SELECT * FROM compute(5);
```

result_1	result_2
6	50

生成一个包含1到3的序列，并对序列的每个值调用一次compute函数。

```
SELECT i, (compute(i)).*        --先调用compute函数，再展开函数返回的所有列
FROM generate_series(1, 3)      --生成一个包含1，2，3的临时序列，用于i的赋值
AS t(i);                        --将结果作为临时表，别名为t，列名为i
```

i	result_1	result_2
1	2	10
2	3	20
3	4	30

与一个表test_data的数据结合使用，表中的value列作为函数compute的入参值，计算结果：

```
DROP TABLE IF EXISTS test_data;
CREATE TABLE test_data (id serial, value int);
INSERT INTO test_data (value) VALUES (2), (4), (6);

SELECT id, value, (compute(value)).*
FROM test_data ORDER BY 1;
```

id	value	result_1	result_2
1	2	3	20
2	4	5	40
3	6	7	60

示例：创建一个返回结果包含多个输出参数的 SQL 语言函数

语法如下：

```
CREATE OR REPLACE FUNCTION func_dup_sql(
in int,                          --输入参数，整数类型
out f1 int,                      --输出参数1，整数类型
out f2 text)                     --输出参数2，文本类型
 AS $$
SELECT $1, CAST($1 AS text) || ' is text'       --函数体，查询返回两列，第一列是输入参数$1，第二列是将参数转为文本后拼接'is text'
 $$   LANGUAGE SQL;
```

执行函数。

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM func_dup_sql(42);
```

f1	f2
42	42 is text

示例：创建 package 属性的重载函数

定义了两个重载的函数，名为package_func_overload，它们位于名为package的包中。这两个函数根据参数类型的不同有不同的实现。

```
CREATE OR REPLACE FUNCTION package_func_overload(
col int,                                    --输入参数1，整数类型
col2  int                                    --输入参数2，整数类型
)
RETURN integer PACKAGE
AS
DECLARE
    col_type text;                  --声明局部变量
BEGIN
     col := 122;                    --将第一个参数的值固定为122
     dbms_output.put_line('two int parameters ' || col2);
     RETURN 0;                       --返回整数值0
END;
/

CREATE OR REPLACE FUNCTION package_func_overload(
col int,
col2 smallint
)
RETURN integer PACKAGE
AS
DECLARE
    col_type text;
BEGIN
     col := 122;
     dbms_output.put_line('two smallint parameters ' || col2);
     RETURN 0;
END;
/
```

函数下推逻辑及示例

函数下推逻辑分为两部分：

1. 函数体是否下推：通过函数属性（SHIPPABLE）判断。
2. 函数体内的语句是否下推：当函数体不下推的时候，函数体内的SQL语句进行独立的下推判断（优化器自动判断）。

调用函数的语句不包含用户表

当调用函数的语句不包含用户表的时候，这个语句总是在CN上执行，函数体不下推，此时执行逻辑如下：

1. 先执行函数：语句中没有用户表，这个语句总是在CN上执行。
2. 接着执行函数体的语句，如果不存在不下推的因素，那么总是可以下推的（走下推框架：FQS 或 stream计划）。

版权所有 © 华为云计算技术有限公司

⚠ 注意

- 单独调用函数（调用函数的语句不包含用户表），函数中包含表关联查询的SQL，查询结果是准确的。
- 函数下推只有在函数调用语句中直接出现用户表的时候才有意义。
- 函数体里面的执行计划没办法直接对外展示，可以设置enable_track_record_subsql=on记录函数体内部的语句。

示例：函数体为NOT SHIPPABLE，调用函数的语句不包含用户表。

```
CREATE TABLE student (
        id integer,
        name character varying(50),
        age integer
)
WITH (orientation=row, compression=no)
DISTRIBUTE BY HASH(id);

CREATE TABLE stuscore(
        id integer,
        score integer
)
WITH (orientation=row, compression=no)
DISTRIBUTE BY HASH(id);

INSERT INTO student VALUES (1,'bob',17),(3,'peter',18),(4,'david',15),(8,'alice',17);
INSERT INTO stuscore VALUES (1,88),(3,68),(8,100),(4,90);

--创建一个NOT SHIPPABLE的函数
CREATE OR REPLACE FUNCTION get_student_id(name character varying)
    RETURNS TABLE(id integer)
    LANGUAGE plpgsql
    VOLATILE NOT FENCED NOT SHIPPABLE
AS $function$
DECLARE
BEGIN
    RETURN QUERY (SELECT id FROM mytest.student WHERE name=$1);
END $function$;
```

查看单独调用函数的计划：

```
EXPLAIN VERBOSE SELECT * FROM get_student_id('alice');
```

从返回结果看，计划中没有函数是否下推的说明，因为函数下推只有在函数调用语句中直接出现用户表的时候才有意义。

版权所有 © 华为云计算技术有限公司

```
QUERY PLAN
 id |            operation             | E-rows | E-distinct | E-width | E-costs
----+----------------------------------+--------+------------+---------+---------
  1 | ->  Function Scan on public.get_student_id |  1000 |            |       4 | 10.25


        Predicate Information (identified by plan id)
-------------------------------------------------------------
  1 --Function Scan on public.get_student_id
        Function Call: get_student_id('alice'::character varying)


      Targetlist Information (identified by plan id)
-----------------------------------------------------
  1 --Function Scan on public.get_student_id
        Output: id
```

调用函数的语句包含用户表

当调用函数的语句包含用户表的时候，语句是否下推由函数属性（SHIPPABLE）决定。

示例：函数属性为NOT SHIPPABLE，调用函数的语句包含用户表。

查看调用函数的计划，函数不可下推：

```
EXPLAIN VERBOSE SELECT get_student_id('alice') FROM stuscore;
```

 版权所有 © 华为云计算技术有限公司

```
QUERY PLAN
 id |                operation                | E-rows | E-distinct | E-width | E-costs
----+-----------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on stuscore "_REMOTE_TABLE_QUERY_" | 30000 |        |       0 | 157.35

            SQL Diagnostic Information

-------------------------------------------------

SQL is not plan-shipping

        reason: Function get_student_id() can not be shipped
```

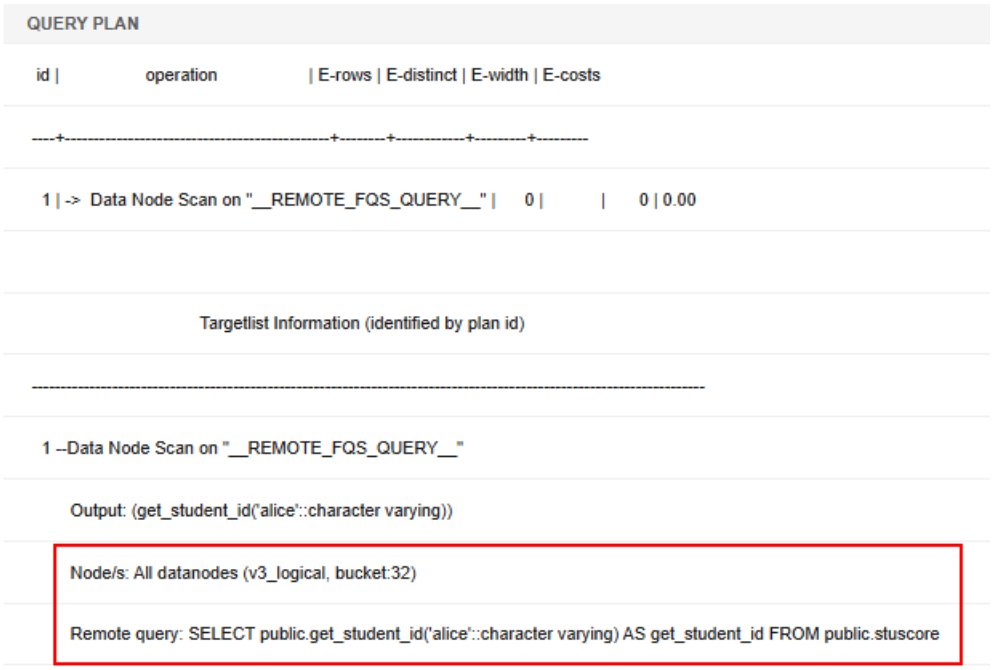
示例：函数属性SHIPPABLE，调用函数的语句包含用户表。

```
--修改函数属性为SHIPPABLE。
CREATE OR REPLACE FUNCTION get_student_id(name character varying)
    RETURNS TABLE(id integer)
    LANGUAGE plpgsql
    VOLATILE NOT FENCED SHIPPABLE
AS $function$
DECLARE
BEGIN
    RETURN QUERY (SELECT id FROM mytest.student WHERE name=$1);
END $function$;
```

查看调用函数的计划，函数可下推。

```
EXPLAIN VERBOSE SELECT get_student_id('alice') FROM stuscore;
```

 版权所有 © 华为云计算技术有限公司



相关链接

ALTER FUNCTION，DROP FUNCTION

21.13.7.2 ALTER FUNCTION

功能描述

修改自定义函数的属性。

注意事项

- 只有该函数的所有者或者授予了函数ALTER权限的用户才有权限执行ALTER FUNCTION，系统管理员默认拥有该权限。
- 要修改函数的所有者的用户必须是新拥有角色的直接或间接成员。
- 要修改函数的所有者，新所有者必须拥有该函数所属模式的CREATE权限。
- 要修改函数的所属模式，必须拥有新模式的CREATE权限。
- 如果函数中涉及对临时表相关的操作，则无法使用ALTER FUNCTION。

语法格式

- 修改自定义函数的附加参数。

```
ALTER FUNCTION function_name ( [ { [ argmode ] [ argname ] argtype} [, ...] ] )
    action [ ... ] [ RESTRICT ];
```

其中附加参数action子句语法为：

```
{CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT}
 | {IMMUTABLE | STABLE | VOLATILE}
 | {SHIPPABLE | NOT SHIPPABLE}
 | {NOT FENCED | FENCED}
 | {MASKED | NOT MASKED}
 | [ NOT ] LEAKPROOF
 | { [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER }
```

 版权所有 © 华为云计算技术有限公司

```
| AUTHID { DEFINER | CURRENT_USER }
| COST execution_cost
| ROWS result_rows
| SET configuration_parameter { { TO | = } { value | DEFAULT }| FROM CURRENT}
| RESET {configuration_parameter | ALL}
```

- 修改自定义函数的名称，支持新函数名前带有原函数所在的schema名，不支持同时修改schema名。

```
ALTER FUNCTION funname ( [ { [ argmode ] [ argname ] argtype} [, ...] ] )
    RENAME TO new_name;
ALTER FUNCTION funname ( [ { [ argmode ] [ argname ] argtype} [, ...] ] )
    RENAME TO schema.new_name;
```

- 修改自定义函数的所属者。

```
ALTER FUNCTION funname ( [ { [ argmode ] [ argname ] argtype} [, ...] ] )
    OWNER TO new_owner;
```

- 修改自定义函数的模式。

```
ALTER FUNCTION funname ( [ { [ argmode ] [ argname ] argtype} [, ...] ] )
    SET SCHEMA new_schema;
```

参数说明

表 21-225 ALTER FUNCTION 参数说明

参数	描述	取值范围
function_name	需要修改的函数名称。	已存在的函数名。
argmode	参数的模式。标识该参数是输入、输出还是输入输出参数。	IN、OUT或者IN OUT。 缺省值是IN。
argname	参数名称。	字符串，需符合**标识符命名规范**。
argtype	函数的参数类型。	有效的类型，请参考**数据类型**。
new_name	函数的新名称。	字符串，需符合**标识符命名规范**。
new_owner	函数的新所有者。	有效的用户名或角色名。
new_schema	函数的新模式。	已存在的模式。
CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT	• CALLED ON NULL INPUT，表明该函数的某些参数是NULL的时可以按照正常的方式调用。缺省时与指定此参数的作用相同。 • STRICT，如果函数的某个参数是NULL时，此函数总是返回NULL。如果指定了该参数，当参数为NULL值时不被调用而只是自动返回一个NULL结果。 • RETURNS NULL ON NULL INPUT和STRICT的功能相同。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
IMMUTABLE | STABLE | VOLATILE	• IMMUTABLE，表示该函数在给出同样的参数值时总是返回同样的结果。 • STABLE，表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。 • VOLATILE，表示该函数值可以在一次表扫描内改变，不会做任何优化。	IMMUTABLE、STABLE和VOLATILE的详细介绍可参考**DWS函数易变属性的三种行为**。
SHIPPABLE | NOT SHIPPABLE	表示该函数是否可以下推到DN上执行。 • 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。 • 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。	-
MASKED | NOT MASKED	表示该函数在guc参数enable_redactcol_computable开启的情况下，是否进行脱敏。 MASKED支持修改系统函数，无法和其他ALTER FUNCTION中的action一起使用。	-
LEAKPROOF	表示该函数没有副作用，指出参数只包括返回值。LEAKPROOF只能由系统管理员设置。	-
EXTERNAL（可选）	允许使用EXTERNAL关键字实现SQL兼容，此功能适用于所有函数而不仅是外部函数。	-
SECURITY INVOKER/AUTHID CURRENT_USER	表明该函数将以调用它的用户的权限执行。缺省时与指定此参数的作用相同。 SECURITY INVOKER和AUTHID CURRENT_USER的功能相同。	-
SECURITY DEFINER/AUTHID DEFINER	声明该函数将以创建它的用户的权限执行。缺省时与指定此参数的作用相同。 AUTHID DEFINER和SECURITY DEFINER的功能相同。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
COST execution_cost	用来估计函数的执行成本。 execution_cost以cpu_operator_cost为单位。	正数。
ROWS result_rows	估计函数返回的行数。用于函数返回的是一个集合。	正数，默认值为1000行。
configuration_parameter	• value 把指定的数据库会话参数值设置为给定的值。 – DEFAULT或RESET，表示新的会话中使用系统的缺省设置。 – OFF，表示关闭设置。 指定默认值。 • from current 取当前会话中的值设置为configuration_parameter的值。	字符串。

示例

创建计算两个整数和的函数func_add_sql2并返回结果。如果输入为null，则返回null。

```
DROP FUNCTION IF EXISTS func_add_sql2;
CREATE FUNCTION func_add_sql2(num1 integer, num2 integer) RETURN integer
AS
BEGIN
RETURN num1 + num2;
END;
/
;
```

修改函数func_add_sql2的执行规则为IMMUTABLE，即参数不变时返回相同结果。

```
ALTER FUNCTION func_add_sql2(INTEGER, INTEGER) IMMUTABLE;
```

将函数func_add_sql2的名称修改为add_two_number。

```
ALTER FUNCTION func_add_sql2(INTEGER, INTEGER) RENAME TO add_two_number;
```

将函数add_two_number的所有者修改为dbadmin。

```
ALTER FUNCTION add_two_number(INTEGER, INTEGER) OWNER TO dbadmin;
```

相关链接

CREATE FUNCTION，**DROP FUNCTION**

21.13.7.3 DROP FUNCTION

功能描述

删除一个已存在的函数。

版权所有 © 华为云计算技术有限公司

注意事项

- 如果所删除的函数为重载函数，则删除时需要指明该函数的参数类型，如为非重载函数，则可直接指定函数名进行删除。
- 如果函数中涉及对临时表相关操作，则无法使用DROP FUNCTION删除函数。

语法格式

```
DROP FUNCTION [ IF EXISTS ] function_name
[ ( [ {[ argmode ] [ argname ] argtype} [, ...] ] ) [ CASCADE | RESTRICT ] ];
```

参数说明

表 21-226 DROP FUNCTION 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的外表不存在，则发出一个notice而不是抛出一个错误。	-
function_name	要删除的函数名字。	已存在的函数名。
argmode	函数参数的模式。	-
argname	函数参数的名称。	-
argtype	函数参数的数据类型。	-
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖于函数的对象（比如操作符）。 • RESTRICT：如果有任何依赖对象存在，则拒绝删除该函数（缺省行为）。

示例

删除名称为add_two_number的函数。

```
DROP FUNCTION add_two_number;
```

相关链接

ALTER FUNCTION，**CREATE FUNCTION**

21.13.7.4 CREATE PROCEDURE

功能描述

创建一个新的存储过程。

存储过程（PROCEDURE）是由一组预编译的SQL语句组成的集合，可以在数据库中进行存储并反复调用。

版权所有 © 华为云计算技术有限公司

注意事项

- 创建存储过程时，请显式指定表对象的所属模式（如schema.table），否则可能引发执行异常。
- 创建存储过程时，存储过程内部通过SET语句设置的current_schema/search_path无效，函数search_path和current_schema执行完成后与函数执行前保持一致。
- 创建存储过程时，禁止在avg函数外面嵌套其他聚合函数或系统函数。
- 创建存储过程时，参数或返回值带有精度，不进行精度检测。
- 在多个CN的集群中，禁止将存储过程的入参/出参设置为临时表类型（因为在非当前连接CN上执行创建存储过程时，无法单独根据表名获取到正确的临时schema，进而无法获取到准确的表类型。）
- 存储过程可以有多个返回值，也可以没有返回值。无返回值的存储过程调用后会有空的回显。
- 如果存储过程参数中带有出参：SELECT调用存储过程必须缺省出参；CALL调用存储过程适配Oracle；调用非重载函数时必须指定出参；对于重载的PACKAGE函数，out参数可以缺省。具体信息参见**CALL**的示例。
- 存储过程指定PACKAGE属性时支持重载。
- 存储过程不支持分段提交。
- 函数创建也适用于存储过程，更多内容请参考**CREATE FUNCTION**。创建存储过程后查询定义时会返回CREATE FUNCTION的定义。

语法格式

```
CREATE [ OR REPLACE ] PROCEDURE procedure_name
    [ ( {[ argmode ] [ argname ] argtype [ { DEFAULT | := | = } expression ]}[,...]) ]
    [
      { IMMUTABLE | STABLE | VOLATILE }
      | { SHIPPABLE | NOT SHIPPABLE }
      | {PACKAGE}
      | [ NOT ] LEAKPROOF
      | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
      | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER | AUTHID
CURRENT_USER}
      | COST execution_cost
      | ROWS result_rows
      | SET configuration_parameter { [ TO | = ] value | FROM CURRENT }
    ][ ... ]
 { IS | AS }
plsql_body
/
```

参数说明

表 21-227 CREATE PROCEDURE 参数说明

参数	描述	取值范围
OR REPLACE	当存在同名的存储过程时，替换原来的定义。	-
procedure_name	要创建的存储过程名字，可以用模式修饰。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
argmode	参数的模式。 OUT和INOUT模式的参数不能用在RETURNS TABLE的存储过程定义中。	IN，OUT，INOUT或VARIADIC。 ● IN：输入参数，缺省值是IN。 ● OUT：输出参数，只有OUT模式的参数后面能跟VARIADIC。 ● INOUT：既是输入又是输出参数。 ● OUT VARIADIC: 表示输出参数，且声明为数组类型。
argname	参数的名字。 argname和argmode的顺序没有严格要求，推荐按照argname、argmode、argtype的顺序使用。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
argtype	参数的数据类型。	可用的数据类型。
IMMUTABLE | STABLE | VOLATILE	行为约束可选项。各参数的功能与CREATE FUNCTION类似，详细说明见**CREATE FUNCTION**。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
SHIPPABLE | NOT SHIPPABLE	表示该存储过程是否可以下推到DN上执行。 ● 对于IMMUTABLE类型的存储过程，存储过程始终可以下推到DN上执行。 ● 对于STABLE/VOLATILE类型的存储过程，仅当存储过程的属性是SHIPPABLE的时候，存储过程可以下推到DN执行。 用户在定义存储过程的SHIPPABLE属性时也需特别慎重，SHIPPABLE意味着整个存储过程会下推到DN上执行，如果设置不当，会导致结果错误等严重问题。 与定义IMMUTABLE属性一样，SHIPPABLE属性的定义也有诸多约束，简单来说就是存储过程内不能有不可下推的因素，存储过程下推到单DN执行后，存储过程内部的计算逻辑仅依赖当前DN的数据集合。 举例如下： 1. 如果存储过程内部引用了表，并且表为HASH分布，那么该存储过程通常不能定义为SHIPPABLE。 2. 存储过程内部有不可下推的因素，存储过程，语法等，那么该存储过程不能定义为SHIPPABLE，可参考语句下推调优。 3. 存储过程内部的计算过程可能需要跨DN数据，这种情况该存储过程通常不能定义为SHIPPABLE，例如一些聚合运算等。	-
PACKAGE	表示该存储过程是否支持重载。PostgreSQL风格的存储过程本身就支持重载，此参数主要是针对Oracle风格的存储过程。 ● 不允许package存储过程和非package存储过程重载或者替换。 ● package存储过程不支持VARIADIC类型的参数。 ● 不允许修改存储过程的package属性。	-
LEAKPROOF	指出该存储过程的参数只包括返回值。LEAKPROOF只能由系统管理员设置。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CALLED ON NULL INPUT	表明该存储过程的某些参数是NULL的时候可以按照正常的方式调用。该参数可以省略。	-
RETURNS NULL ON NULL INPUT | STRICT	STRICT用于指定如果存储过程的某个参数是NULL，此存储过程总是返回NULL。如果声明了这个参数，当有NULL值参数时该存储过程不会被执行；而只是自动返回一个NULL结果。 RETURNS NULL ON NULL INPUT和STRICT的功能相同。	-
EXTERNAL	目的是和SQL兼容，是可选的，这个特性适合于所有存储过程，而不仅是外部存储过程。	-
SECURITY INVOKER | AUTHID CURRENT_USER	表明该存储过程将带着调用它的用户的权限执行。该参数可以省略。 SECURITY INVOKER和AUTHID CURRENT_USER的功能相同。	-
SECURITY DEFINER | AUTHID DEFINER	声明该存储过程将以创建它的用户的权限执行。 AUTHID DEFINER和SECURITY DEFINER的功能相同。	-
COST execution_cost	用来估计存储过程的执行成本。execution_cost以cpu_operator_cost为单位。	正数。
ROWS result_rows	估计存储过程返回的行数。用于存储过程返回的是一个集合。	正数，默认值是1000行。
configuration_parameter	该存储过程配置的参数值。 ● **value** 把指定的数据库会话参数值设置为给定的值。取值范围：字符串。需指定默认值。 – DEFAULT，在新的会话中使用系统的缺省设置。 – OFF，关闭设置。 – RESET，在新的会话中使用系统的缺省设置。 ● **from current** 取当前会话中的值设置为configuration_parameter的值。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
plsql_body	PL/SQL存储过程体。 当在存储过程体中进行创建用户等涉及用户密码相关操作时，系统表及csv日志中会记录密码的明文。因此不建议用户在存储过程体中进行涉及用户密码的相关操作。	-

示例

- 创建存储过程prc_add，param1为输入参数，param2为输入输出参数，计算param1+param2，并将结果赋值给param2，并输出结果。

```
CREATE OR REPLACE PROCEDURE prc_add
(
    param1 IN INTEGER,
    param2 IN OUT INTEGER
)
AS
BEGIN
   param2:= param1 + param2;
   dbms_output.put_line('result is: '||to_char(param2));
END;
/
```

  调用存储过程prc_add。

```
CALL prc_add(2,3);
```

- 创建存储过程pro_variadic，param1参数模式为VARIADIC，将传入的整数数组转换为TEXT格式，并通过param2输出。

```
CREATE OR REPLACE PROCEDURE pro_variadic (param1 VARIADIC int4[],param2 OUT TEXT)
AS
BEGIN
    param2:= param1::text;
END;
/
```

  执行存储过程pro_variadic。

```
SELECT pro_variadic(VARIADIC param1=> array[1,2,3,4]);
```

- 创建带有PACKAGE属性的存储过程package_func_overload。

```
CREATE OR REPLACE PROCEDURE package_func_overload(col int, col2 out varchar)
PACKAGE
AS
DECLARE
    col_type text;
BEGIN
     col2 := '122';
         dbms_output.put_line('two varchar parameters ' || col2);
END;
/
```

相关链接

删除存储过程参见**DROP PROCEDURE**，调用存储过程参见**CALL**。

版权所有 © 华为云计算技术有限公司

21.13.7.5 DROP PROCEDURE

功能描述

删除已存在的存储过程。

注意事项

无。

语法格式

```
DROP PROCEDURE [ IF EXISTS ] procedure_name ;
```

参数说明

表 21-228 DROP PROCEDURE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的存储过程不存在，发出一个notice而不是抛出一个错误。	-
procedure_name	要删除的存储过程名字。	已存在的存储过程名。

示例

删除存储过程。

```
DROP PROCEDURE prc_add;
```

相关链接

CREATE PROCEDURE

21.13.7.6 CREATE TRIGGER

功能描述

创建一个新触发器。 触发器将与指定的表或视图关联，并在特定条件下执行指定的函数。

注意事项

- 当前仅支持在普通行存表上创建触发器，不支持在列存表、临时表、unlogged表等类型表上创建触发器。
- 如果为同一事件定义了多个相同类型的触发器，则按触发器的名称字母顺序触发它们。
- 一个触发器只能作用在一张表上，对创建的触发器数量无限制但一个表上的触发器越多，性能消耗越大。

版权所有 © 华为云计算技术有限公司

- 触发器常用于多表间数据关联同步场景，对SQL执行性能影响较大，不建议在大数据量同步及对性能要求高的场景中使用。
- 当触发器满足如下条件时，触发语句能和触发器一起下推到DN执行并提升触发器执行性能：
  - 开关enable_trigger_shipping和enable_fast_query_shipping开启（默认均开启）。
  - 源表触发器使用的触发器函数为plpgsql类型（推荐类型）。
  - 源表与触发表分布键的类型、数量完全相同，均为行存表，且所属的nodegroup相同。
  - 原INSERT/UPDATE/DELETE语句条件中包含所有分布键与NEW/OLD等值比较表达式。
  - 原INSERT/UPDATE/DELETE语句在没有触发器的情况下原本就能query shipping。
  - 源表上只有INSERT/UPDATE/DELETE AFTER/BEFORE FOR EACH ROW六类触发器，且所有触发器都可下推。
- 关于触发器种类：
  - INSTEAD OF的触发器必须标记为FOR EACH ROW，并且只能在视图上定义。
  - BEFORE和AFTER触发器作用在视图上时，只能标记为FOR EACH STATEMENT。
  - TRUNCATE类型触发器仅限FOR EACH STATEMENT。

语法格式

```
CREATE [ CONSTRAINT ] TRIGGER trigger_name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }
   ON table_name
   [ FROM referenced_table_name ]
   { NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } }
   [ FOR [ EACH ] { ROW | STATEMENT } ]
   [ WHEN ( condition ) ]
   EXECUTE PROCEDURE function_name ( arguments );
```

其中event包含以下几种：

```
   INSERT
   UPDATE [ OF column_name [, ... ] ]
   DELETE
   TRUNCATE
```

参数说明

表 21-229 CREATE TRIGGER 参数说明

参数	描述	取值范围
CONSTRAINT	指定此参数将创建约束触发器，即触发器作为约束来使用。除了可以使用SET CONSTRAINTS调整触发器触发的时间之外，这与常规触发器相同。 约束触发器必须是AFTER ROW触发器。	-

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
trigger_name	要创建的触发器的名称。 该名称不能限定模式，因为触发器自动继承其所在表的模式，且同一个表的触发器不能重名。 对于约束触发器，使用**SET CONSTRAINTS**修改触发器行为时也使用此名称。	字符串，需符合**标识符命名规范**。
BEFORE	触发器函数是在触发事件发生前执行。	-
AFTER	触发器函数是在触发事件发生后执行，约束触发器只能指定为AFTER。	-
INSTEAD OF	触发器函数直接替代触发事件。	-
event	启动触发器的事件。	取值范围包括：INSERT、UPDATE、DELETE或TRUNCATE，也可以通过OR同时指定多个触发事件。 对于UPDATE事件类型，可以使用下面语法指定列： `UPDATE OF column_name1 [, column_name2 ... ]` 表示只有这些列作为UPDATE语句的目标列时，才会启动触发器，但是INSTEAD OF UPDATE类型不支持指定列信息
table_name	需要创建触发器的表名称。	数据库中已经存在的表名称。
referenced_table_name	约束引用的另一个表的名称。 只能为约束触发器指定，常见于外键约束。由于当前不支持外键，因此不建议使用。	数据库中已经存在的表名称。
DEFERRABLE | NOT DEFERRABLE	约束触发器的启动时机，仅作用于约束触发器。这两个关键字设置该约束是否可推迟。 详细介绍请参见**CREATE TABLE**。	-
INITIALLY IMMEDIATE | INITIALLY DEFERRED	如果约束是可推迟的，则这个子句声明检查约束的缺省时间，仅作用于约束触发器。 详细介绍请参见**CREATE TABLE**。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FOR EACH ROW | FOR EACH STATEMENT	触发器的触发频率。 ● FOR EACH ROW是指该触发器是受触发事件影响的每一行触发一次。 ● FOR EACH STATEMENT是指该触发器是每个SQL语句只触发一次。 未指定时默认值为FOR EACH STATEMENT。约束触发器只能指定为FOR EACH ROW。	-
condition	决定是否实际执行触发器函数的条件表达式。当指定WHEN时，只有在条件返回true时才会调用该函数。 ● 在FOR EACH ROW触发器中，WHEN条件可以通过分别写入OLD.column_name或NEW.column_name来引用旧行或新行值的列。 需要注意，INSERT触发器不能引用OLD和DELETE触发器不能引用NEW。 ● INSTEAD OF触发器不支持WHEN条件。 ● WHEN表达式不能包含子查询。 ● 对于约束触发器，WHEN条件的评估不会延迟，而是在执行更新操作后立即发生。 如果条件返回值不为true，则触发器不会排队等待延迟执行。	-
function_name	用户定义的函数，必须声明为不带参数并返回类型为触发器，在触发器触发时执行。	-
arguments	执行触发器时要提供给函数的可选的以逗号分隔的参数列表。参数是文字字符串常量，简单的名称和数字常量也可以写在这里，但它们都将被转换为字符串。 请检查触发器函数的实现语言的描述，以了解如何在函数内访问这些参数。	-

表 21-230 表和视图上支持的触发器种类：

触发时机	触发事件	行级	语句级
BEFORE	INSERT/UPDATE/DELETE	表	表和视图
	TRUNCATE	不支持	表

版权所有 © 华为云计算技术有限公司

触发时机	触发事件	行级	语句级
AFTER	INSERT/UPDATE/DELETE	表	表和视图
	TRUNCATE	不支持	表
INSTEAD OF	INSERT/UPDATE/DELETE	视图	不支持
	TRUNCATE	不支持	不支持

表 21-231 PLPGSQL 类型触发器函数特殊变量：

变量名	变量含义
NEW	INSERT及UPDATE操作涉及tuple信息中的新值，对DELETE为空。
OLD	UPDATE及DELETE操作涉及tuple信息中的旧值，对INSERT为空。
TG_NAME	触发器名称。
TG_WHEN	触发器触发时机（BEFORE/AFTER/INSTEAD OF）。
TG_LEVEL	触发频率（ROW/STATEMENT）。
TG_OP	触发操作（INSERT/UPDATE/DELETE/TRUNCATE）。
TG_RELID	触发器所在表OID。
TG_RELNAME	触发器所在表名（已废弃，现用TG_TABLE_NAME替代）。
TG_TABLE_NAME	触发器所在表名。
TG_TABLE_SCHEMA	触发器所在表的SCHEMA信息。
TG_NARGS	触发器函数参数个数。
TG_ARGV[]	触发器函数参数列表。

示例

创建源表及触发表。

```
CREATE TABLE test_trigger_src_tbl(id1 INT, id2 INT, id3 INT);
CREATE TABLE test_trigger_des_tbl(id1 INT, id2 INT, id3 INT);
```

创建触发器函数tri_insert_func()。

```
CREATE OR REPLACE FUNCTION tri_insert_func() RETURNS TRIGGER AS
        $$
        DECLARE
        BEGIN
                INSERT INTO test_trigger_des_tbl VALUES(NEW.id1, NEW.id2, NEW.id3);
```

 版权所有 © 华为云计算技术有限公司

```
        RETURN NEW;
    END
    $$ LANGUAGE PLPGSQL;
```

创建触发器函数tri_update_func()。

```
CREATE OR REPLACE FUNCTION tri_update_func() RETURNS TRIGGER AS
    $$
    DECLARE
    BEGIN
        UPDATE test_trigger_des_tbl SET id3 = NEW.id3 WHERE id1=OLD.id1;
        RETURN OLD;
    END
    $$ LANGUAGE PLPGSQL;
```

创建触发器函数tri_delete_func()。

```
CREATE OR REPLACE FUNCTION tri_delete_func() RETURNS TRIGGER AS
    $$
    DECLARE
    BEGIN
        DELETE FROM test_trigger_des_tbl WHERE id1=OLD.id1;
        RETURN OLD;
    END
    $$ LANGUAGE PLPGSQL;
```

创建INSERT触发器。

```
CREATE TRIGGER insert_trigger
    BEFORE INSERT ON test_trigger_src_tbl
    FOR EACH ROW
    EXECUTE PROCEDURE tri_insert_func();
```

创建UPDATE触发器。

```
CREATE TRIGGER update_trigger
    AFTER UPDATE ON test_trigger_src_tbl
    FOR EACH ROW
    EXECUTE PROCEDURE tri_update_func();
```

创建DELETE触发器。

```
CREATE TRIGGER delete_trigger
    BEFORE DELETE ON test_trigger_src_tbl
    FOR EACH ROW
    EXECUTE PROCEDURE tri_delete_func();
```

相关链接

ALTER TRIGGER，**DROP TRIGGER**，**ALTER TABLE**

21.13.7.7 ALTER TRIGGER

功能描述

修改触发器定义。

注意事项

只有触发器所在表的所有者可以执行ALTER TRIGGER操作，系统管理员默认拥有此权限。

版权所有 © 华为云计算技术有限公司

语法格式

```
ALTER TRIGGER trigger_name ON table_name RENAME TO new_name;
```

参数说明

表 21-232 ALTER TRIGGER 参数说明

参数	描述	取值范围
trigger_name	需要修改的触发器的名称。	有效的触发器名。
table_name	需要修改的触发器所在的表名称。	触发器所在的有效表名。
new_name	新的触发器名称。	字符串，需符合**标识符命名规范**。

示例

创建示例表test_trigger_src_tbl和test_trigger_des_tbl。

```
DROP TABLE IF EXISTS test_trigger_src_tbl;
DROP TABLE IF EXISTS test_trigger_des_tbl;

CREATE TABLE test_trigger_src_tbl(id1 INT, id2 INT, id3 INT);
CREATE TABLE test_trigger_des_tbl(id1 INT, id2 INT, id3 INT);
```

创建触发器函数tri_insert_func()。

```
DROP FUNCTION IF EXISTS tri_insert_func;
CREATE OR REPLACE FUNCTION tri_insert_func() RETURNS TRIGGER AS
        $$
        DECLARE
        BEGIN
                INSERT INTO test_trigger_des_tbl VALUES(NEW.id1, NEW.id2, NEW.id3);
                RETURN NEW;
        END
        $$ LANGUAGE PLPGSQL;
```

创建触发器insert_trigger。

```
CREATE TRIGGER insert_trigger
        BEFORE INSERT ON test_trigger_src_tbl
        FOR EACH ROW
        EXECUTE PROCEDURE tri_insert_func();
```

修改触发器insert_trigger。

```
ALTER TRIGGER insert_trigger ON test_trigger_src_tbl RENAME TO delete_trigger_renamed;
```

禁用触发器delete_trigger_renamed。

```
ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER delete_trigger_renamed;
```

禁用表test_trigger_src_tbl所有触发器。

```
ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER ALL;
```

相关链接

CREATE TRIGGER，**DROP TRIGGER**，**ALTER TABLE**

版权所有 © 华为云计算技术有限公司

21.13.7.8 DROP TRIGGER

功能描述

删除触发器。

注意事项

只有触发器的所有者可以执行DROP TRIGGER操作，系统管理员默认拥有此权限。

语法格式

```
DROP TRIGGER [ IF EXISTS ] trigger_name ON table_name [ CASCADE | RESTRICT ];
```

参数说明

表 21-233 DROP TRIGGER 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的触发器不存在，则发出一个notice而不是抛出一个错误。	-
trigger_name	要删除的触发器名字。	已存在的触发器。
table_name	要删除的触发器所在的表名称。	已存在的含触发器的表。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖此触发器的对象。 • RESTRICT：如果有依赖对象存在，则拒绝删除此触发器。此选项为缺省值。

示例

删除触发器insert_trigger。

```
DROP TRIGGER insert_trigger ON test_trigger_src_tbl;
```

相关链接

CREATE TRIGGER，**ALTER TRIGGER**，**ALTER TABLE**

21.13.7.9 FETCH

功能描述

FETCH通过已创建的游标来检索数据。

每个游标都有一个供FETCH使用的关联位置。游标的关联位置可以在查询结果的第一行之前，或者在结果中的任意行，或者在结果的最后一行之后：

版权所有 © 华为云计算技术有限公司

- 游标刚创建完之后，关联位置在第一行之前的。
- 在抓取了一些移动行之后，关联位置在检索到的最后一行上。
- 如果FETCH抓取完了所有可用行，它就停在最后一行后面，或者在反向抓取的情况下是停在第一行前面。
- FETCH ALL或FETCH BACKWARD ALL将总是把游标的关联位置放在最后一行或者在第一行前面。

注意事项

- 如果游标定义了NO SCROLL，则不允许使用例如FETCH BACKWARD之类的反向抓取。
- NEXT，PRIOR，FIRST，LAST，ABSOLUTE，RELATIVE形式在恰当地移动游标之后抓取一条记录。如果后面没有数据行，就返回一个空的结果，此时游标就会停在查询结果的最后一行之后（向后查询时）或者第一行之前（向前查询时）。
- FORWARD和BACKWARD形式在向前或者向后移动的过程中抓取指定的行数，然后把游标定位在最后返回的行上；或者是，如果count大于可用的行数，则在所有行之后（向后查询时）或者之前（向前查询时）。
- RELATIVE 0，FORWARD 0，BACKWARD 0都要求在不移动游标的前提下抓取当前行，也就是重新抓取最近刚抓取过的行。除非游标定位在第一行之前或者最后一行之后，这个动作都应该成功，而在那两种情况下，不返回任何行。
- 当FETCH的游标上涉及列存表时，不支持BACKWARD、PRIOR、FIRST等涉及反向获取操作。

语法格式

```
FETCH [ direction { FROM | IN } ] cursor_name;
```

其中direction子句为可选参数。

```
NEXT
  | PRIOR
  | FIRST
  | LAST
  | ABSOLUTE count
  | RELATIVE count
  | count
  | ALL
  | FORWARD
  | FORWARD count
  | FORWARD ALL
  | BACKWARD
  | BACKWARD count
  | BACKWARD ALL
```

参数说明

表 21-234 FETCH 参数说明

参数	描述	取值范围
direction_clause	定义抓取数据的方向。	参见**表21-235**。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
{ FROM | IN } cursor_name	使用关键字FROM或IN指定游标名称。	已创建的游标的名称。

表 21-235 direction_clause 取值范围

取值范围	描述	备注
NEXT（缺省值）	从当前关联位置开始，抓取下一行。	-
PRIOR	从当前关联位置开始，抓取上一行。	-
FIRST	抓取查询的第一行（和ABSOLUTE 1相同）。	-
LAST	抓取查询的最后一行（和ABSOLUTE -1相同）。	-
ABSOLUTE count	抓取查询中第count行。 ABSOLUTE抓取不会比用相对位移移动到需要的数据行更快，因为下层的实现必须遍历所有中间的行。	count取值范围：有符号的整数 ● count为正数，就从查询结果的第一行开始，抓取第count行。当count小于当前游标位置时，涉及到rewind操作，暂不支持。 ● count为负数或0，涉及到反向扫描操作，暂不支持。
RELATIVE count	从当前关联位置开始，抓取随后或前面的第count行。	count取值范围：有符号的整数 ● count为正数就抓取当前关联位置之后的第count行。 ● count为负数，涉及到反向扫描操作，暂不支持。 ● 如果有数据的话，RELATIVE 0重新抓取当前行。
count	抓取随后的count行（和FORWARD count一样）。	-
ALL	从当前关联位置开始，抓取所有剩余的行（和FORWARD ALL一样）。	-
FORWARD	抓取下一行（和NEXT一样）。	-
FORWARD count	与RELATIVE count的效果相同，从当前关联位置开始，抓取随后或前面的第count行。	-

版权所有 © 华为云计算技术有限公司

取值范围	描述	备注
FORWARD ALL	从当前关联位置开始，抓取所有剩余行。	-
BACKWARD	从当前关联位置开始，抓取前面一行（和PRIOR一样）。	-
BACKWARD count	从当前关联位置开始，抓取前面的count行（向后扫描）。	count取值范围：有符号的整数 • count为正数就抓取当前关联位置之前的第count行。 • count为负数就抓取当前关联位置之后的第abs（count）行。 • 如果有数据的话，BACKWARD 0重新抓取当前行。
BACKWARD ALL	从当前关联位置开始，抓取所有前面的行（向后扫描）。	-

示例

示例一：SELECT语句，用一个游标读取一个表。

建立一个名为cursor1的游标。

```
CURSOR cursor1 FOR SELECT * FROM tpcds.customer_address ORDER BY 1;
```

抓取头3行到游标cursor1里。

```
FETCH FORWARD 3 FROM cursor1;
```

示例二：VALUES子句，用一个游标读取VALUES子句中的内容。

建立一个名为cursor2的游标。

```
CURSOR cursor2 FOR VALUES(1,2),(0,3) ORDER BY 1;
```

抓取头2行到游标cursor2里。

```
FETCH FORWARD 2 FROM cursor2;
```

相关链接

CLOSE，**MOVE**

21.13.7.10 MOVE

功能描述

MOVE在不检索数据的情况下重新定位一个游标。MOVE的作用类似于**FETCH**命令，但只是重定位游标而不返回行。

注意事项

无。

版权所有 © 华为云计算技术有限公司

语法格式

```
MOVE [ direction [ FROM | IN ] ] cursor_name;
```

其中direction子句为可选参数。

```
NEXT
  | PRIOR
  | FIRST
  | LAST
  | ABSOLUTE count
  | RELATIVE count
  | count
  | ALL
  | FORWARD
  | FORWARD count
  | FORWARD ALL
  | BACKWARD
  | BACKWARD count
  | BACKWARD ALL
```

参数说明

MOVE命令的参数与FETCH的相同，详细请参见FETCH的**参数说明**。

说明

成功完成时，MOVE命令将返回一个“MOVE count”的标签，count是一个使用相同参数的FETCH命令会返回的行数（可能为零）。

示例

创建表reason，并向表中插入数据。

```
DROP TABLE IF EXISTS reason;
CREATE TABLE reason
(
  a    int primary key,
  b    int,
  c    int
);

INSERT INTO reason VALUES (1, 2, 3);
```

开始一个事务。

```
START TRANSACTION;
```

定义一个名为cursor1的游标。

```
CURSOR cursor1 FOR SELECT * FROM tpcds.reason;
```

忽略游标cursor1的前3行。

```
MOVE FORWARD 3 FROM cursor1;
```

抓取游标cursor1的前4行。

```
FETCH 4 FROM cursor1;
```

关闭游标。

```
CLOSE cursor1;
```

结束一个事务。

```
END;
```

 版权所有 © 华为云计算技术有限公司

相关链接

CLOSE，FETCH

21.13.7.11 CURSOR

功能描述

CURSOR语句定义游标，用于从查询里面检索出指定的几行数据。

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

注意事项

- 游标命令只能在事务块里使用。
- 通常游标和SELECT一样返回文本格式。因为数据在系统内部是用二进制格式存储的，系统必须对数据做一定转换以生成文本格式。一旦数据是以文本形式返回，客户端应用需要把它们转换成二进制进行操作。使用FETCH语句，游标可以返回文本或二进制格式。
- 应该谨慎使用二进制游标。文本格式一般都比对应的二进制格式占用的存储空间大。二进制游标返回内部二进制形态的数据，可能更易于操作。如果想以文本方式显示数据，则以文本方式检索会为用户节约很多客户端的工作。比如，如果查询从某个整数列返回1，在缺省的游标里将获得一个字符串1，但在二进制游标里将得到一个4字节的包含该数值内部形式的数值（大端顺序）。

语法格式

```
CURSOR cursor_name
    [ BINARY ]  [ NO SCROLL ]  [ { WITH | WITHOUT } HOLD ]
    FOR query ;
```

参数说明

表 21-236 CURSOR 参数说明

参数	描述	取值范围
cursor_name	要创建的游标名。	字符串，需符合标识符命名规范。
BINARY	指明游标以二进制而不是文本格式返回数据。	-
NO SCROLL	声明游标检索数据行的方式。 • NO SCROLL：声明该游标不能用于以倒序的方式检索数据行。 • 未声明：根据执行计划的不同，自动判断该游标是否可以用于以倒序的方式检索数据行。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
WITH HOLD | WITHOUT HOLD	声明当创建游标的事务结束后，游标是否能继续使用。 ● WITH HOLD：声明该游标在创建它的事务结束后仍可继续使用。 ● WITHOUT HOLD：声明该游标在创建它的事务之外不能再继续使用，此游标将在事务结束时被自动关闭。（默认行为）	-
query	使用SELECT或VALUES子句指定游标返回的行。	SELECT或VALUES子句。

示例

开启事务。

```
START TRANSACTION;
```

创建名称为cursor1的游标。

```
CURSOR cursor1 FOR SELECT * FROM tpcds.customer_address ORDER BY 1;
```

创建名称为cursor2的游标。

```
CURSOR cursor2 FOR VALUES(1,2),(0,3) ORDER BY 1;
```

WITH HOLD游标的使用：

1. 创建一个with hold游标。
   ```
   DECLARE cursor3 CURSOR WITH HOLD FOR SELECT * FROM tpcds.customer_address ORDER BY 1;
   ```
2. 抓取头2行到游标cursor3中。
   ```
   FETCH FORWARD 2 FROM cursor3;
   ```
3. 结束事务：
   ```
   END;
   ```
4. 抓取下一行到游标cursor3中。
   ```
   FETCH FORWARD 1 FROM cursor3;
   ```
5. 关闭游标。
   ```
   CLOSE cursor3;
   ```

相关链接

FETCH

21.13.7.12 CLOSE

功能描述

CLOSE用于释放与游标关联的所有资源。

版权所有 © 华为云计算技术有限公司

注意事项

- 不允许对一个已关闭的游标再进行任何操作。
- 不再使用的游标应将其关闭。
- 当创建游标的事务用COMMIT或ROLLBACK终止之后，每个非持有的已打开游标都隐式关闭。
- 当创建游标的事务通过ROLLBACK退出之后，则该游标将隐式关闭。
- 当创建游标的事务成功提交，则持有的游标将保持打开状态，直到执行了显式的CLOSE或者客户端断开连接为止。
- DWS没有显式的打开游标的OPEN语句，因为一个游标在使用CURSOR命令定义的时候就打开了。可以通过查询系统视图pg_cursors查看所有可用的游标。

语法格式

```
CLOSE { cursor_name | ALL };
```

参数说明

表 21-237 CLOSE 参数说明

参数	描述	取值范围
cursor_name	要关闭的游标的名称。	已存在的游标名称。
ALL	关闭所有已打开的游标。	-

示例

关闭游标。

```
CLOSE cursor1;
```

相关链接

FETCH，**MOVE**

21.13.8 数据脱敏和安全相关

21.13.8.1 CREATE ROW LEVEL SECURITY POLICY

功能描述

对表创建行访问控制策略。

对表创建行访问控制策略时，需打开该表的行访问控制开关（ALTER TABLE ... ENABLE ROW LEVEL SECURITY | ALTER FOREIGN TABLE ... ENABLE ROW LEVEL SECURITY）策略才能生效，否则不生效。

当前行访问控制会影响数据表的读取操作（SELECT、UPDATE、DELETE），暂不影响数据表的写入操作（INSERT、MERGE INTO）。表所有者或系统管理员可以在USING

版权所有 © 华为云计算技术有限公司

子句中创建表达式，并在客户端执行数据表读取操作时，数据库后台在查询重写阶段会将满足条件的表达式拼接并应用到执行计划中。针对数据表的每一条元组，当USING表达式返回TRUE时，元组对当前用户可见，当USING表达式返回FALSE或NULL时，元组对当前用户不可见。

行访问控制策略名称是针对表的，同一个数据表上不能有同名的行访问控制策略；对不同的数据表，可以有同名的行访问控制策略。

行访问控制策略可以应用到指定的操作（SELECT、UPDATE、DELETE、ALL），ALL表示会影响SELECT、UPDATE、DELETE三种操作；定义行访问控制策略时，若未指定受影响的相关操作，默认为ALL。

行访问控制策略可以应用到指定的用户（角色），也可应用到全部用户（PUBLIC）；定义行访问控制策略时，若未指定受影响的用户，默认为PUBLIC。

注意事项

- 支持对行存表、行存分区表、列存表、列存分区表、复制表、unlogged表、hash表、非EXTERNAL SCHEMA的外表定义行访问控制策略。
- 不支持HDFS表、EXTERNAL SCHEMA的外表、临时表定义行访问控制策略。
- 不支持对视图定义行访问控制策略。
- 同一张表上可以创建多个行访问控制策略，一张表最多创建100个行访问控制策略。
- 具有管理员权限的用户、初始运维用户(Ruby)、表的owner和表的owner角色组成员不受行访问控制影响，可以查看表的全量数据。
- 通过SQL语句、视图、函数、存储过程查询包含行访问控制策略的表，都会受影响。
- 不支持对行访问控制策略依赖的列进行类型修改。例如，不支持如下修改：
  ```
  ALTER TABLE public.all_data ALTER COLUMN role TYPE text;
  ```

语法格式

```
CREATE [ ROW LEVEL SECURITY ] POLICY policy_name ON table_name
    [ AS { PERMISSIVE | RESTRICTIVE } ]
    [ FOR { ALL | SELECT | UPDATE | DELETE } ]
    [ TO { role_name | PUBLIC } [, ...] ]
    USING ( using_expression )
```

参数说明

表 21-238 CREATE ROW LEVEL SECURITY POLICY 参数说明

参数	描述	取值范围
policy_name	行访问控制策略名称，同一个数据表上行访问控制策略名称不能相同。	-
table_name	行访问控制策略的表名。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
PERMISSIVE	指定行访问控制策略的类型为宽容性策略。对于一个给定的查询，将使用“OR”操作符将所有的宽容性策略组合。行访问控制策略的类型默认为宽容性策略。	-
RESTRICTIVE	指定行访问控制策略的类型为限制性策略。对于一个给定的查询，将使用“AND”操作符将所有的限制性策略组合。 至少需要一条宽容性策略允许对记录的访问。如果只有限制性策略存在，则不能访问任何记录。当宽容性和限制性策略共存时，只有当记录能通过至少一条宽容性策略以及所有的限制性策略时，该记录才能访问。	-
command	当前行访问控制影响的SQL操作，可指定操作包括：ALL、SELECT、UPDATE、DELETE。当未指定时，ALL为默认值，涵盖SELECT、UPDATE、DELETE操作。	当command为SELECT时，SELECT类操作受行访问控制的影响，只能查看到满足条件（using_expression返回值为TRUE）的元组数据，受影响的操作包括SELECT，UPDATE ... RETURNING，DELETE ... RETURNING。 当command为UPDATE时，UPDATE类操作受行访问控制的影响，只能更新满足条件（using_expression返回值为TRUE）的元组数据，受影响的操作包括UPDATE，UPDATE ... RETURNING，SELECT ... FOR UPDATE/SHARE。 当command为DELETE时，DELETE类操作受行访问控制的影响，只能删除满足条件（using_expression返回值为TRUE）的元组数据，受影响的操作包括DELETE，DELETE ... RETURNING。 行访问控制策略与适配的SQL语法关系参见**表21-239**：

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
role_name	行访问控制影响的数据库用户。 **须知** 系统管理员不受行访问控制特性影响。	当未指定时，PUBLIC为默认值，PUBLIC表示影响所有数据库用户，可以指定多个受影响的数据库用户。
using_expression	行访问控制的表达式（返回boolean值）。	条件表达式中不能包含AGG函数和窗口（WINDOW）函数。在查询重写阶段，如果数据表的行访问控制开关打开，满足条件的表达式会添加到计划树中。针对数据表的每条元组，会进行表达式计算，只有表达式返回值为TRUE时，行数据对用户才可见（SELECT、UPDATE、DELETE）；当表达式返回FALSE时，该元组对当前用户不可见，用户无法通过SELECT语句查看此元组，无法通过UPDATE语句更新此元组，无法通过DELETE语句删除此元组。

表 21-239 ROW LEVEL SECURITY 策略与适配 SQL 语法关系

Command	SELECT/ALL policy	UPDATE/ALL policy	DELETE/ALL policy
SELECT	Existing row	No	No
SELECT FOR UPDATE/SHARE	Existing row	Existing row	No
UPDATE	No	Existing row	No
UPDATE RETURNING	Existing row	Existing row	No
DELETE	No	No	Existing row
DELETE RETURNING	Existing row	No	Existing row

示例 1：创建行访问控制策略，当前用户只能查看用户自身的数据

1. 创建用户alice和bob。
```
CREATE ROLE alice PASSWORD '{password}';
CREATE ROLE bob PASSWORD '{password}';
```
2. 创建数据表public.all_data。
```
CREATE TABLE public.all_data(id int, role varchar(100), data varchar(100));
```


版权所有 © 华为云计算技术有限公司

3. 向数据表插入数据。

```
INSERT INTO all_data VALUES(1, 'alice', 'alice data');
INSERT INTO all_data VALUES(2, 'bob', 'bob data');
INSERT INTO all_data VALUES(3, 'peter', 'peter data');
```

4. 将表all_data的读取权限赋予alice和bob用户。

```
GRANT SELECT ON all_data TO alice, bob;
```

5. 打开行访问控制策略开关。

```
ALTER TABLE all_data ENABLE ROW LEVEL SECURITY;
```

6. 创建行访问控制策略，当前用户只能查看用户自身的数据。

```
CREATE ROW LEVEL SECURITY POLICY all_data_rls ON all_data USING(role = CURRENT_USER);
```

7. 查看表all_data相关信息。

```
\d+ all_data
                                Table "public.all_data"
 Column |          Type          | Modifiers | Storage  | Stats target | Description
--------+------------------------+-----------+----------+--------------+-------------
 id     | integer                |           | plain    |              |
 role   | character varying(100) |           | extended |              |
 data   | character varying(100) |           | extended |              |
Row Level Security Policies:
    POLICY "all_data_rls"
      USING (((role)::name = "current_user"()))
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, enable_rowsecurity=true
```

8. 当前用户执行SELECT操作。

```
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  1 | alice | alice data
  2 | bob   | bob data
  3 | peter | peter data
(3 rows)
EXPLAIN(COSTS OFF) SELECT * FROM all_data;
         QUERY PLAN
----------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
(3 rows)
```

9. 切换至alice用户。

```
set role alice password '{password}';
```

10. 执行SELECT操作。

```
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  1 | alice | alice data
(1 row)

EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                          QUERY PLAN
---------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'alice'::name)
 Notice: This query is influenced by row level security feature
(5 rows)
```

示例 2：通过行级控制实现分区权限管理

1. 创建用户alice。

```
CREATE ROLE alice PASSWORD '{password1}';
```


版权所有 © 华为云计算技术有限公司

2. 创建范围分区表web_returns_p1，并插入数据。

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(800) END(830) EVERY(1)
);

INSERT INTO web_returns_p1 values (801,17,11,102);
INSERT INTO web_returns_p1 values (802,18,12,103);
```

3. 将表web_returns_p1的读取权限赋予alice用户。

```
GRANT SELECT ON web_returns_p1 TO alice;
```

4. 打开行访问控制策略开关。

```
ALTER TABLE web_returns_p1 ENABLE ROW LEVEL SECURITY;
```

5. 创建行级访问控制策略web_returns_rsl。其中wr_returned_date_sk为web_returns_p1分区表的分区名，801为分区值。

```
CREATE ROW LEVEL SECURITY POLICY web_returns_rsl ON web_returns_p1
USING('wr_returned_date_sk' = '801');
```

6. 将行级访问控制策略web_returns_rsl的赋予用户alice。

```
ALTER ROW LEVEL SECURITY POLICY web_returns_rsl ON web_returns_p1 TO alice;
```

7. 切换至alice用户。

```
set role alice password '{password1}';
```

8. 查询表web_returns_p1。

```
select * from web_returns_p1;
```

相关链接

ALTER ROW LEVEL SECURITY POLICY，**DROP ROW LEVEL SECURITY POLICY**

21.13.8.2 ALTER ROW LEVEL SECURITY POLICY

功能描述

对已存在的行访问控制策略（包括行访问控制策略的名称，行访问控制指定的用户，行访问控制的策略表达式）进行修改。

注意事项

表的所有者或管理员用户才能进行此操作。

语法格式

```
ALTER [ ROW LEVEL SECURITY ] POLICY [ IF EXISTS ] policy_name ON table_name RENAME TO
new_policy_name

ALTER [ ROW LEVEL SECURITY ] POLICY policy_name ON table_name
    [ TO { role_name | PUBLIC } [, ...] ]
    [ USING ( using_expression ) ]
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-240 ALTER ROW LEVEL SECURITY POLICY 参数说明

参数	描述	取值范围
policy_name	行访问控制策略名称。	字符串，符合标识符命名规范。
table_name	行访问控制策略的表名。	已存在的表名。
new_policy_name	新的行访问控制策略名称。	-
role_name	行访问控制策略应用的数据库用户，可以指定多个用户，PUBLIC表示应用到所有用户。	已存在的角色名。
using_expression	行访问控制的表达式，返回值为boolean类型。	-

示例

打开行访问控制策略开关：

```
ALTER TABLE all_data ENABLE ROW LEVEL SECURITY;
```

修改行访问控制all_data_rls的名称：

```
ALTER ROW LEVEL SECURITY POLICY all_data_rls ON all_data RENAME TO all_data_new_rls;
```

修改行访问控制策略影响的用户：

```
ALTER ROW LEVEL SECURITY POLICY all_data_new_rls ON all_data TO alice, bob;
```

修改行访问控制策略表达式：

```
ALTER ROW LEVEL SECURITY POLICY all_data_new_rls ON all_data USING (id > 100 AND role = current_user);
```

相关链接

CREATE ROW LEVEL SECURITY POLICY，**DROP ROW LEVEL SECURITY POLICY**

21.13.8.3 DROP ROW LEVEL SECURITY POLICY

功能描述

删除表上指定的行访问控制策略。

注意事项

仅表的所有者或者管理员用户才能删除表的行访问控制策略。

语法格式

```
DROP [ ROW LEVEL SECURITY ] POLICY [ IF EXISTS ] policy_name ON table_name [ CASCADE | RESTRICT ]
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-241 DROP ROW LEVEL SECURITY POLICY 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的行访问控制策略不存在，发出一个notice而不是抛出一个错误。	-
policy_name	要删除的行访问控制策略的名称。	已存在的行访问控制策略的名称。
table_name	要删除的行访问控制策略所在的数据表名。	已存在的行访问控制策略所在的数据表名。
CASCADE/RESTRICT	仅适配此语法，无对象依赖于该行访问控制策略，CASCADE和RESTRICT效果相同。	-

示例

删除表all_data上的行访问控制策略all_data_rls。

```
DROP ROW LEVEL SECURITY POLICY all_data_rls ON all_data;
```

相关链接

ALTER ROW LEVEL SECURITY POLICY，**CREATE ROW LEVEL SECURITY POLICY**

21.13.8.4 CREATE REDACTION POLICY

功能描述

对表创建数据脱敏策略。

注意事项

- 数据脱敏特性使用feature_support_options参数进行控制，当出现“REDACTION POLICY is not yet supported, please add enable_data_redaction into feature_support_options.“或者“Cannot use the feature of data redaction, please drop existed redaction policy or add enable_data_redaction into feature_support_options.“的报错信息时，说明没有开启数据脱敏特性，需联系技术支持人员设置feature_support_options参数，开启数据脱敏特性。
- 只有表对象的属主和被授予gs_redaction_policy预置角色的用户具有创建脱敏策略的权限。
- 仅支持在普通表和非EXTERNAL SCHEMA的外表（OBS、HDFS、GDS、协调分析外表）创建数据脱敏策略，不支持为系统表、EXTERNAL SCHEMA的外表、临时表、UNLOGGED表以及视图和函数对象创建脱敏策略。
- 不支持通过同义词向普通表对象创建脱敏策略。

版权所有 © 华为云计算技术有限公司

- 表对象与脱敏策略间一一对应。一个脱敏策略是表对象所有脱敏列的集合，可以给脱敏表对象的多个列字段指定脱敏函数，且不同脱敏列对象可以采用不同的脱敏函数。
- 创建脱敏策略时，默认策略生效，即属性enable值为true。
- 具有sysadmin权限的用户，可跳过脱敏策略检查，对脱敏列数据一直具有可见性，即脱敏策略不生效。
- 支持通过指定角色匹配脱敏策略。

语法格式

```
CREATE REDACTION POLICY [IF NOT EXISTS] policy_name ON table_name [ { BEFORE | AFTER }
old_policy_name ]
   [ WHEN (when_expression) ]
   [ ADD COLUMN column_name WITH redaction_function_name ( [ argument [, ...] ] )] [, ... ];
```

参数说明

表 21-242 CREATE REDACTION POLICY 参数说明

参数	描述	取值范围
policy_name	脱敏策略名称。	-
table_name	应用脱敏策略的表名。	-
BEFORE | AFTER	指定当前策略创建的相对位置，通常一张表使用一个脱敏策略即可。缺省时表示：默认当前策略创建在当前系统表中记录的目标表的候选策略的最后一个之后。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
WHEN (when_expression)	WHEN子句指定一个生效表达式。仅当此表达式为真时，脱敏策略才可能生效。	查询语句涉及脱敏表对象时，仅当脱敏策略的WHEN子句表达式为真时，查询对脱敏列数据才可能不可见，即脱敏策略生效。通常，采用WHEN子句来限定脱敏策略的生效用户范围，具有较严格的约束规格。 WHEN子句的规格约束如下： 1. 表达式可以是AND、OR连接的多个子表达式。 2. 每个子表达式仅支持=、<>、!=、>=、>、<=、<七种运算符，左右值只能取常量值或者下列系统常量值之一：SESSION_USER、CURRENT_USER、USER、CURRENT_ROLE、CURRENT_SCHEMA系统常量或者SYS_CONTEXT系统函数。 3. 每个子表达式可以是IN、NOT IN表达式，左值可以是上述2中所列系统常量值，右值数组的每个元素必须是常量值。 4. 每个子表达式可以是**pg_has_role(user, role, privilege)**系统函数。 5. 当脱敏策略永远成立时，即对所有用户（含表对象属主）均生效，建议使用表达式（1=1）创建脱敏策略。 6. WHEN子句缺省时，脱敏策略默认不生效，需用户手动指定WHEN子句表达式。
column_name	脱敏策略应用的表对象的列名。	-
redaction_function_name	对脱敏列应用的脱敏函数。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
arguments	脱敏函数的参数列表。 除系统提供MASK_NONE、MASK_FULL、MASK_PARTIAL三种内置脱敏函数，也支持使用C语言、PL/PGSQL语言创建的用户自定义脱敏函数，函数规格，请参考**数据脱敏函数**。	• MASK_NONE，表示不进行任何脱敏处理。 • MASK_FULL，表示全脱敏成固定值。 • MASK_PARTIAL，表示按指定的字符类型，数值类型或时间类型进行部分脱敏处理。

示例

对指定用户创建脱敏策略。

1. 创建用户alice和matu。
```
CREATE ROLE alice PASSWORD '{password}';
CREATE ROLE matu PASSWORD '{password}';
```
2. 用户alice创建表对象emp并插入数据。
```
CREATE TABLE emp(id int, name varchar(20), salary NUMERIC(10,2));
INSERT INTO emp VALUES(1, 'July', 1230.10), (2, 'David', 999.99);
```
3. 用户alice为表对象emp创建脱敏策略mask_emp，字段salary对用户matu不可见。
```
CREATE REDACTION POLICY mask_emp ON emp WHEN(current_user = 'matu') ADD COLUMN salary
WITH mask_full(salary);
```
4. 用户alice授予用户matu表emp的SELECT权限。
```
GRANT SELECT ON emp TO matu;
```
5. 切至用户matu。
```
SET ROLE matu PASSWORD '{password}';
```
6. 查询表emp，字段salary数据已脱敏。
```
SELECT * FROM emp;
```

对角色创建脱敏策略。

1. 创建角色redact_role。
```
CREATE ROLE redact_role PASSWORD '{password}';
```
2. 将用户matu，alice加入角色redact_role。
```
GRANT redact_role to matu,alice;
```
3. 管理员用户创建表对象emp1并插入数据。
```
CREATE TABLE emp1(id int, name varchar(20), salary NUMERIC(10,2));
INSERT INTO emp1 VALUES(3, 'Rose', 2230.20), (4, 'Jack', 899.88);
```
4. 管理员用户为表对象emp1创建脱敏策略mask_emp1，使字段salary对角色redact_role不可见。
```
CREATE REDACTION POLICY mask_emp1 ON emp1 WHEN(pg_has_role(current_user, 'redact_role',
'member')) ADD COLUMN salary WITH mask_full(salary);
```
   若不指定用户，默认为当前用户current_user。
```
CREATE REDACTION POLICY mask_emp1 ON emp1 WHEN (pg_has_role('redact_role', 'member'))
ADD COLUMN salary WITH mask_full(salary);
```
5. 管理员用户授予用户matu表emp1的SELECT权限。
```
GRANT SELECT ON emp1 TO matu;
```


版权所有 © 华为云计算技术有限公司

6. 切换至用户matu。
```
SET ROLE matu PASSWORD '{password}';
```
7. 查询表emp，字段salary数据已脱敏。
```
SELECT * FROM emp1;
```

相关链接

ALTER REDACTION POLICY，DROP REDACTION POLICY

21.13.8.5 ALTER REDACTION POLICY

功能描述

修改应用在指定表的脱敏策略。

注意事项

只有待修改脱敏策略所应用表对象的属主和被授予gs_redaction_policy预置角色的用户才有修改脱敏策略的权限。

语法格式

- 修改脱敏策略生效表达式。
```
ALTER REDACTION POLICY policy_name ON table_name WHEN (when_expression);
```
- 使脱敏策略生效或失效。
```
ALTER REDACTION POLICY policy_name ON table_name ENABLE | DISABLE;
```
- 重命名脱敏策略。
```
ALTER REDACTION POLICY policy_name ON table_name RENAME TO new_policy_name;
```
- 修改脱敏列，包括新增、修改、删除脱敏列。
```
ALTER REDACTION POLICY policy_name ON table_name
  action;
```
其中，脱敏列操作action可以是以下子句之一：
```
[INHERIT] ADD COLUMN column_name WITH redaction_function_name ( [ argument [, ...] ] )
 | [INHERIT] MODIFY COLUMN column_name WITH redaction_function_name ( [ argument [, ...] ] )
 | DROP COLUMN column_name
```

参数说明

表 21-243 ALTER REDACTION POLICY 参数说明

参数	描述	取值范围
policy_name	待修改的脱敏策略名。	已存在的脱敏策略名。
table_name	待修改的脱敏策略应用的表对象名。	已存在的表名。
when_expression	脱敏策略新的生效表达式。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ENABLE | DISABLE	当前脱敏策略是否生效。 • ENABLE，使之前失效的表对象的脱敏策略重新生效。 • DISABLE，使当前应用在表对象的脱敏策略失效。	-
new_policy_name	新的脱敏策略名。	字符串，需符合**标识符命名规范**。
column_name	脱敏表对象的列字段名。 • 如果新增脱敏列，则指定列字段尚未绑定任何脱敏函数。 • 如果修改脱敏列，则指定列字段为已存在的脱敏列。 • 如果删除脱敏列，则指定列字段为已存在的脱敏列。	-
redaction_function_name	脱敏函数名。	支持的函数请参考**数据脱敏函数**。
arguments	脱敏函数的参数列表。 • MASK_NONE，表示不进行任何脱敏处理。 • MASK_FULL，表示全脱敏成固定值。 • MASK_PARTIAL，表示按指定的字符类型，数值类型或时间类型进行部分脱敏处理。	支持的函数请参考**数据脱敏函数**。

示例

创建用户test_role和示例表emp并插入数据。

```
CREATE ROLE test_role PASSWORD '{Password}';

DROP TABLE IF EXISTS emp;
CREATE TABLE emp(id int, name varchar(20), salary NUMERIC(10,2));
INSERT INTO emp VALUES(1, 'July', 1230.10), (2, 'David', 999.99);
```

为表对象emp创建脱敏策略mask_emp，字段salary对用户test_role不可见。

```
CREATE REDACTION POLICY mask_emp ON emp WHEN(current_user = 'test_role') ADD COLUMN salary
WITH mask_full(salary);
```

修改脱敏策略生效表达式，使其对指定角色生效（若不指定用户，默认对当前用户生效）。

```
ALTER REDACTION POLICY mask_emp ON emp WHEN (pg_has_role(current_user, 'redact_role', 'member'));
ALTER REDACTION POLICY mask_emp ON emp WHEN (pg_has_role('redact_role', 'member'));
```

修改脱敏策略生效表达式，使其对所有用户均生效。

```
ALTER REDACTION POLICY mask_emp ON emp WHEN (1=1);
```

修改脱敏策略，使其失效。

版权所有 © 华为云计算技术有限公司

```
ALTER REDACTION POLICY mask_emp ON emp DISABLE;
```

重新使脱敏策略生效。

```
ALTER REDACTION POLICY mask_emp ON emp ENABLE;
```

重命名脱敏策略为mask_emp_new。

```
ALTER REDACTION POLICY mask_emp ON emp RENAME TO mask_emp_new;
```

新增脱敏列。

```
ALTER REDACTION POLICY mask_emp_new ON emp ADD COLUMN name WITH mask_partial(name, '*', 1, length(name));
```

修改脱敏列name，采用脱敏函数MASK_FULL对name字段数据全脱敏。

```
ALTER REDACTION POLICY mask_emp_new ON emp MODIFY COLUMN name WITH mask_full(name);
```

删除已存在的脱敏列。

```
ALTER REDACTION POLICY mask_emp_new ON emp DROP COLUMN name;
```

相关链接

CREATE REDACTION POLICY，**DROP REDACTION POLICY**

21.13.8.6 DROP REDACTION POLICY

功能描述

删除应用在指定表上的脱敏策略。

注意事项

只有表的属主和被授予gs_redaction_policy预置角色的用户才有权限删除脱敏策略。

语法格式

```
DROP REDACTION POLICY [ IF EXISTS ] policy_name ON table_name;
```

参数说明

表 21-244 DROP REDACTION POLICY 参数说明

参数	描述	取值范围
IF EXISTS	如果待删除的脱敏策略不存在，则发出一个notice而不是抛出一个错误。	-
policy_name	要删除的脱敏策略名称。	已存在的脱敏策略名称。
table_name	要删除的脱敏策略所应用的表名。	已存在的脱敏策略所应用的表名。

版权所有 © 华为云计算技术有限公司

示例

删除表emp上的脱敏策略mask_emp。

```
DROP REDACTION POLICY mask_emp ON emp;
```

相关链接

ALTER REDACTION POLICY，**CREATE REDACTION POLICY**

21.13.9 资源池和过滤规则

21.13.9.1 CREATE RESOURCE POOL

功能描述

创建一个资源池，并指定此资源池相关联的控制组。数据库的系统资源（CPU资源、内存资源、IO资源和存储资源）是有限的，DWS在同时运行多种类型的业务（如数据加载、批量分析、实时查询等）时，各类型业务之间可能会竞争资源，从而出现资源性能瓶颈，导致吞吐量下降，造成整体性能低下。那么对系统的资源进行合理的分配，能避免产生资源不合理利用而导致的系统运行效率下降等问题，因此DWS提供了**资源管理功能**，用户可以根据自身业务将资源按需划分成不同的资源池，不同资源池之间资源互相隔离。

本语法为基础语法仅供参考，当前**推荐通过管理控制台进行资源池创建**，了解资源池更多内容，请参见《用户指南》的“DWS资源负载管理”章节。

注意事项

只要用户对当前数据库有CREATE权限，就可以创建资源池。

语法格式

```
CREATE RESOURCE POOL pool_name
    [WITH ({MEM_PERCENT=pct | CONTROL_GROUP="group_name" | ACTIVE_STATEMENTS=stmt |
MAX_DOP = dop | MEMORY_LIMIT='memory_size' | io_limits=io_limits | io_priority='priority' |
nodegroup='nodegroup_name' | is_foreign = boolean | except_rule='except_rule' | weight=bandwidth_weight
| enable_concurrency_scaling=boolean}[, ... ])];
```

参数说明

表 21-245 CREATE RESOURCE POOL 参数说明

参数	描述	取值范围或示例
pool_name	资源池名称。 资源池名称不能和当前数据库里已有的资源池重名。	字符串，需符合**标识符命名规范**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例

CONTROL_GROUP	控制组名称。
	控制组（Cgroups）是control groups的缩写，是Linux内核提供的一种可以限制、记录、隔离进程组所使用的物理资源（如：CPU、内存、IO等）的机制。如果一个进程加入了某一个控制组，该控制组对Linux的系统资源都有严格的限制，进程在使用这些资源时，不能超过其最大限制。更多Cgroups的原理介绍，请查看相关操作系统的产品手册。
	环境中已预置存在**DefaultClass控制组**，而DefaultClass控制组下层Timeshare控制组又分为以下4个层级：“Rush”、“High”、“Medium”或“Low”。
	● 设置控制组名称时，语法可以使用双引号，也可以使用单引号。
	● group_name对大小写敏感。
	● 不指定group_name时，默认指定的字符串为“Medium”，代表指定DefaultClass控制组的“Medium” Timeshare控制组。
		创建一个默认资源池，其控制组为“DefaultClass”组下属的“Medium” Timeshare Workload控制组。
		`CREATE RESOURCE POOL pool1;`
		创建一个资源池，其控制组指定为“DefaultClass”组下属的“High” Timeshare Workload控制组。
		`CREATE RESOURCE POOL pool2 WITH (CONTROL_GROUP="High");`

ACTIVE_STATEMENTS	资源池**复杂语句**执行的最大并发数量。数值型，-1~INT_MAX，不指定时，默认是10。	创建资源池pool3，设定复杂语句执行的最大并发数量为5，内存使用比例50%，网络权重为5。 `CREATE RESOURCE POOL pool3 WITH (ACTIVE_STATEMENTS=5, MEM_PERCENT=50, weight=5);`
MAX_DOP	资源池**简单语句**执行的最大并发数量。数值型，-1~INT_MAX，-1和0代表不限制，不指定时，默认为-1，不限制。	创建资源池pool4，设定简单语句执行最大并发数量为10。 `CREATE RESOURCE POOL pool4 WITH (MAX_DOP=10);`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
MEMORY_LIMIT	使用该资源池的作业估算内存上限。 取值为字符串。 ● default，表示最大为资源池内存一半。 ● unlimited，表示不限制。 ● 数值范围为1KB~2047GB。 不指定时，默认为default。	-
MEM_PERCENT	资源池可用内存占全部内存或者组用户内存使用的比例。 ● 取值范围为0-100，默认值为0。	-
io_limits	该参数8.1.2版本中已废弃，为兼容历史版本保留该参数。	-
io_priority	该参数8.1.2版本中已废弃，为兼容历史版本保留该参数。	-
nodegroup	在逻辑集群模式下，指定资源池所属的逻辑集群名称。必须是存在的逻辑集群。	如果逻辑集群名称包含大写字符、特殊符号或以数字开头，SQL语句中对逻辑集群名称需要加双引号。
is_foreign	在逻辑集群模式下，指定当前资源池用于控制没有关联本逻辑集群的普通用户的资源。这里的逻辑集群是由资源池nodegroup字段指定的。	● nodegroup必须是存在的逻辑集群，不能是elastic_group和安装的nodegroup (group_version1)。 ● 如果指定了is_foreign为true，则资源池不能再关联用户，即不允许通过CREATE USER ... RESOURCE POOL语句来将该资源池配置给用户。该资源池自动检查用户是否关联到资源池指定的逻辑集群，如果用户没有关联到该逻辑集群，则这些用户在逻辑集群所包含的DN上运行将受到该资源池的资源控制。

 版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
except_rule	异常规则集。 **异常规则集**是由一组定义了作业执行时间、作业排队时间、CPU使用率、最大落盘等各项规则阈值的集合，与资源池绑定搭配后使用，当用户执行作业时，触发了异常规则，则作业会选择终止或者降级。 了解更多，请参见**CREATE EXCEPT RULE**。	**示例：创建异常规则，当作业执行时间超过10秒，则终止该作业**
weight	资源池网络带宽权重。取值范围为-1~2147483647的整数，默认配置为-1，不设置权重。	-
enable_concurrency_scaling	弹性并发扩展功能开关。该功能仅9.1.0.100及以上集群版本支持。	• true表示开启该资源池作业的弹性并发扩展功能。 • false表示关闭该资源池作业的弹性并发扩展功能。 默认值：false

示例

本示例假定用户已预先成功创建控制组。

创建一个默认资源池，其控制组为“DefaultClass”组下属的“Medium”Timeshare Workload控制组。

```
CREATE RESOURCE POOL pool1;
```

创建一个资源池，其控制组指定为“DefaultClass”组下属的“High”Timeshare Workload控制组。

```
CREATE RESOURCE POOL pool2 WITH (CONTROL_GROUP="High");
```

创建资源池pool_p，设定复杂语句执行的最大并发数量为5，内存使用比例50%，网络权重为5。

```
CREATE RESOURCE POOL pool_p WITH (ACTIVE_STATEMENTS=5, MEM_PERCENT=50, weight=5);
```

可通过PG_RESOURCE_POOL系统表查询资源池的配置参数。

```
SELECT * FROM PG_RESOURCE_POOL WHERE respool_name = 'pool_p';
```

```
gaussdb=> SELECT * FROM PG_RESOURCE_POOL WHERE respool_name = 'pool_p';                                M    at isa0060/serio1/
 respool_name | mem_percent | cpu_affinity |    control_group    | active_statements | max_dop | memory_limit | parentid | io_li
mits | io_priority |  nodegroup   | is_foreign | short_acc | except_rule | weight | enable_concurrency_scaling
--------------+-------------+--------------+---------------------+-------------------+---------+--------------+----------+------
-----+-------------+-------------+------------+-----------+-------------+--------+---------------------------
 pool_p       |          50 |           -1 | DefaultClass:Medium |                 5 |      -1 | default      |        0 |
   0 | None        | installation | f          | t         | None        |      5 | f
(1 row)
```


版权所有 © 华为云计算技术有限公司

创建一个异常规则集except_rule4，设置执行的最大时间elapsedtime为10秒，并创建一个资源池同时绑定异常规则。

```
CREATE EXCEPT RULE except_rule4 WITH (elapsedtime=10);
CREATE resource pool resource_pool_a4 WITH (except_rule='except_rule4');
```

创建一个资源池，并打开该资源池的弹性并发扩展功能。

```
CREATE RESOURCE POOL pool6 WITH (enable_concurrency_scaling=true);
```

示例：创建异常规则，当作业执行时间超过 10 秒，则终止该作业

在DWS资源负载管理中，异常规则需要绑定到资源池后生效，紧接着将执行作业的用户绑定到指定资源池后，总体上才能达到资源负载管理的效果，当用户执行某作业时，若触发异常规则，则作业将被终止，以下示例模拟了这一终止过程。

步骤1 先创建一个测试表并导入数据。

1. 以系统管理员连接数据库执行以下SQL，创建一个名为src的临时表，为后续的INSERT操作提供数据源。

   ```
   CREATE SCHEMA test;
   CREATE TABLE test.src AS SELECT 1;
   ```

2. 执行以下SQL，创建表test.test。

   ```
   DROP TABLE IF EXISTS test.test;
   CREATE TABLE test.test(a int, b numeric(15,2)) WITH(orientation=column);
   ```

3. 执行以下SQL，分别为两张表插入2000万行数据。执行时间约半分钟，请等待。

   ```
   INSERT INTO test.test SELECT generate_series(1,20000000) % 1000,generate_series(1,20000000)
   FROM test.src;
   ```

步骤2 创建一个异常规则集except_rule4，设置执行的最大时间elapsedtime为10秒

```
CREATE EXCEPT RULE except_rule4 WITH (elapsedtime=10);
```

步骤3 创建一个资源池resource_pool_a4，并将异常规则绑定到该资源池。

```
CREATE resource pool resource_pool_a4 WITH (except_rule='except_rule4');
```

步骤4 创建一个作业用户，并绑定到资源池resource_pool_a4中，密码请用户自定义。

```
CREATE USER user4 RESOURCE POOL 'resource_pool_a4' PASSWORD 'password';
```

步骤5 授权作业用户user4可以访问表test.test。

```
GRANT USAGE ON SCHEMA test TO user4;
GRANT SELECT ON TABLE test.test TO user4;
```

步骤6 以作业用户user4连接数据库，执行以下SQL，超过10秒，语句被终止了。

```
SELECT * FROM test.test;
```

```
gaussdb=> SELECT * FROM test.test;
ERROR:  canceling statement due to workload manager exception.
DETAIL: except rule [except_rule4] is meet condition:rule [elapsedtime] is over limit, current value is:11.
gaussdb=>
```

步骤7 通过**PG_EXCEPT_RULE**系统表，可以查看异常规则定义。

```
SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule4';
```

```
gaussdb=> SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule4';
     name     |    rule     | value
--------------+-------------+-------
 except_rule4 | action      | abort
 except_rule4 | elapsedtime | 10
(2 rows)
```

----结束

版权所有 © 华为云计算技术有限公司

相关链接

ALTER RESOURCE POOL，DROP RESOURCE POOL，CREATE EXCEPT RULE

21.13.9.2 ALTER RESOURCE POOL

功能描述

修改一个资源池，指定其他控制组。

注意事项

只要用户对当前数据库有ALTER权限，就可以修改资源池。

语法格式

```
ALTER RESOURCE POOL pool_name
    WITH ({MEM_PERCENT=pct | CONTROL_GROUP='group_name' | ACTIVE_STATEMENTS=stmt | MAX_DOP
= dop | MEMORY_LIMIT='memory_size' | io_limits=io_limits | io_priority='priority' |
nodegroup='nodegroup_name' | except_rule='except_rule' | weight=bandwidth_weight |
enable_concurrency_scaling=boolean}[, ... ]);
```

参数说明

表 21-246 ALTER RESOURCE POOL 参数说明

参数	描述	取值范围
pool_name	需要修改的资源池名称。	已存在的资源池。
group_name	控制组名称。	已存在的控制组。
stmt	资源池语句执行的最大并发数量。	数值型，-1~INT_MAX。
dop	资源池简单语句执行的最大并发数量。	数值型，1~INT_MAX。
memory_size	使用该资源池的作业估算内存上限。	取值范围：字符串 ● default，表示最大为资源池内存一半。 ● unlimited，表示不限制。 ● 数值范围为1KB~2047GB。
mem_percent	资源池可用内存占全部内存或者组用户内存使用的比例。	普通用户的mem_percent范围为0-100的整数，默认值为0。
io_limits	该参数8.1.2集群版本中已废弃，为兼容历史版本保留该参数。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
io_priority	该参数8.1.2集群版本中已废弃，为兼容历史版本保留该参数。	-
except_rule	异常规则集。	-
weight	资源池网络带宽权重。	-
enable_concurrency_scaling	弹性并发扩展功能开关。该功能仅9.1.0.100及以上集群版本支持。默认值为false。	true或false。 • true，表示开启该资源池作业的弹性并发扩展功能。 • false，表示关闭该资源池作业的弹性并发扩展功能。

说明

- 设置控制组名称“**group_name**”时，语法可以使用双引号，也可以使用单引号。
- group_name对大小写敏感。
- 不指定group_name时，默认指定的字符串为 "Medium"，代表指定DefaultClass控制组的"Medium" Timeshare控制组。
- 若数据库用户指定Timeshare控制组代表的字符串，即"Rush"、"High"、"Medium"或"Low"其中一种，如control_group的字符串为"High"；代表资源池指定到DefaultClass控制组下的"High" Timeshare控制组。

示例

创建示例资源池pool_test，其控制组为“DefaultClass”组下属的“Medium” Timeshare Workload控制组。

```
DROP RESOURCE POOL IF EXISTS pool_test;
CREATE RESOURCE POOL pool_test;
```

修改资源池pool_test，其控制组指定为“DefaultClass”组下属的“High”Timeshare Workload控制组。

```
ALTER RESOURCE POOL pool_test WITH (CONTROL_GROUP="High");
```

关闭指定资源池作业的弹性并发扩展功能。

```
ALTER RESOURCE POOL pool_test WITH (enable_concurrency_scaling=false);
```

相关链接

CREATE RESOURCE POOL，**DROP RESOURCE POOL**

21.13.9.3 DROP RESOURCE POOL

功能描述

删除一个资源池。

版权所有 © 华为云计算技术有限公司

说明

如果某个角色已关联到该资源池，则无法删除该资源池。

注意事项

只要用户对当前数据库有DROP权限，就可以删除资源池。

语法格式

```
DROP RESOURCE POOL [ IF EXISTS ] pool_name;
```

参数说明

表 21-247 DROP RESOURCE POOL 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的资源池不存在，发出一个notice而不是抛出一个错误。	-
pool_name	已创建过的资源池名称。	字符串，要符合标识符的命名规范。

示例

删除资源池pool。

```
DROP RESOURCE POOL pool;
```

相关链接

ALTER RESOURCE POOL，**CREATE RESOURCE POOL**

21.13.9.4 CREATE EXCEPT RULE

功能描述

用于创建一个异常规则集合。创建时可以指定任意一个或者多个具体的规则阈值，也可以指定触发异常规则后所采取的操作，当前支持终止（abort）和降级（penalty）操作。

分布式数据库中，查询任务由多个节点并行执行，每个节点独立处理部分数据并汇总结果。但若任一节点出现性能瓶颈，即使其他节点快速完成，整个查询仍会被拖慢。在某些极端情况下，一个查询语句可能会占用大量资源，导致节点资源使用率过高，从而引发计算资源或存储资源不足的问题，不仅影响业务运行，还可能导致整个集群性能下降甚至不可用。

DWS提供配置**异常规则**功能，允许用户设置查询的资源限制（如执行时间、内存、磁盘等），SQL语句在执行时触发异常规则会自动终止，避免异常查询拖垮集群，保障稳定性并减轻DBA运维负担。例如设置查询的最大执行时间，超时则自动终止；设置查询可以使用的磁盘空间大小，超过阈值则自动终止；或者设置查询可以使用的内存量，超过限制则终止。

版权所有 © 华为云计算技术有限公司

在DWS资源负载管理中，异常规则需要绑定到资源池后生效，紧接着将执行作业的用户绑定到指定资源池后，总体上才能达到资源负载管理的效果，当用户执行某作业时，若触发异常规则，则作业将被终止。

您也可以**通过管理控制台**进行资源池和异常规则设置，了解**资源池、异常规则**介绍及配置，可参见《用户指南》的“DWS资源负载管理”。

注意事项

同一组异常规则中的多个规则以“且”的关系生效，例如：同时设置执行时间elapsedtime=1000，内存大小memsize=500。表示一个作业，执行时间达到1000秒且内存使用达到500MB时，才会终止该作业。**仅触发其中一个不会生效。**

语法格式

```
CREATE EXCEPT RULE except_rule_name
    WITH (
        | BLOCKTIME = VALUE,
        | CPUTIME = VALUE,
        | ELAPSEDTIME = VALUE,
        | CPUSKEWPERCENT = VALUE,
        | SPILLSIZE = VALUE,
        | BROADCASTSIZE = VALUE,
        | MEMSIZE = VALUE,
        | CPUAVGPERCENT = VALUE,
        | BANDWIDTH = VALUE,
        | ACTION = ['abort' | 'penalty']
        );
```

参数说明

表 21-248 CREATE EXCEPT RULE 参数说明

参数	描述	取值范围
except_rule_name	异常规则集名称。	字符串，需符合**标识符命名规范**。
blocktime	作业排队阻塞的最大时间，单位：秒。	数值型，-1，1~INT64_MAX。-1表示不设置，即不限制，下同。
elapsedtime	作业执行的最大时间，单位：秒。	数值型，-1，1~INT64_MAX。
allcputime	作业运行中使用的最大CPU时间，单位：秒。	数值型，-1，1~INT64_MAX。
cpuskewpercent	作业执行时的平均CPU使用率，单位：百分比。	数值型，-1，1~100。
cpuavgpercent	作业执行时的CPU使用倾斜率，单位：百分比。	数值型，-1，1~100。
spillsize	作业执行的最大下盘大小，单位：MB。	数值型，-1，1~INT64_MAX。
broadcastsize	作业执行的最大广播大小，单位：MB。	数值型，-1，1~INT64_MAX。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
memsize	作业执行使用的最大内存大小，单位：MB。	数值型，-1，1~INT64_MAX。
bandwidth	作业执行可使用的最大带宽，单位：MB。	数值型，-1，1~INT64_MAX。

基础示例

创建一个异常规则集except_rule1并指定其排队阻塞时间blocktime为3000秒，下盘空间为4000MB，即某个语句排队阻塞超过3000秒且下盘空间超过4000MB，该语句会被终止。

```
CREATE EXCEPT RULE except_rule1 WITH (blocktime=3000, spillsize=4000, action=abort);
```

创建一个异常规则集except_rule2并指定其memsize规则阈值为5000MB，默认所采取的异常规则操作为abort。

```
CREATE EXCEPT RULE except_rule2 WITH (memsize=3000);
```

创建一个资源池并绑定异常规则集except_rule3。

```
CREATE resource pool resource_pool_a1 WITH (except_rule='except_rule3');
```

示例：创建异常规则，当作业执行时间超过 10 秒，则终止该作业

在DWS资源负载管理中，异常规则需要绑定到资源池后生效，紧接着将执行作业的用户绑定到指定资源池后，总体上才能达到资源负载管理的效果，当用户执行某作业时，若触发异常规则，则作业将被终止，以下示例模拟了这一终止过程。

步骤1 先创建一个测试表并导入数据。

1. 以系统管理员连接数据库执行以下SQL，创建一个名为src的临时表，为后续的INSERT操作提供数据源。
   ```
   CREATE SCHEMA test;
   CREATE TABLE test.src AS SELECT 1;
   ```
2. 执行以下SQL，创建表test.test。
   ```
   DROP TABLE IF EXISTS test.test;
   CREATE TABLE test.test(a int, b numeric(15,2)) WITH(orientation=column);
   ```
3. 执行以下SQL，分别为两张表插入2000万行数据。执行时间约半分钟，请等待。
   ```
   INSERT INTO test.test SELECT generate_series(1,20000000) % 1000,generate_series(1,20000000)
   FROM test.src;
   ```

步骤2 创建一个异常规则集except_rule4，设置执行的最大时间elapsedtime为10秒

```
CREATE EXCEPT RULE except_rule4 WITH (elapsedtime=10);
```

步骤3 创建一个资源池resource_pool_a4，并将异常规则绑定到该资源池。

```
CREATE resource pool resource_pool_a4 WITH (except_rule='except_rule4');
```

步骤4 创建一个作业用户，并绑定到资源池resource_pool_a4中，密码请用户自定义。

```
CREATE USER user4 RESOURCE POOL 'resource_pool_a4' PASSWORD 'password';
```

步骤5 授权作业用户user4可以访问表test.test。

```
GRANT USAGE ON SCHEMA test TO user4;
GRANT SELECT ON TABLE test.test TO user4;
```

 版权所有 © 华为云计算技术有限公司

步骤6 以作业用户user4连接数据库，执行以下SQL，超过10秒，语句被终止了。

```
SELECT * FROM test.test;
```

```
gaussdb=> SELECT * FROM test.test;
ERROR:  canceling statement due to workload manager exception.
DETAIL: except rule [except_rule4] is meet condition:rule [elapsedtime] is over limit, current value is:11.
gaussdb=>
```

步骤7 通过**PG_EXCEPT_RULE**系统表，可以查看异常规则定义。

```
SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule4';
```

```
gaussdb=> SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule4';
     name     |    rule     | value
--------------+-------------+-------
 except_rule4 | action      | abort
 except_rule4 | elapsedtime | 10
(2 rows)
```

----结束

相关链接

ALTER EXCEPT RULE，**DROP EXCEPT RULE**，**CREATE RESOURCE POOL**

21.13.9.5 ALTER EXCEPT RULE

功能描述

修改一个异常规则集合，可以修改一组规则集中的任意一个或者多个具体的规则阈值。

注意事项

使用ALTER语法添加某个新规则阈值时，WITH参数中原来的其他的规则会保留，并不是全部清空覆盖。例如原来一组规则中只有一条规则：作业排队阻塞的最大时间blocktime = 100秒，通过以下语法新添加一条作业执行最大时间elapsedtime为1000秒，则最终的规则包含2条，可参见以下示例。

语法格式

```
ALTER EXCEPT RULE except_rule_name
    WITH (
        | BLOCKTIME = VALUE,
        | CPUTIME = VALUE,
        | ELAPSEDTIME = VALUE,
        | CPUSKEWPERCENT = VALUE,
        | SPILLSIZE = VALUE,
        | BROADCASTSIZE = VALUE,
        | MEMSIZE = VALUE,
        | CPUAVGPERCENT = VALUE,
        | BANDWIDTH = VALUE,
        | ACTION = ['abort' | 'penalty']
        );
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-249 ALTER EXCEPT RULE 参数说明

参数	描述	取值范围
rule_name	异常规则集名称。	字符串，需符合**标识符命名规范**。
blocktime	作业排队阻塞的最大时间。单位：秒。	数值型，-1，1~INT64_MAX。
elapsedtime	作业执行的最大时间。单位：秒。	数值型，-1，1~INT64_MAX。
allcputime	作业运行中使用的最大CPU时间。单位：秒。	数值型，-1，1~INT64_MAX。
cpuskewpercent	作业执行时的CPU使用倾斜率。单位：百分比。	数值型，-1，1~100。
cpuavgpercent	作业执行时的平均CPU使用率。单位：百分比。	数值型，-1，1~100。
spillsize	作业执行的最大下盘大小。单位：MB。	数值型，-1，1~INT64_MAX。
broadcastsize	作业执行的最大广播大小。单位：MB。	数值型，-1，1~INT64_MAX。
memsize	作业执行使用的最大内存大小。单位：MB。	数值型，-1，1~INT64_MAX。
bandwidth	作业执行可使用的最大网络带宽。单位：MB。	数值型，-1，1~INT64_MAX。

示例

创建异常规则except_rule1。

```
CREATE EXCEPT RULE except_rule_1 WITH (blocktime=2000, spillsize=3000, action=abort);
```

修改异常规则except_rule1的blocktime规则阈值为3000秒，下盘空间为4000MB。

```
ALTER EXCEPT RULE except_rule_1 WITH (blocktime=3000, spillsize=4000);
```

通过**PG_EXCEPT_RULE**系统表，可以查看异常规则已修改成功。

```
SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule_1';
```

```
gaussdb=> SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule_1';
     name      |   rule    | value
---------------+-----------+-------
 except_rule_1 | action    | abort
 except_rule_1 | blocktime | 3000
 except_rule_1 | spillsize | 4000
(3 rows)
```


版权所有 © 华为云计算技术有限公司

同时，需要再叠加一条新规则，作业执行的最大时间elapsedtime为2000秒。

```
ALTER EXCEPT RULE except_rule_1 WITH (elapsedtime=2000);
```

再次查看异常规则，已修改成功。**注意，原先的规则会保留，并不是清空。**

```
SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule_1';
```

```
gaussdb=> SELECT * FROM PG_EXCEPT_RULE WHERE name = 'except_rule_1';
     name      |    rule     | value
---------------+-------------+-------
 except_rule_1 | action      | abort
 except_rule_1 | blocktime   | 3000
 except_rule_1 | spillsize   | 4000
 except_rule_1 | elapsedtime | 2000
(4 rows)
```

创建一个资源池并绑定异常规则集except_rule_1。

```
CREATE resource pool resource_pool_a1 WITH (except_rule='except_rule_1');
```

解除资源池resource_pool_a1绑定的异常规则集。

```
ALTER resource pool resource_pool_a1 WITH (except_rule='None');
```

相关链接

CREATE EXCEPT RULE，DROP EXCEPT RULE

21.13.9.6 DROP EXCEPT RULE

功能描述

用于删除指定异常规则集合。

注意事项

只有系统管理员才能执行DROP EXCEPT RULE操作。

语法格式

```
DROP EXCEPT RULE [ IF EXISTS ]
  rule_name ;
```

参数说明

表 21-250 DROP EXCEPT RULE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的异常规则不存在时返回一个提示而不是错误。	-
rule_name	需要删除的异常规则集名称。	已存在的异常规则集名称。

版权所有 © 华为云计算技术有限公司

示例

创建异常规则集合except_rule_1。

```
CREATE EXCEPT RULE except_rule_1 WITH (blocktime=2000, spillsize=3000, action=abort);
```

删除名称为except_rule_1的规则集合。

```
DROP EXCEPT RULE except_rule_1;
```

如果存在名为except_rule_2的规则集合，则将其删除。

```
DROP EXCEPT RULE IF EXISTS except_rule_2;
```

相关链接

CREATE EXCEPT RULE，**ALTER EXCEPT RULE**

21.13.9.7 CREATE BLOCK RULE

功能描述

该语法用于创建**一条查询过滤规则**，包括查询过滤规则名称、绑定的客户端名称、客户端IP、用户以及匹配方式等。在日常数据库操作中，异常SQL的执行可能会导致资源过度消耗，影响数据库性能和稳定性，通过创建查询过滤规则，可以将此类SQL语句进行拦截。

当前支持**指定某个SQL语句的正则匹配**进行拦截，例如针对某用户对某张表的SELECT进行拦截；也支持**指定某SQL语句的unique_sql_id或sql_hash取值**进行拦截，例如已知某SQL语句的**unique_sql_id**值，将此SQL语句直接拦截。

本文档提供基础SQL语法，您也可以通过控制台进行配置，参见《 用户指南 》的“使用DWS查询过滤器拦截慢SQL”章节 。

注意事项

- 该语法仅9.1.0.100及以上集群版本支持。
- 只有拥有数据库所有者权限的用户或者授予gs_role_block角色权限的用户才能执行CREATE BLOCK RULE命令，系统管理员默认拥有此权限。
- 普通用户无法执行CREATE BLOCK RULE，可通过系统管理员进行授权，授权语法：
  ```
  GRANT gs_role_block TO user;
  ```
  建议**创建查询过滤规则时尽量缩小适用范围，避免误过滤，或者范围过大导致性能劣化**。

语法格式

```
CREATE BLOCK RULE [ IF NOT EXISTS ] block_name
   [ [ TO user_name@'host' ] | [ TO user_name ] | [ TO 'host' ] ] |
   [ FOR UPDATE | SELECT | INSERT | DELETE | MERGE ] |
   FILTER BY
   { SQL ( 'text' ) | TEMPLATE ( template_parameter = value ) }
   [ WITH ( { with_parameter = value }, [, ... ] ) ];
```


版权所有 © 华为云计算技术有限公司

参数说明

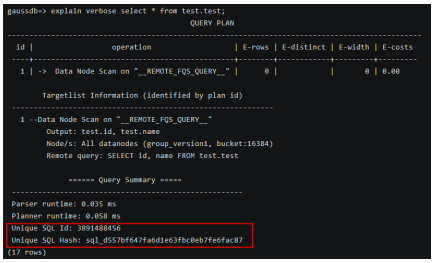
表 21-251 CREATE BLOCK RULE 参数说明

参数	描述	取值范围或示例
block_name	要创建的查询过滤规则名称。	字符串，需符合**标识符命名规范**。

TO	对查询过滤规则生效的对象或客户端IP，支持以下方式：
	• **TO user_name@'host'**：用户名以及客户端IP，两者都要满足，查询过滤规则才能生效。
	• **TO user_name**：用户名，即只判定满足用户名条件，查询过滤规则就生效，不区分客户端IP。
	• **TO 'host'**：客户端IP，即只判定满足客户端IP条件，查询过滤规则就生效，不区分用户。
		举例：查询过滤规则对来自客户端IP为192.168.0.59且登录用户名为user1时生效。
		`CREATE BLOCK RULE query_block TO user1@'192.168.0.59' FOR SELECT  FILTER BY SQL('test.test');`
		查询过滤规则对登录用户名为user1时生效（无论客户端IP是哪个）。
		`CREATE BLOCK RULE query_block TO user1 FOR SELECT  FILTER BY SQL('test.test');`
		查询过滤规则对来自客户端IP 192.168.0.59的访问时都生效（无论哪个用户登录）。
		`CREATE BLOCK RULE query_block TO '192.168.0.59' FOR SELECT  FILTER BY SQL('test.test');`

FOR	语句类型，支持对UPDATE/SELECT/INSERT/DELETE/MERGE INTO五种类型语句进行限制。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
FILTER BY	过滤方法，支持两种形式： • **SQL**：根据SQL语句内容进行过滤，通过关键词对语句进行正则匹配，例如表名，其长度不能超过1024，建议尽量精简。 • **TEMPLATE**：根据模板进行过滤，**此方法不区分用户，针对所有用户执行的特定语句都实施拦截，拦截效果更高效**，取值如下： – **unique_sql_id**：归一化的64位哈希值，重复概率较sql_hash大一些。 – **sql_hash**：归一化的哈希值（MD5），一般不会重复，相较unique_sql_id更推荐使用。	unique_sql_id和sql_hash取值，可以通过EXPLAIN VERBOSE查询，例如： EXPLAIN VERBOSE SELECT * FROM test.test; 
with_parameter	查询过滤规则选项参数。	以下选项可以一起设置，满足任何一个参数的限制查询将会被过滤。 • application_name，客户端名称。 • query_band，负载标识。 • table_num，语句扫描的基表个数。 • partition_num，算子预估扫描的最大分区数。 • estimate_row，算子预估扫描的最大表行数。 • resource_pool，需要切换到的资源池名称。 • max_active_num，规则对应语句的最大并发数。 • is_warning，控制拦截到语句的行为是告警还是报错，on表示告警，off表示报错，默认为off。

示例：基于匹配 SQL 语句进行 SQL 拦截

以系统管理员连接数据库执行以下SQL，创建一个名为src的临时表，为后续的INSERT操作提供数据源。

 版权所有 © 华为云计算技术有限公司

```
CREATE SCHEMA test;
CREATE TABLE test.src AS SELECT 1;
```

执行以下SQL，创建表test.test。

```
DROP TABLE IF EXISTS test.test;
CREATE TABLE test.test(a int, b numeric(15,2)) WITH(orientation=column);
```

执行以下SQL，分别为两张表插入2000万行数据。执行时间约半分钟，请等待。

```
INSERT INTO test.test SELECT generate_series(1,20000000) % 1000,generate_series(1,20000000) FROM test.src;
```

创建用户user1，并授权该用户可以访问表test.test。

```
CREATE USER user1 PASSWORD "password";
GRANT USAGE ON SCHEMA test TO user1;
GRANT SELECT ON TABLE test.test TO user1;
```

以用户user1连接数据库，查询表数据，结果表明在没有创建过滤规则的情况下，用户user1可以正常查询表test.test数据。

```
SELECT sum(b) FROM test.test GROUP BY a;
```

```
gaussdb=> SELECT sum(b) FROM test.test GROUP BY a;
        sum
------------------
 199990060000.00
 199990120000.00
 199990180000.00
 199990240000.00
 199990300000.00
 199990360000.00
 199990420000.00
 199990480000.00
 199990540000.00
 199990600000.00
 199990660000.00
 199990720000.00
 199990780000.00
```

以系统管理员连接数据库，创建名称为query_block的查询过滤规则，对于用户user1查询的SQL语句进行拦截，过滤方式为SQL语句，SQL语句中出现关键字test(即表名)会被过滤。

```
CREATE BLOCK RULE query_block
TO user1@'192.168.0.59'        --规则作用于用户user1，且客户端IP为192.168.0.59。
FOR SELECT                     --过滤SELECT的相关操作。
FILTER BY
SQL('test.test')               --过滤方式为SQL语句，SQL语句中出现关键字test(即表名)会被过滤。
WITH                           --满足以下条件即被过滤。
(
application_name='gsql',       --客户端名称为gsql
table_num='1',                 --扫描基表次数为1
estimate_row='10',             --扫描基表行数为10
is_warning='off'               --关闭告警，直接报错
)
;
```

通过视图**pg_get_blockruledef**查询已创建的过滤规则，从回显判断出规则已生效。

```
SELECT * FROM pg_get_blockruledef('query_block');
```


版权所有 © 华为云计算技术有限公司

```
gaussdb=> SELECT * FROM pg_get_blockruledef('query_block');
                                                
```

```
DROP BLOCK RULE query_block;
CREATE BLOCK RULE query_block FILTER BY
TEMPLATE(sql_hash='sql_170d0c75c472043b0cd35e05903c382f');
```

再次切到user1用户，查询报错，语句被拦截了。

```
gaussdb=> SELECT sum(b) FROM test.test GROUP BY a;
ERROR:  hit block rule query_block(sql_hash: sql_170d0c75c472043b0cd35e05903c382f)
```

相关链接

ALTER BLOCK RULE，**DROP BLOCK RULE**

本文档提供基础SQL语法，您也可以通过控制台进行配置，参见《 用户指南 》的“使用DWS查询过滤器拦截慢SQL”章节 。

常见问题：创建了查询过滤规则后，并未成功拦截语句

不同的计划，可能部分字段无法按照预期进行拦截，例如：

```
CREATE BLOCK RULE test filter BY SQL('test')with(estimate_row='3');
SELECT * FROM test;
```

```
 c1 | c2
----+----
  1 |  2
  1 |  2
  1 |  2
  1 |  2
  1 |  2
(5 rows)
```

此时，语句关键字是匹配的，查询的行数也超过了3行的限制，但是没有成功拦截。

```
postgres=# explain verbose select * from test;
                                  QUERY PLAN
------------------------------------------------------------------------------------------
 id |                operation                 | E-rows | E-distinct | E-width | E-costs
----+------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

    Targetlist Information (identified by plan id)
--------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: test.c1, test.c2
        Node/s: All datanodes (node_group, bucket:16384)
        Remote query: SELECT c1, c2 FROM public.test
```

通过计划可以看出，此时是**FQS（fast_query_shipping）**计划，即参数enable_fast_query_shipping设置为on，导致没有估算信息。因此无法进行拦截，对于CN轻量化的计划也是一样的，如果让语句强制使用stream计划，那么就可以拦截成功：

版权所有 © 华为云计算技术有限公司

```
SET enable_stream_operator=on;
SET enable_fast_query_shipping=off;
```

```
postgres=# set enable_stream_operator=on;
SET
postgres=# set enable_fast_query_shipping=off;
SET
postgres=# select * from test;
ERROR:  hit block rule test(regexp_sql: test, estimate_row: 3(5))
postgres=#  explain verbose select * from test;
                                    QUERY PLAN
-------------------------------------------------------------------------------------------
 id |              operation               | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------+--------+------------+---------+---------
  1 | ->  Row Adapter                      |      5 |            |       8 | 69.00
  2 |    ->  Vector Streaming (type: GATHER) |      5 |            |       8 | 69.00
  3 |       ->  CStore Scan on public.test |      5 |            |       8 | 59.01

     Targetlist Information (identified by plan id)
-----------------------------------------------------------
  1 --Row Adapter
        Output: c1, c2
  2 --Vector Streaming (type: GATHER)
        Output: c1, c2
        Node/s: All datanodes (node_group, bucket:16384)
  3 --CStore Scan on public.test
        Output: c1, c2
        Distribute Key: c1
```

综上所述，如果估算信息不准确，也会导致误拦截或者漏拦截的情况，因为计划的信息是通过估算所得，通常这种情况无法避免。

常见问题：如何查看拦截记录

创建查询过滤规则后会拦截很多慢SQL，可以通过TopSQL系统表GS_WLM_SESSION_INFO查看已拦截的语句，abort_info会记录拦截信息，也就是查询的报错信息。

需连接postgres数据库执行：

```
SELECT abort_info,query FROM GS_WLM_SESSION_INFO WHERE abort_info like '%hit block rule%';    --查询所有拦截记录
SELECT abort_info,query FROM GS_WLM_SESSION_INFO WHERE abort_info like '%hit block rule query_block_new%'   --只查询规则名称为query_block_new的拦截记录
```


版权所有 © 华为云计算技术有限公司

21.13.9.8 ALTER BLOCK RULE

功能描述

修改查询过滤规则属性，包括查询过滤规则名称、绑定的客户端名称、客户端IP、用户以及匹配方式等。

在日常数据库操作中，异常SQL的执行可能会导致资源过度消耗，影响数据库性能和稳定性，通过创建查询过滤规则，可以将此类SQL语句进行拦截。

注意事项

- 该语法仅9.1.0.100及以上集群版本支持。
- 只有拥有数据库所有者权限的用户或者授予gs_role_block角色权限的用户才能执行ALTER BLOCK RULE命令，系统管理员默认拥有此权限。
- 普通用户无法执行CREATE BLOCK RULE，可通过系统管理员进行授权，授权语法：
  ```
  GRANT gs_role_block TO user;
  ```
  建议**创建查询过滤规则时尽量缩小适用范围，避免误过滤，或者范围过大导致性能劣化**。
- DEFAULT语法用于对某一属性进行重置，但仅适用于非必选项。

语法格式

```
ALTER BLOCK RULE block_name
   [ [ TO user_name@'host' ] | [ TO user_name ] | [ TO 'host' ] | [ TO DEFAULT ] ] |
   [ FOR UPDATE | SELECT | INSERT | DELETE | MERGE | DEFAULT ] |
   FILTER BY
   { SQL ( 'text' ) | TEMPLATE ( template_parameter = value ) }
   [ WITH ( { with_parameter = value }, [, ... ] ) ];
ALTER BLOCK RULE block_name RENAME to new_block_name;
```


版权所有 © 华为云计算技术有限公司

参数说明

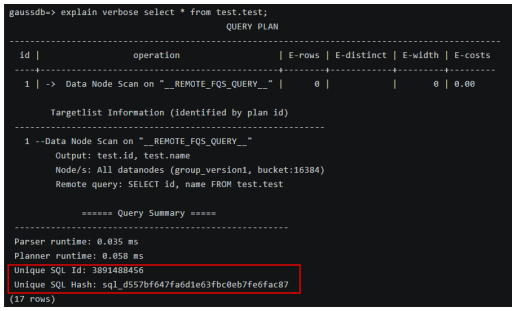
表 21-252 ALTER BLOCK RULE 参数说明

参数	描述	取值范围
block_name	需要修改属性的查询过滤规则名称。	字符串，需符合**标识符命名规范**。

TO	对查询过滤规则生效的对象或客户端IP，支持以下方式：
	• **TO user_name@'host'**：用户名以及客户端IP，两者都要满足，查询过滤规则才能生效。
	• **TO user_name**：用户名，即只判定满足用户名条件，查询过滤规则就生效，不区分客户端IP。
	• **TO 'host'**：客户端IP，即只判定满足客户端IP条件，查询过滤规则就生效，不区分用户。
		举例：查询过滤规则对来自客户端IP为192.168.0.59且登录用户名为user1时生效。
		`ALTER BLOCK RULE query_block TO user1@'192.168.0.59' FOR SELECT  FILTER BY SQL('test.test');`
		查询过滤规则对登录用户名为user1时生效（无论客户端IP是哪个）。
		`ALTER BLOCK RULE query_block TO user1 FOR SELECT  FILTER BY SQL('test.test');`
		查询过滤规则对来自客户端IP 192.168.0.59的访问时都生效（无论哪个用户登录）。
		`ALTER BLOCK RULE query_block TO '192.168.0.59' FOR SELECT  FILTER BY SQL('test.test');`

FOR	语句类型，支持对UPDATE/SELECT/INSERT/DELETE/MERGE INTO五种类型语句进行限制。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FILTER BY	过滤方法，支持两种形式： ● **SQL**：根据SQL语句内容进行过滤，通过关键词对语句进行正则匹配，例如表名，其长度不能超过1024，建议尽量精简。 ● **TEMPLATE**：根据模板进行过滤，**此方法不区分用户，针对所有用户执行的特定语句都实施拦截，拦截效果更高效**，取值如下： – **unique_sql_id**：归一化的64位哈希值，重复概率较sql_hash大一些。 – **sql_hash**：归一化的哈希值（md5），一般不会重复，相较unique_sql_id更推荐使用。	unique_sql_id和sql_hash取值，可以通过EXPLAIN VERBOSE查询，例如： EXPLAIN VERBOSE SELECT * FROM test.test; 
with_parameter	查询过滤规则选项参数。	● application_name，客户端名称。 ● query_band，负载标识。 ● table_num，语句扫描的基表个数。 ● partition_num，算子预估扫描的最大分区数。 ● estimate_row，算子预估扫描的最大表行数。 ● resource_pool，需要切换到的资源池名称。 ● max_active_num，规则对应语句的最大并发数。 ● is_warning，控制拦截到语句的行为是告警还是报错，on表示告警，off为报错，默认为off。

示例

创建查询规则query_block。

 版权所有 © 华为云计算技术有限公司

```
CREATE BLOCK RULE query_block FILTER BY SQL('update table_table set a=1');
```

修改查询规则query_block的属性。

```
ALTER BLOCK RULE query_block
TO user1@'192.168.0.59'
FOR SELECT
FILTER BY
SQL('select * from table_name')
WITH (application_name='gsql',
query_band='test1',
table_num='2',
partition_num='3',
estimate_row='1000',
resource_pool='rsp1',
max_active_num='3',
is_warning='off'
);
```

通过视图**pg_get_blockruledef**查询修改后的过滤规则。

```
SELECT * FROM pg_get_blockruledef('query_block');
```

```
gaussdb=> SELECT * FROM pg_get_blockruledef('query_block');
                                                
```

```
);
SELECT * FROM pg_get_blockruledef('query_block');
```

从回显看出，原来的application_name、query_band参数仍保留。

相关链接

CREATE BLOCK RULE，DROP BLOCK RULE

21.13.9.9 DROP BLOCK RULE

功能描述

该语法用于删除查询过滤规则。**查询过滤规则**，包括查询过滤规则名称、绑定的客户端名称、客户端IP、用户以及匹配方式等。在日常数据库操作中，异常SQL的执行可能会导致资源过度消耗，影响数据库性能和稳定性，通过创建查询过滤规则，可以将此类SQL语句进行拦截。

注意事项

- 该语法仅9.1.0.100及以上集群版本支持。
- 只有拥有数据库所有者权限的用户或者授予gs_role_block角色权限的用户才能执行DROP BLOCK RULE命令，系统管理员默认拥有此权限。
- 普通用户无法执行CREATE BLOCK RULE，可通过系统管理员进行授权，授权语法：
  ```
  GRANT gs_role_block TO user;
  ```
  建议**查询过滤规则操作尽量缩小适用范围，避免误过滤，或者范围过大导致性能劣化**。

语法格式

```
DROP BLOCK RULE [ IF EXISTS ] block_name;
```

参数说明

表 21-253 DROP BLOCK RULE 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的查询过滤规则不存在时返回一个提示而不是错误。	-
block_name	需要删除的查询过滤规则名称。	已存在的查询过滤规则名称。

版权所有 © 华为云计算技术有限公司

示例

创建名称为query_block的查询过滤规则。

```
CREATE BLOCK RULE query_block FILTER BY SQL('update table_table set a=1');
```

删除名称为query_block的查询过滤规则。

```
DROP BLOCK RULE query_block;
DROP BLOCK RULE IF EXISTS query_block;
```

相关链接

CREATE BLOCK RULE，**ALTER BLOCK RULE**

21.13.10 性能和执行计划

21.13.10.1 CREATE STATISTICS

功能描述

在当前数据库中为表创建一个扩展统计信息对象，该对象由命令的执行者所有。该语法仅8.2.1.200及以上集群版本支持（8.3.0集群版本不支持）。

注意事项

- 目前该命令仅支持创建表达式统计信息。
- 不支持对临时表创建表达式统计信息。
- 不支持在表的分区上创建表达式统计信息。
- 不支持对包含聚集函数、子查询、窗口函数、返回集合类型的表达式创建统计信息。

语法格式

```
CREATE STATISTICS [ IF NOT EXISTS ] statistics_name
    ON ( expression ) [, ...]
    FROM table_name
```

参数说明

表 21-254 CREATE STATISTICS 参数说明

参数	描述	取值范围
statistics_name	新统计信息对象的名称。	字符串，需符合**标识符命名规范**。
expression	要收集统计信息的表达式，支持单个或多个表达式，多个表达式通过逗号分隔。	-
table_name	用于收集统计信息的表名。	-

版权所有 © 华为云计算技术有限公司

示例

创建示例表并插入数据。

```
CREATE TABLE t1_row(id int, dt timestamp) WITH (orientation = row) DISTRIBUTE BY HASH (id);
INSERT INTO t1_row values (generate_series(1, 721), generate_series('2023-01-01'::timestamp,
'2023-01-31'::timestamp, '1 hour'::interval));
```

为表创建一个表达式统计信息对象，使用ANALYZE命令收集统计信息。

```
CREATE STATISTICS s1_t1_row ON EXTRACT(HOUR FROM dt), EXTRACT(DAY FROM dt) FROM t1_row;
ANALYZE t1_row;
```

查看语句的查询计划。

```
SET enable_fast_query_shipping = OFF;
EXPLAIN VERBOSE SELECT * FROM t1_row WHERE EXTRACT(HOUR FROM dt) BETWEEN 0 AND 12;
```

相关链接

DROP STATISTICS

21.13.10.2 DROP STATISTICS

功能描述

删除一个扩展统计信息对象。该语法仅8.2.1.200及以上集群版本支持（8.3.0集群版本不支持）。

注意事项

无

语法格式

```
DROP STATISTICS [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

参数说明

表 21-255 DROP STATISTICS 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的扩展统计信息对象不存在，不会抛出一个错误，而会发出一个通知，告知对象不存在。	-
name	要删除的扩展统计信息对象名。	已存在的扩展统计信息对象名。
CASCADE | RESTRICT	该参数不起作用，没有对象依赖于扩展统计信息对象。	-

示例

删除一个扩展统计信息对象。

版权所有 © 华为云计算技术有限公司

```
DROP STATISTICS s1_t1_row;
```

相关链接

CREATE STATISTICS

21.13.11 连接&会话&资源清理

21.13.11.1 CLEAN CONNECTION

功能描述

当数据库有异常时，用来清理数据库连接。允许在指定节点上清理指定数据库的指定用户的相关连接。

注意事项

参数**dbname**和**username**必须至少指定一个。

语法格式

```
CLEAN CONNECTION
   TO { COORDINATOR ( nodename [, ... ] ) | NODE ( nodename [, ... ] )| ALL [ CHECK ] [ FORCE ] }
   [ FOR DATABASE dbname ]
   [ TO USER username ];
```

参数说明

表 21-256 CLEAN CONNECTION 参数说明

参数	描述	取值范围
CHECK	仅在节点列表为TO ALL时可以指定。如果指定该参数，会在清理连接之前检查数据库是否被其他会话连接访问。此参数主要用于DROP DATABASE之前的连接访问检查，如果发现有其他会话连接，则将报错并停止删除数据库。	-
FORCE	仅在节点列表为TO ALL时可以指定，如果指定该参数，所有和指定dbname和username相关的线程都会收到SIGTERM信号，然后被强制关闭。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
COORDINATOR (nodename [, ...]) | NODE (nodename [, ...]) | ALL	删除指定节点上的连接。有三种场景： ● 删除指定CN上的连接。 ● 删除指定DN上的连接。 ● 删除所有节点上的连接，包括CN和DN。	可替换其中的nodename为已存在的节点名。
dbname	删除指定数据库上的连接。如果不指定，则删除所有数据库的连接。	有效的数据库名。
username	删除指定用户上的连接。如果不指定，则删除所有用户的连接。	有效的用户名。

示例

删除数据库template1在dn1和dn2节点上的连接。

```
CLEAN CONNECTION TO NODE (dn1,dn2) FOR DATABASE template1;
```

删除用户jack在dn1节点上的连接。

```
CLEAN CONNECTION TO NODE (dn1) TO USER jack;
```

删除在数据库gaussdb上的所有连接。

```
CLEAN CONNECTION TO ALL FORCE FOR DATABASE gaussdb;
```

21.13.11.2 ALTER SYSTEM KILL SESSION

功能描述

ALTER SYSTEM KILL SESSION命令用于结束一个会话。

注意事项

无。

语法格式

```
ALTER SYSTEM KILL SESSION 'session_sid, serial' [ IMMEDIATE ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-257 ALTER SYSTEM KILL SESSION 参数说明

参数	描述	取值范围
session_sid, serial	会话的SID和SERIAL。	通过查看系统表V$SESSION可查看所有会话的SID和SERIAL。
IMMEDIATE	表示会话将在命令执行后立即结束。	-

示例

查询会话信息。

```
SELECT sid,serial#,username FROM V$SESSION;

       sid        | serial# | username
------------------+---------+----------
 140131075880720 |       0 |
 140131025549072 |       0 |
 140131073779472 |       0 |
 140131071678224 |       0 |
 140131125774096 |       0 |
 140131127875344 |       0 |
 140131113629456 |       0 |
 140131094742800 |       0 |
(8 rows)
```

结束SID为140131075880720的会话。

```
ALTER SYSTEM KILL SESSION '140131075880720,0' IMMEDIATE;
```

21.13.11.3 ALTER SESSION

功能描述

ALTER SESSION命令用于定义或修改那些对当前会话有影响的条件或参数。修改后的会话参数会一直保持，直到断开当前会话。

注意事项

- 如果执行SET TRANSACTION之前没有执行START TRANSACTION，则事务立即结束，命令无法显示效果。
- 可以用START TRANSACTION里面声明所需要的transaction_mode(s)的方法来避免使用SET TRANSACTION。

语法格式

- 设置会话的事务参数。

```
ALTER SESSION SET [ SESSION CHARACTERISTICS AS ] TRANSACTION
    { ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED } | { READ ONLY  | READ
WRITE } } [, ...] ;
```


版权所有 © 华为云计算技术有限公司

- 设置会话的其他运行时参数。

```
ALTER SESSION SET
    {{config_parameter { { TO  | =  } { value | DEFAULT }
      | FROM CURRENT }} | CURRENT_SCHEMA [ TO | = ] { schema | DEFAULT }
      | TIME ZONE time_zone
      | SCHEMA schema
      | NAMES encoding_name
      | ROLE role_name PASSWORD 'password'
      | SESSION AUTHORIZATION { role_name PASSWORD 'password' | DEFAULT }
      | XML OPTION { DOCUMENT | CONTENT }
    } ;
```

参数说明

表 21-258 ALTER SESSION 参数说明

参数	描述	取值范围
SESSION	声明的参数只对当前会话起作用。如果SESSION和LOCAL都没出现，则SESSION为缺省值。 如果在事务中执行了此命令，命令的产生影响将在事务回滚之后消失。如果该事务已提交，影响将持续到会话的结束，除非被另外一个SET命令重置参数。	-
config_parameter	可设置的运行时参数的名称。可用的运行时参数可以使用SHOW ALL命令查看。 **说明** 部分通过SHOW ALL查看的参数不能通过SET设置。如max_datanodes。	-
value	config_parameter的新值。	可以声明为字符串常量、标识符、数字，或者逗号分隔的列表。DEFAULT用于把这些参数设置为它们的缺省值。
TIME ZONE timezone	用于指定当前会话的本地时区。	有效的本地时区。该选项对应的运行时参数名称为TimeZone，DEFAULT缺省值为PRC。
CURRENT_SCHEMA	CURRENT_SCHEMA用于指定当前的模式。	已存在的模式名称。
SCHEMA schema	同CURRENT_SCHEMA。此处的schema是个字符串。 例如：set schema 'public';	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
NAMES encoding_name	用于设置客户端的字符编码。等价于set client_encoding to encoding_name。	有效的字符编码。该选项对应的运行时参数名称为client_encoding，默认编码为UTF8。
XML OPTION option	用于设置XML的解析方式。	• CONTENT（缺省） • DOCUMENT

示例

创建模式ds：

```
CREATE SCHEMA ds;
```

设置模式搜索路径：

```
SET SEARCH_PATH TO ds, public;
```

设置日期时间风格为传统的POSTGRES风格（日在月前）：

```
SET DATESTYLE TO postgres, dmy;
```

设置当前会话的字符编码为UTF8：

```
ALTER SESSION SET NAMES 'UTF8';
```

设置时区为加州伯克利：

```
SET TIME ZONE 'PST8PDT';
```

设置时区为意大利：

```
SET TIME ZONE 'Europe/Rome';
```

设置当前模式：

```
ALTER SESSION SET CURRENT_SCHEMA TO tpcds;
```

设置XML OPTION为DOCUMENT：

```
ALTER SESSION SET XML OPTION DOCUMENT;
```

创建角色joe，并设置会话的角色为joe：

```
CREATE ROLE joe WITH PASSWORD '{password}';
ALTER SESSION SET SESSION AUTHORIZATION joe PASSWORD '{password}';
```

切换到默认用户：

```
ALTER SESSION SET SESSION AUTHORIZATION default;
```

相关链接

SET

版权所有 © 华为云计算技术有限公司

21.13.11.4 DISCARD

功能描述

释放与数据库会话相关的内部资源。

DISCARD命令用于重设部分或全部的会话的状态，不同的DISCARD子命令释放不同类型的资源；DISCARD ALL释放所有与当前会话相关的临时资源，并重置到其初始状态。

注意事项

- 执行DISCARD VOLATILE { TEMPORARY | TEMP }操作后，当前session内所有volatile临时表资源都会被清理，不支持清理单个volatile临时表资源。
- 如果某张全局临时表在某个会话占用资源，需要DISCARD依次清理所有会话的资源后，才能执行DDL操作。
- 执行DISCARD ALL成功之后，以pg_temp和pg_toast_temp开头的schema也会被删除。
- DISCARD ALL不允许在事务中执行。

语法格式

```
DISCARD {{ GLOBAL { TEMPORARY | TEMP } [ TABLE table_name ] } |
      { VOLATILE { TEMPORARY | TEMP } } |
      { ALL | TEMP | TEMPORARY | PLANS | SEQUENCES }}
```

参数说明

表 21-259 DISCARD 参数说明

参数	描述
VOLATILE { TEMPORARY | TEMP }	表示释放当前会话中VOLATILE临时表相关资源。
GLOBAL { TEMPORARY | TEMP } [TABLE table_name]	• DISCARD GLOBAL TEMP，释放当前会话中GLOBAL临时表相关资源。 • DISCARD GLOBAL TEMP TABLE table_name 释放当前会话中指定某张GLOBAL临时表的相关资源。
TEMP | TEMPORARY	释放当前会话中所有临时表的相关资源，包括VOLATILE临时表和GLOBAL临时表。
PLANS	释放当前会话中所有缓存的查询计划，并在下次使用相关prepare语句时强制进行重新计划。
SEQUENCES	丢弃缓存的所有序列相关的状态，包括currval()/lastval()信息和尚未由nextval()返回的任何预分配的序列值。

版权所有 © 华为云计算技术有限公司

参数	描述
ALL	释放所有与当前会话相关的临时资源，并将会话重置为其初始状态，这与执行以下语句序列具有几乎相同的效果： SET SESSION AUTHORIZATION DEFAULT; RESET ALL; DEALLOCATE ALL; CLOSE ALL; UNLISTEN *; SELECT pg_advisory_unlock_all(); DISCARD PLANS; DISCARD SEQUENCES; DISCARD TEMP;

示例

DISCARD全局临时表。

```
CREATE GLOBAL TEMP TABLE t_global_temp(a int,b int);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE

INSERT INTO t_global_temp VALUES(1,1),(2,2);
INSERT 0 2

DROP TABLE t_global_temp;
ERROR:  can not DROP TABLE when global temp table "t_global_temp" is in use

DISCARD GLOBAL TEMP TABLE t_global_temp;

DROP TABLE t_global_temp;
DROP TABLE
```

DISCARD VOLATILE清理当前会话中所有volatile临时表相关资源。

```
CREATE VOLATILE TEMP TABLE TX1(A INT) DISTRIBUTE BY HASH(A);
CREATE TABLE
CREATE VOLATILE TEMP TABLE TX2(A INT) DISTRIBUTE BY HASH(A);
CREATE TABLE

SELECT * FROM TX1;
 a
---
(0 rows)
SELECT * FROM TX2;
 a
---
(0 rows)

DISCARD VOLATILE TEMP;

SELECT * FROM TX1;
ERROR:  relation "tx1" does not exist
LINE 1: SELECT * FROM TX1;
                      ^
SELECT * FROM TX2;
ERROR:  relation "tx2" does not exist
LINE 1: SELECT * FROM TX2;
```

DISCARD TEMP操作清理当前会话中所有临时表相关资源。

```
CREATE GLOBAL TEMP TABLE t_global_temp(a int,b int);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO t_global_temp VALUES(1,1),(2,2);
INSERT 0 2

CREATE VOLATILE TEMP TABLE t_volatile_temp(a int,b int);
CREATE TEMP TABLE t_temp(a int,b int);

DISCARD TEMP;

SELECT * FROM t_global_temp;
 a | b
---+---
(0 rows)

SELECT * FROM t_volatile_temp;
ERROR:  relation "t_volatile_temp" does not exist
LINE 1: select * from t_volatile_temp;

SELECT * FROM t_temp;
ERROR:  relation "t_temp" does not exist
LINE 1: select * from t_temp;
```

21.13.11.5 TRUNCATE

功能描述

TRUNCATE语法用于清理表数据，快速地从表中删除所有行。

它与在目标表上进行无条件的DELETE的效果相同，但由于TRUNCATE实际上并不扫描表，因此速度更快，尤其在大表上操作效果更明显。

TRUNCATE TABLE、DELETE TABLE 和 DROP TABLE 差异

表 21-260 TRUNCATE TABLE、DELETE TABLE 和 DROP TABLE 差异

差异类别	TRUNCATE TABLE	DELETE TABLE	DROP TABLE
语法	DDL	DML	DDL
删除内容	删除表中的所有数据，但不会删除表结构	删除内容，不删除定义	删除内容和定义
是否释放空间	释放空间	不释放空间	释放空间
执行速度	速度较快 释放存储表数据所用数据页来删除数据，并且只在事务日志中记录页的释放，使用系统和事务日志资源少。	速度慢 每删除一行，在事务日志中为所删除每行记录一项。	速度最快
适用场景	如果需要快速删除整个表中的数据，保留表结构，且是大批量数据的场景。	需要根据特定条件删除数据且数据量可控的情况。	需要删除整个表，包括表结构和数据。

版权所有 © 华为云计算技术有限公司

注意事项

- 请谨慎执行TRUNCATE TABLE，确认表数据可以删除或者已有备份后，再执行删除操作。TRUNCATE TABLE命令删除表数据后，数据将无法恢复。
- TRUNCATE全局临时表只对当前会话的数据进行TRUNCATE。不影响其他会话的数据。
- 存算分离架构下，不支持同时TRUNCATE普通表和临时表。

警告

- 避免在业务高峰期执行ALTER TABLE/ALTER TABLE PARTITION（增删改查、DROP PARTITION）、TRUNCATE操作，避免有长SQL阻塞ALTER、TRUNCATE操作或SQL业务被ALTER、TRUNCATE阻塞。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

清理表数据。

```
TRUNCATE [ TABLE ] [ ONLY ] {[[database_name.]schema_name.]table_name [ * ]} [, ... ]
    [ CONTINUE IDENTITY ] [ CASCADE | RESTRICT ] ;
```

参数说明

表 21-261 TRUNCATE 参数说明

参数	描述	取值范围
ONLY	指定清空表的范围。 该参数仅为了兼容PG保留，DWS不支持继承表。	• 如果指定ONLY，则只有指定表会被清空。 • 如果没有指定ONLY，则指定表及其所有继承表（若有）都会被清空。
database_name	要清空的表所在的数据库名称。	已存在的数据库名。
schema_name	要清空的表的模式名称。	已存在的模式名。
table_name	指定要清空的表名（可以用模式修饰）。	已存在的表名。
CONTINUE IDENTITY	不更改序列的值。	该参数为缺省值。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
CASCADE | RESTRICT	可选参数，清空有外键引用的表的方式。	• CASCADE：表示级联清空所有在该表上有外键引用的表，或者由于CASCADE而被添加到组中的表。 • RESTRICT：表示如果其他表在该表上有外键引用，则拒绝清空。 缺省值为RESTRICT。

示例

1. 创建示例表test_t并插入数据。

```
CREATE TABLE test_t
(
    col_id INT PRIMARY KEY,
    col_number INT NOT NULL,
    col_date DATE NOT NULL,
    col_price NUMERIC (10,2),
    col_status TEXT
)
WITH (ORIENTATION = COLUMN)
DISTRIBUTE BY HASH (col_id);

INSERT INTO test_t VALUES
(01,300,'2025-03-01',99.20,'sold'),
(02,400,'2025-04-01',95.50,'sold'),
(03,450,'2025-05-01',420.50,'sold'),
(04,100,'2025-06-01',100.85,'restock');
```

2. 查看表test_t的数据。

```
SELECT * FROM test_t;
 col_id | col_number |      col_date       | col_price | col_status
--------+------------+---------------------+-----------+------------
      1 |        300 | 2025-03-01 00:00:00 |     99.20 | sold
      2 |        400 | 2025-04-01 00:00:00 |     95.50 | sold
      3 |        450 | 2025-05-01 00:00:00 |    420.50 | sold
      4 |        100 | 2025-06-01 00:00:00 |    100.85 | restock
(4 rows)
```

3. 清理表test_t。

```
TRUNCATE test_t;
```

返回结果如下，表示已成功清理。

```
TRUNCATE TABLE
```

4. 查看表定义。TRUNCATE TABLE操作清空表，保留表定义。

```
SELECT * FROM pg_get_tabledef('public.test_t');
                              pg_get_tabledef
-------------------------------------------------------------------------------
 SET search_path = public;                                                    +
 CREATE  TABLE test_t (                                                       +
        col_id integer NOT NULL,                                               +
        col_number integer NOT NULL,                                               +
        col_date timestamp(0) without time zone NOT NULL,                                  +
        col_price numeric(10,2),                                              +
        col_status text                                                   +
 )                                                                        +
 WITH (orientation=column, compression=middle, colversion=2.0, enable_delta=false)+
 DISTRIBUTE BY HASH(col_id)                                                   +
;
(1 row)
```


版权所有 © 华为云计算技术有限公司

21.13.11.6 VACUUM

功能描述

VACUUM回收表或B-Tree索引中已经删除的行所占据的存储空间。在一般的数据库操作里，那些已经DELETE的行并没有从它们所属的表中物理删除；在完成VACUUM之前它们仍然存在。因此有必要周期地运行VACUUM，特别是在经常更新的表上。

VACUUM 与 VACUUM FULL 的区别

VACUUM命令存在两种形式，VACUUM和VACUUM FULL，VACUUM命令做的是LAZY VACUUM。从字面意思就可以看出来，LAZY VACUUM是VACUUM FULL的简化版，具体区别见下表。

表 21-262 VACUUM 与 VACUUM FULL 的区别

比较维度	VACUUM	VACUUM FULL
空间清理	如果删除记录位于表的末端，其所占用的空间将会被物理释放并归还操作系统。如果删除记录不是末端数据，会将表中或索引中dead tuple所占用的空间置为可用状态，从而复用这些空间。	不论被清理的数据处于何处，这些数据所占用的空间都将被物理释放并归还于操作系统。当再有数据插入后，系统会分配新的磁盘页面使用。
锁类型	4级锁，共享更新排他锁（ShareUpdateExclusiveLock），可以与其他操作并行。	8级锁，访问排他锁（AccessExclusiveLock），执行期间会同时锁表，基于该表的其他操作全部挂起。
物理空间	不会释放。	会释放。
事务ID	不回收。	回收
执行开销	开销较小，可以定期执行。	开销巨大，建议确认数据库所占磁盘页面空间接近临界值再执行操作，且最好选择数据量操作较少的时段完成。
执行效果	执行后会有所提升。	执行完后，基于该表的操作效率大大提升。

关于自动清理(AUTOVACUUM）

由于VACUUM操作可释放空间，对提升查询性能至关重要，DWS提供了自动清理的能力，用户可根据自身业务需要，在指定条件下，让系统触发自动清理功能，通过管理控制台配置，参见《数据仓库服务用户指南》的“管理运维计划”章节。

版权所有 © 华为云计算技术有限公司

注意事项

- 如果没有参数，VACUUM处理当前数据库里用户拥有相应权限的每个表。如果参数指定了一个表，VACUUM只处理指定的那个表。
- 要对一个表进行VACUUM操作，通常用户必须是表的所有者，被授予了指定表VACUUM权限的用户或者被授予了gs_role_vacuum_any角色的用户，系统管理员默认拥有此权限。数据库的所有者允许对数据库中除了共享目录以外的所有表进行VACUUM操作（该限制意味着只有系统管理员才能真正对一个数据库进行VACUUM操作）。VACUUM命令会跳过那些用户没有权限的表进行垃圾回收操作。
- VACUUM不能在事务块内执行。
- 建议生产数据库经常清理（至少每晚一次），以保证不断地删除失效的行。VACUUM ANALYZE先执行一个VACUUM操作，然后给每个选定的表执行一个ANALYZE。尤其是在增删了大量记录之后，对受影响的表执行VACUUM ANALYZE命令是一个很好的习惯。这样将更新系统目录为最近的更改，并且允许查询优化器在规划用户查询时有更好的选择。
- VACUUM会导致I/O流量的大幅增加，这可能会影响其他活动会话的性能。因此，有时候会建议使用基于开销的VACUUM延迟特性。
- 如果指定了VERBOSE选项，VACUUM将打印处理过程中的信息，以表明当前正在处理的表。各种有关当前表的统计信息也会打印出来。
- 语法格式中含有带括号的选项列表时，选项可以以任何顺序写入。如果没有括号，则选项必须按语法显示的顺序给出。
- VACUUM列存表内部执行的操作包括四个：VACUUM主表、VACUUM主表的desc表、VACUUM主表的delta表、迁移delta表中的数据到主表。该操作不会回收delta表的存储空间，如果要回收delta表的冗余存储空间，需要对该列存表执行VACUUM DELTAMERGE。VACUUM主表操作默认开启，可以通过调节参数colvacuum_threshold_scale_factor来关闭该功能。
- VACUUM列存主表功能不支持临时表、冷热表和时序表。
- VACUUM列存主表功能对空间的回收是延后的，若需要立即回收空间，在执行VACUUM后，运行vac_fileclear_relation函数，将会对指定表施加排他锁，完成空间回收。
- 简单的VACUUM（不带FULL选项）只是简单地回收空间并且令其可以再次使用。这种形式的命令可以和对表的普通读写并发操作，因为没有请求排他锁。VACUUM FULL执行更广泛的处理，包括跨块移动行，以便把表压缩到最少的磁盘块数目里。这种形式要慢许多并且在处理的时候需要在表上施加一个排他锁。
- 不建议日常使用FULL选项，但是可以在特殊情况下使用。例如在用户删除了一个表的大部分行之后，希望从物理上缩小该表以减少磁盘空间占用。VACUUM FULL通常要比单纯的VACUUM收缩更多的表尺寸。如果执行此命令后所占用物理空间无变化（未减少），请确认是否有其他活跃事务（删除数据事务开始之前开始的事务，并在VACUUM FULL执行前未结束）存在，如果有等其他活跃事务退出进行重试。
- VACUUM和VACUUM FULL时，会根据参数vacuum_defer_cleanup_age延迟清理行存表记录，即不会立即清理刚删除的元组。
- VACUUM FULL系统表只能离线操作，在线VACUUM FULL系统表除了会锁表，还可能导致一些异常情况并产生报错。
- 对列存分区表执行VACUUM FULL，会同时锁表和锁分区。
- VACUUM FULL操作分区表时与用户DML语句在如下特定场景有并发时可能发生分布式死锁，请谨慎操作：

版权所有 © 华为云计算技术有限公司

  - VACUUM FULL子分区与insert/update/delete主表。
  - VACUUM FULL全表与select全表/select子分区。
- 在存算分离架构下，对于全库VACUUM/全库VACUUM FULL/全库VACUUM ANALYZE/全库VACUUM DELTAMERGE提示不支持。
- 在存算分离架构下，正在重分布的列存3.0的表执行VACUUM操作时，系统不会进行实际的数据清理动作，但会返回如下提示信息：

  执行VACUUM操作时，会跳过VACUUM操作。

```
VACUUM tpcds.web_returns_p1;
NOTICE:  skip vacuum relation "web_returns_p1" since it is in redistribution state
```

  执行VACUUM ANALYZE操作时，会跳过VACUUM操作，但是会收集统计信息。

```
VACUUM ANALYZE tpcds.web_returns_p1;
NOTICE:  skip vacuum relation "web_returns_p1" since it is in redistribution state
```

警告

- 对表执行VACUUM FULL操作时会触发表重建和表上的索引重建（重建过程中会先把数据写到一个新的数据文件中，重建完成之后删除原始文件），当表比较大时，重建会消耗较多的磁盘空间。当磁盘空间不足时，要谨慎对待大表VACUUM FULL操作，防止触发集群只读。
- 定期对脏页率、小CU占比超过25%的表执行VACUUM FULL，普通表需在低峰期执行，系统表需离线执行。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

- 回收空间并更新统计信息，关键字顺序必须按语法显示的顺序给出。

```
VACUUM [ ( { FULL | FREEZE | VERBOSE | {ANALYZE | ANALYSE }} [,...] ) ]
    [ table_name [ (column_name [, ...] ) ] ] [ PARTITION ( partition_name ) ];
```

- 仅回收空间，不更新统计信息。

```
VACUUM [ FULL [COMPACT] ] [ FREEZE ] [ VERBOSE ] [ table_name ] [ PARTITION
( partition_name ) ];
```

- 回收空间并更新统计信息，且对关键字顺序有要求。

```
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] { ANALYZE | ANALYSE } [ VERBOSE ]
    [ table_name [ (column_name [, ...] ) ] ] [ PARTITION ( partition_name ) ];
```

- 针对HDFS表和列存表，将delta table中的数据转移到主表存储。（partition_name参数仅8.2.1.200及以上集群版本支持）

```
VACUUM DELTAMERGE [ table_name ][partition_name];
```

- 针对HDFS表，删除HDFS表在HDFS存储上的空值分区目录。

```
VACUUM HDFSDIRECTORY [ table_name ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-263 VACUUM 参数说明

参数	描述	取值范围
FULL	选择该参数，可以恢复更多的空间，但是会耗时更多，并且在表上施加了排他锁。 FULL选项还可以带有COMPACT参数，只针对HDFS表，指定该参数的VACUUM FULL操作性能要好于未指定该参数的VACUUM FULL操作。 **须知** • COMPACT和PARTITION参数不能同时使用。 • 使用FULL参数会导致统计信息丢失，如果需要收集统计信息，请在VACUUM FULL语句中加上analyze关键字。	-
FREEZE	用于解决事务ID回绕(Transaction ID Wraparound)问题，指定FREEZE相当于执行VACUUM时将vacuum_freeze_min_age参数设为0。 **须知** DWS使用64位xid，理论上不会存在绕回问题，所以不建议客户自己操作VACUUM FREEZE。	-
VERBOSE	为每个表打印一份详细的清理工作报告。	-
ANALYZE | ANALYSE	更新用于优化器的统计信息，以决定执行查询的最有效方法。	-
table_name	需要清理的表的名称（可以带有模式修饰）。	缺省时为当前数据库中的所有表。
column_name	需要分析的具体的字段名称。	缺省时为所有字段。
PARTITION	指定清理的表为分区表。HDFS表不支持PARTITION参数。 **说明** PARTITION参数和COMPACT同时使用会报错： COMPACT can not be used with PARTITION.	-
partition_name	要清理的表的分区名称。	缺省时为所有分区。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
DELTAMERGE	只针对HDFS表和列存表，将HDFS表或列存表的delta表中的数据转移到主表上存储。对HDFS表而言，当delta表中数据量小于六万行，则不作迁移，只有在大于或等于六万行数据时，将delta表中所有数据迁移到HDFS上，并通过truncate清理delta表的存储空间。对列存表而言，此操作总是会将delta表中的数据全部转移到CU中。 **说明** 为了检查列存delta表中的信息（HDFS表的delta表数据可以通过explain analyze查到），提供下述DFX函数，用于获取某个列存表的delta表中数据存储情况： • pgxc_get_delta_info(TEXT)，传入参数为列存表名，搜集并显示各个节点上的对应delta表信息，包括当前存活tuple数量、表大小、使用的最大block ID。 • get_delta_info(TEXT)，传入参数为列存表名，汇总pgxc_get_delta_info得到的结果，返回其delta表整体的当前存活tuple数量、表大小、使用的最大block ID。查询临时表delta信息时，需要显示指定临时表的schema，否则会报错找不到表。	-
HDFSDIRECTORY	只针对HDFS表，清理HDFS表（存储在HDFS表的目录下）的空值分区目录。	-

示例

清理当前数据库中的所有表：

```
VACUUM;
```

仅回收表tpcds.web_returns_p1分区P2的空间，不更新统计信息：

```
VACUUM FULL tpcds.web_returns_p1 PARTITION(P2);
```

回收表tpcds.web_returns_p1空间，并更新统计信息：

```
VACUUM FULL ANALYZE tpcds.web_returns_p1;
```

清理当前数据库中的所有表并收集查询优化器的统计信息：

```
VACUUM ANALYZE;
```

仅清理特定表reason：

```
VACUUM (VERBOSE, ANALYZE) tpcds.reason;
```

对列存表table_delta进行DELTAMERGE操作：

```
VACUUM DELTAMERGE tpcds.table_delta;
```

仅对列存表table_delta的分区p1进行DELTAMERGE操作：

```
VACUUM DELTAMERGE tpcds.table_delta partition(p1);
```

21.13.12 设置与注释

版权所有 © 华为云计算技术有限公司

21.13.12.1 SHOW

功能描述

SHOW将显示当前运行时参数的数值。可以使用SET语句来设置这些参数。

注意事项

SHOW可以查看的某些参数是只读的，可以查看但不能设置它们的值。

语法格式

```
SHOW
 {
   configuration_parameter |
   CURRENT_SCHEMA |
   TIME ZONE |
   TRANSACTION ISOLATION LEVEL |
   SESSION AUTHORIZATION |
   ALL
 };
```

参数说明

显示变量的参数请参见RESET的**参数说明**。

示例

显示timezone参数值。

```
SHOW timezone;
```

显示参数DateStyle的当前设置。

```
SHOW DateStyle;
```

显示所有参数的当前设置。

```
SHOW ALL;
```

相关链接

SET，**RESET**

21.13.12.2 SET

功能描述

用于修改运行时配置参数。

注意事项

大多数运行时参数都可以用SET在运行时设置，但有些则在服务运行过程中或会话开始之后不能修改。

版权所有 © 华为云计算技术有限公司

语法格式

- 设置所处的时区。
```
SET [ SESSION | LOCAL ] TIME ZONE { timezone | LOCAL | DEFAULT };
```
- 设置所属的模式。
```
SET [ SESSION | LOCAL ]
    {CURRENT_SCHEMA { TO | = } { schema | DEFAULT }
    | SCHEMA 'schema'};
```
- 设置客户端编码集。
```
SET [ SESSION | LOCAL ] NAMES encoding_name;
```
- 设置XML的解析方式。
```
SET [ SESSION | LOCAL ] XML OPTION { DOCUMENT | CONTENT };
```
- 设置其他运行时参数。
```
SET [ LOCAL | SESSION ]
    { {config_parameter { { TO | = } { value | DEFAULT }
                        | FROM CURRENT }}};
```

参数说明

表 21-264 SET 参数说明

参数	描述	取值范围
SESSION	声明的参数只对当前会话起作用。如果SESSION和LOCAL都没出现，则SESSION为缺省值。 如果在事务中执行了此命令，命令的产生影响将在事务回滚之后消失。如果该事务已提交，影响将持续到会话的结束，除非被另外一个SET命令重置参数。	-
LOCAL	声明的参数只在当前事务中有效。在COMMIT或ROLLBACK之后，会话级别的设置将再次生效。 不论事务是否提交，此命令的影响只持续到当前事务结束。一个特例是：在一个事务里面，既有SET命令，又有SET LOCAL命令，且SET LOCAL在SET后面，则在事务结束之前，SET LOCAL命令会起作用，但事务提交之后，则是SET命令会生效。	-
TIME ZONE timezone	用于指定当前会话的本地时区。	有效的本地时区。该选项对应的运行时参数名称为TimeZone，DEFAULT缺省值为PRC。
CURRENT_SCHEMA schema	CURRENT_SCHEMA用于指定当前的模式。	已存在模式名称。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
SCHEMA schema	同CURRENT_SCHEMA。此处的schema是个字符串。 例如：set schema 'public';	-
NAMES encoding_name	用于设置客户端的字符编码。等价于set client_encoding to encoding_name。	有效的字符编码。该选项对应的运行时参数名称为client_encoding，默认编码为UTF8。
XML OPTION option	用于设置XML的解析方式。	CONTENT（缺省）、DOCUMENT
config_parameter	可设置的运行时参数的名称。可用的运行时参数可以使用SHOW ALL命令查看。 **说明** 部分通过SHOW ALL查看的参数不能通过SET设置。如max_datanodes。	-
value	config_parameter的新值。可以声明为字符串常量、标识符、数字，或者逗号分隔的列表。DEFAULT用于把这些参数设置为它们的缺省值。	-

示例

设置模式tpcds搜索路径。

```
SET search_path TO tpcds, public;
```

把日期时间风格设置为传统的POSTGRES风格（日在月前）。

```
SET datestyle TO postgres;
```

相关链接

RESET，**SHOW**

21.13.12.3 RESET

功能描述

RESET将指定的运行时参数恢复为缺省值。这些参数的缺省值是指postgresql.conf配置文件中所描述的参数缺省值。

RESET命令与如下命令的作用相同：

SET configuration_parameter TO DEFAULT

注意事项

RESET的事务性行为和SET相同：它的影响将会被事务回滚撤销。

版权所有 © 华为云计算技术有限公司

语法格式

```
RESET {configuration_parameter | CURRENT_SCHEMA | TIME ZONE | TRANSACTION ISOLATION LEVEL | SESSION AUTHORIZATION | ALL };
```

参数说明

表 21-265 RESET 参数说明

参数	描述	取值范围
configuration_parameter	运行时参数的名称。	可以使用SHOW ALL命令查看运行时参数。 **说明** 部分通过SHOW ALL查看的参数不能通过SET设置。如max_datanodes。
CURRENT_SCHEMA	当前模式。	-
TIME ZONE	时区。	-
TRANSACTION ISOLATION LEVEL	事务的隔离级别。	-
SESSION AUTHORIZATION	当前会话的用户标识符。	-
ALL	所有运行时参数。	-

示例

将timezone设为缺省值。

```
RESET timezone;
```

将所有参数设置为缺省值。

```
RESET ALL;
```

相关链接

SET，**SHOW**

21.13.12.4 SET CONSTRAINTS

功能描述

SET CONSTRAINTS设置当前事务检查行为的约束条件。

IMMEDIATE约束是在每条语句后面进行检查。DEFERRED约束一直到事务提交时才检查。每个约束都有自己的模式。

从创建约束条件开始，一个约束总是设定为DEFERRABLE INITIALLY DEFERRED，DEFERRABLE INITIALLY IMMEDIATE，NOT DEFERRABLE三个特性之一。第三种总是

版权所有 © 华为云计算技术有限公司

IMMEDIATE，并且不会受SET CONSTRAINTS影响。前两种以指定的方式启动每个事务，但是其行为可以在事务里用SET CONSTRAINTS改变。

带着一个约束名列表的SET CONSTRAINTS改变这些约束的模式（都必须是可推迟的）。如果有多个约束匹配某个名字，则所有都会被影响。SET CONSTRAINTS ALL改变所有可推迟约束的模式。

当SET CONSTRAINTS把一个约束从DEFERRED改成IMMEDIATE的时候，新模式反作用式地起作用：任何将在事务结束准备进行的数据修改都将在SET CONSTRAINTS的时候执行检查。如果违反了任何约束，SET CONSTRAINTS都会失败（并且不会修改约束模式）。因此，SET CONSTRAINTS可以用于强制在事务中某一点进行约束检查。

目前，只有外键约束被该设置影响。检查和唯一约束总是不可推迟的。

注意事项

SET CONSTRAINTS只在当前事务里设置约束的行为。因此，如果用户在事务块之外（START TRANSACTION/COMMIT对）执行这个命令，它将没有任何作用。

语法格式

```
SET CONSTRAINTS { ALL | { name } [, ...] } { DEFERRED | IMMEDIATE };
```

参数说明

表 21-266 SET CONSTRAINTS 参数说明

参数	描述	取值范围
name	约束名。	已存在的约束名。可以在系统表pg_constraint中查到。
ALL	所有约束。	-
DEFERRED	约束一直到事务提交时才检查。	-
IMMEDIATE	约束在每条语句后进行检查。	-

示例

设置所有约束在事务提交时检查。

```
SET CONSTRAINTS ALL DEFERRED;
```

21.13.12.5 SET SESSION AUTHORIZATION

功能描述

把当前会话里的会话用户标识和当前用户标识都设置为指定的用户。

版权所有 © 华为云计算技术有限公司

注意事项

只有在初始会话用户有系统管理员权限的时候，会话用户标识符才能改变。否则，只有在指定了被认证的用户名的情况下，系统才接受该命令。

语法格式

- 为当前会话设置会话用户标识符和当前用户标识符。
  ```
  SET [ SESSION | LOCAL ] SESSION AUTHORIZATION role_name PASSWORD '{password}';
  ```
- 重置会话和当前用户标识符为初始认证的用户名。
  ```
  {SET [ SESSION | LOCAL ] SESSION AUTHORIZATION DEFAULT
     | RESET SESSION AUTHORIZATION};
  ```

参数说明

表 21-267 SET SESSION AUTHORIZATION 参数说明

参数	描述	取值范围
SESSION	声明这个命令只对当前会话起作用。	-
LOCALE	声明该命令只在当前事务中有效。	-
role_name	角色名。	已存在的角色名。
password	角色的密码。要求符合密码的命名规则。	-
DEFAULT	重置会话和当前用户标识符为初始认证的用户名。	-

示例

设置当前用户为paul。

```
SET SESSION AUTHORIZATION paul password '{password}';
```

查看当前会话用户，当前用户。

```
SELECT SESSION_USER, CURRENT_USER;
```

重置当前用户。

```
RESET SESSION AUTHORIZATION;
```

相关参考

SET ROLE

21.13.12.6 COMMENT

功能描述

定义或修改对象的注释。

版权所有 © 华为云计算技术有限公司

注意事项

- 每个对象只存储一条注释，因此要修改对象的注释，对同一个对象发出一条新的COMMENT命令即可。要删除注释，在文本字符串的位置写入NULL。删除对象时，注释将自动删除。
- 目前查看注释没有相关安全机制：连接到数据库的任何用户都可以查看该数据库中对象的所有注释。共享对象（比如数据库、角色、表空间）的注释是全局存储的，连接到集群中任何数据库的任何用户都可以查看共享对象的所有注释。因此，不要在注释里添加与安全有关的敏感信息。
- 对大多数对象，只有对象的所有者才能设置注释。角色没有所有者，所以COMMENT ON ROLE命令仅可以由系统管理员对系统管理员角色执行，或者具有CREATE ROLE权限的角色也可以为非系统管理员角色添加注释。系统管理员可以对所有对象添加注释。

语法格式

```
COMMENT ON
{
  AGGREGATE agg_name (agg_type [, ...] ) |
  CAST (source_type AS target_type) |
  COLLATION object_name |
  COLUMN { table_name.column_name | view_name.column_name } |
  CONSTRAINT constraint_name ON table_name |
  CONVERSION object_name |
  DATABASE object_name |
  DOMAIN object_name |
  EXTENSION object_name |
  FOREIGN DATA WRAPPER object_name |
  FOREIGN TABLE object_name |
  FUNCTION function_name ( [ {[ argmode ] [ argname ] argtype} [, ...] ] ) |
  INDEX object_name |
  LARGE OBJECT large_object_oid |
  OPERATOR operator_name (left_type, right_type) |
  OPERATOR CLASS object_name USING index_method |
  OPERATOR FAMILY object_name USING index_method |
  [ PROCEDURAL ] LANGUAGE object_name |
  ROLE object_name |
  RULE rule_name ON table_name |
  SCHEMA object_name |
  SERVER object_name |
  TABLE object_name |
  TABLESPACE object_name |
  TEXT SEARCH CONFIGURATION object_name |
  TEXT SEARCH DICTIONARY object_name |
  TEXT SEARCH PARSER object_name |
  TEXT SEARCH TEMPLATE object_name |
  TYPE object_name |
  VIEW object_name
}
   IS 'text';
```

参数说明

表 21-268 COMMENT 参数说明

参数	描述	取值范围
agg_name	需要注释的聚集函数的名称。	-
agg_type	聚集函数参数的类型。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
source_type	类型转换的源数据类型。	-
target_type	类型转换的目标数据类型。	-
object_name	对象的名称。	-
table_name.column_name /view_name.column_name	定义/修改注释的列名称。前缀可加表名称或者视图名称。	-
constraint_name	定义/修改注释的表约束的名称。	已存在的表约束名称。
table_name	表名称。	已存在的表名称。
function_name	定义/修改注释的函数名称。	已存在的函数名称。
argmode/argname/ argtype	函数参数的模式、名称、类型。	-
large_object_oid	定义/修改注释的大对象的OID。	-
operator_name	操作符名称。	-
left_type/right_type	操作参数的数据类型（可以用模式指定）。当前置或者后置操作符不存在时，可以增加NONE选项。	-
text	新注释。	字符串。

示例

创建示例表CUSTOMER，并在表CUSTOMER的customer.c_customer_sk列上添加注释。

```
CREATE TABLE IF NOT EXISTS CUSTOMER
(
    C_CUSTKEY     BIGINT NOT NULL CONSTRAINT C_CUSTKEY_pk PRIMARY KEY  ,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT          ,
    C_PHONE       CHAR(15)     ,
    C_ACCTBAL     DECIMAL(15,2)
)
DISTRIBUTE BY HASH(C_CUSTKEY);

COMMENT ON COLUMN customer.c_customer_sk IS 'Primary key of customer demographics table.';
```

创建视图customer_details_view并给视图添加注释。

```
CREATE OR REPLACE VIEW customer_details_view AS SELECT * FROM CUSTOMER;
COMMENT ON VIEW customer_details_view_v2 IS 'View of customer detail';
```

给表customer添加注释。

 版权所有 © 华为云计算技术有限公司

```
COMMENT ON TABLE customer IS 'This is my table';
```

21.13.13 发布和订阅

21.13.13.1 CREATE PUBLICATION

功能描述

向当前数据库添加一个新的发布，发布的名称必须与当前数据库中任何现有发布的名称不同。发布本质上是通过逻辑复制将一组表的数据变更进行复制。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 如果既没有指定FOR TABLE，也没有指定FOR ALL TABLES， 那么这个发布就是以一组空表开始的，可以在后续添加表。
- 创建发布不会开始复制。它只为未来的订阅者定义一个分组和过滤逻辑。 要创建一个发布，调用者必须拥有当前数据库的CREATE权限。
- 要将表添加到发布中，调用者必须拥有该表的所有权。FOR ALL TABLES和FOR ALL TABLES IN SCHEMA子句要求调用者具有系统管理员权限。
- 对一个待发布表，不能同时通过FOR TABLE和FOR ALL TABLES IN SCHEMA方式添加到同一个发布中。

语法格式

```
CREATE PUBLICATION name
    [ FOR ALL TABLES
      | FOR publication_object [, ... ] ]
    [ WITH ( publication_parameter [=value] [, ... ] ) ];
```

其中发布对象publication_object为：

```
TABLE table_name [, ...]
| ALL TABLES IN SCHEMA schema_name [, ... ]
```

参数说明

表 21-269 CREATE PUBLICATION 参数说明

参数	描述	取值范围
name	新发布的名称。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
FOR ALL TABLES	将发布标记为复制数据库中所有细粒度容灾主表的更改，包括在将来创建的表。	-
FOR TABLE	指定要添加到发布的表的列表。只有细粒度容灾主表才能成为发布的一部分。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
table_name	要添加到发布的表的名字，可以带模式名。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
FOR ALL TABLES IN SCHEMA	将发布标记为复制指定模式列表中所有细粒度容灾主表的更改，包括在将来创建的表。	-
schema_name	要添加到发布的模式的名字。	字符串，要符合标识符的命名规范。参见**标识符命名规范**。
WITH (publication_parameter [=value] [, ...])	该子句指定发布的可选参数，当前支持右列的参数。	publish：这个参数决定了哪些DML操作将由新的发布给订阅者。 取值范围：字符串，用逗号分隔的操作列表。允许的操作是insert， update，delete和truncate。 默认发布所有动作，所以这个选项的默认值是：'insert, update, delete, truncate'。

示例

- 创建示例表tpcds.ship_mode_t1。

```
CREATE TABLE tpcds.ship_mode_t1
(
    SM_SHIP_MODE_SK           INTEGER               NOT NULL,
    SM_SHIP_MODE_ID           CHAR(16)              NOT NULL,
    SM_TYPE                   CHAR(30)                      ,
    SM_CODE                   CHAR(10)                      ,
    SM_CARRIER                CHAR(20)                      ,
    SM_CONTRACT               CHAR(20)
) WITH (ORIENTATION = COLUMN,enable_disaster_cstore='on')
DISTRIBUTE BY HASH(SM_SHIP_MODE_SK);
```

创建示例表tpcds.customer_address_p1。

```
CREATE TABLE tpcds.customer_address_p1
(
    CA_ADDRESS_SK             INTEGER               NOT NULL,
    CA_ADDRESS_ID             CHAR(16)              NOT NULL,
    CA_STREET_NUMBER          CHAR(10)                      ,
    CA_STREET_NAME            VARCHAR(60)                   ,
    CA_STREET_TYPE            CHAR(15)                      ,
    CA_SUITE_NUMBER           CHAR(10)                      ,
    CA_CITY                   VARCHAR(60)                   ,
    CA_COUNTY                 VARCHAR(30)                   ,
    CA_STATE                  CHAR(2)                       ,
    CA_ZIP                    CHAR(10)                      ,
    CA_COUNTRY                VARCHAR(20)                   ,
    CA_GMT_OFFSET             DECIMAL(5,2)                  ,
    CA_LOCATION_TYPE          CHAR(20)
) WITH (ORIENTATION = COLUMN,enable_disaster_cstore='on')
DISTRIBUTE BY HASH(CA_ADDRESS_SK);
```

创建示例模式myschema1。

```
CREATE SCHEMA myschema1;
```

 版权所有 © 华为云计算技术有限公司

创建示例模式myschema2。

```
CREATE SCHEMA myschema2;
```

- 创建发布，发布两个表和两个模式中所有更改。
  ```
  CREATE PUBLICATION mypublication FOR TABLE users, departments, ALL TABLES IN SCHEMA myschema1, myschema2;
  ```
- 创建一个发布，发布所有表中的所有更改。
  ```
  CREATE PUBLICATION alltables FOR ALL TABLES;
  ```

相关链接

ALTER PUBLICATION，**DROP PUBLICATION**

21.13.13.2 ALTER PUBLICATION

功能描述

更改发布的属性。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 发布的所有者和系统管理员才能执行ALTER PUBLICATION。
- 要改变发布的所有者，用户必须是新所属角色的直接或者间接的成员，并且此角色必须有当前数据库的CREATE权限。
- FOR ALL TABLES发布的新所有者必须是系统管理员。
- 管理员用户可以更改任何发布的所属关系。

语法格式

- 向发布中添加一个或多个发布对象。
  ```
  ALTER PUBLICATION name ADD publication_object [, ...]
  ```
- 从发布中删除一个或多个发布对象。
  ```
  ALTER PUBLICATION name DROP publication_object [, ...]
  ```
- 用指定的发布对象替换当前发布对象。
  ```
  ALTER PUBLICATION name SET publication_object [, ...]
  ```
- 设置发布的参数，未提及的参数保留其原有值。
  ```
  ALTER PUBLICATION name SET ( publication_parameter [= value] [, ... ] )
  ```
- 更改发布的所有者。
  ```
  ALTER PUBLICATION name OWNER TO new_owner
  ```
- 重命名发布。
  ```
  ALTER PUBLICATION name RENAME TO new_name
  ```

其中发布对象publication_object为：

```
TABLE table_name [, ...]
| ALL TABLES IN SCHEMA schema_name [, ... ]
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-270 ALTER PUBLICATION 参数说明

参数	描述	取值范围
name	需要修改的发布的名称。	-
table_name	表名称。	已存在的表名。
schema_name	模式的名称。	已存在的模式名。
SET (publication_parameter [= value] [, ...])	该子句修改最初由CREATE PUBLICATION设置的发布参数。	详细的参数说明请参考CREATE PUBLICATION的**参数说明**。
new_owner	发布的新所有者。	已存在的用户名。
new_name	发布的新名称。	字符串，需符合**标识符命名规范**。

示例

- 向发布中添加表。
  ```
  ALTER PUBLICATION mypublication ADD TABLE mydata2;
  ```
- 从发布中删除模式。
  ```
  ALTER PUBLICATION mypublication DROP ALL TABLES IN SCHEMA myschema1;
  ```
- 重新设置发布的发布对象。
  ```
  ALTER PUBLICATION mypublication SET TABLE mydata2, ALL TABLES IN SCHEMA myschema2;
  ```
- 修改发布的所有者。
  ```
  ALTER PUBLICATION mypublication OWNER TO user1;
  ```
- 修改发布名称。
  ```
  ALTER PUBLICATION mypublication RENAME TO mypublication1;
  ```

相关链接

CREATE PUBLICATION，**DROP PUBLICATION**

21.13.13.3 DROP PUBLICATION

功能描述

删除一个发布。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 只有发布的所有者可以执行DROP PUBLICATION操作，系统管理员默认拥有此权限。

语法格式

```
DROP PUBLICATION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-271 DROP PUBLICATION 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的发布不存在，不会抛出一个错误，而会发出一个通知，告知发布不存在。	-
name	要删除的发布名字。	已存在的发布。
CASCADE | RESTRICT	当前这些关键词没有任何作用，因为发布没有依赖关系。	-

示例

删除发布。

```
DROP PUBLICATION mypublication;
```

相关链接

ALTER PUBLICATION，**CREATE PUBLICATION**

21.13.13.4 CREATE SUBSCRIPTION

功能描述

为当前数据库添加一个新的订阅。订阅名称必须与数据库中任何现有的订阅不同，订阅表示到发布者的复制连接。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 只有系统管理员才可以创建订阅。

语法格式

```
CREATE SUBSCRIPTION name
    CONNECTION 'conninfo'
    PUBLICATION publication_name [, ...]
    [ WITH ( subscription_parameter [= value] [, ... ] ) ]
```

参数说明

表 21-272 CREATE SUBSCRIPTION 参数说明

参数	描述	取值范围
name	新订阅的名称。	字符串，需符合**标识符命名规范**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
conninfo	连接发布端的字符串。 如host=1.1.1.1,2.2.2.2 port=10000,20000 dbname=postgres user=repusr1 password=*password*。	• **host** 发布端IP地址，可以同时指定发布端主机和备机的IP地址，如果同时指定了多个IP，以英文逗号分隔。 • **port** 发布端端口，此处的端口不能使用主端口，而应该使用主端口+1端口，否则会与线程池冲突。可以同时指定发布端主机和备机的端口，如果同时指定了多个端口，以英文逗号分隔。 **注意** host和port的数量要一致，并且要一一对应。 • **dbname** 发布所在的数据库。 • **user和password** 用于连接发布端且具有系统管理员权限(SYSADMIN)或者运维管理员权限(OPRADMIN)的用户名和密码。
publication_name	要订阅的发布端的发布名称，一个订阅可以对应多个发布。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
WITH	该子句指定订阅的可选参数，支持的参数如右列所示。	• **enabled** 指定订阅是否应该主动复制，或者是否应该只是设置，但尚未启动。 取值范围：true，false。 默认值：true。 • **create_slot** 指定该命令是否要在发布者上创建复制槽。 取值范围：true，false。 默认值：true。 • **slot_name** 要使用的复制槽的名称。 取值范围：字符串。 默认值：默认使用订阅名称作为复制槽的名称。

示例

创建一个到远程服务器的订阅，复制发布mypublication中的表。

```
CREATE SUBSCRIPTION mysub
    CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb password=********'
    PUBLICATION mypublication;
```

相关链接

ALTER SUBSCRIPTION， **DROP SUBSCRIPTION**

21.13.13.5 ALTER SUBSCRIPTION

功能描述

修改订阅的属性。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 订阅的所有者才能执行ALTER SUBSCRIPTION，并且新的所有者必须是系统管理员。

版权所有 © 华为云计算技术有限公司

语法格式

- 更新订阅的连接信息。
  ```
  ALTER SUBSCRIPTION name CONNECTION 'conninfo';
  ```
- 更新订阅的发布端的发布名称。
  ```
  ALTER SUBSCRIPTION name SET PUBLICATION publication_name [, ...];
  ```
- 激活订阅。
  ```
  ALTER SUBSCRIPTION name ENABLE;
  ```
- 禁用订阅。
  ```
  ALTER SUBSCRIPTION name DISABLE;
  ```
- 设置订阅的参数。
  ```
  ALTER SUBSCRIPTION name SET ( subscription_parameter [= value] [, ... ] );
  ```
- 更改订阅的所有者。
  ```
  ALTER SUBSCRIPTION name OWNER TO new_owner;
  ```
- 重命名订阅。
  ```
  ALTER SUBSCRIPTION name RENAME TO new_name;
  ```

参数说明

表 21-273 ALTER SUBSCRIPTION 参数说明

参数	描述	取值范围
name	需要修改的订阅名称。	已存在的订阅名。
CONNECTION 'conninfo'	该子句修改最初由CREATE SUBSCRIPTION设置的连接属性。	详细的参数说明请参考CREATE SUBSCRIPTION的**参数说明**。
publication_name	新发布的名称。	字符串，需符合**标识符命名规范**。
ENABLE	启用先前禁用的订阅，在事务结束时启动逻辑复制工作。	-
DISABLE	禁用正在运行的订阅，在事务结束时停止逻辑复制工作。	-
SET (publication_parameter [= value] [, ...])	该子句修改最初由CREATE PUBLICATION设置的发布参数。	详细的参数说明请参考CREATE SUBSCRIPTION的**参数说明**。
new_owner	订阅新所有者的用户名称。	已存在的用户名。
new_name	订阅的新名称。	字符串，需符合**标识符命名规范**。

示例

修改订阅的连接信息：

```
ALTER SUBSCRIPTION mysub CONNECTION 'host=192.168.1.51 port=5432 user=foo dbname=foodb password=xxxx';
```


版权所有 © 华为云计算技术有限公司

相关链接

CREATE SUBSCRIPTION，DROP SUBSCRIPTION

21.13.13.6 DROP SUBSCRIPTION

功能描述

删除一个订阅。

注意事项

- 该语法仅8.2.0.100及以上集群版本支持。
- 只有系统管理员才可以删除订阅。

语法格式

```
DROP SUBSCRIPTION [ IF EXISTS ] name
```

参数说明

表 21-274 DROP SUBSCRIPTION 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的订阅不存在，不会抛出一个错误，而会发出一个通知，告知订阅不存在。	-
name	要删除的订阅名字。	已存在的订阅。

示例

删除订阅。

```
DROP SUBSCRIPTION mysub;
```

相关链接

ALTER SUBSCRIPTION，CREATE SUBSCRIPTION

21.13.14 其他

21.13.14.1 CREATE SYNONYM

功能描述

创建一个同义词对象。同义词是数据库对象的别名，用于记录与其他数据库对象名间的映射关系，用户可以使用同义词访问关联的数据库对象。

版权所有 © 华为云计算技术有限公司

注意事项

- 定义同义词的用户成为其所有者。
- 若指定模式名称，则同义词在指定模式中创建。否则，在当前模式创建。
- 支持通过同义词访问的数据库对象包括：表、视图、函数和存储过程。
- 使用同义词时，用户需要具有对关联对象的相应权限。
- 支持使用同义词的DML语句包括：SELECT、INSERT、UPDATE、DELETE、EXPLAIN、CALL。
- 不支持关联函数或存储过程的CREATE SYNONYM语句出现在存储过程中，建议存储过程中使用系统表pg_synonym中已存在的同义词对象。

语法格式

```
CREATE [ OR REPLACE ] SYNONYM synonym_name
    FOR object_name;
```

参数说明

表 21-275 CREATE SYNONYM 参数说明

参数	描述	取值范围
synonym	创建的同义词名字，可以带模式名。	字符串，需符合**标识符命名规范**。
object_name	关联的对象名字，可以带模式名。 **说明** object_name可以是不存在的对象名称。	字符串，需符合**标识符命名规范**。

示例

创建模式ot。

```
CREATE SCHEMA ot;
```

创建表ot.t1及其同义词t1。

```
CREATE TABLE ot.t1(id int, name varchar2(10)) DISTRIBUTE BY hash(id);
CREATE OR REPLACE SYNONYM t1 FOR ot.t1;
```

使用同义词t1。

```
SELECT * FROM t1;
INSERT INTO t1 VALUES (1, 'ada'), (2, 'bob');
UPDATE t1 SET t1.name = 'cici' WHERE t1.id = 2;
```

创建同义词v1及其关联视图ot.v_t1。

```
CREATE SYNONYM v1 FOR ot.v_t1;
CREATE VIEW ot.v_t1 AS SELECT * FROM ot.t1;
```

使用同义词v1。

```
SELECT * FROM v1;
```


版权所有 © 华为云计算技术有限公司

创建重载函数ot.add及其同义词add。

```
CREATE OR REPLACE FUNCTION ot.add(a integer, b integer) RETURNS integer AS
$$
SELECT $1 + $2
$$
LANGUAGE sql;

CREATE OR REPLACE FUNCTION ot.add(a decimal(5,2), b decimal(5,2)) RETURNS decimal(5,2) AS
$$
SELECT $1 + $2
$$
LANGUAGE sql;

CREATE OR REPLACE SYNONYM add FOR ot.add;
```

使用同义词add。

```
SELECT add(1,2);
SELECT add(1.2,2.3);
```

创建存储过程ot.register及其同义词register。

```
CREATE PROCEDURE ot.register(n_id integer, n_name varchar2(10))
SECURITY INVOKER
AS
BEGIN
    INSERT INTO ot.t1 VALUES(n_id, n_name);
END;
/

CREATE OR REPLACE SYNONYM register FOR ot.register;
```

使用同义词register，调用存储过程。

```
CALL register(3,'mia');
```

相关链接

ALTER SYNONYM，**DROP SYNONYM**

21.13.14.2 ALTER SYNONYM

功能描述

修改SYNONYM对象的属性。

注意事项

- 目前仅支持修改SYNONYM对象的所有者。
- 只有系统管理员和同义词的所有者有权限修改SYNONYM对象的所有者信息。
- 修改者必须是新所有者的直接或间接成员，并且新的所有者必须拥有该同义词所属模式的CREATE权限。

语法格式

```
ALTER SYNONYM synonym_name
    OWNER TO new_owner;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-276 ALTER SYNONYM 参数说明

参数	描述	取值范围
synonym_name	需要修改的同义词名称，可以带模式名。	已存在的同义词名称。
OWNER TO	用于修改同义词所有者的子句。	-
new_owner	同义词的新所有者。	有效的用户名。

示例

创建同义词t1。

```
CREATE OR REPLACE SYNONYM t1 FOR ot.t1;
```

创建新用户u1。

```
CREATE USER u1 PASSWORD '{password}';
```

修改同义词t1的所有者为u1。

```
ALTER SYNONYM t1 OWNER TO u1;
```

相关链接

CREATE SYNONYM，**DROP SYNONYM**

21.13.14.3 DROP SYNONYM

功能描述

删除指定的SYNONYM对象。

注意事项

只有SYNONYM的所有者有权限执行DROP SYNONYM命令，系统管理员默认拥有此权限。

语法格式

```
DROP SYNONYM [ IF EXISTS ] synonym_name [ CASCADE | RESTRICT ];
```


版权所有 © 华为云计算技术有限公司

参数描述

表 21-277 DROP SYNONYM 参数说明

参数	描述	取值范围
IF EXISTS	如果指定的同义词不存在，则发出一个notice而不是抛出一个错误。	-
synonym_name	要删除的同义词名字，可以带模式名。	已存在的同义词名字。
CASCADE | RESTRICT	当执行删除操作时，如何处理依赖对象中的相关数据。	• CASCADE：级联删除依赖同义词的对象（比如视图）。 • RESTRICT：如果有依赖对象存在，则拒绝删除同义词。此选项为缺省值。

示例

删除同义词。

```
DROP SYNONYM t1;
```

相关链接

ALTER SYNONYM，**CREATE SYNONYM**

21.13.14.4 ALTER LARGE OBJECT

功能描述

用于修改大对象的定义。ALTER LARGE OBJECT的唯一的功能是分配一个新的所有者。

注意事项

只有大对象的所有者有权限执行此命令，系统管理员默认拥有此权限。

语法格式

```
ALTER LARGE OBJECT large_object_oid
    OWNER TO new_owner;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-278 ALTER LARGE OBJECT 参数说明

参数	描述	取值范围
large_object_oid	需要修改的大对象的OID 。	已存在的大对象名。
OWNER TO	用于修改大对象所有者的子句。	-
new_owner	大对象的新所有者。	已存在的用户名/角色名。

示例

无。

21.13.14.5 CREATE BARRIER

功能描述

创建一个新集群节点间的同步点。该同步点可用于数据恢复。

注意事项

在创建同步点之前，首先要确认集群中CN和DN的gtm_backup_barrier和enable_cbm_tracking参数都已设置为on。

语法格式

```
CREATE BARRIER [ barrier_name  ] ;
```

参数说明

barrier_name

可选参数。同步点名称。

取值范围：字符串，需符合**标识符命名规范**。

示例

创建一个同步点，不指定名称。

```
CREATE BARRIER;
```

创建一个同步点并指定其名称为barrier1。

```
CREATE BARRIER 'barrier1';
```

21.14 DML 语法

版权所有 © 华为云计算技术有限公司

21.14.1 DML 语法一览表

DML（Data Manipulation Language数据操作语言），用于对数据库表中的数据进行操作。如：插入、更新、查询、删除。

插入数据

插入数据是往数据库表中添加一条或多条记录，请参考**INSERT**。

修改数据

修改数据是修改数据库表中的一条或多条记录，请参考**UPDATE**。

查询数据

数据库查询语句SELECT是用于在数据库中检索适合条件的信息，请参考**SELECT**。

删除数据

删除表中指定条件的数据，请参考**DELETE**。

拷贝数据

DWS提供了在表和文件之间拷贝数据的语句，请参考**COPY**。

锁定表

DWS提供了多种锁模式用于控制对表中数据的并发访问，请参考**LOCK**。

调用函数

DWS提供了三个用于调用函数的语句，它们在语法结构上没有差别，请参考**CALL**。

21.14.2 CALL

功能描述

使用CALL命令可以调用已定义的函数和存储过程。

注意事项

如果自定义函数的函数名与系统函数同名，调用自定义函数时需指定schema，否则系统会优先调用系统函数。

语法格式

```
CALL [schema.] {func_name| procedure_name} ( param_expr );
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-279 CALL 参数说明

参数	描述	取值范围
schema	函数或存储过程所在的模式名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
func_name	所调用函数或存储过程的名称。	已存在的函数名称。
param_expr	参数列表可以用符号“:=”或者“=>”将参数名和参数值隔开，优点是参数可以以任意顺序排列。若参数列表中仅出现参数值，则参数值的排列顺序必须和函数或存储过程定义时的相同。 **说明** 参数可以包含入参（参数名和类型之间指定“IN”关键字）和出参（参数名和类型之间指定“OUT”关键字），使用CALL命令调用函数或存储过程时，对于非重载的函数，参数列表必须包含出参，出参可以传入一个变量或者任一常量，详见**示例**。对于重载的package函数，参数列表里可以忽略出参，忽略出参时可能会导致函数找不到。包含出参时，出参只能是常量。	已存在的函数参数名称或存储过程参数名称。

示例

创建一个函数func_add_sql，计算两个整数的和，并返回结果：

```
CREATE FUNCTION func_add_sql(num1 integer, num2 integer) RETURN integer
AS
BEGIN
RETURN num1 + num2;
END;
/
```

按参数值传递：

```
CALL func_add_sql(1, 3);
```

使用命名标记法传参：

```
CALL func_add_sql(num1 => 1,num2 => 3);
CALL func_add_sql(num2 := 2, num1 := 3);
```

删除函数：

```
DROP FUNCTION func_add_sql;
```

创建带出参的函数：

```
CREATE FUNCTION func_increment_sql(num1 IN integer, num2 IN integer, res OUT integer)
RETURN integer
AS
BEGIN
```

 版权所有 © 华为云计算技术有限公司

```
res := num1 + num2;
END;
/
```

出参传入常量：

```
CALL func_increment_sql(1,2,1);
```

出参传入变量：

```
DECLARE
res int;
BEGIN
func_increment_sql(1, 2, res);
dbms_output.put_line(res);
END;
/
```

创建重载的函数：

```
create or replace procedure package_func_overload(col int, col2 out int) package
as
declare
    col_type text;
begin
     col := 122;
         dbms_output.put_line('two out parameters ' || col2);
end;
/
create or replace procedure package_func_overload(col int, col2 out varchar)
package
as
declare
    col_type text;
begin
     col2 := '122';
         dbms_output.put_line('two varchar parameters ' || col2);
end;
/
```

函数调用：

```
call package_func_overload(1, 'test');
call package_func_overload(1, 1);
```

删除函数：

```
DROP FUNCTION func_increment_sql;
```

21.14.3 COPY

功能描述

通过COPY命令实现在表和文件之间拷贝数据。COPY FROM从一个文件拷贝数据到一个表，COPY TO把一个表的数据拷贝到一个文件。

COPY FROM/TO适用于低并发，本地小数据量导入导出的场景。

版权所有 © 华为云计算技术有限公司

须知

- 由于云上DWS是全托管服务，用户无法登录后台集群节点，所以云上禁止使用COPY语法，可采用“\COPY”命令进行操作（仅gsql客户端支持，其他客户端包括SQL编辑器均不支持“\COPY”命令）。
- 云上推荐将数据文件放到OBS上，使用OBS外表进行入库。

注意事项

- COPY只能用于表，不能用于视图。
- 对任何要插入数据的表必须有插入权限。
- 如果声明了一个字段列表，COPY将只在文件和表之间拷贝已声明字段的数据。如果表中有任何不在字段列表里的字段，COPY FROM将为这些字段插入缺省值。
- 如果声明了数据源文件，服务器必须可以访问该文件；如果指定了STDIN，数据将在客户前端和服务器之间流动，输入时，表的列与列之间使用TAB键分隔，在新的一行中以反斜杠和句点（\.）表示输入结束。
- 如果数据文件的任意行包含比预期多或者少的字段，COPY FROM将抛出一个错误。
- 数据的结束可以用一个只包含反斜杠和句点（\.）的行表示。如果从文件中读取数据，数据结束的标记是不必要的；如果在客户端应用之间拷贝数据，必须要有结束标记。
- COPY FROM中\N为空字符串，如果要输入实际数据值\N ，使用\\N。
- COPY FROM不支持在导入过程中对数据做预处理（比如说表达式运算，填充指定默认值等）。如果需要在导入过程中对数据做预处理，用户需先把数据导入到临时表中，然后执行SQL语句通过运算插入到表中，但此方法会导致I/O膨胀，降低导入性能。
- COPY FROM在遇到数据格式错误时会回滚事务，但没有足够的错误信息，不方便用户从大量的原始数据中定位错误数据。
- 支持使用COPY TO向OBS导出TEXT或CSV格式数据，但不支持使用COPY FROM从OBS导入数据。对于导出到OBS上的带utf8 BOM的CSV格式文件，如果有导入需求，建议使用OBS导入，否则可能无法正确识别BOM字段。

语法格式

- 从一个文件拷贝数据到一个表：

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | STDIN }
    [ [ USING ] DELIMITERS 'delimiters' ]
    [ WITHOUT ESCAPING ]
    [ LOG ERRORS ]
    [ LOG ERRORS data ]
    [ REJECT LIMIT 'limit' ]
    [ [ WITH ] ( option [, ...] ) ]
    | copy_option
    | FIXED FORMATTER ( { column_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy_option
[ ...] ] ];
```

说明

语法中的FIXED FORMATTER ({ column_name(offset, length) } [, ...])以及 [(option [, ...]) | copy_option [...]] 可以任意排列组合。

版权所有 © 华为云计算技术有限公司

- 把一个表的数据拷贝到一个文件：

```
COPY table_name [ ( column_name [, ...] ) ]
    TO { 'filename' | STDOUT }
    [ [ USING ] DELIMITERS 'delimiters' ]
    [ WITHOUT ESCAPING ]
    [ [ WITH ] ( option [, ...] ) ]
    | copy_option
    | FIXED FORMATTER ( { column_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy_option
[  ...] ] ];

COPY query
    TO { 'filename' | STDOUT }
    [ WITHOUT ESCAPING ]
    [ [ WITH ] ( option [, ...] ) ]
    | copy_option
    | FIXED FORMATTER ( { column_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy_option
[  ...] ] ];
```

说明

- COPY TO语法形式约束如下：

  (query)与[USING] DELIMITER不兼容，即若COPY TO的数据来自于一个query的查询结果，那么COPY TO语法不能再指定[USING] DELIMITERS语法子句。
- 对于FIXED FORMATTER语法后面跟随的copy_option是以空格进行分隔的。
- copy_option是指COPY原生的参数形式，而option是兼容外表导入的参数形式。
- 语法中的FIXED FORMATTER ({ column_name(offset, length) } [, ...])以及[(option [, ...]) | copy_option [...]] 可以任意排列组合。

参数说明

表 21-280 COPY 参数说明

参数	描述	取值范围
query	将查询结果进行拷贝。	一个必须用圆括弧包围的SELECT或VALUES命令。
table_name	表的名字（可以有模式修饰）。	已存在的表名。
column_name	可选的待拷贝字段列表。	如果没有声明字段列表，将使用所有字段。
STDIN	声明输入是来自标准输入。	-
STDOUT	声明输出打印到标准输出。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FIXED	打开字段固定长度模式。 ● 在字段固定长度模式下，不能声明DELIMITER，NULL，CSV选项。 ● 指定FIXED类型后，不能再通过option或copy_option指定BINARY、CSV、TEXT等类型。	定长格式定义如下： ● 每条记录的每个字段长度相同。 ● 长度不足的字段以空格填充，数字类型字段左对齐，字符字段右对齐。 ● 字段和字段之间没有分隔符。
[USING] DELIMITER 'delimiters'	在文件中分隔各个字段的字符串，分隔符最大长度不超过10个字节。	不允许包含\.abcdefghijklmnopqrstuvwxyz0123456789中的任何一个字符。 在文本模式下，缺省是水平制表符，在CSV模式下是一个逗号。
WITHOUT ESCAPING	在TEXT格式中，不对'\'和后面的字符进行转义。	仅支持TEXT格式。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
LOG ERRORS	若指定该参数，则开启对于COPY FROM语句中数据类型错误的容错机制，相关错误行的错误记录会记录到此库中的public.pgxc_copy_error_log表中，备后续查阅。 **须知** 此容错选项的使用限制如下： • 此容错机制仅捕捉COPY FROM过程中CN节点上数据解析过程中相关的数据类型错误（DATA_EXCEPTION），诸如CN与DN之间的网络交互错误或者是DN上的表达式转换错误等CN数据解析逻辑之外的过程无法涵盖在内。 • 在每个库第一次使用时COPY FROM容错时，请先行检查public.pgxc_copy_error_log（COPY错误表）是否存在，若不存在请调用copy_error_log_create() 函数创建；若存在，请转移此表数据并删除这张表后，调用copy_error_log_create() 函数创建。更多关于表public.pgxc_copy_error_log的字段信息，请参见**表21-60**。 • 若在指定了LOG ERRORS的COPY FROM运行时，public.pgxc_copy_error_log不存在（未创建或者已删除）或表定义不符合copy_error_log_create() 中的预设表定义，则会报错。因此请确定此COPY错误表是使用copy_error_log_create() 函数创建的，否则可能导致容错的COPY FROM语句无法正常执行。 • COPY已有的容错选项（如IGNORE_EXTRA_DATA）开启时，对应类型的错误会按照已有的方式处理而不会报出异常，因此错误表也不会有相应数据。 • 此容错机制的覆盖范围与GDS Foreign Table的容错范围相同。推荐用户通过表名列以及COPY FROM语句开始时间戳对查询结果进行过滤。错误数据处理可参考处理错误表章节。	仅支持导入（即COPY FROM）时指定。
LOG ERRORS DATA	LOG ERRORS DATA和LOG ERRORS的区别: • LOG ERRORS DATA会登记错误行数据信息（rawrecord）。 • 使用该参数时，若错误内容过于复杂可能存在写入容错表失败的风险，导致任务失败。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
REJECT LIMIT 'limit'	与LOG ERROR选项共同使用，对COPY FROM的容错机制设置数值上限，一旦此COPY FROM语句错误数据超过选项指定条数，则会按照原有机制报错。 **说明** LOG ERRORS中描述的容错机制，REJECT LIMIT的计数也是按照执行COPY FROM的CN上遇到的解析错误数量计算，而不是每个DN上的错误数量，请与GDS容错机制区别开。	正整数（1-INTMAX），'unlimited'（无最大值限制） 缺省值： • 若未指定LOG ERRORS，则会报错； • 若指定LOG ERRORS，则默认为0。
FORMATTER	在固定长度模式中（即声明了FIXED参数），定义每一个字段在数据文件中的位置。按照column(offset,length)格式定义每一列在数据文件中的位置。	• offset取值不能小于0，以字节为单位。 • length取值不能小于0，以字节为单位。 • 所有列的总长度和不能大于1GB。 • 文件中没有出现的列默认以空值代替。
OPTION { option_name ' value ' }	用于指定兼容外表的各类参数。	具体请参见**表21-281**。
COPY_OPTION { option_name ' value ' }	用于指定COPY原生的各类参数。	具体请参见**表21-282**。

版权所有 © 华为云计算技术有限公司

表 21-281 OPTION 参数详情

参数	描述	取值范围
FORMAT	数据源文件的格式。	支持CSV、TEXT、FIXED、BINARY类型的文件。 ● CSV格式的文件，可以有效处理数据列中的换行符，但对一些特殊字符处理有欠缺。 ● TEXT格式的文件，可以有效处理一些特殊字符，但无法正确处理数据列中的换行符。 ● FIXED格式的文件，适用于每条数据的数据列都比较固定的数据，长度不足的列会添加空格补齐，过长的列则会自动截断。 ● BINARY形式的选项会使得所有的数据被存储/读作二进制格式而不是文本。 这比TEXT和CSV格式的要快一些，但是一个BINARY格式文件可移植性比较差。 缺省值：TEXT
OIDS	为每行拷贝内部对象标识（oid）。 **说明** 若COPY FROM对象为query或者对于没有oid的表，指定oids标识报错。	● true/on ● false/off 缺省值：false

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
DELIMITER	指定数据文件行数据的字段分隔符。 ● 分隔符不能是\r和\n。 ● 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。 ● TEXT格式数据的分隔符不能包含：小写字母、数字和特殊字符.\。 ● 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。 ● 分隔符推荐使用多字符和不可见字符。多字符例如'$^&'；不可见字符例如0x07，0x08，0x1b等。 ● 建议TEXT格式下多字符分隔符中的字符不要完全相同，例如不建议使用delimiter '---' 。 **说明**	支持多字符分隔符，但分隔符不能超过10个字节。 缺省值： ● TEXT格式的默认分隔符是水平制表符（tab）。 ● CSV格式的默认分隔符为“,”。 ● FIXED格式没有分隔符。
NULL	用来指定数据文件中空值的表示方式。	● null值不能是\r和\n，最大为100个字符。 ● null值不能和分隔符、quote参数相同。 缺省值： ● CSV格式下默认值是一个没有引号的空字符串。 ● 在TEXT格式下默认值是\N。
HEADER	指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV，FIXED格式的文件中。 ● 在导入数据时，如果header选项为on，则数据文件中第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。 ● 在导出数据时，如果header选项为on，则需要指定fileheader。如果header为off，则导出数据文件不包含标题行。	● true/on ● false/off 缺省值：false

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
QUOTE	CSV格式文件下，指定一个数据值被引用时使用的引用字符。	缺省值：双引号。 ● quote参数不能和分隔符、null参数相同。 ● quote参数只能是单字节的字符。 ● 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。
ESCAPE	CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。	单字节字符。 缺省值：双引号
EOL 'newline_character'	指定导入导出数据文件换行符样式。	支持多字符换行符，但换行符不能超过10个字节。 常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。 ● EOL参数只能用于TEXT格式的导入导出，不支持CSV格式和FIXED格式导入。为了兼容原有EOL参数，仍然支持导出CSV格式和FIXED格式时指定EOL参数为0x0D或0x0D0A。 ● EOL参数不能和分隔符、null参数相同。 ● EOL参数不能包含：.abcdefghijklmnopqrstuvwxyz0123456789。
FORCE_QUOTE { (column_name [, ...]) | * }	在CSV COPY TO模式下，强制对每个声明的字段的所有非NULL值都使用引号包围。NULL输出不会被引号包围。	已存在的字段。
FORCE_NOT_NULL (column_name [, ...])	在CSV COPY FROM模式下，指定的字段输入不能为空。	已存在的字段。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ENCODING	指定数据文件的编码格式名称。	缺省为当前数据库编码格式。 常用的编码格式有UTF8、GBK、GB18030，GB18030有两个版本，GB18030和GB18030_2022。GB18030_2022，用于支持汉字最新的国家标准**GB 18030-2022**。
IGNORE_EXTRA_DATA	若数据源文件比外表定义列数多，是否会忽略对多出的列。该参数只在数据导入过程中使用。	• 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。 • 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。 `extra data after last expected column` 缺省值：false/off **须知** 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。
COMPATIBLE_ILLEGAL_CHARS	导入非法字符容错参数，仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。 导入非法字符容错规则如下： • 对于'\0'，容错后转换为空格； • 对于其他非法字符，容错后转换为问号； • 如果compatible_illegal_chars为true/on标识导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。	• 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。 • 参数为false/off，导入时遇到非法字符进行报错，中断导入。 缺省值：false/off。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FILL_MISSING_FIELDS	当数据加载时，若数据源文件中一行的最后一个字段缺失的处理方式。	• 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 • 参数为false/off，如果最后一个字段缺失会显示如下错误信息。 `missing data for column "tt"` 缺省值：false/off
DATE_FORMAT	导入对于DATE类型指定格式。仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法DATE格式。可参考**时间、日期处理函数和操作符**。 **须知** 对于指定为ORACLE兼容类型的数据库（建库时指定DBCOMPATIBILITY取值为“ORA”），则DATE类型内建为TIMESTAMP类型。在导入的时候，若需指定格式，可以参考下面的timestamp_format参数。
TIME_FORMAT	导入对于TIME类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法TIME格式，不支持时区。可参考**时间、日期处理函数和操作符**。
TIMESTAMP_FORMAT	导入对于TIMESTAMP类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法TIMESTAMP格式，不支持时区。可参考**时间、日期处理函数和操作符**。
SMALLDATETIME_FORMAT	导入对于SMALLDATETIME类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法SMALLDATETIME格式。可参考**时间、日期处理函数和操作符**。
SERVER	指定OBS SERVER，仅在COPY TO时生效。指定该参数时filename为OBS上的路径，表示导出到OBS。	已创建的OBS SERVER名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
BOM	标识是否对导出的CSV格式文件添加utf8 BOM字段。仅在COPY TO并且指定了有效SERVER参数时生效，仅支持导出文件为utf8编码。	● true/on ● false/off 缺省值：false
MAXROW	标识导出文件的最大行数，超出该值会生成新的文件。仅在COPY TO并且指定了有效SERVER参数时生效。当HEADER取值为true时MAXROW需要大于1。该参数需要与FILEPREFIX同时指定。	1~ 2147483647。
FILEPREFIX	指定导出文件名的前缀。仅在COPY TO并且指定了有效SERVER参数时生效。该参数需要与MAXROW同时指定。	合法的字符串，且不能以/开始或结尾。
PRESERVE_BLANKS	用于定长导入时，控制每列数据拖尾的空白字符（包括空格，\t，\v，\f）是否保留。该参数仅8.2.0.100及以上集群版本支持。	● 参数为true/on，表示针对BYTEAOID、CHAROID、NAMEOID、TEXTOID、BPCHAROID、VARCHAROID、NVARCHAR2OID、CSTRINGOID对应类型的列数据，保留拖尾\t，不进行修剪。 ● 参数为false/off，表示针对所有数据类型的数据拖尾的空白字符（包括空格，\t，\v，\f）进行修剪。 缺省值为false/off。

表 21-282 COPY_OPTION 参数详情

参数	描述	取值范围
OIDS	为每行拷贝内部对象标识（oid）。 若COPY FROM对象为query或者对于没有oid的表，指定oids标识报错。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
NULL null_string	用来指定数据文件中空值的表示。 **须知** 在使用COPY FROM的时候，任何匹配这个字符串的字符串将被存储为NULL值，所以应该确保指定的字符串和COPY TO相同。	• null值不能是\r和\n，最大为100个字符。 • null值不能和分隔符、quote参数相同。 缺省值： • 在TEXT格式下默认值是\N。 • CSV格式下默认值是一个没有引号的空字符串。
HEADER	指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV，FIXED格式的文件中。	• 导入数据时，如果header选项为on，则数据文件中第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。 • 导出数据时，如果header选项为on，则需要指定fileheader。如果header为off，则导出数据文件不包含标题行。
FILEHEADER	导出数据时用于定义标题行的文件，一般用来描述每一列的数据信息。 **须知** • 仅在header为on或true的情况下有效。 • fileheader指定的是绝对路径。 • 该文件只能包含一行标题信息，并以换行符结尾，多余的行将被丢弃（标题信息不能包含换行符）。 • 该文件包括换行符在内长度不超过1M。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FREEZE	将COPY加载的数据行设置为已经被frozen，就等同于这些数据行执行过VACUUM FREEZE。 该参数为一个初始数据加载的性能选项。仅当以下三个条件同时满足时，数据行才会被frozen： ● 在同一事务中create或truncate这张表之后执行COPY。 ● 当前事务中没有打开的游标。 ● 当前事务中没有原有的快照。 **须知** COPY完成后，所有其他会话将会立刻识别到这些数据。但这违反了MVCC可见性的一般原则，会导致潜在的风险。	-
FORCE NOT NULL column_name [, ...]	在CSV COPY FROM模式下，指定的字段输入不能为空。	已存在的字段。
FORCE QUOTE { column_name [, ...] | * }	在CSV COPY TO模式下，强制在每个声明的字段周围对所有非NULL值都使用引号包围。NULL输出不会被引号包围。	已存在的字段。
BINARY	使用二进制格式存储和读取，而不是以文本的方式。在二进制模式下，不能声明DELIMITER，NULL，CSV选项。指定BINARY类型后，不能再通过option或copy_option指定CSV、FIXED、TEXT等类型。	-
CSV	打开逗号分隔变量（CSV）模式。指定CSV类型后，不能再通过option或copy_option指定BINARY、FIXED、TEXT等类型。	-
QUOTE [AS] 'quote_character'	CSV格式文件下的引号字符。 ● quote参数不能和分隔符、null参数相同。 ● quote参数只能是单字节的字符。 ● 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。	缺省值：双引号。
ESCAPE [AS] 'escape_character'	CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。	缺省值为双引号。当与quote值相同时，会被替换为'\0'。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
EOL 'newline_character'	指定导入导出数据文件换行符样式。	支持多字符换行符，但换行符不能超过10个字节。 常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。 ● EOL参数只能用于TEXT格式的导入导出，不支持CSV格式和FIXED格式。为了兼容原有EOL参数，仍然支持导出CSV格式和FIXED格式时指定EOL参数为0x0D或0x0D0A。 ● EOL参数不能和分隔符、null参数相同。 ● EOL参数不能包含：.abcdefghijklmnopqrstuvwxyz0123456789。
ENCODING 'encoding_name'	指定文件编码格式名称。	有效的编码格式。 缺省值：当前编码格式。
IGNORE_EXTRA_DATA	指定当数据源文件比外表定义列数多时，忽略行尾多出来的列。该参数只在数据导入过程中使用。 若不使用该参数，在数据源文件比外表定义列数多，会显示如下错误信息。 `extra data after last expected column`	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
COMPATIBLE_ILLEGAL_CHARS	指定导入时对非法字符进行容错处理，非法字符转换后入库。不报错，不中断导入。该参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。仅对COPY FROM导入有效。 若不使用该参数，导入时遇到非法字符进行报错，中断导入。 导入非法字符容错规则如下： • 对于'\0'，容错后转换为空格； • 对于其他非法字符，容错后转换为问号； • 若compatible_illegal_chars为true/on标识，导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。	-
FILL_MISSING_FIELDS	当数据加载时，若数据源文件中一行的最后一个字段缺失的处理方式。 **须知** 目前COPY指定此Option实际不会生效，即不会有相应的容错处理效果（不生效）。需要额外注意的是，打开此选项会导致解析器在CN数据解析阶段（即COPY错误表容错的涵盖范围）忽略此数据问题，而到DN重新报错，从而使得COPY错误表（打开LOG ERRORS REJECT LIMIT）在此选项打开的情况下无法成功捕获这类少列的数据异常。因此请不要指定此选项。	• true/on • false/off 缺省值：false/off。
DATE_FORMAT 'date_format_string'	导入对于DATE类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。 **说明** 对于指定为ORACLE兼容类型的数据库（建库时指定DBCOMPATIBILITY取值为“ORA”），则DATE类型内建为TIMESTAMP类型。在导入的时候，若需指定格式，可以参考下面的timestamp_format参数。	合法DATE格式。可参考**时间、日期处理函数和操作符**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
TIME_FORMAT 'time_format_string'	导入对于TIME类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法TIME格式，不支持时区。可参考**时间、日期处理函数和操作符**。
TIMESTAMP_FORMAT 'timestamp_format_string'	导入对于TIMESTAMP类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法TIMESTAMP格式，不支持时区。可参考**时间、日期处理函数和操作符**。
SMALLDATETIME_FORMAT 'smalldatetime_format_string'	导入对于SMALLDATETIME类型指定格式。此参数仅对COPY FROM导入有效。 此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。	合法SMALLDATETIME格式。可参考**时间、日期处理函数和操作符**。
PRESERVE_BLANKS	用于定长导入时，控制每列数据拖尾的空白字符（包括空格，\t，\v，\f）是否保留。该参数仅8.2.0.100及以上集群版本支持。	• 参数为true/on，表示针对BYTEAOID、CHAROID、NAMEOID、TEXTOID、BPCHAROID、VARCHAROID、NVARCHAR2OID、CSTRINGOID对应类型的列数据，保留拖尾\t，不进行修剪。 • 参数为false/off，表示针对所有数据类型的数据拖尾的空白字符（包括空格，\t，\v，\f）进行修剪。 缺省值为false/off。

COPY FROM能够识别的特殊反斜杠序列如下所示。

表 21-283 特殊反斜杠序列

反斜杠序列	含义
\b	反斜杠 （ASCII 8）
\f	换页（ASCII 12）

版权所有 © 华为云计算技术有限公司

反斜杠序列	含义
\n	换行符（ASCII 10）
\r	回车符（ASCII 13）
\t	水平制表符（ASCII 9）
\v	垂直制表符（ASCII 11）
\digits	反斜杠后面跟着一到三个八进制数，表示ASCII值为该数的字符。
\xdigits	反斜杠x后面跟着一个或两个十六进制位声明指定数值编码的字符。

示例（仅 gsql 客户端支持，其他客户端不支持）

将tpcds.ship_mode中的数据拷贝到/home/omm/ds_ship_mode.dat文件中：

```
\COPY tpcds.ship_mode TO '/home/omm/ds_ship_mode.dat';
```

将tpcds.ship_mode中的查询结果拷贝到/home/omm/ds_ship_mode.dat文件中：

```
\COPY (SELECT * FROM tpcds.ship_mode WHERE a <> null) TO '/home/omm/ds_ship_mode.dat';
```

将tpcds.ship_mode输出到stdout：

```
\COPY tpcds.ship_mode TO stdout;
```

创建tpcds.ship_mode_t1表：

```
CREATE TABLE tpcds.ship_mode_t1
(
    SM_SHIP_MODE_SK           INTEGER               NOT NULL,
    SM_SHIP_MODE_ID           CHAR(16)              NOT NULL,
    SM_TYPE                   CHAR(30)                      ,
    SM_CODE                   CHAR(10)                      ,
    SM_CARRIER                CHAR(20)                      ,
    SM_CONTRACT               CHAR(20)
)
WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE)
DISTRIBUTE BY HASH(SM_SHIP_MODE_SK );
```

从stdin拷贝数据到表tpcds.ship_mode_t1：

```
\COPY tpcds.ship_mode_t1 FROM stdin;
```

从/home/omm/ds_ship_mode.dat文件拷贝数据到表tpcds.ship_mode_t1：

```
\COPY tpcds.ship_mode_t1 FROM '/home/omm/ds_ship_mode.dat';
```

从/home/omm/ds_ship_mode.dat文件拷贝数据到表tpcds.ship_mode_t1，使用参数如下：导入格式为TEXT（format 'text'），分隔符为'\t'（delimiter E'\t'），忽略多余列（ignore_extra_data 'true'），不指定转义（noescaping 'true'）：

```
\COPY tpcds.ship_mode_t1 FROM '/home/omm/ds_ship_mode.dat' WITH(format 'text', delimiter E'\t',
ignore_extra_data 'true', noescaping 'true');
```

从/home/omm/ds_ship_mode.dat文件拷贝数据到表tpcds.ship_mode_t1，使用参数如下：导入格式为FIXED（FIXED），指定定长格式（FORMATTER(SM_SHIP_MODE_SK(0, 2), SM_SHIP_MODE_ID(2,16),

版权所有 © 华为云计算技术有限公司

SM_TYPE(18,30), SM_CODE(50,10), SM_CARRIER(61,20), SM_CONTRACT(82,20))），忽略多余列（ignore_extra_data），有数据头（header）：

```
\COPY tpcds.ship_mode_t1 FROM '/home/omm/ds_ship_mode.dat' FIXED FORMATTER(SM_SHIP_MODE_SK(0, 2), SM_SHIP_MODE_ID(2,16), SM_TYPE(18,30), SM_CODE(50,10), SM_CARRIER(61,20), SM_CONTRACT(82,20)) header ignore_extra_data;
```

从/home/omm/ds_ship_mode.dat文件拷贝数据到表tpcds.ship_mode_t1，使用参数如下：导入格式为FIXED（FIXED），指定定长格式（FORMATTER(SM_SHIP_MODE_SK(0, 2), SM_SHIP_MODE_ID(2,16), SM_TYPE(18,30), SM_CODE(50,10), SM_CARRIER(61,20), SM_CONTRACT(82,20))），忽略多余列（ignore_extra_data），有数据头（header），保留末尾\t字符：

```
\COPY tpcds.ship_mode_t1 FROM '/home/omm/ds_ship_mode.dat' (FORMAT 'fixed', FORMATTER (SM_SHIP_MODE_SK(0,2), SM_SHIP_MODE_ID(2,16), SM_TYPE(18,30), SM_CODE(50,10), SM_CARRIER(61,20), SM_CONTRACT(82,20)), PRESERVE_BLANKS‘true', HEADER 'true', IGNORE_EXTRA_DATA 'true');
```

将tpcds.ship_mode_t1导出为OBS的 '/bucket/path/'路径上的TEXT格式文件ds_ship_mode.dat。需要指定包含OBS访问信息的“server”option参数：

```
COPY tpcds.ship_mode_t1 TO '/bucket/path/ds_ship_mode.dat' WITH (format 'text', encoding 'utf8', server 'obs_server');
```

将tpcds.ship_mode_t1导出为OBS的 '/bucket/path/'路径上的CSV格式文件。需要指定包含OBS访问信息的“server”option参数。其中文件包含标题行，包含BOM头，单文件最大行数1000行（超出1000行生成新的文件），自定义文件名前缀为“justprefix”：

```
COPY (select * from tpcds.ship_mode_t1 where SM_SHIP_MODE_SK=1060) TO '/bucket/path/' WITH (format 'csv', header 'on', encoding 'utf8', server 'obs_server', bom 'on', maxrow '1000', fileprefix 'justprefix');
```

删除tpcds.ship_mode_t1：

```
DROP TABLE tpcds.ship_mode_t1;
```

21.14.4 DELETE

功能描述

从指定的表里删除满足WHERE子句的行。如果WHERE子句不存在，将删除表中所有行，结果只保留表结构。

注意事项

- 要删除表中的数据，用户必须对它有DELETE权限。同样也必须有USING子句引用的表以及condition上读取的表的SELECT权限。
- 对于复制表，仅支持两种场景下的DELETE操作：
  - 有主键约束的场景。
  - 执行计划能下推的场景。
- 对于列存表，暂时不支持RETURNING子句。

 版权所有 © 华为云计算技术有限公司

警告

- 避免使用UPDATE/DELETE大批量刷新和删除数据，考虑使用TRUNCATE PARTITION/DROP PARTITION代替。
- 更多开发设计规范参见《数据仓库服务开发指南》的“DWS总体开发设计规范”章节。

语法格式

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
    DELETE [/*+ plan_hint */] FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ PARTITION ( partition_name ) | PARTITION FOR ( partition_key_value [, ...] ) ]
    [ USING using_list ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING { * | { output_expr [ [ AS ] output_name ] } [, ...] } ];
```

参数说明

表 21-284 DELETE 参数说明

参数	描述	取值范围
WITH [RECURSIVE] with_query [, ...]	用于声明一个或多个可以在主查询中通过名字引用的子查询，相当于临时表。 如果声明了RECURSIVE，则允许SELECT子查询通过名字引用它自己。 其中with_query的详细格式为： `with_query_name [ ( column_name [, ...] ) ] AS ( {select | values | insert | update | delete} )` – with_query_name指定子查询生成的结果集名字，在查询中可使用该名称访问子查询的结果集。 – column_name指定子查询结果集中显示的列名。 – 每个子查询可以是SELECT，VALUES，INSERT，UPDATE或DELETE语句。	-
plan_hint子句	以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见《数据仓库服务开发指南》中的“DWS性能调优>SQL调优>使用Plan Hint进行调优”章节。	-
ONLY	如果指定ONLY，则只有指定表被删除；如果没有指定ONLY，则指定表及其所有继承表都会被删除。 该参数仅为了兼容PG保留，DWS不支持继承表。	-
table_name	目标表的名字（可以有模式修饰）。	已存在的表名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
alias	目标表的别名。	字符串，符合标识符命名规范。详情请参见**标识符命名规范**。
partition_name	分区名。仅8.2.1及以上集群版本支持该选项。	已存在的分区名。
partition_key_value	分区键值。 通过PARTITION FOR (partition_key_value [, ...])子句指定的这一组值，可以唯一确定一个分区。	需要进行重命名分区的分区键取值范围。
using_list	using子句。	-
condition	一个返回boolean值的表达式，用于判断哪些行需要被删除。	-
WHERE CURRENT OF cursor_name	当前不支持，仅保留语法接口。	-
output_expr	DELETE命令删除行之后计算输出结果的表达式。该表达式可以使用表的任意字段。可以使用*返回被删除行的所有字段。	-
output_name	一个字段的输出名称。	字符串，符合标识符命名规范。详情请参见**标识符命名规范**。

示例

创建表tpcds.customer_address_bak：

```
CREATE TABLE tpcds.customer_address_bak AS TABLE tpcds.customer_address;
```

删除tpcds.customer_address_bak中ca_address_sk小于14888的职员：

```
DELETE FROM tpcds.customer_address_bak WHERE ca_address_sk < 14888;
```

删除tpcds.customer_address_bak中ca_address_sk为14891，14893和14895的职员：

```
DELETE FROM tpcds.customer_address_bak WHERE ca_address_sk in (14891,14893,14895);
```

删除tpcds.customer_address_bak中所有数据：

```
DELETE FROM tpcds.customer_address_bak;
```

先通过子查询（删除行存表tpcds.warehouse_t30）得到一张临时表temp_t，然后查询临时表temp_t中的所有数据：

```
WITH temp_t AS (DELETE FROM tpcds.warehouse_t30 RETURNING *) SELECT * FROM temp_t ORDER BY 1;
```

删除分区表test_range_row的指定分区p1：

```
CREATE TABLE test_range_row(a int, d int)
DISTRIBUTE BY hash(a) PARTITION BY RANGE(d)
```


版权所有 © 华为云计算技术有限公司

```
(
   PARTITION p1 values LESS THAN (60),
   PARTITION p2 values LESS THAN (75),
   PARTITION p3 values LESS THAN (90),
   PARTITION p4 VALUES LESS THAN (maxvalue)
);
INSERT OVERWRITE INTO test_range_row PARTITION(p1) VALUES(55,51);
INSERT OVERWRITE INTO test_range_row PARTITION(p3) VALUES(85,80);

DELETE FROM test_range_row PARTITION(p1);
```

21.14.5 EXPLAIN

功能描述

EXPLAIN命令输出SQL语句的执行计划，通过分析执行计划来定位慢SQL，优化SQL语句。

执行计划将显示SQL语句所引用的表采用的扫描方式，如：简单的顺序扫描、索引扫描等。如果引用了多个表，执行计划还会显示使用的JOIN算法。执行计划的最关键部分是语句的预计执行开销，即计划生成器估算执行该语句将花费多长的时间。

指定ANALYZE选项（ EXPLAIN ANALYZE ），不仅显示执行计划，还会实际执行查询，并提供每个步骤的实际执行时间。这有助于验证执行计划的准确性，并评估实际性能。

语法格式

- 显示SQL语句的执行计划，支持多种选项，对选项顺序无要求：
  ```
  EXPLAIN [ (  option  [, ...] ) ] statement;
  ```
  其中选项option子句的语法为：
  ```
  ANALYZE [ boolean ] |
     ANALYSE [ boolean ] |
     VERBOSE [ boolean ] |
     COSTS [ boolean ] |
     CPU [ boolean ] |
     DETAIL [ boolean ] |
     NODES [ boolean ] |
     NUM_NODES [ boolean ] |
     BUFFERS [ boolean ] |
     TIMING [ boolean ] |
     PLAN [ boolean ] |
     FORMAT { TEXT | XML | JSON | YAML } |
     BLOCKNAME [boolean] |
     OUTLINE [boolean]  |
     WARMUP |
     WARMUP HOT
  ```
- 显示SQL语句的执行计划，且要按顺序给出选项：
  ```
  EXPLAIN  { [  { ANALYZE  | ANALYSE  }  ] [ VERBOSE  ]  | PERFORMANCE  } statement;
  ```
- 显示复现SQL语句的执行计划所需的信息，通常用于定位问题。STATS选项必须单独使用：
  ```
  EXPLAIN ( STATS [ boolean ] ) statement;
  ```
- 显示DDL语句执行步骤的详细耗时信息。（该语法仅9.1.0及以上集群版本支持）
  ```
  EXPLAIN PERFORMANCE statement;
  ```
  支持的DDL语句有CREATE、CREATE INDEX、DROP、VACUUM FULL、ANALYZE及COPY。
- 执行预查询，并将预查询的数据缓存到本地磁盘，提升实际查询时的查询速度。（该语法仅9.1.0.200及以上集群版本支持）
  ```
  EXPLAIN WARMUP statement;
  ```

 版权所有 © 华为云计算技术有限公司

```
EXPLAIN WARMUP HOT statement;
```

参数说明

表 21-285 EXPLAIN 参数说明

参数	描述	取值范围
statement	需要分析的SQL语句。	-

ANALYZE boolean | ANALYSE boolean	指定ANALYZE选项时，会实际执行SQL，根据实际的运行结果显示统计数据，包括每个计划节点内时间总开销（毫秒为单位）和实际返回的总行数。
	如果使用EXPLAIN分析INSERT，UPDATE，DELETE，CREATE TABLE AS或EXECUTE语句，但不改动数据（执行这些语句会影响数据），可以放在一个事务中，分析完成后可以直接回滚：
	`START TRANSACTION;`
	`EXPLAIN ANALYZE ...;`
	`ROLLBACK;`
		● TRUE：显示实际运行时间和其他统计数据。
		● FALSE：不显示。
		缺省值：TRUE

VERBOSE boolean	显示查询计划的附加信息。 附加信息包括查询计划中每个节点输出的列（Output），表的SCHEMA信息，函数的模式信息，表达式中列所属表的别名，被触发的触发器名称等。	● TRUE：显示附加信息。 ● FALSE：不显示。 缺省值：TRUE
COSTS boolean	显示每个计划节点的预估总成本，以及预估行数和每行宽度。	● TRUE：显示估计总成本和宽度。 ● FALSE：不显示。 缺省值：TRUE
CPU boolean	显示CPU的使用情况的信息。	● TRUE：显示CPU的使用情况。 ● FALSE：不显示。 缺省值：TRUE
DETAIL boolean	显示DN上的信息。 **说明** 8.2.1及以上集群版本支持explain打开Detail开关时，执行计划中会显示倾斜值比对耗时。	● TRUE：打印DN的信息。 ● FALSE：不打印。 缺省值：TRUE
NODES boolean	打印query执行的节点信息。	● TRUE：打印执行的节点的信息。 ● FALSE：不打印。 缺省值：TRUE

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
NUM_NODES boolean	打印执行中的节点的个数信息。	● TRUE：打印DN个数的信息。 ● FALSE：不打印。 缺省值：TRUE
BUFFERS boolean	显示缓冲区的使用信息。 缓冲区信息包括共享块（常规表或者索引块）、本地块（临时表或者索引块）和临时块（排序或者哈希等涉及到的短期存在的数据块）的命中块数，读取块数，更新块数等。	● TRUE：显示缓冲区的使用情况。 ● FALSE：不显示。 缺省值：FALSE
TIMING boolean	显示每个计划节点的实际启动时间和总的执行时间。	● TRUE：显示启动时间和花费在输出节点上的时间信息。 ● FALSE：不显示。 缺省值：TRUE
PLAN boolean	是否将执行计划存储在plan_table中。当该选项开启时，会将执行计划存储在PLAN_TABLE中，不打印到当前屏幕，因此该选项为on时，不能与其他选项同时使用。	● ON：将执行计划存储在plan_table中，不打印到当前屏幕。执行成功返回EXPLAIN SUCCESS。 ● OFF：不存储执行计划，将执行计划打印到当前屏幕。 缺省值：ON
FORMAT	指定输出格式。 各个格式输出的内容都是相同的，其中XML | JSON | YAML更有利于您通过程序解析SQL语句的查询计划。	支持四种格式：TEXT，XML，JSON和YAML。 缺省值：TEXT
BLOCKNAME boolean	显示算子的blockname信息。 仅在explain_perf_mode取值为pretty时，打印blockname信息。	● TRUE：显示blockname信息。 ● FALSE：不显示blockname信息。 缺省值：TRUE
OUTLINE boolean	显示从计划中提取的outline信息。 仅在explain_perf_mode取值为pretty时，打印outline信息。	● TRUE：显示outline信息。 ● FALSE：不显示outline信息。 缺省值：TRUE
PERFORMANCE	使用此选项时，即打印执行中的所有相关信息。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
STATS boolean	打印复现SQL语句的执行计划所需的信息，包括对象定义、统计信息、配置参数等，通常用于定位问题。	● TRUE：显示复现SQL语句的执行计划所需的信息。 ● FALSE：不显示。 缺省值：TRUE
WARMUP	warmup将查询数据按照A1in > A1out > Am的顺序进行。该参数仅9.1.0.200及以上集群版本支持。	-
WARMUP HOT	warmup hot将查询的数据直接加入am列。该参数仅9.1.0.200及以上集群版本支持。	-

示例

创建一个表tpcds.customer_address_p1。

```
CREATE TABLE tpcds.customer_address_p1 AS TABLE tpcds.customer_address;
```

修改explain_perf_mode为normal。

```
SET explain_perf_mode=normal;
```

显示表简单查询的执行计划。

```
EXPLAIN SELECT * FROM tpcds.customer_address_p1;
                QUERY PLAN
------------------------------------------------------------------------------
 Data Node Scan on "__REMOTE_FQS_QUERY__"  (cost=0.00..0.00 rows=0 width=0)
   Node/s: All datanodes
(2 rows)
```

以JSON格式输出的执行计划（ explain_perf_mode为normal时 ）。

```
EXPLAIN(FORMAT JSON) SELECT * FROM tpcds.customer_address_p1;
                  QUERY PLAN
--------------------------------------------------
 [                                       +
   {                                     +
     "Plan": {                             +
       "Node Type": "Data Node Scan",            +
       "RemoteQuery name": "__REMOTE_FQS_QUERY__",+
       "Alias": "__REMOTE_FQS_QUERY__",          +
       "Startup Cost": 0.00,                      +
       "Total Cost": 0.00,                       +
       "Plan Rows": 0,                           +
       "Plan Width": 0,                          +
       "Nodes": "All datanodes"                       +
     }                                   +
   }                                     +
 ]
(1 row)
```

如果有一个索引，当使用一个带索引WHERE条件的查询，可能会显示一个不同的计划。

```
EXPLAIN SELECT * FROM tpcds.customer_address_p1 WHERE ca_address_sk=10000;
                              QUERY PLAN
```

 版权所有 © 华为云计算技术有限公司

```
------------------------------------------------------------------------------
 Data Node Scan on "__REMOTE_LIGHT_QUERY__"  (cost=0.00..0.00 rows=0 width=0)
   Node/s: datanode2
(2 rows)
```

以YAML格式输出的执行计划（explain_perf_mode为normal时）。

```
EXPLAIN(FORMAT YAML) SELECT * FROM tpcds.customer_address_p1 WHERE ca_address_sk=10000;
                  QUERY PLAN
-------------------------------------------------
 - Plan:                                  +
     Node Type: "Data Node Scan"              +
     RemoteQuery name: "__REMOTE_LIGHT_QUERY__"+
     Alias: "__REMOTE_LIGHT_QUERY__"          +
     Startup Cost: 0.00                       +
     Total Cost: 0.00                         +
     Plan Rows: 0                             +
     Plan Width: 0                            +
     Nodes: "datanode2"
(1 row)
```

禁止开销估计的执行计划。

```
EXPLAIN(COSTS FALSE)SELECT * FROM tpcds.customer_address_p1 WHERE ca_address_sk=10000;
               QUERY PLAN
-----------------------------------------------
 Data Node Scan on "__REMOTE_LIGHT_QUERY__"
   Node/s: datanode2
(2 rows)
```

带有聚集函数查询的执行计划。

```
EXPLAIN SELECT SUM(ca_address_sk) FROM tpcds.customer_address_p1 WHERE ca_address_sk<10000;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
 Aggregate  (cost=18.19..14.32 rows=1 width=4)
   ->  Streaming (type: GATHER)  (cost=18.19..14.32 rows=3 width=4)
         Node/s: All datanodes
         ->  Aggregate  (cost=14.19..14.20 rows=3 width=4)
               ->  Seq Scan on customer_address_p1  (cost=0.00..14.18 rows=10 width=4)
                     Filter: (ca_address_sk < 10000)
(6 rows)
```

删除表tpcds.customer_address_p1。

```
DROP TABLE tpcds.customer_address_p1;
```

对ANALYZE语句执行EXPLAIN PERFORMANCE。

```
EXPLAIN PERFORMANCE ANALYZE t2_dist_row;
                          QUERY EXEC INFO
------------------------------------------------------------------------
 lock FirstCN:
        coordinator1: actual time=0.240 loops=1
 estimate rows: actual time=[datanode3 0.000, datanode1 0.001]
        coordinator1: actual time=0.000 loops=1
        datanode1: actual time=0.001 loops=1
        datanode2: actual time=0.000 loops=1
        datanode3: actual time=0.000 loops=1
 sample rows: actual time=[datanode1 5.109, coordinator1 119.838]
        coordinator1: actual time=119.838 loops=1
        datanode1: actual time=5.109 loops=1
        datanode2: actual time=5.621 loops=1
        datanode3: actual time=5.342 loops=1
 fetch global stats:
        coordinator1: actual time=8.501 loops=1
 calc stats: actual time=[datanode3 80.794, datanode2 109.155]
        coordinator1: actual time=97.452 loops=1
        datanode1: actual time=94.375 loops=1
        datanode2: actual time=109.155 loops=1
```

 版权所有 © 华为云计算技术有限公司

```
       datanode3: actual time=80.794 loops=1
 calc column stats: actual time=[datanode2 0.938, datanode3 9.811]
       coordinator1: actual time=5.162 loops=2
       datanode1: actual time=1.453 loops=2
       datanode2: actual time=0.938 loops=2
       datanode3: actual time=9.811 loops=2
 calc index stats: actual time=[datanode3 12.392, coordinator1 36.113]
       coordinator1: actual time=36.113 loops=1
       datanode1: actual time=15.933 loops=1
       datanode2: actual time=13.419 loops=1
       datanode3: actual time=12.392 loops=1
 calc expr stats: actual time=[datanode3 41.665, datanode2 78.442]
       coordinator1: actual time=55.608 loops=1
       datanode1: actual time=63.179 loops=1
       datanode2: actual time=78.442 loops=1
       datanode3: actual time=41.665 loops=1
 sync stats:
       coordinator1: actual time=7.906 loops=1

 General Tracks
 CN build CN connection:
       coordinator1: actual time=0.002 loops=1
 CN build DN connection:
       coordinator1: actual time=0.070 loops=1
 -> execute ddl on other CN:
       coordinator1: actual time=0.001 loops=1
 -> execute ddl on other DN:
       coordinator1: actual time=0.000 loops=1
 Query Id: 72902018968225366
 Total runtime: 242.211 ms
(48 rows)
```

显示计划的blockname信息。

```
EXPLAIN (BLOCKNAME ON) SELECT SUM(ca_address_sk) FROM tpcds.customer_address_p1 WHERE
ca_address_sk<10000;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
  1 | ->  Aggregate                             |      1 |          |      12 | 16.14
  2 |    ->  Streaming (type: GATHER)           |      2 |          |      12 | 16.14
  3 |       ->  Aggregate                       |      2 | 1MB      |      12 | 10.14
  4 |          ->  Seq Scan on customer_address_p1 |      7 | 1MB      |       4 | 10.12

 Predicate Information (identified by plan id)
 ---------------------------------------------
   4 --Seq Scan on customer_address_p1
         Filter: (ca_address_sk < 10000)

 Query Block Name / Object Alias (identified by plan id)
 -------------------------------------------------------
   1 - sel$1
   4 - sel$1 / customer_address_p1@"sel$1"

   ====== Query Summary =====
 -------------------------------
 System available mem: 4710400KB
 Query Max mem: 4710400KB
 Query estimated mem: 2048KB
(22 rows)
```

显示计划的outline信息。

```
EXPLAIN (OUTLINE ON) SELECT SUM(ca_address_sk) FROM tpcds.customer_address_p1 WHERE
ca_address_sk<10000;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
```

 版权所有 © 华为云计算技术有限公司

```
 1 | ->  Aggregate                          |      1 |         |      12 | 16.14
 2 |    ->  Streaming (type: GATHER)        |      2 |         |      12 | 16.14
 3 |       ->  Aggregate                    |      2 | 1MB     |      12 | 10.14
 4 |          ->  Seq Scan on customer_address_p1 |      7 | 1MB     |       4 | 10.12

Predicate Information (identified by plan id)
-----------------------------------------------
 4 --Seq Scan on customer_address_p1
       Filter: (ca_address_sk < 10000)

                     Outline Data
-------------------------------------------------------------------
 /*+
    begin_outline_data
     TableScan(@"sel$1" tpcds.customer_address_p1@"sel$1")
    end_outline_data
 */

 ====== Query Summary =====
------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 2048KB
(25 rows)
```

相关链接

ANALYZE | ANALYSE

21.14.6 EXPLAIN PLAN

功能描述

通过EXPLAIN PLAN命令可以将查询执行的计划信息存储于PLAN_TABLE表中。与EXPLAIN命令不同的是，EXPLAIN PLAN仅将计划信息进行存储，而不会打印到屏幕。

语法格式

```
EXPLAIN PLAN
[ SET STATEMENT_ID = string ]
FOR statement ;
```

参数说明

表 21-286 EXPLAIN PLAN 参数说明

参数	描述
PLAN	表示需要将计划信息存储于PLAN_TABLE中，存储成功将返回“EXPLAIN SUCCESS”。
STATEMENT_ID	用户可以对查询设置标签，输入的标签信息也将存储于PLAN_TABLE中。 **说明** 用户在执行EXPLAIN PLAN时，如果没有进行SET STATEMENT_ID，则默认为空值。同时，用户可输入的STATEMENT_ID最大长度为30个字节，超过长度将会产生报错。

版权所有 © 华为云计算技术有限公司

注意事项

- EXPLAIN PLAN不支持在DN上执行。
- 对于执行错误的SQL无法进行计划信息的收集。
- PLAN_TABLE中的数据是session级生命周期并且session隔离和用户隔离，用户只能看到当前session、当前用户的数据。
- PLAN_TABLE无法与GDS外表进行关联查询。
- 对于不能下推的查询，无法收集到具体的object信息，object只能收集到REMOTE_QUERY或CTE等信息。详见**示例 2**。

示例 1

使用EXPLAIN PLAN收集SQL语句的执行计划，通常包括以下步骤：

步骤1 导入TPC-H样例数据。

步骤2 执行EXPLAIN PLAN。

说明

执行EXPLAIN PLAN后会将计划信息自动存储于PLAN_TABLE中，不支持对PLAN_TABLE进行INSERT、UPDATE、ANALYZE等操作。

PLAN_TABLE详细介绍见视图PLAN_TABLE。

```
explain plan set statement_id='TPCH-Q4' for
select
o_orderpriority,
count(*) as order_count
from
orders
where
o_orderdate >= '1993-07-01'::date
and o_orderdate < '1993-07-01'::date + interval '3 month'
and exists (
select
*
from
lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority;
```

步骤3 查询PLAN_TABLE。

```
SELECT * FROM PLAN_TABLE;
```

```
 statement_id | plan_id  | id |      operation      |   options   | object_name | object_type | object_owner |                projection
--------------+----------+----+---------------------+-------------+-------------+-------------+--------------+-------------------------------------------
 TPCH-Q4      | 16781167 |  1 | ROW ADAPTER         |             |             |             | ORDERS.O_ORDERPRIORITY, (PG_CATALOG.COUNT(*))
 TPCH-Q4      | 16781167 |  2 | VECTOR SORT         |             |             |             | ORDERS.O_ORDERPRIORITY, (PG_CATALOG.COUNT(*))
 TPCH-Q4      | 16781167 |  3 | VECTOR AGGREGATE    | HASHED      |             |             | ORDERS.O_ORDERPRIORITY, PG_CATALOG.COUNT(*)
 TPCH-Q4      | 16781167 |  4 | VECTOR STREAMING    | GATHER      |             |             | ORDERS.O_ORDERPRIORITY, (COUNT(*))
 TPCH-Q4      | 16781167 |  5 | VECTOR AGGREGATE    | HASHED      |             |             | ORDERS.O_ORDERPRIORITY, COUNT(*)
 TPCH-Q4      | 16781167 |  6 | VECTOR NESTED LOOPS | SEMI        |             |             | ORDERS.O_ORDERPRIORITY
 TPCH-Q4      | 16781167 |  7 | TABLE ACCESS        | CSTORE SCAN | ORDERS      | TABLE       | TPCH         | ORDERS.O_ORDERPRIORITY, ORDERS.O_ORDERKEY
 TPCH-Q4      | 16781167 |  8 | VECTOR MATERIALIZE  |             |             |             |              | LINEITEM.L_ORDERKEY
 TPCH-Q4      | 16781167 |  9 | TABLE ACCESS        | CSTORE SCAN | LINEITEM    | TABLE       | TPCH         | LINEITEM.L_ORDERKEY
```

步骤4 清理PLAN_TABLE表中的数据。

```
DELETE FROM PLAN_TABLE WHERE xxx;
```

----结束

版权所有 © 华为云计算技术有限公司

示例 2

对于不能下推的查询，执行explain plan后plan_table中object仅收集到REMOTE_QUERY或CTE等信息。

优化器生成下发语句的计划，此时仅能收集到REMOTE_QUERY。

```
explain plan set statement_id = 'test remote query' for
select
current_user
from
customer;
```

查询PLAN_TABLE。

```
SELECT * FROM PLAN_TABLE;
```

```
   statement_id    | plan_id  | id |   operation    |  options  |   object_name    | object_type  | object_owner |  projection
-------------------+----------+----+----------------+-----------+------------------+--------------+--------------+--------------
 test remote query | 29360133 |  1 | NESTED LOOPS   | CARTESIAN |                  |              |              | 'apple'::name
 test remote query | 29360133 |  2 | DATA NODE SCAN |           | customer         | REMOTE_QUERY |              |
 test remote query | 29360133 |  3 | DATA NODE SCAN |           | customer_address | REMOTE_QUERY |              |
(3 rows)
```

21.14.7 LOCK

功能描述

LOCK TABLE获取表级锁。

当自动获取引用表的命令的锁时，DWS会始终使用限制最小的锁模式。如果用户需要一种更为严格的锁模式，可以使用LOCK命令。例如，某个应用是在Read Committed隔离级别上运行事务，并且需要保证表中的数据在事务运行期间保持稳定。为实现这个目的，则可以在查询之前对表使用SHARE锁模式进行锁定。这样将防止并发数据更改，并确保后续的查询可以读到已提交的持久化的数据。因为SHARE锁模式与任何写操作需要的ROW EXCLUSIVE模式冲突，并且LOCK TABLE name IN SHARE MODE语句将等到所有当前持有ROW EXCLUSIVE模式锁的事务提交或回滚后才能执行。因此，一旦获得该锁，就不会存在未提交的写操作，此外其他操作也只能等到该锁释放之后才能开始。

注意事项

- LOCK TABLE只能在一个事务块的内部有用，因为锁在事务结束时就会被释放。出现在任意事务块外面的LOCK TABLE都会报错。
- 如果没有声明锁模式，缺省为最严格的模式ACCESS EXCLUSIVE。
- LOCK TABLE ... IN ACCESS SHARE MODE需要在目标表上有SELECT权限。所有其他形式的LOCK需要UPDATE和/或DELETE权限。
- 没有UNLOCK TABLE命令，锁总是在事务结束时释放。
- LOCK TABLE只处理表级的锁，因此那些带“ROW”字样的锁模式都是有歧义的。这些模式名字通常可理解为用户试图在一个被锁定的表中获取行级的锁。同样，ROW EXCLUSIVE模式也是一个可共享的表级锁。注意，只要是涉及到LOCK TABLE ，所有锁模式都有相同的语意，区别仅在于规则中锁与锁之间是否冲突，规则请参见**锁等级与冲突关系**。

语法格式

```
LOCK [ TABLE ] {[ ONLY ] name [, ...]| {name [ * ]} [, ...]}
    [ IN {ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE | UPDATE EXCLUSIVE} MODE ]
    [ NOWAIT ] [LOCAL COORDINATOR ONLY];
```


版权所有 © 华为云计算技术有限公司

锁等级与冲突关系

在DWS中，锁模式用于管理并发事务对资源的访问，以防止数据不一致和丢失。

表21-287列出了DWS支持的8种常规锁模式及其锁冲突关系：

表 21-287 锁模式间的冲突关系

锁模式名称	锁等级	锁用途	冲突关系
访问共享锁（AccessShareLock）	1	SELECT语句，允许其他事务读取数据。	8
行共享锁（RowShareLock）	2	SELECT FOR UPDATE或FOR SHARE，允许其他事务读取但阻止写入。	7|8
行排他锁（RowExclusiveLock）	3	INSERT、UPDATE、DELETE语句。	5|6|7|8
共享更新排他锁（ShareUpdateExclusiveLock）	4	VACUUM（非FULL）、ANALYZE、CREATE INDEX CONCURRENTLY、COMMENT ON，允许其他事务读取但阻止写入。	4|5|6|7|8
共享锁（ShareLock）	5	CREATE INDEX（非CONCURRENTLY），允许其他事务读取但阻止写入。	3|4|6|7|8
共享行排他锁（ShareRowExclusiveLock）	6	ROW SELECT...FOR UPDATE，类似RowExclusiveLock，但允许RowShareLock。	3|4|5|6|7|8
排他锁（ExclusiveLock）	7	阻止RowShareLock或SELECT ... FOR UPDATE操作。	2|3|4|5|6|7|8
访问排他锁（AccessExclusiveLock）	8	ALTER TABLE、DROP TABLE、VACUUM FULL，完全阻止其他事务读取和写入。	1|2|3|4|5|6|7|8

参数说明

表 21-288 LOCK 参数说明

参数	描述	取值范围
name	要锁定的表的名字，可以有模式修饰。 LOCK TABLE命令中声明的表的顺序就是上锁的顺序。	已存在的表名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ONLY	如果指定ONLY，则只有指定表被锁定；如果没有指定ONLY，则指定表及其所有继承表都会被锁定。 该参数仅为了兼容PG保留，DWS不支持继承表。	-
ACCESS SHARE	ACCESS锁只允许对表进行读取，而禁止对表进行修改。所有对表进行读取而不修改的SQL语句都会自动请求这种锁。例如，SELECT命令会自动在被引用的表上请求一个这种锁。	-
ROW SHARE	ROW SHARE锁允许对表进行并发读取，禁止对表进行其他操作。 SELECT FOR UPDATE和SELECT FOR SHARE命令会自动在目标表上请求ROW SHARE锁（且所有被引用但不是FOR SHARE/FOR UPDATE的其他表上，还会自动加上ACCESS SHARE锁）。	-
ROW EXCLUSIVE	与ROW SHARE锁不同，ROW EXCLUSIVE允许并发读取表，也允许修改表中的数据。UPDATE，DELETE，INSERT命令会自动在目标表上请求这个锁（且所有被引用的其他表上还会自动加上的ACCESS SHARE锁）。通常情况下，所有会修改表数据的命令都会请求表的ROW EXCLUSIVE锁。	-
SHARE UPDATE EXCLUSIVE	这个模式保护一个表的模式不被并发修改，以及禁止在目标表上执行垃圾回收命令（VACUUM ）。 VACUUM（不带FULL选项），ANALYZE，CREATE INDEX CONCURRENTLY命令会自动请求这样的锁。	-
SHARE	SHARE锁允许并发的查询，但是禁止对表进行修改。 CREATE INDEX（不带CONCURRENTLY选项）语句会自动请求这种锁。	-
SHARE ROW EXCLUSIVE	SHARE ROW EXCLUSIVE锁禁止对表进行任何的并发修改，而且是独占锁，因此一个会话中只能获取一次。 任何SQL语句都不会自动请求这个锁模式。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
EXCLUSIVE	EXCLUSIVE锁允许对目标表进行并发查询，但是禁止任何其他操作。 这个模式只允许并发加ACCESS SHARE锁，也就是说，只有对表的读动作可以和持有这个锁模式的事务并发执行。 任何SQL语句都不会在用户表上自动请求这个锁模式。然而在某些操作的时候，会在某些系统表上请求它。	-
ACCESS EXCLUSIVE	这个模式保证其所有者（事务）是可以访问该表的唯一事务。 ALTER TABLE，DROP TABLE，TRUNCATE，REINDEX，CLUSTER，VACUUM FULL命令会自动请求这种锁。	在LOCK TABLE命令没有明确声明需要的锁模式时，为缺省锁模式。
UPDATE EXCLUSIVE	UPDATE EXCLUSIVE锁允许并发（AUTO）VACUUM和（AUTO）ANALYZE，但不允许多个（AUTO）VACUUM并发。 **说明** • 该参数仅8.2.1.200及以上集群版本支持。 • UPDATE EXCLUSIVE锁模式仅在VACUUM语法中使用。	-
NOWAIT	声明LOCK TABLE不去等待任何冲突的锁释放，如果无法立即获取该锁，该命令退出并且发出一个错误信息。 在不指定NOWAIT的情况下获取表级锁时，如果有其他互斥锁存在的话，则等待其他锁的释放。	-
LOCAL COORDINATOR ONLY	声明LOCK TABLE只在接收当前会话请求的CN上执行，而不会下发给其他CN和所有DN。该选项仅针对元数据的操作，以提高效率。 **须知** • 该参数仅8.2.0.100以上集群版本支持。 • 当前仅支持ACCESS SHARE锁模式，其他锁模式会报错。	-

示例

向一个外键表上插入数据时，在有主键的表上使用SHARE锁：

```
START TRANSACTION;

LOCK TABLE tpcds.reason IN SHARE MODE;

SELECT r_reason_desc FROM tpcds.reason WHERE r_reason_sk=5;
```


版权所有 © 华为云计算技术有限公司

```
r_reason_desc
-----------
 Parts missing
(1 row)

COMMIT;
```

在执行删除操作时对一个有主键的表进行SHARE ROW EXCLUSIVE锁：

```
CREATE TABLE tpcds.reason_t1 AS TABLE tpcds.reason;

START TRANSACTION;

LOCK TABLE tpcds.reason_t1 IN SHARE ROW EXCLUSIVE MODE;

DELETE FROM tpcds.reason_t1 WHERE r_reason_desc IN(SELECT r_reason_desc FROM tpcds.reason_t1
WHERE r_reason_sk < 6 );

DELETE FROM tpcds.reason_t1 WHERE r_reason_sk = 7;

COMMIT;
```

删除表tpcds.reason_t1：

```
DROP TABLE tpcds.reason_t1;
```

查看表定义时，对表加ACCESS SHARE锁模式、限制范围是当前CN：

```
BEGIN;
LOCK TABLE lock_test IN ACCESS SHARE MODE LOCAL COORDINATOR ONLY;
SELECT pg_get_tabledef('lock_test');
END;
```

21.14.8 MERGE INTO

功能描述

MERGE INTO语句将目标表和源表中数据根据关联条件进行匹配，若关联条件匹配时对目标表进行UPDATE，不匹配时对目标表执行INSERT。MERGE INTO合并执行UPDATE和INSERT，避免多次执行以提高执行效率。

注意事项

- 执行MERGE INTO操作的用户需要同时拥有目标表的UPDATE和INSERT权限，以及源表的SELECT权限。
- 不支持PREPARE。
- 不支持重分布过程中执行MERGE INTO。
- 不支持对包含触发器的目标表执行MERGE INTO。
- 对roundrobin表执行MERGE INTO时，建议关闭GUC参数allow_concurrent_tuple_update，否则会不支持部分MERGE INTO语句。

语法格式

```
MERGE [/*+ plan_hint */] INTO table_name [ partition_clause ] [ [ AS ] alias ]
USING { { table_name | view_name } | subquery } [ partition_clause ] [ [ AS ] alias ]
ON ( condition )
[
  WHEN MATCHED THEN
  UPDATE SET { column_name = { expression | DEFAULT } |
         ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
  [ WHERE condition ]
]
```

 版权所有 © 华为云计算技术有限公司

```
[
  WHEN NOT MATCHED THEN
  INSERT { DEFAULT VALUES |
  [ ( column_name [, ...] ) ] VALUES ( { expression | DEFAULT } [, ...] ) [, ...] [ WHERE condition ] }
];

where partition_clause can be:
PARTITION ( partition_name ) | PARTITION FOR ( partition_key_value [, ...] )
```

参数说明

表 21-289 MERGE INTO 参数说明

参数	描述	取值范围
plan_hint子句	以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见《数据仓库服务开发指南》中的“DWS性能调优>SQL调优>使用Plan Hint进行调优”章节。	-
INTO子句	指定正在更新或插入的目标表。 ● **table_name** 目标表的表名。 ● **partition_clause** 用来指定目标表分区的子句。该参数9.1.0及以上集群版本支持。 只有当目标表是分区表时可以指定分区。指定的分区名需要是目标表已存在的分区或者指定的分区键值所处的分区已存在。 ● **alias** 目标表的别名。	有效的表名。
USING子句	指定源表，源表可以是表、视图或子查询。 ● **partition_clause** 用来指定源表分区的子句。该参数9.1.0及以上集群版本支持。 只有当源表是分区表时可以指定分区，视图或者子查询不支持指定分区。指定的分区名需要是源表已存在的分区或者指定的分区键值所处的分区已存在。 ● **alias** 目标表的别名。	有效的源表名。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
ON子句	关联条件，用于指定目标表和源表的关联条件。 ● 不支持更新关联条件中的字段。 ● ON关联条件可以是ctid，xc_node_id，tableoid这三个系统列。	-
WHEN MATCHED子句	当源表和目标表中数据根据关联条件匹配时，选择WHEN MATCHED子句进行UPDATE操作。 ● 不支持更新分布列。 ● 不支持更新系统表、系统列。 ● 当指定目标表的分区时，若UPDATE目标表的分区键导致分区变化： – 目标表是列存表时，会报错“The inserted partition key is not mapped to the specified 'partition_name' partition”。 – 目标表是行存表时： 若目标表的表属性为ENABLE ROW MOVEMENT，则不会报错，更新的数据会属于新分区。 若目标表的表属性为DISABLE ROW MOVEMENT，会报错“fail to update partitioned table 'table_name' ”。	-
WHEN NOT MATCHED子句	当源表和目标表中数据根据关联条件无法匹配时，选择WHEN NOT MATCHED子句进行INSERT操作。 ● 不支持INSERT子句中包含多个VALUES。 ● WHEN MATCHED和WHEN NOT MATCHED子句顺序可以交换，可以缺省其中一个，但不能同时缺省，不支持同时指定两个WHEN MATCHED或WHEN NOT MATCHED子句。 ● 当指定目标表的分区时，若INSERT了超出该分区范围的数据，会报错“The inserted partition key is not mapped to the specified 'partition_name' partition”。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
DEFAULT	用对应字段的缺省值填充该字段。	如果没有缺省值，则为NULL。
WHERE condition	UPDATE子句和INSERT子句的条件，只有在条件满足时才进行更新操作，可缺省。 不支持WHERE条件中引用系统列。	-

示例

创建目标表products和源表newproducts，并插入数据。

```
CREATE TABLE products
(
product_id INTEGER,
product_name VARCHAR2(60),
category VARCHAR2(60)
);

INSERT INTO products VALUES (1501, 'vivitar 35mm', 'electronics');
INSERT INTO products VALUES (1502, 'olympus is50', 'electronics');
INSERT INTO products VALUES (1600, 'play gym', 'toys');
INSERT INTO products VALUES (1601, 'lamaze', 'toys');
INSERT INTO products VALUES (1666, 'harry potter', 'dvd');

CREATE TABLE newproducts
(
product_id INTEGER,
product_name VARCHAR2(60),
category VARCHAR2(60)
);

INSERT INTO newproducts VALUES (1502, 'olympus camera', 'electronics');
INSERT INTO newproducts VALUES (1601, 'lamaze', 'toys');
INSERT INTO newproducts VALUES (1666, 'harry potter', 'toys');
INSERT INTO newproducts VALUES (1700, 'wait interface', 'books');
```

执行MERGE INTO操作，仅当product_id字段匹配且目标表product_name字段非 'play gym' 时UPDATE，product_id字段不匹配且源表category字段为books时INSERT。

```
MERGE INTO products p
USING newproducts np
ON (p.product_id = np.product_id)
WHEN MATCHED THEN
  UPDATE SET p.product_name = np.product_name, p.category = np.category WHERE p.product_name !=
'play gym'
WHEN NOT MATCHED THEN
  INSERT VALUES (np.product_id, np.product_name, np.category) WHERE np.category = 'books';
```

查询更新后的结果显示：目标表products的对应product_id字段1502、1666行相关数据已更新，1700行为新插入的数据。

```
SELECT * FROM products ORDER BY product_id;
 product_id |  product_name  | category
------------+----------------+-----------
       1501 | vivitar 35mm   | electronics
       1502 | olympus camera | electronics
       1600 | play gym       | toys
       1601 | lamaze         | toys
       1666 | harry potter   | toys
```


版权所有 © 华为云计算技术有限公司

```
    1700 | wait interface | books
(6 rows)
```

21.14.9 INSERT 和 UPSERT

21.14.9.1 INSERT

功能描述

向表中添加一行或多行数据。

注意事项

- 使用INSERT请遵守INSERT操作规范，参见《 数据仓库服务开发指南 》中“INSERT操作规范”章节。
- 只有拥有表INSERT权限的用户，才可以向表中插入数据。
- 如果使用RETURNING子句，用户必须要有该表的SELECT权限。
- 如果使用QUERY子句插入来自查询里的数据行，用户还需要拥有在查询里使用的表的SELECT权限。
- 如果使用OVERWRITE子句覆盖式插入数据，用户还需要拥有该表的SELECT和TRUNCATE权限（因为OVERWRITE操作是先清空目标表原有数据）。
- 当连接到TD兼容的数据库时，td_compatible_truncation参数设置为on时，将启用超长字符串自动截断功能，在后续的INSERT语句中（不包含外表的场景下），对目标表中char和varchar类型的列上插入超长字符串时，系统会自动按照目标表中相应列定义的最大长度对超长字符串进行截断。

须知

如果向字符集为字节类型编码（SQL_ASCII，LATIN1等）的数据库中插入多字节字符数据（如汉字等），且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

语法格式

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
INSERT [/*+ plan_hint */] [ IGNORE | OVERWRITE ] INTO table_name [ partition_clause ] [ AS alias ]
[ ( column_name [, ...] ) ]
    { DEFAULT VALUES
    | VALUES {( { expression | DEFAULT } [, ...] ) }[, ...]
    | query }
    [ ON DUPLICATE KEY duplicate_action | ON CONFLICT [ conflict_target ] conflict_action ]
    [ RETURNING {* | {output_expression [ [ AS ] output_name ] }[, ...]} ];

where partition_clause can be:
    PARTITION ( partition_name )
    | PARTITION FOR ( partition_key_value [, ...] )

and duplicate_action can be:

    UPDATE { column_name = { expression | DEFAULT } |
             ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] )
```


版权所有 © 华为云计算技术有限公司

```
        } [, ...]

and conflict_target can be one of:

    ( { index_column_name | ( index_expression ) } [ COLLATE collation ] [ opclass ] [, ...] ) [ WHERE
index_predicate ]
    ON CONSTRAINT constraint_name

and conflict_action is one of:

    DO NOTHING
    DO UPDATE SET { column_name = { expression | DEFAULT } |
                    ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] )
                  } [, ...]
              [ WHERE condition ]
```

参数说明

表 21-290 INSERT 参数说明

参数	描述	取值范围或示例
WITH [RECURSIVE] with_query [, ...]	用于声明一个或多个可以在主查询中通过名字引用的子查询，相当于临时表。 如果声明了RECURSIVE，则允许SELECT子查询通过名字引用它自己。 其中with_query的详细格式为： `with_query_name [ ( column_name [, ...] ) ] AS ( {select | values | insert | update | delete} )` – with_query_name指定子查询生成的结果集名字，在查询中可使用该名称访问子查询的结果集。 – column_name指定子查询结果集中显示的列名。 – 每个子查询可以是SELECT，VALUES，INSERT，UPDATE或DELETE语句。	**示例：使用WITH语句生成临时表再插入另一张表**
plan_hint子句	以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见《数据仓库服务开发指南》中的“DWS性能调优>SQL调优>使用Plan Hint进行调优”章节。	-
IGNORE	用于主键或者唯一约束冲突时忽略冲突的数据。 详细介绍参见**UPSERT**。	**示例：使用IGNORE忽略主键冲突**

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
OVERWRITE	**仅8.1.1及以上版本支持**，用于标识覆盖式插入方式，使用此种插入方式执行结束后，目标源数据被清空，只存在新插入的数据。 OVERWRITE支持指定列插入的功能，其他列为默认值，若无默认值则为NULL。 INSERT OVERWRITE约束： ● OVERWRITE不要和INSERT INTO这类实时写入的操作并发，否则实时写入数据有被意外清理的风险。 ● 复制表（Replication）不支持INSERT OVERWRITE。OVERWRITE适用于大批量数据导入场景，不建议用于少量数据的插入场景。 ● 避免对同一张表执行并发insert overwrite操作，否则会出现类似报错“tuple concurrently updated.”。 ● 如果集群正在扩缩容，且INSERT OVERWRITE的写入表需要执行数据重分布，则INSERT OVERWRITE会清除当前数据，并自动将插入的数据按扩缩容后的节点来进行数据分布。如果INSERT OVERWRITE和该表的数据重分布过程同时执行，INSERT OVERWRITE会中断该表的数据重分布过程。 ● OVERWRITE支持覆盖式插入OBS外表，此方式会删除原目录中数据文件。 ● 外表支持OVERWRITE单分区，支持Value多级分区，该功能仅9.1.0.100及以上版本支持。 示例：INSERT OVERWRITE into table_name partition for (p1_value, p2_value...) ... ● 对同一张表同时执行INSERT OVERWRITE和ALTER TABLE...ADD PARTITION操作时，偶发可能会出现不同节点的分区数量不一致的情况，从而导致INSERT OVERWRITE报错。遇到这种情况时，建议重新执行INSERT OVERWRITE语句，即可解决。	-
table_name	要插入数据的目标表名。	已存在的表名。
AS	用于给目标表table_name指定别名。alias即为别名的名字。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
column_name	目标表中的字段名： • 字段名可以用子字段名或者数组下标修饰。 • 没有在字段列表中出现的每个字段，将由系统默认值，或者声明时的默认值填充，若都没有则用NULL填充。例如，向一个复合类型中的某些字段插入数据，其他字段将是NULL。 • 目标字段（column_name）可以按顺序排列。如果没有列出任何字段，则默认全部字段，且顺序为表声明时的顺序。 • 如果value子句和query中只提供了N个字段，则目标字段为前N个字段。 • value子句和query提供的值在表中从左到右关联到对应列。	已存在的字段名。
partition_name	分区名。	已存在的分区名。
partition_key_value	分区键值。 通过PARTITION FOR（partition_key_value [, ...])子句指定的这一组值，可以唯一确定一个分区。支持指定多个分区键值，多个分区键值以英文逗号分隔。	需要进行重命名分区的分区键的取值范围。
expression	赋予对应column的一个有效表达式或值： • 向表中字段插入单引号时需要使用单引号自身进行转义。 • 如果插入行的表达式不是正确的数据类型，系统试图进行类型转换，若转换不成功，则插入数据失败，系统返回错误信息。	**示例：向某列中插入一个表达式或值**
DEFAULT	对应字段名的缺省值。	如果没有缺省值，则为NULL，参见**示例：写入数据时对某个字段使用默认值**。
query	一个查询语句（SELECT语句），将查询结果作为插入的数据。	**示例：将查询结果插入表**
ON DUPLICATE KEY	用于主键或者唯一约束冲突时更新冲突的数据。 duplicate_action指定更新列和更新的数据。 详细介绍参见**UPSERT**。	**示例：使用ON DUPLICATE KEY更新主键或唯一约束冲突时插入数据**

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
ON CONFLICT	用于主键或者唯一约束冲突时忽略或者更新冲突的数据。 conflict_target用于指定列名index_column_name 、包含多个列名的表达式index_expression或约束名字constraint_name。用于从列名、包含多个列名的表达式或者约束名推断是否有唯一索引。其中index_column_name和index_expression遵循CREATE INDEX的索引列格式。 conflict_action指定主键或者唯一约束冲突时执行的策略。有两种： • DO NOTHING冲突忽略。 • DO UPDATE SET冲突更新。 后面指定更新列和更新的数据。 详细介绍参见**UPSERT**。	**示例**
RETURNING	返回实际插入的行，RETURNING列表的语法与SELECT的输出列表一致。	**示例：使用RETURNING返回插入结果**
output_expression	INSERT命令在每一行都被插入之后用于计算输出结果的表达式。	该表达式可以使用table的任意字段。可以使用*返回被插入行的所有字段。
output_name	字段的输出名称。	字符串，符合标识符命名规范。详情请参见**标识符命名规范**。

示例：插入单条或多条数据

创建表reason_t1：

```
DROP TABLE IF EXISTS reason_t1;
CREATE TABLE reason_t1
(
    TABLE_SK        INTEGER             ,
    TABLE_ID        VARCHAR(20)         ,
    TABLE_NA         VARCHAR(20)
);
```

向表中插入一条数据：

```
INSERT INTO reason_t1(TABLE_SK, TABLE_ID, TABLE_NA) VALUES (1, 'S01', 'StudentA');
```

向表中插入一条数据，和上一条语法等效：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA');
```

向表中插入TABLE_SK小于1的数据：

版权所有 © 华为云计算技术有限公司

```
INSERT INTO reason_t1 SELECT * FROM reason_t1 WHERE TABLE_SK < 1;
```

向表中插入多条数据：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
SELECT * FROM reason_t1 ORDER BY 1;
```

table_sk	table_id	table_na
1	S01	StudentA
1	S01	StudentA
1	S01	StudentA
2	T01	TeacherA
3	T02	TeacherB

示例：将查询结果插入表

将表reason_t1的数据插入到表reason_t1中：

```
INSERT INTO reason_t1 SELECT * FROM reason_t1;
```

示例：使用 RETURNING 返回插入结果

插入表数据，并返回插入的数据对应的字段table_id和table_na的取值。

```
INSERT INTO reason_t1 SELECT * FROM reason_t1 RETURNING table_id, table_na;
```

table_id	table_na
T01	TeacherA
T02	TeacherB
S01	StudentA
T02	TeacherB
T01	TeacherA

插入数据，并返回插入数据的所有字段取值。

```
INSERT INTO reason_t1 SELECT * FROM reason_t1 RETURNING *;
```

 版权所有 © 华为云计算技术有限公司

table_sk	table_id	table_na
2	T01	TeacherA
3	T02	TeacherB
2	T01	TeacherA
3	T02	TeacherB
1	S01	StudentA

插入数据，并返回重命名后的字段my_id取值。

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA') RETURNING table_id AS my_id;
```

my_id
S01

示例：使用 INSERT OVERWRITE 覆盖式插入

使用INSERT OVERWRITE更新表中的数据，即覆盖式插入数据（先清空目标表数据）：

```
INSERT OVERWRITE INTO reason_t1 values (5, 'S02', 'StudentB');
SELECT * FROM reason_t1 ORDER BY 1;
```

table_sk	table_id	table_na
5	S02	StudentB

示例：写入数据时对某个字段使用默认值

```
DROP TABLE IF EXISTS products;
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    price DECIMAL(10, 2) DEFAULT 0.00
);
```

插入一条数据，未定义price字段的取值，则默认使用0.00。

```
INSERT INTO products (id, name)VALUES (201, 'phone'); -- price字段将使用默认值0.00
```

插入一条数据，显式使用默认值。

```
INSERT INTO products (id, name, price)VALUES (202, 'pad', DEFAULT); -- 显式使用默认值
SELECT * FROM product;
```

id	name	price
201	phone	0.00
202	pad	0.00

版权所有 © 华为云计算技术有限公司

示例：使用 WITH 语句生成临时表再插入另一张表

将一个表中的部分数据插入到另一个表中：先通过WITH子查询得到一张临时表temp_t，然后将临时表temp_t中的所有数据插入另一张表reason_t1中：

```
WITH temp_t AS (SELECT * FROM reason_t1) INSERT INTO reason_t1 SELECT * FROM temp_t ORDER BY 1;
```

示例：使用 IGNORE 忽略主键冲突

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(50)
)
DISTRIBUTE BY HASH (id);
```

插入两条主键都是1的记录，提示冲突，无法执行。

```
INSERT  INTO users (id, username)
VALUES (1, 'john_doe'),
       (2, 'jane_smith'),
       (1, 'duplicate_user'); --主键冲突，无法插入
```

```
error_code: DWS.S0010010
error_msg: Constraint Violation. STATE: 23505, message: ERROR: dn_6005_6006: duplicate key value violates unique constraint "users_pkey"
Detail: Key (id)=(1) already exists.
```

使用IGNORE忽略冲突，冲突数据被忽略跳过。

```
INSERT IGNORE INTO users (id, username)
VALUES (1, 'john_doe'),
       (2, 'jane_smith'),
       (1, 'duplicate_user'); -- 冲突数据会被忽略
SELECT * FROM users;
```

id	username
1	john_doe
2	jane_smith

示例：使用 ON DUPLICATE KEY 更新主键或唯一约束冲突时插入数据

```
DROP TABLE IF EXISTS user_logins;
CREATE TABLE user_logins
 (user_id  int  primary key,
login_count  int)
DISTRIBUTE BY HASH (user_id);
```

插入2条数据，第二条主键冲突，插入失败。

```
INSERT INTO user_logins VALUES (1, 1);
INSERT INTO user_logins VALUES (1, 1);  --主键冲突，插入失败
SELECT * FROM user_logins;
```

user_id	login_count
1	1

版权所有 © 华为云计算技术有限公司

使用ON DUPLICATE KEY声明，如果主键冲突了，则更新login_count字段，使其加1。

```
INSERT INTO user_logins VALUES (1, 1) ON DUPLICATE KEY UPDATE login_count = login_count + 1;
SELECT * FROM user_logins;
```

user_id	login_count
1	2

示例：向某列中插入一个表达式或值

```
CREATE TABLE tt01 (id int,content varchar(50));

INSERT INTO tt01 values (1,'Jack say ''hello''');  --插入Jack say 'hello'，加单引号进行转义
INSERT INTO tt01 values (2,'Rose do 50%');
INSERT INTO tt01 values (3,'Lilei say ''world''');  --插入Lilei say 'world'，加单引号进行转义
INSERT INTO tt01 values (4,'Hanmei do 100%');

SELECT * FROM tt01;
```

id	content
3	Lilei say 'world'
2	Rose do 50%
1	Jack say 'hello'
4	Hanmei do 100%

示例：向分区表的指定分区插入数据

向分区表的指定分区插入数据：

```
CREATE TABLE test_range_row(a int, d int)
DISTRIBUTE BY hash(a) PARTITION BY RANGE(d)
(
    PARTITION p1 values LESS THAN (60),
    PARTITION p2 values LESS THAN (75),
    PARTITION p3 values LESS THAN (90),
    PARTITION p4 VALUES LESS THAN (maxvalue)
);
INSERT OVERWRITE INTO test_range_row PARTITION(p1) VALUES(55,51);
INSERT OVERWRITE INTO test_range_row PARTITION(p3) VALUES(85,80);

DELETE FROM test_range_row PARTITION(p1);
```

向分区外表中的指定分区插入数据：

```
CREATE FOREIGN TABLE test_ft(c1 text, c2 text, c3 text, c4 text)
PARTITION BY (c1,c2,c3) AUTOMAPPED;
```

向一级(p1_1)分区中插入数据：

```
INSERT OVERWRITE INTO test_ft PARTITION FOR(p1_1) VALUES(p1_1,2,3,4);
```

向二级(p1_2,p2_2)分区中插入数据：

```
INSERT OVERWRITE INTO test_ft PARTITION FOR(p1_2,p2_2) VALUES(p1_2,p2_2,3,4);
```

向三级(p1_3,p2_3,p3_3)分区中插入数据：

```
INSERT OVERWRITE INTO test_ft PARTITION FOR(p1_3,p2_3,p3_3) VALUES(p1_3,p2_3,p3_3,4);
```


版权所有 © 华为云计算技术有限公司

示例：MySQL 兼容模式下，通过设置 hive_compat_options 实现插入空串不被当作 0 处理

DWS在MySQL兼容模式下进行数据操作时，如果插入空串，系统会默认**将空串当作0处理**。

在金融行业，例如从Hive导入数据到DWS场景中，如果将空串当作0处理会导致数据被篡改（金融场景下空串和0具有不同的意义），影响数据的准确性和可靠性。

如何确保数据在导入过程中的完整性和一致性？通过设置GUC参数hive_compat_options值为**empty_str_to_null**实现转换逻辑：**当插入空串时，系统会将空串转为NULL**，从而保持数据的一致性和准确性，下面模拟这个过程。

创建一个兼容模式为MySQL的数据库。

```
CREATE DATABASE mysqldb DBCOMPATIBILITY 'MYSQL';
```

重新连接新的数据库mysqldb，执行以下语句，并插入空串。

```
CREATE TABLE test1 (a int, b int);
INSERT INTO test1 VALUES (1, '');
```

查询数据，发现空串被设置为0。

```
SELECT * FROM test1;
```

```
mysqldb=> SELECT * FROM test1;
 a | b
---+---
 1 | 0
(1 row)
```

将GUC参数hive_compat_options设置为**empty_str_to_null**。

```
ALTER DATABASE mysqldb SET hive_compat_options TO empty_str_to_null;
```

再次插入数据。

```
INSERT INTO test1 VALUES (2, '');
```

查询数据，发现空串自动转换为NULL。

```
SELECT * FROM test1;
```

```
mysqldb=> SELECT * FROM test1;
 a | b
---+---
 1 | 0
 2 |
(2 rows)
```

优化建议

- VALUES

  通过insert语句批量插入数据时，建议将多条记录合并入一条语句中执行插入，以提高数据加载性能。例如，INSERT INTO sections VALUES (30, 'Administration', 31, 1900),(40, 'Development', 35, 2000), (50, 'Development' , 60 , 2001);

 版权所有 © 华为云计算技术有限公司

21.14.9.2 UPSERT

功能描述

向表中添加一行或多行数据。当出现主键或者唯一约束冲突时，它可以选择更新冲突的数据，或者忽略这些冲突。

须知

UPSERT语法仅8.1.1及以上版本支持。

注意事项

⚠ 警告

- 避免UPDATE/UPSERT并发更新同一张列存表。
- 更多开发设计规范参见《开发指南》的“DWS总体开发设计规范”章节。

- **权限要求**：只有拥有表INSERT、UPDATE权限的用户，才可以通过UPSERT语句向表中插入或更新数据。
- **索引要求**：目标表上必须包含主键或者唯一索引才可以执行UPSERT的冲突更新语句。所有的唯一索引都不可用时不能执行UPSERT的冲突更新语句，重建索引后可以正常执行。
- **其它约束：**
  - 不支持与WITH子句同时使用。
  - 不支持与INSERT OVERWRITE同时使用。
  - UPDATE子句和UPDATE的WHERE子句不能有子查询。
  - UPDATE子句中VALUES(colname)用法不支持外层嵌套函数，即不支持类似sqrt(VALUES(colname))用法。如需支持，使用EXCLUDED.colname语法。
  - INSERT INTO ON CONFLICT(...) DO UPDATE必须有conflict_target。即必须指定列或者约束名。
  - 不能更新分布列。（**例外：当分布键与更新值相同时**）。

    ```
    CREATE TABLE t1(dist_key int PRIMARY KEY, a int, b int);
    INSERT INTO t1 VALUES(1,2,3) ON CONFLICT(dist_key) DO UPDATE SET dist_key = EXCLUDED.dist_key, a = EXCLUDED.a + 1;
    INSERT INTO t1 VALUES(1,2,3) ON CONFLICT(dist_key) DO UPDATE SET dist_key = dist_key, a = EXCLUDED.a + 1;
    ```

  - 不支持对包含触发器（触发事件为INSERT或UPDATE）的目标表执行UPSERT语句。
  - 不支持对可更新视图执行UPSERT语句。
  - UPDATE子句、UPDATE中WHERE子句或者索引条件表达式不能包含**不下推函数**。
  - 不支持延迟唯一索引。
  - 通过INSERT INTO SELECT语句执行UPSERT的更新操作时，需要注意SELECT语句的查询结果顺序。在分布式环境中未使用ORDER BY语句时每次执行相同

版权所有 © 华为云计算技术有限公司

的SELECT语句返回结果顺序可能不一样，这会导致UPSERT语句的执行结果不符合预期。

- 不支持多次更新。插入多组数据间如果有冲突，则会出现类似“INSERT ON CONFLICT DO UPDATE command cannot affect row a second time”报错（**例外：当查询计划是PGXC计划时**），详细案例参见**常见问题**。

- **性能影响**：
  - 当在列存表上执行UPSERT时，建议开启DELTA表，能够有效防止执行UPSERT时产生小CU（大量的小CU会导致空间膨胀和查询性能差）。
  - 对于列存表上的UPSERT、UPDATE、DELETE并发场景，由于并发更新到同一个CU时需要等待CU锁，无法支持这几个操作的并发执行，开启DELTA也无法解决该问题。如果需要支持列存表的UPSERT、UPDATE、DELETE并发场景，请使用HStore表。
- **死锁问题**：可能存在分布式死锁导致查询hang问题。例如在一个事务中或者通过JDBC(setAutoCommit(false))批量执行多条UPSERT语句，多个类似任务同时执行时，由于不同线程在不同节点更新顺序可能不同，在存在并发更新同一行的场景里可能会有死锁问题。**解决办法**如下：

  a. 减小GUC参数lockwait_timeout值（默认20分钟）。分布式死锁会等待lockwait_timeout时间然后报错。通过减小此参数的数值，降低死锁造成的业务等待时间。

  b. 保证主键相同的数据只从一个数据库连接导入数据库，可以并发执行UPSERT语句。

  c. 每个事务中只执行一条UPSERT语句，这样也可以并发执行UPSERT语句。

  d. 单线程执行多条UPSERT语句，不能并发执行UPSERT语句。

  如上解决办法中，方法1只能降低等待时间，无法解决死锁问题，在业务中有UPSERT语句时，仍建议减小此参数值。方法2、3、4均可以解决死锁问题，但建议采用方法2，其性能优于其他两个方法。

语法格式

详细介绍请参见INSERT的**语法格式**。有两种UPSERT语法格式：

第一种**不指定索引**。会在所有主键或唯一索引上检查冲突，有冲突就会忽略或者更新。

第二种**指定索引**。会从ON CONFLICT子句中指定列名、包含列名的表达式或者约束名上推断主键或者唯一索引。

- **场景一：冲突更新语法**：当插入的数据发生冲突时，使用新数据更新已有数据。
  - **不指定索引的冲突更新**：
    ```
    INSERT INTO ON DUPLICATE KEY UPDATE
    ```
  - **指定索引的冲突更新**：
    ```
    INSERT INTO ON CONFLICT(...) DO UPDATE SET
    INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO UPDATE SET
    ```
- **场景二：冲突忽略语法**：当插入的数据发生冲突时，直接忽略这次插入操作。
  - **不指定索引的冲突忽略**：
    ```
    INSERT IGNORE
    INSERT INTO ON CONFLICT DO NOTHING
    ```
  - **指定索引的冲突忽略**：
    ```
    INSERT INTO ON CONFLICT(...) DO NOTHING
    INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO NOTHING
    ```


版权所有 © 华为云计算技术有限公司

表 21-291 UPSERT 参数说明

参数	描述	取值范围或示例
ON CONFLICT (...)	对于指定索引的语法形式，通过指定列名或者约束名推断主键或者唯一索引。列名可以是单一列名，或者由多个列名组成的表达式，比如（column1，column2，column3）。 由于创建索引时可以指定collation和opclass，所以此处列名后也可以指定。 COLLATE collation指定列的排序规则。opclass指定操作符类的名字。具体参考**CREATE INDEX**。 从指定列名的表达式中推断出唯一索引，整体原则是判断某唯一索引是否能够恰好完全包含conflict_target指定的列名。 • 如果没有指定collation和opclass，那么只要列或者列名的表达式相同（不管索引列上指定的collation和opclass是什么），都认为匹配。 • 如果指定collation和opclass，那么需要与索引的collation和opclass匹配才可以。	-
UPDATE子句	UPDATE子句可以通过VALUES(colname)或者EXCLUDED.colname引用插入的数据。EXCLUDED表示因冲突原本该排除的数据行，语法示例参见**示例**。	-
WHERE子句	• 用于在数据冲突时，判断是否满足指定条件。如果满足，则更新冲突数据；否则忽略。 • 只有第二种语法形式的冲突更新语法可以指定WHERE子句。即INSERT INTO ON CONFLICT(...) DO UPDATE SET WHERE	-

示例

创建表reason_t2，并向表中插入数据：

```
DROP TABLE IF EXISTS reason_t2;
CREATE TABLE reason_t2
(
  a    int primary key,
  b    int,
  c    int
);

INSERT INTO reason_t2 VALUES (1, 2, 3);
SELECT * FROM reason_t2 ORDER BY 1;
 a | b | c
---+---+---
```


版权所有 © 华为云计算技术有限公司

```
 1 | 2 | 3
 (1 rows)
```

向表reason_t2中插入两条数据，一条有冲突，一条无冲突。有冲突的数据进行忽略，无冲突的数据进行插入：

```
INSERT INTO reason_t2 VALUES (1, 4, 5),(2, 6, 7) ON CONFLICT(a) DO NOTHING;
SELECT * FROM reason_t2 ORDER BY 1;
 a | b | c
---+---+----
 1 | 2 | 3
 2 | 6 | 7
(2 rows)
```

向表reason_t2中插入数据，一条有冲突，一条无冲突。有冲突的数据进行更新，无冲突的数据进行插入，EXCLUDED.b和EXCLUDED.c分别表示引用插入值的b列和c列，本例中为4，5。

```
INSERT INTO reason_t2 VALUES (1, 4, 5),(3, 8, 9) ON CONFLICT(a) DO UPDATE SET b = EXCLUDED.b, c =
EXCLUDED.c;
SELECT * FROM reason_t2 ORDER BY 1;
 a | b | c
---+---+----
 1 | 4 | 5
 2 | 6 | 7
 3 | 8 | 9
 (3 rows)
```

根据过滤条件筛选被更新的行：

```
INSERT INTO reason_t2 VALUES (2, 7, 8) ON CONFLICT (a) DO UPDATE SET b = excluded.b, c = excluded.c
WHERE reason_t2.c = 7;
SELECT * FROM reason_t2 ORDER BY 1;
 a | b | c
---+---+---
 1 | 4 | 5
 2 | 7 | 8
 3 | 8 | 9
(3 rows)
```

向表reason_t中插入数据，有冲突的数据进行更新并调整更新映射关系，即c列更新到b，b列更新到c：

```
INSERT INTO reason_t2 VALUES (1, 2, 3) ON CONFLICT (a) DO UPDATE SET b = excluded.c, c = excluded.b;
SELECT * FROM reason_t2 ORDER BY 1;
 a | b | c
---+---+---
 1 | 3 | 2
 2 | 7 | 8
 3 | 8 | 9
(3 rows)
```

常见问题

不支持多次更新。插入多组数据间如果有冲突，则会出现类似“INSERT ON CONFLICT DO UPDATE command cannot affect row a second time”报错（**例外：当查询计划是PGXC计划时**），以下是报错示例。

```
CREATE TABLE t1(id int PRIMARY KEY, a int, b int);
SET explain_perf_mode=pretty;
-- STREAM计划
EXPLAIN (COSTS OFF) INSERT INTO t1 VALUES(1,2,3),(1,5,6) ON CONFLICT(id) DO UPDATE SET a =
EXCLUDED.a + 1;
                  QUERY PLAN
-----------------------------------------------
 id |               operation
 ----+-----------------------------------------
  1 | ->  Streaming (type: GATHER)
```


版权所有 © 华为云计算技术有限公司

```
  2 |    ->  Insert on t1
  3 |       ->  Streaming(type: REDISTRIBUTE)
  4 |          ->  Values Scan on "*VALUES*"
 Predicate Information (identified by plan id)
 -----------------------------------------------
  2 --Insert on t1
        Conflict Resolution: UPDATE
        Conflict Arbiter Indexes: t1_pkey
  ====== Query Summary =====
 ------------------------------
 System available mem: 819200KB
 Query Max mem: 819200KB
 Query estimated mem: 3104KB
(18 rows)
INSERT INTO t1 VALUES(1,2,3),(1,5,6) ON CONFLICT(id) DO UPDATE SET a = EXCLUDED.a + 1;
ERROR:  dn_xxxx: INSERT ON CONFLICT DO UPDATE command cannot affect row a second time
HINT:  Ensure that no rows proposed for insertion within the same command have duplicate constrained values.
-- 关闭stream，生成PGXC计划
set enable_stream_operator = off;
EXPLAIN (COSTS OFF) INSERT INTO t1 VALUES(1,2,3),(1,5,6) ON CONFLICT(id) DO UPDATE SET a = EXCLUDED.a + 1;
                 QUERY PLAN
-----------------------------------------------
 id |           operation
 ---+-------------------------------
  1 | ->  Insert on t1
  2 |    ->  Values Scan on "*VALUES*"
 Predicate Information (identified by plan id)
 -----------------------------------------------
  1 --Insert on t1
        Conflict Resolution: UPDATE
        Conflict Arbiter Indexes: t1_pkey
        Node expr: id
(11 rows)
INSERT INTO t1 VALUES(1,2,3),(1,5,6) ON CONFLICT(id) DO UPDATE SET a = EXCLUDED.a + 1;
INSERT 0 2
```

21.14.10 UPDATE

功能描述

更新表中的数据。UPDATE修改满足条件的所有行中指定的字段值，WHERE子句声明条件，SET子句指定的字段会被修改，没有出现的字段则保持原值。

注意事项

- 使用UPDATE请遵循UPDATE操作规范，详情参见《数据仓库服务开发指南》中“UPDATE&DELETE操作规范”章节。
- **权限要求**：要修改表，用户必须对该表有UPDATE权限，对expression或condition条件里涉及到的任何表要有SELECT权限。
- **不允许对表的分布列（distribute column）进行修改**。
- UPDATE操作频繁的表不建议创建为复制表。
- **列存表约束**：
  - 列存表，暂时不支持RETURNING子句。
  - 列存表不支持结果不确定的更新（non-deterministic update）。试图对列存表用多行数据更新一行时会报错。
  - 列存表的更新操作，旧记录空间不会回收，需要执行VACUUM FULL table_name进行清理。

版权所有 © 华为云计算技术有限公司

- 对于列存表，支持轻量化UPDATE操作。轻量化UPDATE只重写更新列，减少空间使用量。列存轻量化UPDATE通过GUC参数enable_light_colupdate控制是否开启，默认关闭。
- 列存轻量化UPDATE在以下场景不能使用：更新索引列，更新主键列，更新分区列，更新PCK列和在线扩容，会自动转化为普通UPDATE方式。
- 列存轻量化UPDATE与后台列存AUTOVACUUM并发会小概率报错，可以通过ALTER TABLE设置表级参数enable_column_autovacuum_garbage为off来避免。需要注意的是设置表级参数enable_column_autovacuum_garbage为off会关闭该表的后台列存AUTOVACUUM。

语法格式

```
   UPDATE [/*+ plan_hint */] [ ONLY ] table_name [ * ] [ [ AS ] alias ]
   [ PARTITION ( partition_name ) | PARTITION FOR ( partition_key_value [, ...] ) ]
SET {column_name = { expression | DEFAULT }
   |( column_name [, ...] ) = {( { expression | DEFAULT } [, ...] ) |sub_query }}[, ...]
   [ FROM from_list] [ WHERE condition ]
   [ RETURNING {*
            | {output_expression [ [ AS ] output_name ]} [, ...] }];

where sub_query can be:
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
{ * | {expression [ [ AS ] output_name ]} [, ...] }
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY grouping_element [, ...] ]
[ HAVING condition [, ...] ]
```

参数说明

表 21-292 UPDATE 参数说明

参数	描述	示例
plan_hint子句	以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见《数据仓库服务开发指南》中的“DWS性能调优>SQL调优>使用Plan Hint进行调优”章节。	-
table_name	要更新的表名，可以带上schema名。	test.table1，其中test为schema名。
[AS] alias	为表指定一个简短的别名，方便引用，例如alias。字符串，符合标识符命名规范，详情请参见**标识符命名规范**。	更新别名为e的表employees： `UPDATE employees AS e SET e.salary = e.salary * 1.1 WHERE e.department = 'technology';`
partition_name	分区名称，该子句指定要更新的分区名称对应的数据，适用于范围分区、列表分区等已命名的分区表。仅8.2.1及以上集群版本支持该选项。	更新分区名为p1的数据： `UPDATE test_range_row PARTITION(p1) SET d = 41;`

版权所有 © 华为云计算技术有限公司

参数	描述	示例
partition_key_value	分区键值，该子句指定要更新的分区键值对应的数据。 通过PARTITION FOR (partition_key_value [, ...])子句指定的这一组值，可以唯一确定一个分区。	更新分区键值为41的数据行，将相应的b值改为20： `UPDATE test_range_row PARTITION FOR (41) SET b = 20;`

column_name	要修改的字段名。
	支持使用目标表的表名加字段名来引用这个字段。例如：
	`UPDATE foo SET foo.col_name = 'GaussDB';`
	支持使用目标表的别名加字段名来引用这个字段。例如：
	`UPDATE foo AS f SET f.col_name = 'GaussDB';`
		-

expression	赋给字段的值或表达式。	给商品价格打8折： `UPDATE products SET price = price * 0.8 WHERE product_id = 101;`
DEFAULT	用对应字段的缺省值填充该字段。如果没有缺省值，则为NULL。	重置非活跃用户的状态和登录次数： `UPDATE users SET status = DEFAULT, login_count = DEFAULT WHERE last_login < CURRENT_DATE - INTERVAL '30 days';`
sub_query	子查询。使用同一个数据库里的其他表的查询结果，来更新目标表。 其中SELECT子句具体介绍请参考**SELECT**。	使用子查询更新部门平均工资： `UPDATE departments d SET avg_salary = ( SELECT AVG(salary) FROM employees e WHERE e.dept_id = d.dept_id);`
from_list	另一个表的表达式列表，允许在WHERE条件里关联其他表字段来更新目标表。与在一个SELECT语句的FROM子句里声明表列表类似。 **须知** 目标表绝对不能出现在from_list里，除非在使用一个自连接（此时它必须以from_list的别名出现）。	**示例：使用FROM from_list子句，优先处理VIP等级高的客户的订单**

版权所有 © 华为云计算技术有限公司

参数	描述	示例
condition	跟在WHERE后面的一个返回boolean类型结果的表达式。只有这个表达式返回true的行才会被更新。	只更新数学成绩低于80分的学生： UPDATE student_scores SET math_score = math_score + 5, update_time = NOW() WHERE math_score < 80;
output_expression	在所有需要更新的行都被更新之后，用于计算返回值的表达式，便于确认更新结果。 取值范围：使用任何table以及FROM中列出的表的字段。*表示返回所有字段。	更新并返回修改前后的库存信息： UPDATE inventory SET stock = stock - 10 WHERE category = '电子产品' RETURNING item_id AS "商品ID", name AS "商品名称", stock AS "新库存量", stock + 10 AS "原库存量";
output_name	给RETURNING返回的列起的别名，将输出结果优化。字符串，符合标识符命名规范，详情请参见**标识符命名规范**。	-

示例：常用 UPDATE 场景

创建样例表并插入数据：

```
DROP TABLE IF EXISTS products;
CREATE TABLE products (
    product_id INT PRIMARY KEY,         --商品ID
    name VARCHAR(100),                  --商品名称
    price DECIMAL(10,2),                --商品价格
    stock INT,                          --商品库存
    last_updated TIMESTAMP              --数据更新时间
);

INSERT INTO products VALUES
(101, 'Phone', 2999.00, 50, NULL),
(102, 'PC', 5999.00, 30, NULL),
(103, 'PAD', 1999.00, 20, NULL);
```

不含WHERE子句表示直接更新所有记录的值：所有产品价格上浮20%。

```
UPDATE products SET price = price * 1.2 , last_updated = NOW();
SELECT * FROM products ORDER BY 1;
```

product_id	name	price	stock	last_updated
101	Phone	3598.80	50	2025-05-27 21:44:13.759399
102	PC	7198.80	30	2025-05-27 21:44:13.759399
103	PAD	2398.80	20	2025-05-27 21:44:13.759399

版权所有 © 华为云计算技术有限公司

使用WHERE子句更新指定的数据：将平板电脑的价格上涨500元。

```
UPDATE products SET price = price + 500 , last_updated = NOW()
WHERE name = 'PAD';
SELECT * FROM products ORDER BY 1;
```

product_id	name	price	stock	last_updated
101	Phone	3598.80	50	2025-05-27 21:44:13.759399
102	PC	7198.80	30	2025-05-27 21:44:13.759399
103	PAD	2898.80	20	2025-05-27 21:47:20.690807

使用WHERE子句更新特定范围的数据：商品清库存，需要将库存大于40的商品打8折处理。

```
UPDATE products SET price = price * 0.8 , last_updated = NOW()
WHERE stock > 40;
SELECT * FROM products ORDER BY 1;
```

product_id	name	price	stock	last_updated
101	Phone	2879.04	50	2025-05-27 21:53:31.996656
102	PC	7198.80	30	2025-05-27 21:44:13.759399
103	PAD	2898.80	20	2025-05-27 21:47:20.690807

在一个UPDATE命令中更新多个字段，方法是在SET子句中列出更多赋值：调整PC的价格和库存。

```
UPDATE products SET price = price - 1000 , stock = stock - 10,  last_updated = NOW()
WHERE name = 'PC';
SELECT * FROM products ORDER BY 1;
```

product_id	name	price	stock	last_updated
101	Phone	2879.04	50	2025-05-27 21:53:31.996656
102	PC	6198.80	20	2025-05-27 22:00:07.851013
103	PAD	2898.80	20	2025-05-27 21:47:20.690807

示例：更新分区表的指定分区

创建样例表并导入数据：

```
DROP TABLE IF EXISTS test_range_row;
CREATE TABLE test_range_row(a int, b int,d int)
DISTRIBUTE BY hash(a) PARTITION BY RANGE(d)
(
    PARTITION p1 values LESS THAN (60),
    PARTITION p2 values LESS THAN (75),
    PARTITION p3 values LESS THAN (90),
    PARTITION p4 VALUES LESS THAN (maxvalue)
```


版权所有 © 华为云计算技术有限公司

```
);

INSERT INTO test_range_row PARTITION(p1) VALUES(55,20,51);
INSERT INTO test_range_row PARTITION(p1) VALUES(54,18,59);
INSERT INTO test_range_row PARTITION(p2) VALUES(15,18,63);
INSERT INTO test_range_row PARTITION(p3) VALUES(12,21,80);
INSERT INTO test_range_row PARTITION(p3) VALUES(83,21,85);
INSERT INTO test_range_row PARTITION(p4) VALUES(36,21,97);

SELECT * FROM test_range_row ORDER BY d;
```

a	b	d
55	20	51
54	18	59
15	18	63
12	21	80
83	21	85
36	21	97

总条数：6

更新分区名称为p1的行数据，将d值修改为41。

```
UPDATE test_range_row PARTITION(p1) SET d = 41;
SELECT * FROM test_range_row ORDER BY d;
```

a	b	d
54	18	41
55	20	41
15	18	63
12	21	80
83	21	85
36	21	97

总条数：6

版权所有 © 华为云计算技术有限公司

从以上结果看，分区为p1，即d<60的行数据，相应的d值都被修改为41。

继续执行以下SQL，更新分区键值为41的数据行，将相应的b值改为20。

```
UPDATE test_range_row PARTITION FOR (41) SET b = 20;
SELECT * FROM test_range_row ORDER BY d;
```

a	b	d
54	20	41
55	20	41
15	18	63
12	21	80
83	21	85
36	21	97

总条数：6

示例：使用 FROM from_list 子句，优先处理 VIP 等级高的客户的订单

需要更新目标表：订单表orders，但是需要使用FROM子句关联另一张表customers的客户ID字段，来获取客户等级。

创建客户信息表和订单表，包含的字段如下：

```
DROP TABLE IF EXISTS customers;
CREATE TABLE customers (                --创建客户信息表
    customer_id INT PRIMARY KEY,        --客户ID
    name VARCHAR(100),                  --客户姓名
    vip_level INT                       --VIP等级
);
DROP TABLE IF EXISTS orders;
CREATE TABLE orders (                   --创建订单表
    order_id INT PRIMARY KEY,           --订单号
    customer_id INT ,                   --客户ID
    amount DECIMAL(10,2),               --消费金额
    status VARCHAR(20)                  --订单状态
);
```

插入测试数据，订单状态均为pending，即三个客户的订单都未处理。

```
INSERT INTO customers VALUES
(1, '张三', 1),
(2, '李四', 3),
(3, '王五', 2);
INSERT INTO orders VALUES
(1001, 1, 500.00, 'pending'),
(1002, 2, 1200.00, 'pending'),
(1003, 3, 800.00, 'pending');
SELECT * FROM orders;
```

 版权所有 © 华为云计算技术有限公司

order_id	customer_id	amount	status
1002	2	1200.00	pending
1001	1	500.00	pending
1003	3	800.00	pending

总条数：3

由于订单较多，现要求优先处理VIP等级高的客户的订单，此时需要通过FROM from_list子句，关联客户表的客户ID，获取客户VIP等级，本例为，优先处理VIP等级为3级以上的客户订单。

```
UPDATE orders o
SET status = 'priority'      --将订单状态修改为“高优先级”
FROM customers c
WHERE o.customer_id = c.customer_id AND c.vip_level >= 3;  --从客户表中获取等级VIP为3级以上的客户
SELECT * FROM orders;
```

order_id	customer_id	amount	status
1002	2	1200.00	priority
1001	1	500.00	pending
1003	3	800.00	pending

总条数：3

示例：使用 DEFAULT 重置非活跃用户的状态

创建用户表（带默认值），并插入数据：

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
user_id INT PRIMARY KEY,                --用户ID
username VARCHAR(50) ,                   --用户名
login_count INT DEFAULT 0,               --登录次数，默认为0
last_login TIMESTAMP,                   --最后一次登录的时间
status VARCHAR(20) DEFAULT 'active');     --用户状态，默认为active

INSERT INTO users (user_id, username, login_count, last_login) VALUES
(1, 'user1', 5, '2023-06-01 10:00:00'),
(2, 'user2', 3, '2023-06-15 14:30:00');
SELECT * FROM users;
```

 版权所有 © 华为云计算技术有限公司

user_id	username	login_count	last_login	status
1	user1	5	2023-06-01 10:00:00	active
2	user2	3	2023-06-15 14:30:00	active

总条数：2

重置非活跃用户的状态和登录次数：

```
UPDATE users SET
status = DEFAULT,       -- 重置为'active'
login_count = DEFAULT   -- 重置为0
WHERE last_login < CURRENT_DATE - INTERVAL '30 days';
SELECT * FROM users;
```

user_id	username	login_count	last_login	status
1	user1	0	2023-06-01 10:00:00	active
2	user2	0	2023-06-15 14:30:00	active

21.14.11 VALUES

功能描述

根据给定的值表达式计算一个或一组行的值。它通常用于在一个较大的命令内生成一个“常数表”。

注意事项

- 应当避免使用VALUES返回数量非常大的结果行，否则可能会遭遇内存耗尽或者性能低下。出现在INSERT中的VALUES是一个特殊情况，因为目标字段类型可以从INSERT的目标表获知，并不需要通过扫描VALUES列表来推测，所以此情况下可以处理非常大的结果行。
- 如果指定了多行，那么每一行都必须拥有相同的元素个数。

语法格式

```
VALUES {( expression [, ...] )} [, ...]
    [ ORDER BY { sort_expression [ ASC | DESC | USING operator ] } [, ...] ]
    [ { [ LIMIT { count | ALL } ] [ OFFSET start [ ROW | ROWS ] ] } | { LIMIT start, { count | ALL } } ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-293 VALUES 参数说明

参数	描述	取值范围
expression	用于计算或插入结果表指定地点的常量或者表达式。 在一个出现在INSERT顶层的VALUES列表中，expression可以被DEFAULT替换以表示插入目的字段的缺省值。除此以外，VALUES在其他场景时是不能使用DEFAULT的。	-
sort_expression	一个表示如何排序结果行的表达式或者整数常量。	-
ASC	指定按照升序排列。	-
DESC	指定按照降序排列。	-
operator	一个排序操作符。	-
count	返回的最大行数。	-
start	开始返回行之前忽略的行数。	-
FETCH { FIRST | NEXT } [count] { ROW | ROWS } ONLY	FETCH子句限定返回查询结果从第一行开始的总行数。	count的缺省值为1。

示例：插入单条或多条数据

创建表reason_t1：

```
DROP TABLE IF EXISTS reason_t1;
CREATE TABLE reason_t1
(
   TABLE_SK         INTEGER              ,
   TABLE_ID         VARCHAR(20)          ,
   TABLE_NA          VARCHAR(20)
);
```

向表中插入一条数据：

```
INSERT INTO reason_t1(TABLE_SK, TABLE_ID, TABLE_NA) VALUES (1, 'S01', 'StudentA');
```

向表中插入一条数据，和上一条语法等效：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA');
```

向表中插入TABLE_SK小于1的数据：

```
INSERT INTO reason_t1 SELECT * FROM reason_t1 WHERE TABLE_SK < 1;
```

向表中插入多条数据：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
SELECT * FROM reason_t1 ORDER BY 1;
```

 版权所有 © 华为云计算技术有限公司

table_sk	table_id	table_na
1	S01	StudentA
1	S01	StudentA
1	S01	StudentA
2	T01	TeacherA
3	T02	TeacherB

21.15 DCL 语法

21.15.1 DCL 语法一览表

DCL（Data Control Language数据控制语言），是用来设置或更改数据库用户或角色权限的语句。

授权

DWS提供了针对数据对象和角色授权的语句，请参考**GRANT**。

收回权限

DWS提供了收回权限的语句，请参考**REVOKE**。

设置默认权限

DWS允许设置应用于将来创建的对象的权限，请参考**ALTER DEFAULT PRIVILEGES**。

21.15.2 ALTER DEFAULT PRIVILEGES

功能描述

设置应用于将来要创建的对象的权限（不会影响现有对象的权限）。

用户只可以修改由用户本身或者用户本身所属的角色所创建的对象的默认权限，这些权限可以对全局范围设置（即数据库中创建的所有对象），也可以为指定模式下的对象设置。

查看有关数据库用户的默认权限的信息，可以查询PG_DEFAULT_ACL系统表。

注意事项

- 目前只支持表（包括视图）、序列、函数和类型（包括域）的权限更改。
- 不支持将外表的VACUUM，DROP，ALTER的权限赋予用户。

版权所有 © 华为云计算技术有限公司

语法格式

```
ALTER DEFAULT PRIVILEGES
    [ FOR { ROLE | USER } target_role [, ...] ]
    [ IN SCHEMA schema_name [, ...] ]
    abbreviated_grant_or_revoke;
```

abbreviated_grant_or_revoke子句用于指定对哪些对象进行授权或回收权限，可以是以下选项之一：

- grant_on_tables_clause
- grant_on_functions_clause
- grant_on_types_clause
- grant_on_sequences_clause
- revoke_on_tables_clause
- revoke_on_functions_clause
- revoke_on_types_clause
- revoke_on_sequences_clause

grant_on_tables_clause子句用于对表授权。

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | ANALYZE | ANALYSE
| VACUUM | ALTER | DROP }
    [, ...] | ALL [ PRIVILEGES ] }
    ON TABLES
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ]
```

grant_on_functions_clause子句用于对函数授权。

```
GRANT { { EXECUTE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
    ON FUNCTIONS
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ]
```

grant_on_types_clause子句用于对类型授权。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON TYPES
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ]
```

grant_on_sequences_clause子句用于对序列授权。

```
GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON SEQUENCES
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ]
```

revoke_on_tables_clause子句用于回收表对象的权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | ANALYZE | ANALYSE |
VACUUM | ALTER | DROP }
    [, ...] | ALL [ PRIVILEGES ] }
    ON TABLES
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]
```

revoke_on_functions_clause子句用于回收函数的权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { EXECUTE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
    ON FUNCTIONS
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]
```

revoke_on_types_clause子句用于回收类型的权限。

 版权所有 © 华为云计算技术有限公司

```
REVOKE [ GRANT OPTION FOR ]
    { USAGE | ALL [ PRIVILEGES ] }
    ON TYPES
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]
```

revoke_on_sequences_clause子句用于回收序列的权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON SEQUENCES
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]
```

参数说明

表 21-294 ALTER DEFAULT PRIVILEGES 参数说明

参数	描述	取值范围或示例

FOR { ROLE | USER } target_role [, ...]	FOR ROLE/USER 后面跟着的target_role是指将来要创建对象的角色或用户名称，即要访问谁未来创建的对象，就写这个owner对应的角色或用户名称。如果省略FOR ROLE/USER，则缺省值为当前角色/用户。
	target_role必须有schema_name的CREATE权限。查看角色/用户是否具有schema的CREATE权限可使用has_schema_privilege函数。
	`SELECT a.rolname, n.nspname FROM pg_authid as a, pg_namespace as n WHERE has_schema_privilege(a.oid, n.oid, 'CREATE');`
		已有角色或用户名称。
		举例，授权用户jim可访问用户lily未来创建对象的权限。
		`ALTER DEFAULT PRIVILEGES FOR ROLE lily IN SCHEMA lily GRANT SELECT ON tables TO jim;`
		或者
		`ALTER DEFAULT PRIVILEGES FOR USER lily IN SCHEMA lily GRANT SELECT ON tables TO jim;`

schema_name	已有模式名称。 如果指定了模式名，那么之后在这个模式下面创建的所有对象默认的权限都会被修改。如果IN SCHEMA被省略，那么全局权限会被修改。	已有模式名称。
role_name	被授予或者取消权限的角色名称。 **说明** 如果需删除一个被赋予了默认权限的角色，必须撤销其默认权限的更改或者使用DROP OWNED BY删除该角色的默认权限记录。	已有角色名称。

示例

- 将创建在模式tpcds里的所有表（和视图）的SELECT权限授予每一个用户。
  ```
  ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds GRANT SELECT ON TABLES TO PUBLIC;
  ```
- 将tpcds下的所有表的插入权限授予用户jack。
  ```
  ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds GRANT INSERT ON TABLES TO jack;
  ```
- 撤销上述权限。
  ```
  ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds REVOKE SELECT ON TABLES FROM PUBLIC;
  ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds REVOKE INSERT ON TABLES FROM jack;
  ```


版权所有 © 华为云计算技术有限公司

- 假设有两个用户test1、test2，如果需要test2用户对test1用户未来创建的表都有查询权限，可执行如下操作：

  a. 将test1的schema的权限赋予test2用户。
  ```
  GRANT usage, create ON SCHEMA test1 TO test2;
  ```

  b. 将test1用户下的表的查询权限赋予test2用户。**注**，以下的FOR USER也可以改成FOR ROLE，作用效果一致。
  ```
  ALTER DEFAULT PRIVILEGES FOR USER test1 IN SCHEMA test1 GRANT SELECT ON tables TO test2;
  ```

  c. test1用户创建表。
  ```
  SET ROLE test1 password '{password}';
  CREATE TABLE test3( a int, b int);
  ```

  d. 使test2用户执行查询。
  ```
  SET ROLE test2 password '{password}';
  SELECT * FROM test1.test3;
   a | b
  ---+---
  (0 rows)
  ```

相关链接

GRANT，**REVOKE**

21.15.3 ANALYZE | ANALYSE

功能描述

用于收集有关数据库中表内容的统计信息，统计信息分为表级和列级。其中，表级统计信息存储在PG_CLASS中，列级统计结果存储在系统表PG_STATISTIC下。执行计划生成器会使用这些统计数据，以确定最有效的执行计划。

它的主要作用是：

- **优化查询性能**：为查询优化器提供数据分布信息，帮助生成更高效的执行计划。
- **提高查询计划准确性**：使优化器能更准确地估算不同查询路径的成本。
- **支持索引选择**：帮助优化器决定是否使用索引以及使用哪个索引。

什么时候需要做 ANALYZE 操作

- **大规模数据变化**：

  大规模数据导入/UPDATE/DELETE等操作，会导致表数据行数变化，新增的大量数据也会导致数据特征发生大的变化，此时需要对表重新收集统计信息。

- **查询新增数据**：

  常见于业务表新增数据查询场景，这个也是客户现场中最常出现的统计信息没有及时更新而导致的数据查询不准确的问题，这种场景最主要的特征如下：

  - 存在一个按照时间增长的业务表。
  - 业务表每天入库新一天的数据。
  - 数据入库之后查询新增数据进行数据加工分析。

  在最后步骤的数据加工分析时，最常用的方法就是使用Filter条件从分区表中筛选数据，如

  ```
  passtime > ‘2020-01-19 00:00:00’ AND pastime < ‘2020-01-20 00:00:00’
  ```

  假如新增数据入库之后没有做analyze，优化器发现Filter条件中的passtime取值范围超过了统计信息中记录的passtime值的上边界，会把估算满足passtime >

 版权所有 © 华为云计算技术有限公司

‘2020-01-19 00:00:00’ AND pastime < ‘2020-01-20 00:00:00’的tuple个数为1条，导致估算行数验证失真。

注意事项

- 如果没有指定参数，ANALYZE会分析当前数据库中的每个表和分区表。同时也可以通过指定table_name、column和partition_name参数把分析限定在特定的表、列或分区表中。
- 能够执行ANALYZE特定表的用户，包括表的所有者、表所在数据库的所有者、通过GRANT被授予该表上ANALYZE权限的用户或者被授予了gs_role_analyze_any角色的用户以及有SYSADMIN属性的用户。
- 在百分比采样收集统计信息时，用户需要被授予ANALYZE和SELECT权限。
- 开启分区统计信息参数enable_analyze_partition，对分区表设置表级incremental_analyze参数，会先对无统计信息或有数据变化的分区执行ANALYZE，然后通过合并分区统计信息方式生成分区主表的统计信息。
- 仅8.1.1及以上集群版本支持在匿名块、事务块、函数或存储过程内对单表进行ANALYZE操作。
- 对于ANALYZE全库，库中各表的ANALYZE处于不同的事务中，所以不支持在匿名块、事务块、函数或存储过程内对全库执行ANALYZE。
- 统计信息的回滚操作不支持PG_CLASS中相关字段的回滚。
- 全局临时表在每个会话的ANALYZE的统计信息独立。每个会话都有自己的统计信息。
- ANALYZE|ANALYSE VERIFY用于检测数据库中普通表（行存表、列存表）的数据文件是否损坏，目前此命令暂不支持HDFS表。
- ANALYZE VERIFY操作处理的大多为异常场景检测需要使用RELEASE版本。ANALYZE VERIFY场景不触发远程读，因此远程读参数不生效。对于关键系统表出现错误被系统检测出页面损坏时，将直接报错不再继续检测。

⚠ 警告

- 单次新增、修改量占表总量10%以上场景，需在业务中增加显式Analyze操作。
- 更多开发设计规范参见《开发指南》的“DWS总体开发设计规范”章节。

语法格式

- 收集表的统计信息。

```
{ ANALYZE | ANALYSE } [ { VERBOSE | LIGHT | FORCE | PREDICATE } ]
    [ table_name [ ( column_name [, ...] ) ] ];
```

- 收集分区的统计信息。

```
{ ANALYZE | ANALYSE } [ { VERBOSE | LIGHT | FORCE } ]
    [ table_name [ ( column_name [, ...] ) ] ]
    PARTITION ( partition_name ) ;
```

  - 普通分区表目前支持针对某个分区的统计信息的语法，但功能上不支持针对某个分区的统计信息收集。对指定分区执行ANALYZE，会有相应的WARNING提示。
  - 不支持使用临时采样表来收集分区的统计信息。
  - 不支持分区上的多列统计信息和表达式统计信息。

 版权所有 © 华为云计算技术有限公司

- 收集外表的统计信息。
```
{ ANALYZE | ANALYSE } [ VERBOSE ]
    { foreign_table_name | FOREIGN TABLES };
```
- 收集多列统计信息。
```
{ANALYZE | ANALYSE} [ VERBOSE ]
    table_name (( column_1_name, column_2_name [, ...] ));
```
  - 收集多列统计信息时，请设置GUC参数default_statistics_target为负数，以使用百分比采样方式。
  - 每组多列统计信息最多支持32列。
  - 不支持收集多列统计信息的表：系统表、HDFS外表复制表、全局临时表。
- 检测当前库的数据文件。
```
{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE};
```
  - 支持对全库进行操作，由于涉及的表较多，建议以重定向保存结果**gsql -d database -p port -f "verify.sql"> verify_warning.txt 2>&1**。
  - 不支持HDFS表（内表和外表），不支持临时表和unlog表。
  - 只有外部表提示NOTICE，内部表的检测会包含在它所依赖的外部表，不对外显示。
  - 此命令的处理可容错ERROR级别的处理。由于debug版本的Assert可能会导致core无法继续执行命令，建议在release模式下操作。
  - 对于全库操作时，当关键系统表出现损坏则直接报错，不再继续执行。
- 检测表和索引的数据文件。
```
{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE} table_name|index_name [CASCADE];
```
  - 支持对普通表的操作和索引表的操作，但不支持对索引表index使用CASCADE操作。原因是CASCADE模式用于处理主表的所有索引表，当单独对索引表进行检测时，无需使用CASCADE模式。
  - 不支持HDFS表（内表和外表），不支持临时表和unlog表。
  - 对于主表的检测会同步检测主表的内部表，例如toast表、cudesc表等。
  - 当提示索引表损坏时，建议使用reindex命令进行重建索引操作。
- 检测表分区的数据文件。
```
{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE} table_name PARTITION {(partition_name)}
[CASCADE];
```
  - 支持对表的单独分区进行检测操作，但不支持对索引表index使用CASCADE操作。
  - 不支持HDFS表（内表和外表），不支持临时表和unlog表。

参数说明

表 21-295 ANALYZE | ANALYSE 参数说明

参数	描述	取值范围
VERBOSE	启用VERBOSE后，ANALYZE会发出进度信息，表明目前正在处理的表。各种有关表的统计信息也会打印出来。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
LIGHT	轻量化模式下对表收集的统计信息保存到内存中，不写入系统表，执行时对表加一级锁，该选项请加括号使用，例如ANALYZE (LIGHT) t1	-
FORCE	FORCE模式支持表的统计信息被锁定的情况下进行强制执行ANALYZE，该选项请加括号使用，例如ANALYZE (FORCE) t1。	-
PREDICATE	PREDICATE模式表示只计算当前识别到的谓词列的统计信息，该参数仅9.1.0.100及以上集群版本支持。是否支持谓词列的ANALYZE操作由GUC参数analyze_predicate_column_threshold控制，取值为0表示关闭，大于0表示开启谓词列收集功能，且仅对列数大于等于此取值的表进行谓词列ANALYZE。 该选项请加括号使用，例如ANALYZE (PREDICATE) xxx。 谓词信息是在查询解析阶段收集，动态采样也支持谓词列采样。	-
table_name	需要分析的特定表的表名（可能会带模式名），如果省略，将对数据库中的所有表（非外部表）进行分析。 对于ANALYZE收集统计信息，目前仅支持行存表、列存表、HDFS表、ORC格式的OBS外表、CARBONDATA格式的OBS外表、协同分析的外表。	表名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。
column_name	需要分析特定列的列名，默认为所有列。	已有的列名，格式为：column_1_name，column_2_name，column_3_name....
partition_name	表示分析该分区表的统计信息。目前语法上支持分区表做ANALYZE，但功能实现上暂不支持对指定分区统计信息的分析。	字符串，要符合分区名的命名规范。
foreign_table_name	需要分析的特定外表的表名（可能会带模式名），该表的数据存放于HDFS分布式文件系统中。	字符串，要符合外表的命名规范。
FOREIGN TABLES	分析所有当前用户权限下，数据位于HDFS分布式文件系统中的HDFS外表。	字符串，要符合HDFS外表的命名规范。
index_name	需要分析的特定索引表的表名（可能会带模式名）。	字符串，要符合索引表的命名规范。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
FAST| COMPLETE	不同模式下，对行存表和列存表进行校验。	● 行存表 – FAST模式下，主要对于行存表的CRC和page header进行校验，如果校验失败则会告警; – COMPLETE模式下，主要对行存表的指针、tuple进行解析校验。 ● 列存表 – FAST模式下，主要对于列存表的CRC和magic进行校验，如果校验失败则会告警; – COMPLETE模式下，主要对列存表的CU进行解析校验。
CASCADE	CASCADE模式下会对当前表的所有索引进行检测处理。	-

示例

- 使用ANALYZE语句更新表customer_info统计信息：
  ```
  ANALYZE customer_info;
  ```
- 使用ANALYZE VERBOSE语句更新表customer_info统计信息，并输出表的相关信息：
  ```
  ANALYZE VERBOSE customer_info;
  INFO:  analyzing "cstore.pg_delta_3394584009"(cn_5002 pid=53078)
  INFO:  analyzing "public.customer_info"(cn_5002 pid=53078)
  INFO:  analyzing "public.customer_info" inheritance tree(cn_5002 pid=53078)
  ANALYZE
  ```
- 更新表customer_info中特定列name，email的统计信息：
  ```
  ANALYZE customer_info (name, email);
  ```
- 使用轻量化模式更新表customer_info的统计信息。
  ```
  ANALYZE (LIGHT) customer_info;
  ```
- 使用强制模式更新表customer_info的统计信息，即使表已被锁定。
  ```
  ANALYZE (FORCE) customer_info;
  ```
- 只更新谓词列的统计信息，只会对当前为止收集到的谓词列进行采样，最后在查询业务稳定后使用。
  ```
  ANALYZE (PREDICATE) customer_info;
  ```
  通过函数**pg_stat_get_predicate_columns**可以查询当前已收集了表的哪些谓词列，注意以下示例的表名为“customer_info”，实际执行时请替换表名。

版权所有 © 华为云计算技术有限公司

```
SELECT pr.attnum,pa.attname FROM
pg_catalog.pg_stat_get_predicate_columns('customer_info'::regclass) pr LEFT JOIN pg_attribute pa ON
pa.attrelid='customer_info'::regclass AND pa.attnum = pr.attnum;
```

21.15.4 DEALLOCATE

功能描述

DEALLOCATE用于删除先前编写的预备语句。如果未显式删除一个预备语句，则在会话结束时将其删除。

有关预备语句可参考**PREPARE**。

注意事项

无。

语法格式

```
DEALLOCATE [ PREPARE ] { name | ALL };
```

参数说明

表 21-296 DEALLOCATE 参数说明

参数	描述	取值范围
PREPARE	创建一个预备语句，可选关键字，对操作没有影响。	-
name	将要删除的预备语句的名称。	字符串，需符合预备语句的命名规范。
ALL	删除所有预备语句。	字符串，需符合预备语句的格式。

示例

无。

21.15.5 DO

功能描述

执行匿名代码块。

代码块被视为没有参数的函数主体，返回值类型是void。它的解析和执行是同一时刻发生的。

注意事项

- 程序语言在使用之前，必须通过命令CREATE LANGUAGE安装到当前的数据库中。 plpgsql是默认的安装语言，其它语言安装时必须指定。

版权所有 © 华为云计算技术有限公司

- 如果语言是不受信任的，用户必须有使用程序语言的USAGE权限，或者是系统管理员。

语法格式

```
DO [ LANGUAGE lang_name ] code;
```

参数说明

表 21-297 DO 参数说明

参数	描述	取值范围
lang_name	用来解析代码的程序语言的名字。	符合规范的程序语言，默认的语言是plpgsql。
code	可以被执行的程序语言代码。	字符串格式。

示例

授予用户webuser对模式tpcds下视图的所有操作权限：

```
DO $$DECLARE r record;
BEGIN
   FOR r IN SELECT c.relname,n.nspname FROM pg_class c,pg_namespace n
         WHERE c.relnamespace = n.oid AND n.nspname = 'tpcds' AND relkind IN ('r','v')
   LOOP
      EXECUTE 'GRANT ALL ON ' || quote_ident(r.table_schema) || '.' || quote_ident(r.table_name) || ' TO
webuser';
   END LOOP;
END$$;
```

21.15.6 EXECUTE

功能描述

执行一个前面准备好的预备语句。由于预备语句只在会话的生命期里存在，那么该预备语句必须在当前会话中由一个更早执行的PREPARE语句创建。

注意事项

如果创建预备语句的PREPARE语句声明了一些参数，那么传递给EXECUTE语句的必须是一个兼容的参数集，否则就会生成一个错误。

语法格式

```
EXECUTE name [ ( parameter [, ...] ) ];
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-298 EXECUTE 参数说明

参数	描述	取值范围
name	要执行的预备语句的名字。	字符串，需符合预备语句的命名规范。
parameter	给预备语句的一个参数的具体数值。必须是一个生成与创建预备语句时指定参数的数据类型相兼容的值的表达式。	字符串，与指定参数的数据类型相兼容的值的表达式。

示例

为一个INSERT语句创建一个预备语句，然后执行它：

```
PREPARE insert_reason(integer,character(16),character(100)) AS INSERT INTO tpcds.reason_t1
VALUES($1,$2,$3);
EXECUTE insert_reason(52, 'AAAAAAAADDAAAAAA', 'reason 52');
```

21.15.7 EXECUTE DIRECT

功能描述

在指定的节点上执行SQL语句。一般情况下，SQL语句的执行是由集群负载自动分配到合适的节点上，execute direct主要用于数据库维护和测试。

注意事项

- 只有系统管理员才能执行EXECUTE DIRECT。
- 为了各个节点上数据的一致性，SQL语句仅支持SELECT，不允许执行事务语句、DDL、DML。
- 使用此类型语句在指定的DN执行AVG聚集计算时，返回结果集是以数组形式返回，如{4,2}，表示sum结果为4，count结果为2。
- 由于CN节点不存储用户表数据，不允许指定CN节点执行用户表上的SELECT查询。
- 不允许执行嵌套的EXECUTE DIRECT语句，即执行的SQL语句不能同样是EXECUTE DIRECT语句，此时可直接执行最内层EXECUTE DIRECT语句代替。

语法格式

```
EXECUTE DIRECT ON ( nodename [, ... ] ) query ;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-299 EXECUTE DIRECT 参数说明

参数	描述	取值范围
nodename	节点名称。	字符串，已存在的节点。
query	要执行查询语句。	字符串，可以执行的SQL语句。

示例

查询节点dn_6001_6002上tpcds.customer_address记录：

```
EXECUTE DIRECT ON(dn_6001_6002) 'select count(*) from tpcds.customer_address';
 count
-------
 16922
(1 row)
```

21.15.8 GRANT

功能描述及注意事项

对角色和用户进行授权操作。

使用GRANT命令进行用户授权包括以下三种场景：

将系统权限(SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN 和 LOGIN)授权给角色或用户

系统权限又称为用户属性，包括SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN和LOGIN。

表 21-300 系统权限

系统权限	权限描述	是否支持通过GRANT/REVOKE授权或撤销
SYSADMIN	支持查看特殊系统表。	是
CREATEDB	支持创建数据库DATABASE。	否
CREATEROLE	支持创建用户与角色。	
AUDITADMIN	支持查看审计日志。	
LOGIN	支持连接数据库。	

系统权限一般在创建或修改角色阶段，通过CREATE/ALTER ROLE语法来指定。

其中，SYSADMIN权限可以通过GRANT/REVOKE ALL PRIVILEGE授予或撤销。

版权所有 © 华为云计算技术有限公司

但系统权限无法通过ROLE和USER的权限被继承，也无法授予PUBLIC。

将数据库对象(表和视图、指定字段、数据库、函数、模式等)授权给角色或用户

将数据库对象（表和视图、指定字段、数据库、函数、模式等）的相关权限授予特定角色或用户。

- **不允许对表分区进行GRANT操作**，否则会引起告警。
- 如果要重命名一个对象，用户除了必须是该对象的所有者外，还必须拥有该对象所在模式的CREATE权限。
- **将模式中的表或者视图对象授权给其他用户时，需要将表或视图所属的模式的USAGE权限同时授予该用户，**若没有该权限，则只能看到这些对象的名字，并不能实际进行对象访问。
- GRANT命令将数据库对象的特定权限授予一个或多个角色。这些权限会追加到已有的权限上。
- **关键字PUBLIC**表示该权限要赋予所有角色，包括以后创建的用户。PUBLIC可以看做是一个隐含定义好的组，它总是包括所有角色。任何角色或用户都将拥有通过GRANT直接赋予的权限和所属的权限，再加上PUBLIC的权限。
- 如果声明了WITH GRANT OPTION，则被授权的用户也可以将此权限赋予他人，否则就不能授权给他人。这个选项不能赋予PUBLIC，这是DWS特有的属性。
- DWS会将某些类型的对象上的权限授予PUBLIC。默认情况下，对表、表字段、序列、外部数据源、外部服务器、模式或表空间对象的权限不会授予PUBLIC，而以下这些对象的权限会授予PUBLIC：数据库的CONNECT权限和CREATE TEMP TABLE权限、函数的EXECUTE特权、语言和数据类型（包括域）的USAGE特权。对象拥有者可以撤销默认授予PUBLIC的权限并专门授予权限给其他用户。为了更安全，建议在同一个事务中创建对象并设置权限，这样其他用户就没有时间窗口使用该对象。另外，这些初始的默认权限可以使用ALTER DEFAULT PRIVILEGES命令修改。

将角色或用户的权限授权给其他角色或用户

将一个角色或用户的权限授予一个或多个其他角色或用户。在这种情况下，每个角色或用户都可视为拥有一个或多个数据库权限的集合。

- 当声明了WITH ADMIN OPTION，被授权的用户可以将该权限再次授予其他角色或用户，以及撤销所有由该角色或用户继承到的权限。
- 当授权的角色或用户发生变更或被撤销时，所有继承该角色或用户权限的用户拥有的权限都会随之发生变更。
- 数据库系统管理员可以给任何角色或用户授予/撤销任何权限。
- 拥有CREATEROLE权限的角色可以赋予或者撤销任何非系统管理员角色的权限。

权限分类

表 21-301 GRANT 的权限分类

参数	描述
SELECT	允许对指定的表、视图、序列执行SELECT语句。
INSERT	允许对指定的表执行INSERT语句。

 版权所有 © 华为云计算技术有限公司

参数	描述
UPDATE	允许对声明的表中任意字段执行UPDATE语句。SELECT… FOR UPDATE和SELECT… FOR SHARE除了需要SELECT权限外，还需要UPDATE权限。
DELETE	允许执行DELETE语句删除指定表中的数据。
TRUNCATE	允许执行TRUNCATE语句删除指定表中的所有记录。
REFERENCES	创建一个外键约束，必须拥有参考表和被参考表的REFERENCES权限。
TRIGGER	创建一个触发器，必须拥有表或视图的TRIGGER权限。
ANALYZE | ANALYSE	对表执行ANALYZE | ANALYSE操作来收集表的统计信息，必须拥有表的ANALYZE | ANALYSE权限。
CREATE	● 对于数据库，允许在数据库里创建新的模式。 ● 对于模式，允许在模式中创建新的对象。如果要重命名一个对象，用户除了必须是该对象的所有者外，还必须拥有该对象所在模式的CREATE权限。 ● 对于子集群，允许在子集群中创建表对象。
CONNECT	允许用户连接到指定的数据库。
TEMPORARY | TEMP	允许在使用指定数据库时创建临时表。
EXECUTE	允许使用指定的函数，以及利用这些函数实现的操作符。
USAGE	● 对于过程语言，允许用户在创建函数的时候指定过程语言。 ● 对于模式，USAGE允许访问包含在指定模式中的对象，若没有该权限，则只能看到这些对象的名字。 ● 对于序列，USAGE允许使用nextval函数。 ● 对于子集群， 对包含在指定模式中的对象有访问权限时，USAGE允许访问指定子集群下的表对象。
COMPUTE	针对计算子集群，允许用户在具有compute权限的计算子集群上进行弹性计算。
ALTER	允许修改表、模式或函数。
DROP	允许删除表、模式或函数。
VACUUM	允许对表执行VACUUM。
ALL PRIVILEGES	一次赋予指定用户/角色所有可赋予的权限。只有系统管理员有权执行GRANT ALL PRIVILEGES。

版权所有 © 华为云计算技术有限公司

参数	描述
WITH GRANT OPTION	如果声明了WITH GRANT OPTION，则被授权的用户也可以将此权限赋予他人，否则就不能授权给他人。这个选项不能赋予PUBLIC。 • NODE GROUP不支持WITH GRANT OPTION功能。 • 使用WITH GRANT OPTION时需确保security_enable_options参数值中设置了grant_with_grant_option。
WITH ADMIN OPTION	指定权限是否允许转授。如果声明了WITH ADMIN OPTION，角色的成员又可以将角色的成员身份授予其他人。

语法格式

- 将**表或视图的访问权限**赋予指定的用户或角色。不允许对表分区进行GRANT操作，对表分区进行GRANT操作会引起告警。

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | ANALYZE |
ANALYSE | VACUUM | ALTER | DROP } [, ...]
    | ALL [ PRIVILEGES ] }
  ON { [ TABLE ] table_name [, ...]
    | ALL TABLES IN SCHEMA schema_name [, ...] }
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

⚠ 注意

避免对同一张表或视图（包括ON TABLE和ON ALL TABLES IN SCHEMA）执行并发GRANT/REVOKE操作，或者对同一张表或视图同时执行GRANT/REVOKE操作和DDL操作，否则会出现类似报错“tuple concurrently updated.”。

- 将**表中字段的访问权限**赋予指定的用户或角色。

```
GRANT { {{ SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )} [, ...]
    | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
  ON [ TABLE ] table_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

- 将**模式的访问权限**赋予指定的用户或角色。

```
GRANT { { CREATE | USAGE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
  ON SCHEMA schema_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

⚠ 注意

将模式中的表或者视图对象授权给其他用户时，需要将表或视图所属的模式的USAGE权限同时授予该用户，若没有该权限，则只能看到这些对象的名字，并不能实际进行对象访问。

- 将**数据库的访问权限**赋予指定的用户或角色。

版权所有 © 华为云计算技术有限公司

```
GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...]
    | ALL [ PRIVILEGES ] }
  ON DATABASE database_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

- 将**角色的权限**赋予其他用户或角色的语法。

```
GRANT role_name [, ...]
  TO role_name [, ...]
  [ WITH ADMIN OPTION ];
```

- 将**所有权限**赋予指定的角色。

```
GRANT ALL { PRIVILEGES | PRIVILEGE }
  TO role_name;
```

- 将域的访问权限赋予指定的用户或角色。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
  ON DOMAIN domain_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

说明

当前版本暂时不支持赋予域的访问权限。

- 将外部数据源的访问权限赋予给指定的用户或角色。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
  ON FOREIGN DATA WRAPPER fdw_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

- 将外部服务器的访问权限赋予给指定的用户或角色。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
  ON FOREIGN SERVER server_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

- 将函数的访问权限赋予给指定的用户或角色。

```
GRANT { { EXECUTE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
  ON { FUNCTION {function_name ( [ {[ argmode ] [ arg_name ] arg_type} [, ...] ] )} [, ...]
    | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

- 将过程语言的访问权限赋予给指定的用户或角色。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
  ON LANGUAGE lang_name [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

说明

当前版本暂时不支持过程语言。

- 将大对象的访问权限赋予指定的用户或角色。

```
GRANT { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }
  ON LARGE OBJECT loid [, ...]
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```

说明

当前版本暂时不支持大对象。

- 将序列的访问权限赋予指定的用户或角色。

```
GRANT { { SELECT | UPDATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
  ON { SEQUENCE sequence_name [, ...]
    | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
  TO { [ GROUP ] role_name | PUBLIC } [, ...]
  [ WITH GRANT OPTION ];
```


版权所有 © 华为云计算技术有限公司

- 将子集群的访问权限赋予指定的用户或角色。普通用户不能执行针对Node Group的GRANT/REVOKE操作。

```
GRANT { CREATE | USAGE | COMPUTE | ALL [ PRIVILEGES ] }
    ON NODE GROUP group_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ];
```

- 将类型的访问权限赋予指定的用户或角色。

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON TYPE type_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ];
```

说明

当前版本暂时不支持赋予类型的访问权限。

参数说明

表 21-302 GRANT 的参数说明

参数	描述	取值范围
role_name	已存在用户名称，可通过PG_USER视图查询用户列表。	字符串，要符合标识符命名规范。最多为63个字符。
table_name	已存在表名称，可通过PG_TABLES查询表名称。	字符串，长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。
column_name	已存在字段名称，可通过以下语句查询表字段名称。 SELECT * FROM PG_GET_TABLEDEF('table name');	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
schema_name	已存在模式名称，可通过PG_NAMESPACE系统表查询模式名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
database_name	已存在数据库名称，可通过PG_DATABASE系统表查询数据库名称。	字符串，要符合标识符的命名规范：数据库名称是由大小写英文字母、数字、_和$组合而成，并且不能以数字和$开头。
function_name	已存在函数名称，可通过PG_PROC系统表查询函数名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
sequence_name	已存在序列名称，可通PG_SEQUENCES系统视图（仅9.1.0及以上集群支持）查询序列名称。	字符串，仅可以使用小写字母（a~z）、 大写字母（A~Z），数字和特殊字符"#"，"_"，"$"的组合。
domain_name	已存在域类型名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围
fdw_name	已存在外部数据包名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
lang_name	已存在语言名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
type_name	已存在类型名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
group_name	已存在的子集群名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
argmode	函数的参数模式。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
arg_name	函数的参数名称。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
arg_type	函数的参数类型。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。
loid	包含本页的大对象的标识符。	字符串，要符合标识符命名规范。详情请参见**标识符命名规范**。

示例

- **将系统权限授权给用户或者角色。**
  - 将sysadmin所有可用权限授权给joe用户：
    ```
    GRANT ALL PRIVILEGES TO joe;
    ```
    授权成功后，用户joe会拥有sysadmin的所有权限。
- **将对象权限授权给用户或者角色。**
  - 将表tpcds.reason的SELECT权限授权给用户joe：
    ```
    GRANT SELECT ON TABLE tpcds.reason TO joe;
    ```
  - 将表tpcds.reason的所有权限授权给用户kim：
    ```
    GRANT ALL PRIVILEGES ON tpcds.reason TO kim;
    ```
    授权成功后，kim用户就拥有了tpcds.reason表的所有权限，包括增删改查等权限。
  - 将模式tpcds的使用权限授权给用户joe：
    ```
    GRANT USAGE ON SCHEMA tpcds TO joe;
    ```
    授权成功后，joe用户就拥有了模式schema的USAGE权限，允许访问包含在指定模式schema中的对象。

版权所有 © 华为云计算技术有限公司

  - 将tpcds.reason表中r_reason_sk、r_reason_id、r_reason_desc列的查询权限，r_reason_desc的更新权限授权给joe：
    ```
    GRANT select (r_reason_sk,r_reason_id,r_reason_desc),update (r_reason_desc) ON tpcds.reason TO joe;
    ```
    授权成功后，用户joe对tpcds.reason表中r_reason_sk，r_reason_id的查询权限会立即生效。
    ```
    GRANT select (r_reason_sk, r_reason_id) ON tpcds.reason TO joe ;
    ```
  - 将函数func_add_sql的EXECUTE权限授权给用户joe。
    ```
    CREATE FUNCTION func_add_sql(f1 integer,f2 integer) RETURNS integer
        AS 'select $1 + $2;'
        LANGUAGE SQL
        IMMUTABLE
        RETURNS NULL ON NULL INPUT;
    GRANT EXECUTE ON FUNCTION func_add_sql(integer, integer) TO joe;
    ```
  - 将序列serial的UPDATE权限授权给joe用户。
    ```
    GRANT UPDATE ON SEQUENCE serial TO joe;
    ```
  - 将数据库gaussdb的连接权限授权给用户joe，并给予其在gaussdb中创建schema的权限：
    ```
    GRANT create,connect on database gaussdb TO joe ;
    ```
  - 将模式tpcds的访问权限授权给角色tpcds_manager，并授予该角色在tpcds下创建对象的权限，不允许该角色中的用户将权限授权给其他人：
    ```
    GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds_manager;
    ```

- **将用户或者角色的权限授权给其他用户或角色。**
  - 将用户joe的权限授权给用户manager，并允许该角色将权限授权给其他人：
    ```
    GRANT joe TO manager WITH ADMIN OPTION;
    ```
  - 将用户manager的权限授权给senior_manager用户：
    ```
    GRANT manager TO senior_manager;
    ```

相关链接

REVOKE，**ALTER DEFAULT PRIVILEGES**

21.15.9 PREPARE

功能描述

创建一个预备语句。

预备语句是服务端的对象，可以用于优化性能。在执行PREPARE语句的时候，指定的查询被解析、分析、重写。当随后发出EXECUTE语句的时候，预备语句被规划和执行。这种设计避免了重复解析、分析工作。PREPARE语句创建后在整个数据库会话期间一直存在，一旦创建成功，即便是在事务块中创建，事务回滚，PREPARE也不会删除。只能通过显式调用**DEALLOCATE**进行删除，会话结束时，PREPARE也会自动删除。

注意事项

无。

语法格式

```
PREPARE name [ ( data_type [, ...] ) ] AS statement;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-303 PREPARE 参数说明

参数	描述	取值范围
name	指定预备语句的名称。该名称必须在该会话中是唯一的。	字符串。
data_type	参数的数据类型。	-
statement	可以执行的SQL语句。	任何SELECT INSERT、UPDATE、DELETE或VALUES语句。

示例

为一个INSERT语句创建一个预备语句，然后执行它：

```
PREPARE insert_reason(integer,character(16),character(100)) AS INSERT INTO tpcds.reason_t1
VALUES($1,$2,$3);
EXECUTE insert_reason(52, 'AAAAAAAADDAAAAAA', 'reason 52');
```

相关链接

DEALLOCATE

21.15.10 REASSIGN OWNED

功能描述

修改数据库对象的属主。

REASSIGN OWNED要求系统将所有old_role拥有的数据库对象的属主更改为new_role。

注意事项

- REASSIGN OWNED常用于在删除角色之前的准备工作。由于不会影响其他数据库中的对象， 因此通常需要在每个数据库中执行此命令，该数据库包含要删除的角色所拥有的对象。
- 执行REASSIGN OWNED需要有原角色和目标角色上的权限。
- 资源管理不对该语法的数据切换进行监控，需调用select gs_wlm_readjust_user_space(0)手动校准监控数据。

语法格式

```
REASSIGN OWNED BY old_role [, ...] TO new_role;
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-304 REASSIGN OWNED 参数说明

参数	描述	取值范围
old_role	旧属主的角色名。	字符串，角色名称长度不超过63个字符，以字母或下划线开头，中间字符可以是字母、数字、下划线、$。
new_role	将要成为对象属主的新角色的名字。	字符串，角色名称长度不超过63个字符，以字母或下划线开头，中间字符可以是字母、数字、下划线、$。

示例

将由joe和jack角色拥有的所有数据库对象重新分配给admin：

```
REASSIGN OWNED BY joe, jack TO admin;
```

21.15.11 REVOKE

功能描述

REVOKE用于撤销一个或多个角色的权限。

注意事项

非对象所有者试图在对象上REVOKE权限，命令按照以下规则执行：

- 如果授权用户没有该对象上的权限，则命令立即失败。
- 如果授权用户有部分权限，则只撤销那些有授权选项的权限。
- 如果授权用户没有授权选项，REVOKE ALL PRIVILEGES形式将发出一个错误信息，而对于其他形式的命令而言，如果是命令中指定名字的权限没有相应的授权选项，该命令将发出一个警告。
- 不允许对表分区进行REVOKE操作，对分区表进行REVOKE操作会引起告警。

语法格式

- 撤销指定表和视图上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | ANALYZE |
ANALYSE | VACUUM | ALTER | DROP }[, ...]
    | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
        | ALL TABLES IN SCHEMA schema_name [, ...] }
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```


版权所有 © 华为云计算技术有限公司

注意

避免对同一张表或视图（包括ON TABLE和ON ALL TABLES IN SCHEMA）执行并发GRANT/REVOKE操作，或者对同一张表或视图同时执行GRANT/REVOKE操作和DDL操作，否则会出现类似报错“tuple concurrently updated.”。

- 撤销表上指定字段权限。

```
REVOKE [ GRANT OPTION FOR ]
    { {{ SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )}[, ...]
    | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定数据库上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...]
    | ALL [ PRIVILEGES ] }
    ON DATABASE database_name [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定函数上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { EXECUTE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
    ON { FUNCTION {function_name ( [ {[ argmode ] [ arg_name ] arg_type} [, ...] ] )} [, ...]
        | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定大对象上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }
    ON LARGE OBJECT loid [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定序列上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | UPDATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
    ON SEQUENCE sequence_name [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定模式上权限。

```
REVOKE [ GRANT OPTION FOR ]
    { { CREATE | USAGE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定子集群上权限

```
REVOKE [ GRANT OPTION FOR ]
    { CREATE | USAGE | COMPUTE | ALL [ PRIVILEGES ] }
    ON NODE GROUP group_name [, ...]
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ];
```

- 按角色撤销角色上的权限。

```
REVOKE [ ADMIN OPTION FOR ]
    role_name [, ...] FROM role_name [, ...]
    [ CASCADE | RESTRICT ];
```

- 撤销指定角色的sysadmin权限。

```
REVOKE ALL { PRIVILEGES | PRIVILEGE } FROM role_name;
```


版权所有 © 华为云计算技术有限公司

参数说明

关键字PUBLIC表示一个隐式定义的拥有所有角色的组。REVOKE权限类别和参数说明，请参见GRANT的**参数说明**。

说明

- 任何特定角色拥有的特权包括直接授予该角色的特权、从该角色作为其成员的角色中得到的权限以及授予给PUBLIC的权限。因此，从PUBLIC收回SELECT特权并不一定会意味着所有角色都会失去在该对象上的SELECT特权，那些直接被授予的或者通过另一个角色被授予的角色仍然会拥有它。类似地，从一个用户收回SELECT后，如果PUBLIC仍有SELECT权限，该用户还是可以使用SELECT。
- 依赖性权限：如用户A拥有某个表的UPDATE权限，及WITH GRANT OPTION选项，同时A把这个权限赋予了用户B，则用户B持有的权限称为依赖性权限。当用户A持有的权限或者授权选项被撤销时，依赖性权限仍然存在，但如果声明了CASCADE，则所有依赖性权限都被撤销。
- 一个用户只能撤销由它自己直接赋予的权限。例如，如果用户A被指定授权（WITH ADMIN OPTION）选项，且把一个权限赋予了用户B，然后用户B又赋予了用户C，则用户A不能直接将C的权限撤销。但是，用户A可以撤销用户B的授权选项，并且使用CASCADE。这样，用户C的权限就会自动被撤销。同样，如果A和B都赋予了C同样的权限，则A可以撤销他自己的授权选项，但是不能撤销B的，因此C仍然拥有该权限。
- 如果执行REVOKE的角色持有的权限是通过多层成员关系获得的，则具体是哪个包含的角色执行的该命令是不确定的。在这种场合下，最好的方法是使用SET ROLE成为特定角色，然后执行REVOKE，否则可能导致删除了不想删除的权限，或者是任何权限都没有删除。

示例

撤销joe用户的sysadmin权限。

```
REVOKE ALL PRIVILEGES FROM joe;
```

撤销指定模式上授予的权限。

```
REVOKE USAGE,CREATE ON SCHEMA tpcds FROM tpcds_manager;
```

撤销joe用户的CONNECT特权。

```
REVOKE CONNECT ON DATABASE mydatabase FROM joe;
```

从用户joe撤销角色admins中的成员资格。

```
REVOKE admins FROM joe;
```

撤销用户joe对视图myView具有的所有特权。

```
REVOKE ALL PRIVILEGES ON myView FROM joe;
```

撤销针对表customer_t1的public插入特权。

```
REVOKE INSERT ON customer_t1 FROM PUBLIC;
```

撤销用户joe对模式tpcds的使用权限。

```
REVOKE USAGE ON SCHEMA tpcds FROM joe;
```

撤销用户joe对tpcds.reason表中r_reason_sk，r_reason_id的查询权限。

```
REVOKE select (r_reason_sk, r_reason_id) ON tpcds.reason FROM joe;
```

撤销用户joe对函数func_add_sql的执行权限。

```
REVOKE execute ON FUNCTION func_add_sql(integer, integer) FROM joe CASCADE;
```


版权所有 © 华为云计算技术有限公司

相关链接

GRANT

21.16 DQL 语法

21.16.1 DQL 语法一览表

DQL（Data Query Language数据查询语言），用于从表或视图中获取数据。

查询

DWS提供了用于从表或视图中获取数据的语句，请参考**SELECT**。

根据查询结果定义一个新表。

DWS提供了根据查询结果创建一个新表，并且将查询到的数据插入到新表中的语句，请参考**SELECT INTO**。

21.16.2 SELECT

功能描述

SELECT用于从表或视图中读取数据。

SELECT语句就像叠加在数据库表上的过滤器，利用SQL关键字从数据表中过滤出用户需要的数据。

注意事项

- 使用SELECT请遵循SELECT操作规范，参见《数据仓库服务开发指南》的“SELECT操作规范”章节。
- SELECT支持普通表与HDFS之间的JOIN，不支持普通表与GDS外表之间的JOIN。即SELECT语句中不能同时出现普通表和GDS外表。
- 需确认对每个在SELECT命令中使用的字段拥有SELECT权限，使用FOR UPDATE或FOR SHARE还要求拥有UPDATE权限。

语法格式

主句语法格式如下，其中包含的FROM子句、GROUP BY子句中的语法格式单独定义，如需了解各子句的含义，请参见**FROM子句参数说明**和**GROUP BY子句参数说明**。

说明

另外，SELECT支持简化版查询语法，功能相当于select * from table_name。

```
TABLE { ONLY {(table_name)| table_name} | table_name [ * ]};
```

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [/*+ plan_hint */] [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
{ * | {expression [ [ AS ] output_name ]} [, ...] }
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY grouping_element [, ...] ]
[ HAVING condition [, ...] ]
```


版权所有 © 华为云计算技术有限公司

```
[ WINDOW {window_name AS ( window_definition )} [, ...] ]
[ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]
[ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort_expression_clause ] [ NULLS { FIRST |
LAST } ]} [, ...] ]
[ { [ LIMIT { count | ALL } ] [ OFFSET start [ ROW | ROWS ] ] } | { LIMIT start, { count | ALL } } ]
[ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]
[ {FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ]} [...] ];
```

须知

condition和expression中可以使用输出列中表达式的别名。

- 只能同一层引用。
- 只能引用输出列中的别名。
- 只能是后面的表达式引用前面的表达式。
- 不能包含volatile函数。
- 不能包含Window function函数。
- 不支持在join on条件中引用别名。
- 输出列中有多个要应用的别名则报错。

版权所有 © 华为云计算技术有限公司

SELECT 主句参数说明

表 21-305 SELECT 参数说明

参数	描述	取值范围或示例
WITH [RECURSIVE] with_query [, ...]	用于声明一个或多个在主查询中通过名字引用的子查询（公共表达式CTE），相当于临时表，当需要多次引用相同子查询结果时，使用WITH可简化代码逻辑，提高编码效率。 如果声明了RECURSIVE，则允许递归查询，即允许SELECT子查询通过名字引用它自己。 其中with_query的详细格式为： with_query_name [(column_name [, ...])] AS [[NOT] MATERIALIZED] ({select | values | insert | update | delete})	● with_query_name指定子查询生成的结果集名称，可使用该名称访问子查询的结果集。 ● [NOT] MATERIALIZED，指定该参数可改变默认行为： – 指定MATERIALIZED时，将子查询执行一次，并将其结果集进行物化，供主查询多次查询其结果集。 – 指定NOT MATERIALIZED时，with_query只被主查询引用一次，则不再单独执行，而是将其子查询替换到主查询中的引用处，随主查询一起执行。 – 如下几种情况会忽略NOT MATERIALIZED： – 子查询中含有volatile函数。 – 子查询为含有FOR UPDATE/FOR SHARE的SELECT/

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
		VALUES语句。 – 子查询为INSERT/UPDATE/DELETE等语句。 – with_query为RECURSIVE。 – 被引用次数大于1的with_query2引用了外层自引用的with_query1，则with_query2不能被替换到引用处。例如下面示例中，tmp2被引用了两次，tmp2因为引用了外层自引用的tmp1，所以即使tmp2指定了NOT MATERIALIZED也会被物化。 with recursive tmp1(b) as (values(1) union all (with tmp2 as not materialized (select * from tmp1) select tt1.b + tt2.b from tmp2 tt1,

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
		tmp2 tt2)) select * from tmp1; • column_name指定子查询结果集中显示的列名。 • 每个子查询可以是SELECT，VALUES，INSERT，UPDATE或DELETE语句。
plan_hint子句	以/*+ */的形式在SELECT关键字后，用于对SELECT对应的语句块生成的计划进行hint调优 ，详细用法请参见《数据仓库服务开发指南》中的“DWS性能调优>SQL调优>使用Plan Hint进行调优”章节。	-
ALL	声明返回所有符合条件的行，是默认行为，可以省略该关键字。	-
DISTINCT [ON (expression [, ...])]	从SELECT的结果集中删除所有重复的行，确保结果集中每行都是唯一的。 ON (expression [, ...]) 只保留那些在给出的表达式上运算出相同结果的行当中的第一行。 **须知** DISTINCT ON表达式是使用与ORDER BY相同的规则进行解释的。除非使用了ORDER BY来保证需要的行首先出现，否则，"第一行" 是不可预测的。	-
SELECT列表	指定查询表中列名，可以是部分列或者是全部（使用通配符*表示）。 通过使用子句AS output_name可以为输出字段取个别名，这个别名通常用于输出字段的显示。	列名可以用下面几种形式表达： • 手动输入列名，多个列之间用英文逗号（,）分隔。 • 可以是FROM子句里面计算出来的字段。 • 通常关键字AS可以省略，如select 2a，则会隐式转换为 select 2 as a。
FROM子句	为SELECT声明一个或者多个源表。具体请参见**表21-306**。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
WHERE子句	WHERE子句构成一个行选择表达式，用来缩小SELECT的查询范围。	• LIKE操作符 • "(+)"操作符，不建议使用。 具体使用约束请参见**WHERE子句使用约束**。
GROUP BY子句	将查询结果按某一列或多列的值分组，值相等的为一组。	详情请参见**表21-308**。
HAVING子句	用于过滤分组（GROUP BY）后的结果，它与WHERE类似，但是通常与GROUP BY子句配合使用来过滤出特殊的组。 HAVING子句将组的一些属性与一个常数值进行比较，只有满足HAVING子句中的逻辑表达式的组才会被提取出来。	-
WINDOW子句	用于定义窗口函数，一般格式为WINDOW window_name AS（window_definition）[， ...]，window_name是可以被随后的窗口定义所引用的窗口名称，window_definition表示窗口规则，可以是以下格式： [existing_window_name] [PARTITION BY expression [, ...]] [ORDER BY expression [ASC | DESC | USING operator] [NULLS { FIRST | LAST }] [, ...]] [frame_clause] frame_clause为窗口函数定义一个窗口框架window frame，窗口函数（并非所有）依赖于框架，window frame是当前查询行的一组相关行。frame_clause可以是以下的格式： [RANGE | ROWS] frame_start [RANGE | ROWS] BETWEEN frame_start AND frame_end frame_start和frame_end可以是： UNBOUNDED PRECEDING value PRECEDING（RANGE不支持） CURRENT ROW value FOLLOWING（RANGE不支持） UNBOUNDED FOLLOWING **须知** 对列存表的查询目前只支持row_number窗口函数，不支持frame_clause。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
UNION子句	UNION计算多个SELECT语句返回行集合的并集。 UNION子句有如下约束条件： ● 除非声明了ALL子句，否则缺省的UNION结果不包含重复的行。 ● 同一个SELECT语句中的多个UNION操作符是从左向右计算的，除非用圆括弧进行了标识。 ● FOR UPDATE不能在UNION的结果或输入中声明。 一般表达式： `select_statement UNION [ALL] select_statement` ● select_statement可以是任何没有ORDER BY、LIMIT、FOR UPDATE子句的SELECT语句。 ● 如果用圆括弧包围，ORDER BY和LIMIT可以附着在子表达式里。	-
INTERSECT子句	INTERSECT计算多个SELECT语句返回行集合的交集，不含重复的记录。 INTERSECT子句有如下约束条件： ● 同一个SELECT语句中的多个INTERSECT操作符是从左向右计算的，除非用圆括弧进行了标识。 ● 当对多个SELECT语句的执行结果进行UNION和INTERSECT操作的时候，会优先处理INTERSECT。 一般形式： `select_statement INTERSECT select_statement` select_statement可以是任何没有FOR UPDATE子句的SELECT语句。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
EXCEPT子句	EXCEPT操作符计算存在于左边SELECT语句的输出而不存在于右边SELECT语句输出的行。 EXCEPT子句有如下的通用形式： `select_statement EXCEPT [ ALL ] select_statement` ● select_statement是任何没有FOR UPDATE子句的SELECT表达式。 ● EXCEPT的结果不包含任何重复的行，除非声明了ALL选项。使用ALL时，一个在左边表中有m个重复而在右边表中有n个重复的行将在结果中出现 max(m-n,0) 次。 ● 除非用圆括弧指明顺序，否则同一个SELECT语句中的多个EXCEPT操作符是从左向右计算的。EXCEPT和UNION的绑定级别相同。 ● 目前，不能给EXCEPT的结果或者任何EXCEPT的输入声明FOR UPDATE子句。	-
MINUS子句	与EXCEPT子句具有相同的功能和用法。	-
ORDER BY子句	对SELECT语句检索得到的数据进行升序或降序排序。对于ORDER BY表达式中包含多列的情况： ● 首先根据最左边的列进行排序，如果这一列的值相同，则根据下一个表达式进行比较，以此类推。 ● 如果对于所有声明的表达式都相同，则按随机顺序返回。 ● ORDER BY中排序的列必须包括在SELECT语句所检索的结果集的列中。 **须知** ● 如果未指定ORDER BY，则按数据库系统最快生成的顺序返回。 ● 可以选择在ORDER BY子句中的任何表达式之后添加关键字ASC（升序）或DESC（降序）。如果未指定，则默认使用ASC。 ● 如果要支持中文拼音排序和不区分大小写排序，需要在创建数据库时指定编码格式为UTF-8或GBK。	-

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
nlssort_expression_clause	设置排序方式，其格式为： NLSSORT (column_name, ' NLS_SORT = { SCHINESE_PINYIN_M | generic_m_ci } ') NLS_SORT：指定某字段按照特殊方式排序。目前仅支持中文拼音格式排序和不区分大小写排序。取值如下： • SCHINESE_PINYIN_M，按照中文拼音排序（目前只支持GBK字符集内的一级汉字排序）。如果要支持此排序方式，在创建数据库时需要指定编码格式为“GBK”，否则排序无效。 • generic_m_ci，不区分大小写排序。	• **示例：通过NLSSORT子句实现查询结果按中文拼音首字母进行排序** • **示例：通过NLSSORT子句实现查询结果不区分大小写排序**
[{ [LIMIT { count | ALL }] [OFFSET start [ROW | ROWS]] } | { LIMIT start, { count | ALL } }]	LIMIT子句由两个独立的Limit子句、Offset子句和一个多参Limit子句构成： LIMIT { count | ALL } OFFSET start [ROW | ROWS] LIMIT start, { count | ALL } 其中，count声明返回的最大行数，而start声明跳过前面的start行数。如果这两个参数都指定了，会计算跳过start行之后返回的count行数。多参Limit子句不可和单参的Limit子句或Offset子句共同出现。 例如： LIMIT 10 OFFSET 20; --跳过前面20行，返回后面10行，即返回第21行到第30行 LIMIT 5,10; --跳过前面5行，返回后面10行，即返回第6行到第16行	-
FETCH { FIRST | NEXT } [count] { ROW | ROWS } ONLY	FETCH子句限定返回查询结果从第一行开始的总行数。	如果不指定count，默认值为1。

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
FOR UPDATE | SHARE子句	FOR UPDATE子句将对SELECT检索出来的行进行加锁，这样避免在当前事务结束前被其它事务修改或者删除。即执行UPDATE、 DELETE、 SELECT FOR UPDATE这些行的事务将被阻塞，直到当前事务结束。 NOWAIT选项：避免操作等待其它事务提交，如果被选择的行不能立即被锁，执行SELECT FOR UPDATE NOWAIT将会立即报错。 ● 如果在FOR UPDATE/SHARE中明确指定了表名，则只有这些指定的表被锁定，其他在SELECT中使用的表将不会被锁定。否则，将锁定该命令中所有使用的表。 ● 如果FOR UPDATE/SHARE应用于一个视图或者子查询，它同样将锁定所有该视图或子查询中使用到的表。 ● 多个FOR UPDATE/SHARE子句可以用于为不同的表指定不同的锁定模式。 ● 如果一个表中同时出现（或隐含同时出现）在FOR UPDATE/SHARE子句中，则按照FOR UPDATE处理。类似的，如果影响一个表的任意子句中出现了NOWAIT，该表将按照NOWAIT处理。 ● FOR SHARE/UPDATE：是对每个检索出来的行要求一个共享锁，而不是排他锁。一个共享锁阻塞其它事务执行UPDATE、DELETE、SELECT，而不阻塞SELECT FOR SHARE。 **须知** ● 对于FOR UPDATE/SHARE，执行计划不能下推的SQL，直接返回报错信息；对于执行计划可以下推的，下推到DN执行。 ● 对列存表的查询不支持FOR UPDATE/SHARE。	-

FROM 子句参数说明

FROM from_item 子句格式如下：

```
{[ ONLY ] table_name [ * ] [ partition_clause ] [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
|( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
|with_query_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
|function_name ( [ argument [, ...] ] ) [ AS ] alias [ ( column_alias [, ...] | column_definition [, ...] ) ]
|function_name ( [ argument [, ...] ] ) AS ( column_definition [, ...] )
|from_item [ NATURAL ] join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]}
```


版权所有 © 华为云计算技术有限公司

表 21-306 FROM 子句相关参数说明

参数	描述	取值范围或示例
table_name	表名或视图名，名称前可加上模式名，如：schema_name.table_name。	字符串，要符合标识符命名规范。 具体请参见**标识符命名规范**。
alias	给表或复杂的表引用起一个临时的表别名，以便被其余的查询引用。 别名用于缩写或者在自连接中消除歧义。如果提供了别名，它就会完全隐藏表的实际名字。	字符串，要符合标识符命名规范。 具体请参见**标识符命名规范**。
column_alias	列别名。	字符串，要符合标识符命名规范。 具体请参见**标识符命名规范**。
partition_clause	PARTITION子句，查询某个分区表中相应分区的数据。格式如下： `PARTITION { ( partition_name ) | FOR ( partition_value [, ...] ) }` • partition_name：分区名。 • partition_value：指定的分区键值。在创建分区表时，如果指定了多个分区键，可以通过PARTITION FOR子句指定的这一组分区键的值，唯一确定一个分区。 **须知** 指定分区只适合普通表。	示例： `SELECT * FROM tpcds.reason_p PARTITION (P_05_BEFORE);`
subquery	FROM子句中可以出现子查询，创建一个临时表保存子查询的输出。	例如，将子查询结果取别名为e，并作为FROM数据源： `SELECT d.department_name, e.avg_salary FROM ( SELECT department_id, AVG(salary) AS avg_salary FROM employees GROUP BY department_id) AS e JOIN departments d ON e.department_id = d.id;`

版权所有 © 华为云计算技术有限公司

参数	描述	取值范围或示例
with_query_name	WITH子句同样可以作为FROM子句的源，可以通过WITH查询的名字对其进行引用。	字符串，要符合标识符命名规范。 具体请参见**标识符命名规范**。 举例，WITH子句取名为cte，并作为FROM的引用源： `WITH cte AS (SELECT id, name FROM users) SELECT * FROM cte AS user_info;`
function_name	函数名称。函数调用也可以出现在FROM子句中。	字符串，要符合标识符命名规范。 具体请参见**标识符命名规范**。
join_type	JOIN的类型。	支持五种JOIN类型： • [INNER] JOIN • LEFT [OUTER] JOIN • RIGHT [OUTER] JOIN • FULL [OUTER] JOIN • CROSS JOIN 详情请参见**表21-307**。
ON join_condition	连接条件，用于限定连接中的哪些行是匹配的。如：ON left_table.a = right_table.a。	-
USING(join_column[，...])	ON left_table.a = right_table.a AND left_table.b = right_table.b ... 的简写。要求对应的列必须同名。	-
NATURAL	NATURAL是具有相同名称的两个表的所有列的USING列表的简写。	-
from item	用于连接的查询源对象的名称。	-

版权所有 © 华为云计算技术有限公司

表 21-307 join_type 的五种类型

参数	描述
[INNER] JOIN	一个JOIN子句由两个FROM项组合而成。可使用圆括弧以决定嵌套的顺序。如果没有圆括弧，JOIN从左向右嵌套。在任何情况下，JOIN都比逗号分隔的FROM项绑定得更紧。
LEFT [OUTER] JOIN	返回笛卡尔积中所有符合连接条件的行，再加上左表中通过连接条件没有匹配到右表行中的那些行。这样，左边的行将扩展为生成表的全长，方法是在那些右表对应的字段位置填充上NULL。 需要注意的是，只在计算匹配的时候，才使用JOIN子句的条件，外层的条件是在计算完成之后施加的。
RIGHT [OUTER] JOIN	返回所有内连接的结果行，加上每个不匹配的右边行（左边用NULL扩展）。这只是一个符号上的方便，因为总是可以将其转换成一个LEFT OUTER JOIN，只需要把左边和右边的输入互换位置即可。
FULL [OUTER] JOIN	返回所有内连接的结果行，加上每个不匹配的左边行（右边用NULL扩展），再加上每个不匹配的右边行（左边用NULL扩展）。
CROSS JOIN	CROSS JOIN等效于INNER JOIN ON（TRUE），即没有被条件删除的行。这种连接类型只是符号上的方便，与简单的FROM和WHERE的效果相同。 **说明** ● 必须为INNER和OUTER连接类型声明一个连接条件，即NATURAL ON，join_condition，USING (join_column [， ...])之一。但是它们不能出现在CROSS JOIN中。 ● CROSS JOIN和INNER JOIN生成一个简单的笛卡尔积，和在FROM的顶层列出两个项的结果相同。

GROUP BY 子句参数说明

其中group子句为：

```
( )
| expression
| ( expression [, ...] )
| ROLLUP ( { expression | ( expression [, ...] ) } [, ...] )
| CUBE ( { expression | ( expression [, ...] ) } [, ...] )
| GROUPING SETS ( grouping_element [, ...] )
```


版权所有 © 华为云计算技术有限公司

表 21-308 GROUP BY 子句参数说明

参数	描述	示例

ROLLUP ({ expression | (expression [, ...]) } [, ...])	ROLLUP是计算一个有序的分组列在GROUP BY中指定的标准聚集值，然后从右到左进一步创建高层次的部分和，最后创建了累积和。一个分组能够看做一系列的分组集。例如：
	`GROUP BY ROLLUP (a,b,c)`
	等价于：
	`GROUP BY GROUPING SETS((a,b,c), (a,b), (a), ( ))`
	ROLLUP子句中的元素可以是单独的字段或表达式，也可以是使用括号包含的列表。如果是括号中的列表，产生分组集时它们必须作为一个整体。例如：
	`GROUP BY ROLLUP ((a,b), (c,d))`
	等价于：
	`GROUPING SETS ((a,b,c,d), (a,b), (c,d ), ( ))`
		示例

CUBE ({ expression | (expression [, ...]) } [, ...])	CUBE是自动对group by子句中列出的字段进行分组汇总，结果集将包含维度列中各值的所有可能组合，以及与这些维度值组合相匹配的基础行中的聚合值。CUBE从字面意思理解，就是各个维度的意思，也就是说全部组合，即聚合键中所有字段的组合的分组统计结果。
	它会为每个分组返回一行汇总信息， 用户可以使用CUBE来产生交叉表值。比如，在CUBE子句中给出三个表达式（n = 3），运算结果为$2^n = 2^3 = 8$组。 以n个表达式的值分组的行称为常规行，其余的行称为超级聚集行。例如：
	`GROUP BY CUBE (a,b,c)`
	等价于：
	`GROUP BY GROUPING SETS((a,b,c), (a,b), (a,c), (b,c), (a), (b), (c), ( ))`
	CUBE子句中的元素可以是单独的字段或表达式，也可以是使用括号包含的列表。如果是括号中的列表，产生分组集时它们必须作为一个整体。例如：
	`GROUP BY CUBE (a, (b, c), d)`
	等价于：
	`GROUP BY GROUPING SETS ((a,b,c,d), (a,b,c), (a), ( ))`
		示例

GROUPING SETS (grouping_element [, ...])	GROUPING SETS子句是GROUP BY子句的进一步扩展，它可以使用户指定多个GROUP BY选项。选项用于定义分组集，每个分组集都需要包含在单独的括号中，空白的括号（()）表示将所有数据当作一个组处理。 这样做可以通过裁剪用户不需要的数据组来提高效率。 用户可以根据需要指定所需的数据组进行查询。 **须知** 如果SELECT列表的表达式中引用了那些没有分组的字段，则会报错，除非使用了聚集函数，因为对于未分组的字段，可能返回多个数值。	示例

WHERE 子句使用约束

1. 对于WHERE子句的LIKE操作符，当LIKE中要查询特殊字符“%”、“_”、“\”的时候需要使用反斜杠“\”来进行转义。
   ```
   CREATE TABLE tt01 (id int,content varchar(50));
   ```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO tt01 values (1,'Jack say ''hello''');
INSERT INTO tt01 values (2,'Rose do 50%');
INSERT INTO tt01 values (3,'Lilei say ''world''');
INSERT INTO tt01 values (4,'Hanmei do 100%');

SELECT * FROM tt01;
 id |      content
----+-------------------
  3 | Lilei say 'world'
  4 | Hanmei do 100%
  1 | Jack say 'hello'
  2 | Rose do 50%
(4 rows)

SELECT * FROM tt01 WHERE content like '%''he%';
 id |     content
----+------------------
  1 | Jack say 'hello'
(1 row)

SELECT * FROM tt01 WHERE content like '%50\%%';
 id |   content
----+-------------
  2 | Rose do 50%
(1 row)
```

2. WHERE子句中可以通过指定"(+)"操作符的方法将表的连接关系转换为外连接。但是不建议用户使用这种用法，因为这并不是SQL的标准语法，在做平台迁移的时候可能面临语法兼容性的问题。使用"(+)"有很多限制如下：
   a. "(+)"只能出现在where子句中。
   b. 如果from子句中已经有指定表连接关系，那么"(+)"不能在where子句中使用。
   c. "(+)"只能作用在表或者视图的列上，不能作用在表达式上。
   d. 如果表A和表B有多个连接条件，那么必须在所有的连接条件中指定"(+)"，否则"(+)"将不会生效，表连接会转化成内连接，并且不给出任何提示信息。
   e. "(+)"作用的连接条件中的表不能跨查询或者子查询。如果"(+)"作用的表，不在当前查询或者子查询的from子句中，则会报错。如果"(+)"作用的对端的表不存在，则不报错，同时连接关系会转化为内连接。
   f. "(+)"作用的表达式不能直接通过"OR"连接。
   g. 如果"(+)"作用的列是和一个常量的比较关系， 那么这个表达式会成为join条件的一部分。
   h. 同一个表不能对应多个外表。
   i. "(+)"只能出现"比较表达式"，"NOT表达式"，“ANY表达式”，“ALL表达式”，“IN表达式”，“NULLIF表达式”，“IS DISTINCT FROM表达式”，“IS OF”表达式。"(+)"不能出现在其他类型表达式中，并且这些表达式中不允许出现通过“AND”和“OR”连接。
   j. "(+)"只能转化为左外连接或者右外连接，不能转化为全连接，即不能在一个表达式的两个表上同时指定"(+)"。

示例：使用 WITH 子句实现子查询

先通过子查询得到一张临时表temp_t，然后查询表temp_t中的所有数据。

```
WITH temp_t(name,isdba) AS (SELECT usename,usesuper FROM pg_user) SELECT * FROM temp_t;
```

为名为temp_t的with_query显示指定MATERIALIZED，然后查询表temp_t中的所有数据。

 版权所有 © 华为云计算技术有限公司

```
WITH temp_t(name,isdba) AS MATERIALIZED (SELECT usename,usesuper FROM pg_user) SELECT * FROM
temp_t;
```

为名为temp_t的with_query显示指定NOT MATERIALIZED，然后查询表temp_t中的所有数据。

```
WITH temp_t(name,isdba) AS NOT MATERIALIZED (SELECT usename,usesuper FROM pg_user)
 SELECT * FROM temp_t t1 WHERE name LIKE 'A%'
 UNION ALL
 SELECT * FROM temp_t t2 WHERE name LIKE 'B%';
```

示例：其他 SELECT 常用场景

查询tpcds.reason表的所有r_reason_sk记录，且去除重复。

```
SELECT DISTINCT(r_reason_sk) FROM tpcds.reason;
```

LIMIT子句示例：获取表中一条记录。

```
SELECT * FROM tpcds.reason LIMIT 1;
```

LIMIT子句示例：获取表中第三条记录。

```
SELECT * FROM tpcds.reason LIMIT 1 OFFSET 2;
```

LIMIT子句示例：获取表中前两条记录。

```
SELECT * FROM tpcds.reason LIMIT 2;
```

查询所有记录，且按字母升序排列。

```
SELECT r_reason_desc FROM tpcds.reason ORDER BY r_reason_desc;
```

通过表别名，从pg_user和pg_user_status这两张表中获取数据。

```
SELECT a.usename,b.locktime FROM pg_user a,pg_user_status b WHERE a.usesysid=b.roloid;
```

FULL JOIN子句示例：将pg_user和pg_user_status这两张表的数据进行全连接显示，即数据的合集。

```
SELECT a.usename,b.locktime,a.usesuper FROM pg_user a FULL JOIN pg_user_status b on
a.usesysid=b.roloid;
```

GROUP BY子句示例：根据查询条件过滤，并对结果进行分组。

```
SELECT r_reason_id, AVG(r_reason_sk) FROM tpcds.reason GROUP BY r_reason_id HAVING
AVG(r_reason_sk) > 25;
```

GROUP BY子句示例：通过group by别名来对结果进行分组。

```
SELECT r_reason_id AS id FROM tpcds.reason GROUP BY id;
```

UNION子句示例：将表tpcds.reason里r_reason_desc字段中的内容以W开头和以N开头的进行合并。

```
SELECT r_reason_sk, tpcds.reason.r_reason_desc
   FROM tpcds.reason
   WHERE tpcds.reason.r_reason_desc LIKE 'W%'
UNION
SELECT r_reason_sk, tpcds.reason.r_reason_desc
   FROM tpcds.reason
   WHERE tpcds.reason.r_reason_desc LIKE 'N%';
```

 版权所有 © 华为云计算技术有限公司

示例：使用递归查询（WITH RECURSIVE）查询某公司三层部门的归属关系

以某公司的三层部门的递归为例，在一个包含一级、二级、三级部门的排序不规则的表中，通过递归查询（使用关键字RECURSIVE），查询出某个部门（例如技术部）下的所有二、三级部门。

准备数据：

```
DROP TABLE IF EXISTS departments;
CREATE TABLE departments
(id varchar(10),
pid varchar(10),
dept_name varchar(50));
INSERT INTO  departments VALUES('0',null,'Head office');
INSERT INTO  departments VALUES('1','0','Technology');     -- 一级部门
INSERT INTO  departments VALUES('2','0','Marketing');
INSERT INTO  departments VALUES('3','0','HR');
INSERT INTO  departments VALUES('4','0','Business');
INSERT INTO  departments VALUES('5','0','Maintenance');
INSERT INTO  departments VALUES('1-1','1','R&D Center');   -- 二级部门
INSERT INTO  departments VALUES('1-2','1','Pre-Research');
INSERT INTO  departments VALUES('1-3','1','Network');
INSERT INTO  departments VALUES('1-4','1','Test');
INSERT INTO  departments VALUES('1-5','1','Architecture');
INSERT INTO  departments VALUES('1-1-1','1-1','Frontend');   -- 三级部门
INSERT INTO  departments VALUES('1-1-2','1-1','Backend');
INSERT INTO  departments VALUES('1-1-3','1-1','Design');
INSERT INTO  departments VALUES('2-1','2','Branding');
INSERT INTO  departments VALUES('2-2','2','Pricing');
INSERT INTO  departments VALUES('2-3','2','Training');
```

查询表的原始数据，结果显示排序不规则，很难看出三层部门之间的归属关系。

```
SELECT * FROM departments;
```

id	pid	dept_name
1	0	Technology
4	0	Business
1-2	1	Pre-Research
1-5	1	Architecture
1-1-3	1-1	Design
2-3	2	Training
0	*(Null)*	Head office
3	0	HR

此时使用WITH RECURSIVE关键字，查询出技术部下面所有的二、三级部门，可以使用如下SQL：

版权所有 © 华为云计算技术有限公司

```
WITH RECURSIVE  t1 AS (         --定义一个公共表达式名称为t1
SELECT id,pid,dept_name FROM departments WHERE id='1'   --初始化递归的根节点：查询id为1的记录，即技术部
UNION ALL         --递归合并结果（保留重复项）
SELECT            --递归部分：逐层向下查询
t2.id,            --子节点id，各一级部门名称
t2.pid,           --父节点id，即总公司
t2.dept_name          --子节点名称
FROM departments t2
INNER JOIN t1 ON t2.pid = t1.id          --通过父节点ID连接递归结果
)
SELECT id,dept_name FROM t1 ORDER BY id;       --最终查询，按ID排序显示所有二、三级部门
```

id	dept_name
1	Technology
1-1	R&D Center
1-1-1	Frontend
1-1-2	Backend
1-1-3	Design
1-2	Pre-Research
1-3	Network
1-4	Test
1-5	Architecture

以上查询结果看起来还不太直观，可以将结果再稍微改进，将各个层级展示出来。

```
WITH RECURSIVE  t1 AS (

SELECT    --初始化递归的根节点，查询id为1的记录，即技术部，显示转换为text类型
 id,
 pid,
 dept_name::text    --显式转换为text类型（为后续路径拼接做准备）
FROM departments
WHERE id='1'

UNION ALL

SELECT    --递归部分：拼接层级路径
t2.id,
t2.pid,
(t1.dept_name||'->'||t2.dept_name)::text  AS dept_name   --拼接父级路径与当前节点名称，如Technology -> R&D Center
FROM departments t2
INNER JOIN t1 ON t2.pid = t1.id )

SELECT id,dept_name FROM t1 ORDER BY id;   --最终查询，显示带层级路径的结果
```


版权所有 © 华为云计算技术有限公司

id	dept_name
1	Technology
1-1	Technology->R&D Center
1-1-1	Technology->R&D Center->Frontend
1-1-2	Technology->R&D Center->Backend
1-1-3	Technology->R&D Center->Design
1-2	Technology->Pre-Research
1-3	Technology->Network
1-4	Technology->Test
1-5	Technology->Architecture

示例：通过 NLSSORT 子句实现查询结果按中文拼音首字母进行排序

NLS_SORT子句示例：中文拼音排序。

```
DROP TABLE IF EXISTS stu_pinyin_info;
CREATE TABLE stu_pinyin_info (id bigint, name text) DISTRIBUTE BY REPLICATION;
INSERT INTO stu_pinyin_info VALUES (1, '李雷'),(2, '石祥传'), (3,'安南鹏');
SELECT * FROM stu_pinyin_info ORDER BY NLSSORT (name, 'NLS_SORT = SCHINESE_PINYIN_M' );
```

id	name
3	安南鹏
1	李雷
2	石祥传

总条数：3

示例：通过 NLSSORT 子句实现查询结果不区分大小写排序

不区分大小写排序:

```
CREATE TABLE stu_icase_info (id bigint, name text) DISTRIBUTE BY REPLICATION;
INSERT INTO stu_icase_info VALUES (1, 'aaaa'),(2, 'AAAA');
SELECT * FROM stu_icase_info ORDER BY NLSSORT (name, 'NLS_SORT = generic_m_ci');
 id | name
----+------
  1 | aaaa
  2 | AAAA
(2 rows)
```

 版权所有 © 华为云计算技术有限公司

示例：查询分区表的数据

创建分区表tpcds.reason_p，并插入数据，再从tpcds.reason_p的表分区P_05_BEFORE中获取数据。

```
CREATE TABLE tpcds.reason_p
(
  r_reason_sk integer,
  r_reason_id character(16),
  r_reason_desc character(100)
)
PARTITION BY RANGE (r_reason_sk)
(
  partition P_05_BEFORE values less than (05),
  partition P_15 values less than (15),
  partition P_25 values less than (25),
  partition P_35 values less than (35),
  partition P_45_AFTER values less than (MAXVALUE)
);

INSERT INTO tpcds.reason_p values(3,'AAAAAAAABAAAAAAA','reason 1'),
(10,'AAAAAAAABAAAAAAA','reason 2'),(4,'AAAAAAAABAAAAAAA','reason 3'),
(10,'AAAAAAAABAAAAAAA','reason 4'),(10,'AAAAAAAABAAAAAAA','reason 5'),
(20,'AAAAAAAACAAAAAAA','reason 6'),(30,'AAAAAAAACAAAAAAA','reason 7');

SELECT * FROM tpcds.reason_p PARTITION (P_05_BEFORE);
 r_reason_sk |   r_reason_id    |   r_reason_desc
-------------+------------------+----------------------------------
           4 | AAAAAAAABAAAAAAA | reason 3
           3 | AAAAAAAABAAAAAAA | reason 1
(2 rows)
——查询分区P_15的行数：
SELECT count(*) FROM tpcds.reason_p PARTITION (P_15);
 count
--------
     3
(1 row)
```

GROUP BY子句示例：按r_reason_id分组统计tpcds.reason_p表中的记录数。

```
SELECT COUNT(*),r_reason_id FROM tpcds.reason_p GROUP BY r_reason_id;
 count |   r_reason_id
-------+------------------
     2 | AAAAAAAACAAAAAAA
     5 | AAAAAAAABAAAAAAA
(2 rows)
```

HAVING子句示例：按r_reason_id分组统计tpcds.reason_p表中的记录，并只显示r_reason_id个数大于2的信息。

```
SELECT COUNT(*) c,r_reason_id FROM tpcds.reason_p GROUP BY r_reason_id HAVING c>2;
 c |   r_reason_id
---+------------------
 5 | AAAAAAAABAAAAAAA
(1 row)
```

IN子句示例：按r_reason_id分组统计tpcds.reason_p表中的r_reason_id个数，并只显示r_reason_id值为AAAAAAAABAAAAAAA或AAAAAAAADAAAAAAA的个数。

```
SELECT COUNT(*),r_reason_id FROM tpcds.reason_p GROUP BY r_reason_id HAVING r_reason_id
IN('AAAAAAAABAAAAAAA','AAAAAAAADAAAAAAA');
count |   r_reason_id
-------+------------------
     5 | AAAAAAAABAAAAAAA
(1 row)
```

INTERSECT子句示例：查询r_reason_id等于AAAAAAAABAAAAAAA，并且r_reason_sk小于5的信息。

版权所有 © 华为云计算技术有限公司

```
SELECT * FROM tpcds.reason_p WHERE r_reason_id='AAAAAAAABAAAAAAA' INTERSECT SELECT * FROM
tpcds.reason_p WHERE r_reason_sk<5;
 r_reason_sk |   r_reason_id    |      r_reason_desc
-------------+------------------+----------------------------------
           4 | AAAAAAAABAAAAAAA | reason 3
           3 | AAAAAAAABAAAAAAA | reason 1
(2 rows)
```

EXCEPT子句示例：查询r_reason_id等于AAAAAAAABAAAAAAA，并且去除r_reason_sk小于4的信息。

```
SELECT * FROM tpcds.reason_p WHERE r_reason_id='AAAAAAAABAAAAAAA' EXCEPT SELECT * FROM
tpcds.reason_p WHERE r_reason_sk<4;
r_reason_sk |   r_reason_id    |      r_reason_desc
-------------+------------------+----------------------------------
          10 | AAAAAAAABAAAAAAA | reason 2
          10 | AAAAAAAABAAAAAAA | reason 5
          10 | AAAAAAAABAAAAAAA | reason 4
           4 | AAAAAAAABAAAAAAA | reason 3
(4 rows)
```

通过在where子句中指定"(+)"来实现左连接。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where t1.sr_customer_sk  =
t2.c_customer_sk(+)
order by 1 desc limit 1;
 sr_item_sk | c_customer_id
------------+---------------
      18000 |
(1 row)
```

通过在where子句中指定"(+)"来实现右连接。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where t1.sr_customer_sk(+)  =
t2.c_customer_sk
order by 1 desc limit 1;
 sr_item_sk |  c_customer_id
------------+------------------
            | AAAAAAAAJNGEBAAA
(1 row)
```

通过在where子句中指定"(+)"来实现左连接，并且增加连接条件。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where t1.sr_customer_sk  =
t2.c_customer_sk(+) and t2.c_customer_sk(+) < 1 order by 1  limit 1;
 sr_item_sk | c_customer_id
------------+---------------
          1 |
(1 row)
```

不支持在where子句中指定"(+)"的同时使用内层嵌套AND/OR的表达式。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where not(t1.sr_customer_sk  =
t2.c_customer_sk(+) and t2.c_customer_sk(+) < 1);
ERROR:  Operator "(+)" can not be used in nesting expression.
LINE 1: ...tomer_id from store_returns t1, customer t2 where
not(t1.sr_...
```

where子句在不支持表达式宏指定"(+)"会报错。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where (t1.sr_customer_sk  =
t2.c_customer_sk(+))::bool;
ERROR:  Operator "(+)" can only be used in common expression.
```

where子句在表达式的两边都指定"(+)"会报错。

```
select t1.sr_item_sk ,t2.c_customer_id from store_returns t1, customer t2 where t1.sr_customer_sk(+)  =
t2.c_customer_sk(+);
ERROR:  Operator "(+)" can't be specified on more than one relation in one join condition
HINT:  "t1", "t2"...are specified Operator "(+)" in one condition.
```

 版权所有 © 华为云计算技术有限公司

示例：使用 GROUP BY 进行分组查询（ROLLUP、CUBE、GROUPING SETS）

在一些报表场景中，经常会对数据做分组统计（group by），例如对一级部门下辖的二级部门员工数进行统计。

```
DROP TABLE IF EXISTS emp;
CREATE TABLE emp(
id int primary key,    --工号
name text,             --员工名
dep_1 text,            --一级部门
dep_2 text             --二级部门
);

INSERT INTO emp VALUES
(01, 'liwei', 'Product', 'SRE'),
(02, 'liming', 'Product', 'U1'),
(03, 'hanlei', 'Product', 'U1'),
(04, 'hanmeimei', 'Product', 'SRE'),
(05, 'dingjun', 'Product', 'SRE'),
(06, 'huxue', 'Product', 'E1'),
(07, 'liuyi', 'Product', 'E1'),
(08, 'luhua', 'Product', 'E1'),
(09, 'yangjing', 'Product', 'U1'),
(10, 'binbin', 'Product', 'U1'),
(11, 'jim', 'Product', 'SRE'),
(12, 'leo', 'Product', 'SER');


SELECT COUNT(*), dep_2 FROM emp GROUP BY dep_2;   --统计二级部门的员工人数
```

count	dep_2
1	SER
3	E1
4	SRE
4	U1

常见的统计报表业务中，通常需要进一步计算一级部门的“合计”人数，也就是二级部门各分组的累加，就可以借助于rollup，如下所示，比前面的分组计算结果多了一行合计的数据。如下图的“12 Null”，表示一级部门总人数12。

```
SELECT COUNT(*), dep_2 FROM emp GROUP BY ROLLUP(dep_2) ORDER BY 1;
```


版权所有 © 华为云计算技术有限公司

count	dep_2
1	SER
3	E1
4	U1
4	SRE
12	(Null)

接下来，我们解释下ROLLUP，CUBE，GROUPING SETS的用法。

- **使用ROLLUP**：

  首先创建一张用户信息表customer：

```
DROP TABLE IF EXISTS customer;
CREATE TABLE customer
(
c_id char(16) not null,         --用户iD
c_name char(20) ,               --用户名
c_age integer ,                --年龄
c_country varchar(20) ,          --地区
c_class char(10),               --用户级别
c_gender text,                 --性别
c_balance numeric );            --余额

INSERT INTO customer VALUES(1, 'tom', '20', 'Region A', '1', 'male', 300);
INSERT INTO customer VALUES(2, 'jack', '30', 'Region B', '1', 'male', 100);
INSERT INTO customer VALUES(3, 'rose', '40', 'Region C', '1', 'female', 200);
INSERT INTO customer VALUES(4, 'Frank', '60', 'Region D', '1', 'male', 100);
INSERT INTO customer VALUES(5, 'Leon', '20', 'Region A', '2', 'male', 200);
INSERT INTO customer VALUES(6, 'Lucy', '20', 'Region A', '1', 'female', 500);
```

  ROLLUP是在分组计算基础上增加了合计，从字面意思理解，就是从最小聚合级开始，聚合单位逐渐扩大，例如如下语句。

```
--首先对(c_country, c_class)进行GROUP BY，接着对(c_country)进行GROUP BY,最后对全表进行GROUP BY操作。
SELECT c_country, c_class, sum(c_balance) FROM customer GROUP BY ROLLUP(c_country, c_class) ORDER BY 1,2,3;
```

 版权所有 © 华为云计算技术有限公司

c_country	c_class	sum
Region A	1	800
Region A	2	200
Region A	(Null)	1000
Region B	1	100
Region B	(Null)	100
Region C	1	200
Region C	(Null)	200
Region D	1	100
Region D	(Null)	100
(Null)	(Null)	1400

Total Records: 10

以上语句等价于：

```
SELECT c_country, c_class, SUM(c_balance) FROM customer GROUP BY c_country, c_class UNION ALL
SELECT c_country, null, SUM(c_balance) FROM customer GROUP BY c_country
UNION ALL
SELECT null, null, SUM(c_balance) FROM customer ORDER BY 1,2,3;
```

- **使用CUBE**：

  CUBE从字面意思理解就是各个维度的意思，也就是说全部组合，即聚合键中所有字段的组合的分组统计结果，例如如下语句：

  ```
  --首先对(c_country,c_class)进行GROUP BY，然后依次对(c_country)、(c_class)进行GROUP BY，最后对全表进行GROUP BY操作。
  SELECT c_country, c_class, SUM(c_balance) FROM customer GROUP BY CUBE(c_country, c_class) ORDER BY 1,2,3;
  ```

 版权所有 © 华为云计算技术有限公司

c_country	c_class	sum
Region A	1	800
Region A	2	200
Region A	*(Null)*	1000
Region B	1	100
Region B	*(Null)*	100
Region C	1	200
Region C	*(Null)*	200
Region D	1	100
Region D	*(Null)*	100
(Null)	1	1200
(Null)	2	200
(Null)	*(Null)*	1400

Total Records: 12

以上语句等价于：

```
SELECT c_country, c_class, SUM(c_balance) FROM customer GROUP BY c_country, c_class
union all
SELECT c_country, null, SUM(c_balance) FROM customer GROUP BY c_country
union all
SELECT null, null, SUM(c_balance) FROM customer
union all
SELECT NULL, c_class, SUM(c_balance) FROM customer GROUP BY c_class order by 1,2,3;
```

- **使用GROUPING SETS**:

  GROUPING SETS区别于ROLLUP和CUBE，并没有总体的合计功能，相当于从ROLLUP和CUBE的结果中提取出部分记录，例如如下语句：

  ```
  --分别对(c_class)、(c_country)进行GROUP BY计算
  SELECT c_country, c_class, SUM(c_balance) FROM customer GROUP BY GROUPING SETS(c_country, c_class) ORDER BY 1,2,3;
  ```

 版权所有 © 华为云计算技术有限公司

c_country	c_class	sum
Region A	(Null)	1000
Region B	(Null)	100
Region C	(Null)	200
Region D	(Null)	100
(Null)	1	1200
(Null)	2	200

Total Records: 6

以上语句等价于：

```
SELECT c_country, null, SUM(c_balance) FROM customer GROUP BY c_country
UNION ALL
SELECT null, c_class, sum(c_balance) FROM customer GROUP BY c_class ORDER BY 1,2,3;
```

21.16.3 SELECT INTO

功能描述

SELECT INTO用于根据查询结果创建一个新表，并且将查询到的数据插入到新表中。

数据并不返回给客户端，这一点和普通的SELECT不同。新表的字段具有和SELECT的输出字段相同的名字和数据类型。

注意事项

CREATE TABLE AS的作用和SELECT INTO类似，且提供了SELECT INTO所提供功能的超集。建议使用CREATE TABLE AS语法替代SELECT INTO，因为SELECT INTO不能在存储过程中使用并且SELECT INTO字段名，不支持接收空行。

语法格式

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    { * | {expression [ [ AS ] output_name ]} [, ...] }
    INTO [ [ GLOBAL | LOCAL | VOLATILE ] { TEMPORARY | TEMP } | UNLOGGED ] [ TABLE ] new_table
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ WINDOW {window_name AS ( window_definition )} [, ...] ]
    [ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]
    [ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort_expression_clause ] [ NULLS { FIRST |
LAST } ]} [, ...] ]
    [ { [ LIMIT { count | ALL } ] [ OFFSET start [ ROW | ROWS ] ] } | { LIMIT start, { count | ALL } } ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]
    [ {FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ]} [...] ];
```


版权所有 © 华为云计算技术有限公司

参数说明

INTO [[GLOBAL | LOCAL | VOLATILE] { TEMPORARY | TEMP } | UNLOGGED] [TABLE] new_table

[GLOBAL | LOCAL | VOLATILE] { TEMPORARY | TEMP }指定临时表类型，具体可参考**参数说明**章节中GLOBAL | LOCAL | VOLATILE相关说明。

UNLOGGED指定表为非日志表。写入非日志表中的数据不会写入到预写日志中，这样就会比普通表快很多。但是，非日志表也是不安全的，在冲突或异常关机后会被自动删截。非日志表中的内容也不会被复制到备用服务器中。在该类表中创建的索引也不会被自动记录。

new_table指定新建表的名字。

说明

SELECT INTO的其它参数可参考SELECT的**SELECT主句参数说明**。

示例

将reason_t表中TABLE_SK小于3的值加入到新建表中。

```
CREATE TABLE reason_t
(
   TABLE_SK         INTEGER              ,
   TABLE_ID         VARCHAR(20)          ,
   TABLE_NA         VARCHAR(20)
);
INSERT INTO reason_t VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB'),(3, 'S02',
'StudentB');

SELECT * INTO reason_t_bck FROM reason_t WHERE TABLE_SK < 3;
INSERT 0 2
```

相关链接

SELECT

21.17 TCL 语法

21.17.1 TCL 语法一览表

TCL（Transaction Control Language事务控制语言 ），用来控制数据库操纵事务发生的时间及效果，对数据库实行监视等。

提交

DWS通过COMMIT或者END可完成提交事务的功能。请参考**COMMIT | END**。

设置保存点

DWS用于在当前事务里建立一个新的保存点。请参考**SAVEPOINT**。

版权所有 © 华为云计算技术有限公司

回滚

DWS回滚当前事务状态回到上次最后提交的状态。请参考**ROLLBACK**。

21.17.2 ABORT

功能描述

回滚当前事务并且撤销所有当前事务中所做的更改。

作用等同于**ROLLBACK**，早期SQL有用ABORT，现在推荐使用ROLLBACK。

注意事项

在事务外部执行ABORT语句不会影响事务的执行，但是会产生一个警告信息。

语法格式

```
ABORT [ WORK | TRANSACTION ] ;
```

参数说明

WORK | TRANSACTION

可选关键字，对ABORT操作没有影响。

示例

终止事务，执行的更新操作会被撤销掉：

```
ABORT;
```

相关链接

SET TRANSACTION，**COMMIT | END**，**ROLLBACK**

21.17.3 BEGIN

功能描述

BEGIN可以用于开始一个匿名块，也可以用于开始一个事务。本节描述用BEGIN开始匿名块的语法，以BEGIN开始事务的语法见**START TRANSACTION**。

匿名块是能够动态地创建和执行过程代码的结构，而不需要以持久化的方式将代码作为数据库对象储存在数据库中。

注意事项

无。

语法格式

- 开启匿名块

```
[DECLARE [declare_statements]]
BEGIN
```


版权所有 © 华为云计算技术有限公司

```
execution_statements
END;
/
```

- 开启事务

```
BEGIN [ WORK | TRANSACTION ]
 [
  {
    ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE
READ }
    | { READ WRITE | READ ONLY }
   } [, ...]
 ];
```

参数说明

表 21-309 BEGIN 参数说明

参数	描述	取值范围
declare_statements	声明变量，包括变量名和变量类型，如“sales_cnt int”。	-
execution_statements	匿名块中要执行的语句。	已存在的函数名称。

示例

- 开始事务块：

```
BEGIN;
```

- 要以可重复读隔离级别开始事务块：

```
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

- 使用匿名块输出字符串：

```
BEGIN
dbms_output.put_line('Hello');
END;
/
```

相关链接

START TRANSACTION

21.17.4 CHECKPOINT

功能描述

检查点（CHECKPOINT）是一个事务日志中的点，所有数据文件都在该点被更新以反映日志中的信息，所有数据文件都将被刷新到磁盘。

设置事务日志检查点。预写式日志（WAL）缺省时在事务日志中每隔一段时间放置一个检查点。可以设置相关运行时参数（checkpoint_segments和checkpoint_timeout）来调整这个原子化检查点的间隔。

版权所有 © 华为云计算技术有限公司

注意事项

- 只有系统管理员可以调用CHECKPOINT。
- CHECKPOINT强制立即进行检查，而不是等到下一次调度时的检查点。

语法格式

```
CHECKPOINT;
```

参数说明

无。

示例

设置检查点：

```
CHECKPOINT;
```

21.17.5 COMMIT | END

功能描述

通过COMMIT或者END可完成提交事务的功能，即提交事务的所有操作。

注意事项

执行COMMIT这个命令的时候，命令执行者必须是该事务的创建者或系统管理员，且创建和提交操作可以不在同一个会话中。

语法格式

```
{ COMMIT | END } [ WORK | TRANSACTION ] ;
```

参数说明

表 21-310 COMMIT | END 参数说明

参数	描述
COMMIT | END	通过COMMIT或END提交当前事务，让所有当前事务更改为其他事务可见。
WORK | TRANSACTION	可选关键字，仅提供可读性，对提交事务操作没有影响。

示例

提交事务，让所有更改永久化：

```
COMMIT;
```


版权所有 © 华为云计算技术有限公司

相关链接

ROLLBACK

21.17.6 COMMIT PREPARED

功能描述

提交一个早先为两阶段提交准备好的事务。

注意事项

- 该功能仅在维护模式(GUC参数xc_maintenance_mode为on时)下可用。该模式谨慎打开，一般供维护人员排查问题使用，一般用户不应使用该模式。
- 命令执行者必须是该事务的创建者或系统管理员，且创建和提交操作可以不在同一个会话中。
- 事务功能由数据库自动维护，不应显式使用事务功能。

语法格式

```
COMMIT PREPARED transaction_id ;
COMMIT PREPARED transaction_id WITH CSN;
```

参数说明

表 21-311 COMMIT PREPARED 参数说明

参数	描述
transaction_id	待提交事务的标识符，不能和任何当前预备事务已经使用了的标识符同名。
CSN(commit sequence number)	待提交事务的序列号，是一个64位递增无符号数字。

相关链接

PREPARE TRANSACTION，ROLLBACK PREPARED

21.17.7 PREPARE TRANSACTION

功能描述

为当前事务做两阶段提交的准备。

在命令之后，事务就不再和当前会话关联了；它的状态完全保存在磁盘上，它被提交成功的可能性非常高，即使是在请求提交之前数据库发生了崩溃也如此。

一旦准备好了，一个事务就可以在稍后用COMMIT PREPARED或 ROLLBACK PREPARED命令分别进行提交或者回滚。这些命令可以从任何会话中发出，而不光是最初执行事务的那个会话。

版权所有 © 华为云计算技术有限公司

从发出命令的会话的角度来看，PREPARE TRANSACTION不同于ROLLBACK：在执行它之后，就不再有活跃的当前事务了，并且预备事务的效果无法见到 (在事务提交的时候其效果会再次可见)。

如果PREPARE TRANSACTION因为某些原因失败，那么它就会变成一个ROLLBACK，当前事务被取消。

注意事项

- 事务功能由数据库自动维护，不应显式使用事务功能。
- 在运行PREPARE TRANSACTION命令时，必须在postgresql.conf配置文件中增大max_prepared_transactions的数值。建议至少将其设置为等于max_connections，这样每个会话都可以有一个等待中的预备事务。

语法格式

```
PREPARE TRANSACTION transaction_id;
```

参数说明

transaction_id

待提交事务的标识符，用于后面在COMMIT PREPARED或ROLLBACK PREPARED的时候标识这个事务。不能和任何当前预备事务已经使用了的标识符同名。

取值范围：标识符必须以字符串文本的方式书写，并且长度必须小于200字节。

相关链接

COMMIT PREPARED，**ROLLBACK PREPARED**

21.17.8 SAVEPOINT

功能描述

SAVEPOINT用于在当前事务里建立一个新的保存点。

保存点是事务中的一个特殊记号，它允许将那些在它建立后执行的命令全部回滚，把事务的状态恢复到保存点所在的时刻。

注意事项

- 使用ROLLBACK TO SAVEPOINT回滚到一个保存点。使用RELEASE SAVEPOINT删除一个保存点，但是保留该保存点建立后执行的命令的效果。
- 保存点只能在一个事务块里面建立。在一个事务里面可以定义多个保存点。
- 函数、匿名块和存储过程中不支持使用SAVEPOINT语法。
- 由于节点故障或者通信故障引起的分布式节点线程或进程退出导致的报错，以及由于COPY FROM操作中源数据与目标表的表结构不一致导致的报错，均不能正常回滚到保存点之前，而是整个事务回滚。
- SQL标准要求，使用savepoint建立一个同名保存点时，需要自动删除前面那个同名保存点。在DWS数据库里，将保留旧的保存点，但是在回滚或者释放的时候，只使用最近的那个。释放了新的保存点将导致旧的再次成为ROLLBACK TO SAVEPOINT和RELEASE SAVEPOINT可以访问的保存点。除此之外，SAVEPOINT是完全符合SQL标准的。

版权所有 © 华为云计算技术有限公司

语法格式

```
SAVEPOINT savepoint_name;
```

参数说明

savepoint_name

新建保存点的名字。

示例

- 建立一个保存点，然后撤销建立保存点后执行的所有命令的效果：

```
START TRANSACTION;
INSERT INTO table1 VALUES (1);
SAVEPOINT my_savepoint;
INSERT INTO table1 VALUES (2);
ROLLBACK TO SAVEPOINT my_savepoint;
INSERT INTO table1 VALUES (3);
COMMIT;
```

查询表的内容，会同时看到1和3，不能看到2，因为2被回滚。

- 建立并随后销毁一个保存点：

```
START TRANSACTION;
INSERT INTO table1 VALUES (3);
SAVEPOINT my_savepoint;
INSERT INTO table1 VALUES (4);
RELEASE SAVEPOINT my_savepoint;
COMMIT;
```

查询表的内容，会同时看到3和4。

相关链接

RELEASE SAVEPOINT，**ROLLBACK TO SAVEPOINT**

21.17.9 SET TRANSACTION

功能描述

为当前事务设置特性。它对后面的事务没有影响。事务特性包括事务隔离级别、事务访问模式(读/写或者只读)。

注意事项

无。

语法格式

设置事务的隔离级别、读写模式。

```
{ SET [ LOCAL ] TRANSACTION|SET SESSION CHARACTERISTICS AS TRANSACTION }
  { ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }
  | { READ WRITE | READ ONLY } } [, ...]
```


版权所有 © 华为云计算技术有限公司

参数说明

表 21-312 SET TRANSACTION 参数说明

参数	描述	取值范围
LOCAL	声明该命令只在当前事务中有效。	-
SESSION	声明该命令只对当前会话起作用。	字符串，要符合标识符的命名规范。
ISOLATION LEVEL	指定事务隔离级别，该参数决定当一个事务中存在其他并发运行事务时能够看到什么数据。 **说明** 在事务中第一个数据修改语句（INSERT，DELETE，UPDATE，FETCH，COPY）执行之后，事务隔离级别就不能再次设置。	• READ COMMITTED：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。缺省值为READ COMMITTED。 • READ UNCOMMITTED：读未提交隔离级别，DWS不支持READ UNCOMMITTED，如果设置了READ UNCOMMITTED，实际上使用的是READ COMMITTED。 • REPEATABLE READ：可重复读隔离级别，仅能看到事务开始之前提交的数据，不能看到未提交的数据以及在事务执行期间由其它并发事务提交的修改。 • SERIALIZABLE：事务可序列化，DWS不支持SERIALIZABLE，如果设置了SERIALIZABLE，实际上使用的是REPEATABLE READ。
READ WRITE | READ ONLY	指定事务访问模式。	读写模式或者只读模式。

示例

设置当前事务的隔离级别为READ COMMITTED，访问模式为READ ONLY：

```
START TRANSACTION;
SET LOCAL TRANSACTION ISOLATION LEVEL READ COMMITTED READ ONLY;
COMMIT;
```

21.17.10 START TRANSACTION

功能描述

通过START TRANSACTION启动事务。如果声明了隔离级别、读写模式，那么新事务就使用这些特性，类似执行了**SET TRANSACTION**。

注意事项

无。

版权所有 © 华为云计算技术有限公司

语法格式

格式一：START TRANSACTION格式

```
START TRANSACTION
 [
  {
    ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }
    | { READ WRITE | READ ONLY }
  } [, ...]
 ];
```

格式二：BEGIN格式

```
BEGIN [ WORK | TRANSACTION ]
 [
  {
    ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }
    | { READ WRITE | READ ONLY }
   } [, ...]
 ];
```

参数说明

表 21-313 START TRANSACTION 参数说明

参数	描述	取值范围
WORK | TRANSACTION	BEGIN格式中的可选关键字，对操作没有影响。	-
ISOLATION LEVEL	指定事务隔离级别，该参数决定当一个事务中存在其他并发运行事务时能够看到什么数据。 **说明** 在事务中第一个数据修改语句（INSERT，DELETE，UPDATE，FETCH，COPY）执行之后，事务隔离级别就不能再次设置。	• READ COMMITTED：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。缺省值为READ COMMITTED。 • READ UNCOMMITTED：读未提交隔离级别，DWS不支持READ UNCOMMITTED，如果设置了READ UNCOMMITTED，实际上使用的是READ COMMITTED。 • REPEATABLE READ：可重复读隔离级别，仅能看到事务开始之前提交的数据，不能看到未提交的数据以及在事务执行期间由其它并发事务提交的修改。 • SERIALIZABLE：事务可序列化，DWS不支持SERIALIZABLE，如果设置了SERIALIZABLE，实际上使用的是REPEATABLE READ。
READ WRITE | READ ONLY	指定事务访问模式。	读写模式或者只读模式。

版权所有 © 华为云计算技术有限公司

示例

- 以默认方式启动事务：

```
START TRANSACTION;
SELECT * FROM tpcds.reason;
END;
```

- 以隔离级别为READ COMMITTED，读/写方式启动事务：

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED READ WRITE;
SELECT * FROM tpcds.reason;
COMMIT;
```

相关链接

COMMIT | END，**ROLLBACK**，**SET TRANSACTION**

21.17.11 ROLLBACK

功能描述

回滚当前事务并取消当前事务中的所有更新。

在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销，数据库状态回到事务开始时。

注意事项

如果不在一个事务内部发出ROLLBACK不会有问题，但是将抛出一个警告信息。

语法格式

```
ROLLBACK [ WORK | TRANSACTION ];
```

参数说明

WORK | TRANSACTION

可选关键字。除了增加可读性，对ROLLBACK操作没有影响。

示例

取消当前事务中的所有更改：

```
ROLLBACK;
```

相关链接

COMMIT | END

21.17.12 RELEASE SAVEPOINT

功能描述

RELEASE SAVEPOINT删除当前事务先前定义的保存点。

保存点删除后便无法再作为回滚点使用，除此之外没有其它用户可见的行为。删除保存点并不能撤销在保存点建立起来之后执行的命令的影响。要撤销那些命令可以使用

版权所有 © 华为云计算技术有限公司

ROLLBACK TO SAVEPOINT 。在不再需要的时候删除保存点可以令系统在事务结束之前提前回收一些资源。

RELEASE SAVEPOINT也删除所有在指定的保存点建立之后的所有保存点。

注意事项

- 不能RELEASE一个没有定义的保存点，语法上会报错。
- 如果事务在回滚状态，则不能释放保存点。
- 如果多个保存点拥有同样的名字，只释放最近定义的保存点。

语法格式

```
RELEASE [ SAVEPOINT ] savepoint_name;
```

参数说明

savepoint_name

要删除的保存点的名字。

示例

建立并随后销毁一个保存点：

```
BEGIN;
    INSERT INTO tpcds.table1 VALUES (3);
    SAVEPOINT my_savepoint;
    INSERT INTO tpcds.table1 VALUES (4);
    RELEASE SAVEPOINT my_savepoint;
COMMIT;
```

相关链接

SAVEPOINT，**ROLLBACK TO SAVEPOINT**

21.17.13 ROLLBACK PREPARED

功能描述

取消一个先前为两阶段提交准备好的事务。

注意事项

- 该功能仅在维护模式(GUC参数xc_maintenance_mode为on时)下可用。该模式谨慎打开，一般供维护人员排查问题使用，一般用户不应使用该模式。
- 要想回滚一个预备事务，必须是最初发起事务的用户，或者是系统管理员。
- 事务功能由数据库自动维护，不应显式使用事务功能。

语法格式

```
ROLLBACK PREPARED transaction_id ;
```

参数说明

transaction_id

版权所有 © 华为云计算技术有限公司

待提交事务的标识符。它不能和任何当前预备事务已经使用了的标识符同名。

相关链接

COMMIT PREPARED，PREPARE TRANSACTION

21.17.14 ROLLBACK TO SAVEPOINT

功能描述

ROLLBACK TO SAVEPOINT用于回滚到保存点，隐含地删除所有在该保存点之后建立的保存点。

回滚所有指定保存点建立之后执行的命令。保存点仍然有效，并且需要时可以再次回滚到该点。

注意事项

- 不能回滚到未定义的保存点，语法上会报错。
- 在保存点方面，游标有一些非事务性的行为。任何在保存点里打开的游标都会在回滚掉这个保存点之后关闭。如果一个前面打开了的游标在保存点里面，并且游标被一个FETCH命令影响，而这个保存点稍后回滚了，那么这个游标的位置仍然在FETCH让它指向的位置(也就是FETCH不会被回滚)。关闭一个游标的行为也不会被回滚给撤销掉。如果一个游标的操作导致事务回滚，那么这个游标就会置于不可执行状态，所以，尽管一个事务可以用ROLLBACK TO SAVEPOINT重新恢复，但是游标不能再使用了。
- 使用ROLLBACK TO SAVEPOINT回滚到保存点。使用RELEASE SAVEPOINT删除保存点，但是保留该保存点建立后执行的命令的效果。

语法格式

```
ROLLBACK [ WORK | TRANSACTION ] TO [ SAVEPOINT ] savepoint_name;
```

参数说明

savepoint_name

回滚指定保存点的名称。

示例

撤销my_savepoint建立之后执行的命令的影响：

```
ROLLBACK TO SAVEPOINT my_savepoint;
```

游标位置不受保存点回滚的影响：

```
BEGIN;
DECLARE foo CURSOR FOR SELECT 1 UNION SELECT 2;
SAVEPOINT foo;
FETCH 1 FROM foo;
 ?column?
----------
        1
ROLLBACK TO SAVEPOINT foo;
FETCH 1 FROM foo;
 ?column?
```


版权所有 © 华为云计算技术有限公司

```
----------
         2
COMMIT;
```

相关链接

SAVEPOINT，**RELEASE SAVEPOINT**

 版权所有 © 华为云计算技术有限公司