数据仓库服务
9.1.2
开发指南

文档版本 01

发布日期 2026-11-23

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2026。 保留一切权利。

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

商标声明

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

注意

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

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

目 录

1 使用前必读 1
2 DWS 开发设计建议 4
2.1 DWS 总体开发设计规范 4
2.2 DWS 连接管理规范 7
2.3 DWS 对象设计规范 8
2.3.1 DATABASE 对象设计规范 8
2.3.2 USER 对象设计规范 9
2.3.3 SCHEMA 对象设计规范 10
2.3.4 TABLESPACE 对象设计规范 10
2.3.5 TABLE 对象设计规范（重点） 10
2.3.6 INDEX 对象设计规范（重点） 14
2.3.7 VIEW 对象设计规范 14
2.4 DWS SQL 开发规范 14
2.4.1 DDL 操作规范 15
2.4.2 INSERT 操作规范 15
2.4.3 UPDATE&DELETE 操作规范 16
2.4.4 SELECT 操作规范 16
2.5 DWS 外表功能开发规范 19
2.6 DWS 存储过程开发规范 20
2.7 DWS 各对象设计详细规则 21
2.7.1 DWS 数据库对象命名规则 21
2.7.2 DWS 数据库对象设计规则 22
2.7.2.1 DWS Database 和 Schema 设计规则 22
2.7.2.2 DWS 表设计规则 23
2.7.2.3 DWS 字段设计规则 25
2.7.2.4 DWS 约束设计规则 26
2.7.2.5 DWS 视图和关联表设计规则 27
2.7.3 DWS JDBC 配置规则 27
2.7.4 DWS SQL 编写规则 28
2.7.5 自定义 DWS 外部函数(pgSQL/Java)使用规则 31
2.7.6 DWS PL/pgSQL 使用规则 32
3 创建和管理 DWS 数据库对象 35

 版权所有 © 华为云计算技术有限公司

3.1 创建和管理 DWS 数据库 35
3.2 创建和管理 DWS Schema 36
3.3 创建和管理 DWS 表 39
3.4 选择 DWS 表存储模型 43
3.5 创建和管理 DWS 分区表 46
3.6 创建和管理 DWS 索引 53
3.7 创建和使用 DWS 序列 56
3.8 创建和管理 DWS 视图 58
3.9 创建和管理 DWS 定时任务 59
3.10 查看 DWS 系统表 61
4 Oracle、Teradata 和 MySQL 语法兼容性差异 64
5 DWS 数据库安全管理 85
5.1 DWS 用户及权限管理 85
5.1.1 DWS 数据库用户类型 85
5.1.2 DWS 数据库用户管理 86
5.1.3 自定义 DWS 密码策略 87
5.1.4 DWS 数据库权限管理 89
5.1.5 DWS 三权分立 93
5.2 DWS 敏感数据管理 94
5.2.1 DWS 行级访问控制 94
5.2.2 DWS 数据脱敏 96
5.2.3 DWS 字符串加解密 99
5.2.4 使用 pgcrypto 加密 DWS 数据 102
5.2.5 细粒度透明数据加密 110
6 查询 DWS 数据 113
6.1 单表查询 113
6.2 多表连接查询 114
6.3 子查询表达式 119
6.4 WITH 表达式 123
6.5 UNION 操作符的使用 129
6.6 跨计算组数据读写 132
6.7 SQL on Hudi 134
6.7.1 Hudi 简介 134
6.7.2 使用 Hudi 前准备 135
6.7.3 Hudi 用户接口 136
6.7.4 创建 Hudi 数据（外表） 137
6.7.5 Hudi 任务同步 138
6.7.6 Hudi 外表查询 140
6.7.7 支持访问 MRS 上的 Hudi 表 141
6.8 SQL on Iceberg 143
7 DWS 排序规则 146

 版权所有 © 华为云计算技术有限公司

8 DWS 物化视图 149
8.1 物化视图概述 149
8.2 查询重写 154
8.2.1 概述 154
8.2.2 文本匹配查询重写 155
8.2.3 结构匹配查询重写 157
8.3 物化视图刷新 164
8.3.1 全量刷新 164
8.3.2 增量刷新 166
8.3.3 全量刷新与增量刷新对比 168
8.3.4 分区物化视图 168
8.3.4.1 分区物化视图概述 168
8.3.4.2 创建分区物化视图 169
8.3.4.3 刷新分区物化视图 172
8.3.5 嵌套物化视图 173
8.3.5.1 嵌套物化视图概述 173
8.3.5.2 创建嵌套物化视图 174
8.3.5.3 刷新嵌套物化视图 176
8.4 物化视图设计与实践 178
8.4.1 物化视图设计 178
8.4.2 物化视图设计最佳实践 180
8.4.3 物化视图的监控与运维 189
8.5 物化视图应用案例 191
8.5.1 电商平台实时销售报表案例 191
8.5.2 网站用户行为实时分析案例 192
8.5.3 金融交易风控场景案例 194
9 DWS 用户自定义函数 196
9.1 PL/Java 语言函数 196
9.2 PL/pgSQL 语言函数 206
10 DWS 存储过程 207
10.1 DWS 存储过程简介 207
10.2 DWS 存储过程数据类型转换 207
10.3 DWS 存储过程数组和 record 208
10.3.1 数组 208
10.3.2 record 214
10.4 DWS 存储过程声明语法 217
10.5 DWS 存储过程基本语句 219
10.6 DWS 存储过程动态语句 222
10.6.1 执行动态查询语句 222
10.6.2 执行动态非查询语句 224
10.6.3 动态调用存储过程 225
10.6.4 动态调用匿名块 227

 版权所有 © 华为云计算技术有限公司

10.7 DWS 存储过程控制语句 228
10.7.1 返回语句 228
10.7.2 条件语句 230
10.7.3 循环语句 232
10.7.4 分支语句 235
10.7.5 空语句 237
10.7.6 错误捕获语句 237
10.7.7 GOTO 语句 239
10.7.8 事务控制语句 240
10.8 DWS 存储过程其他语句 244
10.9 DWS 存储过程游标 245
10.9.1 游标概述 245
10.9.2 显式游标 245
10.9.3 隐式游标 249
10.9.4 游标循环 250
10.10 DWS 存储过程高级包 251
10.10.1 DBMS_LOB 251
10.10.2 DBMS_RANDOM 259
10.10.3 DBMS_OUTPUT 261
10.10.4 UTL_RAW 262
10.10.5 DBMS_JOB 264
10.10.6 DBMS_SQL 272
10.11 DWS 存储过程调试 281
11 使用 PostGIS Extension 285
11.1 PostGIS 概述 285
11.2 PostGIS 使用 286
11.3 PostGIS 支持和限制 286
11.4 OPEN SOURCE SOFTWARE NOTICE (For PostGIS) 299
12 使用 JDBC 或 ODBC 进行 DWS 二次开发 327
12.1 开发前准备 327
12.2 基于 JDBC 开发 327
12.2.1 JDBC 开发流程 327
12.2.2 JDBC 包与驱动类 328
12.2.3 加载驱动 329
12.2.4 连接数据库 329
12.2.5 执行 SQL 语句 332
12.2.6 处理结果集 334
12.2.7 常用 JDBC 开发示例 336
12.2.8 应用端加工 RoaringBitmap 结果集并入库 DWS 开发示例 346
12.2.9 JDBC 接口参考 348
12.3 基于 ODBC 开发 361
12.3.1 ODBC 包及依赖的库和头文件 361

 版权所有 © 华为云计算技术有限公司

12.3.2 Linux 下配置数据源 362
12.3.3 Windows 下配置数据源 368
12.3.4 ODBC 开发示例 372
12.3.5 ODBC 接口参考 377
13 DWS 资源监控 395
13.1 用户资源监控 395
13.2 资源池资源监控 397
13.3 内存资源监控 399
13.4 实例资源监控 400
13.5 实时 TopSQL 402
13.6 历史 TopSQL 405
13.7 TopSQL 查询示例 409
14 DWS 性能调优 412
14.1 性能调优概述 412
14.2 性能诊断 414
14.2.1 集群性能分析 414
14.2.2 慢 SQL 分析 414
14.2.2.1 查询最耗性能的 SQL 414
14.2.2.2 分析作业是否被阻塞 415
14.2.3 SQL 诊断 416
14.2.4 表诊断 417
14.3 系统调优 418
14.3.1 数据库系统参数调优 418
14.3.2 SMP 并行执行 424
14.3.3 配置 LLVM 427
14.4 SQL 调优 429
14.4.1 SQL 查询执行流程 429
14.4.2 SQL 执行计划 431
14.4.3 执行计划算子 442
14.4.4 SQL 调优流程 446
14.4.5 更新统计信息 446
14.4.6 审视和修改表定义 453
14.4.7 SQL 调优进阶 454
14.4.7.1 SQL 自诊断 454
14.4.7.2 语句下推调优 458
14.4.7.3 子查询调优 463
14.4.7.4 统计信息调优 472
14.4.7.5 算子级调优 476
14.4.7.6 数据倾斜调优 478
14.4.7.7 plan management 485
14.4.7.8 磁盘缓存主动预热调优 497
14.4.7.9 通过主外键约束消除多余的 JOIN 498

 版权所有 © 华为云计算技术有限公司

14.4.7.10 SQL 语句改写规则 501
14.4.8 优化器参数调整 502
14.4.9 使用 Plan Hint 进行调优 504
14.4.9.1 Plan Hint 调优概述 504
14.4.9.2 Join 顺序的 Hint 506
14.4.9.3 Join 方式的 Hint 509
14.4.9.4 行数的 Hint 510
14.4.9.5 Stream 方式的 Hint 511
14.4.9.6 Scan 方式的 Hint 514
14.4.9.7 子链接块名的 hint 515
14.4.9.8 运行倾斜的 hint 516
14.4.9.9 指定子查询不提升的 hint 520
14.4.9.10 Drive 方式的 Hint 522
14.4.9.11 字典编码的 hint 523
14.4.9.12 配置参数的 hint 524
14.4.9.13 Hint 的错误、冲突及告警 527
14.4.9.14 Plan Hint 实际调优案例 529
14.4.10 例行维护表 534
14.4.11 例行重建索引 535
14.4.12 SQL 语句出错自动重试 536
14.4.13 query_band 负载识别 540
14.5 SQL 调优案例 543
14.5.1 案例：选择合适的分布列 543
14.5.2 案例：建立合适的索引 544
14.5.3 案例：增加 JOIN 列非空条件 545
14.5.4 案例：典型不下推语句整改案例集 547
14.5.4.1 案例：使排序下推 547
14.5.4.2 案例：With-Recursive contains only values rte is not shippable 548
14.5.4.3 案例：With-Recursive contains system table is not shippable 552
14.5.4.4 案例：SubPlan exec on CN can not be shipped 553
14.5.4.5 案例：Type of Record in TargetList can not be shipped 554
14.5.4.6 案例：Type of Record in non-real table can not be shipped 555
14.5.4.7 案例：main plan exec on CN and SubPlan exec on DN can not be shipped 555
14.5.4.8 案例：With-Recursive under multi-nodegroup scenario is not shippable 556
14.5.4.9 案例：With-Recursive contains conflict distribution in none-recursive(xxx) recursive(xxx) 560
14.5.4.10 案例：Table in TargetList can not be shipped 562
14.5.4.11 案例："Row () Expression" can not be shipped 563
14.5.4.12 案例：Merge Into can't be shipped in the scene temporarily 565
14.5.4.13 案例：Type of Record in dual that is not a real table can not be shipped 565
14.5.5 案例：设置 cost_param 对查询性能优化 566
14.5.6 案例：调整局部聚簇键 570
14.5.7 案例：调整中间表存储方式 572

 版权所有 © 华为云计算技术有限公司

14.5.8 案例：改建分区表 572
14.5.9 案例：调整 GUC 参数 best_agg_plan 574
14.5.10 案例：改写 SQL 排除剪枝干扰 575
14.5.11 案例：改写 SQL 消除 in-clause 577
14.5.12 案例：使用 partial cluster key 578
14.5.13 案例：NOT IN 转 NOT EXISTS 581
15 DWS AI 583
15.1 DWS AI 概述 583
15.2 向量计算 584
15.3 库内推理 588
15.4 MCP Server 603
15.5 大模型特征算子 604
16 实时数仓 608
16.1 实时数仓简介 608
16.2 支持与限制 611
16.3 实时数仓 Binlog 615
16.3.1 订阅实时数仓 Binlog 615
16.3.2 Flink 实时消费 Binlog 621
17 DWS 系统表和系统视图 627
17.1 系统表和系统视图概述 627
17.2 系统表 627
17.2.1 GS_BLOCKLIST_QUERY 627
17.2.2 GS_BLOCKLIST_SQL 628
17.2.3 GS_OBSSCANINFO 629
17.2.4 GS_RESPOOL_RESOURCE_HISTORY 629
17.2.5 GS_WLM_INSTANCE_HISTORY 632
17.2.6 GS_WLM_OPERATOR_INFO 633
17.2.7 GS_WLM_SESSION_INFO 635
17.2.8 GS_WLM_USER_RESOURCE_HISTORY 646
17.2.9 PG_AGGREGATE 647
17.2.10 PG_AM 648
17.2.11 PG_AMOP 650
17.2.12 PG_AMPROC 650
17.2.13 PG_ATTRDEF 651
17.2.14 PG_ATTRIBUTE 651
17.2.15 PG_AUTHID 653
17.2.16 PG_AUTH_HISTORY 655
17.2.17 PG_AUTH_MEMBERS 655
17.2.18 PG_BLOCKLISTS 656
17.2.19 PG_CAST 657
17.2.20 PG_CLASS 657

 版权所有 © 华为云计算技术有限公司

17.2.21 PG_COLLATION....662
17.2.22 PG_CONSTRAINT....662
17.2.23 PG_CONVERSION....665
17.2.24 PG_DATABASE....665
17.2.25 PG_DB_ROLE_SETTING....666
17.2.26 PG_DB_FILE....666
17.2.27 PG_DELAY_DELETE_FILE....667
17.2.28 PG_DEFAULT_ACL....667
17.2.29 PG_DEPEND....668
17.2.30 PG_DESCRIPTION....670
17.2.31 PG_ENUM....670
17.2.32 PG_EXCEPT_RULE....671
17.2.33 PG_EXTERNAL_NAMESPACE....671
17.2.34 PG_EXTENSION....672
17.2.35 PG_EXTENSION_DATA_SOURCE....673
17.2.36 PG_FILE_RECYCLEBIN....673
17.2.37 PG_FINE_DR_INFO....675
17.2.38 PG_FOREIGN_DATA_WRAPPER....675
17.2.39 PG_FOREIGN_SERVER....676
17.2.40 PG_FOREIGN_TABLE....677
17.2.41 PG_INDEX....677
17.2.42 PG_INHERITS....679
17.2.43 PG_JOB_INFO....679
17.2.44 PG_JOBS....680
17.2.45 PG_LANGUAGE....681
17.2.46 PG_LARGEOBJECT....682
17.2.47 PG_LARGEOBJECT_METADATA....682
17.2.48 PG_MATVIEW....682
17.2.49 PG_MATVIEW_DEPEND....683
17.2.50 PG_MATVIEW_PARTITION....684
17.2.51 PG_NAMESPACE....684
17.2.52 PG_OBJECT....685
17.2.53 PG_OBSSCANINFO....686
17.2.54 PG_OPCLASS....686
17.2.55 PG_OPERATOR....687
17.2.56 PG_OPFAMILY....688
17.2.57 PG_PARTITION....689
17.2.58 PG_PLAN_BASELINE....691
17.2.59 PG_PLTEMPLATE....692
17.2.60 PG_PROC....692
17.2.61 PG_PUBLICATION....695
17.2.62 PG_PUBLICATION_NAMESPACE....696

 版权所有 © 华为云计算技术有限公司

17.2.63 PG_PUBLICATION_REL ... 696
17.2.64 PG_RANGE ... 697
17.2.65 PG_REDACTION_COLUMN ... 697
17.2.66 PG_REDACTION_POLICY ... 698
17.2.67 PG_RELFILENODE_SIZE ... 699
17.2.68 PG_RLSPOLICY ... 700
17.2.69 PG_RESOURCE_POOL ... 700
17.2.70 PG_REWRITE ... 701
17.2.71 PG_SECLABEL ... 702
17.2.72 PG_SHDEPEND ... 702
17.2.73 PG_SHDESCRIPTION ... 703
17.2.74 PG_SHSECLABEL ... 704
17.2.75 PG_STATISTIC ... 704
17.2.76 PG_STATISTIC_DATA ... 705
17.2.77 PG_STATISTIC_EXT ... 706
17.2.78 PG_STATISTIC_INFO ... 707
17.2.79 PG_STAT_OBJECT ... 708
17.2.80 PG_SUBSCRIPTION ... 711
17.2.81 PG_SYNONYM ... 712
17.2.82 PG_TABLESPACE ... 712
17.2.83 PG_TRIGGER ... 713
17.2.84 PG_TDE ... 713
17.2.85 PG_TS_CONFIG ... 714
17.2.86 PG_TS_CONFIG_MAP ... 715
17.2.87 PG_TS_DICT ... 715
17.2.88 PG_TS_PARSER ... 716
17.2.89 PG_TS_TEMPLATE ... 716
17.2.90 PG_TYPE ... 717
17.2.91 PG_USER_MAPPING ... 719
17.2.92 PG_USER_STATUS ... 720
17.2.93 PG_WORKLOAD_ACTION ... 720
17.2.94 PGXC_CLASS ... 721
17.2.95 PGXC_GROUP ... 721
17.2.96 PGXC_NODE ... 723
17.2.97 PLAN_TABLE_DATA ... 724
17.2.98 SNAPSHOT ... 725
17.2.99 SQL_OUTLINE ... 725
17.2.100 TABLES_SNAP_TIMESTAMP ... 726
17.2.101 性能视图快照系统表 ... 727
17.3 系统视图 ... 728
17.3.1 ALL_ALL_TABLES ... 728
17.3.2 ALL_CONSTRAINTS ... 728

 版权所有 © 华为云计算技术有限公司

17.3.3 ALL_CONS_COLUMNS 729
17.3.4 ALL_COL_COMMENTS 729
17.3.5 ALL_DEPENDENCIES 730
17.3.6 ALL_IND_COLUMNS 730
17.3.7 ALL_IND_EXPRESSIONS 731
17.3.8 ALL_INDEXES 731
17.3.9 ALL_OBJECTS 732
17.3.10 ALL_PROCEDURES 732
17.3.11 ALL_SEQUENCES 732
17.3.12 ALL_SOURCE 733
17.3.13 ALL_SYNONYMS 733
17.3.14 ALL_TAB_COLUMNS 734
17.3.15 ALL_TAB_COMMENTS 734
17.3.16 ALL_TABLES 735
17.3.17 ALL_USERS 735
17.3.18 ALL_VIEWS 736
17.3.19 DBA_DATA_FILES 736
17.3.20 DBA_USERS 736
17.3.21 DBA_COL_COMMENTS 736
17.3.22 DBA_CONSTRAINTS 737
17.3.23 DBA_CONS_COLUMNS 737
17.3.24 DBA_IND_COLUMNS 738
17.3.25 DBA_IND_EXPRESSIONS 738
17.3.26 DBA_IND_PARTITIONS 739
17.3.27 DBA_INDEXES 739
17.3.28 DBA_OBJECTS 740
17.3.29 DBA_PART_INDEXES 740
17.3.30 DBA_PART_TABLES 741
17.3.31 DBA_PROCEDURES 742
17.3.32 DBA_SEQUENCES 742
17.3.33 DBA_SOURCE 742
17.3.34 DBA_SYNONYMS 743
17.3.35 DBA_TAB_COLUMNS 743
17.3.36 DBA_TAB_COMMENTS 744
17.3.37 DBA_TAB_PARTITIONS 744
17.3.38 DBA_TABLES 746
17.3.39 DBA_TABLESPACES 746
17.3.40 DBA_TRIGGERS 747
17.3.41 DBA_VIEWS 747
17.3.42 DUAL 747
17.3.43 GET_ALL_TSC_INFO 747
17.3.44 GET_TABLESPACE_OBS_FILE 748

 版权所有 © 华为云计算技术有限公司

17.3.45 GET_TSC_INFO 748
17.3.46 GLOBAL_COLUMN_TABLE_IO_STAT 749
17.3.47 GLOBAL_REDO_STAT 749
17.3.48 GLOBAL_REL_IOSTAT 750
17.3.49 GLOBAL_ROW_TABLE_IO_STAT 750
17.3.50 GLOBAL_STAT_DATABASE 751
17.3.51 GLOBAL_TABLE_CHANGE_STAT 752
17.3.52 GLOBAL_TABLE_STAT 753
17.3.53 GLOBAL_WORKLOAD_SQL_COUNT 754
17.3.54 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME 755
17.3.55 GLOBAL_WORKLOAD_TRANSACTION 756
17.3.56 GS_ALL_CONTROL_GROUP_INFO 756
17.3.57 GS_BLOCKLIST_QUERY 757
17.3.58 GS_BLOCKLIST_SQL 757
17.3.59 GS_CLUSTER_RESOURCE_INFO 758
17.3.60 GS_COLUMN_TABLE_IO_STAT 758
17.3.61 GS_LSC_DBSTAT_INFO 759
17.3.62 GS_GET_KVSTORE_DB_COMP_STATS 760
17.3.63 GS_GET_KVSTORE_DB_STALL_DETAIL_STATS 762
17.3.64 GS_GET_KVSTORE_STATS 764
17.3.65 GS_GET_KV_LOCK_STATUS 768
17.3.66 GET_KV_LOCK_STATUS 769
17.3.67 GS_LSC_MEMORY_DETAIL 770
17.3.68 GS_OBS_READ_TRAFFIC 770
17.3.69 GS_OBS_WRITE_TRAFFIC 771
17.3.70 GS_INSTR_UNIQUE_SQL 772
17.3.71 GS_NODE_STAT_RESET_TIME 775
17.3.72 GS_OBS_LATENCY 775
17.3.73 GS_QUERY_MONITOR 775
17.3.74 GS_QUERY_RESOURCE_INFO 778
17.3.75 GS_REL_IOSTAT 779
17.3.76 GS_RESPOOL_RUNTIME_INFO 779
17.3.77 GS_RESPOOL_RESOURCE_INFO 779
17.3.78 GS_RESPOOL_MONITOR 782
17.3.79 GS_ROW_TABLE_IO_STAT 784
17.3.80 GS_SESSION_CPU_STATISTICS 784
17.3.81 GS_SESSION_MEMORY_STATISTICS 785
17.3.82 GS_SQL_COUNT 786
17.3.83 GS_STAT_DB_CU 787
17.3.84 GS_STAT_SESSION_CU 788
17.3.85 GS_TABLE_CHANGE_STAT 788
17.3.86 GS_TABLE_STAT 789

 版权所有 © 华为云计算技术有限公司

17.3.87 GS_TOTAL_NODEGROUP_MEMORY_DETAIL 790
17.3.88 GS_USER_MONITOR 790
17.3.89 GS_USER_TRANSACTION 792
17.3.90 GS_VIEW_DEPENDENCY 792
17.3.91 GS_VIEW_DEPENDENCY_PATH 793
17.3.92 GS_VIEW_INVALID 793
17.3.93 GS_WAIT_EVENTS 794
17.3.94 GS_WLM_OPERATOR_INFO 795
17.3.95 GS_WLM_OPERATOR_HISTORY 797
17.3.96 GS_WLM_OPERATOR_STATISTICS 799
17.3.97 GS_WLM_SESSION_INFO 801
17.3.98 GS_WLM_SESSION_HISTORY 811
17.3.99 GS_WLM_SESSION_STATISTICS 822
17.3.100 GS_WLM_SQL_ALLOW 826
17.3.101 GS_WORKLOAD_SQL_COUNT 826
17.3.102 GS_WORKLOAD_SQL_ELAPSE_TIME 826
17.3.103 GS_WORKLOAD_TRANSACTION 827
17.3.104 MPP_TABLES 828
17.3.105 PG_AVAILABLE_EXTENSION_VERSIONS 828
17.3.106 PG_AVAILABLE_EXTENSIONS 829
17.3.107 PG_BULKLOAD_STATISTICS 829
17.3.108 PG_COMM_CLIENT_INFO 830
17.3.109 PG_COMM_DELAY 830
17.3.110 PG_COMM_STATUS 831
17.3.111 PG_COMM_RECV_STREAM 832
17.3.112 PG_COMM_SEND_STREAM 833
17.3.113 PG_COMM_QUERY_SPEED 834
17.3.114 PG_CONTROL_GROUP_CONFIG 834
17.3.115 PG_CURSORS 834
17.3.116 PG_EXT_STATS 835
17.3.117 PG_GET_INVALID_BACKENDS 837
17.3.118 PG_GET_SENDERS_CATCHUP_TIME 837
17.3.119 PG_GLOBAL_TEMP_ATTACHED_PIDS 838
17.3.120 PG_GROUP 838
17.3.121 PG_INDEXES 838
17.3.122 PG_JOB 839
17.3.123 PG_JOB_PROC 841
17.3.124 PG_JOB_SINGLE 841
17.3.125 PG_LIFECYCLE_DATA_DISTRIBUTE 843
17.3.126 PG_LOCKS 843
17.3.127 PG_LOCKWAIT_DETAIL 844
17.3.128 PG_LWLOCKS 847

 版权所有 © 华为云计算技术有限公司

17.3.129 PG_NODE_ENV 847
17.3.130 PG_OS_THREADS 848
17.3.131 PG_POOLER_STATUS 848
17.3.132 PG_PREPARED_STATEMENTS 849
17.3.133 PG_PREPARED_XACTS 850
17.3.134 PG_PUBLICATION_TABLES 850
17.3.135 PG_QUERYBAND_ACTION 850
17.3.136 PG_REPLICATION_SLOTS 851
17.3.137 PG_ROLES 851
17.3.138 PG_RULES 853
17.3.139 PG_RUNNING_XACTS 853
17.3.140 PG_SECLABELS 854
17.3.141 PG_SEQUENCES 854
17.3.142 PG_SESSION_WLMSTAT 855
17.3.143 PG_SESSION_IOSTAT 857
17.3.144 PG_SETTINGS 857
17.3.145 PG_SHADOW 858
17.3.146 PG_SHARED_MEMORY_DETAIL 859
17.3.147 PG_STATS 860
17.3.148 PG_STATS_EXT_EXPRS 861
17.3.149 PG_STAT_ACTIVITY 863
17.3.150 PG_STAT_ALL_INDEXES 865
17.3.151 PG_STAT_ALL_TABLES 866
17.3.152 PG_STAT_BAD_BLOCK 867
17.3.153 PG_STAT_BGWRITER 868
17.3.154 PG_STAT_DATABASE 869
17.3.155 PG_STAT_DATABASE_CONFLICTS 870
17.3.156 PG_STAT_GET_MEM_MBYTES_RESERVED 870
17.3.157 PG_STAT_USER_FUNCTIONS 871
17.3.158 PG_STAT_USER_INDEXES 871
17.3.159 PG_STAT_USER_TABLES 872
17.3.160 PG_STAT_REPLICATION 873
17.3.161 PG_STAT_SYS_INDEXES 874
17.3.162 PG_STAT_SYS_TABLES 874
17.3.163 PG_STAT_XACT_ALL_TABLES 876
17.3.164 PG_STAT_XACT_SYS_TABLES 876
17.3.165 PG_STAT_XACT_USER_FUNCTIONS 877
17.3.166 PG_STAT_XACT_USER_TABLES 877
17.3.167 PG_STATIO_ALL_INDEXES 878
17.3.168 PG_STATIO_ALL_SEQUENCES 878
17.3.169 PG_STATIO_ALL_TABLES 879
17.3.170 PG_STATIO_SYS_INDEXES 879

 版权所有 © 华为云计算技术有限公司

17.3.171 PG_STATIO_SYS_SEQUENCES 880
17.3.172 PG_STATIO_SYS_TABLES 880
17.3.173 PG_STATIO_USER_INDEXES 881
17.3.174 PG_STATIO_USER_SEQUENCES 881
17.3.175 PG_STATIO_USER_TABLES 881
17.3.176 PG_THREAD_WAIT_STATUS 882
17.3.177 PG_TABLES 892
17.3.178 PG_TDE_INFO 894
17.3.179 PG_TIMEZONE_ABBREVS 894
17.3.180 PG_TIMEZONE_NAMES 894
17.3.181 PG_TOTAL_MEMORY_DETAIL 895
17.3.182 PG_TOTAL_SCHEMA_INFO 897
17.3.183 PG_TOTAL_USER_RESOURCE_INFO 897
17.3.184 PG_USER 900
17.3.185 PG_USER_MAPPINGS 901
17.3.186 PG_VIEWS 901
17.3.187 PG_WLM_STATISTICS 902
17.3.188 PGXC_BULKLOAD_PROGRESS 903
17.3.189 PGXC_BULKLOAD_INFO 903
17.3.190 PGXC_BULKLOAD_STATISTICS 906
17.3.191 PGXC_COLUMN_TABLE_IO_STAT 907
17.3.192 PGXC_COMM_CLIENT_INFO 907
17.3.193 PGXC_COMM_DELAY 908
17.3.194 PGXC_COMM_RECV_STREAM 908
17.3.195 PGXC_COMM_SEND_STREAM 909
17.3.196 PGXC_COMM_STATUS 910
17.3.197 PGXC_COMM_QUERY_SPEED 911
17.3.198 PGXC_DEADLOCK 912
17.3.199 PGXC_DISK_CACHE_STATS 913
17.3.200 PGXC_DISK_CACHE_ALL_STATS 914
17.3.201 PGXC_DISK_CACHE_PATH_INFO 916
17.3.202 PGXC_GET_STAT_ALL_TABLES 916
17.3.203 PGXC_GET_ALL_TABLES_DIRTY_RATIO 917
17.3.204 PGXC_GET_STAT_ALL_PARTITIONS 918
17.3.205 PGXC_GET_TABLE_SKEWNESS 920
17.3.206 PGXC_GET_TABLESPACE_OBS_FILE 920
17.3.207 PGXC_GLOBAL_TEMP_ATTACHED_PIDS 921
17.3.208 PGXC_GTM_SNAPSHOT_STATUS 921
17.3.209 PGXC_INSTANCE_TIME 922
17.3.210 PGXC_LOCKWAIT_DETAIL 922
17.3.211 PGXC_INSTR_UNIQUE_SQL 924
17.3.212 PGXC_LOCK_CONFLICTS 926

 版权所有 © 华为云计算技术有限公司

17.3.213 PGXC_LWLOCKS 927
17.3.214 PGXC_MEMORY_DEBUG_INFO 928
17.3.215 PGXC_NODE_ENV 929
17.3.216 PGXC_NODE_STAT_RESET_TIME 929
17.3.217 PGXC_OBS_IO_SCHEDULER_STATS 930
17.3.218 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS 931
17.3.219 PGXC_OS_RUN_INFO 933
17.3.220 PGXC_OS_THREADS 933
17.3.221 PGXC_POOLER_STATUS 934
17.3.222 PGXC_PREPARED_XACTS 934
17.3.223 PGXC_QUERY_INFO 934
17.3.224 PGXC_QUERY_SPILL_ACTIVITY 937
17.3.225 PGXC_RECYCLEBIN_DROP 938
17.3.226 PGXC_RECYCLEBIN_TRUNCATE 939
17.3.227 PGXC_REDO_STAT 940
17.3.228 PGXC_REL_IOSTAT 940
17.3.229 PGXC_REPLICATION_SLOTS 941
17.3.230 PGXC_RESPOOL_RUNTIME_INFO 941
17.3.231 PGXC_RESPOOL_RESOURCE_INFO 942
17.3.232 PGXC_RESPOOL_RESOURCE_HISTORY 945
17.3.233 PGXC_ROW_TABLE_IO_STAT 948
17.3.234 PGXC_RUNNING_XACTS 948
17.3.235 PGXC_SETTINGS 949
17.3.236 PGXC_SESSION_WLMSTAT 950
17.3.237 PGXC_STAT_ACTIVITY 952
17.3.238 PGXC_STAT_BAD_BLOCK 955
17.3.239 PGXC_STAT_BGWRITER 955
17.3.240 PGXC_STAT_DATABASE 956
17.3.241 PGXC_STAT_OBJECT 957
17.3.242 PGXC_STAT_REPLICATION 961
17.3.243 PGXC_STAT_TABLE_DIRTY 962
17.3.244 PGXC_STAT_WAL 965
17.3.245 PGXC_SQL_COUNT 966
17.3.246 PGXC_TABLE_CHANGE_STAT 966
17.3.247 PGXC_TABLE_STAT 967
17.3.248 PGXC_THREAD_WAIT_STATUS 968
17.3.249 PGXC_TOTAL_MEMORY_DETAIL 970
17.3.250 PGXC_TOTAL_SCHEMA_INFO 972
17.3.251 PGXC_TOTAL_SCHEMA_INFO_ANALYZE 972
17.3.252 PGXC_TOTAL_USER_RESOURCE_INFO 973
17.3.253 PGXC_USER_TRANSACTION 976
17.3.254 PGXC_VARIABLE_INFO 976

 版权所有 © 华为云计算技术有限公司

17.3.255 PGXC_WAIT_DETAIL 977
17.3.256 PGXC_WAIT_EVENTS 979
17.3.257 PGXC_WLM_OPERATOR_HISTORY 980
17.3.258 PGXC_WLM_OPERATOR_INFO 981
17.3.259 PGXC_WLM_OPERATOR_STATISTICS 983
17.3.260 PGXC_WLM_SESSION_INFO 986
17.3.261 PGXC_WLM_SESSION_HISTORY 996
17.3.262 PGXC_WLM_SESSION_STATISTICS 1007
17.3.263 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS 1011
17.3.264 PGXC_WLM_USER_RESOURCE_HISTORY 1012
17.3.265 PGXC_WLM_WORKLOAD_RECORDS 1015
17.3.266 PGXC_WORKLOAD_SQL_COUNT 1016
17.3.267 PGXC_WORKLOAD_SQL_ELAPSE_TIME 1017
17.3.268 PGXC_WORKLOAD_TRANSACTION 1018
17.3.269 PLAN_TABLE 1018
17.3.270 PV_FILE_STAT 1019
17.3.271 PV_INSTANCE_TIME 1020
17.3.272 PV_MATVIEW_DETAIL 1020
17.3.273 PV_MATVIEW_REFRESH_HISTORY 1021
17.3.274 PV_MATVIEW_REWRITE_HISTORY 1023
17.3.275 PV_OS_RUN_INFO 1023
17.3.276 PV_REDO_STAT 1024
17.3.277 PV_RUNTIME_ATTSTATS 1024
17.3.278 PV_RUNTIME_EXPRSTATS 1026
17.3.279 PV_RUNTIME_RELSTATS 1028
17.3.280 PV_SESSION_MEMORY 1029
17.3.281 PV_SESSION_MEMORY_DETAIL 1029
17.3.282 PV_SESSION_STAT 1031
17.3.283 PV_SESSION_TIME 1031
17.3.284 PV_TOTAL_MEMORY_DETAIL 1031
17.3.285 REDACTION_COLUMNS 1033
17.3.286 REDACTION_POLICIES 1034
17.3.287 REMOTE_TABLE_STAT 1035
17.3.288 SHOW_TSC_INFO 1035
17.3.289 SHOW_ALL_TSC_INFO 1036
17.3.290 SQL_PLAN_BASELINE 1036
17.3.291 USER_COL_COMMENTS 1037
17.3.292 USER_CONSTRAINTS 1038
17.3.293 USER_CONS_COLUMNS 1038
17.3.294 USER_INDEXES 1039
17.3.295 USER_IND_COLUMNS 1039
17.3.296 USER_IND_EXPRESSIONS 1039

 版权所有 © 华为云计算技术有限公司

17.3.297 USER_IND_PARTITIONS....1040
17.3.298 USER_JOBS....1041
17.3.299 USER_OBJECTS....1042
17.3.300 USER_PART_INDEXES....1042
17.3.301 USER_PART_TABLES....1043
17.3.302 USER_PROCEDURES....1043
17.3.303 USER_SEQUENCES....1044
17.3.304 USER_SOURCE....1044
17.3.305 USER_SYNONYMS....1044
17.3.306 USER_TAB_COLUMNS....1045
17.3.307 USER_TAB_COMMENTS....1046
17.3.308 USER_TAB_PARTITIONS....1046
17.3.309 USER_TABLES....1046
17.3.310 USER_TRIGGERS....1047
17.3.311 USER_VIEWS....1047
17.3.312 V$SESSION....1048
17.3.313 V$SESSION_LONGOPS....1048
18 DWS 数据库 GUC 参数....1049
18.1 查看 GUC 参数....1049
18.2 设置 GUC 参数....1050
18.3 GUC 使用说明....1051
18.4 连接和认证....1051
18.4.1 连接设置....1052
18.4.2 安全和认证（postgresql.conf）....1053
18.4.3 通信库参数....1054
18.5 资源消耗....1060
18.5.1 内存....1060
18.5.2 语句磁盘空间管控....1069
18.5.3 内核资源使用....1070
18.5.4 基于开销的清理延迟....1071
18.5.5 异步 IO....1072
18.5.6 磁盘缓存....1075
18.6 并行导入....1075
18.7 预写式日志....1076
18.7.1 设置....1077
18.7.2 检查点....1079
18.8 双机复制....1079
18.8.1 主服务器....1080
18.9 查询规划....1080
18.9.1 优化器方法配置....1080
18.9.2 优化器开销常量....1099
18.9.3 基因查询优化器....1101

 版权所有 © 华为云计算技术有限公司

18.9.4 其他优化器选项 1103
18.10 错误报告和日志 1126
18.10.1 记录日志的位置 1126
18.10.2 记录日志的时间 1126
18.10.3 记录日志的内容 1130
18.11 运行时统计 1134
18.11.1 查询和索引统计收集器 1134
18.11.2 性能统计 1138
18.12 资源管理 1140
18.13 自动清理 1153
18.14 客户端连接缺省设置 1155
18.14.1 语句行为 1155
18.14.2 区域和格式化 1161
18.14.3 其他缺省 1165
18.15 锁管理 1165
18.16 版本和平台兼容性 1169
18.16.1 历史版本兼容性 1169
18.16.2 平台和客户端兼容性 1172
18.17 容错性 1210
18.18 连接池参数 1211
18.19 集群事务 1213
18.20 备份容灾参数 1216
18.21 开发人员选项 1217
18.22 审计 1242
18.22.1 审计开关 1242
18.22.2 操作审计 1244
18.23 事务监控 1244
18.24 plan management 管理 1245
18.25 GTM 相关参数 1246
18.26 扩容相关参数 1247
18.27 其他选项 1248
19 DWS 开发者术语表 1261

 版权所有 © 华为云计算技术有限公司

1 使用前必读

文档面向的读者对象

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

作为数据库管理员和应用程序开发人员，至少需要了解以下知识：

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

前置条件

使用本指南前，需要完成以下任务。

- 创建DWS集群。
- 安装SQL客户端。
- 将SQL客户端连接到集群的默认数据库。

关于上述任务的详细指导，请参见**《数据仓库服务快速入门》**。

阅读指引

对于首次接触 DWS的用户，建议先阅读以下部分：

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

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

DWS进行数据库应用程序开发过程中，下表将帮您找到对应的信息。

 版权所有 © 华为云计算技术有限公司

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

SQL 语法文本格式约定

为了方便对语法使用的理解，在文档中对SQL语法文本按如下格式进行表述。

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

版权所有 © 华为云计算技术有限公司

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

声明

DWS文档基于商用视角，从使用场景和任务完成角度提供内容指引。即使这样，文档中依然可能存在对Postgres内容的引用和参考。对于这类内容，遵从如下的Postgres Copyright：

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

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

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

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

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

 版权所有 © 华为云计算技术有限公司

2 DWS 开发设计建议

2.1 DWS 总体开发设计规范

目的

本规范定义了在进行基于DWS数据库自主开发过程中应遵守的设计、开发准则，以提升开发效率，保证业务的连续性、稳定性。

适用范围

该规范适用于所有基于DWS数据库自主开发的场景，包含应用层面的设计与开发、数据库服务层面的设计与开发。

术语定义

规则：数据库设计与开发时必须遵守的约定。

建议：数据库设计与开发时建议考虑的约定。

说明：对规则/建议进行的详细说明和解释。

总体开发设计规范

下表是DWS开发过程中需遵循的**开发设计规范全集列表**，可以单击链接跳转到对应的规则下了解详细说明。

表 2-1 DWS 开发设计规范全集列表

编号	类别		规则/建议
1	连接管理规范	-	**规则1.1 DWS集群必须配置负载均衡**
2			**规则1.2 连接数据库完成所需操作后，必须关闭数据库连接（连接池场景除外）**
3			**规则1.3 开启的事务最后必须提交或回滚**

版权所有 © 华为云计算技术有限公司

编号	类别		规则/建议
4			规则1.4 应用侧使用连接池场景，其空闲超时配置必须小于服务侧的SESSION_TIMEOUT配置
5			规则1.5 应用侧使用连接池场景，如使用连接SET设置过参数，当将连接归还连接池前，必须进行参数重置
6			规则1.6 应用侧使用连接池场景，如使用连接创建过临时表，当将连接归还连接池前，必须手动清理所创建的临时表
7	对象设计规范	DATABASE对象设计	规则2.1 避免直接使用内置的DATABASE（如postgres、gaussdb等）
8			规则2.2 创建DATABASE时必须选择正确的数据库编码
9			规则2.3 创建DATABASE时必须选择正确的数据库兼容模式
10			建议2.4 存在关联计算的对象放在同一个DATABASE中
11		USER对象设计	规则2.5 禁止使用特殊权限用户运行业务，需遵循权限最小分配原则
12			规则2.6 禁止使用一个数据库用户运行所有业务
13		SCHEMA对象设计	建议2.7 不在其他USER的私有SCHEMA下创建对象
14		TABLESPACE对象设计	规则2.8 禁止自定义TABLESPACE表空间
15		TABLE对象设计（重点）	规则2.9 创建表时必须选择正确的分布方式和分布列
16			规则2.10 创建表时必须选择正确的存储方式
17			规则2.11 创建表时必须选择正确的分区策略
18			建议2.12 表字段的设计要遵循高效、准确原则
19			建议2.13 避免使用自增列或自增数据类型
20		INDEX对象设计（重点）	规则2.14 只创建必要的索引，创建索引必须选择合适的列和顺序
21			建议2.15 列存表通常可不建索引，极致性能场景需正确选择索引类型
22		VIEW对象设计	建议2.16 视图的嵌套需避免超过三层
23	SQL开发规范	DDL操作规范	建议3.1 DDL操作（CREATE除外）避免在业务高峰期和长事务中执行
24			规则3.2 DROP删除对象操作必须明确删除对象范围

版权所有 © 华为云计算技术有限公司

编号	类别		规则/建议
25		INSERT操作规范	规则3.3 INSERT多VALUES批插场景使用COPY替代
26			建议3.4 禁止针对普通列存表进行实时INSERT操作
27		UPDATE/DELETE操作规范	建议3.5 避免并发UPDATE/DELETE行存表的同一行
28			建议3.6 避免对列存表频繁或并发执行UPDATE/DELETE
29		SELECT操作（含所有语法中的查询部分）规范	规则3.7 禁止执行不下推的SQL
30			规则3.8 禁止多表关联时缺少关联条件
31			规则3.9 多表关联字段数据类型要保持一致
32			建议3.10 尽量避免对关联条件字段和过滤条件字段进行函数运算
33			建议3.11 资源高消耗型SQL需做好压测和并发管控
34			规则3.12 禁止针对行存大表的频繁COUNT
35			建议3.13 避免查询返回超大结果集（数据导出场景除外）
36			建议3.14 查询时避免使用“SELECT *”写法
37			建议3.15 谨慎使用递归语句(WITH RECURSIVE)，明确终止条件，确保递归可终止
38			建议3.16 访问对象（表，函数等）时带上SCHEMA名称
39			建议3.17 针对SQL标记注释，唯一标识SQL的归属
40			建议3.18 SQL语句的长度不要超过64K
41	外表功能开发规范	GDS外表	规则4.1 GDS服务须单独使用服务器部署在DWS集群外
42		协同分析外表	规则4.2 避免同时对多个协同分析外表进行跨集群并发访问
43	存储过程开发规范	-	建议5.1 避免使用复杂的存储过程，避免存储过程嵌套
44			规则5.2 存储过程内避免执行非CREATE类的DDL操作

版权所有 © 华为云计算技术有限公司

2.2 DWS 连接管理规范

规则 1.1 DWS 集群必须配置负载均衡

说明

违反规则的影响：

- 负载不均衡导致性能问题，严重情况会导致业务中断。
- CN故障场景，业务无法自恢复或恢复时间长。

方案建议：

- 配置ELB负载均衡，应用连接负载均衡IP。
- 配置JDBC负载均衡方式的操作参见**负载均衡方式配置JDBC**。

规则 1.2 连接数据库完成所需操作后，必须关闭数据库连接（连接池场景除外）

说明

违反规则的影响：

- 空闲连接过多，触发连接上限，导致新建连接报错。
- 空闲连接过多，内部占用过多资源，导致资源过载。

方案建议：

- 应用侧与数据库建连并使用完成后，手动关闭连接。
- 服务侧设置session_timeout参数，连接空闲超时自动断开。

规则 1.3 开启的事务最后必须提交或回滚

说明

违反规则的影响：

- 事务长时间不提交，持锁阻塞ALTER等操作，进而阻塞所有业务。
- 大量“idle in transaction”连接，触发连接上限，导致新建连接报错。

方案建议：

- 默认使用autocommit自动提交方式，如关闭autocommit，则必须手动提交。
- 显式start transaction开启的事务，执行完相关操作后，必须显式commit/rollback结束事务。

规则 1.4 应用侧使用连接池场景，其空闲超时配置必须小于服务侧的 SESSION_TIMEOUT 配置

说明

违反规则的影响：

- 连接池的连接被服务侧的空闲超时机制清理，连接复用效果下降。

方案建议：

- 将连接池的空闲超时参数设置为小于服务侧的SESSION_TIMEOUT值，或调大SESSION_TIMEOUT为大于连接池超时参数（不推荐）。

版权所有 © 华为云计算技术有限公司

规则 1.5 应用侧使用连接池场景，如使用连接 SET 设置过参数，将连接归还连接池前，必须进行参数重置

说明

违反规则的影响：

- 连接被其他业务复用时，可能会因复用其他业务设置的参数出现业务性能、业务报错等问题。

方案建议：

- 在将连接归还连接池之前，使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”重置参数。

注意：

在应用侧使用连接池的场景，如果在DWS服务侧通过GS_GUC RELOAD设置了全局的GUC参数，需要重启应用侧连接池该参数才可以生效，因为该设置只针对新建的连接生效，针对连接池中原有的连接不生效。

规则 1.6 应用侧使用连接池场景，如使用连接创建过临时表，将连接归还连接池前，必须手动清理所创建的临时表

说明

违反规则的影响：

- 连接被其他业务复用时，可能出现创建临时表报错问题。

方案建议：

- 在将连接归还连接池之前，使用DROP清理当前SESSION创建的临时表。

2.3 DWS 对象设计规范

2.3.1 DATABASE 对象设计规范

规则 2.1 避免直接使用内置的 DATABASE（如 postgres、gaussdb 等）

说明

违反规则的影响：

- 内置的数据库编码、兼容模式不符合业务要求时，需要重新迁移数据。
- 所有业务均使用内置数据库，影响变更耗时。

方案建议：

- 根据实际业务需要，合理创建专用DATABASE并划分给业务使用。

规则 2.2 创建 DATABASE 时必须选择正确的数据库编码

说明

违反规则的影响：

- 选错数据库编码可能导致数据乱码，且不支持直接修改数据库编码，需重新建库重新导入数据。

方案建议：

- 通常建议建库时指定ENCODING为UTF-8编码，特殊场景根据实际情况而定。

版权所有 © 华为云计算技术有限公司

规则 2.3 创建 DATABASE 时必须选择正确的数据库兼容模式

说明

违反规则的影响：

- 选错数据库兼容模式，会导致从其他厂商的数据库迁移到DWS后出现行为不一致问题，且不支持直接修改数据库兼容模式，需重新建库重新导数

方案建议：

- 根据源库端数据库类型，在DWS中建库时通过DBCOMPATIBILITY指定兼容模式，当前支持Teradata、Oracle和MySQL等多种兼容模式。

建议 2.4 存在关联计算的对象创建在同一个 DATABASE 中

说明

违反规则的影响：

- 跨库访问无论使用哪种方案，性能均劣于同一DATABASE内的关联操作。

方案建议：

- 创建多个DATABASE的场景，将需要执行关联计算的对象，创建在同一DATABASE中。

2.3.2 USER 对象设计规范

规则 2.5 禁止使用特殊权限用户运行业务，需遵循权限最小分配原则

说明

违反规则的影响：

- 特殊权限用户如管理员用户，均具有特殊用途，使用此类用户运行业务有安全和管控风险。

方案建议：

- 使用普通用户运行业务，特殊权限类用户仅用于管理类操作。

规则 2.6 禁止使用一个数据库用户运行所有业务

说明

违反规则的影响：

- 同一个数据库用户运行所有业务不利于业务管控，异常场景无法针对特定用户做紧急隔离。

方案建议：

- 根据用途规划管理员、业务运行账号、运维账号等。
- 根据业务模块进行用户细分，以便通过用户进行业务和资源的划分和管控。

版权所有 © 华为云计算技术有限公司

2.3.3 SCHEMA 对象设计规范

建议 2.7 不在其他 USER 的私有 SCHEMA 下创建对象

说明

私有SCHEMA是指创建USER时自带的同名SCHEMA，该SCHEMA为该USER私有。

违反规则的影响：

- 在其他用户私有SCHEMA下创建对象，对象权限不受创建者控制，OWNER也非创建者。

方案建议：

- 在自身私有SCHEMA或自身创建SCHEMA下创建对象，不在其他用户私有SCHEMA下创建对象。

2.3.4 TABLESPACE 对象设计规范

规则 2.8 禁止自定义 TABLESPACE 表空间

说明

违反规则的影响：

- 分布式场景建表使用自定义表空间，导致表数据无法按照各DN分布式存储，出现存储倾斜。

方案建议：

- 创建表对象使用内置默认表空间。

2.3.5 TABLE 对象设计规范（重点）

规则 2.9 创建表时必须选择正确的分布方式和分布列

说明

违反规范的影响：

- 分布式和分布列选择错误，导致表数据存储倾斜，访问性能下降，严重情况会触发存储和计算资源过载。

方案建议：

- 创建表时通过DISTRIBUTE BY显式指定分布方式和分布列，分布列选择原则如下表所示。

表 2-2 分布列选择原则

分布方式	描述	适用场景
Hash	表数据按照分布列生成的hash值与DN实例的映射关系，将数据分布到各DN实例。 • **优点**：每个DN仅包含部分数据，占用整体空间小。 • **缺点**：数据分布的均匀程度强依赖分布列的选择；JOIN关联条件不包含各自分布列的场景存在节点间数据通信的消耗。	大表、事实表。

版权所有 © 华为云计算技术有限公司

分布方式	描述	适用场景
RoundRobin	表数据按照轮询的方式依次分布到各DN实例。 ● **优点**：每个DN仅包含部分数据，占用整体空间小；数据轮询均匀分布，不依赖分布列，不存在数据存储倾斜问题。 ● **缺点**：无法通过分布列条件消除和减少节点间通信，此类场景性能不如HASH。	大表、事实表，无合适分布列的表。
Replication	表中的全量数据在集群的每一个DN实例上保留一份。 ● **优点**：每个DN上都有此表的全量数据，JOIN操作中可以完全避免节点间数据通信，从而减少网络开销，同时减少了STREAM线程启停开销。 ● **缺点**：每个DN都保留了表的完整数据，数据的冗余，占用更多存储空间。	小表、维度表。

规则 2.10 创建表时必须选择正确的存储方式

说明

违反规范的影响：

- 行存表使用不当导致查询场景性能差，资源过载。
- 列存表使用不当导致CU膨胀，性能差，资源过载。

方案建议：

- 创建表时通过orientation参数显式指定表的存储类型，存储类型的选择原则如下表所示。

表 2-3 存储类型选择

存储方式	优势场景	劣势场景
行存(row)	● DML增删改：UPDATE和DELETE操作多的场景。 ● DML查询：点查询(返回记录少，基于索引的简单查询)。	DML查询：统计分析类查询（group，join的数据量大的场景）。 **注意** **创建行存表（orientation = row）时，禁止指定compress属性，禁止使用行存压缩表。**

版权所有 © 华为云计算技术有限公司

存储方式	优势场景	劣势场景
列存(column)	• DML增删改：INSERT批量导入场景（单次单分区入库量接近或大于6w*DN数）。 • DML查询：统计分析类查询（group，join的数据量大的场景）。	• DML增删改：UPDATE/DELETE多的场景、INSERT小批量插入的场景。 • DML查询：高并发的点查询。
行列混存(hstore&hstore_opt)	DWS的行列混存表仅**9.1.0**及以上集群版本支持，其提升了实时入库的能力，兼顾部分行存和列存的优点。 • DML增删改：INSERT/UPSERT/COPY实时入库场景和批量入库（除COPY微批场景外，hstore_opt全面优于hstore）。 • DML查询：统计分析类查询（group，join的数据量大的场景）。	主要是极端的TP（事务处理）类场景。 • DML增删改：高频并发UPDATE/DELETE场景。 • DML查询：高频并发点查场景。

规则 2.11 创建表时必须选择正确的分区策略

说明

违反规范的影响：

分区的优点如下，如不做分区，其查询性能和数据治理效率会下降，数据量越大这种劣化越大。

- 改善查询性能：对分区对象的查询可以仅搜索业务关注的分区，提高检索效率。
- 提升数据治理效率：如数据生命周期管理场景，针对历史分区执行TRUNCATE/DROP PARTITION，效率和效果远优于DELETE。

方案建议：

- 针对包含时间类型字段的表设计分区。

表 2-4 分区策略选择

分区策略	描述	适用场景
范围分区（Range Partitioning）	根据分区键值的范围，将数据存储到不同的分区中，分区键范围连续但不重叠。	1. 日期或者时间类的字段作为分区键。 2. 查询中大多包含分区键作为过滤条件。 3. 定期按照分区键清理数据。
列表分区（List Partitioning）	根据分区键值的列表进行分区，各分区的列表值不重复	1. 特定数量的枚举值作为分区键值。 2. 查询中大多包含分区键作为过滤条件。

版权所有 © 华为云计算技术有限公司

建议 2.12 表字段的设计要遵循高效、准确原则

说明

违反规范的影响：

- 存储空间大、查询效率降低。

方案建议：

1. **选择最高效的类型**。
   - 如果能选择整型就不选择浮点型或字符型。
   - 使用变长字符类型时根据数据特征指定最大长度。
2. **选择最准确的类型**。
   - 使用时间类型存储时间，不使用字符类型存储时间。
   - 使用满足需求的最小数值类型，如果int或smallint够用，就不用bigint而浪费空间。
3. **正确使用约束**。
   - 明确不存在NULL值的字段加上NOT NULL约束，优化器会在特定场景下对其进行自动优化。
   - 业务层面能补全的字段，不要使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
4. **避免非必要类型转换**。
   - 当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。
   - 不同类型的比较操作会导致数据类型转换，可能导致索引和分区剪枝失效，影响查询性能。

建议 2.13 避免使用自增列或自增数据类型

说明

违反规范的影响：

- 自增序列或自增数据类型在大量使用时，会造成GTM压力过大及序列生成速度慢。

方案建议：

- 如果只是想获取一个唯一标识，可以使用UUID。
- 如果必须使用自增序列，在没有严格递增的需求，可设置CACHE，比如1000，降低GTM压力。

版权所有 © 华为云计算技术有限公司

2.3.6 INDEX 对象设计规范（重点）

规则 2.14 只创建必要的索引，创建索引必须选择合适的列和顺序

说明

违反规范的影响：

- 冗余索引浪费空间，索引多影响入库效率。
- 组合索引中列顺序错误，影响查询效率。

最佳实践：

索引的使用需兼顾以下条件：

- 索引列必须是常用于过滤条件或JOIN关联条件的列。
- 索引列必须是DISTINCT值多的列。
- 创建多列组合索引时，DISTINCT值多的列往前放。
- 单表索引个数控制在5个以内，可通过组合索引控制索引的个数。
- 数据批量增删改场景，建议先删除索引，完成增删改后再加回索引，提升批量操作性能（实时访问会有影响）。

建议 2.15 列存表通常可不建索引，极致性能场景需正确选择索引类型

说明

违反规范的影响：

- 错误的索引对列存的访问无任何性能帮助，反而可能影响查询性能。

方案建议：

1. 创建索引时指定索引类型，避免使用默认的PSORT类型索引。
2. 极端点查（海量数据中检索极少数据）场景，可使用B-Tree类型索引。
3. 范围查询性能要求高的场景，可以创建Partial Cluster Key（局部聚簇，简称PCK）通过min/max稀疏索引实现事实表快速过滤扫描。PCK的选取遵循以下原则：
   - 【关注】一张表上只能建立一个PCK，一个PCK可以包含多列，但是一般不建议超过2列。
   - 【建议】针对表达式过滤条件列创建PCK（形如col op const，op为操作符 =、>、>=、<=、<，const为常量值）。

2.3.7 VIEW 对象设计规范

建议 2.16 视图的嵌套需避免超过三层

说明

违反规范的影响：

- 视图嵌套过深导致执行计划不稳定、耗时不稳定。
- 视图依赖的对象重建风险高，锁冲突发生概率增大。

方案建议：

- 创建视图直接基于物理表查询，不建议嵌套视图。

2.4 DWS SQL 开发规范

版权所有 © 华为云计算技术有限公司

2.4.1 DDL 操作规范

建议 3.1 DDL 操作（CREATE 除外）避免在业务高峰期和长事务中执行

说明

违反规范的影响：

DDL操作普遍持锁级别高，如ALTER、DROP、TRUNCATE、REINDEX、VACUUM FULL等，执行时会造成业务等锁阻塞。

- 高峰期执行持锁级别高的DDL操作，造成业务等锁阻塞。
- 长事务中执行持锁级别高的DDL操作，长时间持锁或等锁，均造成业务等锁阻塞。

方案建议：

- 根据业务周期，选择低峰期或运维时间窗执行DDL操作，明确DDL执行环境和耗时，避免锁阻塞。

规则 3.2 DROP 删除对象操作必须明确删除对象范围

危险

违反规范的影响：

DROP对象操作（如DATABASE、USER/ROLE、SCHEMA、TABLE、VIEW等对象）存在数据丢失风险，尤其含带CASCADE级联删除场景，会将关联的对象一并删除。

- DROP DATABASE：整个DATABASE中所有对象被删除。
- DROP USER：USER对象、USER所拥有的SCHEMA、TABLE等对象均被删除。
- DROP SCHEMA：整个SCHEMA中的所有对象被删除。
- DROP TABLE：TABLE对象，依赖TABLE的INDEX、VIEW等对象均被删除。

方案建议：

- DROP操作谨慎，操作前考虑数据备份。

2.4.2 INSERT 操作规范

规则 3.3 INSERT 多 VALUES 批插场景使用 COPY 替代

说明

违反规范的影响：

- 多VALUES解析耗时、耗资源，入库效率低。

方案建议：

- 前端使用COPY类接口（如JDBC的CopyManger等）代替INSERT VALUES。

版权所有 © 华为云计算技术有限公司

建议 3.4 禁止针对普通列存表进行实时 INSERT 操作

说明

违反规范的影响：

- 针对普通列存表实时小批量入库会导致小CU膨胀严重，影响存储空间和查询性能。

方案建议：

- 实时INSERT场景评估单次入库数据量和数据总量，总量小的场景可以改为行存表。
- 实时INSERT场景前端攒批，保证单次、单表、单分区、单DN入库数据量接近6W，建议最低不少于5K。
- 实时INSERT场景使用Hstore列存表（8.3.0及以上集群版本）。

2.4.3 UPDATE&DELETE 操作规范

建议 3.5 避免并发 UPDATE/DELETE 行存表的同一行

说明

违反规范的影响：

- 并发UPDATE/DELETE行存表可能导致行锁阻塞和分布式死锁风险，导致业务报错和性能下降。

方案建议：

- 按主键或分布列进行分组UPDATE/DELETE，组间并行，组内串行。

建议 3.6 避免对列存表频繁或并发执行 UPDATE/DELETE

说明

违反规范的影响：

- 针对列存表频繁执行UPDATE/DELETE造成CU膨胀，导致空间膨胀和访问性能下降。
- 针对列存表并发执行UPDATE/DELETE，导致行锁阻塞和分布式死锁风险，导致业务报错和性能下降。

方案建议：

- 频繁执行UPDATE/DELETE操作的表需设计为行存表。
- 按主键或分布列进行分组UPDATE/DELETE，组间并行，组内串行。

2.4.4 SELECT 操作规范

规则 3.7 禁止执行不下推的 SQL

说明

DWS为分布式架构，SQL语句必须下推才能充分利用分布式的计算资源，达到性能最优。

违反规范的影响：

- 不下推的SQL执行性能差，严重情况会导致CN资源瓶颈，影响整体业务。

方案建议：

- 不使用不下推的语法和不下推的函数，具体参考**语句下推调优**。

版权所有 © 华为云计算技术有限公司

规则 3.8 禁止多表关联时缺少关联条件

说明

违反规范的影响：

- 多表关联时不指定关联条件就是求笛卡尔积，极易导致结果集膨胀，造成性能和资源过载风险。

方案建议：

- 多表关联时明确每张表的过滤条件和关联条件，避免出现缺少过滤条件和关联条件的情况。

规则 3.9 多表关联字段数据类型要保持一致

说明

违反规范的影响：

- 关联字段类型不一致，导致额外的类型转换开销，且影响数据重分布的策略，无法生成最优计划。

方案建议：

- 多表关联场景，关联的字段均使用同样的数据类型。

建议 3.10 尽量避免对关联条件字段和过滤条件字段进行函数运算

说明

违反规范的影响：

- 对关联条件字段和过滤条件字段进行函数运算，导致优化器无法获取准确的字段统计信息，无法生成最优计划，影响执行性能。

方案建议：

- 关联条件字段之间直接比较，如有需要运算后比较的场景需在数据入库前进行预处理。
- 过滤条件和常量比较时，只对常量列进行函数运算，字段列不进行函数运算，例如：

```
SELECT id, from_image_id, from_person_id, from_video_id
FROM face_data
WHERE SS.DEL_FLAG = 'N'
AND NVL(SS.DELETE_FLAG, 'N') = 'N'
改写为：
SELECT id, from_image_id, from_person_id, from_video_id
FROM face_data
where SS.DEL_FLAG = 'N'
AND (SS.DELETE_FLAG = 'N' or SS.DELETE_FLAG is null)
```

建议 3.11 资源高消耗型 SQL 需做好压力测试和并发管控

说明

违反规范的影响：

- 存储和计算资源过载，整体运行性能下降。

方案建议：

高资源消耗型SQL的主要特征：

- 大量UNION ALL
- 大量AGG（ COUNT DISTINCT、MAX等 ）
- 大量表JOIN
- 大量STREAM算子（ 计划维度 ）

针对上述SQL需进行压力测试和并发管控，如果超出资源能力，则必须进行业务优化后再重新评估上线。

版权所有 © 华为云计算技术有限公司

规则 3.12 禁止针对行存大表的频繁 COUNT

说明

磁盘能力强的场景（如SSD），本规则可适当放宽，但仍需关注I/O消耗情况。

违反规范的影响：

- 行存表的COUNT需要扫描全表，大表场景频繁COUNT会消耗大量I/O，如触发I/O瓶颈会导致整体性能问题。

方案建议：

- 建议降低COUNT频率、使用结果缓存、分区级统计等方式，降低COUNT的I/O消耗。

建议 3.13 避免查询返回超大结果集（数据导出场景除外）

说明

违反规范的影响：

- 在实际不需要查看所有结果的场景，查询超大结果集会浪费大量资源。

方案建议：

- 查询使用LIMIT，只返回必要数量的结果。
- 真实需要查询大量结果集的场景，使用游标进行分段获取，合理设置FETCH SIZE。

建议 3.14 查询时避免使用“SELECT *”写法

说明

违反规范的影响：

- 查询实际不需要的列，增加计算负担，浪费计算资源。

方案建议：

- SELECT时明确列出查询所需字段，提升查询的有效性能。

建议 3.15 谨慎使用递归语句(WITH RECURSIVE)，明确终止条件，确保递归可终止

说明

违反规范的影响：

- 无明确终止条件，递归陷入死循环，无法完成。
- 重复数据过多，递归产生大量重复数据，占用大量资源。

方案建议：

- 根据业务表数据量和数据特征设计合理的递归终止条件。

建议 3.16 访问对象（表，函数等）时带上 SCHEMA 名称

说明

违反规范的影响：

- 不指定SCHEMA名称前缀，实际会根据当前search_path中表空间列表，依次搜索所有表空间直到找到匹配的表作为目标表，可能因SCHEMA切换导致访问到非预期的表。

方案建议：

- 访问表和函数对象时显式指定“SCHEMA.”前缀，增强可读性、稳定性、可移植性。

版权所有 © 华为云计算技术有限公司

建议 3.17 针对 SQL 标记注释，唯一标识 SQL 的归属

说明

违反规范的影响：

- 业务溯源能力较差，只能通过数据库、用户名、客户端IP信息找开发人员确认。

方案建议：

- 建议使用query_band，例如：
  ```
  SET query_band='JobName=abc;AppName=test;UserName=user';
  ```
- 每个SQL开头标记注释，唯一标识SQL的归属，方便问题定位及应用性能分析，命名建议为：
  /* 模块名_工具名_作业名_步骤 */，如: /* mca_python_xxxxxx_step1 */ insert into xxx select … from

建议 3.18 SQL 语句的长度不要超过 64K

说明

违反规范的影响：

- SQL解析耗时长、维护难度大，如频繁执行会导致日志膨胀严重。

方案建议：

- 控制业务SQL的长度，避免超过64K。

2.5 DWS 外表功能开发规范

规则 4.1 GDS 服务需单独使用服务器部署在 DWS 集群外

说明

违反规范的影响：

GDS如果部署在DWS集群内，会与DWS集群CN/DN节点发生资源争抢，导致双方性能同时劣化。

方案建议：

- GDS服务单独使用服务器部署在DWS集群外。
- GDS所在服务器的磁盘能力、GDS服务器与DWS集群间网络带宽都要按需规划。

规则 4.2 避免同时对多个协同分析外表进行跨集群并发访问

说明

原理说明：在A集群通过协同分析访问B集群数据时，A集群所有DN会与B集群CN建立连接和活跃会话。

违反规范的影响：

B集群（远端集群）中CN压力过大，导致连接和活跃会话资源超限，访问异常。

方案建议：

应尽量使用外表单表访问并避免并发，避免多外表关联查询；无法避免并发场景时，并发数需根据A集群DN数及B集群常规业务量进行计算和限制，并适当调大max_active_statements和max_connections值。

版权所有 © 华为云计算技术有限公司

2.6 DWS 存储过程开发规范

建议 5.1 避免使用复杂的存储过程，避免存储过程嵌套

说明

违反规范的影响：

- 复杂和嵌套的存储过程维护成本高，故障定位难度大，恢复耗时长。

方案建议：

- 不使用存储过程或只使用一层存储过程，不嵌套。
- 开发存储过程设计对应的日志表，将关键步骤前后的信息记录到日志表中，操作步骤如下。

保存并查看日志操作步骤。

步骤1 创建日志表。

```
CREATE TABLE func_exec_log
(
id varchar2(32) default lower(sys_guid()),
pro_name varchar2(60),
exec_times int,
log_date date,
deal_date date,
log_message text
);
```

步骤2 创建表和导入数据。

```
CREATE TABLE demo_table(data_id int, data_number int);
INSERT INTO demo_table values(generate_series(1,1000),generate_series(1,1000));
```

步骤3 创建业务存储过程。

```
CREATE OR REPLACE FUNCTION demo_table_process(out exe_info text)
LANGUAGE plpgsql
AS $$
declare v_count int;
pro_result text;
fun_name   text;
exec_times   int;
begin
fun_name := 'demo_table_process';
select nvl(max(exec_times), '0') + 1 into exec_times from func_exec_log where pro_name = fun_name;
--业务表插入数据
insert into demo_table values (dbms_random.value(1, 1000)::int,generate_series(1,
dbms_random.value(10000, 20000)::int));
get diagnostics v_count = ROW_COUNT;
exe_info = sysdate || '# step1:insert count:' || v_count || ' rows;';
--删除业务表指定数据
delete from demo_table where data_id = dbms_random.value(1, 1000)::int;
get diagnostics v_count = ROW_COUNT;
exe_info = exe_info || sysdate || '# step2:delete count:' || v_count || ' rows;';
--更新业务表数据
update demo_table set data_number = dbms_random.value(1, 100)::int where data_id =
dbms_random.value(1, 1000)::int;
exe_info = exe_info || sysdate || '# step3:update count:' || sql%rowcount || ' rows';
--在整个程序结束前记录日志，也可以在每个步骤结束后分别记录日志，也可以创建记录日志的函数供调用，灵活使用即可
insert into func_exec_log(pro_name, exec_times, log_date, deal_date, log_message) values
(fun_name,exec_times,sysdate,split_part(regexp_split_to_table(exe_info, ';'), '#',
1),split_part(regexp_split_to_table(exe_info, ';'), '#', 2));
--EXCEPTION用于保证当insert/update/delete等操作步骤异常退出时，也能够正常记录日志
EXCEPTION
WHEN OTHERS THEN
```


版权所有 © 华为云计算技术有限公司

```
pro_result := exe_info || sysdate || '# exception error message is: ' || sqlerrm;
insert into func_exec_log(pro_name, exec_times, log_date, deal_date, log_message)
values(fun_name,exec_times,sysdate,split_part(regexp_split_to_table(pro_result, ';'), '#',
1),split_part(regexp_split_to_table(pro_result, ';'), '#', 2));
END; $$;
```

步骤4 调用存储过程（正常执行）。

```
SELECT demo_table_process();
```

步骤5 查看日志（确认业务运行情况）。

```
SELECT * FROM func_exec_log ORDER BY log_date desc,deal_date,log_message;
```

```
demodb=# select * from func_exec_log order by log_date desc,deal_date,log_mesage;
                id                |      pro_name      | exec_times |      log_date       |      deal_date      |          log_mesage
----------------------------------+--------------------+------------+---------------------+---------------------+------------------------------
 637343d9f2f10ec605c7687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:11 | step1:insert count:19125 rows
 637343d9fe850e3105c8687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:33 | step2:delete count:22 rows
 637343d9068a0fd805c9687ff700fffe | demo_table_process |          1 | 2022-11-15 15:46:34 | 2022-11-15 15:46:34 | step3:update count:15 rows
(3 rows)
```

步骤6 再次调用存储过程（构造执行异常）。

```
SELECT demo_table_process();  --先删除demo_table的data_number列构造异常，之后再调用
```

步骤7 查看日志（确认业务运行情况）。

----结束

规则 5.2 存储过程内避免执行非 CREATE 类的 DDL 操作

说明

违反规范的影响：

- 存储过程是一个大事务，如果有非CREATE类DDL操作（持锁级别高），整个存储过程执行时间窗内会阻塞外部对相关表的访问。

方案建议：

- 存储过程中不使用非CREATE类的DDL操作，如必须使用，严格评估存储过程耗时和DDL的影响耗时，与外部访问业务错峰执行。

2.7 DWS 各对象设计详细规则

2.7.1 DWS 数据库对象命名规则

数据库对象命名需要满足约束：长度不超过63个字符，以字母或下划线开头，中间字符可以是字母、数字、下划线、$。数据库字符集为GBK、UTF8、SQL_ASCII时对象命名支持中文汉字，其中UTF8、SQL_ASCII字符集下，一个中文汉字占3~4个字符；GBK字符集下，一个中文汉字占2个字符。Latin1字符集不支持中文汉字命名。字符集格式在创建数据库时指定，详细内容可参考**CREATE DATABASE**。

- 【建议】避免使用保留或者非保留关键字命名数据库对象。

  说明

  可以使用**SELECT * FROM pg_get_keywords();**查询DWS的关键字或者参考**关键字**章节。

- 【建议】避免使用双引号括起来的字符串来定义数据库对象名称，DWS中使用双引号将数据库对象名称括起来时表示对大小写敏感。数据库对象名称大小写敏感会使定位问题难度增加。
- 【建议】数据库对象命名风格务必保持统一。
  - 增量开发的业务系统或进行业务迁移的系统，建议遵守历史的命名风格。

版权所有 © 华为云计算技术有限公司

  - 数据库对象名称由字母、数字和下划线组成，并且不能由数字开头。建议使用多个单词组成，以下划线分割。
  - 数据库对象名称最好能够望文知意，尽量避免使用自定义缩写（可以使用通用的术语缩写进行命名）。例如，在命名中可以使用具有实际业务含义的英文词汇或汉语拼音，但规则应该在集群范围内保持一致。
  - 变量名的关键是要具有描述性，即变量名称要有一定的意义，变量名要有前缀标明该变量的类型。
- 【建议】表对象的命名应该可以表征该表的重要特征。例如，在表对象命名时区分该表是普通表、临时表还是非日志表：
  - 普通表名按照数据集的业务含义命名。
  - 临时表以“tmp_+后缀”命名。
  - 非日志表以“ul_+后缀”命名。
  - 外表以“f_+后缀”命名。

2.7.2 DWS 数据库对象设计规则

2.7.2.1 DWS Database 和 Schema 设计规则

DWS中可以使用Database和Schema实现业务的隔离，区别在于Database的隔离更加彻底，各个Database之间共享资源极少，可实现连接隔离、权限隔离等，Database之间无法直接互访。Schema隔离的方式共用资源较多，可以通过GRANT与REVOKE语法便捷地控制不同用户对各Schema及其下属对象的权限。

- 从便捷性和资源共享效率上考虑，推荐使用Schema进行业务隔离。
- 建议系统管理员创建Schema和Database，再赋予相关用户对应的权限。

Database 设计建议

- 【建议】在实际业务中，根据需要创建新的Database，不建议直接使用集群默认的gaussdb数据库。
- 【建议】一个集群内，用户自定义的Database数量建议不超过3个。
- 【建议】为了适应全球化的需求，使数据库编码能够存储与表示绝大多数的字符，建议创建Database的时候使用UTF-8编码。
- 【关注】创建Database时，需要重点关注字符集编码(ENCODING)和兼容性(DBCOMPATIBILITY)两个配置项。DWS支持Oracle、Teradata和MySQL三种兼容模式，分别兼容Oracle、Teradata和MySQL语法，不同兼容模式下的语法行为可能有一些差异。详细内容可参考**Oracle、Teradata和MySQL语法兼容性差异**。
- 【关注】Database的owner默认拥有该Database下所有对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

Schema 设计建议

- 【关注】如果该用户不具有sysadmin权限或者不是该Schema的owner，要访问Schema下的对象，需要同时给用户赋予Schema的usage权限和对象的相应权限。
- 【关注】如果要在Schema下创建对象，需要授予操作用户该Schema的CREATE权限。
- 【关注】Schema的owner默认拥有该Schema下对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

版权所有 © 华为云计算技术有限公司

2.7.2.2 DWS 表设计规则

DWS是分布式架构。数据分布在各个DN上。总体上讲，良好的表设计需要遵循以下原则：

- 【关注】将表数据均匀分布在各个DN上。数据均匀分布，可以防止数据在部分DN上集中分布，从而导致因存储倾斜造成集群有效容量下降。通过选择合适的分布列，可以避免数据倾斜。
- 【关注】将表的扫描压力均匀分散在各个DN上。避免扫描压力集中在部分DN上，而导致性能瓶颈。例如，在事实表上使用等值过滤条件时，将会导致扫描压力不均匀。
- 【关注】减少需要扫描的数据量。通过分区表的剪枝机制可以大幅减少数据的扫描量。
- 【关注】尽量减少随机I/O。通过聚簇/局部聚簇可以实现热数据的连续存储，将随机I/O转换为连续I/O，从而减少扫描的I/O代价。
- 【关注】尽量避免数据shuffle。shuffle是指在物理上，数据从一个节点传输到另一个节点。shuffle占用了大量宝贵的网络资源，减小不必要的数据shuffle，可以减少网络压力，使数据的处理本地化，提高集群的性能和可支持的并发度。通过对关联条件和分组条件的仔细设计，能够尽可能地减少不必要的数据shuffle。

选择存储方案

【建议】表的存储类型是表定义设计的第一步，用户业务类型是决定表的存储类型的主要因素，表存储类型的选择依据请参考**表2-5**。

表 2-5 表的存储类型及场景

存储类型	适用场景
行存	• 点查询(返回记录少，基于索引的简单查询)。 • 增、删、改操作较多的场景。
列存	• 统计分析类查询 (关联、分组操作较多的场景)。 • 即席查询（查询条件不确定，行存表扫描难以使用索引）。

对于分析场景，建表需显式设置ORIENTATION 选项为列存。

```
CREATE TABLE  public.t1
(
id integer not null,
data integer,
age integer
)
WITH (ORIENTATION =COLUMN);
```

选择分布方案

【建议】表的分布方式的选择一般遵循以下原则：

版权所有 © 华为云计算技术有限公司

表 2-6 表的分布方式及使用场景

分布方式	描述	适用场景
Hash	表数据通过Hash方式散列到集群中的所有DN上。	数据量较大的事实表。
Replication	集群中每一个DN都有一份全量表数据。	维度表、数据量较小的事实表。
Roundrobin	表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。	数据量较大的事实表，且使用Hash分布时找不到合适的分布列。

选择分区方案

当表中的数据量很大时，应当对表进行分区，一般需要遵循以下原则：

- 【建议】使用具有明显区间性的字段进行分区，比如日期、区域等字段上建立分区。
- 【建议】分区名称应当体现分区的数据特征。例如，关键字+区间特征。
- 【建议】将分区上边界的分区值定义为MAXVALUE，以防止可能出现的数据溢出。

典型的分区表定义如下：

```
CREATE TABLE staffS_p1
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(4,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
PARTITION BY RANGE (HIRE_DATE)
(
   PARTITION HIRE_19950501 VALUES LESS THAN ('1995-05-01 00:00:00'),
   PARTITION HIRE_19950502 VALUES LESS THAN ('1995-05-02 00:00:00'),
   PARTITION HIRE_maxvalue VALUES LESS THAN (MAXVALUE)
);
```

选择分布键

Hash表的分布键选取至关重要，如果分布键选择不当，可能会导致数据倾斜，从而导致查询时，I/O负载集中在部分DN上，影响整体查询性能。因此，在确定Hash表的分布策略之后，需要对表数据进行倾斜性检查，以确保数据的均匀分布。分布键的选择一般需要遵循以下原则：

- 【建议】选作分布键的字段取值应该比较离散，以便数据能在各个DN上均匀分布。当单个字段无法满足离散条件时，可以考虑使用多个字段一起作为分布键。一般情况下，可以考虑选择表的主键作为分布键。例如，在人员信息表中选择证件号码作为分布键。

版权所有 © 华为云计算技术有限公司

- 【建议】在满足第一条原则的情况下，尽量不要选取在查询中存在常量过滤条件的字段作为分布键。例如，在表dwcjk相关的查询中，字段zqdh存在常量过滤条件“zqdh='000001'”，那么就应当尽量不选择zqdh字段作为分布键。
- 【建议】在满足前两条原则的情况下，尽量选择查询中的关联条件为分布键。当关联条件作为分布键时，Join任务的相关数据都分布在DN本地，将极大减少DN之间的数据流动代价。

2.7.2.3 DWS 字段设计规则

选择数据类型

在字段设计时，基于查询效率的考虑，一般遵循以下原则：

- 【建议】尽量使用高效数据类型。

  选择数值类型时，在满足业务精度的情况下，选择数据类型的优先级从高到低依次为整数、浮点数、NUMERIC。
- 【建议】当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。
- 【建议】对于字符串数据，建议使用变长字符串数据类型，并指定最大长度。请务必确保指定的最大长度大于需要存储的最大字符数，避免超出最大长度时出现字符截断现象。除非明确知道数据类型为固定长度字符串，否则，不建议使用CHAR(n)、BPCHAR(n)、NCHAR(n)、CHARACTER(n)。

  关于字符串类型的详细说明，请参见**常用字符串类型介绍**。

常用字符串类型介绍

在进行字段设计时，需要根据数据特征选择相应的数据类型。字符串类型在使用时比较容易混淆，下表列出了DWS中常见的字符串类型：

表 2-7 常用字符串类型

名称	描述	最大存储空间
CHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
CHARACTER(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
NCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
BPCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB

版权所有 © 华为云计算技术有限公司

名称	描述	最大存储空间
VARCHAR(n)	变长字符串，n描述了可以存储的最大字节长度。	10MB
CHARACTER VARYING(n)	变长字符串，n描述了可以存储的最大字节长度；此数据类型和VARCHAR(n)是同一数据类型的不同表达形式。	10MB
VARCHAR2(n)	变长字符串，n描述了可以存储的最大字节长度，此数据类型是为兼容Oracle类型新增的，行为和VARCHAR(n)一致。	10MB
NVARCHAR2(n)	变长字符串，n描述了可以存储的最大字符长度。	10MB
TEXT	不限长度(不超过1GB-8203字节)变长字符串。	1GB-8203字节

2.7.2.4 DWS 约束设计规则

DEFAULT 和 NULL 约束

- 【建议】如果能够从业务层面补全字段值，则不建议使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
- 【建议】给明确不存在NULL值的字段加上NOT NULL约束，优化器会在特定场景下对其进行自动优化。
- 【建议】给可以显式命名的约束显式命名。除了NOT NULL和DEFAULT约束外，其他约束都可以显式命名。

局部聚簇

Partial Cluster Key（局部聚簇，简称PCK）是列存表的一种局部聚簇技术，在DWS中，使用PCK可以通过min/max稀疏索引实现事实表快速过滤扫描。PCK的选取遵循以下原则：

- 【关注】一张表上只能建立一个PCK，一个PCK可以包含多列，但是一般不建议超过2列。
- 【建议】在查询中的简单表达式过滤条件上创建PCK。这种过滤条件一般形如col op const，其中col为列名，op为操作符 =、>、>=、<=、<，const为常量值。
- 【建议】在满足上面条件的前提下，选择在distinct值比较少的列上建PCK。

唯一约束

- 【关注】行存表与列存表都支持唯一约束。
- 【建议】从命名上明确标识唯一约束，例如，命名为“UNI+构成字段”。

版权所有 © 华为云计算技术有限公司

主键约束

- 【关注】行存表与列存表都支持主键约束。
- 【建议】从命名上明确标识主键约束，例如，将主键约束命名为“PK+字段名”。

检查约束

- 【关注】行存表支持检查约束，而列存表不支持。
- 【建议】从命名上明确标识检查约束，例如，将检查约束命名为“CK+字段名”。

2.7.2.5 DWS 视图和关联表设计规则

视图设计

- 【建议】除非视图之间存在强依赖关系，否则不建议视图嵌套。
- 【建议】视图定义中尽量避免排序操作。

关联表设计

- 【建议】表之间的关联字段应该尽量少。
- 【建议】关联字段的数据类型应该保持一致。
- 【建议】关联字段在命名上，尽可能体现出明显的关联关系。例如，采用同样名称来命名。

2.7.3 DWS JDBC 配置规则

目前，DWS相关的第三方工具可以通过JDBC进行连接的，此部分将介绍工具配置时的注意事项。

连接参数

- 【关注】第三方工具通过JDBC连接DWS时，JDBC向DWS发起连接请求，会默认添加以下配置参数，详见JDBC代码ConnectionFactoryImpl类的实现。

```
params = {
{ "user", user },
{ "database", database },
{ "client_encoding", "UTF8" },
{ "DateStyle", "ISO" },
{ "extra_float_digits", "2" },
{ "TimeZone",  createPostgresTimeZone() },
};
```

  这些参数可能会导致JDBC客户端的行为与gsql客户端的行为不一致，例如，Date数据显示方式、浮点数精度表示、timezone显示。

  如果实际期望和这些配置不符，建议在Java连接设置代码中显式设定这些参数。

- 【建议】通过JDBC连接数据库时，应该保证下面两个时区设置一致：
  - JDBC客户端所在主机的时区。
  - DWS集群所在主机的时区。

版权所有 © 华为云计算技术有限公司

fetchsize

【关注】在应用程序中，如果需要使用fetchsize，必须关闭autocommit。开启autocommit，会使fetchsize配置失效。

autocommit

【建议】在JDBC向DWS申请连接的代码中，建议显式打开autocommit开关。如果基于性能或者其他方面考虑，需要关闭autocommit时，需要应用程序自己来保证事务的提交。例如，在指定的业务SQL执行完之后做显式提交，特别是客户端退出之前务必保证所有的事务已经提交。

释放连接

【建议】推荐使用连接池限制应用程序的连接数。每执行一条SQL就连接一次数据库，是一种不好的SQL编写习惯。

【建议】在应用程序完成作业任务之后，应当及时断开和DWS的连接，释放资源。建议在任务中设置session超时时间参数。

【建议】使用JDBC连接池，在将连接释放给连接池前，需要执行以下操作重置会话环境。否则，可能会因为历史会话信息导致的对象冲突。

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须执行“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

CopyManager

【建议】在不使用ETL工具、数据入库实时性要求又比较高的情况下，建议在开发应用程序时，使用DWS JDBC驱动的CopyManager接口进行微批量导入。

CopyManager的使用方法请参见**CopyManager**。

2.7.4 DWS SQL 编写规则

DDL

- 【建议】在DWS中，建议DDL（建表、comments等）操作统一执行，在批处理作业中尽量避免DDL操作。避免大量并发事务对性能的影响。
- 【建议】在非日志表（unlogged table）使用完后，立即执行数据清理（TRUNCATE）操作。因为在异常场景下，DWS不保证非日志表（unlogged table）数据的安全性。
- 【建议】临时表和非日志表的存储方式建议和基表相同。当基表为行存（列存）表时，临时表和非日志表也推荐创建为行存（列存）表，可以避免行列混合关联带来的高计算代价。
- 【建议】索引字段的总长度不超过50字节。否则，索引大小会膨胀比较严重，带来较大的存储开销，同时索引性能也会下降。
- 【建议】不要使用DROP…CASCADE方式删除对象，除非已经明确对象间的依赖关系，以免误删。

版权所有 © 华为云计算技术有限公司

数据加载和卸载

- 【建议】在insert语句中显式给出插入的字段列表。例如：
```
INSERT INTO task(name,id,comment) VALUES ('task1','100','第100个任务');
```
- 【建议】在批量数据入库之后，或者数据增量达到一定阈值后，建议对表进行analyze操作，防止统计信息不准确而导致的执行计划劣化。
- 【建议】如果要清理表中的所有数据，建议使用TRUNCATE TABLE方式，不要使用DELETE FROM TABLE方式。DELETE FROM TABLE方式删除性能较差，且不会释放已删除数据所占用的磁盘空间。

类型转换

- 【建议】在需要数据类型转换（不同数据类型进行比较或转换）时，使用强制类型转换，避免隐式类型转换结果与预期不符。
- 【建议】在查询中，对常量要显式指定数据类型，不要试图依赖任何隐式的数据类型转换。
- 【关注】在ORACLE兼容模式下，导入数据时空字符串会自动转化为NULL。如果要保留空字符串需新建兼容性为TD的数据库。

查询操作

- 【建议】除ETL程序外，应该尽量避免向客户端返回大量结果集的操作。如果结果集过大，应考虑业务设计是否合理。
- 【建议】使用事务方式执行DDL和DML操作。例如，truncate table、update table、delete table、drop table等操作，一旦执行提交就无法恢复。对于这类操作，建议使用事务进行封装，必要时可以进行回滚。
- 【建议】在查询编写时，建议明确列出查询涉及的所有字段，不建议使用“SELECT *”这种写法。一方面基于性能考虑，尽量减少查询输出列；另一方面避免增删字段对前端业务兼容性的影响。
- 【建议】在访问表对象时带上schema前缀，可以避免因schema切换导致访问到非预期的表。
- 【建议】超过8张表或视图进行关联（特别是full join）时，执行代价难以估算。建议使用WITH TABLE AS语句或者其他方式创建中间临时表的方式增加SQL语句的可读性。
- 【建议】尽量避免使用笛卡尔积和Full join。这些操作会造成结果集的急剧膨胀，同时其执行性能也很低。
- 【关注】NULL值的比较只能使用IS NULL或者IS NOT NULL的方式判断，其他任何形式的逻辑判断都返回NULL。例如：NULL<>NULL、NULL=NULL和NULL<>1返回结果都是NULL，而不是期望的布尔值。
- 【关注】需要统计表中所有记录数时，不要使用count(col)来替代count(*)。count(*)会统计NULL值（真实行数），而count(col)不会统计。
- 【关注】在执行count(col)时，将“值为NULL”的记录行计数为0。在执行sum(col)时，当所有记录都为NULL时，最终将返回NULL；当不全为NULL时，“值为NULL”的记录行将被计数为0。
- 【关注】count(多个字段)时，多个字段名必须用圆括号括起来。例如，count((col1,col2,col3))。注意：通过多字段统计行数时，即使所选字段都为NULL，该行也被计数，效果与count(*)一致。
- 【关注】count(distinct col)用来计算该列不重复的非NULL的数量，NULL将不被计数。

版权所有 © 华为云计算技术有限公司

- 【关注】count(distinct (col1,col2,...))用来统计多列的唯一值数量，当所有统计字段都为NULL时，也会被计数，同时这些记录被认为是相同的。
- 【关注】通过常量来过滤数据时，会根据常量的数据类型和匹配列的数据类型来查找用于这两种数据类型计算的函数，如果找不到对应的函数，则会相应的进行隐式数据类型转化，然后再根据转化后的数据类型查找用于转化后的数据类型计算的函数。

  ```
  SELECT * FROM test WHERE timestamp_col = 20000101;
  ```

  上述例子中，假设timestamp_col是timestamp类型，则会先查找支持timestamp类型和int类型（常量数字认为是int类型）“等于”运算的函数，如果找不到，则把timestamp_col和常量数字隐式类型转化成text类型来计算。
- 【建议】尽量避免标量子查询语句的出现。标量子查询是出现在select语句输出列表中的子查询，在下面例子中，括号内部分即为一个标量子查询语句：

  ```
  SELECT id, (SELECT COUNT(*) FROM films f WHERE f.did = s.id) FROM staffs_p1 s;
  ```

  标量子查询时常会导致查询性能急剧劣化，在应用开发过程中，应当根据业务逻辑，对标量子查询进行等价转换，将其写为表关联。
- 【建议】在WHERE子句中，应当对过滤条件进行排序，把选择度较小（筛选出的记录数较少）的条件排在前面。
- 【建议】WHERE子句中的过滤条件，尽量符合单边规则（即把字段名放在比较条件的一边），优化器在某些场景下会自动进行剪枝优化。如col op expression，其中col为表的一个列，op为‘=’、‘>’等比较操作符，expression为不含列名的表达式。例如：

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data WHERE
  current_timestamp(6) - time < '1 days'::interval;
  ```

  改写为：

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data WHERE time >
  current_timestamp(6) - '1 days'::interval;
  ```
- 【建议】尽量避免不必要的排序操作。排序需要耗费大量的内存及CPU，如果业务逻辑许可，可以组合使用ORDER BY和LIMIT，减小资源开销。DWS默认按照ASC & NULL LAST进行排序。
- 【建议】使用ORDER BY子句进行排序时，显式指定排序方式（ASC/DESC），NULL的排序方式（NULL FIRST/NULL LAST）。
- 【建议】不要单独依赖LIMIT子句返回特定顺序的结果集。如果只需要获取部分特定结果集，应组合使用ORDER BY子句与LIMIT子句，必要时也可以使用OFFSET跳过特定结果。
- 【建议】在保障业务逻辑准确的情况下，建议尽量使用UNION ALL来代替UNION。
- 【建议】如果过滤条件只有OR表达式，可以将OR表达式转化为UNION ALL以提升性能。使用OR的SQL语句经常无法优化，导致执行速度慢。

  例如：

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301) OR (cdp= 301 AND inline=302) OR (cdp= 302 AND inline=301);
  ```

  转换为：

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301)
  union all
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 301 AND inline=302)
  union all
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 302 AND inline=301);
  ```


版权所有 © 华为云计算技术有限公司

- 【建议】当in(val1, val2, val3...)表达式中字段较多时，建议使用in (values (val1), (val2), (val3)…)语句进行替换。优化器会自动把IN约束转换为非关联子查询，从而提升查询性能。
- 【建议】在关联字段不存在NULL值的情况下，使用(not) exists代替(not) in。例如，在下面查询语句中，当T1.C1列不存在NULL值时，可以先为T1.C1字段添加NOT NULL约束，再进行如下改写。

  ```
  SELECT * FROM T1 WHERE T1.C1 NOT IN (SELECT T2.C2 FROM T2);
  ```

  改写为：

  ```
  SELECT * FROM T1 WHERE NOT EXISTS (SELECT  * FROM T1,T2 WHERE T1.C1=T2.C2);
  ```

  说明

  - 如果不能保证T1.C1列的值为NOT NULL的情况下，就不能进行上述改写。
  - 如果T1.C1为子查询的输出，要根据业务逻辑确认其输出是否为NOT NULL。
- 【建议】通过游标进行翻页查询，而不是使用LIMIT OFFSET语法，避免多次执行带来的资源开销。游标必须在事务中使用，执行完后务必关闭游标并提交事务。

2.7.5 自定义 DWS 外部函数(pgSQL/Java)使用规则

- 【关注】Java UDF可以实现一些Java逻辑计算，禁止在Java UDF中封装业务。
- 【关注】禁止在Java函数中使用任何方式连接数据库，包括但不限于JDBC。
- 【关注】只能选择下表中的数据类型，不支持自定义类型、复杂数据类型（Java Array类及派生类）等：
- 【关注】不支持UDAF（用户定义聚合函数）、UDTF（用户自定义表生成函数）。

表 2-8 PL/Java 默认数据类型映射关系

DWS	Java
BOOLEAN	boolean
"char"	byte
bytea	byte[]
SMALLINT	short
INTEGER	int
BIGINT	long
FLOAT4	float
FLOAT8	double
CHAR	java.lang.String
VARCHAR	java.lang.String
TEXT	java.lang.String
name	java.lang.String
DATE	java.sql.Timestamp

版权所有 © 华为云计算技术有限公司

DWS	Java
TIME	java.sql.Time (stored value treated as local time)
TIMETZ	java.sql.Time
TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	java.sql.Timestamp

2.7.6 DWS PL/pgSQL 使用规则

总体开发原则

1. 应完全按照设计文档进行开发。
2. 程序模块应做到高内聚低耦合。
3. 应有正确、全面的故障对策。
4. 程序编写应做到结构合理，条理清晰。
5. 程序名称命名应按照统一的命名规则进行命名。
6. 应充分考虑程序的运行效率，包括程序的执行效率和数据库的查询、存储效率，在保证应用的同时应使用效率高的处理方法。
7. 程序注释应详细、正确、规范。
8. 除非应用特别需要控制commit和rollback的提交时机，否则应在存储过程结束时执行显式的commit或者rollback操作。
9. 程序处理应支持7*24小时；对于中断，应用程序应提供安全、简单的断点再续处理。
10. 应提供标准、简单的应用输出，为应用维护人员提供明确的进度显示、错误描述和运行结果；为业务人员提供明确、直观的报表、凭证输出。

程序编写原则

1. 在PL/pgSQL中的SQL语句宜使用绑定变量。
2. 在PL/pgSQL中的SQL语句宜使用RETURNING子句。
3. 存储过程使用原则：
   a. 对于单个存储过程中Varchar或者Varchar2类型输出参数个数不应超过50个。
   b. 不应使用long类型作为输入或输出参数。
   c. 对于大小超过10MB的字符串类型输出，应使用CLOB类型。
4. 变量声明原则：
   a. 变量声明时，如果含义和应用表某字段含义或某变量相同时，应使用%TYPE声明。
   b. 记录声明时，如果含义和某应用表行数据相同时，应使用%ROWTYPE声明。
   c. 变量声明每行应只包含一条语句。
   d. 不应声明LONG类型的变量。

版权所有 © 华为云计算技术有限公司

5. 游标使用类型：
   a. 显式游标使用后应关闭。
   b. 游标变量使用后应关闭，若游标变量需要传递数据给调用的应用程序，应在应用程序中进行游标关闭处理；若游标变量仅在存储过程中使用，应显式关闭游标。
   c. 在使用DBMS_SQL.CLOSE_CURSOR关闭游标前，应使用DBMS_SQL.IS_OPEN判断游标是否已打开。
6. 集合使用原则：
   a. 引用集合中的元素时宜使用FORALL语句，不宜使用FOR循环语句。
7. 动态语句使用原则：
   a. 联机系统的交易程序不宜使用动态SQL。
   b. PL/pgSQL中要实现DDL语句和系统控制命令，可使用动态SQL。
   c. 宜尽量使用变量绑定。
8. 拼装SQL的使用原则：
   a. 拼装SQL宜使用绑定变量。
   b. 拼装SQL语句的条件如果有外部输入源，应对输入条件进行字符检查，防止攻击。
   c. 在PL/pgSQL脚本中，单行代码的长度，不宜超过2499字符。
9. Trigger使用原则：
   a. Trigger可用于实现差量数据日志等与业务处理无关的可用性设计场景。
   b. 不应使用Trigger实现业务处理相关功能。

异常处理原则

任何在PL/pgSQL函数中发生的错误会中止该函数的执行，而且实际上会中止其周围的事务。此时可以使用一个带有EXCEPTION子句的BEGIN块捕获错误并且从中恢复。

1. 在使用PL/pgSQL块中，如果使用了不能返回确定结果的SQL语句，宜在EXCEPTION中对程序可能出现的异常进行处理，避免出现未处理的出错被传递到外层块，导致程序逻辑错误。
2. 对于系统已经定义了的异常，可以直接使用。DWS暂不支持自定义异常。
3. 进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大得多。因此，非必要场景不应使用EXCEPTION。

书写规范

1. 变量命名规则：
   a. 过程、函数的输入参数格式宜为：IN_参数名，参数名宜使用大写。
   b. 过程、函数的输出参数格式宜为：OUT_参数名，参数名宜使用大写。
   c. 过程、函数的输入输出参数格式宜为：IO_参数名，参数名宜使用大写。
   d. 过程、函数的程序中用到的变量宜由v_变量名组成，变量名宜使用小写。
   e. 将查询语句做成字符串拼接时，where语句的拼接变量名宜统一为v_where，select语句的拼接变量名宜为v_select。
   f. 记录（RECORD）的类型（TYPE）命名宜由T+变量名组成，名称宜使用大写。

版权所有 © 华为云计算技术有限公司

g. 游标命名宜由CUR+变量名组成，名称宜使用大写。
h. 引用游标（ REF CURSOR ）的命名宜由REF+变量名组成，名称宜使用大写。

2. 变量类型定义：
   a. 变量类型声明时，如果其含义和应用表某字段含义相同时，应使用%TYPE声明。
   b. 记录类型声明时，如果其含义和某应用表行数据相同时，应使用%ROWTYPE声明。
3. 注释规范：
   a. 注释应该是有意义的，而不应是重述代码。
   b. 注释应简洁、易懂，以中文为主。为了表达准确，名词或操作上也可以使用英文。
   c. 应在每个存储过程、函数的开始加入注释，内容应包括：本程序的简要功能描述、编写者、编写日期、程序版本号信息和程序变更信息，而且各存储过程开头注释应保持统一格式。
   d. 应在输入输出参数的旁边添加注释，注明变量的意义。
   e. 每个块或大分支的开始宜添加注释，描述块的简要功能，若使用算法，宜添加注释简单描述算法的目的和结果。
4. 变量声明格式：

   每行应只包含一条语句，如同时需要赋初始值，应在同一行书写。
5. 大小写规范：

   未加双引号的标识符（如列名等）对大小写不敏感，系统会自动将它们转换为小写。带双引号时大小写敏感。
6. 缩进规范：

   创建存储过程语句中，同一层的CREATE、AS/IS、BEGIN、END这几个关键字应位于同一列，其他部分依次缩进。
7. 语句详述：
   a. 变量定义语句。每行应只包含一条语句。
   b. 同一层的IF、ELSEIF、ELSE和END关键字应开始于同一列，执行语句缩进。
   c. CASE和END关键字应位于同一列，WHEN和ELSE关键字应缩进。
   d. 同一层的LOOP和END LOOP关键字应位于同一列，层内语句或嵌套应依次缩进。

版权所有 © 华为云计算技术有限公司

3 创建和管理 DWS 数据库对象

3.1 创建和管理 DWS 数据库

数据库（Database）是表、索引、视图、存储过程、操作符等对象的集合。DWS支持创建多个数据库，但是客户端程序一次只能连接并访问一个数据库，无法跨数据库进行查询。

模板和默认数据

- DWS提供了两个模板数据库template0、template1，以及一个默认的数据库gaussdb。
- 默认情况下，每个新创建的数据库都是基于一个模板数据库。DWS数据库默认使用template1作为模板，编码格式为SQL_ASCII，且不允许自定义字符编码。若创建数据库时需指定字符编码，请使用template0创建数据库。
- 请避免使用客户端或其他手段连接及操作两个模板数据库。

  说明

  通过“show server_encoding”命令可以查看当前数据库存储编码。

创建数据库

使用CREATE DATABASE语句创建一个新的数据库。

```
CREATE DATABASE mydatabase;
```

 版权所有 © 华为云计算技术有限公司

说明

- 创建数据库时，若数据库名称长度超过63字节，server端会对数据库名称进行截断，保留前63个字节，因此建议数据库名称长度不要超过63个字节，不要使用多字节字符作为对象名。（如果出现因为误操作导致在多字节字符的中间截断进而无法删除数据库对象的现象，请使用截断前的数据库对象名进行删除操作，或将该对象从各个数据库节点的相应系统表中依次删掉。）
- 数据库名称遵循SQL标识符的一般规则。当前用户自动成为此新数据库的所有者。
- 如果一个数据库系统用于承载相互独立的用户和项目，建议把它们放在不同的数据库里。
- 如果项目或者用户是相互关联的，并且可以相互使用对方的资源，则应该把它们放在同一个数据库里，但可以规划在不同的Schema中。
- 用户必须拥有数据库创建的权限或者是数据库的系统管理员权限才能创建数据库。

查看数据库

查看数据库的方式：

- 使用\l元命令查看数据库系统的数据库列表。
  ```
  \l
  ```
- 通过系统表pg_database查询数据库列表。
  ```
  SELECT datname FROM pg_database;
  ```

修改数据库

可以使用ALTER DATABASE语句修改数据库属性（比如：owner、名称和默认的配置属性）。

- 为数据库设置默认的模式搜索路径。
  ```
  ALTER DATABASE mydatabase SET search_path TO pa_catalog,public;
  ```
- 重新命名数据库。
  ```
  ALTER DATABASE mydatabase RENAME TO newdatabase;
  ```

删除数据库

可以使用DROP DATABASE语句删除数据库。该语句删除了数据库中的系统目录，并且删除了带有数据的磁盘上的数据库目录。用户必须是数据库的owner或者系统管理员才能删除数据库。当有用户连接数据库时，删除操作会失败。删除数据库时请先连接到其他的数据库。

使用DROP DATABASE语句删除数据库：

```
DROP DATABASE newdatabase;
```

3.2 创建和管理 DWS Schema

Schema又称作模式，从逻辑上组织一个数据库中的对象和数据。通过管理Schema，允许多个用户使用同一数据库而不相互干扰，同时便于将第三方应用添加到相应的Schema下而不引起冲突。

相同的数据库对象名称可以应用在同一数据库的不同Schema中，而没有冲突。例如，a_schema和b_schema都可以包含名为mytable的表。具有所需权限的用户可以访问数据库的多个Schema中的对象。

在当前数据库中创建用户时，系统会在当前数据库中为新用户创建一个同名Schema。

 版权所有 © 华为云计算技术有限公司

public 模式

每个数据库都有一个名为public的模式。 所有的数据库角色（用户）都在public模式上拥有USAGE特权，但是普通角色（用户）没有在public模式上的CREATE权限。

创建 Schema

- 使用CREATE SCHEMA命令来创建一个新的Schema。
  ```
  CREATE SCHEMA myschema;
  ```
  如果需要在模式中创建或者访问对象，其完整的对象名称由模式名称和具体的对象名称组成。中间由符号“.”隔开。例如：myschema.table。
- 用户可以创建一个由他人拥有的schema。例如，创建名为myschema的Schema，并指定Schema的所有者为用户jack。
  ```
  CREATE SCHEMA myschema AUTHORIZATION jack;
  ```
  若不指定authorization username，则其所有者为执行该命令的用户。

修改 Schema

- 使用ALTER SCHEMA修改Schema名称，只有Schema所有者可以更改Schema。
  ```
  ALTER SCHEMA schema_name RENAME TO new_name;
  ```
- 使用ALTER SCHEMA修改Schema所有者：
  ```
  ALTER SCHEMA schema_name OWNER TO new_owner;
  ```

设置 Schema 搜索路径

GUC参数search_path设置Schema的搜索顺序，参数取值形式为采用逗号分隔的Schema名称列表。如果创建对象时未指定目标Schema，则该对象会被添加到搜索路径中列出的第一个Schema中。当不同Schema中存在同名的对象时，查询对象未指定Schema的情况下，将从搜索路径中包含该对象的第一个Schema中返回对象。

- 使用SHOW命令查看当前搜索路径。
  ```
  SHOW SEARCH_PATH;
   search_path
  ----------------
   "$user",public
  (1 row)
  ```
  search_path参数的默认值为："$user"，public。$user表示与当前会话用户名同名的Schema名，如果这样的模式不存在，$user将被忽略。所以默认情况下，用户连接数据库后，如果数据库下存在同名Schema，则对象会添加到同名Schema下，否则对象被添加到Public Schema下。
- 使用SET命令修改当前会话的默认Schema。例如，将搜索路径设置为myschema、public，首先搜索myschema。
  ```
  SET SEARCH_PATH TO myschema, public;
  ```
  也可以使用ALTER ROLE命令为特定的角色（用户）设置search_path。例如：
  ```
  ALTER ROLE jack SET search_path TO myschema, public;
  ```

使用 Schema

在特定Schema下创建对象或者访问特定Schema下的对象，需要使用有Schema修饰的对象名。名称包含Schema名以及对象名，之间用“.”号分开。

- 在myschema下创建mytable表。以schema_name.table_name格式创建表。
  ```
  CREATE TABLE myschema.mytable(id int, name varchar(20));
  ```
- 查询myschema下mytable表的所有数据。

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM myschema.mytable;
 id | name
----+------
(0 rows)
```

查看 Schema

- 使用current_schema()函数查看当前Schema：

```
SELECT current_schema();
 current_schema
----------------
 myschema
(1 row)
```

- 要查看Schema所有者，请对系统表PG_NAMESPACE和PG_USER执行如下关联查询。语句中的schema_name请替换为实际要查找的Schema名称。

```
SELECT s.nspname,u.usename AS nspowner FROM PG_NAMESPACE s, PG_USER u WHERE nspname='schema_name' AND s.nspowner = u.usesysid;
```

- 要查看所有Schema的列表，请查询PG_NAMESPACE系统表。

```
SELECT * FROM PG_NAMESPACE;
```

- 使用PGXC_TOTAL_SCHEMA_INFO视图查询整个集群的Schema空间使用情况。

```
SELECT * FROM PGXC_TOTAL_SCHEMA_INFO;
```

- 要查看属于某Schema下表的列表，请查询系统视图PG_TABLES。例如，以下查询会返回Schema PG_CATALOG中的表列表。

```
SELECT distinct(tablename),schemaname FROM PG_TABLES where schemaname = 'pg_catalog';
```

Schema 的权限控制

默认情况下，用户只能访问属于自己的Schema中的数据库对象。如需要访问其他Schema的对象，则需赋予对应Schema的usage权限。

通过将模式的CREATE权限授予某用户，被授权用户就可以在此模式中创建对象。

- 将myschema的usage权限赋予用户jack。

```
GRANT USAGE ON schema myschema TO jack;
```

- 将用户jack对于myschema的usage权限收回。

```
REVOKE USAGE ON schema myschema FROM jack;
```

删除 Schema

- 使用DROP SCHEMA命令删除一个空的Schema（即该Schema下没有数据库对象）。

```
DROP SCHEMA IF EXISTS myschema;
```

- 默认情况下，删除一个Schema前，它必须为空。 要删除一个Schema及其包含的所有对象（表、数据、函数等），需要使用CASCADE关键字。

```
DROP SCHEMA myschema CASCADE;
```

系统 Schema

- 每个数据库都包含一个pg_catalog schema，它包含系统表和所有内置数据类型、函数、操作符。pg_catalog是搜索路径中的一部分，始终在临时表所属的模式后面，并在search_path中所有模式的前面，即具有第二搜索优先级。这样确保可以搜索到数据库内置对象。如果用户需要使用和系统内置对象重名的自定义对象时，可以在操作自定义对象时带上自己的模式。
- information_schema由一个包含数据库中对象信息的视图集合组成。 这些视图以一种标准化的方式从系统目录表中得到系统信息。

 版权所有 © 华为云计算技术有限公司

3.3 创建和管理 DWS 表

创建表

CREATE TABLE命令创建一个表，创建表时可以定义以下内容：

- 表的列及**数据类型**。
- 表约束的定义，即任何用于限制列或者表中数据的表约束或者列约束，参见**表约束的定义**。
- 表分布的定义，即表的分布策略，它决定DWS数据库如何在片（Segment）之间划分数据，参见**表分布的定义**。
- 表存储格式，参见**选择DWS表存储模型**。
- 分区表定义，参见**创建和管理DWS分区表**。

示例：CREATE TABLE创建了一个表web_returns_p1，使用wr_item_sk作为分布键，并基于wr_returned_date_sk设置了range分布功能。

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2019 START(20191231) END(20221231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
```

表约束的定义

可以在列和表上定义约束来限制表中的数据，但是有以下一些限制：

- 表中的主键约束和唯一约束必须包含分布列。
- 列存表支持PARTIAL CLUSTER KEY、主键和唯一表级约束，不支持外键表级约束。
- 列存表的字段约束只支持NULL、NOT NULL和DEFAULT常量值。

表 3-1 表约束

约束项	说明	示例

检查约束	检查约束允许用户指定一个特定列中的值必须满足一个布尔（真值）表达式。	创建表products，要求产品价格为正：
		`CREATE TABLE products`
		`(`
		`product_no integer,`
		`name text,`
		`price numeric CHECK (price > 0)`
		`);`

版权所有 © 华为云计算技术有限公司

约束项	说明	示例
非空约束	非空约束指定一个列不能有空值。非空约束总是被写作为列约束。	创建表products，要求product_no和name不能为空： CREATE TABLE products (product_no integer NOT NULL, name text NOT NULL, price numeric);
唯一约束	唯一约束确保一列或者一组列中包含的数据对于表中所有的行都是唯一的。 如果没有声明DISTRIBUTE BY REPLICATION，则唯一约束的列集合中必须包含分布列。	创建表products，product_no值不能重复： CREATE TABLE products (product_no integer UNIQUE, name text, price numeric)DISTRIBUTE BY HASH(product_no);
主键约束	主键约束是一个UNIQUE约束和一个NOT NULL约束的组合。 如果没有声明DISTRIBUTE BY REPLICATION，则主键约束的列集合中必须包含分布列。如果一个表具有主键，这个列（或者这一组列）会被默认选中为该表的分布键。	创建表products，主键约束为product_no： CREATE TABLE products (product_no integer PRIMARY KEY, name text, price numeric)DISTRIBUTE BY HASH(product_no);
局部聚簇	局部聚簇通过min/max稀疏索引较快地实现基表扫描的filter过滤。Partial Cluster Key可以指定多列，但是一般不建议超过两列。	创建表products，PCK为product_no： CREATE TABLE products (product_no integer, name text, price numeric, PARTIAL CLUSTER KEY(product_no)) WITH (ORIENTATION = COLUMN);

表分布的定义

DWS支持的分布方式：复制表（Replication）、哈希表（Hash）和轮询表（Roundrobin）。

 版权所有 © 华为云计算技术有限公司

策略	描述	适用场景	优势与劣势
复制表（Replication）	集群中每一个DN实例上都有一份全量表数据。	小表、维度表。	• Replication优点是每个DN上都有此表的全量数据，在join操作中可以避免数据重分布操作，从而减少网络开销，同时减少了plan segment（每个plan segment都会起对应的线程）。 • Replication缺点是每个DN都保留了表的完整数据，造成数据的冗余。一般情况下只有较小的维度表才会定义为Replication表。
哈希表（Hash）	表数据通过hash方式散列到集群中的所有DN实例上。	数据量较大的事实表。	• 在读/写数据时可以利用各个节点的IO资源，大幅度提升表的读/写速度。 • 一般情况下大表（1000000条记录以上）定义为Hash表。
轮询表（Roundrobin）	表的每一行被轮番地发送给各个DN，数据会被均匀地分布在各个DN中。	数据量较大的事实表，且使用Hash分布时找不到合适的分布列。	• Roundrobin优点是保证了数据不会发生倾斜，从而提高了集群的空间利用率。 • Roundrobin缺点是无法像Hash表一样进行DN本地化优化，查询性能通常不如Hash表。 • 一般在大表无法找到合适的分布列时，定义为Roundrobin表，若大表能够找到合适的分布列，优先选择性能更好的Hash分布。

选择分布列

采用Hash分布方式，需要为用户表指定一个分布列（distribute key）。当插入一条记录时，系统会根据分布列的值进行hash运算后，将数据存储在对应的DN中。

所以Hash分布列选取至关重要，需要满足以下原则：

1. **列值应比较离散，以便数据能够均匀分布到各个DN**。例如，考虑选择表的主键为分布列，如在人员信息表中选择身份证号码为分布列。
2. **在满足第一条原则的情况下尽量不要选取存在常量filter的列**。例如，表dwcjk相关的部分查询中出现dwcjk的列zqdh存在常量的约束(例如zqdh=’000001’)，那么就应当尽量不用zqdh做分布列。
3. **在满足前两条原则的情况下，考虑选择查询中的连接条件为分布列**，以便Join任务能够下推到DN中执行，且减少DN之间的通信数据量。

   对于Hash分表策略，如果分布列选择不当，可能导致数据倾斜，查询时出现部分DN的I/O短板，从而影响整体查询性能。因此在采用Hash分表策略之后需对表的数据进行数据倾斜性检查，以确保数据在各个DN上是均匀分布的。可以使用以下SQL检查数据倾斜性

   ```
   select
   xc_node_id, count(1)
   from tablename
   ```

 版权所有 © 华为云计算技术有限公司

```
group by xc_node_id
order by xc_node_id desc;
```

其中xc_node_id对应DN，一般来说，**不同DN的数据量相差5%以上即可视为倾斜，如果相差10%以上就必须要调整分布列**。

4. 一般不建议用户新增一列专门用作分布列，尤其不建议用户新增一列，然后用SEQUENCE的值来填充作为分布列。因为SEQUENCE可能会带来性能瓶颈和不必要的维护成本。

查看表数据

- 使用系统表pg_tables查询数据库所有表的信息。
  ```
  SELECT * FROM pg_tables;
  ```
- 使用gsql的\d+命令查询表的属性。
  ```
  \d+ customer_t1;
  ```
- 执行如下命令查询表customer_t1的数据量。
  ```
  SELECT count(*) FROM customer_t1;
  ```
- 执行如下命令查询表customer_t1的所有数据。
  ```
  SELECT * FROM customer_t1;
  ```
- 执行如下命令只查询字段c_customer_sk的数据。
  ```
  SELECT c_customer_sk FROM customer_t1;
  ```
- 执行如下命令过滤字段c_customer_sk的重复数据。
  ```
  SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
  ```
- 执行如下命令查询字段c_customer_sk为3869的所有数据。
  ```
  SELECT * FROM customer_t1 WHERE c_customer_sk = 3869;
  ```
- 执行如下命令按照字段c_customer_sk进行排序。
  ```
  SELECT * FROM customer_t1 ORDER BY c_customer_sk;
  ```

删除表数据

⚠ 注意

请谨慎执行DROP TABLE和TRUNCATE TABLE命令，删除表后，数据将无法恢复。

- 从数据库中删除表customer_t1。
  ```
  DROP TABLE customer_t1;
  ```
- 清空一个表的行但不移除该表的定义，可使用DELETE或者TRUNCATE。

  删除表customer_t1中所有的行。
  ```
  TRUNCATE TABLE customer_t1;
  ```

  删除表customer_t1中所有的行。
  ```
  DELETE FROM customer_t1;
  ```

  删除表customer_t1中所有c_customer_sk为3869的记录：
  ```
  DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
  ```

管理 UNLOGGED 表

UNLOGGED表即非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动清理，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。

版权所有 © 华为云计算技术有限公司

使用场景：非日志表不能保证数据的安全性，需要在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。在创建非日志表时应将cnretry关闭（即设置GUC参数max_query_retry_times=0）。

故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。

- 9.1.0版本后，UNLOGGED表默认存储在pg_unlogged表空间下，且不可迁移或指定到其他表空间。
- 从低版本升级到9.1.0版本后，旧版本创建的UNLOGGED表还存储在原表空间下。

9.1.0版本提供UNLOGGED表迁移脚本switch_unlogged_tablespace.py，配合GUC参数**enable_unlogged_tablespace_compat**，可以优化RTO。

1. 脚本位于 $GPHOME/script目录，可以通过-?获取帮助信息。

```
[perfadm@linux83108 script]$ python3 switch_unlogged_tablepace.py -?
Usage:
  python3 switch_unlogged_tablepace.py -? | --help
  python3 switch_unlogged_tablepace.py -t query [--dbname=DBNAME] [--verbose]
  python3 switch_unlogged_tablepace.py -t switch [--dbname=DBNAME] [--without-disable]

General options:
  -t                            Type of the command.
  -?, --help                    Show help information for this utility, and exit the command line mode.

Options for query:
  --dbname                      Database name.
  --verbose                     List all unlogged tables.

Options for switch:
  --dbname                      Database name.
  --without-disable             Do not disable unlogged tablespace compatibility after switch all unlogged tables.
```

2. 迁移所有UNLOGGED表（建议）

```
python3 switch_unlogged_tablespace.py -t switch
```

3. 迁移成功后，GUC参数**enable_unlogged_tablespace_compat**会自动设置为off。

须知

建议在升级到9.1.0版本后通过以下两步操作提升实例重启RTO：

1. 使用switch_unlogged_tablespace.py脚本将UNLOGGED表全部迁移到pg_unlogged表空间。
2. 如果旧版本没有使用UNLOGGED表，建议将GUC参数enable_unlogged_tablespace_compat设置为OFF。

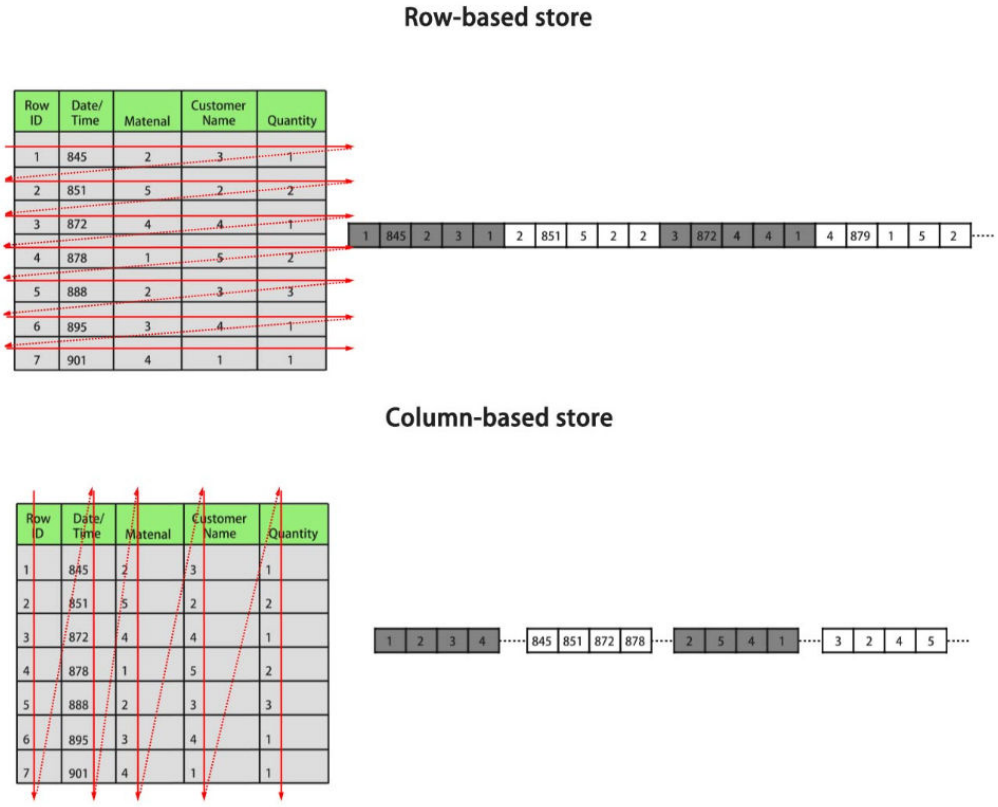
3.4 选择 DWS 表存储模型

DWS支持行列混合存储。当创建一个表时，可以选择表的存储格式为行存储或列存储。

行存储是指将表按行存储到硬盘分区上，列存储是指将表按列存储到硬盘分区上。默认情况下，创建的表为行存储。行存储和列存储的差异请参见图3-1。

版权所有 © 华为云计算技术有限公司

图 3-1 行存储和列存储的差异



上图中，左上为行存表，右上为行存表在硬盘上的存储方式。左下为列存表，右下为列存表在硬盘上的存储方式。

DWS表的行/列存储通过表定义的orientation属性定义。当指定orientation属性为row时，表为行存储；当指定orientation属性为column时，表为列存储；如果不指定，默认为行存储。行、列存储模型各有优劣，建议根据实际情况选择：

表 3-2 表的存储类型及场景

存储模型	优点	缺点	适用场景
行存	数据按照行进行存储，在查询某一行数据时，可以快速定位到目标位置。	查询时即使只涉及某几列，所有数据也都会被读取。	1. 表的字段个数比较少，查询表的大部分字段。 2. 点查询（返回记录少，基于索引的简单查询）。 3. 频繁进行增删改查操作且涉及整行数据。

版权所有 © 华为云计算技术有限公司

存储模型	优点	缺点	适用场景
列存	1. 查询时只有涉及列会被读取。 2. 列数据特征比较相似，能够更有效地进行数据压缩。	不适合少量数据INSERT或UPDATE操作。	1. 表的字段比较多（大宽表），查询中涉及列不多。 2. 统计分析类查询 (关联、分组操作较多的场景)。 3. 即席查询（查询条件不确定，行存表扫描难以使用索引）。

创建一个行存表

例如，创建一个名为customer_t1的行存表：

```
CREATE TABLE customer_t1
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
);
```

创建一个列存表

例如，创建一个名为customer_t2的列存表：

```
CREATE TABLE customer_t2
(
  state_ID   CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID    NUMBER
)
WITH (ORIENTATION = COLUMN);
```

使用压缩

表压缩可以在创建表时开启，压缩表能够使表中的数据以压缩格式存储，意味着占用相对少的内存。

对于I/O读写量大，CPU富足（计算相对小）的场景，选择高压缩比；反之选择低压缩比。建议依据此原则进行不同压缩下的测试和对比，以选择符合自身业务情况的最优压缩比。压缩比通过COMPRESSION参数指定，其支持的取值如下：

- 列存表为：YES/NO/LOW/MIDDLE/HIGH，默认值为LOW。
- 行存表为：YES/NO，默认值为NO。（行存表压缩功能暂未商用，如需使用请联系技术支持工程师）

各压缩级别所适用的业务场景说明如下：

压缩级别	所适用的业务场景
低级别压缩	系统CPU使用率高，存储磁盘空间充足。
中度压缩	系统CPU使用率适中，但存储磁盘空间不是特别充足。

 版权所有 © 华为云计算技术有限公司

压缩级别	所适用的业务场景
高级别压缩	系统CPU使用率低，磁盘空间不充裕。

例如，创建一个名为customer_t3的列存压缩表：

```
CREATE TABLE customer_t3
(
  state_ID    CHAR(2),
  state_NAME VARCHAR2(40),
  area_ID     NUMBER
)
WITH (ORIENTATION = COLUMN,COMPRESSION=middle);
```

3.5 创建和管理 DWS 分区表

分区表就是把逻辑上的一张表根据分区策略分成几张物理块库进行存储，这张逻辑上的表称之为分区表，物理块称之为分区。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。当进行条件查询时，系统只会扫描满足条件的分区，避免全表扫描，从而提升查询性能。

关于分区表维护的详细语法介绍请参见**ALTER TABLE PARTITION**。

分区表的优势：

- 改善查询性能。对分区对象的查询可以仅搜索自己关心的分区，提高检索效率。
- 增强可用性。如果分区表的某个分区出现故障，表在其他分区的数据仍然可用。
- 提升可维护性。对于需要周期性删除的过期历史数据，可以通过drop/truncate分区的方式快速高效处理。

支持的表分区类型

- 范围分区（Range Partitioning），基于一个数值型范围划分数据，例如按日期或价格区间定义。
- 列表分区（List Partitioning），基于一个值列表划分数据，例如按销售范围或产品属性定义。仅8.1.3及以上集群版本支持。

分区策略选择

当表有以下特征时，可以考虑使用表分区策略：

- 数据具有明显区间性的字段。

  分区表需要根据有明显区间性字段进行表分区。比如按照日期、区域、数值等字段进行分区，时间字段是最常见的分区字段。
- 业务查询有明显的区间范围特征。

  查询数据可落到区间范围指定的分区内，这样才能通过分区剪枝，只扫描查询需要的分区，从而提升数据扫描效率，降低数据扫描的IO开销。
- 表数据量比较大。

  小表扫描本身耗时不大，分区表的性能收益不明显，因此只建议对大表采取分区策略。列存储模式下因为每个列是单独的文件存储，且最小的存储单元CU可存储6w行数据，因此对于列存分区表，建议每个分区的数据不小于DN个数*6w。

版权所有 © 华为云计算技术有限公司

创建范围(range)分区表

示例：创建一个按wr_returned_date_sk范围分区的表web_returns_p1。

```
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE (wr_returned_date_sk)
(
    PARTITION p2016 VALUES LESS THAN(20161231),
    PARTITION p2017 VALUES LESS THAN(20171231),
    PARTITION p2018 VALUES LESS THAN(20181231),
    PARTITION p2019 VALUES LESS THAN(20191231),
    PARTITION pxxxx VALUES LESS THAN(maxvalue)
);
```

对于分区间隔固定、批量创建分区的场景。可使用如下示例：

```
CREATE TABLE web_returns_p2
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
```

示例：创建一个按时间日期作为分区的表web_returns_p2，其中time作为分区键。

```
CREATE TABLE web_returns_p2
(
   id integer,
   idle numeric,
   IO numeric,
   scope text,
   IP text,
   time timestamp
)
 WITH (TTL='7 days',PERIOD='1 day')
PARTITION BY RANGE(time)
 (
   PARTITION P1 VALUES LESS THAN('2022-01-05 16:32:45'),
   PARTITION P2 VALUES LESS THAN('2022-01-06 16:56:12')
 );
```

创建列表(list)分区表

LIST分区表可以使用任意允许值比较的列作为分区键列。创建LIST分区表时，必须要为每一个分区声明每一个值分区。

示例：创建LIST分区表sales_info。

```
CREATE TABLE sales_info
(
sale_time  timestamptz,
period     int,
city       text,
price      numeric(10,2),
```

 版权所有 © 华为云计算技术有限公司

```
remark     varchar2(100)
)
DISTRIBUTE BY HASH(sale_time)
PARTITION BY LIST (period, city)
(
PARTITION province1_202201 VALUES (('202201', 'city1'), ('202201', 'city2')),
PARTITION province2_202201 VALUES (('202201', 'city3'), ('202201', 'city4'), ('202201', 'city5'))
);
```

对已有的表进行分区

表只能在创建时被分区。 如果用户有一个表想要分区，用户必须创建一个分过区的表，把原始表的数据载入到新表，再删除原始表并且把分过区的表重命名为原始表的名称。 用户还必须重新授权表上的权限。例如：

```
CREATE TABLE web_returns_p2
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);
INSERT INTO web_returns_p2 SELECT * FROM web_returns_p1;
DROP TABLE web_returns_p1;
ALTER TABLE web_returns_p2 RENAME TO web_returns_p1;
GRANT ALL PRIVILEGES ON web_returns_p1 TO dbadmin;
GRANT SELECT ON web_returns_p1 TO jack;
```

增加一个分区

使用ALTER TABLE语句为范围分区表增加一个分区。例如，为表web_returns_p1增加分区P2020。

```
ALTER TABLE web_returns_p1 ADD PARTITION P2020 VALUES LESS THAN (20201231);
```

为列表分区增加一个分区。例如，为表sales_info增加分区province3_202201。

```
ALTER TABLE sales_info ADD PARTITION province3_202201 VALUES (('202201', 'city6'), ('202201', 'city7'));
```

分割一个分区

范围分区表和列表分区表分割分语法有所区别：

- 使用ALTER TABLE语句为范围分区表分割一个分区。例如，将表*web_returns_p1*分区*pxxxx*以20201231为分割点分割为p2020和p20xx两个分区。
  ```
  ALTER TABLE web_returns_p1 SPLIT PARTITION pxxxx AT(20201231) INTO (PARTITION p2020,PARTITION p20xx);
  ```
- 使用ALTER TABLE语句为列表分区表分割一个分区。例如，将表*sales_info*分区*province2_202201*分割为*province3_202201*和*province4_202201*两个分区。
  ```
  ALTER TABLE sales_info SPLIT PARTITION province2_202201 VALUES(('202201', 'city5')) INTO (PARTITION province3_202201,PARTITION province4_202201);
  ```

合并一个分区

使用ALTER TABLE语句为分区表合并一个分区。例如，将表*web_returns_p1* 分区p2016和p2017合并为一个分区p20162017。

 版权所有 © 华为云计算技术有限公司

```
ALTER TABLE web_returns_p1 MERGE PARTITIONS p2016,p2017 INTO PARTITION p20162017;
```

删除一个分区

使用ALTER TABLE语句从分区表中删除一个分区。例如，删除表web_returns_p1的分区P2020。

```
ALTER TABLE web_returns_p1 DROP PARTITION P2020;
```

查询分区

- 查询分区p2019。

  ```
  SELECT * FROM web_returns_p1 PARTITION (p2019);
  SELECT * FROM web_returns_p1 PARTITION FOR (20201231);
  ```

- 查看分区表信息，可使用系统表dba_tab_partitions。

  ```
  SELECT * FROM dba_tab_partitions where table_name='web_returns_p1';
  ```

删除分区表

使用DROP TABLE语句删除一个分区表。

```
DROP TABLE web_returns_p1;
```

设置分区索引是否可用

给分区表customer_address创建LOCAL索引student_grade_index，并指定分区的索引名称。

```
CREATE INDEX customer_address_index ON customer_address(ca_address_id) LOCAL
(
    PARTITION P1_index,
    PARTITION P2_index,
    PARTITION P3_index
);
```

重建分区表customer_address中分区P1上的所有索引。

```
ALTER TABLE customer_address MODIFY PARTITION P1 REBUILD UNUSABLE LOCAL INDEXES;
```

设置分区表customer_address的分区P3上的所有索引不可用。

```
ALTER TABLE customer_address MODIFY PARTITION P3 UNUSABLE LOCAL INDEXES;
```

重建分区索引

长期运行的分区表，索引可能产生碎片，重建索引，可提升查询效率，以下演示分区插入数据后重建分区索引。

步骤1 创建分区表。

```
DROP TABLE IF EXISTS sales;
CREATE TABLE sales (
   id          INT,
   sale_date   DATE,
   amount      DECIMAL(10,2)
)
PARTITION BY RANGE (sale_date)
(
   PARTITION p202310 VALUES LESS THAN ('2023-11-01'),
   PARTITION p202311 VALUES LESS THAN ('2023-12-01')
);
```

步骤2 创建索引。

 版权所有 © 华为云计算技术有限公司

```
CREATE INDEX idx_sale_date ON sales (sale_date) LOCAL;
```

步骤3 插入10月、11月分区数据。

```
INSERT INTO sales PARTITION (p202310)
VALUES
(1, '2023-10-05', 100.50),
(2, '2023-10-10', 200.75);
INSERT INTO sales PARTITION (p202311)
VALUES
(3, '2023-11-15', 300.00);
```

步骤4 重建特定分区p202310的索引。

```
ALTER TABLE sales  REBUILD PARTITION p202310  WITHOUT UNUSABLE;
```

----结束

交换分区（EXCHANGE PARTITION）

如下示例将演示一个普通表math_grade数据迁移到分区表student_grade中分区（math）的操作。

1. 创建分区表student_grade。

```
DROP TABLE IF EXISTS student_grade;
CREATE TABLE student_grade (
      stu_name     char(5),
      stu_no       integer,
      grade        integer,
      subject      varchar(30)
)
PARTITION BY LIST(subject)
(
      PARTITION gym VALUES('gymnastics'),
      PARTITION phys VALUES('physics'),
      PARTITION history VALUES('history'),
      PARTITION math VALUES('math')
);
```

2. 插入数据到分区表student_grade中。

```
INSERT INTO student_grade VALUES
      ('Ann', 20220101, 75, 'gymnastics'),
      ('Jack', 20220103, 60, 'math'),
      ('Anna', 20220108, 56, 'history'),
      ('John', 20220107, 82, 'physics'),
      ('Molly', 20220104, 91, 'physics'),
      ('Sam', 20220105, 72, 'math');
```

3. 查询分区表student_grade的math分区记录。

```
SELECT * FROM student_grade PARTITION (math);
```

查询结果如下：

```
gaussdb=> SELECT * FROM student_grade PARTITION (math);
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Sam      | 20220105 |    72 | math
 Jack     | 20220103 |    60 | math
(2 rows)
```

4. 创建一个与分区表student_grade定义匹配的普通表math_grade。

```
DROP TABLE IF EXISTS math_grade;
CREATE TABLE math_grade
(
      stu_name     char(5),
      stu_no       integer,
      grade        integer,
      subject      varchar(30)
);
```

 版权所有 © 华为云计算技术有限公司

5. 插入数据到表math_grade中。数据与分区表student_grade的math分区的分区规则一致。

```
INSERT INTO math_grade VALUES
    ('Ann', 20220101, 75, 'math'),
    ('Jack', 20220103, 60, 'math'),
    ('Anna', 20220108, 56, 'math'),
    ('John', 20220107, 82, 'math');
```

6. 将普通表math_grade数据迁移到分区表student_grade的分区（math）。

```
ALTER TABLE student_grade EXCHANGE PARTITION (math) WITH TABLE math_grade;
```

7. 查询分区表student_grade，结果显示表math_grade中的数据已和分区表student_grade的分区math中的数据交换。

```
SELECT * FROM student_grade PARTITION (math);
```

```
gaussdb=> SELECT * FROM student_grade PARTITION (math);
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jack     | 20220103 |    60 | math
 Anna     | 20220108 |    56 | math
 Ann      | 20220101 |    75 | math
 John     | 20220107 |    82 | math
(4 rows)
```

8. 查询表math_grade，显示之前存储在分区表student_grade的分区math中的数据已交换到表math_grade中。

```
SELECT * FROM math_grade;
```

```
gaussdb=> SELECT * FROM math_grade;
 stu_name |  stu_no  | grade | subject
----------+----------+-------+---------
 Jack     | 20220103 |    60 | math
 Sam      | 20220105 |    72 | math
(2 rows)
```

分区表业务锁

分区表的不同业务最终都是作用于目标分区上，数据库会给分区表和目标分区施加不同级别的表锁+分区锁，来控制并发行为。

表 3-3 锁模式间的冲突关系

锁模式名称	锁等级	锁用途	冲突关系
访问共享锁（AccessShareLock）	1	SELECT语句，允许其他事务读取数据。	8
行共享锁（RowShareLock）	2	SELECT FOR UPDATE或FOR SHARE，允许其他事务读取但阻止写入。	7|8
行排他锁（RowExclusiveLock）	3	INSERT、UPDATE、DELETE语句。	5|6|7|8

版权所有 © 华为云计算技术有限公司

锁模式名称	锁等级	锁用途	冲突关系
共享更新排他锁（ShareUpdateExclusiveLock）	4	VACUUM（非FULL）、ANALYZE、CREATE INDEX CONCURRENTLY、COMMENT ON，允许其他事务读取但阻止写入。	4|5|6|7|8
共享锁（ShareLock）	5	CREATE INDEX（非CONCURRENTLY），允许其他事务读取但阻止写入。	3|4|6|7|8
共享行排他锁（ShareRowExclusiveLock）	6	ROW SELECT...FOR UPDATE，类似RowExclusiveLock，但允许RowShareLock。	3|4|5|6|7|8
排他锁（ExclusiveLock）	7	阻止RowShareLock或SELECT ... FOR UPDATE操作。	2|3|4|5|6|7|8
访问排他锁（AccessExclusiveLock）	8	ALTER TABLE、DROP TABLE、VACUUM FULL，完全阻止其他事务读取和写入。	1|2|3|4|5|6|7|8

分区表通过表锁+分区锁两重设计，在表和分区上分别施加不同级别的常规锁，来保证DQL、DML、DDL并发过程中的行为控制。有关分区表锁粒度参考下表：

表 3-4 分区表业务锁粒度

执行语句	表锁+分区锁
SELECT全表	执行CN主表1级锁+所有分区子表1级锁（列存不会加子表锁）
	DN主表1级锁+所有分区子表1级锁
SELECT子分区	执行CN主表1级锁+分区子表1级锁
	DN主表1级锁+分区子表1级锁
INSERT/UPDATE/DELETE主表	执行CN主表3级锁
	主表3级锁+分区子表3级锁
INSERT/UPDATE/DELETE子表	执行CN主表3级锁+分区子表3级锁
	主表3级锁+分区子表3级锁
访问分区子表触发的light runtime analyze	执行CN主表1级锁+分区子表1级锁
	主表1级锁+所有分区子表1级锁（其他子分区及锁用完即放）
访问分区子表触发的normal runtime analyze	执行CN主表4级锁

 版权所有 © 华为云计算技术有限公司

执行语句	表锁+分区锁
	主表4级锁+所有分区子表4级锁
ADD PARTITION	CN主表3级锁+分区子表8级锁+集群锁
	主表3级锁+分区子表8级锁
SPLIT/MERGE/DROP PARTITION	主表8级锁+分区子表8级锁+集群锁
	主表8级锁+分区子表8级锁
TRUNCATE PARTITION	CN主表3级锁+分区子表8级锁
	主表3级锁+分区子表8级锁
EXCHANGE PARTITION	CN主表3级锁+分区子表8级锁
	主表3级锁+分区子表8级锁

3.6 创建和管理 DWS 索引

索引可以提高数据的访问速度，但同时也增加了插入、更新和删除操作的处理时间。所以是否要为表增加索引，索引建立在哪些字段上，是创建索引前必须要考虑的问题。需要分析应用程序的业务处理、数据使用、经常被用作查询的条件或者被要求排序的字段来确定是否建立索引。

索引类型

- B-Tree：B-tree索引使用一种类似于B+树的结构来存储数据的键值，通过这种结构能够快速地查找索引。B-Tree适合支持比较查询以及查询范围。
- GIN：GIN索引是倒排索引，可以处理包含多个键的值（比如数组）。
- GiST：GiST索引适用于几何和地理等多维数据类型和集合数据类型。
- PSORT：PSORT索引。针对列存表进行局部排序索引。

行存表支持的索引类型：B-Tree（行存表缺省值）、GIN、GiST。列存表支持的索引类型：PSORT（列存表缺省值）、B-Tree、GIN。

说明

对于点查询场景，推荐建立B-Tree索引。

索引的选择原则

索引建立在数据库表中的某些列上。因此，在创建索引时，应该仔细考虑在哪些列上创建索引。

- 在经常需要搜索查询的列上创建索引，可以加快搜索的速度。
- 在作为主键的列上创建索引，强制该列的唯一性和组织表中数据的排列结构。
- 在经常使用连接的列上创建索引，这些列主要是一些外键，可以加快连接的速度。
- 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的。

版权所有 © 华为云计算技术有限公司

- 在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间。
- 在经常使用WHERE子句的列上创建索引，加快条件的判断速度。
- 为经常出现在关键字ORDER BY、GROUP BY、DISTINCT后面的字段建立索引。

  说明

  - 索引创建成功后，系统会自动判断何时引用索引。当系统认为使用索引比顺序扫描更快时，就会使用索引。
  - 索引创建成功后，必须和表保持同步以保证能够准确地找到新数据，这样就增加了数据操作的负荷。因此请定期删除无用的索引。

创建索引

DWS支持4种创建索引的方式请参见**表3-5**。

说明

- 索引创建成功后，系统会自动判断何时引用索引。当系统认为使用索引比顺序扫描更快时，就会使用索引。
- 索引创建成功后，必须和表保持同步以保证能够准确地找到新数据，这样就增加了数据操作的负荷。因此请定期删除无用的索引。

表 3-5 索引方式

索引方式	描述
唯一索引	可用于约束索引属性值的唯一性，或者属性组合值的唯一性。如果一个表声明了唯一约束或者主键，则DWS自动在组成主键或唯一约束的字段上创建唯一索引（可能是多字段索引），以实现这些约束。目前，DWS只有B-Tree可以创建唯一索引。
多字段索引	一个索引可以定义在表中的多个属性上。目前，DWS中的B-Tree支持多字段索引，且最多可在32个字段上创建索引。
部分索引	建立在一个表的子集上的索引，这种索引方式只包含满足条件表达式的元组。
表达式索引	索引建立在一个函数或者从表中一个或多个属性计算出来的表达式上。表达式索引只有在查询时使用与创建时相同的表达式才会起作用。

- 创建一个普通表。

```
CREATE TABLE tpcds.customer_address_bak AS TABLE tpcds.customer_address;
```

- 创建普通索引

  如果对于tpcds.customer_address_bak表，需要经常进行以下查询。

```
SELECT ca_address_sk FROM tpcds.customer_address_bak WHERE ca_address_sk=14888;
```

  通常，数据库系统需要逐行扫描整个tpcds.customer_address_bak表以寻找所有匹配的元组。如果表tpcds.customer_address_bak的规模很大，但满足WHERE条件的只有少数几个（可能是零个或一个），则这种顺序扫描的性能就比较差。如果让数据库系统在ca_address_sk属性上维护一个索引，用于快速定位匹配的元组，则数据库系统只需要在搜索树上查找少数的几层就可以找到匹配的元组，这

 版权所有 © 华为云计算技术有限公司

将会大幅度提高数据查询的性能。同样，在数据库中进行更新和删除操作时，索引也可以提升这些操作的性能。

使用以下命令创建索引。

```
CREATE INDEX index_wr_returned_date_sk ON tpcds.customer_address_bak (ca_address_sk);
```

- 创建唯一索引

  如果一个表声明了唯一约束或者主键，则DWS自动在组成主键或唯一约束的字段上创建唯一索引（可能是多字段索引）。如果建表时未声明唯一约束或主键，可使用CREATE INDEX创建。

  ```
  CREATE UNIQUE INDEX unique_index ON tpcds.customer_address_bak(ca_address_sk);
  ```

- 创建多字段索引

  假如用户需要经常查询表tpcds.customer_address_bak中ca_address_sk是5050，且ca_street_number小于1000的记录，使用以下命令进行查询。

  ```
  SELECT ca_address_sk,ca_address_id FROM tpcds.customer_address_bak WHERE ca_address_sk = 5050 AND ca_street_number < 1000;
  ```

  使用以下命令在字段ca_address_sk和ca_street_number上定义一个多字段索引。

  ```
  CREATE INDEX more_column_index ON
  tpcds.customer_address_bak(ca_address_sk ,ca_street_number );
  ```

- 创建部分索引

  如果只需要查询ca_address_sk为5050的记录，可以创建部分索引来提升查询效率。

  ```
  CREATE INDEX part_index ON tpcds.customer_address_bak(ca_address_sk) WHERE ca_address_sk = 5050;
  ```

- 创建表达式索引

  假如经常需要查询ca_street_number小于1000的信息，执行如下命令进行查询。

  ```
  SELECT * FROM tpcds.customer_address_bak WHERE trunc(ca_street_number) < 1000;
  ```

  可以为上面的查询创建表达式索引：

  ```
  CREATE INDEX para_index ON tpcds.customer_address_bak (trunc(ca_street_number));
  ```

查看索引

- 执行如下命令查询系统和用户定义的所有索引。

  ```
  SELECT RELNAME FROM PG_CLASS WHERE RELKIND='i';
  ```

- 执行如下命令查询指定索引的信息。

  ```
  \di+ index_wr_returned_date_sk
  ```

重建索引

- 重建索引index_wr_returned_date_sk：

  ```
  REINDEX INDEX index_wr_returned_date_sk;
  ```

- 重建表上的所有索引：

  ```
  REINDEX TABLE tpcds.customer_address_bak;
  ```

删除索引

使用DROP INDEX命令删除索引：

```
DROP INDEX index_wr_returned_date_sk;
```

 版权所有 © 华为云计算技术有限公司

3.7 创建和使用 DWS 序列

序列Sequence是用来产生唯一整数的数据库对象。序列的值是按照一定规则自增的整数。因为自增所以不重复，因此说Sequence具有唯一标识性。这也是Sequence常被用作主键的原因。

通过序列使某字段成为唯一标识符的方法有两种：

- 一种是声明字段的类型为序列整型，由数据库在后台自动创建一个对应的Sequence。
- 另一种是使用CREATE SEQUENCE自定义一个新的Sequence，然后通过nextval('sequence_name')函数递增序列并返回新值，将nextval('sequence_name')函数读取的序列值，指定为某一字段的默认值，这样该字段就可以作为唯一标识符。

创建序列

方法一： 声明字段类型为序列整型来定义标识符字段。例如：

```
CREATE TABLE T1
(
    id    serial,
    name  text
);
```

方法二： 创建序列，并通过nextval('sequence_name')函数指定为某一字段的默认值。这种方式更灵活，可以为序列定义cache，一次预申请多个序列值，减少与GTM的交互次数，来提高性能。

1. 创建序列
   ```
   CREATE SEQUENCE seq1 cache 100;
   ```
2. 指定为某一字段的默认值，使该字段具有唯一标识属性。
   ```
   CREATE TABLE T2
   (
      id   int not null default nextval('seq1'),
      name text
   );
   ```

说明

- 除了为序列指定了cache，方法二所实现的功能基本与方法一类似。但是一旦定义cache，序列将会产生空洞(序列值为不连贯的数值，如：1.4.5)，并且不能保序。另外为某序列指定从属列后，该列删除，对应的sequence也会被删除。 虽然数据库并不限制序列只能为一列产生默认值，但最好不要多列共用同一个序列。
- 当前版本只支持在定义表的时候指定自增列，或者指定某列的默认值为nextval('seqname')， 不支持在已有表中增加自增列或者增加默认值为nextval('seqname')的列。
- 建议控制SEQUENCE的使用数量（含SERIAL类型），因分布式场景下SEQUENCE中序列值的产生依赖GTM全局管理，频繁的与GTM交互会严重影响性能。建议引用公共的SEQUENCE对象，或者针对需要唯一值的场景使用UUID进行替换。同时为尽量减少与GTM的交互，针对手动创建的SEQUENCE，建议指定CACHE值不小于10000。

修改一个序列

ALTER SEQUENCE命令更改现有序列的属性，包括修改拥有者、归属列和最大值。

- 指定序列与列的归属关系。

 版权所有 © 华为云计算技术有限公司

将序列和一个表的指定字段进行关联。在删除那个字段或其所在表的时候会自动删除已关联的序列。

```
ALTER SEQUENCE seq1 OWNED BY T2.id;
```

- 将序列serial的最大值修改为300：

```
ALTER SEQUENCE seq1 MAXVALUE 300;
```

删除一个序列

使用DROP SEQUENCE命令删除一个序列。 例如，将删除名为seq1的序列：

```
DROP SEQUENCE seq1;
```

注意事项

新序列值的产生是靠GTM维护的，默认情况下，每申请一个序列值都要向GTM发送一次申请，GTM在当前值的基础上加上步长值作为产生的新值返回给调用者。GTM作为全局唯一的节点，势必成为性能的瓶颈，所以对于需要大量频繁产生序列号的操作，如使用Bulkload（批量快速导入数据）功能进行数据导入场景，是非常不推荐产生默认序列值的。比如，在下面所示的场景中， INSERT FROM SELECT语句的性能会非常慢。

```
CREATE SEQUENCE newSeq1;
CREATE TABLE newT1
        (
          id   int not null default nextval('newSeq1'),
          name text
        );
INSERT INTO newT1(name) SELECT name from T1;
```

可以提高性能的写法是（假设T1表导入newT1表中的数据为10000行）：

```
INSERT INTO newT1(id, name) SELECT id,name from T1;
SELECT SETVAL('newSeq1',10000);
```

说明

序列操作函数nextval()，setval() 等均不支持回滚。另外setval设置的新值，会对当前会话的nextval立即生效，但对其他会话，如果定义了cache，不会立即生效，在用尽所有缓存的值后，其变动才被其他会话感知。所以为了避免产生重复值，要谨慎使用setval，设置的新值不能是已经产生的值或者在缓存中的值。

如果必须要在bulkload场景下产生默认序列值，则一定要为newSeq1定义足够大的cache，并且不要定义Maxvalue或者Minvalue。数据库会试图将nextval('sequence_name')的调用下推到Data Node，以提高性能。 目前GTM对并发的连接请求是有限制的，当Data Node很多时，将产生大量并发连接， 这时一定要控制bulkload的并发数目，避免耗尽GTM的连接资源。如果目标表为复制表(DISTRIBUTE BY REPLICATION)时下推将不能进行。当数据量较大时，这对数据库将是个灾难。除了性能问题之外，空间也可能会剧烈膨胀，在导入结束后，需要用vacuum full来恢复。最好的方式还是如上建议的，不要在bulkload的场景中产生默认序列值。

另外，序列创建后，在每个节点上都维护了一张单行表，存储序列的定义及当前值，但此当前值并非GTM上的当前值，只是保存本节点与GTM交互后的状态。如果其他节点也向GTM申请了新值，或者调用了Setval修改了序列的状态，不会刷新本节点的单行表，但因每次申请序列值是向GTM申请，所以对序列正确性没有影响。

 版权所有 © 华为云计算技术有限公司

3.8 创建和管理 DWS 视图

视图允许用户保存常用的或者复杂的查询。视图在磁盘上并没有被物理存储，当用户访问视图时查询会作为一个子查询运行。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。视图每次被引用的时候都会运行一次。

创建视图

使用CREATE VIEW命令创建新视图。

```
CREATE OR REPLACE VIEW MyView AS SELECT * FROM tpcds.customer WHERE c_customer_sk < 150;
```

说明

CREATE VIEW中的OR REPLACE可有可无，当存在OR REPLACE时，表示若以前存在该视图就进行替换。

查看视图

- 查看MyView视图，查询结果为当前实时数据。

  ```
  SELECT * FROM myview;
  ```

- 查看当前用户下的视图。

  ```
  SELECT * FROM user_views;
  ```

- 查看所有视图。

  ```
  SELECT * FROM dba_views;
  ```

- 查看某视图的具体信息。

  执行如下命令查询dba_users视图的详细信息。

  ```
  \d+ dba_users
                        View "PG_CATALOG.DBA_USERS"
   Column  |         Type          | Modifiers | Storage  | Description
  ----------+-----------------------+-----------+----------+-------------
   USERNAME | CHARACTER VARYING(64) |           | extended |
  View definition:
   SELECT PG_AUTHID.ROLNAME::CHARACTER VARYING(64) AS USERNAME
     FROM PG_AUTHID;
  ```

重建视图

使用ALTER VIEW命令在不输入查询语句的情况下重建视图。

```
ALTER VIEW myview REBUILD;
```

删除视图

使用DROP VIEW命令删除一个视图。

```
DROP VIEW myview;
```

DROP VIEW ... CASCADE命令也可以级联删除依赖此视图的对象。 例如，如果A视图依赖于将要被删除的B视图，那么A视图也将被删除。 如果没有CASCADE选项，这个DROP VIEW命令将会失败。

版权所有 © 华为云计算技术有限公司

3.9 创建和管理 DWS 定时任务

DWS支持用户创建定时任务，当设置的任务时间点到达后可以自动触发任务的执行，从而可以减少用户运维的工作量。

数据库兼容Oracle定时任务功能主要通过DBMS.JOB高级包提供的接口，可以实现定时任务的创建、任务到期自动执行、任务删除、修改任务属性（包括：任务id、任务的关闭开启、任务的触发时间、触发时间间隔、任务内容等）。

说明

- 定时任务的执行语句暂不记录到系统的**实时TopSQL**日志中，仅8.2.1及以上集群版本支持记录。
- DWS 时间设置默认为UTC时间，定时任务的执行时间需根据用户所在时区差进行转换。

定时任务管理

步骤1 创建测试表：

```
CREATE TABLE test(id int, time date);
```

当结果显示为如下信息，则表示创建成功。

```
CREATE TABLE
```

步骤2 创建自定义存储过程：

```
CREATE OR REPLACE PROCEDURE PRC_JOB_1()
AS
N_NUM integer :=1;
BEGIN
FOR I IN 1..1000 LOOP
INSERT INTO test VALUES(I,SYSDATE);
END LOOP;
END;
/
```

当结果显示为如下信息，则表示创建成功。

```
CREATE PROCEDURE
```

步骤3 创建任务：

- 新创建的任务（未指定job_id）表示每隔1分钟执行一次存储过程PRC_JOB_1。

```
call dbms_job.submit('call public.prc_job_1(); ', sysdate, 'interval ''1 minute''', :a);
job
-----
1
(1 row)
```

- 指定job_id创建任务。

```
call dbms_job.isubmit(2,'call public.prc_job_1(); ', sysdate, 'interval ''1 minute''');
isubmit
---------


(1 row)
```

步骤4 通过USER_JOBS视图查看当前用户已创建的任务信息。

需要有系统管理员权限才可以访问此系统视图，字段说明详见**表17-396**。

```
select job,dbname,start_date,last_date,this_date,next_date,broken,status,interval,failures,what from
user_jobs;
 job |  dbname  |         start_date         |         last_date          |         this_date          |         next_date          |
```


版权所有 © 华为云计算技术有限公司

```
broken | status |       interval        | failures |            what

-----+----------+-----------------------------+-----------------------------+----------------------------
+--------------------+--------+--------+--------------------+----------+----------------
-----------
   1 | db_demo | 2022-03-25 07:58:01.829436 | 2022-03-25 07:58:03.174817 | 2022-03-25 07:58:01.829436 |
2022-03-25 07:59:01 | n      | s      | interval '1 minute' |        0 | call public.prc
_job_1();
   2 | db_demo | 2022-03-25 07:58:15.893383 | 2022-03-25 07:58:16.608959 | 2022-03-25 07:58:15.893383 |
2022-03-25 07:59:15 | n      | s      | interval '1 minute' |        0 | call public.prc
_job_1();
(2 rows)
```

步骤5 停止任务。

```
call dbms_job.broken(1,true);
broken
--------

(1 row)
```

步骤6 启动任务。

```
call dbms_job.broken(1,false);
broken
--------

(1 row)
```

步骤7 修改任务属性。

- 修改JOB的next_date参数信息。例如，修改Job1的next_date为1小时以后开始执行。

  ```
  call dbms_job.next_date(1, sysdate+1.0/24);
  next_date
  -----------

  (1 row)
  ```

- 修改JOB的Interval参数信息。例如，修改Job1的Interval为每隔1小时执行一次。

  ```
  call dbms_job.interval(1,'sysdate + 1.0/24');
  interval
  ----------

  (1 row)
  ```

- 修改JOB的What参数信息。例如，修改Job1的What为执行SQL语句“insert into public.test values(333, sysdate+5);”。

  ```
  call dbms_job.what(1,'insert into public.test values(333, sysdate+5);');
  what
  ------

  (1 row)
  ```

- 同时修改JOB的next_date、interval、what等多个参数信息。

  ```
  call dbms_job.change(1, 'call public.prc_job_1();', sysdate, 'interval ''1 minute''');
  change
  --------

  (1 row)
  ```

步骤8 删除JOB。

```
call dbms_job.remove(1);
remove
--------

(1 row)
```

步骤9 JOB的权限控制。

 版权所有 © 华为云计算技术有限公司

- 当创建一个JOB时，该JOB会和创建该JOB的数据库和用户绑定（即：pg_job系统视图新增的JOB记录中的dbname和log_user）。
- 如果当前用户是DBA用户、系统管理员、该JOB的创建用户（即：pg_job中的log_user），那么该用户有权限通过高级包接口remove、change、next_date、what、interval删除或修改JOB的参数信息。否则，会提示当前用户没有权限操作该JOB。
- 如果当前数据库是该JOB创建所属的数据库（即：为pg_job系统视图中的dbname），那么连接到当前数据库上可以通过高级包接口remove、change、next_date、what、interval删除或修改JOB的参数信息。
- 当删除JOB所属的数据库（即：为pg_job系统视图中的dbname）时，系统会关联删除该数据库从属的JOB记录。
- 当删除JOB所属的用户（即：为pg_job系统视图中的log_user）时，系统会关联删除该用户从属的JOB记录。

----结束

3.10 查看 DWS 系统表

除了创建的表以外，数据库还包含很多系统表。这些系统表包含集群安装信息以及DWS上运行的各种查询和进程的信息。可以通过查询系统表来收集有关数据库的信息。

查看数据库中包含的表

例如，在PG_TABLES系统表中查看public schema中包含的所有表。

```
SELECT distinct(tablename) FROM pg_tables WHERE SCHEMANAME = 'public';
```

结果类似如下这样：

```
    tablename
--------------------
 err_hr_staffs
 test
 err_hr_staffs_ft3
 web_returns_p1
 mig_seq_table
 films4
(6 rows)
```

查看数据库用户

通过PG_USER可以查看数据库中所有用户的列表，还可以查看用户ID（USESYSID）和用户权限。

```
SELECT * FROM pg_user;
usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd  | valbegin | valuntil |   respool
| parent | spacelimit | useconfig | nodegroup | tempspacelimit | spillspacelim
it
---------+----------+-------------+----------+-----------+---------+----------+----------+----------+--------------
+--------+------------+-----------+-----------+----------------+--------------
---
 Ruby    |       10 | t          | t       | t        | t       | ******** |          |          | default_pool |      0 |
|          |          |              |
 dbadmin |    16393 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 lily    |    16691 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
```


版权所有 © 华为云计算技术有限公司

```
 jack    |    70694 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
(4 rows)
```

DWS在内部使用Ruby执行日常管理和维护任务。可以向SELECT语句添加WHERE usesysid > 10来筛选查询，使其只显示用户定义的用户名称。

```
SELECT * FROM pg_user WHERE usesysid > 10;
 usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |   passwd   | valbegin | valuntil |
respool    | parent | spacelimit | useconfig | nodegroup | tempspacelimit | spillspacelim
it
---------+----------+-------------+----------+-----------+---------+----------+----------+----------+-------------
+--------+------------+-----------+-----------+----------------+--------------
---
 dbadmin |    16393 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 lily    |    16691 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
 jack    |    70694 | f          | f       | f        | f       | ******** |          |          | default_pool |      0 |
|          |          |              |
(3 rows)
```

查看和停止正在运行的查询语句

通过视图**PG_STAT_ACTIVITY**可以查看正在运行的查询语句。方法如下：

步骤1 设置参数track_activities为on。

```
SET track_activities = on;
```

当此参数为on时，数据库系统才会收集当前活动查询的运行信息。

步骤2 查看正在运行的查询语句。以查看正在运行的查询语句所连接的数据库名、执行查询的用户、查询状态及查询对应的PID为例：

```
SELECT datname, usename, state,pid FROM pg_stat_activity;
```

如果state字段显示为idle，则表明此连接处于空闲，等待用户输入命令。

如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pg_stat_activity WHERE state != 'idle';
```

步骤3 若需要取消运行时间过长的查询，通过PG_TERMINATE_BACKEND函数，根据线程ID结束会话。

```
SELECT PG_TERMINATE_BACKEND(139834759993104);
```

显示类似如下信息，表示结束会话成功。

```
PG_TERMINATE_BACKEND
----------------------
 t
(1 row)
```

显示类似如下信息，表示用户执行了结束当前会话的操作。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
```

 版权所有 © 华为云计算技术有限公司

说明

gsql客户端使用PG_TERMINATE_BACKEND函数结束当前会话后台线程时，客户端不会退出而是自动重连。即还会返回“The connection to the server was lost. Attempting reset: Succeeded.”

```
FATAL: terminating connection due to administrator command
FATAL: terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

----结束

 版权所有 © 华为云计算技术有限公司

4 Oracle、Teradata 和 MySQL 语法兼容性差异

DWS支持Oracle（以下简称ORA）、Teradata（以下简称TD）和MySQL三种兼容模式，分别兼容Oracle、Teradata和MySQL语法，不同兼容模式下的语法行为有一些差异。

数据库兼容模型可以在创建数据库时指定（由DBCOMPATIBILITY参数控制），语法示例如下，具体参见**CREATE DATABASE**语法。

```
CREATE DATABASE td_compatible_db DBCOMPATIBILITY 'TD'; --创建兼容Teradata的数据库
CREATE DATABASE ora_compatible_db DBCOMPATIBILITY 'ORA'; --创建兼容Oracle的数据库
CREATE DATABASE mysql_compatible_db DBCOMPATIBILITY 'MYSQL'; --创建兼容MySQL的数据库
```

查询结果：

```
SELECT datname,datcompatibility FROM PG_DATABASE WHERE datname LIKE '%compatible_db';
      datname       | datcompatibility
--------------------+------------------
 td_compatible_db    | TD
 ora_compatible_db   | ORA
 mysql_compatible_db | MYSQL
(3 rows)
```

表 4-1 兼容项差异

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
数据类型date	date会转为timestamp，包含年月日时分秒。	只有年月日。	只有年月日。
空串	只有null	区分空串和null	区分空串和null
空串转数字	null	转换为0。	转换为0。
超长字符自动截断	不支持。	支持(GUC参数td_compatible_truncation设置为ON)。	不支持。
varchar + int运算	转为bigint + int计算。	转为numeric + numeric计算。	转为bigint + int计算。

 版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
null拼接	非null对象与null拼接后返回非null对象。 例如，'abc'||null返回'abc'。	GUC参数behavior_compat_options增加strict_text_concat_td选项后，兼容TD行为，null类型拼接后返回null。 例如，'abc'||null返回null。	兼容MySQL行为，null类型拼接后返回null。 例如，'abc'||null返回null。
char(n)类型拼接	char(n)类型做拼接时移除右侧空格和占位。 例如，cast('a' as char(3))||'b'返回'ab'。	GUC参数behavior_compat_options增加bpchar_text_without_rtrim选项后，char(n)类型做拼接时，保留空格，并补足空格至指定的n长度。 当前不支持“比较字符串时忽略尾部空格”，拼接后结果如果存在尾部空格，进行比较时会对空格敏感。 例如，cast('a' as char(3))||'b'返回'a b'。	移除右侧空格和占位。
concat(str1,str2)	返回所有非null字符串的连接	返回所有非null字符串的连接	入参中存在null时，返回结果为null。
left和right负数处理	返回除最后/前|n|个字符以外的所有字符。	返回除最后/前|n|个字符以外的所有字符。	返回空串。
lpad(string text, length int [, fill text]) rpad(string text, length int [, fill text])	通过填充字符fill(缺省为空格)，把string填充到length长度，如果string已经比length长则将其尾部截断。如果fill为空串或者length为负数则返回null。	如果fill为空串且string长度小于length时，返回原字符串，如果length为负数则返回空串。	如果fill为空串且string长度小于length时，返回空串，如果length为负数则返回null。
substr(str, s[, n])	s = 0时，返回前n个字符。	s = 0时，返回前n个字符。	s = 0时，返回空串。

版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
substring(str, s[, n]) substring(str [from s] [for n])	s = 0时，返回前n - 1个字符 s < 0时，返回前s + n - 1个字符 n < 0时，报错。	s = 0时，返回前n - 1个字符。 s < 0时，返回前s + n - 1个字符。 n < 0时，报错。	s = 0时，返回空串。 s < 0时，倒数第|s|个字符位置开始截取n个字符。 n < 0时，返回空串。
trim、ltrim、rtrim、btrim(string[,characters])	从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。	从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。	从字符串string的指定位置删除等于characters的字符串（缺省为空格）。
log(x)	以10为底的对数。	以10为底的对数。	自然对数。
mod(x, 0)	除数为0时返回x。	除数为0时返回x。	除数为0时报错。
to_char(date)	入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为timestamp。	入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为date，且格式为'YYYY/MM/DD'(GUC参数convert_empty_str_to_null_td打开)。	入参最大值支持timestamp类型的最大值和date类型的最大值；返回值类型为date。
to_date, to_timestamp和to_number空串处理	返回null。	返回null(GUC参数convert_empty_str_to_null_td打开)。	to_date和to_timestamp返回null，to_number中参数为空串时，返回0。
last_day和next_day返回类型	timestamp类型。	timestamp类型。	date类型。
add_months返回类型	timestamp类型。	timestamp类型。	入参为date类型，返回date类型。 入参为timestamp类型，返回timestamp类型。 入参为timestamptz类型，返回timestamptz类型。
CURRENT_TIME CURRENT_TIME(p)	获取当前事务的时间，返回值类型为timetz。	获取当前事务的时间，返回值类型为timetz。	获取当前语句执行时的时间，返回值类型为time。

版权所有 © 华为云计算技术有限公司

兼容项	Oracle兼容	Teradata兼容	MySQL兼容
CURRENT_TIMESTAMP CURRENT_TIMESTAMP(p)	获取当前语句执行时的时间，返回值类型为timestamptz。	获取当前语句执行时的时间，返回值类型为timestamptz。	获取当前语句执行时的时间，返回值类型为timestamp。
CURDATE	不支持。	不支持。	获取当前语句执行时的日期，返回值类型为date。
CURTIME(p)	不支持。	不支持。	获取当前语句执行时的时间，返回值类型为time。
LOCALTIME LOCALTIME(p)	获取当前事务的时间，返回值类型为time。	获取当前事务的时间，返回值类型为time。	获取当前语句执行时的时间，返回值类型为timestamp。
LOCALTIMESTAMP LOCALTIMESTAMP(p)	获取当前事务的时间，返回值类型为timestamp。	获取当前事务的时间，返回值类型为timestamp。	获取当前语句执行时的时间，返回值类型为timestamp。
SYSDATE SYSDATE(p)	获取当前语句执行时的时间，返回值类型为timestamp(0)。	获取当前语句执行时的时间，返回值类型为timestamp(0)。	获取当前系统的时间，返回值类型为timestamp(0)。此函数不可下推，建议用current_date代替。
now()	获取当前事务时间，返回值类型为timestamptz。	获取当前事务时间，返回值类型为timestamptz。	获取语句执行的时间，返回值类型为timestamptz。
操作符'^'	幂运算。	幂运算。	异或。
表达式greatest、least	返回所有非null入参的比较结果。	返回所有非null入参的比较结果。	入参中存在null时，返回结果为null。
表达式case、coalesce、if、ifnull入参类型不同	报错。	兼容TD行为，支持数字和字符串之间的类型转换，比如coalesce参数输入int和varchar类型，解析成varchar类型。	兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。
反引号	不支持。	不支持。	区分MySQL的保留字与普通字符。

举例说明

以下以兼容Teradata模式为例运行SQL，如果需要查看Oracle、MySQL兼容模式下的行为差异，可以切换到对应的数据库ora_compatible_db、mysql_compatible_db，然

版权所有 © 华为云计算技术有限公司

后运行以下相同的SQL语句（表名相应替换为ora_table、mysql_table），即可体会上表所列举的不同兼容模式下的行为差异。

```
CREATE TABLE td_table(a INT,b VARCHAR(5),c date);
INSERT INTO td_table VALUES(1,null,CURRENT_DATE);
INSERT INTO td_table VALUES(2,'',CURRENT_DATE);
```

区分空串和 NULL，date 类型只显示年月日

TD和MySQL模式下，区分空串和NULL，ORA模式下不区分空串和NULL，date类型会转为timestamp，包含年月日时分秒。

TD 模式

```
SELECT a, b, b IS NULL AS null, c FROM td_table;
```

```
 a | b | null |     c
---+---+------+------------
 2 |   | f    | 2025-08-12
 1 |   | t    | 2025-08-12
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
    date
------------
 2025-08-12
(1 row)
```

ORA 模式

```
SELECT a, b, b IS NULL AS null, c FROM ora_table;
```

```
 a | b | null |          c
---+---+------+---------------------
 1 |   | t    | 2025-08-12 00:00:00
 2 |   | t    | 2025-08-12 00:00:00
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
    date
------------
 2025-08-12
(1 row)
```

MySQL 模式

```
SELECT a, b, b IS NULL AS null, c FROM mysql_table;
```


版权所有 © 华为云计算技术有限公司

```
 a | b | null |     c
---+---+------+------------
 2 |   | f    | 2025-08-12
 1 |   | t    | 2025-08-12
(2 rows)
```

```
SELECT CURRENT_DATE;
```

```
 current_date
--------------
 2025-08-12
(1 row)
```

空串转 int，转为 0

TD数据库不同于Oracle，Oracle将空串当作NULL进行处理，TD在将空串转换为数值类型的时候，默认将空串转换为0进行处理，因此查询空串会查询到数值为0的数据。

同样地，在TD兼容模式下，字符串转换数值的过程中，也会将空串默认转换为相应数值类型的0值进行处理。除此之外，' - '、' + '、' '这些字符串也都会在TD兼容模式下默认转换为0进行处理，但是小数点字符串' . '会报错。

TD 模式

```
SELECT b::int FROM td_table WHERE b = '';
```

```
 b
---
 0
(1 row)
```

ORA 模式

```
SELECT b::int FROM ora_table WHERE b = '';
```

```
 b
---
(0 rows)
```

MySQL 模式

```
SELECT b::int FROM mysql_table WHERE b = '';
```

```
 b
---
 0
(1 row)
```


版权所有 © 华为云计算技术有限公司

超长字符自动截断

- TD模式下，**td_compatible_truncation**参数设置为on时，将启用超长字符串自动截断功能，在后续的insert语句中（不包含外表的场景下），对目标表中char和varchar类型的列上插入超长字符串时，系统会自动按照目标表中相应列定义的最大长度对超长字符串进行截断。
- ORA和MySQL模式下，插入超长字符串会报错。

TD 模式

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO td_table VALUES(3,'12345678',CURRENT_DATE);
SELECT * FROM td_table WHERE a = 3;
```

```
 a |   b   |     c
---+-------+------------
 3 | 12345 | 2025-08-12
(1 row)
```

ORA 模式

参数**td_compatible_truncation**设置无效，插入超长字符会报错。

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO ora_table VALUES(3,'12345678',CURRENT_DATE);
```

```
ERROR:  dn_6001_6002: value too long for type character varying(5)
CONTEXT:  referenced column: b
```

MySQL 模式

参数**td_compatible_truncation**设置无效，插入超长字符会报错。

```
SHOW td_compatible_truncation;
SET td_compatible_truncation = ON;
INSERT INTO mysql_table VALUES(3,'12345678',CURRENT_DATE);
```

```
ERROR:  dn_6001_6002: value too long for type character varying(5)
CONTEXT:  referenced column: b
```

varchar + int 运算，转为 numeric + numeric 计算

- TD模式下，varchar + int运算，转为numeric + numeric计算。
- ORA和MySQL模式下，转为bigint + int计算。

TD 模式

```
EXPLAIN VERBOSE SELECT b + a FROM td_table WHERE a = 3;
```


版权所有 © 华为云计算技术有限公司

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((td_table.b)::numeric + (td_table.a)::numeric))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::numeric + a::numeric AS "?column?" FROM public.td_table WHERE a = 3
```

ORA 模式

```
INSERT INTO ora_table VALUES(3,'12345',CURRENT_DATE);
EXPLAIN VERBOSE SELECT b + a FROM ora_table WHERE a = 3;
```

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((ora_table.b)::bigint + ora_table.a))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::bigint + a AS "?column?" FROM public.ora_table WHERE a = 3
```

MySQL 模式

```
INSERT INTO mysql_table VALUES(3,'12345',CURRENT_DATE);
EXPLAIN VERBOSE SELECT b + a FROM mysql_table WHERE a = 3;
```

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                         Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (((mysql_table.b)::bigint + mysql_table.a))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT b::bigint + a AS "?column?" FROM public.mysql_table WHERE a = 3
```

null 拼接返回 null

- TD模式：GUC参数**behavior_compat_options**增加strict_text_concat_td选项，日期、时间、数字、字符串和null拼接均会返回null。
- ORA模式：非null对象与null拼接后返回非null对象。例如，'abc'||null返回'abc'。
- MySQL模式：兼容MySQL行为，null类型拼接后返回null。例如，'abc'||null返回null。

版权所有 © 华为云计算技术有限公司

TD 模式

```
SET behavior_compat_options = 'strict_text_concat_td';
SELECT '2024-02-07 12:12:12'::TIMESTAMP || NULL;
SELECT '12:12:12'::TIME || NULL;
SELECT '12'::TINYINT || NULL;
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------

(1 row)
```

ORA 模式

```
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------
 abc
(1 row)
```

MySQL 模式

```
SELECT 'abc'::CHAR(10) || NULL;
```

```
 ?column?
----------

(1 row)
```

char(n)类型拼接

- TD模式下，GUC参数**behavior_compat_options**增加bpchar_text_without_rtrim选项，char(n)类型做拼接时，保留空格，并补足空格至指定的n长度。
- ORA和MySQL模式下，不保留空格。

TD 模式

例如，a后面带3个空格，强制转换成char(10)后，由于不足10个字符，则会用空格填充到10个字符，占用10个字节的存储空间，每个字节是8位，所以结果是80位。

```
SET behavior_compat_options = 'bpchar_text_without_rtrim';
SELECT bit_length('a   '::char(10));
```

```
 bit_length
------------
          80
(1 row)
```


版权所有 © 华为云计算技术有限公司

ORA 和 MySQL 模式

ORA和MySQL模式下，不保留空格，a只占了1个字符，占用1个字节的存储空间，所以结果是8。

```
SELECT bit_length('a   '::char(10));
```

```
 bit_length
------------
          8
(1 row)
```

concat(str1,str2)

TD和ORA模式下，返回所有非null字符串的连接。MySQL模式下，入参中存在null时，返回结果为null。

TD 和 ORA 模式

```
SELECT concat(null, 'World!');
```

```
 concat
--------
 World!
(1 row)
```

MySQL 模式

```
SELECT concat(null, 'World!');
```

```
 concat
--------

(1 row)
```

left 和 right 负数处理

TD和ORA兼容模式下，left和right函数分别返回除最后、最前|n|个字符以外的所有字符。MySQL模式下，返回空串。

TD 和 ORA 模式

```
SELECT left('abcde', -2);
```

```
 left
------
 abc
(1 row)
```

```
SELECT right('abcde', -2);
```


版权所有 © 华为云计算技术有限公司

```
 right
-------
 cde
(1 row)
```

MySQL 模式

```
SELECT left('abcde', -2);
```

```
 left
------


(1 row)
```

```
SELECT right('abcde', -2);
```

```
 right
-------


(1 row)
```

lpad 和 rpad 空串处理

- lpad和rpad函数在TD兼容模式下：如果fill为空串且string长度小于length时，返回原字符串。如果length为负数则返回空串。
- ORA模式下，都返回null。
- MySQL模式下，如果fill为空串且string长度小于length时，返回空串，如果length为负数则返回null。

TD 模式

```
SELECT lpad('a',7,'');
```

```
 lpad
------
 a
(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------


(1 row)
```

```
SELECT rpad('a',7,'');
```


版权所有 © 华为云计算技术有限公司

```
 rpad
------
 a
(1 row)
```

```
SELECT rpad('a',-7,'');
```

```
 rpad
------

(1 row)
```

ORA 模式

ORA模式下，返回null。 rpad示例类似，不再赘述。

```
SELECT lpad('a',7,'');
```

```
 lpad
------

(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------

(1 row)
```

MySQL 模式

```
SELECT lpad('a',7,'');
```

```
 lpad
------

(1 row)
```

```
SELECT lpad('a',-7,'');
```

```
 lpad
------

(1 row)
```


版权所有 © 华为云计算技术有限公司

substr 返回子串

substr(str, s[, n])函数在TD和ORA模式下，s = 0时，返回前n个字符。在MySQL模式下，返回空串。

TD 和 ORA 模式

```
SELECT SUBSTR('Hello, World!', 0, 5);
```

```
 substr
--------
 Hello
(1 row)
```

MySQL 模式

```
SELECT SUBSTR('Hello, World!', 0, 5);
```

```
 substr
--------

(1 row)
```

substring 返回子串

substring(str [from s] [for n])函数在TD和ORA模式下：

- s = 0时，返回前n - 1个字符。
- s < 0时，返回前s + n - 1个字符。
- n < 0时，报错。

MySQL模式下：

- s = 0时，返回空串。
- s < 0时，倒数第|s|个字符位置开始截取n个字符。
- n < 0时，返回空串。

TD 和 ORA 模式

示例1：s=0，n=5时返回'Hello, World!'前4个字符。

```
SELECT SUBSTRING('Hello, World!' FROM 0 FOR 5);
```

```
 substring
-----------
 Hell
(1 row)
```

示例2：s=-1，n=4时返回'Hello, World!'前两个字符。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR 4);
```


版权所有 © 华为云计算技术有限公司

```
 substring
-----------
 He
(1 row)
```

示例3：n=-1时直接报错。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR -1);
```

```
ERROR:  negative substring length not allowed
CONTEXT:  referenced column: substring
```

MySQL 模式

示例1：s=0，n=5时返回空串。

```
SELECT SUBSTRING('Hello, World!' FROM 0 FOR 5);
```

```
 substring
-----------

(1 row)
```

示例2：s=-1，n=4时返回最后一个字符。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR 4);
```

```
 substring
-----------
 !
(1 row)
```

示例3：n=-1时返回空串。

```
SELECT SUBSTRING('Hello, World!' FROM -1 FOR -1);
```

```
 substring
-----------

(1 row)
```

btrim(string[,characters])

- TD和ORA模式下，从字符串string的指定位置删除只包含characters中字符（缺省为空格）的最长的字符串。
- MySQL模式下，从字符串string的指定位置删除等于characters的字符串（缺省为空格）。

TD 和 ORA 模式

```
SELECT BTRIM('xxHello Worldxx', 'xxz');
```


版权所有 © 华为云计算技术有限公司

```
    btrim
-------------
 Hello World
(1 row)
```

MySQL 模式

```
SELECT BTRIM('xxHello Worldxx', 'xxz');
```

```
      btrim
-----------------
 xxHello Worldxx
(1 row)
```

log(x)对数运算

- TD和ORA模式下log(x)表示以10为底的对数，例如log(100)等同于log(10,100)。
- MySQL模式下表示自然对数，即以e为底（e≈*2.71828*）的对数，例如log(100)约等同于log(2.71828,100)。

TD 和 ORA 模式

```
SELECT log(100);
```

```
 log
-----
   2
(1 row)
```

MySQL 模式

```
SELECT log(100);
```

```
       log
------------------
 4.60517018598809
(1 row)
```

mod(x,0)取模运算

TD和ORA模式下mod(x)除数为0时返回x。MySQL模式下报错。

TD 和 ORA 模式

```
SELECT mod(3,0);
```

```
 mod
-----
   3
(1 row)
```


版权所有 © 华为云计算技术有限公司

MySQL 模式

```
SELECT mod(3,0);
```

```
ERROR:  division by zero
CONTEXT:  referenced column: mod
```

to_char(date)

- TD模式下，入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为date，且格式为'YYYY/MM/DD'(GUC参数behavior_compat_options配置成convert_empty_str_to_null_td)。
- ORA模式下，入参最大值仅支持timestamp类型的最大值，不支持date类型的最大值；返回值类型为timestamp。
- MySQL模式下，入参最大值支持timestamp类型的最大值和date类型的最大值；返回值类型为date。

TD 模式

```
SELECT TO_CHAR(DATE '294276-12-31');
```

```
   to_char
--------------
 294276/12/31
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
ERROR:  date out of range for timestamp
CONTEXT:  referenced column: to_char
```

ORA 模式

```
SELECT TO_CHAR(DATE '294276-12-31');
```

```
        to_char
-----------------------
 294276-12-31 00:00:00
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
ERROR:  timestamp out of range: "5874897-12-31"
LINE 1: SELECT TO_CHAR(DATE '5874897-12-31');
                            ^
CONTEXT:  referenced column: to_char
```

MySQL 模式

```
SELECT TO_CHAR(DATE '294276-12-31');
```


版权所有 © 华为云计算技术有限公司

```
   to_char
--------------
 294276-12-31
(1 row)
```

```
SELECT TO_CHAR(DATE '5874897-12-31');
```

```
    to_char
---------------
 5874897-12-31
(1 row)
```

to_date, to_timestamp 和 to_number 空串处理

- TD兼容模式下，GUC参数**behavior_compat_options**配置成convert_empty_str_to_null_td参数，这些函数都返回null。
- ORA模式下，都返回null。
- MySQL模式下，to_date和to_timestamp返回null，to_number中参数为空串时，返回0。

TD 和 ORA 模式

```
SET behavior_compat_options = 'convert_empty_str_to_null_td'; --仅对TD模式生效。
SELECT TO_DATE('');
SELECT TO_TIMESTAMP('');
SELECT TO_NUMBER('');
```

```
 to_number
-----------


(1 row)
```

MySQL 模式

```
SELECT TO_DATE('');
SELECT TO_TIMESTAMP('');
SELECT TO_NUMBER('');
```

```
 to_date
---------


(1 row)
```

```
 to_timestamp
--------------


(1 row)
```


版权所有 © 华为云计算技术有限公司

```
 to_number
-----------
         0
(1 row)
```

last_day 和 next_day

- TD和ORA模式下，返回均为timestamp类型。
- MySQL模式下，返回date类型。

TD 和 ORA 模式

```
SELECT last_day(to_date('2024-02-07', 'YYYY-MM-DD')) AS cal_result;
```

```
      cal_result
---------------------
 2024-02-29 00:00:00
(1 row)
```

```
SELECT next_day(TIMESTAMP '2024-02-07 00:00:00','Sunday')AS cal_result;
```

```
      cal_result
---------------------
 2024-02-11 00:00:00
(1 row)
```

MySQL 模式

```
SELECT last_day(to_date('2024-02-07', 'YYYY-MM-DD')) AS cal_result;
```

```
 cal_result
------------
 2024-02-29
(1 row)
```

```
SELECT next_day(TIMESTAMP '2024-02-07 00:00:00','Sunday')AS cal_result;
```

```
 cal_result
------------
 2024-02-11
(1 row)
```

add_months 时间加月

TD和ORA模式下，返回均为timestamp类型加月。

MySQL模式下：

- 入参为date类型，返回date类型。

版权所有 © 华为云计算技术有限公司

- 入参为timestamp类型，返回timestamp类型。
- 入参为timestamptz类型，返回timestamptz类型。

TD 和 ORA 模式

```
SELECT add_months('2024-02-07'::date,3);
```

```
     add_months
---------------------
 2024-05-07 00:00:00
(1 row)
```

MySQL 模式

```
SELECT add_months('2024-02-07'::date,3);
```

```
 add_months
------------
 2024-05-07
(1 row)
```

```
SELECT add_months('2024-02-07 00:00:00',3);
```

```
        add_months
-------------------------
 2024-05-07 00:00:00+08
(1 row)
```

操作符'^'

- TD和ORA模式下，表示幂运算。
- MySQL模式下，表示异或。

TD 和 ORA 模式

```
SELECT 10^3;
```

```
 ?column?
----------
     1000
(1 row)
```

MySQL 模式

```
SELECT 10^3;
```

```
 ?column?
----------
        9
(1 row)
```


版权所有 © 华为云计算技术有限公司

表达式 greatest、least

- TD和ORA模式下，返回所有非null入参的比较结果。
- MySQL模式下，入参中存在null时，返回结果为null。

TD 和 ORA 模式

```
SELECT greatest(1,2,3),least(1,2,3),greatest(1,null,3),least(1,null,3);
```

```
 greatest | least | greatest | least
----------+-------+----------+-------
        3 |     1 |        3 |     1
(1 row)
```

MySQL 模式

```
SELECT greatest(1,2,3),least(1,2,3),greatest(1,null,3),least(1,null,3);
```

```
 greatest | least | greatest | least
----------+-------+----------+-------
        3 |     1 |          |
(1 row)
```

case 和 coalesce 表达式

- TD模式：兼容TD行为，支持数字和字符串之间的类型转换，比如coalesce参数输入int和varchar类型，解析成varchar类型。
- ORA模式：报错。
- MySQL模式：兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。

TD 模式

对于case和coalesce，在TD兼容模式下的处理：

- 如果所有输入都是相同的类型，并且不是unknown类型，那么解析成这种类型。
- 如果所有输入都是unknown类型则解析成text类型。
- 如果输入字符串（包括unknown，unknown当text来处理）和数字类型，那么解析成字符串类型，如果是其他不同的类型范畴，则报错。
- 如果输入类型是同一个类型范畴，则选择该类型的优先级较高的类型。
- 把所有输入转换为所选的类型。如果从给定的输入到所选的类型没有隐式转换则失败。

TD模式下，coalesce参数输入int和varchar类型，会解析为varchar类型。查看coalesce参数输入int和varchar类型的查询语句的执行计划如下。

```
EXPLAIN VERBOSE select coalesce(a, b) FROM td_table;
```


版权所有 © 华为云计算技术有限公司

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 id |                 operation                  | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on "__REMOTE_FQS_QUERY__" |      0 |            |       0 | 0.00

                        Targetlist Information (identified by plan id)
-----------------------------------------------------------------------------------------------------------
  1 --Data Node Scan on "__REMOTE_FQS_QUERY__"
        Output: (COALESCE((td_table.a)::character varying, td_table.b))
        Node/s: All datanodes (v3_logical, bucket:24)
        Remote query: SELECT COALESCE(a::character varying, b) AS "coalesce" FROM public.td_table
```

ORA 模式

ORA模式下，coalesce参数输入int和varchar类型，系统会报错。查看coalesce参数输入int和varchar类型的查询语句的执行计划如下。

```
EXPLAIN VERBOSE select coalesce(a, b) FROM ora_table;
```

```
ERROR:  COALESCE types integer and character varying cannot be matched
LINE 1: EXPLAIN VERBOSE select coalesce(a, b) FROM ora_table;
                                                ^
CONTEXT:  referenced column: coalesce
```

MySQL 模式

兼容MySQL行为，支持其他类型和字符串之间的类型转换，比如coalesce参数输入date、int和varchar类型，解析成varchar类型。

版权所有 © 华为云计算技术有限公司

5 DWS 数据库安全管理

5.1 DWS 用户及权限管理

5.1.1 DWS 数据库用户类型

在非三权分立模式下，DWS支持管理员用户和普通用户两种类型的数据库账号。三权分立下，用户类型和权限请参考**DWS三权分立**。

- 管理员用户可以管理所有普通用户和数据库。
- 普通用户可连接和访问数据库，并在授权后进行特定的数据库操作以及执行SQL语句。

在用户登录DWS数据库时会对其进行身份验证。用户可以拥有数据库和数据库对象（例如表），并且可以向用户和角色授予对这些对象的权限以控制谁可以访问哪个对象。除管理员外，具有CREATEDB属性的用户可以创建数据库并授予对这些数据库的权限。

数据库用户类型

表 5-1 数据库用户类型

用户类型	描述	可进行的操作	如何创建
管理员dbadmin	管理员也称作系统管理员，是指具有SYSADMIN属性的账户。	非三权分立模式下，拥有系统的最高权限，能够执行所有的操作。系统管理员具有与对象所有者相同的权限。	• 在DWS 管理控制台创建集群时创建的用户dbadmin是系统管理员。 • 使用CREATE USER或ALTER USER语法创建和设置管理员用户。 CREATE USER *sysadmin* WITH SYSADMIN password '*{Password}*'; ALTER USER *u1* SYSADMIN;

 版权所有 © 华为云计算技术有限公司

用户类型	描述	可进行的操作	如何创建
普通用户	普通用户	• 使用工具连接数据库。 • 拥有数据库系统特定操作的属性，如CREATEDB、CREATEROLE、SYSADMIN。 • 访问数据库对象。 • 执行SQL语句。	使用CREATE USER语法创建普通用户。 CREATE USER *u1* PASSWORD '*{Password}*';
	私有用户	在非三权分立模式下，创建的具有INDEPENDENT属性的私有用户。 数据库管理员在未经其授权前，只能进行控制操作（DROP、ALTER、TRUNCATE），无权进行INSERT、DELETE、SELECT、UPDATE、COPY、GRANT、REVOKE、ALTER OWNER操作。	使用CREATE USER语法创建私有用户。 CREATE USER *user_independent* WITH INDEPENDENT IDENTIFIED BY '*{Password}*';

5.1.2 DWS 数据库用户管理

使用CREATE USER和ALTER USER可以创建和管理数据库用户。

- 非**三权分立**下，DWS用户账户只能由系统管理员或拥有CREATEROLE属性的安全管理员创建和删除。
- 三权分立时，用户账户只能由安全管理员创建。

创建用户

CREATE USER语句用于创建新的DWS用户。创建新用户后，可以使用该用户连接数据库。

- 创建普通用户u1，并设置用户拥有CREATEDB属性。
  ```
  CREATE USER u1 WITH CREATEDB PASSWORD '{Password}';
  ```
- 创建系统管理员mydbadmin，需指定参数SYSADMIN。
  ```
  CREATE USER mydbadmin sysadmin PASSWORD '{Password}';
  ```
- 通过视图**PG_USER**查看已创建的用户。
  ```
  SELECT * FROM pg_user;
  ```
- 要查看用户属性，请查询系统表**PG_AUTHID**。
  ```
  SELECT * FROM pg_authid;
  ```

修改用户属性

ALTER USER语句用于更改用户属性（例如，更改用户密码或权限等内容）。

版权所有 © 华为云计算技术有限公司

示例：

- 用户u1重命名为u2：
  ```
  ALTER USER u1 RENAME TO u2;
  ```
- 为用户u1赋予CREATEROLE权限：
  ```
  ALTER USER u1 CREATEROLE;
  ```
- 修改用户密码可参考**密码设置和修改**。

锁定用户

ALTER USER语句中ACCOUNT LOCK | ACCOUNT UNLOCK参数用于锁定或者解锁用户，被锁定的用户不允许登录。若管理员发现某账户被盗、非法访问等异常情况，可手动锁定该账户；当管理员认为账户恢复正常后，可手动解锁该账户。

示例：

- 锁定用户u1：
  ```
  ALTER USER u1 ACCOUNT LOCK;
  ```
- 解锁用户u1：
  ```
  ALTER USER u1 ACCOUNT UNLOCK;
  ```

删除用户

DROP USER语句用于删除一个或多个DWS用户。当确认账户不再使用，管理员可以删除用户账户。用户删除后不可恢复。

- 同时删除多个用户时，用","隔开。
- 成功删除用户后，该用户的所有权限也会被一同删除。
- 当删除的用户正处于活动状态时，此会话状态不会立即断开，用户在会话状态断开后才会被完全删除。
- DROP USER语句指定CASCADE时，可级联删除依赖用户的表等对象。即删除owner是该用户的对象，并清理掉其他对象对该用户的授权信息。

示例：

- 删除用户u1：
  ```
  DROP USER u1;
  ```
- 级联删除账户u2：
  ```
  DROP USER u2 CASCADE;
  ```

5.1.3 自定义 DWS 密码策略

创建用户或修改用户时需要指定密码，DWS有默认的密码复杂度要求，也支持用户自定义数据库账户密码策略。

DWS 默认密码策略

DWS默认进行密码复杂度校验，默认密码策略要求如下：

- 长度为8~32个字符。
- 至少包含大写字母、小写字母、数字或特殊字符中三种的组合。支持的特殊字符参见**表5-2**。
- 不能是用户名和用户名反序，此条要求为非大小写敏感。

版权所有 © 华为云计算技术有限公司

- 不能与当前密码或当前密码的反序相同。

表 5-2 支持的特殊字符

编号	字符	编号	字符	编号	字符	编号	字符
1	~	9	*	17	|	25	<
2	!	10	(	18	[	26	.
3	@	11	)	19	{	27	>
4	#	12	-	20	}	28	/
5	$	13	_	21	]	29	?
6	%	14	=	22	;	-	-
7	^	15	+	23	:	-	-
8	&	16	\	24	,	-	-

密码设置和修改

- 建议系统管理员和普通用户都要定期修改自己的账户密码，避免账户密码被非法窃取。

  以修改用户u1密码为例，使用管理员用户连接数据库并执行如下命令：

  ```
  ALTER USER u1 IDENTIFIED BY 'newpassword' REPLACE 'oldpassword';
  ```

  说明

  密码要符合**DWS默认密码策略**，否则会执行失败。

- 管理员可以修改自己的或者其他账户的密码。通过修改其他账户的密码，解决用户密码遗失所造成无法登录的问题。

  以修改用户joe账户密码为例，命令格式如下：

  ```
  ALTER USER joe IDENTIFIED BY 'password';
  ```

  说明

  - 系统管理员之间不允许互相修改对方密码。
  - 系统管理员可以修改普通用户密码且不需要用户原密码。
  - 系统管理员可以修改自己的密码但需要管理员原密码。

- 密码验证

  设置当前会话的用户和角色时，需要验证密码。如果输入密码与用户的存储密码不一致，则会报错。

  以设置用户joe为例，命令格式如下：

  ```
  SET ROLE joe PASSWORD 'password';
  ```

  显示如下命令表示设置成功：

  ```
  SET ROLE
  ```


版权所有 © 华为云计算技术有限公司

5.1.4 DWS 数据库权限管理

权限概述

权限表示用户访问某个数据库对象（包括模式、表、函数、序列等）的操作（包括增、删、改、查、创建等）是否被允许。

DWS中的权限管理分为三种场景：

- 系统权限

  系统权限又称为用户属性，包括SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN和LOGIN。

  系统权限一般通过CREATE/ALTER ROLE语法来指定。其中，SYSADMIN权限可以通过GRANT/REVOKE ALL PRIVILEGE授予或撤销。但系统权限无法通过ROLE和USER的权限被继承，也无法授予PUBLIC。

- 数据对象权限

  将数据库对象（表和视图、指定字段、数据库、函数、模式等）的相关权限授予特定角色或用户。GRANT命令将数据库对象的特定权限授予一个或多个角色。这些权限会追加到已有的权限上。

- 用户权限

  将一个角色或用户的权限授予一个或多个其他角色或用户。在这种情况下，每个角色或用户都可视为拥有一个或多个数据库权限的集合。

  当声明了WITH ADMIN OPTION，被授权的用户可以将该权限再次授予其他角色或用户，以及撤销所有由该角色或用户继承到的权限。当授权的角色或用户发生变更或被撤销时，所有继承该角色或用户权限的用户拥有的权限都会随之发生变更。

  数据库系统管理员可以给任何角色或用户授予/撤销任何权限。拥有CREATEROLE权限的角色可以赋予或者撤销任何非系统管理员角色的权限。

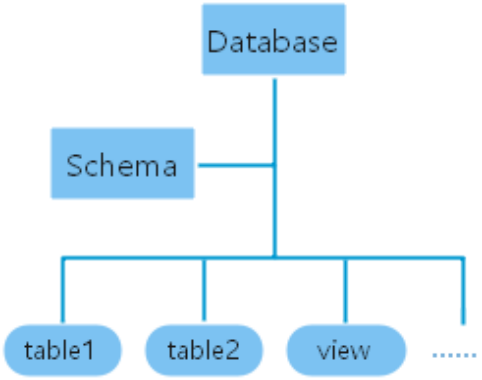
层级权限管理

DWS通过Database、Schema和数据对象权限实现层级权限管理。

- Database之间无法直接互访，通过连接隔离实现彻底的权限隔离。各个Database之间共享资源极少，可实现连接隔离、权限隔离等。数据库集群包含一个或多个已命名数据库。用户和角色在整个集群范围内是共享的，但是其数据并不共享。即用户可以连接任何数据库，但当连接成功后，任何用户都只能访问连接请求里所声明的数据库。
- Schema隔离的方式共用资源较多，可以通过GRANT与REVOKE语法便捷地控制不同用户对各Schema及其下属对象的权限，从而赋予业务更多的灵活性。每个数据库包括一个或多个Schema。每个Schema包含表、函数等其他类型的对象。用户要访问包含在指定Schema中的对象，需要被授予Schema的USAGE权限。
- 对象创建后，默认只有对象所有者或者系统管理员可以查询、修改和删除对象。其他用户要访问包含具体的数据库对象，例如table1，需要首先被授予database的CONNECT权限，再被授予Schema的USAGE权限，最后授予table1的SELECT权限。用户要访问底层的对象，必须先赋予上层对象的权限。比如用户要创建或者删除Schema，需要首先被授予database的CREATE权限；

版权所有 © 华为云计算技术有限公司

图 5-1 层级权限管理



角色

DWS的权限管理模型，是一种典型的RBAC（基于角色的权限控制）的实现。其将用户、角色、权限通过此模型管理起来。

角色是一组权限的集合。

- “用户”概念和“角色”概念实际是等同的，唯一的区别在于“用户”拥有login权限，而“角色”拥有nologin权限。
- 按照数据库系统中承担的责任划分具有不同权限的角色。角色是数据库权限的集合，代表了一个数据库用户或一组数据库用户的行为约束。
- 角色和用户可以转换，通过ALTER语句使角色拥有登录权限。
- 通过GRANT语句把角色授予用户后，用户即具有了角色的所有权限。推荐使用角色进行高效权限分配。例如，可以为设计、开发和维护人员创建不同的角色，将角色GRANT给用户后，再向每个角色中的用户授予其所需数据的差异权限。在角色级别授予或撤销权限时，这些权限更改会对角色下的所有成员生效。
- 非三权分立时，只有系统管理员和具有CREATEROLE属性的用户才能创建、修改或删除角色。三权分立下，只有具有CREATEROLE属性的用户才能创建、修改或删除角色。

要查看所有角色，请查询系统表PG_ROLES：

```
SELECT * FROM PG_ROLES;
```

使用CREATE ROLE、ALTER ROLE、DROP ROLE进行创建、修改和删除角色操作。

预置角色

DWS提供了一组预置角色，以“gs_role_”开头命名，提供对特定的、通常需要高权限的操作的访问，可以将这些角色授权予数据库中的其他用户或角色，使这些用户能够访问或使用特定的信息和功能。请谨慎使用预置角色，以确保预置角色权限的安全使用。

预置角色允许的权限范围可参考下表：

版权所有 © 华为云计算技术有限公司

表 5-3 预置角色允许的权限范围

角色	权限描述
gs_role_signal_backend	具有调用函数pg_cancel_backend()、pg_terminate_backend()、pg_terminate_query()、pg_cancel_query()、pgxc_terminate_query()、pgxc_cancel_query()来取消或终止其他会话的权限，但不能操作属于初始用户的会话。
gs_role_read_all_stats	读取系统状态视图并且使用与扩展相关的各种统计信息，包括有些通常只对系统管理员可见的信息。包括： 资源管理类： ● pgxc_wlm_operator_history ● pgxc_wlm_operator_info ● pgxc_wlm_operator_statistics ● pgxc_wlm_session_info ● pgxc_wlm_session_statistics ● pgxc_wlm_workload_records ● pgxc_workload_sql_count ● pgxc_workload_sql_elapse_time ● pgxc_workload_transaction 状态信息类： ● pgxc_stat_activity ● pgxc_get_table_skewness ● table_distribution ● pgxc_total_memory_detail ● pgxc_os_run_info ● pg_nodes_memory ● pgxc_instance_time ● pgxc_redo_stat 表存储分布信息函数： ● table_distribution() ● table_distribution(regclass) ● table_distribution(schemaname text, tablename text) ● gs_table_distribution() ● gs_table_distribution(schemaname text, tablename text)
gs_role_analyze_any	具有系统级ANALYZE权限类似系统管理员用户，跳过schema权限检查，对所有的表可以执行ANALYZE。
gs_role_vacuum_any	具有系统级VACUUM权限类似系统管理员用户，跳过schema权限检查，对所有的表可以执行VACUUM。

版权所有 © 华为云计算技术有限公司

角色	权限描述
gs_role_sql_management	具有执行CREATE/DROP OUTLINE以及调用plan management管理函数pgxc_bind_plan()、pgxc_unbind_plan()、pgxc_clean_plan()、pgxc_create_auto_clean_plan_task()、pgxc_alter_auto_clean_plan_task()、pgxc_drop_auto_clean_plan_task()、drop_outline()、pgxc_reset_planmgmt_hashtable()的权限。
gs_role_redaction	具有创建、修改、删除脱敏策略的权限，对所有的表都可以执行CREATE/ALTER/DROP REDACTION POLICY。9.1.0.100及以上集群版本支持。
gs_role_block	具有执行CREATE/ALTER/DROP BLOCK RULE的权限。9.1.0.100及以上集群版本支持。

预置角色的使用约束：

- 以gs_role_开头的角色名作为数据库的预置角色保留字，禁止新建以“gs_role_”开头的用户/角色，也禁止将已有的用户/角色重命名为以“gs_role_”开头。
- 禁止对预置角色执行ALTER和DROP操作。
- 预置角色默认没有LOGIN权限，不设置预置登录密码。
- gsql元命令\du和\dg不显示预置角色的相关信息，但若指定了PATTERN（用来指定要被显示的对象名称）则预置角色信息会显示。
- 三权分立关闭时，系统管理员和具有预置角色ADMIN OPTION权限的用户有权对预置角色执行GRANT/REVOKE管理；三权分立打开时，安全管理员（具有CREATEROLE属性）和具有预置角色ADMIN OPTION权限的用户有权对预置角色执行GRANT/REVOKE管理。例如：

  ```
  GRANT gs_role_signal_backend TO user1;
  REVOKE gs_role_signal_backend FROM user1;
  ```

权限授予或撤销

数据库对象创建后，进行对象创建的用户就是该对象的所有者。集群安装后的默认情况下，未开启**三权分立**，数据库系统管理员具有与对象所有者相同的权限。

也就是说对象创建后，默认只有对象所有者或者系统管理员可以查询、修改和删除对象，以及通过GRANT将对象的权限授予其他用户。为使其他用户能够使用对象，可以由对象所有者或管理员通过GRANT/REVOKE对其他用户或角色授予与撤销。

- 使用GRANT语句授予权限。

  例如，将模式myschema的权限赋予角色u1后，将表myschema.t1的SELECT权限授予角色u1。

  ```
  GRANT USAGE ON SCHEMA myschema TO u1;
  GRANT SELECT ON TABLE myschema.t1 to u1;
  ```

- 使用REVOKE撤销已经授予的权限。

  例如：撤销用户u1在指定表myschema.t1上的所有权限。

  ```
  REVOKE ALL PRIVILEGES ON myschema.t1 FROM u1;
  ```


版权所有 © 华为云计算技术有限公司

5.1.5 DWS 三权分立

默认情况下拥有SYSADMIN属性的系统管理员，具备系统最高权限。在实际业务管理中，为了避免系统管理员拥有过度集中的权限带来高风险，可以设置三权分立，将系统管理员的权限分立给安全管理员和审计管理员。

- 三权分立后，系统管理员将不再具有CREATEROLE属性（安全管理员）和AUDITADMIN属性（审计管理员）能力。即不再拥有创建角色和用户的权限，并不再拥有查看和维护数据库审计日志的权限。关于CREATEROLE属性和AUDITADMIN属性的更多信息请参考《数据仓库服务SQL语法参考》中“CREATE ROLE”章节。
- 三权分立后，系统管理员只会对自己作为所有者的对象有权限。

三权分立的设置方法，请参考**设置DWS集群三权分立**章节。

三权分立前的权限详情及三权分立后的权限变化，请分别参见**表5-4**和**表5-5**。

表 5-4 默认的用户权限

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	对表空间有创建、修改、删除、访问、分配操作的权限。	不具有对表空间进行创建、修改、删除、分配的权限，访问需要被赋权。		
表	对所有表有所有的权限。	仅对自己的表有所有的权限，对其他用户的表无权限。		
索引	可以在所有的表上建立索引。	仅可以在自己的表上建立索引。		
模式	对所有模式有所有的权限。	仅对自己的模式有所有的权限，对其他用户的模式无权限。		
函数	对所有的函数有所有的权限。	仅对自己的函数有所有的权限，对其他用户放在public这个公共模式下的函数有调用的权限，对其他用户放在其他模式下的函数无权限。		
自定义视图	对所有的视图有所有的权限。	仅对自己的视图有所有的权限，对其他用户的视图无权限。		
系统表和系统视图	可以查看所有系统表和视图。	只可以查看部分系统表和视图。详细请参见**DWS系统表和系统视图**。		

版权所有 © 华为云计算技术有限公司

表 5-5 三权分立较非三权分立权限变化说明

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	无变化	无变化。		
表	权限缩小。 只对自己的表有所有权限，对其他用户放在属于各自模式下的表无权限。	无变化。		
索引	权限缩小。 只可以在自己的表上建立索引。	无变化。		
模式	权限缩小。 只对自己的模式有所有的权限，对其他用户的模式无权限。	无变化。		
函数	权限缩小。 只对自己的函数有所有的权限，对其他用户放在属于各自模式下的函数无权限。	无变化。		
自定义视图	权限缩小。 只对自己的视图及其他用户放在public模式下的视图有所有的权限，对其他用户放在属于各自模式下的视图无权限。	无变化。		
系统表和系统视图	无变化。	无变化。		

5.2 DWS 敏感数据管理

5.2.1 DWS 行级访问控制

行级访问控制特性将数据库访问控制精确到数据表行级别，使数据库达到行级访问控制的能力。不同用户执行相同的SQL查询操作，读取到的结果是不同的。

用户可以在数据表创建行访问控制(Row Level Security)策略，该策略是指针对特定数据库用户、特定SQL操作生效的表达式。当数据库用户对数据表访问时，若SQL满足数据表特定的Row Level Security策略，在查询优化阶段将满足条件的表达式，按照属性(PERMISSIVE | RESTRICTIVE)类型，通过AND或OR方式拼接，应用到执行计划上。

行级访问控制的目的是控制表中行级数据可见性，通过在数据表上预定义Filter，在查询优化阶段将满足条件的表达式应用到执行计划上，影响最终的执行结果。当前受影响的SQL语句包括SELECT、UPDATE和DELETE。

版权所有 © 华为云计算技术有限公司

示例

某表中汇总了不同用户的数据，但是不同用户只能查看自身相关的数据信息，不能查看其他用户的数据信息。

步骤1 创建用户alice、bob及peter。

```
CREATE ROLE alice PASSWORD 'password';
CREATE ROLE bob PASSWORD 'password';
CREATE ROLE peter PASSWORD 'password';
```

步骤2 创建表public.all_data，包含不同用户数据信息。

```
CREATE TABLE public.all_data(id int, role varchar(100), data varchar(100));
```

步骤3 向数据表插入数据。

```
INSERT INTO all_data VALUES(1, 'alice', 'alice data');
INSERT INTO all_data VALUES(2, 'bob', 'bob data');
INSERT INTO all_data VALUES(3, 'peter', 'peter data');
```

步骤4 将表all_data的读取权限赋予用户alice，bob和peter。

```
GRANT SELECT ON all_data TO alice, bob, peter;
```

步骤5 打开行访问控制策略开关。

```
ALTER TABLE all_data ENABLE ROW LEVEL SECURITY;
```

步骤6 创建行访问控制策略，当前用户只能查看用户自身的数据。

```
CREATE ROW LEVEL SECURITY POLICY all_data_rls ON all_data USING(role = CURRENT_USER);
```

步骤7 查看表详细信息。

```
\d+ all_data
                            Table "public.all_data"
 Column |          Type          | Modifiers | Storage  | Stats target | Description
--------+------------------------+-----------+----------+--------------+-------------
 id     | integer                |           | plain    |              |
 role   | character varying(100) |           | extended |              |
 data   | character varying(100) |           | extended |              |
Row Level Security Policies:
    POLICY "all_data_rls"
      USING (((role)::name = "current_user"()))
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, enable_rowsecurity=true
```

步骤8 切换至用户alice，查询表及分析查询计划。

```
SET ROLE alice PASSWORD 'password';
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  1 | alice | alice data
(1 row)

EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                        QUERY PLAN
-----------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'alice'::name)
 Notice: This query is influenced by row level security feature
(5 rows)
```

步骤9 切换至用户peter，查询表及分析查询计划。

```
SET ROLE peter PASSWORD 'password';
SELECT * FROM all_data;
 id | role  |    data
```


版权所有 © 华为云计算技术有限公司

```
----+-------+------------
  3 | peter | peter data
(1 row)

 EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                      QUERY PLAN
-------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'peter'::name)
 Notice: This query is influenced by row level security feature
(5 rows)
```

----结束

5.2.2 DWS 数据脱敏

DWS提供列级别的数据脱敏(Dynamic Data Masking)功能。针对某些敏感信息（如身份证号、手机号、银行卡号等），通过应用脱敏函数进行原始数据的变形改写，实现敏感隐私数据的可靠保护，从而增强产品在数据安全和隐私保护方面的能力。

- 创建表的数据脱敏策略

  DWS通过使用**CREATE REDACTION POLICY**语法在表上创建数据脱敏策略并使用函数（MASK_NONE，不进行任何脱敏处理；MASK_FULL，全脱敏成固定值；MASK_PARTIAL，按指定的字符类型，数值类型或时间类型进行部分脱敏处理。）限定脱敏策略生效范围。

- 修改表的数据脱敏策略

  ALTER REDACTION POLICY语法用于修改脱敏策略生效表达式、重命名脱敏策略，以及新增、修改、删除脱敏列信息。

- 删除表的脱敏策略

  DROP REDACTION POLICY语法用于删除脱敏策略在表的所有列字段上的脱敏函数信息。

- 查看脱敏策略和脱敏列信息

  脱敏策略信息存储在系统表**PG_REDACTION_POLICY**，脱敏列信息存储在系统表**PG_REDACTION_COLUMN**。用户也可以通过系统视图**REDACTION_POLICIES**和**REDACTION_COLUMNS**，更加方便地查看脱敏策略及脱敏列信息。

注意事项

- 数据脱敏特性使用了参数进行控制，当出现“REDACTION POLICY is not yet supported, please add enable_data_redaction into feature_support_options.“或者“Cannot use the feature of data redaction, please drop existed redaction policy or add enable_data_redaction into feature_support_options.“的报错信息时，说明没有开启数据脱敏特性，此时需要将GUC参数feature_support_options加上enable_data_redaction的取值，需联系技术支持人员设置feature_support_options参数，开启数据脱敏特性。
- 通常，用户可以执行SELECT语句查看敏感信息的脱敏效果。如果语句有如下特征，则可能存在故意套取敏感数据的可能性，造成语句执行报错。
  - GROUP BY子句引用与目标列一样含有脱敏列的Target Entry作为分组。
  - DISTINCT作用在输出的脱敏列上。
  - 带有CTE的语句。
  - 涉及集合操作。

版权所有 © 华为云计算技术有限公司

  - 子查询的目标列不是基表的脱敏列，而是基表脱敏列的表达式或者函数调用。
- 支持脱敏数据的COPY TO或者GDS导出功能。由于脱敏数据的不可逆性，针对脱敏数据的二次运算无任何实际意义。
- 禁止UPDATE、MERGE INTO、DELETE语句的目标列涉及脱敏列。
- UPSERT语句允许通过EXCLUDED更新插入数据。如果引用脱敏列更新基表数据，存在误改数据的可能，执行会报错。
- 自8.2.1集群版本针对同一张表可以允许创建多个脱敏策略，实现敏感数据分级分类场景下的多样化配置。多脱敏策略的选择和应用原则如下：
  - 选择满足当前Session的多个候选策略中policy_order最大的那一个策略，即策略次序越大表示越晚创建。
  - 脱敏可算场景下，DML语句发生策略的继承，也仅继承源表相关的候选策略中policy_order最大的那一个策略。

示例

以员工表emp，表的属主alice及角色matu、july为例，简要介绍数据脱敏过程。其中，表emp包含员工的姓名、手机号、邮箱、银行卡号、薪资等隐私数据

步骤1 使用管理员用户连接数据库后，创建角色alice、matu和july。

```
CREATE ROLE alice PASSWORD 'password';
CREATE ROLE matu PASSWORD 'password';
CREATE ROLE july PASSWORD 'password';
```

步骤2 赋予alice、matu和july当前数据库的模式权限。

```
GRANT ALL PRIVILEGES on schema public to alice,matu,july;
```

步骤3 切换至角色alice，创建表emp并插入三条员工信息。

```
SET ROLE alice PASSWORD 'password';

CREATE TABLE emp(id int, name varchar(20), phone_no varchar(11), card_no number, card_string varchar(19), email text, salary numeric(100, 4), birthday date);

INSERT INTO emp VALUES(1, 'anny', '13420002340', 1234123412341234, '1234-1234-1234-1234', 'smithWu@163.com', 10000.00, '1999-10-02');
INSERT INTO emp VALUES(2, 'bob', '18299023211', 3456345634563456, '3456-3456-3456-3456', '66allen_mm@qq.com', 9999.99, '1989-12-12');
INSERT INTO emp VALUES(3, 'cici', '15512231233', NULL, NULL, 'jonesishere@sina.com', NULL, '1992-11-06');
```

步骤4 alice将表emp的读取权限授予matu、july。

```
GRANT SELECT ON emp TO matu, july;
```

步骤5 创建脱敏策略mask_emp，仅alice可查看员工所有信息，matu和july对员工银行卡号和薪资数据不可见。字段card_no是数值类型，采用MASK_FULL全脱敏成固定值0；字段card_string是字符类型，采用MASK_PARTIAL按指定的输入输出格式对原始数据进行部分脱敏；字段salary是数值类型，采用MASK_PARTIAL指定数字9部分脱敏倒数第二位前的所有数位值。

```
CREATE REDACTION POLICY mask_emp ON emp WHEN (current_user IN ('matu', 'july'))
 ADD COLUMN card_no WITH mask_full(card_no),
 ADD COLUMN card_string WITH mask_partial(card_string, 'VVVVFVVVVFVVVVFVVVV','VVVV-VVVV-VVVV-VVVV','#',1,12),
 ADD COLUMN salary WITH mask_partial(salary, '9', 1, length(salary) - 2);
```

步骤6 切换到matu和july，查看员工表emp。

```
SET ROLE matu PASSWORD 'password';
SELECT * FROM emp;
```

 版权所有 © 华为云计算技术有限公司

```
 id | name |  phone_no   | card_no |     card_string     |        email         |   salary   |      birthday
----+------+-------------+---------+---------------------+----------------------+------------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com      | 99999.9990 |
1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com    |  9999.9990 |
1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |            | 1992-11-06 00:00:00
(3 rows)

SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |        email         |   salary   |      birthday
----+------+-------------+---------+---------------------+----------------------+------------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com      | 99999.9990 |
1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com    |  9999.9990 |
1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |            | 1992-11-06 00:00:00
(3 rows)
```

步骤7 若需要matu也有员工所有信息的查看权限，只有july不可见，修改策略生效范围即可。

```
SET ROLE alice PASSWORD 'password';
ALTER REDACTION POLICY mask_emp ON emp WHEN(current_user = 'july');
```

步骤8 切换到matu和july，重新查看员工表emp。

```
SET ROLE matu PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   |     card_no      |     card_string     |        email         |   salary   |      birthday
----+------+-------------+------------------+---------------------+----------------------+------------+---------------------
+---------------------
  1 | anny | 13420002340 | 1234123412341234 | 1234-1234-1234-1234 | smithWu@163.com      |
10000.0000 | 1999-10-02 00:00:00
  2 | bob  | 18299023211 | 3456345634563456 | 3456-3456-3456-3456 | 66allen_mm@qq.com    |
9999.9900 | 1989-12-12 00:00:00
  3 | cici | 15512231233 |                  |                     | jonesishere@sina.com |            | 1992-11-06 00:00:00
(3 rows)

SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |        email         |   salary   |      birthday
----+------+-------------+---------+---------------------+----------------------+------------+---------------------
  1 | anny | 13420002340 |       0 | ####-####-####-1234 | smithWu@163.com      | 99999.9990 |
1999-10-02 00:00:00
  2 | bob  | 18299023211 |       0 | ####-####-####-3456 | 66allen_mm@qq.com    |  9999.9990 |
1989-12-12 00:00:00
  3 | cici | 15512231233 |         |                     | jonesishere@sina.com |            | 1992-11-06 00:00:00
(3 rows)
```

步骤9 员工信息phone_no、email和birthday也是隐私数据，更新脱敏策略mask_emp，新增三个脱敏列。

```
SET ROLE alice PASSWORD 'password';
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN phone_no WITH mask_partial(phone_no, '*',
4);
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN email WITH mask_partial(email, '*', 1,
position('@' in email));
ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN birthday WITH mask_full(birthday);
```

步骤10 切换到july，查看表emp数据。

```
SET ROLE july PASSWORD 'password';
SELECT * FROM emp;
 id | name |  phone_no   | card_no |     card_string     |        email         |   salary   |      birthday
----+------+-------------+---------+---------------------+----------------------+------------+---------------------
  1 | anny | 134******** |       0 | ####-####-####-1234 | ********163.com      | 99999.9990 | 1970-01-01
00:00:00
  2 | bob  | 182******** |       0 | ####-####-####-3456 | **********qq.com     |  9999.9990 | 1970-01-01
00:00:00
```


版权所有 © 华为云计算技术有限公司

```
 3 | cici | 155******** |          |                    | ************sina.com |            | 1970-01-01 00:00:00
(3 rows)
```

步骤11 通过视图redaction_policies和redaction_columns查看当前脱敏策略mask_emp的详细信息。

```
SELECT * FROM redaction_policies;
 object_schema | object_owner | object_name | policy_name |            expression             | enable |
policy_description | inherited
---------------+--------------+-------------+-------------+-----------------------------------+--------
+--------------------+-----------
 public        | alice        | emp         | mask_emp    | ("current_user"() = 'july'::name) | t      |                    |
f
(1 row)

SELECT object_name, column_name, function_info FROM redaction_columns;
 object_name | column_name |                                                  function_info
-------------+-------------
+-----------------------------------------------------------------------------------------------------------
 emp         | card_no     | mask_full(card_no)
 emp         | card_string | mask_partial(card_string, 'VVVVFVVVVFVVVVFVVVV'::text, 'VVVV-VVVV-VVVV-
VVVV'::text, '#'::text, 1, 12)
 emp         | email       | mask_partial(email, '*'::text, 1, "position"(email, '@'::text))
 emp         | salary      | mask_partial(salary, '9'::text, 1, (length((salary)::text) - 2))
 emp         | birthday    | mask_full(birthday)
 emp         | phone_no    | mask_partial(phone_no, '*'::text, 4)
(6 rows)
```

步骤12 新增一列salary_info，若需要将文本类型的薪资信息统一脱敏成“*.*”，可以创建自定义脱敏函数实现。此处采用PL/pgSQL语言定义脱敏函数mask_regexp_salary，创建脱敏列时，只需自定义脱敏的函数名和参数列表，详细内容可参考**DWS用户自定义函数**。

```
SET ROLE alice PASSWORD 'password';

ALTER TABLE emp ADD COLUMN salary_info TEXT;
UPDATE emp SET salary_info = salary::text;

CREATE FUNCTION mask_regexp_salary(salary_info text) RETURNS text AS
$$
 SELECT regexp_replace($1, '[0-9]+','*','g');
$$
 LANGUAGE SQL
STRICT SHIPPABLE;

ALTER REDACTION POLICY mask_emp ON emp ADD COLUMN salary_info WITH
mask_regexp_salary(salary_info);

SET ROLE july PASSWORD 'password';
SELECT id, name, salary_info FROM emp;
 id | name | salary_info
----+------+-------------
  1 | anny | *.*
  2 | bob  | *.*
  3 | cici |
(3 rows)
```

步骤13 无需为表emp设置敏感策略，删除脱敏策略mask_emp。

```
SET ROLE alice PASSWORD 'password';
DROP REDACTION POLICY mask_emp ON emp;
```

----结束

5.2.3 DWS 字符串加解密

DWS支持使用以下函数对字符串进行加解密。

 版权所有 © 华为云计算技术有限公司

- gs_encrypt(encryptstr, keystr, cryptotype, cryptomode, hashmethod)

  描述：采用cryptotype和cryptomode组成的加密算法以及hashmethod指定的HMAC算法，以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。支持的cryptotype：aes128, aes192, aes256。支持的cryptomode：cbc, gcm。支持的hashmethod：sha256, sha384, sha512, sm3。支持的加密数据类型：目前数据库支持的数值类型，字符类型，二进制类型中的RAW，日期/时间类型中的DATE、TIMESTAMP、SMALLDATETIME。keystr的长度范围与加密算法相关，为1~KeyLen字节。当cryptotype为aes128时，KeyLen为16，aes192时KeyLen为24，aes256时KeyLen为32。

  返回值类型：text

  返回值长度：至少为 4*[(maclen + 56)/3] 字节，不超过 4*[(Len + maclen + 56)/3] 字节，其中Len为加密前数据长度（单位为字节），maclen为HMAC值的长度，当hashmethod为sha256和sm3时maclen为32，sha384时maclen为48，sha512时maclen为64。即当hashmethod为sha256和sm3时，返回值长度至少为120字节，不超过4*[(Len + 88)/3] 字节；当hashmethod为sha384时，返回值长度至少为140字节，不超过4*[(Len + 104)/3] 字节；当hashmethod为sha512时，返回值长度至少为160字节，不超过4*[(Len + 120)/3] 字节；

  示例：

  ```
  SELECT gs_encrypt('database', '1234', 'aes128', 'cbc',  'sha256');
                                                  gs_encrypt
  ----------------------------------------------------------------------------------------------------------------------------
  ---
   AAAAAAAAAAD74GDr4c23kEjr3p7F+hIDBWUBhwWr9E70a6XXWWel+lh31z+kz1y4CK
  +yOITyw3B13tm9ZTIUDm+baNjl4TDOKai+zLqoPoEWK0CWYEKgbA==
  (1 row)
  ```

  说明

  - 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
  - 在同一张数据表中，加密函数gs_encrypt与gs_encrypt_aes128不要混合使用。

- gs_decrypt(decryptstr, keystr, cryptotype, cryptomode, hashmethod)

  描述：采用cryptotype和cryptomode组成的加密算法以及hashmethod指定的HMAC算法，以keystr为密钥对decryptstr字符串进行解密，返回解密后的字符串。解密使用的keystr必须保证与加密时使用的keystr一致才能正常解密。keystr不得为空。

  返回值类型：text

  示例：

  ```
  SELECT gs_decrypt('AAAAAAAAAAD74GDr4c23kEjr3p7F+hIDBWUBhwWr9E70a6XXWWel+lh31z
  +kz1y4CK+yOITyw3B13tm9ZTIUDm+baNjl4TDOKai+zLqoPoEWK0CWYEKgbA==', '1234', 'aes128', 'cbc',
  'sha256');
   gs_decrypt
  ------------
   database
  (1 row)
  ```

  说明

  - 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
  - 此函数需要结合gs_encrypt加密函数共同使用，且加密算法和HMAC算法要保证一致。

- gs_encrypt_aes128(encryptstr,keystr)

  描述：以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。keystr的长度范围为1~16字节。支持的加密数据类型：目前数据库支持的数值类

 版权所有 © 华为云计算技术有限公司

型，字符类型，二进制类型中的RAW，日期/时间类型中的DATE、TIMESTAMP、SMALLDATETIME。

返回值类型：text

返回值长度：至少为92字节，不超过 4*[(Len+68)/3] 字节，其中Len为加密前数据长度（单位为字节）。

示例：

```
SELECT gs_encrypt_aes128('DWS','1234');
                                    gs_encrypt_aes128
------------------------------------------------------------------------------------------------
 1gh4kKbkJ86NQ0A3w/VAvRZ1wM
+l4p6nbl0GIMvxgaCZbUFh0NZIrGE6N2Wdpp8hG8Fle6GJZAGKmjXuusJbgGa4y5U=
(1 row)
```

说明

- 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
- 在同一张数据表中，加密函数gs_encrypt_aes128与gs_encrypt不要混合使用。

- gs_decrypt_aes128(decryptstr,keystr)

  描述：以keystr为密钥对decryptstr字符串进行解密，返回解密后的字符串。解密使用的keystr必须保证与加密时使用的keystr一致才能正常解密。keystr不得为空。

  返回值类型：text

  示例：

  ```
  SELECT gs_decrypt_aes128('1gh4kKbkJ86NQ0A3w/VAvRZ1wM
  +l4p6nbl0GIMvxgaCZbUFh0NZIrGE6N2Wdpp8hG8Fle6GJZAGKmjXuusJbgGa4y5U=','1234');
   gs_decrypt_aes128
  -------------------
   DWS
  (1 row)
  ```

  说明

  - 由于该函数的执行过程需要传入解密口令，为了安全起见，gsql工具不会将该函数记录入执行历史。即无法在gsql里通过上下翻页功能找到该函数的执行历史。
  - 此函数需要结合gs_encrypt_aes128加密函数共同使用。

- gs_hash(hashstr, hashmethod)

  描述：以hashmethod算法对hashstr字符串进行信息摘要，返回信息摘要字符串。支持的hashmethod：sha256, sha384, sha512, sm3。

  返回值类型：text

  返回值长度：sha256和sm3返回64字节，sha384返回96字节，sha512返回128字节。

  示例：

  ```
  SELECT gs_hash('DWS', 'sha256');
                               gs_hash
  ------------------------------------------------------------------
   1d595d9e342bccc3b158fabfcdfece54d9f821cf9c7461c5d3b6c0157c442f94
  (1 row)
  ```

- md5(string)

  描述：将string使用MD5加密，并以16进制数作为返回值。

  说明

  MD5的安全性较低，不建议使用。

 版权所有 © 华为云计算技术有限公司

返回值类型：text

示例：

```
SELECT md5('ABC');
               md5
----------------------------------
 902fbdd2b1df0c4f70b4a5d23525e932
(1 row)
```

5.2.4 使用 pgcrypto 加密 DWS 数据

DWS数据库自8.2.0集群版本开始内置加密解密模块pgcrypto。pgcrypto模块允许数据库用户以加密形式存储数据的某些列，为敏感数据增加了一层额外的保护。因此在没有加密密钥的情况下，任何人都无法读取以加密形式存储在DWS数据库中的数据。

pgcrypto函数在数据库服务器内部运行，这意味着所有数据和密码都以明文形式在pgcrypto和客户端应用程序之间传输。为了获得最佳安全性，建议在客户端和DWS服务器之间使用SSL连接。

有关pgcrypto模块中各个函数的详细信息如下：

通用哈希函数

- digest()

  digest()函数可以根据不同的算法生成数据的二进制哈希值，语法如下：

  ```
  digest(data text, type text) returns bytea
  digest(data bytea, type text) returns bytea
  ```

  其中，data是原始数据，type是加密算法包括md5、sha1、sha224、sha256、sha384、sha512和sm3；函数的返回结果为二进制字符串。

  示例：

  使用digest() 函数对字符串DWS进行sha256加密存储：

  ```
  select digest('DWS', 'sha256');
                                digest
  ------------------------------------------------------------------
   \x1d595d9e342bccc3b158fabfcdfece54d9f821cf9c7461c5d3b6c0157c442f94
  (1 row)
  ```

- hmac()

  hmac()函数可以根据不同的算法为带有密钥的数据计算出MAC值，语法如下：

  ```
  hmac(data text, key text, type text) returns bytea
  hmac(data bytea, key bytea, type text) returns bytea
  ```

  其中，data是原始数据，key是加密密钥，type是加密算法，包括md5、sha1、sha224、sha256、sha384、sha512以及sm3；函数的返回结果为二进制字符串。

  示例：

  使用key123密钥和sha256加密算法对字符串DWS计算MAC值：

  ```
  select hmac('DWS', 'key123', 'sha256');
                                 hmac
  ------------------------------------------------------------------
   \xa2ee3b0d627a93166ac2e6a313deb86e0710ad4505cd7f7b88d563579174b8f9
  (1 row)
  ```

  对于digest() 函数，如果同时被修改了原始数据和加密结果，无法进行识别；hmac()函数只要密钥未泄露，可发现被篡改的数据。

  如果密钥长度大于哈希块长度，系统会先对密钥进行哈希运算，再将哈希结果作为密钥使用。

版权所有 © 华为云计算技术有限公司

密码哈希函数

crypt()和gen_salt()函数专用于哈希密码。crypt()执行哈希用于加密数据，gen_salt()用于生成加盐哈希。

crypt()中的算法和普通的MD5或者SHA1哈希算法存在以下不同之处：

- crypt()中算法很慢。由于密码包含的数据量很小，这是增加暴力破解难度的唯一方法。
- 它们使用一个随机值（称为*salt*，即盐值），因此这样具有相同口令的用户将得到不同的密文口令。这也是针对破解算法提供一种额外的安全保护。
- 它们的结果中包括了算法类型，因此可以针对不同用户使用不同的算法对密码进行加密。
- 其中一些算法具有自适应性，意味着当计算机性能变得更快时，可以调整该算法使其变得更慢，而不会产生与已有密码的不兼容性。

crypt()函数所支持的算法如下表：

表 5-6 crypt()支持的算法

算法	密码最大长度	自适应性	Salt位数	输出结果长度	描述
bf	72	√	128	60	基于Blowfish的2a变种算法
md5	unlimited	×	48	34	基于MD5的加密算法
xdes	8	√	24	20	扩展DES
des	8	×	12	13	原生UNIX加密算法

- crypt()

  crypt()语法格式如下：

  ```
  crypt(password text, salt text) returns text
  ```

  该函数返回password字符串crypt(3)格式的哈希值，salt参数由gen_salt()函数生成。

  对于相同的密码，crypt()函数每次也会返回不同的结果，因为gen_salt()函数每次都会生成不同的salt。校验密码时可以将之前生成的哈希结果作为salt。

  例如：设置一个新密码：

  ```
  UPDATE ... SET pswhash = crypt('new password', gen_salt('bf',10));
  ```

  通过比较存储的密码哈希值验证输入的密码的正确性。

  ```
  SELECT (pswhash = crypt('entered password', pswhash)) AS pswmatch FROM ... ;
  ```

  如果输入的口令正确，这会返回true。

  示例：

  ```
  create table userpwd(userid int8, pwd text);
  CREATE TABLE

  insert into userpwd values (1, crypt('this is a pwd', gen_salt('bf',10)));
  INSERT 0 1
  ```


版权所有 © 华为云计算技术有限公司

```
select crypt('this is a pwd', pwd)=pwd as result from userpwd where userid =1;
 result
--------
 t
(1 row)

select crypt('this is a wrong pwd', pwd)=pwd as result from userpwd where userid =1;
 result
--------
 f
(1 row)
```

- gen_salt()

  gen_salt()函数用来产生随机的参数给crypt，语法如下：

  ```
  gen_salt(type text [, iter_count integer ]) returns text
  ```

  该函数每次都会生成一个随机的盐值(salt)字符串，该字符串同时决定了crypt()函数使用的算法；type参数用于指定一个生成字符串的哈希算法，取值包括des、xdes、md5以及bf。对于xdes和bf算法，iter_count指迭代次数，数字越大加密时间越长，被破解需要的时间也越长。

  ```
  SELECT gen_salt('des'), gen_salt('xdes'), gen_salt('md5'), gen_salt('bf');
   gen_salt | gen_salt  |  gen_salt   |              gen_salt
  ----------+-----------+-------------+------------------------------
   qh       | _J9..uEUi | $1$SNgqyKAi | $2a$06$B/Etc3J8zYBV49LrDU97MO
  (1 row)
  ```

  每种算法生成的salt拥有固定的格式，例如bf算法结果中的$2a$06$，2a表示Blowfish的2a变种算法，06表示迭代的次数。

  如果忽略 iter_count，将会使用默认的迭代次数。允许的iter_count值与算法相关，如下表所示。对于xdes算法，迭代次数必须是一个奇数。

  表 5-7 crypt()的迭代计数

算法	默认值	最小值	最大值
xdes	725	1	16777215
bf	6	4	31

PGP 加密函数

DWS的PGP加密函数遵循OpenPGP(RFC 4880)标准，包括对称密钥加密（私钥加密）和非对称密钥加密（公钥加密）。

加密后的PGP消息由两部分组成：

- 这个消息的会话密钥（加密过的对称密钥或者公钥）。
- 使用该会话密钥加密的数据。

对于对称密钥（也就是密码）加密：

1. 使用String2Key（S2K）算法对密钥进行加密，类似于执行一个特意减慢并且包含随机salt的crypt() 算法，生成一个完整长度的二进制密钥。
2. 如果要求一个单独的会话密钥，生成一个随机的密钥。否则使用上面的S2K密钥直接作为会话密钥。
3. 如果直接使用S2K密钥，只将S2K设置加入会话密钥包中。否则，使用S2K密钥对会话密钥进行加密，然后放入会话密钥包中。

版权所有 © 华为云计算技术有限公司

对于公钥加密：

1. 生成一个随机的会话密钥。
2. 使用公钥对其进行加密后放入会话密钥包中。

无论哪种情况，数据的加密过程如下：

1. 执行可选的数据操作：压缩、转换成UTF-8或者换行符的转换。
2. 在数据前面增加一个随机字节组成的块，相当于使用了一个随机的初始值（IV）。
3. 追加随机前缀和数据的SHA1哈希值到数据后面。
4. 将所有内容使用会话密钥进行加密后放入数据包中。

支持的PGP加密函数：

- pgp_sym_encrypt()

  描述：用于对称密钥加密。

  语法格式：

  ```
  pgp_sym_encrypt(data text, psw text [, options text ]) returns bytea
  pgp_sym_encrypt_bytea(data bytea, psw text [, options text ]) returns bytea
  ```

  其中，data是要加密的数据；psw是PGP对称密钥；options 参数用于设置选项，参考**表5-8**。

- pgp_sym_decrypt()

  描述：用于解密PGP对称密钥加密后的消息。

  语法格式：

  ```
  pgp_sym_decrypt(msg bytea, psw text [, options text ]) returns text
  pgp_sym_decrypt_bytea(msg bytea, psw text [, options text ]) returns bytea
  ```

  其中，msg是要解密的消息；psw是PGP对称密钥；options参数用于设置选项，参考**表5-8**。为了避免输出无效的字符，不允许使用pgp_sym_decrypt函数对bytea数据进行解密；可以使用 pgp_sym_decrypt_bytea 对原始文本数据进行解密。

- pgp_pub_encrypt()

  描述：用于公共密钥加密。

  语法格式：

  ```
  pgp_pub_encrypt(data text, key bytea [, options text ]) returns bytea
  pgp_pub_encrypt_bytea(data bytea, key bytea [, options text ]) returns bytea
  ```

  其中，data是要加密的数据；key是PGP公钥，如果传入一个私钥将会返回错误；options参数用于设置选项，参考**表5-8**。

- pgp_pub_decrypt()

  描述：用于解密PGP公共密钥加密后的消息。

  语法格式：

  ```
  pgp_pub_decrypt(msg bytea, key bytea [, psw text [, options text ]]) returns text
  pgp_pub_decrypt_bytea(msg bytea, key bytea [, psw text [, options text ]]) returns bytea
  ```

  解密一个公共密钥加密的消息。*key*必须是对应于用来加密的公钥的私钥。如果私钥是用口令保护的，必须在*psw*中给出该口令。如果没有口令，但想要指定选项，需要给出一个空口令。

  为了避免输出无效的字符，不允许使用pgp_pub_decrypt函数对bytea数据进行解密；可以使用 pgp_pub_decrypt_bytea 对原始文本数据进行解密。

  其中，key是公共密钥对应的私钥；如果私钥使用了密码保护功能，必须在psw参数中指定密码；如果没有使用密码保护，想要指定options参数时必须指定一个空的psw。options参数用于设置选项，参考**表5-8**。

版权所有 © 华为云计算技术有限公司

- pgp_key_id()

  描述：用于提取PGP公钥或者私钥的密钥ID；如果传入一个加密后的消息，将会返回加密该消息使用的密钥ID。

  语法格式：

  ```
  pgp_key_id(bytea) returns text
  ```

  该函数可能返回两个特殊密钥ID：

  - SYMKEY，表示该消息使用对称密钥进行加密。
  - ANYKEY，表示该消息使用公钥进行加密，但是密钥ID已经被删除。这意味着需要尝试所有的密钥，查找可以解密该消息的私钥。pgcrypto不会产生这种加密消息。

  说明

  不同的密钥可能拥有相同的ID，这种情况很少见但可能存在。客户端应用程序需要自己尝试使用不同的密钥进行解密，就像处理ANYKEY一样。

- armor()

  描述：用于将二进制数据转换为PGP ASCII-armor格式，相当于Base64加上CRC以及额外的格式化。

  语法格式：

  ```
  armor(data bytea [ , keys text[], values text[] ]) returns text
  ```

- dearmor()

  描述：用于执行相反的转换。

  语法格式：

  ```
  dearmor(data text) returns bytea
  ```

  将加密后的数据bytea，转换成PGP ASCII-armor格式.或者反向转换。

  其中，data是需要转换的数据；如果指定了keys和values数值，每个key/value对都会生成一个armor header并添加到编码格式中；两个数组都是一维数组，长度相同，并且不能包含非ASCII字符。

- pgp_armor_headers()

  描述：函数用于返回数据中的armor header。

  ```
  pgp_armor_headers(data text, key out text, value out text) returns setof record
  ```

  返回结果是一个包含key和value两个字段的数据行集，如果其中包含任何非ASCII字符，都会被视作UTF-8字符。

  用GnuPG生成PGP 密钥

  要生成一个新密钥：

  ```
  gpg --gen-key
  ```

  更好的密钥类型是“DSA和Elgamal”。

  对于RSA密钥，必须创建仅用于签名的DSA或RSA密钥作为主控密钥，然后用gpg --edit-key增加一个RSA加密子密钥。

  要列举密钥：

  ```
  gpg --list-secret-keys
  ```

  要以ASCII-保护格式导出一个公钥：

  ```
  gpg -a --export KEYID > public.key
  ```

  要以 ASCII-保护格式导出一个私钥：

  ```
  gpg -a --export-secret-keys KEYID > secret.key
  ```

  在把这些密钥交给PGP函数之前，需要对它们使用dearmor()。或者如果你能处理二进制数据，可以从命令中去掉-a。

版权所有 © 华为云计算技术有限公司

须知

PGP加密函数的存在以下限制：

- 不支持签名。这也意味着它不会检查加密子密钥是否属于主控密钥。
- 不支持加密密钥作为主控密钥。由于通常并不鼓励这种用法，因此这应该不是问题。
- 不支持多个子密钥。由于实际应用中经常需要多个子密钥，这可能是个问题。另一方面，不要使用常规GPG/PGP密钥作为pgcrypto加密密钥，而应该创建新的密钥，因为这是非常不同的使用场景。

PGP函数的选项

pgcrypto函数中的选项名称和GnuPG类似，选项的值使用等号设置，每个选项使用逗号进行分隔。例如：

```
pgp_sym_encrypt(data, psw, 'compress-algo=1, cipher-algo=aes256')
```

除了convert-crlf之外，其他选项仅适用于加密函数。解密函数会从PGP数据中获取参数。

最常设置的选项包括compress-algo和unicode-mode。其他选项通常使用默认值。

表 5-8 pgcrypto 加密选项

选项	描述	默认值	取值	适用函数
cipher-algo	使用的密码算法。	aes128	bf, aes128, aes192, aes256, 3des, cast5	pgp_sym_encrypt, pgp_pub_encrypt
compress-algo	使用的压缩算法。	0	• 0，表示不压缩。 • 1，表示ZIP压缩。 • 2，表示ZLIB压缩（= ZIP加上元数据和CRC）	pgp_sym_encrypt, pgp_pub_encrypt
compress-level	压缩级别。级别越高压缩得越小但速度也越慢。0表示不压缩。	6	0，1-9	pgp_sym_encrypt, pgp_pub_encrypt
convert-crlf	加密时是否将\n转换成\r\n并且解密时执行相反的转换是否把\r\n转换成\n。RFC4880指定文本数据存储时需要使用\r\n作为换行符。	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt, pgp_sym_decrypt, pgp_pub_decrypt

版权所有 © 华为云计算技术有限公司

选项	描述	默认值	取值	适用函数
disable-mdc	不用 SHA-1 保护数据。仅用于兼容老旧的PGP产品。	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt
sess-key	使用单独的会话密钥。公钥加密总是使用一个单独的会话密钥。该选项用于对称密钥加密，因为对称密钥加密默认直接使用 S2K 密钥。	0	0，1	pgp_sym_encrypt
s2k-mode	使用的S2K 算法。	3	• 0 ，表示不使用 salt。不推荐！ • 1，表示使用 salt ，但是迭代固定次数。 • 3 ，可变的迭代计数。	pgp_sym_encrypt
s2k-count	S2K 算法的迭代次数。	65536 和 253952 之间的一个随机数值	大于等于1024并且小于等于65011712	pgp_sym_encrypt，并且 s2k-mode=3
s2k-digest-algo	S2K计算时的摘要算法。	sha1	md5, sha1	pgp_sym_encrypt
s2k-cipher-algo	加密单独会话密钥时使用的密码。	默认使用 cipher-algo 的算法	bf, aes, aes128, aes192, aes256	pgp_sym_encrypt

版权所有 © 华为云计算技术有限公司

选项	描述	默认值	取值	适用函数
unicode-mode	是否将文本数据在数据库内部编码和UTF-8 之间来回转换。如果当前数据库已经是UTF-8，不会执行转换，但是消息将被标记为UTF-8。没有指定该选项就不会被标记	0	0，1	pgp_sym_encrypt, pgp_pub_encrypt

原始加密函数

原始加密函数仅对数据运行一次加密，不支持PGP加密的任何高级功能，因此存在以下问题：

- 直接将用户密钥作为加密密钥。
- 不提供任何完整性检查来校验加密后的数据是否被修改。
- 需要用户自己关联所有加密参数，包括初始值（IV）。
- 不支持处理文本数据。

因此，在引入了PGP加密后，不建议使用这些原始加密函数。

```
encrypt(data bytea, key bytea, type text) returns bytea
decrypt(data bytea, key bytea, type text) returns bytea
encrypt_iv(data bytea, key bytea, iv bytea, type text) returns bytea
decrypt_iv(data bytea, key bytea, iv bytea, type text) returns bytea
```

其中，data是需要加密的数据；type用于指定加密/解密方法。type参数的语法如下：

```
algorithm [ - mode ] [ /pad: padding ]
```

其中algorithm的取值范围如下：

- bf，Blowfish算法。包括近义词：BF，BF-CBC；BLOWFISH，BF-CBC；BLOWFISH-CBC，BF-CBC；BLOWFISH-ECB，BF-ECB；BLOWFISH-CFB，BF-CFB。
- aes，AES算法(Rijndael-128, -192或-256)。包括近义词：AES，AES-CBC；RIJNDAEL，AES-CBC；RIJNDAEL，AES-CBC；RIJNDAEL-CBC，AES-CBC；RIJNDAEL-ECB，AES-ECB。
- DES算法。包括近义词：DES，DES-CBC；3DES，DES3-CBC；3DES-ECB，DES3-ECB；3DES-CBC，DES3-CBC
- CAST5算法。包括近义词：CAST5-CBC

mode的可能取值范围如下：

- cbc，下一个块依赖前一个块（默认值）
- ecb，每个块独立加密（不推荐，仅用于测试）

padding的取值范围如下：

版权所有 © 华为云计算技术有限公司

- pkcs，数据可以是任意长度（默认值）
- none，数据长度必须是密码块大小的倍数

例如，以下函数的加密结果相同：

```
encrypt(data, 'fooz', 'bf')
encrypt(data, 'fooz', 'bf-cbc/pad:pkcs')
```

对于函数encrypt_iv和decrypt_iv，参数iv表示CBC模式的初始值，ECB模式会忽略该参数。如果它的长度不是准确的块大小，可能会被截断或者使用0进行填充。对于没有该参数的两个函数，默认全部使用0填充。

随机数据函数

- gen_random_bytes()函数用于生成具有强加密性的随机字节。

  ```
  gen_random_bytes(count integer) returns bytea
  ```

  其中，count表示返回的字节数，取值为1到1024。

  示例：

  ```
  SELECT gen_random_bytes(16);
          gen_random_bytes
  ------------------------------------
   \x1f1eddc11153afdde0f9e1229f8f4caf
  (1 row)
  ```

- gen_random_uuid()函数用于返回一个version 4的随机UUID。

  ```
  SELECT gen_random_uuid();
  gen_random_uuid
  --------------------------------------
  2bd664a2-b760-4859-8af6-8d09ccc5b830
  ```

5.2.5 细粒度透明数据加密

数据库加密是保护敏感数据安全的关键手段，通过加密可以防止未经授权的用户访问和读取敏感数据。DWS的细粒度透明加密功能通过**表级加密**方式和**列级加密**方式提供了细粒度的加密能力，对数据库进行更加精准的数据安全保护，同时加密过程完全透明对应用层和SQL访问不会产生影响。用户可根据不同的数据敏感级别和访问需求来选择合适的加密方式。

细粒度透明加密支持两种加密算法：AES-CTR-128（非国密场景）和SM4-CTR-128（国密场景）。

开启国密SM4-CTR-128算法时，建议开启KAE（鲲鹏加速引擎，需联系技术支持开启GUC参数enable_kae_accelerate），以降低对集群性能的影响。（未开启KAE，集群性能开销将大幅增加；开启KAE后，集群性能开销显著降低，因SM4加密本身会带来一定的性能开销，在TPC-DS复杂查询场景下的仍会出现：列存整体性能劣化通常在5%至15%之间及行存整体性能劣化通常在50%以上的情况，具体的劣化幅度取决于硬件是否提供KAE鲲鹏加速引擎）。

具体的细粒度透明数据加密语法，请参考**CREATE TABLE**。

密钥缓存和轮转

执行ALTER CLUSTER ENCRYPTION KEY ROTATION命令进行密钥轮转后，系统会生成新的数据加密密钥（DEK）。每次轮转产生的密钥都会在首次访问时从KMS解密并缓存到本地缓存中，以减少后续访问KMS的网络开销。

TDE缓存容量通过GUC参数tde_cache_config（该参数需联系技术支持设置）的cachesize字段设置，默认值为100000。当缓存中的密钥数量达到容量上限时，系统会按照最近最少使用策略淘汰旧密钥，为新密钥腾出空间。

版权所有 © 华为云计算技术有限公司

若缓存容量设置过小，无法容纳所有历史轮转密钥，则被淘汰的密钥在下次访问时将产生缓存未命中，系统需重新从KMS获取并解密该密钥后再次写入缓存，这会增加I/O访问延迟。

缓存超时时间通过GUC参数tde_cache_config（该参数需联系技术支持设置）的cachetime字段设置，默认值-1表示不启用超时淘汰。当设置为正数时，系统会定期清理超过该时间未被访问的缓存密钥，同样会导致后续访问时需要重新连接KMS。

建议合理规划密钥轮转的频率，根据业务实际需要缓存的密钥数量设置cachesize，确保缓存能够容纳所有活跃密钥，避免因缓存淘汰产生额外的KMS访问开销。同时，根据密钥访问的时间局部性特征合理设置cachetime，在内存占用与访问性能之间取得平衡。

注意事项

使用细粒度透明加密功能时，需遵守以下约束，避免影响业务正常运行：

- 细粒度透明加密仅9.1.1.200及以上集群版本支持，且需联系技术支持配置GUC参数tde_version为2。
- 细粒度透明加密仅行存表、列存2.0表支持。
- 行存表不支持列级的透明加密配置，仅列存表支持列级。
- 集群级透明加密和细粒度透明加密功能互斥，无法同时开启。已经开启了集群级透明加密，则无法开启细粒度透明加密功能。
- 集群升级观察期内，不支持创建透明加密表。
- 索引不支持修改透明加密配置。
- 不支持基于加密表创建物化视图，也不支持对物化视图加密。
- 普通列存+delta表、Hstore非opt表、行列混存表、LSM表不支持表级加密。
- 临时表不支持加密。
- 细粒度透明加密功能强依赖KMS（密钥管理服务）才能获取密钥，如果DWS集群使用细粒度加密功能时出现报错，需联系技术支持修复KMS故障。故障修复前不能重启DWS集群或者实例，否则会导致集群不可用。

操作示例

以员工信息表employees为例，使用细粒度透明加密功能。

步骤1 创建表employees为初始未加密表。

```
CREATE TABLE employees (
    emp_id int PRIMARY KEY,
    emp_name text,
    emp_email text,
    emp_phone text,
    salary decimal(10,2),
    hire_date date );
```

步骤2 创建表级加密表employees_encrypted（AES-CTR-128算法），并将employees表数据迁移到employees_encrypted。

```
CREATE TABLE employees_encrypted (
    emp_id int PRIMARY KEY,
    emp_name text,
    emp_email text,
    emp_phone text,
    salary decimal(10,2),
    hire_date date )
```

 版权所有 © 华为云计算技术有限公司

```
WITH (encrypt_option = table,encrypt_algo = AES_CTR_128);
INSERT INTO employees_encrypted SELECT * FROM employees;
```

步骤3 创建SM4加密（国密算法）的薪资表salaries。

```
CREATE TABLE salaries (
    emp_id int,
    salary decimal(10,2),
    bonus decimal(10,2),
    effective_date date )
WITH (encrypt_option = table, encrypt_algo = SM4_CTR_128);
```

步骤4 创建列级加密表hr_analysis，仅针对敏感列“salary”信息加密。

```
CREATE TABLE hr_analysis (
    emp_id int,
    emp_name text,
    emp_department text,
    salary decimal(10,2) ENCRYPT,  -- 仅薪资列加密
    performance_score int )
WITH (orientation = column, encrypt_option = column);
```

步骤5 列级加密表hr_analysis由指定列加密修改为全列加密。

```
ALTER TABLE hr_analysis MODIFY (emp_id to encrypt);
ALTER TABLE hr_analysis MODIFY (emp_name to encrypt);
ALTER TABLE hr_analysis MODIFY (emp_department to encrypt);
ALTER TABLE hr_analysis MODIFY (performance_score to encrypt);
VACUUM FULL hr_analysis;
```

步骤6 对加密表进行解密。

```
ALTER TABLE employees_encrypted SET (encrypt_option = off);
VACUUM FULL employees_encrypted;
```

对初始未加密表开启加密。

```
ALTER TABLE employees SET (encrypt_option = table);
VACUUM FULL employees;
```

步骤7 修改表的加密算法为SM4。

```
ALTER TABLE employees SET (encrypt_algo = SM4_CTR_128);
VACUUM FULL employees;
ALTER TABLE hr_analysis SET (encrypt_algo = SM4_CTR_128);
VACUUM FULL hr_analysis;
```

步骤8 定期密钥轮转，按安全策略每半年执行一次密钥轮转。

```
ALTER CLUSTER ENCRYPTION KEY ROTATION;
```

----结束

 版权所有 © 华为云计算技术有限公司

6 查询 DWS 数据

6.1 单表查询

示例表：

```
CREATE TABLE newproducts
(
product_id INTEGER NOT NULL,
product_name VARCHAR2(60),
category VARCHAR2(60),
quantity  INTEGER
)
WITH (ORIENTATION = COLUMN) DISTRIBUTE BY HASH(product_id);


INSERT INTO newproducts VALUES (1502, 'earphones', 'electronics',150);
INSERT INTO newproducts VALUES (1601, 'telescope', 'toys',80);
INSERT INTO newproducts VALUES (1666, 'Frisbee', 'toys',244);
INSERT INTO newproducts VALUES (1700, 'interface', 'books',100);
INSERT INTO newproducts VALUES (2344, 'milklotion', 'skin care',320);
INSERT INTO newproducts VALUES (3577, 'dumbbell', 'sports',550);
INSERT INTO newproducts VALUES (1210, 'necklace', 'jewels', 200);
```

简单查询

通过SELECT ... FROM ... 语句从数据库中获取结果。

```
SELECT category FROM newproducts;
  category
------------
 electronics
 sports
 jewels
 toys
 books
 skin care
 toys
(7 rows)
```

对结果进行筛选

通过WHERE语句对查询的结果进行过滤，找到想要查询的部分。

```
SELECT * FROM newproducts WHERE category='toys';
 product_id | product_name | category | quantity
```

 版权所有 © 华为云计算技术有限公司

```
------------+--------------+----------+----------
      1601 | telescope    | toys     |       80
      1666 | Frisbee      | toys     |      244
(2 rows)
```

对结果进行排序

使用ORDER BY语句可以让查询结果按照期望的方式进行排序。

```
SELECT product_id,product_name,category,quantity FROM newproducts ORDER BY quantity DESC;
 product_id | product_name |  category   | quantity
------------+--------------+-------------+----------
      3577 | dumbbell     | sports      |      550
      2344 | milklotion   | skin care   |      320
      1666 | Frisbee      | toys        |      244
      1210 | necklace     | jewels      |      200
      1502 | earphones    | electronics |      150
      1700 | interface    | books       |      100
      1601 | telescope    | toys        |       80
(7 rows)
```

限制结果查询数量

如果需要查询只返回部分结果，可以使用LIMIT语句限制查询结果返回的记录数。

```
SELECT product_id,product_name,category,quantity FROM newproducts ORDER BY quantity DESC limit 5;
 product_id | product_name |  category   | quantity
------------+--------------+-------------+----------
      3577 | dumbbell     | sports      |      550
      2344 | milklotion   | skin care   |      320
      1666 | Frisbee      | toys        |      244
      1210 | necklace     | jewels      |      200
      1502 | earphones    | electronics |      150
(5 rows)
```

聚合查询

可以通过使用GROUP BY语句配合聚合函数，构建一个聚合查询来关注数据的整体情况。

```
SELECT category, string_agg(quantity,',') FROM newproducts group by category;
  category   | string_agg
-------------+------------
 toys        | 80,244
 books       | 100
 sports      | 550
 jewels      | 200
 skin care   | 320
 electronics | 150
```

6.2 多表连接查询

连接类型介绍

通过SQL完成各种复杂的查询，多表之间的连接是必不可少的。连接分为：内连接和外连接两大类，每大类中还可进行细分。

- 内连接：标准内连接（INNER JOIN），交叉连接（CROSS JOIN）和自然连接（NATURAL JOIN）。
- 外连接：左外连接（LEFT OUTER JOIN），右外连接（RIGHT OUTER JOIN）和全外连接（FULL JOIN）。

版权所有 © 华为云计算技术有限公司

为了能更好的说明各种连接之间的区别，下面通过具体示例进行详细的阐述。

创建示例表student和math_score，并插入数据，设置enable_fast_query_shipping为off（默认为on）即查询优化器使用分布式框架；参数explain_perf_mode为pretty（默认值为pretty）指定explain的显示格式。

```
CREATE TABLE student(
  id INTEGER,
  name varchar(50)
);

CREATE TABLE math_score(
  id INTEGER,
  score INTEGER
);

INSERT INTO student VALUES(1, 'Tom');
INSERT INTO student VALUES(2, 'Lily');
INSERT INTO student VALUES(3, 'Tina');
INSERT INTO student VALUES(4, 'Perry');

INSERT INTO math_score VALUES(1, 80);
INSERT INTO math_score VALUES(2, 75);
INSERT INTO math_score VALUES(4, 95);
INSERT INTO math_score VALUES(6, NULL);

SET enable_fast_query_shipping = off;
SET explain_perf_mode = pretty;
```

内连接

- 标准内连接（INNER JOIN）

  语法：

  ```
  left_table [INNER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：表示left_table和right_table中满足join_condition的行拼接在一起作为结果输出，不满足条件的元组不会输出。

  示例1：查询学生的数学成绩。

  ```
  SELECT s.id, s.name, ms.score FROM student s JOIN math_score ms on s.id = ms.id;
   id | name  | score
  ----+-------+-------
    2 | Lily  |    75
    1 | Tom   |    80
    4 | Perry |    95
  (3 rows)


  EXPLAIN SELECT s.id, s.name, ms.score FROM student s JOIN math_score ms on s.id = ms.id;
                                    QUERY PLAN
  ------------------------------------------------------------------------------------
   id |                operation                | E-rows | E-memory | E-width | E-costs
  ----+-----------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)            |      4 |          |      13 | 19.47
    2 |    ->  Hash Join (3,4)                  |      4 | 1MB      |      13 | 11.47
    3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       8 | 10.10
    4 |       ->  Hash                          |     12 | 16MB     |       9 | 1.28
    5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
    6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Join (3,4)
          Hash Cond: (ms.id = s.id)

    ====== Query Summary =====
  -------------------------------
  ```


版权所有 © 华为云计算技术有限公司

```
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 4400KB
(19 rows)
```

- 交叉连接（CROSS JOIN）

  语法：

  ```
  left_table CROSS JOIN right_table
  ```

  说明：表示left_table中所有行和right_table中的所有行分别进行连接，最终结果行数等于两边行数的乘积。又称笛卡尔积。

  示例2：学生表和数学成绩表的交叉连接。

  ```
  SELECT s.id, s.name, ms.score FROM student s CROSS JOIN math_score ms;
   id | name  | score
  ----+-------+-------
    3 | Tina  |    80
    2 | Lily  |    80
    1 | Tom   |    80
    4 | Perry |    80
    3 | Tina  |
    2 | Lily  |
    1 | Tom   |
    4 | Perry |
    3 | Tina  |    95
    2 | Lily  |    95
    1 | Tom   |    95
    4 | Perry |    95
    2 | Lily  |    75
    3 | Tina  |    75
    1 | Tom   |    75
    4 | Perry |    75
  (16 rows)

  EXPLAIN SELECT s.id, s.name, ms.score FROM student s CROSS JOIN math_score ms;
                                    QUERY PLAN
  ------------------------------------------------------------------------------------------
    id |                operation                | E-rows | E-memory | E-width | E-costs
   ----+-----------------------------------------+--------+----------+---------+---------
     1 | ->  Streaming (type: GATHER)            |    120 |          |      13 | 19.89
     2 |    ->  Nested Loop (3,4)                |    120 | 1MB      |      13 | 11.89
     3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       4 | 10.10
     4 |       ->  Materialize                   |     12 | 16MB     |       9 | 1.30
     5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
     6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

    ====== Query Summary =====
   -------------------------------
   System available mem: 1761280KB
   Query Max mem: 1761280KB
   Query estimated mem: 4144KB
  (14 rows)
  ```

- 自然连接（NATURAL JOIN）

  语法：

  ```
  left_table NATURAL JOIN right_table
  ```

  说明：表示left_table和right_table中列名相同的列进行等值连接，且自动将同名列只保留一份。

  示例3：学生表和数学成绩表的自然连接，两表的同名列为id列，将按照id列进行等值连接。

  ```
  SELECT * FROM student s NATURAL JOIN math_score ms;
   id | name  | score
  ----+-------+-------
    1 | Tom   |    80
    4 | Perry |    95
  ```


版权所有 © 华为云计算技术有限公司

```
 2 | Lily |    75
(3 rows)

EXPLAIN SELECT * FROM student s NATURAL JOIN math_score ms;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
 id |               operation                | E-rows | E-memory | E-width | E-costs
----+----------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)           |      4 |          |      13 | 19.47
  2 |    ->  Hash Join (3,4)                 |      4 | 1MB      |      13 | 11.47
  3 |       ->  Seq Scan on math_score ms    |     30 | 1MB      |       8 | 10.10
  4 |       ->  Hash                         |     12 | 16MB     |       9 | 1.28
  5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
  6 |             ->  Seq Scan on student s  |      4 | 1MB      |       9 | 1.01

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
       Hash Cond: (ms.id = s.id)

  ====== Query Summary =====
-------------------------------
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 4400KB
(19 rows)
```

外连接

- 左外连接（LEFT JOIN）

  语法：

  ```
  left_table LEFT [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：左外连接的结果集包括left_table的所有行，而不仅是连接列所匹配的行。如果left_table的某行在right_table中没有匹配行，则在相关联的结果集行中输出right_table的列均为空值。

  示例4：学生表和数学成绩表进行左外连接，学生表中id为3的行在结果集中对应的右表数据用NULL填充。

  ```
  SELECT s.id, s.name, ms.score FROM student s LEFT JOIN math_score ms on (s.id = ms.id);
   id | name  | score
  ----+-------+-------
   3 | Tina  |
   1 | Tom   |    80
   2 | Lily  |    75
   4 | Perry |    95
  (4 rows)

  EXPLAIN SELECT s.id, s.name, ms.score FROM student s LEFT JOIN math_score ms on (s.id = ms.id);
                                    QUERY PLAN
  ---------------------------------------------------------------------------------------------
   id |                operation                 | E-rows | E-memory | E-width | E-costs
  ----+------------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)             |      4 |          |      13 | 10.26
    2 |    ->  Hash Left Join (3, 5)             |      4 | 1MB      |      13 | 2.26
    3 |       ->  Streaming(type: REDISTRIBUTE)  |      4 | 2MB      |       9 | 1.11
    4 |          ->  Seq Scan on student s       |      4 | 1MB      |       9 | 1.01
    5 |       ->  Hash                           |      4 | 16MB     |       8 | 1.11
    6 |          ->  Streaming(type: REDISTRIBUTE) |      4 | 2MB      |       8 | 1.11
    7 |             ->  Seq Scan on math_score ms |      4 | 1MB      |       8 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Left Join (3, 5)
         Hash Cond: (s.id = ms.id)

    ====== Query Summary =====
  ```


版权所有 © 华为云计算技术有限公司

```
------------------------------
System available mem: 901120KB
Query Max mem: 901120KB
Query estimated mem: 7520KB
(20 rows)
```

- 右外连接（RIGHT JOIN）

  语法：

  ```
  left_table RIGHT [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：与左外连接相反，右外连接的结果集包括right_table的所有行，而不仅是连接列所匹配的行。如果right_table的某行在left_table中没有匹配行，则在相关联的结果集行中left_table的列均为空值。

  示例5：学生表和数学成绩表进行右外连接，数学成绩表中id为6的行在结果集中对应的左表数据用NULL填充。

  ```
  SELECT ms.id, s.name, ms.score FROM student s RIGHT JOIN math_score ms on (s.id = ms.id);
   id | name  | score
  ----+-------+-------
    1 | Tom   |    80
    6 |       |
    4 | Perry |    95
    2 | Lily  |    75

  EXPLAIN SELECT ms.id, s.name, ms.score FROM student s RIGHT JOIN math_score ms on (s.id = ms.id);
                                    QUERY PLAN
  ------------------------------------------------------------------------------------------
   id |                operation                | E-rows | E-memory | E-width | E-costs
  ----+-----------------------------------------+--------+----------+---------+---------
    1 | ->  Streaming (type: GATHER)            |     30 |          |      13 | 19.47
    2 |    ->  Hash Left Join (3, 4)            |     30 | 1MB      |      13 | 11.47
    3 |       ->  Seq Scan on math_score ms     |     30 | 1MB      |       8 | 10.10
    4 |       ->  Hash                          |     12 | 16MB     |       9 | 1.28
    5 |          ->  Streaming(type: BROADCAST) |     12 | 2MB      |       9 | 1.28
    6 |             ->  Seq Scan on student s   |      4 | 1MB      |       9 | 1.01

   Predicate Information (identified by plan id)
  -----------------------------------------------
    2 --Hash Left Join (3, 4)
          Hash Cond: (ms.id = s.id)

    ====== Query Summary =====
  -------------------------------
  System available mem: 1761280KB
  Query Max mem: 1761280KB
  Query estimated mem: 5424KB
  (19 rows)
  ```

  对于右外连接，在join算子中却显示的是Left。这是因为，右外连接其实就是将左右表进行交互后的左外连接，所以数据库内部实现为了减少处理逻辑，会将右外连接转为左外连接。

- 全外连接（FULL JOIN）

  语法：

  ```
  left_table FULL [OUTER] JOIN right_table [ ON join_condition | USING ( join_column )]
  ```

  说明：全外连接是左外连接和右外连接的综合。全外连接的结果集包括left_table和right_table的所有行，而不仅是连接列所匹配的行。如果left_table的某行在right_table中没有匹配行，则在相关联的结果集行中right_table的列均为空值。如果right_table的某行在left_table中没有匹配行，则在相关联的结果集行中left_table的列均为空值。

  示例6：学生表和数学成绩表进行全外连接，学生表中id为3的行在结果集中对应的右表数据用NULL填充，数学成绩表中id为6的行在结果集中对应的左表数据用NULL填充。

 版权所有 © 华为云计算技术有限公司

```
SELECT s.id, s.name, ms.id, ms.score FROM student s FULL JOIN math_score ms ON (s.id = ms.id);
 id | name  | id | score
----+-------+----+-------
  2 | Lily  |  2 |    75
  4 | Perry |  4 |    95
  1 | Tom   |  1 |    80
  3 | Tina  |    |
    |       |  6 |
(5 rows)

EXPLAIN SELECT s.id, s.name, ms.id, ms.score FROM student s FULL JOIN math_score ms ON (s.id =
ms.id);
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                operation                 | E-rows | E-memory | E-width | E-costs
----+------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)             |     30 |          |      17 | 20.24
  2 |    ->  Hash Full Join (3, 5)             |     30 | 1MB      |      17 | 12.24
  3 |       ->  Streaming(type: REDISTRIBUTE)  |     30 | 2MB      |       8 | 11.06
  4 |          ->  Seq Scan on math_score ms   |     30 | 1MB      |       8 | 10.10
  5 |       ->  Hash                           |      4 | 16MB     |       9 | 1.11
  6 |          ->  Streaming(type: REDISTRIBUTE) |   4 | 2MB      |       9 | 1.11
  7 |             ->  Seq Scan on student s    |      4 | 1MB      |       9 | 1.01

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Full Join (3, 5)
        Hash Cond: (ms.id = s.id)

  ====== Query Summary =====
--------------------------------
 System available mem: 1761280KB
 Query Max mem: 1761280KB
 Query estimated mem: 6496KB
(20 rows)
```

多表查询中 on 条件和 where 条件的区别

从上面各种连接语法中可见，除自然连接和交叉连接外，其他都需要有on条件（using在查询解析过程中会被转为on条件）来限制两表连接的结果。通常在查询的语句中也都会有where条件限制查询结果。这里说的on连接条件和where过滤条件是指不含可以下推到表上的过滤条件。on和where的区别是：

- on条件是两表连接的约束条件。
- where是对两表连接后产生的结果集再次进行过滤。

简单总结就是：on条件优先于where条件，在两表进行连接时被应用；生成两表连接结果集后，再应用where条件。

6.3 子查询表达式

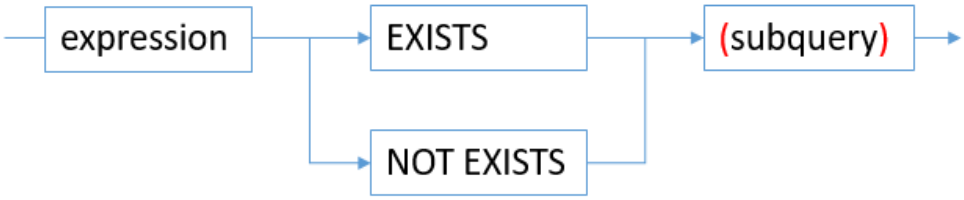
子查询允许在一个查询中嵌套另一个查询，从而实现更复杂的数据查询和分析。

子查询表达式

- EXISTS/NOT EXISTS

  首先执行子查询，然后根据子查询的结果是否满足条件来决定是否继续执行主查询。如果子查询至少返回一行，则EXISTS结果就为“真”。如果子查询没有返回任何行， NOT EXISTS的结果是“真”。

 版权所有 © 华为云计算技术有限公司

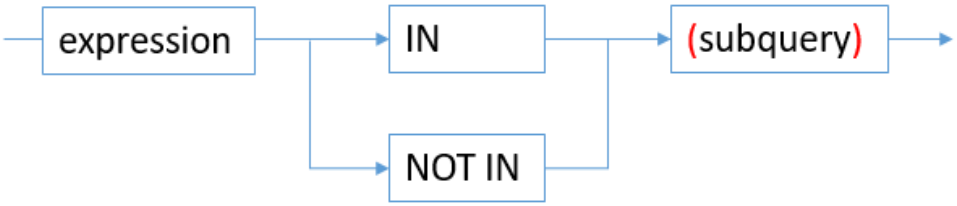


语法：

```
WHERE column_name EXISTS/NOT EXISTS (subquery)
```

- IN/NOT IN

  用于测试某个给定的比较值是否存在于某一组值里。如果外层查询里的行与子查询返回的某一个行相匹配，那么IN的结果为“真”。如果外层查询中的行与子查询返回的所有行都不匹配，那么NOT IN的结果为“真”。



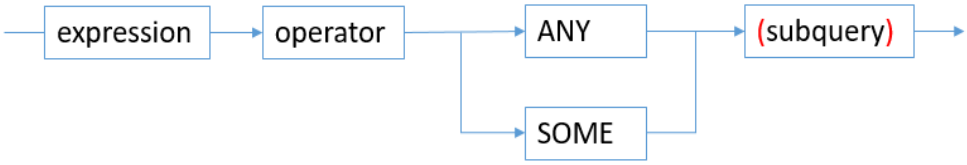
语法：

```
WHERE column_name IN/NOT IN (subquery)
```

- ANY/SOME

  ANY表示子查询中的任何值都可以与外部查询中的值匹配。SOME与ANY相同，只是在语法上的差别。

  右边的子查询必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较（=、<>、<、<=、>、>=），其结果必须是布尔值。如果至少获得一个真值，则ANY结果为“真”。如果全部获得假值，则结果是“假”（包括子查询没有返回任何行的情况）。

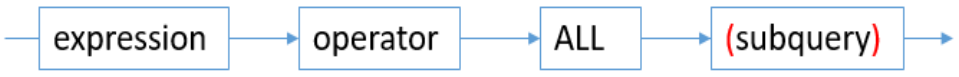


语法：

```
WHERE column_name operator ANY/SOME (subquery)
```

- ALL

  右边的子查询必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较（=、<>、<、<=、>、>=），其结果必须是布尔值。如果全部获得真值，ALL结果为“真”（包括子查询没有返回任何行的情况）。如果至少获得一个假值，则结果是“假”。



语法：

```
WHERE column_name operator ALL (subquery)
```


版权所有 © 华为云计算技术有限公司

表 6-1 ALL 查询条件

条件	描述
column_name > ALL(…)	column_name列中的值必须大于要评估为true的集合中的最大值。
column_name >= ALL(…)	column_name列中的值必须大于或等于要评估为true的集合中的最大值。
column_name < ALL(…)	column_name列中的值必须小于要评估为true的集合中的最小值。
column_name <= ALL(…)	column_name列中的值必须小于或等于要评估为true的集合中的最小值。
column_name <> ALL(…)	column_name列中的值不得等于要评估为true的集合中的任何值。
column_name = ALL(…)	column_name列中的值必须等于要评估为true的集合中的任何值。

示例

创建记录课程信息的表course，并插入数据：

```
CREATE TABLE course(cid VARCHAR(10) COMMENT 'No.course',cname VARCHAR(10) COMMENT 'course name',teid VARCHAR(10) COMMENT 'No.teacher');

INSERT INTO course VALUES('01' , 'course1' , '02');
INSERT INTO course VALUES('02' , 'course2' , '01');
INSERT INTO course VALUES('03' , 'course3' , '03');
```

创建教师表teacher，并插入数据：

```
CREATE TABLE teacher(teid VARCHAR(10) COMMENT '教师编号',tname VARCHAR(10) COMMENT '教师姓名');

INSERT INTO teacher VALUES('01' , 'teacher1');
INSERT INTO teacher VALUES('02' , 'teacher2');
INSERT INTO teacher VALUES('03' , 'teacher3');
INSERT INTO teacher VALUES('04' , 'teacher4');
```

- EXISTS/NOT EXISTS示例

  查询在course表中的教师记录：

  ```
  SELECT * FROM teacher WHERE EXISTS (SELECT * FROM course WHERE course.teid = teacher.teid);
  ```

  ```
   teid |  tname
  ------+----------
   02   | teacher2
   01   | teacher1
   03   | teacher3
  (3 rows)
  ```

  查询没有在course表中的教师记录：

  ```
  SELECT * FROM teacher WHERE NOT EXISTS (SELECT * FROM course WHERE course.teid = teacher.teid);
  ```

  ```
   teid |  tname
  ------+----------
   04   | teacher4
  (1 row)
  ```


版权所有 © 华为云计算技术有限公司

- IN/NOT IN 示例

  根据教师id匹配course表，查询在course表的教师信息：

  ```
  SELECT * FROM course WHERE teid IN (SELECT teid FROM teacher );
  ```

  ```
   cid |  cname  | teid
  -----+---------+------
   01  | course1 | 02
   03  | course3 | 03
   02  | course2 | 01
  (3 rows)
  ```

  查询不在course表的教师信息：

  ```
  SELECT * FROM teacher WHERE teid NOT IN (SELECT teid FROM course );
  ```

  ```
   teid |  tname
  ------+----------
   04   | teacher4
  (1 row)
  ```

- ANY/SOME 示例

  左侧主句与右侧子查询进行字段比对，获取需要的结果集：

  ```
  SELECT * FROM course WHERE teid < ANY (SELECT teid FROM teacher where teid<>'04');
  ```

  或

  ```
  SELECT * FROM course WHERE teid < some (SELECT teid FROM teacher where teid<>'04');
  ```

  ```
  cid |  cname  | teid
  -----+---------+------
   01  | course1 | 02
   02  | course2 | 01
  (2 rows)
  ```

- ALL示例

  查询teid列中的值必须小于要评估为true的集合中的最小值：

  ```
  SELECT * FROM course WHERE teid < ALL(SELECT teid FROM teacher WHERE teid<>'01');
  ```

  ```
   cid |  cname  | teid
  -----+---------+------
   02  | course2 | 01
  (1 row)
  ```

注意事项

- 禁止一条SQL语句中，出现重复子查询语句。
- 尽量少用标量子查询（标量子查询指结果为1个值，并且条件表达式为等值的子查询）。
- 避免在SELECT目标列中使用子查询，可能导致计划无法下推影响执行性能。
- 子查询嵌套深度建议不超过2层。由于子查询会带来临时表开销，过于复杂的查询应考虑从业务逻辑上进行优化。

子查询可以在 SELECT 语句中嵌套其他查询，从而实现更复杂的查询。子查询还可以在WHERE子句中使用其他查询的结果，从而更好地过滤数据。但是子查询可能会导致查询性能问题和代码难阅读和理解。 所以在DWS数据库中使用SQL子查询时，请结合实际业务情况进行操作。

 版权所有 © 华为云计算技术有限公司

6.4 WITH 表达式

WITH表达式用于定义在大型查询中使用的辅助语句，这些辅助语句通常被称为公共表达式或CTE（即common table expr），可以理解为一个带名称的子查询，之后该子查询可以以其名称在查询中被多次引用。

WITH表达式中的辅助语句可以是SELECT、INSERT、UPDATE或DELETE，并且WITH子句本身也可以被附加到一个主语句中，主语句可以是SELECT、INSERT或DELETE。

WITH 中的 SELECT

在WITH子句中使用SELECT的相关信息。

语法格式

```
[WITH [RECURSIVE] with_query [,…] ] SELECT …
```

其中，with_query的语法为：

```
with_query_name [ ( column_name [, ...] ) ]
    AS [ [ NOT ] MATERIALIZED ] ( {select | values | insert | update | delete} )
```


版权所有 © 华为云计算技术有限公司

⚠ 注意

- 显式指定MATERIALIZED时，将子查询执行一次，并将其结果集进行物化；指定NOT MATERIALIZED时，则将其子查询替换到主查询中的引用处。
- 每个CTE的AS语句指定的SQL语句，必须是可以返回查询结果的语句，可以是普通的SELECT查询语句，也可以是INSERT、UPDATE、DELETE、VALUES等其他数据修改语句，使用数据修改语句时需要通过RETURNING子句返回元组。例如：
  ```
  WITH s AS (INSERT INTO t VALUES(1) RETURNING a) SELECT * FROM s;
  ```
- 单个WITH表达式表示一个SQL语句块中的CTE定义，可以同时定义多个CTE，每个CTE可以指定列名，也可以默认使用查询输出列的别名。例如：
  ```
  WITH s1(a, b) AS (SELECT x, y FROM t1), s2 AS (SELECT x, y FROM t2) SELECT * FROM s1 JOIN s2 ON s1.a=s2.x;
  ```
  该语句中定义了两个CTE，s1和s2，其中s1指定了列名为a，b，s2未指定列名，则列名为输出列名x，y。
- 每个CTE可以在主查询中引用0次、1次或多次。
- 同一个语句块中不能出现同名的CTE，但不同语句块中可以出现同名的CTE，此时，语句中引用的CTE则是距离引用位置最近的语句块中的CTE。
- 由于SQL语句中可能包含多个SQL语句块，每个语句块都可以包含一个WITH表达式，每个WITH表达式中的CTE可以在当前语句块、当前语句块的后续CTE中，以及子层语句块中引用，但不能在父层语句块中引用。由于每个CTE的定义也是个语句块，因此也支持在该语句块中定义WITH表达式。
- 当CTE由于指定MATERIALIZED或被引用多次而物化时，CTE整体将作为一个表看待，所以CTE中查询或filter条件无法单独提升到CTE外进行其他优化。如下示例，CTE中的filter条件无法与join条件进行等价类推导，所以无法使join的分区表采用分区剪枝优化：

```
CREATE TABLE date_part(ds date) distribute by hash(ds);

CREATE TABLE t1(id1 int, ds date) distribute by hash(ds)
partition by range(ds)
 ( partition p1 values less than('20250101'),
   partition p2 values less than('20250201'),
   partition p3 values less than('20250301'),
   partition p4 values less than('20250401'),
   partition p5 values less than('20250501'),
   partition p6 values less than('20250601'));

CREATE TABLE t3(id3 int, ds date) distribute by hash(ds)
partition by range(ds)
 ( partition p1 values less than('20250101'),
   partition p2 values less than('20250201'),
   partition p3 values less than('20250301'),
   partition p4 values less than('20250401'),
   partition p5 values less than('20250501'),
   partition p6 values less than('20250601'));

explain verbose -- CTE被物化时，无法将CTE中的filter条件提升与join条件进行等价类推导来使join的分区表应用分区剪枝
 with dt as (select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
 t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
 t2 as (select id3 from t3 join dt t2 on t3.ds = t2.ds) select * from t1 join t2 on t1.id1 = 100;

explain with dt as (select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
t2 as(select id3 from t3 join dt t2 on t3.ds = t2.ds)
select * from t1 join t2 on t1.id1 = 100;
                                                  QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
 id |                       operation                       | E-rows | E-memory | E-width | E-costs
```


版权所有 © 华为云计算技术有限公司

```
----+-----------------------------------------------------------------+--------+----------+---------+---------
 1 | ->  Row Adapter                                    |      2 |          |       8 | 66.27
 2 |    ->  Vector Streaming (type: GATHER)                    |      2 |          |       8 | 66.27
 3 |       ->  Vector Nest Loop (4,15)                  |      2 | 1MB      |       8 | 60.27
 4 |         ->  Vector CTE Append(5, 7)                |      2 | 1MB      |      12 | 40.26
 5 |           ->  Vector Streaming(type: LOCAL GATHER)         |      1 | 2MB      |       8 | 20.01
 6 |             ->  CStore Scan on date_part  [5, CTE dt(0)] |      1 | 1MB      |       8 | 20.01
 7 |           ->  Vector Nest Loop (8,9)               |      2 | 1MB      |      12 | 20.25
 8 |             ->  Vector CTE Scan on dt(0) t2              |      1 | 1MB      |       8 | 0.02
 9 |             ->  Vector Materialize                 |      4 | 16MB     |       4 | 20.22
10 |               ->  Vector Streaming(type: BROADCAST)      |      4 | 2MB      |       4 | 20.21
11 |                 ->  Vector Nest Loop (12,14)             |      2 | 1MB      |       4 | 20.04
12 |                   ->  Vector Partition Iterator        |      1 | 1MB      |      12 | 20.00
13 |                     ->  Partitioned CStore Scan on t1 |      1 | 1MB      |      12 | 20.00
14 |                   ->  Vector CTE Scan on dt(0) t2      |      1 | 1MB      |       8 | 0.02
15 |         ->  Vector Partition Iterator              |      1 | 1MB      |      12 | 20.00
16 |           ->  Partitioned CStore Scan on t3              |      1 | 1MB      |      12 | 20.00

                                        Predicate Information (identified by plan id)

-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------
  3 --Vector Nest Loop (4,15)
        Join Filter: (t2.ds = t3.ds)
  6 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
 11 --Vector Nest Loop (12,14)
        Join Filter: (public.t1.ds = t2.ds)
 12 --Vector Partition Iterator
        Iterations: 6
 13 --Partitioned CStore Scan on t1
        CU Predicate Filter: (id1 = 100)
        Pushdown Predicate Filter: (id1 = 100)
        Partitions Selected by Static Prune: 1..6
 15 --Vector Partition Iterator
        Iterations: 6
 16 --Partitioned CStore Scan on t3
        Partitions Selected by Static Prune: 1..6

  ====== Query Summary =====
-------------------------------
System available mem: 2990080KB
Query Max mem: 2990080KB
Query estimated mem: 11312KB
Turbo Engine: true
(45 rows)


explain  -- CTE不被物化时，会将CTE被展开为子查询，其中的filter条件可与join条件进行等价类推导使join
的分区表应用分区剪枝
with dt as not materialized
(select ds from date_part where ds>='2025-05-04' and ds < '2025-05-06'),
t1 as(select id1 from t1 join dt t2 on t1.ds = t2.ds),
t2 as(select id3 from t3 join dt t2 on t3.ds = t2.ds) select * from t1 join t2 on t1.id1 = 100;
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------
 id |                       operation                       | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                                    |      2 |          |       8 | 52.26
  2 |    ->  Vector Streaming (type: GATHER)                    |      2 |          |       8 | 52.26
  3 |       ->  Vector Nest Loop (4,6)                   |      2 | 1MB      |       8 | 46.26
  4 |         ->  Vector Streaming(type: BROADCAST)         |      2 | 2MB      |       8 | 20.10
  5 |           ->  CStore Scan on date_part               |      1 | 1MB      |       8 | 20.01
  6 |         ->  Vector Materialize                     |      2 | 16MB     |      16 | 26.15
  7 |           ->  Vector Nest Loop (8,11)                |      2 | 1MB      |      16 | 26.14
```


版权所有 © 华为云计算技术有限公司

```
  8 |              ->  Vector Streaming(type: BROADCAST)   |      2 | 2MB       |      12 | 3.10
  9 |                ->  Vector Partition Iterator        |      1 | 1MB       |      12 | 3.01
 10 |                  ->  Partitioned CStore Scan on t3 |      1 | 1MB       |      12 | 3.01
 11 |              ->  Vector Nest Loop (12,14)            |      2 | 1MB       |       4 | 23.03
 12 |                ->  Vector Partition Iterator        |      1 | 1MB       |      12 | 3.01
 13 |                  ->  Partitioned CStore Scan on t1 |      1 | 1MB       |      12 | 3.01
 14 |                ->  CStore Scan on date_part         |      1 | 1MB       |       8 | 20.01

                                                  Predicate Information (identified by plan id)

-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------
  3 --Vector Nest Loop (4,6)
        Join Filter: (t3.ds = public.date_part.ds)
  5 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
  9 --Vector Partition Iterator
        Iterations: 1
 10 --Partitioned CStore Scan on t3
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))
        Partitions Selected by Static Prune: 6
 11 --Vector Nest Loop (12,14)
        Join Filter: (public.t1.ds = public.date_part.ds)
 12 --Vector Partition Iterator
        Iterations: 1
 13 --Partitioned CStore Scan on t1
        CU Predicate Filter: ((id1 = 100) AND (ds >= '2025-05-04 00:00:00'::timestamp without time zone)
AND (ds < '2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((id1 = 100) AND (ds >= '2025-05-04 00:00:00'::timestamp without
time zone) AND (ds < '2025-05-06 00:00:00'::timestamp without time zone))
        Partitions Selected by Static Prune: 6
 14 --CStore Scan on date_part
        CU Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds <
'2025-05-06 00:00:00'::timestamp without time zone))
        Pushdown Predicate Filter: ((ds >= '2025-05-04 00:00:00'::timestamp without time zone) AND (ds
< '2025-05-06 00:00:00'::timestamp without time zone))

  ====== Query Summary =====
 ------------------------------
 System available mem: 2990080KB
 Query Max mem: 2990080KB
 Query estimated mem: 12352KB
 Turbo Engine: true
(48 rows)
```

WITH中SELECT的基本价值是将复杂的查询分解为简单的部分。示例如下：

```
WITH regional_sales AS (
    SELECT region, SUM(amount) AS total_sales
    FROM orders
    GROUP BY region
), top_regions AS (
    SELECT region
    FROM regional_sales
    WHERE total_sales > (SELECT SUM(total_sales)/10 FROM regional_sales)
)
SELECT region,
       product,
       SUM(quantity) AS product_units,
       SUM(amount) AS product_sales
FROM orders
WHERE region IN (SELECT region FROM top_regions)
GROUP BY region, product;
```


版权所有 © 华为云计算技术有限公司

WITH子句定义了两个辅助语句regional_sales和top_regions，其中regional_sales的输出用在top_regions中而top_regions的输出用在主SELECT查询。这个例子可以不用WITH来书写，但是就必须要用两层嵌套的子SELECT，使得查询更长更难以维护。

WITH 递归查询

通过声明RECURSIVE关键字，一个WITH查询可以引用它自己的输出。

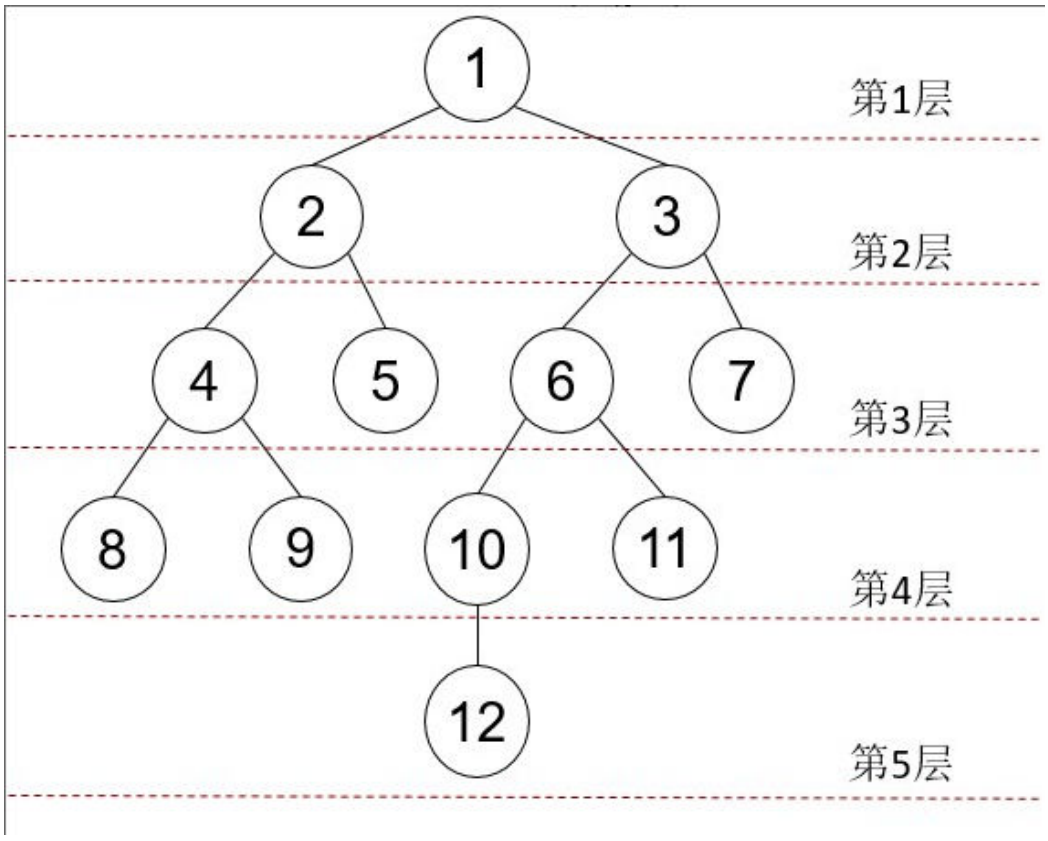
递归WITH查询的通常形式如下：

```
non_recursive_term UNION [ALL] recursive_term
```

其中：UNION在合并集合时会执行去重操作，而UNION ALL则直接将结果集合并、不执行去重；只有递归项能够包含对于查询自身输出的引用。

使用递归WITH时，必须确保查询的递归项最终不会返回元组，否则查询将无限循环。

使用表tree来存储下图中的所有节点信息：



表定义语句如下：

```
CREATE TABLE tree(id INT, parentid INT);
```

表中数据如下：

```
INSERT INTO tree VALUES(1,0),(2,1),(3,1),(4,2),(5,2),(6,3),(7,3),(8,4),(9,4),(10,6),(11,6),(12,10);

SELECT * FROM tree;
 id | parentid
----+----------
  1 |        0
  2 |        1
  3 |        1
```


版权所有 © 华为云计算技术有限公司

```
 4 |       2
 5 |       2
 6 |       3
 7 |       3
 8 |       4
 9 |       4
10 |       6
11 |       6
12 |      10
(12 rows)
```

通过以下WITH RECURSIVE语句，可以返回从顶层1号节点开始，整个树的节点，以及层次信息：

```
WITH RECURSIVE nodeset AS
(
-- recursive initializing query
SELECT id, parentid, 1 AS level FROM tree
WHERE id = 1
UNION ALL
-- recursive join query
SELECT tree.id, tree.parentid, level + 1 FROM tree, nodeset
WHERE tree.parentid = nodeset.id
)
SELECT * FROM nodeset ORDER BY id;
```

上述查询中可以看出，一个典型的WITH RECURSIVE表达式包含至少一个递归查询的CTE，该CTE中的定义为一个UNION ALL集合操作，第一个分支为递归起始查询，第二个分支为递归关联查询，需要自引用第一部分进行不断递归关联。该语句执行时，递归起始查询执行一次，关联查询执行若干次并将结果叠加到起始查询结果集中，直到某一些关联查询结果为空，则返回。

上述查询的执行结果如下：

```
 id | parentid | level
----+----------+-------
  1 |        0 |     1
  2 |        1 |     2
  3 |        1 |     2
  4 |        2 |     3
  5 |        2 |     3
  6 |        3 |     3
  7 |        3 |     3
  8 |        4 |     4
  9 |        4 |     4
 10 |        6 |     4
 11 |        6 |     4
 12 |       10 |     5
(12 rows)
```

从返回结果可以看出，起始查询结果包含level=1的结果集，关联查询执行了五次，前四次分别输出level=2,3,4,5的结果集，在第五次执行时，由于没有parentid和输出结果集id相等的记录，也就是再没有多余的孩子节点，因此查询结束。

说明

对于WITH RECURSIVE表达式，DWS支持其分布式执行。由于WITH RECURSIVE涉及循环运算，DWS引入了参数max_recursive_times，用于控制WITH RECURSIVE的最大循环次数，默认值为200，超过该次数则报错。

WITH 中的数据修改语句

在WITH子句中使用数据修改命令INSERT、UPDATE、DELETE。这允许用户在同一个查询中执行多个不同操作。示例如下所示：

版权所有 © 华为云计算技术有限公司

```
WITH moved_tree AS (
    DELETE FROM tree
    WHERE parentid = 4
    RETURNING * )
 INSERT INTO tree_log
 SELECT * FROM moved_tree;
```

上述查询示例实际上从tree把行移动到tree_log。WITH中的DELETE删除来自tree的指定行，以它的RETURNING子句返回它们的内容，并且接着主查询读该输出并将它插入到tree_log。

WITH子句中的数据修改语句必须有RETURNING子句，用来返回RETURNING子句的输出，而不是数据修改语句的目标表，RETURNING子句形成了可以被查询的其余部分引用的临时表。如果一个WITH中的数据修改语句缺少一个RETURNING子句，则它形不成临时表并且不能在剩余的查询中被引用。

如果声明了RECURSIVE关键字，则不允许在数据修改语句中进行递归自引用。在某些情况中可以通过引用递归WITH的输出来绕过这个限制，例如：

```
WITH RECURSIVE included_parts(sub_part, part) AS (
    SELECT sub_part, part FROM parts WHERE part = 'our_product'
  UNION ALL
    SELECT p.sub_part, p.part
    FROM included_parts pr, parts p
    WHERE p.part = pr.sub_part
  )
DELETE FROM parts
  WHERE part IN (SELECT part FROM included_parts);
```

这个查询将会移除一个产品的所有直接或间接子部件。

WITH子句中的子语句与主查询同时执行。因此，在使用WITH中的数据修改语句时，指定更新的顺序实际是以不可预测的方式发生的。所有的语句都使用同一个快照中执行，语句的效果在目标表上不可见。这减轻了行更新的实际顺序的不可预见性的影响，并且意味着RETURNING数据是在不同WITH子语句和主查询之间传达改变的唯一方法。

本示例中外层SELECT可以返回更新之前的数据：

```
WITH t AS (
    UPDATE tree SET id = id + 1
    RETURNING * )
SELECT * FROM tree;
```

本示例中外部SELECT将返回更新过的数据：

```
WITH t AS (
UPDATE tree SET id = id + 1
    RETURNING * )
SELECT * FROM t;
```

不支持在单个语句中更新同一行两次。这种语句的效果是不可预测的。如果只有一个修改发生了，但却不容易（有时也不可能）预测哪一个发生了修改。

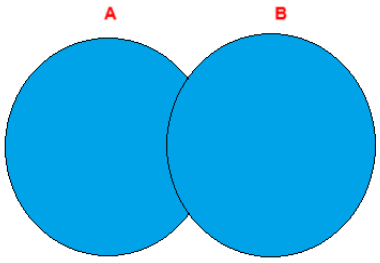
6.5 UNION 操作符的使用

在SQL中，UNION是一个非常强大的功能，UNION操作符用于合并两个或多个SELECT语句的结果集。合并时，两个表对应的列数和数据类型必须相同，并且相互对应。各个SELECT语句之间使用UNION或UNION ALL关键字分隔。

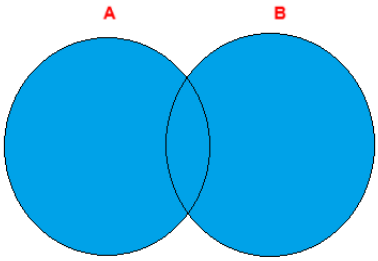
UNION在合并两个或多个表时会执行去重操作，而UNION ALL则直接将两个或者多个结果集合并，不执行去重。 另外，执行去重会消耗大量的时间，因此，在一些实际应

 版权所有 © 华为云计算技术有限公司

用场景中，如果通过业务逻辑已确认了两个集合不存在重复数据时，可直接用UNION ALL替代UNION，以便提升性能。



UNION操作符返回两个查询的结果集的并集，去除重复记录。



UNION ALL操作符返回两个查询的结果集的并集。对于两个结果集的重复部分，不去重。

语法格式

```
SELECT column,... FROM table1 UNION [ALL]SELECT column,... FROM table2
```

示例

步骤1 创建学生信息表student（ID、姓名、性别、学校）。

```
SET current_schema=public;
DROP TABLE IF EXISTS student;
CREATE table student(
sId VARCHAR(10) NOT NULL,
sname VARCHAR(10) NOT NULL,
sgender VARCHAR(10) NOT NULL,
sschool VARCHAR(10) NOT NULL);
```

步骤2 给表student插入数据。

```
INSERT INTO student VALUES('s01' , 'ZhaoLei' , 'male', 'NENU');
INSERT INTO student VALUES('s02' , 'QianDian' , 'male', 'SJTU');
INSERT INTO student VALUES('s03' , 'SunFeng' , 'male', 'Tongji');
INSERT INTO student VALUES('s04' , 'LiYun' , 'male', 'CCOM');
INSERT INTO student VALUES('s05' , 'ZhouMei' , 'female', 'FuDan');
INSERT INTO student VALUES('s06' , 'WuLan' , 'female', 'WHU');
INSERT INTO student VALUES('s07' , 'ZhengZhu' , 'female', 'NWAFU');
INSERT INTO student VALUES('s08' , 'ZhangShan' , 'female', 'Tongji');
```

步骤3 查看表student。

```
SELECT * FROM student;
```

回显如下：

```
 sid |   sname   | sgender | sschool
-----+-----------+---------+---------
 s02 | QianDian  | male    | SJTU
 s05 | ZhouMei   | female  | FuDan
 s08 | ZhangShan | female  | Tongji
 s03 | SunFeng   | male    | Tongji
 s06 | WuLan     | female  | WHU
 s01 | ZhaoLei   | male    | NENU
 s04 | LiYun     | male    | CCOM
 s07 | ZhengZhu  | female  | NWAFU
(8 rows)
```

步骤4 创建教师信息表teacher（ID、姓名、性别、学校）。

```
DROP TABLE IF EXISTS teacher;
CREATE table teacher(
```

 版权所有 © 华为云计算技术有限公司

```
tid VARCHAR(10) NOT NULL,
tname VARCHAR(10) NOT NULL,
tgender VARCHAR(10) NOT NULL,
tschool VARCHAR(10) NOT NULL);
```

步骤5 给表teacher插入数据。

```
INSERT INTO teacher VALUES('t01' , 'ZhangLei', 'male', 'FuDan');
INSERT INTO teacher VALUES('t02' , 'LiLiang', 'male', 'WHU');
INSERT INTO teacher VALUES('t03' , 'WangGang', 'male', 'Tongji');
```

步骤6 查询表teacher。

```
SELECT * FROM teacher;
```

```
 tid |  tname   |tgender| tschool
-----+----------+------+---------
 t03 | WangGang | male | Tongji
 t02 | LiLiang  | male | WHU
 t01 | ZhangLei | male | FuDan
(3 rows)
```

步骤7 使用UNION（合并且去重）获取学生和教师所在学校，并按学校名称首字母升序排序。

```
SELECT t.school  FROM (
    SELECT sschool AS school
     FROM student
     UNION
     SELECT tschool AS school
     FROM teacher
 ) t
 ORDER BY t.school ASC;
```

回显如下：

```
 school
--------
 CCOM
 FuDan
 NENU
 NWAFU
 SJTU
 Tongji
 WHU
(7 rows)
```

步骤8 使用UNION ALL（合并不去重）获取所有学生和教师所在学校，并按学校名称首字母升序排序。

```
SELECT t.school  FROM (
    SELECT sschool AS school
     FROM student
     UNION ALL
     SELECT tschool AS school
     FROM teacher
 ) t
 ORDER BY t.school ASC;
```

 版权所有 © 华为云计算技术有限公司

```
 school
--------
 CCOM
 FuDan
 FuDan
 NENU
 NWAFU
 SJTU
 Tongji
 Tongji
 Tongji
 WHU
 WHU
(11 rows)
```

步骤9 使用UNION ALL（合并带有WHERE子句SQL结果集）获取来自“Tongji”的学生和教师的所有信息，并按学生和教师的编号升序排序。

```
SELECT t.*  FROM  (
  SELECT Sid AS id,Sname AS name,Sgender AS gender,Sschool AS school
  FROM student
  WHERE Sschool='Tongji'
  UNION ALL
  SELECT Tid AS id,Tname AS name,Tgender AS gender,Tschool AS school
  FROM teacher
  WHERE Tschool='Tongji'
) t
  ORDER BY t.id ASC;
```

----结束

小结

在实际业务场景中，使用UNION和UNION ALL时需要注意以下几点：

- 左右两侧的SQL字段数量和字段类型需要保持一致。
- 业务需求是否需要考虑数据除重（合并前除重还是合并时除重）。
- 根据表中数据量的大小，需要对SQL的执行效率进行评估，从而考虑是否需要选择临时表进行过渡后再合并。
- 需要考虑SQL编写的复杂度，不能为了写SQL而写SQL，需要结合业务需求进行选择。

6.6 跨计算组数据读写

场景介绍

创建关联计算组用户后，用户提交的查询或修改（包括Insert、Delete、Update等）会在其关联的计算组上进行计算执行。当提交查询或修改的用户与需要查询和修改的基表在不同的计算组上时，数据需要在表所在的计算组和用户关联的计算组间进行查询和修改，此时优化器会生成一个跨计算组的查询或修改计划，保证用户关联的计算组可以查询或修改表的数据。

版权所有 © 华为云计算技术有限公司

图 6-1 跨计算组实现数据查询

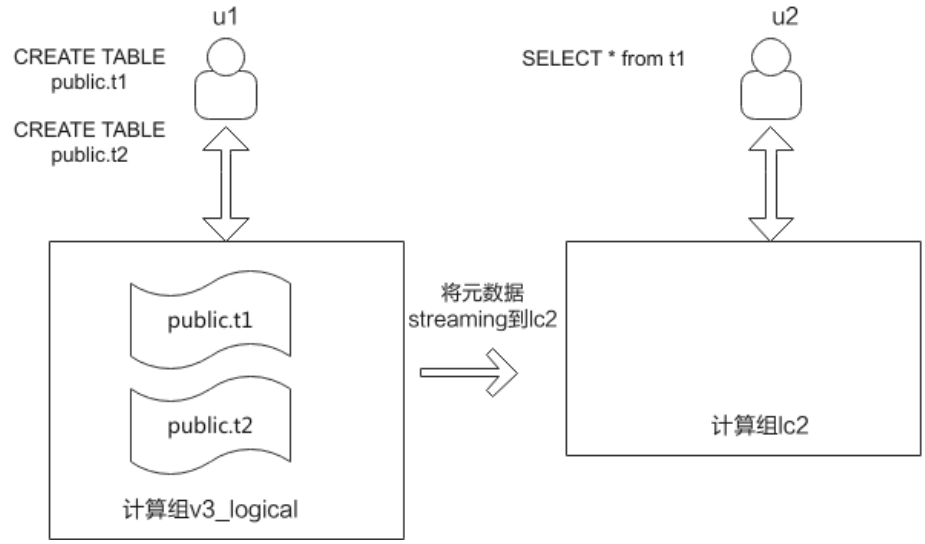
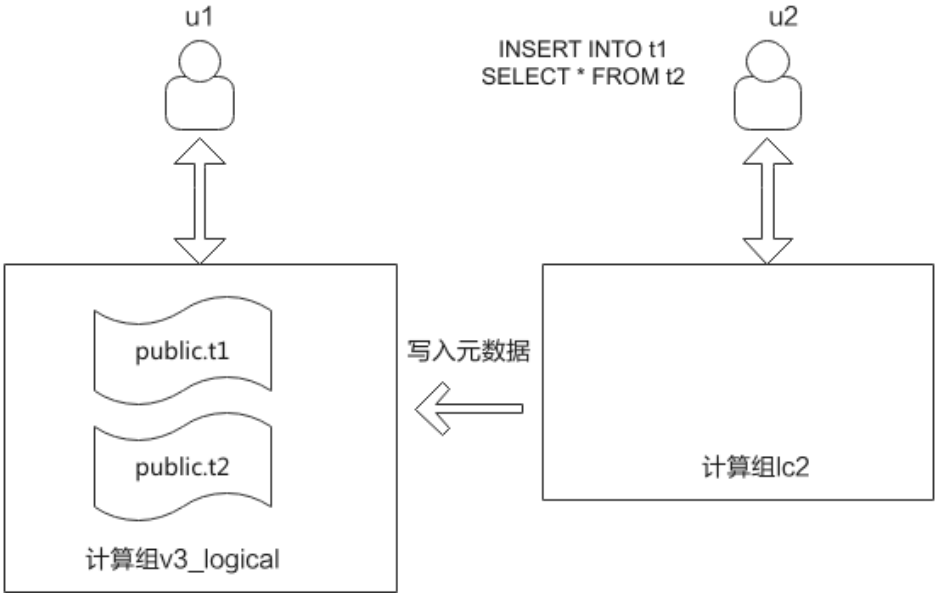


图 6-2 跨计算组实现数据写入



约束和限制

跨计算组写操作需要保证DDL与DML在CN上锁互斥，否则会存在并发写数据问题。

- 非分区表：在CN上默认存在主表锁互斥，无需额外配置。
- 分区表：如需支持跨计算组写操作，需确保在**behavior_compat_options**参数中设置disable_select_truncate_parallel选项。

操作步骤

步骤1 参见**创建DWS存算分离集群**，集群创建后默认会转换成计算组v3_logical。

版权所有 © 华为云计算技术有限公司

步骤2 通过节点扩容方式增加3个节点到弹性集群，再添加计算组lc2。

步骤3 创建用户u1，并关联计算组v3_logical。

```
CREATE USER u1 with SYSADMIN NODE GROUP "v3_logical" password  "Password";
```

步骤4 创建用户u2，并关联计算组lc2。

```
CREATE USER u2 with SYSADMIN NODE GROUP "lc2" password  "Password";
```

步骤5 以u1登录数据库，创建表t1和t2，并插入测试数据。

```
CREATE TABLE  public.t1
(
id integer not null,
data integer,
age integer
)
WITH (ORIENTATION =COLUMN, COLVERSION =3.0)
DISTRIBUTE BY ROUNDROBIN;

CREATE TABLE public.t2
(
id integer not null,
data integer,
age integer
)
WITH (ORIENTATION = COLUMN, COLVERSION =3.0)
DISTRIBUTE BY ROUNDROBIN;

INSERT INTO public.t1 VALUES (1,2,10),(2,3,11);
INSERT INTO public.t2 VALUES (1,2,10),(2,3,11);
```

步骤6 以u2登录数据库，执行以下命令查询t1和写入数据。

从结果可得出，实现用户u2跨计算组进行查询和写入数据的能力。

```
SELECT * FROM t1;
INSERT INTO t1 SELECT * FROM t2;
```

----结束

6.7 SQL on Hudi

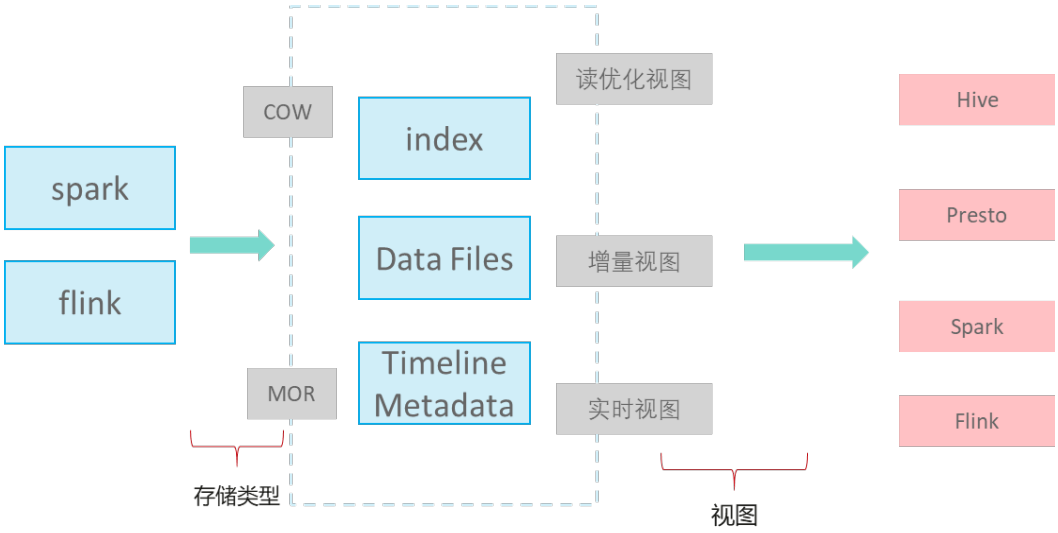
6.7.1 Hudi 简介

DWS的SQL on Hudi特性仅9.1.0.100及以上版本支持。

Apache Hudi表示Hadoop Upserts Deletes and Incrementals。用来管理Hadoop大数据体系下存储在DFS（Distributed File System，分布式文件系统）上大型分析数据集。

Hudi不是单纯的数据格式，而是一套数据访问方法（类似DWS存储的access层），在Apache Hudi 0.9版本，大数据的Spark，Flink等组件都单独实现各自客户端。Hudi的逻辑存储如下图所示：

版权所有 © 华为云计算技术有限公司



- 写入模式

  COW：写时复制，适合更新少的场景。

  MOR：读时复制，对于UPDATE&DELETE增量写delta log文件，分析时进行base和delta log文件合并，异步compaction合并文件。
- 存储格式

  index：对主键进行索引，默认是file group级别的bloomfilter。

  data files：base file + delta log file（主要面向对base file的update&delete）。

  timeline metadata：版本log的管理。
- 视图

  读优化视图：读取Compaction后生成的base file，未Compaction数据时效性有一定延迟（高效读取）。

  实时视图：读取最新的数据，在读取时进行Base file和Delta file合并（频繁update场景）。

  增量视图：类似CDC方式持续读取增量写入Hudi的数据（流批一体）。

6.7.2 使用 Hudi 前准备

前提条件

- 已完成创建OBS委托以及创建OBS数据源，具体可参见**管理OBS数据源**。
- 使用SQL on Hudi前需确保enable_dws_bigdata参数为打开状态。如果使用时出现类似“flight into error”报错，需联系技术支持开启该参数。

授权使用数据源

执行GRANT命令为用户授予使用数据源的权限：

```
GRANT USAGE ON FOREIGN SERVER server_name TO role_name;
```

示例：

给用户u1授予数据源hudi_server的访问权限：

```
GRANT USAGE ON FOREIGN SERVER hudi_server TO u1;
```


版权所有 © 华为云计算技术有限公司

授权使用外表

执行如下命令给用户授予使用外表的权限：

```
ALTER USER role_name USEFT;
```

示例：

给用户u1授予外表访问权限：

```
ALTER USER u1 USEFT;
```

6.7.3 Hudi 用户接口

实时视图与增量视图查询

DWS提供了类似spark-sql风格的表级参数，用于支持实时视图和增量视图。

具体参数说明如下，其中SCHEMA.FOREIGN_TABLE需要替换为实际的schema和外表名。

表 6-2 实时视图与增量视图查询的参数

参数	取值	说明
hoodie.SCHEMA.FOREIGN_TABLE.consume.mode	SNAPSHOT	查询实时视图。
	INCREMENTAL	查询增量视图。
hoodie.SCHEMA.FOREIGN_TABLE.consume.start.timestamp	hudi时间戳	指定增量同步的起始commit。
hoodie.SCHEMA.FOREIGN_TABLE.consume.ending.timestamp	hudi时间戳	指定增量同步的结束commit，不指定则采用最新commit。

说明

- 以上参数支持使用set命令设置，并且仅在当前SESSION中有效。可使用reset命令恢复缺省值。
- 可以通过系统函数pg_catalog.pg_show_custom_settings()来查询相关参数的设置详情。
- 查询MOR表的增量视图时，需要使用where条件过滤_hoodie_commit_time字段，避免读取到未合并的不符合条件的log文件数据。COW表无需该操作。

Hudi 外表信息查询与自动同步任务

DWS提供一系列系统函数来实现Hudi外表信息获取、创建Hudi自动同步任务等功能。其中，Hudi自动同步任务实现了从Hudi外表周期性同步数据到DWS内表功能。

版权所有 © 华为云计算技术有限公司

表 6-3 Hudi 系统函数

名称	类型	功能
pg_show_custom_settings()	内置函数	查询Hudi外表参数设置详情。
hudi_get_options(regclass)	内置函数	查询Hudi外表的属性信息（hoodie.properties）。
hudi_get_max_commit(regclass)	内置函数	获取当前Hudi外表最新commit的时间戳。
hudi_sync_task_submit(regclass, regclass)	内置函数	提交Hudi自动同步任务。
hudi_sync_task_submit(regclass, regclass, text, text)		
hudi_show_sync_state()	内置函数	获取Hudi自动同步任务的同步状态。
hudi_sync(regclass, regclass)	存储过程	Hudi自动同步任务调用入口。
hudi_sync_custom(regclass, regclass, text)	存储过程	Hudi自动同步任务调用入口，支持用户自定义目标表和数据源表的字段同步对应关系。
hudi_set_sync_commit(regclass, regclass, text)	内置函数	设置Hudi自动同步任务首次同步的起点时间戳，避免在已经同步了部分数据的情况下，重新同步已有数据。
hudi_set_sync_commit(text, text)		设置Hudi自动同步任务下一次同步的起点时间戳，可以用于重复同步历史数据或者跳过某些数据。

6.7.4 创建 Hudi 数据（外表）

DWS通过外表方式访问OBS上的Hudi数据，与一般外表相比，Hudi外表无需额外设置更多的参数，只需指定format参数为“hudi”，即可指定数据源文件的格式。具体参见**CREATE FOREIGN TABLE (SQL on OBS or Hadoop)**。

获取 MRS 上表定义

DWS Hudi外表为只读，因此在创建外表之前需要明确目标数据定义的字段及字段类型情况。

例如，对于MRS上的Hudi表，可以使用spark-sql来查询原始表定义：

```
SHOW create table rtd_mfdt_int_currency_t;
```

编写 DWS 表定义

复制MRS表所有列的定义，创建hudi外表，进行适当的类型转换以适配DWS语法：

```
CREATE FOREIGN TABLE rtd_mfdt_int_currency_ft(
_hoodie_commit_time text,
```


版权所有 © 华为云计算技术有限公司

```
_hoodie_commit_seqno text,
_hoodie_record_key text,
_hoodie_partition_path text,
_hoodie_file_name text,
...
)SERVER hudi_server OPTIONS (
foldername '/erpgc-test-01/s000/u1/rtd_mfdt_int_currency_t/',
format 'hudi',
encoding 'utf-8'
)distribute by roundrobin;
```

其中，foldername为hudi数据在OBS上存储路径，对应MRS中Spark-sql表定义中的LOCATION，末尾要以“/”结尾。

6.7.5 Hudi 任务同步

创建 Hudi 任务

迁移场景

如果DWS表已经通过CDL导入数据，改为用SQL on Hudi方式迁移数据。或者使用CDM做全量初始化后，继续使用SQL on Hudi方式同步增量数据。

步骤1 创建hudi.hudi_sync_state同步状态表，需要管理员权限。

```
SELECT pg_catalog.create_hudi_sync_table();
```

通常情况下，每个数据库中只创建一次hudi.hudi_sync_state。

步骤2 设置同步进度，用户需要有同步目标表的INSERT和UPDATE权限、Hudi外表的SELECT权限，否则无法正常设置同步进度。

```
SELECT hudi_set_sync_commit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE', 'LATEST_COMMIT');
```

其中：

- SCHEMA.TABLE，表示同步数据的目标表名，schema为模式名。
- SCHEMA.FOREIGN_TABLE，表示OBS外表名，schema为模式名。
- LATEST_COMMIT，表示已同步的Hudi数据时间截点。

示例：目标表public.in_rel，已经同步hudi的数据到20220913152131，切换到SQL on Hudi方式从OBS外表hudi_read1中继续导出数据。

```
SELECT hudi_set_sync_commit('public.in_rel', 'public.hudi_read1', '20220913152131');
```

步骤3 提交Hudi同步任务。

```
SELECT hudi_sync_task_submit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE');
```

例如：目标表public.in_rel，切换到SQL on Hudi方式从OBS外表hudi_read1中继续导出数据。

```
SELECT hudi_sync_task_submit('public.in_rel', 'public.hudi_read1');
```

----结束

新建场景

DWS表为空，第一次从Hudi同步数据，可执行以下命令直接创建任务。

```
SELECT hudi_sync_task_submit('SCHEMA.TABLE', 'SCHEMA.FOREIGN_TABLE');
```


版权所有 © 华为云计算技术有限公司

查询 Hudi 同步任务

查询Hudi同步任务。

语法：

```
SELECT * FROM pg_task_show('SQLonHudi');
```

查询结果中的**task_id**是Hudi同步任务唯一标识。

示例：

```
SELECT * FROM pg_task_show('SQLonHudi');
              task_id                 |
what                                                                                             | category_id | userid | is_broken |
interval  | time_cons |           start_time            | end_time | parameter | task_rank |         next_start_time
|          next_end_time          | last_log | failure_times
--------------------------------------
+-----------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+-------------+--------+-----------+------------+-----------
+------------------------------+----------+-----------+-----------+-------------------------------
+------------------------------+----------+---------------
 64d257e9-1e9b-0d00-3ce3-7e61b5e0fffe | call pg_catalog.hudi_sync_custom('public.hudi_read_target',
'public.hudi_read101', '{"_hoodie_commit_seqno" : "_hoodie_commit_seqno", "id" : "id", "ts" : "ts",
"long_field" : "ts"}'); | SQLonHudi   |      10 | f          | '00:00:10' |            | 2023-08-08 22:58:15.846903+08
|           |            |          5 | 2023-08-08 22:58:15.846903+08 | 2023-08-08 22:58:24.846903+08 |
|             0
(1 row)
```

暂停 Hudi 同步任务

通过指定的task_id，暂停对应的Hudi同步任务。

语法：

```
SELECT pg_task_pause('task_id');
```

示例：

暂停task_id为64479410-a04c-0700-d150-3037d700fffe的同步任务。

```
SELECT pg_task_pause('64479410-a04c-0700-d150-3037d700fffe');
```

恢复 Hudi 同步任务

通过指定的task_id，恢复对应的Hudi同步任务。

语法：

```
SELECT pg_task_resume('task_id');
```

示例：

恢复task_id为64479410-a04c-0700-d150-3037d700fffe的同步任务。

```
SELECT pg_task_resume('64479410-a04c-0700-d150-3037d700fffe');
```

复位连续失败的 Hudi 同步任务

通过指定连续失败Hudi任务的task_id，复位连续失败的Hudi同步任务。

连续失败次数大于等于10的任务会自动暂停，需手动调用pg_task_resume()函数复位。

```
SELECT pg_task_resume('task_id');
```

删除 Hudi 同步任务

通过指定的task_id，删除对应的Hudi同步任务。

语法：

```
SELECT pg_task_remove('task_id');
```

示例：

删除task_id为64479410-a04c-0700-d150-3037d700fffe的同步任务。

```
SELECT pg_task_remove('64479410-a04c-0700-d150-3037d700fffe');
```

查询历史 Hudi 同步任务信息

使用视图hudi_sync_state_history_view查询Hudi历史同步任务信息，该视图仅9.1.0及以上集群版本支持。

```
SELECT * FROM pg_catalog.hudi_sync_state_history_view;
```

表 6-4 hudi_sync_state_history_view 字段

名称	类型	描述
task_id	text	任务ID。
target_tbl	text	同步目标表名。
source_ftbl	text	同步源表名（外表）。
latest_commit	text	最近一次同步成功的时间戳。
latest_sync_count	bigint	最近一次同步成功的行数。
latest_sync_start	timestamp with time zone	最近一次同步任务开始的时间。
latest_sync_end	timestamp with time zone	最近一次同步任务结束的时间。
hudi_flushdisk_time	text	hudi文件落盘时间。

查询 Hudi 同步任务状态

使用函数hudi_show_sync_state()查询Hudi同步任务状态。

```
SELECT * FROM hudi_show_sync_state();
```

6.7.6 Hudi 外表查询

Hudi外表可以直接查询数据，默认查询实时视图，也可以通过设置增量查询参数实现增量视图查询。

版权所有 © 华为云计算技术有限公司

说明

DWS的hudi外表不支持读取使用了Schema演进特性的hudi表，如果读取Schema演进的hudi表，会导致修改列无数据（全为null）或者有报错的情况。（Schema演进，即Schema Evolution，是指在数据表的生命周期中，允许用户能够方便地修改Hudi表的当前Schema，以适应数据需求的变化。这包括添加、删除或修改表中的字段。）

hudi 外表增量查询

通过设置增量查询参数，进行hudi外表增量查询。

```
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.mode=incremental;
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.start.timestamp=起始时间戳;
SET hoodie.SCHEMA.FOREIGN_TABLE.consume.ending.timestamp=结束时间戳;
SELECT * FROM SCHEMA.FOREIGN_TABLE;
```

示例：

查询MOR hudi外表public.rtd_mfdt_int_currency_ft从20221207164617到20221207170234之间的增量数据。

```
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.mode=incremental;
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.start.timestamp=20221207164617;
SET hoodie.public.rtd_mfdt_int_currency_ft.consume.ending.timestamp=20221207170234;
SELECT * FROM public.rtd_mfdt_int_currency_ft where _hoodie_commit_time>20221207164617 and
_hoodie_commit_time<=20221207170234;
```

查询 hudi 外表已设置的增量参数

通过函数pg_show_custom_settings()查看hudi外表已设置的增量参数，检查参数设置是否正确。

```
SELECT * FROM pg_show_custom_settings();
```

查询 hudi 外表属性（hoodie.properties）

查询OBS上hudi数据的外表属性hoodie.properties。

语法：

```
SELECT * FROM hudi_get_options('SCHEMA.FOREIGN_TABLE');
```

示例：查询当前schema下的OBS外表rtd_mfdt_int_currency_ft的hudi属性。

```
SELECT * FROM hudi_get_options('rtd_mfdt_int_currency_ft');
```

查询 hudi 外表最大时间线

查询OBS上hudi数据最大时间线，即最新的提交记录。

语法：

```
SELECT * FROM hudi_get_max_commit('SCHEMA.FOREIGN_TABLE');
```

示例：查询当前schema下的OBS外表rtd_mfdt_int_currency_ft的最大时间线。

```
SELECT * FROM hudi_get_max_commit('rtd_mfdt_int_currency_ft');
```

6.7.7 支持访问 MRS 上的 Hudi 表

SQL on Hudi支持对存储在MRS上的hudi表进行访问。

版权所有 © 华为云计算技术有限公司

> **说明**
>
> DWS的hudi外表不支持读取使用了Schema演进特性的hudi表。（Schema演进（Schema Evolution）是指在数据表的生命周期中，允许用户能够方便地修改Hudi表的当前Schema，以适应数据需求的变化。这包括添加、删除或修改表中的字段。）

前提条件

已完成创建MRS数据源，具体可参见**管理MRS数据源**。

> **说明**
>
> SQL on Hudi读取存储在MRS上的hudi表，在使用上仅创建数据源与OBS不同，其他无差异。

对接多套 MRS 集群

通常情况下由于JDK的限制，同一JVM同时仅能保存一份kerberos配置文件信息，导致一套DWS集群无法同时通过SQL on Hudi并发访问多套MRS集群上的hudi表。但通过执行以下操作可访问多套MRS集群上的hudi表。

步骤1 获取各MRS集群已下载客户端中的krb5.conf文件。

步骤2 任取一套MRS集群的krb5.conf文件作为待合并文件，以下简称集群A。

步骤3 将集群B的KDC域信息添加到集群A配置文件realms中。

> **注意**
>
> 实际操作时需要将集群realms或者domain_realm中实际的KDC域信息进行合并。

示例：

```
[realms]
CLUSTER.A.COM = {
admin_server =  ClusterA_SERVER_IP:PORT
kdc = ClusterA_KDC_IP:PORT
kdc = ClusterA_KDC_IP:PORT
}
CLUSTER.B.COM = {
admin_server = ClusterB_SERVER_IP:PORT
kdc = ClusterB_KDC_IP:PORT
kdc = ClusterB_KDC_IP:PORT
}
```

步骤4 将集群B的域信息添加到集群A配置文件domain_realm中。

示例：

```
[domain_realm]
.cluster.a.com = CLUSTER.A.COM
.cluster.b.com = CLUSTER.B.COM
```

步骤5 将合并后的krb5.conf配置文件更新到各集群各个节点原路径下替换原krb5.conf文件。

----结束

 版权所有 © 华为云计算技术有限公司

6.8 SQL on Iceberg

Iceberg 简介

Iceberg是一种开放的数据湖表格式，可以基于Iceberg快速地在HDFS或OBS上构建高性能数据湖存储服务，为数据处理提供可扩展、高性能、易用的数据表管理解决方案。

DWS的SQL on Iceberg特性仅9.1.2及以上集群版本支持。

Iceberg 外表支持的功能

DWS通过创建Iceberg格式外表或者使用external schema访问OBS或HDFS上的Iceberg数据。当前Iceberg外表支持以下能力：

- 支持访问OBS或HDFS上的Iceberg数据。
- 提供查询常用Iceberg表信息的函数，包括snapshot、tag/branch、元信息等。
- 支持时间旅行（通过指定snapshot或tag查询历史数据）。
- 支持的catalog为lakeformation、hive和hadoop。
- 支持谓词下推、分区剪枝等查询优化。

Iceberg 外表的功能约束

- 不支持写Iceberg表。
- 手动创建Iceberg外表，需要保证外表字段名与Iceberg schema中定义的字段名保持一致，否则可能导致字段查询结果为空。
- 对Iceberg数据文件格式有要求如下（其他格式组合不支持）：
  - write.format.default=parquet且write.delete.format.default=parquet
  - write.format.default=parquet且write.delete.format.default=avro
- 仅支持Iceberg v1/v2表，不支持v3表。不支持读复杂类型（map、struct、array嵌套）。
- Iceberg外表不支持创建为分区表，外表自动识别分区字段。
- 默认情况下不支持schema演进，除非创建外表指定cow_improve 为off，或者对external schema外表设置behavior_foreign_options=lf_use_bigdata。
- 使用hivemetastore+hdfs external schema外表时，要求schema和hdfs server的配置文件路径一致。
- 对于存放在HDFS上的Iceberg表，仅在HDFS集群开启kerberos认证后支持读取。

Iceberg 表服务函数

DWS提供11个系统函数查询Iceberg表元数据信息。所有函数接受单个regclass参数（Iceberg外表OID）。具体可参考《SQL语法参考》中“Iceberg表服务函数”章节。

1. gs_iceberg_snapshots()，查看Iceberg表快照历史记录。
2. gs_iceberg_schemas()，查看Iceberg表DDL信息。
3. gs_iceberg_partition_specs()，查看Iceberg表分区键。

版权所有 © 华为云计算技术有限公司

4. gs_iceberg_refs()，查看Iceberg表快照引用信息。
5. gs_iceberg_manifests()，查看Iceberg表清单文件。
6. gs_iceberg_partitions()，查看Iceberg表分区统计信息。
7. gs_iceberg_manifest_entries()，查看Iceberg表清单文件状态条目。
8. gs_iceberg_files()，查看Iceberg表数据文件信息。
9. gs_iceberg_history()，查看Iceberg表时间线历史。
10. gs_iceberg_metadata_log_entries()，查看Iceberg表元数据日志条目。
11. gs_iceberg_pos_delete_files()，查看Iceberg表删除记录。

使用 DWS 外表访问 Iceberg 数据

步骤1 创建obs_server。

```
CREATE SERVER obs_server FOREIGN DATA WRAPPER dfs_fdw OPTIONS
(address 'obs.xxx.com', type 'obs', encrypt 'on', access_key 'xxx', secret_access_key 'xxx');
```

步骤2 根据原始Iceberg定义确认字段名和类型，创建Iceberg格式外表。

```
CREATE FOREIGN TABLE iceberg_all_types (
id              INT,
f_boolean       BOOLEAN,
f_short         SMALLINT,
f_byte          TINYINT,
f_integer       INT,
f_long          BIGINT,
f_float         FLOAT4,
f_double        FLOAT8,
f_date          DATE,
f_timestamp     TIMESTAMP,
f_char          TEXT,
f_varchar       TEXT,
f_string        TEXT,
f_binary        BYTEA,
f_decimal       NUMERIC(10,2)
) SERVER obs_server
OPTIONS (encoding 'utf-8',
format 'iceberg',
foldername '/user/hive/warehouse/test_db.db/iceberg_all_types/',
cow_improve 'on')
DISTRIBUTE BY ROUNDROBIN;
```

----结束

通过设置表级参数查询历史版本数据（时间旅行）

通过在查询外表前设置表级GUC参数“iceberg.schema名.外表名.version”或“iceberg.schema名.外表名.timestamp”，来指定具体的branch/tag或者时间戳。具体的参数取值可通过gs_iceberg_refs函数查询。

⚠ 注意

“iceberg.schema名.外表名.version”与“iceberg.schema名.外表名.timestamp”不能同时指定。

- 支持timestamp指定毫秒时间戳。

```
SET iceberg.public.iceberg_mor_timetravel.timestamp=1784944184903;
SELECT * FROM iceberg_mor_timetravel;
```

 版权所有 © 华为云计算技术有限公司

- 支持timestamp指定为带格式的时间戳。
```
SET iceberg.public.iceberg_mor_timetravel.timestamp='2026-07-25 09:29:47.456+08';
SELECT * FROM iceberg_mor_timetravel;
```
- 支持version指定tag。
```
RESET iceberg.public.iceberg_mor_timetravel.timestamp;//version和timestamp不支持同时设置，因此设置其中一个参数前，确保另一个参数处于未设置状态
SET iceberg.public.iceberg_mor_timetravel.version='release-2026-08';
SELECT * FROM iceberg_mor_timetravel;
```
- 支持version指定branch。
```
SET iceberg.public.iceberg_mor_timetravel.version='latest-branch';
SELECT * FROM iceberg_mor_timetravel;
```

 版权所有 © 华为云计算技术有限公司

7 DWS 排序规则

排序规则(collation)是在字符集中指定数据排序顺序及对数据进行分类的规则。排序规则支持不再受限于数据库的LC_COLLATE和LC_CTYPE设置创建后就不能更改的约束。

概述

一种可排序数据类型的每一种表达式都有一个排序规则（系统内部的可排序数据类型可以是text、varchar和char等字符类型。用户定义的基础类型也可以被标记为可排序的，并且在一种可排序数据类型上的域也是可排序的）。如果该表达式是一个列引用，该表达式的排序规则就是列所定义的排序规则。如果该表达式是一个常量，排序规则就是该常量数据类型的默认排序规则。更复杂表达式的排序规则根据其输入的排序规则得来。

排序规则组合原则

- 当表达式的collation未指定时，则认为是默认的排序规则default，它表示数据库的区域设置。表达式的collation也可能是不确定的，此时，排序操作和其他不确定的排序规则的操作就会失败。
- 对于函数或操作符调用，其排序规则将通过检查所有参数的collation来决定。如果该函数或操作符调用的结果是一种可排序的数据类型，若有外层表达式要用到排序规则，那么该外层的表达式将继承对应函数和操作符所调用结果集的排序规则。
- 表达式的排序规则派生可以是显式或隐式。该区别会影响多个不同的排序规则出现在同一个表达式中时如何对collation进行组合。当执行语句使用COLLATE子句时，将发生显式派生，否则为隐式派生。当多个排序规则组合时，规则如下：
  - 如果输入表达式中存在显式COLLATE派生，则在输入表达式之间的所有显式派生的COLLATE必须相同，否则将产生冲突错误。如果存在显式COLLATE，那它就是排序规则组合的结果。
  - 如果不存在显式COLLATE，那所有输入表达式必须具有相同的隐式COLLATE或默认COLLATE。如果存在非默认COLLATE，那它就是排序规则组合的结果。否则，结果是默认COLLATE。
  - 如果在输入表达式之间存在多个冲突的非默认COLLATE，则组合被认为是具有不确定排序规则，这并非一种错误。如果被调用的函数或表达式需要用到排序规则，运行时将产生排序规则未知的错误。
- CASE表达式中，比较行为使用的规则以WHEN子句中的COLLATE设置为准。
- 显示COLLATE的派生仅在当前查询（CTE或SUBQUERY）中生效，查询外则降为隐式派生。

 版权所有 © 华为云计算技术有限公司

排序规则使用建议

- 同一条查询语句中，避免使用多种排序规则，可能导致非预期的结果集。
- 使用collate子句指定排序规则时，避免连续使用多个collate子句变更排序规则。

大小写不敏感排序规则支持

从集群8.1.3版本开始，DWS增加内置排序规则case_insensitive，即对字符类型的大小写不敏感行为（如排序、比较、哈希）。

约束条件：

- 支持字符类型：char/character/nchar、varchar/character varying/varchar2/nvarchar2/clob/text。
- 不支持字符类型：“char”和name。
- 不支持的编码：PG_EUC_JIS_2004、PG_MULE_INTERNAL、PG_LATIN10、PG_WIN874。
- 不支持CREATE DATABASE时指定到LC_COLLATE。
- 不支持正则表达式。
- 不支持字符类型的record比较（如record_eq）。
- 不支持时序表。
- 不支持倾斜优化。
- 不支持RoughCheck优化。

示例

语句中显式指定COLLATE子句。

```
SELECT 'a' = 'A', 'a' = 'A' COLLATE case_insensitive;
 ?column? | ?column?
----------+----------
 f        | t
(1 row)
```

建表时指定列属性为case_insensitive。

```
CREATE TABLE t1 (a text collate case_insensitive);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
\d t1
            Table "public.t1"
 Column | Type |         Modifiers
--------+------+---------------------------
 a      | text | collate case_insensitive

INSERT INTO t1 values('a'),('A'),('b'),('B');
INSERT 0 4
```

建表时指定，查询时无需指定。

```
SELECT a, a='a' FROM t1;
 a | ?column?
---+----------
 A | t
 B | f
 a | t
 b | f
(4 rows)
```


版权所有 © 华为云计算技术有限公司

```
SELECT a, count(1) FROM t1 GROUP BY a;
 a | count
---+-------
 a |     2
 B |     2
(2 rows)
```

CASE表达式，以WHEN子句中的COLLATE设置为准。

```
SELECT a,case a when 'a' collate case_insensitive then 'case1' when 'b' collate "C" then 'case2' else 'case3'
end FROM t1;
 a | case
---+-------
 A | case1
 B | case3
 a | case1
 b | case2
(4 rows)
```

跨子查询隐式派生。

```
SELECT * FROM (SELECT a collate "C" from t1) WHERE a in ('a','b');
 a
---
 a
 b
(2 rows)
SELECT * FROM t1,(SELECT a collate "C" from t1) t2 WHERE t1.a=t2.a;
ERROR:  could not determine which collation to use for string hashing
HINT:  Use the COLLATE clause to set the collation explicitly.
```

⚠ 注意

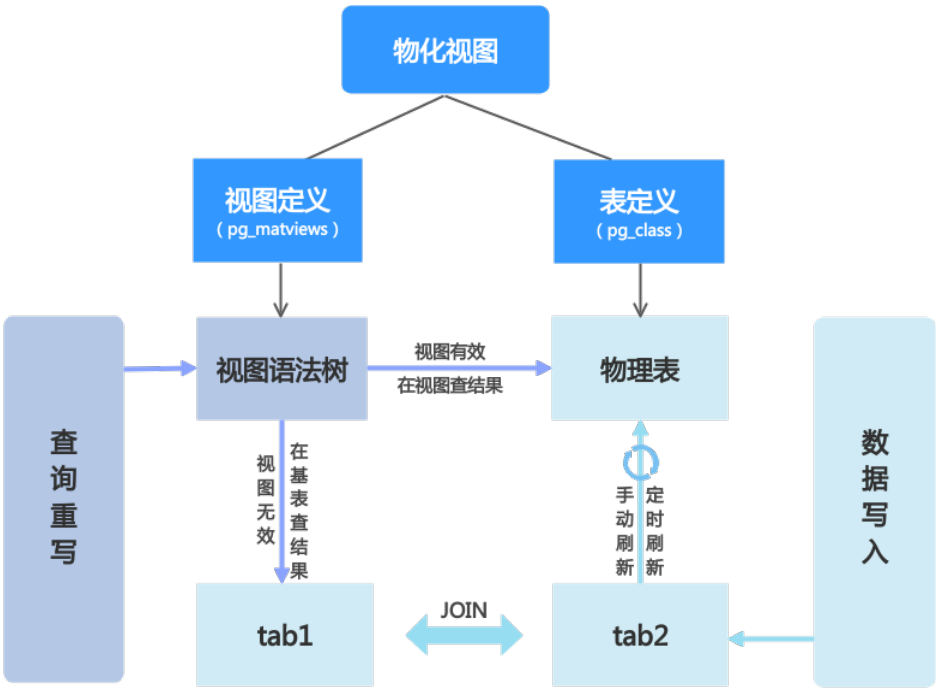
- 由于collate case_insensitive为不敏感排序，结果集不确定，再使用敏感排序筛选，会有结果集不稳定的问题，因此语句中避免出现敏感排序和不敏感排序混用。
- 使用collate case_insensitive指定字符类型为大小写不敏感后，性能较使用前会有所下降，因此性能敏感场景需谨慎评估后使用。

 版权所有 © 华为云计算技术有限公司

8 DWS 物化视图

8.1 物化视图概述

物化视图（Materialized View）是一种特殊的数据库对象。它将复杂计算的查询结果预先进行计算并持久化到存储介质中，业务查询时直接查询已经预计算好的数据，最终实现加速查询的目的。



物化视图主要有两大核心应用场景：透明加速和动态数据流。

表 8-1 物化视图两大核心应用场景

场景	说明	典型用例
透明加速	查询自动重写为物化视图，无需修改业务	报表加速、仪表盘

版权所有 © 华为云计算技术有限公司

场景	说明	典型用例
动态数据流	分层物化视图，通过增量计算实现实时数据流动	数仓分层(DWD→DWS→ADS)

物化视图中的基本概念

物化视图中的基本概念解释如下表：

表 8-2 物化视图基本概念

概念	说明
基表	物化视图定义中所依赖的表，支持DWS 内表、外表、分区表、视图或物化视图，不支持临时表。
失效	基表发生数据变化后，物化视图会被标记成失效状态。可直接查询，不能自动重写。
刷新	创建物化视图后，物化视图中存储的数据只反映创建时刻的状态。当基表中的数据发生变化时，需要通过刷新物化视图更新数据变化。 根据刷新模式可划分为：全量刷新和增量刷新。 ● 全量刷新：每次刷新重新计算物化视图中的数据。 ● 增量刷新：每次刷新将基表自上次刷新后数据变更的部分刷新到物化视图。
透明加速	查询重写自动将SQL定向到物化视图，无需修改业务。
动态数据流	分层嵌套创建多个物化视图，通过增量计算实现自底向上实时的动态数据流动。满足实时数仓的数据分层要求。

物化视图与普通视图及表的区别

- 普通视图是虚拟表，只存储视图的SQL定义，不存储查询结果的数据。在查询重写时会把对视图的查询转化为对基表的查询。
- 物化视图是实体表，既存储视图的SQL定义，也存储查询结果的数据。基表发生数据变化时，触发物化视图的失效。物化视图刷新时再重新计算为最新的数据。

查询重写时，根据用户SQL与物化视图SQL的匹配情况，选择直接从物化视图读取预计算的最新数据。

表 8-3 物化视图与普通视图及表的区别

对比维度	物化视图	普通视图	普通表
数据存储	✓	✗	✓
索引支持	✓	✗	✓

版权所有 © 华为云计算技术有限公司

对比维度	物化视图	普通视图	普通表
查询性能提升	✓	✗	✗
空间占用	✓	✗	✓
数据一致性	需刷新	实时一致	实时一致
业务逻辑简化	✓	✓	✗

物化视图的基本原理

1. 数据预计算：基于指定的基础查询语句（可包含关联、聚合、过滤等操作），在物化视图创建/刷新时，将查询结果一次性计算并存储为物理表，数据与基础表解耦。

2. 数据一致性：基础表数据发生变更（增删改）时，物化视图的预存数据会过期，需通过刷新机制同步基础表变化，平衡查询性能与数据实时性。

3. 透明加速：当用户SQL与物化视图SQL匹配时，查询重写会自动选择从物化视图读取数据。无需修改业务SQL，即可缩短复杂查询的响应时间，降低系统负载。

简单来说，物化视图是“空间换时间”的典型优化方案，以额外的存储开销和少量的刷新性能损耗，换取复杂查询的毫秒级响应。

异步物化视图和同步物化视图

创建物化视图后，当基表中的数据发生变化时，需要通过刷新物化视图更新数据变化。根据刷新方式，可分为异步物化视图和同步物化视图。

- 异步物化视图：基表数据变化后需要定期触发物化视图刷新，更新物化视图中数据。
  - 手动刷新：在业务数据修改后通过执行REFRESH MATERIALIZED VIEW 进行相应物化视图的刷新。
  - 自动刷新：在物化视图创建时指定刷新开始时间和刷新间隔。由数据库负责定期刷新。
- 同步物化视图：基表数据变化后物化视图数据自动变化。通过CREATE MATERIALIZED VIEW语法的以下两个参数设置：
  - ON COMMIT：基表修改的事务提交时，会触发基表上所有物化视图刷新。保证了基表与物化视图的数据强一致性。
  - ON STATEMENT：基表修改的DML语句执行时，触发基表上所有物化视图刷新。保证了基表与物化视图的数据强一致性。

表 8-4 异步物化视图和同步物化视图区别

对比维度	异步物化视图	同步物化视图
触发刷新时机	手动或定时轮询	基表发生数据修改时
对基表影响	无影响	拖慢基表入库性能
触发失效情况	基表写入后，物化视图失效	数据严格一致，物化视图不失效
适用场景	批量更新或微批更新	严格要求实时一致的场景

版权所有 © 华为云计算技术有限公司

物化视图的优势

- **声明式编程**

  用一条SQL实现数据的搬运，不再需要编写复杂的存储过程来实现两张表直接的数据搬运。索引、锁、数据一致性都自动考虑，对业务影响小。

- **灵活刷新**

  刷新方式：支持全量刷新和增量刷新。

  刷新模式：支持手动刷新，定时刷新和同步实时刷新。还支持按嵌套的链路级联刷新。

- **流式加工**

  将数仓的DWD（Data Warehouse Detail，明细层，负责清洗去重标准化原始数据）>DWB（Data Warehouse Base，基础层，整合关联宽表化）>DWS（Data WareHouse Service：汇总层，常用聚合指标）>ADS（Application Data Store，应用层，例如具体报表）都创建为物化视图。通过增量计算实现实时的数据流式动态加工。

- **透明加速**

  物化视图预先计算并存储查询结果，当发起查询时，可以查询物化视图直接返回结果，从而提高查询性能，且无需修改业务。

- **增效降本**

  多次执行复杂查询通常需要显著的CPU和内存资源。物化视图通过透明加速将复杂的重复计算重写为简单的查询物化视图的SQL，可显著降低集群负载。

- **瞬时物化**

  将物化视图进行分区，使用通用的分区自动管理功能，实现旧分区的自动淘汰，可减轻数据管理的负担。

- **冷热分仓**

  将物化视图进行分区，使用通用的分区自动转冷功能，实现旧分区的自动转冷。可自动降低存储成本。

物化视图相关语法

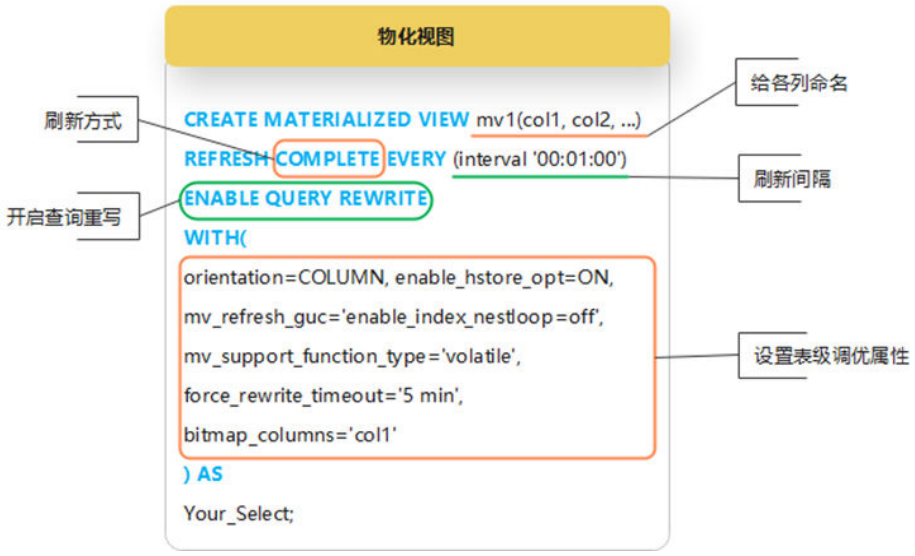
前提条件：

DWS的物化视图特性仅8.2.1.220及以上集群支持，使用物化视图需确保**enable_matview**设置为on。

- 创建物化视图。

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ [ DEFAULT ] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] |
[ CASCADE ALL ] ]
        [ COMPLETE | FAST ]
        [ [ ON DEMAND ] | [ ON STATEMENT ] | [ ON COMMIT ] ]
        [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```


版权所有 © 华为云计算技术有限公司



- 刷新物化视图。

```
REFRESH MATERIALIZED VIEW
{[schema.]materialized_view_name} [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] |
[ CASCADE BACKWARD ] | [ CASCADE ALL ] ]
```

- 删除物化视图。

```
DROP MATERIALIZED VIEW [ IF EXISTS ]
{[schema.]materialized_view_name} [, ...] [ CASCADE | RESTRICT ];
```

- 修改物化视图。

```
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }[ ENABLE | DISABLE ] QUERY
REWRITE;
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] |
[ CASCADE ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY
(interval) ] ];
ALTER MATERIALIZED VIEW [ IF EXISTS ] { materialized_view_name }
    REFRESH [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] |
[ CASCADE ALL ] ;
ALTER MATERIALIZED VIEW { materialized_view_name }
    OWNER TO new_owner;
ALTER MATERIALIZED VIEW { materialized_view_name }
    SET ( {storage_parameter = value} [, ... ] )
    | RESET ( storage_parameter [, ... ] );
```

物化视图语法具体参数说明，请参考**CREATE MATERIALIZED VIEW**和**ALTER MATERIALIZED VIEW**章节。

物化视图的应用场景

物化视图适用如下场景：

- 查询结果不经常改变，即基表稳定且执行频次高的查询。
- 查询包含非常耗时的操作，比如聚合、连接操作等。对于涉及多个表连接和聚合操作的查询，物化视图可以将这些耗时的操作预先计算并存储起来，从而避免在每次查询时都重新执行这些操作。
- 查询结果仅涉及表中的很小部分数据。

使用物化视图所需操作权限

- 创建物化视图时需要schema的CREATE权限和基表或列的SELECT权限。

版权所有 © 华为云计算技术有限公司

- 查询物化视图需要物化视图的SELECT权限。
- 刷新需要物化视图的REFRESH权限。
- 删除物化视图需要有DROP TABLE的权限。

物化视图使用限制

- 不支持包含临时表，包括全局临时表、volatile临时表和普通临时表。
- 不支持物化视图定义中的CTE包含除SELECT语句以外的其他DML。
- 不支持基表上存在脱敏策略和行级访问控制或者基表属于私有用户。脱敏策略和行级访问控制可能会导致结果集错误，而私有用户可能导致刷新物化视图失败等问题。
- 9.1.0.200及以上版本支持stable和volatile函数。物化视图无法感知函数内容变化，如果需要函数变化及时反馈到物化视图结果中，需执行REFRESH MATERIALIZED VIEW RESTRICT进行物化视图刷新或者跟随数据更新反馈到物化视图。
- 物化视图无法自动感知影响其查询执行的全局变量或会话参数变更，需要再次刷新物化视图更新结果。

物化视图使用建议

- 存储资源：物化视图会占用额外的存储空间，需要考虑磁盘容量。
- 刷新维护：自动或手动刷新物化视图会消耗系统资源，如果基础表数据变化频繁，刷新操作可能会影响系统性能。
- 数据一致性和实时性：异步物化视图中的数据可能不是实时的，即数据不会随着原始数据的更新而自动更新。如果基础表数据发生变化，视图中的数据可能会过时，需要定期刷新以保持数据一致性。
- 设计复杂性：物化视图的设计和创建需要仔细考虑预期的查询模式和数据访问模式，以便实现最佳的性能优化。

8.2 查询重写

8.2.1 概述

DWS不仅支持查询物化视图，还支持查询重写。物化视图查询重写，让用户在不改变业务情况下自动使用物化视图加速查询执行。通过查询重写功能，物化视图可以像索引一样添加或删除，不会影响业务。

使用场景

物化视图查询重写，能够避免查询中复杂的关联、聚集以及表达式运算操作。

- 文本匹配查询重写，查询与物化视图的定义基本一致（允许格式、注释有差异）。因为能够支持cte，窗口函数等复杂查询，复杂查询性能较差时可以通过定义与查询语句相同的物化视图，加快查询。
- 结构匹配查询重写，可以匹配多种语句，适合用于多表关联关系不变，过滤条件以及输出列变化的场景。

 版权所有 © 华为云计算技术有限公司

默认情况下，物化视图必须有效才能进行查询重写。也可以设置CREATE MATERIALIZED VIEW语法中force_rewrite_timeout参数，设置物化视图失效指定的时间内，依然可以强制重写。

物化视图查询重写方式

DWS支持以下两种方式的物化视图查询重写：

- 基于文本匹配查询重写

  当查询和物化视图的SQL或者语法树匹配时，可以使用物化视图进行查询改写。
- 基于结构匹配查询重写

  提取查询的特定结构进行重写。目前针对查询中投影、过滤条件、Join和聚合部分与物化视图进行匹配和重写。

设置物化视图查询重写

DWS通过GUC参数**mv_rewrite_rule**控制查询重写方式。

8.2.2 文本匹配查询重写

DWS支持文本匹配查询重写，包括JOIN、AGG、CTE、子查询等复杂查询。

使用约束

- 查询的子查询也可以使用物化视图进行文本匹配。
- 如果查询语句有ORDER BY，要求ORDER BY列在物化视图输出列中存在才可以进行重写。
- 支持查询和物化视图间有空格、回车、注释的差异。
- 物化视图输出列指定别名时，需要查询也指定别名才能匹配成功。
- 物化视图中的查询不能包含FOR UPDATE。

示例

以tpch Q1为例，基于Q1的复杂查询创建物化视图，查看查询计划时，显示语句已经被重写成了对物化视图的扫描。

创建LINEITEM表：

```
CREATE TABLE LINEITEM (
    L_ORDERKEY BIGINT NOT NULL,
    L_PARTKEY BIGINT NOT NULL,
    L_SUPPKEY BIGINT NOT NULL,
    L_LINENUMBER BIGINT NOT NULL,
    L_QUANTITY DECIMAL ( 15, 2 ) NOT NULL,
    L_EXTENDEDPRICE DECIMAL ( 15, 2 ) NOT NULL,
    L_DISCOUNT DECIMAL ( 15, 2 ) NOT NULL,
    L_TAX DECIMAL ( 15, 2 ) NOT NULL,
    L_RETURNFLAG CHAR ( 1 ) NOT NULL,
    L_LINESTATUS CHAR ( 1 ) NOT NULL,
    L_SHIPDATE DATE NOT NULL,
    L_COMMITDATE DATE NOT NULL,
    L_RECEIPTDATE DATE NOT NULL,
    L_SHIPINSTRUCT CHAR ( 25 ) NOT NULL,
    L_SHIPMODE CHAR ( 10 ) NOT NULL,
    L_COMMENT VARCHAR ( 44 ) NOT NULL
)
```

 版权所有 © 华为云计算技术有限公司

```
WITH (orientation = COLUMN)
DISTRIBUTE BY hash(L_ORDERKEY)
PARTITION BY RANGE(L_SHIPDATE) (
    PARTITION L_SHIPDATE_1 VALUES LESS THAN ( '1993-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_2 VALUES LESS THAN ( '1994-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_3 VALUES LESS THAN ( '1995-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_4 VALUES LESS THAN ( '1996-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_5 VALUES LESS THAN ( '1997-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_6 VALUES LESS THAN ( '1998-01-01 00:00:00' ),
    PARTITION L_SHIPDATE_7 VALUES LESS THAN ( '1999-01-01 00:00:00' )
);
```

创建物化视图：

```
CREATE MATERIALIZED VIEW mv_tpch_q1 enable query rewrite as
SELECT
        l_returnflag,
        l_linestatus,
        sum(l_quantity) AS sum_qty,
        sum(l_extendedprice) AS sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
        avg(l_quantity) AS avg_qty,
        avg(l_extendedprice) AS avg_price,
        avg(l_discount) AS avg_disc,
        count(*) AS count_order
FROM
        lineitem
WHERE l_shipdate <= DATE '1998-12-01' - INTERVAL '3 day'
GROUP BY l_returnflag,l_linestatus
ORDER BY sum_qty;

explain (costs off)
SELECT
        l_returnflag,
        l_linestatus,
        sum(l_quantity) AS sum_qty,
        sum(l_extendedprice) AS sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
        avg(l_quantity) AS avg_qty,
        avg(l_extendedprice) AS avg_price,
        avg(l_discount) AS avg_disc,
        count(*) AS count_order
FROM
        lineitem
WHERE l_shipdate <= DATE '1998-12-01' - INTERVAL '3 day'
GROUP BY l_returnflag,l_linestatus
ORDER BY sum_qty;
                QUERY PLAN
-----------------------------------------------
  id |              operation
 ----+---------------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |    ->  Sort
   3 |       ->  Seq Scan on mv_tpch_q1
```

查询除了ORDER BY部分如果与物化视图定义一致，也可以进行全文匹配。

```
CREATE TABLE t1 (a int,b int);
CREATE MATERIALIZED VIEW mv_t1 enable query rewrite AS SELECT sum(a), b FROM t1 GROUP BY b;

EXPLAIN (costs off) SELECT sum(a), b FROM t1 GROUP BY b ORDER BY 1;
            QUERY PLAN
--------------------------------------
  id |          operation
 ----+-----------------------------
   1 | ->  Streaming (type: GATHER)
   2 |    ->  Sort
   3 |       ->  Seq Scan on mv_t1
```

 版权所有 © 华为云计算技术有限公司

8.2.3 结构匹配查询重写

结构匹配查询重写不要求查询语句与物化视图定义完全一致，只要求查询语句的结果能够通过物化视图的结果改写得到，通过添加投影列、过滤条件、聚集列、排序列以及limit的方式，使重写后的结果与原始查询语句的结果一致。

DWS支持四种结果匹配查询重写：条件重写、投影重写、聚合重写和JOIN重写。

使用约束

- 结构匹配查询重写功能语句较为复杂，建议使用前联系技术支持。
- 只支持SELECT(条件)-PROJECTION(投影)-JOIN-GROUP BY (SPJG) 形式的查询。
- 查询中不能包含RELATION、VIEW、MATVIEW、FOREIGN TABLE之外的对象。
- 物化视图不能包含子查询。
- 物化视图不能包含CTE、窗口函数、集合操作、同义词、行级访问控制、FOR UPDATE语法。
- 查询可以包含limit offset/having/grouping sets，物化视图不能包含limit offset/having/grouping sets。
- 指定分区查询时，不支持结构匹配重写。
- distinct on不支持重写。
- 支持聚合上卷的聚合类型包含：sum/count/min/max/avg/roaringbitmap。

条件重写

物化视图和查询语句中都可能存在过滤条件。若满足以下条件可以进行条件重写：

- 查询有过滤条件，而物化视图没有过滤条件。该情况下，可以通过添加过滤条件的方式进行重写。
- 查询有过滤条件，物化视图也有过滤条件，但是查询的过滤条件所表示的范围是物化视图的过滤条件所表示的范围的子集。该情况下，同样可以通过添加过滤条件的方式，来减小物化视图数据的范围进行重写。
- 查询有过滤条件，物化视图也有过滤条件，但是物化视图的过滤条件都在查询的过滤条件中。该情况下，可以通过添加查询比物化视图多出来的过滤条件来进行重写。

条件重写示例：

创建基表t1及物化视图mv1_t1，mv2_t1，mv3_t1。

```
CREATE TABLE t1(a int, b int, c int);
CREATE MATERIALIZED VIEW mv1_t1 enable query rewrite as select c from t1;
CREATE MATERIALIZED VIEW mv2_t1 enable query rewrite as select a, b, c from t1 where a > 2;
CREATE MATERIALIZED VIEW mv3_t1 enable query rewrite as select a, b, c from t1 where a > 1 and b < 2;
```

场景一：查询有过滤条件，物化视图没有过滤条件。

```
EXPLAIN (costs off, verbose off, nodes off) select c from t1 where c > 2;
                QUERY PLAN
-------------------------------------------------
 id |          operation
 ----+-------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |    ->  Seq Scan on mv1_t1

 Predicate Information (identified by plan id)
```

 版权所有 © 华为云计算技术有限公司

```
-----------------------------------------------
  2 --Seq Scan on mv1_t1
        Filter: (c >= 3)
```

场景二：查询的过滤条件是物化视图的过滤条件所表示的范围的子集。

```
EXPLAIN (costs off, verbose on, nodes off) select a from t1 where a > 3;
                 QUERY PLAN
-----------------------------------------------
 id |          operation
----+-----------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on mv2_t1

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on mv2_t1
        Filter: (a >= 4)
```

场景三：物化视图的过滤条件都在查询的过滤条件中。

```
explain (costs off, verbose on, nodes off) select a, b, c from t1 where a > 1 and b < 2 and c > 1;
                 QUERY PLAN
-----------------------------------------------
 id |          operation
----+-----------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on mv3_t1

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on mv3_t1
        Filter: (c >= 2)
```

约束场景：添加的过滤条件不在物化视图的输出列中。

```
CREATE TABLE t1(a int, b int, c int);
CREATE MATERIALIZED VIEW mv1_t1 enable query rewrite as select c from t1;
```

当查询多出来的过滤条件不是关于物化视图输出列c时，无法通过mv1_t1重写。

```
EXPLAIN (costs off, verbose on, nodes off) select c from t1 where a > 0;
                 QUERY PLAN
-----------------------------------------------
 id |          operation
----+-----------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on t1

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on t1
        Filter: (a > 0)
```

投影重写

物化视图结构匹配查询重写不要求查询和物化视图的输出列完全一致，只要满足以下场景便可重写：

- 查询的输出列是物化视图输出列的组合。
- 查询的输出列是在物化视图输出列的基础上增加函数或者表达式。
- 查询的输出列可以通过等价类替代成物化视图的输出列。

条件重写示例：

创建基表t2和物化视图mv1_t2和mv2_t2。

 版权所有 © 华为云计算技术有限公司

```
CREATE TABLE t2(a int, b int, c int);
CREATE MATERIALIZED VIEW mv1_t2 enable query rewrite as select a, c from t2;
CREATE MATERIALIZED VIEW mv2_t2 enable query rewrite as select a from t2 where a = b;
```

场景一：查询的输出列是物化视图输出列的组合。

```
EXPLAIN (costs off, verbose on, nodes off) select a + c from t2;
                    QUERY PLAN
-----------------------------------------------------
 id |            operation
----+----------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on public.mv1_t2

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: ((a + c))
  2 --Seq Scan on public.mv1_t2
        Output: (a + c)
```

场景二：查询的输出列是在物化视图输出列的基础上增加函数或者表达式。

```
EXPLAIN (costs off, verbose on, nodes off) select nvl(a, 10) from t2;
                    QUERY PLAN
-------------------------------------------------------
 id |            operation
----+----------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on public.mv1_t2

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: (COALESCE(a, 10))
  2 --Seq Scan on public.mv1_t2
        Output: COALESCE(a, 10)
```

场景三：查询的输出列可以通过等价类替代成物化视图的输出列。

```
EXPLAIN (costs off, verbose on, nodes off) select b from t2 where a = b;
                    QUERY PLAN
-----------------------------------------------------------
 id |            operation
----+----------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on public.mv2_t2

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a
  2 --Seq Scan on public.mv2_t2
        Output: a
```

聚合重写

物化视图结构匹配查询重写不要求查询和物化视图的group by列一样，只要满足以下情况便可进行重写。

- group by列的顺序没有区别，物化视图和查询的group by列相同，顺序不同时也可重写。
- 当物化视图没有group by列，查询有group by列时，通过给物化视图添加group by的方式进行重写。

 版权所有 © 华为云计算技术有限公司

- 当物化视图和查询都有group by列，并且查询的group by列是物化视图的子集时，可以通过上卷重新group by的方式重写，min/max/count/sum等聚集函数会被重写为min(min)，max(max)，sum(count)，sum(sum)。
- 当物化视图group by列没有grouping sets/rollup/cube，而查询有grouping sets/rollup/cube，可以进行查询。

聚合重写示例：

创建示例表t3。

```
CREATE TABLE t3(a int, b int, c int);
```

场景一：group by列完全匹配。

```
CREATE MATERIALIZED VIEW mv1_t3 enable query rewrite as select a,b,sum(c) from t3 group by a,b;
EXPLAIN (costs off, verbose on, nodes off) select sum(c) from t3 group by b,a;
                    QUERY PLAN
----------------------------------------------------------
 id |            operation
 ----+----------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |    ->  Seq Scan on public.mv1_t3

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: sum
   2 --Seq Scan on public.mv1_t3
         Output: sum
```

场景二：物化视图没有group by，查询有group by。

```
CREATE MATERIALIZED VIEW mv2_t3 enable query rewrite as select a,b,c from t3
EXPLAIN (costs off, verbose on, nodes off) select sum(a) from t3 group by b,c;
                    QUERY PLAN
----------------------------------------------------------
 id |               operation
 ----+----------------------------------------
   1 | ->  HashAggregate
   2 |    ->  Streaming (type: GATHER)
   3 |       ->  HashAggregate
   4 |          ->  Seq Scan on public.mv2_t3

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --HashAggregate
         Output: pg_catalog.sum((sum(a))), b, c
         Group By Key: mv2_t3.b, mv2_t3.c
   2 --Streaming (type: GATHER)
         Output: (sum(a)), b, c
   3 --HashAggregate
         Output: sum(a), b, c
         Group By Key: mv2_t3.b, mv2_t3.c
   4 --Seq Scan on public.mv2_t3
         Output: b, c, a
```

场景三：查询的group by是物化视图的group by的子集。

```
CREATE MATERIALIZED VIEW mv3_t3 enable query rewrite as select sum(b),min(b),max(b),count(b),a,c
from t3 group by a,c;
EXPLAIN (costs off, verbose on, nodes off) select sum(b),min(b),max(b),count(b),avg(b) from t3 group by a;
                                                    QUERY
PLAN
--------------------------------------------------------------------------------------------------------------------------
-------------------------------
 id |               operation
 ----+-----------------------------------------
   1 | ->  Streaming (type: GATHER)
```

```
  2 |   ->  HashAggregate
  3 |     ->  Streaming(type: REDISTRIBUTE)
  4 |       ->  Seq Scan on public.mv3_t3

                                        Targetlist Information (identified by plan
id)

------------------------------------------------------------------------------------------------------------------------
------------------------------
  1 --Streaming (type: GATHER)
      Output: ((sum(sum))::bigint), (min(min)), (max(max)), ((sum(count))::bigint),
((((sum(sum))::bigint)::numeric / ((sum(count))::bigint)::numeric)), a
  2 --HashAggregate
      Output: (sum(sum))::bigint, min(min), max(max), (sum(count))::bigint,
(((sum(sum))::bigint)::numeric / ((sum(count))::bigint)::numeric), a
      Group By Key: mv3_t3.a
  3 --Streaming(type: REDISTRIBUTE)
      Output: a, sum, min, max, count
      Distribute Key: a
  4 --Seq Scan on public.mv3_t3
      Output: a, sum, min, max, count
```

JOIN 重写

查询和物化视图存在多表关联时，除了满足其他重写条件，需要表一一对应，并且在JOIN部分需要满足关联顺序、关联类型以及关联条件的限制。

满足以下条件可以进行多表关联的重写：

- 如果关联均为内连接，则无需满足关联顺序的限制，关联条件上，如果物化视图中两列是等价类，则必须在查询中也是等价类。

  说明

  等价类，即当查询中有t1.a=t2.a这种条件时，认为t1.a和t2.a等价，如果再加上t2.a=t3.a，则可认为t2.a和t3.a等价，最后t1.a,t2.a,t3.a等价，构成一个等价类。

- 如果存在外连接，且物化视图和查询的外连接类型相同，则要求物化视图和查询外连接的内外表顺序一致、关联条件一致、可能被补null的一侧一致。
- 如果连接类型不同，则需要满足派生重写的条件，如**表1 JOIN派生重写**所示。此外，连接类型不同的位置，不能是可能被补null的一侧。
- 如果表个数不同，查询涉及的表是物化视图的表的子集，并且物化视图多出来的表能够通过主外键关联消除掉，可以进行重写。

表 8-5 JOIN 派生重写

查询Join类型	物化视图 Join类型	补偿谓词
INNER JOIN	LEFT JOIN	LEFT JOIN右侧表的列 is not null
ANTI JOIN	LEFT JOIN	LEFT JOIN右侧表的列 is null
INNER JOIN	RIGHT JOIN	RIGHT JOIN左侧表的列 is not null
RIGHT ANTI JOIN	RIGHT JOIN	RIGHT JOIN左侧表的列 is null
SEMI JOIN	INNER JOIN	主键/unique 直接转换，无需添加补偿谓词

版权所有 © 华为云计算技术有限公司

查询Join类型	物化视图 Join类型	补偿谓词
RIGHT SEMI JOIN	INNER JOIN	主键/unique 直接转换，无需添加补偿谓词
LEFT JOIN	FULL JOIN	FULL JOIN左侧表的列 is not null
RIGHT JOIN	FULL JOIN	FULL JOIN右侧表的列 is not null
INNER JOIN	FULL JOIN	FULL JOIN两侧表的列 is not null

关联重写示例：

创建基表join_t1和join_t2。

```
CREATE TABLE join_t1 (a1 int, b1 int, c1 int);
CREATE TABLE join_t2 (a2 int, b2 int, c2 int);
```

场景一：物化视图和查询均为内连接。

```
CREATE MATERIALIZED VIEW mv_join_01 enable query rewrite as select * from join_t1 join join_t2 on
join_t1.a1 = join_t2.a2;
EXPLAIN (costs off, verbose on, nodes off) select sum(a1), b1 + c1 from join_t1 join join_t2 on join_t1.a1 =
join_t2.a2 and join_t1.b1 = join_t2.b2 and join_t1.c1 = join_t2.c2 where c1 > 10 group by b1,c1;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 id |               operation
----+-------------------------------------------
  1 | ->  HashAggregate
  2 |    ->  Streaming (type: GATHER)
  3 |       ->  HashAggregate
  4 |          ->  Seq Scan on public.mv_join_01

                            Predicate Information (identified by plan id)
-----------------------------------------------------------------------------------------------------------------
  4 --Seq Scan on public.mv_join_01
        Filter: ((mv_join_01.c1 >= 11) AND (mv_join_01.b1 = mv_join_01.b2) AND (mv_join_01.c1 =
mv_join_01.c2))

        Targetlist Information (identified by plan id)
-----------------------------------------------------------------
  1 --HashAggregate
        Output: pg_catalog.sum((sum(a1))), ((b1 + c1)), b1, c1
        Group By Key: mv_join_01.b1, mv_join_01.c1
  2 --Streaming (type: GATHER)
        Output: (sum(a1)), ((b1 + c1)), b1, c1
  3 --HashAggregate
        Output: sum(a1), (b1 + c1), b1, c1
        Group By Key: mv_join_01.b1, mv_join_01.c1
  4 --Seq Scan on public.mv_join_01
        Output: b1, c1, a1, b2, c2
```

场景二：物化视图和查询存在外连接。

```
CREATE MATERIALIZED VIEW mv_join_02 enable query rewrite as select * from join_t1 left join join_t2 on
join_t1.a1 = join_t2.a2;
EXPLAIN (costs off, verbose on, nodes off) select sum(a1), b1 + c1 from join_t1 left join join_t2 on
join_t1.a1 = join_t2.a2 where join_t1.b1 = join_t2.b2 and join_t1.c1 = join_t2.c2 and c1 > 10 group by b1,c1;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 id |               operation
----+-------------------------------------------
```

 版权所有 © 华为云计算技术有限公司

```
  1 | ->  HashAggregate
  2 |    ->  Streaming (type: GATHER)
  3 |       ->  HashAggregate
  4 |          ->  Seq Scan on public.mv_join_02

                          Predicate Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------------
  4 --Seq Scan on public.mv_join_02
        Filter: ((mv_join_02.c1 >= 11) AND (mv_join_02.b1 = mv_join_02.b2) AND (mv_join_02.c1 =
mv_join_02.c2))

        Targetlist Information (identified by plan id)
-----------------------------------------------------------------
  1 --HashAggregate
        Output: pg_catalog.sum((sum(a1))), ((b1 + c1)), b1, c1
        Group By Key: mv_join_02.b1, mv_join_02.c1
  2 --Streaming (type: GATHER)
        Output: (sum(a1)), ((b1 + c1)), b1, c1
  3 --HashAggregate
        Output: sum(a1), (b1 + c1), b1, c1
        Group By Key: mv_join_02.b1, mv_join_02.c1
  4 --Seq Scan on public.mv_join_02
        Output: b1, c1, a1, b2, c2
```

场景三：物化视图和查询的关联类型不同。

物化视图是full join，查询是right join，通过给右侧添加is not null成功重写。

```
CREATE MATERIALIZED VIEW mv_join_03 enable query rewrite as select * from join_t1 full join join_t2 on
join_t1.a1 = join_t2.a2;
EXPLAIN (costs off, verbose on, nodes off) select sum(a1), b1 + c1 from join_t1 right join join_t2 on
join_t1.a1 = join_t2.a2 where join_t1.b1 = join_t2.b2 and join_t1.c1 = join_t2.c2 group by b1,c1;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 id |               operation
----+-----------------------------------------
  1 | ->  HashAggregate
  2 |    ->  Streaming (type: GATHER)
  3 |       ->  HashAggregate
  4 |          ->  Seq Scan on public.mv_join_03

                              Predicate Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------------------
  4 --Seq Scan on public.mv_join_03
        Filter: ((mv_join_03.a2 IS NOT NULL) AND (mv_join_03.b1 = mv_join_03.b2) AND (mv_join_03.c1 =
mv_join_03.c2))

        Targetlist Information (identified by plan id)
-----------------------------------------------------------------
  1 --HashAggregate
        Output: pg_catalog.sum((sum(a1))), ((b1 + c1)), b1, c1
        Group By Key: mv_join_03.b1, mv_join_03.c1
  2 --Streaming (type: GATHER)
        Output: (sum(a1)), ((b1 + c1)), b1, c1
  3 --HashAggregate
        Output: sum(a1), (b1 + c1), b1, c1
        Group By Key: mv_join_03.b1, mv_join_03.c1
  4 --Seq Scan on public.mv_join_03
        Output: b1, c1, a1, b2, c2
```

场景四：查询的表是物化视图的表的子集，并且物化视图中关联条件符合主外键关系。

使用场景：宽表是将业务相关的指标、维度、属性关联在一起的一张表。类似大宽表，可以基于多张表的关联创建物化视图，业务查询只涉及其中的几张表。 通过物化视图改写查询后，多表关联的查询可以达到与查询相应大宽表相同的效果。注意：导入数据时，不进行外键约束校验。业务必须确保导入的数据满足主外键约束条件，再在本场景使用物化视图重写。

 版权所有 © 华为云计算技术有限公司

```
SET info_constraint_options = 'foreign_key';
ALTER TABLE join_t1 add constraint pkey_join_t1 primary key(a1);
ALTER TABLE join_t2 add constraint fkey_join_t2 foreign key(a2) references join_t1(a1);
ALTER TABLE join_t2 modify a2 not null;
EXPLAIN (costs off, verbose on, nodes off) select a2 from join_t2;
                    QUERY PLAN
-----------------------------------------------------------
 id |              operation
----+---------------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on public.mv_join_01

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a1
  2 --Seq Scan on public.mv_join_01
        Output: a1
```

使用 HINT 指定物化视图进行重写

可以使用hint mvrewrite指定物化视图名称进行重写。mvrewrite后面可以指定多个表名，或者schema.表名。

```
CREATE TABLE t1(a int,b int);
CREATE MATERIALIZED VIEW hintmv1 ENABLE QUERY REWRITE as select * from t1 where a > 1;
CREATE MATERIALIZED VIEW hintmv2 ENABLE QUERY REWRITE as select * from t1 where a > 2;

EXPLAIN (verbose on, costs off, nodes off) select /*+ mvrewrite (hintmv2) */ * from t1 where a>2;
                     QUERY PLAN
-------------------------------------------------------
 id |             operation
----+-----------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on public.hintmv2

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b
  2 --Seq Scan on public.hintmv2
        Output: a, b
```

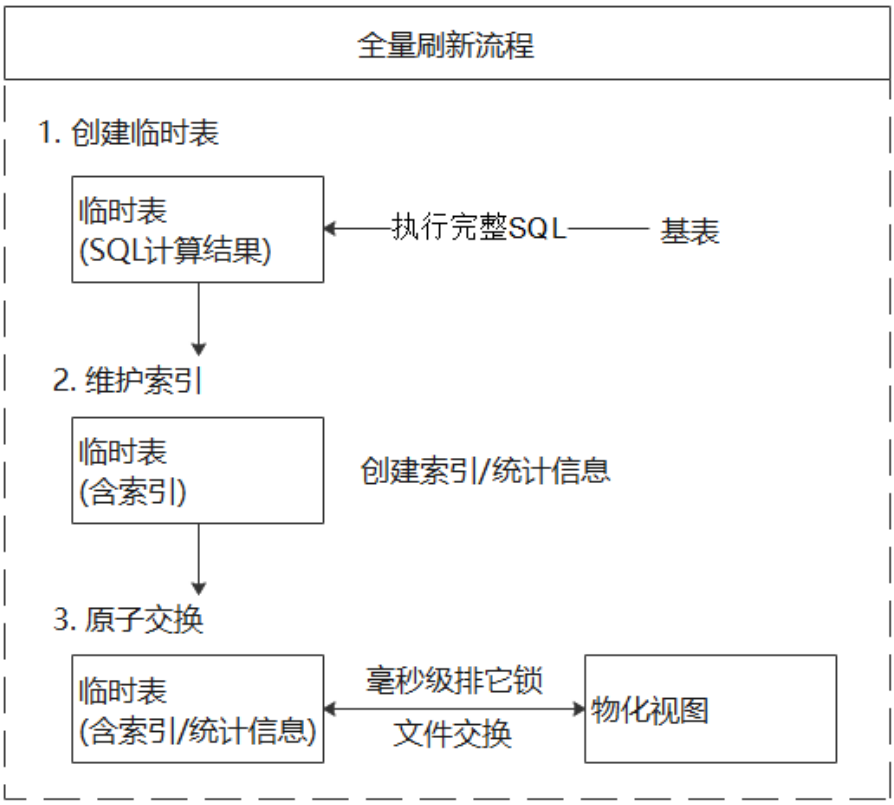
8.3 物化视图刷新

8.3.1 全量刷新

全量刷新原理

物化视图的全量刷新借助INSERT OVERWRITE逻辑，实现物化视图SQL的全量重新计算，并进行结果集的物化存储。

版权所有 © 华为云计算技术有限公司



全量刷新示例

创建基表：

```
CREATE TABLE source_sales (
    sale_id      int primary key,
    sale_date    date,
    region       varchar(20),
    product      varchar(20),
    total_amount numeric(12,2),
    order_count  int,
    status       varchar(10)
)
WITH (
    orientation = column,
    colversion = 2.0,
    enable_hstore_opt = on
)
DISTRIBUTE BY hash(sale_id);
```

创建全量刷新的物化视图：

```
CREATE MATERIALIZED VIEW test_mv1
REFRESH COMPLETE -- 全量刷新
EVERY (interval '01:00:00') --每1小时刷新一次
ENABLE QUERY REWRITE --开启查询重写
WITH(orientation=COLUMN, enable_hstore_opt=ON) --使用列存hstore_opt存储格式
AS
SELECT
    sale_id,
    sale_date,
    region,
    product,
    total_amount,
```

 版权所有 © 华为云计算技术有限公司

```
    order_count
FROM source_sales
WHERE status = 'valid';
```

约束限制

会重新执行整个SQL语句，因此没有特别的要求，适用性最为广泛。基表既可以是内表、外表或其他物化视图。基表存储格式可以是行存或列存。

物化视图本身的存储格式也没有特别的要求。

对业务影响

- 刷新时，先从基表查询计算全量的数据写入到临时表中，再维护好索引和统计信息，仅最后临时表与物化视图交换数据文件的瞬间需要对物化视图加排它锁堵塞查询。对查询的影响在毫秒级。
- 如果刷新时物化视图上有复杂的耗时长的SQL会堵塞全量刷新的交换文件，可以在**ddl_select_concurrent_mode**中设置INSERT_OVERWRITE终止查询快速完成刷新，且被终止的查询会自动重试。

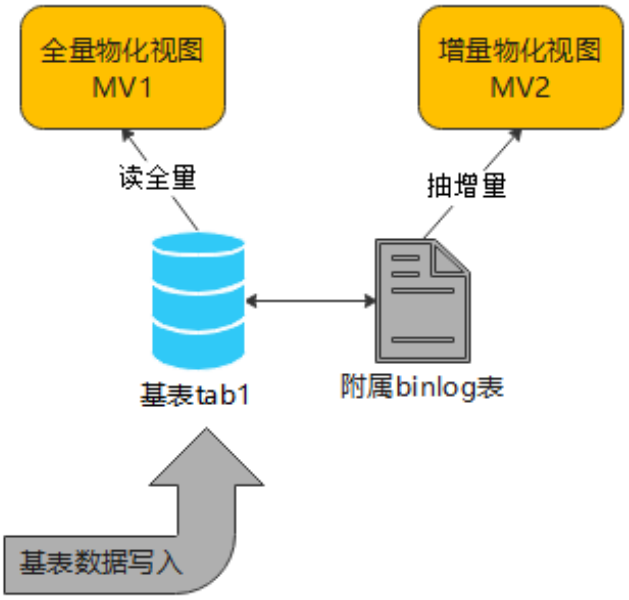
适用场景

全量刷新适用于历史数据经常发生变化的批量更新场景，一般为小时级的刷新频率。具体情况根据实际业务复杂度和资源情况确定。

8.3.2 增量刷新

刷新原理

增量刷新仅处理自上次刷新以来的增量数据，因此从计算量上比全量刷新要小很多，消耗资源也比较少。



增量刷新示例

创建基表：

版权所有 © 华为云计算技术有限公司

```
CREATE TABLE source_sales (
    sale_id       int primary key,
    sale_date     date,
    region        varchar(20),
    product       varchar(20),
    total_amount numeric(12,2),
    order_count  int,
    status        varchar(10)
)
WITH (
    orientation = column,
    colversion = 2.0,
    enable_hstore_opt = on,
    enable_binlog = true
)
DISTRIBUTE BY hash(sale_id);
```

创建增量刷新的物化视图：

```
CREATE MATERIALIZED VIEW test_mv2
REFRESH FAST -- 增量刷新
EVERY (interval '00:10:00') --每10分钟刷新一次
ENABLE QUERY REWRITE --开启查询重写
WITH(orientation=COLUMN, enable_hstore_opt=ON) --使用列存hstore_opt存储格式
AS
SELECT
    sale_date,
    region,
    sum(total_amount) AS amount,
    sum(order_count)  AS orders
FROM source_sales
GROUP BY sale_date, region;
```

约束限制

- 增量刷新前置条件：基表必须开启Binlog及需要增量计算支持。
- 不满足增量刷新的前置条件时自动转全量：
  - Binlog消费记录被清理。
  - 基表发生DDL。
  - 扩容重分布期间。
  - 其他需要全量数据的场景。
- 支持的增量算子：
  - 聚合：SUM/COUNT、AVG、MIN/MAX
  - JOIN：INNER JOIN、OUTER JOIN
  - 其他：HAVING、WHERE、GROUP BY
  - 暂不支持：Subquery、UNION ALL、Window、CTE

对业务影响

- 要求基表开启Binlog以记录增量数据，因此会有额外的存储开销。
- 计算出来的增量数据，最终以INSERT、DELETE、MERGE方式写入物化视图，因此对物化视图上的查询没有影响。

适用场景

- 实时性要求高 (分钟级/秒级)
- 基表变化频率高但变化量小 (<30%)

 版权所有 © 华为云计算技术有限公司

- 刷新频率：分钟级，根据资源情况也可秒级。

8.3.3 全量刷新与增量刷新对比

表 8-6 全量刷新与增量刷新对比

对比维度	全量刷新	增量刷新
基表类型	内表，外表，普通视图，物化视图	内表，物化视图
查询约束	无约束	支持agg、join、where、group by、having等。 不支持subquery、union all、window agg、cte。
存储格式	行列存	行列存
表分区	支持	支持
查询重写	支持	支持
刷新间隔	天、小时级	分钟、秒级
刷新代价	基表数据量	增量大小，计算复杂度
弹性计算	支持	支持

全量刷新与增量刷新的选择

表 8-7 全量刷新与增量刷新的选择

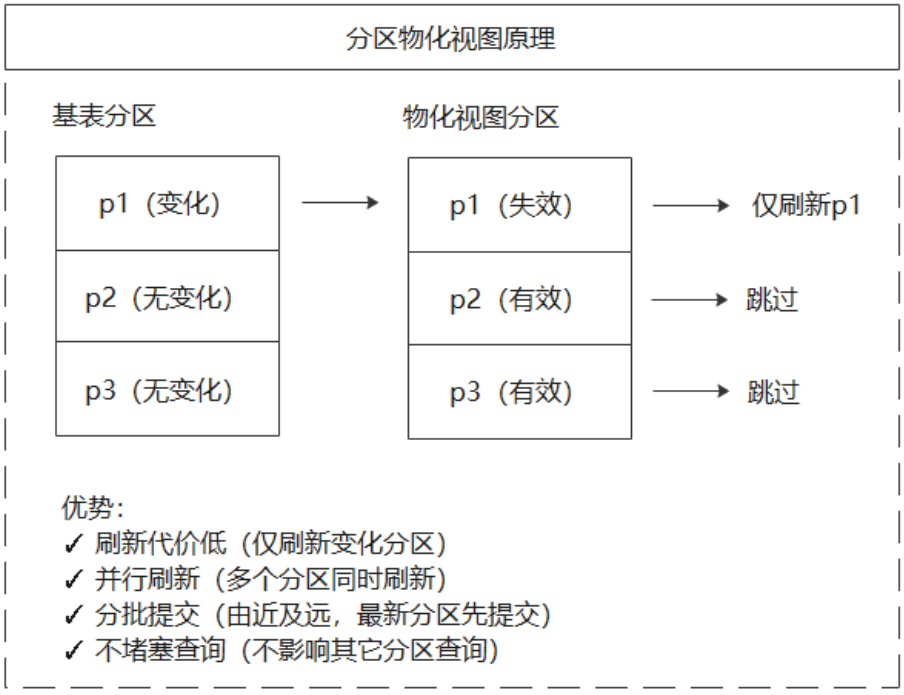
参考维度	推荐全量	推荐增量
基本约束	无法支持增量的SQL	增量全部支持的SQL
基表变化	频率很低，经常批量更新或发生DDL	频率很高且都是少量修改（30%以下）
时效要求	天/小时级即可满足	分钟/秒级读到最新数据
SQL复杂度	较复杂增量会慢	较简单不影响增量

8.3.4 分区物化视图

8.3.4.1 分区物化视图概述

为了实现物化视图能够按分区快速刷新，可以把物化视图建成分区表，并且与被查询的基表建立分区映射关系。当基表的部分分区发生变化时，刷新语句就可以只查询变化分区，然后把最新数据导入物化视图对应分区即可。

版权所有 © 华为云计算技术有限公司



- **视图创建**

  创建时指定用哪张被查询的基表分区键作为物化视图的分区键，即可实现物化视图与基表的自动分区映射。

  全量刷新和增量刷新都可以创建为分区物化视图。

- **分区失效**
  - 当基表有分区发生数据变化时，会自动将物化视图对应的分区进行失效。
  - 基表并发数据修改时，可并行对物化视图标记失效，且已失效的不会重复标记。
- **增量刷新**
  - 物化视图刷新时会自动识别失效的分区，并仅对失效分区进行刷新。
  - 并发刷新时，如果当前分区正在被刷新则会自动跳过，避免重复刷新。
  - 物化视图的刷新不阻塞基表的数据修改。
- **分区DDL**

  基表分区增加或者删除时，物化视图的分区也会根据映射规则随之增加或删除。

8.3.4.2 创建分区物化视图

DWS提供多种场景的分区物化视图创建：

- 等比例对齐分区，物化视图分区条件与基表的分区条件一致。
- 时间粒度上卷对齐分区，例如：基表按天分区，物化视图按月分区。
- 多基表对齐分区，多个基表的分区条件完全一致时，都会自动和物化视图进行映射。
- 基表为多分区键时，物化视图选第一个键为分区键创建物化视图。

版权所有 © 华为云计算技术有限公司

语法

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ [DEFAULT] | [ RESTRICT ] | [ CASCADE FORWARD ] | [ CASCADE BACKWARD ] | [ CASCADE
ALL ] ] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```

使用约束

- 必须是range分区方式。
- 物化视图只支持单分区键且为基表第一个分区键，基表可以是组合分区键。
- 上卷方式，只支持时间类型且用date_trunc进行上卷计算，仅支持日、月、季、年的上卷。
- 不支持OUTER JOIN中会产生空值的列做分区键的情况。
- 不支持MySQL兼容模式下的disable full group by。
- 指定分区的查询不支持重写。
- 分区物化视图基表不支持SPLIT、MERGE。
- 如果union的两边都满足物化视图的分区约束检测，则创建多表绑定的物化视图。
- 使用JOIN 连接基表，并将分区键设置为公共列，并且连接的基表满足物化视图的分区约束检测，则创建多表绑定的物化视图。
- 不支持查询语句中含有子链接。
- 不支持查询语句中含有子计划。
- 不支持DML和DDL修改数据，只能靠刷新更新数据。

场景示例

创建事实表fact_table和fact_table1。

```
CREATE TABLE fact_table (start_time TIMESTAMPTZ NOT NULL, id int)PARTITION BY RANGE(start_time)(
PARTITION p1 VALUES LESS THAN ('2024-01-01'),
PARTITION p2 VALUES LESS THAN ('2024-01-02'),
PARTITION p3 VALUES LESS THAN ('2024-02-03'),
PARTITION p4 VALUES LESS THAN ('2024-03-01'),
PARTITION p5 VALUES LESS THAN ('2024-03-21'),
PARTITION p6 VALUES LESS THAN ('2024-04-02'));
CREATE TABLE fact_table1 (start_time TIMESTAMPTZ NOT NULL, id int)PARTITION BY RANGE(start_time)(
PARTITION p1 VALUES LESS THAN ('2024-01-01'),
PARTITION p2 VALUES LESS THAN ('2024-01-02'),
PARTITION p3 VALUES LESS THAN ('2024-02-03'),
PARTITION p4 VALUES LESS THAN ('2024-03-01'),
PARTITION p5 VALUES LESS THAN ('2024-03-21'),
PARTITION p6 VALUES LESS THAN ('2024-04-02'));
```

创建维度表dimen_table。

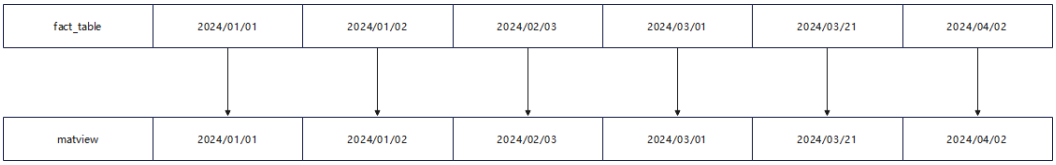
```
CREATE TABLE dimen_table (start_time TIMESTAMPTZ NOT NULL, id INT);
```

场景一：等比例对齐分区。

创建物化视图level_normal，它的分区与fact_table表分区，分区条件完全一样。对数据类型无特殊要求。

版权所有 © 华为云计算技术有限公司

```
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
```

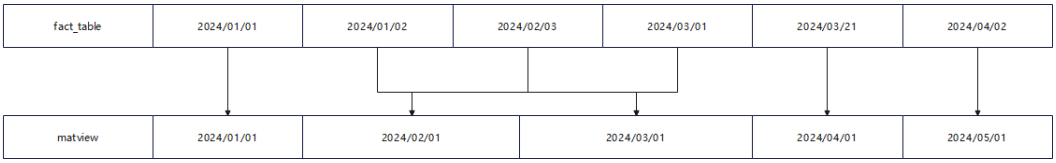


场景二：时间粒度上卷对齐分区。

通过在分区键上使用date_trunc参数，创建一个分区时间比基表更粗粒度的物化视图。当检测到基表分区中的数据变更后，DWS数据库将会刷新物化视图中对应的上卷分区。

通过date_trunc('time_unit', column_name)参数指定上卷，物化视图分区范围大于基表分区，比如：基表按天分区，物化视图按月分区。一般通过date_trunc参数，实现时间的向上截取。支持上卷的时间单位：年，季，月，日。数据类型：只支持time, timestamp, timestamptz类型。

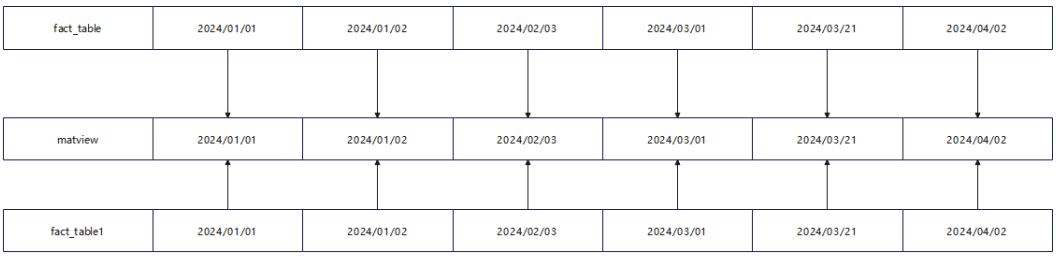
```
CREATE MATERIALIZED VIEW level_month enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY date_trunc('month', start_time)
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
```



场景三：多基表对齐分区。

当多个基表分区键的种类和分区范围一致，即基表使用相同的分区键，可以创建多基表绑定的物化视图。

```
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT start_time, id FROM fact_table union SELECT start_time, id FROM fact_table1;
CREATE MATERIALIZED VIEW level_normal enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY start_time
AS SELECT fact_table1.start_time, fact_table.id, fact_table1.id as id1
FROM fact_table inner join fact_table1 on fact_table.start_time=fact_table1.start_time;
```



场景四：基表为多分区键时，支持以基表的第一个分区键创建物化视图。

```
CREATE TABLE IF NOT EXISTS  multi_time_base (
    new_start_time timestamp with time zone NOT NULL,
```

 版权所有 © 华为云计算技术有限公司

```
    id int,
    v1 int
) PARTITION BY RANGE(new_start_time,id) (
    PARTITION p1 VALUES LESS THAN ('2022-01-01',400),
    PARTITION p2 VALUES LESS THAN ('2022-01-05',400),
    PARTITION p3 VALUES LESS THAN ('2022-01-05',500),
    PARTITION p4 VALUES LESS THAN ('2022-02-15',500),
    PARTITION p5 VALUES LESS THAN ('2022-02-16',500),
    PARTITION p6 VALUES LESS THAN ('2022-03-10',500)
);
CREATE MATERIALIZED VIEW multi_time_mv
PARTITION BY (date_trunc('month', new_start_time))
AS SELECT new_start_time,id FROM multi_time_base;
```

8.3.4.3 刷新分区物化视图

当创建成功分区物化视图时，可以使用分区刷新减少刷新代价。

自动刷新

- 由系统内部识别自动刷新，通过以下命令刷新。

  ```
  REFRESH MATERIALIZED VIEW materialized_view_name;
  ```

  该命令会尝试刷新物化视图，如果满足分区刷新的条件则进行分区刷新，否则转为全量刷新。
- 设置GUC参数**mv_refresh_parts_per_trans**（该参数9.1.1.200及以上集群版本支持），可实现分区由近及远的分批刷新和提交，先刷最新分区，保证最新分区的时效性。推荐按分区访问的业务场景开启此参数。
- 并发与冲突，当自动刷新时遇到当前分区正在被刷新，会自动跳过当前分区，避免重复浪费计算资源。

手动刷新指定分区

通过以下命令指定需要刷新的物化视图分区。

```
REFRESH MATERIALIZED VIEW materialized_view_name PARTITION (partname1,partname2);
```

该命令会尝试刷新物化视图分区partname1，partname2，如果当前分区不需要刷新，则不会刷新，并会提示数据为最新。

强制手动刷新指定分区

通过以下命令强制刷新物化视图指定分区。

```
REFRESH MATERIALIZED VIEW materialized_view_name PARTITION (partname) RESTRICT;
```

查看分区物化视图

查看分区的数据是否更新。

通过pg_matview_partition的active字段可以查看当前分区是否已更新。

```
\d  pg_matview_partition;
Table "pg_catalog.pg_matview_partition"
      Column       |           Type           | Modifiers
-------------------+--------------------------+-----------
matpartid          | oid                      | not null
active             | boolean                  | not null
refresh_start_time  | timestamp with time zone | not null
refresh_finish_time | timestamp with time zone | not null
```


版权所有 © 华为云计算技术有限公司

字段含义如下：

- matpartid，表示物化视图分区OID
- active，表示当前分区是否已更新。 t表示已更新，f表示失效。
- refresh_start_time，表示最后一次分区刷新开始时间。
- refresh_finish_time，表示最后一次分区刷新结束时间。

示例

步骤1 创建时间上卷物化视图level_month并查看视图信息。

```
CREATE MATERIALIZED VIEW level_month enable query rewrite
DISTRIBUTE BY HASH(start_time)
PARTITION BY date_trunc('month', start_time)
AS SELECT fact_table.start_time, dimen_table.id FROM fact_table JOIN dimen_table ON fact_table.id =
dimen_table.id;
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17092  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
```

步骤2 向基表插入数据。

```
INSERT INTO fact_table (start_time, id) VALUES ('2023-05-01',1);
```

步骤3 查看分区失效情况。

```
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17092  | f      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
```

步骤4 刷新物化视图。

```
REFRESH MATERIALIZED VIEW level_month;
```

由于只是分区失效，则会触发分区刷新。

步骤5 查看刷新后的状态。

```
SELECT * FROM pg_matview_partition;
matpartid | active |      refresh_start_time       |      refresh_finish_time
----------+--------+-------------------------------+-------------------------------
  17093  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17094  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17095  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17096  | t      | 2024-11-18 15:15:32.177697+08 | 2024-11-18 15:15:32.204009+08
  17092  | t      | 2024-11-18 15:16:28.902902+08 | 2024-11-18 15:16:28.940159+08
```

----结束

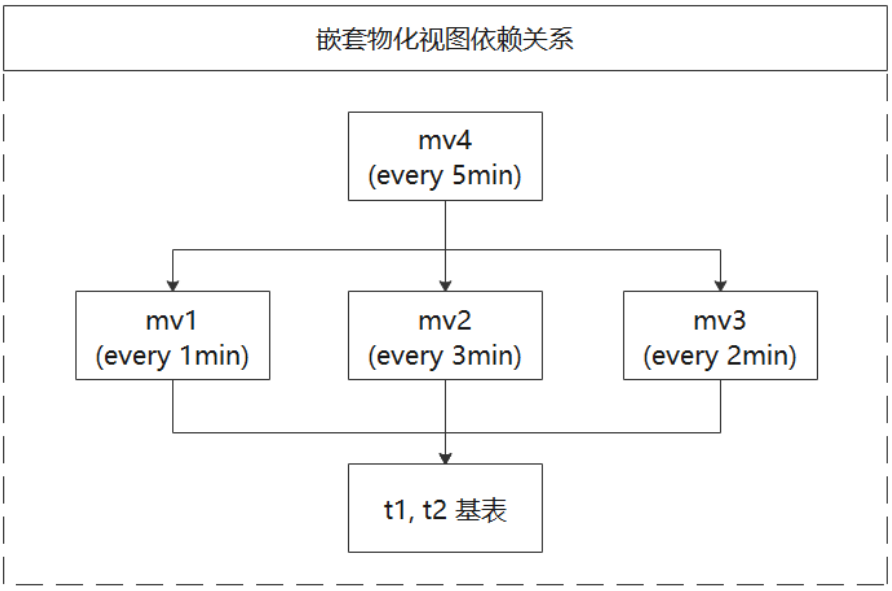
8.3.5 嵌套物化视图

8.3.5.1 嵌套物化视图概述

DWS物化视图支持嵌套创建、嵌套物化视图刷新以及嵌套物化视图重写。嵌套物化视图的应用，可以有效地减少单个物化视图定义的复杂程度。

版权所有 © 华为云计算技术有限公司

- 创建嵌套物化视图，可以基于物化视图构建物化视图。
- 级联刷新，嵌套物化视图刷新时，不仅能刷新当前物化视图，也支持自动级联刷新。
- 定时级联刷新，针对定时刷新的物化视图，支持用户指定和修改刷新方式。
- 嵌套物化视图重写，查询语句能自动被重写成查询物化视图的SQL。



8.3.5.2 创建嵌套物化视图

DWS支持创建嵌套物化视图，嵌套物化视图是指在物化视图的定义中使用其他物化视图的过程。 通过嵌套物化视图，可以将多个物化视图组合在一起，形成更复杂和综合的查询结果。

通过GUC参数**mv_max_cascade_depth**指定嵌套物化视图的最大深度，为了刷新性能上的考虑，当前最大深度为10，默认值为3。

使用约束

- 不支持预聚合物化视图嵌套。
- 不支持基于物化视图创建预聚合物化视图。
- 不支持非定时刷新物化视图指定或者修改刷新方式。
- 嵌套物化视图最大深度为10，暂不支持更深的嵌套层数。
- 升级场景不支持创建嵌套物化视图。

语法格式

创建嵌套物化视图的语法基本同物化视图的语法一致，物化视图的定义语句中允许基表是物化视图。增加refresh_behavior用于定时刷新物化视图指定刷新方式。

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] materialized_view_name
[ ( column_name [, ...] ) ]
[ BUILD { DEFERRED | IMMEDIATE } ]
[ REFRESH [ refresh_behavior] [ COMPLETE ] [ ON DEMAND ] [ [ START WITH (timestamptz) ] | [ EVERY (interval) ] ] ]
```

 版权所有 © 华为云计算技术有限公司

```
[ { ENABLE | DISABLE } QUERY REWRITE ]
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY { REPLICATION | ROUNDROBIN | { HASH ( column_name [,...] ) } } ]
[ PARTITION BY {(column_name) | date_trunc('time_unit', column_name)} ]
AS query;
```

查看嵌套物化视图

系统表pg_matview_depend中的relid字段，存放物化视图的OID。

```
\d+ pg_matview_depend
Table "pg_catalog.pg_matview_depend"
Column             |   Type    | Modifiers | Storage | Stats target | Description
-------------------+-----------+-----------+---------+--------------+-------------
matviewid          | oid       | not null  | plain   |              |
relid              | oid       | not null  | plain   |              |
partids            | oidvector | not null  | plain   |              |
contain_entire_rel | boolean   | not null  | plain   |              |
Indexes:
"pg_matview_depend_mvid_relid_index" UNIQUE, btree (matviewid, relid) TABLESPACE pg_default
"pg_matview_depend_relid_index" btree (relid) TABLESPACE pg_default
Replica Identity: NOTHING
Has OIDs: no
```

示例

1. 创建基表、物化视图和嵌套物化视图。

```
CREATE TABLE t1(a int, b int, c int);
CREATE TABLE t2(a int, b int, c int);
CREATE MATERIALIZED VIEW mv1 ENABLE QUERY REWRITE as select * from t1 WHERE a = 1;
CREATE MATERIALIZED VIEW mv2 ENABLE QUERY REWRITE as select * from t2 WHERE b = 1;
CREATE MATERIALIZED VIEW mv3 ENABLE QUERY REWRITE as select * from mv2 WHERE b = 1;
CREATE MATERIALIZED VIEW mv4 ENABLE QUERY REWRITE as select t1.a as t1a,t1.b as t1b,t1.c as t1c,t2.a as t2a,t2.b as t2b,t2.c as t2c from t1 left join t2 on t1.a = t2.a WHERE t2.b = 1;
CREATE MATERIALIZED VIEW mv5 ENABLE QUERY REWRITE as (select * from mv1) union all (select * from t1 WHERE a = 1);
CREATE MATERIALIZED VIEW mv6 ENABLE QUERY REWRITE as select * from mv4 left join (select * from mv1 union all select * from mv5)mv5 on mv4.t1a = mv5.a;
```

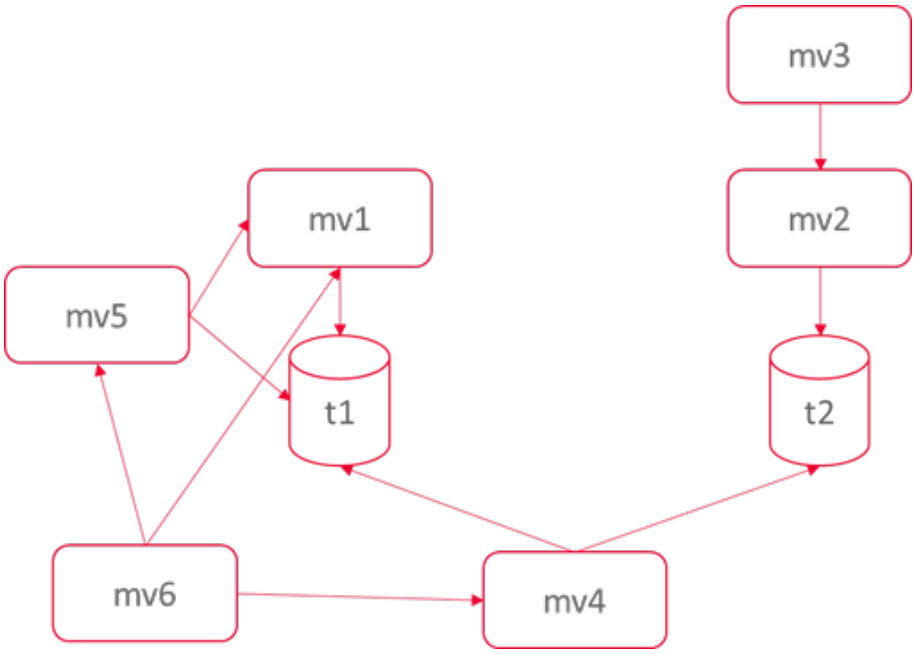
2. 上述SQL中先创建两个表t1和t2，然后以t1和t2为基表，创建了物化视图mv1和mv2；再以t1、t2、mv1、mv2为基表依次创建物化视图mv3、mv4、mv5和mv6。

和单层物化视图相比，嵌套物化视图顶层的定义更为简洁，使用上更为灵活。

3. 可通过下图了解基表和物化视图的创建关系及依赖关系。

 版权所有 © 华为云计算技术有限公司

图 8-1 嵌套物化视图依赖关系



4. 当底层数据发生变更时，需要刷新物化视图，保证物化视图中的数据都是有效的。即t1表数据发生变化，mv1、mv4、mv5和mv6中的数据都需要更新。使用级联刷新单个物化视图，自动将相关联的物化视图都进行更新。

```
REFRESH MATERIALIZED VIEW mv6 cascade backward;
```

5. 当嵌套物化视图中的数据都是有效的，执行查询语句时，会进行自动识别和匹配，将查询语句自动重写为查询物化视图，大大减少查询语句中的计算。

8.3.5.3 刷新嵌套物化视图

级联刷新

嵌套物化视图中，为了避免单个物化视图刷新繁琐操作，支持自动级联刷新嵌套物化视图。

当前支持五种刷新方式，默认、强制、级联向上、级联向下和全部级联。即对应refresh_behavior取值：DEFAULT、RESTRICT、CASCADE FORWARD、CASCADE BACKWARD和CASCADE ALL。

- 默认方式：刷新单个物化视图，当物化视图的基表未发生数据变化时，不进行刷新操作，直接跳过。
- 强制方式：刷新单个物化视图。
- 级联向上：先刷新当前物化视图，然后依次刷新依赖当前物化视图的嵌套物化视图。
- 级联向下：找到当前物化视图依赖的所有物化视图，依次从最底层开始刷新，最后刷新当前物化视图。
- 全部级联：先执行级联向下刷新，再执行级联向上刷新。

版权所有 © 华为云计算技术有限公司

> **说明**
>
> 级联刷新方式中，采用的是默认刷新方式，即下层物化视图/表数据未发生变化，不进行刷新操作。

示例：

以**创建嵌套物化视图**中创建的嵌套物化视图为例。

对物化视图mv5进行全部级联刷新。

```
REFRESH MATERIALIZED VIEW mv5 cascade all;
```

级联刷新流程：先刷新mv1，再刷新mv5，最后刷新mv6。

相当于先对mv5执行级联向下刷新，然后进行级联向上刷新。

```
REFRESH MATERIALIZED VIEW mv5 cascade backward;
REFRESH MATERIALIZED VIEW mv5 cascade forward;
```

对于级联向下刷新，会先找到mv5依赖的所有底层物化视图，然后从底层向上层，逐层进行物化视图刷新。即执行：

```
REFRESH MATERIALIZED VIEW mv5 cascade backward;
```

级联刷新流程：先刷新mv1，再刷新mv5。

对于级联向上刷新，会先找到所有依赖mv5的物化视图，然后从底层向上层，逐层进行物化视图刷新。

级联刷新流程：先刷新mv5，再刷新mv6。

定时刷新物化视图

对于后台定时刷新的物化视图，需要用户指定刷新方式，否则都采用默认的刷新方式，而不会进行级联刷新。

因此针对此类物化视图，支持在创建时指定级联刷新方式，支持修改级联刷新方式。

示例：

1. 创建定时刷新物化视图。

```
CREATE TABLE test_table(a int2, b int, c int8, d int) WITH (ORIENTATION = COLUMN, COLVERSION = 2.0) DISTRIBUTE BY hash(c);

CREATE MATERIALIZED VIEW mv_test_sync_default REFRESH START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s')  ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;

CREATE MATERIALIZED VIEW mv_test_sync_restrict REFRESH restrict START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select a,sum(b) b from test_table group by a;

CREATE MATERIALIZED VIEW mv_test_sync_forward REFRESH cascade forward START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select a,sum(b) b from test_table group by a;

CREATE MATERIALIZED VIEW mv_test_sync_backward REFRESH cascade backward START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select a,sum(b) b from test_table group by a;

CREATE MATERIALIZED VIEW mv_test_sync_all REFRESH cascade all START WITH('2020-01-01 12:12:12'::timestamptz) EVERY (interval '1 s') ENABLE QUERY REWRITE as select  a,sum(b) b from test_table group by a;
```

 版权所有 © 华为云计算技术有限公司

2. 查看定时刷新任务。

```
SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%'
ORDER BY 1;
```

```
postgres=# SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%' ORDER BY 1;
                                 what                                 |    category_id    | is_broken | interval
----------------------------------------------------------------------+-------------------+-----------+----------
 REFRESH MATERIALIZED VIEW public.mv_test_sync_all CASCADE ALL;       | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_backward CASCADE BACKWARD; | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_default;               | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_forward CASCADE FORWARD;   | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_restrict RESTRICT;     | MatViewKernelTask | f         | 00:00:01
(5 rows)
```

3. 修改级联刷新方式。

```
ALTER MATERIALIZED VIEW mv_test_sync_backward REFRESH START WITH('2020-01-01
15:15:15'::timestamptz) EVERY (interval '10 s') ;
ALTER MATERIALIZED VIEW mv_test_sync_restrict REFRESH cascade all START WITH('2020-01-01
12:12:12'::timestamptz) EVERY (interval '20 s') ;
ALTER MATERIALIZED VIEW mv_test_sync_all REFRESH  default;
ALTER MATERIALIZED VIEW mv_test_sync_forward REFRESH default START WITH('2020-01-01
12:12:12'::timestamptz) EVERY (interval '50 s') ;
```

```
postgres=# SELECT what, category_id, is_broken, interval FROM scheduler.pg_task WHERE what LIKE '%mv_test%' ORDER BY 1;
                                 what                                 |    category_id    | is_broken | interval
----------------------------------------------------------------------+-------------------+-----------+----------
 REFRESH MATERIALIZED VIEW public.mv_test_sync_all;                   | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_backward CASCADE BACKWARD; | MatViewKernelTask | f         | 00:00:10
 REFRESH MATERIALIZED VIEW public.mv_test_sync_default;               | MatViewKernelTask | f         | 00:00:01
 REFRESH MATERIALIZED VIEW public.mv_test_sync_forward;               | MatViewKernelTask | f         | 00:00:50
 REFRESH MATERIALIZED VIEW public.mv_test_sync_restrict CASCADE ALL;  | MatViewKernelTask | f         | 00:00:20
(5 rows)
```

8.4 物化视图设计与实践

8.4.1 物化视图设计

物化视图设计的核心原则：“相似归一，找公共集合”，减少重复计算。

此原则是指通过分析历史高频相似SQL（基于Unique SQL ID），提取归纳公共子查询并创建物化视图的过程，其步骤主要为：

1. TopSQL分析

```
SELECT unique_sql_id, count(*) as freq
FROM pgxc_wlm_session_info
WHERE query LIKE '%table_name%'
GROUP BY unique_sql_id
ORDER BY freq DESC;
```

2. 归纳公共计算。
   - 相同表上的相似聚合。
   - 相同表连接的不同查询。
   - 常用的过滤条件组合。
3. 设计物化视图
   - 提取更基础、更简单的计算。
   - 复用频率越高越好。
   - 优先物化底层视图。

示例

步骤1 建表并插入数据。

```
CREATE TABLE orders (
    order_id    int,
```


版权所有 © 华为云计算技术有限公司

```
    region      varchar(20),
    product     varchar(20),
    amount      numeric(10,2),
    order_date  date
)
WITH (orientation=column)
DISTRIBUTE BY hash(order_id);

INSERT INTO orders VALUES
(1, 'North', 'phones', 1000.00, '2024-01-05'),
(2, 'North', 'Laptops', 3000.00, '2024-01-10'),
(3, 'North', 'phones', 1500.00, '2024-01-15'),
(4, 'East', 'phones', 2000.00, '2024-01-08'),
(5, 'East', 'Laptops', 5000.00, '2024-01-12'),
(6, 'East', 'Tablets', 800.00, '2024-01-20'),
(7, 'South', 'Laptops', 4500.00, '2024-01-06'),
(8, 'South', 'Tablets', 1200.00, '2024-01-18'),
(9, 'South', 'phones', 1800.00, '2024-01-22'),
(10,'North', 'Tablets', 600.00, '2024-01-25');
```

步骤2 创建物化视图。

创建物化视图mv_region_product，使用最细粒度（region+product），同时存储SUM和COUNT。

```
CREATE MATERIALIZED VIEW mv_region_product
REFRESH COMPLETE
ENABLE QUERY REWRITE
WITH (orientation=column,enable_hstore_opt = ON)
DISTRIBUTE BY hash(region)
AS
SELECT
    region,
    product,
    sum(amount)   AS total_amount,
    count(amount) AS cnt_amount
FROM orders
GROUP BY region, product;
```

步骤3 刷新物化视图。

```
REFRESH MATERIALIZED VIEW mv_region_product;
```

步骤4 查询按照region聚合。

```
SELECT region, sum(amount)
FROM orders
GROUP BY region;
```

步骤5 查看执行计划，预期使用物化视图重写。

```
EXPLAIN (verbose on, costs off, nodes off)
SELECT region, sum(amount)
FROM orders
GROUP BY region;
                        QUERY PLAN
------------------------------------------------------------
 id |                    operation
 ---+-----------------------------------------------------
  1 | ->  Row Adapter
  2 |    ->  Vector Streaming (type: GATHER)
  3 |       ->  Vector Sonic Hash Aggregate
  4 |          ->  CStore Scan on public.mv_region_product

 Targetlist Information (identified by plan id)
 -----------------------------------------------
  1 --Row Adapter
       Output: region, (sum(total_amount))
  2 --Vector Streaming (type: GATHER)
       Output: region, (sum(total_amount))
  3 --Vector Sonic Hash Aggregate
```

 版权所有 © 华为云计算技术有限公司

```
      Output: region, sum(total_amount)
      Group By Key: mv_region_product.region
4 --CStore Scan on public.mv_region_product
      Output: region, total_amount
      Distribute Key: region
```

----结束

物化视图设计原则

- 物化视图的设计越基础，复用频率越高，收益越大。
- 避免过度物化，增加系统负担。
- 合理设计刷新策略，平衡性能与实时性。

8.4.2 物化视图设计最佳实践

刷新任务调度

当集群中存在多个物化视图时，如果所有刷新任务在同一时刻集中触发，会导致CPU、内存和I/O资源瞬间争抢，影响整体查询性能。以下从错峰编排、执行计划稳定性和级联刷新三个方面给出调度建议。

- 刷新任务错峰编排

  核心原则：通过START WITH和EVERY为不同物化视图设定不同的首次刷新时间和刷新间隔，避免多个刷新任务在同一时刻并发执行。推荐刷新时：

  1. 错开首次刷新时间：不同物化视图的 START WITH 至少间隔数分钟，避免“整点齐刷”。

  2. 按业务优先级设置间隔：核心报表的刷新间隔可以较短，辅助报表的间隔适当拉长。

  3. 利用低峰窗口：将全量刷新的首次时间设定在业务低峰期（如凌晨），避开白天查询高峰。

  示例：

  ```
  -- 报表A：每日凌晨 01:00 开始，每 24 小时刷新一次
  CREATE MATERIALIZED VIEW mv_report_a
    REFRESH COMPLETE ON DEMAND
    START WITH('2026-09-02 01:00:00'::timestamptz) EVERY(interval '1 day')
    AS SELECT ...;

  -- 报表B：每日凌晨 02:30 开始，每 24 小时刷新一次（与报表A错开 1.5 小时）
  CREATE MATERIALIZED VIEW mv_report_b
    REFRESH COMPLETE ON DEMAND
    START WITH('2026-09-02 02:30:00'::timestamptz) EVERY(interval '1 day')
    AS SELECT ...;

  -- 看板C：从当前时间起，每 10 分钟刷新一次
  CREATE MATERIALIZED VIEW mv_dashboard_c
    REFRESH FAST ON DEMAND
    START WITH(sysdate) EVERY(interval '10 min')
    AS SELECT ...;
  ```

- 保持执行计划稳定

  物化视图的刷新本质上是执行一条查询并将结果写入。如果刷新时的执行计划发生跳变（例如从Hash Join变成NestLoop），刷新耗时可能突然增大数倍，导致资源占用异常。

- 防止重复刷新

版权所有 © 华为云计算技术有限公司

如果级联使用物化视图，不必每一层都创建刷新任务，可以按数据流管道级联刷新；使用CASCADE BACKWARD控制。推荐刷新时：

1.通过CREATE MATERIALIZED VIEW的mv_refresh_guc选项为刷新会话固定关键执行参数（如Join方式、资源队列等），减少执行计划跳变的风险。

2.执行计划通过hint固定，避免计划跳变引起性能问题。

- 级联物化视图的刷新编排

  当物化视图之间存在依赖关系（即上层物化视图基于下层物化视图创建）时，不需要为每一层都单独创建定时刷新任务。可以使用级联刷新（CASCADE BACKWARD）按数据流方向一次性刷新整条链路，减少重复刷新的开销。可参考下表：

表 8-8 级联刷新方向说明

选项	含义	适用场景
CASCADE BACKWARD	刷新当前物化视图及其所有上游依赖	从最终物化视图触发，先刷新上游再刷新自身
CASCADE FORWARD	刷新当前物化视图及其所有下游依赖	从基表层物化视图触发，向后逐层刷新
CASCADE ALL	刷新当前物化视图的所有上下游依赖	需要全链路刷新时使用

级联物化视图的刷新示例：

只需在最上层物化视图创建定时任务，使用CASCADE BACKWARD自动刷新上游。

```
CREATE MATERIALIZED VIEW mv_layer_top
  REFRESH COMPLETE ON DEMAND
  START_WITH('2025-09-01 03:00:00'::timestamptz) EVERY(interval '10 min')
  AS SELECT ... FROM mv_layer_middle;
```

刷新时自动先刷新上游mv_layer_middle，再刷新mv_layer_top。

```
REFRESH MATERIALIZED VIEW mv_layer_top CASCADE BACKWARD;
```

级联场景下的间隔设置原则：

上层物化视图的刷新间隔必须大于或等于下层物化视图的刷新间隔，否则上层刷新时读到的仍是下层的旧数据，刷新没有实际意义。

场景示例：

表 8-9

是否正确	上层间隔	下层间隔	说明
正确	10分钟	5分钟	上层刷新时下层已更新，能读到最新数据。
正确	10分钟	10分钟	上下层同步刷新。
错误	2分钟	5分钟	上层每2分钟刷新一次，而下层5分钟才更新，因此大多数情况下读到的是旧数据。

 版权所有 © 华为云计算技术有限公司

存储格式选择

推荐使用hstore_opt存储格式。

```
CREATE MATERIALIZED VIEW mv_test
WITH (orientation = COLUMN, enable_hstore_opt = ON)
AS SELECT ...
```

典型应用模式一：报表加速

- 业务场景：业务系统每天凌晨批量写入数据，白天有大量重复的聚合查询（如按地区统计销售额、按时间段统计订单量）。
- 基表：销售明细表（亿级数据）。
- 查询：每日销售汇总。

通过全量刷新物化视图预计算聚合结果，配合查询重写自动将用户查询路由到物化视图，大幅降低查询响应时间。

示例：

步骤1 创建基表，订单明细表orders和客户维度表customers。

```
CREATE TABLE orders (
    order_id    BIGINT NOT NULL,
    region      VARCHAR(50) NOT NULL,
    product     VARCHAR(100) NOT NULL,
    amount      NUMERIC(18,2) NOT NULL,
    order_date  DATE NOT NULL,
    customer_id BIGINT
)
WITH (orientation=column)
DISTRIBUTE BY HASH(order_id);

CREATE TABLE customers (
    customer_id BIGINT NOT NULL PRIMARY KEY,
    customer_name VARCHAR(200),
    city        VARCHAR(100),
    level       INT
)
WITH (orientation=column)
DISTRIBUTE BY HASH(customer_id);
```

步骤2 插入测试数据。

```
INSERT INTO orders SELECT
    i,
    CASE (i % 5) WHEN 0 THEN 'East' WHEN 1 THEN 'South' WHEN 2 THEN 'North' WHEN 3 THEN
'southwest' ELSE 'northwest' END,
    'product_' || (i % 100),
    (i % 10000) * 1.5 + 10,
    DATE '2024-01-01' + (i % 365),
    i % 10000
FROM generate_series(1, 1000) AS i;

INSERT INTO customers SELECT
    i,
    'customer_' || i,
    CASE (i % 10) WHEN 0 THEN 'Shanghai' WHEN 1 THEN 'Guangzhou' WHEN 2 THEN 'Beijing' WHEN 3
THEN 'Chengdu' ELSE 'Xian' END,
    (i % 5) + 1
FROM generate_series(1, 100) AS i;
```

步骤3 创建物化视图MV1、MV2和MV3。

- MV1：按地区统计销售额（全量刷新+查询重写）。

  ```
  CREATE MATERIALIZED VIEW mv_region_sales
    BUILD IMMEDIATE
  ```

 版权所有 © 华为云计算技术有限公司

```
    REFRESH COMPLETE ON DEMAND
    ENABLE QUERY REWRITE
    WITH (ORIENTATION = COLUMN, enable_hstore_opt = on)
    DISTRIBUTE BY HASH(region)
AS
SELECT
    region,
    COUNT(*)          AS order_count,
    SUM(amount)       AS total_amount,
    AVG(amount)       AS avg_amount,
    MAX(amount)       AS max_amount,
    MIN(amount)       AS min_amount
FROM orders
GROUP BY region;
```

- MV2：按地区+产品统计（多维度聚合）。

```
CREATE MATERIALIZED VIEW mv_region_product_sales
    BUILD IMMEDIATE
    REFRESH COMPLETE ON DEMAND
    ENABLE QUERY REWRITE
    WITH (ORIENTATION = COLUMN, enable_hstore_opt = on)
    DISTRIBUTE BY HASH(region)
AS
SELECT
    o.region,
    o.product,
    COUNT(*)          AS order_count,
    SUM(o.amount)     AS total_amount
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY o.region, o.product;
```

- MV3：按月统计销售额。

```
CREATE MATERIALIZED VIEW mv_monthly_sales
    BUILD IMMEDIATE
    REFRESH COMPLETE ON DEMAND
    ENABLE QUERY REWRITE
    WITH (ORIENTATION = COLUMN, enable_hstore_opt = on)
    DISTRIBUTE BY HASH(month)
AS
SELECT
    date_trunc('month', order_date) AS month,
    region,
    COUNT(*)                        AS order_count,
    SUM(amount)                     AS total_amount
FROM orders
GROUP BY date_trunc('month', order_date), region;
```

步骤4 查询重写验证。用户查询（会被自动重写到mv_region_sales）

```
EXPLAIN (VERBOSE, COSTS OFF)
SELECT
    region,
    COUNT(*)      AS order_count,
    SUM(amount)  AS total_amount,
    AVG(amount)  AS avg_amount
FROM orders
GROUP BY region
ORDER BY region;
                        QUERY PLAN
------------------------------------------------------------------
 id |                    operation
----+------------------------------------------------
  1 | ->  Row Adapter
  2 |    ->  Vector Streaming (type: GATHER)
  3 |       ->  Vector Sort
  4 |          ->  CStore Scan on public.mv_region_sales

      Targetlist Information (identified by plan id)
------------------------------------------------------------------
  1 --Row Adapter
```

 版权所有 © 华为云计算技术有限公司

```
     Output: region, order_count, total_amount, avg_amount
2 --Vector Streaming (type: GATHER)
     Output: region, order_count, total_amount, avg_amount
     Merge Sort Key: mv_region_sales.region
     Node/s: All datanodes (group1, bucket:16384)
3 --Vector Sort
     Output: region, order_count, total_amount, avg_amount
     Sort Key: mv_region_sales.region
4 --CStore Scan on public.mv_region_sales
     Output: region, order_count, total_amount, avg_amount
     Distribute Key: region
```

步骤5 刷新物化视图。

- 手动刷新单个物化视图。
  ```
  REFRESH MATERIALIZED VIEW mv_region_sales;
  ```
- 强制刷新（忽略刷新抑制）。
  ```
  REFRESH MATERIALIZED VIEW mv_region_sales RESTRICT;
  ```
- 批量数据导入后刷新。
  ```
  INSERT INTO orders SELECT
     i + 100000,
     'East', 'product_99', 999.99, DATE '2025-01-01', 1
  FROM generate_series(1, 1000) AS i;

  REFRESH MATERIALIZED VIEW mv_region_sales;
  REFRESH MATERIALIZED VIEW mv_region_product_sales;
  REFRESH MATERIALIZED VIEW mv_monthly_sales;
  ```

----结束

模式二：实时数仓分层

通过级联构建三层实时数仓：DWD明细清洗 → DWS基础汇总 → ADS 应用报表。

各层按不同频率定时增量刷新，上层MV自动从下层MV获取增量数据。

架构总览：

```
source_sales (基表, binlog)
    │
    │  REFRESH FAST EVERY (5 min)     ← DWD 层：明细清洗
    ▼
mv_dwd_sales (过滤 + 清洗)
    │
    │  REFRESH COMPLETE EVERY (10 min)    ← DWS 层：基础汇总
    ▼
mv_dws_sales (GROUP BY 聚合)
    │
    │  REFRESH COMPLETE EVERY (30 min)    ← ADS 层：应用报表
    ▼
mv_ads_sales (派生指标)
```

物化视图分层功能概述如下表所示：

表 8-10 物化视图分层功能

层级	物化视图名称	功能	刷新频率	级联深度
DWD	mv_dwd_sales	过滤有效订单、字段标准化。	5 min	1
DWS	mv_dws_sales	按日期+区域聚合汇总。	10 min	2

版权所有 © 华为云计算技术有限公司

层级	物化视图名称	功能	刷新频率	级联深度
ADS	mv_ads_sales	计算客单价等派生指标。	30 min	3

示例：

创建基表，必须开启binlog（FAST增量刷新的前提）。

```
CREATE TABLE source_sales (
    sale_id      int primary key,
    sale_date    date,
    region       varchar(20),
    product      varchar(20),
    total_amount numeric(12,2),
    order_count  int,
    status       varchar(10)
)
WITH (
    orientation = column,
    colversion = 2.0,
    enable_hstore_opt = on,
    enable_binlog = true
)
DISTRIBUTE BY hash(sale_id);
```

插入测试数据。

```
INSERT INTO source_sales VALUES
(1,  '2024-06-01', 'North', 'phones', 1000.00, 2, 'valid'),
(2,  '2024-06-01', 'North', 'Laptops', 3000.00, 1, 'valid'),
(3,  '2024-06-01', 'East', 'phones', 2000.00, 3, 'valid'),
(4,  '2024-06-01', 'East', 'Tablets',  800.00, 1, 'valid'),
(5,  '2024-06-01', 'South', 'Laptops', 4500.00, 2, 'valid'),
(6,  '2024-06-02', 'North', 'phones', 1500.00, 1, 'valid'),
(7,  '2024-06-02', 'East', 'Laptops', 5000.00, 2, 'valid'),
(8,  '2024-06-02', 'South', 'Tablets', 1200.00, 1, 'valid'),
(9,  '2024-06-02', 'North', 'Laptops', 2800.00, 1, 'invalid'),   -- 无效订单，DWD 过滤
(10, '2024-06-02', 'East', 'phones', 1800.00, 2, 'pending'),    -- 待处理，DWD 过滤
(11, '2024-06-03', 'South', 'phones', 2200.00, 2, 'valid'),
(12, '2024-06-03', 'North', 'Tablets',  600.00, 1, 'valid');
```

1. **DWD层：明细清洗。**

   DWD层：过滤有效订单，字段透传。

   刷新频率：5 分钟（最频繁，保证明细层近实时）

   级联深度：1（直接建立在基表上）

   创建物化视图mv_dwd_sales。

   ```
   CREATE MATERIALIZED VIEW mv_dwd_sales
   REFRESH FAST
   START WITH (sysdate)
   EVERY (interval '5 min')
   ENABLE QUERY REWRITE
   WITH (
       orientation = column,
       colversion = 2.0,
       enable_hstore_opt = on
   )
   DISTRIBUTE BY hash(sale_id)
   AS
   SELECT
       sale_id,
       sale_date,
       region,
       product,
   ```

 版权所有 © 华为云计算技术有限公司

```
    total_amount,
    order_count
FROM source_sales
WHERE status = 'valid';
```

手动刷新。

```
REFRESH MATERIALIZED VIEW mv_dwd_sales;
```

验证DWD已过滤invalid和pending记录。

```
SELECT count(*) AS dwd_rows FROM mv_dwd_sales;
```

2. **DWS层：基础汇总。**

   DWS 层：按日期+区域聚合

   刷新频率：10 分钟（比DWD慢，因为汇总层容忍稍高延迟）

   级联深度：2（建立在mv_dwd_sales上）

   创建物化视图mv_dws_sales（MV-on-MV，DWS的基表是DWD MV而非原始基表）。

```
CREATE MATERIALIZED VIEW mv_dws_sales
REFRESH COMPLETE
START WITH (sysdate)
EVERY (interval '10 min')
ENABLE QUERY REWRITE
WITH (
    orientation = column,
    colversion = 2.0,
    enable_hstore_opt = on
)
DISTRIBUTE BY hash(sale_date, region)
AS
SELECT
    sale_date,
    region,
    sum(total_amount) AS amount,
    sum(order_count)  AS orders
FROM mv_dwd_sales
GROUP BY sale_date, region;
```

   手动刷新。

```
REFRESH MATERIALIZED VIEW mv_dws_sales;
```

   验证DWS数据。

```
SELECT sale_date, region, amount, orders
FROM mv_dws_sales
ORDER BY sale_date, region;
```

3. ADS层：应用报表。

   ADS 层：派生指标（客单价 = 总金额 / 订单数）

   刷新频率：30 分钟（应用层容忍最高延迟）

   级联深度：3（建立在mv_dws_sales上，达到默认上限）

   创建物化视图mv_ads_sales。

```
CREATE MATERIALIZED VIEW mv_ads_sales
    REFRESH COMPLETE
    START WITH (sysdate)
    EVERY (interval '30 min')
    ENABLE QUERY REWRITE
    WITH (
        orientation = column,
        colversion = 2.0,
        enable_hstore_opt = on
    )
    DISTRIBUTE BY hash(sale_date, region)
AS
SELECT
```

 版权所有 © 华为云计算技术有限公司

```
    sale_date,
    region,
    sum(amount) AS amount,
    sum(orders)  AS orders,
    sum(amount) / sum(orders) AS avg_order_value
FROM mv_dws_sales
GROUP BY sale_date, region;
```

手动刷新。

```
REFRESH MATERIALIZED VIEW mv_ads_sales;
```

验证ADS数据。

```
SELECT sale_date, region, amount, orders,
       round(avg_order_value, 2) AS avg_order_value
FROM mv_ads_sales
ORDER BY sale_date, region;
```

模式三：分区物化视图

场景描述：基表按时间分区，每天产生新分区数据。分区物化视图与基表分区建立映射关系，基表分区数据变更时只失效对应物化视图分区，刷新时只刷新失效分区，实现精准的增量分区刷新。

示例：

步骤1 创建基表daily_metrics（数据按天分区）并插入数据。

```
CREATE TABLE daily_metrics (
    metric_date  DATE NOT NULL,
    region       VARCHAR(50) NOT NULL,
    metric_name  VARCHAR(100) NOT NULL,
    value        NUMERIC(18,4) NOT NULL,
    record_count INT NOT NULL
)
WITH (
    orientation = column,
    colversion = 2.0,
    enable_hstore_opt = on
)
DISTRIBUTE BY HASH(metric_date)
PARTITION BY RANGE(metric_date) (
    PARTITION p20240101 VALUES LESS THAN('2024-01-02'),
    PARTITION p20240102 VALUES LESS THAN('2024-01-03'),
    PARTITION p20240103 VALUES LESS THAN('2024-01-04'),
    PARTITION p20240104 VALUES LESS THAN('2024-01-05'),
    PARTITION p20240105 VALUES LESS THAN('2024-01-06'),
    PARTITION p20240106 VALUES LESS THAN('2024-01-07'),
    PARTITION p20240107 VALUES LESS THAN('2024-01-08'),
    PARTITION p20240201 VALUES LESS THAN('2024-02-02'),
    PARTITION p20240202 VALUES LESS THAN('2024-02-03'),
    PARTITION p20240203 VALUES LESS THAN('2024-02-04'),
    PARTITION p20240204 VALUES LESS THAN('2024-02-05'),
    PARTITION p20240205 VALUES LESS THAN('2024-02-06'),
    PARTITION p20240206 VALUES LESS THAN('2024-02-07'),
    PARTITION p20240207 VALUES LESS THAN('2024-02-08')
);

INSERT INTO daily_metrics SELECT
    DATE '2024-01-01' + (i % 7),
    CASE (i % 3) WHEN 0 THEN 'North' WHEN 1 THEN 'East' ELSE 'South' END,
    'metric_' || (i % 50),
    (i % 10000) * 1.5,
    (i % 100) + 1
FROM generate_series(1, 10000) AS i;

INSERT INTO daily_metrics SELECT
    DATE '2024-02-01' + (i % 7),
    CASE (i % 3) WHEN 0 THEN 'North' WHEN 1 THEN 'East' ELSE 'South' END,
```


版权所有 © 华为云计算技术有限公司

```
    'metric_' || (i % 50),
    (i % 10000) * 1.5,
    (i % 100) + 1
FROM generate_series(1, 10000) AS i;
```

步骤2 创建分区物化视图mv_daily_agg。按天聚合，分区键与基表一对一映射。

```
CREATE MATERIALIZED VIEW mv_daily_agg
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE
WITH
(
    ORIENTATION = COLUMN,
    colversion = 2.0,
    enable_hstore_opt = on
)
DISTRIBUTE BY HASH(metric_date)
PARTITION BY metric_date
AS
SELECT
    metric_date,
    region,
    metric_name,
    SUM(value)        AS total_value,
    SUM(record_count) AS total_count
FROM daily_metrics
GROUP BY metric_date, region, metric_name;
```

创建分区物化视图mv_monthly_agg。按月汇总，多对一映射，使用date_trunc函数。

```
CREATE MATERIALIZED VIEW mv_monthly_agg
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE
WITH
(
    ORIENTATION = COLUMN,
    colversion = 2.0,
    enable_hstore_opt = on
)
DISTRIBUTE BY HASH(region)
PARTITION BY metric_date
AS
SELECT
    date_trunc('month', metric_date) as metric_date,
    region,
    SUM(total_value) AS total_value,
    SUM(total_count) AS total_count
FROM mv_daily_agg
GROUP BY metric_date, region;
```

步骤3 验证分区刷新。

1. 查看分区映射关系和状态。

```
SELECT matview, baserel, mapping, active
FROM pv_matview_detail
WHERE matview = 'mv_daily_agg';
```

```
   matview    |    baserel    |                        mapping                         | active
--------------+---------------+--------------------------------------------------------+--------
 mv_daily_agg | daily_metrics |                                                        | t
 mv_daily_agg | daily_metrics | p20240101 (2024-01-02 00:00:00) -> p1 (2024-01-02 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240102 (2024-01-03 00:00:00) -> p2 (2024-01-03 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240103 (2024-01-04 00:00:00) -> p3 (2024-01-04 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240104 (2024-01-05 00:00:00) -> p4 (2024-01-05 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240105 (2024-01-06 00:00:00) -> p5 (2024-01-06 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240106 (2024-01-07 00:00:00) -> p6 (2024-01-07 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240107 (2024-01-08 00:00:00) -> p7 (2024-01-08 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240201 (2024-02-02 00:00:00) -> p8 (2024-02-02 00:00:00)  | t
```

 版权所有 © 华为云计算技术有限公司

```
 mv_daily_agg | daily_metrics | p20240202 (2024-02-03 00:00:00) -> p9 (2024-02-03 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240203 (2024-02-04 00:00:00) -> p10 (2024-02-04 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240204 (2024-02-05 00:00:00) -> p11 (2024-02-05 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240205 (2024-02-06 00:00:00) -> p12 (2024-02-06 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240206 (2024-02-07 00:00:00) -> p13 (2024-02-07 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240207 (2024-02-08 00:00:00) -> p14 (2024-02-08 00:00:00) | t
(15 rows)
```

2. 向分区插入新数据。

```
INSERT INTO daily_metrics VALUES
    ('2024-01-03', 'East', 'metric_5', 999.0, 10),
    ('2024-01-03', 'South', 'metric_10', 888.0, 20);
```

3. 查看分区映射关系和状态。

```
SELECT matview, baserel, mapping, active
FROM pv_matview_detail
WHERE matview = 'mv_daily_agg';

   matview    |    baserel    |                        mapping                         | active
--------------+---------------+--------------------------------------------------------+--------
 mv_daily_agg | daily_metrics |                                                        | t
 mv_daily_agg | daily_metrics | p20240101 (2024-01-02 00:00:00) -> p1 (2024-01-02 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240102 (2024-01-03 00:00:00) -> p2 (2024-01-03 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240103 (2024-01-04 00:00:00) -> p3 (2024-01-04 00:00:00)  | f
 mv_daily_agg | daily_metrics | p20240104 (2024-01-05 00:00:00) -> p4 (2024-01-05 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240105 (2024-01-06 00:00:00) -> p5 (2024-01-06 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240106 (2024-01-07 00:00:00) -> p6 (2024-01-07 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240107 (2024-01-08 00:00:00) -> p7 (2024-01-08 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240201 (2024-02-02 00:00:00) -> p8 (2024-02-02 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240202 (2024-02-03 00:00:00) -> p9 (2024-02-03 00:00:00)  | t
 mv_daily_agg | daily_metrics | p20240203 (2024-02-04 00:00:00) -> p10 (2024-02-04 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240204 (2024-02-05 00:00:00) -> p11 (2024-02-05 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240205 (2024-02-06 00:00:00) -> p12 (2024-02-06 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240206 (2024-02-07 00:00:00) -> p13 (2024-02-07 00:00:00) | t
 mv_daily_agg | daily_metrics | p20240207 (2024-02-08 00:00:00) -> p14 (2024-02-08 00:00:00) | t
(15 rows)
```

4. 分区级刷新（只刷新失效分区），刷新后分区状态显示都是活跃状态。

```
REFRESH MATERIALIZED VIEW mv_daily_agg;
```

----结束

8.4.3 物化视图的监控与运维

物化视图的关键监控指标

表 8-11 关键监控指标

指标分类	监控指标	查询视图
数据时效性	• 最近刷新时间。 • 分区失效数量。	• PV_MATVIEW_DETAIL • PG_MATVIEW_PARTITION
查询性能	• 查询重写命中率。 • 重写后在物化视图上的查询性能。	PV_MATVIEW_REWRITE_HISTORY
刷新性能	• 刷新耗时。 • 刷新失败次数。	PV_MATVIEW_REFRESH_HISTORY
存储使用	物化视图和辅助表存储大小。	-

版权所有 © 华为云计算技术有限公司

物化视图常用监控系统视图

1. 物化视图属性信息：**PG_MATVIEW**视图。
2. 物化视图分区属性信息：**PG_MATVIEW_PARTITION**视图。
3. 物化视图与基表的依赖信息：**PG_MATVIEW_DEPEND**视图。
4. 综合显示属性，依赖和刷新信息：**PV_MATVIEW_DETAIL**视图。

物化视图常用监控 SQL

- 查看所有物化视图状态。
  ```
  SELECT * FROM pv_matview_detail ORDER BY matview;
  ```
- 查看刷新统计。
  ```
  SELECT
  matview,refresh_mode,refresh_method,mapping,active,refresh_count,last_refresh_start,last_refresh_end,
  (last_refresh_end-last_refresh_start) duration
  FROM pv_matview_detail ORDER BY matview;
  ```
- 查看刷新耗时的历史记录（需连接postgres数据库）。
  ```
  SELECT
    matview_name,
    matview_refresh_type,
    start_time,
    finish_time,
    duration,
    inserted_rows,
    deleted_rows,
    updated_rows,
    status
  FROM pv_matview_refresh_history
  WHERE start_time >= current_date
  ORDER BY duration DESC
  LIMIT 20;
  ```
- 查看查询重写统计。
  ```
  SELECT matview,active,rewrite_count FROM pv_matview_detail ORDER BY matview;
  ```
- 查看重写历史记录（需连接postgres数据库）。
  ```
  SELECT
    matview,
    rewrite_count,
    start_time,
    duration,
    left(query, 200) AS query_preview
  FROM pv_matview_rewrite_history
  WHERE start_time >= current_date - interval '7 days'
  ORDER BY start_time DESC
  LIMIT 50;
  ```
- 查看物化视图与基表分区映射关系。
  ```
  SELECT mapping as "base partition -> matview partition"  FROM pv_matview_detail  WHERE matview
  = 'test_base_mv1';
  ```

物化视图刷新任务运维

- 查看所有物化视图的定时刷新任务：
  ```
  SELECT * FROM scheduler.pg_task WHERE category_id = 'MatViewKernelTask';
  ```
- 修改物化视图的刷新计划：
  ```
  ALTER MATERIALIZED VIEW mv1 REFRESH START WITH('2024-06-01 00:00:00'::timestamptz)
  EVERY(interval '30 minutes');
  ```
- 停止物化视图的自动刷新：
  ```
  ALTER MATERIALIZED VIEW mv1 REFRESH ON DEMAND;
  ```
- 修复中断的物化视图定时任务

版权所有 © 华为云计算技术有限公司

当is_broken = true时，表示物化视图定时刷新任务已中断，可以通过以下方式修复：

```
ALTER MATERIALIZED VIEW mv1 REFRESH ON DEMAND;
ALTER MATERIALIZED VIEW mv1 REFRESH START WITH(current_timestamp + interval '5 minutes')
EVERY(interval '1 hour');
```

8.5 物化视图应用案例

8.5.1 电商平台实时销售报表案例

场景描述

某电商平台需要实时查看销售报表。

数据量： 销售明细表10亿+条记录

查询模式： 按日期、地区、商品类别聚合统计

业务需求

业务需求：实时销售大屏。

刷新频率：分钟级。

查询延迟要求：< 1秒。

解决方案

三层物化视图架构。

将数仓的DWD>DWS>ADS都创建为物化视图。通过增量计算实现实时的数据加工，满足电商平台需要实时查看销售报表的需求。

应用示例

步骤1 DWD层：数据清洗与标准化。

```
CREATE MATERIALIZED VIEW mv_dwd_sales
REFRESH FAST EVERY (interval '5 min')
ENABLE QUERY REWRITE
WITH (orientation = COLUMN, enable_hstore_opt = ON)
AS
SELECT
TO_DATE(sale_date) as sale_date,
region_id,
category_id,
product_id,
customer_id,
COUNT(*) as order_count,
SUM(sale_amount) as total_amount,
SUM(cost_amount) as total_cost,
COUNT(ORDER BY DISTINCT customer_id) as customer_count
FROM raw_sales
WHERE status = 'completed'
AND sale_date >= '2024-01-01'
GROUP BY TO_DATE(sale_date), region_id, category_id, product_id, customer_id;
```

步骤2 DWS层：多维度汇总。

版权所有 © 华为云计算技术有限公司

```
CREATE MATERIALIZED VIEW mv_dws_sales
REFRESH COMPLETE EVERY (interval '15 min')
ENABLE QUERY REWRITE
PARTITION BY date_trunc('month', sale_date)
WITH (orientation = COLUMN, enable_hstore_opt = ON)
AS
SELECT
sale_date,
region_id,
category_id,
SUM(order_count) as order_count,
SUM(total_amount) as total_amount,
SUM(total_cost) as total_cost,
SUM(customer_count) as customer_count
FROM mv_dwd_sales
GROUP BY sale_date, region_id, category_id;
```

步骤3 ADS层：即时查询。

```
CREATE MATERIALIZED VIEW mv_ads_daily_summary
REFRESH COMPLETE EVERY (interval '30 min')
ENABLE QUERY REWRITE
WITH (orientation = COLUMN, enable_hstore_opt = ON)
AS
SELECT
sale_date,
SUM(order_count) as total_orders,
SUM(total_amount) as total_sales,
SUM(total_cost) as total_cost,
SUM(customer_count) as total_customers,
SUM(total_amount - total_cost) as total_profit
FROM mv_dws_sales
GROUP BY sale_date;
```

步骤4 业务查询示例：今日销售概况。

```
SELECT * FROM mv_ads_daily_summary
WHERE sale_date = CURRENT_DATE;
```

----结束

8.5.2 网站用户行为实时分析案例

场景描述

某网站需要对用户行为数据进行实时统计。

业务需求

实时分析浏览网站的用户的行为数据，分析数据包括：按小时/天统计UV（独立访客数）、PV（页面浏览量）、访问时长、留存率等。

解决方案

使用物化视图的增量刷新及分区物化视图功能。

应用示例

步骤1 创建用户行为日志表。

```
CREATE TABLE user_behavior_log (
  user_id BIGINT,
  event_type VARCHAR(50),
  page_url VARCHAR(200),
  duration INT, -- 停留时长(秒)
```

 版权所有 © 华为云计算技术有限公司

```
  event_time TIMESTAMP
)
WITH (orientation = COLUMN)
PARTITION BY RANGE (event_time)
(
  PARTITION p0 VALUES LESS THAN ('2024-01-01'),
  PARTITION p1 VALUES LESS THAN ('2024-02-01'),
  PARTITION p2 VALUES LESS THAN (MAXVALUE)
);
```

步骤2 小时级统计物化视图，利用分区物化视图进行分区上卷计算。

```
CREATE MATERIALIZED VIEW mv_user_stats_daily
REFRESH COMPLETE EVERY (interval '10 min')
ENABLE QUERY REWRITE
WITH (orientation = COLUMN)
PARTITION BY (date_trunc('day', stat_hour))
AS
SELECT
  DATE_TRUNC('day', event_time) as stat_hour,
  event_type,
  page_url,
  COUNT(*) as pv, -- 页面访问次数
  COUNT(DISTINCT user_id) as uv, -- 独立访客数
  AVG(duration) as avg_duration, -- 平均停留时长
  COUNT(CASE WHEN duration > 60 THEN 1 END) as long_stay_count -- 停留>1分钟
FROM user_behavior_log
GROUP BY stat_hour, event_type, page_url;
```

步骤3 留存率统计。

1. 首次访问物化视图。

   ```
   CREATE MATERIALIZED VIEW mv_first_visit
   REFRESH COMPLETE EVERY (interval '10 min')
   ENABLE QUERY REWRITE
   AS
   SELECT
     user_id,
     DATE_TRUNC('day', MIN(event_time)) AS first_date
   FROM user_behavior_log
   GROUP BY user_id;
   ```

2. 留存明细物化视图（依赖mv_first_visit）。

   ```
   CREATE MATERIALIZED VIEW mv_retention_calc
   REFRESH COMPLETE EVERY (interval '10 min')
   ENABLE QUERY REWRITE
   AS
   SELECT
     fv.first_date,
     (DATE_TRUNC('day', u.event_time) - fv.first_date) AS days_diff,
     COUNT(DISTINCT u.user_id) AS user_count
   FROM user_behavior_log u
   JOIN mv_first_visit fv ON u.user_id = fv.user_id
   GROUP BY fv.first_date,
           (DATE_TRUNC('day', u.event_time) - fv.first_date);
   ```

3. 留存率物化视图（依赖mv_retention_calc + mv_first_visit）。

   ```
   CREATE MATERIALIZED VIEW mv_user_retention
   REFRESH COMPLETE EVERY (interval '1 hour')
   ENABLE QUERY REWRITE
   AS
   SELECT
     rc.first_date,
     rc.days_diff,
     rc.user_count,
     ROUND(100.0 * rc.user_count /
       (SELECT COUNT(*) FROM mv_first_visit fv2
        WHERE fv2.first_date = rc.first_date), 2
   ```

 版权所有 © 华为云计算技术有限公司

```
 ) AS retention_rate
FROM mv_retention_calc rc;
```

----结束

8.5.3 金融交易风控场景案例

场景描述

某金融平台需实时统计交易数据，进行交易风控规则计算。

业务需求

实时统计交易金额、笔数及异常交易检测。

解决方案

通过物化视图的增量刷新功能实现实时计算。

操作示例

步骤1 创建交易记录表。

```
CREATE TABLE transaction_log (
transaction_id BIGINT,
user_id BIGINT,
merchant_id BIGINT,
amount DECIMAL(15,2),
transaction_type VARCHAR(20),
risk_level VARCHAR(10),
transaction_time TIMESTAMP
) WITH (orientation = COLUMN);
```

步骤2 创建物化视图存储实时风控指标。

```
CREATE MATERIALIZED VIEW mv_risk_indicators
REFRESH COMPLETE EVERY (interval '1 min')
ENABLE QUERY REWRITE
WITH (orientation = COLUMN)
AS
SELECT
user_id,
DATE_TRUNC('hour', transaction_time) as stat_hour,
COUNT(*) as transaction_count,
SUM(amount) as total_amount,
AVG(amount) as avg_amount,
MAX(amount) as max_amount,
COUNT(CASE WHEN risk_level = 'HIGH' THEN 1 END) as high_risk_count,
COUNT(CASE WHEN risk_level = 'MEDIUM' THEN 1 END) as medium_risk_count
FROM transaction_log
GROUP BY user_id, DATE_TRUNC('hour', transaction_time);
```

步骤3 创建物化视图进行实时异常交易检测。

```
CREATE MATERIALIZED VIEW mv_anomaly_detection
REFRESH COMPLETE EVERY (interval '5 min')
ENABLE QUERY REWRITE
AS
SELECT
ri.user_id,
ri.stat_hour,
ri.transaction_count,
ri.total_amount,
ri.avg_amount,
ri.high_risk_count,
```


版权所有 © 华为云计算技术有限公司

```
CASE
WHEN ri.transaction_count > 100 THEN '异常: 高频交易'  -- 笔数异常检测规则
WHEN ri.total_amount > 100000 THEN '异常: 大额交易'    -- 金额异常检测规则
WHEN ri.high_risk_count > 5 THEN '异常: 高风险交易'    -- 异常交易检测规则
ELSE '正常'
END as anomaly_type
FROM mv_risk_indicators ri;
```

----结束

版权所有 © 华为云计算技术有限公司

9 DWS 用户自定义函数

9.1 PL/Java 语言函数

使用DWS数据库的PL/Java函数，用户可以使用自己喜欢的Java IDE编写Java方法，并将包含这些方法的jar文件安装到DWS数据库中，然后使用该方法。DWS PL/Java基于开源PL/Java 1.5.5开发，所使用的JRE版本为1.8.0_322。

使用限制

Java UDF可以实现一些Java逻辑计算，强烈建议不要在Java UDF中封装业务

- 强烈建议不要在Java函数中使用任何方式连接数据库，包括但不限于JDBC。
- 暂不支持的数据类型：除**表9-2**内容之外的数据类型，包括自定义类型，复杂数据类型（Java Array类及派生类）。
- 暂不支持UDAF，UDTF。

示例

使用PL/Java函数时，需要首先将Java方法的实现打包为jar包并且部署到数据库中，然后使用数据库管理员账号创建函数，考虑兼容性问题，请使用1.8.0_322版本的JRE进行编译。

步骤1 编译jar包。

Java方法的实现和出包可以借助IDE来实现，以下是一个通过命令行来进行编译和出包的简单的示例，通过这个简单示例可以创建出一个包含单个方法的jar包文件。

首先，编写一个Example.java文件（通常需要使用static关键字），在此文件中实现子字符串大写转换的方法，本例中类名为Example，方法名为upperString，内容如下：

```
public class Example
{
    public static String upperString (String text, int beginIndex, int endIndex)
    {
        return text.substring(beginIndex, endIndex).toUpperCase();
    }
}
```

然后，创建manifest.txt清单文件，文件内容如下：各参数的解释如**表9-1**所示：

 版权所有 © 华为云计算技术有限公司

```
Manifest-Version: 1.0
Main-Class: Example
Specification-Title: "Example"
Specification-Version: "1.0"
Created-By: 1.6.0_35-b10-428-11M3811
Build-Date: 08/14/2018 10:09 AM
```

表 9-1 manifest.txt 清单文件

参数项	描述	示例
Manifest-Version	表示清单文件的版本，固定为1.0，与JAR规范兼容。 无需修改，保持默认即可。	1.0
Main-Class	指定JAR包的入口类（包含main方法的类），即主应用程序类的类名。当通过java -jar执行JAR包时，会调用此类。	Example
Specification-Title	项目或模块的名称标识符，用于描述JAR包的功能，例如本例：子字符串大写转换。	Example
Specification-Version	项目的规范版本号，遵循语义化版本，用户自定义，例如首次发布为1.0，后续迭代为2.0。	1.0
Created-By	定义了生成此清单文件所基于的Java实现的版本和供应商，此属性由jar工具生成，格式为 JDK版本号-构建号。 • 默认由构建工具（如 Maven）自动填充，通常无需手动修改。 • 确保构建环境和运行环境的JDK版本一致。	1.6.0_35-b10-428-11M3811
Build-Date	JAR 的构建时间戳，用于追踪构建历史，便于回滚或调试。	08/14/2018 10:09 AM

最后，编译Java文件并打包得到javaudf-example.jar

```
javac Example.java
jar cfm javaudf-example.jar manifest.txt Example.class
```

须知

jar包的命名规则应符合JDK命名要求，如果含有非法字符，在部署或者使用函数时将出错。

步骤2 部署jar包。

Jar包首先需要放置到OBS服务器中，放置方法具体请参见《*对象存储服务控制台指南*》的上传文件章节。接着创建访问密钥AK/SK，获取访问密钥的具体步骤，请参见**创建访问密钥（AK和SK）**章节。登录数据库，使用具有管理员权限用户运行gs_extend_library函数，将文件导入到DWS中：

版权所有 © 华为云计算技术有限公司

```
SELECT gs_extend_library('addjar', 'obs://bucket/path/javaudf-example.jar
accesskey=access_key_value_to_be_replaced  secretkey=secret_access_key_value_to_be_replaced
region=region_name libraryname=example');
```

gs_extend_library函数如何使用请参见**管理jar包和文件**。函数中的AK/SK值，请用户根据实际获取值替换。region_name请用户根据实际所在的区域名称替换。bucket为OBS服务器上的桶名称。

步骤3 使用PL/Java函数。

首先，使用具有sysadmin权限的数据库用户（例如：dbadmin）登录数据库并创建java_upperstring函数如下：

```
CREATE FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER)
    RETURNS VARCHAR
    AS 'Example.upperString'
LANGUAGE JAVA;
```

说明

- 函数java_upperstring中定义的数据类型为DWS的数据类型。该数据类型需要和**步骤1**中Java定义的方法upperString中数据类型一一对应。DWS与Java数据类型的对应关系，请参见**表9-2**。
- AS子句用于指定该函数所调用的Java方法的类名和static方法名，格式为“类名.方法名”。该字段需要和**步骤1**中Java定义的类名和方法名一致。
- 使用PL/Java函数时，LANGUAGE字段应指定为JAVA。
- CREATE FUNCTION更多说明，请参见**创建函数**。

然后，执行java_upperstring函数：

```
SELECT java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
----------------------
 T
(1 row)
```

步骤4 授权普通用户使用PL/Java函数。

创建普通用户，名称为udf_user。

```
CREATE USER udf_user PASSWORD 'password';
```

授权普通用户udf_user对java_upperstring函数的使用权限。注意，此处需要把函数所在模式和函数的使用权限同时赋予给用户，用户才可以使用此函数。

```
GRANT ALL PRIVILEGES ON SCHEMA public TO udf_user;
GRANT ALL PRIVILEGES ON FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER) TO udf_user;
```

以普通用户udf_user登录数据库。

```
SET SESSION AUTHORIZATION udf_user PASSWORD 'password';
```

执行java_upperstring函数：

```
SELECT public.java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
----------------------
 T
(1 row)
```

 版权所有 © 华为云计算技术有限公司

步骤5 删除函数。

如果不再使用该函数可以进行删除：

```
DROP FUNCTION java_upperstring;
```

步骤6 卸载jar包。

使用gs_extend_library函数卸载jar包：

```
SELECT gs_extend_library('rmjar', 'libraryname=example');
```

----结束

SQL 定义与使用

- **管理jar包和文件**

  具有sysadmin权限的数据库用户可以使用gs_extend_library函数部署、查看和删除数据库中的jar包，语法如下：

  ```
  SELECT gs_extend_library('[action]', '[operation]');
  ```

  说明

  - **action**表明操作动作，值可以为：
    - ls表示查看数据库中jar包，会对各节点的文件进行MD5值一致性检查。
    - addjar表示将OBS服务器中的jar包部署到数据库中。
    - rmjar表示将数据库中的jar包删除。
    - lsclass表示查看数据库中每个jar包对应的节点和类名。
  - **operation**表明操作字符串，格式为：

    obs://[bucket]/[source_filepath] accesskey=[accesskey] secretkey=[secretkey] region=[region] libraryname=[libraryname]
    - bucket: OBS文件所属桶名，不可缺省。
    - source_filepath: OBS服务器上的文件路径，仅支持jar文件。
    - accesskey: obs服务获得的accesskey，不可缺省。
    - secret_key: obs服务获得的secretkey，不可缺省。
    - region: 自定义函数jar包所存放的OBS桶所属的区域，不可缺省。
    - libraryname: 自定义库名，此自定义命名用于DWS内对jar文件的调用。当action为addjar和rmjar时，该参数不可缺省，当action为ls时，该参数可以缺省。注意，自定义库名不允许含有/|;&$<>\'{}"()[]~*?!等字符。

- **创建函数**

  PL/Java函数通过CREATE FUNCTION语法创建，并且定义为LANGUAGE JAVA，且包含RETURNS和AS子句。

  - CREATE FUNCTION时指定所创建函数的名称，以及参数类型；
  - RETURNS子句用于指定该函数的返回类型；
  - AS子句与用于指定该函数所调用的Java方法的类名和static方法名，如果需要向Java方法传递NULL值作为入参，还需要指定该参数类型所对应的Java封装类名（详见**NULL值处理**）。
  - 更多语法说明，请参见CREATE FUNCTION。

    ```
    CREATE [ OR REPLACE ] FUNCTION function_name
    ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ]} [, ···] ])
    [ RETURNS rettype [ DETERMINISTIC ] ]
    LANGUAGE JAVA
    [
      { IMMUTABLE | STABLE | VOLATILE }
    ```

 版权所有 © 华为云计算技术有限公司

```
    | [ NOT ] LEAKPROOF
    | WINDOW
    | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT |STRICT }
    | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER |
AUTHID CURRENT_USER}
    | { FENCED }
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { {TO |=} value | FROM CURRENT}
] [···]
{
    AS 'class_name.method_name' ( { argtype } [, ···] )
}
```

- **使用函数**

  执行时，PL/Java会根据jar包名的字母序列，在所有部署的jar包中寻找函数指定的Java类，并调用首次找到的类中函数所指定的Java方法，并返回调用结果。

- **删除函数**

  PL/Java函数通过DROP FUNCTION语法删除函数。更多语法说明，请参见DROP FUNCTION。

  ```
  DROP FUNCTION [ IF EXISTS ] function_name [ ( [ {[ argmode ] [ argname ] argtype} [, ...] ] )
  [ CASCADE | RESTRICT ] ];
  ```

  需要注意的是，如果所删除的函数为重载函数（详见**重载函数**），则删除时需要指明该函数的参数类型，如为非重载函数，则可直接指定函数名进行删除。

- **函数授权**

  非sysadmin用户无法创建PL/Java函数，sysadmin用户可以赋予其他类型用户使用函数的权限。

  ```
  GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION {function_name ( [ {[ argmode ] [ arg_name ] arg_type} [, ...] ] )} [, ...]
       | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ];
  ```

基本数据类型映射关系

表 9-2 PL/Java 默认数据类型映射关系

DWS	Java
BOOLEAN	boolean
"char"	byte
bytea	byte[]
SMALLINT	short
INTEGER	int
BIGINT	long
FLOAT4	float
FLOAT8	double
CHAR	java.lang.String
VARCHAR	java.lang.String

版权所有 © 华为云计算技术有限公司

DWS	Java
TEXT	java.lang.String
name	java.lang.String
DATE	java.sql.Timestamp
TIME	java.sql.Time (stored value treated as local time)
TIMETZ	java.sql.Time
TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	java.sql.Timestamp

数组类型处理

DWS支持基础数组类型的转换，只需要在创建函数时在数据类型后追加“[]”即可，例如：

```
CREATE FUNCTION java_arrayLength(INTEGER[])
    RETURNS INTEGER
    AS 'Example.getArrayLength'
LANGUAGE JAVA;
```

Java代码类似于：

```
public class Example
{
    public static int getArrayLength(Integer[] intArray)
    {
        return intArray.length;
    }
}
```

调用函数java_arrayLength：

```
SELECT java_arrayLength(ARRAY[1, 2, 3]);
```

返回结果如下所示：

```
java_arrayLength
---------------------
3
(1 row)
```

NULL 值处理

对于默认与Java的简单类型进行映射转换的那些DWS数据类型，是无法处理NULL值的，如果希望在Java方法里能够获得并处理从DWS中传入的NULL值，可以使用Java的封装类，并通过以下方式在AS子句中指定该Java封装类：

```
CREATE FUNCTION java_countnulls(INTEGER[])
    RETURNS INTEGER
    AS 'Example.countNulls(java.lang.Integer[])'
LANGUAGE JAVA;
```

Java代码类似于：

```
public class Example
{
```

 版权所有 © 华为云计算技术有限公司

```
    public static int countNulls(Integer[] intArray)
    {
        int nullCount = 0;
        for (int idx = 0; idx < intArray.length; ++idx)
        {
            if (intArray[idx] == null)
            nullCount++;
        }
        return nullCount;
    }
}
```

调用函数java_countNulls：

```
SELECT java_countNulls(ARRAY[null, 1, null, 2, null]);
```

返回结果如下所示：

```
java_countNulls
---------------------
3
(1 row)
```

重载函数

PL/Java支持重载函数，因此可以创建同名函数，或者调用Java代码中的重载方法。步骤如下：

步骤1 创建重载函数

例如，在Java中可以实现两个方法名相同，输入参数类型不同的方法dummy(int) 和dummy(String)

```
public class Example
{
    public static int dummy(int value)
    {
        return value*2;
    }
    public static String dummy(String value)
    {
        return value;
    }
}
```

并在DWS中创建两个同名函数分别指定为上述两个方法：

```
CREATE FUNCTION java_dummy(INTEGER)
    RETURNS INTEGER
    AS 'Example.dummy'
LANGUAGE JAVA;

CREATE FUNCTION java_dummy(VARCHAR)
    RETURNS VARCHAR
    AS 'Example.dummy'
LANGUAGE JAVA;
```

步骤2 调用重载函数

在调用重载函数时，DWS会根据输入的参数类型去调用匹配该类型的Java方法。因此上述两个函数的调用结果如下所示：

```
SELECT java_dummy(5);
 java_dummy
-----------------
          10
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
SELECT java_dummy('5');
 java_dummy
---------------
5
(1 row)
```

需要注意的是，由于DWS对数据类型存在隐式转换的情况，因此建议在调用重载函数时，指定输入参数的类型，例如：

```
SELECT java_dummy(5::varchar);
 java_dummy
----------------
5
(1 row)
```

此时会优先匹配所指定的参数类型，如果不存在指定参数类型的Java方法，则会对参数进行隐式转换匹配转换后的参数类型对应的Java方法。

```
SELECT java_dummy(5::INTEGER);
 java_dummy
-----------------
10
(1 row)

DROP FUNCTION java_dummy(INTEGER);

SELECT java_dummy(5::INTEGER);
 java_dummy
----------------
5
(1 row)
```

须知

隐式转换的数据类型包括：

- 可以默认转换为INTEGER类型的包括：SMALLINT
- 可以默认转换为BIGINT类型的包括：SMALLINT、INTEGER
- 可以默认转换为BOOL类型的包括：TINYINT、SMALLINT、INTEGER、BIGINT
- 可以默认转换为TEXT类型的包括：CHAR、NAME、BIGINT、INTEGER、SMALLINT、 TINYINT、RAW、FLOAT4、FLOAT8、BPCHAR、VARCHAR、NVARCHAR2、DATE、TIMESTAMP、TIMESTAMPTZ、NUMERIC、SMALLDATETIME
- 可以默认转换为VARCHAR类型的包括：TEXT、CHAR、BIGINT、INTEGER、SMALLINT、TINYINT、RAW、FLOAT4、FLOAT8、BPCHAR、DATE、NVARCHAR2、TIMESTAMP、NUMERIC、SMALLDATETIME

步骤3 删除重载函数

对于重载函数，删除时需要指定函数的参数类型，否则无法删除。

```
DROP FUNCTION java_dummy(INTEGER);
```

----结束

相关 GUC 参数

- **udf_memory_limit**

 版权所有 © 华为云计算技术有限公司

系统级别的GUC参数，用于限制每个CN、DN执行UDF可以使用的物理内存量。具体参数内容可参考**udf_memory_limit**，可通过修改postgresql.conf文件进行配置，配置后需要重启数据库服务后才可生效。

须知

- udf_memory_limit是max_process_memory的一部分。每个CN、DN启动时，会预留（udf_memory_limit - 200MB）内存供UDF Worker进程使用。CN、DN和UDF Worker是不同的进程，但CN、DN自动少用一部分内存，把这部分内存节省下来供UDF Worker进程使用。

  例如：在某DN上把max_process_memory设置为10GB，udf_memory_limit设置为4GB，则此DN最多使用10GB - (4GB - 200MB)=6.2GB内存。即使用户没有执行任何UDF，则此DN也最多只能使用6.2GB内存。查询pv_total_memory_detail视图时可以发现，process_used_memory永远不会超过max_process_memory - (udf_memory_limit - 200MB)。
- UDF进程断连时会有报错提示。当出现“memory in UDF Work Process is limited by cgroup: [usage: xxx, max_usage_history: xxx, limit: xxx]”时，可以根据提示判断当前内存使用情况。报错中的usage表示UDF进程被KILL后剩余UDF进程的总物理内存使用量；max_usage_history表示UDF实例启动后历史上使用过的最大内存量，limit表示UDF进程使用内存的最大限制量。如果max_usage_history值接近limit值说明当前集群存在内存超过限制的风险，需要优化业务或按实际情况调整udf_memory_limit参数限制。
- 一个CN执行最简单的Java UDF函数，使用的物理内存量大约为50MB，用户可以根据自己Java函数的内存使用量和并发度设置此参数。新增此参数后，不再建议用户设置UDFWorkerMemHardLimit和FencedUDFMemoryLimit。
- 当UDF进程并发度过大，内存超出udf_memory_limit设置值时会导致进程退出等非预期情况，该场景下执行结果可能不可靠，强烈建议根据实际情况进行参数设置，保留足够内存余量。如果系统记录有/var/log/messages，可查看该日志文件是否存在因超过cgroup内存限制而造成内存不足。内存严重不足时，甚至可能导致UDF master进程退出，可以查看UDF日志进行分析，默认的UDF日志路径在$GAUSSLOG/cm/cm_agent/pg_log下。例如，出现以下日志时就说明内存资源严重不足，导致了UDF master进程退出，需要检查udf_memory_limit参数设置。

  0 [BACKEND] FATAL: poll() failed: Bad address, please check the parameter:udf_memory_limit to make sure there is enough memory.

- **FencedUDFMemoryLimit**

  会话级别的GUC参数，用户限制会话发起的单个Fenced UDF Worker进程的最大虚拟内存使用量，设置方法如下：

  ```
  SET FencedUDFMemoryLimit='512MB';
  ```

  该参数的取值范围为 (150MB, 1G]，当设置大于1G时会立即报错，当设置小于等于150MB时，则会在调用函数时报错。

 版权所有 © 华为云计算技术有限公司

须知

- FencedUDFMemoryLimit设置为0，表示不控制Fenced UDF Worker的虚拟内存使用量。
- 建议通过设置udf_memory_limit控制Fenced UDF Worker使用的物理内存量。不建议用户使用FencedUDFMemoryLimit，尤其在使用Java UDF时不建议用户设置此参数。但是如果用户非常清楚设置该参数带来的影响，可以参考下列信息进行设置：
  - C UDF worker启动之后，占用的虚拟内存约为200MB，占用的物理内存约为16MB。
  - Java UDF worker启动之后，占用的虚拟内存约为2.5GB，占用的物理内存约为50MB。

异常处理

如果在JVM中发生异常，PL/Java的异常处理机制会将异常时JVM的堆栈信息输出到客户端。

日志

PL/Java使用标准的Java Logger。 因此，用户可以通过如下方式记录日志：

```
Logger.getAnonymousLogger().config( "Time is " + new
Date(System.currentTimeMillis()));
```

初始化的Java Logger类会默认设置为CONFIG级别，对应为DWS的LOG级别。Java Logger类输出的日志消息都会重定向到DWS后端，并写入到服务器日志或显示在用户界面上。MPPDB服务器日志将记录LOG、WARNING、ERROR级别的信息，而SQL用户界面将显示WARNING和ERROR级别的日志消息。Java Logger级别与DWS的日志级别对应关系见下表。

表 9-3 PL/Java 日志级别

java.util.logging.Level	DWS 日志级别
SERVER	ERROR
WARNING	WARNING
CONFIG	LOG
INFO	INFO
FINE	DEBUG1
FINER	DEBUG2
FINEST	DEBUG3

用户可以通过以下方式更改Java Logger的记录级别。例如通过下面的Java代码修改Java Logger级别为SEVERE，此时再记录WARNING级别的日志时，日志消息（msg）就不会再写入到DWS日志中。

版权所有 © 华为云计算技术有限公司

```
Logger log = Logger.getAnonymousLogger();
Log.setLevel(Level.SEVERE);
log.log(Level.WARNING, msg);
```

安全问题

在DWS中，PL/Java是一种untrusted语言，PL/Java函数只能由数据库sysadmin用户进行创建，通过GRANT方式赋予其他用户使用权限（详见**函数授权**）。

同时PL/Java控制用户对文件系统的访问权限，不允许用户在Java方法中对大部分的系统文件进行读操作，不允许所有的写、删除和执行操作。

9.2 PL/pgSQL 语言函数

PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。

用PL/pgSQL创建的函数可以被用在任何可以使用内建函数的地方。例如，可以创建复杂条件的计算函数并且后面用它们来定义操作符或把它们用于索引表达式。

SQL是大部分数据库系统通用的查询语言，具备良好的可移植性且易于掌握。但每条语句均需由数据库服务器单独执行，这意味着客户端应用必须发送每一个查询到数据库服务器、等待它被处理、接收并处理结果、做一些计算，然后发送更多查询给服务器。如果客户端和数据库服务器不在同一台机器上，所有这些会引起进程间通信并且将带来网络负担。

通过PL/pgSQL，可以将一整块计算和一系列查询分组在数据库服务器内部，这样就有了一种过程语言的能力并且使SQL更易用，同时能节省的客户端/服务器通信开销。

- 客户端和服务器之间的额外往返通信被消除。
- 客户端不需要的中间结果不必被整理或者在服务器和客户端之间传送。
- 多轮的查询解析可以被避免。

PL/pgSQL可以使用SQL中所有的数据类型、操作符和函数。

应用PL/pgSQL创建函数的语法为CREATE FUNCTION。正如前面所说，PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。其应用方法与**DWS存储过程**相似，只是存储过程无返回值，函数有返回值。

版权所有 © 华为云计算技术有限公司

10 DWS 存储过程

10.1 DWS 存储过程简介

什么是 DWS 存储过程

商业规则和业务逻辑可以通过程序存储在DWS中，这个程序就是存储过程。

存储过程是SQL、PL/SQL及Java语句的组合。存储过程使执行商业规则的代码可以从应用程序中移动到数据库。从而，代码存储一次能够被多个程序使用。

存储过程的创建及调用方法请参考**CREATE PROCEDURE**。

PL/pgSQL语言函数节所提到的PL/pgSQL语言创建的函数与存储过程的应用方法相同。下面各节中，除非特别声明，否则内容通用于存储过程和PL/pgSQL语言函数。

DWS 存储过程数据类型

数据类型是一组值的集合以及定义在这个值集上的一组操作。DWS数据库是由表的集合组成的，而各表中的列定义了该表，每一列都属于一种数据类型，DWS根据数据类型有相应函数对其内容进行操作，例如DWS可对数值型数据进行加、减、乘、除操作。

10.2 DWS 存储过程数据类型转换

数据库中允许有些数据类型进行隐式类型转换（赋值、函数调用的参数等），有些数据类型间不允许进行隐式数据类型转换，可尝试使用DWS提供的类型转换函数，例如CAST进行数据类型强转。

DWS数据库常见的隐式类型转换，请参见**表10-1**。

须知

DWS支持的DATE的效限范围是：公元前4713年到公元294276年。

 版权所有 © 华为云计算技术有限公司

表 10-1 隐式类型转换表

原始数据类型	目标数据类型	备注
CHAR	VARCHAR2	-
CHAR	NUMBER	原数据必须由数字组成。
CHAR	DATE	原数据不能超出合法日期范围。
CHAR	RAW	-
CHAR	CLOB	-
VARCHAR2	CHAR	-
VARCHAR2	NUMBER	原数据必须由数字组成。
VARCHAR2	DATE	原数据不能超出合法日期范围。
VARCHAR2	CLOB	-
NUMBER	CHAR	-
NUMBER	VARCHAR2	-
DATE	CHAR	-
DATE	VARCHAR2	-
RAW	CHAR	-
RAW	VARCHAR2	-
CLOB	CHAR	-
CLOB	VARCHAR2	-
CLOB	NUMBER	原数据必须由数字组成。
INT4	CHAR	-

10.3 DWS 存储过程数组和 record

10.3.1 数组

数组类型的使用

在使用数组之前，需要自定义一个数组类型。

在存储过程中紧跟AS关键字后面定义数组类型。定义方法为：

```
TYPE array_type IS VARRAY(size) OF data_type [NOT NULL];
```

其中：

版权所有 © 华为云计算技术有限公司

- array_type：要定义的数组类型名。
- VARRAY：表示要定义的数组类型。
- size：取值为正整数，表示可以容纳的成员的最大数量。
- data_type：要创建的数组中成员的类型。
- NOT NULL: 可选约束，可以约束该数组中的元素均不为NULL。

说明

- 在DWS中，数组会自动增长，访问越界会返回一个NULL，不会报错。越界写入数组会提示：Subscript outside of limit.
- 在存储过程中定义的数组类型，其作用域仅在该存储过程中。
- 建议选择上述定义方法的一种来自定义数组类型，当同时使用两种方法定义同名的数组类型时，DWS会优先选择存储过程中定义的数组类型来声明数组变量。

DWS 8.1.0之前版本， 由于数组可以自动增长，系统不会校验数组越界以及数组元素的长度限制。当前版本为了兼容Oracle的用法增加了相关约束。如果已经存在越界写入等场景，可通过在**behavior_compat_options**参数中配置varray_verification，来兼容之前不校验的行为。

示例：

```
--演示在存储过程中对数组声明操作。
CREATE OR REPLACE PROCEDURE array_proc
AS
	TYPE ARRAY_INTEGER IS VARRAY(1024) OF INTEGER;--定义数组类型
	TYPE ARRAY_INTEGER_NOT_NULL IS VARRAY(1024) OF INTEGER NOT NULL;--定义非空数组类型
	ARRINT ARRAY_INTEGER := ARRAY_INTEGER();  --声明数组类型的变量
BEGIN
	ARRINT.extend(10);
	FOR I IN 1..10 LOOP
		ARRINT(I) := I;
	END LOOP;
	DBMS_OUTPUT.PUT_LINE(ARRINT.COUNT);
	DBMS_OUTPUT.PUT_LINE(ARRINT(1));
	DBMS_OUTPUT.PUT_LINE(ARRINT(10));
	DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.FIRST));
	DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.last));
END;
/

--调用该存储过程。
CALL array_proc();
10
1
10
1
10
--删除存储过程。
DROP PROCEDURE array_proc;
```

ROWTYPE 类型数组的声明及使用

除了上述示例中普通数组以及not null数组的声明及使用，数组中同时支持rowtype类型数组的声明及使用。

示例：

```
--演示在存储过程中对数组COUNT函数的用法。
CREATE TABLE tbl (a int, b int);
INSERT INTO tbl VALUES(1, 2),(2, 3),(3, 4);
CREATE OR REPLACE PROCEDURE array_proc
AS
```

 版权所有 © 华为云计算技术有限公司

```
    CURSOR all_tbl IS SELECT * FROM tbl ORDER BY a;
    TYPE tbl_array_type IS varray(50) OF tbl%rowtype; --定义rowtype类型的数组，tbl是任意表。
    tbl_array tbl_array_type;
    tbl_item tbl%rowtype;
    inx1 int;
BEGIN
    tbl_array := tbl_array_type();
    inx1 := 0;
    FOR tbl_item IN all_tbl LOOP
        inx1 := inx1 + 1;
        tbl_array(inx1) := tbl_item;
    END LOOP;
    WHILE inx1 IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('tbl_array(inx1).a=' || tbl_array(inx1).a || ' tbl_array(inx1).b=' ||
tbl_array(inx1).b);
        inx1 := tbl_array.PRIOR(inx1);
    END LOOP;
END;
/
```

执行结果：

```
call array_proc();
tbl_array(inx1).a=3 tbl_array(inx1).b=4
tbl_array(inx1).a=2 tbl_array(inx1).b=3
tbl_array(inx1).a=1 tbl_array(inx1).b=2
```

数组相关函数使用

在DWS中，提供了类似Oracle相关的数组函数的支持，可以通过以下函数获取数组的一些属性或者对数组内容进行操作。

COUNT

COUNT函数可以返回当前数组元素的数量，仅统计初始化过的元素或者经过EXTEND函数扩展后的元素。

用法如下：

varray.COUNT或*varray*.COUNT()

示例：

```
--演示在存储过程中对数组COUNT函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    v_varray.extend;
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count=3
v_varray.count=4
```

FIRST 和 LAST

FIRST函数可以返回第一个元素的下标。LAST函数可以返回最后一个元素的下标。

版权所有 © 华为云计算技术有限公司

用法如下：

varray.FIRST或*varray*.FIRST()

varray.LAST或*varray*.LAST()

示例：

```
--演示在存储过程中对数组FIRST和LAST函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    DBMS_OUTPUT.PUT_LINE('v_varray.first=' || v_varray.first);
    DBMS_OUTPUT.PUT_LINE('v_varray.last=' || v_varray.last);
END;
/
```

执行结果：

```
call test_varray();
v_varray.first=1
v_varray.last=3
```

EXTEND

说明

EXTEND函数主要是为了兼容Oracle的两种用法。在DWS中，数组会自动增长，EXTEND函数不是必须的。如果是新写的存储过程，完全没有必要使用EXTEND函数。

EXTEND函数可以对数组进行扩展，EXTEND有两种调用方式。

- 方式一：

  EXTEND包含一个整型入参，表示数组向后扩展size大小的长度，EXTEND后COUNT和LAST函数的值也会有相应的变化。

  用法如下：

  varray.EXTEND(size)

  其中*varray*.EXTEND这种无参的调用默认会向后扩展1位等价于*varray*.EXTEND(1)

- 方式二：

  EXTEND包含两个整型入参，第一个参数代表向后扩展size大小的长度，第二个参数表示扩展后的数组元素值和之下标为index的元素相同。

  用法如下：

  varray.EXTEND(size, index)

示例：

```
--演示在存储过程中对数组EXTEND函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    v_varray.extend(3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    v_varray.extend(2,3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count=' || v_varray.count);
    DBMS_OUTPUT.PUT_LINE('v_varray(7)=' || v_varray(7));
```


版权所有 © 华为云计算技术有限公司

```
    DBMS_OUTPUT.PUT_LINE('v_varray(8)=' || v_varray(8));
END;
/
```

执行结果：

```
call test_varray();
v_varray.count=6
v_varray.count=8
v_varray(7)=3
v_varray(8)=3
```

NEXT 和 PRIOR

NEXT函数和PRIOR函数主要用于数组的循环遍历中，NEXT函数会根据入参index值，返回下一个数组元素的下标，若已经到达数组下标最大值则返回NULL。PRIOR函数会根据入参index值，返回上一个数组元素的下标，若已经到达数组下标最小值则返回NULL。

用法如下：

varray.NEXT(index)

varray.PRIOR(index)

示例：

```
--演示在存储过程中对数组NEXT和PRIOR函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
    i int;
BEGIN
    v_varray := varray_type(1, 2, 3);

    i := v_varray.COUNT;
    WHILE i IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('test prior v_varray('||i||')=' || v_varray(i));
        i := v_varray.PRIOR(i);
    END LOOP;

    i := 1;
    WHILE i IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('test next v_varray('||i||')=' || v_varray(i));
        i := v_varray.NEXT(i);
    END LOOP;
END;
/
```

执行结果：

```
call test_varray();
test prior v_varray(3)=3
test prior v_varray(2)=2
test prior v_varray(1)=1
test next v_varray(1)=1
test next v_varray(2)=2
test next v_varray(3)=3
```

EXISTS

EXISTS函数可以判断数组下标是否存在。

用法如下：

版权所有 © 华为云计算技术有限公司

varray.EXISTS(index)

示例：

```
--演示在存储过程中对数组EXISTS函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3);
    IF v_varray.EXISTS(1) THEN
        DBMS_OUTPUT.PUT_LINE('v_varray.EXISTS(1)');
    END IF;
    IF NOT v_varray.EXISTS(10) THEN
        DBMS_OUTPUT.PUT_LINE('NOT v_varray.EXISTS(10)');
    END IF;
END;
/
```

执行结果：

```
call test_varray();
v_varray.EXISTS(1)
NOT v_varray.EXISTS(10)
```

TRIM

TRIM函数可以从数组尾部删除指定数量的元素。

用法如下：

varray.TRIM(size)

其中*varray*.TRIM这种无参的调用会默认入参为1，等价于*varray*.TRIM(1)

示例：

```
--演示在存储过程中对数组TRIM函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    v_varray.trim(3);
    DBMS_OUTPUT.PUT_LINE('v_varray.count' || v_varray.count);
    v_varray.trim;
    DBMS_OUTPUT.PUT_LINE('v_varray.count:' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count:2
v_varray.count:1
```

DELETE

DELETE函数可以从数组删除数组中的所有元素。

用法如下：

varray.DELETE或*varray*.DELETE()

示例：

版权所有 © 华为云计算技术有限公司

```
--演示在存储过程中对数组DELETE函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    v_varray.delete;
    DBMS_OUTPUT.PUT_LINE('v_varray.count:' || v_varray.count);
END;
/
```

执行结果：

```
call test_varray();
v_varray.count:0
```

LIMIT

LIMIT函数可以返回数组的最大长度限制。

用法如下：

varray.LIMIT或*varray*.LIMIT()

示例：

```
--演示在存储过程中对数组LIMIT函数的用法。
CREATE OR REPLACE PROCEDURE test_varray
AS
    TYPE varray_type IS VARRAY(20) OF INT;
    v_varray varray_type;
BEGIN
    v_varray := varray_type(1, 2, 3, 4, 5);
    DBMS_OUTPUT.PUT_LINE('v_varray.limit:' || v_varray.limit);
END;
/
```

执行结果：

```
call test_varray();
v_varray.limit:20
```

10.3.2 record

record 类型的变量

创建一个record变量的方式：

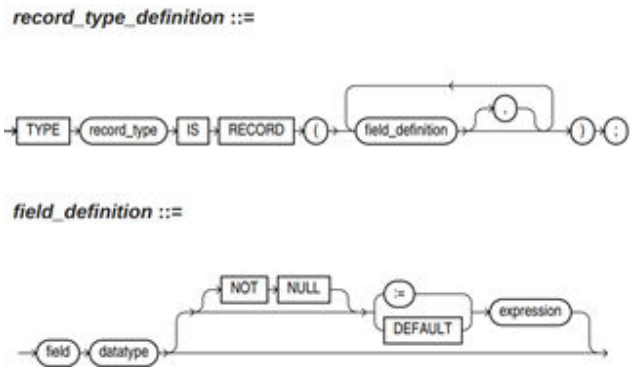
定义一个record类型 ， 然后使用该类型来声明一个变量。

语法

record类型的语法参见**图10-1**。

 版权所有 © 华为云计算技术有限公司

图 10-1 record 类型的语法



对以上语法格式的解释如下：

- record_type：声明的类型名称。
- field：record类型中的成员名称。
- datatype：record类型中成员的类型。
- expression：设置默认值的表达式。

说明

record类型的变量需遵循以下约束：

- 在函数或存储过程的声明阶段，声明一个record类型，并且可以在该类型中定义成员变量。
- 一个record变量到另一个record变量的赋值。
- SELECT INTO和FETCH向一个record类型的变量中赋值。
- 将一个NULL值赋值给一个record变量。
- 不支持INSERT和UPDATE语句使用record变量进行插入数据和更新数据。
- 如果成员有复合类型，在声明阶段不支持指定默认值，该行为与声明阶段的变量一样。

示例

```
下面存储过程中用到的表定义如下：
CREATE TABLE emp_rec
(
    empno           numeric(4,0),
    ename           character varying(10),
    job             character varying(9),
    mgr             numeric(4,0),
    hiredate        timestamp(0) without time zone,
    sal             numeric(7,2),
    comm            numeric(7,2),
    deptno          numeric(2,0)
)
with (orientation = column,compression=middle)
distribute by hash (sal);
\d emp_rec
              Table "public.emp_rec"
  Column  |            Type             | Modifiers
----------+-----------------------------+-----------
 empno    | numeric(4,0)                | not null
 ename    | character varying(10)       |
 job      | character varying(9)        |
 mgr      | numeric(4,0)                |
 hiredate | timestamp(0) without time zone |
```


版权所有 © 华为云计算技术有限公司

```
 sal      | numeric(7,2)                  |
 comm     | numeric(7,2)                  |
 deptno   | numeric(2,0)                  |

--演示在存储过程中对数组进行操作。
CREATE OR REPLACE FUNCTION regress_record(p_w VARCHAR2)
RETURNS
VARCHAR2  AS $$
DECLARE

  --声明一个record类型。
  type rec_type is record (name  varchar2(100), epno int);
  employer rec_type;

  --使用%type声明record类型。
  type rec_type1 is record (name  emp_rec.ename%type, epno int not null :=10);
  employer1 rec_type1;

  --声明带有默认值的record类型。
  type rec_type2 is record (
        name varchar2 not null := 'SCOTT',
        epno int not null :=10);
   employer2 rec_type2;
   CURSOR C1 IS  select ename,empno from emp_rec order by 1 limit 1;

BEGIN
     --对一个record类型的变量的成员赋值。
    employer.name := 'WARD';
    employer.epno := 18;
    raise info 'employer name: % , epno:%', employer.name, employer.epno;

    --将一个record类型的变量赋值给另一个变量。
    employer1 := employer;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --将一个record类型变量赋值为NULL。
    employer1 := NULL;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --获取record变量的默认值。
    raise info 'employer2 name: % ,epno: %', employer2.name, employer2.epno;

     --在for循环中使用record变量
     for employer in select ename,empno from emp_rec order by 1  limit 1
         loop
             raise info 'employer name: % , epno: %', employer.name, employer.epno;
         end loop;

     --在select into 中使用record变量。
     select ename,empno  into employer2 from emp_rec order by 1 limit 1;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;

     --在cursor中使用record变量。
     OPEN C1;
     FETCH C1 INTO employer2;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;
     CLOSE C1;
     RETURN employer.name;
END;
$$
LANGUAGE plpgsql;

--调用该存储过程。
CALL regress_record('abc');
INFO:  employer name: WARD , epno:18
INFO:  employer1 name: WARD , epno: 18
INFO:  employer1 name: <NULL> , epno: <NULL>
INFO:  employer2 name: SCOTT ,epno: 10
```


版权所有 © 华为云计算技术有限公司

```
--删除存储过程。
DROP PROCEDURE regress_record;
```

10.4 DWS 存储过程声明语法

基本结构

PL/SQL块中可以包含子块，子块可以位于PL/SQL中任何部分。PL/SQL块的结构如下：

- 声明部分：声明PL/SQL用到的变量，类型及游标，以及局部的存储过程和函数。
  ```
  DECLARE
  ```

  说明

  不涉及变量声明时声明部分可以没有。

  - 对匿名块来说，没有变量声明部分时，可以省去DECLARE关键字。
  - 对存储过程来说，没有DECLARE， AS相当于DECLARE。即便没有变量声明的部分，关键字AS也必须保留。
- 执行部分：过程及SQL语句，程序的主要部分。必选。
  ```
  BEGIN
  ```
- 执行异常部分：错误处理。可选。
  ```
  EXCEPTION
  ```
- 结束
  ```
  END;
  /
  ```

须知

禁止在PL/SQL块中使用连续的Tab，连续的Tab可能会造成在使用gsql工具带“-r”参数执行PL/SQL块时出现异常。

PL/SQL 块分类

PL/SQL块可以分为以下几类：

- 匿名块：动态构造，只能执行一次。语法请参考**匿名块**。
- 子程序：存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

匿名块

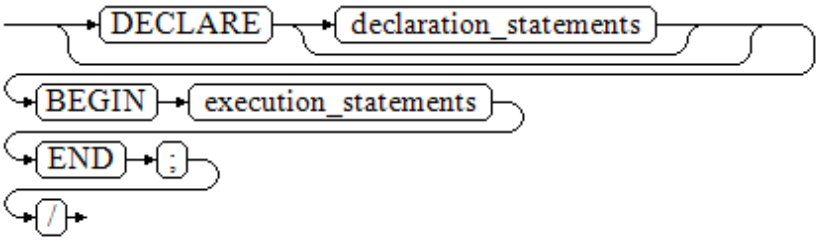
匿名块（Anonymous Block）一般用于不频繁执行的脚本或不重复进行的活动。它们在一个会话中执行，并不被存储。

语法

匿名块的语法参见**图10-2**。

版权所有 © 华为云计算技术有限公司

图 10-2 anonymous_block::=



对以上语法图的解释如下：

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。输入“/”按回车执行它。

  须知

  最后的结束符“/”必须独占一行，不能直接跟在END后面。

- 声明部分包括变量定义、类型、游标定义等。
- 最简单的匿名块不执行任何命令。但一定要在任意实施块里至少有一个语句，甚至是一个NULL语句。

示例

下面列举了基本的匿名块程序：

```
--空语句块
BEGIN
    NULL;
END;
/

--将信息打印到控制台：
BEGIN
    dbms_output.put_line('hello world!');
END;
/

--将变量内容打印到控制台：
DECLARE
    my_var VARCHAR2(30);
BEGIN
    my_var :='world';
    dbms_output.put_line('hello'||my_var);
END;
/
```

子程序

存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

版权所有 © 华为云计算技术有限公司

10.5 DWS 存储过程基本语句

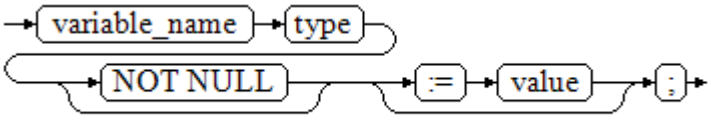
定义变量

介绍PL/SQL中变量的声明，以及该变量在代码中的作用域。

变量声明

变量声明语法请参见图10-3。

图 10-3 declare_variable::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- type，为变量类型。
- value，是该变量的初始值（如果不给定初始值，则初始为NULL）。value也可以是表达式。

示例

```
DECLARE
    emp_id  INTEGER := 7788;  --定义变量并赋值
BEGIN
    emp_id := 5*7784;  --变量赋值
END;
/
```

变量类型除了支持基本类型，还可使用%TYPE和%ROWTYPE去声明一些与其他表字段或表结构本身相关的变量。

%TYPE属性

%TYPE主要用于声明某个与其他变量类型（例如，表中某列的类型）相同的变量。假如想定义一个my_name变量，它的变量类型与employee的firstname类型相同，可使用如下定义：

```
my_name employee.firstname%TYPE
```

这样定义可以带来两个好处，首先，不用预先知道employee 表的firstname类型具体是什么。其次，即使之后firstname类型有了变化，也不需要再次修改my_name的类型。

%ROWTYPE属性

%ROWTYPE属性主要用于对一组数据的类型声明，用于存储表中的一行数据，或从游标匹配的结果。假如需要一组数据，该组数据的字段名称与字段类型都与employee表相同。可以通过如下定义：

```
my_employee employee%ROWTYPE
```


版权所有 © 华为云计算技术有限公司

须知

多个CN的环境下，存储过程中无法声明临时表的%ROWTYPE及%TYPE属性。因为临时表仅在当前session有效，在编译阶段其他CN无法看到当前CN的临时表。故多个CN的环境下，会提示该临时表不存在。

变量作用域

变量的作用域表示变量在代码块中的可访问性和可用性。只有在它的作用域内，变量才有效。

- 变量必须在declare部分声明，即必须建立BEGIN-END块。块结构也强制变量必须先声明后使用，即变量在过程内有不同作用域、不同的生存期。
- 同一变量可以在不同的作用域内定义多次，内层的定义会覆盖外层的定义。
- 在外部块定义的变量，可以在嵌套块中使用。但外部块不能访问嵌套块中的变量。

示例

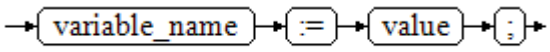
```
DECLARE
    emp_id  INTEGER :=7788;  --定义变量并赋值
    outer_var  INTEGER :=6688;  --定义变量并赋值
BEGIN
    DECLARE
        emp_id INTEGER :=7799;  --定义变量并赋值
        inner_var  INTEGER :=6688;  --定义变量并赋值
    BEGIN
        dbms_output.put_line('inner emp_id ='||emp_id);  --显示值为7799
        dbms_output.put_line('outer_var ='||outer_var);  --引用外部块的变量
    END;
    dbms_output.put_line('outer emp_id ='||emp_id);  --显示值为7788
END;
/
```

赋值语句

语法

给变量赋值的语法请参见**图10-4**。

图 10-4 assignment_value::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- value，可以是值或表达式。值value的类型需要和变量variable_name的类型兼容才能正确赋值。

示例

```
DECLARE
    emp_id  INTEGER := 7788;--赋值
BEGIN
    emp_id := 5;--赋值
    emp_id := 5*7784;
```


版权所有 © 华为云计算技术有限公司

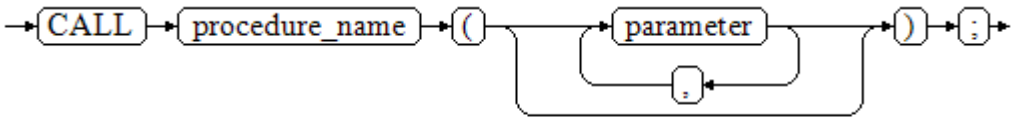
```
END;
/
```

调用语句

语法

调用一个语句的语法请参见图10-5。

图 10-5 call_clause::=



对以上语法格式的解释如下：

- procedure_name，为存储过程名。
- parameter，为存储过程的参数，可以没有或者有多个参数。

示例

```
--创建存储过程proc_staffs
CREATE OR REPLACE PROCEDURE proc_staffs
(
section     NUMBER(6),
salary_sum out NUMBER(8,2),
staffs_count out INTEGER
)
IS
BEGIN
SELECT sum(salary), count(*) INTO salary_sum, staffs_count FROM staffs where section_id = section;
END;
/

--创建存储过程proc_return.
CREATE OR REPLACE PROCEDURE proc_return
AS
v_num NUMBER(8,2);
v_sum INTEGER;
BEGIN
proc_staffs(30, v_sum, v_num);  --调用语句
dbms_output.put_line(v_sum||'#'||v_num);
RETURN;  --返回语句
END;
/

--调用存储过程proc_return.
CALL proc_return();

--清除存储过程
DROP PROCEDURE proc_staffs;
DROP PROCEDURE proc_return;

--创建函数func_return.
CREATE OR REPLACE FUNCTION func_return returns void
language plpgsql
AS $$
DECLARE
v_num INTEGER := 1;
BEGIN
dbms_output.put_line(v_num);
```


版权所有 © 华为云计算技术有限公司

```
RETURN;  --返回语句
END $$;


-- 调用函数func_return
CALL func_return();
1

-- 清除函数
DROP FUNCTION func_return;
```

10.6 DWS 存储过程动态语句

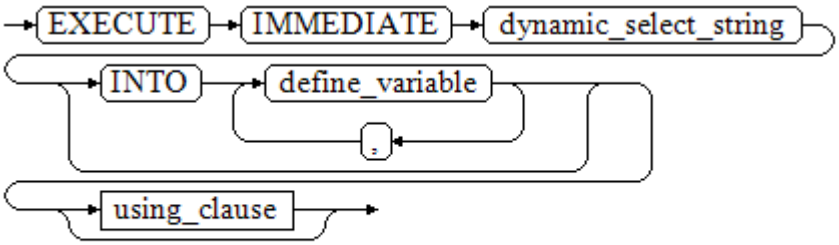
10.6.1 执行动态查询语句

介绍执行动态查询语句。DWS提供两种方式：使用EXECUTE IMMEDIATE、OPEN FOR实现动态查询。前者通过动态执行SELECT语句，后者结合了游标的使用。当需要将查询的结果保存在一个数据集用于提取时，可使用OPEN FOR实现动态查询。

EXECUTE IMMEDIATE

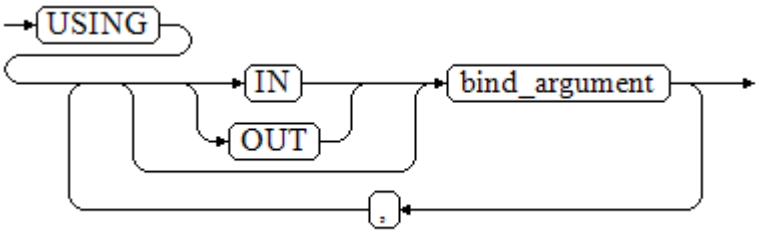
语法图请参见图10-6。

图 10-6 EXECUTE IMMEDIATE dynamic_select_clause::=



using_clause子句的语法图参见图10-7。

图 10-7 using_clause-1



对以上语法格式的解释如下：

- define_variable，用于指定存放单行查询结果的变量。
- USING IN bind_argument，用于指定存放传递给动态SQL值的变量，即在dynamic_select_string中存在占位符时使用。

版权所有 © 华为云计算技术有限公司

- USING OUT bind_argument，用于指定存放动态SQL返回值的变量。

须知

- 查询语句中，into和out不能同时存在；
- 占位符命名以“:”开始，后面可跟数字、字符或字符串，与USING子句的bind_argument一一对应；
- bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象，即不支持使用bind_argument为动态SQL语句传递模式对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause；
- 动态PL/SQL块允许出现重复的占位符，即相同占位符只能与USING子句的一个bind_argument按位置对应。

示例

```
--从动态语句检索值（INTO 子句）：
DECLARE
   staff_count  VARCHAR2(20);
BEGIN
   EXECUTE IMMEDIATE 'select count(*) from staffs'
      INTO staff_count;
   dbms_output.put_line(staff_count);
END;
/

--传递并检索值（INTO子句用在USING子句前）：
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
   EXECUTE IMMEDIATE 'select first_name, salary from staffs where staff_id = :1'
      INTO first_name, salary
      USING IN staff_id;
   dbms_output.put_line(first_name || ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

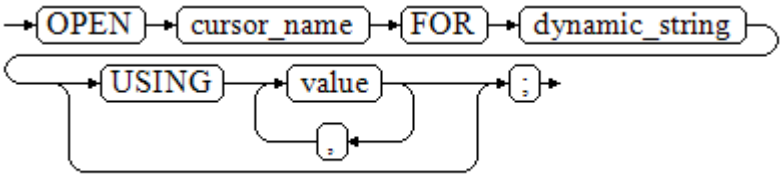
OPEN FOR

动态查询语句还可以使用OPEN FOR打开动态游标来执行。

语法参见**图10-8**。

版权所有 © 华为云计算技术有限公司

图 10-8 open_for::=



参数说明：

- cursor_name：要打开的游标名。
- dynamic_string：动态查询语句。
- USING value：在dynamic_string中存在占位符时使用。

游标的使用请参考**DWS存储过程游标**。

示例

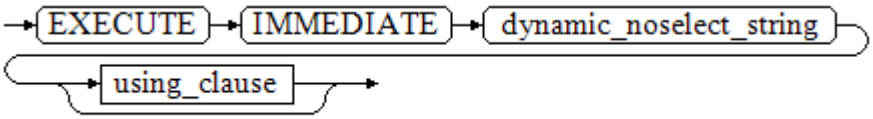
```
DECLARE
   name          VARCHAR2(20);
   phone_number  VARCHAR2(20);
   salary        NUMBER(8,2);
   sqlstr        VARCHAR2(1024);

   TYPE app_ref_cur_type IS REF CURSOR;  --定义游标类型
   my_cur app_ref_cur_type;  --定义游标变量

BEGIN
   sqlstr := 'select first_name,phone_number,salary from staffs
       where section_id = :1';
   OPEN my_cur FOR sqlstr USING '30';  --打开游标, using是可选的
   FETCH my_cur INTO name, phone_number, salary; --获取数据
   WHILE my_cur%FOUND LOOP
         dbms_output.put_line(name||'#'||phone_number||'#'||salary);
         FETCH my_cur INTO name, phone_number, salary;
   END LOOP;
   CLOSE my_cur;   --关闭游标
END;
/
```

10.6.2 执行动态非查询语句

语法

语法请参见**图10-9**。

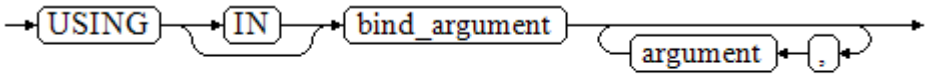
图 10-9 noselect::=



using_clause子句的语法参见**图10-10**。

版权所有 © 华为云计算技术有限公司

图 10-10 using_clause-2



对以上语法格式的解释如下：

USING IN bind_argument用于指定存放传递给动态SQL值的变量，在dynamic_noselect_string中存在占位符时使用，即动态SQL语句执行时，bind_argument将替换相对应的占位符。要注意的是，bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause。另外，动态语句允许出现重复的占位符，相同占位符只能与唯一一个bind_argument按位置一一对应。

示例

```
--创建表
CREATE TABLE sections_t1
(
   section       NUMBER(4) ,
   section_name  VARCHAR2(30),
   manager_id    NUMBER(6),
   place_id      NUMBER(4)
)
DISTRIBUTE BY hash(manager_id);

--声明变量
DECLARE
   section       NUMBER(4) := 280;
   section_name  VARCHAR2(30) := 'Info support';
   manager_id    NUMBER(6) := 103;
   place_id      NUMBER(4) := 1400;
   new_colname   VARCHAR2(10) := 'sec_name';
BEGIN
--执行查询
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :4)'
       USING section, section_name, manager_id,place_id;
--执行查询（重复占位符）
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :1)'
       USING section, section_name, manager_id;
--执行ALTER语句（建议采用“||”拼接数据库对象构造DDL语句）
    EXECUTE IMMEDIATE 'alter table sections_t1 rename section_name to ' || new_colname;
END;
/

--查询数据
SELECT * FROM sections_t1;

--删除表
DROP TABLE sections_t1;
```

10.6.3 动态调用存储过程

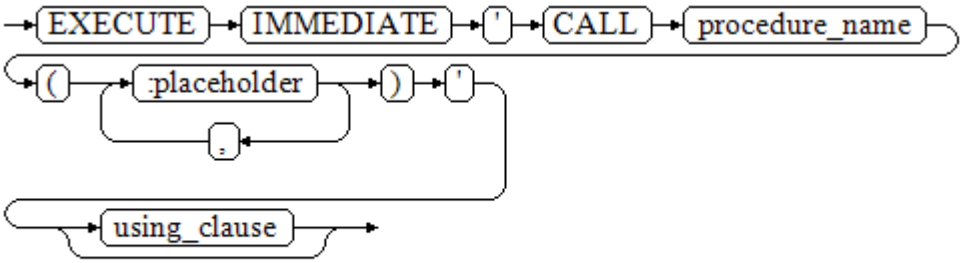
动态调用存储过程必须使用匿名的语句块将存储过程或语句块包在里面，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

语法请参见图10-11。

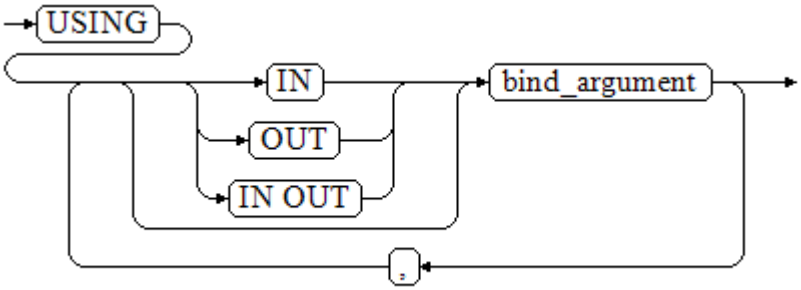
版权所有 © 华为云计算技术有限公司

图 10-11 call_procedure::=



using_clause子句的语法参见图10-12。

图 10-12 using_clause-3



对以上语法格式的解释如下：

- CALL procedure_name，调用存储过程。
- [:placeholder1，:placeholder2，…]，存储过程参数占位符列表，占位符个数与参数个数相同。占位符命名以“:”或“$”开始，“:”后面可跟数字、字符或字符串（不能使用带引号的数字、字符或字符串），“$”后面仅可跟数字。占位符与USING子句的bind_argument一一对应。
- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。

示例

创建存储过程proc_add。

```
CREATE OR REPLACE PROCEDURE proc_add
(
    param1    in   INTEGER,
    param2    out  INTEGER,
    param3    in   INTEGER
)
AS
BEGIN
   param2:= param1 + param3;
END;
/
```

调用存储过程。

```
DECLARE
    input1 INTEGER:=1;
```


版权所有 © 华为云计算技术有限公司

```
    input2 INTEGER:=2;
    statement  VARCHAR2(200);
    param2     INTEGER;
BEGIN
  --声明调用语句
    statement := 'call proc_add(:col_1, :col_2, :col_3)'; ( 或者statement := 'call proc_add($1, $2, $3)'; )
    --执行语句
    EXECUTE IMMEDIATE statement
        USING IN input1, OUT param2, IN input2;
    dbms_output.put_line('result is: '||to_char(param2));
END;
/
```

删除存储过程。

```
DROP PROCEDURE proc_add;
```

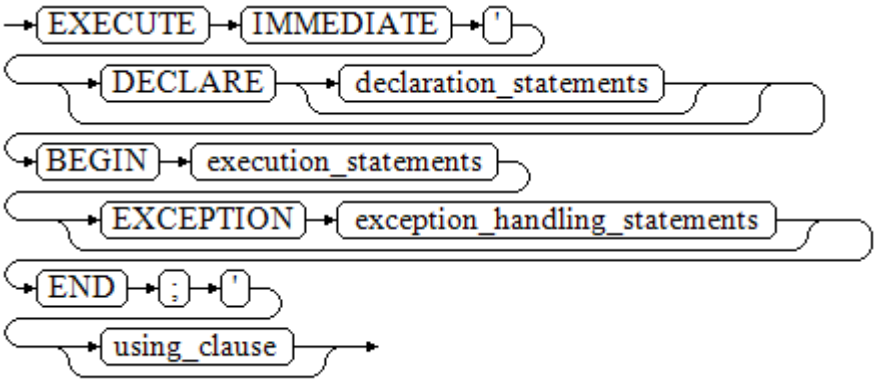
10.6.4 动态调用匿名块

动态调用匿名块是指在动态语句中执行匿名块，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

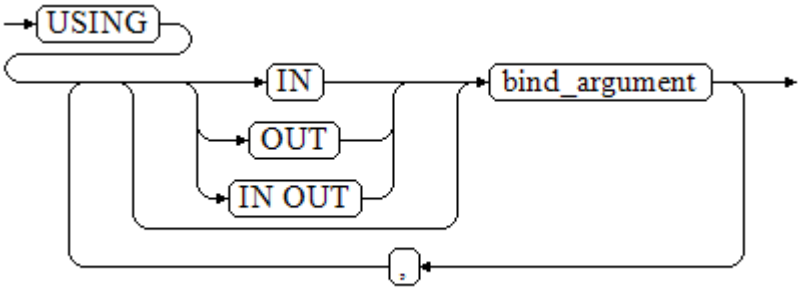
语法请参见图10-13。

图 10-13 call_anonymous_block::=



using_clause子句的语法参见图10-14。

图 10-14 using_clause-4



对以上语法格式的解释如下：

版权所有 © 华为云计算技术有限公司

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。
- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。
- 匿名块中间的输入输出参数使用占位符来指明，要求占位符个数与参数个数相同，并且占位符所对应参数的顺序和USING中参数的顺序一致。
- 目前DWS在动态语句调用匿名块时，EXCEPTION语句中暂不支持使用占位符进行输入输出参数的传递。

示例

```
--创建存储过程dynamic_proc
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
    --执行匿名块
    EXECUTE IMMEDIATE 'begin select first_name, salary into :first_name, :salary from staffs where
staff_id= :dno; end;'
       USING OUT first_name, OUT salary, IN staff_id;
   dbms_output.put_line(first_name|| ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

10.7 DWS 存储过程控制语句

10.7.1 返回语句

DWS提供两种方式返回数据：RETURN（或RETURN NEXT）及RETURN QUERY。其中，RETURN NEXT和RETURN QUERY只适用于函数，不适用存储过程。

RETURN

语法

返回语句的语法请参见图10-15。

图 10-15 return_clause::=

对以上语法的解释如下：

用于将控制从存储过程或函数返回给调用者。

示例

```
--创建存储过程proc_staffs
CREATE OR REPLACE PROCEDURE proc_staffs
```


版权所有 © 华为云计算技术有限公司

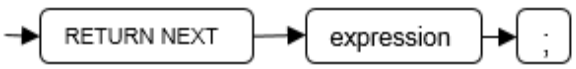
```
(
section     NUMBER(6),
salary_sum out NUMBER(8,2),
staffs_count out INTEGER
)
IS
BEGIN
SELECT sum(salary), count(*) INTO salary_sum, staffs_count FROM staffs where section_id = section;
END;
/

--创建存储过程proc_return.
CREATE OR REPLACE PROCEDURE proc_return
AS
v_num NUMBER(8,2);
v_sum INTEGER;
BEGIN
proc_staffs(30, v_sum, v_num);  --调用语句
dbms_output.put_line(v_sum||'#'||v_num);
RETURN;  --返回语句
END;
/

--调用存储过程proc_return.
CALL proc_return();

--清除存储过程
DROP PROCEDURE proc_staffs;
DROP PROCEDURE proc_return;

--创建函数func_return.
CREATE OR REPLACE FUNCTION func_return returns void
language plpgsql
AS $$
DECLARE
v_num INTEGER := 1;
BEGIN
dbms_output.put_line(v_num);
RETURN;  --返回语句
END $$;


-- 调用函数func_return
CALL func_return();
1

-- 清除函数
DROP FUNCTION func_return;
```

RETURN NEXT 及 RETURN QUERY

语法

创建函数时需要指定返回值SETOF datatype。

return_next_clause::=



return_query_clause::=

对以上语法的解释如下：

版权所有 © 华为云计算技术有限公司

当需要函数返回一个集合时，使用RETURN NEXT或者RETURN QUERY向结果集追加结果，然后继续执行函数的下一条语句。随着后续的RETURN NEXT或RETURN QUERY命令的执行，结果集中会有多个结果。函数执行完成后会一起返回所有结果。

RETURN NEXT可用于标量和复合数据类型。

RETURN QUERY有一种变体RETURN QUERY EXECUTE，后面还可以增加动态查询，通过USING向查询插入参数。

示例

```
CREATE TABLE t1(a int);
INSERT INTO t1 VALUES(1),(10);

--RETURN NEXT
CREATE OR REPLACE FUNCTION fun_for_return_next() RETURNS SETOF t1 AS $$
DECLARE
  r t1%ROWTYPE;
BEGIN
  FOR r IN select * from t1
  LOOP
    RETURN NEXT r;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE PLPGSQL;
call fun_for_return_next();
 a
---
 1
 10
(2 rows)

-- RETURN QUERY
CREATE OR REPLACE FUNCTION fun_for_return_query() RETURNS SETOF t1 AS $$
DECLARE
  r t1%ROWTYPE;
BEGIN
  RETURN QUERY select * from t1;
END;
$$
language plpgsql;
call fun_for_return_query();
 a
---
 1
 10
(2 rows)
```

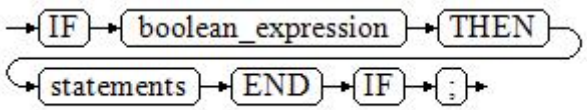
10.7.2 条件语句

条件语句的主要作用是判断参数或者语句是否满足已给定的条件，根据判定结果执行相应的操作。

DWS有五种形式的IF：

- IF_THEN

  图 10-16 IF_THEN::=

版权所有 © 华为云计算技术有限公司

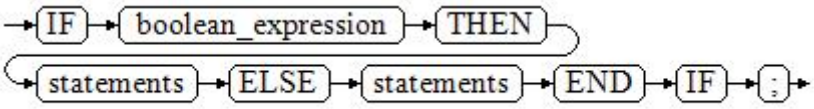
IF_THEN语句是IF的最简单形式。如果条件为真，statements将被执行。否则，将忽略它们的结果使该IF_THEN语句执行结束。

示例

```
IF v_user_id <> 0 THEN
    UPDATE users SET email = v_email WHERE user_id = v_user_id;
END IF;
```

- IF_THEN_ELSE

图 10-17 IF_THEN_ELSE::=



IF_THEN_ELSE语句增加了ELSE的分支，可以声明在条件为假的时候执行的语句。

示例

```
IF parentid IS NULL OR parentid = ''
THEN
    RETURN;
ELSE
    hp_true_filename(parentid);--表示调用存储过程
END IF;
```

- IF_THEN_ELSE IF

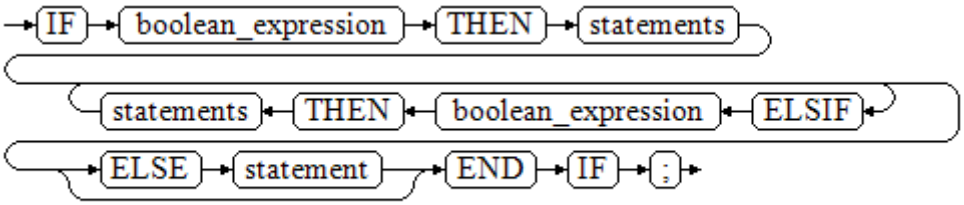
  IF语句可以嵌套，嵌套方式如下：

```
IF gender = 'm' THEN
    pretty_gender := 'man';
ELSE
    IF gender = 'f' THEN
        pretty_gender := 'woman';
    END IF;
END IF;
```

  这种形式实际上就是在一个IF语句的ELSE部分嵌套了另一个IF语句。因此需要给每个嵌套的IF一个END IF语句，另外还需要一个END IF语句结束父IF-ELSE。如果有多个选项，可使用IF_THEN_ELSIF_ELSE形式。

- IF_THEN_ELSIF_ELSE

图 10-18 IF_THEN_ELSIF_ELSE::=



示例

```
IF number_tmp = 0 THEN
    result := 'zero';
ELSIF number_tmp > 0 THEN
    result := 'positive';
ELSIF number_tmp < 0 THEN
    result := 'negative';
```


版权所有 © 华为云计算技术有限公司

```
ELSE
    result := 'NULL';
END IF;
```

- IF_THEN_ELSEIF_ELSE

  ELSEIF是ELSIF的别名。

  综合示例

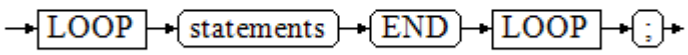
```
CREATE OR REPLACE PROCEDURE proc_control_structure(i in integer)
AS
    BEGIN
        IF i > 0 THEN
            raise info 'i:% is greater than 0. ',i;
        ELSIF i < 0 THEN
            raise info 'i:% is smaller than 0. ',i;
        ELSE
            raise info 'i:% is equal to 0. ',i;
        END IF;
        RETURN;
    END;
/

CALL proc_control_structure(3);

--删除存储过程
DROP PROCEDURE proc_control_structure;
```

10.7.3 循环语句

简单 LOOP 语句

语法图

图 10-19 loop::=



示例

```
CREATE OR REPLACE PROCEDURE proc_loop(i in integer, count out integer)
AS
    BEGIN
        count:=0;
        LOOP
        IF count > i THEN
            raise info 'count is %. ', count;
            EXIT;
        ELSE
            count:=count+1;
        END IF;
        END LOOP;
    END;
/

CALL proc_loop(10,5);
```


版权所有 © 华为云计算技术有限公司

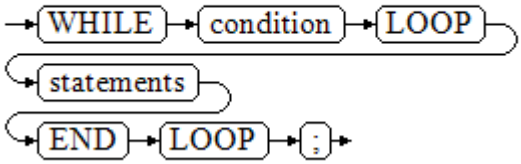
须知

该循环必须要结合EXIT使用，否则将陷入死循环。

WHILE_LOOP 语句

语法图

图 10-20 while_loop::=



只要条件表达式为真，WHILE语句就会不停地在一系列语句上进行循环，在每次进入循环体的时候进行条件判断。

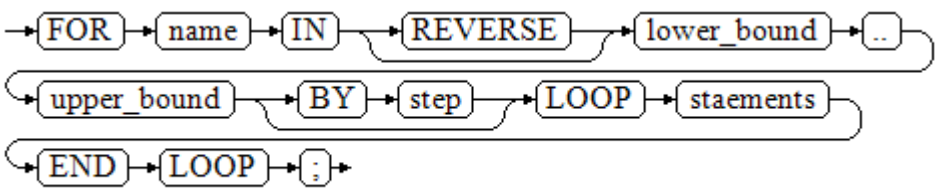
示例

```
CREATE TABLE integertable(c1 integer) DISTRIBUTE BY hash(c1);
CREATE OR REPLACE PROCEDURE proc_while_loop(maxval in integer)
AS
    DECLARE
    i int :=1;
    BEGIN
        WHILE i < maxval LOOP
            INSERT INTO integertable VALUES(i);
            i:=i+1;
        END LOOP;
    END;
/

--调用函数
CALL proc_while_loop(10);

--删除存储过程和表
DROP PROCEDURE proc_while_loop;
DROP TABLE integertable;
```

FOR_LOOP（integer 变量）语句

语法图

图 10-21 for_loop::=



版权所有 © 华为云计算技术有限公司

说明

- 变量name会自动定义为integer类型并且只在此循环里存在。变量name介于lower_bound和upper_bound之间。
- 当使用REVERSE关键字时，lower_bound必须大于等于upper_bound，否则循环体不会被执行。

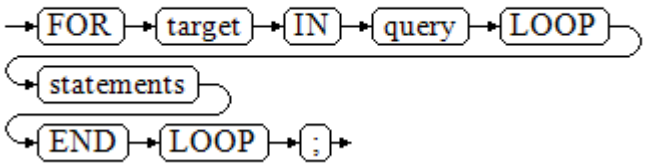
示例

```
--从0到5进行循环
CREATE OR REPLACE PROCEDURE proc_for_loop()
AS
    BEGIN
    FOR I IN 0..5 LOOP
        DBMS_OUTPUT.PUT_LINE('It is '||to_char(I) || ' time;') ;
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop();

--删除存储过程
DROP PROCEDURE proc_for_loop;
```

FOR_LOOP 查询语句

语法图

图 10-22 for_loop_query::=



说明

变量target会自动定义，类型和query的查询结果的类型一致，并且只在此循环中有效。target的取值就是query的查询结果。

示例

```
--循环输出查询结果。
CREATE OR REPLACE PROCEDURE proc_for_loop_query()
AS
    record VARCHAR2(50);
BEGIN
    FOR record IN SELECT spcname FROM pg_tablespace LOOP
    dbms_output.put_line(record);
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop_query();
```

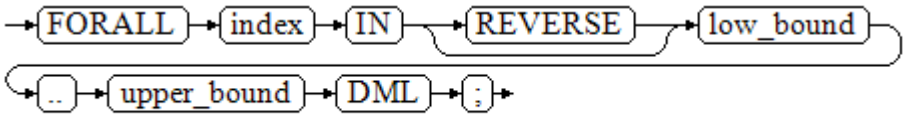

版权所有 © 华为云计算技术有限公司

```
--删除存储过程
DROP PROCEDURE proc_for_loop_query;
```

FORALL 批量查询语句

语法图

图 10-23 forall::=



说明

变量index会自动定义为integer类型并且只在此循环里存在。index的取值介于low_bound和upper_bound之间。

示例

```
CREATE TABLE hdfs_t1 (
  title NUMBER(6),
  did VARCHAR2(20),
  data_period VARCHAR2(25),
  kind VARCHAR2(25),
  interval VARCHAR2(20),
  time DATE,
  isModified VARCHAR2(10)
)
DISTRIBUTE BY hash(did);

INSERT INTO hdfs_t1 VALUES( 8, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', to_date('21-06-1999',
'dd-mm-yyyy'), 'SH_CLERK' );

CREATE OR REPLACE PROCEDURE proc_forall()
AS
BEGIN
    FORALL i IN 100..120
        insert into hdfs_t1(title) values(i);
END;
/

--调用函数
CALL proc_forall();

--查询存储过程调用结果
SELECT * FROM hdfs_t1 WHERE title BETWEEN 100 AND 120;

--删除存储过程和表
DROP PROCEDURE proc_forall;
DROP TABLE hdfs_t1;
```

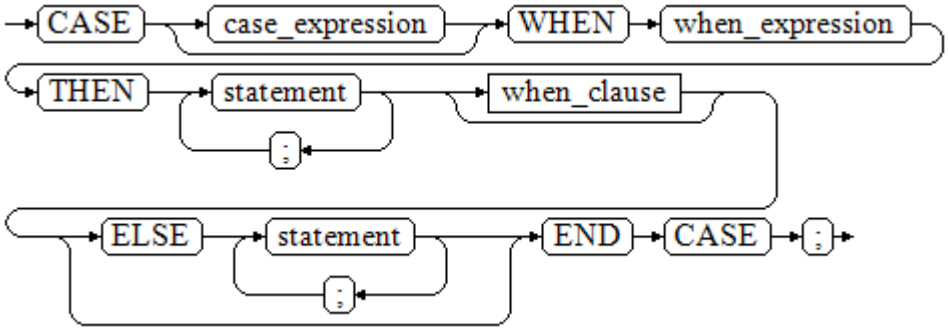
10.7.4 分支语句

语法

分支语句的语法请参见图10-24。

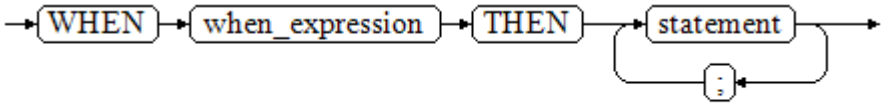
版权所有 © 华为云计算技术有限公司

图 10-24 case_when::=



when_clause子句的语法图参见图10-25。

图 10-25 when_clause::=



参数说明：

- case_expression：变量或表达式。
- when_expression：常量或者条件表达式。
- statement：执行语句。

示例

```
CREATE OR REPLACE PROCEDURE proc_case_branch(pi_result in integer, pi_return out integer)
AS
   BEGIN
      CASE pi_result
         WHEN 1 THEN
            pi_return := 111;
         WHEN 2 THEN
            pi_return := 222;
         WHEN 3 THEN
            pi_return := 333;
         WHEN 6 THEN
            pi_return := 444;
         WHEN 7 THEN
            pi_return := 555;
         WHEN 8 THEN
            pi_return := 666;
         WHEN 9 THEN
            pi_return := 777;
         WHEN 10 THEN
            pi_return := 888;
         ELSE
            pi_return := 999;
      END CASE;
      raise info 'pi_return : %',pi_return ;
END;
/

CALL proc_case_branch(3,0);
```


版权所有 © 华为云计算技术有限公司

```
--删除存储过程
DROP PROCEDURE proc_case_branch;
```

10.7.5 空语句

在PL/SQL程序中，可以用NULL语句来说明“不用做任何事情”，即空语句。

空语句相当于一个占位符，可以使某些语句变得有意义，提高程序的可读性。

语法

空语句的用法如下：

```
DECLARE
   ...
BEGIN
   ...
   IF v_num IS NULL THEN
      NULL; -- 不需要处理任何数据。
   END IF;
END;
/
```

10.7.6 错误捕获语句

缺省时，当PL/SQL函数执行过程中发生错误时退出函数执行，并且周围的事务也会回滚。可以用一个带有EXCEPTION子句的BEGIN块捕获错误并且从中恢复。其语法是正常的BEGIN块语法的一个扩展：

```
[<<label>>]
[DECLARE
   declarations]
BEGIN
   statements
EXCEPTION
   WHEN condition [OR condition ...] THEN
      handler_statements
   [WHEN condition [OR condition ...] THEN
      handler_statements
   ...]
END;
```

如果没有发生错误，这种形式的块只是简单地执行所有语句，然后转到END之后的下一个语句。但是如果在执行的语句内部发生了一个错误，则这个语句将会回滚，然后转到EXCEPTION列表寻找匹配错误的第一个条件。若找到匹配，则执行对应的handler_statements，然后转到END之后的下一个语句。如果没有找到匹配，则会向事务的外层报告错误，和没有EXCEPTION子句一样。

也就是说该错误可以被一个包围块用EXCEPTION捕获，如果没有包围块，则进行退出函数处理。

condition的名字遵循SQL标准错误码编号说明的任意值。特殊的条件名OTHERS匹配除了QUERY_CANCELED之外的所有错误类型。

如果在选中的handler_statements里发生了新错误，则不能被这个EXCEPTION子句捕获，而是向事务的外层报告错误。一个外层的EXCEPTION子句可以捕获它。

如果一个错误被EXCEPTION捕获，PL/SQL函数的局部变量保持错误发生时的原值，但是所有该块中想写入数据库中的状态都回滚。

示例：

```
CREATE TABLE mytab(id INT,firstname VARCHAR(20),lastname VARCHAR(20)) DISTRIBUTE BY hash(id);
```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO mytab(firstname, lastname) VALUES('Tom', 'Jones');

CREATE FUNCTION fun_exp() RETURNS INT
AS $$
DECLARE
    x INT :=0;
    y INT;
BEGIN
    UPDATE mytab SET firstname = 'Joe' WHERE lastname = 'Jones';
    x := x + 1;
    y := x / 0;
EXCEPTION
    WHEN division_by_zero THEN
        RAISE NOTICE 'caught division_by_zero';
        RETURN x;
END;$$
LANGUAGE plpgsql;

CALL fun_exp();
NOTICE:  caught division_by_zero
 fun_exp
---------
       1
(1 row)

SELECT * FROM mytab;
 id | firstname | lastname
----+-----------+----------
    | Tom       | Jones
(1 row)

DROP FUNCTION fun_exp();
DROP TABLE mytab;
```

当执行到给y赋值处时，将触发division_by_zero错误。该错误被EXCEPTION子句捕获，最终RETURN语句返回x的增量值。

说明

- 进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大得多。因此，非必要不建议使用EXCEPTION。
- 以下示例场景中，无法捕获处理异常，整个存储过程回滚：
  1. 节点故障、网络故障引起的存储过程参与节点线程退出。
  2. COPY FROM操作中源数据与目标表的表结构不一致造成的异常。

示例：UPDATE/INSERT异常。

根据使用异常处理器执行恰当的UPDATE或INSERT 。

```
CREATE TABLE db (a INT, b TEXT);

CREATE FUNCTION merge_db(key INT, data TEXT) RETURNS VOID AS
$$
BEGIN
    LOOP

--第一次尝试更新key
        UPDATE db SET b = data WHERE a = key;
        IF found THEN
            RETURN;
        END IF;
--不存在，所以尝试插入key，如果其他人同时插入相同的key，可能得到唯一key失败。
        BEGIN
            INSERT INTO db(a,b) VALUES (key, data);
            RETURN;
        EXCEPTION WHEN unique_violation THEN
        --什么也不做，并且循环尝试再次更新。
```


版权所有 © 华为云计算技术有限公司

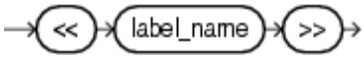
```
        END;
    END LOOP;
END;
$$
LANGUAGE plpgsql;

SELECT merge_db(1, 'david');
SELECT merge_db(1, 'dennis');

--删除FUNCTION和TABLE
DROP FUNCTION merge_db;
DROP TABLE db ;
```

10.7.7 GOTO 语句

GOTO语句可以实现从GOTO位置到目标语句的无条件跳转。GOTO语句会改变原本的执行逻辑，因此应该慎重使用，或者也可以使用EXCEPTION处理特殊场景。当执行GOTO语句时，目标Label必须是唯一的。

语法

label declaration ::=



goto statement ::=

示例

```
CREATE OR REPLACE PROCEDURE GOTO_test()
AS
DECLARE
   v1  int;
BEGIN
   v1  := 0;
      LOOP
      EXIT WHEN v1 > 100;
            v1 := v1 + 2;
            if v1 > 25 THEN
                  GOTO pos1;
            END IF;
      END LOOP;
<<pos1>>
v1 := v1 + 10;
raise info 'v1 is %. ', v1;
END;
/

call GOTO_test();
DROP PROCEDURE GOTO_test();
```

限制场景

GOTO使用有以下限制场景

- 不支持有多个相同的GOTO labels目标场景，无论是否在同一个block中。

  ```
  BEGIN
    GOTO pos1;
    <<pos1>>
    SELECT * FROM ...
  ```


版权所有 © 华为云计算技术有限公司

```
 <<pos1>>
 UPDATE t1 SET ...
END;
```

- 不支持GOTO跳转到IF语句、CASE语句、LOOP语句中。

```
BEGIN
  GOTO pos1;
  IF valid THEN
    <<pos1>>
    SELECT * FROM ...
  END IF;
END;
```

- 不支持GOTO语句从一个IF子句跳转到另一个IF子句，或从一个CASE语句的WHEN子句跳转到另一个WHEN子句。

```
BEGIN
  IF valid THEN
    GOTO pos1;
    SELECT * FROM ...
  ELSE
    <<pos1>>
    UPDATE t1 SET ...
  END IF;
END;
```

- 不支持从外部块跳转到内部的BEGIN-END块。

```
BEGIN
  GOTO pos1;
  BEGIN
    <<pos1>>
    UPDATE t1 SET ...
  END;
END;
```

- 不支持从异常处理部分跳转到当前的BEGIN-END块，但可以跳转到上层BEGIN-END块。

```
BEGIN
  <<pos1>>
  UPDATE t1 SET ...
  EXCEPTION
    WHEN condition THEN
      GOTO pos1;
END;
```

- 如果从GOTO到一个不包含执行语句的位置，需要添加NULL语句。

```
DECLARE
  done  BOOLEAN;
BEGIN
  FOR i IN 1..50 LOOP
    IF done THEN
      GOTO end_loop;
    END IF;
    <<end_loop>>  -- not allowed unless an executable statement follows
    NULL; -- add NULL statement to avoid error
  END LOOP;  -- raises an error without the previous NULL
END;
/
```

10.7.8 事务控制语句

存储过程本身就处于一个事务中，开始调用最外围存储过程时会自动开启一个事务，在调用结束时自动提交或者发生异常时回滚。除了系统自动的事务控制外，也可以使用COMMIT/ROLLBACK来控制存储过程中的事务。在存储过程中调用COMMIT/ROLLBACK命令，将提交/回滚当前事务并自动开启一个新的事务，后续的所有操作都会在此新事务中运行。

本功能仅9.1.1.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

使用场景

- 存储过程支持包含DML语句的COMMIT/ROLLBACK。
- 存储过程支持包含DDL语句的COMMIT/ROLLBACK。
- 嵌套存储过程的调用支持使用COMMIT/ROLLBACK。
- 包含EXCEPTION的存储过程中支持使用COMMIT/ROLLBACK。
- 匿名块内支持COMMIT/ROLLBACK。
- 事务块中支持存储过程带COMMIT/ROLLBACK。

约束限制

- 函数（function）中不允许调用COMMIT/ROLLBACK语句。
- 函数（function）中不允许调用带有COMMIT/ROLLBACK语句的存储过程。
- 写cursor loop过程中不允许COMMIT/ROLLBACK。
- 不支持存储过程中设置参数后COMMIT/ROLLBACK。
- cursor loop中不支持调用COMMIT/ROLLBACK语句：提交回滚时游标 (cursor) 被关闭。
- 不支持出现在SQL中的调用（SELECT PROCEDURE除外）。
- 不支持出现在显式子事务savepoint后的存储过程中进行COMMIT、ROLLBACK。

示例

- 示例1：存储过程支持包含DML语句的COMMIT/ROLLBACK。

```
CREATE TABLE IF NOT EXISTS test1(a int, b int);

CREATE OR REPLACE PROCEDURE transaction_test1()
AS
BEGIN
    FOR i IN 0..9 LOOP
        INSERT INTO test1 (a) VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
END;
/

CALL transaction_test1();
```

- 示例2：存储过程支持包含DDL语句的COMMIT/ROLLBACK。

```
DROP TABLE IF EXISTS test2;

CREATE OR REPLACE PROCEDURE transaction_test2()
AS
BEGIN
    CREATE TABLE test2(a int);
    COMMIT;
DROP TABLE test2;
END;
/

CALL transaction_test2();
```

- 示例3：嵌套存储过程的调用（带有COMMIT/ROLLBACK）。

```
CREATE OR REPLACE PROCEDURE transaction_test3()
AS
BEGIN
```


版权所有 © 华为云计算技术有限公司

```
    PERFORM transaction_test1();
    ROLLBACK;
PERFORM transaction_test2();
ROLLBACK;
END;
/

CALL transaction_test3();
```

- 示例4：包含EXCEPTION的存储过程（COMMIT/ROLLBACK）。

```
CREATE OR REPLACE PROCEDURE TEST_COMMIT_EXCEPTION()
AS
BEGIN
    INSERT INTO test1 (a) VALUES (1);
    INSERT INTO test1 (a) VALUES (2);
    COMMIT;
    INSERT INTO test1 (a) VALUES (3);
    RAISE EXCEPTION 'RAISE EXCEPTION-A';
EXCEPTION
    WHEN OTHERS THEN
    INSERT INTO test1 (a) VALUES (4);
END;
/

CALL TEST_COMMIT_EXCEPTION();
```

- 示例5：匿名块内支持COMMIT/ROLLBACK。

```
DO LANGUAGE plpgsql $$
BEGIN
    BEGIN
        INSERT INTO test1 (a) VALUES (1);
        COMMIT;
        INSERT INTO test1 (a) VALUES (1/0);
        COMMIT;
    EXCEPTION
        WHEN division_by_zero THEN
            RAISE NOTICE 'caught division_by_zero';
    END;
END;
$$;
```

- 示例6：事务块中支持存储过程带COMMIT/ROLLBACK。

```
CREATE TABLE IF NOT EXISTS test1(a int, b int);

CREATE OR REPLACE PROCEDURE transaction_test1()
AS
BEGIN
    FOR i IN 0..9 LOOP
        INSERT INTO test1 (a) VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
END;
/

START TRANSACTION;
CALL transaction_test1();
COMMIT;
```

- 示例7：函数（function）中不允许调用COMMIT/ROLLBACK语句。

```
CREATE FUNCTION transaction_test2() RETURNS int
LANGUAGE plpgsql
AS $$
BEGIN
    FOR i IN 0..9 LOOP
        INSERT INTO test1 (a) VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
```

 版权所有 © 华为云计算技术有限公司

```
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
    RETURN 1;
END
$$;

SELECT transaction_test2();
```

- 示例8：函数（function）中不允许调用带有COMMIT/ROLLBACK语句的存储过程。

```
CREATE OR REPLACE PROCEDURE transaction_test1()
AS
BEGIN
    INSERT INTO test1 (a) VALUES (1);
    ROLLBACK;
    INSERT INTO test1 (a) VALUES (2);
END;
/
CREATE FUNCTION transaction_test3() RETURNS int
LANGUAGE plpgsql
AS $$
BEGIN
    CALL transaction_test1();
    RETURN 1;
END;
$$;

SELECT transaction_test3();
```

- 示例9：写cursor loop过程中不允许COMMIT/ROLLBACK。

```
DO LANGUAGE plpgsql $$
DECLARE
    r RECORD;
BEGIN
    FOR r IN UPDATE test2 SET x = x * 2 RETURNING x LOOP
        INSERT INTO test1 (a) VALUES (r.x);
        ROLLBACK;
    END LOOP;
END;
$$;
```

- 示例10：不支持存储过程中设置参数后COMMIT/ROLLBACK。

```
CREATE OR REPLACE PROCEDURE transaction_test5()
SET work_mem = 555
AS
BEGIN
    insert into test1 select setting from pg_settings where name like 'work_mem';
    COMMIT;
END;
/

CALL transaction_test5();
```

- 示例11：cursor loop中不支持调用COMMIT/ROLLBACK语句，提交回滚时游标(cursor) 被关闭。

```
DROP TABLE IF EXISTS cursor_back_tab2;
CREATE TABLE cursor_back_tab2(i_int bigint,i_char varchar ,i_numeric numeric(18,3),i_data date);
CREATE OR REPLACE PROCEDURE cursor_back_pro_2()
AS
declare
o_int bigint;
o_char varchar;
o_numeric numeric;
o_data date;
cursor cursor1 for select * from cursor_back_tab1 order by 1,2,3,4;
begin
    open cursor1;
    fetch  next from cursor1 into o_int,o_char,o_numeric,o_data;
```

 版权所有 © 华为云计算技术有限公司

```
    LOOP
        fetch  next from cursor1 into o_int,o_char,o_numeric,o_data;
exit when cursor1%NOTFOUND;
insert into cursor_back_tab2 values(o_int,o_char,o_numeric,o_data);
    END LOOP;
COMMIT;
ROLLBACK;
CLOSE cursor1;
END;
/

CALL cursor_back_pro_2();
```

- 示例12：不支持出现在SQL中的调用（SELECT PROCEDURE除外）。

```
CREATE TABLE IF NOT EXISTS test1(a int, b int);

CREATE OR REPLACE PROCEDURE transaction_test1()
AS
BEGIN
    FOR i IN 0..9 LOOP
        INSERT INTO test1 (a) VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
END;
/

SELECT transaction_test1() from test1;
```

- 示例13：不支持出现在显式子事务savepoint后的存储过程中进行COMMIT、ROLLBACK。

```
CREATE OR REPLACE PROCEDURE sp_txn_test1_1()
AS
BEGIN
INSERT INTO test1 (a) VALUES (1);
COMMIT;
INSERT INTO test1 (a) VALUES (2);
END;
/

BEGIN;
SAVEPOINT s1;
CALL sp_txn_test1_1();
ROLLBACK;
```

10.8 DWS 存储过程其他语句

锁操作

DWS提供了多种锁模式用于控制对表中数据的并发访问。这些模式可以用在MVCC（多版本并发控制）无法给出期望行为的场合。同样，大多数DWS命令自动施加恰当的锁，以保证被引用的表在命令的执行过程中不会以一种不兼容的方式被删除或者修改。比如，在存在其他并发操作的时候，ALTER TABLE是不能在同一个表上执行的。

游标操作

DWS中游标（cursor）是系统为用户开设的一个数据缓冲区，存放着SQL语句的执行结果。每个游标区都有一个名字。用户可以用SQL语句逐一从游标中获取记录，并赋予主变量，交由主语言进一步处理。

游标的操作主要有游标的定义、打开、获取和关闭。

版权所有 © 华为云计算技术有限公司

完整的游标操作示例可参考**显式游标**。

10.9 DWS 存储过程游标

10.9.1 游标概述

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

须知

当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。

游标的使用分为显式游标和隐式游标。对于不同的SQL语句，游标的使用情况不同，详细信息请参见**表10-2**。

表 10-2 游标使用情况

SQL语句	游标
非查询语句	隐式的
结果是单行的查询语句	隐式的或显式的
结果是多行的查询语句	显式的

10.9.2 显式游标

显式游标主要用于对查询语句的处理，尤其是查询结果为多条记录的情况。

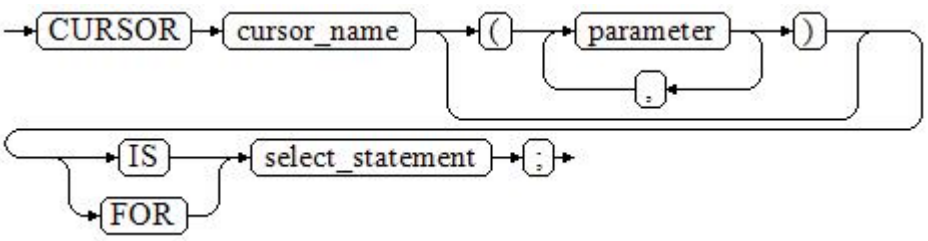
处理步骤

显式游标处理需六个PL/SQL步骤：

步骤1 **定义静态游标：**就是定义一个游标名，以及与其相对应的SELECT语句。

定义静态游标的语法图，请参见**图10-26**。

图 10-26 static_cursor_define::=



版权所有 © 华为云计算技术有限公司

参数说明：

- cursor_name：定义的游标名。
- parameter：游标参数，只能为输入参数，其格式为：
  ```
  parameter_name datatype
  ```
- select_statement：查询语句。

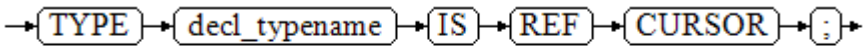
说明

根据执行计划的不同，系统会自动判断该游标是否可以用于以倒序的方式检索数据行。

定义动态游标：指ref游标，可以通过一组静态的SQL语句动态的打开游标。首先定义ref游标类型，然后定义该游标类型的游标变量，在打开游标时通过OPEN FOR动态绑定SELECT语句。

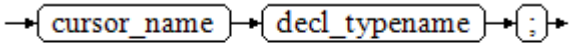
定义动态游标的语法图，请参见图10-27和图10-28。

图 10-27 cursor_typename::=



DWS支持sys_refcursor动态游标类型，函数或存储过程可以通过sys_refcursor参数传入或传出游标结果集合，函数也可以通过返回sys_refcursor来返回游标结果集合。

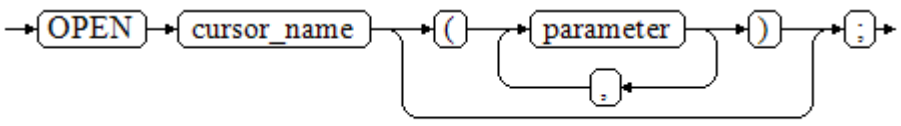
图 10-28 dynamic_cursor_define::=



步骤2 **打开静态游标：**就是执行游标所对应的SELECT语句，将其查询结果放入工作区，并且指针指向工作区的首部，标识游标结果集合。如果游标查询语句中带有FOR UPDATE选项，OPEN语句还将锁定数据库表中游标结果集合对应的数据行。

打开静态游标的语法图，请参见图10-29。

图 10-29 open_static_cursor::=

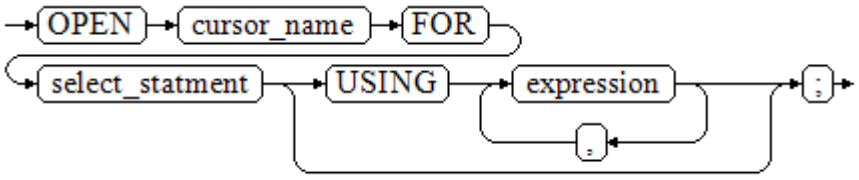


打开动态游标：可以通过OPEN FOR语句打开动态游标，动态绑定SQL语句。

打开动态游标的语法图，请参见图10-30。

版权所有 © 华为云计算技术有限公司

图 10-30 open_dynamic_cursor::=

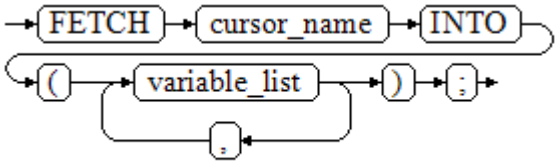


PL/SQL程序不能用OPEN语句重复打开一个游标。

步骤3 提取游标数据：检索结果集合中的数据行，放入指定的输出变量中。

提取游标数据的语法图，请参见**图10-31**。

图 10-31 fetch_cursor::=



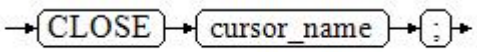
步骤4 对该记录进行处理。

步骤5 继续处理，直到活动集合中没有记录。

步骤6 关闭游标：当提取和处理完游标结果集合数据后，应及时关闭游标，以释放该游标所占用的系统资源，并使该游标的工作区变成无效，不能再使用FETCH语句获取其中数据。关闭后的游标可以使用OPEN语句重新打开。

关闭游标的语法图，请参见**图10-32**。

图 10-32 close_cursor::=



----结束

属性

游标的属性用于控制程序流程或者了解程序的状态。当运行DML语句时，PL/SQL打开一个内建游标并处理结果，游标是维护查询结果的内存中的一个区域，游标在运行DML语句时打开，完成后关闭。显式游标的属性为：

- %FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- %NOTFOUND布尔型属性：与%FOUND相反。
- %ISOPEN布尔型属性：当游标已打开时返回TRUE。
- %ROWCOUNT数值型属性：返回已从游标中读取的记录数。

示例

```
--游标参数的传递方法。
CREATE OR REPLACE PROCEDURE cursor_proc1()
```


版权所有 © 华为云计算技术有限公司

```
AS
DECLARE
    DEPT_NAME VARCHAR(100);
    DEPT_LOC NUMBER(4);
    --定义游标
    CURSOR C1 IS
        SELECT section_name, place_id FROM sections WHERE section_id <= 50;
    CURSOR C2(sect_id INTEGER) IS
        SELECT section_name, place_id FROM sections WHERE section_id <= sect_id;
    TYPE CURSOR_TYPE IS REF CURSOR;
    C3 CURSOR_TYPE;
    SQL_STR VARCHAR(100);
BEGIN
    OPEN C1;--打开游标
    LOOP
        --通过游标取值
        FETCH C1 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C1;--关闭游标

    OPEN C2(10);
    LOOP
        FETCH C2 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C2%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C2;

    SQL_STR := 'SELECT section_name, place_id FROM sections WHERE section_id <= :DEPT_NO;';
    OPEN C3 FOR SQL_STR USING 50;
    LOOP
        FETCH C3 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C3%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C3;
END;
/

CALL cursor_proc1();

DROP PROCEDURE cursor_proc1;
--给工资低于3000的员工增加工资500。
CREATE TABLE staffs_t1 AS TABLE staffs;

CREATE OR REPLACE PROCEDURE cursor_proc2()
AS
DECLARE
  V_EMPNO  NUMBER(6);
  V_SAL    NUMBER(8,2);
  CURSOR C IS SELECT staff_id, salary FROM staffs_t1;
BEGIN
  OPEN C;
  LOOP
    FETCH C INTO V_EMPNO, V_SAL;
    EXIT WHEN C%NOTFOUND;
    IF V_SAL<=3000 THEN
         UPDATE staffs_t1 SET salary =salary + 500 WHERE staff_id = V_EMPNO;
    END IF;
  END LOOP;
  CLOSE C;
END;
/

CALL cursor_proc2();

--删除存储过程
```


版权所有 © 华为云计算技术有限公司

```
DROP PROCEDURE cursor_proc2;
DROP TABLE staffs_t1;
--SYS_REFCURSOR类型作为函数参数
CREATE OR REPLACE PROCEDURE proc_sys_ref(O OUT SYS_REFCURSOR)
IS
C1 SYS_REFCURSOR;
BEGIN
OPEN C1 FOR SELECT section_ID FROM sections ORDER BY section_ID;
O := C1;
END;
/

DECLARE
C1 SYS_REFCURSOR;
TEMP NUMBER(4);
BEGIN
proc_sys_ref(C1);
LOOP
  FETCH C1 INTO TEMP;
  DBMS_OUTPUT.PUT_LINE(C1%ROWCOUNT);
  EXIT WHEN C1%NOTFOUND;
END LOOP;
END;
/

--删除存储过程
DROP PROCEDURE proc_sys_ref;
```

10.9.3 隐式游标

对于非查询语句，如修改、删除操作，则由系统自动地为这些操作设置游标并创建其工作区，这些由系统隐含创建的游标称为隐式游标，隐式游标的名字为SQL，这是由系统定义的。

简介

对于隐式游标的操作，如定义、打开、取值及关闭操作，都由系统自动地完成，无需用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。在隐式游标的工作区中，所存放的数据是最新处理的一条SQL语句所包含的数据，与用户自定义的显式游标无关。

格式调用为： SQL%

说明

INSERT，UPDATE，DROP，SELECT语句中不必明确定义游标。

属性

隐式游标属性为：

- SQL%FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- SQL%NOTFOUND布尔型属性：与%FOUND相反。
- SQL%ROWCOUNT数值型属性：返回已从游标中读取的记录数。
- SQL%ISOPEN布尔型属性：取值总是FALSE。SQL语句执行完毕立即关闭隐式游标。

示例

```
--删除EMP表中某部门的所有员工，如果该部门中已没有员工，则在DEPT表中删除该部门。
CREATE TABLE staffs_t1 AS TABLE staffs;
```


版权所有 © 华为云计算技术有限公司

```
CREATE TABLE sections_t1 AS TABLE sections;

CREATE OR REPLACE PROCEDURE proc_cursor3()
AS
   DECLARE
   V_DEPTNO NUMBER(4) := 100;
   BEGIN
      DELETE FROM staffs WHERE section_ID = V_DEPTNO;
      --根据游标状态做进一步处理
      IF SQL%NOTFOUND THEN
      DELETE FROM sections_t1 WHERE section_ID = V_DEPTNO;
      END IF;
   END;
/

CALL proc_cursor3();

--删除存储过程和临时表
DROP PROCEDURE proc_cursor3;
DROP TABLE staffs_t1;
DROP TABLE sections_t1;
```

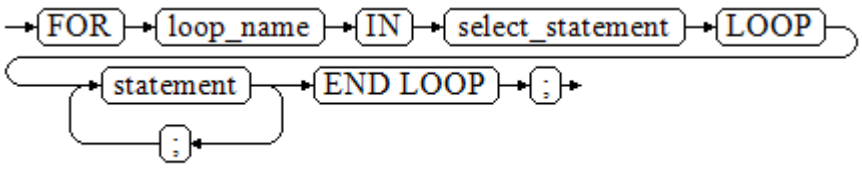
10.9.4 游标循环

游标在WHILE语句、LOOP语句中的使用称为游标循环，一般这种循环都需要使用OPEN、FETCH和CLOSE语句。下面要介绍的一种循环不需要这些操作，可以简化游标循环的操作，这种循环方式适用于静态游标的循环，不用执行静态游标的四个步骤。

语法

FOR AS循环的语法请参见图10-33。

图 10-33 FOR_AS_loop::=



注意事项

- 不能在该循环语句中对查询的表进行更新操作。
- 变量loop_name会自动定义且只在此循环中有效，类型和select_statement的查询结果类型一致。loop_name的取值就是select_statement的查询结果。
- 游标的属性中%FOUND、%NOTFOUND、%ROWCOUNT在DWS数据库中都是访问同一个内部变量，事务和匿名块不支持多个游标同时访问。

示例

```
BEGIN
FOR ROW_TRANS IN
      SELECT first_name FROM staffs
   LOOP
      DBMS_OUTPUT.PUT_LINE (ROW_TRANS.first_name );
   END LOOP;
END;
/
```


版权所有 © 华为云计算技术有限公司

```
--创建表
CREATE TABLE integerTable1( A INTEGER) DISTRIBUTE BY hash(A);
CREATE TABLE integerTable2( B INTEGER) DISTRIBUTE BY hash(B);
INSERT INTO integerTable2 VALUES(2);

--多游标共享游标属性的标量
DECLARE
  CURSOR C1 IS SELECT A FROM integerTable1;--声明游标
  CURSOR C2 IS SELECT B FROM integerTable2;
  PI_A INTEGER;
  PI_B INTEGER;
BEGIN
  OPEN C1;--打开游标
  OPEN C2;
  FETCH C1 INTO PI_A; ----  C1%FOUND 和 C2%FOUND 值为 FALSE
  FETCH C2 INTO PI_B; ----  C1%FOUND 和 C2%FOUND 的值都为 TRUE
  --判断游标状态
  IF C1%FOUND THEN
     IF C2%FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Dual cursor share parameter.');
    END IF;
  END IF;
  CLOSE C1;--关闭游标
  CLOSE C2;
END;
/

--删除临时表
DROP TABLE integerTable1;
DROP TABLE integerTable2;
```

10.10 DWS 存储过程高级包

10.10.1 DBMS_LOB

接口介绍

高级功能包DBMS_LOB支持的所有接口请参见**表10-3**。

表 10-3 DBMS_LOB

接口名称	描述
DBMS_LOB.GETLENGTH	获取并返回指定的LOB类型对象的长度。
DBMS_LOB.OPEN	打开一个LOB返回一个LOB的描述符。
DBMS_LOB.READ	根据指定的长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。
DBMS_LOB.WRITE	根据指定长度及起始位置偏移将BUFFER中内容写入到LOB中。
DBMS_LOB.WRITEAPPEND	根据指定长度将BUFFER中内容写入到LOB的尾部。
DBMS_LOB.COPY	根据指定长度及起始位置偏移将BLOB内容写入到另一个BLOB中。
DBMS_LOB.ERASE	根据指定长度及起始位置偏移删除BLOB中的内容。

版权所有 © 华为云计算技术有限公司

接口名称	描述
DBMS_LOB.CLOSE	关闭已经打开的LOB描述符。
DBMS_LOB.INSTR	返回一个字符串在LOB中第N次出现的位置。
DBMS_LOB.COMPARE	比较两个LOB或者两个LOB的某一部分。
DBMS_LOB.SUBSTR	用于读取一个LOB的子串，返回读取的字节个数或者字符个数。
DBMS_LOB.TRIM	用于截断指定长度的LOB，执行完会将LOB的长度设置为newlen参数指定的长度。
DBMS_LOB.CREATETEMPORARY	创建一个临时的BLOB或者CLOB。
DBMS_LOB.APPEND	将源LOB的内容拼接到目标LOB中。

- DBMS_LOB.GETLENGTH

  存储过程GETLENGTH获取并返回指定的LOB类型对象的长度。

  DBMS_LOB.GETLENGTH函数原型为：

  ```
  DBMS_LOB.GETLENGTH (
  lob_loc    IN   BLOB)
  RETURN INTEGER;

  DBMS_LOB.GETLENGTH (
  lob_loc    IN   CLOB)
  RETURN INTEGER;
  ```

  表 10-4 DBMS_LOB.GETLENGTH 接口参数说明

参数	描述
lob_loc	待获取长度的指定的LOB类型的对象。

- DBMS_LOB.OPEN

  存储过程打开一个LOB，并返回一个LOB描述符，该过程无实际意义，仅用于兼容。

  DBMS_LOB.OPEN函数原型为：

  ```
  DBMS_LOB.LOB (
  lob_loc    INOUT   BLOB,
  open_mode IN BINARY_INTEGER);

  DBMS_LOB.LOB (
  lob_loc  INOUT    CLOB,
  open_mode IN BINARY_INTEGER);
  ```

  表 10-5 DBMS_LOB.OPEN 接口参数说明

参数	描述
lob_loc	被打开的一个BLOB或者CLOB描述符。

版权所有 © 华为云计算技术有限公司

参数	描述
open_mode IN BINARY_INTEGER	打开的模式，目前支持DBMS_LOB.LOB_READWRITE

- DBMS_LOB.READ

  存储过程READ根据指定长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。

  DBMS_LOB.READ函数原型为：

```
DBMS_LOB.READ (
lob_loc     IN          BLOB,
amount      IN          INTEGER,
offset      IN          INTEGER,
buffer      OUT         RAW);

DBMS_LOB.READ (
lob_loc     IN          CLOB,
amount      IN OUT      INTEGER,
offset      IN          INTEGER,
buffer      OUT         VARCHAR2);
```

表 10-6 DBMS_LOB.READ 接口参数说明

参数	说明
lob_loc	待读入的指定的LOB类型的对象。
amount	读入长度。 **说明** 如果读入长度为负，会收到错误提示“ERROR: argument 2 is null, invalid, or out of range.”
offset	指定从LOB内容的哪个位置开始读取的偏移（即相对LOB内容起始位置的字节数）。
buffer	读取LOB内容后存放的目标缓冲区。

- DBMS_LOB.WRITE

  存储过程WRITE根据指定长度及起始位置偏移将BUFFER中内容写入到LOB变量中。

  DBMS_LOB.WRITE函数原型为：

```
DBMS_LOB.WRITE (
lob_loc     IN OUT      BLOB,
amount      IN          INTEGER,
offset      IN          INTEGER,
buffer      IN          RAW);

DBMS_LOB.WRITE (
lob_loc    IN OUT       CLOB,
amount     IN           INTEGER,
offset     IN           INTEGER,
buffer     IN           VARCHAR2);
```


版权所有 © 华为云计算技术有限公司

表 10-7 DBMS_LOB.WRITE 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。
offset	指定从LOB内容的哪个位置开始写入的偏移（即相对LOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于LOB类型最大长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.WRITEAPPEND

  存储过程WRITEAPPEND根据指定长度将BUFFER中内容写入到LOB的尾部。

  DBMS_LOB.WRITEAPPEND函数原型为：

```
DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     BLOB,
amount      IN         INTEGER,
buffer      IN         RAW);

DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     CLOB,
amount      IN         INTEGER,
buffer      IN         VARCHAR2);
```

表 10-8 DBMS_LOB.WRITEAPPEND 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.COPY

  存储过程COPY根据指定长度及起始位置偏移将BLOB内容复制到另一个BLOB中。

  DBMS_LOB.COPY函数原型为：

```
DBMS_LOB.COPY (
dest_lob      IN OUT     BLOB,
src_lob       IN         BLOB,
amount        IN         INTEGER,
dest_offset   IN         INTEGER  DEFAULT 1,
src_offset    IN         INTEGER  DEFAULT 1);
```


版权所有 © 华为云计算技术有限公司

表 10-9 DBMS_LOB.COPY 接口参数说明

参数	说明
dest_lob	待拷入的BLOB类型对象。
src_lob	待拷出的BLOB类型对象。
amount	复制长度。 **说明** 如果拷入长度小于1或拷入长度大于BLOB类型最大长度，则报错。
dest_offset	指定从BLOB内容的哪个位置开始拷入的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。
src_offset	指定从BLOB内容的哪个位置开始拷出的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于复制来源BLOB的长度，则报错。

- DBMS_LOB.ERASE

  存储过程ERASE根据指定长度及起始位置偏移删除BLOB中的内容。

  DBMS_LOB.ERASE函数原型为：

```
DBMS_LOB.ERASE (
lob_loc          IN OUT   BLOB,
amount           IN OUT   INTEGER,
offset           IN       INTEGER DEFAULT 1);
```

表 10-10 DBMS_LOB.ERASE 接口参数说明

参数	说明
lob_loc	待删除内容的BLOB类型对象。
amount	待删除的长度。 **说明** 如果删除长度小于1或删除长度大于BLOB类型最大长度，则报错。
offset	指定从BLOB内容的哪个位置开始删除的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。

- DBMS_LOB.CLOSE

  存储过程CLOSE用于关闭已经打开的LOB内容。

  DBMS_LOB.CLOSE函数原型为：

```
DBMS_LOB.CLOSE(
src_lob      IN             BLOB);

DBMS_LOB.CLOSE (
src_lob      IN             CLOB);
```


版权所有 © 华为云计算技术有限公司

表 10-11 DBMS_LOB.CLOSE 接口参数说明

参数	说明
src_lob	待关闭的LOB类型对象。

- DBMS_LOB.INSTR

  该函数返回在LOB中第N次出现的位置，如果输入的是一些无效值会返回NULL值。offset < 1 or offset > LOBMAXSIZE，nth < 1，nth > LOBMAXSIZE。

  DBMS_LOB.INSTR函数原型为：

```
DBMS_LOB.INSTR (
lob_loc     IN     BLOB,
pattern     IN     RAW,
offset      IN     INTEGER := 1,
nth         IN     INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.INSTR (
lob_loc    IN     CLOB,
pattern    IN     VARCHAR2 ,
offset     IN     INTEGER := 1,
nth        IN     INTEGER := 1)
RETURN INTEGER;
```

表 10-12 DBMS_LOB.INSTR 接口参数说明

参数	说明
lob_loc	要查找的LOB描述符。
pattern	要匹配的模式，对于BLOB是由一组RAW类型的数据组成，对于CLOB是由一组text类型的数据组成。
offset	对于BLOB是以字节为单位的绝对偏移量，对于CLOB是以字符为单位的偏移量，模式匹配的起始位置是1。
nth	模式匹配的次数，最小值为1。

- DBMS_LOB.COMPARE

  这个函数比较两个LOB或者两个LOB的一部分。

  - 如果比较的结果相等返回0，否则返回非零的值。
  - 如果第一个CLOB比第二个小，返回-1；如果第一个CLOB比第二个大，返回1。
  - 如果amount，offset_1，offset_2这几个参数有无效参数返回NULL，有效的偏移量范围是1~LOBMAXSIZE。

  DBMS_LOB.COMPARE函数原型为：

```
DBMS_LOB.COMPARE (
lob_1     IN     BLOB,
lob_2     IN     BLOB,
amount    IN     INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN     INTEGER := 1,
offset_2  IN     INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.COMPARE (
lob_1     IN     CLOB,
```


版权所有 © 华为云计算技术有限公司

```
lob_2     IN     CLOB,
amount    IN     INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN     INTEGER := 1,
offset_2  IN     INTEGER := 1)
RETURN INTEGER;
```

表 10-13 DBMS_LOB.COMPARE 接口参数说明

参数	说明
lob_1	第一个要比较的LOB描述符。
lob_2	第二个要比较的LOB描述符。
amount	要比较的字符数或者字节数， 最大值为DBMS_LOB.LOBMAXSIZE。
offset_1	第一个LOB描述符的偏移量，初始位置是1。
offset_2	第二个LOB描述符的偏移量，初始位置是1。

- DBMS_LOB.SUBSTR

  用于读取一个LOB的子串，返回读取的字节个数或者字符个数，当amount > 1或者amount < 32767，offset < 1或者offset > LOBMAXSIZE时返回值是NULL。

  DBMS_LOB.SUBSTR函数原型为：

```
DBMS_LOB.SUBSTR (
lob_loc     IN     BLOB,
amount      IN     INTEGER := 32767,
offset      IN     INTEGER := 1)
RETURN RAW;

DBMS_LOB.SUBSTR (
lob_loc     IN     CLOB,
amount      IN     INTEGER := 32767,
offset      IN     INTEGER := 1)
RETURN VARCHAR2;
```

表 10-14 DBMS_LOB.SUBSTR 接口参数说明

参数	说明
lob_loc	将要读取子串的LOB描述符，对于BLOB类型的返回值是读取的字节个数，对于CLOB类型的返回值是字符个数。
amount	要读取的字节数或者字符数量。
offset	从开始位置偏移的字符数或者字节数量。

- DBMS_LOB.TRIM

  这个存储过程用于截断指定长度的LOB，执行完这个存储过程会将LOB的长度设置为newlen参数指定的长度。如果对一个空的LOB执行截断操作，不会有任何执行结果；如果指定的长度比LOB的长度长，会产生一个异常。

  DBMS_LOB.TRIM函数原型为：

```
DBMS_LOB.TRIM (
lob_loc     IN OUT     BLOB,
newlen      IN         INTEGER);
```


版权所有 © 华为云计算技术有限公司

```
DBMS_LOB.TRIM (
lob_loc    IN          OUT CLOB,
newlen     IN          INTEGER);
```

表 10-15 DBMS_LOB.TRIM 接口参数说明

参数	说明
lob_loc	待读入的指定BLOB类型的对象。
newlen	截断后LOB的新长度对于BLOB是字节数，对于CLOB是字符数。

- DBMS_LOB.CREATETEMPORARY

  这个存储过程创建一个临时的BLOB或者CLOB，这个存储过程仅用于语法上的兼容，并无实际意义。

  DBMS_LOB.CREATETEMPORARY函数原型为：

```
DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT      BLOB,
cache      IN          BOOLEAN,
dur        IN          INTEGER);

DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT      CLOB,
cache      IN          BOOLEAN,
dur        IN          INTEGER);
```

表 10-16 DBMS_LOB.CREATETEMPORARY 接口参数说明

参数	说明
lob_loc	LOB描述符。
cache	仅用于语法上的兼容。
dur	仅用于语法上的兼容。

- DBMS_LOB.APPEND

  存储过程APPEND用于将源LOB内容拼接到目标LOB中。

  DBMS_LOB.APPEND函数原型为：

```
DBMS_LOB.APPEND (
dest_lob    IN OUT      BLOB,
src_lob     IN          BLOB);

DBMS_LOB.APPEND (
dest_lob    IN OUT      CLOB,
src_lob     IN          CLOB);
```

表 10-17 DBMS_LOB.APPEND 接口参数说明

参数	说明
dest_lob	要写入的LOB描述符。
src_lob	读取的LOB描述符。

版权所有 © 华为云计算技术有限公司

示例

```
--获取字符串的长度
SELECT DBMS_LOB.GETLENGTH('12345678');

DECLARE
myraw  RAW(100);
amount INTEGER :=2;
buffer INTEGER :=1;
begin
DBMS_LOB.READ('123456789012345',amount,buffer,myraw);
dbms_output.put_line(myraw);
end;
/

CREATE TABLE blob_Table (t1 blob) DISTRIBUTE BY REPLICATION;
CREATE TABLE blob_Table_bak (t2 blob) DISTRIBUTE BY REPLICATION;
INSERT INTO blob_Table VALUES('abcdef');
INSERT INTO blob_Table_bak VALUES('22222');

DECLARE
str varchar2(100) := 'abcdef';
source raw(100);
dest blob;
copyto blob;
amount int;
PSV_SQL varchar2(100);
PSV_SQL1 varchar2(100);
a int :=1;
len int;
BEGIN
source := utl_raw.cast_to_raw(str);
amount := utl_raw.length(source);

PSV_SQL :='select * from blob_Table for update';
PSV_SQL1 := 'select * from blob_Table_bak for update';

EXECUTE IMMEDIATE PSV_SQL into dest;
EXECUTE IMMEDIATE PSV_SQL1 into copyto;

DBMS_LOB.WRITE(dest, amount, 1, source);
DBMS_LOB.WRITEAPPEND(dest, amount, source);

DBMS_LOB.ERASE(dest, a, 1);
DBMS_OUTPUT.PUT_LINE(a);
DBMS_LOB.COPY(copyto, dest, amount, 10, 1);
DBMS_LOB.CLOSE(dest);
RETURN;
END;
/

--删除表
DROP TABLE blob_Table;
DROP TABLE blob_Table_bak;
```

10.10.2 DBMS_RANDOM

接口介绍

高级功能包DBMS_RANDOM支持的所有接口请参见**表10-18**。

版权所有 © 华为云计算技术有限公司

表 10-18 DBMS_RANDOM 接口参数说明

接口名称	描述
DBMS_RANDOM.SEED	设置一个随机数的种子。
DBMS_RANDOM.VALUE	生成一个大小介于指定的low及high之间的随机数。

- DBMS_RANDOM.SEED

  存储过程SEED用于设置一个随机数的种子。DBMS_RANDOM.SEED函数原型为：

  ```
  DBMS_RANDOM.SEED (seed  IN  INTEGER);
  ```

  表 10-19 DBMS_RANDOM.SEED 接口参数说明

参数	描述
seed	用于产生一个随机数的种子。

- DBMS_RANDOM.VALUE

  存储过程VALUE生成一个大小介于指定的low及high之间的随机数。
  DBMS_RANDOM.VALUE函数原型为：

  ```
  DBMS_RANDOM.VALUE(
  low  IN  NUMBER,
  high IN  NUMBER)
  RETURN NUMBER;
  ```

  表 10-20 DBMS_RANDOM.VALUE 接口参数说明

参数	描述
low	指定随机数大小的下边界，生成的随机数大于或等于low。
high	指定随机数大小的上边界，生成的随机数小于high。

  说明

  实际上，只要求这里的参数类型是NUMERIC即可，对于左右边界的大小并没有要求。

示例

生成0到1之间的随机数：

```
SELECT DBMS_RANDOM.VALUE(0,1);
```

生成0到100之间的随机整数。随机整数大于等于low参数的指定值，小于high参数的指定值。

```
SELECT TRUNC(DBMS_RANDOM.VALUE(0,100));
```


版权所有 © 华为云计算技术有限公司

10.10.3 DBMS_OUTPUT

接口介绍

高级功能包DBMS_OUTPUT支持的所有接口请参见**表10-21**。

表 10-21 DBMS_OUTPUT

接口名称	描述
DBMS_OUTPUT.PUT_LINE	输出指定的文本，文本长度不能超过32767字节。
DBMS_OUTPUT.PUT	将指定的文本输出到指定文本的前面，不添加换行符，文本长度不能超过32767字节。
DBMS_OUTPUT.ENABLE	设置输出缓冲区的大小。若不指定，缓冲区最大只能容纳20000字节，缓冲区最小可设置为2000字节，若设置小于2000字节将按2000字节处理。

- DBMS_OUTPUT.PUT_LINE

存储过程PUT_LINE向消息缓冲区写入一行带有行结束符的文本。
DBMS_OUTPUT.PUT_LINE函数原型为：

```
DBMS_OUTPUT.PUT_LINE (
item IN VARCHAR2);
```

表 10-22 DBMS_OUTPUT.PUT_LINE 接口参数说明

参数	描述
item	写入消息缓冲区的文本。

- DBMS_OUTPUT.PUT

存储过程PUT将指定的文本输出到指定文本的前面，不添加换行符。
DBMS_OUTPUT.PUT函数原型为：

```
DBMS_OUTPUT.PUT (
item IN VARCHAR2);
```

表 10-23 DBMS_OUTPUT.PUT 接口参数说明

参数	描述
item	写入指定文本前的文本。

- DBMS_OUTPUT.ENABLE

版权所有 © 华为云计算技术有限公司

存储过程ENABLE设置输出缓冲区的大小。如果不指定，则缓冲区最大只能容纳20000字节。DBMS_OUTPUT.ENABLE函数原型为：

```
DBMS_OUTPUT.ENABLE (
buf IN INTEGER);
```

表 10-24 DBMS_OUTPUT.ENABLE 接口参数说明

参数	描述
buf	设置输出缓冲区的大小。

示例

```
BEGIN
    DBMS_OUTPUT.ENABLE(50);
    DBMS_OUTPUT.PUT ('hello, ');
    DBMS_OUTPUT.PUT_LINE('database!');--输出hello, database!
END;
/
```

10.10.4 UTL_RAW

接口介绍

高级功能包UTL_RAW支持的所有接口请参见**表10-25**。

表 10-25 UTL_RAW

接口名称	描述
UTL_RAW.CAST_FROM_BINARY_INTEGER	将INTEGER类型值转换为二进制表示形式（即RAW类型）。
UTL_RAW.CAST_TO_BINARY_INTEGER	将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。
UTL_RAW.LENGTH	获取RAW类型对象的长度。
UTL_RAW.CAST_TO_RAW	将VARCHAR2类型值转化为二进制表示形式（即RAW类型）。

须知

RAW类型的外部表现形式是十六进制，内部存储形式是二进制。例如一个RAW类型的数据11001011的表现形式为‘CB’，即在实际的类型转换中输入的是‘CB’。

- UTL_RAW.CAST_FROM_BINARY_INTEGER

  存储过程CAST_FROM_BINARY_INTEGER将INTEGER类型值转换为二进制表示形式（即RAW类型）。

  UTL_RAW.CAST_FROM_BINARY_INTEGER函数原型为：

版权所有 © 华为云计算技术有限公司

```
UTL_RAW.CAST_FROM_BINARY_INTEGER (
n           IN  INTEGER,
endianess   IN  INTEGER)
RETURN RAW;
```

表 10-26 UTL_RAW.CAST_FROM_BINARY_INTEGER 接口参数说明

参数	描述
n	待转成RAW类型的整型数值。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.CAST_TO_BINARY_INTEGER

  存储过程CAST_TO_BINARY_INTEGER将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。

  UTL_RAW.CAST_TO_BINARY_INTEGER函数原型为：

  ```
  UTL_RAW.CAST_TO_BINARY_INTEGER (
  r           IN  RAW,
  endianess  IN  INTEGER)
  RETURN BINARY_INTEGER;
  ```

  表 10-27 UTL_RAW.CAST_TO_BINARY_INTEGER 接口参数说明

参数	描述
r	二进制表示形式的整型值（即RAW类型）。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.LENGTH

  存储过程LENGTH返回RAW类型对象的长度。

  UTL_RAW.LENGTH函数原型为：

  ```
  UTL_RAW.LENGTH(
  r      IN RAW)
  RETURN INTEGER;
  ```

  表 10-28 UTL_RAW.LENGTH 接口参数说明

参数	描述
r	RAW类型对象

- UTL_RAW.CAST_TO_RAW

  存储过程CAST_TO_RAW将VARCHAR2类型的对象转换成RAW类型。

  UTL_RAW.CAST_TO_RAW函数原型为：

  ```
  UTL_RAW.CAST_TO_RAW(
  c      IN VARCHAR2)
  RETURN RAW;
  ```

 版权所有 © 华为云计算技术有限公司

表 10-29 UTL_RAW.CAST_TO_RAW 接口参数说明

参数	描述
c	待转换的VARCHAR2类型对象

示例

在存储过程中操作RAW数据：

```
CREATE OR REPLACE PROCEDURE proc_raw
AS
str varchar2(100) := 'abcdef';
source raw(100);
amount integer;
BEGIN
source := utl_raw.cast_to_raw(str);--类型转换
amount := utl_raw.length(source);--获取长度
dbms_output.put_line(amount);
END;
/
```

调用存储过程：

```
CALL proc_raw();
```

10.10.5 DBMS_JOB

接口介绍

高级功能包DBMS_JOB支持的所有接口请参见**表10-30**。

表 10-30 DBMS_JOB

接口名称	描述
DBMS_JOB.SUBMIT	提交一个定时任务。作业号由系统自动生成。
DBMS_JOB.SUBMIT_NODE	提交一个定时任务。执行节点由用户指定，作业号由系统自动生成。
DBMS_JOB.ISUBMIT	提交一个定时任务。作业号由用户指定。
DBMS_JOB.REMOVE	通过作业号来删除定时任务。
DBMS_JOB.BROKEN	禁用或者启用定时任务。
DBMS_JOB.CHANGE	修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。
DBMS_JOB.WHAT	修改定时任务的任务内容属性。
DBMS_JOB.NEXT_DATE	修改定时任务的下次执行时间属性。

版权所有 © 华为云计算技术有限公司

接口名称	描述
DBMS_JOB.INTERVAL	修改定时任务的执行间隔属性。
DBMS_JOB.CHANGE_OWNER	修改定时任务的属主。
DBMS_JOB.CHANGE_NODE	修改定时任务的执行节点。

- DBMS_JOB.SUBMIT

  存储过程SUBMIT提交一个系统提供的定时任务。

  DBMS_JOB.SUBMIT函数原型为：

```
DBMS_JOB.SUBMIT(
what         IN   TEXT,
next_date    IN   TIMESTAMP DEFAULT sysdate,
job_interval IN   TEXT  DEFAULT 'null',
job          OUT  INTEGER);
```

  说明

  当创建一个定时任务（DBMS_JOB）时，系统默认将当前数据库和用户名与当前创建的定时任务（DBMS_JOB）绑定起来。该接口函数可以通过call或select调用，如果通过select调用，可以不填写出参。如果在存储过程中，则需要通过perform调用该接口函数。

  表 10-31 DBMS_JOB.SUBMIT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
what	text	IN	否	要执行的SQL语句。支持一个或多个“DML”、“匿名块”、“调用存储过程的语句”或3种混合的场景。
next_date	timestamp	IN	否	下次作业运行时间。默认值为当前系统时间（sysdate）。如果是过去时间，在提交作业时表示立即执行。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式，可以是interval表达式，也可以是sysdate加上一个numeric值（例如：sysdate+1.0/24）。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。
job	integer	OUT	否	作业号。范围为1～32767。当使用select调用dbms.submit时，该参数可以省略。

  示例：

版权所有 © 华为云计算技术有限公司

```
select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1');

select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1.0/24');

CALL DBMS_JOB.SUBMIT('INSERT INTO T_JOB  VALUES(1);  call pro_1(); call pro_2();',
add_months(to_date('201701','yyyymm'),1), 'date_trunc(''day'',SYSDATE) + 1 +(8*60+30.0)/
(24*60)' ,:jobid);
```

- DBMS_JOB.SUBMIT_NODE

  存储过程SUBMIT提交一个系统提供的定时任务。执行节点由用户指定。该接口仅8.3.0及以上集群版本支持。

  DBMS_JOB.SUBMIT_NODE函数原型为：

```
DBMS_JOB.SUBMIT_NODE(
what         IN   TEXT,
next_date    IN   TIMESTAMP DEFAULT sysdate,
job_interval IN   TEXT DEFAULT 'null',
job_node     IN   TEXT DEFAULT NULL,
job          OUT  INTEGER);
```

表 10-32 DBMS_JOB.SUBMIT_NODE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
what	text	IN	否	要执行的SQL语句。支持一个或多个DML、匿名块、调用存储过程的语句或3种混合的场景。
next_date	timestamp	IN	否	下次作业运行时间。默认值为当前系统时间（sysdate）。如果是过去时间，在提交作业时表示立即执行。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式，可以是interval表达式，也可以是sysdate加上一个numeric值（例如：sysdate+1.0/24）。如果为空值或字符串"null"则表示只执行一次，执行后JOB状态STATUS变成'd'则不再执行。
node	text	IN	是	作业执行节点名称。
job	integer	OUT	否	作业号。范围为1～32767。当使用select调用dbms.submit时，该参数可以省略。

  示例：

```
select DBMS_JOB.SUBMIT_NODE('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate
+1','coordinator1');

select DBMS_JOB.SUBMIT_NODE('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1.0/24');

CALL DBMS_JOB.SUBMIT('INSERT INTO T_JOB  VALUES(1); call pro_1(); call pro_2();',
add_months(to_date('201701','yyyymm'),1), 'date_trunc(''day'',SYSDATE) + 1 +(8*60+30.0)/(24*60)',
'coordinator1', :jobid);
```

- DBMS_JOB.ISUBMIT

版权所有 © 华为云计算技术有限公司

ISUBMIT与SUBMIT语法功能相同，但其第一个参数是入参，即指定的作业号，SUBMIT最后一个参数是出参，表示系统自动生成的作业号。

示例：

```
CALL dbms_job.isubmit(101, 'insert_msg_statistic1;', sysdate, 'sysdate+3.0/24');
```

说明

DWS的pgstats持久化功能将内存中的统计信息写入pg_stat_object系统表，如果新安装的9.1.0.100及以上集群版本，会占用1作为job_id。如果为低版本升级到9.1.0.100及以上集群版本，并且之前pg_job中有任务，那么会找到一个未被占用的job_id作为持久化任务的ID。所以在使用dbms_job.isubmit接口时需注意，不能和已经存在的pgstats持久化任务的ID重复，否则会导致任务注册失败。

- DBMS_JOB.REMOVE

  存储过程REMOVE删除指定的定时任务。

  DBMS_JOB.REMOVE函数原型为：

  ```
  REMOVE(job IN INTEGER);
  ```

  表 10-33 DBMS_JOB.REMOVE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。

  示例：

  ```
  CALL dbms_job.remove(101);
  ```

- DBMS_JOB.BROKEN

  存储过程BROKEN禁用或者启用定时任务。

  DBMS_JOB.BROKEN函数原型为：

  ```
  DBMS_JOB.BROKEN(
  job          IN   INTEGER,
  broken       IN   BOOLEAN,
  next_date    IN   TIMESTAMP  DEFAULT  sysdate);
  ```

  表 10-34 DBMS_JOB.BROKEN 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
broken	boolean	IN	否	状态标志位，true代表禁用，false代表启用。具体true或false值更新当前job；如果为空值，则不改变原有job的状态。

版权所有 © 华为云计算技术有限公司

参数	类型	入参/出参	是否可以为空	描述
next_date	timestamp	IN	是	下次运行时间，默认为当前系统时间。如果参数broken状态为true，则更新该参数为'4000-1-1'；如果参数broken状态为false，且如果参数next_date不为空值，则更新指定job的next_date值，如果next_date为空值，则不更新next_date值。该参数可以省略，为默认值。

示例：

```
CALL dbms_job.broken(101,true);
CALL dbms_job.broken(101,false,sysdate);
```

- DBMS_JOB.CHANGE

  存储过程CHANGE修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。

  DBMS_JOB.CHANGE函数原型为：

```
DBMS_JOB.CHANGE(
job          IN   INTEGER,
what         IN   TEXT,
next_date    IN   TIMESTAMP,
interval     IN   TEXT);
```

表 10-35 DBMS_JOB.CHANGE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	是	执行的存储过程名或者sql语句块。如果该参数为空值，则不更新指定job的what值，否则更新指定job的what值。
next_date	timestamp	IN	是	下次运行时间。如果该参数为空值，则不更新指定job的next_date值，否则更新指定job的next_date值。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果该参数为空值，则不更新指定job的interval值；如果该参数不为空值，会校验interval是否为有效的时间类型或interval类型，则更新指定job的interval值。如果为字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。

版权所有 © 华为云计算技术有限公司

示例：

```
CALL dbms_job.change(101, 'call userproc();', sysdate, 'sysdate + 1.0/1440');
CALL dbms_job.change(101, 'insert into tbl_a values(sysdate);', sysdate, 'sysdate + 1.0/1440');
```

- DBMS_JOB.WHAT

  存储过程WHAT修改定时任务的任务内容属性。

  DBMS_JOB.WHAT函数原型为：

```
DBMS_JOB.WHAT(
job             IN     INTEGER,
what            IN     TEXT);
```

表 10-36 DBMS_JOB.WHAT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	否	执行的存储过程调用或者sql语句块。

说明

- 当what参数是一个或多个可以执行成功的sql语句/程序块/调用存储过程时，该接口函数才能被执行成功，否则会执行失败。
- 若what参数为一个简单的insert、update等语句，需要在表前加模式名。

示例：

```
CALL dbms_job.what(101, 'call userproc();');
CALL dbms_job.what(101, 'insert into tbl_a values(sysdate);');
```

- DBMS_JOB.NEXT_DATE

  存储过程NEXT_DATE修改定时任务的下次执行时间属性。

  DBMS_JOB.NEXT_DATE函数原型为：

```
DBMS_JOB.NEXT_DATE(
job          IN     INTEGER,
next_date    IN     TIMESTAMP);
```

表 10-37 DBMS_JOB.NEXT_DATE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
next_date	timestamp	IN	否	下次运行时间。

版权所有 © 华为云计算技术有限公司

说明

如果输入的next_date的值小于当前日期值，该job会立即执行一次。

示例：

```
CALL dbms_job.next_date(101,sysdate);
```

- DBMS_JOB.INTERVAL

  存储过程INTERVAL修改定时任务的执行间隔属性。

  DBMS_JOB.INTERVAL函数原型为：

```
DBMS_JOB.INTERVAL(
job             IN   INTEGER,
interval         IN    TEXT);
```

表 10-38 DBMS_JOB.INTERVAL 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。interval是否为有效的时间类型或interval类型。

示例：

```
CALL dbms_job.interval(101, 'sysdate + 1.0/1440');
```

说明

对于指定job正在运行状态（即job_status为'r'）时，不允许通过remove、change、next_date、what、interval等接口删除或修改job的参数信息。

- DBMS_JOB.CHANGE_OWNER

  存储过程CHANGE_OWNER修改定时任务的属主。

  DBMS_JOB.CHANGE_OWNER函数原型为：

```
DBMS_JOB.CHANGE_OWNER(
job             IN     INTEGER,
new_owner       IN     NAME);
```

表 10-39 DBMS_JOB.CHANGE_OWNER 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
new_owner	name	IN	否	新的用户名。

版权所有 © 华为云计算技术有限公司

示例：

```
CALL dbms_job.change_owner(101, 'alice');
```

- DBMS_JOB.CHANGE_NODE

  存储过程CHANGE_NODE修改定时任务的执行节点。该接口仅8.3.0及以上集群版本支持。

  DBMS_JOB.CHANGE_NODE函数原型为：

```
DBMS_JOB.CHANGE_NODE(
job             IN     INTEGER,
new_node        IN     text);
```

表 10-40 DBMS_JOB.CHANGE_NODE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
new_node	text	IN	否	新的执行节点。

示例：

```
CALL dbms_job.change_node(101, 'coordinator2');
```

约束说明

1. 创建一个新job后，该job从属于当前coordinator（即：该job仅在当前coordinator上调度和执行），其他coordinator不会调度和执行该job。所有coordinator都可以查看、修改、删除其他CN创建的job。
2. job只能通过dbms_job高级包提供的接口进行创建、更新、删除操作，因为高级包的接口中会考虑所有CN间job信息的同步和pg_jobs表主键的关联操作，如果通过DML语句对pg_jobs表进行增删改，会导致job信息在CN间不一致和系统表无法关联变更的混乱问题，会严重影响job内部的管理。
3. 由于用户创建的每个任务和CN绑定，若不开启CN故障自动迁移功能，当任务运行过程中，该CN故障，则该任务的状态无法实时刷新。如果在任务未执行时CN故障，则该CN上的任务都得不到正常的调度和执行。建议开启CN故障自动迁移功能，故障CN上的作业会迁移至其他CN继续调度。
4. job在定时执行过程中，需要在当前job所属的CN上实时更新该job的运行状态、最近执行开始时间、最近执行结束时间、下次开始时间、失败次数（如果job执行失败）等相关参数信息到pg_jobs系统表中，并同步到其他CN，保证job信息的一致性。如果其他CN存在节点故障，那么job所属CN会同步超时重发的处理，导致job执行时间变长，但CN间同步超时失败后，原CN上pg_jobs表中job的相关信息仍然能正常更新，且job能正常执行成功。当故障CN恢复正常后，可能出现该CN上pg_jobs表中当前job的执行时间、运行状态等参数与原CN上不一致的情况，需要原CN上再次执行该job后才能保证job信息的同步。
5. 对于并发同时有多个job到达执行时间的场景，由于会为每个job创建一个线程来执行job，由于系统内部启动每个线程的时间会有延迟，因此会导致同时并发执行的job的开始时间有延迟，每个job的延迟时间在0.1ms左右。
6. job中待执行SQL语句有长度限制，最长为8K。

版权所有 © 华为云计算技术有限公司

10.10.6 DBMS_SQL

接口介绍

高级功能包DBMS_SQL支持的接口请参见**表10-41**。

表 10-41 DBMS_SQL

接口名称	描述
DBMS_SQL.OPEN_CURSOR	打开一个游标。
DBMS_SQL.CLOSE_CURSOR	关闭一个已打开的游标。
DBMS_SQL.PARSE	向游标传递一组SQL语句，目前只支持SELECT。
DBMS_SQL.EXECUTE	在游标上执行一组动态定义操作。
DBMS_SQL.FETCH_ROWS	读取游标一行数据。
DBMS_SQL.DEFINE_COLUMN	动态定义一个列。
DBMS_SQL.DEFINE_COLUMN_CHAR	动态定义一个char类型的列。
DBMS_SQL.DEFINE_COLUMN_INT	动态定义一个int类型的列。
DBMS_SQL.DEFINE_COLUMN_LONG	动态定义一个long类型的列。
DBMS_SQL.DEFINE_COLUMN_RAW	动态定义一个raw类型的列。
DBMS_SQL.DEFINE_COLUMN_TEXT	动态定义一个text类型的列。
DBMS_SQL.DEFINE_COLUMN_UNKNOWN	动态定义一个未知列（类型不识别时入此接口）。
DBMS_SQL.COLUMN_VALUE	读取一个已动态定义的列值。
DBMS_SQL.COLUMN_VALUE_CHAR	读取一个已动态定义的列值（指定char类型）。
DBMS_SQL.COLUMN_VALUE_INT	读取一个已动态定义的列值（指定int类型）。
DBMS_SQL.COLUMN_VALUE_LONG	读取一个已动态定义的列值（指定long类型）。
DBMS_SQL.COLUMN_VALUE_RAW	读取一个已动态定义的列值（指定raw类型）。
DBMS_SQL.COLUMN_VALUE_TEXT	读取一个已动态定义的列值（指定text类型）。
DBMS_SQL.COLUMN_VALUE_UNKNOWN	读取一个已动态定义的列值（类型不识别时入此接口）。
DBMS_SQL.IS_OPEN	检查游标是否已打开。

版权所有 © 华为云计算技术有限公司

说明

- 建议使用dbms_sql.define_column及dbms_sql.column_value定义参数列。
- 当结果集大于work_mem设定值时会触发结果集临时下盘，但最大阈值不超过512MB。

- DBMS_SQL.OPEN_CURSOR

  该函数用来打开一个游标，是后续dbms_sql各项操作的前提。该函数不传入任何参数，内部自动递增生成游标ID，并作为返回值返回给integer定义的变量。

  DBMS_SQL.OPEN_CURSOR函数原型为：

  ```
  DBMS_SQL.OPEN_CURSOR (
  )
  RETURN INTEGER;
  ```

- DBMS_SQL.CLOSE_CURSOR

  该函数用来关闭一个游标，是dbms_sql各项操作的结束。如果在存储过程结束时没有调用该函数，则该游标占用的内存仍然会保存，因此关闭游标非常重要。由于异常情况的发生会中途退出存储过程，导致游标未能关闭，因此建议存储过程中有异常处理，将该接口包含在内。

  DBMS_SQL.CLOSE_CURSOR函数原型为：

  ```
  DBMS_SQL.CLOSE_CURSOR (
  cursorid     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 10-42 DBMS_SQL.CLOSE_CURSOR 接口说明

参数名称	描述
cursorid	需要关闭的游标ID号。

- DBMS_SQL.PARSE

  该函数用来解析给定游标的查询语句，被传入的查询语句会立即执行。目前仅支持SELECT查询语句的解析，且语句参数仅可通过text类型传递，长度不大于1G。

  DBMS_SQL.PARSE函数的原型为：

  ```
  DBMS_SQL.PARSE (
  cursorid      IN INTEGER,
  query_string  IN TEXT,
  language_flag IN INTEGER
  )
  RETURN BOOLEAN;
  ```

  表 10-43 DBMS_SQL.PARSE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID。
query_string	执行的查询语句。
language_flag	版本语言号，目前只支持1。

- DBMS_SQL.EXECUTE

  该函数用来执行一个给定的游标。该函数接收一个游标ID，运行后获得的数据用于后续操作。目前仅支持SELECT查询语句的执行。

版权所有 © 华为云计算技术有限公司

DBMS_SQL.EXECUTE函数的原型为：

```
DBMS_SQL.EXECUTE(
cursorid     IN INTEGER
)
RETURN INTEGER;
```

表 10-44 DBMS_SQL.EXECUTE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID。

- DBMS_SQL.FETCH_ROWS

  该函数返回符合查询条件的数据行数，每一次运行该接口都会获取到新的行数的集合，直到数据读取完毕获取不到新行为止。

  DBMS_SQL.FETCH_ROWS函数的原型为：

  ```
  DBMS_SQL.FETCH_ROWS(
  cursorid     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 10-45 DBMS_SQL.FETCH_ROWS 接口说明

参数名称	描述
cursorid	执行的游标ID。

- DBMS_SQL.DEFINE_COLUMN

  该函数用来定义从给定游标返回的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN(
  cursorid     IN INTEGER,
  position     IN INTEGER,
  column_ref   IN ANYELEMENT,
  column_size  IN INTEGER default 1024
  )
  RETURN INTEGER;
  ```

  表 10-46 DBMS_SQL.DEFINE_COLUMN 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column_ref	任意类型的变量，可根据变量类型选择适当的接口动态定义列。
column_size	定义的列的长度。

- DBMS_SQL.DEFINE_COLUMN_CHAR

 版权所有 © 华为云计算技术有限公司

该函数用来定义从给定游标返回的CHAR类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

DBMS_SQL.DEFINE_COLUMN_CHAR函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT,
column_size     IN INTEGER
)
RETURN INTEGER;
```

表 10-47 DBMS_SQL.DEFINE_COLUMN_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column	需要定义的某类型的参数变量。
column_size	动态定义列长度。

- DBMS_SQL.DEFINE_COLUMN_INT

  该函数用来定义从给定游标返回的INT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_INT函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_INT(
  cursorid     IN INTEGER,
  position     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 10-48 DBMS_SQL.DEFINE_COLUMN_INT 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。

- DBMS_SQL.DEFINE_COLUMN_LONG

  该函数用来定义从给定游标返回的长列类型（非数据类型long）的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。长列的大小限制为1G。

  DBMS_SQL.DEFINE_COLUMN_LONG函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_LONG(
  cursorid     IN INTEGER,
  position     IN INTEGER
  )
  RETURN INTEGER;
  ```


版权所有 © 华为云计算技术有限公司

表 10-49 DBMS_SQL.DEFINE_COLUMN_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。

- DBMS_SQL.DEFINE_COLUMN_RAW

  该函数用来定义从给定游标返回的RAW类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_RAW函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_RAW(
  cursorid     IN INTEGER,
  position     IN INTEGER,
  column       IN BYTEA,
  column_size     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 10-50 DBMS_SQL.DEFINE_COLUMN_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column	RAW类型的参数变量。
column_size	列的长度。

- DBMS_SQL.DEFINE_COLUMN_TEXT

  该函数用来定义从给定游标返回的TEXT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_TEXT函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_TEXT(
  cursorid     IN INTEGER,
  position     IN INTEGER,
  max_size     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 10-51 DBMS_SQL.DEFINE_COLUMN_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
max_size	定义的TEXT类型的最大长度。

版权所有 © 华为云计算技术有限公司

- DBMS_SQL.DEFINE_COLUMN_UNKNOWN

  该函数用来处理从给定游标返回的未知数据类型的列，该接口仅用于类型不识别时的报错退出。

  DBMS_SQL.DEFINE_COLUMN_UNKNOWN函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_UNKNOWN(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT
)
RETURN INTEGER;
```

  表 10-52 DBMS_SQL.DEFINE_COLUMN_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column	动态定义的参数。

- DBMS_SQL.COLUMN_VALUE

  该函数用来返回给定游标给定位置的游标元素值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE函数的原型为：

```
DBMS_SQL.COLUMN_VALUE(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT ANYELEMENT
)
RETURN ANYELEMENT;
```

  表 10-53 DBMS_SQL.COLUMN_VALUE 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column_value	定义的列的返回值。

- DBMS_SQL.COLUMN_VALUE_CHAR

  该函数用来返回给定游标给定位置的游标CHAR类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_CHAR函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_CHAR(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT CHARACTER,
err_num               INOUT NUMERIC default 0,
actual_length           INOUT INTEGER default 1024
)
RETURN RECORD;
```


版权所有 © 华为云计算技术有限公司

表 10-54 DBMS_SQL.COLUMN_VALUE_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column_value	返回值。
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度。

- DBMS_SQL.COLUMN_VALUE_INT

  该函数用来返回给定游标给定位置的游标INT类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。DBMS_SQL.COLUMN_VALUE_INT函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_INT(
cursorid              IN    INTEGER,
position              IN    INTEGER
)
RETURN INTEGER;
```

表 10-55 DBMS_SQL.COLUMN_VALUE_INT 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。

- DBMS_SQL.COLUMN_VALUE_LONG

  该函数用来返回给定游标给定位置的游标长列（非long/bigint整型）类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_LONG函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_LONG(
cursorid              IN    INTEGER,
position              IN    INTEGER,
length                IN    INTEGER,
off_set               IN    INTEGER,
column_value          INOUT TEXT,
actual_length         INOUT INTEGER default 1024
)
RETURN RECORD;
```

表 10-56 DBMS_SQL.COLUMN_VALUE_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
length	返回值的长度。

版权所有 © 华为云计算技术有限公司

参数名称	描述
off_set	返回值的起始位置。
column_value	返回值。
actual_length	实际返回值的长度。

- DBMS_SQL.COLUMN_VALUE_RAW

  该函数用来返回给定游标给定位置的游标RAW类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_RAW函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_RAW(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT BYTEA,
err_num               INOUT NUMERIC default 0,
actual_length           INOUT INTEGER default 1024
)
RETURN RECORD;
```

  表 10-57 DBMS_SQL.COLUMN_VALUE_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column_value	返回的列值。
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度，不能长于此值，否则截断。

- DBMS_SQL.COLUMN_VALUE_TEXT

  该函数用来返回给定游标中给定位置的游标TEXT类型值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_TEXT函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_TEXT(
cursorid              IN    INTEGER,
position              IN    INTEGER
)
RETURN TEXT;
```

  表 10-58 DBMS_SQL.COLUMN_VALUE_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。

版权所有 © 华为云计算技术有限公司

- DBMS_SQL.COLUMN_VALUE_UNKNOWN

  该函数用来返回给定游标给定位置的游标未知类型的值，该接口为类型不支持时的报错处理接口。

  DBMS_SQL.COLUMN_VALUE_UNKNOWN函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_UNKNOWN(
cursorid              IN    INTEGER,
position              IN    INTEGER,
COLUMN_TYPE           IN    TEXT
)
RETURN TEXT;
```

  表 10-59 DBMS_SQL.COLUMN_VALUE_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID。
position	动态定义列在查询中的位置。
column_type	返回的参数类型。

- DBMS_SQL.IS_OPEN

  该函数用来返回游标的当前状态：打开、解析、执行、定义。取值为TRUE，关闭后为FALSE，未知时报错，其余默认为关闭。

  DBMS_SQL.IS_OPEN函数的原型为：

```
DBMS_SQL.IS_OPEN(
cursorid              IN    INTEGER
)
RETURN BOOLEAN;
```

  表 10-60 DBMS_SQL.IS_OPEN 接口说明

参数名称	描述
cursorid	需要查询的游标ID。

示例

```
--在存储过程中操作raw数据
create or replace procedure pro_dbms_sql_all_02(in_raw raw,v_in int,v_offset int)
as
cursorid int;
v_id int;
v_info bytea :=1;
query varchar(2000);
execute_ret int;
define_column_ret_raw bytea :='1';
define_column_ret int;
begin
drop table if exists pro_dbms_sql_all_tb1_02 ;
create table pro_dbms_sql_all_tb1_02(a int ,b blob);
insert into pro_dbms_sql_all_tb1_02 values(1,HEXTORAW('DEADBEEE'));
insert into pro_dbms_sql_all_tb1_02 values(2,in_raw);
query := 'select * from pro_dbms_sql_all_tb1_02 order by 1';
--打开游标
cursorid := dbms_sql.open_cursor();
--编译游标
dbms_sql.parse(cursorid, query, 1);
```


版权所有 © 华为云计算技术有限公司

```
--定义列
define_column_ret:= dbms_sql.define_column(cursorid,1,v_id);
define_column_ret_raw:= dbms_sql.define_column_raw(cursorid,2,v_info,10);
--执行
execute_ret := dbms_sql.execute(cursorid);
loop
exit when (dbms_sql.fetch_rows(cursorid) <= 0);
--获取值
dbms_sql.column_value(cursorid,1,v_id);
dbms_sql.column_value_raw(cursorid,2,v_info,v_in,v_offset);
--输出结果
dbms_output.put_line('id:'|| v_id || ' info:' || v_info);
end loop;
--关闭游标
dbms_sql.close_cursor(cursorid);
end;
/
--调用存储过程
call pro_dbms_sql_all_02(HEXTORAW('DEADBEEF'),0,1);

--删除存储过程
DROP PROCEDURE pro_dbms_sql_all_02;
```

10.11 DWS 存储过程调试

语法

RAISE有以下五种语法格式：

图 10-34 raise_format::=

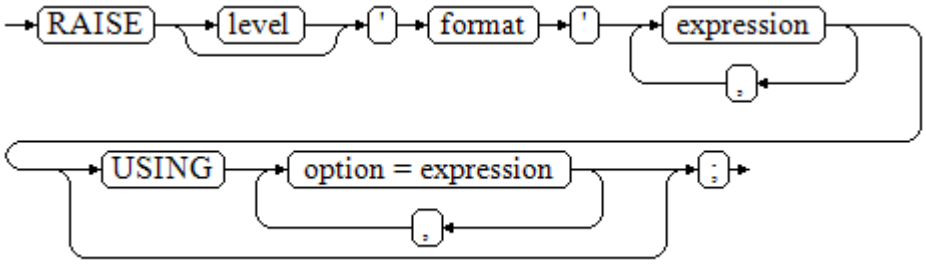
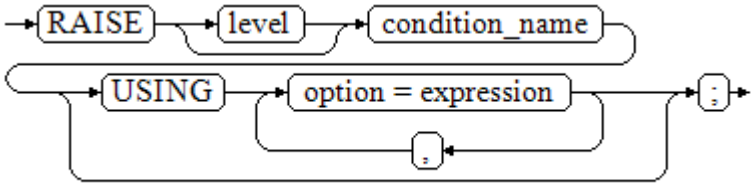


图 10-35 raise_condition::=



版权所有 © 华为云计算技术有限公司

图 10-36 raise_sqlstate::=

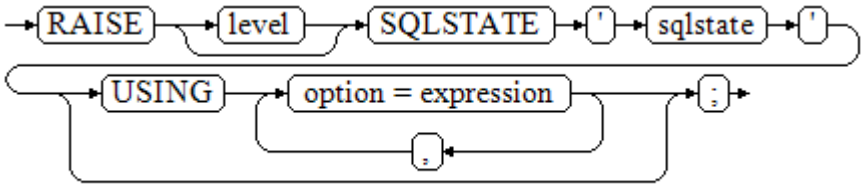


图 10-37 raise_option::=

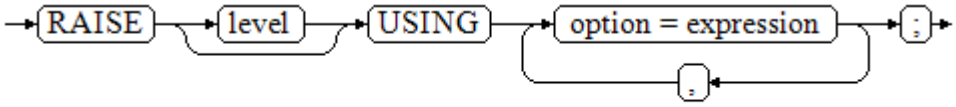
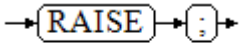


图 10-38 raise::=



参数说明：

- level选项用于指定错误级别，有DEBUG，LOG，INFO，NOTICE，WARNING以及EXCEPTION（默认值）。EXCEPTION上报一个正常终止当前事务的异常，其他的仅产生不同异常级别的信息。特殊级别的错误信息是否报告到客户端、写到服务器日志由**log_min_messages**和**client_min_messages**这两个配置参数控制。
- format：格式字符串，指定要报告的错误消息文本。格式字符串后可跟表达式，用于向消息文本中插入。在格式字符串中，%由format后面跟着的参数的值替换，%%用于打印出%。例如：

  ```
  --v_job_id 将替换字符串中的 %:
  RAISE NOTICE 'Calling cs_create_job(%)',v_job_id;
  ```

- option = expression：向错误报告中添加另外的信息。关键字option可以是MESSAGE、DETAIL、HINT以及ERRCODE，并且每一个expression可以是任意的字符串。
  - MESSAGE，指定错误消息文本，这个选项不能用于在USING前包含一个格式字符串的RAISE语句中。
  - DETAIL，说明错误的详细信息。
  - HINT，用于打印出提示信息。
  - ERRCODE，向报告中指定错误码（SQLSTATE）。可以使用条件名称或者直接用五位字符的SQLSTATE错误码。
- condition_name：错误码对应的条件名。
- sqlstate：错误码。

如果在RAISE EXCEPTION命令中既没有指定条件名也没有指定SQLSTATE，默认用RAISE EXCEPTION (P0001)。如果没有指定消息文本，默认用条件名或者SQLSTATE作为消息文本。

版权所有 © 华为云计算技术有限公司

须知

当由SQLSTATE指定了错误码，则不局限于已定义的错误码，可以选择任意包含五个数字或者大写的ASCII字母的错误码，而不是00000。建议避免使用以三个0结尾的错误码，因为这种错误码是类别码，会被整个种类捕获。

说明

图10-38所示的语法不接任何参数。这种形式仅用于一个BEGIN块中的EXCEPTION语句，它使得错误重新被处理。

示例

终止事务时，给出错误和提示信息：

```
CREATE OR REPLACE PROCEDURE proc_raise1(user_id in integer)
AS
BEGIN
RAISE EXCEPTION 'Noexistence ID --> %',user_id USING HINT = 'Please check your user ID';
END;
/

call proc_raise1(300011);

--执行结果
ERROR:  Noexistence ID --> 300011
HINT:  Please check your user ID
```

两种设置SQLSTATE的方式：

```
CREATE OR REPLACE PROCEDURE proc_raise2(user_id in integer)
AS
BEGIN
RAISE 'Duplicate user ID: %',user_id USING ERRCODE = 'unique_violation';
END;
/

\set VERBOSITY verbose
call proc_raise2(300011);

--执行结果
ERROR:  Duplicate user ID: 300011
SQLSTATE: 23505
LOCATION:  exec_stmt_raise, pl_exec.cpp:3482
```

如果主要的参数是条件名或者是SQLSTATE，可以使用：

RAISE division_by_zero;

RAISE SQLSTATE '22012';

例如：

```
CREATE OR REPLACE PROCEDURE division(div in integer, dividend in integer)
AS
DECLARE
res int;
    BEGIN
    IF dividend=0 THEN
        RAISE division_by_zero;
        RETURN;
    ELSE
        res := div/dividend;
        RAISE INFO 'division result: %', res;
        RETURN;
    END IF;
```


版权所有 © 华为云计算技术有限公司

```
    END;
/
call division(3,0);

--执行结果
ERROR:  division_by_zero
```

或者另一种方式：

```
RAISE unique_violation USING MESSAGE = 'Duplicate user ID: ' || user_id;
```


版权所有 © 华为云计算技术有限公司

11 使用 PostGIS Extension

11.1 PostGIS 概述

说明

- PostGIS Extension依赖的第三方软件需要用户进行单独安装，用户如需使用PostGIS功能，请提交工单或联系技术支持人员提交申请。
- 如果用户在使用中出现“ERROR：EXTENSION is not yet supported.”这种报错，则表示没有安装PostGIS软件包，请联系技术支持进行获取。
- 在存算分离架构下，PostGIS扩展功能（CREATE EXTENSION语句）仅支持在主VW上执行。

DWS提供PostGIS Extension（版本为PostGIS-2.4.2和PostGIS-3.2.2）。PostGIS Extension是PostgreSQL的空间数据库扩展，提供如下空间信息服务功能：空间对象、空间索引、空间操作函数和空间操作符。PostGIS Extension完全遵循OpenGIS规范。

DWS中PostGIS Extension依赖第三方开源软件如下。

- PostGIS 2.4.2依赖第三方开源软件：
  - Geos 3.6.2
  - Proj 4.9.2
  - Json 0.12.1
  - Libxml2 2.7.1
  - Gdal 1.11.0
- PostGIS 3.2.2依赖第三方开源软件：
  - Geos-3.11.0
  - Proj-6.0.0
  - Json 0.12.1
  - Libxml2 2.7.1
  - Sqlite3
  - protobuf-c 1.4.1
  - protobuf 3.6.1

 版权所有 © 华为云计算技术有限公司

11.2 PostGIS 使用

说明

PostGIS Extension依赖的第三方软件需要用户进行单独安装，用户如需使用PostGIS功能，请提交工单或联系技术支持人员提交申请。

如果用户在使用中出现“ERROR：EXTENSION is not yet supported.”这种报错，则表示没有安装PostGIS软件包，请联系技术支持进行获取。

创建 Extension

创建PostGIS Extension可直接使用CREATE EXTENSION命令进行创建，建议在public模式下创建PostGIS：

```
CREATE EXTENSION postgis;
```

使用 Extension

PostGIS Extension函数调用格式为：

```
SELECT GisFunction (Param1, Param2,.....);
```

其中GisFunction为函数名，Param1、Param2等为函数参数名。下列SQL语句展示PostGIS的简单使用，对于各函数的具体使用，请参考《**PostGIS-2.4.2用户手册**》。

示例1：几何表的创建。

```
CREATE TABLE cities ( id integer, city_name varchar(50) );
SELECT AddGeometryColumn('cities', 'position', 4326, 'POINT', 2);
```

示例2：几何数据的插入。

```
INSERT INTO cities (id, position, city_name) VALUES (1,ST_GeomFromText('POINT(-9.5 23)',4326),'CityA');
INSERT INTO cities (id, position, city_name) VALUES (2,ST_GeomFromText('POINT(-10.6 40.3)',4326),'CityB');
INSERT INTO cities (id, position, city_name) VALUES (3,ST_GeomFromText('POINT(20.8 30.3)',4326), 'CityC');
```

示例3：计算三个城市间任意两个城市距离。

```
SELECT p1.city_name,p2.city_name,ST_Distance(p1.position,p2.position) FROM cities AS p1, cities AS p2
WHERE p1.id > p2.id;
```

删除 Extension

在DWS中删除PostGIS Extension的方法如下所示：

```
DROP EXTENSION postgis [CASCADE];
```

说明

如果Extension被其他对象依赖（如创建的几何表），需要加入CASCADE（级联）关键字，删除所有依赖对象。

11.3 PostGIS 支持和限制

支持数据类型

DWS的PostGIS Extension支持如下数据类型:

版权所有 © 华为云计算技术有限公司

- box2d
- box3d
- geometry_dump
- geometry
- geography
- raster

⚠ 注意

- PostGIS-3.2.2暂不支持raster相关功能，如需使用相关功能，请使用PostGIS-2.4.2。
- 创建Postgis和使用Postgis非同一用户时，请设置如下GUC参数：
  ```
  SET behavior_compat_options = 'bind_procedure_searchpath';
  ```

支持的操作符和函数列表

说明

PostGIS笛卡尔积ST_Intersects函数拥有缓存机制，外表的空间数据结构缓存命中率较高，在内表和外表的宽度有较大差异时，缓存宽表的数据可以避免多次加载大对象至缓存，对于性能的优化尤为明显，实际使用时可以通过**Join顺序的Hint**指定宽表作为外表使计划符合该场景。

表 11-1 PostGIS-2.4.2 支持的操作符和函数列表

函数分类	包含函数
Management Functions	AddGeometryColumn、DropGeometryColumn、DropGeometryTable、PostGIS_Full_Version、PostGIS_GEOS_Version、PostGIS_Liblwgeom_Version、PostGIS_Lib_Build_Date、PostGIS_Lib_Version、PostGIS_PROJ_Version、PostGIS_Scripts_Build_Date、PostGIS_Scripts_Installed、PostGIS_Version、PostGIS_LibXML_Version、PostGIS_Scripts_Released、Populate_Geometry_Columns 、UpdateGeometrySRID

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Constructors	ST_BdPolyFromText 、ST_BdMPolyFromText 、ST_Box2dFromGeoHash、ST_GeogFromText、ST_GeographyFromText、ST_GeogFromWKB、ST_GeomCollFromText、ST_GeomFromEWKB、ST_GeomFromEWKT、ST_GeometryFromText、ST_GeomFromGeoHash、ST_GeomFromGML、ST_GeomFromGeoJSON、ST_GeomFromKML、ST_GMLToSQL、ST_GeomFromText 、ST_GeomFromWKB、ST_LineFromMultiPoint、ST_LineFromText、ST_LineFromWKB、ST_LinestringFromWKB、ST_MakeBox2D、ST_3DMakeBox、ST_MakeEnvelope、ST_MakePolygon、ST_MakePoint、ST_MakePointM、ST_MLineFromText、ST_MPointFromText、ST_MPolyFromText、ST_Point、ST_PointFromGeoHash、ST_PointFromText、ST_PointFromWKB、ST_Polygon、ST_PolygonFromText、ST_WKBToSQL、ST_WKTToSQL
Geometry Accessors	GeometryType、ST_Boundary、ST_CoordDim、ST_Dimension、ST_EndPoint、ST_Envelope、ST_ExteriorRing、ST_GeometryN、ST_GeometryType、ST_InteriorRingN、ST_IsClosed、ST_IsCollection、ST_IsEmpty、ST_IsRing、ST_IsSimple、ST_IsValid、ST_IsValidReason、ST_IsValidDetail、ST_M、ST_NDims、ST_NPoints、ST_NRings、ST_NumGeometries、ST_NumInteriorRings、ST_NumInteriorRing、ST_NumPatches、ST_NumPoints、ST_PatchN、ST_PointN、ST_SRID、ST_StartPoint、ST_Summary、ST_X、ST_XMax、ST_XMin、ST_Y、ST_YMax、ST_YMin、ST_Z、ST_ZMax、ST_Zmflag、ST_ZMin
Geometry Editors	ST_AddPoint、ST_Affine、ST_Force2D、ST_Force3D、ST_Force3DZ、ST_Force3DM、ST_Force4D、ST_ForceCollection、ST_ForceSFS、ST_ForceRHR、ST_LineMerge、ST_CollectionExtract、ST_CollectionHomogenize、ST_Multi、ST_RemovePoint、ST_Reverse、ST_Rotate、ST_RotateX、ST_RotateY、ST_RotateZ、ST_Scale、ST_Segmentize、ST_SetPoint、ST_SetSRID、ST_SnapToGrid、ST_Snap、ST_Transform、ST_Translate、ST_TransScale
Geometry Outputs	ST_AsBinary、ST_AsEWKB、ST_AsEWKT、ST_AsGeoJSON、ST_AsGML、ST_AsHEXEWKB、ST_AsKML、ST_AsLatLonText 、ST_AsSVG、ST_AsText、ST_AsX3D、ST_GeoHash
Operators	&&、&&&、&<、&<|、&>、<<、<<|、=、>>、@ 、|&> 、|>>、~、~=、<->、<#>

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Spatial Relationships and Measurements	ST_3DClosestPoint、ST_3DDistance、ST_3DDWithin、ST_3DDFullyWithin、ST_3DIntersects、ST_3DLongestLine、ST_3DMaxDistance、ST_3DShortestLine、ST_Area、ST_Azimuth、ST_Centroid、ST_ClosestPoint、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_LineCrossingDirection、ST_Disjoint、ST_Distance、ST_HausdorffDistance、ST_MaxDistance、ST_DistanceSphere、ST_DistanceSpheroid、ST_DFullyWithin、ST_DWithin、ST_Equals、ST_HasArc、ST_Intersects、ST_Length、ST_Length2D、ST_3DLength、ST_Length_Spheroid、ST_Length2D_Spheroid、ST_3DLength_Spheroid、ST_LongestLine、ST_OrderingEquals、ST_Overlaps、ST_Perimeter、ST_Perimeter2D、ST_3DPerimeter、ST_PointOnSurface、ST_Project、ST_Relate、ST_RelateMatch、ST_ShortestLine、ST_Touches、ST_Within
Geometry Processing	ST_Buffer、ST_BuildArea、ST_Collect、ST_ConcaveHull、ST_ConvexHull、ST_CurveToLine、ST_DelaunayTriangles、ST_Difference、ST_Dump、ST_DumpPoints、ST_DumpRings、ST_FlipCoordinates、ST_Intersection、ST_LineToCurve、ST_MakeValid、ST_MemUnion、ST_MinimumBoundingCircle、ST_Polygonize、ST_Node、ST_OffsetCurve、ST_RemoveRepeatedPoints、ST_SharedPaths、ST_Shift_Longitude、ST_Simplify、ST_SimplifyPreserveTopology、ST_Split、ST_SymDifference、ST_Union、ST_UnaryUnion
Linear Referencing	ST_LineInterpolatePoint、ST_LineLocatePoint、ST_LineSubstring、ST_LocateAlong、ST_LocateBetween、ST_LocateBetweenElevations、ST_InterpolatePoint、ST_AddMeasure
Miscellaneous Functions	ST_Accum、Box2D、Box3D、ST_Expand、ST_Extent、ST_3Dextent、Find_SRID、ST_MemSize
Exceptional Functions	PostGIS_AddBBox、PostGIS_DropBBox、PostGIS_HasBBox
Raster Management Functions	AddRasterConstraints、DropRasterConstraints、AddOverviewConstraints、DropOverviewConstraints、PostGIS_GDAL_Version、PostGIS_Raster_Lib_Build_Date、PostGIS_Raster_Lib_Version、ST_GDALDrivers、UpdateRasterSRID
Raster Constructors	ST_AddBand、ST_AsRaster、ST_Band、ST_MakeEmptyRaster、ST_Tile、ST_FromGDALRaster

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Raster Accessors	ST_GeoReference、ST_Height、ST_IsEmpty、ST_MetaData、ST_NumBands、ST_PixelHeight、ST_PixelWidth、ST_ScaleX、ST_ScaleY、ST_RasterToWorldCoord、ST_RasterToWorldCoordX、ST_RasterToWorldCoordY、ST_Rotation、ST_SkewX、ST_SkewY、ST_SRID、ST_Summary、ST_UpperLeftX、ST_UpperLeftY、ST_Width、ST_WorldToRasterCoord、ST_WorldToRasterCoordX、ST_WorldToRasterCoordY
Raster Band Accessors	ST_BandMetaData、ST_BandNoDataValue、ST_BandIsNoData、ST_BandPath、ST_BandPixelType、ST_HasNoBand
Raster Pixel Accessors and Setters	ST_PixelAsPolygon、ST_PixelAsPolygons、ST_PixelAsPoint、ST_PixelAsPoints、ST_PixelAsCentroid、ST_PixelAsCentroids、ST_Value、ST_NearestValue、ST_Neighborhood、ST_SetValue、ST_SetValues、ST_DumpValues、ST_PixelOfValue
Raster Editors	ST_SetGeoReference、ST_SetRotation、ST_SetScale、ST_SetSkew、ST_SetSRID、ST_SetUpperLeft、ST_Resample、ST_Rescale、ST_Reskew、ST_SnapToGrid、ST_Resize、ST_Transform
Raster Band Editors	ST_SetBandNoDataValue、ST_SetBandIsNoData
Raster Band Statistics and Analytics	ST_Count、ST_CountAgg、ST_Histogram、ST_Quantile、ST_SummaryStats、ST_SummaryStatsAgg、ST_ValueCount
Raster Outputs	ST_AsBinary、ST_AsGDALRaster、ST_AsJPEG、ST_AsPNG、ST_AsTIFF
Raster Processing	ST_Clip、ST_ColorMap、ST_Intersection、ST_MapAlgebra、ST_Reclass、ST_Union、ST_Distinct4ma、ST_InvDistWeight4ma、ST_Max4ma、ST_Mean4ma、ST_Min4ma、ST_MinDist4ma、ST_Range4ma、ST_StdDev4ma、ST_Sum4ma、ST_Aspect、ST_HillShade、ST_Roughness、ST_Slope、ST_TPI、ST_TRI、Box3D、ST_ConvexHull、ST_DumpAsPolygons、ST_Envelope、ST_MinConvexHull、ST_Polygon、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Disjoint、ST_Intersects、ST_Overlaps、ST_Touches、ST_SameAlignment、ST_NotSameAlignmentReason、ST_Within、ST_DWithin、ST_DFullyWithin
Raster Operators	&&、&<、&>、=、@、~=、~

版权所有 © 华为云计算技术有限公司

表 11-2 PostGIS-3.2.2 支持的操作符和函数列表

函数分类	包含函数
Management Functions	AddGeometryColumn、DropGeometryColumn、DropGeometryTable、PostGIS_Full_Version、PostGIS_GEOS_Version、PostGIS_Liblwgeom_Version、PostGIS_Lib_Build_Date、PostGIS_Lib_Version、PostGIS_PROJ_Version、PostGIS_Scripts_Build_Date、PostGIS_Scripts_Installed、PostGIS_Version、PostGIS_LibXML_Version、PostGIS_Scripts_Released、Populate_Geometry_Columns 、UpdateGeometrySRID 、PostGIS_Libprotobuf_Version 、PostGIS_Wagyu_Version
Geometry Constructors	ST_BdPolyFromText 、ST_BdMPolyFromText 、ST_Box2dFromGeoHash、ST_GeneratePoints、ST_GeogFromText、ST_GeographyFromText、ST_GeogFromWKB、ST_GeomCollFromText、ST_GeomFromEWKB、ST_GeomFromEWKT、ST_GeometryFromText、ST_GeomFromGeoHash、ST_GeomFromGML、ST_GeomFromGeoJSON、ST_GeomFromKML、ST_GMLToSQL、ST_GeomFromText 、ST_GeomFromWKB、ST_LineFromMultiPoint、ST_LineFromText、ST_LineFromWKB、ST_LinestringFromWKB、ST_MakeBox2D、ST_3DMakeBox、ST_MakeEnvelope、ST_MakePolygon、ST_MakePoint、ST_MakePointM、ST_MLineFromText、ST_MPointFromText、ST_MPolyFromText、ST_Point、ST_Points、ST_PointFromGeoHash、ST_PointFromText、ST_PointFromWKB、ST_Polygon、ST_PolygonFromText、ST_WKBToSQL、ST_WKTToSQL、Geography_Distance_Knn、Geometry_Distance_Cpa、Geometry_Hash、ST_3Dlineinterpolate、ST_AsEncodedPolyline
Geometry Accessors	GeometryType、ST_Boundary、ST_CoordDim、ST_Dimension、ST_EndPoint、ST_Envelope、ST_ExteriorRing、ST_GeometryN、ST_GeometryType、ST_InteriorRingN、ST_IsClosed、ST_IsCollection、ST_IsEmpty、ST_IsPolygonCCW、ST_IsPolygonCW、ST_IsRing、ST_IsSimple、ST_IsValid、ST_IsValidReason、ST_IsValidDetail、ST_M、ST_NDims、ST_NPoints、ST_NRings、ST_NumGeometries、ST_NumInteriorRings、ST_NumInteriorRing、ST_NumPatches、ST_NumPoints、ST_PatchN、ST_PointN、ST_SRID、ST_StartPoint、ST_Summary、ST_X、ST_XMax、ST_XMin、ST_Y、ST_YMax、ST_YMin、ST_Z、ST_ZMax、ST_Zmflag、ST_ZMin、ST_Wrapx、ST_Asmvt

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Editors	ST_AddPoint、ST_Affine、ST_Force2D、ST_Force3D、ST_Force3DZ、ST_Force3DM、ST_Force4D、ST_ForceCollection、ST_ForcePolygonCCW、ST_ForcePolygonCW、ST_ForceSFS、ST_ForceRHR、ST_LineMerge、ST_CollectionExtract、ST_CollectionHomogenize、ST_Multi、ST_Normalize、ST_RemovePoint、ST_Reverse、ST_Rotate、ST_RotateX、ST_RotateY、ST_RotateZ、ST_Scale、ST_Segmentize、ST_SetPoint、ST_SetSRID、ST_SnapToGrid、ST_Snap、ST_Transform、ST_Translate、ST_TransScale、ST_AsmvtGeom、ST_isvalidTrajectory、ST_linefromencodedpolyline、ST_lineinterpolatepoints、ST_MaximuminScribedCircle、ST_OrientedEnvelope、ST_QuantizeCoordinates、ST_ReducePrecision、ST_Scroll、ST_SetEffectiveArea、ST_simplifyvw、ST_square、ST_squaregrid、ST_Swapordinates、ST_Voronoilines、ST_VoronoiPolygons
Geometry Outputs	ST_AsBinary、ST_AsEWKB、ST_AsEWKT、ST_AsGeoJSON、ST_AsGML、ST_AsHEXEWKB、ST_AsKML、ST_AsLatLonText 、ST_AsSVG、ST_AsText、ST_AsTwkb、ST_AsX3D、ST_GeoHash、Json、Jsonb、ST_GeomfromGeojson
Operators	&&、&&&、&<、&<|、&>、<<、<<|、=、>>、@ 、|&> 、|>>、~、~=、<->、<#>、<->、|=|、<<->>
Spatial Relationships and Measurements	ST_3DClosestPoint、ST_3DDistance、ST_3DDWithin、ST_3DDFullyWithin、ST_3DIntersects、ST_3DLongestLine、ST_3DMaxDistance、ST_3DShortestLine、ST_Area、ST_Azimuth、ST_Centroid、ST_ClosestPoint、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_LineCrossingDirection、ST_Disjoint、ST_Distance、ST_HausdorffDistance、ST_MaxDistance、ST_DistanceSphere、ST_DistanceSpheroid、ST_DFullyWithin、ST_DWithin、ST_Equals、ST_HasArc、ST_Intersects、ST_Length、ST_Length2D、ST_3DLength、ST_LengthSpheroid、ST_Length2DSpheroid、ST_LongestLine、ST_MinimumBoundingRadius、ST_OrderingEquals、ST_Overlaps、ST_Perimeter、ST_Perimeter2D、ST_3DPerimeter、ST_PointOnSurface、ST_Project、ST_Relate、ST_RelateMatch、ST_ShortestLine、ST_Touches、ST_Within、_ST_DistancerectTree、_ST_DistancerectTreeCached、_ST_SorTableHash

版权所有 © 华为云计算技术有限公司

函数分类	包含函数
Geometry Processing	ST_Buffer、ST_BuildArea、ST_ClipByBox2D、ST_ClusterDBSCAN、ST_ClusterIntersecting、ST_ClusterKMeans、ST_ClusterWithin、ST_Collect、ST_ConcaveHull、ST_ConvexHull、ST_CurveToLine、ST_DelaunayTriangles、ST_Difference、ST_Dump、ST_DumpPoints、ST_DumpRings、ST_FlipCoordinates、ST_Intersection、ST_LineToCurve、ST_MakeValid、ST_MemUnion、ST_MinimumBoundingCircle、ST_Polygonize、ST_Node、ST_OffsetCurve、ST_RemoveRepeatedPoints、ST_SharedPaths、ST_ShiftLongitude、ST_Simplify、ST_SimplifyPreserveTopology、ST_Split、ST_Subdivide、ST_SymDifference、ST_Union、ST_UnaryUnion、ST_BoundingDiagonal、ST_ChaikinsMoothing、ST_ClosestPointofApproach、ST_CollectionExtract、ST_CPAwithin、ST_DistanceCPA、ST_DumpSegments、ST_EstimatedExtent、ST_Filterbym、ST_SetEffectiveArea、ST_Forcecurve
Linear Referencing	ST_LineInterpolatePoint、ST_LineLocatePoint、ST_LineSubstring、ST_LocateAlong、ST_LocateBetween、ST_LocateBetweenElevations、ST_InterpolatePoint、ST_AddMeasure
Miscellaneous Functions	Array_Agg、Box2D、Box3D、ST_Expand、ST_Extent、ST_3Dextent、Find_SRID、ST_MemSize
Exceptional Functions	PostGIS_AddBBox、PostGIS_DropBBox、PostGIS_HasBBox

空间索引

DWS数据库的PostGIS Extension支持GiST (Generalized Search Tree) 空间索引（分区表除外）。相比于B-tree索引，GiST索引适应于任意类型的非常规数据结构，可有效提高几何和地理数据信息的检索效率。

使用如下命令创建GiST索引：

```
CREATE INDEX indexname ON tablename USING gist ( geometryfield );
```

扩展限制

- 只支持行存表，不支持列存索引。
- 只支持Oracle兼容格式数据库。
- 不支持拓扑对象管理模块Topology。
- 不支持BRIN索引。
- spatial_ref_sys表在扩容期间只支持查询操作。

版权所有 © 华为云计算技术有限公司

插件升级兼容性

PostGIS 2.4.2版本升级到3.2.2版本时，存在部分函数不兼容或没有完全的前向兼容，导致出现升级前后函数功能表现不一致的情况，所以在PostGIS升级时，需从业务自身角度来评估升级不兼容带来的影响。

不兼容函数列表如下。

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
3.2.2新增函数	无	ST_IsPolygonCW(geometry)
	无	ST_IsPolygonCCW(geometry)
	无	ST_PointInsideCircle(geometry,float8,float8,float8)
	无	ST_ForcePolygonCW(geometry)
	无	ST_ForcePolygonCCW(geometry)
	无	ST_Normalize(geom geometry)
	无	ST_AsTWKB(geom geometry, prec int4 default 0, prec_z int4 default 0, prec_m int4 default 0, with_sizes boolean default false, with_boxes boolean default false)
	无	ST_AsTWKB(geom geometry[], ids bigint[], prec int4 default 0, prec_z int4 default 0, prec_m int4 default 0, with_sizes boolean default false, with_boxes boolean default false)
	无	ST_MakeLine (geometry[])
	无	ST_TileEnvelope(zoom integer, x integer, y integer, bounds geometry DEFAULT 'SRID=3857;LINESTRING(-20037508.342789244 -20037508.342789244, 20037508.342789244 20037508.342789244)'::geometry, margin float8 DEFAULT 0.0)
	无	ST_ClusterIntersecting(geometry[])
	无	ST_ClusterWithin(geometry[], float8)
	无	ST_ClusterDBSCAN (geometry, eps float8, minpoints int)

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	无	ST_Scale(geometry,geometry,origin geometry)
	无	ST_GeneratePoints(area geometry, npoints integer, seed integer)
	无	ST_FrechetDistance(geom1 geometry, geom2 geometry, float8 default -1)
	无	ST_Points(geometry)
	无	ST_ClipByBox2d(geom geometry, box box2d)
	无	ST_Subdivide(geom geometry, maxvertices integer DEFAULT 256, gridSize float8 DEFAULT -1.0)
	无	ST_ClusterIntersecting (geometry)
	无	ST_ClusterWithin (geometry, float8)
	无	ST_ClusterKMeans(geom geometry, k integer, max_radius float8 default null)
	无	ST_AsText(geometry, int4)
	无	ST_AsEWKT(geography, int4)
	无	_ST_CoveredBy(geog1 geography, geog2 geography)
	无	ST_Point(float8, float8, srid integer)
3.2.2不再支持	ST_3DLength_spheroid(geometry, spheroid)	无
	ST_length2d_spheroid(geometry, spheroid)	无
	ST_locate_between_measures(geometry, float8, float8)	无
	ST_locate_along_measure(geometry, float8)	无
	ST_Buffer(geometry,float8, text)	无
	ST_GeneratePoints(area geometry, npoints integer)	无

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_Combine_BBox(box3d, geometry)	无
	ST_Combine_BBox(box2d, geometry)	无
	pgis_abs_in(cstring)	无
	pgis_abs_out(pgis_abs)	无
	pgis_abs (internallength = 16, input = pgis_abs_in, output = pgis_abs_out, alignment = double)	无
	ST_MemUnion(geometry)	无
	pgis_geometry_accum_finalfn(pgis_abs)	无
	ST_MakeLine (geometry)	无
	ST_Accum (geometry)	输入单geometry数据，会直接输出该geometry的值，多行geometry数据则会报错不支持。
	_ST_AsKML(int4,geometry, int4, text)	无
	ST_MemUnion(geometry)	无
	_ST_AsGeoJson(int4, geometry, int4, int4)	无
	ST_AsGeoJson(gj_version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0)	无
	_ST_DWithin(geography, geography, float8, boolean)	无
	ST_point_inside_circle(geometry,float8,float8,float8)	无
	ST_CurveToLine(geometry)	无
	ST_Shift_Longitude(geometry)	无，改用ST_ShiftLongitude
	ST_find_extent(text,text,text) , ST_find_extent(text,text)	无，改用ST_FindExtent

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_mem_size(geometry)	无，改用ST_MemSize
	ST_length_spheroid(geometry, spheroid)	无，改用ST_LengthSpheroid
	ST_distance_spheroid(geom1 geometry, geom2 geometry,spheroid)	无，改用ST_DistanceSpheroid
	ST_force_2d(geometry)	无，改用ST_Force2D
	ST_force_3dz(geometry)	无，改用ST_Force3DZ
	ST_force_3d(geometry)	无，改用ST_Force3D
	ST_force_3dm(geometry)	无，改用ST_Force3DM
	ST_force_4d(geometry)	无，改用ST_Force4D
	ST_force_collection(geometry)	无，改用ST_ForceCollection
	ST_line_locate_point(geom1 geometry, geom2 geometry)	无，改用ST_LineLocatePoint
	ST_line_interpolate_point(geometry, float8)	无，改用ST_LineInterpolatePoint
	ST_Buffer(geometry,float8)	无，改用ST_Buffer
3.2.2参数类型变化	pgis_geometry_accum_transfn(pgis_abs, geometry)	pgis_geometry_accum_transfn(internal, geometry)
	pgis_geometry_accum_transfn(pgis_abs, geometry, float8)	pgis_geometry_accum_transfn(internal, geometry, float8)
	pgis_geometry_accum_transfn(pgis_abs, geometry, float8, int)	pgis_geometry_accum_transfn(internal, geometry, float8, int)
	pgis_geometry_union_finalfn(pgis_abs)	pgis_geometry_union_finalfn(internal)
	pgis_geometry_collect_finalfn(pgis_abs)	pgis_geometry_collect_finalfn(internal)
	pgis_geometry_polygonize_finalfn(pgis_abs)	pgis_geometry_polygonize_finalfn(internal)
	pgis_geometry_clusterintersecting_finalfn(pgis_abs)	pgis_geometry_clusterintersecting_finalfn(internal)
	ST_Union (geometry)	ST_Union (geometry)

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
	ST_Collect (geometry)	ST_Collect (geometry)
	ST_Buffer(geometry,float8, integer)	ST_Buffer(geom geometry, radius float8, quadsegs integer)
3.2.2接口变更	ST_AsKML(version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix text DEFAULT null)	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix TEXT default ' ')
	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15)	ST_AsKML(geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix TEXT default ' ')
	ST_AsKML(version int4, geom geometry, maxdecimaldigits int4 DEFAULT 15, nprefix text DEFAULT null)	ST_AsGML(version int4, geog geography, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0, nprefix text DEFAULT 'gml', id text DEFAULT '')
3.2.2默认参数变化	ST_SymDifference(geom1 geometry, geom2 geometry)	ST_SymDifference(geom1 geometry, geom2 geometry, gridSize float8 DEFAULT -1.0)
	ST_UnaryUnion(geometry)	ST_UnaryUnion(geometry, gridSize float8 DEFAULT -1.0)
	ST_AsGeoJson(geom geometry, maxdecimaldigits int4 DEFAULT 15, options int4 DEFAULT 0)	ST_AsGeoJson(geom geometry, maxdecimaldigits int4 DEFAULT 9, options int4 DEFAULT 8)
	ST_Buffer(geometry,float8, cstring)	ST_Buffer(geom geometry, radius float8, options text DEFAULT ' ')
	_ST_DWithin(geography, geography, float8, boolean)	_ST_DWithin(geog1 geography, geog2 geography, tolerance float8, use_spheroid boolean DEFAULT true)
	ST_IsValidDetail(geometry)	ST_IsValidDetail(geom geometry, flags int4 DEFAULT 0)
	ST_CurveToLine(geom geometry, tol float8, toltype integer, flags integer)	ST_CurveToLine(geom geometry, tol float8 DEFAULT 32, toltype integer DEFAULT 0, flags integer DEFAULT 0)
3.2.2算子支持hash join和merge join	OPERATOR =	OPERATOR =

版权所有 © 华为云计算技术有限公司

兼容性差异	PostGIS 2.4.2	PostGIS 3.2.2
3.2.2功能变更	ST_ConcaveHull弱校验	ST_ConcaveHull添加强校验，校验更为严格。
3.2.2不定义commutor	OPERATOR &< ,OPERATOR &<|,OPERATOR |&>	OPERATOR &< ,OPERATOR &<|,OPERATOR |&>

11.4 OPEN SOURCE SOFTWARE NOTICE (For PostGIS)

This document contains open source software notice for the product. And this document is confidential information of copyright holder. Recipient shall protect it in due care and shall not disseminate it without permission.

Warranty Disclaimer

This document is provided “as is” without any warranty whatsoever, including the accuracy or comprehensiveness. Copyright holder of this document may change the contents of this document at any time without prior notice, and copyright holder disclaims any liability in relation to recipient’s use of this document.

Open source software is provided by the author “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of open source software, even if advised of the possibility of such damage.

Copyright Notice And License Texts

Software: postgis-2.4.2

Copyright notice:

"Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Copyright 2008 Kevin Neufeld

Copyright (c) 2009 Walter Bruce Sinclair

Copyright 2006-2013 Stephen Woodbridge.

Copyright (c) 2008 Walter Bruce Sinclair

Copyright (c) 2012 TJ Holowaychuk <tj@vision-media.ca>

Copyright (c) 2008, by Attractive Chaos <attractivechaos@aol.co.uk>

 版权所有 © 华为云计算技术有限公司

Copyright (c) 2001-2012 Walter Bruce Sinclair

Copyright (c) 2010 Walter Bruce Sinclair

Copyright 2006 Stephen Woodbridge

Copyright 2006-2010 Stephen Woodbridge.

Copyright (c) 2006-2014 Stephen Woodbridge.

Copyright (c) 2017, Even Rouault <even.rouault at spatialys.com>

Copyright (C) 2004-2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2015 Nicklas Avén <nicklas.aven@jordogskog.no>

Copyright 2008 Paul Ramsey

Copyright (C) 2012 Sandro Santilli <strk@kbt.io>

Copyright 2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2014 Sandro Santilli <strk@kbt.io>

Copyright 2013 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2011 Sandro Santilli <strk@kbt.io>

Copyright 2015 Daniel Baston

Copyright 2009 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Kashif Rasul <kashif.rasul@gmail.com> and

Shoaib Burq <saburq@gmail.com>

Copyright 2013 Sandro Santilli <strk@kbt.io>

Copyright 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright 2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Nicklas Avén

Copyright 2011-2016 Regina Obe

Copyright (C) 2008 Paul Ramsey

Copyright (C) 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2010-2012 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2015 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2013 Nicklas Avén

Copyright (C) 2016 Sandro Santilli <strk@kbt.io>

Copyright 2017 Darafei Praliaskouski <me@komzpa.net>

Copyright (c) 2016, Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011-2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007-2008 Mark Cave-Ayland

Copyright (C) 2001-2006 Refractions Research Inc.

Copyright 2015 Daniel Baston <dbaston@gmail.com>

Copyright 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2012-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2012-2015 Sandro Santilli <strk@kbt.io>

Copyright 2001-2006 Refractions Research Inc.

Copyright (C) 2004 Refractions Research Inc.

Copyright 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright 2009-2010 Sandro Santilli <strk@kbt.io>

Copyright 2015-2016 Daniel Baston <dbaston@gmail.com>

Copyright 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2007-2008 Mark Cave-Ayland

Copyright 2012-2013 Oslandia <infos@oslandia.com>

Copyright (C) 2015-2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2001-2003 Refractions Research Inc.

Copyright 2016 Sandro Santilli <strk@kbt.io>

Copyright 2011 Kashif Rasul <kashif.rasul@gmail.com>

Copyright (C) 2014 Nicklas Avén

Copyright (C) 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 1984, 1989-1990, 2000-2015 Free Software Foundation, Inc.

Copyright (C) 2011 Paul Ramsey

 版权所有 © 华为云计算技术有限公司

Copyright 2001-2003 Refractions Research Inc.

Copyright 2009-2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2010-2012 Oslandia

Copyright 2006 Corporacion Autonoma Regional de Santander

Copyright 2013 Nicklas Avén

Copyright 2011-2016 Arrival 3D, Regina Obe

Copyright (C) 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009-2012 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 - Oslandia

Copyright (C) 2006 Mark Leslie <mark.leslie@lisasoft.com>

Copyright (C) 2008-2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2009-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 Olivier Courtin <olivier.courtin@camptocamp.com>

Copyright 2010 Nicklas Avén

Copyright 2012 Paul Ramsey

Copyright 2011 Nicklas Avén

Copyright 2002 Thamer Alharbash

Copyright 2011 OSGeo

Copyright (C) 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2004-2007 Refractions Research Inc.

Copyright 2010 LISAsoft Pty Ltd

Copyright 2010 Mark Leslie

Copyright (c) 1999, Frank Warmerdam

Copyright 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 2007, Frank Warmerdam

Copyright 2008 OpenGeo.org

Copyright (C) 2008 OpenGeo.org

Copyright (C) 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2010 LISAsoft

Copyright (C) 2010 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 1999, 2001, Frank Warmerdam

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2016-2017 Björn Harrtell <bjorn@wololo.org>

Copyright (C) 2017 Danny Götte <danny.goette@fem.tu-ilmenau.de>

Copyright 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2012 (C) Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2006 Refractions Research Inc.

Copyright 2009 Paul Ramsey <pramsey@opengeo.org>

Copyright 2001-2009 Refractions Research Inc.

Copyright (C) 2010 Olivier Courtin <olivier.courtin@oslandia.com>

By Nathan Wagner, copyright disclaimed,

this entire file is in the public domain

Copyright 2009-2011 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2001-2005 Refractions Research Inc.

Copyright 2001-2011 Refractions Research Inc.

Copyright 2009-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2010 Sandro Santilli <strk@kbt.io>

Copyright 2012 J Smith <dark.panda@gmail.com>

Copyright 2009 - 2010 Oslandia

Copyright 2009 Oslandia

Copyright 2001-2005 Refractions Research Inc.

Copyright 2016 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2016 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2011 OpenGeo.org

Copyright (c) 2003-2017, Troy D. Hanson http:troydhanson.github.com/uthash/

Copyright (C) 2011 Regents of the University of California

Copyright (C) 2011-2013 Regents of the University of California

Copyright (C) 2010-2011 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2010-2011 David Zwarg <dzwarg@azavea.com>

Copyright (C) 2009-2011 Pierre Racine <pierre.racine@sbf.ulaval.ca>

Copyright (C) 2009-2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2008-2009 Sandro Santilli <strk@kbt.io>

Copyright (C) 2013 Nathaneil Hunter Clay <clay.nathaniel@gmail.com

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

Copyright (C) 2013 Bborie Park <dustymugs@gmail.com>

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

(C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2012 Regents of the University of California

Copyright (C) 2013 Regents of the University of California

Copyright (C) 2012-2013 Regents of the University of California

Copyright (C) 2009 Sandro Santilli <strk@kbt.io>"

License: The GPL v2 License.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

 版权所有 © 华为云计算技术有限公司

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

 版权所有 © 华为云计算技术有限公司

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

 版权所有 © 华为云计算技术有限公司

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

 版权所有 © 华为云计算技术有限公司

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A

 版权所有 © 华为云计算技术有限公司

PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author

 版权所有 © 华为云计算技术有限公司

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Software:Geos

Copyright notice：

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2013 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

 版权所有 © 华为云计算技术有限公司

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public

Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom

 版权所有 © 华为云计算技术有限公司

to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and

is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary

 版权所有 © 华为云计算技术有限公司

General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in

non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The

former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under

copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code

 版权所有 © 华为云计算技术有限公司

for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the

Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1

above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of

its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

 版权所有 © 华为云计算技术有限公司

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in

themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or

collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany

it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to

distribute the source code, even though third parties are not compelled to copy the source along with the object code.

 版权所有 © 华为云计算技术有限公司

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a

work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License.

Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object

file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6,

whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work

under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work

 版权所有 © 华为云计算技术有限公司

that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system,

rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception,

the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot

use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

 版权所有 © 华为云计算技术有限公司

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the

integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot

impose that choice.

 版权所有 © 华为云计算技术有限公司

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time.

Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is

copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR

 版权所有 © 华为云计算技术有限公司

CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING

RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

 版权所有 © 华为云计算技术有限公司

That's all there is to it!

Software: JSON-C

Copyright notice:

Copyright (c) 2004, 2005 Metaparadigm Pte. Ltd.

Copyright (c) 2009-2012 Eric Haszlakiewicz

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Copyright (c) 2009 Hewlett-Packard Development Company, L.P.

Copyright 2011, John Resig

Copyright 2011, The Dojo Foundation

Copyright (c) 2012 Eric Haszlakiewicz

Copyright (c) 2009-2012 Hewlett-Packard Development Company, L.P.

Copyright (c) 2008-2009 Yahoo! Inc. All rights reserved.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) 2013 Metaparadigm Pte. Ltd.

License: MIT License

Copyright (c) 2009-2012 Eric Haszlakiewicz

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Software: proj

Copyright notice:

"Copyright (C) 2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2007 Douglas Gregor <doug.gregor@gmail.com>

Copyright (C) 2007 Troy Straszheim

CMake, Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>)

Copyright (C) 2011 Nicolas David <nicolas.david@ign.fr>

Copyright (c) 2000, Frank Warmerdam

Copyright (c) 2011, Open Geospatial Consortium, Inc.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) Charles Karney (2012-2015) <charles@karney.com> and licensed

Copyright (c) 2005, Antonello Andrea

Copyright (c) 2010, Frank Warmerdam

Copyright (c) 1995, Gerald Evenden

Copyright (c) 2000, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2010, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2013, Frank Warmerdam

Copyright (c) 2003 Gerald I. Evenden

Copyright (c) 2012, Frank Warmerdam <warmerdam@pobox.com>

 版权所有 © 华为云计算技术有限公司

Copyright (c) 2002, Frank Warmerdam

Copyright (c) 2004 Gerald I. Evenden

Copyright (c) 2012 Martin Raspaud

Copyright (c) 2001, Thomas Flemming, tf@ttqv.com

Copyright (c) 2002, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2009, Frank Warmerdam

Copyright (c) 2003, 2006 Gerald I. Evenden

Copyright (c) 2011, 2012 Martin Lambers <marlam@marlam.de>

Copyright (c) 2006, Andrey Kiselev

Copyright (c) 2008-2012, Even Rouault <even dot rouault at mines-paris dot org>

Copyright (c) 2001, Frank Warmerdam

Copyright (c) 2001, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2008 Gerald I. Evenden

"

License: MIT License

Please see above

Software: libxml2

Copyright notice:

"See Copyright for the status of this software.

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Copyright (C) 2003 Daniel Veillard.

copy: see Copyright for the status of this software.

copy: see Copyright for the status of this software

copy: see Copyright for the status of this software.

Copyright (C) 2000 Bjorn Reese and Daniel Veillard.

Copy: See Copyright for the status of this software.

See COPYRIGHT for the status of this software

Copyright (C) 2000 Gary Pennington and Daniel Veillard.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007 Free Software Foundation, Inc.

Copyright (C) 1998 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese <breese@users.sourceforge.net>

 版权所有 © 华为云计算技术有限公司

Copyright (C) 2000 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese and Daniel Stenberg.

See Copyright for the status of this software

"

License: MIT License

Please see above

 版权所有 © 华为云计算技术有限公司

12 使用 JDBC 或 ODBC 进行 DWS 二次开发

12.1 开发前准备

如果用户在APP的开发中，使用了连接池机制，那么需要遵循如下规范：

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

否则，连接池里面的连接就是有状态的，会对用户后续使用连接池进行操作的正确性带来影响。

驱动下载

请参见**下载JDBC或ODBC驱动**。

12.2 基于 JDBC 开发

12.2.1 JDBC 开发流程

JDBC（Java Database Connectivity，java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问接口，应用程序可基于它操作数据。DWS库提供了对JDBC 4.0特性的支持，需要使用JDK1.6及以上版本编译程序代码，不支持JDBC桥接ODBC方式。JDBC开发应用程序的具体流程如下：

 版权所有 © 华为云计算技术有限公司

图 12-1 采用 JDBC 开发应用程序的流程

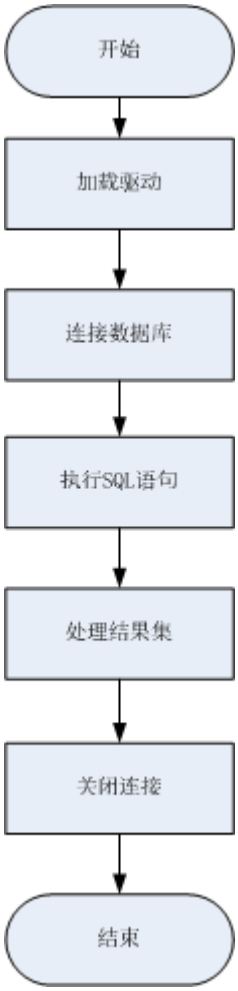


表 12-1 JDBC 开发流程

步骤	描述
加载驱动	下载JDBC驱动并在程序中编译和加载。
连接数据库	通过JDBC驱动连接数据库。
执行SQL语句	应用程序通过执行SQL语句来操作数据库的数据。
处理结果集	不同类型的结果集有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。
关闭连接	在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

12.2.2 JDBC 包与驱动类

JDBC 包

从管理控制台下载包名为dws_9.1.x_jdbc_driver.zip的软件包。

版权所有 © 华为云计算技术有限公司

请参见**下载JDBC或ODBC驱动**。

解压后有两个JDBC的驱动jar包：

- gsjdbc4.jar：与PostgreSQL保持兼容的驱动包，其中类名、类结构与PostgreSQL驱动完全一致，曾经运行于PostgreSQL的应用程序可以直接移植到当前系统使用。
- gsjdbc200.jar：如果同一JVM进程内需要同时访问PostgreSQL及DWS请使用此驱动包，它的主类名为“com.huawei.gauss200.jdbc.Driver”（即将“org.postgresql”替换为“com.huawei.gauss200.jdbc”），数据库连接的URL前缀为“jdbc:gaussdb”，其余与gsjdbc4.jar相同。

驱动类

在创建数据库连接之前，需要加载数据库驱动类“org.postgresql.Driver”（对应包gsjdbc4.jar）或者“com.huawei.gauss200.jdbc.Driver”（对应gsjdbc200.jar）。

说明

由于DWS在JDBC的使用上与PG的使用方法保持兼容，所以同时在同一进程内使用两个JDBC的驱动的时候，可能会类名冲突。

12.2.3 加载驱动

在创建数据库连接之前，需要先加载数据库驱动程序。

加载驱动有两种方法：

- 在代码中创建连接之前任意位置隐含装载：
  Class.forName("org.postgresql.Driver");
- 在JVM启动时参数传递：java -Djdbc.drivers=org.postgresql.Driver jdbctest

  说明

  - 上述jdbctest为测试用例程序的名称。
  - 当使用gsjdbc200.jar时，上面的Driver类名相应修改为"com.huawei.gauss200.jdbc.Driver"

12.2.4 连接数据库

在创建数据库连接之后，才能使用它来执行SQL语句操作数据。

如果您使用的是开源的JDBC驱动程序，应确保数据库参数password_encryption_type取值设置为1，如果参数值不为1，可能会出现连接失败，典型的报错信息比如：“none of the server's SASL authentication mechanisms are supported”，可参考如下操作进行处理：

1. 联系技术支持人员检查数据库参数password_encryption_type是否为1，如果取值不为1，需要将该参数修改为1。
2. 新建一个数据库用户用于连接，或者重置准备使用的数据库用户的密码。
   - 如果您使用的是管理员账号，参见**重置密码**。
   - 如果是普通用户，可以先通过其他客户端工具（例如Data Studio）连接数据库后，使用ALTER USER语句来修改密码。
3. 再尝试连接数据库。

需要执行以上操作的原因：

版权所有 © 华为云计算技术有限公司

- 调整参数的原因：当前MD5算法已被证实可以人工碰撞，已严禁将之用于密码校验算法。DWS 采用默认安全设计，默认禁止MD5算法的密码校验，而PostgreSQL的开源libpq通信协议恰好使用的是MD5算法。所以需要调整密码算法参数password_encryption_type，打开MD5算法。
- 修改密码的原因：DWS 中是不会存储您的密码原文的，而是存储的密码HASH摘要（默认是SHA256摘要），在密码校验时该摘要会与客户端发来的密码摘要进行比对（中间会有加盐操作）。故当您只是单纯调整了密码算法策略时，数据库是无法还原您的密码进而再生成MD5的摘要值的，必须要求您手动修改一次密码或者创建一个新用户，这时新的密码将会采用您设置的HASH算法进行摘要存储，用于下次连接认证。

函数原型

JDBC提供了三个方法，用于创建数据库连接。

- DriverManager.getConnection(String url);
- DriverManager.getConnection(String url, Properties info);
- DriverManager.getConnection(String url, String user, String password);

参数

表 12-2 数据库连接参数

参数	描述
url	gsjdbc4.jar数据库连接描述符。格式如下： ● jdbc:postgresql:database ● jdbc:postgresql://host/database ● jdbc:postgresql://host:port/database ● jdbc:postgresql://host:port[,host:port][...]/database **说明** 使用gsjdbc200.jar时，将“jdbc:postgresql”修改为“jdbc:gaussdb” ● database为要连接的数据库名称。 ● host为数据库服务器名称或IP地址，当集群绑定弹性负载均衡(ELB)时，应设置为ELB的IP地址。 由于安全原因，数据库CN禁止集群内部其他节点无认证接入。如果要在集群内部访问CN，请将JDBC程序部署在CN所在机器，host使用"127.0.0.1"。否则可能会出现“FATAL: Forbid remote connection with trust method!”错误。 建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。 ● port为数据库服务器端口。缺省情况下，会尝试连接到localhost的8000端口的database。 ● 支持多ip端口配置形式，jdbc自动实现了负载均衡，多ip端口配置形式是采取随机访问+failover的方式，这个过程系统会自动忽略不可达IP。 以","隔开，例如jdbc:postgresql://10.10.0.13:8000,10.10.0.14:8000/database ● 使用JDBC连接集群时集群连接地址只支持指定jdbc连接参数，不支持增加变量参数。

版权所有 © 华为云计算技术有限公司

参数	描述
info	数据库连接属性。常用的属性如下： • user：String类型。表示创建连接的数据库用户。 • password：String类型。表示数据库用户的密码。 • ssl：Boolean类型。表示是否使用SSL连接。 • loggerLevel：string类型。为LogStream或LogWriter设置记录到DriverManager中的日志详细级别。目前支持"OFF"、"DEBUG"和"TRACE"。 值为"DEBUG"时，表示只打印DEBUG级别以上的日志，将记录非常少的信息。值等于TRACE时，表示打印DEBUG和TRACE级别的日志，将产生详细的日志信息。默认值为OFF，表示不打印日志。 • prepareThreshold：integer类型。用于确定在转换为服务器端的预备语句之前，要求执行方法PreparedStatement的次数。缺省值是5。 • batchMode : boolean类型，用于确定是否使用batch模式连接。 • fetchsize： integer类型，用于设置数据库链接所创建statement的默认fetchsize。 • ApplicationName：string类型。应用名称，在不做设置时，缺省值为PostgreSQL JDBC Driver。 • allowReadOnly:boolean类型，用于设置connection是否允许设置readonly模式，默认为false，若该参数不被设置为true，则执行connection.setReadOnly不生效。 • blobMode:string类型，用于设置setBinaryStream方法为不同的数据类型赋值，设置为on时表示为blob数据类型赋值，设置为off时表示为bytea数据类型赋值，默认为on。 • connectionExtraInfo：Boolean类型。表示驱动是否上报当前驱动的部署路径、进程属主用户到数据库。 **说明** 取值范围：true或false，默认值为true。设置connectionExtraInfo为true，JDBC驱动会将当前驱动的部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。
user	数据库用户。
password	数据库用户的密码。

关闭连接

在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

关闭数据库连接可以直接调用其close方法即可。如：**conn.close()**

示例

```
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
//以下代码将获取数据库连接操作封装为一个接口，可通过给定用户名和密码来连接数据库。
```

 版权所有 © 华为云计算技术有限公司

```
public static Connection GetConnection(String username, String passwd) {
      //驱动类。
      String driver = "org.postgresql.Driver";
      //数据库连接描述符。
      String sourceURL = "jdbc:postgresql://10.10.0.13:8000/postgres?currentSchema=test";
      Connection conn = null;

      try {
         //加载驱动。
         Class.forName(driver);
      } catch (ClassNotFoundException e ){
         e.printStackTrace();
         return null;
      }

      try {
          //创建连接。
         conn = DriverManager.getConnection(sourceURL, username, passwd);
         System.out.println("Connection succeed!");
      } catch (SQLException e) {
         e.printStackTrace();
         return null;
      }

      return conn;
   }
```

12.2.5 执行 SQL 语句

执行普通 SQL 语句

应用程序通过执行SQL语句来操作数据库的数据（不用传递参数的语句），需要按以下步骤执行：

步骤1 调用Connection的createStatement方法创建语句对象。

```
Statement stmt = con.createStatement();
```

步骤2 调用Statement的executeUpdate方法执行SQL语句。

```
int rc = stmt.executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");
```

说明

数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，事务块中不支持vacuum操作。如果其中有一个语句失败，那么整个请求都将会被回滚。

步骤3 关闭语句对象。

```
stmt.close();
```

----结束

执行预编译 SQL 语句

预编译语句是只编译和优化一次，然后可以通过设置不同的参数值多次使用。由于已经预先编译好，后续使用会减少执行时间。因此，如果多次执行一条语句，请选择使用预编译语句。可以按以下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("UPDATE customer_t1 SET c_customer_name = ?
WHERE c_customer_sk = 1");
```

步骤2 调用PreparedStatement的setShort设置参数。

版权所有 © 华为云计算技术有限公司

```
pstmt.setShort(1, (short)2);
```

步骤3 调用PreparedStatement的executeUpdate方法执行预编译SQL语句。

```
int rowcount = pstmt.executeUpdate();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

----结束

调用存储过程

DWS支持通过JDBC直接调用事先创建的存储过程，步骤如下：

步骤1 调用Connection的prepareCall方法创建调用语句对象。

```
CallableStatement cstmt = myConn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
```

步骤2 调用CallableStatement的setInt方法设置参数。

```
cstmt.setInt(2, 50);
cstmt.setInt(1, 20);
cstmt.setInt(3, 90);
```

步骤3 调用CallableStatement的registerOutParameter方法注册输出参数。

```
cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
```

步骤4 调用CallableStatement的execute执行方法调用。

```
cstmt.execute();
```

步骤5 调用CallableStatement的getInt方法获取输出参数。

```
int out = cstmt.getInt(4);  //获取out参数
```

示例：

```
//在数据库中已创建了如下存储过程，它带有out参数。
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
```

步骤6 调用CallableStatement的close方法关闭调用语句。

```
cstmt.close();
```

说明

- 很多的数据库例如Connection、Statement和ResultSet都有close()方法，在使用完对象后应把它们关闭。要注意的是，Connection的关闭将间接关闭所有与它关联的Statement，Statement的关闭间接关闭了ResultSet。
- 一些JDBC驱动程序还提供命名参数的方法来设置参数。命名参数的方法允许根据名称而不是顺序来设置参数，若参数有默认值，则可以不用指定参数值就可以使用此参数的默认值。即使存储过程中参数的顺序发生了变更，也不必修改应用程序。目前DWS数据库的JDBC驱动程序不支持此方法。
- DWS数据库不支持带有输出参数的函数，也不支持存储过程和函数参数默认值。

----结束

 版权所有 © 华为云计算技术有限公司

须知

- 当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。
- 存储过程不能和普通SQL在同一条语句中执行。

执行批处理

用一条预处理语句处理多条相似的数据，数据库只创建一次执行计划，节省了语句的编译和优化时间。可以按如下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("INSERT INTO customer_t1 VALUES (?)");
```

步骤2 针对每条数据都要调用setShort设置参数，以及调用addBatch确认该条设置完毕。

```
pstmt.setShort(1, (short)2);
pstmt.addBatch();
```

步骤3 调用PreparedStatement的executeBatch方法执行批处理。

```
int[] rowcount = pstmt.executeBatch();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

说明

在实际的批处理过程中，通常不终止批处理程序的执行，否则会降低数据库的性能。因此在批处理程序时，应该关闭自动提交功能，每几行提交一次。关闭自动提交功能的语句为：
conn.setAutoCommit(false);

----结束

12.2.6 处理结果集

设置结果集类型

不同类型的结果集有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。在执行SQL语句过程中，都需要先创建相应的语句对象，而部分创建语句对象的方法提供了设置结果集类型的功能。具体的参数设置如表12-3所示。涉及的Connection的方法如下：

```
//创建一个Statement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
createStatement(int resultSetType, int resultSetConcurrency);

//创建一个PreparedStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareStatement(String sql, int resultSetType, int resultSetConcurrency);

//创建一个CallableStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareCall(String sql, int resultSetType, int resultSetConcurrency);
```

 版权所有 © 华为云计算技术有限公司

表 12-3 结果集类型

参数	描述
resultSetType	表示结果集的类型，具体有三种类型： • ResultSet.TYPE_FORWARD_ONLY：ResultSet只能向前移动。是缺省值。 • ResultSet.TYPE_SCROLL_SENSITIVE：在修改后重新滚动到修改所在行，可以查看修改后的结果。 • ResultSet.TYPE_SCROLL_INSENSITIVE：结果集可以前后滚动，但在读取数据期间，其他操作对数据进行的修改，该结果集无法感知。 **说明** 结果集从数据库中读取了数据之后，即使类型是ResultSet.TYPE_SCROLL_SENSITIVE，也不会看到由其他事务在这之后引起的改变。调用ResultSet的refreshRow()方法，可进入数据库并从其中取得当前游标所指记录的最新数据。
resultSetConcurrency	表示结果集的并发，具体有两种类型： • ResultSet.CONCUR_READ_ONLY：如果不从结果集中的数据建立一个新的更新语句，不能对结果集中的数据进行更新。 • ResultSet.CONCUR_UPDATABLE：可改变的结果集。对于可滚动的结果集，可对结果集进行适当的改变。

在结果集中定位

ResultSet对象具有指向其当前数据行的光标。最初，光标被置于第一行之前。next方法将光标移动到下一行；因为该方法在ResultSet对象没有下一行时返回false，所以可以在while循环中使用它来迭代结果集。但对于可滚动的结果集，JDBC驱动程序提供更多的定位方法，使ResultSet指向特定的行。定位方法如表12-4所示。

表 12-4 在结果集中定位的方法

方法	描述
next()	把ResultSet向下移动一行。
previous()	把ResultSet向上移动一行。
beforeFirst()	把ResultSet定位到第一行之前。
afterLast()	把ResultSet定位到最后一行之后。
first()	把ResultSet定位到第一行。
last()	把ResultSet定位到最后一行。
absolute(int)	把ResultSet移动到参数指定的行数。
relative(int)	向前或者向后移动参数指定的行。

版权所有 © 华为云计算技术有限公司

获取结果集中光标的位置

对于可滚动的结果集，可能会调用定位方法来改变光标的位置。JDBC驱动程序提供了获取结果集中光标所处位置的方法。获取光标位置的方法如表12-5所示。

表 12-5 获取结果集光标的位置

方法	描述
isFirst()	是否在一行。
isLast()	是否在最后一行。
isBeforeFirst()	是否在第一行之前。
isAfterLast()	是否在最后一行之后。
getRow()	获取当前在第几行。

获取结果集中的数据

ResultSet对象提供了丰富的方法，以获取结果集中的数据。获取数据常用的方法如表12-6所示，其他方法请参考JDK官方文档。

表 12-6 ResultSet 对象的常用方法

方法	描述
int getInt(int columnIndex)	按列标获取int型数据。
int getInt(String columnLabel)	按列名获取int型数据。
String getString(int columnIndex)	按列标获取String型数据。
String getString(String columnLabel)	按列名获取String型数据。
Date getDate(int columnIndex)	按列标获取Date型数据
Date getDate(String columnLabel)	按列名获取Date型数据。

12.2.7 常用 JDBC 开发示例

示例 1

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
```


版权所有 © 华为云计算技术有限公司

```
end;
/
```

此示例将演示如何基于DWS提供的JDBC接口开发应用程序。

```
//DBtest.java
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
//演示基于JDBC开发的主要步骤，会涉及创建数据库、创建表、插入数据等。

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.CallableStatement;

public class DBTest {

 //创建数据库连接。
 public static Connection GetConnection(String username, String passwd) {
   String driver = "org.postgresql.Driver";
   String sourceURL = "jdbc:postgresql://localhost:8000/gaussdb";
   Connection conn = null;
   try {
     //加载数据库驱动。
     Class.forName(driver).newInstance();
   } catch (Exception e) {
     e.printStackTrace();
     return null;
   }

   try {
     //创建数据库连接。
     conn = DriverManager.getConnection(sourceURL, username, passwd);
     System.out.println("Connection succeed!");
   } catch (Exception e) {
     e.printStackTrace();
     return null;
   }

   return conn;
 };

 //执行普通SQL语句，创建customer_t1表。
 public static void CreateTable(Connection conn) {
   Statement stmt = null;
   try {
     stmt = conn.createStatement();

     //执行普通SQL语句。
     int rc = stmt
         .executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");

     stmt.close();
   } catch (SQLException e) {
     if (stmt != null) {
       try {
         stmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
   }
 }

 //执行预处理语句，批量插入数据。
 public static void BatchInsertData(Connection conn) {
```


版权所有 © 华为云计算技术有限公司

```
  PreparedStatement pst = null;

  try {
    //生成预处理语句。
    pst = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?,?)");
    for (int i = 0; i < 3; i++) {
      //添加参数。
      pst.setInt(1, i);
      pst.setString(2, "data " + i);
      pst.addBatch();
    }
    //执行批处理。
    pst.executeBatch();
    pst.close();
  } catch (SQLException e) {
    if (pst != null) {
      try {
        pst.close();
      } catch (SQLException e1) {
      e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
 }

 //执行预编译语句，更新数据。
 public static void ExecPreparedSQL(Connection conn) {
   PreparedStatement pstmt = null;
   try {
     pstmt = conn
         .prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
     pstmt.setString(1, "new Data");
     int rowcount = pstmt.executeUpdate();
     pstmt.close();
   } catch (SQLException e) {
     if (pstmt != null) {
       try {
         pstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
   }
 }


//执行存储过程。
 public static void ExecCallableSQL(Connection conn) {
   CallableStatement cstmt = null;
   try {

     cstmt=conn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
     cstmt.setInt(2, 50);
     cstmt.setInt(1, 20);
     cstmt.setInt(3, 90);
     cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
     cstmt.execute();
     int out = cstmt.getInt(4);  //获取out参数
     System.out.println("The CallableStatment TESTPROC returns:"+out);
     cstmt.close();
   } catch (SQLException e) {
     if (cstmt != null) {
       try {
         cstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
```

 版权所有 © 华为云计算技术有限公司

```
    }
    e.printStackTrace();
  }
 }


 /**
  * 主程序，逐步调用各静态方法。
  * @param args
 */
 public static void main(String[] args) {
  //创建数据库连接。
  Connection conn = GetConnection("tester", "password");

  //创建表。
  CreateTable(conn);

  //批插数据。
  BatchInsertData(conn);

  //执行预编译语句，更新数据。
  ExecPreparedSQL(conn);

  //执行存储过程。
  ExecCallableSQL(conn);

  //关闭数据库连接。
  try {
   conn.close();
  } catch (SQLException e) {
   e.printStackTrace();
  }

 }

}
```

示例 2 客户端内存占用过多解决

此示例主要使用setFetchSize来调整客户端内存使用，它的原理是通过数据库游标来分批获取服务器端数据，但它会加大网络交互，可能会损失部分性能。

由于游标事务内有效，故需要先关闭自动提交。

```
// 关闭掉自动提交
conn.setAutoCommit(false);
Statement st = conn.createStatement();

// 打开游标，每次获取50行数据
st.setFetchSize(50);
ResultSet rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next()){
    System.out.print("a row was returned.");
}
rs.close();

// 关闭服务器游标。
st.setFetchSize(0);
rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next()){
    System.out.print("many rows were returned.");
}
rs.close();

// Close the statement.
st.close();
```

 版权所有 © 华为云计算技术有限公司

重新执行应用 SQL

当主DN故障且40s未恢复时，DWS会自动将对应的备DN升主，使集群正常运行。备升主期间正在运行的作业会失败；备升主后启动的作业不会再受影响。如果要做到DN主备切换过程中，上层业务不感知，可参考此示例构建业务层SQL重试机制。

以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 *
 */

class ExitHandler extends Thread {
    private Statement cancel_stmt = null;

    public ExitHandler(Statement stmt) {
        super("Exit Handler");
        this.cancel_stmt = stmt;
    }
    public void run() {
        System.out.println("exit handle");
        try {
            this.cancel_stmt.cancel();
        } catch (SQLException e) {
            System.out.println("cancel query failed.");
            e.printStackTrace();
        }
    }
}

public class SQLRetry {
   //创建数据库连接。
   public static Connection GetConnection(String username, String passwd) {
     String driver = "org.postgresql.Driver";
     String sourceURL = "jdbc:postgresql://10.131.72.136:8000/gaussdb";
     Connection conn = null;
     try {
       //加载数据库驱动。
       Class.forName(driver).newInstance();
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     try {
       //创建数据库连接。
       conn = DriverManager.getConnection(sourceURL, username, passwd);
       System.out.println("Connection succeed!");
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     return conn;
}
```

执行普通SQL语句，创建jdbc_test1表。

版权所有 © 华为云计算技术有限公司

```
public static void CreateTable(Connection conn) {
    Statement stmt = null;
    try {
      stmt = conn.createStatement();

      // add ctrl+c handler
      Runtime.getRuntime().addShutdownHook(new ExitHandler(stmt));

      //执行普通SQL语句。
      int rc2 = stmt
         .executeUpdate("DROP TABLE if exists jdbc_test1;");

      int rc1 = stmt
         .executeUpdate("CREATE TABLE jdbc_test1(col1 INTEGER, col2 VARCHAR(10));");

      stmt.close();
    } catch (SQLException e) {
      if (stmt != null) {
        try {
          stmt.close();
        } catch (SQLException e1) {
          e1.printStackTrace();
        }
      }
      e.printStackTrace();
    }
  }
```

执行预处理语句，批量插入数据。

```
public static void BatchInsertData(Connection conn) {
    PreparedStatement pst = null;

    try {
      //生成预处理语句。
      pst = conn.prepareStatement("INSERT INTO jdbc_test1 VALUES (?,?)");
      for (int i = 0; i < 100; i++) {
       //添加参数。
        pst.setInt(1, i);
        pst.setString(2, "data " + i);
        pst.addBatch();
      }
      //执行批处理。
      pst.executeBatch();
      pst.close();
    } catch (SQLException e) {
      if (pst != null) {
        try {
          pst.close();
        } catch (SQLException e1) {
        e1.printStackTrace();
        }
      }
      e.printStackTrace();
    }
  }
```

执行预编译语句，更新数据。

```
 private static boolean QueryRedo(Connection conn){
    PreparedStatement pstmt = null;
    boolean retValue = false;
    try {
      pstmt = conn
          .prepareStatement("SELECT col1 FROM jdbc_test1 WHERE col2 = ?");

          pstmt.setString(1, "data 10");
          ResultSet rs = pstmt.executeQuery();

          while (rs.next()) {
```


版权所有 © 华为云计算技术有限公司

```
            System.out.println("col1 = " + rs.getString("col1"));
        }
        rs.close();

      pstmt.close();
       retValue = true;
     } catch (SQLException e) {
      System.out.println("catch..... retValue " + retValue);
      if (pstmt != null) {
        try {
         pstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
      }
      e.printStackTrace();
    }

     System.out.println("finish.....");
    return retValue;
  }
```

查询语句，执行失败重试，重试次数可配置。

```
public static void ExecPreparedSQL(Connection conn) throws InterruptedException {
        int maxRetryTime = 50;
        int time = 0;
        String result = null;
        do {
            time++;
            try {
 System.out.println("time:" + time);
 boolean ret = QueryRedo(conn);
 if(ret == false){
  System.out.println("retry, time:" + time);
  Thread.sleep(10000);
  QueryRedo(conn);
 }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } while (null == result && time < maxRetryTime);

 }

 /**
  * 主程序，逐步调用各静态方法。
  * @param args
 * @throws InterruptedException
 */
 public static void main(String[] args) throws InterruptedException {
   //创建数据库连接。
   Connection conn = GetConnection("testuser", "test@123");

   //创建表。
   CreateTable(conn);

   //批插数据。
   BatchInsertData(conn);

   //执行预编译语句，更新数据。
   ExecPreparedSQL(conn);

   //关闭数据库连接。
   try {
     conn.close();
   } catch (SQLException e) {
     e.printStackTrace();
   }
```

 版权所有 © 华为云计算技术有限公司

```
  }

}
```

通过本地文件导入导出数据

在使用Java语言基于DWS进行二次开发时，可以使用CopyManager接口，通过流方式，将数据库中的数据导出到本地文件或者将本地文件导入数据库中，文件格式支持CSV、TEXT等格式。

样例程序如下，执行时需要加载DWS jdbc驱动。

以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.SQLException;
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Copy{

    public static void main(String[] args)
    {
     String urls = new String("jdbc:postgresql://10.180.155.74:8000/gaussdb"); //数据库URL
     String username = new String("jack");            //用户名
     String password = new String("********");        //密码
     String tablename = new String("migration_table"); //定义表信息
     String tablename1 = new String("migration_table_1"); //定义表信息
     String driver = "org.postgresql.Driver";
     Connection conn = null;

     try {
          Class.forName(driver);
          conn = DriverManager.getConnection(urls, username, password);
        } catch (ClassNotFoundException e) {
            e.printStackTrace(System.out);
        } catch (SQLException e) {
            e.printStackTrace(System.out);
        }
```

执行数据导入导出：

```
    // 将migration_table查询结果导出到本地文件d:/data.txt
     try {
    copyToFile(conn, "d:/data.txt", "(SELECT * FROM migration_table)");
  } catch (SQLException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  }
     //将d:/data.txt中的数据导入到migration_table_1中。
     try {
    copyFromFile(conn, "d:/data.txt", migration_table_1);
  } catch (SQLException e) {
 // TODO Auto-generated catch block
        e.printStackTrace();
} catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
```


版权所有 © 华为云计算技术有限公司

```
}

    // 将migration_table_1中的数据导出到本地文件d:/data1.txt
    try {
    copyToFile(conn, "d:/data1.txt", migration_table_1);
 } catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
 } catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
   }

public static void copyFromFile(Connection connection, String filePath, String tableName)
        throws SQLException, IOException {

   FileInputStream fileInputStream = null;

   try {
      CopyManager copyManager = new CopyManager((BaseConnection)connection);
      fileInputStream = new FileInputStream(filePath);
      copyManager.copyIn("COPY " + tableName + " FROM STDIN", fileInputStream);
   } finally {
      if (fileInputStream != null) {
         try {
            fileInputStream.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
}

public static void copyToFile(Connection connection, String filePath, String tableOrQuery)
        throws SQLException, IOException {

    FileOutputStream fileOutputStream = null;

    try {
       CopyManager copyManager = new CopyManager((BaseConnection)connection);
       fileOutputStream = new FileOutputStream(filePath);
       copyManager.copyOut("COPY " + tableOrQuery + " TO STDOUT", fileOutputStream);
    } finally {
       if (fileOutputStream != null) {
          try {
             fileOutputStream.close();
          } catch (IOException e) {
             e.printStackTrace();
          }
       }
    }
 }
}
```

从 MySQL 向 DWS 进行数据迁移

下面示例演示如何通过CopyManager从mysql向DWS进行数据迁移的过程。

以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。

```
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```


版权所有 © 华为云计算技术有限公司

```
import java.sql.SQLException;
import java.sql.Statement;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Migration{

    public static void main(String[] args) {
        String url = new String("jdbc:postgresql://10.180.155.74:8000/gaussdb"); //数据库URL
        String user = new String("jack");            //DWS用户名
        String pass = new String("********");            //DWS密码
        String tablename = new String("migration_table"); //定义表信息
        String delimiter = new String("|");              //定义分隔符
        String encoding = new String("UTF8");            //定义字符集
        String driver = "org.postgresql.Driver";
        StringBuffer buffer = new StringBuffer();      //定义存放格式  化数据的缓存

        try {
            //获取源数据库查询结果集
            ResultSet rs = getDataSet();

            //遍历结果集，逐行获取记录
            //将每条记录中各字段值，按指定分隔符分割，由换行符结束，拼成一个字符串
            //把拼成的字符串，添加到缓存buffer
            while (rs.next()) {
                buffer.append(rs.getString(1) + delimiter
                        + rs.getString(2) + delimiter
                        + rs.getString(3) + delimiter
                        + rs.getString(4)
                        + "\n");
            }
            rs.close();

            try {
                //建立目标数据库连接
                Class.forName(driver);
                Connection conn = DriverManager.getConnection(url, user, pass);
                BaseConnection baseConn = (BaseConnection) conn;
                baseConn.setAutoCommit(false);

                //初始化表信息
                String sql = "Copy " + tablename + " from STDIN DELIMITER " + "'" + delimiter + "'" + "
ENCODING " + "'" + encoding + "'";

                //提交缓存buffer中的数据
                CopyManager cp = new CopyManager(baseConn);
                StringReader reader = new StringReader(buffer.toString());
                cp.copyIn(sql, reader);
                baseConn.commit();
                reader.close();
                baseConn.close();
            } catch (ClassNotFoundException e) {
                e.printStackTrace(System.out);
            } catch (SQLException e) {
                e.printStackTrace(System.out);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

从源数据库返回查询结果集：

```
    private static ResultSet getDataSet() {
        ResultSet rs = null;
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
```

 版权所有 © 华为云计算技术有限公司

```
            Connection conn = DriverManager.getConnection("jdbc:mysql://10.119.179.227:3306/jack?
useSSL=false&allowPublicKeyRetrieval=true", "jack", "********");
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery("select * from migration_table");
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rs;
    }
}
```

12.2.8 应用端加工 RoaringBitmap 结果集并入库 DWS 开发示例

DWS 自8.1.3版本后支持位图功能（RoaringBitmap），在使用Java语言基于DWS进行二次开发时，可以使用CopyManager接口，实现小批量RoaringBitmap的数据入库至DWS。

说明

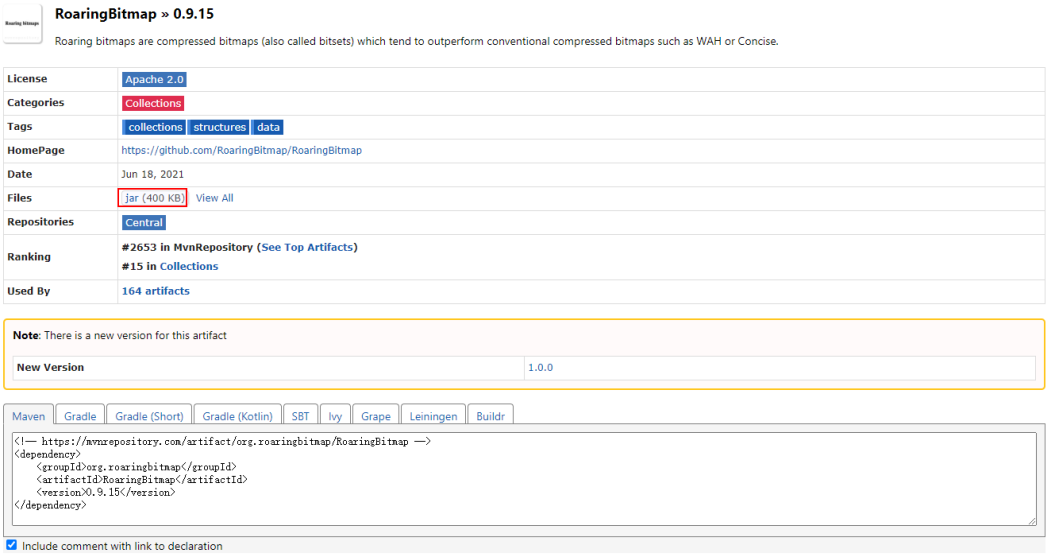
- 如针对大批量数据入库，需要在应用侧增加算力进行优化，否则会影响入库性能。
- Java生成的64位Roaringbitmap不支持入库。

加工 RoaringBitmap

步骤1 访问Maven下载开源的RoaringBitmap的jar包，推荐下载0.9.15版本。

POM文件依赖项配置如下：

```
<dependencies>
 <dependency>
 <groupId>org.roaringbitmap</groupId>
 <artifactId>RoaringBitmap</artifactId>
 <version>0.9.15</version>
 </dependency>
 </dependencies>
```



步骤2 调用jar包实现Roaringbitmap的转换。

大致过程是声明一个RoaringBitmap，调用add方法，将要转化的int转化成Roaringbitmap类型，再对数据进行序列化。代码示例如下：

版权所有 © 华为云计算技术有限公司

```
RoaringBitmap rr2 = new RoaringBitmap ();
for (int i = 1; i < 10000000;  i++)  {
     rr2.add(i);
}
ByteArrayOutputStream a = new ByteArrayOutputStream();
DataOutputStream b = new DataOutputStream(a);
rr2.serialize(b);
```

----结束

数据入库

调用CopyManager进行入库，实现数据不落地，小批量RoaringBitmap入库。

```
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。

package rb_demo;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;
import org.roaringbitmap.RoaringBitmap;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class rb_demo {

   private static String hexStr =  "0123456789ABCDEF";

   public static String bytesToHex(byte[] bytes) {
      StringBuffer sb = new StringBuffer();
      for (int i = 0; i < bytes.length; i++) {
         String hex = Integer.toHexString(bytes[i] & 0xFF);
         if (hex.length() < 2) {
            sb.append(0);
         }
         sb.append(hex);
      }
      return sb.toString();
   }

   public static Connection GetConnection(String username, String passwd) {
      String driver = "org.postgresql.Driver";
      String sourceURL = "jdbc:postgresql://10.185.180.161:8000/gaussdb";  //数据库URL
      Connection conn = null;
      try {
         // 加载数据库驱动。
         Class.forName(driver).newInstance();
      } catch (Exception e) {
         e.printStackTrace();
         return null;
      }

      try {
         // 创建数据库连接。
         conn = DriverManager.getConnection(sourceURL, username, passwd);
         System.out.println("Connection succeed!");
      } catch (Exception e) {
```

 版权所有 © 华为云计算技术有限公司

```
            e.printStackTrace();
            return null;
        }

        return conn;
    }

    public static void main(String[] args) throws IOException {

        RoaringBitmap rr2 = new RoaringBitmap();

        for (int i = 1; i < 10000000; i++) {
            rr2.add(i);
        }

        ByteArrayOutputStream a = new ByteArrayOutputStream();

        DataOutputStream b = new DataOutputStream(a);
        rr2.serialize(b);

        Connection conn = GetConnection("test", "password");  //数据库用户名、密码
        Statement pstmt = null;
        try {
            conn.setAutoCommit(true);
            pstmt = conn.createStatement();

            pstmt.execute("drop table if exists t_rb");
            pstmt.execute("create table t_rb(c1 int, c2 roaringbitmap) distribute by hash (c1);");

            StringReader sr = null;
            CopyManager cm = null;
            cm = new CopyManager((BaseConnection) conn);

            String delimiter = "|";
            StringBuffer tuples = new StringBuffer();
            tuples.append("1" + delimiter + "\\x" + bytesToHex(a.toByteArray()));


            StringBuffer sb = new StringBuffer();
            sb.append(tuples.toString());

            sr = new StringReader(tuples.toString());
            String sql = "copy t_rb from STDIN with (delimiter '|', NOESCAPING)";

            long rows = cm.copyIn(sql, sr);// 执行copy入库

            pstmt.close();
        } catch (SQLException e) {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }
}
```

12.2.9 JDBC 接口参考

JDBC接口是一套提供给用户的API方法，本节将对部分常用接口做具体描述，若涉及其他接口可参考JDK1.6（软件包）/JDBC4.0中相关内容。

java.sql.Connection

java.sql.Connection是数据库连接接口。

版权所有 © 华为云计算技术有限公司

表 12-7 对 java.sql.Connection 接口的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes
commit()	void	Yes
createStatement()	Statement	Yes
getAutoCommit()	boolean	Yes
getClientInfo()	Properties	Yes
getClientInfo(String name)	String	Yes
getTransactionIsolation()	int	Yes
isClosed()	boolean	Yes
isReadOnly()	boolean	Yes
prepareStatement(String sql)	PreparedStatement	Yes
rollback()	void	Yes
setAutoCommit(boolean autoCommit)	void	Yes
setClientInfo(Properties properties)	void	Yes
setClientInfo(String name,String value)	void	Yes

须知

接口内部默认使用自动提交模式，若通过setAutoCommit(false)关闭自动提交，将会导致后面执行的语句都受到显式事务包裹，数据库中不支持事务中执行的语句不能在此模式下执行。

java.sql.CallableStatement

java.sql.CallableStatement是存储过程执行接口。

表 12-8 对 java.sql.CallableStatement 的支持情况

方法名	返回值类型	支持JDBC 4
registerOutParameter(int parameterIndex, int type)	void	Yes
wasNull()	boolean	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getString(int parameterIndex)	String	Yes
getBoolean(int parameterIndex)	boolean	Yes
getByte(int parameterIndex)	byte	Yes
getShort(int parameterIndex)	short	Yes
getInt(int parameterIndex)	int	Yes
getLong(int parameterIndex)	long	Yes
getFloat(int parameterIndex)	float	Yes
getDouble(int parameterIndex)	double	Yes
getBigDecimal(int parameterIndex)	BigDecimal	Yes
getBytes(int parameterIndex)	byte[]	Yes
getDate(int parameterIndex)	Date	Yes
getTime(int parameterIndex)	Time	Yes
getTimestamp(int parameterIndex)	Timestamp	Yes
getObject(int parameterIndex)	Object	Yes

说明

- 不允许含有OUT参数的语句执行批量操作。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows , setFetchSize。
- 以下方法是从java.sql.PreparedStatement继承而来：addBatch，clearParameters，execute，executeQuery，executeUpdate，getMetaData，setBigDecimal，setBoolean，setByte，setBytes，setDate，setDouble，setFloat，setInt，setLong，setNull，setObject，setString，setTime，setTimestamp 。

java.sql.DatabaseMetaData

java.sql.DatabaseMetaData是数据库对象定义接口。

版权所有 © 华为云计算技术有限公司

表 12-9 对 java.sql.DatabaseMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	ResultSet	Yes
getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	ResultSet	Yes
getTableTypes()	ResultSet	Yes
getUserName()	String	Yes
isReadOnly()	boolean	Yes
nullsAreSortedHigh()	boolean	Yes
nullsAreSortedLow()	boolean	Yes
nullsAreSortedAtStart()	boolean	Yes
nullsAreSortedAtEnd()	boolean	Yes
getDatabaseProductName()	String	Yes
getDatabaseProductVersion()	String	Yes
getDriverName()	String	Yes
getDriverVersion()	String	Yes
getDriverMajorVersion()	int	Yes
getDriverMinorVersion()	int	Yes
usesLocalFiles()	boolean	Yes
usesLocalFilePerTable()	boolean	Yes
supportsMixedCaseIdentifiers()	boolean	Yes
storesUpperCaseIdentifiers()	boolean	Yes
storesLowerCaseIdentifiers()	boolean	Yes
supportsMixedCaseQuotedIdentifiers()	boolean	Yes
storesUpperCaseQuotedIdentifiers()	boolean	Yes
storesLowerCaseQuotedIdentifiers()	boolean	Yes

 版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
storesMixedCaseQuotedIdentifiers()	boolean	Yes
supportsAlterTableWithAddColumn()	boolean	Yes
supportsAlterTableWithDropColumn()	boolean	Yes
supportsColumnAliasing()	boolean	Yes
nullPlusNonNullIsNull()	boolean	Yes
supportsConvert()	boolean	Yes
supportsConvert(int fromType, int toType)	boolean	Yes
supportsTableCorrelationNames()	boolean	Yes
supportsDifferentTableCorrelationNames()	boolean	Yes
supportsExpressionsInOrderBy()	boolean	Yes
supportsOrderByUnrelated()	boolean	Yes
supportsGroupBy()	boolean	Yes
supportsGroupByUnrelated()	boolean	Yes
supportsGroupByBeyondSelect()	boolean	Yes
supportsLikeEscapeClause()	boolean	Yes
supportsMultipleResultSets()	boolean	Yes
supportsMultipleTransactions()	boolean	Yes
supportsNonNullableColumns()	boolean	Yes
supportsMinimumSQLGrammar()	boolean	Yes
supportsCoreSQLGrammar()	boolean	Yes
supportsExtendedSQLGrammar()	boolean	Yes
supportsANSI92EntryLevelSQL()	boolean	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
supportsANSI92IntermediateSQL()	boolean	Yes
supportsANSI92FullSQL()	boolean	Yes
supportsIntegrityEnhancementFacility()	boolean	Yes
supportsOuterJoins()	boolean	Yes
supportsFullOuterJoins()	boolean	Yes
supportsLimitedOuterJoins()	boolean	Yes
isCatalogAtStart()	boolean	Yes
supportsSchemasInDataManipulation()	boolean	Yes
supportsSavepoints()	boolean	Yes
supportsResultSetHoldability(int holdability)	boolean	Yes
getResultSetHoldability()	int	Yes
getDatabaseMajorVersion()	int	Yes
getDatabaseMinorVersion()	int	Yes
getJDBCMajorVersion()	int	Yes
getJDBCMinorVersion()	int	Yes

java.sql.Driver

java.sql.Driver是数据库驱动接口。

表 12-10 对 java.sql.Driver 的支持情况

方法名	返回值类型	支持JDBC 4
acceptsURL(String url)	boolean	Yes
connect(String url, Properties info)	Connection	Yes
jdbcCompliant()	boolean	Yes
getMajorVersion()	int	Yes
getMinorVersion()	int	Yes

版权所有 © 华为云计算技术有限公司

java.sql.PreparedStatement

java.sql.PreparedStatement是预处理语句接口。

表 12-11 对 java.sql.PreparedStatement 的支持情况

方法名	返回值类型	支持JDBC 4
clearParameters()	void	Yes
execute()	boolean	Yes
executeQuery()	ResultSet	Yes
executeUpdate()	int	Yes
getMetaData()	ResultSetMetaData	Yes
setBoolean(int parameterIndex, boolean x)	void	Yes
setBigDecimal(int parameterIndex, BigDecimal x)	void	Yes
setByte(int parameterIndex, byte x)	void	Yes
setBytes(int parameterIndex, byte[] x)	void	Yes
setDate(int parameterIndex, Date x)	void	Yes
setDouble(int parameterIndex, double x)	void	Yes
setFloat(int parameterIndex, float x)	void	Yes
setInt(int parameterIndex, int x)	void	Yes
setLong(int parameterIndex, long x)	void	Yes
setNString(int parameterIndex, String value)	void	Yes
setShort(int parameterIndex, short x)	void	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
setString(int parameterIndex, String x)	void	Yes
addBatch()	void	Yes
executeBatch()	int[]	Yes
clearBatch()	void	Yes

说明

- addBatch()、execute()必须在clearBatch()之后才能执行。
- 调用executeBatch()方法并不会清除batch。用户必须显式使用clearBatch()清除 。
- 在添加了一个batch的绑定变量后，用户若想重用这些值(再次添加一个batch)，无需再次使用set*()方法 。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows, setFetchSize。

java.sql.ResultSet

java.sql.ResultSet是执行结果集接口。

表 12-12 对 java.sql.ResultSet 的支持情况

方法名	返回值类型	支持JDBC 4
findColumn(String columnLabel)	int	Yes
getBigDecimal(int columnIndex)	BigDecimal	Yes
getBigDecimal(String columnLabel)	BigDecimal	Yes
getBoolean(int columnIndex)	boolean	Yes
getBoolean(String columnLabel)	boolean	Yes
getByte(int columnIndex)	byte	Yes
getBytes(int columnIndex)	byte[]	Yes
getByte(String columnLabel)	byte	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
getBytes(String columnLabel)	byte[]	Yes
getDate(int columnIndex)	Date	Yes
getDate(String columnLabel)	Date	Yes
getDouble(int columnIndex)	double	Yes
getDouble(String columnLabel)	double	Yes
getFloat(int columnIndex)	float	Yes
getFloat(String columnLabel)	float	Yes
getInt(int columnIndex)	int	Yes
getInt(String columnLabel)	int	Yes
getLong(int columnIndex)	long	Yes
getLong(String columnLabel)	long	Yes
getShort(int columnIndex)	short	Yes
getShort(String columnLabel)	short	Yes
getString(int columnIndex)	String	Yes
getString(String columnLabel)	String	Yes
getTime(int columnIndex)	Time	Yes
getTime(String columnLabel)	Time	Yes
getTimestamp(int columnIndex)	Timestamp	Yes
getTimestamp(String columnLabel)	Timestamp	Yes
isAfterLast()	boolean	Yes
isBeforeFirst()	boolean	Yes
isFirst()	boolean	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
next()	boolean	Yes

说明

- 一个Statement不能有多个处于“open”状态的ResultSet。
- 用于遍历结果集(ResultSet)的游标(Cursor)在被提交后不能保持“open”的状态。

java.sql.ResultSetMetaData

java.sql.ResultSetMetaData是对ResultSet对象相关信息的具体描述。

表 12-13 对 java.sql.ResultSetMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getColumnCount()	int	Yes
getColumnName(int column)	String	Yes
getColumnType(int column)	int	Yes
getColumnTypeName(int column)	String	Yes

java.sql.Statement

java.sql.Statement是SQL语句接口。

表 12-14 对 java.sql.Statement 的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes
execute(String sql)	boolean	Yes
executeQuery(String sql)	ResultSet	Yes
executeUpdate(String sql)	int	Yes
getConnection()	Connection	Yes
getResultSet()	ResultSet	Yes
getQueryTimeout()	int	Yes
getUpdateCount()	int	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
isClosed()	boolean	Yes
setQueryTimeout(int seconds)	void	Yes
setFetchSize(int rows)	void	Yes
cancel()	void	Yes

说明

通过setFetchSize可以减少结果集在客户端的内存占用情况。它的原理是通过将结果集打包成游标，然后分段处理，所以会加大数据库与客户端的通信量，会有性能损耗。

由于数据库游标是事务内有效，所以，在设置setFetchSize的同时，需要将连接设置为非自动提交模式，setAutoCommit(false)。同时在业务数据需要持久化到数据库中时，在连接上执行提交操作。

javax.sql.ConnectionPoolDataSource

javax.sql.ConnectionPoolDataSource是数据源连接池接口。

表 12-15 对 javax.sql.ConnectionPoolDataSource 的支持情况

方法名	返回值类型	支持JDBC 4
getLoginTimeout()	int	Yes
getLogWriter()	PrintWriter	Yes
getPooledConnection()	PooledConnection	Yes
getPooledConnection(String user,String password)	PooledConnection	Yes
setLoginTimeout(int seconds)	void	Yes
setLogWriter(PrintWriter out)	void	Yes

javax.sql.DataSource

javax.sql.DataSource是数据源接口。

版权所有 © 华为云计算技术有限公司

表 12-16 对 javax.sql.DataSource 接口的支持情况

方法名	返回值类型	支持JDBC 4
getConnection()	Connection	Yes
getConnection(String username,String password)	Connection	Yes
getLoginTimeout()	int	Yes
getLogWriter()	PrintWriter	Yes
setLoginTimeout(int seconds)	void	Yes
setLogWriter(PrintWriter out)	void	Yes

javax.sql.PooledConnection

javax.sql.PooledConnection是由连接池创建的连接接口。

表 12-17 对 javax.sql.PooledConnection 的支持情况

方法名	返回值类型	支持JDBC 4
addConnectionEventListener (ConnectionEventListener listener)	void	Yes
close()	void	Yes
getConnection()	Connection	Yes
removeConnectionEventListener (ConnectionEventListener listener)	void	Yes
addStatementEventListener (StatementEventListener listener)	void	Yes
removeStatementEventListener (StatementEventListener listener)	void	Yes

javax.naming.Context

javax.naming.Context是连接配置的上下文接口。

表 12-18 对 javax.naming.Context 的支持情况

方法名	返回值类型	支持JDBC 4
bind(Name name, Object obj)	void	Yes

版权所有 © 华为云计算技术有限公司

方法名	返回值类型	支持JDBC 4
bind(String name, Object obj)	void	Yes
lookup(Name name)	Object	Yes
lookup(String name)	Object	Yes
rebind(Name name, Object obj)	void	Yes
rebind(String name, Object obj)	void	Yes
rename(Name oldName, Name newName)	void	Yes
rename(String oldName, String newName)	void	Yes
unbind(Name name)	void	Yes
unbind(String name)	void	Yes

javax.naming.spi.InitialContextFactory

javax.naming.spi.InitialContextFactory是初始连接上下文工厂接口。

表 12-19 对 javax.naming.spi.InitialContextFactory 的支持情况

方法名	返回值类型	支持JDBC 4
getInitialContext(Hashtable<?,?> environment)	Context	Yes

CopyManager

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

CopyManager的继承关系

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

```
public class CopyManager
extends Object
```

构造方法

```
public CopyManager(BaseConnection connection)
throws SQLException
```

常用方法

版权所有 © 华为云计算技术有限公司

表 12-20 CopyManager 常用方法

返回值	方法	描述	throws
CopyIn	copyIn(String sql)	-	SQLException
long	copyIn(String sql, InputStream from)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, InputStream from, int bufferSize)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from, int bufferSize)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
CopyOut	copyOut(String sql)	-	SQLException
long	copyOut(String sql, OutputStream to)	将一个COPY TO STDOUT的结果集从数据库发送到OutputStream类中。	SQLException,IOException
long	copyOut(String sql, Writer to)	将一个COPY TO STDOUT的结果集从数据库发送到Writer类中。	SQLException,IOException

12.3 基于 ODBC 开发

12.3.1 ODBC 包及依赖的库和头文件

从管理控制台下载ODBC的软件包。

请参见**下载JDBC或ODBC驱动**。

版权所有 © 华为云计算技术有限公司

Linux 下的 ODBC 包

从软件包中获取，包名为dws_9.1.x_odbc_driver_for_xxx_xxx.zip。Linux环境下，开发应用程序要用到unixODBC提供的头文件（sql.h、sqlext.h等）和库libodbc.so。这些头文件和库可从unixODBC-2.3.0的安装包中获得。

Windows 下的 ODBC 包

从软件包中获取，包名为dws_9.1.x_odbc_driver_for_windows.zip。Windows环境下，开发应用程序用到的相关头文件和库文件都由系统自带。

12.3.2 Linux 下配置数据源

将DWS提供的ODBC DRIVER（psqlodbcw.so）配置到数据源中便可使用。配置数据源需要配置“odbc.ini”和“odbcinst.ini”两个文件（在编译安装unixODBC过程中生成且默认放在etc目录下），并在服务器端进行配置。

操作步骤

步骤1 获取unixODBC源码包。

获取参考地址：**https://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download**

步骤2 目前不支持unixODBC-2.2.1版本。以unixODBC-2.3.0版本为例，在客户端执行如下命令安装unixODBC。安装时通过--prefix=[your_path]指定安装目录，生成数据源文件到“[your_path]/etc”目录下，库文件生成在“[your_path]/lib”目录。

```
tar zxvf unixODBC-2.3.0.tar.gz
cd unixODBC-2.3.0
#修改configure文件(如果不存在，那么请修改configure.ac)，找到LIB_VERSION
#将它的值修改为"1:0:0"，这样将编译出*.so.1的动态库，与psqlodbcw.so的依赖关系相同。
vim configure

./configure --enable-gui=no --prefix=[your_path] #如果要在ARM服务器上编译，请追加一个configure参数：--build=aarch64-unknown-linux-gnu
make
make install
```

安装unixODBC。如果机器上已经安装了其他版本的unixODBC，可以直接覆盖安装。

步骤3 替换客户端DWS驱动程序。

将dws_9.1.x_odbc_driver_for_xxx_xxx.zip解压，在“/dws_9.1.x_odbc_driver_for_xxx_xxx/odbc/lib”目录下得到“psqlodbcw.la”和“psqlodbcw.so”两个文件。

步骤4 配置数据源。

1. 配置ODBC驱动文件。

   在“[your_path]/etc/odbcinst.ini”文件中追加以下内容。

   ```
   [DRIVER_DWS]
   Driver64=[your_path]/lib/odbc/psqlodbcw.so
   setup=[your_path]/lib/odbc/psqlodbcw.so
   ```

   odbcinst.ini文件中的配置参数说明如**表12-21**所示。

 版权所有 © 华为云计算技术有限公司

表 12-21 odbcinst.ini 文件配置参数

参数	描述	示例
[DriverName]	驱动器名称，对应数据源DSN中的驱动名。	[DRIVER_DWS]
Driver64	驱动动态库的路径。	Driver64=/xxx/odbc/lib/odbc/psqlodbcw.so
setup	驱动安装路径，与Driver64中动态库的路径一致。	setup=/xxx/odbc/lib/odbc/psqlodbcw.so

2. 配置数据源文件。

   在“[your_path]/etc/odbc.ini ”文件中追加以下内容。

```
[DWSODBC]
Driver=DRIVER_DWS
Servername=10.10.0.13（数据库Server IP）
Database=gaussdb （数据库名）
Username=dbadmin （数据库用户名）
Password= （数据库用户密码）
Port=8000（数据库监听端口）
Sslmode=allow
```

   odbc.ini文件配置参数说明如表12-22所示。

表 12-22 odbc.ini 文件配置参数

参数	描述	示例
[DSN]	数据源的名称。	[DWSODBC]
Driver	驱动名，对应odbcinst.ini中的DriverName。	Driver=DRIVER_DWS
Servername	服务器的IP地址。	Servername=10.145.130.26
Database	要连接的数据库的名称。	Database=gaussdb
Username	数据库用户名称。	Username=dbadmin
Password	数据库用户密码。	Password= **说明** ODBC驱动本身已经对内存密码进行过清理，以保证用户密码在连接后不会再在内存中保留。 但是如果配置了此参数，由于UnixODBC对数据源文件等进行缓存，可能导致密码长期保留在内存中。 推荐在应用程序连接时，将密码传递给相应API，而非写在数据源配置文件中。同时连接成功后，应当及时清理保存密码的内存段。

 版权所有 © 华为云计算技术有限公司

参数	描述	示例
Port	服务器的端口号。	Port=8000
Sslmode	开启SSL模式。	Sslmode=allow
UseServerSidePrepare	是否开启数据库端扩展查询协议。 可选值0或1，默认为1，表示打开扩展查询协议。	UseServerSidePrepare=1
UseBatchProtocol	是否开启批量查询协议（打开可提高DML性能）；可选值0或者1，默认为1。 当此值为0时，不使用批量查询协议（主要用于与早期数据库版本通信兼容）。 当此值为1，并且数据库support_batch_bind参数存在且为on时，将打开批量查询协议。	UseBatchProtocol=1
ConnectionExtraInfo	GUC参数connection_info（参见**connection_info**）中显示驱动部署路径和进程属主用户的开关。	ConnectionExtraInfo=1 **说明** 默认值为1。当设置为0时，ODBC驱动会将当前驱动的名称、驱动版本上报到数据库中；当设置为1时，ODBC驱动会将当前驱动的名称、部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。
ForExtensionConnector	ETL工具性能优化参数，可进行内存优化，降低对端的CN内存占用，避免因CN内存使用过多导致系统不稳定。 可选值0或者1，默认为0，表示不开启优化项。 请勿在数据库系统之外的其他业务中配置此参数，以免影响业务的正确性。	ForExtensionConnector=1

版权所有 © 华为云计算技术有限公司

参数	描述	示例
KeepDisallowPremature	当UseDeclareFetch=1时，应用程序调用SQLPrepare后调用SQLNumResultCols、SQLDescribeCol或SQLColAttribute获取结果集列信息时，SQL语句中的游标是否具有with hold属性。 可选值0或者1，0表示具有with hold属性，1表示不具有with hold属性，默认为0。	KeepDisallowPremature=1 **说明** UseServerSidePrepare=1时，KeepDisallowPremature参数不生效，使用时需要指定UseServerSidePrepare为0，例如： UseDeclareFetch=1 KeepDisallowPremature=1 UseServerSidePrepare=0

其中关于sslmode的选项的允许值，具体信息见下表：

表 12-23 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机构签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。DWS不支持此模式。

步骤5 SSL模式

如果需要使用SSL证书连接，那么请将DWS安装包中的SSLCERT的证书包解压，在shell环境下，执行“source sslcert_env.sh”，即在**当前会话**完成证书的默认位置的部署。

或者手动声明如下环境变量，同时保证client.key*系列文件为600权限：

```
export PGSSLCERT="/YOUR/PATH/OF/client.crt" #请修改该路径到client.crt的绝对路径
export PGSSLKEY="/YOUR/PATH/OF/client.key"  #请修改该路径到client.key的绝对路径
```

同时将数据源中的Sslmode选项调整至“verify-ca”。

步骤6 将客户端所在主机的IP网段加入DWS的安全组规则，确保客户端主机与DWS网络互通。

 版权所有 © 华为云计算技术有限公司

步骤7 配置环境变量。

```
vim ~/.bashrc
```

在配置文件中追加以下内容。

```
export LD_LIBRARY_PATH=[your_path]/lib/:$LD_LIBRARY_PATH
export ODBCSYSINI=[your_path]/etc
export ODBCINI=[your_path]/etc/odbc.ini
```

说明

麒麟OS环境中，此处不建议追加LD_LIBRARY_PATH，可能造成libssl.so动态库冲突。集群9.1.0及以上最新版本中已增加rpath，不需要LD_LIBRARY_PATH即可找到依赖。

步骤8 执行如下命令使设置生效。

```
source ~/.bashrc
```

----结束

测试数据源配置

执行isql -v DWSODBC（数据源名称）命令。

- 如果显示如下信息，表明配置正确，连接成功。

```
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

- 如果显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

常见问题处理

- [UnixODBC][Driver Manager]Can't open lib 'xxx/xxx/psqlodbcw.so' : file not found.

  此问题的可能原因：

  - odbcinst.ini文件中配置的路径不正确

    确认的方法：执行**ls**查询错误信息中的路径，以确保该psqlodbcw.so文件存在，同时具有执行权限。

  - psqlodbcw.so的依赖库不存在，或者不在系统环境变量中

    确认的方法：执行**ldd**查询错误信息中的路径，如果是缺少libodbc.so.1等UnixODBC的库，那么按照“操作步骤”中的方法重新配置UnixODBC，并确保它的安装路径下的lib目录添加到了LD_LIBRARY_PATH中；如果是缺少其他库，请将ODBC驱动包中的lib目录添加到LD_LIBRARY_PATH中。或者将psqlodbcw.so的依赖库放到psqlodbcw.so的rpath对应的路径中，通过readelf -d可查看rpath。

- [UnixODBC]connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

版权所有 © 华为云计算技术有限公司

如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

- 防火墙及网闸设备

  请确认防火墙设置，将数据库的通信端口添加到可信端口中。

  如果有网闸设备，请确认相关的设置。

- [unixODBC]The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项。

  解决方法：

  请配置该选项至allow或以上选项。此配置的更多信息，见**表12-23**。

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因：

  使用了SSL加密的“verify-full”选项，驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。

  解决方法：

  碰到此问题可以使用“verify-ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- Driver's SQLAllocHandle on SQL_HANDLE_DBC failed

  此问题的可能原因：

  可执行文件（比如UnixODBC的isql，以下都以isql为例）与数据库驱动（psqlodbcw.so）依赖于不同的odbc的库版本：libodbc.so.1或者libodbc.so.2。此问题可以通过如下方式确认：

  ```
  ldd `which isql` | grep odbc
  ldd psqlodbcw.so | grep odbc
  ```

  这时，如果输出的libodbc.so最后的后缀数字不同或者指向不同的磁盘物理文件，那么基本就可以断定是此问题。isql与psqlodbcw.so都会要求加载libodbc.so，这时如果它们加载的是不同的物理文件，便会导致两套完全同名的函数列表，同时出现在同一个可见域里（UnixODBC的libodbc.so.*的函数导出列表完全一致），产生冲突，无法加载数据库驱动。

  解决方法：

  确定一个要使用的UnixODBC，然后卸载另外一个（比如卸载库版本号为.so.2的UnixODBC），然后将剩下的.so.1的库，新建一个同名但是后缀为.so.2的软链接，便可解决此问题。

- FATAL: Forbid remote connection with trust method!

  由于安全原因，数据库CN禁止集群内部其他节点无认证接入。

  如果要在集群内部访问CN，请将ODBC程序部署在CN所在机器，服务器地址使用"127.0.0.1"。建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

 版权所有 © 华为云计算技术有限公司

说明

- 数据库并不存储用户口令，只存储用户口令的哈希码。
- 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
- 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
- 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

要解决该问题，可以更新用户口令；或者新建一个用户，赋予同等权限，使用新用户连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

12.3.3 Windows 下配置数据源

Windows操作系统自带ODBC数据源管理器，无需用户手动安装管理器便可直接进行配置。

操作步骤

步骤1 替换客户端DWS驱动程序。

将GaussDB-9.1.2-Windows-Odbc.tar.gz解压后，根据需要，双击psqlodbc.msi（32位）或者psqlodbc_x64.msi（64 位）进行驱动安装。

步骤2 打开驱动管理器。

在配置数据源时，请使用对应的驱动管理器（假设操作系统安装盘符为C:盘，如果是其他盘符，请对路径做相应修改）：

- **64**位操作系统上进行**32**位程序开发，安装**32**位驱动程序后，使用**32**位的驱动管理器：C:\Windows**SysWOW64**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

  说明

  WoW64的全称是"Windows 32-bit on Windows 64-bit"，C:\Windows\SysWOW64\存放的是64位系统上的32位运行环境。

- **64**位操作系统上进行**64**位程序开发，安装**64**位驱动程序后，使用**64**位的驱动管理器：C:\Windows**System32**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

  说明

  C:\Windows\System32\存放的是与操作系统一致的运行环境，具体的技术信息请查阅Windows的相关技术文档。

- 32位操作系统请使用：C:\Windows\System32\odbcad32.exe

  或者单击计算机->控制面板->管理工具->数据源(ODBC)打开驱动管理器。

步骤3 配置数据源。

 版权所有 © 华为云计算技术有限公司

在打开的驱动管理器上，选择用户DSN->添加->PostgreSQL Unicode（如果是64位驱动，将会有64位标识），然后进行配置：

须知

此界面上配置的用户名及密码信息，将会被记录在Windows注册表中，再次连接数据库时就不再需要输入认证信息。但是出于安全考虑，建议在单击“Save”按钮保存配置信息前，清空相关敏感信息；在使用ODBC的连接API时，再传入所需的用户名、密码信息。

步骤4 SSL模式。

如果需要使用SSL证书连接，那么请将DWS安装包中的SSLCERT的证书包解压，双击“sslcert_env.bat”文件，即可完成证书的默认位置的部署。

须知

该sslcert_env.bat为了保证证书环境的纯净，在%APPDATA%\postgresql目录存在时，会提示是否需要移除相关目录。如果有需要，请备份该目录中的文件。

或者手动将client.crt、client.key、client.key.cipher、client.key.rand文件放至%APPDATA%\postgresql(该目录需手动建立)目录下，并且将文件名中的client改为postgres，例如client.key修改为postgres.key；将cacert.pem文件放至%APPDATA%\postgresql目录，并更名为root.crt。

同时将步骤2中的设置窗口的“SSL Mode”选项调整至“verify-ca”。

表 12-24 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。

 版权所有 © 华为云计算技术有限公司

sslmode	是否会启用SSL加密	描述
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机构签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。 **说明** 不支持此模式。

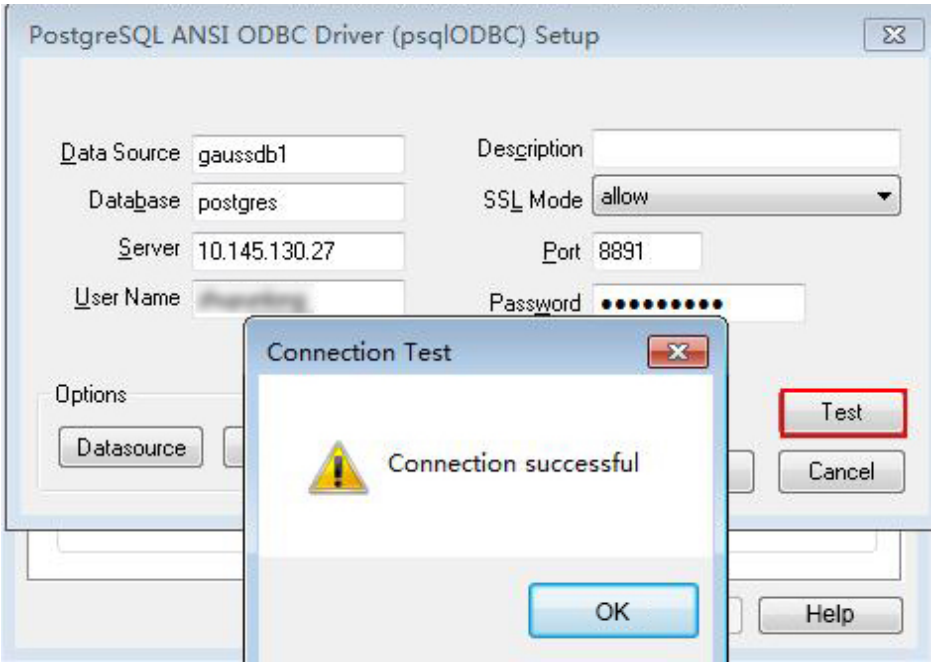
步骤5 将客户端所在主机的IP网段加入DWS的安全组规则，确保客户端主机与DWS网络互通。

----结束

测试数据源配置

单击Test进行测试。

- 如果显示如下，则表明配置正确，连接成功。



- 如果显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

 版权所有 © 华为云计算技术有限公司

常见问题处理

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因是使用了SSL加密的“verify-full”选项，这时驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。碰到此问题可以使用“verify-ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

    如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

  - 防火墙及网闸设备

    请确认防火墙设置，将数据库的通信端口添加到可信端口中。

    如果有网闸设备，请确认相关的设置。

- 在指定的DSN中，驱动程序和应用程序之间的体系结构不匹配

  此问题可能的原因：在64位程序中使用了32位驱动，或者相反。

  C:\Windows\SysWOW64\odbcad32.exe：这是32位ODBC驱动管理器。

  C:\Windows\System32\odbcad32.exe：这是64位ODBC驱动管理器。

- The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项，请调整此项至allow或以上级别，允许SSL连接，此选项的更多说明，请见**表12-24**。

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

  说明

  - 数据库并不存储用户口令，只存储用户口令的哈希码。
  - 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
  - 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
  - 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

  要解决该问题，参见以下操作：

  a. 联系技术支持人员检查数据库参数password_encryption_type是否为1，如果取值不为1，需要将该参数修改为1。

  b. 新建一个数据库用户用于连接，或者重置准备使用的数据库用户的密码。

     - 如果您使用的是管理员账号，参见**重置密码**。

 版权所有 © 华为云计算技术有限公司

- 如果是普通用户，可以先通过其他客户端工具（例如Data Studio）连接数据库后，使用ALTER USER语句来修改密码。

c. 再尝试连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

- FATAL: GSS authentication method is not allowed because XXXX user password is not disabled.

  或：GSSAPI authentication not supported.

  目标CN的pg_hba.conf里配置了当前客户端IP使用“gss”方式来做认证，该认证算法不支持用作客户端的身份认证，请修改到“sha256”后再试。

  同时请注意，数据库当前不支持在集群内跨节点连接数据库，如果是在集群内跨节点连接CN出现此问题，请将业务程序调整到集群外后重试。

12.3.4 ODBC 开发示例

常用功能示例代码

此示例演示如何通过ODBC方式获取DWS中的数据。

```
// DBtest.c (compile with: libodbc.so)
#include <stdlib.h>
#include <stdio.h>
#include <sqlext.h>
#ifdef WIN32
#include <windows.h>
#endif
SQLHENV       V_OD_Env;        // Handle ODBC environment
SQLHSTMT      V_OD_hstmt;      // Handle statement
SQLHDBC       V_OD_hdbc;       // Handle connection
char          typename[100];
SQLINTEGER    value = 100;
SQLINTEGER    V_OD_erg,V_OD_buffer,V_OD_err,V_OD_id;
SQLLEN        V_StrLen_or_IndPtr;
int main(int argc,char *argv[])
{
    // 1. 申请环境句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&V_OD_Env);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error AllocHandle\n");
        exit(0);
    }
    // 2. 设置环境属性（版本信息）
    SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
    // 3. 申请连接句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    // 4. 设置连接属性
    SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, 0);
    // 5. 连接数据源，这里的“userName”与“password”分别表示连接数据库的用户名和用户密码，请根据实际情况修改。
    // 如果odbc.ini文件中已经配置了用户名密码，那么这里可以留空（""）；但是不建议这么做，因为一旦odbc.ini权限管理不善，将导致数据库用户密码泄露。
    V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*) "gaussdb", SQL_NTS,
                          (SQLCHAR*) "userName", SQL_NTS,  (SQLCHAR*) "password", SQL_NTS);
```

 版权所有 © 华为云计算技术有限公司

```
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error SQLConnect %d\n",V_OD_erg);
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    printf("Connected !\n");
    // 6. 设置语句属性
    SQLSetStmtAttr(V_OD_hstmt,SQL_ATTR_QUERY_TIMEOUT,(SQLPOINTER *)3,0);
    // 7. 申请语句句柄
    SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);
    // 8. 直接执行SQL语句。
    SQLExecDirect(V_OD_hstmt,"drop table IF EXISTS customer_t1",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"insert into customer_t1 values(25,'li')",SQL_NTS);
    // 9. 准备执行
    SQLPrepare(V_OD_hstmt,"insert into customer_t1 values(?)",SQL_NTS);
    // 10. 绑定参数
    SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                     &value,0,NULL);
    // 11. 执行准备好的语句
    SQLExecute(V_OD_hstmt);
    SQLExecDirect(V_OD_hstmt,"select c_customer_sk from customer_t1",SQL_NTS);
    // 12. 获取结果集某一列的属性

SQLColAttribute(V_OD_hstmt,1,SQL_DESC_TYPE_NAME,typename,sizeof(typename),NULL,NULL);

    printf("SQLColAttribute %s\n",typename);
    // 13. 绑定结果集
    SQLBindCol(V_OD_hstmt,1,SQL_C_SLONG, (SQLPOINTER)&V_OD_buffer,150,
            (SQLLEN *)&V_StrLen_or_IndPtr);
    // 14. 通过SQLFetch取结果集中数据
    V_OD_erg=SQLFetch(V_OD_hstmt);
    // 15. 通过SQLGetData获取并返回数据。
    while(V_OD_erg != SQL_NO_DATA)
    {
        SQLGetData(V_OD_hstmt,1,SQL_C_SLONG,(SQLPOINTER)&V_OD_id,0,NULL);
        printf("SQLGetData ----ID = %d\n",V_OD_id);
        V_OD_erg=SQLFetch(V_OD_hstmt);
    };
    printf("Done !\n");
    // 16. 断开数据源连接并释放句柄资源
    SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
    SQLDisconnect(V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC,V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
    return(0);
}
```

批量绑定示例代码

- 请在数据源中打开UseBatchProtocol，同时指定数据库中参数support_batch_bind为on。
- CHECK_ERROR的作用是检查并打印错误信息。
- 此示例将与用户交互式获取DSN、模拟的数据量，忽略的数据量，并将最终数据入库到test_odbc_batch_insert中。

```
#include <stdio.h>
#include <stdlib.h>
#include <sql.h>
#include <sqlext.h>
#include <string.h>

#include "util.c"

void Exec(SQLHDBC hdbc, SQLCHAR* sql)
```

 版权所有 © 华为云计算技术有限公司

```
{
    SQLRETURN retcode;                  // Return status
    SQLHSTMT hstmt = SQL_NULL_HSTMT;    // Statement handle
    SQLCHAR     loginfo[2048];

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
                hstmt, SQL_HANDLE_STMT);

    // Prepare Statement
    retcode = SQLPrepare(hstmt, (SQLCHAR*) sql, SQL_NTS);
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLExecute(hstmt);
    sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    sprintf((char*)loginfo, "SQLFreeHandle stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);
}

int main ()
{
    SQLHENV  henv  = SQL_NULL_HENV;
    SQLHDBC  hdbc  = SQL_NULL_HDBC;
    int      batchCount = 1000;
    SQLLEN   rowsCount = 0;
    int      ignoreCount = 0;

    SQLRETURN   retcode;
    SQLCHAR     dsn[1024] = {'\0'};
    SQLCHAR     loginfo[2048];

    // 交互获取数据源名称
    getStr("Please input your DSN", (char*)dsn, sizeof(dsn), 'N');
    // 交互获取批量绑定的数据量
    getInt("batchCount", &batchCount, 'N', 1);
    do
    {
        // 交互获取批量绑定的数据中，不要入库的数据量
        getInt("ignoreCount", &ignoreCount, 'N', 1);
        if (ignoreCount > batchCount)
        {
            printf("ignoreCount(%d) should be less than batchCount(%d)\n", ignoreCount, batchCount);
        }
    }while(ignoreCount > batchCount);

    retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_ENV)",
                henv, SQL_HANDLE_ENV);

    // Set ODBC Version
    retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
                                (SQLPOINTER*)SQL_OV_ODBC3, 0);
    CHECK_ERROR(retcode, "SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION)",
                henv, SQL_HANDLE_ENV);

    // Allocate Connection
    retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_DBC)",
                henv, SQL_HANDLE_DBC);

    // Set Login Timeout
    retcode = SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (SQLPOINTER)5, 0);
    CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_LOGIN_TIMEOUT)",
                hdbc, SQL_HANDLE_DBC);
```


版权所有 © 华为云计算技术有限公司

```
// Set Auto Commit
retcode = SQLSetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT,
                            (SQLPOINTER)(1), 0);
CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_ATTR_AUTOCOMMIT)",
          hdbc, SQL_HANDLE_DBC);

// Connect to DSN
sprintf(loginfo, "SQLConnect(DSN:%s)", dsn);
retcode = SQLConnect(hdbc, (SQLCHAR*) dsn, SQL_NTS,
                     (SQLCHAR*) NULL, 0, NULL, 0);
CHECK_ERROR(retcode, loginfo, hdbc, SQL_HANDLE_DBC);

// init table info.
Exec(hdbc, "drop table if exists test_odbc_batch_insert");
Exec(hdbc, "create table test_odbc_batch_insert(id int primary key, col varchar2(50))");

// 下面的代码根据用户输入的数据量，构造出将要入库的数据:
{
    SQLRETURN retcode;
    SQLHSTMT hstmtinesrt = SQL_NULL_HSTMT;
    int          i;
    SQLCHAR      *sql = NULL;
    SQLINTEGER   *ids  = NULL;
    SQLCHAR      *cols = NULL;
    SQLLEN       *bufLenIds = NULL;
    SQLLEN       *bufLenCols = NULL;
    SQLUSMALLINT *operptr = NULL;
    SQLUSMALLINT *statusptr = NULL;
    SQLULEN      process = 0;

    // 这里是按列构造，每个字段的内存连续存放在一起。
    ids = (SQLINTEGER*)malloc(sizeof(ids[0]) * batchCount);
    cols = (SQLCHAR*)malloc(sizeof(cols[0]) * batchCount * 50);
    // 这里是每个字段中，每一行数据的内存长度。
    bufLenIds = (SQLLEN*)malloc(sizeof(bufLenIds[0]) * batchCount);
    bufLenCols = (SQLLEN*)malloc(sizeof(bufLenCols[0]) * batchCount);
    // 该行是否需要被处理，SQL_PARAM_IGNORE 或 SQL_PARAM_PROCEED
    operptr = (SQLUSMALLINT*)malloc(sizeof(operptr[0]) * batchCount);
    memset(operptr, 0, sizeof(operptr[0]) * batchCount);
    // 该行的处理结果。
    // 注：由于数据库中处理方式是同一语句隶属同一事务中，所以如果出错，那么待处理数据都将是出错的，
并不会部分入库。
    statusptr = (SQLUSMALLINT*)malloc(sizeof(statusptr[0]) * batchCount);
    memset(statusptr, 88, sizeof(statusptr[0]) * batchCount);

    if (NULL == ids || NULL == cols || NULL == bufLenCols || NULL == bufLenIds)
    {
        fprintf(stderr, "FAILED:\tmalloc data memory failed\n");
        goto exit;
    }

    for (int i = 0; i < batchCount; i++)
    {
        ids[i] = i;
        sprintf(cols + 50 * i, "column test value %d", i);
        bufLenIds[i] = sizeof(ids[i]);
        bufLenCols[i] = strlen(cols + 50 * i);
        operptr[i] = (i < ignoreCount) ? SQL_PARAM_IGNORE : SQL_PARAM_PROCEED;
    }

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmtinesrt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
              hstmtinesrt, SQL_HANDLE_STMT);

    // Prepare Statement
    sql = (SQLCHAR*)"insert into test_odbc_batch_insert values(?, ?)";
    retcode = SQLPrepare(hstmtinesrt, (SQLCHAR*) sql, SQL_NTS);
```


版权所有 © 华为云计算技术有限公司

```
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMSET_SIZE, (SQLPOINTER)batchCount,
sizeof(batchCount));
    CHECK_ERROR(retcode, "SQLSetStmtAttr", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 1, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER,
sizeof(ids[0]), 0,&(ids[0]), 0, bufLenIds);
    CHECK_ERROR(retcode, "SQLBindParameter for id", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 2, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR, 50, 50,
cols, 50, bufLenCols);
    CHECK_ERROR(retcode, "SQLBindParameter for cols", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMS_PROCESSED_PTR, (SQLPOINTER)&process,
sizeof(process));
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAMS_PROCESSED_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_STATUS_PTR, (SQLPOINTER)statusptr,
sizeof(statusptr[0]) * batchCount);
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_STATUS_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_OPERATION_PTR, (SQLPOINTER)operptr,
sizeof(operptr[0]) * batchCount);
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_OPERATION_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

    retcode = SQLExecute(hstmtinesrt);
    sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLRowCount(hstmtinesrt, &rowsCount);
    CHECK_ERROR(retcode, "SQLRowCount execution", hstmtinesrt, SQL_HANDLE_STMT);

    if (rowsCount != (batchCount - ignoreCount))
    {
        sprintf(loginfo, "(batchCount - ignoreCount)(%d) != rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
        CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
    }
    else
    {
        sprintf(loginfo, "(batchCount - ignoreCount)(%d) == rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
        CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
    }

    if (rowsCount != process)
    {
        sprintf(loginfo, "process(%d) != rowsCount(%d)", process, rowsCount);
        CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
    }
    else
    {
        sprintf(loginfo, "process(%d) == rowsCount(%d)", process, rowsCount);
        CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
    }

    for (int i = 0; i < batchCount; i++)
    {
        if (i < ignoreCount)
        {
            if (statusptr[i] != SQL_PARAM_UNUSED)
            {
                sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_UNUSED", i, statusptr[i]);
                CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
```

 版权所有 © 华为云计算技术有限公司

```
            }
        }
        else if (statusptr[i] != SQL_PARAM_SUCCESS)
        {
            sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_SUCCESS", i, statusptr[i]);
            CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
        }
    }

    retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmtinesrt);
    sprintf((char*)loginfo, "SQLFreeHandle hstmtinesrt");
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);
  }


exit:
   printf ("\nComplete.\n");

   // Connection
   if (hdbc != SQL_NULL_HDBC) {
      SQLDisconnect(hdbc);
      SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
   }

   // Environment
   if (henv != SQL_NULL_HENV)
      SQLFreeHandle(SQL_HANDLE_ENV, henv);

   return 0;
}
```

12.3.5 ODBC 接口参考

ODBC接口是一套提供给用户的API函数，本节将对部分常用接口做具体描述，若涉及其他接口可参考**msdn网址**中ODBC Programmer's Reference项的相关内容。

SQLAllocEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocEnv已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLAllocConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocConnect已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLAllocHandle

功能描述

分配环境、连接、语句或描述符的句柄，它替代了ODBC 2.x函数SQLAllocEnv、SQLAllocConnect及SQLAllocStmt。

原型

```
SQLRETURN SQLAllocHandle(SQLSMALLINT   HandleType,
                         SQLHANDLE     InputHandle,
                         SQLHANDLE     *OutputHandlePtr);
```

参数

版权所有 © 华为云计算技术有限公司

表 12-25 SQLAllocHandle 参数

关键字	参数说明
HandleType	由SQLAllocHandle分配的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV（环境句柄） • SQL_HANDLE_DBC（连接句柄） • SQL_HANDLE_STMT（语句句柄） • SQL_HANDLE_DESC（描述句柄） 申请句柄顺序为，先申请环境句柄，再申请连接句柄，最后申请语句句柄，后申请的句柄都要依赖它前面申请的句柄。
InputHandle	将要分配的新句柄的类型。 • 如果HandleType为SQL_HANDLE_ENV，则这个值为SQL_NULL_HANDLE。 • 如果HandleType为SQL_HANDLE_DBC，则这一定是一个环境句柄。 • 如果HandleType为SQL_HANDLE_STMT或SQL_HANDLE_DESC，则它一定是一个连接句柄。
OutputHandlePtr	**输出参数**：一个缓冲区的指针，此缓冲区以新分配的数据结构存放返回的句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当分配的句柄并非环境句柄时，如果SQLAllocHandle返回的值为SQL_ERROR，则它会将OutputHandlePtr的值设置为SQL_NULL_HDBC、SQL_NULL_HSTMT或SQL_NULL_HDESC。之后，通过调用带有适当参数的**SQLGetDiagRec**，其中HandleType和Handle被设置为InputHandle的值，可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLAllocStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocStmt已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

SQLBindCol

功能描述

 版权所有 © 华为云计算技术有限公司

将应用程序数据缓冲区绑定到结果集的列中。

原型

```
SQLRETURN SQLBindCol(SQLHSTMT       StatementHandle,
                     SQLUSMALLINT   ColumnNumber,
                     SQLSMALLINT    TargetType,
                     SQLPOINTER     TargetValuePtr,
                     SQLLEN         BufferLength,
                     SQLLEN         *StrLen_or_IndPtr);
```

参数

表 12-26 SQLBindCol 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要绑定结果集的列号。起始列号为0，以递增的顺序计算列号，第0列是书签列。若未设置书签，则起始列号为1。
TargetType	缓冲区中C数据类型的标识符。
TargetValuePtr	**输出参数**：指向与列绑定的数据缓冲区的指针。SQLFetch函数返回这个缓冲区中的数据。如果此参数为一个空指针，则StrLen_or_IndPtr是一个有效值。
BufferLength	TargetValuePtr指向缓冲区的长度，以字节为单位。
StrLen_or_IndPtr	**输出参数**：缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLBindParameter

功能描述

将一条SQL语句中的一个参数标志和一个缓冲区绑定起来。

版权所有 © 华为云计算技术有限公司

原型

```
SQLRETURN SQLBindParameter(SQLHSTMT       StatementHandle,
                           SQLUSMALLINT   ParameterNumber,
                           SQLSMALLINT    InputOutputType,
                           SQLSMALLINT    ValueType,
                           SQLSMALLINT    ParameterType,
                           SQLULEN        ColumnSize,
                           SQLSMALLINT    DecimalDigits,
                           SQLPOINTER     ParameterValuePtr,
                           SQLLEN         BufferLength,
                           SQLLEN         *StrLen_or_IndPtr);
```

参数

表 12-27 SQLBindParameter

关键词	参数说明
StatementHandle	语句句柄。
ParameterNumber	参数序号，起始为1，依次递增。
InputOutputType	输入输出参数类型。
ValueType	参数的C数据类型。
ParameterType	参数的SQL数据类型。
ColumnSize	列的大小或相应参数标记的表达式。
DecimalDigits	列的十进制数字或相应参数标记的表达式。
ParameterValuePtr	指向存储参数数据缓冲区的指针。
BufferLength	ParameterValuePtr指向缓冲区的长度，以字节为单位。
StrLen_or_IndPtr	缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

版权所有 © 华为云计算技术有限公司

参见：**示例**

SQLColAttribute

功能描述

返回结果集中一列的描述符信息。

原型

```
SQLRETURN SQLColAttribute(SQLHSTMT        StatementHandle,
                          SQLUSMALLINT    ColumnNumber,
                          SQLUSMALLINT    FieldIdentifier,
                          SQLPOINTER      CharacterAttributePtr,
                          SQLSMALLINT     BufferLength,
                          SQLSMALLINT     *StringLengthPtr,
                          SQLPOINTER      NumericAttributePtr);
```

参数

表 12-28 SQLColAttribute 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要检索字段的列号，起始为1，依次递增。
FieldIdentifier	IRD中ColumnNumber行的字段。
CharacterAttributePtr	**输出参数**：一个缓冲区指针，返回FieldIdentifier字段值。
BufferLength	• 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个字符串或二进制缓冲区，则此参数为该缓冲区的长度。 • 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个整数，则会忽略该字段。
StringLengthPtr	**输出参数**：缓冲区指针，存放*CharacterAttributePtr中字符类型数据的字节总数，对于非字符类型，忽略BufferLength的值。
NumericAttributePtr	**输出参数**：指向一个整型缓冲区的指针，返回IRD中ColumnNumber行FieldIdentifier字段的值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

版权所有 © 华为云计算技术有限公司

当SQLColAttribute返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用SQLGetDiagRec函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：示例

SQLConnect

功能描述

在驱动程序和数据源之间建立连接。连接上数据源之后，可以通过连接句柄访问到所有有关连接数据源的信息，包括程序运行状态、事务处理状态和错误信息。

原型

```
SQLRETURN  SQLConnect(SQLHDBC        ConnectionHandle,
                     SQLCHAR        *ServerName,
                     SQLSMALLINT    NameLength1,
                     SQLCHAR        *UserName,
                     SQLSMALLINT    NameLength2,
                     SQLCHAR        *Authentication,
                     SQLSMALLINT    NameLength3);
```

参数

表 12-29 SQLConnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。
ServerName	要连接数据源的名称。
NameLength1	ServerName的长度。
UserName	数据源中数据库用户名。
NameLength2	UserName的长度。
Authentication	数据源中数据库用户密码。
NameLength3	Authentication的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

版权所有 © 华为云计算技术有限公司

当调用SQLConnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLDisconnect

功能描述

关闭一个与特定连接句柄相关的连接。

原型

```
SQLRETURN SQLDisconnect(SQLHDBC     ConnectionHandle);
```

参数

表 12-30 SQLDisconnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当调用SQLDisconnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLExecDirect

功能描述

使用参数的当前值，执行一条准备好的语句。对于一次只执行一条SQL语句，SQLExecDirect是最快的执行方式。

原型

版权所有 © 华为云计算技术有限公司

```
SQLRETURN SQLExecDirect(SQLHSTMT         StatementHandle,
                        SQLCHAR          *StatementText,
                        SQLINTEGER       TextLength);
```

参数

表 12-31 SQLExecDirect 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
StatementText	要执行的SQL语句。不支持一次执行多条语句。
TextLength	StatementText的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。
- SQL_NO_DATA：表示SQL语句不返回结果集。

注意事项

当调用SQLExecDirect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLExecute

功能描述

如果语句中存在参数标记，SQLExecute函数使用参数标记参数的当前值，执行一条准备好的SQL语句。

原型

```
SQLRETURN SQLExecute(SQLHSTMT    StatementHandle);
```

参数

版权所有 © 华为云计算技术有限公司

表 12-32 SQLExecute 参数

关键字	参数说明
StatementHandle	要执行语句的语句句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：表示在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当SQLExecute函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，可通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLFetch

功能描述

从结果集中取下一个行集的数据，并返回所有被绑定列的数据。

原型

```
SQLRETURN SQLFetch(SQLHSTMT    StatementHandle);
```

参数

表 12-33 SQLFetch 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。

版权所有 © 华为云计算技术有限公司

- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLFreeStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeStmt已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

SQLFreeConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeConnect已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

SQLFreeHandle

功能描述

释放与指定环境、连接、语句或描述符相关联的资源，它替代了ODBC 2.x函数SQLFreeEnv、SQLFreeConnect及SQLFreeStmt。

原型

```
SQLRETURN SQLFreeHandle(SQLSMALLINT   HandleType,
                        SQLHANDLE     Handle);
```

参数

表 12-34 SQLFreeHandle 参数

关键字	参数说明
HandleType	SQLFreeHandle要释放的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC 如果HandleType不是这些值之一，SQLFreeHandle返回SQL_INVALID_HANDLE。

版权所有 © 华为云计算技术有限公司

关键字	参数说明
Handle	要释放的句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

如果SQLFreeHandle返回SQL_ERROR，句柄仍然有效。

示例

参见：**示例**

SQLFreeEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeEnv已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

SQLPrepare

功能描述

准备一个将要进行的SQL语句。

原型

```
SQLRETURN SQLPrepare(SQLHSTMT      StatementHandle,
                     SQLCHAR       *StatementText,
                     SQLINTEGER    TextLength);
```

参数

表 12-35 SQLPrepare 参数

关键字	参数说明
StatementHandle	语句句柄。
StatementText	SQL文本串。
TextLength	StatementText的长度。

返回值

版权所有 © 华为云计算技术有限公司

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当SQLPrepare返回的值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLGetData

功能描述

SQLGetData返回结果集中某一列的数据。可以多次调用它来部分地检索不定长度的数据。

原型

```
SQLRETURN SQLGetData(SQLHSTMT        StatementHandle,
                     SQLUSMALLINT    Col_or_Param_Num,
                     SQLSMALLINT     TargetType,
                     SQLPOINTER      TargetValuePtr,
                     SQLLEN          BufferLength,
                     SQLLEN          *StrLen_or_IndPtr);
```

参数

表 12-36 SQLGetData 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
Col_or_Param_Num	要返回数据的列号。结果集的列按增序从1开始编号。书签列的列号为0。
TargetType	TargetValuePtr缓冲中的C数据类型的类型标识符。若TargetType为SQL_ARD_TYPE，驱动使用ARD中SQL_DESC_CONCISE_TYPE字段的类型标识符。若为SQL_C_DEFAULT，驱动根据源的SQL数据类型选择缺省的数据类型。
TargetValuePtr	**输出参数**：指向返回数据所在缓冲区的指针。
BufferLength	TargetValuePtr所指向缓冲区的长度。
StrLen_or_IndPtr	**输出参数**：指向缓冲区的指针，在此缓冲区中返回长度或标识符的值。

版权所有 © 华为云计算技术有限公司

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLGetDiagRec

功能描述

返回诊断记录的多个字段的当前值，其中诊断记录包含错误、警告及状态信息。

原型

```
SQLRETURN  SQLGetDiagRec(SQLSMALLINT    HandleType,
                         SQLHANDLE      Handle,
                         SQLSMALLINT    RecNumber,
                         SQLCHAR        *SQLState,
                         SQLINTEGER     *NativeErrorPtr,
                         SQLCHAR        *MessageText,
                         SQLSMALLINT    BufferLength,
                         SQLSMALLINT    *TextLengthPtr);
```

参数

表 12-37 SQLGetDiagRec 参数

关键字	参数说明
HandleType	句柄类型标识符，它说明诊断所要求的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC
Handle	诊断数据结构的句柄，其类型由HandleType来指出。如果HandleType是SQL_HANDLE_ENV，Handle可以是共享的或非共享的环境句柄。

版权所有 © 华为云计算技术有限公司

关键字	参数说明
RecNumber	指出应用从查找信息的状态记录。状态记录从1开始编号。
SQLState	**输出参数**：指向缓冲区的指针，该缓冲区存储着有关RecNumber的五字符的SQLSTATE码。
NativeErrorPtr	**输出参数**：指向缓冲区的指针，该缓冲区存储着本地的错误码。
MessageText	指向缓冲区的指针，该缓冲区存储着诊断信息文本串。
BufferLength	MessageText的长度。
TextLengthPtr	**输出参数**：指向缓冲区的指针，返回MessageText中的字节总数。如果返回字节数大于BufferLength，则MessageText中的诊断信息文本被截断成BufferLength减去NULL结尾字符的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

SQLGetDiagRec不发布自己的诊断记录。它用下列返回值来报告它自己的执行结果：

- SQL_SUCCESS：函数成功返回诊断信息。
- SQL_SUCCESS_WITH_INFO：*MessageText太小以致不能容纳所请求的诊断信息。没有诊断记录生成。
- SQL_INVALID_HANDLE：由HandType和Handle所指出的句柄是不合法句柄。
- SQL_ERROR：RecNumber小于等于0或BufferLength小于0。

如果调用ODBC函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO，可调用SQLGetDiagRec返回诊断信息值SQLSTATE，SQLSTATE值的如下表。

表 12-38 SQLSTATE 值

SQLSTATE	错误	描述
HY000	一般错误	未定义特定的SQLSTATE所产生的一个错误。
HY001	内存分配错误	驱动程序不能分配所需要的内存来支持函数的执行或完成。
HY008	取消操作	调用SQLCancel取消执行语句后，依然在StatementHandle上调用函数。

版权所有 © 华为云计算技术有限公司

SQLSTATE	错误	描述
HY010	函数系列错误	在为执行中的所有数据参数或列发送数据前就调用了执行函数。
HY013	内存管理错误	不能处理函数调用，可能由当前内存条件差引起。
HYT01	连接超时	数据源响应请求之前，连接超时。
IM001	驱动程序不支持此函数	调用了StatementHandle相关的驱动程序不支持的函数

示例

参见：**示例**

SQLSetConnectAttr

功能描述

设置控制连接各方面的属性。

原型

```
SQLRETURN SQLSetConnectAttr(SQLHDBC      ConnectionHandle,
                            SQLINTEGER   Attribute,
                            SQLPOINTER   ValuePtr,
                            SQLINTEGER   StringLength);
```

参数

表 12-39 SQLSetConnectAttr 参数

关键字	参数说明
ConnectionHandle	连接句柄。
Attribute	设置属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr是32位无符号整型值或指向以空结束的字符串。如果ValuePtr参数是驱动程序指定值，ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度。如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。

版权所有 © 华为云计算技术有限公司

- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetConnectAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_DBC的HandleType和ConnectionHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLSetEnvAttr

功能描述

设置控制环境各方面的属性。

原型

```
SQLRETURN SQLSetEnvAttr(SQLHENV      EnvironmentHandle,
                        SQLINTEGER   Attribute,
                        SQLPOINTER   ValuePtr,
                        SQLINTEGER   StringLength);
```

参数

表 12-40 SQLSetEnvAttr 参数

关键字	参数说明
EnvironmentHandle	环境句柄。
Attribute	需设置的环境属性，可为如下值： • SQL_ATTR_ODBC_VERSION：指定ODBC版本。 • SQL_CONNECTION_POOLING：连接池属性。 • SQL_OUTPUT_NTS：指明驱动器返回字符串的形式。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位整型值，或为以空结束的字符串。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

版权所有 © 华为云计算技术有限公司

当SQLSetEnvAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_ENV的HandleType和EnvironmentHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

SQLSetStmtAttr

功能描述

设置相关语句的属性。

原型

```
SQLRETURN SQLSetStmtAttr(SQLHSTMT      StatementHandle,
                         SQLINTEGER    Attribute,
                         SQLPOINTER    ValuePtr,
                         SQLINTEGER    StringLength);
```

参数

表 12-41 SQLSetStmtAttr 参数

关键字	参数说明
StatementHandle	语句句柄。
Attribute	需设置的属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位无符号整型值，或指向以空结束的字符串，二进制缓冲区，或者驱动定义值。注意，如果ValuePtr参数是驱动程序指定值。ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetStmtAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_STMT的HandleType和StatementHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

 版权所有 © 华为云计算技术有限公司

参见：**示例**

 版权所有 © 华为云计算技术有限公司

13 DWS 资源监控

DWS为用户提供了多维度的资源监控视图。可以查看作业的实时资源记录和历史资源记录。

13.1 用户资源监控

在多租户管理的框架下，用户可以通过系统视图**PG_TOTAL_USER_RESOURCE_INFO**、**PGXC_TOTAL_USER_RESOURCE_INFO**和函数GS_WLM_USER_RESOURCE_INFO实时查询所有用户资源（包括内存，CPU核数，存储空间、临时空间、算子落盘空间和IO）实时使用情况，也可以通过系统表**GS_WLM_USER_RESOURCE_HISTORY**和系统视图**PGXC_WLM_USER_RESOURCE_HISTORY**查询用户资源的历史使用情况。

注意事项

- 用户监控可以同时监控快慢车道（快车道管控简单作业，慢车道管控复杂作业）所有作业的CPU、IO和内存使用情况，不再受限于仅监控慢车道作业。
- 当前快车道作业内存和CPU不受控，在快车道运行作业占用资源较多情况下，可能出现已用资源大于资源限制的情况。
- DN监控视图中，IO、内存和CPU显示的是本DN上资源池资源使用和资源限制信息。
- CN监控视图中，IO、内存和CPU显示的是集群内所有DN资源池资源使用和资源限制的累积和。
- DN每隔5秒更新一次监控信息，CN每隔5秒从DN收集一次用户监控信息，因为各实例单独更新/收集用户监控信息，因此各实例监控信息更新时间可能不一致。
- 辅助线程中每隔30秒自动调用持久化函数，持久化用户监控数据，正常情况下不需要用户单独调用持久化函数持久化用户监控数据。
- 当用户数量较多，集群规模较大时，查询此类实时视图，因CN/DN间实时通信开销，会有一定的网络延时。
- 初始管理用户不进行资源监控。

操作步骤

- 查询所有用户的资源限额和资源实时使用情况。

```
SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO;
```

  得到的结果视图如下：

 版权所有 © 华为云计算技术有限公司

```
username       | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
---------------------+-------------+--------------+----------+-----------+------------+-------------
+-----------------+-----------------+-----------------+------------------+------------+-------------
+-------------+--------------+------------+-------------+------------+------------
perfadm            |          0 |          0 |        0 |        0 |         0 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
usern              |          0 |      17250 |        0 |       48 |         0 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
(2 rows)
```

其中，IO资源监控字段(read_kbytes、write_kbytes、read_counts、write_counts、read_speed和write_speed)需要在GUC参数**enable_user_metric_persistent**开启时才有监控数据。

所查各字段说明详见**PG_TOTAL_USER_RESOURCE_INFO** 。

- 查询具体某个用户的资源限额和资源实时使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_INFO('username');
```

查询结果如下：

```
userid | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
--------+-------------+--------------+----------+-----------+------------+-------------
+-----------------+-----------------+-----------------+------------------+------------+-------------
+-------------+--------------+------------+-------------+------------+------------
16407 |          18 |       1655 |        6 |         19 |         13787176 |         -1 |             0 |            -1
|             0 |            -1 |          0 |          0 |          0 |          0 |          0 |          0 |         0
|          0
(1 row)
```

- 查询所有用户的资源限额和资源历史使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_HISTORY;
```

查询结果如下：

```
username       |          timestamp          | used_memory | total_memory | used_cpu | total_cpu |
used_space | total_space | used_temp_space | total_temp_space | used_spill_space | total_spill_space |
read_kbytes | write_kbytes | read_counts | write_counts | read_speed  | write_speed | send_speed |
recv_speed
----------------------+-------------------------------+-------------+--------------+----------+-----------
+------------+-------------+-----------------+------------------+------------------+-------------------
+-------------+--------------+-------------+--------------+-------------+-------------+------------
usern              | 2020-01-08 22:56:06.456855+08 |           0 |        17250 |        0 |        48 |          0
|          -1 |             0 |             -1 |         88349078 |             -1 |        45680 |          34 |        5710
|           8 |         320 |           0 |          0 |          0
userg              | 2020-01-08 22:56:06.458659+08 |           0 |        15525 |    33.48 |        48 |          0
|          -1 |             0 |             -1 |        110169581 |             -1 |        17648 |          23 |
2206 |           5 |         123 |           0 |          0 |          0
userg1             | 2020-01-08 22:56:06.460252+08 |           0 |        13972 |    33.48 |        48 |          0
|          -1 |             0 |             -1 |        136106277 |             -1 |        17648 |          23 |
2206 |           5 |         123 |           0 |          0 |          0
```

对于系统表**GS_WLM_USER_RESOURCE_HISTORY**，仅当GUC参数**enable_user_metric_persistent**开启时，才会定期将视图**PG_TOTAL_USER_RESOURCE_INFO**中的数据保存到历史表中。

所查各字段说明详见**GS_WLM_USER_RESOURCE_HISTORY**。

版权所有 © 华为云计算技术有限公司

13.2 资源池资源监控

简介

多租户管理框架下，用户关联资源池执行查询，用户执行查询所占用的资源将汇总至关联资源池上，通过资源池监控视图，用户可以直观的查询到所有资源池的实时资源使用情况，同时也可以通过资源池监控历史表查询资源池资源的历史使用情况。

资源池监控数据每5秒更新一次，但是因为CN和DN时间差，实际监控数据更新时间可能会大于5秒，正常不会超过10秒。资源池监控数据每30秒持久化一次，资源池监控和用户监控逻辑基本一致，因此共用GUC参数控制持久化和老化，使用GUC参数enable_user_metric_persistent控制是否进行资源池监控数据持久化，使用GUC参数user_metric_retention_time控制资源池监控数据老化。

资源池监控的资源包含：快慢车道作业运行和排队信息，CPU、内存以及逻辑IO资源监控信息。涉及的监控视图和历史表如下：

- 资源池实时运行信息监控视图(单CN)：**GS_RESPOOL_RUNTIME_INFO**。
- 资源池实时运行信息监控视图(所有CN)：**PGXC_RESPOOL_RUNTIME_INFO**。
- 资源池实时资源监控视图(单CN)：**GS_RESPOOL_RESOURCE_INFO**。
- 资源池实时资源监控视图(所有CN)：**PGXC_RESPOOL_RESOURCE_INFO**。
- 资源池历史资源监控表(单CN)：**GS_RESPOOL_RESOURCE_HISTORY**。
- 资源池历史资源监控视图(所有CN)：**PGXC_RESPOOL_RESOURCE_HISTORY**。

说明

- 资源池监控可以同时监控快慢车道所有作业的CPU、IO和内存使用情况，不再受限于仅监控慢车道作业。
- 当前快车道作业内存和CPU不受控，在快车道运行作业占用资源较多情况下，可能出现已用资源大于资源限制的情况。
- DN资源池监控视图中，IO、内存和CPU显示的是本DN上资源池资源使用和资源限制信息。
- CN资源池监控视图中，IO、内存和CPU显示的是集群内所有DN资源池资源使用和资源限制的累积和。
- DN每隔5秒更新一次资源池监控信息，CN每隔5秒从DN收集一次资源池监控信息，因为各实例单独更新/收集资源池监控信息，因此各实例监控信息更新时间可能不一致。
- 辅助线程中每隔30秒自动调用持久化函数，持久化资源池监控数据，正常情况下不需要用户单独调用持久化函数持久化资源池监控数据。

操作步骤

- 查询资源池的作业实时运行情况。

```
SELECT * FROM GS_RESPOOL_RUNTIME_INFO;
```

得到的结果视图如下：

```
 nodegroup |     rpname     | ref_count | fast_run | fast_wait | slow_run | slow_wait
-----------+----------------+-----------+----------+-----------+----------+-----------
 vc1       | p2             |        10 |        0 |         0 |        0 |         0
 vc2       | p3             |        10 |        5 |         5 |        0 |         0
 vc2       | p4             |         0 |        0 |         0 |        0 |         0
 vc1       | default_pool |         0 |        0 |         0 |        0 |         0
 vc2       | default_pool |         0 |        0 |         0 |        0 |         0
 vc1       | p1             |        20 |        5 |         5 |        3 |         7
(6 rows)
```


版权所有 © 华为云计算技术有限公司

其中：

a. ref_count为引用当前资源池信息的作业数，语句从进入管控到结束一直占用该计数；

b. fast_run和slow_run为负载管理记账信息，只有管控(fast_limit/slow_limit大于0)时该值才有效；

c. 该视图仅在CN上有效，持久化信息保存在GS_RESPOOL_RESOURCE_HISTORY中；

d. 各字段说明详见**GS_RESPOOL_RUNTIME_INFO**。

- 查询资源池的资源限额和资源实时使用情况。

```
SELECT * FROM GS_RESPOOL_RESOURCE_INFO;
```

得到的结果视图如下：

```
nodegroup |     rpname     |       cgroup        | ref_count | fast_run | fast_wait | fast_limit | slow_run |
slow_wait | slow_limit | used_cpu | cpu_limit | used_mem | estimate_mem | mem_limit |read_kbytes |
write_kbytes | read_counts | write_counts | read_speed | write_speed | send_speed | recv_speed
-----------+----------------+---------------------+-----------+----------+-----------+------------+----------
+-----------+------------+----------+-----------+----------+--------------+-----------+-------------+--------------
+-------------+--------------+-------------+------------+------------+------------
 vc1       | p2             | DefaultClass:Rush   |        10 |        0 |         0 |         -1 |        0 |         0 |        10
|     9.97 |        48 |       20 |           0 |     11555 |         8 |        2880 |          1 |         360 |         1
|         589 |          0 |          0
 vc2       | p3             | DefaultClass:Rush   |        10 |        5 |         5 |          5 |        0 |         0 |        10
|     4.98 |        48 |       11 |           0 |     11555 |         0 |         848 |          0 |         106 |         0
|         173 |          0 |          0
 vc2       | p4             | DefaultClass:Rush   |         0 |        0 |         0 |         -1 |        0 |         0 |        10
|        0 |        48 |        0 |           0 |     11555 |         0 |           0 |          0 |           0 |         0
|           0 |          0 |          0
 vc1       | default_pool | DefaultClass:Medium |         0 |        0 |         0 |         -1 |        0 |         0
|         -1 |        0 |       48 |        0 |           0 |     11555 |         0 |          0 |           0 |          0
|          0 |          0 |          0 |          0
 vc2       | default_pool | DefaultClass:Medium |         0 |        0 |         0 |         -1 |        0 |         0
|         -1 |        0 |       48 |        0 |           0 |     11555 |         0 |          0 |           0 |          0
|          0 |          0 |          0 |          0
 vc1       | p1             | DefaultClass:Rush   |        20 |        5 |         5 |          5 |        3 |         7 |         3
|     7.98 |        48 |       16 |         768 |     11555 |         8 |        2656 |          1 |         332 |         1
|         543 |          0 |          0
(6 rows)
```

a. 该视图在CN和DN上均有效，DN上CPU、内存和IO为本DN资源消耗情况，CN上CPU、内存和IO为集群内所有DN上资源消耗的累加和；

b. estimate_mem仅在动态负载管理情况下CN上有效，显示资源池估算内存记账情况；

c. IO监控信息仅在enable_logical_io_statistics开启时才会记录；

d. 各字段说明详见**GS_RESPOOL_RESOURCE_INFO**。

- 查询资源池的资源限额和资源历史使用情况。

```
SELECT * FROM GS_RESPOOL_RESOURCE_HISTORY ORDER BY timestamp DESC;
```

得到的结果视图如下：

```
timestamp           |  nodegroup   |     rpname     |       cgroup        | ref_count | fast_run | fast_wait |
fast_limit | slow_run | slow_wait | slow_limit | used_cpu | cpu_limit | used_mem | estimate_mem |
mem_limit | read_kbytes | write_kbytes | read_counts | write_counts | read_speed | write_speed |
send_speed | recv_speed
----------------------------+--------------+----------------+---------------------+-----------+----------+-----------+
+-----------+------------+----------+-----------+----------+--------------+-----------+-------------+--------------
+-------------+--------------+-------------+------------+------------+------------
 2022-03-04 09:41:57.53739+08  | vc1          | p2             | DefaultClass:Rush   |        10 |        0 |         0
|         -1 |        0 |         0 |         10 |     9.97 |        48 |   20 |           0 |     11555 |          0 |
2320 |          0 |         290 |          0 |         474 |          0 |          0
 2022-03-04 09:41:57.53739+08  | vc1          | p1             | DefaultClass:Rush   |        20 |        5 |         5
|          5 |        3 |         7 |          3 |     7.98 |        48 |   16 |         768 |     11555 |          0 |
```


版权所有 © 华为云计算技术有限公司

```
1896 |          0 |         237 |          0 |         387 |          0 |          0
 2022-03-04 09:41:57.53739+08  | vc2          | default_pool | DefaultClass:Medium |        0 |       0
|        0 |        -1 |       0 |        0 |        -1 |       0 |       48 |    0 |           0 |     11555 |          0
|           0 |         0 |          0 |          0 |          0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc1          | default_pool | DefaultClass:Medium |        0 |       0
|        0 |        -1 |       0 |        0 |        -1 |       0 |       48 |    0 |           0 |     11555 |          0
|           0 |         0 |          0 |          0 |          0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc2          | p4           | DefaultClass:Rush   |        0 |       0 |       0
|        -1 |       0 |        0 |        10 |       0 |       48 |    0 |           0 |     11555 |          0 |          0
|           0 |         0 |          0 |          0 |          0 |         0 |         0
 2022-03-04 09:41:57.53739+08  | vc2          | p3           | DefaultClass:Rush   |       10 |       5 |       5
|         5 |       0 |        0 |        10 |     4.99 |       48 |   11 |           0 |     11555 |          0 |        880
|           0 |         110 |          0 |         180 |          0 |         0
 2022-03-04 09:41:27.335234+08 | vc2          | p3           | DefaultClass:Rush   |       10 |       5 |       5
|         5 |       0 |        0 |        10 |     4.98 |       48 |   11 |           0 |     11555 |          0 |        856
|           0 |         107 |          0 |         175 |          0 |         0
```

a. 该监控信息来自资源池监控历史表，enable_user_metric_persistent开启时每30秒记录一次；

b. 该表数据保存时间由GUC参数user_metric_retention_time控制；

c. 各字段说明详见**GS_RESPOOL_RESOURCE_HISTORY**。

13.3 内存资源监控

内存监控

DWS支持在监控面板(DMS)查看集群内存使用情况，详情参考**在监控面板(DMS)查看DWS集群监控**章节。

DWS也提供了监控整个集群内存使用状态的视图：

查询pgxc_total_memory_detail视图，必须具有sysadmin权限。

```
SELECT * FROM pgxc_total_memory_detail;
```

如果查询该视图时出现以下错误，请开启内存管理功能（该功能由GUC参数enable_memory_limit控制，该参数默认开启。若确认参数已关闭，请联系技术支持开启）。

```
SELECT * FROM pgxc_total_memory_detail;
ERROR:  unsupported view for memory protection feature is disabled.
CONTEXT:  PL/pgSQL function pgxc_total_memory_detail() line 12 at FOR over EXECUTE statement
```

用户可通过DWS控制台修改max_process_memory参数，调整数据库节点可用的最大物理内存，方法如下：

1. 登录**DWS控制台**。
2. 在左侧导航栏中，单击**“集群 > 集群列表”**。
3. 在集群列表中找到所需要的集群，单击集群名称，进入**“集群详情”**页面。
4. 单击“参数修改”页签，并在“参数列表”模块修改“max_process_memory”参数值，修改建议请参见**max_process_memory**，然后单击**“保存”**。
5. 在**“修改预览”**窗口，确认修改无误后，单击**“保存”**。

共享内存监控

用户可以通过视图pg_shared_memory_detail查询共享内存上下文信息：

```
SELECT * FROM pg_shared_memory_detail;
          contextname          | level |            parent              | totalsize | freesize | usedsize
-------------------------------+-------+--------------------------------+-----------+----------+----------
```


版权所有 © 华为云计算技术有限公司

```
ProcessMemory                 |     0 |                                |    24576 |     9840 |    14736
Workload manager memory context |     1 | ProcessMemory                |  2105400 |     7304 |  2098096
wlm collector hash table      |     2 | Workload manager memory context |     8192 |     3736 |     4456
Resource pool hash table      |     2 | Workload manager memory context |    24576 |    15968 |     8608
wlm cgroup hash table         |     2 | Workload manager memory context |    24576 |    15968 |     8608
(5 rows)
```

该视图列举了内存使用的上下文名称、级别、上级内存上下文、共享内存总大小等。

另外，在数据库中，GUC参数“memory_tracking_mode”用来设置内存信息统计的模式，共支持四种模式：

- none，不启动内存统计功能。
- normal，仅做内存实时统计，不生成文件。
- executor，生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。

  当为executor模式时，将在DN进程的pg_log目录下生成csv格式文件，命名方式为：memory_track_<DN名称>_query_<queryid>.csv。作业执行时，执行器postgres线程和所有stream线程执行的算子信息，都将输入该文件。

  文件内容根据以下内容组成示例如下：

  ```
  0, 0, ExecutorState, 0, PortalHeapMemory, 0, 40K, 602K, 23
  1, 3, CStoreScan_29360131_25, 0, ExecutorState, 1, 265K, 554K, 23
  2, 128, cstore scan per scan memory context, 1, CStoreScan_29360131_25, 2, 24K, 24K, 23
  3, 127, cstore scan memory context, 1, CStoreScan_29360131_25, 2, 264K, 264K, 23
  4, 7, InitPartitionMapTmpMemoryContext, 1, CStoreScan_29360131_25, 2, 31K, 31K, 23
  5, 2, VecPartIterator_29360131_24, 0, ExecutorState, 1, 16K, 16K, 23
  0, 0, ExecutorState, 0, PortalHeapMemory, 0, 24K, 1163K, 20
  1, 3, CStoreScan_29360131_22, 0, ExecutorState, 1, 390K, 1122K, 20
  2, 20, cstore scan per scan memory context, 1, CStoreScan_29360131_22, 2, 476K, 476K, 20
  3, 19, cstore scan memory context, 1, CStoreScan_29360131_22, 2, 264K, 264K, 20
  4, 7, InitPartitionMapTmpMemoryContext, 1, CStoreScan_29360131_22, 2, 23K, 23K, 20
  5, 2, VecPartIterator_29360131_21, 0, ExecutorState, 1, 16K, 16K, 20
  ```

  其中各字段分别为：输出顺序号、线程内分配内存上下文的顺序号、当前内存上下文的名称、父内存上下文的输出顺序号、父内存上下文的名称、内存上下文树形层次级别号、当前内存上下文使用的内存峰值、当前内存上下文及其所有子内存上下文使用的内存峰值、当前线程所在query的plannodeid。

  在本例中，记录“1, 3, CStoreScan_29360131_22, 0, ExecutorState, 1, 390K, 1122K, 20”和Explain Analyze的对应关系如下：

  - CstoreScan_29360131_22代表CstoreScan算子。
  - 1122K代表CstoreScan算子的PeakMemory。
- fullexec，生成文件包含执行层申请过的所有内存上下文信息。

  当设置为fullexec模式时，输出信息和executor模式相同，但可能增加部分内存上下文分配信息，因为所有申请内存（无论是否申请成功）相关的信息都会被打印出来。由于仅记录内存申请信息，故记录中内存上下文使用的峰值均为0。

13.4 实例资源监控

DWS提供了监控CN、DN实例资源使用状态（包括内存，CPU，磁盘IO，进程物理IO和进程逻辑IO）的系统表

GS_WLM_INSTANCE_HISTORY及监控整个集群资源使用状态的函数pgxc_get_wlm_history_instance_info()和pgxc_get_wlm_history_instance_info()。

版权所有 © 华为云计算技术有限公司

操作步骤

- 查询当前实例最近的资源使用情况。

```
SELECT * FROM GS_WLM_INSTANCE_HISTORY ORDER BY TIMESTAMP DESC;
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
-------------+-------------------------------+----------+----------+----------+----------+----------+-----------
+------------+--------------+---------------+----------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2026-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |      357717 |        37667
dn_6015_6016 | 2026-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |      358103 |        38584
dn_6015_6016 | 2026-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |      316085 |        39079
dn_6015_6016 | 2026-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |      303351 |        38136
```

- 查询当前实例某一段时间内的资源使用情况。

```
SELECT * FROM GS_WLM_INSTANCE_HISTORY WHERE TIMESTAMP > '2026-01-10' AND TIMESTAMP
< '2026-01-11' ORDER BY TIMESTAMP DESC;
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
-------------+-------------------------------+----------+----------+----------+----------+----------+-----------
+------------+--------------+---------------+----------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2026-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |      357717 |        37667
dn_6015_6016 | 2026-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |      358103 |        38584
dn_6015_6016 | 2026-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |      316085 |        39079
dn_6015_6016 | 2026-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |      303351 |        38136
```

- 查询集群最近的资源使用情况，可以在CN节点上调用pgxc_get_wlm_current_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_current_instance_info('ALL');
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
-------------+-------------------------------+----------+----------+----------+----------+---------+-----------
+------------+--------------+---------------+----------------+--------------+---------------+-------------+--------------
coordinator2 | 2026-01-14 21:58:29.290894+08 |        0 |    12010 |      278 |  16.0445 | 7.19561 |
184.431 |    27959.3 |            0 |             10 |            0 |             0 |           0 |            0
coordinator3 | 2026-01-14 21:58:27.567655+08 |        0 |    12000 |      288 |  .964557 | 3.40659 |
332.468 |    3375.02 |           26 |             13 |            0 |             0 |           0 |            0
datanode1    | 2026-01-14 21:58:23.900321+08 |        0 |    11899 |      389 |  1.17296 |    3.25 |
329.6 |     2870.4 |           28 |              8 |           13 |             3 |          18 |            6
datanode2    | 2026-01-14 21:58:32.832989+08 |        0 |    11904 |      384 |   17.948 | 8.52148 |
214.186 |    25894.1 |           28 |             10 |           13 |             3 |          18 |            6
datanode3    | 2026-01-14 21:58:24.826694+08 |        0 |    11894 |      394 |  1.16088 |    3.15 |      328
|     2868.8 |           25 |             10 |           13 |             3 |          18 |           6
coordinator1 | 2026-01-14 21:58:33.367649+08 |        0 |    11988 |      300 |  9.53286 |   10.05 |
43.2 |      55232 |            0 |              0 |            0 |             0 |           0 |            0
coordinator1 | 2026-01-14 21:58:23.216645+08 |        0 |    11988 |      300 |  1.17085 | 3.21182 |
324.729 |    2831.13 |            8 |             13 |            0 |             0 |           0 |            0
(7 rows)
```

- 查询集群历史的资源使用情况，可以在CN节点上调用pgxc_get_wlm_history_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_history_instance_info('ALL', '2026-01-14 21:00:00', '2026-01-14
22:00:00', 3);
```


版权所有 © 华为云计算技术有限公司

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await |  io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
-------------+-------------------------------+----------+----------+----------+----------+-----------
+------------+-------------+--------------+---------------+--------------+---------------+-------------
coordinator2 | 2026-01-14 21:50:49.778902+08 |        0 |    12020 |      268 |  .127371 |   .789211 |
15.984 |    3994.41 |           0 |           0 |           0 |           0 |           0 |           0
coordinator2 | 2026-01-14 21:53:49.043646+08 |        0 |    12018 |      270 |  30.2902 |   8.65404 |
276.77 |    16741.8 |           3 |           1 |           0 |           0 |           0 |           0
coordinator2 | 2026-01-14 21:57:09.202654+08 |        0 |    12018 |      270 |   .16051 |   .979021 |
59.9401 |       5596 |           0 |           0 |           0 |           0 |           0 |           0
coordinator3 | 2026-01-14 21:38:48.948646+08 |        0 |    12012 |      276 | .0769231 | .00999001
|         0 |    35.1648 |           0 |           1 |           0 |           0 |           0 |           0
coordinator3 | 2026-01-14 21:40:29.061178+08 |        0 |    12012 |      276 |  .118421 |   .0199601
|         0 |    970.858 |           0 |           0 |           0 |           0 |           0 |           0
coordinator3 | 2026-01-14 21:50:19.612777+08 |        0 |    12010 |      278 |   24.411 |   11.7665 |
8.78244 |    44641.1 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2026-01-14 21:49:42.758649+08 |        0 |    11909 |      379 |  .798776 |      8.02 |
51.2 |    20924.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2026-01-14 21:49:52.760188+08 |        0 |    11909 |      379 |  23.8972 |      14.1 |
0 |      74760 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2026-01-14 21:50:22.769226+08 |        0 |    11909 |      379 |  39.5868 |       7.4 |         0
|    19760.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2026-01-14 21:58:02.826185+08 |        0 |    11905 |      383 |  .351648 |       .32 |
20.8 |      504.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2026-01-14 21:56:42.80793+08  |        0 |    11906 |      382 |  .559748 |       .04 |         0
|      326.4 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2026-01-14 21:45:21.632407+08 |        0 |    11901 |      387 |  12.1313 |   4.55544 |
3.1968 |    45177.2 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2026-01-14 21:58:14.823317+08 |        0 |    11898 |      390 |  .378205 |       .99 |
48 |    23353.6 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2026-01-14 21:47:50.665028+08 |        0 |    11901 |      387 |  1.07494 |       1.19 |
0 |    15506.4 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2026-01-14 21:51:21.720117+08 |        0 |    11903 |      385 |  10.2795 |       3.11 |
0 |    11031.2 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2026-01-14 21:42:59.121945+08 |        0 |    12020 |      268 | .0857143 | .0699301
|         0 |    6579.02 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2026-01-14 21:41:49.042646+08 |        0 |    12020 |      268 |  20.9039 |   11.3786 |
6042.76 |    57903.7 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2026-01-14 21:41:09.007652+08 |        0 |    12020 |      268 | .0446429 |    .03996 |
0 |    1109.29 |           0 |           0 |           0 |           0 |           0 |           0
(18 rows)
```

13.5 实时 TopSQL

系统提供了不同级别的资源监控实时视图用来查询实时TopSQL。资源监控实时视图记录了查询作业运行时的资源使用情况(包括内存、下盘、CPU时间等)以及性能告警信息。

实时视图具体的对外接口如下表所示：

表 13-1 资源监控实时视图

视图级别	节点范围	查询视图
query级别/perf级别	当前CN	GS_WLM_SESSION_STATISTICS
	所有CN	PGXC_WLM_SESSION_STATISTICS
operator_realtime级别/operator级别	当前CN	GS_WLM_OPERATOR_STATISTICS
	所有CN	PGXC_WLM_OPERATOR_STATISTICS

版权所有 © 华为云计算技术有限公司

约束和限制

- 视图级别取决于资源监控的等级，即参数**resource_track_level**的配置。
- perf和operator级别会影响**GS_WLM_SESSION_STATISTICS**/**PGXC_WLM_SESSION_INFO**中的query_plan和warning字段的取值，详细内容参见**SQL自诊断**。
- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 查询此类实时视图时，因需要获取作业运行实时资源使用情况，会有一定的网络延时。
- 实例故障时，实时TopSQL视图有可能记录不全。
- 实时TopSQL中能够记录的SQL语句的规格是：
  - 不记录特殊数据定义语句，如：SET、RESET、SHOW、ALTER SESSION SET、SET CONSTRAINTS语句。
  - 记录数据定义语句，例如：执行CREATE、ALTER、DROP、GRANT、REVOKE和VACUUM语句。
  - 记录数据操作语句，例如：
    - 执行SELECT、INSERT、UPDATE和DELETE语句。
    - 执行explain analyze和explain performance场景。
    - 使用query级别/perf级别视图。
  - 记录函数与存储过程的调用入口语句，当GUC参数**enable_track_record_subsql**开启的情况下，可记录存储过程的部分内部语句(declare定义语句除外)，仅会记录其中下发到DN执行的内部语句，未下发到DN执行的内部语句会被过滤掉。
  - 记录匿名块语句，当GUC参数**enable_track_record_subsql**开启的情况下，可记录匿名块中的部分内部语句，仅会记录其中下发到DN执行的内部语句，未下发到DN执行的内部语句会被过滤掉。
  - 记录游标语句，当游标没有从缓存中读取数据，而触发语句下发到DN上执行的条件下，该游标语句会被记录，并且会进行语句、执行计划增强，但当游标从缓存中读取数据时，不进行记录；当游标语句在匿名块或者函数中使用且游标从DN上读取较多数据但不完全使用时，因当前架构限制，无法记录该游标在DN上的监控信息。对于With Hold游标，该语法执行逻辑特殊，会在事务提交阶段执行实际查询动作，当语句在该阶段执行报错时，作业的aborted状态无法反馈到TopSQL历史表中。
  - 重分布过程中的作业不统计。
  - JDBC执行的带占位符语句，通常会补齐参数内容，但当参数和原语句合起来长度超过64KB，则不记录参数，或者是轻量化语句直接下发到DN上执行，不记录参数。
  - 从8.1.3集群版本开始，query、perf级别TopSQL运行时监控功能已完全不影响查询性能，对当前会话的语句进行资源监控的GUC参数resource_track_cost默认值已修改为0，查询TopSQL实时监控视图时，默认会显示所有正在执行的语句。
  - 从8.1.3集群版本开始，对于存储过程中的子语句监控功能，如果在查询TopSQL实时监控视图的会话中，开启控制子语句记录归档功能的GUC参数

版权所有 © 华为云计算技术有限公司

enable_track_record_subsql，不论业务语句中是否开启子语句监控开关，查询TopSQL实时监控视图的结果都能看到执行语句的子语句运行信息。

- 关于存储过程中子语句的监控功能即enable_track_record_subsql，8.1.3集群版本中建议不要全面开启，由于没有按时间过滤子语句的功能，全面开启可能会记录过多子语句，导致归档的监控表占用大量磁盘空间；8.1.3集群版本建议仅用于查询实时监控信息，或对个别存储过程业务做定位分析时，仅开启对应会话中的参数。8.2.1及以上集群版本新增GUC参数**resource_track_subsql_duration**，可以通过执行时间过滤需要归档的子语句，用户可以按需调整该值大小。
- 9.1.2及以上集群版本，新增GUC参数**resource_track_subsql_max_depth**，用于控制存储过程中嵌套子语句的最大监控层数。TopSQL实时监控视图支持按子语句调用层级展示当前正在执行的嵌套子语句信息。
- 由于规格限制，对于未下盘的主语句，TopSQL历史表中的记录会有延时，等待下次作业下发时才会显示在TopSQL历史表中。
- 8.2.1.200及以上集群版本，新增operator_realtime级别TopSQL运行时监控，提供算子级实时监控的能力，开启此级别的监控可以查询语句的执行计划以及具体执行信息，查询TopSQL算子级实时监控视图时，默认会显示所有正在执行的语句。但是对于存储过程和游标场景，暂时不支持显示算子级实时监控信息。该版本暂时不支持算子历史信息查询。
- operator_realtime级别TopSQL运行时监控对于CN轻量化和存储过程的情况，暂时不支持。另由于算子执行速度较快的原因，对于算子信息的显示会有一定滞后性。
- query级别的作业监控和operator的算子监控中的spill_size字段，由于统计维度不同，会有一定差异，query级别监控的语句实际下盘文件大小，算子监控的是具体算子在逻辑层IO读写的数据量。
- 当GUC参数enable_stream_operator设置为off状态时，算子执行信息存在显示不准的情况。

前提条件

- GUC参数**enable_resource_track**为on（默认为on）。
- GUC参数**resource_track_level**为query、perf、operator_realtime或operator（9.1.2及以上集群版本中默认为perf）。
- 监控作业的类型为：优化器估算的执行代价大于或等于**resource_track_cost**取值的作业。
- Cgroups功能正常加载，可通过gs_cgroup -P查看控制组信息。
- GUC参数**enable_track_record_subsql**控制是否记录存储过程、匿名块内部语句。

在上述条件中，**enable_resource_track**为系统级参数，用于设置是否开启资源监控功能。**resource_track_level**为session级参数，可以对某个session的资源监控级别进行灵活设置。这两个参数的设置方法如下表：

表 13-2 设置资源监控信息统计级别

enable_resource_track	resource_track_level	query级别信息	算子级别信息
on (default)	none	不统计	不统计

版权所有 © 华为云计算技术有限公司

enable_resource_track	resource_track_level	query级别信息	算子级别信息
	query	统计	不统计
	perf (default)	统计	不统计
	operator	统计	统计
	operator_realtime	统计	统计实时算子监控
off	none/query/ operator	不统计	不统计

操作步骤

步骤1 通过视图gs_session_cpu_statistics查询实时CPU信息。

```
SELECT * FROM gs_session_cpu_statistics;
```

步骤2 通过视图gs_session_memory_statistics查询实时memory信息。

```
SELECT * FROM gs_session_memory_statistics;
```

步骤3 通过视图gs_wlm_session_statistics查询当前CN的实时资源。

```
SELECT * FROM gs_wlm_session_statistics;
```

步骤4 通过视图pgxc_wlm_session_statistics查询所有CN的实时资源。

```
SELECT * FROM pgxc_wlm_session_statistics;
```

步骤5 通过视图gs_wlm_operator_statistics查询当前CN作业算子执行实时资源信息。

```
SELECT * FROM gs_wlm_operator_statistics;
```

步骤6 通过视图pgxc_wlm_operator_statistics查询所有CN作业算子执行实时资源信息。

```
SELECT * FROM pgxc_wlm_operator_statistics;
```

步骤7 通过视图pg_session_wlmstat查询当前用户执行作业正在运行时的负载管理信息。

```
SELECT * FROM pg_session_wlmstat;
```

步骤8 通过视图pgxc_wlm_workload_records（动态负载功能开启，即enable_dynamic_workload为on时该视图有效）查询当前用户在每个CN上作业执行时的状态信息。

```
SELECT * FROM pgxc_wlm_workload_records;
```

----结束

13.6 历史 TopSQL

系统提供了资源监控历史视图用来查询历史TopSQL。资源监控历史视图记录了查询作业运行结束时的资源使用情况(包括内存、下盘、CPU时间等)和运行状态信息(包括报错、终止、异常等)以及性能告警信息。但对于由于FATAL、PANIC错误导致查询异常结束时，状态信息列只显示aborted，无法记录详细异常信息。特别的，对于查询解析，优化阶段的状态信息则无法监控。

历史视图具体的对外接口如下表所示：

 版权所有 © 华为云计算技术有限公司

视图级别	节点范围	查询视图	
query级别/perf级别（推荐使用）	当前CN	历史（内部转储接口，仅显示最近三分钟内结束的语句）	GS_WLM_SESSION_HISTORY
		历史（全部语句）	GS_WLM_SESSION_INFO
	所有CN	历史（内部转储接口，仅显示最近三分钟内结束的语句）	PGXC_WLM_SESSION_HISTORY
		历史（全部语句）	PGXC_WLM_SESSION_INFO
		历史（全部语句，该视图显示经典字段信息）	PGXC_QUERY_INFO
operator级别	当前CN	历史（仅显示最近三分钟内结束的语句）	GS_WLM_OPERATOR_HISTORY
		历史（内部转储接口，全部语句）	GS_WLM_OPERATOR_INFO
	所有CN	历史（仅显示最近三分钟内结束的语句）	PGXC_WLM_OPERATOR_HISTORY
		历史（内部转储接口，全部语句）	PGXC_WLM_OPERATOR_INFO

约束和限制

- 视图级别取决于资源监控的等级，即参数**resource_track_level**的配置。
- perf和operator级别会影响**GS_WLM_SESSION_STATISTICS**/**PGXC_WLM_SESSION_INFO**中的query_plan和warning字段的取值，详细内容参见**SQL自诊断**。
- 9.1.2及以上集群版本中，当子语句使用游标且未一次性获取全部结果时，因当前系统限制，perf级别资源监控记录的执行计划可能无法完整显示实际执行时间、实际执行行数等信息。
- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 实例故障时，历史TopSQL视图有可能记录不全。
- 在某些异常的情况下，历史TopSQL中的状态信息列可能会显示为unknown，其记录的监控信息会导致不准确。
- 历史TopSQL能够记录的SQL语句的规格与实时TopSQL能够记录的SQL语句的规格一致。请参考**实时TopSQL中能够记录的SQL语句的规格**。
- 历史TopSQL只有在GUC参数**enable_resource_record**开启时才会记录数据。
- 查询历史TopSQL Query以及算子级别数据时，仅能通过postgres数据库进行访问。
- 历史TopSQL侧重于查询性能的定位定界辅助分析，不作为审计功能使用，不记录语法分析报错类语句。

版权所有 © 华为云计算技术有限公司

- 8.2.1及以上集群版本，新增GUC参数**resource_track_subsql_duration**，用于过滤存储过程中执行时间小于该参数的子语句，仅归档执行时间大于该参数的子语句。同时将GUC参数**enable_track_record_subsql**默认值由off变更为on，默认将会记录存储过程中的子语句。9.1.2及以上集群版本，新增GUC参数**resource_track_subsql_max_depth**，用于控制嵌套子语句的最大记录层数。如果一条子语句被记录，那么它必然满足以下几个条件：
  - 语句所在的会话中，参数enable_track_record_subsql开启。
  - 该子语句一定要下推到DN端执行（为避免TopSQL记录过多的子语句，不下推到DN端执行的子语句会被过滤）。
  - 该子语句的执行时间超过所在会话中resource_track_subsql_duration参数值。
  - 该子语句的嵌套层数未超过所在会话中resource_track_subsql_max_depth参数设置的最大记录层数。
- 关于history视图，默认会查询最近3分钟内结束的语句，该视图内部实际查询的是表，属于因性能考虑而实现的临时视图，从8.1.3集群版本开始，TopSQL监控功能中的实时监控和归档功能已有大幅度提升，在性能问题查询中，不建议使用的history视图已删除。
- 从8.1.3集群版本开始，TopSQL实时监控功能几乎对语句性能无影响，完全可以将GUC参数resource_track_cost设置为0来监控所有语句的运行时的信息；而TopSQL历史监控中对语句归档的能力，对语句执行性能也无影响，但当TPS较高时，需要考虑以下两个因素：
  - 记录所有语句的磁盘开销，用户可以将一条语句归档所需要的磁盘空间估算为8KB，按照业务TPS的峰值来计算相应的空间占用情况，来合理调整resource_track_duration和resource_track_subsql_duration参数。
  - 缓存所有语句的内存开销，用户可以将一条语句归档所需要的内存大小估算为16KB，语句批量归档的间隙为5秒，按照业务TPS的峰值来计算所需的内存峰值，计算方法为：5秒*TPS*16KB，需要GUC参数**session_history_memory**（9.1.2及以上集群版本中默认值为500MB）大于所需值，才能保证所有语句都能被记录。
  - 历史TopSQL视图记录所有DDL语句。

前提条件

- GUC参数**enable_resource_track**为on（默认为on）。
- GUC参数**resource_track_level**为query、perf、operator_realtime或operator（9.1.2及以上集群版本中默认为perf）。设置方法详见**表13-2**。
- GUC参数**enable_resource_record**为on（默认为on）。
- GUC参数**resource_track_duration**小于作业执行时间和排队时间之和（9.1.2及以上集群版本中默认为0）。
- GUC参数**enable_track_record_subsql**控制是否记录存储过程、匿名块内部语句（默认为on）。
- GUC参数**resource_track_subsql_duration**小于存储过程中内部语句的执行时间（9.1.2及以上集群版本中默认为100ms）。
- GUC参数**resource_track_application**指定TopSQL记录特定客户端下发的语句。
- 监控作业类型为：资源监控实时视图（参见**表13-1**）中记录的作业执行时间和排队时间之和大于或等于**resource_track_duration**的作业，且优化器估算的执行代价大于或等于**resource_track_cost**的作业。

版权所有 © 华为云计算技术有限公司

- Cgroups功能正常加载，可通过gs_cgroup -P查看控制组信息。

操作步骤

步骤1 通过视图gs_wlm_session_history查询当前CN最近执行作业结束后的负载记录。

```
SELECT * FROM gs_wlm_session_history;
```

步骤2 通过视图pgxc_wlm_session_history查询所有CN最近执行作业结束后的负载记录。

```
SELECT * FROM pgxc_wlm_session_history;
```

步骤3 通过视图pgxc_query_info查询所有CN的作业执行结束后的负载记录，要查到历史记录，必须保证**enable_resource_record**为on。

```
SELECT * FROM pgxc_query_info;
```

步骤4 通过数据表gs_wlm_session_info查询当前CN作业执行结束后的负载记录。要查到历史记录，必须保证**enable_resource_record**为on。

```
SELECT * FROM gs_wlm_session_info;
```

- 消耗内存最多的10个Query（可指定查询时间段）。

```
SELECT * FROM gs_wlm_session_info order by max_peak_memory desc limit 10;
SELECT * FROM gs_wlm_session_info WHERE start_time >= '2022-05-15 21:00:00' and finish_time <='2022-05-15 23:30:00' order by max_peak_memory desc limit 10;
```

- 消耗CPU最多的10个Query。

```
SELECT * FROM gs_wlm_session_info order by total_cpu_time desc limit 10;
SELECT * FROM gs_wlm_session_info WHERE start_time >= '2022-05-15 21:00:00' and finish_time <='2022-05-15 23:30:00' order by total_cpu_time desc limit 10;
```

步骤5 通过视图pgxc_wlm_session_info查询所有CN的作业执行结束后的负载记录。要查到历史记录，必须保证**enable_resource_record**为on。

```
SELECT * FROM pgxc_wlm_session_info;
```

- 查询所有CN消耗时间最多的10个query

  ```
  SELECT * FROM pgxc_wlm_session_info order by duration desc limit 10;
  ```

- 查询已经完成执行的query语句的执行信息。如：查询queryid为7656119369502647 8的语句执行信息。

  ```
  SELECT * FROM pgxc_wlm_session_info where queryid = '76561193695026478';
  ```

步骤6 通过函数pgxc_get_wlm_session_info_bytime对视图pgxc_wlm_session_info进行筛选查询，要查到历史记录，必须保证**enable_resource_record**为on。在统计数据量很大的场景中，建议使用该函数进行查询。

说明

DWS集群默认使用时区为UTC时间，与系统时间存在8h时差，请确保数据库时间与系统时间一致后进行以下查询。

- 查询所有CN上开始时间介于“2019-09-10 15:30:00”和“2019-09-10 15:35:00”之间的query，每个CN最多返回10条记录

```
SELECT * FROM pgxc_get_wlm_session_info_bytime('start_time', '2019-09-10 15:30:00', '2019-09-10 15:35:00', 10);
```

- 查询所有CN上结束时间介于“2019-09-10 15:30:00”和“2019-09-10 15:35:00”之间的query，每个CN最多返回10条记录

```
SELECT * FROM pgxc_get_wlm_session_info_bytime('finish_time', '2019-09-10 15:30:00', '2019-09-10 15:35:00', 10);
```

步骤7 通过视图gs_wlm_operator_history查询当前CN作业算子最近执行资源信息。要查到记录，必须保证**resource_track_level**为operator。

```
SELECT * FROM gs_wlm_operator_history;
```


版权所有 © 华为云计算技术有限公司

步骤8 通过视图pgxc_wlm_operator_history查询所有CN作业算子最近执行资源信息。要查到记录，必须保证**resource_track_level**为operator。

```
SELECT * FROM pgxc_wlm_operator_history;
```

步骤9 通过数据表gs_wlm_operator_info查询当前CN作业算子历史执行资源信息。要查到记录，必须保证**resource_track_level**为operator和**enable_resource_record**为on。

```
SELECT * FROM gs_wlm_operator_info;
```

步骤10 通过视图pgxc_wlm_operator_info查询所有CN作业算子历史执行资源信息。要查到记录，必须保证**resource_track_level**为operator和**enable_resource_record**为on。

```
SELECT * FROM pgxc_wlm_operator_info;
```

----结束

说明

- 对于上述的视图信息，由于预设内存的限制，内存中能够保留的数据记录数量有限，实时查询在结束后会导入到历史相关的视图中。关于记录上限，对于query级别视图，当新的需要记录的查询超过内存约束记录数上限时，则当前查询无法记录，下条查询重新进行规则判断；在每个CN上，记query级别历史视图的内存占用（默认100MB）可通过**PG_TOTAL_MEMORY_DETAIL**视图进行查询。
- 对于算子级别视图，当需要记录的查询的plan_node数量加上当前内存中已有的记录数量超过内存约束记录数上限时，则当前查询的所有算子节点不记录，下条查询重新按照算子规则判定。在每个CN上，算子级别视图在内存中可记录的最大实时和历史记录数分别为max_oper_realt_num(当前系统值为56987)，max_oper_hist_num(113975)；当前用户业务系统的平均每个查询的节点数为num_plan_node，则在每个CN上，实时视图允许客户执行的最大并发数：num_realt_active = max_oper_realt_num/num_plan_node；历史视图允许客户执行的最大并发数：num_hist_active = max_oper_hist_num/(180/run_time)/num_plan_node。
- 如果并发过高，避免需要记录的查询数量超过query级别视图和算子级别视图的值，可以通过参数**session_history_memory**修改历史查询视图的内存，内存增大和查询数量成正比。

13.7 TopSQL 查询示例

本章节以查询TPC-DS样例数据的作业为例，演示如何查看**实时TopSQL**和**历史TopSQL**。

配置集群参数

查询TopSQL资源监控信息之前，需要先配置相关的GUC参数，以便能查询到作业的资源监控历史信息或归档信息。步骤如下：

1. 登录**DWS控制台**。
2. 在左侧导航栏中，单击**“集群 > 集群列表”**。
3. 在集群列表中找到所需要的集群，单击集群名称，进入**“集群详情”**页面。
4. 单击“参数修改”页签，并在“参数列表”修改参数**resource_track_duration**为合适的值，然后单击**“保存”**。
5. 在**“修改预览”**窗口，确认修改无误后，单击**“保存”**。

TopSQL 查询示例

本示例以TPC-DS样例数据为例。

步骤1 打开SQL客户端工具，连接到您的数据库。

 版权所有 © 华为云计算技术有限公司

步骤2 使用explain语句查询所要执行的SQL语句的预估代价，从而可以确定该SQL语句是否会进行资源监控。

默认执行代价大于**resource_track_cost**的查询才会进行资源监控，用户才可以查询到相关的资源监控信息。

例如，执行如下语句查询该SQL语句的预估执行代价：

```
SET CURRENT_SCHEMA = tpcds;
EXPLAIN WITH customer_total_return AS
( SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
FROM store_returns, date_dim
WHERE sr_returned_date_sk = d_date_sk AND d_year =2000
GROUP BY sr_customer_sk, sr_store_sk )
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

查询结果如下所示，第一行的E-costs列的值即为当前语句的预估代价。

图 13-1 explain 查询结果

```
 id |                                operation                                | E-rows | E-width | E-costs
----+-------------------------------------------------------------------------+--------+---------+---------
  1 | ->  Row Adapter                                                         |      6 |      20 | 153.06
  2 |    ->  Vector Limit                                                     |      6 |      20 | 153.06
  3 |       ->  Vector Streaming (type: GATHER)                               |      6 |      20 | 153.06
  4 |          ->  Vector Limit                                               |      6 |      20 | 152.84
  5 |             ->  Vector Sort                                             |      6 |      20 | 152.84
  6 |                ->  Vector Hash Join (7,26)                              |      6 |      20 | 152.83
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |      6 |       4 | 134.57
  8 |                      ->  Vector Hash Join (9,18)                        |      6 |       4 | 134.46
  9 |                         ->  Vector Hash Join (10,11)                    |      1 |      44 | 97.33
 10 |                            ->  CStore Scan on store                     |      1 |       4 | 60.23
 11 |                            ->  Vector Subquery Scan on ctr1             |      6 |      40 | 37.07
 12 |                               ->  Vector Hash Aggregate                 |      6 |      54 | 37.06
 13 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
 14 |                                     ->  Vector Hash Join (15,17)        |      6 |      22 | 37.00
 15 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
 16 |                                           ->  CStore Scan on date_dim   |      1 |       4 | 18.06
 17 |                                        ->  CStore Scan on store_returns |     60 |      26 | 18.02
 18 |                         ->  Vector Hash Aggregate                       |      6 |      68 | 37.09
 19 |                            ->  Vector Subquery Scan on ctr2             |      6 |      36 | 37.07
 20 |                               ->  Vector Hash Aggregate                 |      6 |      54 | 37.06
 21 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
 22 |                                     ->  Vector Hash Join (23,25)        |      6 |      22 | 37.00
 23 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
 24 |                                           ->  CStore Scan on date_dim   |      1 |       4 | 18.06
 25 |                                        ->  CStore Scan on store_returns |     60 |      26 | 18.02
 26 |                   ->  CStore Scan on customer                           |     60 |      24 | 18.02
(26 rows)
```

本示例为了演示TopSQL的资源监控功能，需要将resource_track_cost参数设置为比explain查询结果中的预估代价小的一个值，即将resource_track_cost参数设置为100，设置方法请参见**resource_track_cost**。

说明

在完成本示例后，仍然要将**resource_track_cost**设置为原始的默认值100000或者一个比较合理的值，否则参数值太小会影响数据库性能。

步骤3 执行SQL语句。

```
SET CURRENT_SCHEMA = tpcds;
WITH customer_total_return AS
(SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
```

 版权所有 © 华为云计算技术有限公司

```
FROM store_returns,date_dim
WHERE sr_returned_date_sk = d_date_sk
AND d_year =2000
GROUP BY sr_customer_sk ,sr_store_sk)
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

步骤4 在SQL语句执行期间，查询该条SQL语句在当前CN上的Memory峰值的实时信息。

```
SELECT query,max_peak_memory,average_peak_memory,memory_skew_percent FROM
gs_wlm_session_statistics ORDER BY start_time DESC;
```

含义：查询**query级别**的SQL语句的Memory峰值**实时信息**（语句在所有DN上的每秒最大Memory峰值，在所有DN上的每秒平均Memory峰值，在DN间的Memory倾斜率）

实时TopSQL资源监控信息的更多查询示例，请参见**实时TopSQL**。

步骤5 等待**步骤3**中的SQL执行完成，然后查询该语句执行期间的资源监控历史信息。

```
SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_history ORDER BY start_time
desc;
```

含义：查询**query级别**的SQL语句执行期间的**历史信息**（语句执行的开始时间，结束时间，实际执行时间，执行状态），时间单位为ms。

历史TopSQL资源监控信息的更多查询示例，请参见**历史TopSQL**。

步骤6 等待**步骤3**中的SQL执行结束的三分钟后，在info视图中查询该语句的资源监控历史信息。

如果设置参数enable_resource_record为“on”，且**步骤3**中SQL的执行时间不小于resource_track_duration所设置的值，该条语句的历史信息将会在三分钟后被归档到gs_wlm_session_info视图中。

对于info视图，只支持在连接postgres数据库时查询。因此，请切换为连接postgres数据库后，再执行以下语句进行查询：

```
SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_info ORDER BY start_time desc;
```

----结束

 版权所有 © 华为云计算技术有限公司

14 DWS 性能调优

14.1 性能调优概述

数据库性能调优是指通过优化数据库系统的配置及SQL查询，以提高数据库性能和效率的过程。目的为消除性能瓶颈、减少响应时间、提高系统吞吐量和资源利用率，降低业务成本，从而提高系统稳定性，给用户带来更大的价值。

本章通过性能诊断、系统调优及SQL调优及常见SQL调优案例等性能调优的实际操作，为数据库性能调优人员提供全方位的指导。

注意事项

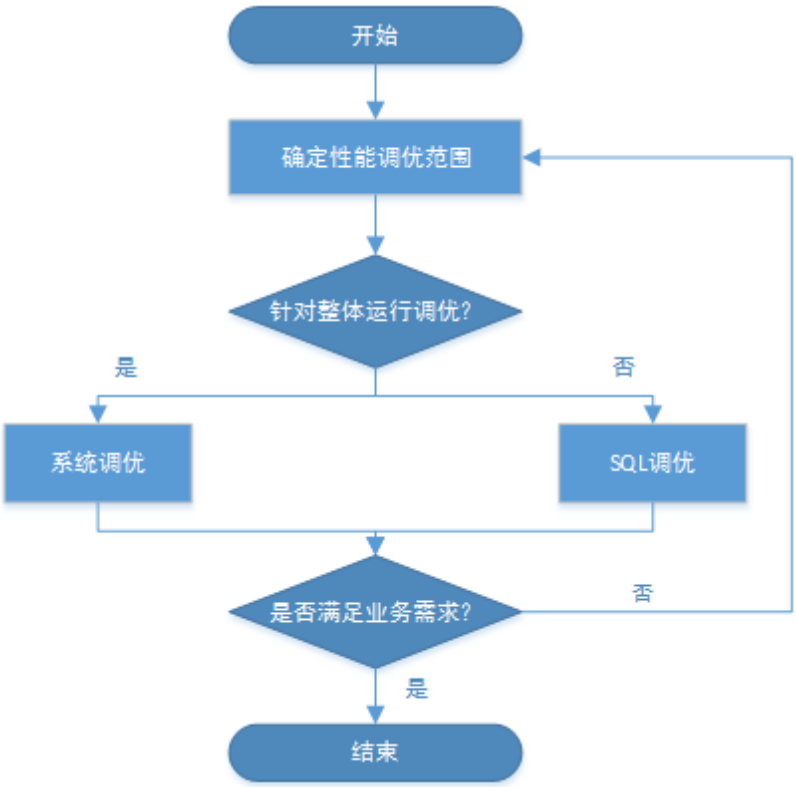
- 数据库调优是一个复杂和细致的过程，需熟悉数据库系统的内部工作原理和相关技术。它需要综合考虑硬件、软件、查询、配置和数据结构等多个方面的因素，以达到最佳的性能和效率。因此，要求调优人员应对系统软件架构、软硬件配置、数据库配置参数、并发控制、查询处理和数据库应用有广泛而深刻的理解。
- 性能调优过程有时需要重启集群，可能会中断当前业务。建议在业务低峰期进行需要重启集群的性能调优操作，避免业务异常中断。

调优流程

调优流程如图14-1所示。

版权所有 © 华为云计算技术有限公司

图 14-1 DWS 性能调优流程



调优各阶段说明，如表14-1所示。

表 14-1 DWS 性能调优流程说明

阶段	描述
性能诊断	获取集群各节点的CPU、内存、I/O和网络资源使用情况，确认这些资源是否已被充分利用，是否存在瓶颈点。
系统调优	进行操作系统级以及数据库系统级的调优，更充分地利用机器的CPU、内存、I/O和网络资源，避免资源冲突，提升整个系统查询的吞吐量。

版权所有 © 华为云计算技术有限公司

阶段	描述
SQL调优	审视业务所用SQL语句是否存在可优化空间，包括： • 通过ANALYZE语句生成表统计信息：ANALYZE语句可收集与数据库中表内容相关的统计信息，统计结果存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以确定最有效的执行计划。 • 分析执行计划：EXPLAIN语句可显示SQL语句的执行计划，EXPLAIN PERFORMANCE语句可显示SQL语句中各算子的执行时间。 • 查找问题根因并进行调优：通过分析执行计划，找到可能存在的原因，进行针对性的调优，通常为调整数据库级SQL调优参数。 • 编写更优的SQL：介绍一些复杂查询中的中间临时数据缓存、结果集缓存、结果集合并等场景中的更优SQL语法。

14.2 性能诊断

14.2.1 集群性能分析

DWS不同集群规格的CPU核数、内存大小和节点存储容量不同，处理业务能力和性能也就不同，用户在创建集群前需要结合实际业务量和具体使用场景来选择集群规格。

在使用集群过程中，当用户的业务量过大，则需要更多的资源（CPU、内存、网络带宽等）来支撑逐渐增长的业务量，如果用户当前使用的集群资源不足，会降低数据库运行速度并影响数据库性能。

DWS监控信息功能提供了丰富的监控指标，您可以通过集群的各项监控指标（CPU使用率、内存使用率、磁盘使用率、磁盘I/O、网络I/O等），掌握集群的性能和运行状况。当您发现监控指标存在异常时，可以通过查看集群监控指标排查出现异常的原因。

如果需要更多的计算资源或存储资源以满足业务需要时，可以在管理控制台对已有集群，进行规格变更或扩容操作。

14.2.2 慢 SQL 分析

14.2.2.1 查询最耗性能的 SQL

系统中有些SQL语句运行了很长时间仍未结束，这些语句会消耗大量系统资源，可参考本章内容查询长时间运行的SQL语句。

操作步骤

步骤1 设置参数track_activities为on。

```
SET track_activities = on;
```

当此参数为on时，数据库系统才会收集当前活动查询的运行信息。

 版权所有 © 华为云计算技术有限公司

步骤2 查询系统中长时间运行的查询语句。

```
SELECT current_timestamp - query_start AS runtime, datname, usename, query FROM pgxc_stat_activity
WHERE state != 'idle' ORDER BY 1 desc;
```

查询后会按执行时间从长到短顺序返回查询语句列表，第一条结果就是当前系统中执行时间最长的查询语句。返回结果中包含了系统调用的SQL语句和用户执行SQL语句，请根据实际返回结果中query字段对应的信息，查找到用户执行时间长的语句。

若当前系统较为繁忙，可以通过限制current_timestamp - query_start大于某一阈值查看执行时间超过此阈值的查询语句。

```
SELECT query FROM pgxc_stat_activity WHERE current_timestamp - query_start > interval '1 days';
```

步骤3 查看正在运行的查询语句。

以查看视图pgxc_stat_activity为例：

```
SELECT datname, usename, state FROM pgxc_stat_activity;
 datname  | usename | state  |
----------+---------+--------+
 postgres |   omm   | idle   |
 postgres |   omm   | active |
(2 rows)
```

如果state字段显示为idle，则表明此连接处于空闲，等待用户输入命令。

如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pgxc_stat_activity WHERE state != 'idle';
```

步骤4 分析长时间运行的查询语句状态。

- 若查询语句处于正常状态，则等待其执行完毕。
- 若查询语句阻塞，则通过如下命令查看当前处于阻塞状态的查询语句：
  ```
  SELECT datname, usename, state, query FROM pgxc_stat_activity WHERE waiting = true;
  ```
  查询结果中包含了当前被阻塞的查询语句，该查询语句所请求的锁资源可能被其他会话持有，正在等待持有会话释放锁资源。

  说明

  只有当查询阻塞在系统内部锁资源时，waiting字段才显示为true。尽管等待锁资源是数据库系统最常见的阻塞行为，但是在某些场景下查询也会阻塞在等待其他系统资源上，例如写文件、定时器等。但是这种情况的查询阻塞，不会在视图pgxc_stat_activity中体现。

----结束

14.2.2.2 分析作业是否被阻塞

数据库系统运行时，在某些业务场景下查询语句会被阻塞，导致语句运行时间过长，可以强制结束有问题的会话。

操作步骤

步骤1 查看阻塞的查询语句及阻塞查询的表、模式信息。

```
SELECT w.query as waiting_query,
w.coorname as w_cn,
w.pid as w_pid,
w.usename as w_user,
l.query as locking_query,
l.pid as l_pid,
l.usename as l_user,
t.schemaname || '.' || t.relname as tablename
from pgxc_stat_activity w join pg_locks l1 on w.pid = l1.pid
```


版权所有 © 华为云计算技术有限公司

```
and not l1.granted join pg_locks l2 on l1.relation = l2.relation
and l2.granted join pgxc_stat_activity l on l2.pid = l.pid join pg_stat_user_tables t on l1.relation = t.relid
where w.waiting;
```

该查询返回线程ID、CN名称、用户信息、查询状态，以及导致阻塞的表、模式信息。

步骤2 使用如下命令结束相应的会话。其中，w_cn、w_pid分别为上面查到的对应CN名称、线程ID。

```
EXECUTE DIRECT ON(w_cn) 'SELECT pg_terminate_backend(w_pid)';
```

显示类似如下信息，表示结束会话成功。

```
 PG_TERMINATE_BACKEND
----------------------
 t
(1 row)
```

显示类似如下信息，表示用户正在尝试结束当前会话，此时仅会重连会话，而不是结束会话。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

说明

gsql客户端使用PG_TERMINATE_BACKEND函数终止本会话后台线程时，客户端不会退出而是自动重连。

----结束

14.2.3 SQL 诊断

DWS集群提供了SQL诊断功能，支持通过多种条件检索出符合条件的SQL查询（如慢查询），并完整展示执行计划。

SQL诊断功能使用方式：

步骤1 登录DWS管理控制台。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的“**操作**”列，单击“**监控面板**”，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择“**工具 > SQL诊断**”，进入SQL诊断页面。

步骤5 在SQL诊断页面您可查看SQL的诊断信息。指标详情请参见表14-2。

表 14-2 SQL 诊断指标说明

指标名称	指标说明
查询ID	语句执行使用的内部query_ID。
数据库	所属数据库名称。
模式名	所属模式名称。
用户名称	所属用户名称。
客户端	连接到后端的应用名。

版权所有 © 华为云计算技术有限公司

指标名称	指标说明
客户端IP地址	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者是内部进程，如autovacuum。
运行时间（ms）	语句实际执行的时间，单位ms。
CPU时间（ms）	语句在所有DN上的CPU总时间，单位ms。
开始时间	语句执行开始时间。
完成时间	语句执行结束时间。
详情	单击“**查看**”键，可浏览到SQL的详细诊断结果。其中包括： • 告警信息：该SQL语句的告警信息。 • SQL语句：该SQL语句。 • 执行计划：该SQL语句的执行计划分析。

----结束

14.2.4 表诊断

DWS提供了集群中数据表关键运行状态的统计数据与诊断工具。其中包括：

- 表倾斜率：对于集群中数据表统计信息进行监控分析，展示倾斜率高于5%且表大小TOP50的表信息。
- 表脏页率：对于集群中数据表统计信息进行监控分析，展示脏页率高于50%且表大小TOP50的表信息。

表倾斜率

不合理的分布列选择，将引发算子计算/数据下盘倾斜严重，导致不同DN的处理压力不同，影响业务性能，并容易造成单DN磁盘使用率过高。用户可通过查询表倾斜率，根据表的大小和倾斜率，对倾斜严重的表重新选择分布列。参见**如何调整分布列？** 。

操作步骤

步骤1 **登录DWS管理控制台**。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的“**操作**”列，单击“**监控面板**”，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择“**工具 > 表诊断 > 表倾斜率**”，页面将展示集群中符合统计条件的表信息（倾斜率高于5%且表大小Top50的表数据）。

该界面表倾斜率的算法和后台的算法不同，该算法为：**表倾斜率 =（表在各DN占空间的最大值-表在各DN占空间的最小值）/ 表在各DN占空间的平均值**。

其中该表在各DN占空间的最大值/最小值/平均值是通过查询视图**gs_table_distribution**的“**dnsize**”（表在该节点上的存储空间大小）列得出。

版权所有 © 华为云计算技术有限公司

步骤5 （可选）在表倾斜率页面单击**“一键导出”**，将会导出集群中符合统计条件的表信息到CSV文件，可直接在本地打开查看。

----结束

表脏页率

对于数据表的DML操作将影响数据表数据导致存在无用的脏数据， 过多的脏数据将占用磁盘空间，影响集群可用容量。用户可通过查询表的脏页率，根据表的大小和脏页率，对较大表和脏页率过高的表进行处理， 处理方式参考**磁盘使用率高&集群只读处理方案**。

操作步骤

步骤1 **登录DWS管理控制台**。

步骤2 找到需要查看监控的集群。

步骤3 在指定集群所在行的**“操作”**列，单击**“监控面板”**，系统将显示数据库监控页面。

步骤4 在左侧导航栏选择**“工具 > 表诊断 > 表脏页率”**，页面将展示集群中符合统计条件的表信息。

步骤5 （可选）在表脏页率页面单击**“一键导出”**，将会导出集群中符合统计条件的表信息到CSV文件，可直接在本地打开查看。

----结束

14.3 系统调优

14.3.1 数据库系统参数调优

为了保证数据库尽可能高性能地运行，建议依据资源情况和业务实际进行数据库系统GUC参数的设置。本章节旨在介绍一些常用参数以及推荐配置，关于参数的详细设置方法请参考**设置GUC参数**。

 版权所有 © 华为云计算技术有限公司

数据库内存相关参数

表 14-3 数据库内存相关参数

GUC参数	描述	建议
max_process_memory	max_process_memory设置单个CN/DN可用的最大物理内存。	• DN上该数值需要根据系统物理内存及单节点部署主DN个数决定的。计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.8 / (n+主DN个数)。该参数目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min_free_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即， max_process_memory=物理内存*0.8/(n+主DN个数)，其中，当集群规模小于256时，n=1；当集群规模大于256且小于512时， n=2；当集群规模超过512时，n=3。 • CN上该数值内存可设置与DN数值一样。 • RAM：集群规划时分配给集群的最大使用内存。
shared_buffers	设置DWS使用的共享内存大小。增加此参数值会使DWS比系统默认设置需要更多的System V共享内存。	建议设置shared_buffers值为内存的40%以内。行存列存分别进行设置。行存设大，列存设小。列存：(单服务器内存/单服务器DN个数)*0.4*0.25。 如果设置较大的shared_buffers需要同时增加checkpoint_segments的值，因为写入大量新增、修改数据需要消耗更多的时间周期。
cstore_buffers	设置列存和OBS、HDFS外表列存格式（orc、parquet、carbondata）所使用的共享缓冲区的大小。	列存表使用cstore_buffers设置的共享缓冲区，几乎不用shared_buffers。因此在列存表为主的场景中，应减少shared_buffers，增加cstore_buffers。 OBS、HDFS外表使用cstore_buffers设置ORC、Parquet、Carbondata的元数据和数据的缓存，元数据缓存大小为cstore_buffers的1/4，最大不超过2GB，其余缓存空间为列存数据和外表列存格式数据共享使用。

 版权所有 © 华为云计算技术有限公司

GUC参数	描述	建议
work_mem	设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。	该参数默认小规格内存为512MB，大规格内存为2GB（max_process_memory大于等于30GB为大规格内存，否则为小规格内存）。 建议参数work_mem依据查询特点和并发来确定，一旦work_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。 • 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work_mem=50%内存/10。 • 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work_mem=50%内存/5。 • 对于并发场景，建议work_mem=串行下的work_mem/物理并发数。
maintenance_work_mem	设置维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。	建议设置此参数的值等于work_mem，可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。 当自动清理进程运行时，autovacuum_max_workers倍数的内存将会被分配，所以此时设置maintenance_work_mem的值应该不小于work_mem。

 版权所有 © 华为云计算技术有限公司

数据库并发队列相关参数

GUC参数	描述	建议
max_active_statements（全局并发队列）	控制单个CN上运行并发执行的作业数量。	采用全局并发队列机制将控制所有普通用户的执行作业，不区分复杂度，即执行语句都将作为一个执行单元，当并发执行的作业数量达到此参数阈值时，将进入队列等待。对于管理员执行的作业，不受全局并发控制的限制。 需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持最大限度的并发作业，且防止并发执行作业过多，引起系统崩溃。
parctl_min_cost（局部并发队列）	控制在一个CN上同一资源池内的并发作业数量。该GUC参数需联系技术支持设置。	局部并发控制机制根据执行作业的cost，控制复杂查询的并发作业数量。

说明

全局并发队列参数max_active_statements调整需注意：

- 如果参数max_active_statements设置为-1，也就是不限制全局并发数，会导致通信大并发下存在连接断开报错。
- 在点查询的场景下，参数max_active_statements建议设置为100。
- 在分析类查询的场景下，参数max_active_statements的值设置为CPU的核数除以DN个数，一般可以设置5~8个。

数据库通信相关参数

数据库集群默认使用TCP代理通信库进行节点间通信。

表 14-4 数据库通信相关参数

GUC参数	描述	建议
comm_quota_size	参数comm_quota_size控制每个流通道每次数据传输的大小，默认数值1M。	大并发场景下，提升该数值时，可提升通信性能，但消耗更多的内存，需要根据实际场景进行调优。若通过查询DN的视图pg_total_memory_detail，发现通信层使用内存已达参数comm_usable_memory的阈值时，需要减少该数值，如修改为512K。

版权所有 © 华为云计算技术有限公司

GUC参数	描述	建议
comm_usable_memory	参数comm_usable_memory用于控制数据库传输时，整个DN上可以用于通信的内存使用量。	该参数的取值并不是用于总量控制，仅作为内存流控的手段，即默认流控数值为1M，若内存使用量超过了该参数取值的1/2时，流控数值自动调整为默认流控的一半，若仅剩20%时，则流控数值改为最小流控值8K。

数据库连接相关参数

表 14-5 数据库连接相关参数

GUC参数	描述	建议
max_connections	允许和数据库连接的最大并发连接数。此参数会影响集群的并发能力。	CN中此参数建议保持默认值。DN中此参数建议设置为CN的个数乘以CN中此参数的值。 增大这个参数可能导致DWS要求更多的System V共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。
max_prepared_transactions	设置可以同时处于“预备”状态的事务的最大数目。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。	max_connections取值的设置受max_prepared_transactions的影响，在设置max_connections之前，应确保max_prepared_transactions的值大于或等于max_connections的值，这样可确保每个会话都有一个等待中的预备事务。
session_timeout	客户端连接数据库后处于空闲状态时会根据该参数的默认值自动断开连接。	0-86400，最小单位为秒（s），0表示关闭超时设置。一般不建议设置为0。

版权所有 © 华为云计算技术有限公司

其他性能相关参数

表 14-6 其他性能相关参数

GUC参数	描述	建议
enable_dynamic_workload	设置是否开启动态负载管理。 动态负载管理指数据库内部根据用户负载情况，自动对复杂查询进行队列控制，不再需要手动设置参数，做到系统参数免调优。	该参数默认打开，需要注意以下几点： ● 简单查询作业（估算值<32MB）、非DML（即非INSERT、UPDATE、DELETE和SELECT）语句，不走自适应负载，需要通过max_active_statements来进行单CN的上限控制。 ● 在自适应负载特性下，参数work_mem的默认数值不能变大，否则会引起内存不受控（例如未做Analyze的语句）。 ● 以下场景或语句由于内存使用的特殊性和不确定性，可能导致大并发场景内存不受控，遇到需要降低并发数： – 单条元组占用内存过大，例如，基表包含超过MB级别的宽列。 – 完全下推语句的查询。 – 需要在CN上耗费大量内存的语句，例如，不能下推的语句、withhold cursor场景。 – 由于计划生成不当导致hashjoin算子建立的hash表重复值过多，占用大量内存。 – 包含UDF，且UDF中使用大量内存。 ● 该参数可配合query_dop=0使用，当query_dop设置为0（自适应），系统会根据资源情况和计划特征，动态为每个查询选取最优并行度。enable_dynamic_workload参数会动态分配内存。
bulk_write_ring_size	数据并行导入使用的环形缓冲区大小。	该参数主要影响入库性能，建议导入压力大的场景增加DN上的该参数配置。 默认值为2GB。
data_replicate_buffer_size	发送端与接收端传递数据页时，队列占用内存的大小。	该参数会影响主备之间复制的缓冲大小。 CN默认值为16MB，DN默认值为128MB。若服务器内存为256GB，可适当增大到512MB。

版权所有 © 华为云计算技术有限公司

14.3.2 SMP 并行执行

在复杂查询场景中，单个查询的执行较长，系统并发度低，通过SMP并行执行技术实现算子级的并行，能够有效减少查询执行时间，提升查询性能及资源利用率。

通过算子并行来提升性能，同时会占用更多的系统资源，包括CPU、内存、网络、I/O等等。本质上SMP是一种以资源换取时间的方式，在合适的场景以及资源充足的情况下，能够起到较好的性能提升效果；但是如果在不合适的场景下，或者资源不足的情况下，反而可能引起性能的劣化。同时，生成SMP需要考虑更多的候选计划，将会导致生成时间较长，相比串行场景也会引起性能的劣化。

DWS的SMP特性由GUC参数query_dop控制，该参数可设置用户自定义的查询并行度。

SMP 适用场景与限制

SMP适用场景：

- 支持并行的算子

  计划中存在以下算子支持并行：

  a. Scan：支持行存普通表和行存分区表顺序扫描、列存普通表和列存分区表顺序扫描、HDFS内外表顺序扫描；支持GDS数据导入的外表扫描并行。以上均不支持复制表。

  b. Join：HashJoin、NestLoop

  c. Agg：HashAgg、SortAgg、PlainAgg、WindowAgg(只支持partition by，不支持order by)

  d. Stream：Redistribute、Broadcast

  e. 其他：Result、Subqueryscan、Unique、Material、Setop、Append、VectoRow、RowToVec

- SMP特有算子

  为了实现并行，新增了并行线程间的数据交换Stream算子供SMP特性使用。以下新增的算子可视为Stream算子的子类：

  a. Local Gather：实现DN内部并行线程的数据汇总。

  b. Local Redistribute：在DN内部各线程之间，按照分布键进行数据重分布。

  c. Local Broadcast：将数据广播到DN内部的每个线程。

  d. Local RoundRobin：在DN内部各线程之间实现数据轮询分发。

  e. Split Redistribute：在集群跨DN的并行线程之间实现数据重分布。

  f. Split Broadcast：将数据广播到集群所有DN的并行线程。

  上述新增算子可以分为Local与非Local两类，Local类算子实现了DN内部并行线程间的数据交换，而非Local类算子实现了跨DN的并行线程间的数据交换。

- 示例说明

  以TPCH Q1的并行计划为例：

 版权所有 © 华为云计算技术有限公司

```
 id |                                operation                                 |
----+--------------------------------------------------------------------------+
  1 | ->  Row Adapter                                                          |
  2 |    ->  Vector Streaming (type: GATHER)                                   |
  3 |       ->  Vector Sort                                                    |
  4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/4)               |
  5 |             ->  Vector Hash Aggregate                                    |
  6 |                ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 4/4)   |
  7 |                   ->  Vector Hash Aggregate                              |
  8 |                      ->  Vector Append(9, 10)                            |
  9 |                         ->  Dfs Scan on lineitem                         |
 10 |                         ->  Vector Adapter                               |
 11 |                            ->  Seq Scan on pg_delta_1423863972 lineitem  |
(11 rows)
```

在这个计划中，实现了Hdfs Scan以及HashAgg算子的并行，并且新增了Local Gather和Split Redistribute数据交换算子。

其中6号算子为Split Redistribute算子，上面标有的“dop: 4/4”表明Split Redistribute的发送端和接收端线程的并行度均为4。4号算子为Local Gather，上面标有“dop: 1/4”，该算子的发送端线程并行度为4，而接收端线程并行度为1，即下层的5号Hash Aggregate算子按照4并行度执行，而上层的1~3号算子按照串行执行，4号算子实现了DN内并行线程的数据汇总。

通过计划Stream算子上标明的dop信息即可看出各个算子的并行情况。

非适用场景：

1. 生成计划时间占比很高的短查询场景。
2. 不支持CN上的算子并行。
3. 不支持不能下推的查询并行执行。
4. 不支持子查询subplan的并行，以及包含子查询的算子并行。

资源对 SMP 性能的影响

SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，包括CPU、内存、I/O和网络带宽等资源的消耗都会出现明显的增长，而且随着并行度的增大，资源消耗也随之增大。当上述资源成为瓶颈的情况下，SMP无法提升性能，反而可能导致集群整体性能的劣化。SMP支持自适应特性，该特性会根据当前资源和查询特征，动态选取最优的并行度。下面对各种资源对SMP性能的影响情况分别进行说明：

- **CPU资源**

  在一般客户场景中，系统CPU利用率不高的情况下，利用SMP并行架构能够更充分地利用系统CPU资源，提升系统性能。但当数据库服务器的CPU核数较少，CPU利用率已经比较高的情况下，如果打开SMP并行，不仅性能提升不明显，反而可能因为多线程间的资源竞争而导致性能劣化。

- **内存资源**

  查询并行后会导致内存使用量的增长，但每个算子使用内存上限仍受到work_mem等参数的限制。假设work_mem为4GB，并行度为2，那么每个并行线程所分配到的内存上限为2GB。在work_mem较小或者系统内存不充裕的情况下，使用SMP并行后，可能出现数据下盘，导致查询性能劣化的问题。

- **网络带宽资源**

  为了实现查询并行执行，会新增并行线程间的数据交换算子。对于Local类Stream算子，所需要进行数据交换的线程在同一个DN内，通过内存交换，不会增加网络

 版权所有 © 华为云计算技术有限公司

负担。而非Local类算子，需要通过网络进行数据交换，因此会加重网络负担。当网络资源成为瓶颈的情况下，并行可能会导致一定程度的劣化。

- **I/O资源**

  要实现并行扫描必定会增加I/O的资源消耗，因此只有在I/O资源充足的情况下，并行扫描才能够提高扫描性能。

其他因素对 SMP 性能的影响

除了资源因素外，还有一些因素也会对SMP并行性能造成影响。例如分区表中分区数据不均，以及系统并发度等因素。

- **数据倾斜对SMP性能的影响**

  当数据中存在严重数据倾斜时，并行效果较差。例如某表join列上某个值的数据量远大于其他值，开启并行后，根据join列的值对该表数据做hash重分布，使得某个并行线程的数据量远多于其他线程，造成长尾问题，导致并行后效果差。

- **系统并发度对SMP性能的影响**

  SMP特性会增加资源的使用，而在高并发场景下资源剩余较少。所以，如果在高并发场景下，开启SMP并行，会导致各查询之间严重的资源竞争问题。一旦出现了资源竞争的现象，无论是CPU、I/O、内存或者网络资源，都会导致整体性能的下降。因此在高并发场景下，开启SMP经常不能达到性能提升的效果，甚至可能引起性能劣化。

SMP 相关参数配置建议

如果要打开SMP自适应功能，要设置query_dop=0，需同步调整以下相关参数值，以获取更佳的dop选择：

- comm_usable_memory

  当系统内存较大时，max_process_memory设置较大，可适当调大该值，建议设置为max_process_memory的5%，默认值为max_process_memory/8。

- comm_max_stream

  设置建议值为：comm_max_stream=Min(dop_limit * dop_limit * 20 * 2, max_process_memory(字节数) * 0.025 / 总DN数 / 260)，且该值在comm_max_stream取值范围内。

- max_connections

  设置建议值为：max_connections=dop_limit * 20 * 6 + 24，且该值在max_connections取值范围内。

⚠ 注意

公式中的dop_limit为集群中每个DN对应的CPU数，计算公式为：dop_limit = 单机器的CPU逻辑核数 / 单机器的DN数。

版权所有 © 华为云计算技术有限公司

SMP 配置方式

须知

系统的CPU、内存、I/O和网络带宽等资源充足。SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，当上述资源成为瓶颈的情况下，SMP无法提升性能，反而可能导致性能的劣化。同时，SMP计划的生成时间较串行要长。因此，在短查询为主的TP类业务中，或者出现资源瓶颈的情况下，建议关闭SMP，即设置query_dop=1。

配置步骤：

1. 观察当前系统负载情况，如果系统资源充足（资源利用率小于50%），执行步骤2；否则退出。
2. 设置query_dop=1，利用explain打出执行计划，观察计划是否符合**SMP适用场景与限制**适用场景。如果符合，进入下一步。
3. 设置query_dop=-value，在考虑资源情况和计划特征基础上，限制dop选取的范围为[1,value]。
4. 设置query_dop=value，不考虑资源情况和计划特征，强制选取dop为1或value。
5. 在符合条件的查询语句执行前设置合适的query_dop值，在语句执行结束后关闭query_dop。例如，

```
SET query_dop = 0;
SELECT COUNT(*) FROM t1 GROUP BY a;
......
SET query_dop = 1;
```

说明

- 资源许可的情况下，并行度越高，性能提升效果越好。
- SMP并行度支持会话级设置，推荐客户在执行符合要求的查询前，打开smp，执行结束后，关闭smp。以免在业务峰值时，对业务造成冲击。
- SMP自适应（query_dop<=0）依赖资源管理，如果资源管理禁用，那么只会产生1或2并行度的计划。

14.3.3 配置 LLVM

LLVM（Low Level Virtual Machine）动态编译技术可以为每个查询生成定制化的机器码用于替换原本的通用函数。通过减少实际查询时冗余的条件逻辑判断、虚函数调用并提高数据局域性，从而达到提升查询整体性能的目的。

由于LLVM需要消耗额外的时间预生成IR中间态表示并编译成机器码，因此在小数据量场景或查询本身耗时较少时，可能引起性能的劣化。

LLVM 适用场景与限制

适用场景

- 支持LLVM的表达式。查询语句中存在以下的表达式支持LLVM优化：
  a. Case…when… 表达式
  b. In表达式
  c. Bool表达式 (And/Or/Not)

版权所有 © 华为云计算技术有限公司

d. BooleanTest表达式 (IS_NOT_KNOWN/IS_UNKNOWN/IS_TRUE/IS_NOT_TRUE/IS_FALSE/IS_NOT_FALSE)
e. NullTest表达式 (IS_NOT_NULL/IS_NULL)
f. Operator表达式
g. Function表达式 (lpad, substring, btrim, rtrim, length)
h. Nullif表达式

表达式计算支持的数据类型包括bool，tinyint，smallint，int，bigint，float4，float8，numeric，date，time，timetz，timestamp，timestamptz，interval，bpchar，varchar，text，oid。

仅当表达式出现在以下场景时才会考虑是否使用LLVM动态编译优化：

- 向量化执行引擎中Scan节点的filter；
- Hash Join节点中的complicate hash condition、hash join filter、hash join target；
- Nested Loop节点中的filter、join filter；
- Merge Join节点的merge join filter、merge join target；
- Group节点中的filter表达式。

- 支持LLVM的算子：

  a. Join ： HashJoin
  b. Agg ： HashAgg
  c. Sort

  其中，

  - HashJoin算子仅支持Hash Inner Join，对应的hash cond仅支持int4、bigint、bpchar类型的比较；
  - HashAgg算子仅支持针对bigint、numeric类型的sum及avg操作，且group by语句仅支持int4、bigint、bpchar，text，varchar，timestamp类型操作，同时支持count(*)聚集操作；
  - Sort算子仅支持对int4，bigint，numeric，bpchar，text，varchar数据类型的比较操作；

  除此之外，无法使用LLVM动态编译优化，具体可通过explain performance语句进行查看。

非适用场景：

- 不支持CN上LLVM动态编译优化。
- 不支持小数据量表使用LLVM动态编译优化。
- 不支持生成非向量化执行路径的查询作业。

其他因素对 LLVM 性能的影响

LLVM优化效果不仅依赖于数据库内部具体的实现，还与当前所选择的硬件环境等有关。

- 表达式调用C-函数个数

  数据库内部针对表达式计算并未实现全codegen，即在整个表达式计算中部分表达式实现了codegen，部分直接调用原本的C代码。如果整个表达式计算中后者占据了主要部分，使用LLVM动态编译优化，可能会导致性能劣化。通过设置

 版权所有 © 华为云计算技术有限公司

log_min_messages的级别为DEBUG1可以查看到哪些表达式直接调用了C代码实现。

- 内存资源

  LLVM特性的一个重要思想是保障数据的局域特性，即数据应尽可能地存放在寄存器中。同时应减少数据加载，因此在使用LLVM优化时应设置足够大的work_mem，保证对应使用LLVM优化的执行代码整个过程在内存中实现，否则可能引起性能劣化。

- 优化器代价估算

  LLVM特性实现了简易的代价估算模型，即依据当前参与节点运算的表大小决定当前节点是否考虑使用LLVM动态编译优化。如果优化器低估了实际参与运算的行数，则原本可获得收益的未正常获得收益。反之亦然。

LLVM 使用建议

目前LLVM在数据库内核侧已默认打开，用户可结合上述的分析进行配置，总体建议如下：

1. 设置合理的work_mem，在允许的条件下尽可能设置较大的work_mem，如果出现大量下盘，则建议关闭LLVM动态编译优化（通过设置enable_codegen=off实现）。
2. 设置合理的codegen_cost_threshold(默认值为10000)，确保小数据量场景下避免使用LLVM动态编译优化。当codegen_cost_threshold的值设定后，因使用LLVM动态编译优化引入性能劣化，则建议增加codegen_cost_threshold的取值。
3. 对于表达式计算使用LLVM动态编译优化，如果存在大量的调用C-函数的场景，建议关闭LLVM动态编译优化。
4. In表达式后常量列表长度不能超过10，否则不能执行LLVM编译优化。

   说明

   在资源许可的情况下，数据量越大，可获得的性能提升效果越好。

14.4 SQL 调优

14.4.1 SQL 查询执行流程

SQL引擎从接收SQL语句到执行SQL语句需要经历的步骤如**图14-2**和**表14-7**所示。其中，红色字体部分为DBA可以介入实施调优的环节。

版权所有 © 华为云计算技术有限公司

图 14-2 SQL 引擎执行查询类 SQL 语句的流程

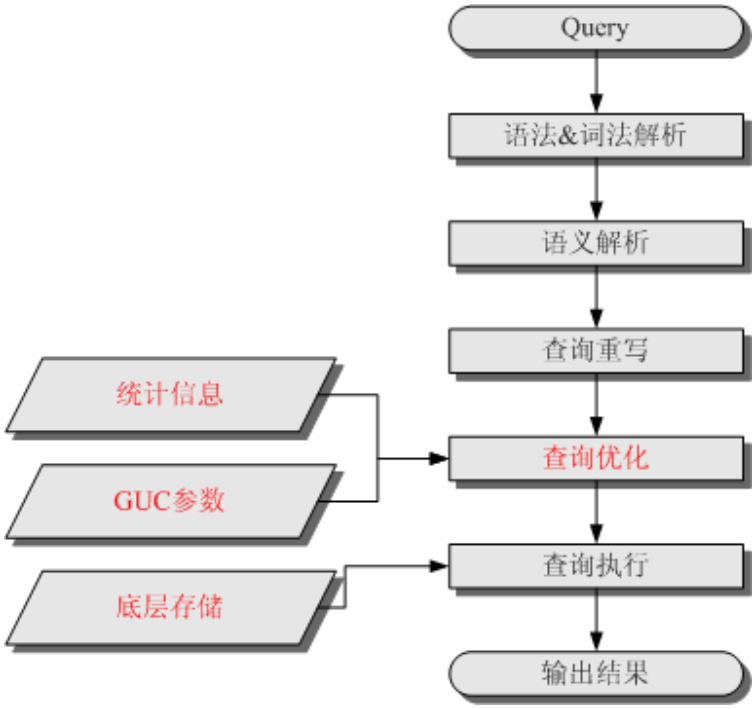


表 14-7 SQL 引擎执行查询类 SQL 语句的步骤说明

步骤	说明
1、语法&词法解析	按照约定的SQL语句规则，把输入的SQL语句从字符串转化为格式化结构(Stmt)。
2、语义解析	将“语法&词法解析”输出的格式化结构转化为数据库可以识别的对象。
3、查询重写	根据规则把“语义解析”的输出等价转化为执行上更为优化的结构。
4、查询优化	根据“查询重写”的输出和数据库内部的统计信息规划SQL语句具体的执行方式，也就是执行计划。统计信息和GUC参数对查询优化（执行计划）的影响，请参见**调优手段之统计信息**和**调优手段之GUC参数**。
5、查询执行	根据“查询优化”规划的执行路径执行SQL查询语句。底层存储方式的选择合理性，将影响查询执行效率。详见**调优手段之底层存储**。

调优手段之统计信息

DWS优化器是典型的基于代价的优化 (Cost-Based Optimization，简称CBO)。在这种优化器模型下，数据库根据表的元组数、字段宽度、NULL记录比率、distinct值、MCV值、HB值等表的特征值，以及一定的代价计算模型，计算出每一个执行步骤的不同执行方式的输出元组数和执行代价（cost），进而选出整体执行代价最小/首元组返回代价最小的执行方式进行执行。这些特征值就是统计信息。从上面描述可以看出统计信息是查询优化的核心输入，准确的统计信息将帮助优化器选择最合适的查询规

 版权所有 © 华为云计算技术有限公司

划，一般来说通过ANALYZE语法收集整个表或者表的若干个字段的统计信息，周期性地运行ANALYZE，或者在对表的大部分内容做了更改之后马上运行它是个好习惯。

调优手段之 GUC 参数

查询优化的主要目的是为查询语句选择高效的执行方式。

如下SQL语句:

```
SELECT count(1)
FROM customer inner join store_sales on (ss_customer_sk = c_customer_sk);
```

在执行customer inner join store_sales的时候，DWS支持Nested Loop、Merge Join和Hash Join三种不同的Join方式。优化器会根据表customer和表store_sales的统计信息估算结果集的大小以及每种join方式的执行代价，然后对比选出执行代价最小的执行计划。

如前所述，执行代价估算依赖于特定模型和统计信息。当估算结果与实际执行成本存在偏差时，可通过调整GUC参数让优化器选择更优的执行计划。

调优手段之底层存储

DWS的表支持行存表、列存表，底层存储方式的选择严格依赖于客户的具体业务场景。一般来说计算型业务查询场景（以关联、聚合操作为主）建议使用列存表；点查询、大批量UPDATE/DELETE业务场景适合行存表。

对于每种存储方式还有对应的存储层优化手段，这部分会在后续的调优章节深入介绍。

调优手段之 SQL 重写

除了上述干预SQL引擎所生成执行计划的执行性能外，根据数据库的SQL执行机制以及大量的实践发现，有些场景下，在保证客户业务SQL逻辑的前提下，通过一定规则由DBA重写SQL语句，可以大幅度的提升SQL语句的性能。

这种调优场景对DBA的要求比较高，需要对客户业务有足够的了解，同时也需要扎实的SQL语句基本功，后续会介绍几个常见的SQL改写场景。

14.4.2 SQL 执行计划

SQL执行计划是一个节点树，显示DWS执行一条SQL语句时执行的详细步骤。

使用EXPLAIN命令可以查看优化器为每个查询生成的具体执行计划。EXPLAIN给每个执行节点都输出一行，显示基本的节点类型和优化器为执行这个节点预计的开销值。

执行计划显示信息

除了设置不同的执行计划显示格式外，还可以通过不同的EXPLAIN用法，显示不同详细程度的执行计划信息。常见有如下几种，更多用法请参见**EXPLAIN语法**章节说明。

- EXPLAIN *statement*： 只生成执行计划，不实际执行。其中statement代表SQL语句。
- EXPLAIN ANALYZE *statement*：生成执行计划，进行执行，并显示执行的概要信息。显示中加入了实际的运行时间统计，包括在每个规划节点内部花掉的总时间(以毫秒计)和它实际返回的行数。

 版权所有 © 华为云计算技术有限公司

- EXPLAIN PERFORMANCE *statement*：生成执行计划，进行执行，并显示执行期间的全部信息。

为了测量运行时在执行计划中每个节点的开销，EXPLAIN ANALYZE或EXPLAIN PERFORMANCE会在当前查询执行上增加性能分析的开销。在一个查询上运行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE有时会比普通查询明显地花费更多的时间。超支的数量依赖于查询的本质和使用的平台。

因此，当定位SQL运行慢问题时，如果SQL长时间运行未结束，建议通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及其实际的运行信息，以便更精准地定位问题原因。

执行计划中的常见关键字说明：

1. 表访问方式
   - Seq Scan/CStore Scan

     全表顺序扫描。最基本的扫描算子，用于行/列存表的顺序扫描。
   - Index Scan/CStore Index Scan

     行/列存表的索引扫描。行/列存表上存在索引，条件列为索引列。

     优化器决定使用两步的规划：最底层的规划节点访问一个索引，找出匹配索引条件的行的位置，然后上层规划节点真实地从表中抓取出对应行。独立地抓取数据行比顺序地读取数据的开销高很多，但是因为并非所有表的页面都被访问了，这么做实际上仍然比一次顺序扫描开销要少。使用两层规划的原因是，上层规划节点在读取索引标识出来的行位置之前，会先将它们按照物理位置排序，这样可以最小化独立抓取的开销。

     如果在WHERE中使用的多个字段上都存在索引，那么优化器可能会使用索引的AND或OR的组合，此方式要求访问两个索引，与只使用一个索引，而把另外一个条件只当作过滤器相比，这个方法未必是更优。

     索引扫描可以分为以下几类，其差异在于索引的排序机制。

     - Bitmap Index Scan

       使用位图索引抓取数据页，需要索引扫描获取位图后再到基表上扫描。

     - Index Scan using index_name

       使用简单索引搜索，该方式表的数据行是以索引顺序抓取的，这样就令读取它们的开销更大，但是这里的行少得可怜，因此对行位置的额外排序并不值得。最常见的就是看到这种规划类型只抓取一行，以及那些要求ORDER BY条件匹配索引顺序的查询。因为那时候没有多余的排序步骤是必要的以满足ORDER BY。
2. 表连接方式
   - Nested Loop

     嵌套循环，适用于被连接的数据子集较小的查询。在嵌套循环中，外表驱动内表，外表返回的每一行都要在内表中检索找到它匹配的行，因此整个查询返回的结果集不能太大（不能大于10000），要把返回子集较小的表作为外表，而且在内表的连接字段上建议要有索引。
   - (Sonic) Hash Join

     哈希连接，适用于数据量大的表的连接方式。优化器使用两个表中较小的表，利用连接键在内存中建立hash表，然后扫描较大的表并探测散列，找到与散列匹配的行。Sonic和非Sonic的Hash Join的区别在于所使用hash表结构不同，不影响执行的结果集。

 版权所有 © 华为云计算技术有限公司

- Merge Join

  归并连接或融合连接，是先将关联表的关联列各自做排序，然后从各自的排序表中抽取数据，到另一个排序表中做匹配。

  Merge join需要执行更多的排序操作，所以资源消耗更多，因此通常情况下执行性能差于Hash Join。 如果源数据已经被排序过，在执行融合连接时无需再排序，此时Merge Join的性能优于Hash Join。

3. 运算符
   - sort

     对结果集进行排序。
   - filter

     EXPLAIN输出显示WHERE子句当作一个"filter"条件附属于顺序扫描计划节点。这意味着规划节点为它扫描的每一行检查该条件，并且只输出符合条件的行。预计的输出行数降低了，因为有WHERE子句。不过，扫描仍将必须访问所有 10000 行，因此开销没有降低；实际上它还增加了一些（确切的说，通过10000 * cpu_operator_cost）以反映检查WHERE条件的额外CPU时间。
   - LIMIT

     LIMIT限定了执行结果的输出记录数。如果增加了LIMIT，那么不是所有的行都会被检索到。

执行计划显示格式

DWS对执行计划提供了normal、pretty、summary、run四种显示格式。通过设置GUC参数explain_perf_mode，可以显示不同格式的执行计划。

- normal：代表使用默认的打印格式。图14-3中即为此显示格式。

  图 14-3 normal 格式执行计划示例

```
postgres=# explain select * from test where a < 1;
                            QUERY PLAN
------------------------------------------------------------------
 Streaming (type: GATHER)  (cost=0.25..19.16 rows=7 width=8)
   Node/s: All datanodes
   ->  Seq Scan on test  (cost=0.00..13.16 rows=7 width=8)
         Filter: (a < 1)
(4 rows)
```

- pretty：代表使用DWS改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。如图14-4。

  图 14-4 pretty 格式执行计划示例

```
postgres=# explain select cjxh, count(1) from dwcjk group by cjxh;
 id |              operation               | E-rows | E-memory | E-width | E-costs
----+--------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                      |      1 |          |      52 | 58.42
  2 |    ->  Vector Streaming (type: GATHER) |      1 |          |      52 | 58.42
  3 |       ->  Vector Hash Aggregate      |      1 | 16MB     |      52 | 58.02
  4 |          ->  CStore Scan on dwcjk    |      1 | 1MB      |      44 | 58.00
(4 rows)
```

- summary：是在pretty的基础上增加了对打印信息的分析。
- run：在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

版权所有 © 华为云计算技术有限公司

常见类型计划

DWS中当前主要存在三类分布式计划：

- FQS(fast query shipping)计划

  CN直接将原语句下发到DN，各DN单独执行，并将执行结果在CN上进行汇总。

- Stream计划

  CN根据原语句生成计划并将计划下发给DN进行执行，各DN执行过程中使用Stream算子进行数据交互。

- Remote-Query计划

  CN生成计划后，将部分原语句下发到DN，各DN单独执行，执行后将结果发送给CN，CN执行剩余计划。

现有表tt01和tt02定义如下：

```
CREATE TABLE tt01(c1 int, c2 int) DISTRIBUTE BY hash(c1);
CREATE TABLE tt02(c1 int, c2 int) DISTRIBUTE BY hash(c2);
```

类型一：FQS计划，完全下推

两表JOIN，且其连接条件为各表的分布列，在关闭stream算子的情况下，CN会直接将该语句发送至各DN执行，最后结果在CN汇总。

```
SET enable_stream_operator=off;
SET explain_perf_mode=normal;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Data Node Scan on "__REMOTE_FQS_QUERY__"
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   Remote query: SELECT tt01.c1, tt01.c2, tt02.c1, tt02.c2 FROM dbadmin.tt01, dbadmin.tt02 WHERE tt01.c1
= tt02.c2
(4 rows)
```

类型二：非FQS计划，部分语句下推

两表JOIN，且连接条件中包含非分布列，此时在关闭stream算子的情况下，CN会将基表扫描语句下发至各DN，然后在CN上进行JOIN。

```
SET enable_stream_operator=off;
SET explain_perf_mode=normal;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c1;
                               QUERY PLAN
-------------------------------------------------------------------------------
 Hash Join
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Hash Cond: (tt01.c1 = tt02.c1)
   ->  Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"
         Output: tt01.c1, tt01.c2
         Node/s: All datanodes
         Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt01 WHERE true
   ->  Hash
         Output: tt02.c1, tt02.c2
         ->  Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_"
               Output: tt02.c1, tt02.c2
               Node/s: All datanodes
               Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt02 WHERE true
(13 rows)
```

类型三：Stream计划，DN之间无数据交换

版权所有 © 华为云计算技术有限公司

两表JOIN，且连接条件为各表的分布列，因此各DN无需数据交换。CN生成stream计划后，将除Gather Stream的计划下发给DN执行，在各个DN上进行基表扫描，并进行哈希连接后，发送给CN。

```
SET enable_fast_query_shipping=off;
SET enable_stream_operator=on;

EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
                QUERY PLAN
-----------------------------------------------------
 Streaming (type: GATHER)
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   ->  Hash Join
         Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
         Hash Cond: (tt01.c1 = tt02.c2)
         ->  Seq Scan on dbadmin.tt01
               Output: tt01.c1, tt01.c2
               Distribute Key: tt01.c1
         ->  Hash
               Output: tt02.c1, tt02.c2
               ->  Seq Scan on dbadmin.tt02
                     Output: tt02.c1, tt02.c2
                     Distribute Key: tt02.c2
(14 rows)
```

类型四：Stream计划，DN之间存在数据交换

两表JOIN，且连接条件包含非分布列，在开启stream算子(SET enable_stream_operator=on)的情况下，会生成stream计划，其DN间存在数据交换。此时对于tt02表，会在各DN进行基表扫描，扫描后会通过Redistribute Stream算子，按照JOIN条件中的tt02.c1进行哈希计算后重新发送给各DN，然后在各DN上做JOIN，最后汇总到CN。

```
postgres=> SET enable_stream_operator=on;
SET
postgres=> SET enable_fast_query_shipping=off;
SET
postgres=> SET explain_perf_mode=normal;
SET
postgres=> EXPLAIN (VERBOSE on,COSTS off) SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c1;
                      QUERY PLAN
-------------------------------------------------------
 Streaming (type: GATHER)
   Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
   Node/s: All datanodes
   ->  Hash Join
         Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
         Hash Cond: (tt02.c1 = tt01.c1)
         ->  Streaming(type: REDISTRIBUTE)
               Output: tt02.c1, tt02.c2
               Distribute Key: tt02.c1
               Spawn on: All datanodes
               Consumer Nodes: All datanodes
               ->  Seq Scan on dbadmin.tt02
                     Output: tt02.c1, tt02.c2
                     Distribute Key: tt02.c2
         ->  Hash
               Output: tt01.c1, tt01.c2
               ->  Seq Scan on dbadmin.tt01
                     Output: tt01.c1, tt01.c2
                     Distribute Key: tt01.c1
(19 rows)
```

类型五：Remote-Query计划场景

因unship_func不能下推，且不满足部分下推要求（子查询下推），所以只能发送基表扫描的语句到DN，将基表数据收集到CN上来计算。

```
postgres=> CREATE FUNCTION unship_func(integer,integer) returns integer
postgres-> AS 'select $1 + $2;'
postgres-> LANGUAGE SQL volatile
postgres-> returns null on null input;
CREATE FUNCTION
```

 版权所有 © 华为云计算技术有限公司

```
postgres=> SET explain_perf_mode=pretty;
SET
postgres=> EXPLAIN VERBOSE SELECT unship_func(tt01.c1,tt01.c2) FROM tt01 JOIN tt02 on tt01.c1=tt02.c1;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 id |                      operation                          | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Hash Join (2,3)                                     |     30 |            |       8 | 0.86
  2 |    ->  Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"    |     30 |            |       8 | 0.00
  3 |    ->  Hash                                             |     30 |            |       4 | 0.00
  4 |       ->  Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_" |     30 |            |       4 | 0.00

             SQL Diagnostic Information
 ---------------------------------------------------
 SQL is not plan-shipping
        reason: Function unship_func() can not be shipped

 Predicate Information (identified by plan id)
 ---------------------------------------------
   1 --Hash Join (2,3)
         Hash Cond: (tt01.c1 = tt02.c1)

            Targetlist Information (identified by plan id)
 --------------------------------------------------------------------
   1 --Hash Join (2,3)
         Output: (tt01.c1 + tt01.c2)
   2 --Data Node Scan on tt01 "_REMOTE_TABLE_QUERY_"
         Output: tt01.c1, tt01.c2
         Node/s: All datanodes
         Remote query: SELECT c1, c2 FROM ONLY dbadmin.tt01 WHERE true
   3 --Hash
         Output: tt02.c1
   4 --Data Node Scan on tt02 "_REMOTE_TABLE_QUERY_"
         Output: tt02.c1
         Node/s: All datanodes
         Remote query: SELECT c1 FROM ONLY dbadmin.tt02 WHERE true

 ====== Query Summary =====
 --------------------------
 Parser runtime: 0.055 ms
 Planner runtime: 0.528 ms
 Unique SQL Id: 1780774145
(37 rows)
```

EXPLAIN PERFORMANCE 详解

在SQL调优过程中经常需要执行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看SQL语句实际执行信息，通过对比实际执行与优化器的估算之间的差别来为优化提供依据。EXPLAIN PERFORMANCE相对于EXPLAIN ANALYZE增加了每个DN上的执行信息。

表定义如下：

```
CREATE TABLE tt01(c1 int, c2 int) DISTRIBUTE BY hash(c1);
CREATE TABLE tt02(c1 int, c2 int) DISTRIBUTE BY hash(c2);
```

以如下SQL查询语句为例：

```
SELECT * FROM tt01,tt02 WHERE tt01.c1=tt02.c2;
```

执行EXPLAIN PERFORMANCE输出的显示执行信息分为以下8个部分：

1. 执行计划

```
                                                    QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
 id |               operation                |      A-time      | A-rows | E-rows | E-distinct | Peak Memory  | E-memory | A-width | E-width | E-costs
----+----------------------------------------+------------------+--------+--------+------------+--------------+----------+---------+---------+---------
  1 | ->  Streaming (type: GATHER)           | 2.566            |      0 |     30 |            | 24KB         |          |         |      16 | 36.59
  2 |    ->  Hash Join (3,4)                 | [0.007, 0.009]   |      0 |     30 |            | [8KB, 8KB]   | 1MB      |         |      16 | 28.59
  3 |       ->  Seq Scan on dbadmin.tt01     | [0.002, 0.003]   |      0 |     30 | 14         | [16KB, 16KB] | 1MB      |         |       8 | 14.14
  4 |       ->  Hash                         | [0,0]            |      0 |     29 | 14         | [0, 0]       | 16MB     |         |       8 | 14.14
  5 |          ->  Seq Scan on dbadmin.tt02  | [0,0]            |      0 |     30 |            | [0, 0]       | 1MB      |         |       8 | 14.14
```

以表格的形式将计划显示出来，包含有11个字段，分别是：id、operation、A-time、A-rows、E-rows、E-distinct、Peak Memory、E-memory、A-width、E-width和E-costs。字段含义如下**表14-8**。

表 14-8 执行字段说明

字段	描述
id	执行算子节点编号。

版权所有 © 华为云计算技术有限公司

字段	描述
operation	具体的执行节点算子名称。 Vector前缀的算子是指向量化执行引擎算子，一般出现含有列存表的Query中。 Streaming是一个特殊的算子，它实现了分布式架构的核心数据shuffle功能，Streaming共有三种形态，分别对应了分布式结构下不同的数据shuffle功能： ● Streaming (type: GATHER)：作用是coordinator从DN收集数据。 ● Streaming(type: REDISTRIBUTE)：作用是DN根据选定的列把数据重分布到所有的DN。 ● Streaming(type: BROADCAST)：作用是把当前DN的数据广播给其他所有的DN。
A-time	各DN相应算子执行时间，一般DN上执行的算子的A-time是由[]括起来的两个值，分别表示此算子在所有DN上完成的最短时间和最长时间，包括下层算子执行时间。 注意：在整个计划中，除了叶子节点的执行时间是算子本身的执行时间，其余算子的执行时间均包含子节点的执行时间。
A-rows	表示相应算子输出的全局总行数。
E-rows	每个算子估算的输出行数。
E-distinct	表示hashjoin算子的distinct估计值。
Peak Memory	此算子在每个DN上执行时使用的内存峰值，[]中左侧为最小值，右侧为最大值。
E-memory	DN上每个算子估算的内存使用量，只有DN上执行的算子会显示。某些场景会在估算的内存使用量后使用括号显示该算子在内存资源充足下可以自动扩展的内存上限。
A-width	表示当前算子每行元组的实际宽度，仅对于重内存使用算子会显示，包括：(Vec)HashJoin、(Vec)HashAgg、(Vec) HashSetOp、(Vec)Sort、(Vec)Materialize算子等，其中(Vec)HashJoin计算的宽度是其右子树算子的宽度，会显示在其右子树上。
E-width	每个算子输出元组的估算宽度。
E-costs	每个算子估算的执行代价。 ● E-costs是优化器根据成本参数定义的单位来衡量的，习惯上以磁盘页面抓取为1个单位， 其他开销参数将参照它来设置。 ● 每个节点的开销（ E-costs值 ）包括它的所有子节点的开销。 ● 开销只反映了优化器关心的东西，并没有把结果行传递给客户端的时间考虑进去。虽然这个时间可能在实际的总时间里占据相当重要的分量，但是被优化器忽略了，因为它无法通过修改规划来改变。

 版权所有 © 华为云计算技术有限公司

2. SQL Diagnostic Information

   SQL自诊断信息。优化和执行过程中识别到的性能优化点，当对DML语句进行带VERBOSE属性的EXPLAIN（EXPLAIN PERFORMANCE内置VERBOSE属性）时，SQL自诊断信息也会输出，以辅助性能问题定位。

3. Predicate Information (identified by plan id)

```
Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
        Hash Cond: (tt01.c1 = tt02.c2)
  3 --Seq Scan on dbadmin.tt01
        Filter: (tt01.c1 >= 202007)
  5 --Seq Scan on dbadmin.tt02
        Filter: (tt02.c2 >= 202007)
```

   谓词过滤这部分主要显示的是对应执行算子节点的过滤条件，即在整个计划执行过程中不会变的信息，主要是一些join条件和一些filter信息。

   8.3.0及以上集群版本支持显示与字典计划相关的CU Predicate Filter和Pushdown Predicate Filter(will be pruned)信息。

4. Memory Information (identified by plan id)

```
          Memory Information (identified by plan id)
------------------------------------------------------------------
Coordinator Query Peak Memory:
        Query Peak Memory: 2MB
DataNode Query Peak Memory
        dn_6001_6002 Query Peak Memory: 0MB
        dn_6003_6004 Query Peak Memory: 0MB
        dn_6005_6006 Query Peak Memory: 0MB
  1 --Streaming (type: GATHER)
        Peak Memory: 56KB, Estimate Memory: 512MB
  2 --Hash Join (3,4)
        dn_6001_6002 Peak Memory: 8KB, Estimate Memory: 1024KB
        dn_6003_6004 Peak Memory: 8KB, Estimate Memory: 1024KB
        dn_6005_6006 Peak Memory: 8KB, Estimate Memory: 1024KB
  3 --Seq Scan on dbadmin.tt01
        dn_6001_6002 Peak Memory: 32KB, Estimate Memory: 1024KB
        dn_6003_6004 Peak Memory: 32KB, Estimate Memory: 1024KB
        dn_6005_6006 Peak Memory: 32KB, Estimate Memory: 1024KB
  4 --Hash
        dn_6001_6002 Buckets: 0  Batches: 0  Memory Usage: 0kB
        dn_6003_6004 Buckets: 0  Batches: 0  Memory Usage: 0kB
        dn_6005_6006 Buckets: 0  Batches: 0  Memory Usage: 0kB
```

   内存使用信息这部分显示的是整个计划中会将内存的使用情况打印出来的算子的内存使用信息，主要是Hash、Sort算子，包括算子峰值内存（peak memory），优化器预估的内存（estimate memory），控制内存（control memory），估算内存使用（operator memory），执行时实际宽度（width），内存使用自动扩展次数（auto spread num），是否提前下盘（early spilled），以及下盘信息，包括重复下盘次数（spill Time(s)），内外表下盘分区数（inner/outer partition spill num），下盘文件数（temp file num），下盘数据量及最小和最大分区的下盘数据量（written disk IO [min, max] ）。其中sort算子不会显示具体的下盘文件数，仅在显示排序方法时显示Disk。

5. Targetlist Information (identified by plan id)

 版权所有 © 华为云计算技术有限公司

```
  Targetlist Information (identified by plan id)
--------------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
        Node/s: All datanodes
  2 --Hash Join (3,4)
        Output: tt01.c1, tt01.c2, tt02.c1, tt02.c2
  3 --Seq Scan on dbadmin.tt01
        Output: tt01.c1, tt01.c2
        Distribute Key: tt01.c1
  4 --Hash
        Output: tt02.c1, tt02.c2
  5 --Seq Scan on dbadmin.tt02
        Output: tt02.c1, tt02.c2
        Distribute Key: tt02.c2
```

这一部分显示的是每一个算子对应的输出目标列信息。

8.3.0及以上集群版本支持显示与字典相关的Dict Optimized和Dict Decoded，分别表示字典列和字典编码。

6. DataNode Information (identified by plan id)

```
                                        Datanode Information (identified by plan id)
-----------------------------------------------------------------------------------------------------------------------------------------------
 1 --Row Adapter
      (actual time=68637.768..68637.772 rows=1 loops=1)
      (CPU: ex c/r=7589244, ex row=1, ex cyc=7589244, inc cyc=205913250248)
 2 --Vector Aggregate
      (actual time=68635.241..68635.241 rows=1 loops=1) (projection time=0.014)
      (Buffers: shared hit=1)
      (CPU: ex c/r=1675582, ex row=2, ex cyc=3351164, inc cyc=205905661004)
 3 --Vector Streaming (type: GATHER)
      (actual time=54126.686..68634.124 rows=2 loops=1)
      (Buffers: shared hit=1)
      (CPU: ex c/r=102951154920, ex row=2, ex cyc=205902309840, inc cyc=205902309840)
 4 --Vector Aggregate
      dn_6001_6002 (actual time=54076.307..54076.308 rows=1 loops=1) (projection time=65.581)
      dn_6003_6004 (actual time=68597.121..68597.122 rows=1 loops=1) (projection time=64.801)
      dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
      dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
      dn_6001_6002 (CPU: ex c/r=44, ex row=29999729, ex cyc=1346672758, inc cyc=162228870602)
      dn_6003_6004 (CPU: ex c/r=44, ex row=29999728, ex cyc=1343711688, inc cyc=205791299646)
 5 --CStore Scan on public.lineitem_v3
      dn_6001_6002 (actual time=594.261..53628.924 rows=29999729 loops=1) (CU ScanInfo: smallCu: 0, totalCu: 500, avrCuRow: 59999, totalDeadRows: 0)
      dn_6003_6004 (actual time=865.099..68150.512 rows=29999728 loops=1) (CU ScanInfo: smallCu: 0, totalCu: 500, avrCuRow: 59999, totalDeadRows: 0)
      dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
      dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
      dn_6001_6002 (CStore Buffers: shared hit=29500 read=500)
      dn_6003_6004 (CStore Buffers: shared hit=29500 read=500)
      dn_6001_6002 (Disk Cache: miss=576 scanBytes=80MB remoteReadBytes=576MB)
      dn_6003_6004 (Disk Cache: miss=579 scanBytes=80MB remoteReadBytes=579MB)
      dn_6001_6002 (CPU: ex c/r=5362, ex row=29999729, ex cyc=160882197844, inc cyc=160882197844)
      dn_6003_6004 (CPU: ex c/r=6814, ex row=29999728, ex cyc=204447587958, inc cyc=204447587958)
```

这部分将各个算子的执行时间（若包含过滤及投影也会显示对应的执行时间）、CPU、buffer的使用情况全部打印出来。

- 算子执行信息

  ```
  4 --Vector Aggregate
        dn_6001_6002 (actual time=54076.307..54076.308 rows=1 loops=1) (projection time=65.581)
        dn_6003_6004 (actual time=68597.121..68597.122 rows=1 loops=1) (projection time=64.801)
  ```

  每个算子的执行信息都包含三个部分：

  - dn_6001_6002/dn_6003_6004表示具体执行的节点信息，括号中的信息是实际的执行信息。
  - actual time表示实际的执行时间，第一个数字表示执行时进入当前算子到输出第一条数据所花费的时间，第二个数字表示输出所有数据的总执行时间。
  - rows表示当前算子输出数据行数。
  - loops表示当前算子的执行次数。需要注意，对于分区表来说，每一个分区表的扫描就是一次完整的扫描操作，当切换到下一个分区的时候，又是一次新的扫描操作。

- CPU信息

 版权所有 © 华为云计算技术有限公司

```
dn_6001_6002 (CPU: ex c/r=44, ex row=29999729, ex cyc=1346672758, inc cyc=162228870602)
dn_6003_6004 (CPU: ex c/r=44, ex row=29999728, ex cyc=1343711688, inc cyc=205791299646)
```

每个算子执行的过程都有CPU信息，其中cyc代表的是CPU的周期数，ex cyc表示的是当前算子的周期数，不包含其子节点；inc cyc是包含子节点的周期数；ex row是当前算子输出的数据行数；ex c/r则是ex cyc/ex row得到的每条数据所用的平均周期数。

- Buffer信息

```
dn_6001_6002 (Buffers: shared hit=1725 dirtied=117)
dn_6003_6004 (Buffers: shared hit=1723 dirtied=117)
```

buffers显示缓冲区信息，包括共享块和临时块的读和写。

共享块包含表和索引，临时块在排序和物化中使用的磁盘块。上层节点显示出来的块数据包含了其所有子节点使用的块数。

- 磁盘缓存信息（仅9.1.0.100及以上版本且colversion=3.0的存算分离V3表或外表支持）

```
dn_6001_6002 (Disk Cache: hit=145611 miss=47 error=47 errorCode=152 scanBytes=18325.02MB remoteReadBytes=959.00MB loadTime=13109.5)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1230.6 preloadWaitTime=1724.9 preloadWaitCount=28)
             (OBS I/O: count=349 averageRTT=238.7 averageLatency=233.0)
dn_6003_6004 (Disk Cache: hit=140641 miss=45 error=45 errorCode=152 scanBytes=17508.64MB remoteReadBytes=902.00MB loadTime=13271.7)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1202.9 preloadWaitTime=3061.5 preloadWaitCount=20)
             (OBS I/O: count=327 averageRTT=231.5 averageLatency=226.3 latencyGt1s=1)
dn_6005_6006 (Disk Cache: hit=139736 miss=47 error=47 errorCode=152 scanBytes=17494.91MB remoteReadBytes=909.00MB loadTime=12752.5)
             (Column 3.0: preloadStep=20 preloadSubmitTime=1169.4 preloadWaitTime=2723.4 preloadWaitCount=13)
             (OBS I/O: count=327 averageRTT=194.3 averageLatency=188.8 latencyGt1s=3)
```

Disk Cache：表示磁盘缓存的命中信息和数据读取信息。（存算分离V3表或外表支持）

miss代表磁盘缓存未命中的次数，hit表示磁盘缓存命中的次数，errorCode见**表17-195**，disk_cache_error_code，error表示产生errorCode的次数。scanBytes表示scan查询的数据量，remoteReadBytes表示在OBS上读取的数据量，loadTime表示从磁盘缓存加载数据的时间。为了提升OBS的效率会对相邻的请求块合并，或者因为请求写磁盘缓存的最小粒度是block(默认1M)，可能会使得scanBytes会小于remoteReadBytes。

Column 3.0：存算分离V3表的预读参数与预读过程信息。（仅存算分离V3表支持，开启预读相关参数后显示）

preloadStep表示预读的步长，preloadSubmitTime表示预读流程中提交IO请求的时间，preloadWaitTime表示预读流程中等待IO请求的时间，preloadWaitCount表示预读流程中等待IO请求的次数。

OBS I/O ： 表示OBS IO请求的详细信息。（存算分离V3表或外表支持）

count表示OBS IO请求的总数量，averageRTT表示OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs，averageLatency表示OBS IO请求的平均延迟，单位为μs，latencyGt1s表示OBS IO请求延迟超过1s的数量，latencyGt10s表示OBS IO请求延迟超过10s的数量，retryCount表示OBS IO请求重试的总次数，rateLimitCount表示OBS IO请求被流控的总次数。

7. User Define Profiling

```
                  User Define Profiling
------------------------------------------------------------
Plan Node id: 1  Track name: coordinator get datanode connection
      cn_5001 (time=9.306 total_calls=1 loops=1)
Plan Node id: 1  Track name: coordinator begin transaction
      cn_5001 (time=0.002 total_calls=1 loops=1)
Plan Node id: 1  Track name: coordinator send command
      cn_5001 (time=0.113 total_calls=3 loops=1)
Plan Node id: 1  Track name: coordinator get the first tuple
      cn_5001 (time=0.091 total_calls=12 loops=1)
```

 版权所有 © 华为云计算技术有限公司

自定义信息，这一部分显示的是CN和DN、DN和DN建连的时间，以及存储层的一些执行信息。

8. Query Summary

```
                    ====== Query Summary =====
---------------------------------------------------------------------------
Datanode executor start time [dn_6005_6006, dn_6001_6002]: [0.360 ms,0.483 ms]
Datanode executor run time [dn_6001_6002, dn_6003_6004]: [0.008 ms,0.009 ms]
Datanode executor end time [dn_6003_6004, dn_6005_6006]: [0.036 ms,0.066 ms]
Remote query poll time: 2.649 ms, Deserialze time: 0.000 ms
System available mem: 1761280KB
Query Max mem: 1761280KB
Query estimated mem: 3328KB
Enqueue time: 0.030 ms
Coordinator executor start time: 0.083 ms
Coordinator executor run time: 13.044 ms
Coordinator executor end time: 0.034 ms
Parser runtime: 0.060 ms
Planner runtime: 0.539 ms
Query Id: 218706056932222840
Unique SQL Id: 2641724793
Total runtime: 13.906 ms
```

这一部分主要打印总的执行时间和网络流量，包括了各个DN上初始化和结束阶段的最大最小执行时间、CN上的初始化、执行、结束阶段的时间，以及当前语句执行时系统可用内存、语句估算内存等信息。

- DataNode executor start time：DN执行器开始时间，[min_node_name, max_node_name] : [min_time, max_time]。
- DataNode executor run time：DN执行器运行时间，[min_node_name, max_node_name] : [min_time, max_time]。
- DataNode executor end time：DN执行器结束时间，[min_node_name, max_node_name] : [min_time, max_time]。
- Remote query poll time：接收结果时用于poll等待的时间。
- System available mem：系统可用内存。
- Query Max mem：查询最大内存。
- Enqueue time：入队时间。
- Coordinator executor start time：CN执行器开始时间。
- Coordinator executor run time：CN执行器运行时间。
- Coordinator executor end time：CN执行器结束时间。
- Parser runtime：解析器运行时间。
- Planner runtime：优化器执行时间。
- 网络流量，stream算子发送的数据量。
- Query Id：查询ID。
- Unique SQL ID：约束SQL ID。
- Total runtime：总执行时间。

 版权所有 © 华为云计算技术有限公司

须知

- A-rows和E-rows的差异体现了优化器估算和实际执行的偏差度。一般情况下两者偏差越大，则可以认为优化器生成的计划的越不可信，人工干预调优的必要性越大。
- A-time中的两个值偏差越大，表明此算子的计算偏斜(在不同DN上执行时间差异)越大，人工干预调优的必要性越大。一般来说，两个相邻的算子，上层算子的执行时间包含下层算子的执行时间，但如果上层算子为stream算子，由于各线程不存在驱动关系，上层算子执行时间可能小于下层算子的执行时间，即不存在包含关系。
- Max Query Peak Memory经常用来估算SQL语句耗费内存，也被用来作为SQL语句调优时运行态内存参数设置的重要依据。一般会以EXPLAIN ANALYZE或EXPLAIN PERFORMANCE的输出作为进一步调优的输入。

14.4.3 执行计划算子

算子介绍

SQL执行计划中每一个步骤为一个数据库运算符，也叫作一个执行算子。DWS中算子是基本的数据处理单元，合理地组合算子、优化算子的顺序和执行方式，可以提升数据的处理效率。

DWS算子可分为：扫描算子、控制算子、物化算子、连接算子、其他算子等。

扫描算子

扫描算子用来扫描表中的数据，每次获取一条元组作为上层节点的输入， 存在于查询计划树的叶子节点，它不仅可以扫描表，还可以扫描函数的结果集、链表结构、子查询结果集。常见的扫描算子如下表所示：

表 14-9 扫描算子

算子	含义	场景
SeqScan	顺序扫描	最基本的扫描算子，用于扫描物理表（没有索引辅助的顺序扫描）。
IndexScan	索引扫描	选择条件涉及的属性上建立了索引。
IndexOnlyScan	直接从索引返回元组	索引列完全覆盖结果集列。
BitmapScan(BitmapIndexScan, BitmapHeapScan)	利用Bitmap获取元组	BitmapIndexScan利用属性上的索引进行扫描，返回结果为一个位图；BitmapHeapScan从BitmapIndexScan输出的位图中获取元组。
TidScan	通过元组tid获取元组	1. WHERE conditions(like CTID = tid or CTID IN (tid1, tid2, ···))； 2. UPDATE/DELETE ··· WHERE CURRENT OF cursor；

版权所有 © 华为云计算技术有限公司

算子	含义	场景
SubqueryScan	子查询扫描	以另一个查询计划树（子计划）为扫描对象进行元组的扫描。
FunctionScan	函数扫描	FROM function_name
ValuesScan	扫描values链表	对VALUES子句给出的元组集合进行扫描。
ForeignScan	外部表扫描	查询外部表。
CteScan	CTE表扫描	扫描SELECT查询中用WITH子句定义的子查询。

连接算子

连接算子对应了关系代数中的连接操作，以表 t1 join t2 为例，主要的集中连接类型如下：inner join、left join、right join、full join、semi join、 anti join，其实现方式包括Nestloop、HashJoin及MergeJoin。

表 14-10 连接算子

算子	含义	场景	实现特点
NestLoop	嵌套循环连接，暴力连接，对每一行都扫描内表。	Inner Join, Left Outer Join, Semi Join, Anti Join	适用于被连接的数据子集较小的查询。在嵌套循环中，外表驱动内表，外表返回的每一行都要在内表中检索找到它匹配的行，因此整个查询返回的结果集不能太大（不能大于10000），要把返回子集较小的表作为外表，而且在内表的连接字段上建议要有索引。
MergeJoin	归并连接（输入有序），内外表排序，定位首尾两端，一次性连接元组。等值连接。	Inner Join, Left Outer Join, Right Outer Join, Full Outer Join, Semi Join, Anti Join	也称作“融合连接”，是先将关联表的关联列各自做排序，然后从各自的排序表中抽取数据，到另一个排序表中做匹配。 Merge join需要执行更多的排序操作，所以资源消耗更多，因此通常情况下执行性能差于Hash Join。 如果源数据已经被排序过，在执行融合连接时无需再排序，此时Merge Join的性能优于Hash Join。

版权所有 © 华为云计算技术有限公司

算子	含义	场景	实现特点
(Sonic) HashJoin	哈希连接，内外表使用join列的hash值建立hash表，相同值的必在同一个hash桶。等值连接的连接两端必须为类型相同的等值连接，且支持hash散列。	Inner Join, Left Outer Join, Right Outer Join, Full Outer Join, Semi Join, Anti Join	哈希连接，适用于数据量大的表的连接方式。优化器使用两个表中较小的表，利用连接键在内存中建立hash表，然后扫描较大的表并探测散列，找到与散列匹配的行。Sonic和非Sonic的Hash Join的区别在于所使用hash表结构不同，不影响执行的结果集。

物化算子

物化算子是一类可缓存元组的节点。在执行过程中，很多扩展的物理操作符需要首先获取所有的元组才能进行操作（例如聚集函数操作、没有索引辅助的排序等），这是要用物化算子将元组缓存起来。

表 14-11 物化算子

算子	含义	场景
Material	物化	缓存子节点结果。
Sort	排序	ORDER BY子句，连接操作，分组操作，集合操作，配合Unique。
Group	分组操作	GROUP BY子句。
Agg	执行聚集函数。包含三种类型： • HashAggregate，哈希聚集算子，适用于大部分场景，以及数据无序情况下的聚集计算。 • GroupAggregate，排序聚集算子，基于排序或已排序输入做分组聚合。 • Aggregate，普通聚合算子，不分组，只简单聚集。	1. COUNT/SUM/AVG/MAX/MIN等聚集函数。 2. DISTINCT子句。 3. UNION去重。 4. GROUP BY子句。
WindowAgg	窗口函数	WINDOW子句。
Unique	去重（下层已排序）	1. DISTINCT子句。 2. UNION去重。
Hash	HashJoin辅助节点	构造Hash表，配合HashJoin。

版权所有 © 华为云计算技术有限公司

算子	含义	场景
SetOp	处理集合操作	INTERSECT/INTERSECT ALL，EXCEPT/EXCEPT ALL
LockRows	处理行级锁	SELECT … FOR SHARE/ UPDATE

控制算子

控制算子是一类用于处理特殊情况的节点，用于实现特殊的执行流程。

表 14-12 控制算子

算子	含义	场景
Result	直接进行计算	1. 不包含表扫描。 2. INSERT语句中只有一个VALUES子句。
ModifyTable	INSERT/UPDATE/ DELETE上层节点	INSERT/UPDATE/DELETE
Append	追加	1. UNION(ALL)。 2. 继承表。
MergeAppend	追加（输入有序）	1. UNION(ALL)。 2. 继承表。
RecursiveUnion	处理WITH子句中递归定义的UNION子查询	WITH RECURSIVE … SELECT … 语句。
BitmapAnd	Bitmap逻辑与操作	多维索引扫描的BitmapScan。
BitmapOr	Bitmap逻辑或操作	多维索引扫描的BitmapScan。
Limit	处理LIMIT子句	OFFSET … LIMIT …

其他算子

其他算子包括Stream算子，以及RemoteQuery等算子。Stream算子主要有三种类型：Gather stream、Broadcast stream及Redistribute stream。

- Gather stream：每个源节点都将其数据发送给目标节点进行汇聚。
- Broadcast stream：由一个源节点将其数据发给N个目标节点进行运算。
- Redistribute stream：每个源节点将其数据根据连接条件计算Hash值，根据重新计算的Hash值进行分布，发给对应的目标节点。

 版权所有 © 华为云计算技术有限公司

表 14-13 其他算子

算子	含义	场景
Stream	多节点数据交换	执行分布式查询计划，节点间存在数据交换。
Partition Iterator	分区迭代器	分区表扫描，迭代扫描每个分区。
RowToVec	行转列	行列混合场景。
DfsScan / DfsIndexScan	HDFS表（索引）扫描	HDFS表扫描。

14.4.4 SQL 调优流程

对慢SQL语句进行分析，通常包括以下步骤：

操作步骤

步骤1 收集SQL中涉及到的所有表的统计信息。在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧会造成执行计划严重劣化，从而导致性能问题。从经验数据来看，10%左右性能问题是因为没有收集统计信息。具体请参见**更新统计信息**。

步骤2 **审视和修改表定义**。

步骤3 通常情况下，有些SQL语句可以通过查询重写转换成等价的，或特定场景下等价的语句。重写后的语句比原语句更简单，且可以简化某些执行步骤达到提升性能的目的。查询重写方法在各个数据库中基本是通用的。**SQL语句改写规则**介绍了几种常用的通过改写SQL进行调优的方法。

步骤4 通过查看执行计划来查找原因。如果SQL长时间运行未结束，通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及实际运行情况，以便更精准地定位问题原因。有关执行计划的详细介绍请参见**SQL执行计划**。

步骤5 针对EXPLAIN或EXPLAIN PERFORMANCE信息，定位SQL慢的具体原因以及改进措施，具体参见**SQL调优进阶**。

步骤6 用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息等多个手段来进行执行计划的调优，以提升查询的性能。详细请参见**使用Plan Hint进行调优**。

步骤7 为了保证数据库性能的持续优质，建议**例行维护表**和**例行重建索引**。

步骤8 （可选）DWS支持在资源富足的情况下，通过算子并行来提升性能。详细请参见**SMP并行执行**。

----结束

14.4.5 更新统计信息

在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧会造成执行计划严重劣化，从而导致性能问题。

版权所有 © 华为云计算技术有限公司

背景信息

ANALYZE语句可收集与数据库中表内容相关的统计信息，统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，以生成最有效的执行计划。

建议在执行了大批量插入/删除操作后，例行对表或全库执行ANALYZE语句更新统计信息。目前默认收集统计信息的采样比例是30000行（即：guc参数default_statistics_target默认设置为100），如果表的总行数超过一定行数（大于1600000），建议设置guc参数default_statistics_target为-2，即按2%收集样本估算统计信息。

对于在批处理脚本或者存储过程中生成的中间表，也需要在完成数据生成之后显式的调用ANALYZE。

对于表中多个列有相关性且查询中有同时基于这些列的条件或分组操作的情况，可尝试收集多列统计信息，以便查询优化器可以更准确地估算行数，并生成更有效的执行计划。

生成统计信息

- 更新单个表的统计信息。
  ```
  ANALYZE tablename;
  ```
- 更新全库的统计信息。
  ```
  ANALYZE;
  ```
- 多列统计信息相关操作。
  - 收集tablename表的column_1、column_2列的多列统计信息
    ```
    ANALYZE tablename ((column_1, column_2));
    ```
  - 添加tablename表的column_1、column_2列的多列统计信息声明
    ```
    ALTER TABLE tablename ADD STATISTICS ((column_1, column_2));
    ```
  - 收集单列统计信息，并收集已声明的多列统计信息
    ```
    ANALYZE tablename;
    ```
  - 删除tablename表的column_1、column_2列的多列统计信息或其声明
    ```
    ALTER TABLE tablename DELETE STATISTICS ((column_1, column_2));
    ```

须知

- 在使用ALTER TABLE tablename ADD STATISTICS语句添加了多列统计信息声明后，系统并不会立刻收集多列统计信息，而是在下次对该表或全库进行ANALYZE时，进行多列统计信息的收集。如果想直接收集多列统计信息，请使用ANALYZE命令进行收集。
- 使用EXPLAIN查看各SQL的执行计划时，如果发现某个表SEQ SCAN的输出中rows=10，rows=10是系统给的默认值，有可能该表没有进行ANALYZE，需要对该表执行ANALYZE。

提升统计信息质量

ANALYZE是按照随机采样算法从表上采样，根据样本计算表数据特征。采样数可以通过配置参数default_statistics_target进行控制，default_statistics_target取值范围为-100~10000，默认值为100。

版权所有 © 华为云计算技术有限公司

- 当default_statistics_target > 0时；采样的样本数为300*default_statistics_target，default_statistics_target取值越大，采样的样本也越大，样本占用的内存空间也越大，统计信息计算耗时也越长。
- 当default_statistics_target < 0时，采样的样本数为 (default_statistics_target)/100*表的总行数，default_statistics_target取值越小，采样的样本也越大。当default_statistics_target < 0时会把采样数据下盘，不存在样本占用的内存空间的问题，但是因为样本过大，计算耗时长的问题同样存在。

  default_statistics_target < 0时，实际采样数是 (default_statistics_target)/100*表的总行数，所以又称之为百分比采样。

自动收集统计信息

当配置参数autoanalyze打开时，查询语句进入优化器发现表不存在统计信息或数据变化超过阈值时，将自动触发统计信息收集，以满足优化器的需求。

基于代价的优化器模型（CBO, cost base optimizer）中，统计信息决定了查询计划生成的好坏。因此，统计信息的及时有效很重要。

- 表级统计信息，存储在pg_class的relpages, reltuples中。
- 列级统计信息，存储在pg_statistics中，可以通过pg_stats视图查看。包括：NULL值比例，distinct值占比，高频值MCV，直方图histogram等。

收集条件：当数据量变化超过10%时，认为数据特征已发生变化，需重新收集统计信息。

总体策略：开启动态采样保证统计信息及时性，开启轮询采样保证统计信息持久化，查询性能敏感（秒级响应）靠手动采样。

基本规则

表 14-14 统计信息收集相关基础功能

功能	介绍	特点	约束
手动采样	作业中修改大量数据后，手动执行ANALYZE。	• normal模式，统计信息存系统表，全局共享。四级锁，同一张表不能并发。 • light模式，统计信息存内存，全局共享。一级锁，同一张表可以并发。 • force模式，在锁定统计信息情况下，也可以强制采样，行为与normal一致。 语法：ANALYZE tablename; ANALYZE (light|force) tablename;	-

 版权所有 © 华为云计算技术有限公司

功能	介绍	特点	约束
轮询采样	后台线程，根据阈值轮询维护统计信息。	仅支持normal模式，统计信息存系统表，共享。四级锁，同一张表不能并发。 相关GUC参数： ● autovacuum ● autovacuum_mode ● autovacuum_analyze_threshold ● autovacuum_analyze_scale_factor	异步轮询触发。
动态采样	查询解析时，根据阈值，用几十秒的代价实时维护统计信息。	● normal模式，统计信息存系统表，全局共享。四级锁，同一张表不能并发。 ● light模式，统计信息存内存，全局共享。一级锁，同一张表可以并发。 相关GUC参数： ● autoanalyze ● autoanalyze_mode	随查询实时触发。轻量化时依赖轮询采样进行持久化。
强制采样	SQL中通过hint，强制每次查询都收集统计信息。	用于数据特征敏感场景，确保查询具有最新的统计信息。 用法：select /*+ lightanalyze (t1 1) */ from t1; 1：强制采样，0：禁止采样	需要修改SQL。
分区统计信息	按分区收集，仅收增量，自动合并全局信息。	用于超大分区表的场景，确保分区剪枝后的查询代价估算准确。	多占存储，信息更准。
多列统计信息	根据多个列组合收集统计信息。	用于多列同时过滤场景，确保对多列组合的查询代价估算准确。	手动识别，临时表方式。
表达式统计信息	根据表达式函数对某列收集统计信息。	用于批量的表达式过滤的场景，确保查询表达式的查询代价估算准确。	手动识别。
表达式索引信息	创建的表达式索引会自动收集统计信息。	用于点查的表达式过滤的场景，确保查询表达式的查询代价估算准确。	手动识别。
冻结统计信息	将表级的统计信息冻结，防止发生变化。	用于数据特征极稳定场景，禁止采样，防止查询计划跳变。 用于数据特征极易变场景，强制采样，确保每次查询都采样。 参数：表级属性analyze_mode	-

 版权所有 © 华为云计算技术有限公司

功能	介绍	特点	约束

修改统计信息	手动计算后，直接修改统计信息。	继续低采样率，但可以手动计算后进行校准。用法：
		select approx_count_distinct(col_name) from table_name;
		alter table set (n_distinct=xxx);
			-

分区信息拷贝	可将旧分区的统计信息拷贝到新分区。	用于数据特征变化不大的分区表，减少统计信息收集的开销。	-
统计信息推理	基于旧的统计信息自动推算更准确的统计信息。	通过GUC参数enable_extrapolation_stats设置。	-
统计信息备份恢复	通过explain (stat on)将统计信息备份成SQL。	用于场景复现或统计信息还原。	导出SQL的形式。

场景和策略

列举了常见的数据加工场景和对应的统计信息收集策略。

表 14-15 统计信息收集策略

场景	特点	策略
流式增量加工	数据流式增量变化，无合理ANALYZE时机。	开启动态采样，查询按需自动收集统计信息，且全局共享。

在线批量加工（数据湖）	数据加工与查询会并发，要求查询稳定。	开启动态采样，或一个事务中完成数据加工和ANALYZE。
		begin;
		truncate table or partition;
		copy/merge/insert overwrite
		ANALYZE (light) tablename;
		end;

分区并行加工	不同分区并发加工数据。	开启“动态采样”或“手动light采样”，同表可并发收集
宽表场景	百列以上的宽表。	1. 启动动态采样的自动谓词管理。 2. 仅收集前N列的统计信息。 3. 根据查询中常用谓词，可列级设置是否参与采样。
大表场景	• 表的数据量大，变化难以达到阈值。 • 统计信息易变。	调低动态采样触发阈值。

版权所有 © 华为云计算技术有限公司

场景	特点	策略
特征敏感场景	• 数据特征易变，查询计划不稳定。 • 需要强制收集。	1. 调低动态采样触发阈值。 2. 在SQL中通过HINT方式强制light动态采样。 3. 清空并冻结统计信息，查询每次都会重新收集且不共享。
高并发场景	同一张表会高并发查询（10并发以上）同时触发动态采样占用资源多。	1. 关闭并发，其他查询不使用最新统计信息。 2. 其他查询等最新统计信息生成，再执行查询（开发中）。
流式性能敏感	流式加工，秒级查询或整体资源高水位。	表级或SQL级禁止动态采样，依靠后台轮询采样。
批量性能敏感	批量加工，秒级查询或整体资源高水位。	加工时手动收集统计信息。

资源消耗

表 14-16 资源消耗

类别	详细分类	说明
CPU	谓词列管理	自动管理谓词，仅对查询中使用的列收集统计信息。 手动屏蔽非谓词列，可手动设置哪些列不收集统计信息。
	超长列统计	可截断的数据类型，仅收取前1024字符计算统计信息。
IO	默认采集3万条样本	与表大小无关，与列数，分区数，小CU数有关。
内存	Buffer占用	最多只占用cstore buffer中的一个槽位。
	内存零拷贝	从buffer取出样本后，直接进行统计信息计算，无需组织成tuple再传递。
	内存自适应	可配置为内存不足时自动转为使用临时表采样。动态采样为防止查询触发建临时表，不会自适应。 通过GUC参数analyze_stats_mode设置。
	内存的大小	控制ANALYZE时最大使用内存，超过后或下盘或自动减少样本。 通过GUC参数maintenance_work_mem设置。
锁	四级锁	normal模式，分布式申请四级锁。与增删改不冲突，与DDL，VACUUM，ANALYZE，REINDEX冲突。
	一级锁	light模式，仅本地一级锁。仅与DDL冲突。

版权所有 © 华为云计算技术有限公司

准确性及可靠性

表 14-17 准确性及可靠性

准确性及可靠性	分类	说明
准确性	采样大小	可配置为按表大小自适应。由参数default_statistics_target控制。
	采样随机性	• analyze_sample_mode参数设置新支持优化蓄水池和range采样随机性更优。 • random_function_version参数增强了随机数计算函数的随机性。
	全局共享	统计信息可以跨会话和跨节点进行共享。
	修改计数广播	后台线程会轮询全局检查修改计数并进行广播。 作业线程也可直接广播修改计数。通过参数tuple_change_sync_threshold设置。 跨CN修改和查询也影响不大，修改计数会异步方式广播同步。
	调整CU采样比	CU填充率不高的场景，可调大CU采样比。通过列级参数cstore_cu_sample_ratio设置。
	distinct固定	随机采样后distinct值不稳定场景，不调高采样率情况下，可进行distinct值固定。通过列级参数n_distinct设置。
	统计信息推算	enable_extrapolation_stats参数可以控制估算失真时，基于旧的统计信息自动推算更准确的统计信息。
可靠性	CN故障	其他CN故障时，不影响动态采样，统计信息不同步，不影响当前CN查询的统计信息质量。
	CN恢复	其他CN恢复后，再次查询时会强制动态采样，并全局同步。
	DN故障	非当前逻辑集群的DN故障，不影响本逻辑集群的动态采样。

运维监控

DWS通过在ANALYZE命令后面追加注释的方式，详细展示了ANALYZE运行模式和各种执行阶段，主要通过以下视图体现：

- 当前活跃会话视图pgxc_stat_activity的query字段。
- 当前线程等待视图pgxc_thread_wait_status的wait_status字段。

ANALYZE命令追加注释的内容格式为：--Action-RunMode-StatsMode-SyncMode

 版权所有 © 华为云计算技术有限公司

- Action取值及含义：
  ```
  {"begin", "finished", "lock FirstCN", "estimate rows", "statistics", "sample rows", "calc stats"};
  ```
  分别表示：开始，结束，在向FirstCN申锁，在一阶段估算行数信息，在二阶段真正执行ANALYZE，在采集样本，在计算统计信息。
- RunMode取值及含义：
  ```
  {"manual", "backend", "normal runtime", "light runtime", "light runtime inxact", "light estimate rows", "light manual"};
  ```
  分别表示：手动模式，后台轮询模式，normal动态采样，light动态采样，事务内的light动态采样，light仅估算函数，手动light模式。
- StatsMode取值及含义：
  ```
  {"dynamic", "memory", "smptbl"};
  ```
  分别表示：自适应选择内存或临时表放样本，仅使用内存放样本，仅使用临时表放样本。
- SyncMode取值及含义：
  ```
  {"sync", "nosync"};
  ```
  分别表示：向所有CN同步统计信息，不同步统计信息。

示例：

```
SELECT coorname,datid,datname,pid,usename,application_name,query_id,query
FROM pgxc_stat_activity WHERE query like '%analyze%' and query not like '%application_name%';
    coorname | datid | datname | pid | usename | application_name | query_id | query
--------------+-------+----------+-----------------+-----------+------------------+-------------------
+----------------------------------------
coordinator1 | 15676 | postgres | 139919333779200 | test | gsql | 73183493944770822 | analyze t_1;
coordinator2 | 15676 | postgres | 140217336461056 | test | coordinator1 | 73183493944770822 | analyze public.t_1;--push stats-manual-memory-sync
coordinator3 | 15676 | postgres | 139944245847808 | test | coordinator1 | 73183493944770822 | analyze public.t_1;--push stats-manual-memory-sync
(3 rows)
```

统计信息查看

- 查看动态采样的内存统计信息。
  - 查内存中表级统计信息：
    ```
    SELECT * FROM pv_runtime_relstats;
    ```
  - 查内存中列级统计信息：
    ```
    SELECT * FROM pv_runtime_attstats;
    ```
- 查看系统表中统计信息。
  - 查系统表中表级统计信息：
    ```
    select relname, relpages, reltuples from pg_class;
    ```
  - 查系统表中列级统计信息：
    ```
    SELECT * FROM pg_stats;
    ```
- 查看统计信息最近收集时间。

  动态采样把统计信息放内存中，不更新该系统表的时间。

  ```
  SELECT * FROM pg_object;
  ```

14.4.6 审视和修改表定义

在分布式框架下，数据分布在各个DN上。一个或者几个DN的数据存在一块物理存储设备上，较好的表定义需要满足以下要求：

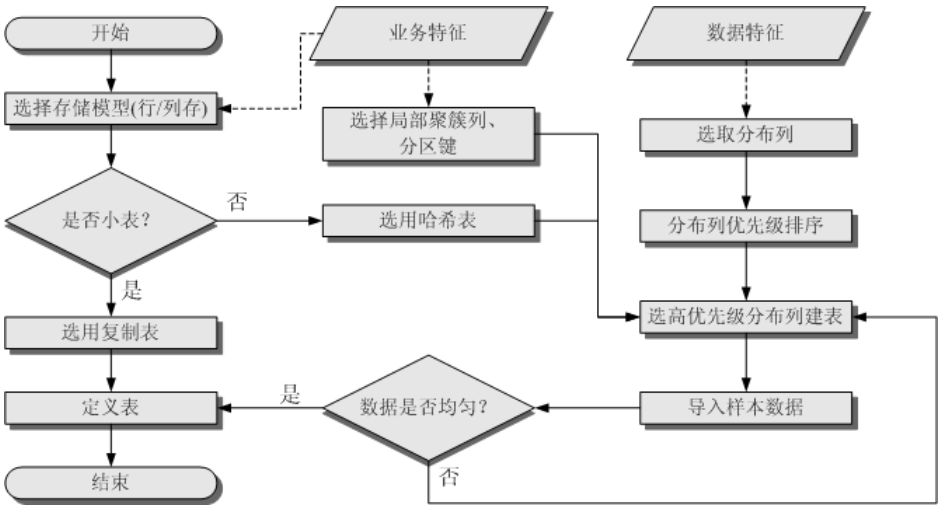
1. **表数据均匀分布在各个DN上**，以防止单个DN对应的存储设备空间不足造成集群有效容量下降。选择合适分布列，避免数据分布倾斜可以实现该点。

版权所有 © 华为云计算技术有限公司

2. **表Scan压力均匀分散在各个DN上**，以避免单DN的Scan压力过大，形成Scan的单节点瓶颈。分布列不选择基表上等值filter中的列可以实现该点。
3. **减少扫描数据量**。通过分区的剪枝机制可以实现该点。
4. **尽量减少随机IO**。通过聚簇/局部聚簇可以实现该点。
5. **尽量避免数据shuffle**，减小网络压力。通过选择join-condition或者group by列为分布列可以最大程度的实现这点。

从上述描述来看表定义中最重要的一点是分布列的选择。创建表定义一般遵循图14-5所示流程。表定义在数据库设计阶段创建，在SQL调优过程中进行审视和修改。

图 14-5 表定义流程



审视和修改表定义的具体操作方法，请参见**基于表结构设计和调优提升DWS查询性能**。

14.4.7 SQL 调优进阶

14.4.7.1 SQL 自诊断

用户在执行INSERT/UPDATE/DELETE/SELECT/MERGE INTO或者CREATE TABLE AS语句时，可能会遇到性能问题。产品内置集成了性能自动诊断功能，并把相关的诊断信息保存到**实时TopSQL**中，当配置参数**enable_resource_track**为on的时候，这些诊断信息会转存到**历史TopSQL**中。通过查询**GS_WLM_SESSION_STATISTICS**，**GS_WLM_SESSION_HISTORY**，**GS_WLM_SESSION_INFO**视图的warning字段可以获得对应的性能诊断信息，为性能调优提供参考。

- SQL自诊断的告警类型与**resource_track_level**的设置相关。

  当“resource_track_level”设置为query时，可以诊断多列/单列统计信息未收集、分区未剪枝告警和SQL不下推的告警等非执行态的诊断信息；“resource_track_level”设置为perf或operator时，可以诊断所有的告警场景。
- SQL自诊断的诊断范围与**resource_track_cost**的设置相关。

  当SQL的代价大于“resource_track_cost”时，SQL才会被诊断。SQL的代价可以通过explain来确认。
- 执行EXPLAIN PERFORMANCE或者EXPLAIN VERBOSE的时候，除缺乏多列统计信息之外的SQL自诊断信息也会输出，具体请参考**SQL执行计划**。

版权所有 © 华为云计算技术有限公司

SQL 执行性能相关告警场景

目前支持对以下10种导致性能问题的场景上报告警。

1. 多列/单列统计信息未收集。

   如果存在单列或者多列统计信息未收集，则上报相关告警。对于这种告警，建议的优化方案是对相关表进行ANALYZE，可参考**更新统计信息**和**统计信息调优**。

   需要特别注意的是，如果查询语句中的OBS外表和HDFS外表未收集统计信息，也会上报统计信息未收集的告警，因为OBS外表和HDFS外表的ANALYZE性能比较差，一般不建议对其进行ANALYZE，但是建议使用ALTER FOREIGN TABLE的语法修改外表的totalrows属性，从而修正外表的行数估算。

   告警信息示例：

   整表的统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t1
   ```

   单列统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t2(c1)
   ```

   多列统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t3((c1,c2))
   ```

   单列和多列统计信息未收集：

   ```
   Statistic Not Collect
         schema_test.t4(c1)
         schema_test.t5((c1,c2))
   ```

2. 分区不剪枝。

   分区表查询时，常会期望通过分区键上的约束条件进行分区剪枝，从而提升分区表查询性能，但有时候会因为约束条件书写不当，导致分区表没有剪枝，出现查询性能问题，具体请参见**案例：改写SQL排除剪枝干扰**。

3. SQL不下推。

   对于不下推的SQL，尽可能详细上报导致不下推的原因。调优方法可参考案例**语句下推调优**。

   导致不下推的因素主要有以下两点：

   - 函数导致的不下推

     诊断信息中会给出具体的函数名称。函数下推是由函数的shippable属性决定，具体可参见《数据仓库服务SQL语法参考》中“CREATE FUNCTION”语法。

   - 语法导致的不下推

     诊断信息中会提示导致不下推的具体语法，例如：含有With Recursive，Distinct On，row表达式，返回值为record类型的，会告警相应语法不支持下推等等。

   告警信息示例：

   ```
   SQL is not plan-shipping
         "enable_stream_operator" is off

   SQL is not plan-shipping
         "Distinct On" can not be shipped

   SQL is not plan-shipping
         "v_test_unshipping_log" is VIEW that will be treated as Record type can't be shipped
   ```


版权所有 © 华为云计算技术有限公司

4. 不支持向量化计划

   对于不能生成向量化计划的SQL，尽可能详细地上报不支持向量化的原因。

   常见的不支持向量化的因素主要有：

   - 目标列含有返回类型是集合类型的函数。
   - 目标列或查询条件、Stream算子的分布键、Limit和Offset子句含有不可向量化的表达式（地理空间类型、数组表达式、Row表达式、XML表达式、参数或返回值有refcursor类型的函数等）。
   - Group By子句含有数组等值判断表达式。
   - GC_FDW和LOG_FDW不支持向量化。
   - 计划含有Cte Scan、Recursive Union、Merge Append、Lock Rows等算子。

   告警信息示例：

   ```
   SQL is un-vectorized
         Function regexp_split_to_table that returns set is un-vectorized

   SQL is un-vectorized
         Array expression is un-vectorized

   SQL is un-vectorized
         Function array_agg is un-vectorized

   SQL is un-vectorized
         RecursiveUnion is un-vectorized
   ```

5. HashJoin中大表做内表。

   如果在表连接过程中使用了Hashjoin（可通过**GS_WLM_SESSION_HISTORY**的“query_plan”字段查看），且连接的内表行数是外表行数的10倍或以上；同时内表在每个DN上的平均行数大于10万行，且发生了下盘，则上报相关告警。针对这种场景，需要调整HashJoin内外表顺序，具体调优方法参考**Join顺序的Hint**。

   告警信息示例：

   ```
   Execute diagnostic information
         PlanNode[7] Large Table is INNER in HashJoin “Vector Hash Aggregate”
   ```

   其中，7为“query_plan”字段文本中编号为7的算子。

6. 大表等值连接使用Nestloop

   如果在表连接过程中使用了nestloop（可通过**GS_WLM_SESSION_HISTORY**的query_plan字段查看），并且两个表中较大表的行数平均每个DN上的行数大于10万行、表的连接中存在等值连接，则上报相关告警。针对这种场景，需要调整表关联方式，禁止当前内外表之间使用NestLoop的关联方式，具体调优方法参考**Join方式的Hint**。

   告警信息示例：

   ```
   Execute diagnostic information
         PlanNode[5] Large Table with Equal-Condition use Nestloop"Nested Loop"
   ```

7. 大表Broadcast

   如果在Broadcast算子中，平均每DN的行数大于10万行，则告警大表Broadcast。针对这种场景，需要禁止Broadcast下层算子做Broadcast动作，具体调优方法参考**Stream方式的Hint**。

   告警信息示例：

   ```
   Execute diagnostic information
         PlanNode[5] Large Table in Broadcast "Streaming(type: BROADCAST dop: 1/2)"
   ```

8. 数据倾斜

 版权所有 © 华为云计算技术有限公司

某表在各DN上的分布，存在某DN上的行数是另一DN上行数的10倍或以上，且有DN中的行数大于10万行，则上报相关告警。该告警一般分为存储层倾斜和计算层倾斜，具体调优方法参考**数据倾斜调优**。

告警信息示例：

```
Execute diagnostic information
    PlanNode[6] DataSkew:"Seq Scan", min_dn_tuples:0, max_dn_tuples:524288
```

9. 索引不合理

   在基表扫描时，满足下述条件则上报相关告警：

   - 对于行存表：
     - 使用索引扫描，输出行数/扫描行数>1/1000且输出行数>1万行。
     - 使用顺序扫描，输出行数/扫描行数<1/1000且输出行数<=1万行、扫描行数>1万行。
   - 对于列存表：
     - 使用索引扫描，输出行数/扫描行数>1/10000且输出行数>100。
     - 使用顺序扫描，输出行数/扫描行数<1/10000且输出行数<=100、扫描行数>1万行。

   调优方法可参考**算子级调优**，也可参考案例**案例：建立合适的索引**和**案例：使用partial cluster key**。

   告警信息示例：

   ```
   Execute diagnostic information
       PlanNode[4] Indexscan is not properly used:"Index Only Scan", output:524288, filtered:0, rate:1.00000
       PlanNode[5] Indexscan is ought to be used:"Seq Scan", output:1, filtered:524288, rate:0.00000
   ```

   需要注意的是，这个诊断结果只是针对当前SQL的建议，是否创建索引要结合整体业务综合分析，对于高频的过滤条件才建议创建索引。

10. 估算不准

对于平均每DN行数如果优化器的估算行数和实际行数中的较大值大于10万行，并且估算行数和实际行数中较大值是较小值的10倍或以上，则上报相关告警。针对这种场景，可以参照**行数的Hint**修正行数估算，让优化器在正确的行数基础上重新规划执行计划。

告警信息示例：

```
Execute diagnostic information
    PlanNode[5] Inaccurate Estimation-Rows: "Hash Join" A-Rows:0, E-Rows:52488
```

规格约束

1. 告警字符串长度上限为2048。如果告警信息超过这个长度（例如存在大量未收集统计信息的超长表名，列名等信息）则不告警，只上报warning：
   ```
   WARNING, "Planner issue report is truncated, the rest of planner issues will be skipped"
   ```
2. 如果query存在limit节点（即查询语句中包含limit），则不会上报limit节点以下的Operator级别的告警。
3. 对于“数据倾斜”和“估算不准”两种类型告警，在某一个plan树结构下，只上报下层节点的告警，上层节点不再重复告警。这主要是因为这两种类型的告警可能是因为底层触发上层的。例如，如果在scan节点已经存在数据倾斜，那么在上层的hashagg等其他算子很可能也出现数据倾斜。

 版权所有 © 华为云计算技术有限公司

14.4.7.2 语句下推调优

语句下推介绍

目前，DWS优化器在分布式框架下指定语句的执行策略时，有三种执行计划方式：生成下推语句计划、生成分布式执行计划以及发送语句的分布式执行计划。

- 下推语句计划：指直接将查询语句从CN发送到DN进行执行，然后将执行结果返回给CN。
- 分布式执行计划：指CN对查询语句进行编译和优化，生成计划树，再将计划树发送给DN进行执行，并在执行完毕后返回结果到CN。
- 发送语句的分布式执行计划：上述两种方式都不可行时，将可下推的查询部分组成查询语句（多为基表扫描语句）下推到DN进行执行，获取中间结果到CN，然后在CN执行剩下的部分。

在“发送语句的分布式执行计划”策略中，要将大量中间结果从DN发送到CN，并且要在CN运行不能下推的部分语句，会导致CN成为性能瓶颈（带宽、存储、计算等）。在进行性能调优的时候，应尽量避免只能选择该策略的查询语句。

执行语句不能下推是因为语句中含有**不支持下推的函数**或者**不支持下推的语法**。一般都可以通过等价改写规避执行计划不能下推的问题。

查看执行计划是否下推

执行计划是否下推可以依靠如下方法快速判断：

步骤1 将GUC参数“**enable_fast_query_shipping**”设置为off，使查询优化器使用分布式框架策略。

```
SET enable_fast_query_shipping = off;
```

步骤2 查看执行计划。

如果执行计划中有Data Node Scan节点，那么此执行计划为不可下推的执行计划；如果执行计划中有Streaming节点，那么计划是可以下推的。

例如如下业务SQL：

```
select
count(ss.ss_sold_date_sk order by ss.ss_sold_date_sk)c1
from store_sales ss, store_returns sr
where
sr.sr_customer_sk = ss.ss_customer_sk;
```

执行计划如下，可以看出此SQL语句不能下推。

```
                          QUERY PLAN
------------------------------------------------------------------------------
Aggregate
->  Hash Join
Hash Cond: (ss.ss_customer_sk = sr.sr_customer_sk)
->  Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
Node/s: All datanodes
->  Hash
->  Data Node Scan on store_returns "_REMOTE_TABLE_QUERY_"
Node/s: All datanodes
(8 rows)
```

----结束

版权所有 © 华为云计算技术有限公司

不支持下推的语法

以如下三个表定义说明不支持下推的SQL语法。

```
CREATE TABLE CUSTOMER1
(
    C_CUSTKEY     BIGINT NOT NULL
  , C_NAME        VARCHAR(25) NOT NULL
  , C_ADDRESS     VARCHAR(40) NOT NULL
  , C_NATIONKEY   INT NOT NULL
  , C_PHONE       CHAR(15) NOT NULL
  , C_ACCTBAL     DECIMAL(15,2)   NOT NULL
  , C_MKTSEGMENT  CHAR(10) NOT NULL
  , C_COMMENT     VARCHAR(117) NOT NULL
)
DISTRIBUTE BY hash(C_CUSTKEY);
CREATE TABLE test_stream(a int,b float); --float不支持重分布
CREATE TABLE sal_emp ( c1 integer[] ) DISTRIBUTE BY replication;
```

- 不支持returning语句下推。

```
explain update customer1 set C_NAME = 'a' returning c_name;
                              QUERY PLAN
---------------------------------------------------------------------
 Update on customer1  (cost=0.00..0.00 rows=30 width=187)
   Node/s: All datanodes
   Node expr: c_custkey
   ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=187)
         Node/s: All datanodes
(5 rows)
```

- count(distinct expr)中的字段不支持重分布，则不支持下推。

```
explain verbose select count(distinct b) from test_stream;
                                    QUERY PLAN
---------------------------------------------------------------------------- Aggregate  (cost=2.50..2.51 rows=1 width=8)
   Output: count(DISTINCT test_stream.b)
   ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=8)
         Output: test_stream.b
         Node/s: All datanodes
         Remote query: SELECT b FROM ONLY public.test_stream WHERE true
(6 rows)
```

- 不支持distinct on用法下推。

```
explain verbose select distinct on (c_custkey) c_custkey from customer1 order by c_custkey;
                                     QUERY PLAN
---------------------------------------------------------------------------- Unique  (cost=49.83..54.83 rows=30 width=8)
   Output: customer1.c_custkey
   ->  Sort  (cost=49.83..52.33 rows=30 width=8)
         Output: customer1.c_custkey
         Sort Key: customer1.c_custkey
         ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
width=8)
               Output: customer1.c_custkey
               Node/s: All datanodes
               Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
(9 rows)
```

- Fulljoin的join列如果不支持重分布，则不支持下推。

```
explain select * from test_stream t1 full join test_stream t2 on t1.a=t2.b;
                                     QUERY PLAN
---------------------------------------------------------------------------- Hash Full Join  (cost=0.38..0.82 rows=30
width=24)
   Hash Cond: ((t1.a)::double precision = t2.b)
   ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=12)
         Node/s: All datanodes
   ->  Hash  (cost=0.00..0.00 rows=30 width=12)
         ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
width=12)
               Node/s: All datanodes
(7 rows)
```

 版权所有 © 华为云计算技术有限公司

- 不支持数组表达式下推。

```
explain verbose select array[c_custkey,1] from customer1 order by c_custkey;

                        QUERY PLAN
------------------------------------------------------------------------ Sort  (cost=49.83..52.33 rows=30 width=8)
   Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
   Sort Key: customer1.c_custkey
   ->  Data Node Scan on "__REMOTE_SORT_QUERY__"  (cost=0.00..0.00 rows=30 width=8)
         Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
         Node/s: All datanodes
         Remote query: SELECT ARRAY[c_custkey, 1::bigint], c_custkey FROM ONLY public.customer1
WHERE true ORDER BY 2
(7 rows)
```

- Subplan在多个线程间共享不支持下推。

```
postgres=# explain verbose  select c_custkey in (select c_custkey from customer1) b from customer1;
                        QUERY PLAN
------------------------------------------------------------------------
Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=2.50..5.00 rows=1000 width=8)
   Output: (hashed SubPlan 1)
   Node/s: All datanodes
   Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
   SubPlan 1
     ->  Data Node Scan on customer "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=1000
width=8)
           Output: public.customer.c_custkey
           Node/s: All datanodes
           Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
(9 rows)
```

- With Recursive当前版本不支持下推的场景和原因如下：

序号	场景	不下推原因
1	包含外表、HDFS表的查询场景	LOG: SQL can't be shipped, reason: RecursiveUnion contains HDFS Table or ForeignScan is not shippable（LOG为CN日志中打印的不下推原因，下同） 外表、HDFS表，当前版本暂不支持下推。
2	多nodegroup场景	LOG: SQL can't be shipped, reason: With-Recursive under multi-nodegroup scenario is not shippable 基表存储nodegroup不相同，或者计算nodegroup与基表不相同，当前版本暂不支持下推。
3	WITH recursive t_result AS (SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm > 10 UNION SELECT t2.dm,t2.sj_dm,t2.name||' > '|| t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive does not contain "ALL" to bind recursive & none-recursive branches UNION不带ALL，需要去重。

版权所有 © 华为云计算技术有限公司

序号	场景	不下推原因
4	WITH RECURSIVE x(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 5), y(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 10) SELECT y.*, x.* FROM y LEFT JOIN x USING (id) ORDER BY 1;	LOG: SQL can't be shipped, reason: With-Recursive contains system table is not shippable 基表中有系统表。
5	WITH RECURSIVE t(n) AS (VALUES (1) UNION ALL SELECT n+1 FROM t WHERE n < 100) SELECT sum(n) FROM t;	LOG: SQL can't be shipped, reason: With-Recursive contains only values rte is not shippable 基表扫描只有VALUES子句，仅在CN上即可完成执行。
6	select a.ID,a.Name, (with recursive cte as (select ID, PID, NAME from b where b.ID = 1 union all select parent.ID,parent.PID,parent.NAME from cte as child join b as parent on child.pid=parent.id where child.ID = a.ID) select NAME from cte limit 1) cName from (select id, name, count(*) as cnt from a group by id,name) a order by 1,2;	LOG: SQL can't be shipped, reason: With-Recursive recursive term correlated only is not shippable 相关子查询的关联条件仅在递归部分，非递归部分无关联条件。
7	WITH recursive t_result AS (select * from(SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm < 10 order by dm limit 6 offset 2) UNION all SELECT t2.dm,t2.sj_dm,t2.name||' > '|| t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive contains conflict distribution in none-recursive(Replicate) recursive(Hash) 非递归部分带limit为Replicate计划，递归部分为Hash计划，计划存在冲突。

版权所有 © 华为云计算技术有限公司

序号	场景	不下推原因
8	with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from (with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from rec_tb4 h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID) h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID,1,2,3;	LOG: SQL can't be shipped, reason: Recursive CTE references recursive CTE "cte" 多层Recursive嵌套，即recursive的递归部分又嵌套另一个recursive查询。

不支持下推的函数

首先介绍函数的易变性。在DWS中共分三种形态：

- **IMMUTABLE**

  表示该函数在给出同样的参数值时总是返回同样的结果。

- **STABLE**

  表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。

- **VOLATILE**

  表示该函数值可以在一次表扫描内改变，因此不会做任何优化。

函数易变性可以查询pg_proc的provolatile字段获得，i代表IMMUTABLE，s代表STABLE，v代表VOLATILE。另外，在pg_proc中的proshippable字段，取值范围为t/f/NULL，这个字段与provolatile字段一起用于描述函数是否下推。

- 如果函数的provolatile属性为i，则无论proshippable的值是否为t，则函数始终可以下推。
- 如果函数的provolatile属性为s或v，则仅当proshippable的值为t时，函数可以下推。
- random如果出现CTE中，也不下推。因为此场景中下推可能出现结果错误。

对于用户自定义函数，可以在创建函数的时候指定provolatile和proshippable属性的值，详细请参考CREATE FUNCTION。

对于函数不能下推的场景：

- 如果是系统函数，建议根据业务等价替换这个函数。
- 如果是自定义函数，建议分析客户业务场景，看函数的provolatile和proshippable属性定义是否正确。

 版权所有 © 华为云计算技术有限公司

实例分析：自定义函数

对于自定义函数，如果对于确定的输入，有确定的输出，则应将函数定义为immutable类型。

利用TPCDS的销售信息举例，需要写一个函数来获取商品的打折情况，即定义一个计算折扣的函数：

```
CREATE FUNCTION func_percent_2 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
VOLATILE;
```

执行下列语句：

```
SELECT func_percent_2(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
   Output: func_percent_2(store_sales.ss_sales_price, store_sales.ss_list_price)
   Remote query: SELECT ss_sales_price, ss_list_price FROM ONLY store_sales WHERE true
(3 rows)
```

可见，func_percent_2并没有被下推，而是将ss_sales_price和ss_list_price收到CN上，再进行计算，消耗大量CN的资源，而且计算缓慢。

由于该自定义函数对确定的输入有确定的输出，如果将该自定义函数改为：

```
CREATE FUNCTION func_percent_1 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
IMMUTABLE;
```

执行语句：

```
SELECT func_percent_1(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan on "__REMOTE_FQS_QUERY__"  (cost=0.00..0.00 rows=0 width=0)
   Output: (func_percent_1(store_sales.ss_sales_price, store_sales.ss_list_price))
   Node/s: All datanodes
   Remote query: SELECT public.func_percent_1(ss_sales_price, ss_list_price) AS func_percent_1 FROM public.store_sales
(4 rows)
```

可见函数func_percent_1被下推到DN执行，提升了执行效率（TPCDS 1000X，3CN18DN，查询效率提升100倍以上）。

其他典型不下推案例集

请参考**案例：典型不下推语句整改案例集**。

14.4.7.3 子查询调优

子查询背景介绍

应用程序通过SQL语句来操作数据库时会使用大量的子查询，这种写法比直接对两个表做连接操作在结构上和思路上更清晰，尤其是在一些比较复杂的查询语句中，子查询有更完整、更独立的语义，会使SQL对业务逻辑的表达更清晰更容易理解，因此得到了广泛的应用。

版权所有 © 华为云计算技术有限公司

DWS根据子查询在SQL语句中的位置把子查询分成了子查询、子链接两种形式。

- 子查询SubQuery：对应于查询解析树中的范围表RangeTblEntry，更通俗一些指的是出现在FROM语句后面的独立的SELECT语句。
- 子链接SubLink：对应于查询解析树中的表达式，更通俗一些指的是出现在where/on子句、targetlist里面的语句。

  综上，对于查询解析树而言，SubQuery的本质是范围表，而SubLink的本质是表达式。针对SubLink场景而言，由于SubLink可以出现在约束条件、表达式中，按照DWS对sublink的实现，sublink可以分为以下几类：

  - exist_sublink：对应EXISTS、NOT EXISTS语句
  - any_sublink：对应op Any(select…)语句，其中OP可以是IN,<,>,=操作符
  - all_sublink：对应op ALL(select…)语句，其中OP可以是IN,<,>,=操作符
  - rowcompare_sublink：对应record op (select …)语句
  - expr_sublink：对应(SELECT with single targetlist item ...)语句
  - array_sublink：对应ARRAY(select…)语句
  - cte_sublink：对应with query(…)语句

须知

在逻辑集群模式下，当执行的查询语句中存在子查询与父查询执行集群不一致，且子查询中存在与父查询关联的过滤条件时，由于相关条件无法跨逻辑集群下推，导致计划产生额外的算子开销。该场景下性能可能会劣于单逻辑集群场景。

其中OLAP、HTAP场景中常用的sublink为exist_sublink、any_sublink，在DWS的优化引擎中对其应用场景做了优化（子链接提升），由于SQL语句中子查询使用的灵活性，会带来SQL子查询过于复杂而造成的性能问题。子查询从大类上来看，分为非相关子查询和相关子查询：

- **非相关子查询None-Correlated SubQuery**

  子查询的执行不依赖于外层父查询的任何属性值。这样子查询具有独立性，可独自求解，形成一个子查询计划先于外层的查询求解。

  例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c2 IN (2,3,4)
);
                          QUERY PLAN
------------------------------------------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Hash Right Semi Join
        Hash Cond: (t2.c2 = t1.c1)
        ->  Streaming(type: REDISTRIBUTE)
              Spawn on: All datanodes
              ->  Seq Scan on t2
                    Filter: (c2 = ANY ('{2,3,4}'::integer[]))
        ->  Hash
              ->  Seq Scan on t1
(10 rows)
```

 版权所有 © 华为云计算技术有限公司

- **相关子查询Correlated-SubQuery**

  子查询的执行依赖于外层父查询的一些属性值（如下列示例t2.c1 = t1.c1条件中的t1.c1）作为内层查询的一个AND-ed条件。这样的子查询不具备独立性，需要和外层查询按分组进行求解。

  例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = t1.c1 AND t2.c2 in (2,3,4)
);
                                QUERY PLAN
-------------------------------------------------------------------------
Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t2.c1 = t1.c1)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                           Spawn on: All datanodes
                           ->  Seq Scan on t2
                                 Filter: (c2 = ANY ('{2,3,4}'::integer[]))
(12 rows)
```

DWS 对 SubLink 的优化

针对SubLink的优化策略主要是让内层的子查询提升（pullup），能够和外表直接做关联查询，从而避免生成SubPlan+Broadcast内表的执行计划。判断子查询是否存在性能风险，可以通过explain查询语句查看Sublink的部分是否被转换成SubPlan+Broadcast的执行计划。

例如：

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = t1.c1
);
```

```
QUERY PLAN
-------------------------------------------------------------
Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t2.c1 = t1.c1)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                           Spawn on: All datanodes
                           ->  Seq Scan on t2
(11 rows)
```

- **目前DWS支持的Sublink-Release场景**
  - IN-Sublink无相关条件
    - 不能包含上一层查询的表中的列（可以包含更高层查询表中的列）。
    - 不能包含易变函数。

 版权所有 © 华为云计算技术有限公司

```
select t1.c1,t1.c2
from t1
where t1.c1 in (
    select c2
    from t2
    where t2.c1 = 1
);
```

```
QUERY PLAN
-------------------------------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Nested Loop Semi Join
        Join Filter: (t1.c1 = t2.c2)
        ->  Seq Scan on t1
        ->  Materialize
              ->  Streaming(type: REDISTRIBUTE)
                    Spawn on: datanode1
                    ->  Seq Scan on t2
                          Filter: (c1 = 1)
(10 rows)
```

- Exists-Sublink包含相关条件

  Where子句中必须包含上一层查询的表中的列，子查询的其他部分不能含有上层查询的表中的列。其他限制如下：

  - 子查询必须有from子句。
  - 子查询不能含有with子句。
  - 子查询不能含有聚集函数。
  - 子查询里不能包含集合操作、排序、limit、windowagg、having操作。
  - 不能包含易变函数。

```
select t1.c1,t1.c2
from t1
where exists (
    select c2
    from t2
    where t2.c1 = t1.c1
);
```

```
QUERY PLAN
-----------------------------------
Streaming (type: GATHER)
  Node/s: All datanodes
  ->  Hash Semi Join
        Hash Cond: (t1.c1 = t2.c1)
        ->  Seq Scan on t1
        ->  Hash
              ->  Seq Scan on t2
(7 rows)
```

- 包含聚集函数的等值相关子查询的提升

  子查询的where条件中必须含有来自上一层的列，而且此列必须和子查询本层涉及表中的列做相等判断，且这些条件必须用and连接。其他地方不能包含上层的列。其他限制条件如下：

  - 子查询中where条件包含的表达式（列名）必须是表中的列。
  - 子查询的Select关键字后，必须有且仅有一个输出列，此输出列必须是聚集函数（如max），并且聚集函数的参数（t2.c2）不能是来自外层表（t1）中的列。聚集函数不能是count。

    下列示例可以提升：

    ```
    select * from t1 where c1 >(
        select max(t2.c1) from t2 where t2.c1=t1.c1
    );
    ```

    下列示例不能提升，因为子查询没有聚集函数：

    ```
    select * from t1 where c1 >(
        select  t2.c1 from t2 where t2.c1=t1.c1
    );
    ```

    下列示例不能提升，因为子查询有两个输出列：

    ```
    select * from t1 where (c1,c2) >(
        select  max(t2.c1),min(t2.c2) from t2 where t2.c1=t1.c1
    );
    ```

  - 子查询必须是from子句。

 版权所有 © 华为云计算技术有限公司

- 子查询中不能有groupby、having、集合操作。
- 子查询只能是inner join。

  下列示例不能提升：

```
select * from t1 where c1 >(
      select max(t2.c1) from t2 full join t3 on (t2.c2=t3.c2) where t2.c1=t1.c1
);
```

- 子查询的targetlist中不能包含返回set的函数。
- 子查询的where条件中必须含有来自上一层的列，而且此列必须和子查询层涉及表中的列做相等判断，且这些条件必须用and连接。其他地方不能包含上层中的列。下列示例中的最内层子链接可以提升：

```
select * from t3 where t3.c1=(
       select t1.c1
       from t1 where c1 >(
             select max(t2.c1) from t2 where t2.c1=t1.c1
));
```

  基于上面的示例，再加一个条件，则不能提升，因为最内层子查询引用了上层中的列。示例如下：

```
select * from t3 where t3.c1=(
       select t1.c1
       from t1 where c1 >(
            select max(t2.c1) from t2 where t2.c1=t1.c1 and t3.c1>t2.c2

));
```

– 提升OR子句中的SubLink

  当WHERE过滤条件中有OR连接的EXISTS相关SubLink，

  例如：

```
select a, c from t1
where t1.a = (select avg(a) from t3 where t1.b = t3.b) or
exists (select * from t4 where t1.c = t4.c);
```

  将OR-ed连接的EXISTS相关子查询OR子句的提升过程：

  i. 提取where条件中，or子句中的opExpr。为：t1.a = (select avg(a) from t3 where t1.b = t3.b)

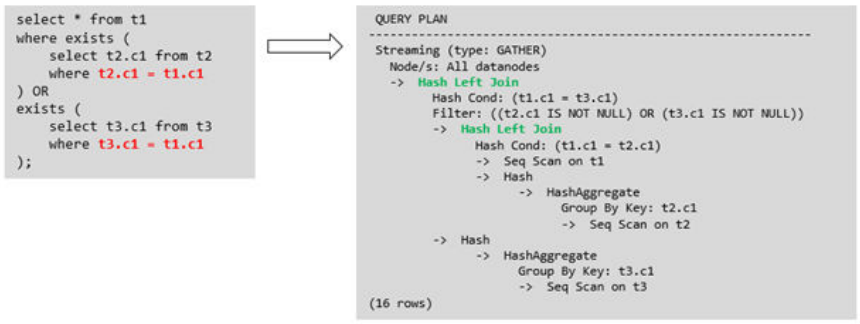
  ii. 这个op操作中包含subquery，判断是否可以提升，如果可以提升，重写subquery为：select avg(a), t3.b from t3 group by t3.b，生成not null条件t3.b is not null，并将这个opexpr用这个not null条件替换。此时SQL变为：

```
select a, c
from t1 left join (select avg(a) avg, t3.b from t3 group by t3.b)  as t3 on (t1.a = avg
and t1.b = t3.b)
where t3.b is not null or exists (select * from t4 where t1.c = t4.c);
```

  iii. 再次提取or子句中的exists sublink，exists (select * from t4 where t1.c = t4.c)，判断是否可以提升，如果可以提升，转换subquery为：select t4.c from t4 group by t4.c生成NotNull条件t4.c is not null提升查询，SQL变为：

```
select a, c
from t1 left join (select avg(a) avg, t3.b from t3 group by t3.b)  as t3 on (t1.a = avg and
t1.b = t3.b)
left join (select t4.c from t4 group by t4.c) where t3.b is not null or t4.c is not null;
```


版权所有 © 华为云计算技术有限公司



- **目前DWS不支持的Sublink-Release场景**

  除了以上场景之外都不支持Sublink提升，因此关联子查询会被计划成SubPlan +Broadcast的执行计划，当inner表的数据量较大时则会产生性能风险。

  如果相关子查询中跟外层的两张表做join，那么无法提升该子查询，需要通过将父SQL创建成with子句，然后再跟子查询中的表做相关子查询。

  例如：

  ```
  select distinct t1.a, t2.a
  from t1 left join t2 on t1.a=t2.a and not exists (select a,b from test1 where test1.a=t1.a and test1.b=t2.a);
  ```

  改写为

  ```
  with temp as
  (
        select * from (select t1.a as a, t2.a as b from t1 left join t2 on t1.a=t2.a)

  )
  select distinct a,b
  from temp
  where not exists (select a,b from test1 where temp.a=test1.a and temp.b=test1.b);
  ```

  – 出现在targetlist里的相关子查询无法提升(不含count)

    例如：

    ```
    explain (costs off)
    select (select c2 from t2 where t1.c1 = t2.c1) ssq, t1.c2
    from t1
    where t1.c2 > 10;
    ```

    执行计划为：

    ```
    explain (costs off)
    select (select c2 from t2 where t1.c1 = t2.c1) ssq, t1.c2
    from t1
    where t1.c2 > 10;
                        QUERY PLAN
    ------------------------------------------------------------
     Streaming (type: GATHER)
       Node/s: All datanodes
       ->  Seq Scan on t1
             Filter: (c2 > 10)
             SubPlan 1
               ->  Result
                     Filter: (t1.c1 = t2.c1)
                     ->  Materialize
                           ->  Streaming(type: BROADCAST)
                                 Spawn on: All datanodes
                                 ->  Seq Scan on t2
    (11 rows)
    ```

    由于相关子查询出现在targetlist（查询返回列表）里，对于t1.c1=t2.c1不匹配的场景仍然需要输出值，因此使用left-outerjoin关联T1&T2确保t1.c1=t2.c1在不匹配时，子SSQ能够返回不匹配的补空值。

 版权所有 © 华为云计算技术有限公司

说明

SSQ和CSSQ的解释如下：

- SSQ：ScalarSubQuery一般指返回1行1列scalar值的sublink，简称SSQ。
- CSSQ：Correlated-ScalarSubQuery和SSQ相同不过是指包含相关条件的SSQ。

上述SQL语句可以改写为：

```
with ssq as
(
   select t2.c1, t2.c2 from t2
)
select ssq.c2, t1.c2
from t1 left join ssq on t1.c1 = ssq.c1
where t1.c2 > 10;
```

改写后的执行计划为：

```
                QUERY PLAN
-------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Hash Right Join
         Hash Cond: (t2.c1 = t1.c1)
         ->  Seq Scan on t2
         ->  Hash
               ->  Seq Scan on t1
                     Filter: (c2 > 10)
(8 rows)
```

可以看到出现在SSQ返回列表里的相关子查询SSQ，已经被提升成Right Join，从而避免当内表T2较大时出现SubPlan+Broadcast计划导致性能变差。

- 出现在targetlist里的相关子查询无法提升(带count)

  例如：

```
select (select count(*) from t2 where t2.c1=t1.c1) cnt, t1.c1, t3.c1
from t1,t3
where t1.c1=t3.c1 order by cnt, t1.c1;
```

  执行计划为：

```
                          QUERY PLAN
-------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Sort
         Sort Key: ((SubPlan 1)), t1.c1
         ->  Hash Join
               Hash Cond: (t1.c1 = t3.c1)
               ->  Seq Scan on t1
               ->  Hash
                     ->  Seq Scan on t3
               SubPlan 1
                 ->  Aggregate
                       ->  Result
                             Filter: (t2.c1 = t1.c1)
                             ->  Materialize
                                   ->  Streaming(type: BROADCAST)
                                         Spawn on: All datanodes
                                         ->  Seq Scan on t2
(17 rows)
```

  相关子查询出现在targetlist（查询返回列表）中，对于t1.c1=t2.c1不匹配的场景仍需要输出值，因此使用LEFT OUTER JOIN关联t1和t2，确保t1.c1=t2.c1不匹配时子SSQ能够返回不匹配的补空值。但由于这里使用了COUNT语句，在t1.c1=t2.c1不匹配时需要输出0，因此可将count(*)改写为case when null then 0 else count(*) end来处理。

  上述SQL语句可以改写为：

 版权所有 © 华为云计算技术有限公司

```
with ssq as
(
   select count(*) cnt, c1 from t2 group by c1
)
select case when
          ssq.cnt is null then 0
          else ssq.cnt
      end cnt, t1.c1, t3.c1
from t1 left join ssq on ssq.c1 = t1.c1,t3
where t1.c1 = t3.c1
order by ssq.cnt, t1.c1;
```

改写后的执行计划为：

```
 QUERY PLAN
-----------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Sort
         Sort Key: (count(*)), t1.c1
         ->  Hash Join
               Hash Cond: (t1.c1 = t3.c1)
               ->  Hash Left Join
                     Hash Cond: (t1.c1 = t2.c1)
                     ->  Seq Scan on t1
                     ->  Hash
                           ->  HashAggregate
                                 Group By Key: t2.c1
                                 ->  Seq Scan on t2
               ->  Hash
                     ->  Seq Scan on t3
(15 rows)
```

- 相关条件为不等值场景

  例如：

  ```
  select t1.c1, t1.c2
  from t1
  where t1.c1 = (select agg() from t2.c2 > t1.c2);
  ```

  对于非等值相关条件的SubLink目前无法提升，从语义上可以通过做2次join（一次CorrelationKey，一次rownum自关联）达到提升改写的目的。

  改写方案有两种：

  - 子查询改写方式

    ```
    select t1.c1, t1.c2
    from t1, (
       select t1.rowid, agg() aggref
       from t1,t2
       where t1.c2 > t2.c2 group by t1.rowid
    ) dt /* derived table */
    where t1.rowid = dt.rowid AND t1.c1 = dt.aggref;
    ```

  - CTE改写方式

    ```
    WITH dt as
    (
       select t1.rowid, agg() aggref
       from t1,t2
       where t1.c2 > t2.c2 group by t1.rowid
    )
    select t1.c1, t1.c2
    from t1, dt
    where t1.rowid = dt.rowid AND
    t1.c1 = dt.aggref;
    ```


版权所有 © 华为云计算技术有限公司

须知

- 目前DWS尚无高效的实现表、中间结果集的全局唯一rowid因此目前此类场景很难改写，建议通过业务层进行规避，或者可以使用t1.xc_node_id + t1.ctid进行rowid关联，但是xc_node_id的重复率较高会导致join关联效率变低，而xc_node_id+ctid类型无法作为hashjoin的关联条件。
- 对于AGG类型为count(*)时需要进行CASE-WHEN对没有match的场景补0处理，非COUNT(*)场景NULL处理。
- CTE改写方式如果有sharescan支持性能上能够更优。

更多优化示例

示例1：修改基表为REPLICATION表，并且在过滤列上创建索引。

```
create table master_table (a int);
create table sub_table(a int, b int);
select a from master_table group by a having a in (select a from sub_table);
```

上述示例中存在一个相关性子查询，为了提升查询的性能，可以将sub_table修改为一个REPLICATION表，并且在字段a上创建一个index。

示例2：修改select语句，将子查询修改为和主表的join，或者修改为可以提升的subquery，但是在修改前后需要保证语义的正确性。

```
explain (costs off)select * from master_table as t1 where t1.a in (select t2.a from sub_table as t2 where t1.a
= t2.b);
                      QUERY PLAN
------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on master_table t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t1.a = t2.b)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                             Spawn on: All datanodes
                             ->  Seq Scan on sub_table t2
(11 rows)
```

上面示例计划中存在一个subPlan，为了消除这个subPlan可以修改语句为：

```
explain(costs off) select * from master_table as t1 where exists (select t2.a from sub_table as t2 where t1.a
= t2.b and t1.a = t2.a);
                   QUERY PLAN
-------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Hash Semi Join
         Hash Cond: (t1.a = t2.b)
         ->  Seq Scan on master_table t1
         ->  Hash
               ->  Streaming(type: REDISTRIBUTE)
                     Spawn on: All datanodes
                     ->  Seq Scan on sub_table t2
(9 rows)
```

从计划可以看出，subPlan消除了，计划变成了两个表的semi join，这样会大幅度提高执行效率。

 版权所有 © 华为云计算技术有限公司

14.4.7.4 统计信息调优

统计信息调优介绍

DWS是基于代价估算生成的最优执行计划。优化器需要根据ANALYZE收集的统计信息行数估算和代价估算，因此统计信息对优化器行数估算和代价估算起着至关重要的作用。通过ANALYZE收集全局统计信息，主要包括：pg_class表中的relpages和reltuples；pg_statistic表中的stadistinct、stanullfrac、stanumbersN、stavaluesN、histogram_bounds等。

实例分析 1：未收集统计信息导致查询性能差

在很多场景中，由于查询中涉及到的表或列没有收集统计信息，对查询性能产生很大的影响。

示例表的表结构如下：

```
CREATE TABLE LINEITEM
(
L_ORDERKEY        BIGINT        NOT NULL
, L_PARTKEY       BIGINT        NOT NULL
, L_SUPPKEY       BIGINT        NOT NULL
, L_LINENUMBER    BIGINT        NOT NULL
, L_QUANTITY      DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT      DECIMAL(15,2) NOT NULL
, L_TAX           DECIMAL(15,2) NOT NULL
, L_RETURNFLAG    CHAR(1)       NOT NULL
, L_LINESTATUS    CHAR(1)       NOT NULL
, L_SHIPDATE      DATE          NOT NULL
, L_COMMITDATE    DATE          NOT NULL
, L_RECEIPTDATE   DATE          NOT NULL
, L_SHIPINSTRUCT  CHAR(25)      NOT NULL
, L_SHIPMODE      CHAR(10)      NOT NULL
, L_COMMENT       VARCHAR(44)   NOT NULL
) with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
(
O_ORDERKEY       BIGINT        NOT NULL
, O_CUSTKEY      BIGINT        NOT NULL
, O_ORDERSTATUS  CHAR(1)       NOT NULL
, O_TOTALPRICE   DECIMAL(15,2) NOT NULL
, O_ORDERDATE    DATE NOT NULL
, O_ORDERPRIORITY CHAR(15)     NOT NULL
, O_CLERK        CHAR(15)      NOT NULL
, O_SHIPPRIORITY  BIGINT       NOT NULL
, O_COMMENT      VARCHAR(79)   NOT NULL
)with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
```


版权所有 © 华为云计算技术有限公司

```
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
order by
numwait desc;
```

可以通过如下方法确认查询中涉及的表或列是否执行过ANALYZE收集统计信息。

1. 通过explain verbose执行query分析执行计划时会提示WARNING信息：
   ```
   WARNING:Statistics in some tables or columns(public.lineitem(l_receiptdate,l_commitdate,l_orderkey, l_suppkey), public.orders(o_orderstatus,o_orderkey)) are not collected.
   HINT:Do analyze for them in order to generate optimized plan.
   ```
2. 通过在pg_log目录下的日志文件中查找以下类似信息来确认是当前执行的query是否由于没有收集统计信息导致查询性能变差。
   ```
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] LOG:Statistics in some tables or columns(public.lineitem(l_receiptdate, l_commitdate,l_orderkey,
   .l_suppkey), public.orders(o_orderstatus,o_orderkey)) are not collected.
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] HINT:Do analyze for them in order to generate optimized plan.
   ```

查看和确认涉及的表或列没有执行ANALYZE后，可以通过对WARNING或日志中上报的表或列执行ANALYZE，以解决未收集统计信息导致查询变慢的问题。

实例分析 2：设置 cost_param 对查询性能优化

请参考**案例：设置cost_param对查询性能优化**。

实例分析 3：多表 join 的复杂查询存在中间结果不准调优

现象描述：查询与指定人在前后15分钟内、同一网吧登记上网的人员信息：

```
SELECT
C.WBM,
C.DZQH,
C.DZ,
B.ZJHM,
B.SWKSSJ,
B.XWSJ
FROM
b_zyk_wbswxx A,
b_zyk_wbswxx B,
b_zyk_wbcs C
WHERE
A.ZJHM = '522522******3824'
AND A.WBDM = B.WBDM
AND A.WBDM = C.WBDM
AND abs(to_date(A.SWKSSJ,'yyyymmddHH24MISS') - to_date(B.SWKSSJ,'yyyymmddHH24MISS')) <
INTERVAL '15 MINUTES'
ORDER BY
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

执行计划如**图14-6**所示。该查询实际耗时约12秒。

 版权所有 © 华为云计算技术有限公司

图 14-6 应用 unlogged table 案例（一）

```
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=221021.41..221021.43 rows=10 width=120)
   ->  Sort  (cost=221021.41..221022.01 rows=240 width=120)
         Sort Key: b.swkssj, b.zjhm
         ->  Streaming (type: GATHER)  (cost=221015.62..221016.22 rows=240 width=120)
               Node/s: All datanodes
               ->  Limit  (cost=9208.98..9209.01 rows=10 width=120)
                     ->  Sort  (cost=9208.98..9211.60 rows=1048 width=120)
                           Sort Key: b.swkssj, b.zjhm
                           ->  Nested Loop  (cost=23.27..9186.34 rows=1048 width=120)
                                 Join Filter: (((a.zjhm)::text <> (b.zjhm)::text) AND ((a.wbdm)::text = (b.wbdm)::text)
                                 AND (abs(((to_date((a.swkssj)::text, 'yyyymmddHH24MISS'::text)
                                    - to_date((b.swkssj)::text, 'yyyymmddHH24MISS'::text)))::numeric) < .0104166666666667))
                                 ->  Streaming(type: BROADCAST)  (cost=0.00..6.33 rows=24 width=135)
                                       Spawn on: All datanodes
                                       ->  Nested Loop  (cost=0.00..106.80 rows=1 width=135)
                                             ->  Streaming(type: BROADCAST)  (cost=0.00..24.75 rows=264 width=48)
                                                   Spawn on: All datanodes
                                                   ->  Partition Iterator  (cost=0.00..48.44 rows=11 width=48)
                                                         Iterations: 25
                                                         ->  Partitioned Index Scan using idx_b_zyk_wbswxx_zjhm on b_zyk_wbswxx a  (cost=0.00..48.44 rows=11 width=48)
                                                               Index Cond: ((zjhm)::text = '522522******3824'::text)
                                                               Selected Partitions:  1..25
                                             ->  Index Scan using idx_b_zyk_wbcs_wbdm on b_zyk_wbcs c  (cost=0.00..2.82 rows=1 width=87)
                                                   Index Cond: ((wbdm)::text = (a.wbdm)::text)
                                 ->  Partition Iterator  (cost=23.27..7306.33 rows=2454 width=63)
                                       Iterations: 25
                                       ->  Partitioned Bitmap Heap Scan on b_zyk_wbswxx b  (cost=23.27..7306.33 rows=2454 width=63)
                                             Recheck Cond: ((wbdm)::text = (c.wbdm)::text)
                                             Filter: ('522522198405243824'::text <> (zjhm)::text)
                                             Selected Partitions:  1..25
                                             ->  Partitioned Bitmap Index Scan on idx_b_zyk_wbswxx_wbdm  (cost=0.00..22.65 rows=2454 width=0)
                                                   Index Cond: ((wbdm)::text = (c.wbdm)::text)
(30 rows)
```

优化分析：分析过程如下：

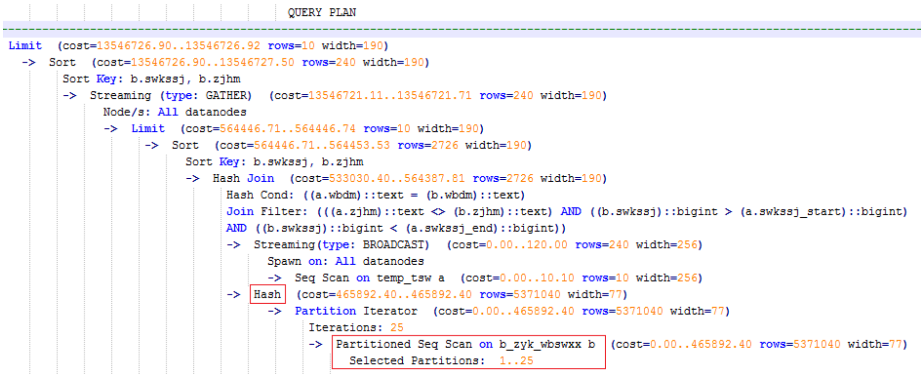
1. 分析该执行计划发现，扫描节点已使用Index Scan，耗时主要在最外层Nest Loop Join的Join Filter计算中，且该计算执行了字符串的加减法和不等值比较。
2. 考虑使用unlogged table保存目标人的上网信息，且在插入时处理上网开始时间和终止时间，以避免后续进行时间加减。

```
//创建临时unlogged table
CREATE UNLOGGED TABLE temp_tsw
(
ZJHM         NVARCHAR2(18),
WBDM          NVARCHAR2(14),
SWKSSJ_START NVARCHAR2(14),
SWKSSJ_END   NVARCHAR2(14),
WBM          NVARCHAR2(70),
DZQH         NVARCHAR2(6),
DZ          NVARCHAR2(70),
IPDZ         NVARCHAR2(39)
)
;
//插入目标人的上网记录，并处理上网开始和结束时间。
INSERT INTO
temp_tsw
SELECT
A.ZJHM,
A.WBDM,
to_char((to_date(A.SWKSSJ,'yyyymmddHH24MISS') - INTERVAL '15
MINUTES'),'yyyymmddHH24MISS'),
to_char((to_date(A.SWKSSJ,'yyyymmddHH24MISS') + INTERVAL '15
MINUTES'),'yyyymmddHH24MISS'),
B.WBM,B.DZQH,B.DZ,B.IPDZ
FROM
b_zyk_wbswxx A,
b_zyk_wbcs B
WHERE
A.ZJHM='522522******3824' AND A.WBDM = B.WBDM
;

//查询和目标人在前后十五分钟内在同一网吧上网的人员信息，比较大小时强制转换为int8。
SELECT
A.WBM,
A.DZQH,
A.DZ,
A.IPDZ,
```

```
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
order by
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

上述查询耗时约7秒，执行计划如图14-7所示。

图 14-7 应用 unlogged table 案例（二）



```
QUERY PLAN
--------------------------------------------------------------------------------
Limit  (cost=13546726.90..13546726.92 rows=10 width=190)
  ->  Sort  (cost=13546726.90..13546727.50 rows=240 width=190)
        Sort Key: b.swkssj, b.zjhm
        ->  Streaming (type: GATHER)  (cost=13546721.11..13546721.71 rows=240 width=190)
              Node/s: All datanodes
              ->  Limit  (cost=564446.71..564446.74 rows=10 width=190)
                    ->  Sort  (cost=564446.71..564453.53 rows=2726 width=190)
                          Sort Key: b.swkssj, b.zjhm
                          ->  Hash Join  (cost=533030.40..564387.81 rows=2726 width=190)
                                Hash Cond: ((a.wbdm)::text = (b.wbdm)::text)
                                Join Filter: (((a.zjhm)::text <> (b.zjhm)::text) AND ((b.swkssj)::bigint > (a.swkssj_start)::bigint)
                                AND ((b.swkssj)::bigint < (a.swkssj_end)::bigint))
                                ->  Streaming(type: BROADCAST)  (cost=0.00..120.00 rows=240 width=256)
                                      Spawn on: All datanodes
                                      ->  Seq Scan on temp_tsw a  (cost=0.00..10.10 rows=10 width=256)
                                ->  Hash  (cost=465892.40..465892.40 rows=5371040 width=77)
                                      ->  Partition Iterator  (cost=0.00..465892.40 rows=5371040 width=77)
                                            Iterations: 25
                                            ->  Partitioned Seq Scan on b_zyk_wbswxx b  (cost=0.00..465892.40 rows=5371040 width=77)
                                                  Selected Partitions:  1..25
```

3. 分析上述执行计划，发现执行了Hash Join，对大表b_zyk_wbswxx建立了Hash Table。由于该表数据量大，创建过程耗时较长。

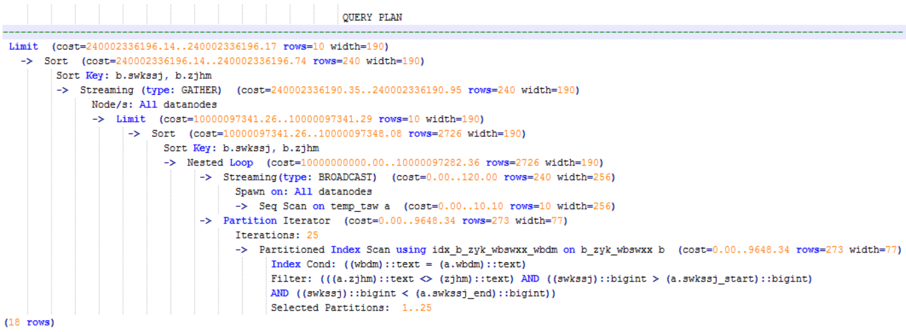
   由于temp_tsw中仅包含几百条记录，且temp_tsw和b_zyk_wbswxx均通过wbdm（网吧代码）执行等值连接。因此，如果Join方式改为Nest Loop Join，则扫描节点可以实现Index Scan，性能预计将会提升。

4. 执行如下语句，将Join方式改为Nest Loop Join。

   ```
   SET enable_hashjoin = off;
   ```

   执行计划如图14-8所示。查询耗时约3秒。

图 14-8 应用 unlogged table 案例（三）



```
QUERY PLAN
--------------------------------------------------------------------------------
Limit  (cost=240002336196.14..240002336196.17 rows=10 width=190)
  ->  Sort  (cost=240002336196.14..240002336196.74 rows=240 width=190)
        Sort Key: b.swkssj, b.zjhm
        ->  Streaming (type: GATHER)  (cost=240002336190.35..240002336190.95 rows=240 width=190)
              Node/s: All datanodes
              ->  Limit  (cost=10000097341.26..10000097341.29 rows=10 width=190)
                    ->  Sort  (cost=10000097341.26..10000097348.08 rows=2726 width=190)
                          Sort Key: b.swkssj, b.zjhm
                          ->  Nested Loop  (cost=10000000000.00..10000097282.36 rows=2726 width=190)
                                ->  Streaming(type: BROADCAST)  (cost=0.00..120.00 rows=240 width=256)
                                      Spawn on: All datanodes
                                      ->  Seq Scan on temp_tsw a  (cost=0.00..10.10 rows=10 width=256)
                                ->  Partition Iterator  (cost=0.00..9648.34 rows=273 width=77)
                                      Iterations: 25
                                      ->  Partitioned Index Scan using idx_b_zyk_wbswxx_wbdm on b_zyk_wbswxx b  (cost=0.00..9648.34 rows=273 width=77)
                                            Index Cond: ((wbdm)::text = (a.wbdm)::text)
                                            Filter: (((a.zjhm)::text <> (zjhm)::text) AND ((swkssj)::bigint > (a.swkssj_start)::bigint)
                                            AND ((swkssj)::bigint < (a.swkssj_end)::bigint))
                                            Selected Partitions:  1..25
(18 rows)
```

5. 使用unlogged table保存结果集并用于分页显示。

   如果需要在上层应用页面实现分页显示，需要修改offset值确定显示目标页的结果集。按此实现，每次翻页时均执行上面查询语句，耗时较长。

 版权所有 © 华为云计算技术有限公司

为解决上述问题，建议使用unlogged table保存结果集。

```
//创建保存结果集的unlogged table
CREATE UNLOGGED TABLE temp_result
(
WBM      NVARCHAR2(70),
DZQH     NVARCHAR2(6),
DZ       NVARCHAR2(70),
IPDZ     NVARCHAR2(39),
ZJHM     NVARCHAR2(18),
XM       NVARCHAR2(30),
SWKSSJ   date,
XWSJ     date,
SWZDH    NVARCHAR2(32)
);

//将结果集插入unlogged table，插入耗时约3秒。
INSERT INTO
temp_result
SELECT
A.WBM,
A.DZQH,
A.DZ,
A.IPDZ,
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
;

//查询结果集表进行分页显示，分页查询耗时约10ms。
SELECT
*
FROM
temp_result
ORDER BY
SWKSSJ,
ZJHM
LIMIT 10 OFFSET 0;
```

⚠ 注意

通过ANALYZE收集全局统计信息，通常会改善查询性能。

如果遇到性能问题：可以使用plan hint来调整到之前的查询计划，详情请参见**使用Plan Hint进行调优**。

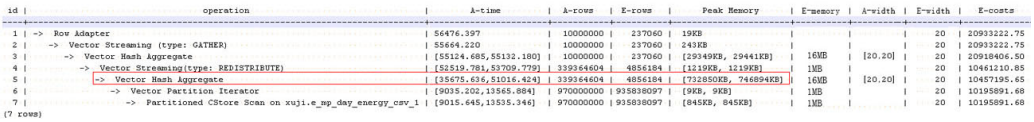
14.4.7.5 算子级调优

算子级调优介绍

一个查询语句要经过多个算子步骤才会输出最终的结果。由于个别算子耗时过长导致整体查询性能下降的情况比较常见。这些算子是整个查询的瓶颈算子。通用的优化手段是EXPLAIN ANALYZE/PERFORMANCE命令查看执行过程的瓶颈算子，然后进行针对性优化。

版权所有 © 华为云计算技术有限公司

如下面的执行过程信息中，Hashagg算子的执行时间占总时间的：(51016-13535)/56476 ≈66%，此处Hashagg算子就是这个查询的瓶颈算子，在进行性能优化时应当优先考虑此算子的优化。



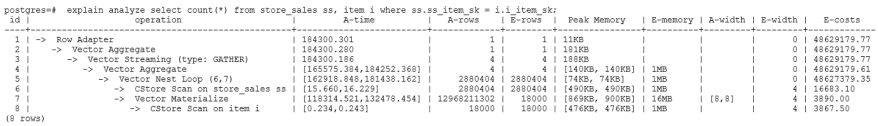
算子级调优示例

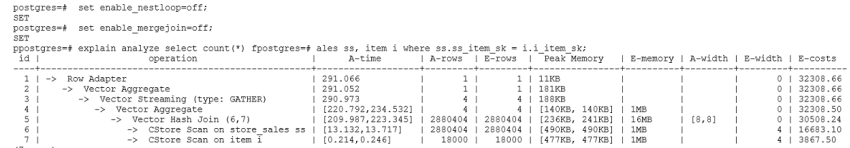
示例1：基表扫描时，对于点查或者范围扫描等过滤大量数据的查询，如果使用SeqScan全表扫描会比较耗时，可以在条件列上建立索引选择IndexScan进行索引扫描提升扫描效率。

```
explain (analyze on, costs off) select * from store_sales where ss_sold_date_sk = 2450944;
 id |             operation             |       A-time        | A-rows | Peak Memory  | A-width
----+-----------------------------------+---------------------+--------+--------------+---------
  1 | ->  Streaming (type: GATHER)   | 3666.020            |   3360 | 195KB        |
  2 |    ->  Seq Scan on store_sales | [3594.611,3594.611] |   3360 | [34KB, 34KB] |

 Predicate Information (identified by plan id)
-------------------------------------------------
   2 --Seq Scan on store_sales
         Filter: (ss_sold_date_sk = 2450944)
         Rows Removed by Filter: 4968936
 create index idx on store_sales_row(ss_sold_date_sk);
CREATE INDEX
 explain (analyze on, costs off) select * from store_sales_row where ss_sold_date_sk = 2450944;
 id |                    operation                    |     A-time      | A-rows | Peak Memory  | A-width
----+-------------------------------------------------+-----------------+--------+--------------+----------
  1 | ->  Streaming (type: GATHER)                    | 81.524          |   3360 | 195KB        |
  2 |    ->  Index Scan using idx on store_sales_row | [13.352,13.352] |   3360 | [34KB, 34KB] |
```

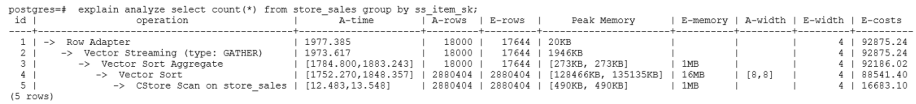
上述例子中，全表扫描返回3360条数据，过滤掉大量数据，在ss_sold_date_sk列上建立索引后，使用IndexScan扫描效率显著提高，从3.6秒提升到13毫秒。

示例2：如果从执行计划中看，两表join选择了NestLoop，而实际行数比较大时，NestLoop Join可能执行比较慢。如下的例子中NestLoop耗时181秒，如果设置参数enable_mergejoin=off关掉Merge Join，同时设置参数enable_nestloop=off关掉NestLoop，让优化器选择HashJoin，则Join耗时提升至200多毫秒。





示例3：通常情况下Agg选择HashAgg性能较好，如果大结果集选择了Sort+GroupAgg，则需要设置enable_sort=off，HashAgg耗时明显优于Sort+GroupAgg。



版权所有 © 华为云计算技术有限公司

```
postgres=#  set enable_sort=off;
SET
postgres=#  explain analyze select count(*) from store_sales group by ss_item_sk;
 id |                operation                 |       A-time        | A-rows  | E-rows  |      Peak Memory       | E-memory | A-width | E-width | E-costs
----+------------------------------------------+---------------------+---------+---------+------------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                          | 838.218             |   18000 |   17644 | 20KB                   |          |         |       4 | 21016.93
  2 |    ->  Vector Streaming (type: GATHER)   | 834.264             |   18000 |   17644 | 228KB                  |          |         |       4 | 21016.93
  3 |      ->  Vector Hash Aggregate           | [585.017,758.204]   |   18000 |   17644 | [262552KB, 262564KB]   | 16MB     | [8,8]   |       4 | 20327.72
  4 |        ->  CStore Scan on store_sales    | [12.540,13.941]     | 2880404 | 2880404 | [490KB, 490KB]         | 1MB      |         |       4 | 16683.10
(4 rows)
```

14.4.7.6 数据倾斜调优

数据倾斜问题是分布式架构的重要难题，它破坏了MPP架构中各个节点对等的要求，导致单节点（倾斜节点）所存储或者计算的数据量远大于其他节点，所以会造成以下危害：

- 存储上的倾斜会严重限制系统容量，在系统容量不饱和的情况下，由于单节点倾斜的限制，使得整个系统容量无法继续增长。
- 计算上的倾斜会严重影响系统性能，由于倾斜节点所需要运算的数据量远大于其他节点，导致倾斜节点降低系统整体性能。
- 数据倾斜还严重影响了MPP架构的扩展性。由于在存储或者计算时，会将相同值的数据放到同一节点，因此当倾斜数据（大量数据的值相同）出现之后，即使增加节点，系统瓶颈仍然受限于倾斜节点的容量或者性能。

DWS数据库针对数据倾斜问题给出了完整的解决方案，包括存储倾斜和计算倾斜两大问题，下面分别进行介绍。

存储层数据倾斜

DWS数据库中，数据分布存储在各个DN上，通过分布式执行提高查询的效率。但是，如果数据分布存在倾斜，则会导致分布式执行某些DN成为瓶颈，影响查询性能。这种情况通常是由于分布列选择不合理，可以通过调整分布列的方式解决。

分析过程：

步骤1 登录**DWS控制台**。在“集群列表”页面，找到需要查看监控的集群。在指定集群所在行的“操作”列，单击“监控面板”。选择“监控 > 节点监控 > 磁盘”，查看磁盘使用率。

说明

各个数据磁盘的利用率，会有不均衡的现象。正常情况下，利用率最高和利用率最低的磁盘空间相差不大，如果磁盘利用率相差超过了5%就要注意是不是有资源倾斜的情况。

步骤2 连接数据库，通过等待视图查看作业的运行情况，发现作业总是等待部分DN或者个别DN。

```
SELECT wait_status, count(*) as cnt FROM pgxc_thread_wait_status WHERE wait_status not like '%cmd%'
AND wait_status not like '%none%' and wait_status not like '%quit%' group by 1 order by 2 desc;
```

步骤3 执行慢语句的explain performance显示，发现各个DN的基表scan的时间和行数不均衡。

```
explain performance select avg(ss_wholesale_cost) from store_sales;
```

```
rmance select avg(ss_wholesale_cost) from store_sales;

      operation                                    |       A-time          |  A-rows  | E-rows  | E-distinct |  Peak Memory     | E-memory | A-wi
---------------------------------------------------+-----------------------+----------+---------+------------+------------------+----------+-----
                                                   | 2452.321              |        1 |       1 |            | 11KB             |          |
te                                                 | 2452.238              |        1 |       1 |            | 184KB            |          |
aming (type: GATHER)                               | 2452.010              |        4 |       4 |            | 108KB            |          |
ggregate                                           | [12.219,2425.225]     |        4 |       4 |            | [183KB, 184KB]   | 1MB      |
r Partition Iterator                               | [12.139,1189.187]     | 22831616 | 2880404 |            | [17KB, 17KB]     | 1MB      |
rtitioned CStore Scan on public.store_sales        | [5.929,1173.555]      | 22831616 | 2880404 |            | [482KB, 499KB]   | 1MB      |

identified by plan id)
--------------------------
rator

can on public.store_sales
s:  1..7
```

 版权所有 © 华为云计算技术有限公司

- 基表scan的时间：最快的DN耗时5ms，最慢的DN耗时1173ms。
- 数据分布情况：某些DN有22831616行，其他DN都是0行，数据有严重倾斜。

```
5 --Vector Partition Iterator
      datanode1 (actual time=0.620..1189.187 rows=22831616 loops=1)
      datanode2 (actual time=14.346..14.346 rows=0 loops=1)
      datanode3 (actual time=14.424..14.424 rows=0 loops=1)
      datanode4 (actual time=12.139..12.139 rows=0 loops=1)
      datanode1 (CPU: ex c/r=1, ex row=22831616, ex cyc=40540820, inc cyc=3088825848)
      datanode2 (CPU: ex c/r=0, ex row=0, ex cyc=20852952, inc cyc=37297912)
      datanode3 (CPU: ex c/r=0, ex row=0, ex cyc=20005612, inc cyc=37501436)
      datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=16147884, inc cyc=31560524)
6 --Partitioned CStore Scan on public.store_sales
      datanode1 (actual time=2.611..1173.555 rows=22831616 loops=7)
      datanode2 (actual time=6.327..6.327 rows=0 loops=7)
      datanode3 (actual time=6.732..6.732 rows=0 loops=7)
      datanode4 (actual time=5.929..5.929 rows=0 loops=7)
      datanode1 (Buffers: shared hit=1359)
      datanode2 (Buffers: shared hit=55)
      datanode3 (Buffers: shared hit=55)
      datanode4 (Buffers: shared hit=55)
      datanode1 (CPU: ex c/r=133, ex row=22831616, ex cyc=3048285028, inc cyc=3048285028)
      datanode2 (CPU: ex c/r=0, ex row=0, ex cyc=16444960, inc cyc=16444960)
      datanode3 (CPU: ex c/r=0, ex row=0, ex cyc=17495824, inc cyc=17495824)
      datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=15412640, inc cyc=15412640)
```

步骤4 通过倾斜检查接口可以发现数据倾斜。

```
SELECT table_skewness('store_sales');
```

```
tpcds1xcpm=# select table_skewness('store_sales');
                table_skewness
-------------------------------------------
 ("datanode1            ",22831616,100.000%)
 ("datanode2            ",0,0.000%)
 ("datanode3            ",0,0.000%)
 ("datanode4            ",0,0.000%)
(4 rows)
```

```
SELECT table_distribution('public','store_sales');
```

```
tpcds1xcpm=# select table_distribution('public','store_sales');
           table_distribution
-----------------------------------------
 (public,store_sales,datanode1,1011752960)
 (public,store_sales,datanode4,33792000)
 (public,store_sales,datanode2,32129024)
 (public,store_sales,datanode3,33349632)
(4 rows)
```

----结束

处理方法--如何找到倾斜的表：

1. 数据库中表个数少于1万的场景下，可直接使用倾斜视图查询当前库内所有表的数据倾斜情况。
   ```
   SELECT * FROM pgxc_get_table_skewness ORDER BY totalsize DESC;
   ```
2. 数据库中表个数非常多（至少大于1万）的场景，因PGXC_GET_TABLE_SKEWNESS视图涉及全库查并计算非常全面的倾斜字段，所以可能会花费比较长的时间（小时级），建议参考PGXC_GET_TABLE_SKEWNESS视图定义，执行以下操作：
   - 8.1.2及之前集群版本中使用table_distribution()函数自定义输出，减少输出列进行计算优化，例如：

 版权所有 © 华为云计算技术有限公司

```
SELECT schemaname,tablename,max(dnsize) AS maxsize, min(dnsize) AS minsize
FROM pg_catalog.pg_class c
INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
INNER JOIN pg_catalog.table_distribution() s ON s.schemaname = n.nspname AND s.tablename
= c.relname
INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype = 'H'
GROUP BY schemaname,tablename;
```

- 8.1.3及以上集群版本中支持使用**gs_table_distribution()**函数，全库查询所有表的数据倾斜情况。全库表查询时，gs_table_distribution()函数优于table_distribution()函数；在大集群大数据量场景下，如果进行全库表查询，建议优先使用gs_table_distribution()函数。

```
SELECT schemaname,tablename,max(dnsize) AS maxsize, min(dnsize) AS minsize
FROM pg_catalog.pg_class c
INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
INNER JOIN pg_catalog.gs_table_distribution() s ON s.schemaname = n.nspname AND
s.tablename = c.relname
INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype = 'H'
GROUP BY schemaname,tablename;
```

说明

使用如下语句可快速查询到大表：

```
SELECT schemaname||'.'||tablename as table, sum(dnsize) as size FROM
gs_table_distribution() group by 1 order by 2 desc limit 10;
```

使用如下语句可快速查询表的倾斜率：

```
WITH skew AS
(
        SELECT
                schemaname,
                tablename,
                pg_catalog.sum(dnsize) AS totalsize,
                pg_catalog.avg(dnsize) AS avgsize,
                pg_catalog.max(dnsize) AS maxsize,
                pg_catalog.min(dnsize) AS minsize,
                (pg_catalog.max(dnsize) - pg_catalog.min(dnsize)) AS skewsize,
                pg_catalog.stddev(dnsize) AS skewstddev
        FROM pg_catalog.pg_class c
        INNER JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
        INNER JOIN pg_catalog.gs_table_distribution() s ON s.schemaname = n.nspname
AND s.tablename = c.relname
        INNER JOIN pg_catalog.pgxc_class x ON c.oid = x.pcrelid AND x.pclocatortype IN('H',
'N')
        GROUP BY schemaname,tablename
)
SELECT
        schemaname,
        tablename,
        totalsize,
        avgsize::numeric(1000),
        (maxsize/totalsize)::numeric(4,3)  AS maxratio,
        (minsize/totalsize)::numeric(4,3)  AS minratio,
        skewsize,
        (skewsize/avgsize)::numeric(4,3)  AS skewratio,
        skewstddev::numeric(1000)
FROM skew
WHERE totalsize > 0;
```

处理方法--表的分布键的选择方法：

1. 如果此列的distinct值比较大，并且没有明显的数据倾斜，也可以把多列定义成分布列。

   查看distinct的大小：

   ```
   SELECT count(distinct column1) FROM table;
   ```

   查看数据是否存在倾斜：

   ```
   SELECT count(*) cnt, column1 FROM table group by column1 order by cnt limit 100;
   ```


版权所有 © 华为云计算技术有限公司

2. 选用经常做JOIN或group by的列，可以减少STREAM运算。
3. 不推荐以下分布键选择方式：
   a. 分布列用默认值（第一列）。
   b. 分布列用sequence自增生成。
   c. 分布列用随机数生成（除非任意列，或者任意两列的组合做分布键都是倾斜的，一般不选用这种方法）。

计算层数据倾斜

即使通过修改表的分布键，使得数据存储在各个节点上是均衡的，但是在执行查询的过程中，仍然可能出现数据倾斜的问题。在运算过程中某个算子在DN上输出的结果集出现倾斜，从而导致此算子上层的运算出现计算倾斜。一般来说，这是由于在执行过程中，数据重分布导致的。

在查询执行的过程中，join key、group by key等不是表的分布列，因此需要按照join key、group by key上数据的hash值，让数据在各个DN之间进行重新分布，这个过程对应于计划中的Redistribute算子。当重分布列上的数据存在倾斜时，就会导致运行时的数据倾斜，即重分布后部分节点的数据远大于其他。倾斜节点需要处理更多的数据，导致倾斜节点的计算性能远低于其他节点。

如下例中，s表和t表join，join条件中的s.x和t.x均不是表的分布列，因此需要重分布（REDISTRIBUTE算子）。其中s.x列上存在倾斜值，t.x上不存在倾斜。id=6的stream算子在datanode2节点输出的结果集是其他DN的3倍，从而导致了计算倾斜。

```
select * from skew s,test t where s.x = t.x order by s.a limit 1;
 id |                    operation                    |        A-time
----+-------------------------------------------------+-----------------------
  1 | ->  Limit                                       | 52622.382
  2 |    ->  Streaming (type: GATHER)                 | 52622.374
  3 |       ->  Limit                                 | [30138.494,52598.994]
  4 |          ->  Sort                               | [30138.486,52598.986]
  5 |             ->  Hash Join (6,8)                 | [30127.013,41483.275]
  6 |                ->  Streaming(type: REDISTRIBUTE)    | [11365.110,22024.845]
  7 |                   ->  Seq Scan on public.skew s     | [2019.168,2175.369]
  8 |                ->  Hash                         | [2460.108,2499.850]
  9 |                   ->  Streaming(type: REDISTRIBUTE) | [1056.214,1121.887]
 10 |                      ->  Seq Scan on public.test t  | [310.848,325.569]

6 --Streaming(type: REDISTRIBUTE)
        datanode1 (rows=5050368)
        datanode2 (rows=15276032)
        datanode3 (rows=5174272)
        datanode4 (rows=5219328)
```

和存储倾斜相比，计算倾斜更难以提前识别，因此DWS提出了RLBT(Runtime Load Balance Technology)方案，用于解决运行时的计算倾斜问题，该特性由参数**skew_option**控制。RLBT方案主要分为两个层面，第一步是计算倾斜识别，第二步是计算倾斜解决。下面分别进行介绍。

1. 倾斜识别

   计算倾斜的识别，即预先识别计算过程中的重分布列是否存在倾斜数据。RLBT方案中给出了三个解决手段，统计信息识别，hint方式指定以及规则识别：

   - 统计信息识别

     需要用户先执行ANALYZE收集各表的统计信息，然后优化器能够自动利用统计信息对重分布键上的倾斜数据进行提前识别，对于存在倾斜的查询，生成相应的优化计划。在重分布键有多列的情况，只有所有列都属于同一个基表才能利用统计信息进行识别。

 版权所有 © 华为云计算技术有限公司

统计信息只能给出基表的倾斜情况，当基表某一列存在倾斜，其他列上带有过滤条件，或者经过和其他表的join之后，无法准确判断倾斜列上倾斜数据是否依旧存在。当skew_option为normal时，这里认为倾斜数据依旧存在，仍然会对基表中识别到的倾斜进行优化；当skew_option为lazy时，这里认为倾斜数据已经不再存在，也就不会进行相应的优化。

- hint方式指定

  统计信息有着一定的局限性，对于较为复杂的查询，其中间结果难以通过统计信息进行估算和识别倾斜数据。对于这种情况，DWS设计了hint手段，通过用户手动指定的方式，给定倾斜信息。优化器根据用户给定的倾斜信息，来对查询进行优化。详细hint使用语法参见**运行倾斜的hint**。

- 规则识别

  现在BI系统会产生大量带有outer join（left join、right join、full join）的SQL，outer join在匹配失败的情况下会补空产生大量NULL值，如果接下来在补空列上进行join或者group by操作，就会导致NULL值倾斜。当前RLBT技术会自动识别这种场景，并生成相应的NULL值倾斜优化计划。

2. 计算倾斜解决

   在解决倾斜时，目前针对最常见的join和agg算子进行了优化。

   - join优化

   基本思路是将倾斜数据和非倾斜数据进行隔离处理。主要分为以下三种情况：

   a. join两侧都需要做重分布：

      对倾斜侧做PART_REDISTRIBUTE_PART_ROUNDROBIN，其中对倾斜数据做roundrobin，非倾斜数据做redistribute；

      对非倾斜侧做PART_REDISTRIBUTE_PART_BROADCAST，其中对倾斜数据做broadcast，非倾斜数据做redistribute；

   b. join一侧需要重分布，另一侧不需要重分布：

      对需要重分布的一侧做PART_REDISTRIBUTE_PART_ROUNDROBIN；

      对不需要重分布的一侧做PART_LOCAL_PART_BROADCAST，其中对等于倾斜值的部分做broadcast，其余数据保留在本地。

   c. 对于有补NULL值的表：

      对该表做PART_REDISTRIBUTE_PART_LOCAL，其中将NULL值保留在本地，其余数据做redistribute。

   以前面的查询为例，s.x列上存在倾斜数据，倾斜数据的值为0。优化器通过统计信息，识别到了该倾斜数据，生成了倾斜优化计划如下：

```
 id |                                operation                                 |        A-time
----+--------------------------------------------------------------------------+-----------------------
  1 | ->  Limit                                                                | 23642.049
  2 |    ->  Streaming (type: GATHER)                                          | 23642.041
  3 |       ->  Limit                                                          | [23310.768,23618.021]
  4 |          ->  Sort                                                        | [23310.761,23618.012]
  5 |             ->  Hash Join (6,8)                                          | [20898.341,21115.272]
  6 |                ->  Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)    |
[7125.834,7472.111]
  7 |                   ->  Seq Scan on public.skew s                          | [1837.079,1911.025]
  8 |                ->  Hash                                                  | [2612.484,2640.572]
  9 |                   ->  Streaming(type: PART REDISTRIBUTE PART BROADCAST) | [1193.548,1297.894]
 10 |                      ->  Seq Scan on public.test t                       | [314.343,328.707]

  5 --Vector Hash Join (6,8)
      Hash Cond: s.x = t.x
      Skew Join Optimized by Statistic
  6 --Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)
      datanode1  (rows=7635968)
```

 版权所有 © 华为云计算技术有限公司

```
      datanode2 (rows=7517184)
      datanode3 (rows=7748608)
      datanode4 (rows=7818240)
```

上述执行计划中，可以看到Skew Join Optimized by Statistic的字样，代表该计划为倾斜优化计划，其中Statistic关键字代表该倾斜优化来自于统计信息，除此之外还有Hint和Rule，分别代表倾斜优化来自于hint语句和规则。对比前面的计划可以看到，这里对于非倾斜数据和倾斜数据做了分别处理。对于s表中的非倾斜数据，依旧按照原有的方案，根据数据的hash值进行重分布；而对于倾斜数据（即等于0的数据），则通过轮询发送的方式，均衡地发送到所有节点。通过这样的方式，解决了倾斜数据分布不均衡的问题。

同时，为了保证结果的正确性，需要对t表做相应的处理。对于t表中等于0（s.x表中的倾斜值）的数据做广播，对于其他数据，依旧根据数据的hash值进行重分布。

通过这样的方式，就解决了join操作中，数据倾斜的问题。从上面的结果来看，id=6的stream算子各个DN的输出结果已经非常均衡，同时查询端到端性能提升了1倍。

如果执行计划中Stream算子类型显示为HYBRID，则表示对不同的倾斜数据所应用的stream方式不同，例如以下计划：

```
EXPLAIN (nodes OFF, costs OFF) SELECT COUNT(*) FROM skew_scol s, skew_scol1 s1 WHERE s.b =
s1.c;
QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------
id |                                                operation
----
+----------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------
1 | ->  Aggregate
2 |    ->  Streaming (type: GATHER)
3 |       ->  Aggregate
4 |          ->  Hash Join (5,7)
5 |             ->  Streaming(type: HYBRID)
6 |                ->  Seq Scan on skew_scol s
7 |             ->  Hash
8 |                ->  Streaming(type: HYBRID)
9 |                   ->  Seq Scan on skew_scol1 s1

Predicate Information (identified by plan id)
-----------------------------------------------------------------------------------------------------------------------------------------
--------------------
4 --Hash Join (5,7)
Hash Cond: (s.b = s1.c)
Skew Join Optimized by Statistic
5 --Streaming(type: HYBRID)
Skew Filter: (b = 1)
Skew Filter: (b = 0)
8 --Streaming(type: HYBRID)
Skew Filter: (c = 0)
Skew Filter: (c = 1)
```

数据1在skew_scol表上有倾斜，对倾斜数据做roundrobin，非倾斜数据做redistribute。

数据0在skew_scol表上是非倾斜侧，对倾斜数据做broadcast，非倾斜数据做redistribute。

由此可以看到两个stream类型分别是PART REDISTRIBUTE PART ROUNDROBIN和PART REDISTRIBUTE PART BROADCAST，这里标记stream类型为HYBRID类型。

- agg优化

对于agg操作，解决倾斜的思路与join操作不同，这里是通过首先在本DN内按照group by key进行去重操作，然后再进行重分布。因为经过DN内部去重之后，从

版权所有 © 华为云计算技术有限公司

全局来看，每个值的数量都不会超过DN数，因此不会出现严重的数据倾斜问题。以如下query为例：

```
select c1, c2, c3, c4, c5, c6, c7, c8, c9, count(*) from t group by c1, c2, c3, c4, c5, c6, c7, c8, c9 limit 10;
```

原执行结果如下：

```
 id |                operation                 |        A-time         | A-rows
----+------------------------------------------+-----------------------+----------
  1 | ->  Streaming (type: GATHER)             | 130621.783            |      12
  2 |   ->  GroupAggregate                     | [85499.711,130432.341] |      12
  3 |     ->  Sort                             | [85499.509,103145.632] | 36679237
  4 |       ->  Streaming(type: REDISTRIBUTE)  | [25668.897,85499.050]  | 36679237
  5 |         ->  Seq Scan on public.t         | [9835.069,10416.388]   | 36679237

  4 --Streaming(type: REDISTRIBUTE)
      datanode1 (rows=36678837)
      datanode2 (rows=100)
      datanode3 (rows=100)
      datanode4 (rows=200)
```

其中存在大量倾斜数据，导致数据按照group by key进行重分布之后，datanode1的数据量是其他节点的数十万倍。在倾斜优化之后，首先在本DN进行一次group by操作，达到数据去重的效果，然后再进行重分布，可以发现基本没有数据倾斜的问题出现。

```
 id |                operation                 |        A-time
----+------------------------------------------+-----------------------
  1 | ->  Streaming (type: GATHER)             | 10961.337
  2 |   ->  HashAggregate                      | [10953.014,10953.705]
  3 |     ->  HashAggregate                    | [10952.957,10953.632]
  4 |       ->  Streaming(type: REDISTRIBUTE)  | [10952.859,10953.502]
  5 |         ->  HashAggregate                | [10084.280,10947.139]
  6 |           ->  Seq Scan on public.t       | [4757.031,5201.168]

 Predicate Information (identified by plan id)
-----------------------------------------------
  3 --HashAggregate
      Skew Agg Optimized by Statistic

  4 --Streaming(type: REDISTRIBUTE)
      datanode1 (rows=17)
      datanode2 (rows=8)
      datanode3 (rows=8)
      datanode4 (rows=14)
```

适用范围

- join算子
  - 支持nest loop，merge join，hash join等join方式。
  - 当倾斜数据位于join的left侧时，支持inner join，left join，semi join，anti join；当倾斜位于join的right侧时，支持inner join，right join，right semi join，right anti join。
  - 通过统计信息获取的倾斜优化计划，优化器会根据代价判断该计划是否为最优计划。通过hint和规则会强制生成倾斜优化计划。
- agg算子
  - array_agg、string_agg、subplan in agg qual这几种场景不支持优化。
  - 通过统计信息识别到的倾斜优化计划会受到代价、plan_mode_seed参数、best_agg_plan参数影响，而hint、规则识别到的不会。

 版权所有 © 华为云计算技术有限公司

14.4.7.7 plan management

由于数据库参数设置的更改、版本的升级、数据的变化等，导致同一个语句的执行计划会存在差异，同时造成执行性能的劣化。plan management可通过为语句绑定指定的计划来防止计划跳变而产生的性能劣化。

plan management主要包括为计划生成唯一标识、为语句和计划生成sql_hash和plan_hash、计划保存和使用、计划失效等部分。

为计划生成唯一标识

如果要为语句指定一个固定的计划，那么每个语句及其计划必须拥有一个唯一的标识，即sql_hash和plan_hash。

可以通过开启GUC参数**enable_generate_plan_hash**为outline生成唯一标识plan_hash。

适用于生成 outline 的计划

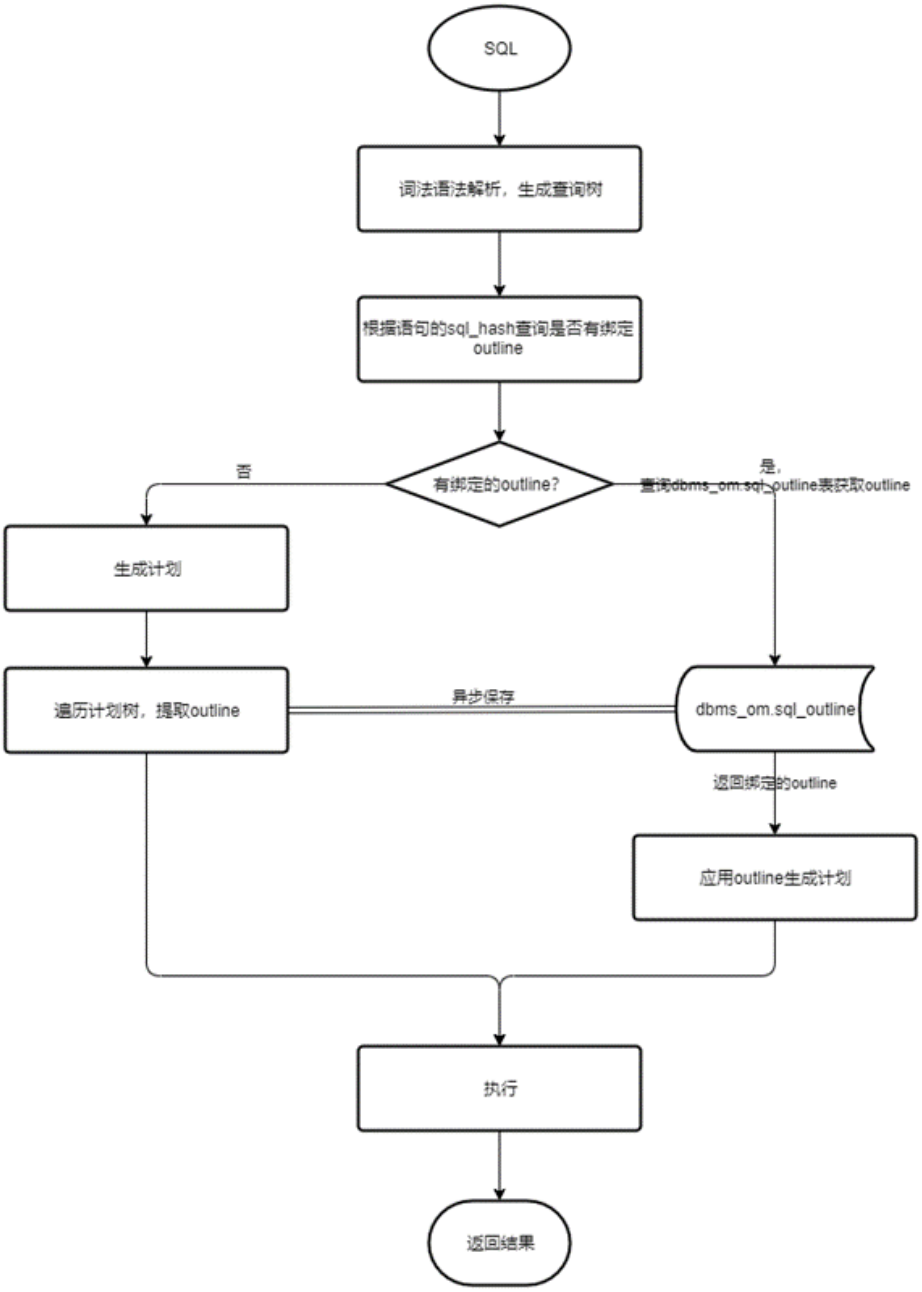
除FQS和CN强量化计划外，其他计划都会生成outline。

outline 保存和使用

下图是outline保存和使用在语句执行过程中处理流程。

版权所有 © 华为云计算技术有限公司

图 14-9 outline 保存和使用流程



outline保存

outline保存，即将计划自动生成的outline记录到**SQL_OUTLINE**系统表中，以供后续将其绑定给SQL语句。

- 如果同一个sql_hash的语句不存在该outline_name的outline，则保存该outline；若已经存在同名的outline，则不会重复保存。
- outline是在语句正确执行完成后才进行保存的，如果执行出错，则不会保存。

可以通过设置GUC参数**planmgmt_options**来保存outline，例如：

- 设置planmgmt_options = 'plan_save_mode_outline, plan_save_level_topsql'，满足记录topsql的语句才将其outline进行保存。

 版权所有 © 华为云计算技术有限公司

或

- 设置planmgmt_options = 'plan_save_mode_outline, plan_save_level_all'，表示将outline进行保存。

说明

plan management虽可以使语句执行性能稳定，但启用plan management也存在性能消耗，不适用于非资源消耗型的简单语句。因此建议设置planmgmt_options = 'plan_save_mode_outline, plan_save_level_topsql'，可避免将简单语句的计划也进行保存。

outline使用

使用pgxc_bind_plan函数为语句绑定一个outline后，可设置planmgmt_options = 'plan_save_mode_outline, plan_save_level_topsql, enable_plan_baseline'，使得执行语句时可以使用该绑定的outline。

须知

当前通过plan management支持自动提取以下hint：支持join顺序的leading hint、join方式的hint、scan方式的hint、指定倾斜列的skew hint和指定分布方式的stream hint。

通过plan management绑定outline来固定计划的根本是通过outline中的hint来固定计划，由于outline中支持自动提取的hint类型有限，有以下约束：

1. 如果sql中手工增删或修改了outline中不支持的hint，可能会出现绑定outline后，计划不能被固定：
   - no merge hint：sql1中指定了no merge hint，sql2和sql1的差别仅在于没有指定no merge hint，sql1和sql2的sql_hash是一样的，所以进行outline绑定就会同时给sql1和sql2都绑定，如果绑定的是sql1生成的outline，则sql2执行时绑定的outline就会不生效，同理反过来也是成立的。这是因为no merge hint限制了子查询的提升，语句块的层次不同，生成的outline中的hint应用的语句块在子查询提升和不提升时是不一样的，所以会导致绑定后无法找到hint中对应的表而无法生效。
   - GUC hint：与no merge hint同理。
2. 可通过人工修改自动生成的outline，添加outline中不支持自动提取的hint，来固定计划。

plan management 使用示例

步骤1 设置开启plan management。

```
SET resource_track_cost=0;
SET resource_track_duration=0;
SET enable_generate_plan_hash = on;
SET planmgmt_options='plan_save_mode_outline,plan_save_level_topsql,enable_plan_baseline';
```

步骤2 连接数据库创建表和索引。

```
CREATE TABLE t1(a1 int, b1 int, c1 serial) DISTRIBUTE BY hash(a1);
INSERT INTO t1(a1, b1) VALUES(generate_series(1, 10), generate_series(1, 8));
CREATE INDEX idx_t1_b1 on t1(b1);
CREATE TABLE t2(a2, b2, c2) AS SELECT * FROM t1;
```

步骤3 查看不使用索引和使用索引两个语句的计划。

查看不使用索引的语句计划：

 版权所有 © 华为云计算技术有限公司

```
EXPLAIN (verbose on, blockname on, outline on) select count(*) from t1, t2 where b1 = 1 and a1 = a2;
```

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 id |                   operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
----+-----------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Aggregate                                 |      1 |            |          |       8 | 9.02
  2 |    ->  Streaming (type: GATHER)               |      2 |            |          |       8 | 9.02
  3 |       ->  Aggregate                           |      2 |            | 1MB      |       8 | 3.02
  4 |          ->  Hash Join (5,6)                  |     20 |            | 1MB      |       0 | 2.98
  5 |             ->  Seq Scan on public.t2         |     40 |          5 | 1MB      |       4 | 1.20
  6 |             ->  Hash                          |     10 |          1 | 16MB     |       4 | 1.56
  7 |                ->  Streaming(type: BROADCAST) |     10 |            | 2MB      |       4 | 1.56
  8 |                   ->  Seq Scan on public.t1   |      5 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  4 --Hash Join (5,6)
        Hash Cond: (t2.a2 = t1.a1)
  8 --Seq Scan on public.t1
        Filter: (t1.b1 = 1)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Aggregate
        Output: pg_catalog.count(*)
  2 --Streaming (type: GATHER)
        Output: (count(*))
        Node/s: All datanodes
  3 --Aggregate
        Output: count(*)
  5 --Seq Scan on public.t2
        Output: t2.a2, t2.b2, t2.c2
  6 --Hash
        Output: t1.a1
  7 --Streaming(type: BROADCAST)
        Output: t1.a1
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  8 --Seq Scan on public.t1
        Output: t1.a1
        Distribute Key: t1.a1

Query Block Name / Object Alias (identified by plan id)
---------------------------------------------------------
  1 - sel$1
  5 - sel$1 / public.t2@"sel$1"
  8 - sel$1 / public.t1@"sel$1"

                        Outline Data
---------------------------------------------------------------
  /*+
      begin_outline_data
       Leading(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t2@"sel$1")
       Broadcast(@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t1@"sel$1")
      end_outline_data
  */

              ====== Query Summary =====
-------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 5408KB
Parser runtime: 0.044 ms
Planner runtime: 1.167 ms
Unique SQL Id: 1960277016
Unique SQL Hash: sql_6e8591b8291c3b458b26ce7419fd3704
Unique Plan Hash: plan_414b7354a6cfc22ccc44aa8244024921
(67 rows)
```

查看使用索引的语句计划：

```
EXPLAIN (verbose on, blockname on, outline on) select /*+ indexscan(t1 idx_t1_b1) */ count(*) from t1, t2
where b1 = 1 and a1 = a2;
```

 版权所有 © 华为云计算技术有限公司

```
                                             QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 id |                          operation                           | E-rows | E-distinct | E-memory | E-width | E-costs
----+--------------------------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Aggregate                                                |      1 |            |          |       8 | 20.05
  2 |    ->  Streaming (type: GATHER)                              |      2 |            |          |       8 | 20.05
  3 |       ->  Aggregate                                          |      2 |            | 1MB      |       8 | 14.05
  4 |          ->  Hash Join (5,6)                                 |     20 |            | 1MB      |       0 | 14.02
  5 |             ->  Seq Scan on public.t2                        |     40 | 5          | 1MB      |       4 | 1.20
  6 |             ->  Hash                                         |     10 | 1          | 16MB     |       4 | 12.60
  7 |                ->  Streaming(type: BROADCAST)                |     10 |            | 2MB      |       4 | 12.60
  8 |                   ->  Index Scan using idx_t1_b1 on public.t1 |      5 |            | 1MB      |       4 | 12.29

 Predicate Information (identified by plan id)
 ---------------------------------------------
   4 --Hash Join (5,6)
         Hash Cond: (t2.a2 = t1.a1)
   8 --Index Scan using idx_t1_b1 on public.t1
         Index Cond: (t1.b1 = 1)

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Aggregate
         Output: pg_catalog.count(*)
   2 --Streaming (type: GATHER)
         Output: (count(*))
         Node/s: All datanodes
   3 --Aggregate
         Output: count(*)
   5 --Seq Scan on public.t2
         Output: t2.a2, t2.b2, t2.c2
   6 --Hash
         Output: t1.a1
   7 --Streaming(type: BROADCAST)
         Output: t1.a1
         Spawn on: All datanodes
         Consumer Nodes: All datanodes
   8 --Index Scan using idx_t1_b1 on public.t1
         Output: t1.a1
         Distribute Key: t1.a1

 Query Block Name / Object Alias (identified by plan id)
 -------------------------------------------------------
   1 - sel$1
   5 - sel$1 / public.t2@"sel$1"
   8 - sel$1 / public.t1@"sel$1"

                        Outline Data
 -------------------------------------------------------------
   /*+
       begin_outline_data
        Leading[@"sel$1" public.t2@"sel$1" public.t1@"sel$1"]
        HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
        TableScan(@"sel$1" public.t2@"sel$1")
        Broadcast(@"sel$1" public.t1@"sel$1")
        IndexScan(@"sel$1" public.t1@"sel$1" idx_t1_b1)
       end_outline_data
   */

               ====== Query Summary =====
 -------------------------------------------------------
 System available mem: 4710400KB
 Query Max mem: 4710400KB
 Query estimated mem: 5408KB
 Parser runtime: 0.069 ms
 Planner runtime: 1.115 ms
 Unique SQL Id: 1960277016
 Unique SQL Hash: sql_6e8591b8291c3b458b26ce7419fd3704
 Unique Plan Hash: plan_00df934ed2620cb09c5f51a33f1945fd
(67 rows)
```

步骤4 查看sql_outline表中保存的计划。

```
SELECT sql_hash, plan_hash, outline_name, outline FROM dbms_om.sql_outline WHERE outline like '%public.t1%public.t2%';
```

```
               sql_hash                |               plan_hash                |              outline_name               |                          outline
---------------------------------------+----------------------------------------+-----------------------------------------+------------------------------------------------------------
 sql_6e8591b8291c3b458b26ce7419fd3704 | plan_414b7354a6cfc22ccc44aa8244024921 | outline_b390d2877f110c2ab5d03e39dca8180b | /*+                                                        +
                                       |                                        |                                         |     begin_outline_data                                     +
                                       |                                        |                                         |      Leading[@"sel$1" public.t2@"sel$1" public.t1@"sel$1"] +
                                       |                                        |                                         |      HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")+
                                       |                                        |                                         |      TableScan(@"sel$1" public.t2@"sel$1")                 +
                                       |                                        |                                         |      Broadcast(@"sel$1" public.t1@"sel$1")                 +
                                       |                                        |                                         |      TableScan(@"sel$1" public.t1@"sel$1")                 +
                                       |                                        |                                         |     end_outline_data                                       +
                                       |                                        |                                         | */                                                         +
                                       |                                        |                                         |
 sql_6e8591b8291c3b458b26ce7419fd3704 | plan_00df934ed2620cb09c5f51a33f1945fd | outline_ae16ae6300416c3efc96378c7b3a7e7b | /*+                                                        +
                                       |                                        |                                         |     begin_outline_data                                     +
                                       |                                        |                                         |      Leading[@"sel$1" public.t2@"sel$1" public.t1@"sel$1"] +
                                       |                                        |                                         |      HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")+
                                       |                                        |                                         |      TableScan(@"sel$1" public.t2@"sel$1")                 +
                                       |                                        |                                         |      Broadcast(@"sel$1" public.t1@"sel$1")                 +
                                       |                                        |                                         |      IndexScan(@"sel$1" public.t1@"sel$1" idx_t1_b1)       +
                                       |                                        |                                         |     end_outline_data                                       +
                                       |                                        |                                         | */                                                         +
                                       |                                        |                                         |
(2 rows)
```

使用索引的语句对应的sql_hash、plan_hash和outline_name分别为'sql_6e8591b8291c3b458b26ce7419fd3704'和

'plan_00df934ed2620cb09c5f51a33f1945fd'和'outline_ae16ae6300416c3efc96378c7b3a7e7b'。

步骤5 为语句绑定使用索引的计划。

```
SELECT pgxc_bind_plan('sql_6e8591b8291c3b458b26ce7419fd3704', 
'outline_ae16ae6300416c3efc96378c7b3a7e7b');
 pgxc_bind_plan 
----------------
 t
(1 row)
```

步骤6 查看语句是否使用了绑定的索引计划。

```
EXPLAIN (verbose on, blockname on, outline on) select count(*) from t1, t2 where b1 = 1 and a1 = a2;
```

```
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 id |                          operation                            | E-rows | E-distinct | E-memory | E-width | E-costs
----+---------------------------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Aggregate                                                 |      1 |            |          |       8 | 20.05
  2 |    ->  Streaming (type: GATHER)                               |      2 |            |          |       8 | 20.05
  3 |       ->  Aggregate                                           |      2 |            | 1MB      |       8 | 14.05
  4 |          ->  Hash Join (5,6)                                  |     20 |            | 1MB      |       0 | 14.02
  5 |             ->  Seq Scan on public.t2                         |     40 | 5          | 1MB      |       4 | 1.20
  6 |             ->  Hash                                          |     10 | 1          | 16MB     |       4 | 12.60
  7 |                ->  Streaming(type: BROADCAST)                 |     10 |            | 2MB      |       4 | 12.60
  8 |                   ->  Index Scan using idx_t1_b1 on public.t1 |      5 |            | 1MB      |       4 | 12.29

 Predicate Information (identified by plan id)
 ---------------------------------------------
   4 --Hash Join (5,6)
         Hash Cond: (t2.a2 = t1.a1)
   8 --Index Scan using idx_t1_b1 on public.t1
         Index Cond: (t1.b1 = 1)

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Aggregate
         Output: pg_catalog.count(*)
   2 --Streaming (type: GATHER)
         Output: (count(*))
         Node/s: All datanodes
   3 --Aggregate
         Output: count(*)
   5 --Seq Scan on public.t2
         Output: t2.a2, t2.b2, t2.c2
   6 --Hash
         Output: t1.a1
   7 --Streaming(type: BROADCAST)
         Output: t1.a1
         Spawn on: All datanodes
         Consumer Nodes: All datanodes
   8 --Index Scan using idx_t1_b1 on public.t1
         Output: t1.a1
         Distribute Key: t1.a1

 Query Block Name / Object Alias (identified by plan id)
 -------------------------------------------------------
   1 - sel$1
   5 - sel$1 / public.t2@"sel$1"
   8 - sel$1 / public.t1@"sel$1"

                        Outline Data
 -------------------------------------------------------------
   /*+
       begin_outline_data
        Leading(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
        HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
        TableScan(@"sel$1" public.t2@"sel$1")
        Broadcast(@"sel$1" public.t1@"sel$1")
        IndexScan(@"sel$1" public.t1@"sel$1" idx_t1_b1)
       end_outline_data
   */

                                         ====== Query Summary =====
 ---------------------------------------------------------------------------------------------------------------
 System available mem: 4710400KB
 Query Max mem: 4710400KB
 Query estimated mem: 5408KB
 Parser runtime: 0.056 ms
 Planner runtime: 1.117 ms
 Unique SQL Id: 1960277016
 Unique SQL Hash: sql_6e8591b8291c3b458b26ce7419fd3704
 Unique Plan Hash: plan_00df934ed2620cb09c5f51a33f1945fd
 Bind plan ('sql_6e8591b8291c3b458b26ce7419fd3704', 'outline_ae16ae6300416c3efc96378c7b3a7e7b') from cache is used
(68 rows)
```

从上面的计划可见，绑定的索引计划被使用。

步骤7 为语句解绑计划。

```
SELECT pgxc_unbind_plan('sql_6e8591b8291c3b458b26ce7419fd3704');
 pgxc_unbind_plan
------------------
 t
(1 row)
```

步骤8 查看解绑计划后的执行计划。

```
EXPLAIN (verbose on, blockname on, outline on) select count(*) from t1, t2 where b1 = 1 and a1 = a2;
```

```
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 id |                   operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
----+-----------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Aggregate                                 |      1 |            |          |       8 | 9.02
  2 |    ->  Streaming (type: GATHER)               |      2 |            |          |       8 | 9.02
  3 |       ->  Aggregate                           |      2 |            | 1MB      |       8 | 3.02
  4 |          ->  Hash Join (5,6)                  |     20 |            | 1MB      |       0 | 2.98
  5 |             ->  Seq Scan on public.t2         |     40 | 5          | 1MB      |       4 | 1.20
  6 |             ->  Hash                          |     10 | 1          | 16MB     |       4 | 1.56
  7 |                ->  Streaming(type: BROADCAST) |     10 |            | 2MB      |       4 | 1.56
  8 |                   ->  Seq Scan on public.t1   |      5 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  4 --Hash Join (5,6)
        Hash Cond: (t2.a2 = t1.a1)
  8 --Seq Scan on public.t1
        Filter: (t1.b1 = 1)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Aggregate
        Output: pg_catalog.count(*)
  2 --Streaming (type: GATHER)
        Output: (count(*))
        Node/s: All datanodes
  3 --Aggregate
        Output: count(*)
  5 --Seq Scan on public.t2
        Output: t2.a2, t2.b2, t2.c2
  6 --Hash
        Output: t1.a1
  7 --Streaming(type: BROADCAST)
        Output: t1.a1
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  8 --Seq Scan on public.t1
        Output: t1.a1
        Distribute Key: t1.a1

Query Block Name / Object Alias (identified by plan id)
---------------------------------------------------------
  1 - sel$1
  5 - sel$1 / public.t2@"sel$1"
  8 - sel$1 / public.t1@"sel$1"

                       Outline Data
-------------------------------------------------------------
  /*+
      begin_outline_data
       Leading(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t2@"sel$1")
       Broadcast(@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t1@"sel$1")
      end_outline_data
  */

              ====== Query Summary =====
-------------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 5408KB
Parser runtime: 0.044 ms
Planner runtime: 1.167 ms
Unique SQL Id: 1960277016
Unique SQL Hash: sql_6e8591b8291c3b458b26ce7419fd3704
Unique Plan Hash: plan_414b7354a6cfc22ccc44aa8244024921
(67 rows)
```

 版权所有 © 华为云计算技术有限公司

从上面的计划可见，索引计划不再被使用。

步骤9 重新绑定计划，删除索引。

```
SELECT pgxc_bind_plan('sql_6e8591b8291c3b458b26ce7419fd3704',
'outline_ae16ae6300416c3efc96378c7b3a7e7b');
 pgxc_bind_plan
----------------
 t
(1 row)

DROP INDEX idx_t1_b1;
```

步骤10 查看计划失效后的语句执行计划。

```
EXPLAIN (verbose on, blockname on, outline on) select count(*) from t1, t2 where b1 = 1 and a1 = a2;
```

```
WARNING:  Error hint: BIND HINT -- IndexScan(@"sel$1" public.t1@"sel$1" idx_t1_b1), index "idx_t1_b1" doesn't exist.
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 id |                  operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
----+----------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Aggregate                                |      1 |            |          |       8 | 9.02
  2 |    ->  Streaming (type: GATHER)              |      2 |            |          |       8 | 9.02
  3 |       ->  Aggregate                          |      2 |            | 1MB      |       8 | 3.02
  4 |          ->  Hash Join (5,6)                 |     20 |            | 1MB      |       0 | 2.98
  5 |             ->  Seq Scan on public.t2        |     40 | 5          | 1MB      |       4 | 1.20
  6 |             ->  Hash                         |     10 | 1          | 16MB     |       4 | 1.56
  7 |                ->  Streaming(type: BROADCAST) |     10 |            | 2MB      |       4 | 1.56
  8 |                   ->  Seq Scan on public.t1  |      5 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  4 --Hash Join (5,6)
        Hash Cond: (t2.a2 = t1.a1)
  8 --Seq Scan on public.t1
        Filter: (t1.b1 = 1)

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Aggregate
        Output: pg_catalog.count(*)
  2 --Streaming (type: GATHER)
        Output: (count(*))
        Node/s: All datanodes
  3 --Aggregate
        Output: count(*)
  5 --Seq Scan on public.t2
        Output: t2.a2, t2.b2, t2.c2
  6 --Hash
        Output: t1.a1
  7 --Streaming(type: BROADCAST)
        Output: t1.a1
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  8 --Seq Scan on public.t1
        Output: t1.a1
        Distribute Key: t1.a1

Query Block Name / Object Alias (identified by plan id)
-------------------------------------------------------
  1 - sel$1
  5 - sel$1 / public.t2@"sel$1"
  8 - sel$1 / public.t1@"sel$1"

                      Outline Data
-------------------------------------------------------------
  /*+
      begin_outline_data
       Leading(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       HashJoin(@"sel$1" public.t2@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t2@"sel$1")
       Broadcast(@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$1" public.t1@"sel$1")
      end_outline_data
  */

                                          ====== Query Summary =====
-------------------------------------------------------------------------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 5408KB
Parser runtime: 0.118 ms
Planner runtime: 5.432 ms
Unique SQL Id: 1960277016
Unique SQL Hash: sql_6e8591b8291c3b458b26ce7419fd3704
Unique Plan Hash: plan_414b7354a6cfc22ccc44aa8244024921
Bind plan ('sql_6e8591b8291c3b458b26ce7419fd3704', 'outline_ae16ae6300416c3efc96378c7b3a7e7b') from system table is used
(68 rows)
```

 版权所有 © 华为云计算技术有限公司

从上面的计划可见，虽然绑定的outline被使用，但绑定的outline中的索引hint由于索引删除，导致索引hint未生效。

步骤11 含no merge hint的语句的计划。

```
EXPLAIN (verbose on, blockname on, outline on) select * from t1 where a1 in (select /*+ no merge */ a2 from t2 where b2 < 10);
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 id |                operation                | E-rows | E-distinct | E-memory | E-width | E-costs
----+-----------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)            |     20 |            |          |      12 | 8.88
  2 |    ->  Seq Scan on public.t1            |     20 |            | 1MB      |      12 | 2.88
  3 |       ->  Materialize  [2, SubPlan 1]   |     80 |            | 16MB     |       4 | 1.53
  4 |          ->  Streaming(type: BROADCAST) |     80 |            | 2MB      |       4 | 1.33
  5 |             ->  Seq Scan on public.t2   |     40 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on public.t1
        Filter: (hashed SubPlan 1)
  5 --Seq Scan on public.t2
        Filter: (t2.b2 < 10)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: t1.a1, t1.b1, t1.c1
        Node/s: All datanodes
  2 --Seq Scan on public.t1
        Output: t1.a1, t1.b1, t1.c1
        Distribute Key: t1.a1
  3 --Materialize
        Output: t2.a2
  4 --Streaming(type: BROADCAST)
        Output: t2.a2
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  5 --Seq Scan on public.t2
        Output: t2.a2

Query Block Name / Object Alias (identified by plan id)
---------------------------------------------------------
  1 - sel$1
  2 - sel$1 / public.t1@"sel$1"
  5 - sel$2 / public.t2@"sel$2"

                Outline Data
-----------------------------------------
  /*+
      begin_outline_data
       TableScan(@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$2" public.t2@"sel$2")
      end_outline_data
  */

              ====== Query Summary =====
---------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 3108KB
Parser runtime: 0.094 ms
Planner runtime: 1.416 ms
Unique SQL Id: 856581108
Unique SQL Hash: sql_14150502dc61e290b4c81c5c43d7446c
Unique Plan Hash: plan_a65528a87d6ebf5fb44766f192bf9383
(57 rows)
```

步骤12 无no merge hint的语句的计划。

```
EXPLAIN (verbose on, blockname on, outline on) select * from t1 where a1 in (select a2 from t2 where b2 < 10);
```

 版权所有 © 华为云计算技术有限公司

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 id |                  operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
----+----------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                 |     40 |            |          |      12 | 9.74
  2 |    ->  Hash Join (3,4)                       |     40 |            | 1MB      |      12 | 3.74
  3 |       ->  Seq Scan on public.t1              |     40 | 5          | 1MB      |      12 | 1.20
  4 |       ->  Hash                               |     10 | 5          | 16MB     |       4 | 2.21
  5 |          ->  HashAggregate                   |     10 |            | 16MB     |       4 | 2.21
  6 |             ->  Streaming(type: REDISTRIBUTE) |     18 |            | 2MB      |       4 | 2.13
  7 |                ->  HashAggregate             |     18 |            | 16MB     |       4 | 1.39
  8 |                   ->  Seq Scan on public.t2  |     40 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
        Hash Cond: (t1.a1 = t2.a2)
  8 --Seq Scan on public.t2
        Filter: (t2.b2 < 10)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: t1.a1, t1.b1, t1.c1
        Node/s: All datanodes
  2 --Hash Join (3,4)
        Output: t1.a1, t1.b1, t1.c1
  3 --Seq Scan on public.t1
        Output: t1.a1, t1.b1, t1.c1
        Distribute Key: t1.a1
  4 --Hash
        Output: t2.a2
  5 --HashAggregate
        Output: t2.a2
        Group By Key: t2.a2
  6 --Streaming(type: REDISTRIBUTE)
        Output: t2.a2
        Distribute Key: t2.a2
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  7 --HashAggregate
        Output: t2.a2
        Group By Key: t2.a2
  8 --Seq Scan on public.t2
        Output: t2.a2

Query Block Name / Object Alias (identified by plan id)
---------------------------------------------------------
  1 - sel$1
  3 - sel$1 / public.t1@"sel$1"
  8 - sel$1 / public.t2@"sel$2"

                       Outline Data
---------------------------------------------------------------
  /*+
      begin_outline_data
       Leading(@"sel$1" public.t1@"sel$1" public.t2@"sel$2")
       HashJoin(@"sel$1" public.t1@"sel$1" public.t2@"sel$2")
       TableScan(@"sel$1" public.t1@"sel$1")
       Redistribute(@"sel$1" public.t2@"sel$2")
       TableScan(@"sel$1" public.t2@"sel$2")
      end_outline_data
  */

              ====== Query Summary =====
-----------------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 4384KB
Parser runtime: 0.048 ms
Planner runtime: 1.402 ms
Unique SQL Id: 856581108
Unique SQL Hash: sql_14150502dc61e290b4c81c5c43d7446c
Unique Plan Hash: plan_3057ac5254814929809 3f4100167838a
(72 rows)
```

步骤13 查询含no merge hint和无no merge hint生成的outline。

```
SELECT sql_hash, plan_hash, outline_name, outline FROM dbms_om.sql_outline WHERE sql_hash =
'sql_14150502dc61e290b4c81c5c43d7446c';
```

 版权所有 © 华为云计算技术有限公司

```
             sql_hash             |            plan_hash             |            outline_name            |                        outline
----------------------------------+----------------------------------+------------------------------------+-------------------------------------------------------
 sql_14150502dc61e290b4c81c5c43d7446c | plan_3057ac5254814929809 3f4100167838a | outline_7e9d299e8285e1ce916672d52d8fabal | /*+                                                   +
                                  |                                  |                                    |     begin_outline_data                                +
                                  |                                  |                                    |      Leading(@"sel$1" public.t1@"sel$1" public.t2@"sel$2"] +
                                  |                                  |                                    |      HashJoin(@"sel$1" public.t1@"sel$1" public.t2@"sel$2")+
                                  |                                  |                                    |      TableScan(@"sel$1" public.t1@"sel$1")            +
                                  |                                  |                                    |      Redistribute(@"sel$1" public.t2@"sel$2")         +
                                  |                                  |                                    |      TableScan(@"sel$1" public.t2@"sel$2")            +
                                  |                                  |                                    |     end_outline_data                                  +
                                  |                                  |                                    | */                                                    +
                                  |                                  |                                    |
 sql_14150502dc61e290b4c81c5c43d7446c | plan_a65528a87d6ebf5fb44766f192bf9383 | outline_a2cdc54106b991174cc0a35b8e0f0b29 | /*+                                                   +
                                  |                                  |                                    |     begin_outline_data                                +
                                  |                                  |                                    |      TableScan(@"sel$1" public.t1@"sel$1")            +
                                  |                                  |                                    |      TableScan(@"sel$2" public.t2@"sel$2")            +
                                  |                                  |                                    |     end_outline_data                                  +
                                  |                                  |                                    | */                                                    +
                                  |                                  |                                    |
(2 rows)
```

步骤14 绑定含no merge hint的outline。

```
SELECT pgxc_bind_plan('sql_14150502dc61e290b4c81c5c43d7446c',
'outline_a2cdc54106b991174cc0a35b8e0f0b29');
 pgxc_bind_plan
----------------
 t
(1 row)
```

步骤15 对无no merge hint的语句，应用绑定的outline后，仍产生了子查询提升。

```
EXPLAIN (verbose on, blockname on, outline on) select * from t1 where a1 in (select a2 from t2 where b2 < 10);
```

 版权所有 © 华为云计算技术有限公司

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 id |                  operation                  | E-rows | E-distinct | E-memory | E-width | E-costs
----+---------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                |     40 |            |          |      12 | 9.74
  2 |    ->  Hash Join (3,4)                      |     40 |            | 1MB      |      12 | 3.74
  3 |       ->  Seq Scan on public.t1             |     40 | 5          | 1MB      |      12 | 1.20
  4 |       ->  Hash                              |     10 | 5          | 16MB     |       4 | 2.21
  5 |          ->  HashAggregate                  |     10 |            | 16MB     |       4 | 2.21
  6 |             ->  Streaming(type: REDISTRIBUTE) |     18 |            | 2MB      |       4 | 2.13
  7 |                ->  HashAggregate            |     18 |            | 16MB     |       4 | 1.39
  8 |                   ->  Seq Scan on public.t2 |     40 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Join (3,4)
        Hash Cond: (t1.a1 = t2.a2)
  8 --Seq Scan on public.t2
        Filter: (t2.b2 < 10)

Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: t1.a1, t1.b1, t1.c1
        Node/s: All datanodes
  2 --Hash Join (3,4)
        Output: t1.a1, t1.b1, t1.c1
  3 --Seq Scan on public.t1
        Output: t1.a1, t1.b1, t1.c1
        Distribute Key: t1.a1
  4 --Hash
        Output: t2.a2
  5 --HashAggregate
        Output: t2.a2
        Group By Key: t2.a2
  6 --Streaming(type: REDISTRIBUTE)
        Output: t2.a2
        Distribute Key: t2.a2
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  7 --HashAggregate
        Output: t2.a2
        Group By Key: t2.a2
  8 --Seq Scan on public.t2
        Output: t2.a2

Query Block Name / Object Alias (identified by plan id)
-------------------------------------------------------
  1 - sel$1
  3 - sel$1 / public.t1@"sel$1"
  8 - sel$1 / public.t2@"sel$2"

                      Outline Data
-------------------------------------------------------------
  /*+
      begin_outline_data
       Leading(@"sel$1" public.t1@"sel$1" public.t2@"sel$2")
       HashJoin(@"sel$1" public.t1@"sel$1" public.t2@"sel$2")
       TableScan(@"sel$1" public.t1@"sel$1")
       Redistribute(@"sel$1" public.t2@"sel$2")
       TableScan(@"sel$1" public.t2@"sel$2")
      end_outline_data
  */

                                        ====== Query Summary =====
---------------------------------------------------------------------------------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 4384KB
Parser runtime: 0.039 ms
Planner runtime: 1.354 ms
Unique SQL Id: 856581108
Unique SQL Hash: sql_14150502dc61e290b4c81c5c43d7446c
Unique Plan Hash: plan_3057ac5254814929809 3f4100167838a
Bind plan ('sql_14150502dc61e290b4c81c5c43d7446c', 'outline_a2cdc54106b991174cc0a35b8e0f0b29') from cache is used
(73 rows)
```

步骤16 人工修改绑定的outline添加no merge hint，并重新创建到dbms_om.sql_outline中。

```
CREATE OUTLINE outline_nomerge for sql_14150502dc61e290b4c81c5c43d7446c using
'/*+
   begin_outline_data
    no merge@"sel$2"
    TableScan(@"sel$1" public.t1@"sel$1")
    TableScan(@"sel$2" public.t2@"sel$2")
   end_outline_data
*/'
```

 版权所有 © 华为云计算技术有限公司

步骤17 将新创建的含no merge hint的outline进行绑定。

```
SELECT pgxc_bind_plan('sql_14150502dc61e290b4c81c5c43d7446c', 'outline_nomerge');
 pgxc_bind_plan
----------------
 t
(1 row)
```

步骤18 对无no merge hint的语句，应用绑定的outline后，不再产生子查询提升。

```
EXPLAIN (verbose on, blockname on, outline on) select * from t1 where a1 in (select a2 from t2 where b2 < 10);
```

```
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 id |                operation                | E-rows | E-distinct | E-memory | E-width | E-costs
----+-----------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)            |     20 |            |          |      12 | 8.88
  2 |    ->  Seq Scan on public.t1            |     20 |            | 1MB      |      12 | 2.88
  3 |       ->  Materialize  [2, SubPlan 1]   |     80 |            | 16MB     |       4 | 1.53
  4 |          ->  Streaming(type: BROADCAST) |     80 |            | 2MB      |       4 | 1.33
  5 |             ->  Seq Scan on public.t2   |     40 |            | 1MB      |       4 | 1.25

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on public.t1
        Filter: (hashed SubPlan 1)
  5 --Seq Scan on public.t2
        Filter: (t2.b2 < 10)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: t1.a1, t1.b1, t1.c1
        Node/s: All datanodes
  2 --Seq Scan on public.t1
        Output: t1.a1, t1.b1, t1.c1
        Distribute Key: t1.a1
  3 --Materialize
        Output: t2.a2
  4 --Streaming(type: BROADCAST)
        Output: t2.a2
        Spawn on: All datanodes
        Consumer Nodes: All datanodes
  5 --Seq Scan on public.t2
        Output: t2.a2

Query Block Name / Object Alias (identified by plan id)
---------------------------------------------------------
  1 - sel$1
  2 - sel$1 / public.t1@"sel$1"
  5 - sel$2 / public.t2@"sel$2"

                Outline Data
-------------------------------------------
  /*+
      begin_outline_data
       TableScan(@"sel$1" public.t1@"sel$1")
       TableScan(@"sel$2" public.t2@"sel$2")
      end_outline_data
  */

                                  ====== Query Summary =====
-----------------------------------------------------------------------------------------------
System available mem: 4710400KB
Query Max mem: 4710400KB
Query estimated mem: 3108KB
Parser runtime: 0.111 ms
Planner runtime: 4.668 ms
Unique SQL Id: 856581108
Unique SQL Hash: sql_14150502dc61e290b4c81c5c43d7446c
Unique Plan Hash: plan_a65528a87d6ebf5fb44766f192bf9383
Bind plan ('sql_14150502dc61e290b4c81c5c43d7446c', 'outline_nomerge') from system table is used
(58 rows)
```

----结束

14.4.7.8 磁盘缓存主动预热调优

该功能仅9.1.0.200及以上版本支持。

背景介绍

当前存算分离架构为了降低存储成本会将用户数据存储到obs，这样会导致用户每次查询数据时都要发生网络IO去obs取数据，因此当前存算分离架构提供了磁盘缓存的能

版权所有 © 华为云计算技术有限公司

力，将用户预查询的数据缓存到本地磁盘，当实际查询数据流程时，数据已经缓存在本地，从而提升查询速度，在降低用户存储成本的同时尽可能地降低性能损耗。

当前采用了LRU2Q的算法来管理磁盘缓存。LRU2Q共包含3个队列，分别是A1in/A1out/Am，其中数据首次命中时会进入到A1in队列中，后续如果A1in队列满了（可调整guc参数，通常默认值为整个队列大小的0.25倍），会淘汰到A1out（实际并不存储数据到缓存，只是A1in队列淘汰的延长记忆），等A1out队列中的block被命中时，才会插入到Am队列。在Am队列中的数据，是最热的数据。

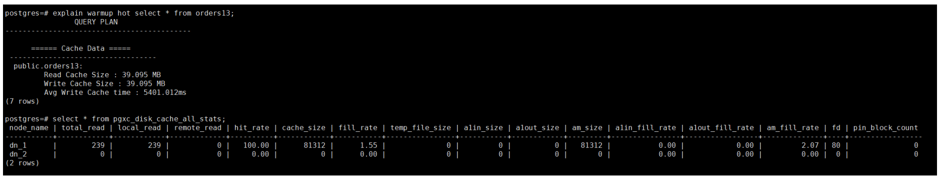
上述LRU2Q的淘汰策略对普通查询来说已经足够了，但是可能会存在的场景是：用户会将一些大表（比如查询数据接近磁盘缓存大小）和一些小表频繁join，此时如果继续按照普通LRU2Q的淘汰策略，对于频繁使用的小表来说可能会发生A1in满了然后淘汰到A1out，再次命中进入Am的过程，这样会频繁发生网络IO去重新缓存小表的数据，此时会极大的降低大表join小表的性能。

调优语法

当前提供了一种新的调优策略，即允许用户将上述描述场景中的小表数据直接入到Am队列中，保证小表的数据一直是热数据，join时不会频繁的发生网络IO而去重新加载小表数据，语法格式支持以下三种：

1. 发生实际查询操作，并将查询的数据直接进入到Am队列。

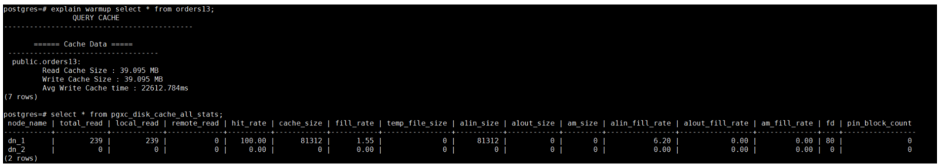
```
explain warmup hot select ...;
```



```
postgres=# explain warmup hot select * from orders13;
                QUERY PLAN
-------------------------------------------

      ====== Cache Data =====
 ----------------------------------
  public.orders13:
        Read Cache Size : 39.095 MB
        Write Cache Size : 39.095 MB
        Avg Write Cache time : 5401.012ms
(7 rows)

postgres=# select * from pgxc_disk_cache_all_stats;
 node_name | total_read | local_read | remote_read | hit_rate | cache_size | fill_rate | temp_file_size | a1in_size | a1out_size | am_size | a1in_fill_rate | a1out_fill_rate | am_fill_rate | fd | pin_block_count
 dn_1      |        239 |        239 |           0 |   100.00 |      81312 |      1.55 |              0 |         0 |          0 |   81312 |           0.00 |            0.00 |         2.07 | 80 |               0
 dn_2      |          0 |          0 |           0 |     0.00 |          0 |      0.00 |              0 |         0 |          0 |       0 |           0.00 |            0.00 |         0.00 |  0 |               0
(2 rows)
```

2. 发生实际查询操作，查询的数据按照A1in > A1out > Am的顺序进行。

```
explain warmup select ...;
```



```
postgres=# explain warmup select * from orders13;
                QUERY CACHE
-------------------------------------------

      ====== Cache Data =====
 ----------------------------------
  public.orders13:
        Read Cache Size : 39.095 MB
        Write Cache Size : 39.095 MB
        Avg Write Cache time : 22612.784ms
(7 rows)

postgres=# select * from pgxc_disk_cache_all_stats;
 node_name | total_read | local_read | remote_read | hit_rate | cache_size | fill_rate | temp_file_size | a1in_size | a1out_size | am_size | a1in_fill_rate | a1out_fill_rate | am_fill_rate | fd | pin_block_count
 dn_1      |        239 |        239 |           0 |   100.00 |      81312 |      1.55 |              0 |     81312 |          0 |       0 |           6.20 |            0.00 |         0.00 | 80 |               0
 dn_2      |          0 |          0 |           0 |     0.00 |          0 |      0.00 |              0 |         0 |          0 |       0 |           0.00 |            0.00 |         0.00 |  0 |               0
(2 rows)
```

3. 不发生实际的查询操作，和之前explain逻辑保持不变。

```
explain select ...;
```

14.4.7.9 通过主外键约束消除多余的 JOIN

本文介绍如何使用主键（PRIMARY KEY）与外键（FOREIGN KEY）之间的约束关系来优化查询计划，消除多余的JOIN操作。

前提条件

- 仅9.1.0.200及以上集群版本支持通过主外键约束消除多余的JOIN。
- 创建FOREIGN KEY前，需开启声明外键约束功能：

```
set info_constraint_options = 'foreign_key';
```

功能介绍

在关系型数据库中，使用JOIN连接可以将多张表数据组合在一起。在大型数据分析场景中，多表JOIN是非常常见的操作。与此同时，JOIN操作SQL执行中也是最为耗时的

操作之一。因此JOIN消除是一种能极大提升SQL执行性能的优化技术之一。其中，根据用户定义的表直接的主外键约束信息进行JOIN消除是一种具体的优化方法。这些约束提供了关于表与表之间关系的信息，可以用来减少不必要的JOIN操作，减少查询时间，提高数据库性能。

DWS数据库支持通过设置GUC参数**rewrite_rule**来开启或关闭使用外键约束信息消除多余JOIN操作的功能。具体设置如下：

- 开启JOIN消除优化功能。开启后，系统在SQL执行过程中，会获取表之间已经定义好的主外键约束关系，并根据此信息结合JOIN消除规则，进行JOIN消除。
  ```
  SET rewrite_rule = 'join_elimination';
  ```
- 关闭JOIN消除优化功能。
  ```
  SET rewrite_rule = '';
  ```

操作流程

1. 声明FOREIGN KEY。
2. 应用JOIN消除优化查询。

声明 FOREIGN KEY

- 新建表时，通过CREATE TABLE创建FOREIGN KEY。
- 对于已存在的表，通过ALTER TABLE创建或删除FOREIGN KEY。

⚠ 注意

- 您需要自行确保主键和外键之间的数据约束关系。DWS不会进行数据的约束检查。如果用户未能保证正确的主键与外键约束关系，开启join消除后可能会造成查询返回的结果集错误问题。
- DROP TABLE命令会自动删除外键，DWS不会进行检验和报错。
- 如需使用JOIN消除优化，外键列须同时带有NOT NULL约束；若无NOT NULL约束，则优化器会自动添加外键列的IS NOT NULL过滤条件来确保外键不为NULL，然后应用JOIN消除优化。

1. **通过CREATE TABLE创建FOREIGN KEY。**
   - 创建一个带有主键的表item。
     ```
     CREATE TABLE item(a1 INT PRIMARY KEY, b1 INT) WITH (ORIENTATION=COLUMN) distribute by HASH(a1);
     ```
   - 通过CREATE TABLE语句创建带有外键约束的表store, 外键列为a2带有NOT NULL 约束，指向item表的主键列a1。
     ```
     CREATE TABLE store(a2 int NOT NULL REFERENCES item(a1), b2 int);
     ```
2. **通过ALTER TABLE创建或删除FOREIGN KEY。**
   - 创建一个带有主键的表item。
     ```
     CREATE TABLE item(a1 INT PRIMARY KEY, b1 INT) WITH (ORIENTATION=COLUMN) distribute by HASH(a1);
     ```
   - 通过CREATE TABLE语句创建一张非外键表store。
     ```
     CREATE TABLE store(a2 int NOT NULL , b2 int);
     ```
   - 通过ALTER TABLE为store表增加外键。
     ```
     ALTER TABLE store ADD FOREIGN KEY (a2) REFERENCES item(a1);
     ```
   - 通过ALTER TABLE 为store表删除外键。
     ```
     ALTER TABLE store DROP CONSTRAINT store_a2_fkey;
     ```


版权所有 © 华为云计算技术有限公司

3. **查看表上的外键约束。**

```
postgres=# \d+ store
                        Table "public.store"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a2     | integer | not null  | plain   |              |
 b2     | integer |           | plain   |              |
Foreign-key constraints:
    "store_a2_fkey" FOREIGN KEY (a2) REFERENCES item(a1)
    "store_a2_fkey1" FOREIGN KEY (a2) REFERENCES item(a1)
Has OIDs: no
Distribute By: HASH(a2)
Location Nodes: ALL DATANODES
```

4. **DWS还提供函数验证外键约束的合法性，用户需手动调用。**

```
SELECT check_foreign_key_constraint('store');
```

```
postgres=# select check_foreign_key_constraint('store');
        check_foreign_key_constraint
---------------------------------------------
 store satisfied all Foreign key constraints
(1 row)
```

应用 JOIN 消除优化查询

JOIN消除是在JOIN连接中，利用主外键约束消除不必要的JOIN关系，以简化查询计划并提升查询性能。

前提条件：将join_elimination加入rewrite_rule中。

```
SET rewrite_rule='join_elimination';
```

- **两表内连接主外键消除。**

  test2.a2 为外键，test1.a1为主键。

```
EXPLAIN (COSTS OFF, NODES OFF) SELECT a1, a2 FROM test1, test2 WHERE a1 = a2 ORDER BY 1,2;
          QUERY PLAN
-----------------------------------
 id |         operation
 ----+-------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Sort
  3 |       ->  Seq Scan on test2
EXPLAIN (COSTS OFF, NODES OFF) SELECT a1, a2 FROM test1 inner join test2 on a1 = a2 ORDER BY 1,2;
          QUERY PLAN
-----------------------------------
 id |         operation
 ----+-------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Sort
  3 |       ->  Seq Scan on test2
```

- **多表内连接消除 vtest2 -> (vtest1, vtest3)。**

  vtest2是外键表，vtest1、vtest3是主键表。

```
EXPLAIN (COSTS OFF, NODES OFF)select vtest2.* from vtest2, vtest1, vtest3 where a2 = a1 and b2 = a3;
              QUERY PLAN
-----------------------------------------------
 id |            operation
 ----+-----------------------------------------
  1 | ->  Row Adapter
```


版权所有 © 华为云计算技术有限公司

```
  2 |    ->  Vector Streaming (type: GATHER)
  3 |       ->  CStore Scan on vtest2
```

- **多表串联内连接消除 test2 -> test1 -> test3。**

  test2是外键表，test1既是外键表也是主键表，test3是主键表。

```
EXPLAIN (COSTS OFF, NODES OFF) SELECT a1, a2 FROM test1, test2, test3 WHERE a1 = a2 and b1 = a3 ORDER BY 1,2;
            QUERY PLAN
-------------------------------------
 id |          operation
 ----+-----------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Sort
  3 |       ->  Seq Scan on test2
EXPLAIN (COSTS OFF, NODES OFF) SELECT a1, a2 FROM test2 inner join (select test1.* from test1 inner join test3 on b1 = a3) on a1 = a2;;
            QUERY PLAN
-------------------------------------
 id |          operation
 ----+-----------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Seq Scan on test2
```

- **无法消除的场景。**

  如果查询中引用了主键表的非主键列，内连接也无法消除 （ 如下示例需要返回test1.b1，则无法消除 ）。

```
EXPLAIN (COSTS OFF, NODES OFF)
SELECT b1, a2 FROM test1, test2 WHERE a1 = a2 ORDER BY 1,2;
                QUERY PLAN
-------------------------------------------------
 id |            operation
 ----+-----------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |    ->  Sort
  3 |       ->  Hash Join (4,5)
  4 |          ->  Seq Scan on test2
  5 |          ->  Hash
  6 |             ->  Seq Scan on test1

 Predicate Information (identified by plan id)
 -------------------------------------------------
  3 --Hash Join (4,5)
       Hash Cond: (test2.a2 = test1.a1)
```

14.4.7.10 SQL 语句改写规则

根据数据库的SQL执行机制以及大量的实践，总结发现：通过一定的规则调整SQL语句，在保证结果正确的基础上，能够提高SQL执行效率。如果遵守下列规则，能够大幅度提升业务查询效率。

- **使用union all代替union**

  union在合并两个集合时会执行去重操作，而union all则直接将两个结果集合并、不执行去重。执行去重会消耗大量的时间，因此，在一些实际应用场景中，如果通过业务逻辑已确认两个集合不存在重叠，可用union all替代union以便提升性能。

- **join列增加非空过滤条件**

  若join列上的NULL值较多，则可以加上is not null过滤条件，以实现数据的提前过滤，提高join效率。

- **not in转not exists**

  not in语句需要使用nestloop anti join来实现，而not exists则可以通过hash anti join来实现。在join列不存在null值的情况下，not exists和not in等价。因此在确

 版权所有 © 华为云计算技术有限公司

保没有null值时，可以通过将not in转换为not exists，通过生成hash join来提升查询效率。

如下所示，如果t2.d2字段中没有null值（t2.d2字段在表定义中not null）查询可以修改为

```
SELECT * FROM t1 WHERE  NOT EXISTS (SELECT * FROM t2 WHERE t1.c1=t2.d2);
```

产生的计划如下：

图 14-10 not exists 执行计划

```
 id |                 operation
----+-------------------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |     ->  Hash Right Anti Join (3, 5)
  3 |        ->  Streaming(type: REDISTRIBUTE)
  4 |           ->  Seq Scan on t2
  5 |        ->  Hash
  6 |           ->  Seq Scan on t1

Predicate Information (identified by plan id)
---------------------------------------------
  2 --Hash Right Anti Join (3, 5)
        Hash Cond: (t2.d2 = t1.c1)
(13 rows)
```

- **选择hashagg。**

  查询中GROUP BY语句如果生成了groupagg+sort的执行计划，性能通常较差，可以通过调大GUC参数work_mem促使优化器生成hashagg的执行计划，从而避免排序操作，提升性能。

- **尝试将函数替换为case语句。**

  如果出现过多的函数调用导致性能下降较大，可根据实际情况将可下推的函数改写为CASE表达式。

- **避免对索引使用函数或表达式运算。**

  对索引使用函数或表达式运算会停止使用索引转而执行全表扫描。

- **尽量避免在where子句中使用!=或<>操作符、null值判断、or连接、参数隐式转换。**

- **对复杂SQL语句进行拆分。**

  对于过于复杂并且不易通过以上方法调整性能的SQL可以考虑拆分的方法，把SQL中某一部分拆分成独立的SQL并把执行结果存入临时表，拆分常见的场景包括但不限于：

  - 作业中多个SQL有同样的子查询，并且子查询数据量较大。
  - Plan cost计算不准，导致子查询hash bucket太小，比如实际数据1000W行，hash bucket只有1000。
  - 函数（如substr,to_number）导致大数据量子查询选择度计算不准。
  - 多DN环境下对大表做broadcast的子查询。

14.4.8 优化器参数调整

本节将介绍影响DWS SQL调优性能的关键CN配置参数，配置方法参见**设置GUC参数**。

版权所有 © 华为云计算技术有限公司

表 14-18 CN 配置参数

参数/参考值	描述
enable_nestloop=on	控制查询优化器对嵌套循环连接（Nest Loop Join）类型的使用。当设置为“on”后，优化器优先使用Nest Loop Join；当设置为“off”后，优化器在存在其他方法时将优先选择其他方法。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_nestloop to off;` 此参数默认设置为“on”，但实际调优中应根据情况选择是否关闭。一般情况下，在三种join方式（Nested Loop、Merge Join和Hash Join）里，Nested Loop性能较差，实际调优中可以选择关闭。
enable_bitmapscan=on	控制查询优化器对位图扫描规划类型的使用。设置为“on”，表示使用；设置为“off”，表示不使用。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_bitmapscan to off;` bitmapscan扫描方式适用于“where a > 1 and b > 1”且a列和b列都有索引这种查询条件，但有时其性能不如indexscan。因此，现场调优如发现查询性能较差且计划中有bitmapscan算子，可以关闭bitmapscan，看性能是否有提升。
enable_fast_query_shipping=on	控制查询优化器是否使用分布式框架，执行快速执行计划。设置为“on”，表示执行计划在CN和DN上各自生成；设置为“off”，表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_fast_query_shipping to off;`
enable_hashagg=on	控制优化器对Hash聚集规划类型的使用。
enable_hashjoin=on	控制优化器对Hash连接规划类型的使用。
enable_mergejoin=on	控制优化器对融合连接规划类型的使用。
enable_indexscan=on	控制优化器对索引扫描规划类型的使用。
enable_indexonlyscan=on	控制优化器对仅索引扫描规划类型的使用。
enable_seqscan=on	控制优化器对顺序扫描规划类型的使用。无法完全消除顺序扫描，但关闭该变量会让优化器在存在其他方法的时候优先选择其他方法。

版权所有 © 华为云计算技术有限公司

参数/参考值	描述
enable_sort=on	控制优化器使用的排序步骤。无法完全消除明确的排序，但关闭该变量可以让优化器在存在其他方法的时候优先选择其他方法。
enable_broadcast=on	控制查询优化器对于broadcast广播模式数据传输的使用。此方式网络传输数据量较大，因此当网络传输节点（Stream）实际数据量较大而估算不准时，可以将该参数设置为off，看性能是否有提升。
enable_redistribute=on	控制查询优化器对于local redistribute和split redistribute重分布模式数据传输的使用。此参数与enable_broadcast是对应关系。优化器可能会对local broadcast和split broadcast代价估计偏大，从而选择了local redistribute或者split redistribute重分布的计划。这有可能导致性能劣化。因此当网络传输节点（Stream）实际数据量较小时，可以将该参数设置为off，让优化器优先选择broadcast广播模式，看性能是否有提升。
rewrite_rule	控制优化器是否启用特定组合的重写规则。

14.4.9 使用 Plan Hint 进行调优

14.4.9.1 Plan Hint 调优概述

Plan Hint为用户提供了直接影响执行计划生成的手段，用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息，指定配置参数的值等多个手段来进行执行计划的调优，以提升查询的性能。

功能描述

Plan Hint支持在SELECT、INSERT、UPDATE、MERGE、DELETE关键字后通过如下形式指定：

```
/*+ <plan hint> */
```

可以同时指定多个hint，之间使用空格分隔。hint只能hint当前层的计划，对于子查询计划的hint，需要在子查询对应的关键字后指定hint。

例如：

```
SELECT /*+ <plan_hint1> <plan_hint2> */ * FROM t1, (SELECT /*+ <plan_hint3> */ * FROM t2) WHERE 1=1;
```

其中<plan_hint1>，<plan_hint2>为外层查询的hint，<plan_hint3>为内层子查询的hint。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

如果在视图定义（CREATE VIEW）时指定hint，则在该视图每次被应用时会使用该hint。

当使用random plan功能（参数plan_mode_seed不为0）时，查询指定的plan hint不会被使用。

支持范围

当前版本Plan Hint支持的范围如下，后续版本会进行增强。

- 指定Join顺序的Hint - leading hint。
- 指定Join方式的Hint，仅支持除semi/anti join，unique plan之外的常用hint。
- 指定结果集行数的Hint。
- 指定Stream方式的Hint。
- 指定Scan方式的Hint，仅支持常用的tablescan，indexscan和indexonlyscan的hint。
- 指定子链接块名的Hint。
- 指定倾斜信息的Hint，仅支持Join与HashAgg的重分布过程倾斜。
- 指定Agg重分布列Hint。仅8.1.3.100及以上集群版本支持。
- 指定子查询不提升的Hint。仅8.2.0及以上集群版本支持。
- 指定配置参数值的Hint，仅支持部分配置参数，详见**配置参数的hint**。

注意事项

- 不支持Sort、Setop和Subplan的hint。
- 不支持SMP和Node Group场景下的Hint。
- 不支持对INSERT语句的目标表使用Hint。

示例

本章节使用同一个语句进行示例，便于Plan Hint支持的各方法作对比，示例语句及不带hint的原计划如下所示：

```
explain
select i_product_name product_name
,i_item_sk item_sk
,s_store_name store_name
,s_zip store_zip
,ad2.ca_street_number c_street_number
,ad2.ca_street_name c_street_name
,ad2.ca_city c_city
,ad2.ca_zip c_zip
,count(*) cnt
,sum(ss_wholesale_cost) s1
,sum(ss_list_price) s2
,sum(ss_coupon_amt) s3
FROM   store_sales
,store_returns
,store
,customer
,promotion
,customer_address ad2
```


版权所有 © 华为云计算技术有限公司

```
,item
WHERE  ss_store_sk = s_store_sk AND
ss_customer_sk = c_customer_sk AND
ss_item_sk = i_item_sk and
ss_item_sk = sr_item_sk and
ss_ticket_number = sr_ticket_number and
c_current_addr_sk = ad2.ca_address_sk and
ss_promo_sk = p_promo_sk and
i_color in ('maroon','burnished','dim','steel','navajo','chocolate') and
i_current_price between 35 and 35 + 10 and
i_current_price between 35 + 1 and 35 + 15
group by i_product_name
,i_item_sk
,s_store_name
,s_zip
,ad2.ca_street_number
,ad2.ca_street_name
,ad2.ca_city
,ad2.ca_zip
;
```

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |  E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 3401632.49
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 3401632.49
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 3401630.82
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 3401630.78
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 16MB     |     169 | 3401630.42
  6 |                ->  Vector Hash Join (7,20)                               |          7 | 43MB     |     173 | 3400343.15
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)               |          7 | 1MB      |     123 | 3395775.64
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 3395775.48
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 3386294.72
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 3386294.56
 11 |                               ->  Vector Hash Join (12,14)               |          7 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator           |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns |  287999764 | 1MB      |      12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)            |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB      |      66 | 2756066.50
 17 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 18 |                                  ->  CStore Scan on store                |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                         ->  CStore Scan on customer_address ad2          |    6000000 | 1MB      |      58 | 5770.00
 21 |                      ->  CStore Scan on promotion                        |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```

14.4.9.2 Join 顺序的 Hint

功能描述

指明join的顺序，包括内外表顺序。

语法格式

- 单层圆括号()，仅指定join顺序，不指定内外表顺序。

```
leading(join_table_list)
leading(@block_name join_table_list)
```

- 双层圆括号(())，同时指定join顺序和内外表顺序，内外表顺序仅在最外层生效。

```
leading((join_table_list))
leading(@block_name (join_table_list))
```

- 单层方括号[]，同时指定[]这层的join顺序和内外表顺序。

```
leading[join_table_list]
leading[@block_name join_table_list]
```

- 单层圆括号()和单层方括号[]混合使用，同时指定join顺序和任意层内外表顺序。圆括号()这一层只指定join顺序，不指定内外表顺序，方括号[]这一层同时指定join顺序和内外表顺序。

```
leading(join_table_list1 [join_table_list2])
leading[join_table_list1 [join_table_list2]]
```


版权所有 © 华为云计算技术有限公司

```
leading[join_table_list1 (join_table_list2)]
leading(@block_name join_table_list1 [join_table_list2])
leading[@block_name join_table_list1 [join_table_list2]]
leading[@block_name join_table_list1 (join_table_list2)]
```

须知

单层方括号[]可以和单层圆括号()混合使用，指定任意层的内外表顺序。不支持单层[]和双层(())一起使用。

参数说明

- **join_table_list**

  为表示表join顺序的hint字符串，可以包含当前层的任意个表（别名），或对于子查询提升的场景，也可以包含子查询的hint别名，同时任意表可以使用括号指定优先级，表之间使用空格分隔。

  join table list中指定的表需要满足以下要求，否则会报语义错误：

  - list中的表必须在当前层或提升的子查询中存在。
  - list中的表在当前层或提升的子查询中必须是唯一的。如果不唯一，需要使用不同的别名进行区分。
  - 同一个表只能在list里出现一次。
  - 如果表存在别名，则list中的表必须使用别名。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

- **block_name**

  语句块的block_name。表示该hint在block_name对应的子查询语句块中生效。

 版权所有 © 华为云计算技术有限公司

须知

- 默认为语句生成block_name。
- CN轻量化不生成block_name。
- create table as select、select into、select、insert、update、delete、merge语句生成block_name。
- block_name的命名规则：
  - 为select、insert、update、delete、merge语句自动生成blockname，block_name的命名格式分别为sel$n、ins$n、upd$n、del$n、mer$n，n从1开始计数，不同类型语句之间，计数不累加，同一类型语句之间计数累加。

    例如：

    ```
    INSERT INTO t SELECT * FROM t1 WHERE a1 IN (select * from t2);
    ```

    --------sel$2-------

    ----------------------sel$1----------------------

    --------------------------------ins$1--------------------------
  - 在优化器之前递归为每个语句块分配block_name。

    首先为当前语句块根据语句类型分配block_name，然后按照下面的顺序进行遍历，并为其中的语句块分配block_name：

    1. 遍历目标列
    2. 遍历merge语句的源表中的目标列
    3. 遍历merge语句中的action（update/insert）
    4. 遍历returning子句
    5. 遍历from中的join条件和where条件（join条件优先于where条件）
    6. 如果是集合操作，遍历集合的各个分支（union/intersect/except）
    7. 遍历having子句
    8. 遍历limit offset子句
    9. 遍历limit count子句
    10. 遍历cte
    11. 遍历from后的表
    12. 遍历upsert子句
  - 在优化器的重写阶段，由于fulljoin、cte inline、物化视图重写、inlist2join、or转换、multi count(distinct)、magicset、lazyagg、子查询/子链接提升等重写优化，都会构造新的子查询，此时会对新构造的子查询也应用上面分配block_name的递归处理，block_name的编号计数在之前的基础上进行累加。
  - 优化器重写阶段，发生子查询提升时，内层子查询中的表被提升到外层查询中，内层子查询被消除。此时，被提升的表可能会和外层查询中的表同名，所以在表中记录其原本所属block_name来区分来自不同查询块中的两个相同表。

例如：

leading(t1 t2 t3 t4 t5)表示：t1, t2, t3, t4, t5先join，五表join顺序及内外表不限。

版权所有 © 华为云计算技术有限公司

leading((t1 t2 t3 t4 t5))表示：t1和t2先join，t2做内表；再和t3 join，t3做内表；再和t4 join，t4做内表；再和t5 join，t5做内表。

leading(t1 (t2 t3 t4) t5)表示：t2, t3, t4先join，内外表不限；再和t1, t5 join，内外表不限。

leading((t1 (t2 t3 t4) t5))表示：t2, t3, t4先join，内外表不限；在最外层，t1再和t2, t3, t4的join表join，t1为外表，再和t5 join，t5为内表。

leading((t1 (t2 t3) t4 t5)) leading((t3 t2))表示：t2, t3先join，t2做内表；然后再和t1 join，t2, t3的join表做内表；然后再依次跟t4, t5做join，t4, t5做内表。

leading[t1 [t2 t3]]等价于leading((t1 (t2 t3))) leading((t2 t3))。

leading(t1 [t2 t3])等价于leading(t1 t2 t3) leading((t2 t3))。

leading[@sel$1 t1@sel$1 [t2@sel$2 t3@sel$2]]表示：t2、t3位于子查询中，子查询被提升后，和t1进行join，其表示先t2和t3 join，t2为外表，t3为内表；然后再和t1 join，t1为外表，t2, t3的join表做内表。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ leading((((((store_sales store) promotion) item) customer) ad2) store_returns) leading((store store_sales))*/ i_product_name product_name ...
```

该hint表示：表之间的join关系是：store_sales和store先join， store_sales做内表，然后依次跟promotion, item, customer, ad2, store_returns做join。生成计划如下所示：

```
WARNING:  Duplicated or conflict hint: Leading(store_sales store), will be discarded.
 id |                                operation                                |   E-rows   | E-memory | E-width |   E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+-------------
  1 | ->  Row Adapter                                                         |          6 |          |     273 | 16308094.34
  2 |    ->  Vector Streaming (type: GATHER)                                  |          6 |          |     273 | 16308094.34
  3 |       ->  Vector Hash Aggregate                                         |          6 | 16MB     |     273 | 16308092.67
  4 |          ->  Vector Hash Join (5,20)                                    |          6 | 585MB    |     169 | 16308092.63
  5 |             ->  Vector Streaming(type: REDISTRIBUTE)                    |    1320811 | 1MB      |     181 | 16069870.93
  6 |                ->  Vector Hash Join (7,19)                              |    1320811 | 43MB     |     181 | 16061891.00
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |    1320811 | 1MB      |     131 | 16056566.78
  8 |                      ->  Vector Hash Join (9,18)                        |    1320811 | 27MB     |     131 | 16048586.85
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)        |    1383248 | 1MB      |     131 | 16038321.62
 10 |                            ->  Vector Hash Join (11,17)                 |    1383248 | 16MB     |     131 | 16029664.50
 11 |                               ->  Vector Hash Join (12,16)              | 2626366951 | 16MB     |      73 | 15751384.88
 12 |                                  ->  Vector Hash Join (13,14)           | 2750085660 | 2156MB   |      77 | 14226077.19
 13 |                                     ->  CStore Scan on store            |      24048 | 1MB      |      19 | 2264.00
 14 |                                     ->  Vector Partition Iterator       | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 16 |                                  ->  CStore Scan on promotion           |      36000 | 1MB      |       4 | 1268.50
 17 |                               ->  CStore Scan on item                   |        158 | 1MB      |      58 | 4051.25
 18 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 19 |                         ->  CStore Scan on customer_address ad2         |    6000000 | 1MB      |      58 | 5770.00
 20 |                   ->  Vector Partition Iterator                         |  287999764 | 1MB      |      12 | 227383.99
 21 |                      ->  Partitioned CStore Scan on store_returns       |  287999764 | 1MB      |      12 | 227383.99
(21 rows)
```

图中计划顶端warning的提示详见**Hint的错误、冲突及告警**的说明。

14.4.9.3 Join 方式的 Hint

功能描述

指明Join使用的方法，可以为Nested Loop，Hash Join和Merge Join。

语法格式

```
[no] nestloop|hashjoin|mergejoin([@block_name] table_list)
```

 版权所有 © 华为云计算技术有限公司

参数说明

- **no**表示hint的join方式不使用。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table_list**为表示hint表集合的字符串，该字符串中的表与**join_table_list**相同，只是中间不允许出现括号指定join的优先级。

例如：

no nestloop(t1 t2 t3)表示：生成t1，t2，t3三表连接计划时，不使用nestloop。三表连接计划可能是t2 t3先join，再跟t1 join，或t1 t2先join，再跟t3 join。此hint只hint最后一次join的join方式，对于两表连接的方法不hint。如果需要，可以单独指定，例如：任意表均不允许nestloop连接，且希望t2 t3先join，则增加hint：no nestloop(t2 t3)。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ nestloop(store_sales store_returns item) */ i_product_name product_name ...
```

该hint表示：生成store_sales，store_returns和item三表的结果集时，最后的两表关联使用nestloop。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061693161.06
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061693161.06
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061693159.40
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061693159.36
  5 |             ->  Vector Hash Join (6,22)                                  |          6 | 43MB     |     169 | 100061693158.99
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061688591.48
  7 |                   ->  Vector Hash Join (8,21)                            |          6 | 16MB     |     119 | 100061688591.30
  8 |                      ->  Vector Hash Join (9,20)                         |          7 | 27MB     |     123 | 100061687304.04
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061677823.27
 10 |                            ->  Vector Hash Join (11,19)                  |          7 | 16MB     |     123 | 100061677823.12
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061675546.57
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB | 12 | 227383.99
 17 |                                  ->  Vector Materialize                  |        158 | 16MB     |      58 | 4051.28
 18 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 19 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 20 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 21 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 22 |                      ->  CStore Scan on customer_address ad2             |    6000000 | 1MB      |      58 | 5770.00
(22 rows)
```

14.4.9.4 行数的 Hint

功能描述

指明中间结果集的大小，支持绝对值和相对值的hint。

语法格式

```
rows([@block_name] table_list #|+|-|* const)
```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **#,+,-,***，进行行数估算hint的四种操作符号。#表示直接使用后面的行数进行hint。+,-,*表示对原来估算的行数进行加、减、乘操作，运算后的行数最小值为1

版权所有 © 华为云计算技术有限公司

行。table_list为hint对应的单表或多表join结果集，与**Join方式的Hint**中table_list相同。

- **const**可以是任意非负数，支持科学计数法。

例如：

rows(t1 #5)表示：指定t1表的结果集为5行。

rows(t1 t2 t3 *1000)表示：指定t1, t2, t3 join完的结果集的行数乘以1000。

建议

- 推荐使用两个表*的hint。对于两个表的采用*操作符的hint，只要两个表出现在join的两端，都会触发hint。例如：设置hint为rows(t1 t2 * 3)，对于(t1 t3 t4)和(t2 t5 t6)join时，由于t1和t2出现在join的两端，所以其join的结果集也会应用该hint规则乘以3。
- rows hint支持在单表、多表、function table及subquery scan table的结果集上指定hint。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ rows(store_sales store_returns *50) */ i_product_name product_name ...
```

该hint表示：store_sales，store_returns关联的结果集估算行数在原估算行数基础上乘以50。生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |        312 |          |     273 | 3401656.58
  2 |    ->  Vector Streaming (type: GATHER)                                  |        312 |          |     273 | 3401656.58
  3 |       ->  Vector Hash Aggregate                                         |        312 | 16MB     |     273 | 3401634.91
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |        313 | 1MB      |     169 | 3401634.39
  5 |             ->  Vector Hash Join (6,21)                                 |        313 | 43MB     |     169 | 3401633.06
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                 |        313 | 1MB      |     119 | 3397065.38
  7 |                   ->  Vector Hash Join (8,20)                           |        313 | 27MB     |     119 | 3397064.31
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)           |        328 | 1MB      |     119 | 3387583.37
  9 |                         ->  Vector Hash Join (10,19)                    |        328 | 16MB     |     119 | 3387582.18
 10 |                            ->  Vector Hash Join (11,18)                 |        344 | 16MB     |     123 | 3386294.74
 11 |                               ->  Vector Hash Join (12,14)              |        360 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator          |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |  12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)           |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator       | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 17 |                                     ->  CStore Scan on item             |        158 | 1MB      |      58 | 4051.25
 18 |                                  ->  CStore Scan on store               |      24048 | 1MB      |      19 | 2264.00
 19 |                               ->  CStore Scan on promotion              |      36000 | 1MB      |       4 | 1268.50
 20 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 21 |                      ->  CStore Scan on customer_address ad2            |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

第11行算子的估算行数修正为360行，原估算行数为7行（四舍五入后取值）。

14.4.9.5 Stream 方式的 Hint

功能描述

指明stream使用的方法，可以为broadcast和redistribute以及指定AGG重分布的分布键。

说明

指定Agg重分布列Hint，仅8.1.3.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

语法格式

```
[no] broadcast | redistribute([@block_name] table_list) | redistribute ([@block_name] (*) (columns))
```

参数说明

- **no**表示hint的stream方式不使用，当指定AGG分布列的hint时指定no关键字无效。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table_list**为进行stream操作的单表或多表join结果集，见**参数说明**。
- 当指定分布列的hint时，*为固定写法，不支持指定表名。
- **columns**指定group by子句中的一个或者多个列，没有group by子句也可以指定distinct子句中的列。

  说明

  - 指定的分布列，需要用group by或distinct中的列序号或列名来表示，count（distinct）中的列只能通过列名指定。
  - 对于多层的查询，可以在每层指定对应层的分布列hint，只在当前层生效。
  - 指定的count(distinct)列仅针对生成双层hashagg的计划时才生效，否则指定的分布列无效。
  - 指定了分布列，如果优化器估算后发现不需要重分布，则指定的分布列无效。

建议

- 通常优化器会根据统计信息选择一组不倾斜的分布键进行数据重分布。当默认选择的分布键有倾斜时，可以手动指定重分布的列，避免数据倾斜。
- 在选择分布键的时候，通常要根据数据分布特征选取一组distinct值比较高的列作为分布列，这样可以保证重分布后，数据均匀的分布到各个DN。
- 在编写好hint后，可以通过explain verbose+SQL打印执行计划，查看指定的分布键是否有效，如果指定的分布键无效会有warning提示。

示例

- 对**示例**中原语句使用如下hint：

```
explain
select /*+ no redistribute(store_sales store_returns item store) leading(((store_sales store_returns item store) customer)) */ i_product_name product_name ...
```

原计划中，(store_sales store_returns item store)和customer做join时，前者做了重分布，此hint表示禁止前者混合表做重分布，但仍然保持join顺序，则生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |          6 |          |     273 | 5718448.94
  2 |    ->  Vector Streaming (type: GATHER)                                  |          6 |          |     273 | 5718448.94
  3 |       ->  Vector Hash Aggregate                                         |          6 | 16MB     |     273 | 5718447.27
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |          6 | 1MB      |     169 | 5718447.23
  5 |             ->  Vector Hash Join (6,21)                                 |          6 | 16MB     |     169 | 5718446.86
  6 |                ->  Vector Hash Join (7,20)                              |          7 | 43MB     |     173 | 5717159.60
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |          7 | 1MB      |     123 | 5712592.09
  8 |                      ->  Vector Hash Join (9,18)                        |          7 | 585MB    |     123 | 5712591.93
  9 |                         ->  Vector Hash Join (10,17)                    |          7 | 16MB     |     123 | 3386294.56
 10 |                            ->  Vector Hash Join (11,13)                 |          7 | 19MB     |     112 | 3384018.02
 11 |                               ->  Vector Partition Iterator             |  287999764 | 1MB      |      12 | 227383.99
 12 |                                  ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      12 | 227383.99
 13 |                               ->  Vector Hash Join (14,16)              |    1516824 | 16MB     |     124 | 3065686.08
 14 |                                  ->  Vector Partition Iterator          | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                     ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 16 |                                  ->  CStore Scan on item                |        158 | 1MB      |      58 | 4051.25
 17 |                            ->  CStore Scan on store                     |      24048 | 1MB      |      19 | 2264.00
 18 |                         ->  Vector Streaming(type: BROADCAST)           |  288000000 | 1MB      |       8 | 2176297.36
 19 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 20 |                      ->  CStore Scan on customer_address ad2            |    6000000 | 1MB      |      58 | 5770.00
 21 |                   ->  CStore Scan on promotion                          |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```


版权所有 © 华为云计算技术有限公司

- 指定Agg重分布的分布列。

```
explain (verbose on, costs off, nodes off)
select /*+ redistribute ((*) (2 3)) */ a1, b1, c1, count(c1)  from t1 group by a1, b1, c1 having
count(c1) > 10 and sum(d1) > 100
```

通过下面的示例可以看到指定的group by列的后两列作为分布键。

```
                           QUERY PLAN
-----------------------------------------------------------------
  id |                operation
 ----+-----------------------------------------
   1 | ->  Streaming (type: GATHER)
   2 |     ->  HashAggregate
   3 |        ->  Streaming(type: REDISTRIBUTE)
   4 |           ->  Seq Scan on public.t1

         Predicate Information (identified by plan id)
 ----------------------------------------------------------------
   2 --HashAggregate
         Filter: ((count(t1.c1) > 10) AND (sum(t1.d1) > 100))

 Targetlist Information (identified by plan id)
 ------------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a1, b1, c1, (count(c1))
   2 --HashAggregate
         Output: a1, b1, c1, count(c1)
         Group By Key: t1.a1, t1.b1, t1.c1
   3 --Streaming(type: REDISTRIBUTE)
         Output: a1, b1, c1, d1
         Distribute Key: b1, c1
   4 --Seq Scan on public.t1
         Output: a1, b1, c1, d1

    ====== Query Summary =====
 ---------------------------------
 System available mem: 24862720KB
 Query Max mem: 24862720KB
 Query estimated mem: 3138KB
(30 rows)
```

- 当语句中不包含group by子句时，指定distinct列作为重分布列。

```
explain (verbose on, costs off, nodes off)
select /*+ redistribute ((*) (3 1)) */ distinct a1, b1, c1 from t1;
```

 版权所有 © 华为云计算技术有限公司

```
                    QUERY PLAN
-------------------------------------------------
 id |                 operation
----+-----------------------------------------
  1 | ->  Streaming (type: GATHER)
  2 |     ->  HashAggregate
  3 |        ->  Streaming(type: REDISTRIBUTE)
  4 |           ->  Seq Scan on public.t1

Targetlist Information (identified by plan id)
-------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a1, b1, c1
  2 --HashAggregate
        Output: a1, b1, c1
        Group By Key: t1.a1, t1.b1, t1.c1
  3 --Streaming(type: REDISTRIBUTE)
        Output: a1, b1, c1
        Distribute Key: c1, a1
  4 --Seq Scan on public.t1
        Output: a1, b1, c1

   ====== Query Summary =====
---------------------------------
System available mem: 24862720KB
Query Max mem: 24862720KB
Query estimated mem: 3136KB
(25 rows)
```

14.4.9.6 Scan 方式的 Hint

功能描述

指明scan使用的方法，可以是tablescan、indexscan和indexonlyscan。

语法格式

```
[no] tablescan|indexscan|indexonlyscan([@block_name] table [index])
```

参数说明

- **no**表示hint的scan方式不使用。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示hint指定的表，只能指定一个表，如果表存在别名应优先使用别名进行hint。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

- **index**表示使用indexscan或indexonlyscan的hint时，指定的索引名称，当前只能指定一个。

 版权所有 © 华为云计算技术有限公司

说明

对于indexscan或indexonlyscan，只有hint的索引属于hint的表时，才能使用该hint。

scan hint支持在行列存表、hdfs内外表、obs表、子查询表上指定。对于hdfs内表，由主表和delta表组成，delta表对用户不可见，故hint仅作用在主表上。

指定indexscan可生效indexscan或indexonlyscan，indexscan或indexonlyscan也可同时出现。当indexscan和indexonlyscan hint同时出现，优先生效indexonlyscan。

示例

为了hint使用索引扫描，需要首先在表item的i_item_sk列上创建索引，名称为i：

```
create index i on item(i_item_sk);
```

对示例中原语句使用如下hint：

```
explain
select /*+ indexscan(item i) */ i_product_name product_name ...
```

该hint表示：item表使用索引i进行扫描。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061674938.26
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061674938.26
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061674936.59
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061674936.55
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 43MB     |     169 | 100061674936.19
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061670368.67
  7 |                   ->  Vector Hash Join (8,20)                            |          6 | 16MB     |     119 | 100061670368.50
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 100061669081.23
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061659600.47
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 100061659600.31
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061657323.77
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB | 66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB | 12 | 227383.99
 17 |                                  ->  CStore Index Scan using i on item   |          1 | 1MB      |      58 | 4.01
 18 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 21 |                   ->  CStore Scan on customer_address ad2                |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

14.4.9.7 子链接块名的 hint

功能描述

指明子链接块的名称。

语法格式

```
blockname ([@block_name] table)
```

注意事项

- block_name hint仅在对应的子链接块提升时才会被上层查询使用。目前支持的子链接提升包括IN子链接提升、EXISTS子链接提升和包含Agg等值相关子链接提升。该hint通常会和前面章节提到的hint联合使用。
- 对于FROM关键字后的子查询，则需要使用子查询的别名进行hint，block_name hint不会被用到。
- 如果子链接中含有多个表，则提升后这些表可与外层表以任意优化顺序连接，hint也不会被用到。

版权所有 © 华为云计算技术有限公司

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示为该子链接块hint的别名的名称。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。

示例

```
explain select /*+nestloop(store_sales tt) */ * from store_sales where ss_item_sk in (select /*
+blockname(tt)*/ i_item_sk from item group by 1);
```

该hint表示：子链接的别名为tt，提升后与上层的store_sales表关联时使用nestloop。生成计划如下所示：

```
 id |                       operation                       |   E-rows   | E-memory | E-width |     E-costs
----+-------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                       | 1439994000 |          |     216 | 325105765847.91
  2 |    ->  Vector Streaming (type: GATHER)                | 1439994000 |          |     216 | 325105765847.91
  3 |       ->  Vector Nest Loop Semi Join (4, 6)           | 1439994000 | 1MB      |     216 | 325026664615.00
  4 |          ->  Vector Partition Iterator                | 2879987999 | 1MB      |     216 | 2756066.50
  5 |             ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB      |     216 | 2756066.50
  6 |          ->  Vector Materialize                       |     300000 | 16MB     |       4 | 4176.25
  7 |             ->  Vector Hash Aggregate                 |     300000 | 16MB     |       4 | 3988.75
  8 |                ->  CStore Scan on item                |     300000 | 1MB      |       4 | 3832.50
(8 rows)
```

14.4.9.8 运行倾斜的 hint

功能描述

指明查询运行时重分布过程中存在倾斜的重分布键和倾斜值，针对Join和HashAgg运算中的重分布进行优化。

注意事项

- skew hint仅在需要重分布且指定的倾斜信息与查询执行过程中的重分布信息相匹配时才会被使用。
- skew hint受GUC参数**skew_option**限制，如果参数处于关闭状态，则无法进行skew hint倾斜调优。
- skew hint目前仅处理普通表和子查询类型的表关系，支持基表hint、子查询hint、with as子句hint。对于子查询，无论提升与否都支持在skew hint中使用，这点与其他hint不一样。
- 对于倾斜表，如果定义了别名，则在hint中必须使用别名。
- 对于倾斜列，在不产生歧义的情况下，可以使用原名也可以使用别名。skew hint的column不支持表达式，如果需要指定采用分布键为表达式的重分布存在倾斜，需要将重分布键指定为新的列，以新的列进行hint。
- 对于倾斜值，个数需为列数的整数倍并按列的顺序进行组合，组合的个数不能超过10个。如果各倾斜列的倾斜值的个数不一样，为了满足按列组合，值可以重复指定。如，表t1的c1和c2存在倾斜，c1列的倾斜值只有a1，而c2列的倾斜有b1和

版权所有 © 华为云计算技术有限公司

b2，则skew hint如下：skew(t1 (c1 c2) ((a1 b1)(a1 b2)))。例中(a1 b1)为一个值组合，NULL可以作为倾斜值出现，每个hint中的值组合不超过十个， 且需为列的整数倍。

- 在Join的重分布优化中，skew hint中的value不可缺省，在HashAgg中可以缺省。
- 对于表、列、值中若指定多个，则同类间需以空格分离。
- 对于倾斜值，不支持在hint中进行类型强转；对于string类型，需要使用单引号。

语法格式

- 指定单表倾斜：
  ```
  skew([@block_name] table (column) [(value)])
  ```
- 指定中间结果倾斜：
  ```
  skew([@block_name] (join_rel) (column) [(value)])
  ```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **table**表示存在倾斜的单个表名。

  说明

  表的语法格式如下：

  ```
  [schema.]table[@block_name]
  ```

  表名可以带schema，也可以带所在子查询语句块提升前的block_name。

  子查询语句块在优化器进行优化重写时发生提升，则该block_name会与leading中block_name不同。
- **join_rel**表示参与join的两个或多个表，如（t1 t2）表示t1和t2 join后的结果存在倾斜。
- **column**表示倾斜表中存在倾斜的一个或多个列。
- **value**表示倾斜的列中存在倾斜的一个或多个值。

例如：

- 指定单表倾斜

  每一个skew hint用来表示一个表关系存在的倾斜信息，如果想要指定在查询中的多个表关系存在的倾斜信息，则通过指定多个skew hint实现。

  在指定skew时，包括以下四个场景的用法：

  - 单列单值： skew(t (c1) (v1))

    说明：表关系t的c1列中的v1值在查询执行中存在倾斜。
  - 单列多值：skew(t (c1) (v1 v2 v3 ...))

    说明：表关系t的c1列中的v1、v2、v3…等值在查询执行中存在倾斜。
  - 多列单值：skew(t (c1 c2) (v1 v2))

    说明：表关系t的c1列的v1值和c2列的v2值在查询执行中存在倾斜。
  - 多列多值：skew(t (c1 c2) ((v1 v2) (v3 v4) (v5 v6) ...))

    说明：表关系t的c1列的v1、v3、v5…值和c2列的v2、v4、v6…值在查询执行中存在倾斜。

 版权所有 © 华为云计算技术有限公司

须知

多列多值时，各组倾斜值间也可以不使用括号，如：skew(t (c1 c2) (v1 v2 v3 v4 v5 v6 ...))。是否使用括号必须统一，不可混合，

如：skew(t (c1 c2) (v1 v2 v3 v4 （ v5 v6 ） ...)) 将会产生语法报错。

- 指定中间结果倾斜

  如果基表不存在倾斜，而是查询执行中的中间结果出现倾斜，则需要通过指定中间结果倾斜的skew hint来进行倾斜的调优。skew((t1 t2) (c1) (v1))

  说明：表关系t1和t2 Join后的结果存在倾斜，倾斜的是t1表的c1列，c1列的倾斜值是v1。

  为了避免产生歧义，“c1”只能存在于join_rel的一个表关系中，如果存在同名列则通过别名进行规避。

建议

- 如果查询具有多层，则哪一层出现倾斜，则将hint写在哪一层中。
- 对于提升的子查询，skew hint支持直接使用子查询名进行hint。如果明确子查询提升后的哪一个基表存在倾斜，则直接使用基表进行hint的可用性更高。
- 无论对于表或列，若存在别名，则优先使用别名进行hint。

示例

指定单表倾斜

- 原query中进行hint。

  采用如下查询进行skew hint倾斜调优的举例，查询语句及不带hint的原计划如下所示：

```
explain
with customer_total_return as
(select sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```


版权所有 © 华为云计算技术有限公司

```
 id |                                operation                                |  E-rows   |   E-memory    | E-width |  E-costs
----+-------------------------------------------------------------------------+-----------+---------------+---------+-----------
  1 | ->  Row Adapter                                                         |       100 |               |      20 | 911254.47
  2 |    ->  Vector Limit                                                     |       100 |               |      20 | 911254.47
  3 |       ->  Vector Streaming (type: GATHER)                               |      2400 |               |      20 | 911325.75
  4 |          ->  Vector Limit                                               |      2400 | 1MB           |      20 | 911247.62
  5 |             ->  Vector Sort                                             |   3684816 | 16MB          |      20 | 911631.21
  6 |                ->  Vector Hash Join (7,29)                              |   3684817 | 41MB(12374MB) |      20 | 905379.41
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |   3684817 | 384KB         |       4 | 883010.31
  8 |                      ->  Vector Hash Join (9,19)                        |   3684817 | 16MB          |       4 | 861302.05
  9 |                         ->  Vector Hash Join (10,18)                    |  11054450 | 16MB          |      44 | 427109.71
 10 |                            ->  Vector Hash Aggregate                    |  50247501 | 397MB(12671MB) |      54 | 395302.57
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)  |  50247501 | 384KB         |      22 | 358663.76
 12 |                                  ->  Vector Hash Join (13,15)           |  50247501 | 16MB          |      22 | 294300.51
 13 |                                     ->  Vector Partition Iterator       | 287999764 | 1MB           |      26 | 227383.99
 14 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      26 | 227383.99
 15 |                                     ->  Vector Streaming(type: BROADCAST) |      8712 | 384KB         |       4 | 975.56
 16 |                                        ->  Vector Partition Iterator    |       363 | 1MB           |       4 | 910.65
 17 |                                           ->  Partitioned CStore Scan on date_dim | 363 | 1MB |       4 | 910.65
 18 |                               ->  CStore Scan on store                  |        44 | 1MB           |       4 | 1006.39
 19 |                            ->  Vector Hash Aggregate                    |       192 | 16MB          |      68 | 426707.38
 20 |                               ->  Vector Subquery Scan on ctr2          |  50247501 | 1MB           |      36 | 416239.03
 21 |                                  ->  Vector Hash Aggregate              |  50247501 | 397MB(12671MB) |      54 | 395302.57
 22 |                                     ->  Vector Streaming(type: REDISTRIBUTE) |  50247501 | 384KB |      22 | 358663.76
 23 |                                        ->  Vector Hash Join (24,26)     |  50247501 | 16MB          |      22 | 294300.51
 24 |                                           ->  Vector Partition Iterator | 287999764 | 1MB           |      26 | 227383.99
 25 |                                              ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      26 | 227383.99
 26 |                                           ->  Vector Streaming(type: BROADCAST) |      8712 | 384KB |       4 | 975.56
 27 |                                              ->  Vector Partition Iterator |       363 | 1MB           |       4 | 910.65
 28 |                                                 ->  Partitioned CStore Scan on date_dim |  363 | 1MB |       4 | 910.65
 29 |                   ->  CStore Scan on customer                           |  12000000 | 1MB           |      24 | 12923.00
(29 rows)
```

对内层with子句中的HashAgg和外层的Hash Join进行hint指定，带hint的查询如下：

```
explain
with customer_total_return as
(select /*+ skew(store_returns(sr_store_sk sr_customer_sk)) */sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  /*+ skew(ctr1(ctr_customer_sk)(11))*/  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```

该hint表示：内层with子句中的group by在做HashAgg中进行重分布时存在倾斜，对应原计划的10和21号Hash Agg算子；外层ctr1表的ctr_customer_sk列在做Hash Join中进行重分布时存在倾斜，对应原计划的6号算子。生成计划如下所示：

```
 id |                                operation                                |  E-rows   |   E-memory    | E-width |  E-costs
----+-------------------------------------------------------------------------+-----------+---------------+---------+-----------
  1 | ->  Row Adapter                                                         |       100 |               |      20 | 1061778.14
  2 |    ->  Vector Limit                                                     |       100 |               |      20 | 1061778.14
  3 |       ->  Vector Streaming (type: GATHER)                               |      2400 |               |      20 | 1061849.41
  4 |          ->  Vector Limit                                               |      2400 | 1MB           |      20 | 1061771.29
  5 |             ->  Vector Sort                                             |   3684816 | 16MB          |      20 | 1062154.87
  6 |                ->  Vector Hash Join (7,31)                              |   3684817 | 41MB(12344MB) |      20 | 1055903.08
  7 |                   ->  Vector Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN) |   3684817 | 384KB |       4 | 1013056.49
  8 |                      ->  Vector Hash Join (9,20)                        |   3684817 | 16MB          |       4 | 1000006.10
  9 |                         ->  Vector Hash Join (10,19)                    |  11054450 | 16MB          |      44 | 496461.73
 10 |                            ->  Vector Hash Aggregate                    |  50247501 | 397MB(12010MB) |      54 | 464654.59
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)  |  50247501 | 384KB         |      54 | 428015.79
 12 |                                  ->  Vector Hash Aggregate              |  50247501 | 397MB(12010MB) |      54 | 330939.31
 13 |                                     ->  Vector Hash Join (14,16)        |  50247501 | 16MB          |      22 | 294300.51
 14 |                                        ->  Vector Partition Iterator    | 287999764 | 1MB           |      26 | 227383.99
 15 |                                           ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      26 | 227383.99
 16 |                                        ->  Vector Streaming(type: BROADCAST) |      8712 | 384KB |       4 | 975.56
 17 |                                           ->  Vector Partition Iterator |       363 | 1MB           |       4 | 910.65
 18 |                                              ->  Partitioned CStore Scan on date_dim |  363 | 1MB |       4 | 910.65
 19 |                               ->  CStore Scan on store                  |        44 | 1MB           |       4 | 1006.39
 20 |                            ->  Vector Hash Aggregate                    |       192 | 16MB          |      68 | 496059.40
 21 |                               ->  Vector Subquery Scan on ctr2          |  50247501 | 1MB           |      36 | 485591.05
 22 |                                  ->  Vector Hash Aggregate              |  50247501 | 397MB(12010MB) |      54 | 464654.59
 23 |                                     ->  Vector Streaming(type: REDISTRIBUTE) |  50247501 | 384KB |      54 | 428015.79
 24 |                                        ->  Vector Hash Aggregate        |  50247501 | 397MB(12010MB) |      54 | 330939.31
 25 |                                           ->  Vector Hash Join (26,28)  |  50247501 | 16MB          |      22 | 294300.51
 26 |                                              ->  Vector Partition Iterator | 287999764 | 1MB |      26 | 227383.99
 27 |                                                 ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      26 | 227383.99
 28 |                                              ->  Vector Streaming(type: BROADCAST) |      8712 | 384KB |       4 | 975.56
 29 |                                                 ->  Vector Partition Iterator |       363 | 1MB           |       4 | 910.65
 30 |                                                    ->  Partitioned CStore Scan on date_dim |  363 | 1MB |       4 | 910.65
 31 |                   ->  Vector Streaming(type: PART LOCAL PART BROADCAST)  | 12000000 | 384KB         |      24 | 34485.50
 32 |                      ->  CStore Scan on customer                        |  12000000 | 1MB           |      24 | 12923.00
(32 rows)
```


版权所有 © 华为云计算技术有限公司

从优化后的计划可以看出：①对于Hash Agg，由于其重分布存在倾斜，所以优化为双层Agg；②对于Hash Join，同样由于其重分布存在倾斜，所以优化为采用新的重分布算子。

- 需要改写query后进行hint

  不带hint的查询和计划如下：

```
explain select count(*) from store_sales_1 group by round(ss_list_price);
```

```
 1 | ->  Row Adapter                                                    |   16672 |          |      14 | 62261.28
 2 |    ->  Vector Streaming (type: GATHER)                             |   16672 |          |      14 | 62261.28
 3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)            |   16672 | 32KB     |      14 | 61479.78
 4 |          ->  Vector Hash Aggregate                                 |   16672 | 16MB     |      14 | 61452.00
 5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) | 3112836 | 128KB    |       6 | 57498.43
 6 |                ->  CStore Scan on store_sales_1                    | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

  由于hint中列不支持表达式，在进行倾斜优化时需要借助subquery改写查询，改写后的查询和计划如下：

```
explain
select count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
 1 | ->  Row Adapter                                                    |   16672 |          |      14 | 62261.28
 2 |    ->  Vector Streaming (type: GATHER)                             |   16672 |          |      14 | 62261.28
 3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)            |   16672 | 32KB     |      14 | 61479.78
 4 |          ->  Vector Hash Aggregate                                 |   16672 | 16MB     |      14 | 61452.00
 5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) | 3112836 | 128KB    |       6 | 57498.43
 6 |                ->  CStore Scan on store_sales_1                    | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

  改写注意不要影响到业务逻辑。

  采用改写后的查询进行hint，带hint的查询和计划如下：

```
explain
select /*+ skew(tmp(a)) */ count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
 id |                           operation                            | E-rows  | E-memory | E-width | E-costs
----+----------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                                |   16672 |          |      14 | 27771.82
  2 |    ->  Vector Streaming (type: GATHER)                         |   16672 |          |      14 | 27771.82
  3 |       ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)        |   16672 | 32KB     |      14 | 26990.32
  4 |          ->  Vector Hash Aggregate                             |   16671 | 16MB     |      14 | 26962.54
  5 |             ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) |   66216 | 128KB    |      14 | 26838.09
  6 |                ->  Vector Hash Aggregate                       |   66216 | 16MB     |      14 | 25949.61
  7 |                   ->  CStore Scan on store_sales_1             | 3112836 | 1MB      |       6 | 21810.25
(7 rows)
```

  从计划可以看出，对Hash Agg进行倾斜优化后，采用了双层agg实现，大大过滤了进行重分布时的数据量，减少了重分布时间。

  此外，需要说明的是，对于子查询，支持使用查询内部的列进行hint，如：

```
explain
select /*+ skew(tmp(b)) */ count(*)
from (select round(ss_list_price) b,ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

14.4.9.9 指定子查询不提升的 hint

功能描述

优化器在对查询进行逻辑优化时通常会将可以提升的子查询提升到上层以避免嵌套执行，但对于某些场景，嵌套执行不会导致性能下降过多，而提升之后扩大了查询路径的搜索范围，可能导致性能变差。对于此类情况，可以使用no merge hint指定子查询不提升进行调试。大多数情况下不建议使用此hint。

版权所有 © 华为云计算技术有限公司

语法格式

```
no merge[@block_name]
no merge ([@block_name1] subquery_name[@block_name2])
```

参数说明

- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **subquery_name**为目标子查询名，亦可以是view或cte名，表示该子查询在逻辑优化时不会进行提升；当不指定subquery_name时，表示当前查询不提升。

示例

创建表t1、t2、t3：

```
create table t1(a1 int,b1 int,c1 int,d1 int);
create table t2(a2 int,b2 int,c2 int,d2 int);
create table t3(a3 int,b3 int,c3 int,d3 int);
```

原语句为：

```
explain select * from t3, (select a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
```

```
 id |           operation           | E-rows | E-width | E-costs
----+-------------------------------+--------+---------+---------
  1 | ->  Hash Join (2,6)           |  44450 |      32 | 754.31
  2 |    ->  Hash Join (3,4)        |   8885 |      28 | 182.11
  3 |       ->  Seq Scan on t3      |   1776 |      16 | 27.76
  4 |       ->  Hash                |   1776 |      12 | 27.76
  5 |          ->  Seq Scan on t2   |   1776 |      12 | 27.76
  6 |    ->  Hash                   |   1776 |       8 | 27.76
  7 |       ->  Seq Scan on t1      |   1776 |       8 | 27.76
```

上述查询中，可以使用以下两种方式禁止子查询s1进行提升：

- 方式一：
  ```
  explain select /*+ no merge(s1) */ * from t3, (select a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
  ```
- 方式二：
  ```
  explain select * from t3, (select /*+ no merge */ a1,b2,c1,d2 from t1,t2 where t1.a1=t2.a2) s1 where t3.b3=s1.b2;
  ```

提升后效果：

```
 id |           operation           | E-rows | E-width | E-costs
----+-------------------------------+--------+---------+---------
  1 | ->  Hash Join (2,6)           |   8880 |      32 | 443.03
  2 |    ->  Hash Join (3,4)        |   8885 |      16 | 182.11
  3 |       ->  Seq Scan on t1      |   1776 |       8 | 27.76
  4 |       ->  Hash                |   1776 |      12 | 27.76
  5 |          ->  Seq Scan on t2   |   1776 |      12 | 27.76
  6 |    ->  Hash                   |   1776 |      16 | 27.76
  7 |       ->  Seq Scan on t3      |   1776 |      16 | 27.76
```


版权所有 © 华为云计算技术有限公司

14.4.9.10 Drive 方式的 Hint

功能描述

优化器在进行查询计划生成时，会使用动态规划或遗传基因算法枚举可能的Join路径，并在其中选择最优的路径，当连接表的数目增大时，会导致搜索空间的过度膨胀。对于此类情况，可以通过Drive Hint指定查询中的事实表，使用启发式搜索的方式减少路径搜索空间的范围。该hint只有当GUC参数**join_search_mode**设置为heuristic时生效。

语法格式

```
[no] drive (table)
```

参数说明

- **no**表示hint对应的表为非Drive表。
- **table**表示hint指定的表，只能指定一个表，如果表存在别名应优先使用别名进行hint。

示例

创建表t1、t2、t3：

```
create table t1(a1 int,b1 int,c1 int,d1 int);
create table t2(a2 int,b2 int,c2 int,d2 int);
create table t3(a3 int,b3 int,c3 int,d3 int);
```

原语句为：

```
explain select * from t1,t2,t3 where t1.b1=t2.b2 and t1.c1=t3.c3 and t2.d2=t3.d3;
```

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      4 |          |      48 | 40.03
  2 |    ->  Hash Join (3,9)                    |      4 | 1MB      |      48 | 34.03
  3 |       ->  Streaming(type: BROADCAST)      |     40 | 2MB      |      32 | 23.64
  4 |          ->  Hash Join (5,7)              |     20 | 1MB      |      32 | 22.06
  5 |             ->  Streaming(type: BROADCAST) |     40 | 2MB      |      16 | 11.67
  6 |                ->  Seq Scan on t1         |     20 | 1MB      |      16 | 10.10
  7 |             ->  Hash                      |     20 | 16MB     |      16 | 10.10
  8 |                ->  Seq Scan on t2         |     20 | 1MB      |      16 | 10.10
  9 |       ->  Hash                            |     20 | 16MB     |      16 | 10.10
 10 |          ->  Seq Scan on t3               |     20 | 1MB      |      16 | 10.10
```

上述查询中，可以通过指定t3表为drive表，使得t3表优先和其他表join，减少计划生成路径搜索，同时更改join顺序。：

```
explain select /*+ drive(t3) */ * from t1, t2, t3 where t1.b1=t2.b2 and t1.c1=t3.c3 and t2.d2=t3.d3;
```

使用hint后的效果：

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-memory | E-width | E-costs
----+-------------------------------------------+--------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      4 |          |      48 | 40.03
  2 |    ->  Hash Join (3,9)                    |      4 | 1MB      |      48 | 34.03
  3 |       ->  Streaming(type: BROADCAST)      |     40 | 2MB      |      32 | 23.64
  4 |          ->  Hash Join (5,7)              |     20 | 1MB      |      32 | 22.06
  5 |             ->  Streaming(type: BROADCAST) |     40 | 2MB      |      16 | 11.67
  6 |                ->  Seq Scan on t3(DRIVE)  |     20 | 1MB      |      16 | 10.10
  7 |             ->  Hash                      |     20 | 16MB     |      16 | 10.10
  8 |                ->  Seq Scan on t1         |     20 | 1MB      |      16 | 10.10
  9 |       ->  Hash                            |     20 | 16MB     |      16 | 10.10
 10 |          ->  Seq Scan on t2               |     20 | 1MB      |      16 | 10.10
```


版权所有 © 华为云计算技术有限公司

> **⚠ 注意**
>
> 当GUC参数**join_search_mode**取值为heuristic且连接表个数超过from_collapse_limit时，优化器会自动识别Drive表，并在计划中显示。如果识别优化器最终选择的Drive不对，可以使用no drive hint进行纠偏。

14.4.9.11 字典编码的 hint

功能描述

通过设置指定列来构建字典编码，将字典编码的字符串的比较转化为数字的比较，加快group by、filter等查询速度。该hint仅8.3.0及以上集群版本支持。

注意事项

目前只支持新版的hstore表（表级参数为enable_hstore_opt为on）。

语法格式

```
/* + ( no ) dict(table (column)) */
```

参数说明

- dict(table (column))
  使用字典编码表的某列。
- no dict(table (column))
  禁用字典编码表的某列。

示例

```
SELECT /*+ dict (bitmaptbl_high (server_ip)) */ distinct(server_ip) FROM bitmaptbl_high WHERE scope_name='saetataetaeta' ORDER BY server_ip;
```

生成的计划如下，server_ip使用了字典编码：

版权所有 © 华为云计算技术有限公司

```
postgres=# explain (verbose on, costs off, nodes off)
select distinct(server_ip) from bitmaptbl_high where scope_name='saetataetaeta' order by server_ip;
                                  QUERY PLAN
------------------------------------------------------------------------------------
 id |                       operation
----+-----------------------------------------------------
  1 | ->  Row Adapter
  2 |    ->  Vector Sort
  3 |       ->  Vector Sonic Hash Aggregate
  4 |          ->  Vector Streaming (type: GATHER)
  5 |             ->  Vector Sonic Hash Aggregate
  6 |                ->  CStore Scan on public.bitmaptbl_high

                  Predicate Information (identified by plan id)
------------------------------------------------------------------------------------
  6 --CStore Scan on public.bitmaptbl_high
       CU Predicate Filter: (bitmaptbl_high.scope_name = 'saetataetaeta'::text)
       Pushdown Predicate Filter: (bitmaptbl_high.scope_name = 'saetataetaeta'::text)

 Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Row Adapter
       Output: server_ip
  2 --Vector Sort
       Output: server_ip
       Sort Key: bitmaptbl_high.server_ip
  3 --Vector Sonic Hash Aggregate
       Output: server_ip
       Group By Key: bitmaptbl_high.server_ip
  4 --Vector Streaming (type: GATHER)
       Output: server_ip
  5 --Vector Sonic Hash Aggregate
       Output: server_ip
       Group By Key: bitmaptbl_high.server_ip
       Dict Optimized: bitmaptbl_high.server_ip
       Dict Decoded: bitmaptbl_high.server_ip
  6 --CStore Scan on public.bitmaptbl_high
       Output: server_ip
       Dict Encoded: server_ip
       Distribute Key: scope_name
```

使用no dict可以禁用server_ip使用字典编码：

```
SELECT /*+ no dict (bitmaptbl_high (server_ip)) */ distinct(server_ip) FROM bitmaptbl_high WHERE
scope_name='saetataetaeta' ORDER BY server_ip;
```

14.4.9.12 配置参数的 hint

功能描述

指明计划生成时配置参数的值，又称作guc hint。

注意事项

- 如果hint设置的配置参数在语句级别生效，则该hint必须写在顶层查询中，而不能写在子查询中。对于UNION、INTERSECT、EXCEPT和MINUS语句，可以将在语句级别的guc hint写在参与集合运算的任意一个SELECT子句上，该guc hint设置的配置参数会在参与集合运算的每个SELECT子句上生效。
- 子查询提升时，该子查询上的所有guc hint会被丢弃。
- 如果一个配置参数既被语句级别的guc hint设置，又被子查询级别的guc hint设置，则子查询级别的guc hint在对应的子查询中生效，语句级别的guc hint在语句的其他子查询中生效。

语法格式

```
set [global]([@block_name] guc_name guc_value)
```

 版权所有 © 华为云计算技术有限公司

参数说明

- **global**表示hint设置的配置参数在语句级别生效，不加global表示hint设置的配置参数在子查询级别生效，即仅在hint所在的子查询中生效，在该语句的其他子查询中不生效。
- **block_name**表示语句块的block_name，详细说明请参考**block_name**。
- **guc_name**表示hint指定的配置参数的名称。
- **guc_value**表示hint指定的配置参数的值。

guc hint当前仅支持部分配置参数，并且有些配置参数不支持在子查询级别设置，只能在语句级别设置，以下为支持的参数列表：

表 14-19 guc hint 支持的配置参数

配置参数名	是否支持在子查询级别设置
agg_max_mem	是
agg_redistribute_enhancement	是
best_agg_plan	是
cost_model_version	否
cost_param	否
enable_array_optimization	否
enable_bitmapscan	是
enable_broadcast	是
enable_csqual_pushdown	否
enable_redistribute	是
enable_extrapolation_stats	是
enable_fast_query_shipping	否
enable_force_vector_engine	否
enable_hashagg	是
enable_hashfilter	否
enable_hashjoin	是
enable_index_nestloop	是
enable_indexonlyscan	是
enable_indexscan	是
enable_join_pseudoconst	是
enable_mergejoin	是
enable_mixedagg	否

版权所有 © 华为云计算技术有限公司

配置参数名	是否支持在子查询级别设置
enable_nestloop	是
enable_nodegroup_debug	否
enable_partition_dynamic_pruning	是
enable_seqscan	是
enable_sonic_hashagg	否
enable_sonic_hashjoin	是
enable_sort	是
enable_stream_ctescan	否
enable_tidscan	是
enable_value_redistribute	是
enable_vector_engine	否
expected_computing_nodegroup	否
force_bitmapand	是
from_collapse_limit	是
join_collapse_limit	是
join_num_distinct	是
outer_join_max_rows_multipler	是
prefer_hashjoin_path	否
qrw_inlist2join_optmode	是
qual_num_distinct	是
query_dop	否
query_max_mem	否
query_mem	否
rewrite_rule	否
setop_optmode	是
skew_option	是
stream_ctescan_max_estimate_mem	否
stream_ctescan_pred_threshold	否
stream_ctescan_refcount_threshold	否
windowagg_pushdown_enhancement	否

 版权所有 © 华为云计算技术有限公司

配置参数名	是否支持在子查询级别设置
index_selectivity_cost	是
index_cost_limit	是

示例

对示例中原语句使用如下hint：

```
explain
select /*+ set global(query_dop 0) */ i_product_name product_name
...
```

该hint表示：在整个语句的计划生成过程中，将配置参数query_dop设置为0，即打开SMP自适应功能。生成的计划如下图所示：

```
 id |                                    operation                                    | E-rows  | E-memory | E-width | E-costs
----+---------------------------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                                                 |       1 |          |     230 | 19595.89
  2 |    ->  Vector Sonic Hash Aggregate                                              |       1 |          |     230 | 19595.89
  3 |       ->  Vector Streaming (type: GATHER)                                       |       3 |          |     230 | 19595.89
  4 |          ->  Vector Sonic Hash Aggregate                                        |       3 | 16MB     |     230 | 19595.66
  5 |             ->  Vector Nest Loop (6,28)                                         |       3 | 1MB      |     126 | 19595.62
  6 |                ->  Vector Nest Loop (7,27)                                      |       3 | 1MB      |     130 | 19291.57
  7 |                   ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)             |       3 | 4MB      |     118 | 19279.41
  8 |                      ->  Vector Nest Loop (9,24)                                |       3 | 1MB      |     118 | 19279.38
  9 |                         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) |       3 | 4MB      |      82 | 18117.66
 10 |                            ->  Vector Nest Loop (11,21)                         |       3 | 1MB      |      82 | 18117.61
 11 |                               ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) |   3 | 4MB      |      82 | 16195.20
 12 |                                  ->  Vector Sonic Hash Join (13,15)            |       3 | 16MB     |      82 | 16195.15
 13 |                                     ->  Vector Partition Iterator              |  287514 | 1MB      |      12 | 1110.42
 14 |                                        ->  Partitioned CStore Scan on store_returns | 287514 | 1MB   |      12 | 1110.42
 15 |                                     ->  Vector Streaming(type: LOCAL BROADCAST dop: 2/2) | 2764 | 4MB  |      94 | 14718.42
 16 |                                        ->  Vector Sonic Hash Join (17,19)       |    1382 | 16MB     |      94 | 14699.69
 17 |                                           ->  Vector Partition Iterator         | 2880404 | 1MB      |      39 | 11541.07
 18 |                                              ->  Partitioned CStore Scan on store_sales | 2880404 | 1MB |  39 | 11541.07
 19 |                                           ->  Vector Streaming(type: LOCAL BROADCAST dop: 2/2) |  16 | 4MB  |      55 | 1947.12
 20 |                                              ->  CStore Scan on item            |       8 | 1MB      |      55 | 1947.00
 21 |                               ->  Vector Materialize                            |  100000 | 16MB     |       8 | 1797.41
 22 |                                  ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 2/2) | 100000 | 4MB |  8 | 1714.07
 23 |                                     ->  CStore Scan on customer                |  100000 | 1MB      |       8 | 703.67
 24 |                         ->  Vector Materialize                                  |   50000 | 16MB     |      44 | 1099.22
 25 |                            ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 2/2) |  50000 | 4MB  |      44 | 1057.55
 26 |                               ->  CStore Scan on customer_address ad2           |   50000 | 1MB      |      44 | 552.33
 27 |                   ->  CStore Scan on store                                      |      36 | 1MB      |      20 | 12.01
 28 |                ->  CStore Scan on promotion                                     |     900 | 1MB      |       4 | 300.30
(28 rows)
```

14.4.9.13 Hint 的错误、冲突及告警

Plan Hint的结果会体现在计划的变化上，可以通过explain来查看变化。

Hint中的错误不会影响语句的执行，只是不能生效，该错误会根据语句类型以不同方式提示用户。对于explain语句，hint的错误会以warning形式显示在界面上，对于非explain语句，会以debug1级别日志显示在日志中，关键字为PLANHINT。

hint 的错误类型

- 语法错误

  语法规则树归约失败，会报错，指出出错的位置。

  例如：hint关键字错误，leading hint或join hint指定2个表以下，其他hint未指定表等。一旦发现语法错误，则立即终止hint的解析，所以此时只有错误前面的解析完的hint有效。

  例如：

  ```
  leading((t1 t2)) nestloop(t1) rows(t1 t2 #10)
  ```

  nestloop(t1)存在语法错误，则终止解析，可用hint只有之前解析的leading((t1 t2))。

版权所有 © 华为云计算技术有限公司

- 语义错误
  - 表不存在，存在多个，或在leading或join中出现多次，均会报语义错误。
  - scanhint中的index不存在，会报语义错误。
  - 另外，如果子查询提升后，同一层出现多个名称相同的表，且其中某个表需要被hint，hint会存在歧义，无法使用，需要为相同表增加别名规避。
- hint重复或冲突

  如果存在hint重复或冲突，只有第一个hint生效，其他hint均会失效，会给出提示。
  - hint重复是指，hint的方法及表名均相同。例如：nestloop(t1 t2) nestloop(t1 t2)。
  - hint冲突是指，table list一样的hint，存在不一样的hint，hint的冲突仅对于每一类hint方法检测冲突。

    例如：nestloop (t1 t2) hashjoin (t1 t2)，则后面与前面冲突，此时hashjoin的hint失效。注意：nestloop(t1 t2)和no mergejoin(t1 t2)不冲突。

    须知

    leading hint中的多个表会进行拆解。例如：leading ((t1 t2 t3))会拆解成：leading((t1 t2)) leading(((t1 t2) t3))，此时如果存在leading((t2 t1))，则两者冲突，后面的会被丢弃。（例外：指定内外表的hint若与不指定内外表的hint重复，则始终丢弃不指定内外表的hint。）
- 子链接提升后hint失效

  子链接提升后的hint失效，会给出提示。通常出现在子链接中存在多个表连接的场景。提升后，子链接中的多个表不再作为一个整体出现在join中。
- 列类型不支持重分布
  - 对于skew hint来说，目的是为了进行重分布时的调优，所以当hint列的类型不支持重分布时，hint将无效。
- hint未被使用
  - 非等值join使用hashjoin hint或mergejoin hint
  - 不包含索引的表使用indexscan hint或indexonlyscan hint
  - 通常只有在索引列上使用过滤条件才会生成相应的索引路径，全表扫描将不会使用索引，因此使用indexscan hint或indexonlyscan hint将不会使用
  - indexonlyscan只有输出列仅包含索引列才会使用，否则指定时hint不会被使用
  - 多个表存在等值连接时，仅尝试有等值连接条件的表的连接，此时没有关联条件的表之间的路径将不会生成，所以指定相应的leading，join，rows hint将不使用，例如：t1 t2 t3表join，t1和t2，t2和t3有等值连接条件，则t1和t3不会优先连接，leading(t1 t3)不会被使用。
  - 生成stream计划时，如果表的分布列与join列相同，则不会生成redistribute的计划；如果不同，且另一表分布列与join列相同，只能生成redistribute的计划，不会生成broadcast的计划，指定相应的hint则不会被使用。
  - 对于AGG重分布列的hint，hint未被使用的可能原因如下：
    - 指定的分布键包含不支持重分布的数据类型。

 版权所有 © 华为云计算技术有限公司

- 执行计划中不需要重分布。
- 执行的分布键的序号有误。
- 对于使用grouping sets子句和cube子句的函数，window agg中的分布键，不支持hint。

  说明

  指定Agg重分布列Hint，仅8.1.3.100及以上集群版本支持。

– 如果子连接未被提升，则blockname hint不会被使用。
– 对于skew hint，hint未被使用的可能原因如下：
  - 计划中不需要进行重分布。
  - hint指定的列为包含分布键。
  - hint指定倾斜信息有误或不完整，如对于join优化未指定值。
  - 倾斜优化的GUC参数处于关闭状态。
– 对于guc hint，hint未被使用的可能原因如下：
  - 配置参数不存在。
  - 配置参数不支持guc hint。
  - 配置参数的值无效。
  - 语句级别的guc hint没有被写在顶层查询中。
  - 子查询级别的guc hint设置的配置参数不支持在子查询级别设置。
  - guc hint所在的子查询被提升。

14.4.9.14 Plan Hint 实际调优案例

本节以TPC-DS标准测试的Q24的部分语句为例，在1000X，24DN环境上，说明使用plan hint进行实际调优的过程。示例如下：

```
select avg(netpaid) from
(select c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size
,sum(ss_sales_price) netpaid
from store_sales
,store_returns
,store
,item
,customer
,customer_address
where ss_ticket_number = sr_ticket_number
and ss_item_sk = sr_item_sk
and ss_customer_sk = c_customer_sk
```


版权所有 © 华为云计算技术有限公司

```
and ss_item_sk = i_item_sk
and ss_store_sk = s_store_sk
and c_birth_country = upper(ca_country)
and s_zip = ca_zip
and s_market_id=7
group by c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size);
```

1. 该语句的初始计划如下，运行时间110s：

图 14-11 语句初始计划

```
 id |                                operation                                  |         A-time          |   A-rows   |   E-rows   |
----+---------------------------------------------------------------------------+-------------------------+------------+------------+
  1 | ->  Row Adapter                                                           | 110324.107              |          1 |          1 |
  2 |    ->  Vector Aggregate                                                   | 110324.093              |          1 |          1 |
  3 |       ->  Vector Streaming (type: GATHER)                                 | 110323.958              |         24 |         24 |
  4 |          ->  Vector Aggregate                                             | [110179.302,110309.653] |         24 |         24 |
  5 |             ->  Vector Hash Aggregate                                     | [110178.388,110308.515] |     647824 |      16656 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                   | [77616.177,96478.771]   |  666834733 |      16664 |
  7 |                   ->  Vector Hash Join (8,22)                             | [81727.257,84728.519]   |  666834733 |      16664 |
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)             | [78770.520,82021.087]   |  666834733 |      16664 |
  9 |                         ->  Vector Hash Join (10,21)                      | [88066.755,90701.860]   |  666834733 |      16664 |
 10 |                            ->  Vector Streaming(type: BROADCAST)          | [7940.962,21430.725]    |  591882336 |      51360 |
 11 |                               ->  Vector Hash Join (12,20)                | [2419.995,5319.606]     |   24661764 |       2140 |
 12 |                                  ->  Vector Streaming(type: REDISTRIBUTE) | [1750.448,4659.581]     |   25258268 |       2241 |
 13 |                                     ->  Vector Hash Join (14,18)          | [15240.666,17159.616]   |   25258268 |       2241 |
 14 |                                        ->  Vector Hash Join (15,17)       | [12112.913,13563.366]   |  252564412 |  472070592 |
 15 |                                           ->  Vector Partition Iterator   | [11148.731,12473.230]   | 2879987999 | 2879987999 |
 16 |                                              ->  Partitioned CStore Scan on public.store_sales | [11097.921,12412.596] | 2879987999 | 2879987999 |
 17 |                                           ->  CStore Scan on public.store | [0.447,0.689]           |       2064 |       2064 |
 18 |                                        ->  Vector Partition Iterator      | [296.805,319.014]       |  287999764 |  287999764 |
 19 |                                           ->  Partitioned CStore Scan on public.store_returns | [292.938,314.787] | 287999764 | 287999764 |
 20 |                                  ->  CStore Scan on public.customer       | [114.358,144.462]       |   12000000 |   12000000 |
 21 |                            ->  CStore Scan on public.customer_address     | [38.426,56.753]         |    6000000 |    6000000 |
 22 |                      ->  CStore Scan on public.item                       | [3.160,5.026]           |     300000 |     300000 |
(22 rows)
```

该计划中，第10层算子使用broadcast性能较差，由于第11层算子估算行数为2140，比实际行数严重低估。错误行数估算主要来源于第13层算子的行数低估，根因是第13层hashjoin中，使用store_sales的(ss_ticket_number, ss_item_sk)列和store_returns的(sr_ticket_number, sr_item_sk)列进行关联，由于缺少多列相关性的估算导致行数严重低估。

2. 使用如下的rows hint进行调优后，计划如下，运行时间318s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns * 11270)*/ c_last_name ...
```

图 14-12 使用 rows hint 进行调优

```
 id |                                operation                                  |         A-time          |   A-rows   |   E-rows   |
----+---------------------------------------------------------------------------+-------------------------+------------+------------+
  1 | ->  Row Adapter                                                           | 318585.246              |          1 |          1 |
  2 |    ->  Vector Aggregate                                                   | 318585.232              |          1 |          1 |
  3 |       ->  Vector Streaming (type: GATHER)                                 | 318585.082              |         24 |         24 |
  4 |          ->  Vector Aggregate                                             | [318323.324,318499.290] |         24 |         24 |
  5 |             ->  Vector Hash Aggregate                                     | [318320.813,318497.054] |     647824 |  187770504 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                   | [288074.860,305601.698] |  666834733 |  187770507 |
  7 |                   ->  Vector Hash Join (8,22)                             | [253642.468,315808.664] |  666834733 |  187770507 |
  8 |                      ->  Vector Hash Join (9,18)                          | [250904.317,315684.018] |  666834733 |  187770507 |
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)          | [4552.500,310602.307]   |  275042158 |  147106999 |
 10 |                            ->  Vector Hash Join (11,17)                   | [7658.951,14053.823]    |  275042158 |  147106999 |
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)    | [3953.255,10264.943]    |  287999764 |  154060900 |
 12 |                                  ->  Vector Hash Join (13,15)             | [28196.188,32838.794]   |  287999764 |  154060900 |
 13 |                                     ->  Vector Partition Iterator         | [11477.673,12324.583]   | 2879987999 | 2879987999 |
 14 |                                        ->  Partitioned CStore Scan on public.store_sales | [11411.382,12250.209] | 2879987999 | 2879987999 |
 15 |                                     ->  Vector Partition Iterator         | [304.188,403.205]       |  287999764 |  287999764 |
 16 |                                        ->  Partitioned CStore Scan on public.store_returns | [299.838,398.255] | 287999764 | 287999764 |
 17 |                               ->  CStore Scan on public.customer          | [122.246,170.128]       |   12000000 |   12000000 |
 18 |                         ->  Vector Streaming(type: REDISTRIBUTE)          | [57.558,117.461]        |     492915 |     146467 |
 19 |                            ->  Vector Hash Join (20,21)                   | [45.554,96.238]         |     492915 |     146467 |
 20 |                               ->  CStore Scan on public.customer_address  | [39.738,89.412]         |    6000000 |    6000000 |
 21 |                               ->  CStore Scan on public.store             | [0.361,1.095]           |       2064 |       2064 |
 22 |                      ->  Vector Streaming(type: BROADCAST)                | [48.986,91.170]         |    7200000 |    7200000 |
 23 |                         ->  CStore Scan on public.item                    | [4.506,6.602]           |     300000 |     300000 |
(23 rows)
```


版权所有 © 华为云计算技术有限公司

时间反而劣化了，原因是第8层hashjoin过慢引起第9层redistribute时间过慢导致，其中第9层redistribute并没有数据倾斜，hashjoin慢的原因是由于第18层redistribute后数据倾斜导致。

3. 经过实际数据验证，customer_address的两个join列的不同值数目较少，使用其进行join容易出现数据倾斜，故把customer_address放到最后进行join。使用如下的hint进行调优后，计划如下，运行时间116s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((store_sales store_returns store item customer) customer_address)*/
c_last_name ...
```

图 14-13 hint 调优

```
 id |                                   operation                                    |        A-time         |  A-rows   |  E-rows   |
----+--------------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                                | 116326.597            |         1 |         1 |
  2 |   ->  Vector Aggregate                                                         | 116326.590            |         1 |         1 |
  3 |     ->  Vector Streaming (type: GATHER)                                        | 116326.473            |        24 |        24 |
  4 |       ->  Vector Aggregate                                                     | [116157.161,116236.494] |        24 |        24 |
  5 |         ->  Vector Hash Aggregate                                              | [116155.328,116233.946] |    647824 | 187770504 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                             | [84103.951,102052.326]  | 666834733 | 187770507 |
  7 |             ->  Vector Hash Join (8,10)                                        | [23229.469,47484.697]   | 666834733 | 187770507 |
  8 |               ->  Vector Streaming(type: REDISTRIBUTE)                         | [38.367,74.930]         |   6000000 |   6000000 |
  9 |                 ->  CStore Scan on public.customer_address                     | [69.877,121.460]        |   6000000 |   6000000 |
 10 |               ->  Vector Streaming(type: REDISTRIBUTE)                         | [17404.744,17567.550]   |  24661764 |  24112909 |
 11 |                 ->  Vector Hash Join (12,22)                                   | [16123.627,16397.246]   |  24661764 |  24112909 |
 12 |                   ->  Vector Streaming(type: REDISTRIBUTE)                     | [15320.663,15741.646]   |  25258268 |  25252751 |
 13 |                     ->  Vector Hash Join (14,21)                               | [14962.342,16375.458]   |  25258268 |  25252751 |
 14 |                       ->  Vector Hash Join (15,19)                             | [14449.031,15825.949]   |  25258268 |  25252751 |
 15 |                         ->  Vector Hash Join (16,18)                           | [11439.959,12510.065]   | 252564412 | 472070592 |
 16 |                           ->  Vector Partition Iterator                        | [10531.986,11536.213]   | 2879987999 | 2879987999 |
 17 |                             ->  Partitioned CStore Scan on public.store_sales  | [10483.634,11474.944]   | 2879987999 | 2879987999 |
 18 |                           ->  CStore Scan on public.store                      | [0.347,0.463]           |      2064 |      2064 |
 19 |                         ->  Vector Partition Iterator                          | [293.977,365.021]       | 287999764 | 287999764 |
 20 |                           ->  Partitioned CStore Scan on public.store_returns  | [289.936,360.808]       | 287999764 | 287999764 |
 21 |                       ->  CStore Scan on public.item                           | [3.109,5.245]           |    300000 |    300000 |
 22 |                   ->  CStore Scan on public.customer                           | [113.871,141.791]       |  12000000 |  12000000 |
(22 rows)
```

发现时间基本花在了第6层redistribute算子上，需要进一步优化。

4. 由于最后一层redistribute包含倾斜，所以时间较长。为了避免倾斜，需要将item表放在最后join，由于item表的join并不能使行数减少。修改hint如下并执行，计划如下，运行时间120s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
c_last_name ...
```

图 14-14 修改 hint 并执行语句

```
 id |                                   operation                                    |        A-time         |  A-rows   |  E-rows   |
----+--------------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                                | 120377.258            |         1 |         1 |
  2 |   ->  Vector Aggregate                                                         | 120377.245            |         1 |         1 |
  3 |     ->  Vector Streaming (type: GATHER)                                        | 120377.091            |        24 |        24 |
  4 |       ->  Vector Aggregate                                                     | [120184.884,120301.704] |        24 |        24 |
  5 |         ->  Vector Hash Aggregate                                              | [120183.119,120297.845] |    647824 | 187770504 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                             | [87775.682,106070.878]  | 666834733 | 187770507 |
  7 |             ->  Vector Hash Join (8,22)                                        | [22323.764,49878.523]   | 666834733 | 187770507 |
  8 |               ->  Vector Hash Join (9,11)                                      | [21129.236,45208.255]   | 666834733 | 187770507 |
  9 |                 ->  Vector Streaming(type: REDISTRIBUTE)                       | [37.859,75.412]         |   6000000 |   6000000 |
 10 |                   ->  CStore Scan on public.customer_address                   | [74.798,114.449]        |   6000000 |   6000000 |
 11 |                 ->  Vector Streaming(type: REDISTRIBUTE)                       | [15714.458,15824.928]   |  24661764 |  24112909 |
 12 |                   ->  Vector Hash Join (13,21)                                 | [14637.516,14955.464]   |  24661764 |  24112909 |
 13 |                     ->  Vector Streaming(type: REDISTRIBUTE)                   | [13898.593,14333.200]   |  25258268 |  25252751 |
 14 |                       ->  Vector Hash Join (15,19)                             | [14166.917,15378.244]   |  25258268 |  25252751 |
 15 |                         ->  Vector Hash Join (16,18)                           | [11272.239,12052.532]   | 252564412 | 472070592 |
 16 |                           ->  Vector Partition Iterator                        | [10409.566,11127.981]   | 2879987999 | 2879987999 |
 17 |                             ->  Partitioned CStore Scan on public.store_sales  | [10365.838,11077.601]   | 2879987999 | 2879987999 |
 18 |                           ->  CStore Scan on public.store                      | [0.431,0.609]           |      2064 |      2064 |
 19 |                         ->  Vector Partition Iterator                          | [343.780,408.254]       | 287999764 | 287999764 |
 20 |                           ->  Partitioned CStore Scan on public.store_returns  | [339.844,403.923]       | 287999764 | 287999764 |
 21 |                       ->  CStore Scan on public.customer                       | [117.234,163.598]       |  12000000 |  12000000 |
 22 |                   ->  Vector Streaming(type: BROADCAST)                        | [44.571,130.129]        |   7200000 |   7200000 |
 23 |                     ->  CStore Scan on public.item                             | [4.169,6.347]           |    300000 |    300000 |
(23 rows)
```

该计划中的redistribute问题并没有解决，因为第22层item表做了broadcast，导致与customer_address表join后的倾斜并没有被消除掉。

5. 增加如下禁止item表做broadcast的hint，使与customer_address join的表做redistribute（也可以进行join表redistribute的hint），计划如下，运行时间105s：

 版权所有 © 华为云计算技术有限公司

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
no broadcast(item)*/
c_last_name ...
```

图 14-15 执行计划

```
 id |                              operation                               |          A-time           |   A-rows   |   E-rows   |
----+----------------------------------------------------------------------+---------------------------+------------+------------+
  1 | ->  Row Adapter                                                      | 105854.957                |          1 |          1 |
  2 |   ->  Vector Aggregate                                               | 105854.948                |          1 |          1 |
  3 |     ->  Vector Streaming (type: GATHER)                              | 105854.825                |         24 |         24 |
  4 |       ->  Vector Aggregate                                           | [105706.709,105776.135]   |         24 |         24 |
  5 |         ->  Vector Hash Aggregate                                    | [105705.061,105773.013]   |     647824 |  187770504 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                   | [70701.966,89973.672]     |  666834733 |  187770507 |
  7 |             ->  Vector Hash Join (8,23)                              | [71759.500,79018.433]     |  666834733 |  187770507 |
  8 |               ->  Vector Streaming(type: REDISTRIBUTE)               | [69794.307,77269.178]     |  666834733 |  187770507 |
  9 |                 ->  Vector Hash Join (10,12)                         | [21443.307,46714.378]     |  666834733 |  187770507 |
 10 |                   ->  Vector Streaming(type: REDISTRIBUTE)           | [41.295,83.419]           |    6000000 |    6000000 |
 11 |                     ->  CStore Scan on public.customer_address       | [70.405,166.072]          |    6000000 |    6000000 |
 12 |                   ->  Vector Streaming(type: REDISTRIBUTE)           | [15689.053,15788.475]     |   24661764 |   24112909 |
 13 |                     ->  Vector Hash Join (14,22)                     | [14517.847,14712.929]     |   24661764 |   24112909 |
 14 |                       ->  Vector Streaming(type: REDISTRIBUTE)       | [13806.733,14089.770]     |   25258268 |   25252751 |
 15 |                         ->  Vector Hash Join (16,20)                 | [13709.384,15095.449]     |   25258268 |   25252751 |
 16 |                           ->  Vector Hash Join (17,19)               | [10944.796,11827.285]     |  252564412 |  472070592 |
 17 |                             ->  Vector Partition Iterator            | [10070.316,10884.728]     | 2879987999 | 2879987999 |
 18 |                               ->  Partitioned CStore Scan on public.store_sales | [10018.966,10828.990] | 2879987999 | 2879987999 |
 19 |                             ->  CStore Scan on public.store          | [0.447,0.568]             |       2064 |       2064 |
 20 |                           ->  Vector Partition Iterator              | [293.042,329.056]         |  287999764 |  287999764 |
 21 |                             ->  Partitioned CStore Scan on public.store_returns | [288.631,324.782] |  287999764 |  287999764 |
 22 |                       ->  CStore Scan on public.customer             | [113.735,138.235]         |   12000000 |   12000000 |
 23 |               ->  CStore Scan on public.item                         | [3.127,5.357]             |     300000 |     300000 |
(23 rows)
```

6. 发现最后一层使用单层Agg，但行数缩减较多。使用相同的hint，同时结合参数best_agg_plan=3进行双层Agg调优，最终计划如下图所示，运行时间94s，完成调优。

图 14-16 最终调优计划

```
 id |                              operation                               |          A-time           |   A-rows   |   E-rows   |
----+----------------------------------------------------------------------+---------------------------+------------+------------+
  1 | ->  Row Adapter                                                      | 94004.670                 |          1 |          1 |
  2 |   ->  Vector Aggregate                                               | 94004.655                 |          1 |          1 |
  3 |     ->  Vector Streaming (type: GATHER)                              | 94004.504                 |         24 |         24 |
  4 |       ->  Vector Aggregate                                           | [93833.832,93928.052]     |         24 |         24 |
  5 |         ->  Vector Hash Aggregate                                    | [93832.460,93926.412]     |     647824 |  187770507 |
  6 |           ->  Vector Streaming(type: REDISTRIBUTE)                   | [93640.866,93787.939]     |     647824 |  183912384 |
  7 |             ->  Vector Hash Aggregate                                | [93687.544,93791.242]     |     647824 |  183912384 |
  8 |               ->  Vector Hash Join (9,24)                            | [70025.469,72773.161]     |  666834733 |  187770507 |
  9 |                 ->  Vector Streaming(type: REDISTRIBUTE)             | [68242.223,71275.972]     |  666834733 |  187770507 |
 10 |                   ->  Vector Hash Join (11,13)                       | [21421.136,44830.306]     |  666834733 |  187770507 |
 11 |                     ->  Vector Streaming(type: REDISTRIBUTE)         | [35.444,71.328]           |    6000000 |    6000000 |
 12 |                       ->  CStore Scan on public.customer_address     | [67.246,119.224]          |    6000000 |    6000000 |
 13 |                     ->  Vector Streaming(type: REDISTRIBUTE)         | [16089.853,16212.570]     |   24661764 |   24112909 |
 14 |                       ->  Vector Hash Join (15,23)                   | [14822.972,15188.942]     |   24661764 |   24112909 |
 15 |                         ->  Vector Streaming(type: REDISTRIBUTE)     | [14061.867,14604.162]     |   25258268 |   25252751 |
 16 |                           ->  Vector Hash Join (17,21)               | [13949.756,15492.311]     |   25258268 |   25252751 |
 17 |                             ->  Vector Hash Join (18,20)             | [10935.742,12160.719]     |  252564412 |  472070592 |
 18 |                               ->  Vector Partition Iterator          | [10052.958,11194.962]     | 2879987999 | 2879987999 |
 19 |                                 ->  Partitioned CStore Scan on public.store_sales | [10008.415,11143.984] | 2879987999 | 2879987999 |
 20 |                               ->  CStore Scan on public.store        | [0.452,0.839]             |       2064 |       2064 |
 21 |                             ->  Vector Partition Iterator            | [298.235,332.736]         |  287999764 |  287999764 |
 22 |                               ->  Partitioned CStore Scan on public.store_returns | [294.067,327.629] |  287999764 |  287999764 |
 23 |                         ->  CStore Scan on public.customer           | [114.377,145.156]         |   12000000 |   12000000 |
 24 |                 ->  CStore Scan on public.item                       | [3.150,3.530]             |     300000 |     300000 |
(24 rows)
```

如果有统计信息变更引起的查询劣化，可以考虑用plan hint来调整到之前的查询计划。这里以TPCH-Q17为例，在收集default_statistics_target设置为–2的统计信息之后，计划相比于默认统计信息发生劣化。

1. 默认统计信息（default_statistics_target设置为100）的计划如下：

版权所有 © 华为云计算技术有限公司

图 14-17 默认统计信息

```
 id |                                                operation                                                |          A-time
----+---------------------------------------------------------------------------------------------------------+---------------------------
  1 | ->  Row Adapter                                                                                         | 265006.779
  2 |    ->  Vector Aggregate                                                                                 | 265006.764
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 265006.071
  4 |          ->  Vector Aggregate                                                                           | [263699.512,264503.084]
  5 |             ->  Vector Hash Join (6,17)                                                                 | [263676.665,264477.932]
  6 |                ->  Vector Streaming(type: LOCAL GATHER dop: 1/4)                                        | [1.998,7.594]
  7 |                   ->  Vector Hash Aggregate                                                             | [201775.393,202432.672]
  8 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 4/4)                            | [201567.130,202231.524]
  9 |                         ->  Vector Hash Join (10,12)                                                    | [170675.231,199908.410]
 10 |                            ->  Vector Partition Iterator                                                | [34847.797,51968.266]
 11 |                               ->  Partitioned CStore Scan on tpch10wx_col.lineitem                      | [33805.013,51137.657]
 12 |                            ->  Vector Hash Aggregate                                                    | [23283.387,25359.493]
 13 |                               ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4)                      | [12850.624,14608.515]
 14 |                                  ->  Vector Hash Aggregate                                              | [2690.439,3616.623]
 15 |                                     ->  Vector Partition Iterator                                       | [2659.700,3579.390]
 16 |                                        ->  Partitioned CStore Scan on tpch10wx_col.part                 | [2642.213,3559.093]
 17 |                ->  Vector Streaming(type: REDISTRIBUTE dop: 1/4)                                        | [262300.732,262961.078]
 18 |                   ->  Vector Hash Join (19,21)                                                          | [225749.727,260990.322]
 19 |                      ->  Vector Partition Iterator                                                      | [40046.078,56220.694]
 20 |                         ->  Partitioned CStore Scan on tpch10wx_col.lineitem                            | [39204.414,55328.448]
 21 |                      ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4)                               | [55748.177,61987.136]
 22 |                         ->  Vector Partition Iterator                                                   | [3042.864,3873.942]
 23 |                            ->  Partitioned CStore Scan on tpch10wx_col.part                             | [3027.023,3848.159]
(23 rows)
```

2. 统计信息变更（default_statistics_target设置为–2）的计划如下：

图 14-18 统计信息变更

```
 id |                                                operation                                                |          A-time
----+---------------------------------------------------------------------------------------------------------+---------------------------
  1 | ->  Row Adapter                                                                                         | 1440492.994
  2 |    ->  Vector Aggregate                                                                                 | 1440492.982
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 1440491.021
  4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/6)                                              | [1439737.284,1440008.568]
  5 |             ->  Vector Aggregate                                                                        | [1439008.369,1439854.148]
  6 |                ->  Vector Hash Join (7,18)                                                              | [1439006.016,1439851.619]
  7 |                   ->  Vector Streaming(type: LOCAL BROADCAST dop: 6/6)                                  | [2.932,139.405]
  8 |                      ->  Vector Hash Aggregate                                                          | [190452.312,195910.748]
  9 |                         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                         | [190171.929,195653.119]
 10 |                            ->  Vector Hash Join (11,13)                                                 | [161076.195,178831.123]
 11 |                               ->  Vector Partition Iterator                                             | [27306.318,45564.565]
 12 |                                  ->  Partitioned CStore Scan on tpch10wx_col.lineitem                   | [26752.444,44912.020]
 13 |                               ->  Vector Hash Aggregate                                                 | [35601.624,39812.058]
 14 |                                  ->  Vector Streaming(type: SPLIT BROADCAST dop: 6/6)                   | [23096.460,27057.137]
 15 |                                     ->  Vector Hash Aggregate                                           | [2372.587,3052.445]
 16 |                                        ->  Vector Partition Iterator                                    | [2345.381,3012.732]
 17 |                                           ->  Partitioned CStore Scan on tpch10wx_col.part              | [2329.874,2989.393]
 18 |                   ->  Vector Hash Join (19,22)                                                          | [1437388.414,1438470.781]
 19 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                            | [1392693.529,1408571.859]
 20 |                         ->  Vector Partition Iterator                                                   | [29065.204,41264.514]
 21 |                            ->  Partitioned CStore Scan on tpch10wx_col.lineitem                         | [28212.219,40133.491]
 22 |                      ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 6/6)                            | [2570.841,3438.567]
 23 |                         ->  Vector Partition Iterator                                                   | [2447.569,3276.369]
 24 |                            ->  Partitioned CStore Scan on tpch10wx_col.part                             | [2432.124,3263.641]
(24 rows)
```

3. 经过对比，劣化的原因主要为lineitem和part表join时stream类型由BroadCast变更为Redistribute导致。可以对语句进行stream方式的hint来调整到之前的计划，例如：

图 14-19 调整语句

```
select /*+ no redistribute(part lineitem) */
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```


版权所有 © 华为云计算技术有限公司

14.4.10 例行维护表

为了保证数据库的有效运行，数据库必须在插入/删除操作后，定期执行VACUUM FULL和ANALYZE，更新统计信息，以便获得更优的性能。

相关概念

使用VACUUM、VACUUM FULL和ANALYZE命令定期对每个表进行维护，主要有以下原因：

- VACUUM FULL可回收已更新或已删除的数据所占据的磁盘空间，同时将小数据文件合并。
- VACUUM对每个表维护了一个可视化映射来跟踪包含对别的活动事务可见的数组的页。一个普通的索引扫描首先通过可视化映射来获取对应的数组，来检查是否对当前事务可见。若无法获取，再通过堆数组抓取的方式来检查。因此更新表的可视化映射，可加速唯一索引扫描。
- VACUUM可避免执行的事务数超过数据库阈值时，事务ID重叠造成的原有数据丢失。
- ANALYZE可收集与数据库中表内容相关的统计信息。统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，生成最有效的执行计划。

操作步骤

步骤1 使用VACUUM或VACUUM FULL命令，进行磁盘空间回收。

- **VACUUM**：

  对表执行VACUUM操作。

  ```
  VACUUM customer;
  ```

  可以与数据库操作命令并行运行。（执行期间，可正常使用的语句：SELECT、INSERT、UPDATE和DELETE。不可正常使用的语句：ALTER TABLE）。

  对表分区执行VACUUM操作。

  ```
  VACUUM customer_par PARTITION ( P1 );
  ```

- **VACUUM FULL**：

  ```
  VACUUM FULL customer;
  ```

  需要向正在执行的表增加排他锁，且需要停止其他所有数据库操作。

  在进行磁盘空间回收时，用户可以使用如下命令查询集群中最早事务对应session，再根据需要结束最早执行的长事务，从而更加高效地利用磁盘空间。

  a. 使用命令从GTM上查询oldestxmin

  ```
  SELECT * FROM pgxc_gtm_snapshot_status();
  ```

  b. 从CN上查询对应的session的pid，此处xmin为上一步的oldestxmin。

  ```
  SELECT * FROM pgxc_running_xacts() where xmin=1400202010;
  ```

步骤2 使用ANALYZE语句更新统计信息。

```
ANALYZE customer;
```

使用ANALYZE VERBOSE语句更新统计信息，并输出表的相关信息。

```
ANALYZE VERBOSE customer;
```

也可以同时执行VACUUM ANALYZE命令进行查询优化。

```
VACUUM ANALYZE customer;
```


版权所有 © 华为云计算技术有限公司

说明

VACUUM和ANALYZE会导致I/O流量的大幅增加，这可能会影响其他活动会话的性能。因此，建议通过“vacuum_cost_delay”参数设置。

----结束

维护建议

- 定期对部分大表执行VACUUM FULL，在性能下降后为全库执行VACUUM FULL，目前暂定每月执行一次VACUUM FULL。
- 定期对系统表执行VACUUM FULL，主要是PG_ATTRIBUTE。
- 启用系统自动清理进程（AUTOVACUUM）自动执行VACUUM和ANALYZE，回收被标识为删除状态的记录空间，并更新表的统计数据。

14.4.11 例行重建索引

背景信息

数据库经过多次删除操作后，索引页面上的索引键将被删除，造成索引膨胀。例行重建索引，可有效地提高查询效率。

数据库支持的索引类型包含B-tree索引、GIN索引和PSORT索引。

- 对于B-tree索引，例行重建索引可有效地提高查询效率。
  - 如果数据发生大量删除后，索引页面上的索引键将被删除，导致索引页面数量的减少，造成索引膨胀。重建索引可回收浪费的空间。
  - 新建的索引中逻辑结构相邻的页面，通常在物理结构中也是相邻的，所以一个新建的索引比更新了多次的索引访问速度要快。
- 对于非B-tree索引，不建议例行重建。

重建索引

重建索引有以下两种方式：

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）。

  在删除索引过程中，会在父表上增加一个短暂的排他锁，阻止相关读写操作。在创建索引过程中，会锁住写操作但是不会锁住读操作，此时读操作只能使用顺序扫描。
- 使用REINDEX语句重建索引。
  - 使用REINDEX TABLE语句重建索引，会在重建过程中增加排他锁，阻止相关读写操作。
  - 使用REINDEX INTERNAL TABLE语句重建desc表（包括列存表的cudesc表）的索引，会在重建过程中增加排他锁，阻止相关读写操作。
  - 使用REINDEX INDEX PARTITION重建索引分区，会在对应的索引分区对象上增加排他锁。在重建过程中，阻止使用该索引分区的读写操作（如索引扫描），不使用该索引分区的读写操作不受影响。

操作步骤

假定在导入表“areaS”上的“area_id”字段上存在普通索引“areaS_idx”。重建索引有以下两种方式：

版权所有 © 华为云计算技术有限公司

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）
  a. 删除索引。
     ```
     DROP INDEX areaS_idx;
     ```
  b. 创建索引。
     ```
     CREATE INDEX areaS_idx ON areaS (area_id);
     ```
- 使用REINDEX重建索引。
  - 使用REINDEX TABLE语句重建索引。
    ```
    REINDEX TABLE areaS;
    ```
  - 使用REINDEX INTERNAL TABLE重建desc表（包括列存表的cudesc表）的索引。
    ```
    REINDEX INTERNAL TABLE areaS;
    ```

14.4.12 SQL 语句出错自动重试

DWS支持在SQL语句执行出错时自动重试（下文简称CN Retry）。对于来自gsql客户端、JDBC、ODBC驱动的SQL语句，在SQL语句执行失败时，CN端能够自动识别语句执行过程中的报错，并重新下发任务进行自动重试。

CN Retry 功能的限制和约束

表 14-20 CN Retry 功能的限制和约束

限制和约束项	详细内容
功能范围	• 仅能提高故障发生时SQL语句执行成功率，不能保证100%的执行成功。 • CN Retry默认开启，开启后temp表会记录日志，关闭CN Retry后，temp表不会记录日志，因此不能在使用temp表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。 • CN Retry默认开启，开启后新创建的unlogged表会忽视unlogged关键字，建成普通表。关闭CN Retry后，unlogged表不会记录日志，因此不能在使用unlogged表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。 • 在使用gds进行数据导出时，支持CN Retry。现有机制导出时会对重复文件进行检测并删除相同的文件，因此建议不要对相同的外表重复导出数据，除非确定数据目录中相同文件名的文件需要删除。
错误类型	SQL语句出错时能够被识别和重试的错误，仅限在错误类型列表（请参考**表14-21**）中定义的错误。
语句类型	支持单语句CN Retry、存储过程、函数、匿名块。不支持事务块中的语句。

版权所有 © 华为云计算技术有限公司

限制和约束项	详细内容
存储过程语句	• 包含EXCEPTION的存储过程，如果在执行过程中（包含语句块执行和EXCEPTION中的语句执行）错误被抛出，可以retry，且系统内部错误发生时，retry会先于EXCEPTION被执行，而如果报错被EXCEPTION捕获则不能retry。 • 不支持使用全局变量的package。 • 不支持DBMS_JOB。 • 不支持UTL_FILE。 • 如果存储过程中有输出打印信息（如dbms_output.put_line或raise info等），则发生retry时会重复输出已打印的消息，并会在重复消息前输出“Notice：Retry triggered, some message may be duplicated.”进行提示。
集群状态	• 仅支持DN、GTM实例故障。 • CN Retry有次数限制，如果在CN Retry达到最大尝试次数（最大次数由max_query_retry_times控制）之前，集群状态无法从故障状态恢复到正常状态，那么CN Retry不能保证执行成功。 • 扩容时不支持CN Retry。
数据导入	• 不支持COPY FROM STDIN语句。 • 不支持gsql \copy from元命令。 • 不支持JDBC CopyManager copyIn导入数据。

CN Retry 支持的错误类型列表

CN Retry支持的错误类型列表和对应的错误码信息见**表14-21**， 可以通过GUC参数**retry_ecode_list**设置CN Retry支持的错误类型列表，但不建议用户直接修改该参数，如有修改需求请联系技术工程师协助处理。

表 14-21 CN Retry 支持的错误类型列表

错误类型	错误码	备注
对端连接重置（CONNECTION_RESET_BY_PEER）	YY001	TCP通信错误：Connection reset by peer（CN和DN间通信）
对端流重置（STREAM_CONNECTION_RESET_BY_PEER）	YY002	TCP通信错误：Stream connection reset by peer（DN和DN间通信）
锁等待超时（LOCK_WAIT_TIMEOUT）	YY003	锁超时，Lock wait timeout
连接超时（CONNECTION_TIMED_OUT）	YY004	TCP通信错误，Connection timed out

版权所有 © 华为云计算技术有限公司

错误类型	错误码	备注
查询设置错误（SET_QUERY_ERROR）	YY005	SET命令发送失败，Set query
超出逻辑内存（OUT_OF_LOGICAL_MEMORY）	YY006	内存申请失败，Out of logical memory
通信库内存分配（SCTP_MEMORY_ALLOC）	YY007	SCTP通信错误，Memory allocate error
无通信库缓存数据（SCTP_NO_DATA_IN_BUFFER）	YY008	SCTP通信错误，SCTP no data in buffer
通信库释放内存关闭（SCTP_RELEASE_MEMORY_CLOSE）	YY009	SCTP通信错误，Release memory close
SCTP、TCP断开（SCTP_TCP_DISCONNECT）	YY010	SCTP通信错误，TCP disconnect
通信库断开（SCTP_DISCONNECT）	YY011	SCTP通信错误，SCTP disconnect
通信库远程关闭（SCTP_REMOTE_CLOSE）	YY012	SCTP通信错误，Stream closed by remote
等待未知通信库通信（SCTP_WAIT_POLL_UNKNOWN）	YY013	等待未知通信库通信，SCTP wait poll unknown
无效快照（SNAPSHOT_INVALID）	YY014	快照非法，Snapshot invalid
通讯接收信息错误（ERRCODE_CONNECTION_RECEIVE_WRONG）	YY015	连接获取错误，Connection receive wrong
内存耗尽（OUT_OF_MEMORY）	53200	内存耗尽，Out of memory
连接失败（CONNECTION_FAILURE）	08006	GTM出错，Connection failure
连接异常（CONNECTION_EXCEPTION）	08000	连接出现错误，和DN的通讯失败，Connection exception
管理员关闭系统（ADMIN_SHUTDOWN）	57P01	管理员关闭系统，Admin shutdown
关闭远程流接口（STREAM_REMOTE_CLOSE_SOCKET）	XX003	关闭远程套接字，Stream remote close socket
重复查询编号（ERRCODE_STREAM_DUPLICATE_QUERY_ID）	XX009	重复查询，Duplicate query id

版权所有 © 华为云计算技术有限公司

错误类型	错误码	备注
stream查询并发更新同一行（ERRCODE_STREAM_CONCURRENT_UPDATE）	YY016	stream查询并发更新同一行，Stream concurrent update
LLVM内存分配错误（ERRCODE_LLVM_BAD_ALLOC_ERROR ）	CG003	内存分配错误， Allocate error
LLVM致命错误（ERRCODE_LLVM_FATAL_ERROR ）	CG004	致命错误，Fatal error
HashJoin临时文件读取错误(ERRCODE_HASHJOIN_TEMP_FILE_ERROR)	F0011	临时文件读取错误，File error
Buffer文件读取错误(ERRCODE_BUFFER_FILE_ERROR)	F0012	文件读取错误，File error
分区个数发生变化(ERRCODE_PARTITION_NUM_CHANGED)	45003	在扫描LIST分区表时，发现此时的分区个数和优化阶段的分区个数不一致，一般出现在查询和ADD/DROP分区并发时。（此错误类型仅8.1.3及以上集群版本支持）
节点间对象SCHEMA名称不一致(ERRCODE_UNMATCH_OBJECT_SCHEMA)	42P30	对象SCHEMA名称不一致，Unmatched schema name

开启 CN Retry 功能

设置以下GUC参数开启CN Retry功能：

- 必选GUC参数（CN和DN都需设置）

  max_query_retry_times，指定SQL语句出错自动重试功能的最大重跑次数，目前支持重跑的错误类型为“Connection reset by peer”、“Lock wait timeout”和“Connection timed out”等，默认值为6。

  ⚠ 注意

  CN Retry功能开启时会为临时表数据记录日志，为保证数据一致性，在使用临时表时不能切换CN Retry开关状态，保持使用临时表的会话中CN Retry开关始终处于打开状态或者关闭状态。

- 可选的GUC参数（如需修改请联系技术支持）

  a. cn_send_buffer_size，指定CN端数据发送数据缓存区的大小。默认值为8KB。

  b. max_cn_temp_file_size，指定SQL语句出错自动重试功能中CN端使用临时文件的最大值。默认值为5GB。

 版权所有 © 华为云计算技术有限公司

14.4.13 query_band 负载识别

概述

DWS实现基于query_band的负载识别和队列内优先级控制，一方面提供了更为灵活的负载识别手段，可根据作业类型、应用名称、脚本名称等识别负载队列，使用户根据业务场景可灵活配置query_band识别队列；另一方面实现了队列内作业下发优先级控制，后续将逐步实现队列内资源优先级控制。

管理员用户可根据业务场景及作业类别配置query_band所关联队列及估算内存限制等实现更为灵活的负载控制与资源管控。如果业务未配置query_band或用户未将query_band关联行为时，作业会默认使用用户关联队列和队列内优先级。

query_band 支持的负载行为

query_band是一个session级别的GUC参数，作为作业标识符，本身没有特殊含义，数据类型为字符型，支持赋值任何字符串。但是为方便区分和设置，query_band负载识别仅支持识别键值对形式的query_band，示例：

```
SET query_band='JobName=abc;AppName=test;UserName=user';
```

其中，‘JobName=abc’、‘AppName=test’以及‘UserName=user’都是一个独立的键值对。query_band键值对规格：

- query_band使用键值对方式设置，即'key=value'；session内支持设置多个query_band键值对，多个键值对之间使用分号分隔。query_band键值对和query_band参数值长度的上限均为1024个字符。
- query_band键值对支持的有效字符包括：数字0~9、大写字母A~Z、小写字母a-z、'.'、'-'、'_' 以及'#'。

query_band负载识别以键值对为单位设置和识别负载行为，目前支持的负载行为，如表14-22所示：

表 14-22 QUERY_BAND 支持负载行为

类别	行为	行为表现
负载管理(workload)	资源池(respool)	query_band关联资源池
负载管理(workload)	优先级(priority)	队列内优先级
次序(order)	队列（respool） 目前为无效字段，主要用于后续扩展。	query_band搜索次序

其中，行为类别用于负载行为归类，不同的负载行为可能属于同一个类别，比如资源池和优先级同属于负载管理类别；负载行为代表query_band键值对关联的是哪种负载行为；行为表现用于记录具体的负载行为是什么；负载类别中的次序用于标记query_band负载行为识别的优先级，一个session内设置多个query_band键值对时，优先使用次序较小的query_band键值对识别负载行为。每一个query_band键值对都可

版权所有 © 华为云计算技术有限公司

以对应0个或多个负载行为，但是一种负载行为只能关联一个。query_band负载行为详细说明如下：

- 资源池：query_band支持关联资源池，作业执行时，若query_band指定了资源池，则使用query_band关联的资源池，否则使用用户关联的资源池。
  - query_band关联资源池时，资源池不存在报错退出，关联失败。
  - query_band关联资源池时，记录query_band与资源池依赖关系。
  - query_band关联资源池删除时，提示有query_band依赖，资源池删除失败。
- 队列内优先级：query_band支持关联作业优先级，支持高中低(High/Medium/Low)三个优先级，同时提供Rush作为特殊优先级（绿色通道），默认优先级为Medium。正常实践过程中，大部分作业使用Medium优先级，优先级较低作业使用Low优先级，特权作业使用High优先级，High作业不建议过多。Rush优先级作为特殊场景下应急使用，平时不建议使用。

  队列内优先级实现队列内排队优先级：
  - 静态负载管理场景下，CN并发不足时，触发CN全局队列排队，CN全局队列为优先级队列。
  - 动态负载管理场景下，DN内存不足时，触发CCN全局排队，CCN全局队列为优先级队列。
  - 资源池并发或内存不足时，触发资源池排队，资源池队列为优先级队列。

  以上优先级队列均遵守以下调度规则：
  - 优先级高作业优先调度。
  - 优先级高作业全部调度完之后调度优先级低作业。
  - 动态负载管理场景下，CN全局队列不支持query_band优先级。
- 次序：支持设置query_band识别次序，未设置识别次序的使用默认次序-1。除默认次序外，不存在次序相同的两个query_band。设置次序时对query_band次序进行校验，存在相同次序时，已存在次序递归+1直到不存在相同次序为止。
  - session内设置多个query_band键值对时，使用次序较小的query_band键值对作为负载识别的query_band。
  - 次序最小为0，默认次序-1为最大次序。
  - 次序都为默认次序时，使用设置靠前的query_band作为负载识别的query_band。
  - 示例：set query_band='b=1;a=3;c=1'; b=1，其中b=1次序-1，a=3次序4，c=1次序1，则使用c=1作为负载识别的query_band，此设计可提供负载管理员对负载调度调整能力。

query_band 应用与配置

- pg_workload_action跨库系统表用于存储query_band行为和次序，详见**PG_WORKLOAD_ACTION**。
- 默认行为和次序在系统表pg_workload_action不存储，query_band有设置非默认行为的，查询其行为默认行为也显示；查询行为和次序都为默认的query_band行为时，显示<query_band information not found>。
- gs_wlm_set_queryband_action函数用于设置query_band行为：其中第一个参数即queryband键值对的长度上限为63个字符；第二个参数action不区分大小写，多个action使用分号分隔；order为缺省参数，默认为-1。具体请参见**gs_wlm_set_queryband_action** 。
- gs_wlm_set_queryband_order函数设置query_band次序：其中第一个参数即queryband键值对的长度上限为63个字符；query_band次序必须大于等于-1，除

 版权所有 © 华为云计算技术有限公司

默认次序-1外不存在两个次序相同的query_band。设置query_band次序时如果存在相同次序的query_band，则将原query_band次序+1。具体请参见**gs_wlm_set_queryband_order**。

- gs_wlm_get_queryband_action函数用于查询query_band行为，具体请参见**gs_wlm_get_queryband_action**。
- pg_queryband_action系统视图用于查询所有query_band行为，具体请参见**PG_QUERYBAND_ACTION**。
- query_band优先级在负载管理视图（**PG_SESSION_WLMSTAT**）中显示为int型，数字和优先级的对应关系如下：
  - 0：表示该作业不受负载管理管控；
  - 1：LOW；
  - 2：MEDIUM；
  - 4：HIGH；
  - 8：RUSH；
- 权限控制：除初始用户外，被授权用户才具有设置和查询query_band权限。

说明

- 批量取消所有运行作业或队列并发上限是1且只有一个队列有作业运行的情况下，可能会触发CN自动唤醒作业导致作业不按照优先级下发。

示例

步骤1 设置query_band“JobName=abc”关联资源池p1、队列内优先级Rush、次序为1。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=p1;priority=rush',1);
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤2 修改query_band“JobName=abc”的关联资源池为p2。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=p2');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤3 修改query_band“JobName=abc”的队列内优先级为High。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','priority=high');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤4 修改query_band“JobName=abc”的次序为3。

```
SELECT * FROM gs_wlm_set_queryband_order('JobName=abc',3);
gs_wlm_set_queryband_order
-----------------------------
 t
(1 row)
```

步骤5 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
    qband     | respool_id | respool | priority | qborder
--------------+------------+---------+----------+---------
 JobName=abc |      17119 | p2      | high     |       1
(1 row)
```


版权所有 © 华为云计算技术有限公司

步骤6 设置query_band“AppName=test”队列内优先级Low，使用用户关联资源池，同时使用默认次序。

```
SELECT * FROM gs_wlm_set_queryband_action('AppName=test','priority=low');
gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤7 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
    qband     | respool_id | respool | priority | qborder
--------------+------------+---------+----------+---------
 AppName=test |          0 | NULL    | low      |      -1
 JobName=abc  |      16754 | p2      | high     |       3
(2 rows)
```

步骤8 取消query_band“JobName=abc”关联的所有负载行为，设置为默认行为即可。

```
SELECT * FROM gs_wlm_set_queryband_action('JobName=abc','respool=null;priority=medium',-1);
NOTICE:  The respool of query_band(JobName=abc) will be removed.
NOTICE:  The priority of query_band(JobName=abc) will be removed.
 gs_wlm_set_queryband_action
-----------------------------
 t
(1 row)
```

步骤9 查询query_band关联的负载行为。

```
SELECT * FROM pg_queryband_action;
    qband     | respool_id | respool | priority | qborder
--------------+------------+---------+----------+---------
 AppName=test |          0 | NULL    | low      |      -1
(1 row)
```

----结束

14.5 SQL 调优案例

14.5.1 案例：选择合适的分布列

分布列用于将数据分布到不同的节点上，划分均衡可以避免数据倾斜。

在进行关联查询时，尽量选择查询中的关联条件作为分布键。当关联条件作为分布键时，相关数据都分布在DN本地，将减少DN之间的数据流动代价，提升查询速度。

优化前

将a作为t1和t2的分布列，表定义如下：

```
CREATE TABLE t1 (a int, b int) DISTRIBUTE BY HASH (a);
CREATE TABLE t2 (a int, b int) DISTRIBUTE BY HASH (a);
```

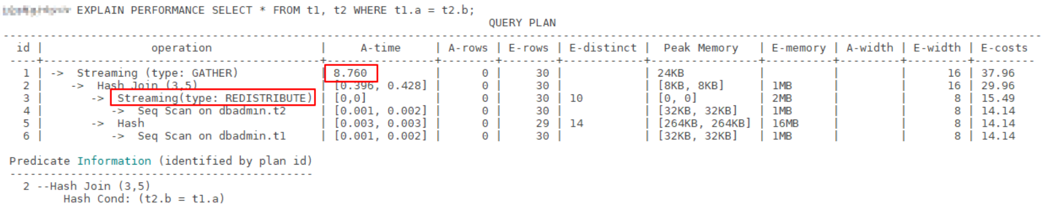
执行如下查询：

```
SELECT * FROM t1, t2 WHERE t1.a = t2.b;
```

则执行计划存在“Streaming(type: REDISTRIBUTE)”，即DN根据选定的列把数据重分布到所有的DN，这将导致DN之间存在较大通信数据量，如图14-20所示。

 版权所有 © 华为云计算技术有限公司

图 14-20 选择合适的分布列案例（一）



```
 EXPLAIN PERFORMANCE SELECT * FROM t1, t2 WHERE t1.a = t2.b;
                                                            QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 id |                 operation                 |       A-time       | A-rows | E-rows | E-distinct |  Peak Memory   | E-memory | A-width | E-width | E-costs
----+-------------------------------------------+--------------------+--------+--------+------------+----------------+----------+---------+---------+---------
  1 | ->  Streaming (type: GATHER)              | 8.760              |      0 |     30 |            | 24KB           |          |         |      16 | 37.96
  2 |    ->  Hash Join (3,5)                    | [0.396, 0.428]     |      0 |     30 |            | [8KB, 8KB]     | 1MB      |         |      16 | 29.96
  3 |       ->  Streaming(type: REDISTRIBUTE)   | [0,0]              |      0 |     30 | 10         | [0, 0]         | 2MB      |         |       8 | 15.49
  4 |          ->  Seq Scan on dbadmin.t2       | [0.001, 0.002]     |      0 |     30 |            | [32KB, 32KB]   | 1MB      |         |       8 | 14.14
  5 |       ->  Hash                            | [0.003, 0.003]     |      0 |     29 | 14         | [264KB, 264KB] | 16MB     |         |       8 | 14.14
  6 |          ->  Seq Scan on dbadmin.t1       | [0.001, 0.002]     |      0 |     30 |            | [32KB, 32KB]   | 1MB      |         |       8 | 14.14

 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Hash Join (3,5)
         Hash Cond: (t2.b = t1.a)
```

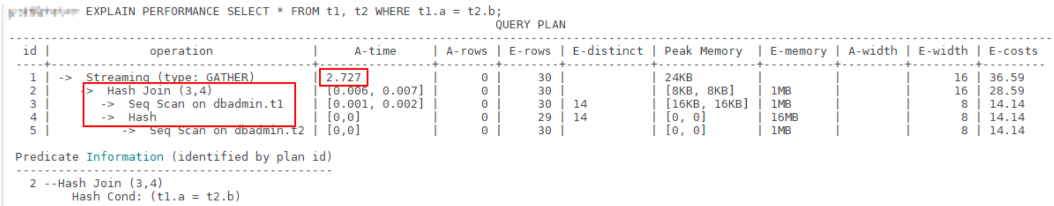
优化后

将查询中的关联条件作为分布键，执行下列语句修改b作为t2的分布列：

```
ALTER TABLE t2 DISTRIBUTE BY HASH (b);
```

将表t2的分布列改为b列之后，执行计划将不再包含“Streaming(type: REDISTRIBUTE)”，减少了DN之间存在的通信数据量的同时，执行时间也从8.7毫秒降低至2.7毫秒，从而提升查询性能，如图14-21所示。

图 14-21 选择合适的分布列案例（二）



```
 EXPLAIN PERFORMANCE SELECT * FROM t1, t2 WHERE t1.a = t2.b;
                                                            QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 id |                operation                |       A-time       | A-rows | E-rows | E-distinct |  Peak Memory   | E-memory | A-width | E-width | E-costs
----+-----------------------------------------+--------------------+--------+--------+------------+----------------+----------+---------+---------+---------
  1 | ->  Streaming (type: GATHER)            | 2.727              |      0 |     30 |            | 24KB           |          |         |      16 | 36.59
  2 |    ->  Hash Join (3,4)                  | [0.006, 0.007]     |      0 |     30 |            | [8KB, 8KB]     | 1MB      |         |      16 | 28.59
  3 |       ->  Seq Scan on dbadmin.t1        | [0.001, 0.002]     |      0 |     30 | 14         | [16KB, 16KB]   | 1MB      |         |       8 | 14.14
  4 |       ->  Hash                          | [0,0]              |      0 |     29 | 14         | [0, 0]         | 16MB     |         |       8 | 14.14
  5 |          ->  Seq Scan on dbadmin.t2     | [0,0]              |      0 |     30 |            | [0, 0]         | 1MB      |         |       8 | 14.14

 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Hash Join (3,4)
         Hash Cond: (t1.a = t2.b)
```

14.5.2 案例：建立合适的索引

创建合适的索引可以加速对表中数据行的检索。索引占用磁盘空间，并且降低添加、删除和更新行的速度。如果需要非常频繁地更新数据或磁盘空间有限，则需要限制索引的数量。在表较大时再建立索引，表中的数据越多，索引的优越性越明显。建议仅在匹配如下某条原则时创建索引：

- 需要经常执行查询的字段。
- 对于存在多字段连接的查询，建议在连接条件字段上建立组合索引。例如select * from t1 join t2 on t1.a=t2.a and t1.b=t2.b，可以在t1表上的a，b字段上建立组合索引。
- where子句过滤条件的字段（尤其是范围条件）。
- 经常出现在order by、group by和distinct后的字段。

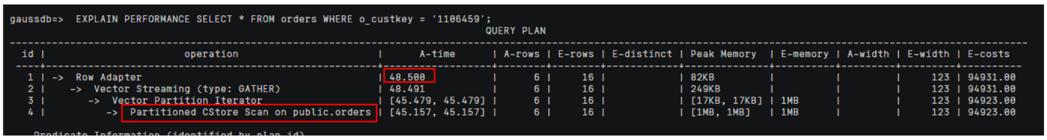
优化前

列存分区表orders表定义如下：

```
                                                  pg_get_tabledef
---------------------------------------------------------------------------------------------------------------------------
SET search_path = dbadmin;                                                                                                +
CREATE  TABLE orders (                                                                                                    +
        o_orderkey bigint NOT NULL,                                                                                       +
        o_custkey bigint NOT NULL,                                                                                        +
        o_orderstatus character(1) NOT NULL,                                                                              +
        o_totalprice numeric(15,2) NOT NULL,                                                                              +
        o_orderdate timestamp(0) without time zone NOT NULL,                                                              +
        o_orderpriority character(15) NOT NULL,                                                                           +
        o_clerk character(15) NOT NULL,                                                                                   +
        o_shippriority bigint NOT NULL,                                                                                   +
        o_comment character varying(79) NOT NULL                                                                          +
)                                                                                                                         +
WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)                                            +
DISTRIBUTE BY HASH(o_orderkey)                                                                                            +
TO GROUP group_version1                                                                                                   +
PARTITION BY RANGE (o_orderdate)                                                                                          +
(                                                                                                                         +
         PARTITION o_orderdate_1 VALUES LESS THAN ('1993-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_2 VALUES LESS THAN ('1994-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_3 VALUES LESS THAN ('1995-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_4 VALUES LESS THAN ('1996-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_5 VALUES LESS THAN ('1997-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_6 VALUES LESS THAN ('1998-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
         PARTITION o_orderdate_7 VALUES LESS THAN ('1999-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default +
)                                                                                                                         +
ENABLE ROW MOVEMENT;
(1 row)
```

执行SQL语句查询没有建立索引情况下的执行计划，发现执行时间为48毫秒。

```
EXPLAIN PERFORMANCE SELECT * FROM orders WHERE o_custkey = '1106459';
```



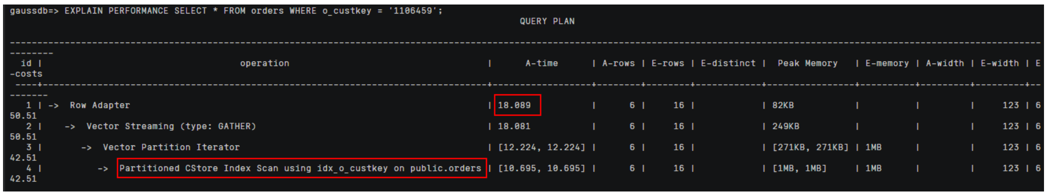
```
gaussdb=>  EXPLAIN PERFORMANCE SELECT * FROM orders WHERE o_custkey = '1106459';
                                                          QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------
 id |                  operation                     |       A-time       | A-rows | E-rows | E-distinct | Peak Memory  | E-memory | A-width | E-width | E-costs
----+------------------------------------------------+--------------------+--------+--------+------------+--------------+----------+---------+---------+---------
  1 | -> Row Adapter                                 | 48.500             |      6 |     16 |            | 82KB         |          |         |     123 | 94931.00
  2 |    -> Vector Streaming (type: GATHER)          | 48.491             |      6 |     16 |            | 249KB        |          |         |     123 | 94931.00
  3 |       -> Vector Partition Iterator             | [45.479, 45.479]   |      6 |     16 |            | [17KB, 17KB] | 1MB      |         |     123 | 94923.00
  4 |          -> Partitioned CStore Scan on public.orders | [45.157, 45.157] |   6 |     16 |            | [1MB, 1MB]   | 1MB      |         |     123 | 94923.00
```

优化后

where子句过滤条件的字段是o_custkey，在o_custkey字段上添加一个索引：

```
CREATE INDEX idx_o_custkey ON orders (o_custkey) LOCAL;
```

执行SQL语句查询建立索引后的执行计划，发现执行时间为18毫秒。



```
gaussdb=> EXPLAIN PERFORMANCE SELECT * FROM orders WHERE o_custkey = '1106459';
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
 id |                          operation                                  |       A-time       | A-rows | E-rows | E-distinct |  Peak Memory   | E-memory | A-width | E-width | E
-costs
----+---------------------------------------------------------------------+--------------------+--------+--------+------------+----------------+----------+---------+---------+--
-------
  1 | -> Row Adapter                                                      | 18.089             |      6 |     16 |            | 82KB           |          |         |     123 | 6
50.51
  2 |    -> Vector Streaming (type: GATHER)                               | 18.081             |      6 |     16 |            | 249KB          |          |         |     123 | 6
50.51
  3 |       -> Vector Partition Iterator                                  | [12.224, 12.224]   |      6 |     16 |            | [271KB, 271KB] | 1MB      |         |     123 | 6
42.51
  4 |          -> Partitioned CStore Index Scan using idx_o_custkey on public.orders | [10.695, 10.695] | 6 | 16 |          | [1MB, 1MB]     | 1MB      |         |     123 | 6
42.51
```

14.5.3 案例：增加 JOIN 列非空条件

若Join列上的NULL值较多，可以加上is not null过滤条件，以实现数据的提前过滤，提高Join效率。

优化前

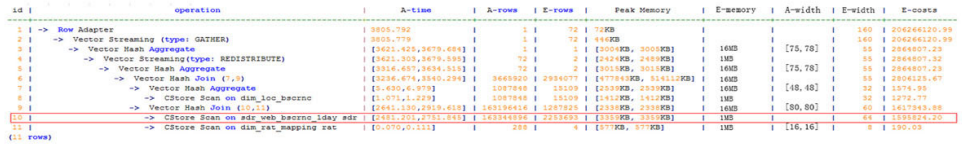
```
SELECT
 *
FROM
( ( SELECT
  STARTTIME STTIME,
  SUM(NVL(PAGE_DELAY_MSEL,0)) PAGE_DELAY_MSEL,
  SUM(NVL(PAGE_SUCCEED_TIMES,0)) PAGE_SUCCEED_TIMES,
  SUM(NVL(FST_PAGE_REQ_NUM,0)) FST_PAGE_REQ_NUM,
  SUM(NVL(PAGE_AVG_SIZE,0)) PAGE_AVG_SIZE,
  SUM(NVL(FST_PAGE_ACK_NUM,0)) FST_PAGE_ACK_NUM,
  SUM(NVL(DATATRANS_DW_DURATION,0)) DATATRANS_DW_DURATION,
  SUM(NVL(PAGE_SR_DELAY_MSEL,0)) PAGE_SR_DELAY_MSEL
 FROM
  PS.SDR_WEB_BSCRNC_1DAY SDR
```


版权所有 © 华为云计算技术有限公司

```
INNER JOIN (SELECT
    BSCRNC_ID,
    BSCRNC_NAME,
    ACCESS_TYPE,
    ACCESS_TYPE_ID
   FROM
    nethouse.DIM_LOC_BSCRNC
   GROUP BY
    BSCRNC_ID,
    BSCRNC_NAME,
    ACCESS_TYPE,
    ACCESS_TYPE_ID) DIM
 ON SDR.BSCRNC_ID = DIM.BSCRNC_ID
 AND DIM.ACCESS_TYPE_ID IN (0,1,2)
 INNER JOIN nethouse.DIM_RAT_MAPPING RAT
 ON (RAT.RAT = SDR.RAT)
WHERE
 ( (STARTTIME >= 1461340800
 AND STARTTIME < 1461427200) )
 AND RAT.ACCESS_TYPE_ID IN (0,1,2)
GROUP BY STTIME ) ) ;
```

执行计划如图14-22所示。

图 14-22 增加 JOIN 列非空条件（一）



优化后

1. 分析执行计划图14-22可知，在顺序扫描阶段耗时较多。
2. 多表JOIN中，由于表PS.SDR_WEB_BSCRNC_1DAY的JOIN列“BSCRNC_ID”存在大量空值，JOIN性能差。

   建议在语句中手动添加JOIN列的非空判断，修改后的语句如下所示。

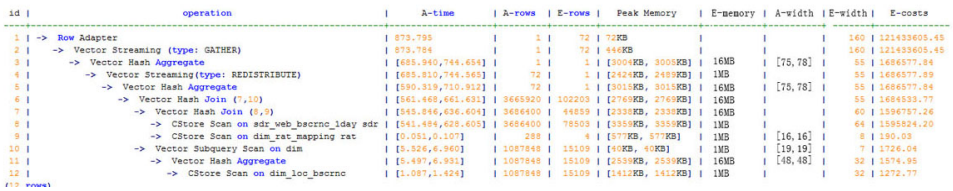
```
SELECT
 *
FROM
( ( SELECT
  STARTTIME STTIME,
  SUM(NVL(PAGE_DELAY_MSEL,0)) PAGE_DELAY_MSEL,
  SUM(NVL(PAGE_SUCCEED_TIMES,0)) PAGE_SUCCEED_TIMES,
  SUM(NVL(FST_PAGE_REQ_NUM,0)) FST_PAGE_REQ_NUM,
  SUM(NVL(PAGE_AVG_SIZE,0)) PAGE_AVG_SIZE,
  SUM(NVL(FST_PAGE_ACK_NUM,0)) FST_PAGE_ACK_NUM,
  SUM(NVL(DATATRANS_DW_DURATION,0)) DATATRANS_DW_DURATION,
  SUM(NVL(PAGE_SR_DELAY_MSEL,0)) PAGE_SR_DELAY_MSEL
 FROM
  PS.SDR_WEB_BSCRNC_1DAY SDR
  INNER JOIN (SELECT
     BSCRNC_ID,
     BSCRNC_NAME,
     ACCESS_TYPE,
     ACCESS_TYPE_ID
    FROM
     nethouse.DIM_LOC_BSCRNC
    GROUP BY
     BSCRNC_ID,
     BSCRNC_NAME,
     ACCESS_TYPE,
     ACCESS_TYPE_ID) DIM
  ON SDR.BSCRNC_ID = DIM.BSCRNC_ID
```


版权所有 © 华为云计算技术有限公司

```
 AND DIM.ACCESS_TYPE_ID IN (0,1,2)
 INNER JOIN nethouse.DIM_RAT_MAPPING RAT
 ON (RAT.RAT = SDR.RAT)
WHERE
 ( (STARTTIME >= 1461340800
 AND STARTTIME < 1461427200) )
 AND RAT.ACCESS_TYPE_ID IN (0,1,2)
 and SDR.BSCRNC_ID is not null
GROUP BY
 STTIME ) ) A;
```

执行计划如图14-23所示。

图 14-23 增加 JOIN 列非空条件（二）



14.5.4 案例：典型不下推语句整改案例集

14.5.4.1 案例：使排序下推

在做场景性能测试时，发现某场景大部分时间是CN端在做window agg，占到总执行时间95%以上，系统资源不能充分利用。研究发现该场景的特点是：将两列分别求sum作为一个子查询，外层对两列的和再求和后做trunc，然后排序。可以尝试将语句改写为子查询，使排序下推。

优化前

表结构如下所示：

```
CREATE TABLE public.test(imsi int,L4_DW_THROUGHPUT int,L4_UL_THROUGHPUT int)
with (orientation = column) DISTRIBUTE BY hash(imsi);
```

查询语句如下所示：

```
SELECT COUNT(1) over() AS DATACNT,
IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))), 0) AS
DECIMAL(20)) AS TOTAL_VOLUME_KPIID
FROM public.test AS test
GROUP BY IMSI
ORDER BY TOTAL_VOLUME_KPIID DESC LIMIT 10;
```

执行计划如下：

```
QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
-----------------------------------------
 id |                operation                |     A-time      | A-rows | E-rows | E-distinct | Peak Memory |  E-
memory  | A-width | E-width | E-costs
----+-----------------------------------------+-----------------+--------+--------+------------+-------------
+-------------+---------+---------+----------
  1 | ->  Row Adapter                         | 2862.008        |     10 |     10 |            | 31KB        |
|         |      28 | 48360.42
  2 |   ->  Vector Limit                      | 2861.969        |     10 |     10 |            | 8KB         |
|         |      28 | 48360.42
  3 |     ->  Vector Sort                     | 2861.946        |     10 | 1000000 |           | 479KB
|           |         |      28 | 50860.39
  4 |       ->  Vector WindowAgg              | 2166.759        | 1000000 | 1000000 |          | 69987KB
```


版权所有 © 华为云计算技术有限公司

```
|            |        |      28 | 26750.75
   5 |           ->  Vector Streaming (type: GATHER)  | 136.813          | 1000000 | 1000000 |            |
208KB        |            |         |      28 | 15500.75
   6 |              ->  Vector Sonic Hash Aggregate   | [71.374, 73.640] | 1000000 | 1000000 |            | [14MB,
14MB] | 96MB(2919MB) | [31,31] |      28 | 15032.00
   7 |                 ->  CStore Scan on public.test | [2.957, 2.994]   | 1000000 | 1000000 |            | [1MB,
1MB]   | 1MB          |         |      12 | 1282.00
```

可以看到window agg和sort全部在CN端执行，耗时非常严重。

优化后

尝试将语句改写为子查询：

```
SELECT COUNT(1) over() AS DATACNT, IMSI_IMSI, TOTAL_VOLUME_KPIID
FROM (SELECT IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))),
0) AS DECIMAL(20)) AS TOTAL_VOLUME_KPIID
FROM public.test AS test
GROUP BY IMSI
ORDER BY TOTAL_VOLUME_KPIID DESC LIMIT 10);
```

将trunc两列的和作为一个子查询，然后在子查询的外面做window agg，这样排序就可以下推了，执行计划如下：

```
 QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  id |                       operation                       |       A-time       | A-rows  | E-rows  | E-distinct |  Peak
Memory   |   E-memory   | A-width | E-width | E-costs
 ----+-------------------------------------------------------+--------------------+---------+---------+------------
+---------------+--------------+---------+---------+----------
   1 | ->  Row Adapter                                       | 955.277            |      10 |       5 |            | 31KB
|            |        |      24 | 25843.13
   2 |    ->  Vector WindowAgg                               | 955.261            |      10 |       5 |            | 1572KB
|            |        |      24 | 25843.13
   3 |       ->  Vector Streaming (type: GATHER)             | 955.015            |      10 |      10 |            |
127KB        |            |         |      24 | 25843.07
   4 |         ->  Vector Limit                              | [0.018, 0.018]     |      10 |      10 |            | [8KB, 8KB]     |
1MB          |        |      28 | 25836.97
   5 |           ->  Vector Streaming(type: BROADCAST)       | [0.014, 0.014]     |      20 |      20 |            |
[719KB, 719KB] | 2MB          |         |      28 | 25837.12
   6 |              ->  Vector Limit                         | [927.730, 934.283] |      20 |      20 |            | [8KB, 8KB]
| 1MB          |        |      28 | 25836.85
   7 |                 ->  Vector Sort                       | [927.720, 934.269] |      20 | 1000000 |            | [463KB,
463KB] | 16MB         | [32,32] |      28 | 27086.82
   8 |                    ->  Vector Sonic Hash Aggregate    | [456.841, 461.077] | 1000000 | 1000000 |            |
[15MB, 15MB]   | 96MB(2916MB) | [31,31] |      28 | 15032.00
   9 |                       ->  CStore Scan on public.test | [2.959, 3.014]     | 1000000 | 1000000 |            | [1MB,
1MB]     | 1MB          |         |      12 | 1282.00
```

经过SQL改写，性能由2.862s提升0.955s，优化效果明显。需注意，本示例中优化结果仅供参考，由于WindowAgg的不确定性，优化后的结果集跟实际业务相关。

14.5.4.2 案例：With-Recursive contains only values rte is not shippable

问题根因

递归语句的某个分支中没有FROM子句（只有 VALUES 或者类似 SELECT 1 这样的语句）。

案例 1：递归驱动分支没有 FROM 子句

原始语句

 版权所有 © 华为云计算技术有限公司

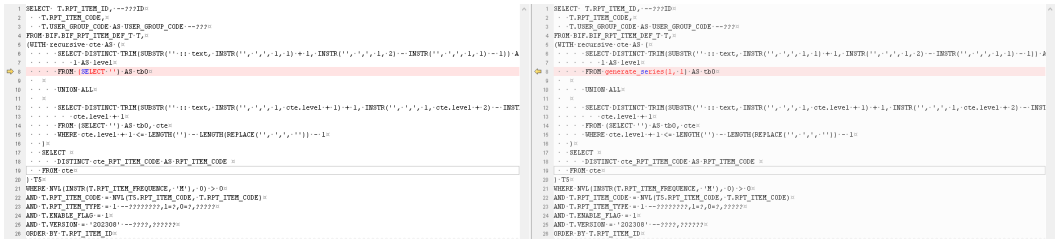
```
SELECT  T.RPT_ITEM_ID, --报表项ID
   T.RPT_ITEM_CODE,
   T.USER_GROUP_CODE AS USER_GROUP_CODE --用户组
FROM BIF.BIF_RPT_ITEM_DEF_T T,
(WITH recursive cte AS (
       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, 1) + 1, INSTR('', ',', 1, 2) - INSTR('', ',', 1, 1) -
1)) AS cte_RPT_ITEM_CODE,
          1 AS level
       FROM (SELECT '') AS tb0

       UNION ALL

       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, cte.level + 1) + 1, INSTR('', ',', 1, cte.level + 2) -
INSTR('', ',', 1, cte.level + 1) - 1)),
          cte.level + 1
       FROM (SELECT '') AS tb0, cte
       WHERE cte.level + 1 <= LENGTH('') - LENGTH(REPLACE('', ',', '')) - 1
   )
   SELECT
      DISTINCT cte_RPT_ITEM_CODE AS RPT_ITEM_CODE
   FROM cte
) T5
WHERE NVL(INSTR(T.RPT_ITEM_FREQUENCE, 'M'), 0) > 0
AND T.RPT_ITEM_CODE = NVL(T5.RPT_ITEM_CODE, T.RPT_ITEM_CODE)
AND T.RPT_ITEM_TYPE = 1 --是否是叶子报表项，1=是，0=否，基本报表项
AND T.ENABLE_FLAG = 1
AND T.VERSION = '202308' --使用快照,增加条件限制
ORDER BY T.RPT_ITEM_ID
```

改写语句

```
SELECT  T.RPT_ITEM_ID, --报表项ID
   T.RPT_ITEM_CODE,
   T.USER_GROUP_CODE AS USER_GROUP_CODE --用户组
FROM BIF.BIF_RPT_ITEM_DEF_T T,
(WITH recursive cte AS (
       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, 1) + 1, INSTR('', ',', 1, 2) - INSTR('', ',', 1, 1) -
1)) AS cte_RPT_ITEM_CODE,
          1 AS level
       FROM generate_series(1, 1) AS tb0

       UNION ALL

       SELECT DISTINCT TRIM(SUBSTR('' :: text, INSTR('', ',', 1, cte.level + 1) + 1, INSTR('', ',', 1, cte.level + 2) -
INSTR('', ',', 1, cte.level + 1) - 1)),
          cte.level + 1
       FROM (SELECT '') AS tb0, cte
       WHERE cte.level + 1 <= LENGTH('') - LENGTH(REPLACE('', ',', '')) - 1
   )
   SELECT
      DISTINCT cte_RPT_ITEM_CODE AS RPT_ITEM_CODE
   FROM cte
) T5
WHERE NVL(INSTR(T.RPT_ITEM_FREQUENCE, 'M'), 0) > 0
AND T.RPT_ITEM_CODE = NVL(T5.RPT_ITEM_CODE, T.RPT_ITEM_CODE)
AND T.RPT_ITEM_TYPE = 1 --是否是叶子报表项，1=是，0=否，基本报表项
AND T.ENABLE_FLAG = 1
AND T.VERSION = '202308' --使用快照,增加条件限制
ORDER BY T.RPT_ITEM_ID
```

修改点比对

版权所有 © 华为云计算技术有限公司



案例 2：递归驱动分支没有 FROM 子句

原始语句

```
SELECT A.DYNM_COMP_ID,
    DECODE(B.LINE_NO, 1, '202308', A.VERSION)
FROM BIF.BIF_DYNM_COMP_SOU_TBL_V A,
(WITH recursive cte AS (
    SELECT 1 AS level
    UNION ALL
    SELECT cte.level + 1
    FROM cte
    WHERE cte.level + 1 < 3
    )
    SELECT
        level as LINE_NO
    FROM cte
) B
WHERE EXISTS (SELECT 1
    FROM BIF.BIF_RPT_ITEM_DEF_MV RPT,
    BIF.BIF_PUB_SUBJECT_AREA_T SBJ,
    BIF.BIF_SNAPSHORT_SUBJECT_V TYP
    WHERE A.DYNM_COMP_ID = RPT.DYNM_COMP_ID
    AND RPT.VERSION = 'current'
    AND RPT.SUBJECT_AREA_ID = SBJ.SUBJECT_AREA_ID
    AND SBJ.SUBJECT_AREA_CODE =TYP.SUBJECT_CODE
    AND TYP.SUBJECT_TYPE ='TAX')
AND A.VERSION = 'current'
```

改写语句

```
SELECT A.DYNM_COMP_ID,
    DECODE(B.LINE_NO, 1, '202308', A.VERSION)
FROM BIF.BIF_DYNM_COMP_SOU_TBL_V A,
(SELECT
    *
    FROM generate_series(1, 2) AS cte(LINE_NO)
) B
WHERE EXISTS (SELECT 1
    FROM BIF.BIF_RPT_ITEM_DEF_MV RPT,
    BIF.BIF_PUB_SUBJECT_AREA_T SBJ,
    BIF.BIF_SNAPSHORT_SUBJECT_V TYP
    WHERE A.DYNM_COMP_ID = RPT.DYNM_COMP_ID
    AND RPT.VERSION = 'current'
    AND RPT.SUBJECT_AREA_ID = SBJ.SUBJECT_AREA_ID
    AND SBJ.SUBJECT_AREA_CODE =TYP.SUBJECT_CODE
    AND TYP.SUBJECT_TYPE ='TAX')
AND A.VERSION = 'current'
```

修改点比对

 版权所有 © 华为云计算技术有限公司

案例 3：递归驱动分支是 VALUES 子句

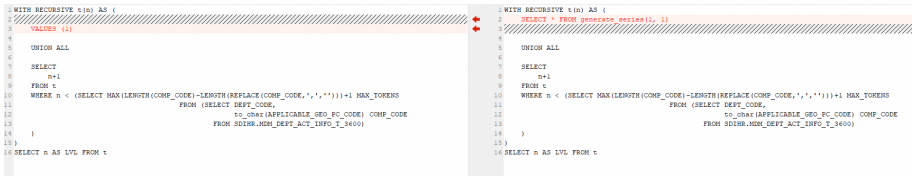
原始语句

```
WITH RECURSIVE t(n) AS (
    VALUES (1)

    UNION ALL

    SELECT
        n+1
    FROM t
    WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
                                FROM (SELECT DEPT_CODE,
                                        to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
                                    FROM SDIHR.MDM_DEPT_ACT_INFO_T_3600)
    )
)
SELECT n AS LVL FROM t
```

改写语句

```
WITH RECURSIVE t(n) AS (
    SELECT * FROM generate_series(1, 1)

    UNION ALL

    SELECT
        n+1
    FROM t
    WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
                                FROM (SELECT DEPT_CODE,
                                        to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
                                    FROM SDIHR.MDM_DEPT_ACT_INFO_T_3600)
    )
)
SELECT n AS LVL FROM t
```

修改点比对



案例 4：递归驱动分支是 VALUES 子句

原始语句

```
WITH RECURSIVE t(n) AS (VALUES (1)
    UNION ALL
    SELECT n+1
    FROM t
    WHERE n < (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1
MAX_TOKENS
    FROM (SELECT DEPT_CODE,
    to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
    FROM SDIHR.MDM_DEPT_ACT_INFO_T_3600))
) SELECT n AS LVL FROM t
```

改写语句

```
SELECT * FROM generate_series
    (1, (SELECT MAX(LENGTH(COMP_CODE)-LENGTH(REPLACE(COMP_CODE,',','')))+1 MAX_TOKENS
```

 版权所有 © 华为云计算技术有限公司

```
   FROM (SELECT DEPT_CODE,
   to_char(APPLICABLE_GEO_PC_CODE) COMP_CODE
   FROM SDIHR.MDM_DEPT_ACT_INFO_T_3600))
) AS t(lvl)
```

修改点比对

14.5.4.3 案例：With-Recursive contains system table is not shippable

问题根因

递归语句的某个分支中没有完整的From子句，仅使用系统表或者系统视图(DUAL也被视为系统视图)代替。

案例 1：递归驱动分支是 FROM DUAL 查询

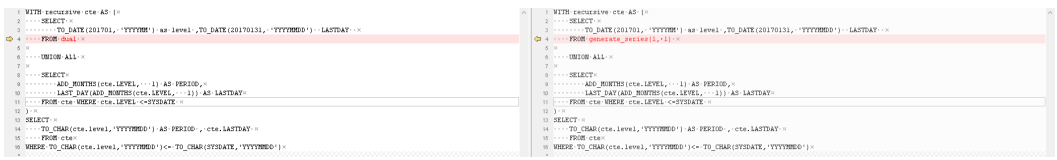
原始语句

```
WITH recursive cte AS (
   SELECT
      TO_DATE(201701, 'YYYYMM') as level ,TO_DATE(20170131, 'YYYYMMDD')  LASTDAY
   FROM dual

   UNION ALL

   SELECT
      ADD_MONTHS(cte.LEVEL,   1) AS PERIOD,
      LAST_DAY(ADD_MONTHS(cte.LEVEL,   1)) AS LASTDAY
   FROM cte WHERE cte.LEVEL <=SYSDATE
)
SELECT
   TO_CHAR(cte.level,'YYYYMMDD') AS PERIOD , cte.LASTDAY
   FROM cte
WHERE TO_CHAR(cte.level,'YYYYMMDD')<= TO_CHAR(SYSDATE,'YYYYMMDD')
```

改写语句

```
WITH recursive cte AS (
   SELECT
      TO_DATE(201701, 'YYYYMM') as level ,TO_DATE(20170131, 'YYYYMMDD')  LASTDAY
   FROM generate_series(1, 1)

   UNION ALL

   SELECT
      ADD_MONTHS(cte.LEVEL,   1) AS PERIOD,
      LAST_DAY(ADD_MONTHS(cte.LEVEL,   1)) AS LASTDAY
   FROM cte WHERE cte.LEVEL <=SYSDATE
)
SELECT
   TO_CHAR(cte.level,'YYYYMMDD') AS PERIOD , cte.LASTDAY
   FROM cte
WHERE TO_CHAR(cte.level,'YYYYMMDD')<= TO_CHAR(SYSDATE,'YYYYMMDD')
```

修改点对比



 版权所有 © 华为云计算技术有限公司

14.5.4.4 案例：SubPlan exec on CN can not be shipped

问题根因

某个子查询语句只能在CN上执行，通常是这个子查询有不下推因素，比如有调用了系统表、系统视图，或者存在不下推函数等。

案例 1：子查询过程中存在系统表/系统视图的查询

原始语句

```
WITH error_log  AS NOT MATERIALIZED (
   SELECT
      upper(log_column_name) AS log_column_name,
      log_error_code,
      s.char_length AS data_length,
      s.data_type,s.nullable
   FROM (SELECT
         distinct
         unnest(string_to_array(bad_log_column_name,',')) AS log_column_name,
         unnest(string_to_array(bad_log_error_code,',')) AS log_error_code
      FROM stgltc.BAD_cfs_inv_invoice_ad_2500
   ) T,
   (SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')) S
   WHERE  upper(T.log_column_name)=upper(S.column_name)
)

SELECT
   CASE WHEN upper('ACTIVITY_NAME') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ACTIVITY_NAME,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ACTIVITY_NAME')))
      ELSE ACTIVITY_NAME
   END AS ACTIVITY_NAME,
   CASE WHEN upper('ADJUSTMENT_ID') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
      THEN SUBSTRB(ADJUSTMENT_ID,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ADJUSTMENT_ID')))
      ELSE ADJUSTMENT_ID
   END AS ADJUSTMENT_ID
FROM stgltc.BAD_cfs_inv_invoice_ad_2500
```

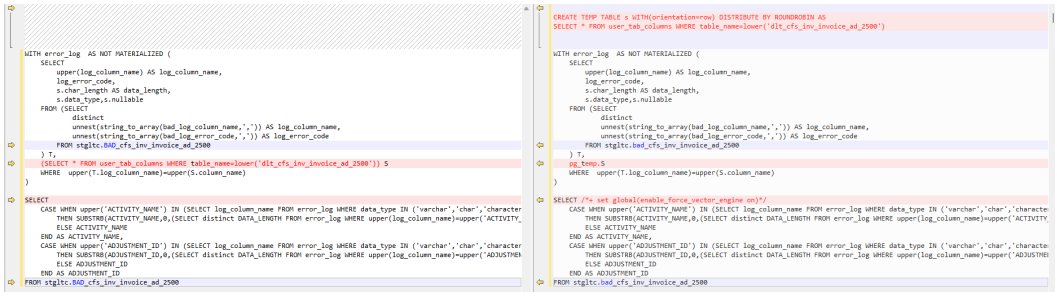
改写语句

```
-- 识别不下推的子查询为WITH error_log子句中的
-- SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')
--
-- 因为这部分为系统表查询，无论如何都不能下推，所以此处把这部分结果转储到一个中间表中
-- 中间表创建成行存表
CREATE TEMP TABLE s WITH(orientation=row) DISTRIBUTE BY ROUNDROBIN AS
SELECT * FROM user_tab_columns WHERE table_name=lower('dlt_cfs_inv_invoice_ad_2500')

-- 因为整个查询涉及到的表都是列存表，之后前面创建的临时表s为行存表
-- 所以此处加一个强制走向量化的hint
WITH error_log  AS NOT MATERIALIZED (
   SELECT
      upper(log_column_name) AS log_column_name,
      log_error_code,
      s.char_length AS data_length,
      s.data_type,s.nullable
   FROM (SELECT
         distinct
         unnest(string_to_array(bad_log_column_name,',')) AS log_column_name,
         unnest(string_to_array(bad_log_error_code,',')) AS log_error_code
      FROM stgltc.bad_cfs_inv_invoice_ad_2500
```


版权所有 © 华为云计算技术有限公司

```
    ) T,
    pg_temp.S
    WHERE  upper(T.log_column_name)=upper(S.column_name)
)

SELECT /*+ set global(enable_force_vector_engine on)*/
    CASE WHEN upper('ACTIVITY_NAME') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
        THEN SUBSTRB(ACTIVITY_NAME,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ACTIVITY_NAME')))
        ELSE ACTIVITY_NAME
    END AS ACTIVITY_NAME,
    CASE WHEN upper('ADJUSTMENT_ID') IN (SELECT log_column_name FROM error_log WHERE data_type
IN ('varchar','char','character','nchar','character varying','varchar2','nvarchar2','clob','text') AND
log_error_code='22001'/*字符超长*/)
        THEN SUBSTRB(ADJUSTMENT_ID,0,(SELECT distinct DATA_LENGTH FROM error_log WHERE
upper(log_column_name)=upper('ADJUSTMENT_ID')))
        ELSE ADJUSTMENT_ID
    END AS ADJUSTMENT_ID
FROM stgltc.bad_cfs_inv_invoice_ad_2500
```

修改点对比



14.5.4.5 案例：Type of Record in TargetList can not be shipped

问题根因

输出列中存在record类型，这种类型的不下推一般是不会体现在最外层的输出列上，一般这类报错有两个场景：

1. SQL书写逻辑错误，导致输出列上出现了（ ... ）形式的输出列。
2. SQL业务逻辑正确， 这种场景需要了解业务含义，把record字段强转为text类型，再使用record字段的地方做特殊适配。

案例 1：输出列书写错误，出现（ ... ）形式的输出列

原始语句

```
SELECT
    d.id,
    coalesce(d.period, 'snull') AS period,
    (d.plan_unit_code, 'snull') AS plan_unit_code,
    coalesce(d.product_type_model, 'snull') AS product_type_model,
    coalesce(d.revision, 'snull') AS revision,
    d.start_date
FROM (SELECT *
    FROM cdcscm.cdc_mp_d_forecast_t_6120 t
    WHERE t.cdc_timestamp > to_date('2023-07-06 00:00:00', 'yyyy-mm-dd hh24:mi:ss') - 7
    AND t.cdc_timestamp < to_date('2023-08-08 00:00:00', 'yyyy-mm-dd hh24:mi:ss')
) t1,
sdiscm.mp_d_forecast_t_6120 d
```


版权所有 © 华为云计算技术有限公司

```
WHERE (t1.audit_op_type = 'delete' AND t1.audit_op_option = 'before')
AND d.id = t1.id
```

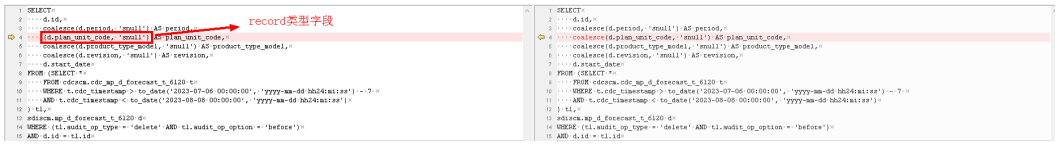
改写语句

```
SELECT
    d.id,
    coalesce(d.period, 'snull') AS period,
    coalesce(d.plan_unit_code, 'snull') AS plan_unit_code,
    coalesce(d.product_type_model, 'snull') AS product_type_model,
    coalesce(d.revision, 'snull') AS revision,
    d.start_date
FROM (SELECT *
    FROM cdcscm.cdc_mp_d_forecast_t_6120 t
    WHERE t.cdc_timestamp > to_date('2023-07-06 00:00:00', 'yyyy-mm-dd hh24:mi:ss') - 7
    AND t.cdc_timestamp < to_date('2023-08-08 00:00:00', 'yyyy-mm-dd hh24:mi:ss')
) t1,
sdiscm.mp_d_forecast_t_6120 d
WHERE (t1.audit_op_type = 'delete' AND t1.audit_op_option = 'before')
AND d.id = t1.id
```

说明

改写前后的语句不等价，不等价是因为原始SQL书写有问题，正确的写法为：
coalesce(d.plan_unit_code, 'snull') AS plan_unit_code

修改点对比



14.5.4.6 案例：Type of Record in non-real table can not be shipped

问题根因

8.2.1.x特定版本下UPDATE语句+WITH语句场景存在下推问题。

案例 1：UPDATE 语句不下推

1. 通过GUC参数**enable_stream_ctescan**设置stream计划下不支持ctescan，在语句执行前进行设置。
   ```
   SET enable_stream_ctescan = off
   ```
2. 业务语句执行完再重置参数。
   ```
   SET enable_stream_ctescan = on
   ```
3. 语句中UPDATE关键字后加hint，具体方式为 UPDATE /*+ set global(enable_stream_ctescan off) */

14.5.4.7 案例：main plan exec on CN and SubPlan exec on DN can not be shipped

问题根因

根因：复杂语句中的主查询为不涉及用户表的查询，但相关的子查询是涉及用户表的查询语句。

版权所有 © 华为云计算技术有限公司

案例 1：主查询来源表是维表 DUAL

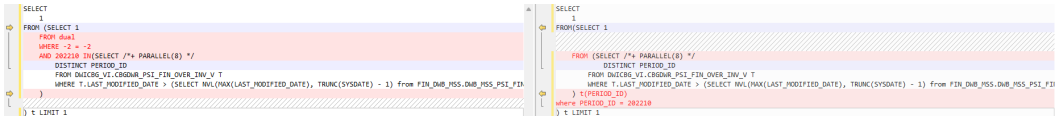
原始语句

```
SELECT
    1
FROM (SELECT 1
    FROM dual
    WHERE -2 = -2
    AND 202210 IN(SELECT /*+ PARALLEL(8) */
        DISTINCT PERIOD_ID
        FROM DWICBG_VI.CBGDWR_PSI_FIN_OVER_INV_V T
        WHERE T.LAST_MODIFIED_DATE > (SELECT NVL(MAX(LAST_MODIFIED_DATE), TRUNC(SYSDATE) - 1)
from FIN_DWB_MSS.DWB_MSS_PSI_FIN_OVER_INV_M_F) AND T.LAST_MODIFIED_DATE <= (SELECT
TRUNC(SYSDATE) + 1)
    )
) t LIMIT 1
```

改写后语句

```
SELECT
    1
FROM(SELECT 1
    FROM (SELECT /*+ PARALLEL(8) */
            DISTINCT PERIOD_ID
        FROM DWICBG_VI.CBGDWR_PSI_FIN_OVER_INV_V T
        WHERE T.LAST_MODIFIED_DATE > (SELECT NVL(MAX(LAST_MODIFIED_DATE), TRUNC(SYSDATE) - 1)
from FIN_DWB_MSS.DWB_MSS_PSI_FIN_OVER_INV_M_F) AND T.LAST_MODIFIED_DATE <= (SELECT
TRUNC(SYSDATE) + 1)
    ) t(PERIOD_ID)
where PERIOD_ID = 202210
) t LIMIT 1
```

修改点对比



14.5.4.8 案例：With-Recursive under multi-nodegroup scenario is not shippable

问题根因

查询语句中所涉及的表分属于多个不同的逻辑集群中，这种场景的处理方案一般是通过中间表的方式让所有表在一个逻辑集群，或者把WITH RECURSIVE语句隔离出来以避免影响这个语句的性能。

案例 1：通过临时表把 WITH RECURSIVE 语句中的非当前逻辑集群的表转储到当前逻辑集群中

适用场景：语句中包含WITH RECURSIVE语句，且WITH RECURSIVE只有少部分（1~2个）表分属于其他逻辑集群，这些表数据量不大或者经过过滤条件之后结果集不大。

原始语句如下，语句中目标表dwltax.dwl_tax_tom_cit_tp_account_tmp在逻辑集群"LC_DW1"中，来源表dwtaxdi.dwi_tax_cit_tp_account_r_i中涉及的表来源于逻辑集群“LC_DL1”。

```
INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)
SELECT
    S.account_id,S.parentid
```


版权所有 © 华为云计算技术有限公司

```
FROM (SELECT
      DISTINCT S.ACCOUNT_ID, S.PARENTID
   FROM (
      WITH RECURSIVE tmp AS (
         SELECT 1 AS LV,
            '/'||T.PARENTID AS path,
            account_ID AS account_ID,
            PARENTID AS PARENTID ,
            ROW_number() over() PATH_NUM,
            T.DEL_FLAG
         FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
         WHERE T.DEL_FLAG = 'N'
         AND T.ISLEAF_FLAG = 1

         UNION ALL

         SELECT 1+lv AS LV,
            p.path||'/'||t.PARENTID AS path,
            p.account_ID,
            t.PARENTID AS PARENTID,
            ROW_number() over() PATH_NUM,
            T.DEL_FLAG
         FROM dwtaxdi.dwi_tax_cit_tp_account_r_i t
         INNER JOIN tmp p ON p.PARENTID = t.account_ID
      )

      SELECT lv,
         '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
         account_ID,
         PARENTID,
         PATH_NUM
      FROM tmp
      WHERE DEL_FLAG='N'
   ) S
) S
```

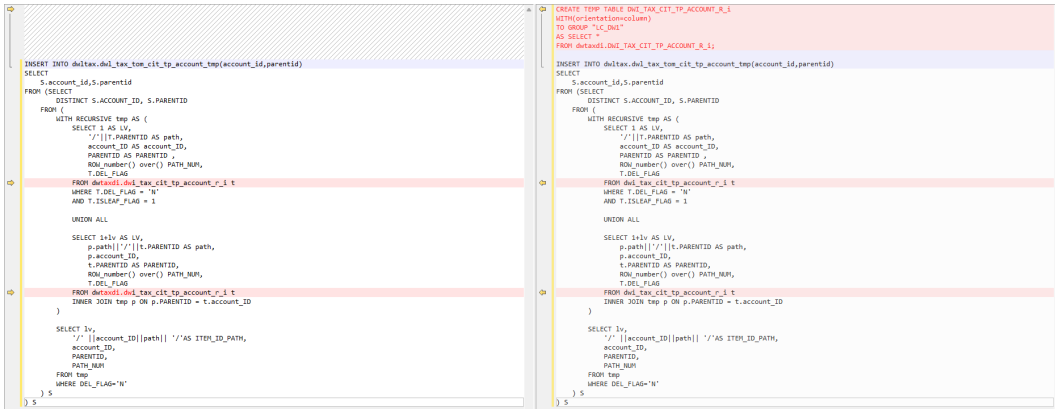
改写后语句

```
CREATE TEMP TABLE DWI_TAX_CIT_TP_ACCOUNT_R_i -- 把来源表数据导入临时表，临时表在逻辑集群
“LC_DW1”
WITH(orientation=column)
TO GROUP "LC_DW1"
AS SELECT *
FROM dwtaxdi.DWI_TAX_CIT_TP_ACCOUNT_R_i;

INSERT INTO dwltax.dwl_tax_tom_cit_tp_account_tmp(account_id,parentid)  -- 来源表和目标表都在逻辑集群
“LC_DW1”
SELECT
   S.account_id,S.parentid
FROM (SELECT
      DISTINCT S.ACCOUNT_ID, S.PARENTID
   FROM (
      WITH RECURSIVE tmp AS (
         SELECT 1 AS LV,
            '/'||T.PARENTID AS path,
            account_ID AS account_ID,
            PARENTID AS PARENTID ,
            ROW_number() over() PATH_NUM,
            T.DEL_FLAG
         FROM dwi_tax_cit_tp_account_r_i t
         WHERE T.DEL_FLAG = 'N'
         AND T.ISLEAF_FLAG = 1

         UNION ALL

         SELECT 1+lv AS LV,
            p.path||'/'||t.PARENTID AS path,
            p.account_ID,
            t.PARENTID AS PARENTID,
            ROW_number() over() PATH_NUM,
```

 版权所有 © 华为云计算技术有限公司

```
            T.DEL_FLAG
        FROM dwi_tax_cit_tp_account_r_i t
        INNER JOIN tmp p ON p.PARENTID = t.account_ID
    )

    SELECT lv,
        '/' ||account_ID||path|| '/'AS ITEM_ID_PATH,
        account_ID,
        PARENTID,
        PATH_NUM
    FROM tmp
    WHERE DEL_FLAG='N'
  ) S
) S
```

修改点对比



优化后的执行计划

```
id |                                                                operation                                                                 |  E-rows  | E-distinct | E-memory | E-width | E-costs
---+------------------------------------------------------------------------------------------------------------------------------------------+----------+------------+----------+---------+---------
 1 |  ->  Row Adapter                                                                                                                         |        1 |            |          |      64 | 92487.39
 2 |    ->  Vector Streaming (type: GATHER)                                                                                                   |        1 |            |          |      64 | 92487.39
 3 |      ->  Vector Insert on dwltax.dwl_tax_tom_cit_tp_account_tmp                                                                          |       72 |            | 16MB     |      64 | 92485.11
 4 |        ->  Vector Adapter                                                                                                                |       72 |            | 1MB      |      64 | 92483.10
 5 |          ->  Streaming(type: ROUNDROBIN)                                                                                                 |       72 |            | 2MB      |      64 | 92483.10
 6 |            ->  HashAggregate                                                                                                             |        1 |            | 16MB     |      64 | 92480.84
 7 |              ->  Result                                                                                                                  |        1 |            | 1MB      |       0 | 92480.81
 8 |                ->  Streaming(type: BROADCAST)                                                                                            |        1 |            | 2MB      |       0 | 0.30
 9 |                  ->  CTE Scan on tmp                                                                                                     |        1 |            | 1MB      |       0 | 0.02
10 |                    ->  Recursive Union (11,15)                                                                                           |  2027304 |            |          |      78 | 92480.51
11 |                      ->  WindowAgg                                                                                                       |    33624 |            |          |      18 | 838.02
12 |                        ->  Streaming(type: BROADCAST)                                                                                    |  2421648 |            |          |      18 | 829.85
13 |                          ->  Row Adapter                                                                                                 |    33624 |            |          |      18 | 534.32
14 |                            ->  CStore Scan on pg_temp_cn_5010_2869_708_140584651829760.dwi_tax_cit_tp_account_r_i t                      |    33634 |            |          |      18 | 534.32
15 |                      ->  WindowAgg                                                                                                       |   199368 |            |          |      78 | 9107.93
16 |                        ->  Streaming(type: BROADCAST) stream_level:1                                                                     | 14354280 |            |          |      78 | 9045.63
17 |                          ->  Hash Join (18,19) <<ruid:[10] ctlid:[10] (SYNC)>>                                                           |   199365 |            |          |      78 | 1452.52
18 |                            ->  WorkTable Scan on tmp p                                                                                   | 24209280 |  67248     |          |     100 | 38.64
19 |                            ->  Hash                                                                                                      |    39888 |  554       |          |      18 | 531.55
20 |                              ->  Row Adapter                                                                                             |    39888 |            |          |      18 | 531.55
21 |                                ->  CStore Scan on pg_temp_cn_5010_2869_708_140584651829760.dwi_tax_cit_tp_account_r_i t                  |    39873 |            |          |      18 | 531.55

   SQL Diagnostic Information
-------------------------------
CTE is inlined
    There are recursive CTEs in query
```

案例 2：把 WITH RECURSIVE 语句部分的结果转储到临时表

适用场景：当WITH RECURSIVE部分结果集比较小，且上一种整改方案的结果集仍比较大的时候，常见的有以下场景

- WITH RECURSIVE语句涉及到较多其他逻辑集中的表，转储比较麻烦。
- 当前语句没有直接调用WITH RECURSIVE语句，但是调用的视图中包含WITH RECURSIVE逻辑。

原始语句

```
INSERT INTO
dwltax.bif_rpt_item_htp_tax_t(PERIOD_ID,REPORT_ITEM_ID,CRT_CYCLE_ID,LAST_UPD_CYCLE_ID,UPD_JOB_IN
```

 版权所有 © 华为云计算技术有限公司

```
STANCE_ID, DEL_FLAG, CRT_JOB_INSTANCE_ID, UCCID )
SELECT
   DISTINCT T.PERIOD_ID ,158004503,-1,-1,-1,'N',-1,T.UCCID
FROM dwltax.dwl_tax_tom_taxpln_htp_tag_v T
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_account_v  TAX_E1502044 ON T.ACCOUNT_ID =
TAX_E1502044.ACCOUNT_ID
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_rf_tag_v  TAX_E1502045 ON T.RF_ID =  TAX_E1502045.RF_ID
WHERE  1 = 1
AND T.SOURCE_SYSTEM = 'HWHTPLOC01'
AND TAX_E1502044.PARENTID_ACCOUNT_CODE IN
('TARFAdjustmentsManual','TARFAdjustmentsExpenseDeferredReclass')
AND TAX_E1502045.PARENTID_RF_CODE = 'TARFTotal'
AND T.DATACATEGORY_CODE IN ('TARFDeferredNonCurrentLossNR')
AND T.PERIOD_ID=202308
```

分析发现如上语句的dwltax.dwl_tax_tom_cit_tp_rf_tag_v包含WITH RECURSIVE逻辑。

```
CREATE OR REPLACE VIEW dwltax.dwl_tax_tom_cit_tp_rf_tag_v AS  SELECT d2.rf_id AS parent_rf_id,
   d2.label AS parentid_rf_code, d1.rf_id, d1.label AS rf_code,
   d1.userdefined1, d1.userdefined2, d1.userdefined3, d1.en_us, d1.en_gb,
   d2.en_us AS parent_en_us, d2.en_gb AS parent_en_gb, d2.source_system
FROM (SELECT
      DISTINCT r1.rf_id, r1.parentid
   FROM (SELECT
            DISTINCT s.rf_id, s.parentid
         FROM (WITH RECURSIVE tmp AS (SELECT 1 AS lv,
               '/'::text || t.parentid::text AS path,
               t.rf_id, t.parentid
            FROM dwtaxdi.dwi_tax_cit_tp_rf_r_i t1
            RIGHT JOIN dwtaxdi.dwi_tax_cit_tp_rf_r_i t ON t.rf_id = t1.rf_id AND t.parentid = t1.parentid AND
t1.del_flag::text = 'N'::text
            WHERE t.del_flag::text = 'N'::text AND t.isleaf_flag = 1::numeric AND 1 = 1 AND 1 = 1 AND 1 = 1

            UNION ALL

            SELECT 1 + p.lv AS lv,
               (p.path || '/'::text) || t.parentid::text AS path,
               p.rf_id, t.parentid
            FROM tmp p, dwtaxdi.dwi_tax_cit_tp_rf_r_i t1
            RIGHT JOIN dwtaxdi.dwi_tax_cit_tp_rf_r_i t ON t.rf_id = t1.rf_id AND t.parentid = t1.parentid AND
t1.del_flag::text = 'N'::text
            WHERE p.parentid = t.rf_id AND t.del_flag::text = 'N'::text AND 1 = 1 AND 1 = 1 AND 1 = 1
         )

         SELECT (('/'::text || tmp.rf_id::text) || tmp.path) || '/'::text AS item_id_path,
            tmp.rf_id, tmp.parentid
         FROM tmp
      ) s

      UNION ALL

      SELECT r.rf_id, r.rf_id AS parentid
      FROM dwtaxdi.dwi_tax_cit_tp_rf_r_i r
      WHERE r.isleaf_flag = 1::numeric AND r.del_flag::text = 'N'::text) r1
) r, dwtaxdi.dwi_tax_cit_tp_rf_i d1, dwtaxdi.dwi_tax_cit_tp_rf_i d2
WHERE r.rf_id = d1.rf_id AND r.parentid = d2.rf_id;
```

视图dwltax.dwl_tax_tom_cit_tp_rf_tag_v的结果集只有1w行左右，可以尝试把事务dwltax.dwl_tax_tom_cit_tp_rf_tag_v的结果集转储到中间表，从而保证主查询可以下推。

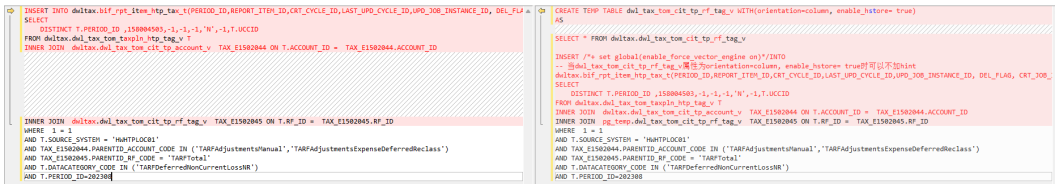
优化后语句：

```
-- 8.2.1及以上版本
CREATE TEMP TABLE dwl_tax_tom_cit_tp_rf_tag_v WITH(orientation=column, enable_hstore= true)
AS
SELECT * FROM dwltax.dwl_tax_tom_cit_tp_rf_tag_v
```


版权所有 © 华为云计算技术有限公司

```
-- 8.2.0及以下版本
CREATE TEMP TABLE dwl_tax_tom_cit_tp_rf_tag_v WITH(orientation=row)
AS
SELECT * FROM dwltax.dwl_tax_tom_cit_tp_rf_tag_v

INSERT /*+ set global(enable_force_vector_engine on)*/INTO
-- 当dwl_tax_tom_cit_tp_rf_tag_v属性为orientation=column, enable_hstore= true时可以不加hint
dwltax.bif_rpt_item_htp_tax_t(PERIOD_ID,REPORT_ITEM_ID,CRT_CYCLE_ID,LAST_UPD_CYCLE_ID,UPD_JOB_IN
STANCE_ID, DEL_FLAG, CRT_JOB_INSTANCE_ID, UCCID )
SELECT
    DISTINCT T.PERIOD_ID ,158004503,-1,-1,-1,'N',-1,T.UCCID
FROM dwltax.dwl_tax_tom_taxpln_htp_tag_v T
INNER JOIN  dwltax.dwl_tax_tom_cit_tp_account_v  TAX_E1502044 ON T.ACCOUNT_ID =
TAX_E1502044.ACCOUNT_ID
INNER JOIN  pg_temp.dwl_tax_tom_cit_tp_rf_tag_v  TAX_E1502045 ON T.RF_ID =  TAX_E1502045.RF_ID
WHERE  1 = 1
AND T.SOURCE_SYSTEM = 'HWHTPLOC01'
AND TAX_E1502044.PARENTID_ACCOUNT_CODE IN
('TARFAdjustmentsManual','TARFAdjustmentsExpenseDeferredReclass')
AND TAX_E1502045.PARENTID_RF_CODE = 'TARFTotal'
AND T.DATACATEGORY_CODE IN ('TARFDeferredNonCurrentLossNR')
AND T.PERIOD_ID=202308
```

修改点对比



14.5.4.9 案例：With-Recursive contains conflict distribution in none-recursive(xxx) recursive(xxx)

问题根因

非递归部分语句（UNION ALL前面部分）的执行计划为replicate计划，递归部分语句（UNION ALL后面部分）的执行计划为Hash计划，计划存在冲突。

处理方案：把非递归部分语句转储为临时表，促使临时表的分布形式和计划的分布式不一致即可。

案例 1：非递归部分语句的执行计划为 replicate 计划

原始语句

```
WITH RECURSIVE CTE AS (
   SELECT
      AOH.ORG_CODE,1 AS LEVEL,
      1 FLAG,
      LAST_UPDATE_DATE
   FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
   WHERE AOH.ORG_CODE IN (SELECT V.ORG_CODE
      FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 V
      WHERE V.ORG_CODE IS NOT NULL
      AND AOH.ORG_CODE = V.ORG_CODE
   )

   UNION ALL

   SELECT
     AOH.ORG_CODE,T.LEVEL + 1 AS LEVEL,
     1 FLAG,
```

 版权所有 © 华为云计算技术有限公司

```
    AOH.LAST_UPDATE_DATE
  FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
  INNER JOIN CTE T ON T.ORG_CODE = AOH.PARENT_ORG_CODE
)
SELECT
  ORG_CODE,
  MAX(LEVEL) -1 AS AOH_LEVEL,
  1 FLAG,
  LAST_UPDATE_DATE AS LAST_UPDATE_DATE1
FROM CTE
GROUP BY ORG_CODE, LAST_UPDATE_DATE
```

SQL自诊断信息

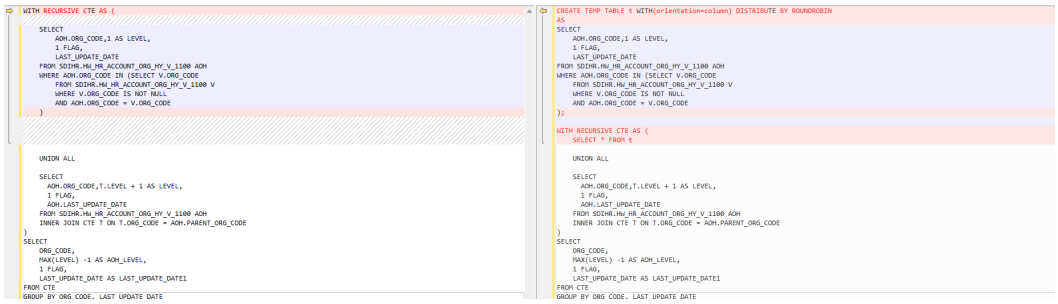
```
id |                                                operation                                                 |
---+----------------------------------------------------------------------------------------------------------+
 1 | ->  HashAggregate                                                                                        |
 2 |    ->  CTE Scan on cte                                                                                   |
 3 |    ->  Recursive Union  (4,7)  [2, CTE cte]                                                              |
 4 |       ->  Subquery Scan on "*SELECT* 1"                                                                  |
 5 |          ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_"                        |
 6 |          ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_" [5, SubPlan 1]         |
 7 |       ->  Hash Join (8,9)                                                                                |
 8 |          ->  Data Node Scan on hw_hr_account_org_hy_v_1100 "_REMOTE_TABLE_QUERY_"                        |
 9 |          ->  Hash                                                                                        |
10 |             ->  WorkTable Scan on cte t                                                                  |

                                    SQL Diagnostic Information
-----------------------------------------------------------------------------------------------------------
SQL is not plan-shipping
    reason: With-Recursive contains conflict distribution in none-recursive(Replicate) recursive(Hash)
```

改写语句

```
CREATE TEMP TABLE t WITH(orientation=column) DISTRIBUTE BY ROUNDROBIN
AS
SELECT
  AOH.ORG_CODE,1 AS LEVEL,
  1 FLAG,
  LAST_UPDATE_DATE
FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
WHERE AOH.ORG_CODE IN (SELECT V.ORG_CODE
  FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 V
  WHERE V.ORG_CODE IS NOT NULL
  AND AOH.ORG_CODE = V.ORG_CODE
);

WITH RECURSIVE CTE AS (
  SELECT * FROM t

  UNION ALL

  SELECT
    AOH.ORG_CODE,T.LEVEL + 1 AS LEVEL,
    1 FLAG,
    AOH.LAST_UPDATE_DATE
  FROM SDIHR.HW_HR_ACCOUNT_ORG_HY_V_1100 AOH
  INNER JOIN CTE T ON T.ORG_CODE = AOH.PARENT_ORG_CODE
)
SELECT
  ORG_CODE,
  MAX(LEVEL) -1 AS AOH_LEVEL,
  1 FLAG,
  LAST_UPDATE_DATE AS LAST_UPDATE_DATE1
FROM CTE
GROUP BY ORG_CODE, LAST_UPDATE_DATE
```

修改点对比

版权所有 © 华为云计算技术有限公司



14.5.4.10 案例：Table in TargetList can not be shipped

问题根因

表名或者表的别名出现在targetlist（输出列SELECT 和 FROM之间）中。

处理方案：改写SQL，消除对表名或者表的别名的直接引用。

案例 1：如场景描述，表 test 的别名 t，出现在 targetlist 中

原始语句

```
SELECT t, 1 FROM test t;
```

```
postgres=#  EXPLAIN VERBOSE SELECT t, 1 FROM test t;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------
 id |                    operation                     | E-rows | E-distinct | E-width | E-costs
----+--------------------------------------------------+--------+------------+---------+---------
  1 | ->  Data Node Scan on test "_REMOTE_TABLE_QUERY_" |     20 |            |      80 | 0.00

          SQL Diagnostic Information
-----------------------------------------------
SQL is not plan-shipping
        reason: Table in TargetList can not be shipped

                  Targetlist Information (identified by plan id)
-------------------------------------------------------------------------------
  1 --Data Node Scan on test "_REMOTE_TABLE_QUERY_"
        Output: t.*, (1)
        Node/s: All datanodes (group_version1, bucket:16384)
        Remote query: SELECT t.*::public.test, 1 FROM ONLY public.test t WHERE true
```

改写语句：强转为text类型，或者优化SQL语句，输出列上去除表名或者表的别名。

```
SELECT t::text, 1 FROM test t;
```


版权所有 © 华为云计算技术有限公司

```
postgres=#  EXPLAIN VERBOSE SELECT t::text, 1 FROM test t;
                                    QUERY PLAN
-----------------------------------------------------------------------------------------------
 id |           operation             | E-rows | E-distinct | E-memory | E-width | E-costs
----+---------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)    |     20 |            |          |      80 | 16.15
  2 |    ->  Seq Scan on public.test t |     20 |            | 1MB      |      80 | 10.15

      Targetlist Information (identified by plan id)
-------------------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: ((t.*)::text), 1
        Node/s: All datanodes (group_version1, bucket:16384)
  2 --Seq Scan on public.test t
        Output: (t.*)::text, 1
```

14.5.4.11 案例："Row () Expression" can not be shipped

问题根因

语句中有row表达式，一般表现为在targetlist（输出列SELECT 和 FROM之间）上存在括号表达式"(.....)"，

处理方案：改写SQL，消除row表达式。

案例 1：row 表达式(k4.businessID, arriveDate)出现在 targetlist 中

原始语句

```
SELECT
   date_format(arriveDate, '%Y-%m-%d') AS "date",
   sales,
   COUNT(DISTINCT (k4.businessID, arriveDate)/*row expression*/)
FROM (
   SELECT k1.businessID, k2.arriveDate, k1.sales
   FROM potential_customers k1
   LEFT JOIN flow_reception k2
      ON k1.businessID = k2.businessID
   WHERE k2.arriveDate BETWEEN '2024-08-31' AND '2024-09-02'
) k4
GROUP BY arriveDate, sales;
```


版权所有 © 华为云计算技术有限公司

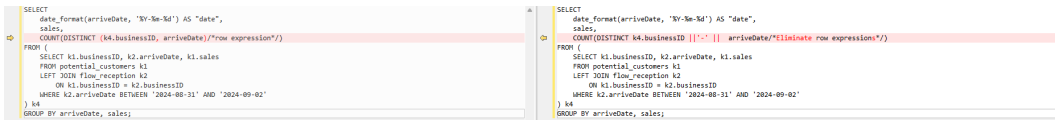
```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 id |                        operation                         | E-rows | E-distinct | E-memory | E-width | E-costs
----+----------------------------------------------------------+--------+------------+----------+---------+--------
  1 | ->  GroupAggregate                                       |      2 |            |          |     104 | 20.63
  2 |    ->  Sort                                              |      2 |            |          |      96 | 20.56
  3 |       ->  Subquery Scan on k4                            |      2 |            |          |      96 | 20.54
  4 |          ->  Streaming (type: SUBQUERY GATHER)           |      2 |            |          |      96 | 26.53
  5 |             ->  Nested Loop (6,8)                        |      2 |            | 1MB      |      96 | 20.53
  6 |                ->  Streaming(type: BROADCAST)            |      2 |            | 2MB      |      64 | 10.31
  7 |                   ->  Seq Scan on public.flow_reception k2 |      1 |            | 1MB      |      64 | 10.15
  8 |                ->  Seq Scan on public.potential_customers k1 |     20 |            | 1MB      |      64 | 10.10

          SQL Diagnostic Information
-------------------------------------------
Partial SQL is not plan-shipping
        reason: "Row () Expression" can not be shipped

                          Predicate Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------
  5 --Nested Loop (6,8)
        Join Filter: (k1.businessid = k2.businessid)
  7 --Seq Scan on public.flow_reception k2
        Filter: ((k2.arrivedate >= '2024-08-31'::text) AND (k2.arrivedate <= '2024-09-02'::text))
```

规避方案：不直接输出row表达式，因此改写row表达式为字符串拼接。

```
SELECT
   date_format(arriveDate, '%Y-%m-%d') AS "date",
   sales,
   COUNT(DISTINCT k4.businessID ||'-' ||  arriveDate/*Eliminate row expressions*/)
FROM (
   SELECT k1.businessID, k2.arriveDate, k1.sales
   FROM potential_customers k1
   LEFT JOIN flow_reception k2
      ON k1.businessID = k2.businessID
   WHERE k2.arriveDate BETWEEN '2024-08-31' AND '2024-09-02'
) k4
GROUP BY arriveDate, sales;
```

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 id |                        operation                         | E-rows | E-distinct | E-memory | E-width | E-costs
----+----------------------------------------------------------+--------+------------+----------+---------+--------
  1 | ->  Streaming (type: GATHER)                             |      2 |            |          |     112 | 26.71
  2 |    ->  HashAggregate                                     |      2 |            | 16MB     |     112 | 20.71
  3 |       ->  HashAggregate                                  |      2 |            | 16MB     |     104 | 20.68
  4 |          ->  Streaming(type: REDISTRIBUTE)               |      2 |            | 2MB      |      96 | 20.64
  5 |             ->  Nested Loop (6,8)                        |      2 |            | 1MB      |      96 | 20.53
  6 |                ->  Streaming(type: BROADCAST)            |      2 |            | 2MB      |      64 | 10.31
  7 |                   ->  Seq Scan on public.flow_reception k2 |      1 |            | 1MB      |      64 | 10.15
  8 |                ->  Seq Scan on public.potential_customers k1 |     20 |            | 1MB      |      64 | 10.10

                          Predicate Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------
  5 --Nested Loop (6,8)
        Join Filter: (k1.businessid = k2.businessid)
  7 --Seq Scan on public.flow_reception k2
        Filter: ((k2.arrivedate >= '2024-08-31'::text) AND (k2.arrivedate <= '2024-09-02'::text))
```

修改点对比



 版权所有 © 华为云计算技术有限公司

14.5.4.12 案例：Merge Into can't be shipped in the scene temporarily

问题根因

MERGE INTO语句的USING子句部分还有非列表达式。

处理方案：改写SQL，去除非列表达式。

案例 1：子查询 d 中的 WHERE 条件含有 AND 1 = 2

原始语句

```
MERGE INTO dwifin.dwi_bill_pay_interface_line l
USING (SELECT to_number(ss_id || invoice_id) as ap_invoice_id,
              to_number(ss_id || invoice_line_id) as credit_invoice_line_id,
              line_number,
              nvl(deleted_flag, 'N') AS src_sys_del_flag
       FROM sdifin.apwf_pay_interface_line_t_6600 t
      WHERE t.adjust_type = 'Y' AND 1 = 2) d
ON (l.ap_invoice_id = d.ap_invoice_id AND l.credit_invoice_line_id = d.credit_invoice_line_id AND
l.line_number = d.line_number AND l.src_sys_del_flag = d.src_sys_del_flag AND l.ss_id = 6600
)
WHEN MATCHED THEN UPDATE SET l.del_flag = 'Y', last_upd_cycle_id = 20250301000000;
```

规避方案：这种情况一般属于SQL语句书写错误，去掉条件“AND 1 = 2”即可。

```
MERGE INTO dwifin.dwi_bill_pay_interface_line l
USING (SELECT to_number(ss_id || invoice_id) as ap_invoice_id,
              to_number(ss_id || invoice_line_id) as credit_invoice_line_id,
              line_number,
              nvl(deleted_flag, 'N') AS src_sys_del_flag
       FROM sdifin.apwf_pay_interface_line_t_6600 t
      WHERE t.adjust_type = 'Y' ) d
ON (l.ap_invoice_id = d.ap_invoice_id AND l.credit_invoice_line_id = d.credit_invoice_line_id AND
l.line_number = d.line_number AND l.src_sys_del_flag = d.src_sys_del_flag AND l.ss_id = 6600
)
WHEN MATCHED THEN UPDATE SET l.del_flag = 'Y', last_upd_cycle_id = 20250301000000;
```

修改点对比

14.5.4.13 案例：Type of Record in dual that is not a real table can not be shipped

问题根因

优化器识别到的record列，且record列对应的主表不是一个实体表。最常见的情况为MERGE INTO语句的USING部分使用了FROM dual子句导致问题出现。

处理方案：改写SQL，消除语句中的dual子句。

案例 1：子查询 f 为(SELECT 0 AS inv_id,0 AS ovs_pack_num,0 AS ovs_pack_id FROM DUAL)

原始语句

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
  SELECT
```

 版权所有 © 华为云计算技术有限公司

```
    f.inv_id,
    d.ovs_pack_id
  FROM (SELECT 0 AS inv_id,0 AS ovs_pack_num,0 AS ovs_pack_id FROM DUAL) f,
  dwicommon.dwi_inv_oversea_pack d
  WHERE f.ovs_pack_num = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```

```
                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 id |                                operation                                | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------------------------------+--------+------------+---------+--------
  1 | ->  Merge on dwiscm.dwi_inv_balance_onhand t                            |      1 |            |    3404 | 0.05
  2 |    ->  Nested Loop (3,6)                                               |      1 |            |    3404 | 0.05
  3 |       ->  Nested Loop (4,5)                                            |      1 |            |    3372 | 0.03
  4 |          ->  Data Node Scan on dwi_inv_balance_onhand "_REMOTE_TABLE_QUERY_" |      1 |            |    3372 | 0.00
  5 |          ->  Result                                                     |      1 |            |       0 | 0.01
  6 |       ->  Data Node Scan on dwi_inv_oversea_pack "_REMOTE_TABLE_QUERY_"  |      1 |            |      32 | 0.00

                     SQL Diagnostic Information
-------------------------------------------------------------------------------
SQL is not plan-shipping
       reason: Type of Record in dual that is not a real table can not be shipped
```

改写语句

```
MERGE INTO dwiscm.dwi_inv_balance_onhand t USING (
  SELECT
    0 AS inv_id,
    d.ovs_pack_id
  FROM dwicommon.dwi_inv_oversea_pack d
  WHERE 0 = d.ovs_pack_num
) S ON (t.inv_id = s.inv_id)
WHEN MATCHED THEN UPDATE
SET t.ovs_pack_id = s.ovs_pack_id, t.last_upd_cycle_id = 20250301000000, t.dw_last_update_date = sysdate;
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 id |                                operation                                | E-rows | E-distinct |   E-memory    | E-width | E-costs
----+-------------------------------------------------------------------------+--------+------------+---------------+---------+--------
  1 | ->  Row Adapter                                                         |      1 |            |               |    3414 | 25.78
  2 |    ->  Vector Streaming (type: GATHER)                                  |      1 |            |               |    3414 | 25.78
  3 |       ->  Vector Merge on dwiscm.dwi_inv_balance_onhand t               |      2 |            | 36MB(3035MB)  |    3414 | 23.28
  4 |          ->  Vector Nest Loop (5,6)                                     |      2 |            | 1MB           |    3414 | 23.28
  5 |             ->  CStore Scan on dwiscm.dwi_inv_balance_onhand t          |      1 |            | 1MB           |    3372 | 10.04
  6 |             ->  Vector Materialize                                      |      2 |            | 16MB          |      42 | 13.24
  7 |                ->  Vector Streaming(type: BROADCAST)                    |      2 |            | 2MB           |      42 | 13.24
  8 |                   ->  CStore Scan on dwicommon.dwi_inv_oversea_pack d    |      1 |            | 1MB           |      42 | 13.08

     Predicate Information (identified by plan id)
-----------------------------------------------------------------
  5 --CStore Scan on dwiscm.dwi_inv_balance_onhand t
       Filter: (t.inv_id = 0::numeric)
       Pushdown Predicate Filter: (t.inv_id = 0::numeric)
  8 --CStore Scan on dwicommon.dwi_inv_oversea_pack d
       Filter: (0 = (d.ovs_pack_num)::bigint)
```

修改点对比

14.5.5 案例：设置 cost_param 对查询性能优化

cost_param参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

版权所有 © 华为云计算技术有限公司

场景一：优化前

cost_param的bit0（set cost_param=1）值为1时，表示对于求!=连接的选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确。下面查询的例子是cost_param的bit0为1时的优化场景。DWS已弃用cost_param & 1不为0时的路径，默认选择已优化的估算公式。

说明

选择率是两表join时，满足join条件的行数在join结果集中所占的比率。

表结构如下所示：

```
CREATE TABLE LINEITEM
(L_ORDERKEY BIGINT NOT NULL,
L_PARTKEY BIGINT NOT NULL,
L_SUPPKEY BIGINT NOT NULL,
L_LINENUMBER BIGINT NOT NULL,
L_QUANTITY DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
L_DISCOUNT DECIMAL(15,2) NOT NULL,
L_TAX DECIMAL(15,2) NOT NULL,
L_RETURNFLAG CHAR(1) NOT NULL,
L_LINESTATUS CHAR(1) NOT NULL,
L_SHIPDATE DATE NOT NULL,
L_COMMITDATE DATE NOT NULL,
L_RECEIPTDATE DATE NOT NULL,
L_SHIPINSTRUCT CHAR(25) NOT NULL,
L_SHIPMODE CHAR(10) NOT NULL,
L_COMMENT VARCHAR(44) NOT NULL)
with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
(O_ORDERKEY BIGINT NOT NULL,
O_CUSTKEY BIGINT NOT NULL,
O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE DECIMAL(15,2) NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY BIGINT NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL)
with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
order by
numwait desc;
```


版权所有 © 华为云计算技术有限公司

执行计划如下图所示（verbose条件下，新增distinct列，受cost off/on控制，hashjoin行显示内外表的distinct估值，其他行为空）：

```
 id |                        operation                        | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Row Adapter                                         |      1 |            |       8 | 39.36
  2 |    ->  Vector Sort                                      |      1 |            |       8 | 39.36
  3 |       ->  Vector Aggregate                              |      1 |            |       8 | 39.34
  4 |          ->  Vector Streaming (type: GATHER)            |      2 |            |       8 | 39.34
  5 |             ->  Vector Aggregate                        |      2 |            |       8 | 39.25
  6 |                ->  Vector Hash Anti Join (7, 10)        |      2 | 4, 5       |       0 | 39.24
  7 |                   ->  Vector Hash Join (8,9)            |      2 | 200, 1     |      16 | 26.12
  8 |                      ->  CStore Scan on public.lineitem 11 |      7 |            |      16 | 13.05
  9 |                      ->  CStore Scan on public.orders   |      1 |            |       8 | 13.05
 10 |                   ->  CStore Scan on public.lineitem 13 |      7 |            |      16 | 13.05
```

场景一：优化后

以上查询为lineitem表自连接的Anti Join，当使用cost_param的bit0为0时，估算Anti Join的行数与实际行数相差很大，导致查询性能下降。可以通过设置cost_param的bit0为1时，使Anti Join的行数估算更准确，从而提高查询性能。优化后的执行计划如下：

```
 id |                        operation                        |  E-rows   | E-memory | E-width |  E-costs
----+---------------------------------------------------------+-----------+----------+---------+-----------
  1 | -> Row Adapter                                          |         1 |          |       0 | 9104892.37 9
  2 |    -> Vector Sort                                       |         1 |          |       0 | 9104892.37 9
  3 |       -> Vector Aggregate                               |         1 |          |       0 | 9104892.35 8
  4 |          -> Vector Streaming (type: GATHER)             |        48 |          |       0 | 9104892.35 8
  5 |             -> Vector Aggregate                         |        48 | 1MB      |       0 | 9104890.82 5
  6 |                -> Vector Hash Join (7.12)               | 2526630903 | 929MB    |       0 | 8973295.45 4
  7 |                   -> Vector Hash Anti Join (8. 10)      | 1999996587 | 3178MB   |       8 | 7198231.14
  8 |                      -> Vector Partition Iterator       | 1999996587 | 1MB      |      16 | 3000158.25
  9 |                         -> Partitioned CStore Scan on public.lineitem 11 | 1999996587 | 1MB | 16 | 3000158.25 1
 10 |                      -> Vector Partition Iterator       | 1999996587 | 1MB      |      16 | 3000158.25
 11 |                         -> Partitioned CStore Scan on public.lineitem 13 | 1999996587 | 1MB | 16 | 3000158.25
 12 |                   -> Vector Partition Iterator          |  730839014 | 1MB      |       8 | 589611.00
 13 |                      -> Partitioned CStore Scan on public.orders | 730839014 | 1MB | 8 | 589611.00
```

场景二：优化前

当cost_param的bit1（set cost_param=2）为1时，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确。下面查询的例子是cost_param的bit1为1时的优化场景。

表结构如下所示：

```
CREATE TABLE NATION
(
N_NATIONKEY INT NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INT NOT NULL,
N_COMMENT VARCHAR(152)
)
distribute by replication;

CREATE TABLE SUPPLIER
(
S_SUPPKEYBIG INT NOT NULL,
S_NAME CHAR(25) NOT NULL,
S_ADDRESS VARCHAR(40) NOT NULL,
S_NATIONKEY INT NOT NULL,
S_PHONE CHAR(15) NOT NULL,
S_ACCTBAL DECIMAL(15,2) NOT NULL,
S_COMMENT VARCHAR(101) NOT NULL) distribute by hash(S_SUPPKEY);

CREATE TABLE PARTSUPP
(
```

 版权所有 © 华为云计算技术有限公司

```
PS_PARTKEYBIG INT NOT NULL,
PS_SUPPKEYBIG INT NOT NULL,
PS_AVAILQTYBIG INT NOT NULL,
PS_SUPPLYCOST DECIMAL(15,2) NOT NULL,
PS_COMMENT VARCHAR(199) NOT NULL
)
distribute by hash(PS_PARTKEY);
```

查询语句如下所示：

```
set cost_param=2;
explain verbose select
nation,
sum(amount) as sum_profit
from
(
select
n_name as nation,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
from
supplier,
lineitem,
partsupp,
nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and s_nationkey = n_nationkey
) as profit
group by nation
order by nation;
```

当cost_param的bit1为0时，执行计划如下图所示：

```
 id |                           operation                           | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Sort                                                      |      1 |            |     208 | 61.52
  2 |    ->  HashAggregate                                          |      1 |            |     208 | 61.51
  3 |       ->  Streaming (type: GATHER)                            |      2 |            |     208 | 61.51
  4 |          ->  HashAggregate                                    |      2 |            |     208 | 61.36
  5 |             ->  Hash Join (6,7)                               |      2 | 20, 15     |     176 | 61.33
  6 |                ->  Seq Scan on public.nation                  |     40 |            |     108 | 20.20
  7 |                ->  Hash                                       |      2 |            |      76 | 41.04
  8 |                   ->  Hash Join (9,16)                        |      2 | 10, 13     |      76 | 41.04
  9 |                      ->  Streaming(type: REDISTRIBUTE)        |      2 |            |      88 | 27.73
 10 |                         ->  Hash Join (11,14)                 |      2 | 10, 13     |      88 | 27.62
 11 |                            ->  Streaming(type: REDISTRIBUTE)  |     20 |            |      70 | 14.19
 12 |                               ->  Row Adapter                 |     21 |            |      70 | 13.01
 13 |                                  ->  CStore Scan on public.lineitem |     20 |            |      70 | 13.01
 14 |                            ->  Hash                           |     21 |            |      34 | 13.13
 15 |                               ->  Seq Scan on public.partsupp |     20 |            |      34 | 13.13
 16 |                         ->  Hash                              |     21 |            |      12 | 13.13
 17 |                            ->  Seq Scan on public.supplier    |     20 |            |      12 | 13.13
```

场景二：优化后

在以上查询中，supplier、lineitem、partsupp三表做hashjoin的条件为(lineitem.l_suppkey = supplier.s_suppkey) AND (lineitem.l_partkey = partsupp.ps_partkey)，此hashjoin条件中存在两个过滤条件，这前一个过滤条件中的lineitem.l_suppkey和后一个过滤条件中的lineitem.l_partkey同为lineitem表的两列，这两列存在强相关的关联关系。这种情况下，估算hashjoin条件的选择率时，如果使用cost_param的bit1为0时，实际是将AND的两个过滤条件分别计算的2个选择率的值相乘来得到hashjoin条件的选择率，导致行数估算不准确，查询性能较差。所以需要将cost_param的bit1为1时，选择最小的选择率作为总的选择率估算行数比较准确，查询性能较好，优化后的计划如下图所示：

 版权所有 © 华为云计算技术有限公司

```
 id |                          operation                          | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------------------+--------+------------+---------+---------
  1 | -> Sort                                                     |     10 |            |     208 | 64.42
  2 |    -> HashAggregate                                         |     10 |            |     208 | 64.23
  3 |       -> Streaming (type: GATHER)                           |     20 |            |     208 | 64.23
  4 |          -> HashAggregate                                   |     20 |            |     208 | 62.71
  5 |             -> Hash Join (6,7)                              |     20 | 20, 10     |     176 | 62.46
  6 |                -> Seg Scan on public.nation                 |     40 |            |     108 | 20.20
  7 |                -> Hash                                      |     20 |            |      76 | 41.97
  8 |                   -> Hash Join (9,16)                       |     20 | 10, 13     |      76 | 41.97
  9 |                      -> Streaming(type: REDISTRIBUTE)       |     20 |            |      82 | 28.54
 10 |                         -> Hash Join (11,14)               |     20 | 10, 13     |      82 | 27.63
 11 |                            -> Streaming(type: REDISTRIBUTE) |     20 |            |      70 | 14.19
 12 |                               -> Row Adapter                |     21 |            |      70 | 13.01
 13 |                                  -> CStore Scan on public.lineitem |     20 |     |      70 | 13.01
 14 |                            -> Hash                          |     21 |            |      12 | 13.13
 15 |                               -> Seg Scan on public.supplier |    20 |            |      12 | 13.13
 16 |                      -> Hash                                |     21 |            |      34 | 13.13
 17 |                         -> Seg Scan on public.partsupp      |     20 |            |      34 | 13.13
```

14.5.6 案例：调整局部聚簇键

局部聚簇（Partial Cluster Key，简称PCK），列存储下一种通过min/max稀疏索引实现基表快速扫描的索引技术。Partial Cluster Key可以指定多列，但是一般不建议超过2列。PCK适用于列存大表点查询加速。

优化前

创建一个无局部聚簇（以下称为PCK）的列存表orders_no_pck，表定义如下：

```
                                pg_get_tabledef
---------------------------------------------------------------------------------
 SET search_path = dbadmin;                                                      +
 CREATE  TABLE orders_no_pck (                                                   +
         o_orderkey bigint NOT NULL,                                             +
         o_custkey bigint NOT NULL,                                              +
         o_orderstatus character(1) NOT NULL,                                    +
         o_totalprice numeric(15,2) NOT NULL,                                    +
         o_orderdate timestamp(0) without time zone NOT NULL,                    +
         o_orderpriority character(15) NOT NULL,                                 +
         o_clerk character(15) NOT NULL,                                         +
         o_shippriority bigint NOT NULL,                                         +
         o_comment character varying(79) NOT NULL                                +
 )                                                                               +
 WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)+
 DISTRIBUTE BY HASH(o_orderkey)                                                  +
 TO GROUP group_version1;
(1 row)
```

执行以下SQL语句，查询某个点查询的执行计划：

```
EXPLAIN PERFORMANCE
SELECT * FROM orders_no_pck
WHERE o_orderkey = '13095143'
ORDER BY o_orderdate;
```

由下图可知执行时间为48毫秒，查看Datanode Information发现filter时间为19毫秒，CUNone比例为0。

```
gaussdb=>  EXPLAIN PERFORMANCE
gaussdb-> SELECT * FROM orders_no_pck
gaussdb-> WHERE o_orderkey = '13095143'
gaussdb-> ORDER BY o_orderdate;
                                                   QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------
 id |                operation                 |      A-time      | A-rows | E-rows | E-distinct |  Peak Memory   | E-memory | A-width | E-width | E-costs
----+------------------------------------------+------------------+--------+--------+------------+----------------+----------+---------+---------+---------
  1 | -> Row Adapter                           | 48.182           |      1 |      3 |            | 82KB           |          |         |     123 | 94838.01
  2 |    -> Vector Streaming (type: GATHER)    | 48.175           |      1 |      3 |            | 825KB          |          |         |     123 | 94838.01
  3 |       -> Vector Sort                     | [44.260, 44.772] |      1 |      3 |            | [330KB, 411KB] | 16MB     | [0,167] |     123 | 94830.01
  4 |          -> CStore Scan on public.orders_no_pck | [44.157, 44.669] | 1 |    1 |            | [1MB, 1MB]     | 1MB      |         |     123 | 94830.00

          Predicate Information (identified by plan id)
```


版权所有 © 华为云计算技术有限公司

```
                         Datanode Information (identified by plan id)
-------------------------------------------------------------------------------------------------
  1 --Row Adapter
        (actual time=48.181..48.182 rows=1 loops=1)
        (CPU: ex c/r=676, ex row=1, ex cyc=676, inc cyc=4818100)
  2 --Vector Streaming (type: GATHER)
        (actual time=48.174..48.175 rows=1 loops=1)
        (Buffers: 0)
        (CPU: ex c/r=4817424, ex row=1, ex cyc=4817424, inc cyc=4817424)
  3 --Vector Sort
        dn_6001_6002 (actual time=44.461..44.461 rows=0 loops=1)
        dn_6003_6004 (actual time=44.259..44.260 rows=1 loops=1)
        dn_6005_6006 (actual time=44.772..44.772 rows=0 loops=1)
        dn_6001_6002 (Buffers: shared hit=389)
        dn_6003_6004 (Buffers: shared hit=389)
        dn_6005_6006 (Buffers: shared hit=389)
        dn_6001_6002 (CPU: ex c/r=0, ex row=0, ex cyc=11343, inc cyc=4446062)
        dn_6003_6004 (CPU: ex c/r=10101, ex row=1, ex cyc=10101, inc cyc=4425810)
        dn_6005_6006 (CPU: ex c/r=0, ex row=0, ex cyc=10257, inc cyc=4477201)
  4 --CStore Scan on public.orders_no_pck
        dn_6001_6002 (actual time=44.348..44.348 rows=0 loops=1) (filter time=19.721) (RoughCheck CU: CUNone: 0, CUSome: 84)
        dn_6003_6004 (actual time=38.704..44.157 rows=1 loops=1) (filter time=19.739) (RoughCheck CU: CUNone: 0, CUSome: 84)
        dn_6005_6006 (actual time=44.669..44.669 rows=0 loops=1) (filter time=19.568) (RoughCheck CU: CUNone: 0, CUSome: 84)
        dn_6001_6002 (Buffers: shared hit=389)
        dn_6003_6004 (Buffers: shared hit=389)
        dn_6005_6006 (Buffers: shared hit=389)
        dn_6001_6002 (CPU: ex c/r=0, ex row=5007635, ex cyc=4434719, inc cyc=4434719)
        dn_6003_6004 (CPU: ex c/r=0, ex row=5002975, ex cyc=4415709, inc cyc=4415709)
        dn_6005_6006 (CPU: ex c/r=0, ex row=4989390, ex cyc=4466944, inc cyc=4466944)
```

优化后

创建的列存表orders_pck。表定义如下：

```
                              pg_get_tabledef
--------------------------------------------------------------------------------
 SET search_path = dbadmin;                                                    +
 CREATE  TABLE orders_pck (                                                    +
         o_orderkey bigint NOT NULL,                                           +
         o_custkey bigint NOT NULL,                                            +
         o_orderstatus character(1) NOT NULL,                                  +
         o_totalprice numeric(15,2) NOT NULL,                                  +
         o_orderdate timestamp(0) without time zone NOT NULL,                  +
         o_orderpriority character(15) NOT NULL,                               +
         o_clerk character(15) NOT NULL,                                       +
         o_shippriority bigint NOT NULL,                                       +
         o_comment character varying(79) NOT NULL                              +
 )                                                                             +
 WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)+
 DISTRIBUTE BY HASH(o_orderkey)                                                +
 TO GROUP group_version1;
(1 row)
```

使用ALTER TABLE将字段o_orderkey设置为PCK：

```
postgres=> ALTER TABLE orders_pck ADD PARTIAL CLUSTER KEY(o_orderkey);
ALTER TABLE
```

执行以下SQL语句，再次查询同样的点查询SQL语句的执行计划：

```
EXPLAIN PERFORMANCE
SELECT * FROM orders_pck
WHERE o_orderkey = '13095143'
ORDER BY o_orderdate;
```

由下图可知执行时间为5毫秒，查看Datanode Information发现filter时间为0.5毫秒，CUNone比例为82。CUNone比例越高，PCK的性能收益越明显。

```
gaussdb=> EXPLAIN PERFORMANCE
gaussdb-> SELECT * FROM orders_pck
gaussdb-> WHERE o_orderkey = '13095143'
gaussdb-> ORDER BY o_orderdate;
                                                         QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 id |                 operation                  |     A-time     | A-rows | E-rows | E-distinct |   Peak Memory    | E-memory | A-width | E-width | E-costs
----+--------------------------------------------+----------------+--------+--------+------------+------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                            | 5.597          |      1 |      3 |            | 82KB             |          |         |     123 | 94838.01
  2 |    ->  Vector Streaming (type: GATHER)     | 5.589          |      1 |      3 |            | 825KB            |          |         |     123 | 94838.01
  3 |       ->  Vector Sort                      | [1.858, 1.929] |      1 |      3 |            | [330KB, 411KB]   | 16MB     | [0,167] |     123 | 94830.01
  4 |          ->  CStore Scan on public.orders_pck | [1.742, 1.804] |   1 |      1 |            | [1MB, 1MB]       | 1MB      |         |     123 | 94830.00

      Predicate Information (identified by plan id)
 ---------------------------------------------------------------
```


版权所有 © 华为云计算技术有限公司

```
                                    Datanode Information (identified by plan id)
-------------------------------------------------------------------------------------------------------------------------------------
  1 --Row Adapter
        (actual time=5.597..5.597 rows=1 loops=1)
        (CPU: ex c/r=815, ex row=1, ex cyc=815, inc cyc=559741)
  2 --Vector Streaming (type: GATHER)
        (actual time=5.589..5.589 rows=1 loops=1)
        (Buffers: shared hit=3)
        (CPU: ex c/r=558926, ex row=1, ex cyc=558926, inc cyc=558926)
  3 --Vector Sort
        dn_6001_6002 (actual time=1.858..1.858 rows=0 loops=1)
        dn_6003_6004 (actual time=1.914..1.914 rows=1 loops=1)
        dn_6005_6006 (actual time=1.929..1.929 rows=0 loops=1)
        dn_6001_6002 (Buffers: shared hit=395)
        dn_6003_6004 (Buffers: shared hit=396)
        dn_6005_6006 (Buffers: shared hit=396)
        dn_6001_6002 (CPU: ex c/r=0, ex row=0, ex cyc=11573, inc cyc=185784)
        dn_6003_6004 (CPU: ex c/r=12187, ex row=1, ex cyc=12187, inc cyc=191420)
        dn_6005_6006 (CPU: ex c/r=0, ex row=0, ex cyc=12455, inc cyc=192864)
  4 --CStore Scan on public.orders_pck
        dn_6001_6002 (actual time=1.742..1.742 rows=0 loops=1) (filter time=0.497) (RoughCheck CU: CUNone: 82, CUSome: 2)
        dn_6003_6004 (actual time=1.694..1.793 rows=1 loops=1) (filter time=0.509) (RoughCheck CU: CUNone: 82, CUSome: 2)
        dn_6005_6006 (actual time=1.804..1.804 rows=0 loops=1) (filter time=0.509) (RoughCheck CU: CUNone: 82, CUSome: 2)
        dn_6001_6002 (Buffers: shared hit=392)
        dn_6003_6004 (Buffers: shared hit=393)
        dn_6005_6006 (Buffers: shared hit=393)
        dn_6001_6002 (CPU: ex c/r=0, ex row=5007635, ex cyc=174211, inc cyc=174211)
        dn_6003_6004 (CPU: ex c/r=0, ex row=5002975, ex cyc=179233, inc cyc=179233)
        dn_6005_6006 (CPU: ex c/r=0, ex row=4989390, ex cyc=180409, inc cyc=180409)
```

14.5.7 案例：调整中间表存储方式

在DWS中行存表使用行执行引擎，列存表使用列执行引擎。如果一个SQL语句涉及的表既有行存表又有列存表，系统会自动选择行执行引擎。由于列执行引擎的性能(除indexscan相关的算子)比行执行引擎性能要好很多，因此一般建议使用列存表。特别是对一些中间结果集转储的表，一定要分析清楚，使用合适的表存储类型。

优化前

某局点测试过程遇到如下的执行计划，客户希望将性能提升至3s内返回结果。

```
id |                                                operation                                                 |      A-time      | A-rows | E-rows |  Peak Memory   | E-memory | A-width | E-width | E-costs
---+----------------------------------------------------------------------------------------------------------+------------------+--------+--------+----------------+----------+---------+---------+---------
 1 | ->  Streaming (type: GATHER)                                                                             | 4651.039         |      7 |     17 | 304KB          |      1MB |         |      41 | 101740.13
 2 |    ->  Hash Join (3,7)                                                                                   | [4.801,4629.889] |      7 |     17 | [8KB, 8KB]     |      1MB |         |      41 | 101739.10
 3 |       ->  Append                                                                                         | [3474.514,3840.836] | 34752433 | 108109436 | [1KB, 1KB]     |      1MB |         |      49 | 88969.75
 4 |          ->  Row Adapter                                                                                 | [2727.901,3040.924] | 33011417 |  99098596 | [49KB, 49KB]   |      1MB |         |      49 | 70615.19
 5 |             ->  Partitioned Dfs Scan on sd_data.act_account_his ta                                       | [2288.421,2558.707] | 33011417 |  99098596 | [1002KB, 1012KB] |      1MB |         |      49 | 70615.19
 6 |          ->  Seq Scan on catore.pg_delta_2425217623 ta                                                   | [163.977,169.707] |  1741016 |   9010840 | [15KB, 15KB]   |      1MB |         |      50 | 18354.56
 7 |       ->  Hash                                                                                           | [4.386,7.999]    |      9 |     32 | [260KB, 292KB] |      16MB |         |      30 | 100.17
 8 |          ->  Streaming(type: REDISTRIBUTE)                                                               | [4.384,7.997]    |      9 |     32 | [1054KB, 1056KB] |      1MB |  [0, 36] |      30 | 100.17
 9 |             ->  Hash Join (10,11)                                                                        | [0.162,1.043]    |      9 |     32 | [5KB, 5KB]     |      1MB |         |      30 | 100.06
10 |                ->  Seq Scan on pg_temp_cn_5001_140148717123328.input_acct_id_tbl tb                      | [0.005,0.176]    |   1030 |  31968 | [11KB, 11KB]   |      1MB |         |      11 | 15.99
11 |                ->  Hash                                                                                  | [0.001,0.849]    |      9 |     32 | [258KB, 290KB] |      16MB |  [0, 37] |      19 | 80.31
12 |                   ->  HashAggregate                                                                      | [0.001,0.849]    |      9 |     32 | [10KB, 13KB]   |      1MB |         |      19 | 80.30
13 |                      ->  Seq Scan on public.row_unlogged_table                                           | [0.000,0.847]    |    449 |    449 | [13KB, 13KB]   |      1MB |         |      19 | 78.70
(13 rows)
```

优化后

经过分析发现计划走了行引擎。根本原因是：临时计划表input_acct_id_tbl和中间结果转储表row_unlogged_table使用了行存表。

修改这两个表为列存表之后，性能提升至1.6s。

```
id |                                                operation                                                 |      A-time      | A-rows | E-rows |  Peak Memory   | E-memory | A-width | E-width | E-costs
---+----------------------------------------------------------------------------------------------------------+------------------+--------+--------+----------------+----------+---------+---------+---------
 1 | ->  Row Adapter                                                                                          | 1567.367         |      7 |     17 | 38KB           |          |         |      41 | 101758.52
 2 |    ->  Vector Streaming (type: GATHER)                                                                   | 1567.349         |      7 |     17 | 393KB          |          |         |      41 | 101758.52
 3 |       ->  Vector Hash Join (4,8)                                                                         | [8.130,1529.101] |      7 |     17 | [2362KB, 2446KB] |      16MB |         |      41 | 101757.48
 4 |          ->  Vector Append                                                                               | [542.823,1452.479] | 5681770 | 108109436 | [1KB, 1KB]     |      1MB |         |      49 | 88969.75
 5 |             ->  Partitioned Dfs Scan on sd_data.act_account_his ta                                       | [295.796,1195.830] | 3940784 |  99098596 | [861KB, 1012KB] |      1MB |         |      49 | 70615.19
 6 |          ->  Vector Adapter                                                                              | [236.065,260.284] |  1741016 |   9010840 | [129KB, 129KB] |      1MB |         |      50 | 18354.56
 7 |             ->  Seq Scan on catore.pg_delta_2425217623 ta                                                | [152.595,168.048] |  1741016 |   9010840 | [15KB, 15KB]   |      1MB |         |      50 | 18354.56
 8 |          ->  Vector Streaming(type: REDISTRIBUTE)                                                        | [7.727,12.981]   |      9 |     32 | [1092KB, 1141KB] |      1MB |  [0, 40] |      30 | 118.56
 9 |             ->  Vector Hash Join (10,11)                                                                 | [0.132,4.958]    |      9 |     32 | [2217KB, 2217KB] |      16MB |         |      30 | 118.48
10 |                ->  CStore Scan on pg_temp_cn_5001_140148158066112.input_acct_id_tbl tb                   | [4.372,4.372]    |    999 |  31968 | [207KB, 207KB] |      1MB |         |      11 | 81.00
11 |                ->  Vector Hash Aggregate                                                                 | [0.062,0.209]    |      9 |     32 | [2225KB, 2225KB] |      16MB |  [0, 35] |      19 | 83.67
12 |                   ->  CStore Scan on public.col_unlogged_table                                           | [0.011,0.107]    |    449 |    449 | [541KB, 598KB] |      1MB |         |      19 | 82.08
```

14.5.8 案例：改建分区表

逻辑上的一张表根据某种策略分成多个物理块进行存储，这张逻辑上的表称之为分区表，每个物理块则称为一个分区。一般对数据和查询都有明显区间段特征的表使用分区策略可通过减少不必要的数据扫描，从而提升查询性能。

在查询时，可通过分区剪枝技术尽可能减少底层数据扫描，即缩小表的扫描范围。分区剪枝是指对于分区表或分区索引来说，优化器可以自动从FROM和WHERE子句里根

 版权所有 © 华为云计算技术有限公司

据分区键提取出需要扫描的分区，从而避免全表扫描，减少扫描的数据块，提高性能。

优化前

创建一个非分区表orders_no_part，表定义如下：

```
                              pg_get_tabledef
--------------------------------------------------------------------------------
 SET search_path = dbadmin;
 CREATE  TABLE orders_no_part (
         o_orderkey bigint NOT NULL,
         o_custkey bigint NOT NULL,
         o_orderstatus character(1) NOT NULL,
         o_totalprice numeric(15,2) NOT NULL,
         o_orderdate timestamp(0) without time zone NOT NULL,
         o_orderpriority character(15) NOT NULL,
         o_clerk character(15) NOT NULL,
         o_shippriority bigint NOT NULL,
         o_comment character varying(79) NOT NULL
 )
 WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)
 DISTRIBUTE BY HASH(o_orderkey)
 TO GROUP group_version1;
(1 row)
```

执行以下SQL语句查询非分区表的执行计划：

```
EXPLAIN PERFORMANCE
SELECT count(*) FROM orders_no_part WHERE
o_orderdate >= '1996-01-01 00:00:00'::timestamp(0);
```

由下图可知执行时间为73毫秒，其中全表扫描的时间为44~45毫秒。

```
gaussdb=> EXPLAIN PERFORMANCE
gaussdb-> SELECT count(*) FROM orders_no_part WHERE
gaussdb-> o_orderdate >= '1996-01-01 00:00:00'::timestamp(0);
                                    QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------
 id |                  operation                   |       A-time       | A-rows  | E-rows  | E-distinct |   Peak Memory    | E-memory | A-width | E-width | E-costs
----+----------------------------------------------+--------------------+---------+---------+------------+------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                              | 73.623             |       1 |       1 |            | 10KB             |          |         |       8 | 99791.27
  2 |    ->  Vector Aggregate                      | 73.611             |       1 |       1 |            | 177KB            |          |         |       8 | 99791.27
  3 |       ->  Vector Streaming (type: GATHER)    | 73.575             |       3 |       3 |            | 89KB             |          |         |       8 | 99791.27
  4 |          ->  Vector Aggregate                | [54.963, 55.561]   |       3 |       3 |            | [138KB, 138KB]   | 1MB      |         |       8 | 99783.27
  5 |             ->  CStore Scan on public.orders_no_part | [44.572, 45.077] | 5890663 | 5943908 |      | [308KB, 308KB]   | 1MB      |         |       0 | 94830.00
```

优化后

创建一个分区表orders。表定义如下：

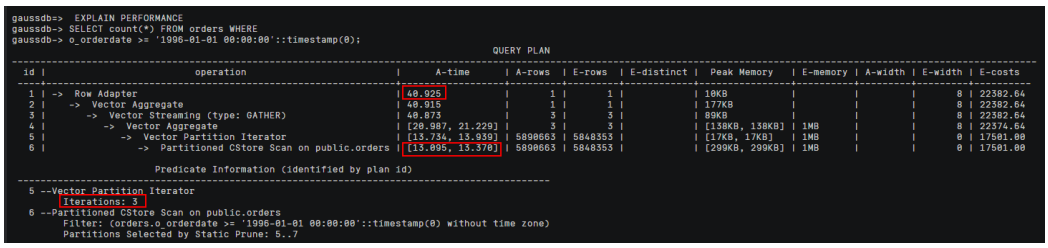
```
                              pg_get_tabledef
--------------------------------------------------------------------------------
 SET search_path = dbadmin;                                                    +
 CREATE  TABLE orders (                                                        +
         o_orderkey bigint NOT NULL,                                           +
         o_custkey bigint NOT NULL,                                            +
         o_orderstatus character(1) NOT NULL,                                  +
         o_totalprice numeric(15,2) NOT NULL,                                  +
         o_orderdate timestamp(0) without time zone NOT NULL,                  +
         o_orderpriority character(15) NOT NULL,                               +
         o_clerk character(15) NOT NULL,                                       +
         o_shippriority bigint NOT NULL,                                       +
         o_comment character varying(79) NOT NULL                              +
 )                                                                             +
 WITH (orientation=column, compression=low, colversion=2.0, enable_delta=false)+
 DISTRIBUTE BY HASH(o_orderkey)                                                +
 TO GROUP group_version1                                                       +
 PARTITION BY RANGE (o_orderdate)                                              +
 (                                                                             +
          PARTITION o_orderdate_1 VALUES LESS THAN ('1993-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_2 VALUES LESS THAN ('1994-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_3 VALUES LESS THAN ('1995-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_4 VALUES LESS THAN ('1996-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_5 VALUES LESS THAN ('1997-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_6 VALUES LESS THAN ('1998-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default,+
          PARTITION o_orderdate_7 VALUES LESS THAN ('1999-01-01 00:00:00'::timestamp(0) without time zone) TABLESPACE pg_default +
 )                                                                             +
 ENABLE ROW MOVEMENT;
(1 row)
```

再次执行SQL语句，查询分区表的执行计划。执行时间为40毫秒，其中表扫描时间仅为13毫秒，Iterations越小，分区剪枝效果越好。

版权所有 © 华为云计算技术有限公司

```
EXPLAIN PERFORMANCE
SELECT count(*) FROM orders_no_part WHERE
o_orderdate >= '1996-01-01 00:00:00'::timestamp(0);
```

由下图可知执行时间为40毫秒，其中表扫描时间仅为13毫秒。另外Iterations越小，分区剪枝效果越好。



14.5.9 案例：调整 GUC 参数 best_agg_plan

现象描述

t1的表定义为：

```
create table t1(a int, b int, c int) distribute by hash(a);
```

假设agg下层算子所输出结果集的分布列为setA，agg操作的group by列为setB，则在Stream框架下，Agg操作可以分为两个场景。

场景一：setA是setB的一个子集。

对于这种场景，直接对下层结果集进行汇聚的结果就是正确的汇聚结果，上层算子直接使用即可。如下图所示：

```
explain select a, count(1) from t1 group by a;
 id |         operation          | E-rows | E-width | E-costs
----+----------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER) |     30 |       4 | 15.56
  2 |    ->  HashAggregate       |     30 |       4 | 14.31
  3 |       ->  Seq Scan on t1   |     30 |       4 | 14.14
(3 rows)
```

场景二：setA不是setB的一个子集。

对于这种场景，Stream执行框架分为如下三种计划形态：

hashagg+gather(redistribute)+hashagg

redistribute+hashagg(+gather)

hashagg+redistribute+hashagg(+gather)

DWS提供了guc参数best_agg_plan来干预执行计划，强制其生成上述对应的执行计划，此参数取值范围为0，1，2，3

- 取值为1时，强制生成第一种计划。
- 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
- 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
- 取值为0时，优化器会根据以上三种计划的估算代价选择最优的一种计划生成。

具体影响如下：

 版权所有 © 华为云计算技术有限公司

```
set best_agg_plan to 1;
SET
explain select b,count(1) from t1 group by b;
 id |            operation            | E-rows | E-width | E-costs
----+---------------------------------+--------+---------+---------
  1 | ->  HashAggregate               |      8 |       4 | 15.83
  2 |    ->  Streaming (type: GATHER) |     25 |       4 | 15.83
  3 |       ->  HashAggregate         |     25 |       4 | 14.33
  4 |          ->  Seq Scan on t1     |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 2;
SET
explain select b,count(1) from t1 group by b;
 id |               operation                | E-rows | E-width | E-costs
----+----------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)           |     30 |       4 | 15.85
  2 |    ->  HashAggregate                   |     30 |       4 | 14.60
  3 |       ->  Streaming(type: REDISTRIBUTE) |     30 |       4 | 14.45
  4 |          ->  Seq Scan on t1            |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 3;
SET
explain select b,count(1) from t1 group by b;
 id |               operation                | E-rows | E-width | E-costs
----+----------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)           |     30 |       4 | 15.84
  2 |    ->  HashAggregate                   |     30 |       4 | 14.59
  3 |       ->  Streaming(type: REDISTRIBUTE) |     25 |       4 | 14.59
  4 |          ->  HashAggregate             |     25 |       4 | 14.33
  5 |             ->  Seq Scan on t1         |     30 |       4 | 14.14
(5 rows)
```

总结

通常优化器总会选择最优的执行计划，但是众所周知代价估算，尤其是中间结果集的代价估算一般会有比较大的偏差，这种比较大的偏差就可能会导致agg的计算方式出现比较大的偏差，这时候就需要通过best_agg_plan进行agg计算模型的干预。

一般来说，当agg汇聚的收敛度很小时，即结果集的个数在agg之后并没有明显变少时（经验上以5倍为临界点），选择redistribute+hashagg执行方式，否则选择hashagg+redistribute+hashagg执行方式。

14.5.10 案例：改写 SQL 排除剪枝干扰

分区表查询中表达式一般不是单纯的分区键，而是包含分区键的表达式的Filter条件，这种类型的Filter条件是不能用来剪枝的。

优化前

t_ddw_f10_op_cust_asset_mon为分区表，分区键为year_mth，此字段是由年月两个值拼接而成的整数。

测试SQL如下：

```
SELECT
   count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth  < substr('20200722',1 ,6 )
AND b1.year_mth + 1 >= substr('20200722',1 ,6 );
```

测试结果显示此SQL的表Scan耗时长达10s，查询SQL语句的执行计划如下

```
EXPLAIN (ANALYZE ON, VERBOSE ON)
SELECT
```


版权所有 © 华为云计算技术有限公司

```
   count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth < substr('20200722',1 ,6 )
AND b1.year_mth + 1 >= cast(substr('20200722',1 ,6 ) AS int);
                                                              QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------
  id |                            operation                             |       A-time         |  A-rows  |  E-rows  | E-
distinct | Peak Memory  | E-memory | A-width | E-width |  E-costs
 ----+--------------------------------------------------------------------------+-----------------------+-----------
+----------+------------+--------------+----------+---------+---------+-----------
   1 | ->  Aggregate                                                   | 10662.260            |        1 |        1 |
| 32KB         |          |          |       8 | 593656.42
   2 |    ->  Streaming (type: GATHER)                                 | 10662.172            |        4 |        4
|           | 136KB        |          |          |       8 | 593656.42
   3 |       ->  Aggregate                                             | [9692.785, 10656.068] |        4 |        4
|           | [24KB, 24KB] | 1MB      |          |       8 | 593646.42
   4 |          ->  Partition Iterator                                 | [8787.198, 9629.138]  | 16384000 |
32752850 |            | [16KB, 16KB] | 1MB      |          |       0 | 573175.88
   5 |             ->  Partitioned Seq Scan on public.t_ddw_f10_op_cust_asset_mon b1 | [8365.655, 9152.115]  |
16384000 | 32752850 |            | [32KB, 32KB] | 1MB      |          |       0 | 573175.88

                            SQL Diagnostic Information
 ------------------------------------------------------------------------------------------
 Partitioned table unprunable Qual
        table public.t_ddw_f10_op_cust_asset_mon b1:
        left side of expression "((year_mth + 1) > 202008)" invokes function-call/type-conversion

                  Predicate Information (identified by plan id)
 ---------------------------------------------------------------------------------
   4 --Partition Iterator
        Iterations: 6
   5 --Partitioned Seq Scan on public.t_ddw_f10_op_cust_asset_mon b1
        Filter: ((b1.year_mth < 202007::bigint) AND ((b1.year_mth + 1) >= 202007))
        Rows Removed by Filter: 81920000
        Partitions Selected by Static Prune: 1..6
```

优化后

分析语句的执行计划，查看执行计划中的SQL自诊断信息，发现如下诊断信息：

```
                            SQL Diagnostic Information
 ------------------------------------------------------------------------------------------
 Partitioned table unprunable Qual
        table public.t_ddw_f10_op_cust_asset_mon b1:
        left side of expression "((year_mth + 1) > 202008)" invokes function-call/type-conversion
```

Filter条件中存在表达式(year_mth + 1) > 202008，这种**表达式一侧不是单纯的分区键、而是包含分区键的表达式的Filter条件是不能用来剪枝的**，因而导致查询语句扫描了几乎整个分区表的数据。

跟原始SQL语句对比，可以确定表达式 '(year_mth + 1) > 202008' 是从表达式 'b1.year_mth + 1 > substr('20200822',1 ,6)' 衍生而来，按照诊断信息把修改SQL语句为如下方式：

```
SELECT
   count(1)
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth <= substr('20200822',1 ,6 )
AND b1.year_mth > cast(substr('20200822',1 ,6 ) AS int) - 1;
```

改写之后，SQL语句的执行信息如下，可以看到不剪枝告警已经消除，剪枝后需要扫描分区数为1，执行时间从10s提升至3s。

```
EXPLAIN (analyze ON, verbose ON)
SELECT
   count(1)
```


版权所有 © 华为云计算技术有限公司

```
FROM t_ddw_f10_op_cust_asset_mon b1
WHERE b1.year_mth < substr('20200722',1 ,6 )
AND b1.year_mth >= cast(substr('20200722',1 ,6 ) AS int) - 1;
                                                                    QUERY PLAN
--------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
  id |                              operation                              |       A-time        |  A-rows  |  E-rows  | E-
distinct | Peak Memory  | E-memory | A-width | E-width |  E-costs
 ----+---------------------------------------------------------------------------------+----------------------+----------
+----------+------------+--------------+----------+---------+---------+-----------
   1 | ->  Aggregate                                                        | 3009.796            |        1 |        1 |            |
32KB         |          |          |       8 | 501541.70
   2 |    ->  Streaming (type: GATHER)                                      | 3009.718            |        4 |        4
|           | 136KB       |          |          |       8 | 501541.70
   3 |       ->  Aggregate                                                  | [2675.509, 3003.298] |        4 |        4
|           | [24KB, 24KB] | 1MB      |          |       8 | 501531.70
   4 |          ->  Partition Iterator                                      | [1820.725, 2053.836] | 16384000 |
16380697 |            | [16KB, 16KB] | 1MB      |          |       0 | 491293.75
   5 |             ->  Partitioned Seq Scan on public.t_ddw_f10_op_cust_asset_mon b1 | [1420.972, 1590.083] |
16384000 | 16380697 |            | [16KB, 16KB] | 1MB      |          |       0 | 491293.75

             Predicate Information (identified by plan id)
 ---------------------------------------------------------------------------
   4 --Partition Iterator
         Iterations: 1
   5 --Partitioned Seq Scan on public.t_ddw_f10_op_cust_asset_mon b1
         Filter: ((b1.year_mth < 202007::bigint) AND (b1.year_mth >= 202006))
         Partitions Selected by Static Prune: 6
```

14.5.11 案例：改写 SQL 消除 in-clause

优化前

in-clause/any-clause是常见的SQL语句约束条件，有时in或any后面的clause都是常量，类似于：

```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in ('20120405','20130405');
```

或者

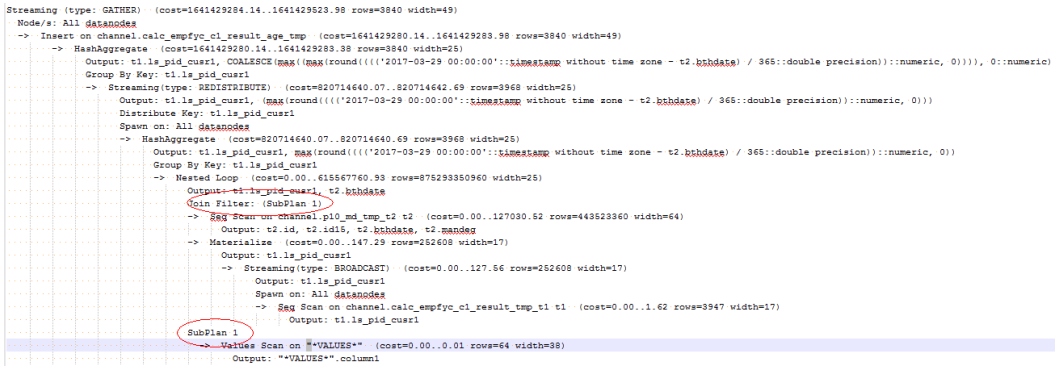
```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in any('20120405','20130405');
```

但是也有一些如下的特殊用法：

```
SELECT
ls_pid_cusr1,COALESCE(max(round((current_date-bthdate)/365)),0)
FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
WHERE t1.ls_pid_cusr1 = any(values(id),(id15))
GROUP BY ls_pid_cusr1;
```

其中，id、id15为p10_md_tmp_t2中的两列，“t1.ls_pid_cusr1 = any(values(id),(id15))”等价于“t1.ls_pid_cusr1 = id or t1.ls_pid_cusr1 = id15”。

因此join-condition实质上是一个不等式，这种不等值的join操作必须走nestloop，对应执行计划如下：



优化后

测试发现由于两表结果集过大，导致nestloop耗时过长，超过一小时未返回结果，因此性能优化的关键是消除nestloop，让join走更高效的hashjoin。从语义等价的角度消除any-clause，SQL改写如下：

```
select
ls_pid_cusr1,COALESCE(max(round(ym/365)),0)
from
(
	(
		SELECT
			ls_pid_cusr1,(current_date-bthdate) as ym
		FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
		WHERE t1.ls_pid_cusr1 = t2.id and t1.ls_pid_cusr1 != t2.id15
	)
	union all
	(
		SELECT
			ls_pid_cusr1,(current_date-bthdate) as ym
		FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
		WHERE t1.ls_pid_cusr1 = id15
	)
)
GROUP BY ls_pid_cusr1;
```

注意：尽量使用union all代替union。union在合并两个集合时会执行去重操作，而union all则直接将两个结果集合并、不执行去重。执行去重会消耗大量的时间，因此，在一些实际应用场景中，如果通过业务逻辑已确认两个集合不存在重叠，可用union all替代union以便提升性能。

优化后的SQL查询由两个等值join的子查询构成，而每个子查询都可以走更适合此场景的hashjoin。优化后的执行计划如下

```
 id |                                operation                                |          A-time          | A-rows |  E-rows   |   Peak Memory    | E-memory | A-width
----+-------------------------------------------------------------------------+--------------------------+--------+-----------+------------------+----------+---------
  1 | ->  Streaming (type: GATHER)                                            | 6737.281                 |      0 |       192 | 292KB            |          |
  2 |    ->  Insert on channel.calc_empfyc_c1_result_age_tmp                  | [4665.024,4990.666]      |      0 |       192 | [1108KB, 1108KB] | 1MB      |
  3 |       ->  HashAggregate                                                 | [4664.996,4990.641]      |      0 |       192 | [12KB, 12KB]     | 16MB     |
  4 |          ->  Streaming(type: REDISTRIBUTE)                              | [4664.991,4990.637]      |      0 |      3392 | [2090KB, 2090KB] | 1MB      |
  5 |             ->  HashAggregate                                           | [3416.939,4958.348]      |      0 |      3392 | [14KB, 14KB]     | 16MB     |
  6 |                ->  Append                                               | [3416.936,4958.340]      |      0 |      4011 | [1KB, 1KB]       | 1MB      |
  7 |                   ->  Hash Join (8,9)                                   | [2011.226,3080.697]      |      0 |      3947 | [6KB, 6KB]       | 1MB      |
  8 |                      ->  Seq Scan on channel.p10_md_tmp_t2 t2           | [803.782,1238.984]       | 443525717 | 443523360 | [12KB, 12KB]     | 1MB      |
  9 |                      ->  Hash                                           | [4.357,328.979]          | 252608 |    252608 | [482KB, 482KB]   | 16MB     | [35,39]
 10 |                         ->  Streaming(type: BROADCAST)                  | [2.345,326.320]          | 252608 |    252608 | [2090KB, 2090KB] | 1MB      |
 11 |                            ->  Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1 | [0.011,0.030]    |   3947 |      3947 | [11KB, 11KB]     | 1MB      |
 12 |                   ->  Hash Join (13,14)                                 | [1376.258,2066.110]      |      0 |        64 | [5KB, 5KB]       | 1MB      |
 13 |                      ->  Seq Scan on channel.p10_md_tmp_t2 t2           | [777.552,1388.499]       | 443525717 | 443523360 | [12KB, 12KB]     | 1MB      |
 14 |                      ->  Hash                                           | [2.812,4.217]            | 252608 |    252608 | [482KB, 482KB]   | 16MB     | [23,27]
 15 |                         ->  Streaming(type: BROADCAST)                  | [1.276,1.868]            | 252608 |    252608 | [2090KB, 2090KB] | 1MB      |
 16 |                            ->  Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1 | [0.010,0.033]    |   3947 |      3947 | [11KB, 11KB]     | 1MB      |
(16 rows)
```

优化后，从超过1个小时未返回结果优化到7s返回结果。

14.5.12 案例：使用 partial cluster key

列存表可以选取某一列或几列设置为partial cluster key(column_name[, ...])。在导入数据时，按设置的列进行局部排序（默认每70个CU即420万行排序一次），生成的CU

 版权所有 © 华为云计算技术有限公司

会聚集在一起，即CU的min，max会在一个较小的区间内。当查询时，where条件含有这些列时，可产生良好的过滤效果。

优化前

未使用partial cluster key。表定义如下：

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
)
with (orientation = column)
distribute by hash(L_ORDERKEY);

select
sum(l_extendedprice * l_discount) as revenue
from
lineitem
where
l_shipdate >= '1994-01-01'::date
and l_shipdate < '1994-01-01'::date + interval '1 year'
and l_discount between 0.06 - 0.01 and 0.06 + 0.01
and l_quantity < 24;
```

导入数据后执行查询，查看执行时间：

图 14-24 未使用 partial cluster key

```
 id |                  operation                  |       A-time       | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
----+---------------------------------------------+--------------------+--------+--------+----------------+---------+---------+-----------
  1 | ->  Row Adapter                             | 1653.156           |      1 |      1 | 12KB           |         |      44 | 205803.90
  2 |    ->  Vector Aggregate                     | 1653.146           |      1 |      1 | 184KB          |         |      44 | 205803.90
  3 |       ->  Vector Streaming (type: GATHER)   | 1653.070           |      1 |      1 | 174KB          |         |      44 | 205803.90
  4 |          ->  Vector Aggregate               | [1481.497,1481.497] |      1 |      1 | [225KB, 225KB] |         |      44 | 205803.84
  5 |             ->  CStore Scan on public.lineitem | [1405.004,1405.004] | 114160 | 111485 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```

图 14-25 未使用 partial cluster key 后 CU 加载情况

```
 5 --CStore Scan on public.lineitem
       datanode1 (actual time=40.623..1405.004 rows=114160 loops=1)
       datanode1 (RoughCheck CU: CUNone: 0, CUSome: 101)
       datanode1 (LLVM Optimized)
       datanode1 (Buffers: shared hit=18385 read=23)
       datanode1 (CPU: ex c/r=31917, ex cyc=3643646206, inc cyc=3643646206)
```

优化后

where条件中l_shipdate和l_quantity的distinct值数量较少且可以做min max过滤，将字段l_shipdate、l_quantity设置为PCK修改表定义如下：

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
```

 版权所有 © 华为云计算技术有限公司

```
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
, partial cluster key(l_shipdate, l_quantity)
)
with (orientation = column)
distribute by hash(L_ORDERKEY);
```

重新导入数据后执行查询，查看执行时间：

图 14-26 使用 partial cluster key

```
id |                operation                |      A-time       | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
---+-----------------------------------------+-------------------+--------+--------+----------------+---------+---------+----------
 1 | ->  Row Adapter                         | 459.539           |      1 |      1 | 12KB           |         |      44 | 205693.85
 2 |    ->  Vector Aggregate                 | 459.528           |      1 |      1 | 184KB          |         |      44 | 205693.85
 3 |       ->  Vector Streaming (type: GATHER) | 459.452         |      1 |      1 | 174KB          |         |      44 | 205693.85
 4 |          ->  Vector Aggregate           | [285.177,285.177] |      1 |      1 | [225KB, 225KB] |         |      44 | 205693.79
 5 |             ->  CStore Scan on public.lineitem | [249.757,249.757] | 114160 |  89475 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```

图 14-27 使用 partial cluster key 后 CU 加载情况

```
 5 --CStore Scan on public.lineitem
       datanode1 (actual time=23.017..249.757 rows=114160 loops=1)
       datanode1 (RoughCheck CU: CUNone: 84, CUSome: 17)
       datanode1 (LLVM Optimized)
       datanode1 (Buffers: shared hit=2853 read=23)
       datanode1 (CPU: ex c/r=5673, ex cyc=647656146, inc cyc=647656146)
```

使用partial cluster key后，5-- CStore Scan on public.lineitem的时间减少了1.2s，得益于有84个CU被过滤掉了。

优化建议

- 选取partial cluster key列。
  - 列存表支持创建partial cluster key的类型character varying(n), varchar(n), character(n), char(n), text, nvarchar2, timestamp with time zone, timestamp without time zone, date, time without time zone, time with time zone。
  - 数据的distinct值数量较少，这样能产生较好的过滤效果。
  - 出现在查询where条件中，优先选取能过滤大量数据的列。
  - partial cluster key中设置多个列时，是先按第一个列排序，当第一个列值相同时，使用第二列比较，后续列依次类推。推荐不要超出3个列。
- 添加partial cluster key后，优化导入性能。

  由于添加了partial cluster key，在导入时会增加排序计算，会对导入性能产生影响。当排序完全在内存中进行时影响较小，如果无法在内存中完成排序时，会下盘写临时文件，这时就会产生较大的影响。

 版权所有 © 华为云计算技术有限公司

排序使用的内存通过GUC参数psort_work_mem来设置，可以设置较大的值来使用更大的内存进行排序。

排序的数据量是通过表的存储参数PARTIAL_CLUSTER_ROWS来设置，降低这个数值，可减少一次排序的数据量。这个参数通常与存储参数MAX_BATCHROW配置使用。PARTIAL_CLUSTER_ROWS设置值必须是MAX_BATCHROW的整数倍，MAX_BATCHROW是设置单个CU中数据的最大行数。

14.5.13 案例：NOT IN 转 NOT EXISTS

NOT IN语句需要使用nestloop anti join来实现，而NOT EXISTS则可以通过hash anti join来实现。在join列不存在null值的情况下，not exists和not in等价。因此在确保没有null值时，可以通过将not in转换为not exists，通过生成hash join来提升查询效率。

优化前

创建两个基表t1、t2：

```
CREATE TABLE t1(a int, b int, c int not null) WITH(orientation=row);
CREATE TABLE t2(a int, b int, c int not null) WITH(orientation=row);
```

执行下列SQL语句，查询NOT IN的执行计划：

```
EXPLAIN VERBOSE SELECT * FROM t1 WHERE t1.c NOT IN (SELECT t2.c FROM t2);
```

返回结果如图：

```
                                  QUERY PLAN
------------------------------------------------------------------------------------------
 id |               operation               | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------+--------+------------+---------+---------
  1 | ->  Streaming (type: GATHER)          |      6 |            |      12 | 78.98
  2 |    ->  Nested Loop Anti Join (3, 4)   |      6 |            |      12 | 64.98
  3 |       ->  Seq Scan on public.t1       |     60 |            |      12 | 18.18
  4 |       ->  Materialize                 |    360 |            |       4 | 30.75
  5 |          ->  Streaming(type: BROADCAST) |    360 |            |       4 | 30.45
  6 |             ->  Seq Scan on public.t2 |     60 |            |       4 | 18.18

            Predicate Information (identified by plan id)
-------------------------------------------------------------------------
  2 --Nested Loop Anti Join (3, 4)
        Join Filter: ((t1.c = t2.c) OR (t1.c IS NULL) OR (t2.c IS NULL))
```

从返回结果可知执行计划走NestLoop，因为NULL值跟任意值的OR运算结果都是NULL，WHERE条件表达式：

```
t1.c NOT IN (SELECT t2.c FROM t2)
```

等价于：

```
t1.c <> ANY(t2.c) AND t1.c IS NOT NULL AND ANY(t2.c) IS NOT NULL
```

优化后

可将查询可以修改为：

```
SELECT * FROM t1 WHERE NOT EXISTS (SELECT * FROM t2 WHERE t2.c = t1.c);
```

执行下列语句，查询NOT EXISTS的执行计划：

```
EXPLAIN VERBOSE SELECT * FROM t1 WHERE NOT EXISTS (SELECT 1 FROM t2 WHERE t2.c = t1.c);
```

 版权所有 © 华为云计算技术有限公司

```
HINT:  Do analyze for them in order to generate optimized plan.
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------
 id |                operation                 | E-rows | E-distinct | E-width | E-costs
----+------------------------------------------+--------+------------+---------+---------
  1 | ->  Streaming (type: GATHER)             |      6 |            |      12 | 54.56
  2 |    ->  Hash Anti Join (3, 5)             |      6 |            |      12 | 40.56
  3 |       ->  Streaming(type: REDISTRIBUTE)  |     60 | 10         |      12 | 20.12
  4 |          ->  Seq Scan on public.t1       |     60 |            |      12 | 18.18
  5 |       ->  Hash                           |     59 | 10         |       4 | 20.12
  6 |          ->  Streaming(type: REDISTRIBUTE) |     60 |            |       4 | 20.12
  7 |             ->  Seq Scan on public.t2    |     60 |            |       4 | 18.18

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Hash Anti Join (3, 5)
        Hash Cond: (t1.c = t2.c)
```


版权所有 © 华为云计算技术有限公司

15 DWS AI

15.1 DWS AI 概述

随着生成式AI与智能分析技术的广泛应用，DWS在AI相关能力上也持续演进，**自9.1.1.200及以上集群版本**提供向量计算、库内推理、MCP Server等AI功能。DWS AI将数据分析、语义搜索与智能预测融合到统一平台，实现“数据即智能”。支持从数据挖掘、解析到智能推理与搜索的完整流程，助力业务实现数据价值的最大化，从而推动决策优化和业务增长，为企业提供更实时、更低成本、更智能的决策支持能力，全面满足用户在AI场景中的多样化需求。

- 向量计算

  DWS集成了插件pgvector，提供高维向量的存储和检索能力。它让数据库能够直接进行向量相似度搜索（如 cosine、L2、inner product），并支持IVFFlat、HNSW等主流高效索引结构，大幅提升海量向量的查询性能。结合DWS的库内推理能力，用户可以在数据库内部实现语义检索、推荐、RAG（检索增强生成）等AI应用场景。

- 库内推理

  DWS通过集成pgai插件，用户无需依赖外部AI平台，即可在数据库内直接调用LLM（大语言模型）和Embedding（嵌入模型）模型，这不仅简化了复杂的RAG应用流程，更显著提升响应速度与数据库安全性，为用户带来了更高效、更灵活的数据分析体验。

- MCP Server

  DWS基于MCP协议实现了DWS MCP Server，能够让AI模型和智能代理（agents）与DWS集群无缝连接，并提供语句执行、元数据查询等功能可让用户借助大模型能力，通过自然语言直接操作数据库，达到自然语言向SQL的内置转换，实现一键式后端SQL查询，用户端能够直接接收查询结果，为用户提供了更灵活和智能的数据处理能力。

- 大模型特征算子

  大模型特征算子以扩展（extension）形式封装为文件置于DWS系统，通过CREATE EXTENSION命令创建，大模型直接调用封装好的Python UDF函数，实现智能化的数据处理与分析。

 版权所有 © 华为云计算技术有限公司

15.2 向量计算

DWS的9.1.1.200及以上集群版本支持通过集成pgvector插件（有关详细的规格和语法请参考**pgvector**），提供向量存储和检索能力。

注意事项

- **该功能9.1.1.200集群版本中为Beta特性，如需使用请联系技术支持；9.1.1.300及以上集群版本中已正式商用**；如需使用，请联系技术支持设置GUC参数**feature_support_options**为**enable_pgvector**。
- 在存算分离架构下，pgvector扩展功能（CREATE EXTENSION语句）仅支持在主VW上执行。

pgvector 插件能力

- 支持向量数据类型：可在表中建立 vector(n) 列，用于存储高维向量。
- 向量相似度计算：支持多种距离或相似度运算符，包括 L2（欧氏距离）、Inner Product、Cosine 距离、L1（曼哈顿距离）、甚至 Hamming/Jaccard 等。
- 支持最近邻搜索（Nearest Neighbour Search, NNS），包括准确搜索和近似搜索（ANN）。
- 支持向量索引： IVFFlat、HNSW 索引结构，用于加速大规模向量检索。

说明

pgvector插件功能当前版本仅支持行存表。

向量数据类型

数据类型	精度	支持维度	典型用途
vector	单精度浮点，4 字节。	<=16000	传统embedding。
halfvec	半精度浮点（16 位浮点数，每元素 2 字节）。	<=16000	高维embedding、内存敏感场景。
bit	二进制向量类型（bit 向量）。	<= 64000	二进制哈希、图像 / 特征布尔表示。
sparsevec	稀疏向量类型。	<=16000非零维度	文本TF-IDF稀疏特征、超高维稀疏embedding。

版权所有 © 华为云计算技术有限公司

说明

以上向量支持的维度为数据类型本身支持的维度限制，ivfflat及hnsw索引对向量维度有额外支持维度范围：

- vector：<=2000
- halfvec：<=4000
- bit：<=64000
- sparsevec：<=1000非零维度（仅hnsw索引支持）

向量距离/相似度函数

操作符	距离 / 相似度类型	说明
<->	Euclidean距离 (L2)	欧氏距离。
<#>	（负）内积 (Inner Product)	内积。
<=>	Cosine距离	向量方向差异。
<+>	L1距离 (Taxicab)	向量坐标绝对差之和。
<~>	Hamming距离	用于二进制bit向量，计算位不同数目。
<%>	Jaccard距离	用于二进制bit向量/集合表示，衡量交集/并集差异。

索引类型

索引类型	原理	构建成本	查询速度 / 召回 (speed- recall)	主要调优参数
IVFFlat	将向量分成若干“lists”（簇/桶），查询时只扫描部分 lists	构建快、内存使用低。	查询速度中等；召回率略低。扫描范围受“probes”参数影响。	lists（簇数）、probes（扫描簇数）、maxProbes（最大扫描簇数）。
HNSW	基于图结构（多层可导航小世界图），通过跳跃访问近邻点	构建较慢、内存占用较高。	查询速度快、召回率高；尤其适合高维、低延迟场景。	m（每层最大连接数）、ef_construction（构建候选数）、ef_search（查询候选数）、max_scan_tuples（最大扫描节点数）、mem_scan_multiplier（内存动态扩张率）。

版权所有 © 华为云计算技术有限公司

pgvector 插件使用

1. 初始化插件。
```
CREATE EXTENSION pgvector;
```
2. 创建一个维度为3的向量列。
```
CREATE TABLE items (id bigserial PRIMARY KEY, embedding vector(3));
```
3. 插入向量数据。
```
INSERT INTO items (embedding) VALUES
('[1, 2, 3]'),
('[0.5, 0.1, 0.9]'),
('[3, 2, 1]'),
('[0, 0, 1]'),
('[1.1, 2.2, 3.3]');
```
4. L2距离查询最近相邻向量。
```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
 id |   embedding
----+---------------
  3 | [3,2,1]
  1 | [1,2,3]
  5 | [1.1,2.2,3.3]
  2 | [0.5,0.1,0.9]
  4 | [0,0,1]
(5 rows)
```
5. 查询距离。
```
SELECT embedding <-> '[3,1,2]' AS distance FROM items;
     distance
------------------
 2.44948974278318
 2.87576084849969
  1.4142135623731
  3.3166247903554
 2.59615095306844
(5 rows)
```

说明

对于内积（inner product），用 -1 * inner product来获取实际距离：

```
SELECT (embedding <#> '[3,1,2]') * -1 AS inner_product FROM items;
```

对于余弦距离（cosine distance）， 用1 - cosine distance来获取实际距离：

```
SELECT 1 - (embedding <=> '[3,1,2]') AS cosine_similarity FROM items;
```

6. 聚合计算。
```
SELECT AVG(embedding) FROM items;
       avg
------------------
 [1.12,1.26,1.84]
(1 row)

SELECT SUM(embedding) FROM items;
      sum
---------------
 [5.6,6.3,9.2]
(1 row)
```

索引使用

pgvector默认使用精确近邻搜索，提供百分之百召回率但查询速度较慢。可以按需使用近似相邻搜索索引来牺牲部分召回率以提高查询速度。不同于传统索引，近似搜索索引可能会返回不同的查询结果。pgvector目前支持的索引类型包括HNSW和IVFFlat。

- **HNSW**

 版权所有 © 华为云计算技术有限公司

a. 添加索引。

```
-- L2 distance
CREATE INDEX ON items USING hnsw (embedding vector_l2_ops);
-- Inner product
CREATE INDEX ON items USING hnsw (embedding vector_ip_ops);
-- Cosine distance
CREATE INDEX ON items USING hnsw (embedding vector_cosine_ops);
-- L1 distance
CREATE INDEX ON items USING hnsw (embedding vector_l1_ops);
-- Hamming distance
CREATE INDEX ON items USING hnsw (embedding bit_hamming_ops);
-- Jaccard distance
CREATE INDEX ON items USING hnsw (embedding bit_jaccard_ops);
```

说明

根据使用向量类型使用对应的距离函数，例如：halfvec向量L2距离使用halfvec_l2_ops，sparsevec向量类型使用 sparsevec_l2_ops。

b. 构建参数。

m - 图层最大连接数 (默认值16)。

ef_construction - 索引构建候选数 (默认值64)。

```
CREATE INDEX ON items USING hnsw (embedding vector_l2_ops) WITH (m = 16,
ef_construction = 64);
```

说明

更大的构建选项值能够提供更高的召回率，但索引的构建以及数据插入时间会增加。

- **IVFFlat**

a. 添加索引

```
-- L2 distance
CREATE INDEX ON items USING ivfflat (embedding vector_l2_ops) WITH (lists = 100);
-- Inner product
CREATE INDEX ON items USING ivfflat (embedding vector_ip_ops) WITH (lists = 100);
-- Cosine distance
CREATE INDEX ON items USING ivfflat (embedding vector_cosine_ops) WITH (lists = 100);
-- Hamming distance
CREATE INDEX ON items USING ivfflat (embedding bit_hamming_ops) WITH (lists = 100);
```

b. 构建参数

lists - 构建索引时计算的聚簇的数量，值越大召回率越高，索引构建时间越长。

说明

- 建议在导入数据后再创建IVFFlat索引。
- 选择合适的lists值，推荐1M以下数据量使用**行数 /1000**，1M以上数据量采用**sqrt**（行数）。

- **索引参数**

– 迭代扫描模式

HNSW和IVFFlat索引都支持迭代扫描模式，自动扫描更多索引来获取足够的数据量。

通过参数hnsw_iterative_scan和ivfflat_iterative_scan控制开关。

参数取值：off（默认值）、relaxed_order、strict_order（仅HNSW索引支持strict_order）。

– HNSW查询选项

 版权所有 © 华为云计算技术有限公司

hnsw_options：包含三个数值，控制查询参数ef_search（查询候选数）、max_scan_tuples（最大扫描节点数）及mem_scan_multiplier（内存动态扩张率）。

默认值：'40, 20000, 1.0'

- IVFFlat查询选项

  ivfflat_options：包含两个数值，控制probes（扫描簇数）和maxProbes（最大扫描簇数）。

  默认值：'1, 32768'

15.3 库内推理

随着人工智能技术的迅猛发展，AI在各行各业中正逐渐渗透，成为推动技术创新和业务发展的关键力量。从自然语言处理到计算机视觉，再到深度学习，AI正以前所未有的速度改变着我们的工作和生活方式。特别是在数据管理和处理领域，AI技术的应用使得信息处理变得更加智能、高效和精准。在数据库的智能化的趋势中，pgai插件作为一项面向数据库的AI增强解决方案，能够将先进AI能力集成至数据操作流程中，实现更智能、高效和精准的数据处理。

DWS作为一种高性能的数据仓库解决方案，一直致力于为用户提供更强大的数据处理能力。随着AI技术在数据分析和决策中的重要性日益凸显，DWS的9.1.1.200及以上集群版本通过集成pgai插件，提供更加智能化的数据分析服务，实现数据即推理、即处理的全新操作模式。这不仅能够提升数据处理的速度和效率，还能够帮助企业在大数据分析中快速获取洞察，提升决策质量。

集成pgai插件后，DWS用户无需依赖外部AI平台，即可在数据库中直接进行LLM和Embedding模型调用，简化了复杂的RAG应用流程，也为用户带来了更高效、更灵活的数据分析体验。这一创新举措标志着DWS在智能数据处理方面迈出了重要的一步，也为未来的AI与数据库融合奠定了坚实的基础。

注意事项

- **该功能9.1.1.200集群版本中为Beta特性。**如需使用，请联系技术支持**安装pgai包**并设置GUC参数**feature_support_options**为**enable_pgai_extension。**
- 在存算分离架构下，pgai扩展功能（CREATE EXTENSION语句）仅支持在主VW上执行。
- 由于受到tiktoken库的限制，ai.openai_tokenize和ai.openai_detokenize两个函数必须依赖OpenAI大模型服务才能正常使用。其他函数均可通过设置base-url和api-key直接使用。
- 推荐购买支持OpenAI大模型的华为云MaaS服务。

库内推理的 Function 列表

通过执行“CREATE EXTENSION ai;”来创建pgai扩展（另外由于该扩展依赖于pgvector，因此需要优先创建pgvector扩展），当前DWS支持的库内推理的Function列表如下：

版权所有 © 华为云计算技术有限公司

表 15-1 库内推理的 Function 列表

Function名称	描述	备注
ai.set_func_model	设置Function默认使用的模型名称。	-
ai.dws_pgai_encrypt_info	设置并且加密所有Function使用的baseurl和apikey。	-
ai.openai_tokenize	将文本转换为token。	仅支持openai提供的gpt系列模型服务。
ai.openai_detokenize	将token转换为文本。	仅支持openai提供的gpt系列模型服务。
ai.openai_list_models	显示可用模型列表。	部分大模型服务可能不支持该功能。
ai.openai_list_models_with_raw_response	显示可用模型列表，以json格式返回。	部分大模型服务可能不支持该功能。
ai.openai_embed	将文本转换成向量。	使用Embedding模型。
ai.openai_embed_with_raw_response	将文本转换成向量，以json格式返回。	使用Embedding模型。
ai.openai_chat_complete	与LLM交互。	使用LLM。
ai.openai_chat_complete_with_raw_response	与LLM交互，以json格式返回。	使用LLM。
ai.openai_moderate	将文本分类为是否有害。	使用moderate模型。
ai.openai_moderate_with_raw_response	将文本分类为是否有害，以json格式返回。	使用moderate模型。
ai.chunk_text	支持单个分隔符的文本分块。	-
ai.chunk_text_recursively	支持多个分隔符递归的文本分块。	-
ai.similarity	计算两个输入文本的相似度。	使用Embedding模型。
ai.vector_cosine_similarity	计算两个向量的文本相似度。	-
ai.classify	根据输入标签对文本进行分类。	使用LLM。
ai.extract	从输入文本中提取关键字的内容。	使用LLM。
ai.mask	从输入文本中脱敏关键字的内容。	使用LLM。

版权所有 © 华为云计算技术有限公司

Function名称	描述	备注
ai.fix_grammar	修正输入文本的语法。	使用LLM。
ai.summarize	生成输入文本的摘要。	使用LLM。
ai.translate	将输入文本翻译成指定语言。	使用LLM。
ai.rank	根据主题对多个文本进行相关性打分。	使用LLM。
ai.sentiment	对输入文本进行情感分析。	使用LLM。
ai.textfilter	根据给定条件对输入文本进行过滤。	使用LLM。

ai.set_func_model

描述：设置Function默认使用的模型名称。未设置前，对应Function默认使用的模型名称为NULL，此时无法使用Function；设置后，调用Function时优先使用默认模型。

参数说明：

- funcname：text，必填，Function的名称。比如：openai_embed、openai_chat_complete等需要使用LLM或者Embedding模型服务的函数。
- modelname：text，必填，使用的模型的名称。根据模型服务提供的模型名称确定。

返回值说明：无返回值。

使用示例：

```
SELECT ai.set_func_model('openai_embed', 'nomic-embed-text');
```

设置后可以通过以下语句查看当前function默认使用模型名称的情况。

```
SELECT * from ai.ai_model_info;
```

查询结果如下：

```
            func_name             |    model_name
----------------------------------------------------------------
 openai_tokenize                  |
 openai_detokenize                |
 openai_embed_with_raw_response   |
 openai_chat_complete             |
 openai_chat_complete_with_raw_response |
 openai_moderate                  |
 openai_moderate_with_raw_response |
 similarity                       |
 classify                         |
 extract                          |
 mask                             |
 fix_grammar                      |
 summarize                        |
 translate                        |
 rank                             |
 sentiment                        |
 textfilter                       |
```


版权所有 © 华为云计算技术有限公司

```
 openai_embed                | nomic-embed-text
(18 rows)
```

ai.dws_pgai_encrypt_info

描述：全局设置并且加密所有Function使用的模型服务的baseurl和apikey。

参数说明：

- base_url：text，必填， 使用的模型服务的baseurl。
- api_key：text，必填，使用的模型服务的apikey。

返回值说明：无返回值。

使用示例：

```
SELECT ai.dws_pgai_encrypt_info('https://example.com', 'your_api_key');
```

ai.openai_tokenize

描述：对于给定的模型，将文本转换为token。只能通过提供openai提供的系列模型服务使用。

参数说明：

- text_input：text，必填，输入文本内容。
- model：text，选填，使用的模型名称。默认为ai.set_func_model设置的模型名称。

返回值说明：返回int[]类型的token。

使用示例：

```
SELECT ai.openai_tokenize('have a test');
```

ai.openai_detokenize

描述：对于给定的模型，将token转换为文本。只能通过openai提供的系列模型服务使用。

参数说明：

- tokens：int[]，必填，需要转为文本的token数组。
- model：text，选填，使用的模型名称。默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的文本。

使用示例：

```
SELECT ai.openai_detokenize(array[15365, 23456, 29889, 11, 9906, 1917, 0]);
```

ai.openai_list_models

描述：显示模型服务平台支持的模型列表，部分模型平台可能不支持该功能。

无输入参数。

返回值说明：返回text[]类型的模型信息。

 版权所有 © 华为云计算技术有限公司

使用示例：

```
SELECT ai.openai_list_models();
                                                     openai_list_models
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------
{"{'id': 'bge-large', 'created': datetime.datetime(2023, 2, 28, 18, 56, 42, tzinfo=datetime.timezone.utc), 'owned_by': 'openai'}","{'id': 'deepseek-r1-distill-qwen-1.5b', 'created': datetime.datetime(2023, 2, 28, 18, 56, 42, tzinfo=datetime.timezone.utc), 'owned_by': 'openai'}"}
(1 row)
```

ai.openai_list_models_with_raw_response

描述：显示模型服务平台支持的模型列表，返回原始响应数据。

无输入参数。

返回值说明：返回text类型的模型信息。

使用示例：

```
SELECT ai.openai_list_models_with_raw_response();
                                                openai_list_models_with_raw_response
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------
{"data": [{"id": "deepseek-v3-0324", "object": "model", "created": 1677610602, "owned_by": "openai"},{"id": "nomic-embed-text", "object": "model", "created": 1677610602, "owned_by": "openai"}], "object": "list"}
(1 row)
```

ai.openai_embed

描述：将文本或者token(int[])转换成向量。

表 15-2 参数说明：

参数名称	参数类型	是否必填	参数说明
input_text | input_texts | input_tokens	text | text[] | int[]	必填	支持三种类型的输入，根据输入类型自动适配。
model	text	选填	使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text | jsonb | text，与输入类型对应。

使用示例：

输入text类型：

```
SELECT ai.openai_embed('have a test');
                    openai_embed
-----------------------------------------------------------------------
[0.012326963, 0.015280011, -0.17099911,..., -0.005275759, -0.03978255]
(1 row)
```

输入text[]类型：

```
SELECT ai.openai_embed(array['have a test1', 'have a test2']);
                                                      openai_embed
```

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
[{"index": 0, "embedding": [-.02023241, .01978845, -.1544127, ..., -.006075094, -.057604033]}","{"index": 1,
"embedding": [-.0066108834, -.0045409035, -.15522258, ..., .0038487795, -.03174798]}]
(1 row)
```

ai.openai_embed_with_raw_response

描述：将文本或者token转换为向量。

表 15-3 参数说明

参数名称	参数类型	是否必填	参数说明
input_text | input_texts | input_tokens	text | text[] | int[]	必填	支持三种类型的输入，根据输入类型自动适配。
model	text	选填	使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型，未经处理的API服务响应。

使用示例：

输入text类型：

```
SELECT ai.openai_embed_with_raw_response('have a test');
                                                    openai_embed_with_raw_response
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
 {"data": [{"index": 0, "object": "embedding", "embedding": [.012326963, .015280011, ..., -.005275759,
-.03978255]}], "model": "nomic-embed-text:latest", "usage": {"total_tokens": 3, "prompt_tokens": 3,
"completion_tokens": 0, "prompt_tokens_details": null, "completion_tokens_details": null}, "object": "list"}
(1 row)
```

输入text[]类型：

```
SELECT ai.openai_embed_with_raw_response(array['have a test1', 'have a test2']);
                                                    openai_embed_with_raw_response
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
 {"data": [{"index": 0, "object": "embedding", "embedding": [-.02023241, .01978845, ..., -.006075094,
-.057604033]}, {"index": 1, "object": "embedding", "embedding": [-.0066108834,
-.0045409035, ..., .0038487795, -.03174798]}], "model": "nomic-embed-text:latest", "usage": {"total_tokens":
8, "prompt_tokens": 8, "completion_tokens": 0, "prompt_tokens_details": null, "completion_tokens_details":
null}, "object": "list"}
(1 row)
```

ai.openai_chat_complete

描述：与LLM交互进行文本生成。

版权所有 © 华为云计算技术有限公司

表 15-4 参数说明

参数名称	参数类型	是否必填	参数说明
messages | input_text	jsonb | text	必填	支持两种类型的输入，与LLM交互。
model	text	选填	使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，直接返回LLM回答的所有内容。

使用示例：

输入jsonb类型：

```
SELECT ai.openai_chat_complete(
'[
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": "What is a Database?"},
{"role": "assistant", "content": "Database is a system software."}
]'::jsonb
);
                                                                                                   openai_chat_complete
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
A database (typically referring to a database management system, or DBMS) is a system software. Just as an operating system (OS) manages a computer’s hardware resources (CPU, memory, disk), this system software is specifically responsible for managing “data,” the core resource. It runs on top of the operating system and provides unified, efficient, and secure data access services to upper-layer application software (such as e-commerce systems, banking apps, and ERP systems).
(1 row)
```

输入text类型：

```
SELECT ai.openai_chat_complete('Do your know Database?');
                                                                                                   openai_chat_complete
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
 Yes, I’m very familiar with databases. Which specific module of the database would you like to explore in depth today? Have you encountered a specific kernel-level bug (such as a concurrency deadlock or memory out-of-bounds error), or would you like to discuss performance optimization strategies for a particular operator (such as SIMD acceleration or hash joins)? We can dive right into the code or architecture.
(1 row)
```

ai.openai_chat_complete_with_raw_response

描述：与LLM交互进行文本生成。

表 15-5 参数说明

参数名称	参数类型	是否必填	参数说明
messages | input	jsonb | text	必填	支持两种类型的输入，与LLM交互。
model	text	选填	使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型的数据，未经处理的api服务响应。

使用示例：

输入jsonb类型：

```
SELECT ai.openai_chat_complete_with_raw_response(
'[
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": "What is a Database?"},
{"role": "assistant", "content": "Database is a system software."}
]'::jsonb
);

                                                    openai_chat_complete_with_raw_response
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
{"id": "chatcmpl-a9298cb4ef", "model": "hosted_vllm/deepseek-r1-671b", "usage": {"tpot": 68, "ttft": 228, "total_tokens": 698, "prompt_tokens": 27, "completion_tokens": 671}, "object": "chat.completion", "choices": [{"index": 0, "message": {"role": "assistant", "content": "A database (typically referring to a database management system, or DBMS) is a system software. Just as an operating system (OS) manages a computer’s hardware resources (CPU, memory, disk), this system software is specifically responsible for managing “data,” the core resource. It runs on top of the operating system and provides unified, efficient, and secure data access services to upper-layer application software (such as e-commerce systems, banking apps, and ERP systems)."}, "finish_reason": "stop", "provider_specific_fields": {}}], "created": 1763017779450, "message": ""}
(1 row)
```

输入text类型：

```
SELECT ai.openai_chat_complete_with_raw_response('Do your know Database?');
                                                    openai_chat_complete_with_raw_response
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
{"id": "chatcmpl-cdff97c366", "model": "hosted_vllm/deepseek-r1-671b", "usage": {"tpot": 69, "ttft": 185, "total_tokens": 736, "prompt_tokens": 15, "completion_tokens": 721}, "object": "chat.completion", "choices": [{"index": 0, "message": {"role": "assistant", "content": "Yes, I’m very familiar with databases. Which specific module of the database would you like to explore in depth today? Have you encountered a specific kernel-level bug (such as a concurrency deadlock or memory out-of-bounds error), or would you like to discuss performance optimization strategies for a particular operator (such as SIMD acceleration or hash joins)? We can dive right into the code or architecture."}, "finish_reason": "timeout", "provider_specific_fields": {}}], "created": 1763017897477, "message": "Reach time limitation! Use stream request to avoid timeout."}
(1 row)
```

ai.openai_moderate

描述：判断文本内容是否存在有害性。

参数说明：

- input_text：text，必填，输入用于判断的文本内容。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型的数据，给出判断结果。

使用示例：

```
SELECT ai.openai_moderate('You're such a fool.');
```

 版权所有 © 华为云计算技术有限公司

ai.openai_moderate_with_raw_response

描述：判断文本内容是否存在有害性。

参数说明：

- input_text：text，必填，输入用于判断的文本内容。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型的数据，未经处理的api服务响应。

使用示例：

```
SELECT ai.openai_moderate_with_raw_response('You are such a fool.');
```

ai.chunk_text

描述：将文本按照分隔符进行分块处理。

参数说明：

- input：text，必填，输入用于分块的文本。
- chunk_size：int，选填，指定每个文本块的最大长度，默认为NULL。
- chunk_overlap：int，选填，相邻文本块之间的重叠字符数，默认为NULL。
- separator：text，选填，自定义的分隔符，默认为NULL。
- is_separator_regex：boolean，选填，判断separator（分隔符）是否为一个正则表达式，默认为false。

返回值说明：返回text[]类型的分块结果数据。

使用示例：

```
SELECT ai.chunk_text('A data warehouse is a centralized data storage system specifically designed to
collect, store, process, and analyze large volumes of data from multiple heterogeneous data sources. By
integrating information from different operating systems, transaction systems, and external data sources, it
enables enterprises to perform complex data analysis and reporting. Unlike traditional database systems,
data warehouses emphasize historical data and query efficiency, typically supporting business intelligence
(BI) and decision support systems (DSS).', 100, 10, '.');
                                          chunk_text
-----------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
{"(0, 'A data warehouse is a centralized data storage system specifically designed to collect, store, process,
and analyze large volumes of data from multiple heterogeneous data sources')","(1, 'By integrating
information from different operating systems, transaction systems, and external data sources, it enables
enterprises to perform complex data analysis and reporting')","(2, 'Unlike traditional database systems,
data warehouses emphasize historical data and query efficiency, typically supporting business intelligence
(BI) and decision support systems (DSS)')"}
(1 row)
```

ai.chunk_text_recursively

描述：将文本按照多个分隔符递归进行分块处理。

参数说明：

- input：text，必填，输入用于分块的文本。
- chunk_size：int，选填，指定每个文本块的最大长度，默认为NULL。
- chunk_overlap：int，选填，相邻文本块之间的重叠字符数，默认为NULL。

- separator：text[]，选填，自定义的分隔符，默认为NULL。
- is_separator_regex：boolean，选填，判断separator（分隔符）是否为一个正则表达式，默认为false。

返回值说明：返回text[]类型的分块结果数据。

使用示例：

```
SELECT ai.chunk_text_recursively('
A data warehouse is a centralized data storage system specifically designed to collect, store, process, and analyze large volumes of data from multiple heterogeneous data sources. By integrating information from different operating systems, transaction systems, and external data sources, it enables enterprises to perform complex data analysis and reporting. Unlike traditional database systems, data warehouses emphasize historical data and query efficiency, typically supporting business intelligence (BI) and decision support systems (DSS).
', 100, 10, array[',', '.']);
                                                    chunk_text_recursively
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
{"(0, 'A data warehouse is a centralized data storage system specifically designed to collect, store')","(1, ', store, process')","(2, ', and analyze large volumes of data from multiple heterogeneous data sources')","(3, '. By integrating information from different operating systems')","(4, ', transaction systems, and external data sources')","(5, ', it enables enterprises to perform complex data analysis and reporting')","(6, '. Unlike traditional database systems')","(7, ', data warehouses emphasize historical data and query efficiency')","(8, ', typically supporting business intelligence (BI) and decision support systems (DSS).')"}
(1 row)
```

ai.similarity

描述：计算两个输入文本的相似度。

参数说明：

- input_text1：text，必填，用于计算相似度的文本1。
- input_text2：text，必填，用于计算相似度的文本2。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回float类型的数据，给出相似度结果，范围在[-1, 1]之间，越接近1相似度越高。

使用示例：

```
SELECT ai.similarity('I like apple.', 'I like pen.');
similarity
------------
 .5875276
(1 row)
```

ai.vector_cosine_similarity

描述：计算两个输入向量的相似度。

参数说明：

- input_vec1：text，必填，用于计算相似度的向量1。
- input_vec2：text，必填，用于计算相似度的向量2。

返回值说明：返回float类型的数据，给出相似度结果，范围在[-1, 1]之间，越接近1相似度越高。

使用示例：

```
SELECT ai.vector_cosine_similarity('[1.0, 2.0, 3.0]', '[-2.0, 5.0, -3.0]');
vector_cosine_similarity
------------------------
-.043355495
(1 row)
```

ai.classify

描述：根据指定标签对输入文本进行分类。

参数说明：

- input_text：text，必填，需要分类的文本。
- category：text[]，必填，文本归类的标签集合。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，内容为category中的一个标签。

使用示例：

```
SELECT ai.classify('The football match between Team A and Team B was thrilling. The game ended with a 3-2 victory for Team A, and the players celebrated with their fans.', array['Sports', 'Technology', 'Culture']);
classify
---------
Sports
(1 row)
```

ai.extract

描述：从输入文本中提取特定标签的相关信息。

参数说明：

- input_text：text，必填，需提取信息的文本。
- extract_keywords：text[]，必填，需要从文本中提取的关键字信息。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型的数据，给出每个标签以及对应的标签信息。

使用示例：

```
SELECT ai.extract('Name: John Smith, Gender: Male, Age: 30, Email: john.smith@example.com, Phone: +1 555-1234, Experience: Highly experienced professional', array['Name', 'Gender', 'Email']);
                                extract
-----------------------------------------------------------------------------
{"Name": "John Smith", "Email": "john.smith@example.com", "Gender": "Male"}
(1 row)
```

ai.mask

描述：从输入文本中将指定的关键信息内容脱敏。

参数说明：

- input_text：text，必填，需要脱敏的输入文本。
- mask_keywords：text[]，必填，需要从文本中脱敏的关键信息。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

 版权所有 © 华为云计算技术有限公司

返回值说明：返回text类型的数据，给出脱敏关键信息后的文本。

使用示例：

```
SELECT ai.mask('Name: John Smith, Gender: Male, Age: 30, Email: john.smith@example.com, Phone: +1 555-1234, Experience: Highly experienced professional', array['Name', 'Gender', 'Email']);
                                                              mask
---------------------------------------------------------------------------------------------------------------------------------
-
Name: [#####], Gender: [#####], Age: 30, Email: [#####], Phone: +1 555-1234, Experience: Highly experienced professional
(1 row)
```

ai.fix_grammar

描述：修正输入文本的语法错误。

参数说明：

- input_text：text，必填，需要修正语法的输入文本。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，给出修正语法后的文本内容。

使用示例：

```
SELECT ai.fix_grammar('She do not like him.');
      fix_grammar
-----------------------
She does not like him.
(1 row)
```

ai.summarize

描述：生成输入文本的摘要。

参数说明：

- input_text：text，必填，需要生成文本摘要的输入文本。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，给出文本摘要内容。

使用示例：

```
SELECT ai.summarize('
Huawei Cloud GaussDB is a high-performance, distributed database service launched by Huawei, supporting multiple database engines including relational databases (such as MySQL and PostgreSQL) and non-relational databases. GaussDB delivers elasticity, high availability, and auto-scaling capabilities to meet enterprises'' demanding requirements for database performance, reliability, and flexibility.
As Huawei Cloud''s flagship database, GaussDB integrates an advanced distributed architecture that supports automatic fault recovery and data backup to ensure data security. Simultaneously, it optimizes query performance and reduces operational costs through intelligent scheduling and resource management technologies. GaussDB also excels in high concurrency and high throughput, making it ideal for large-scale data processing and high-load applications across industries such as finance, e-commerce, and manufacturing.
Furthermore, through deep integration with other Huawei Cloud services, GaussDB delivers comprehensive data processing solutions to empower enterprises in digital transformation and intelligent management. It not only fulfills fundamental database requirements but also supports AI and big data applications, further enhancing the value of enterprise data.
');
                                                                      summarize
---------------------------------------------------------------------------------------------------------------------------------
```

 版权所有 © 华为云计算技术有限公司

```
--------------------------------------------------
Huawei Cloud GaussDB is a high-performance, distributed database service supporting relational and non-relational databases, offering elasticity, high availability, and auto-scaling to meet enterprise needs for performance, reliability, and flexibility. It integrates an advanced distributed architecture for automatic fault recovery, data backup, and optimized query performance with cost reduction. Its high concurrency and throughput make it suitable for large-scale data processing and high-load applications in industries like finance and e-commerce. Integrated with other Huawei Cloud services, it provides comprehensive data processing solutions, supporting AI and big data to drive digital transformation and intelligent management.
(1 row)
```

ai.translate

描述：将输入文本翻译成指定语言。

参数说明：

- input_text：text，必填，输入需翻译的文本。
- target_language: text，必填，翻译后的目标语言。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，给出翻译后的文本。

ai.rank

描述：根据主题对多个输入文本进行相关性打分。

参数说明：

- input_topic：text，必填，主题的内容。
- input_texts: text[]，必填，多个文本内容。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回jsonb类型的数据，给出每个文本字段的相关性评分，取值为[0, 1]，值越大相关性越高。

使用示例：

```
SELECT ai.rank('The Future Development of Artificial Intelligence.',
array['The rapid advancement of artificial intelligence has already made a significant impact across multiple industries, including healthcare and finance.',
'Machine learning and deep learning are core technologies of artificial intelligence, and an increasing number of companies are beginning to apply these technologies.',
'Forecast data for the global technology market in 2023 indicates that the cloud computing market is experiencing steady growth.']);
                                                 rank
--------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------
{"Forecast data for the global technology market in 2023 indicates that the cloud computing market is experiencing steady growth.": 3.0, "The rapid advancement of artificial intelligence has already made a significant impact across multiple industries, including healthcare and finance.": 8.0, "Machine learning and deep learning are core technologies of artificial intelligence, and an increasing number of companies are beginning to apply these technologies.": 7.0}
(1 row)
```

ai.sentiment

描述：对输入文本进行情感分析。判断文本的内容是positive（正面），negative（负面），neutral（中性）还是mixed（混合）。

参数说明：

- input_text：text，必填，需要分析的文本内容。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回text类型的数据，给出文本内容的情感判断结果。

使用示例：

```
SELECT ai.sentiment('You are such a kind person.');
sentiment
-----------
positive
(1 row)
```

ai.textfilter

描述：根据给定条件对输入文本进行过滤。

参数说明：

- input_text：text，必填，需过滤的文本内容。
- filter_condition：text，必填，过滤条件。
- model：text，选填，使用的模型名称，默认为ai.set_func_model设置的模型名称。

返回值说明：返回boolean类型的数据，给出文本是否符合过滤条件的判断。

使用示例：

```
SELECT ai.textfilter('The food at this restaurant is delicious!', 'Is this a negative review?');
textfilter
-----------
f
(1 row)
```

RAG 使用案例

RAG使用案例场景：根据用户对披萨的评价生成一份有关披萨的用户评价报告。

前提条件：需要pgvector插件支持。

1. 创建示例表并插入数据。

```
CREATE TABLE public.pizza_reviews  (
id bigserial NOT NULL,
product text NOT NULL,
customer_message text NULL,
text_length INTEGER,
CONSTRAINT pizza_reviews_pkey PRIMARY KEY (id)
);

INSERT INTO public.pizza_reviews values
(1, 'pizza','The best pizza I''ve ever eaten. The sauce was so tangy!'),
(2, 'pizza','The pizza was disgusting. I think the pepperoni was made from rats.'),
(3, 'pizza','I ordered a hot-dog and was given a pizza, but I ate it anyway.'),
(4, 'pizza','I hate pineapple on pizza. It is a disgrace. Somehow, it worked well on this izza though.'),
(5, 'pizza','I ate 11 slices and threw up. The pizza was tasty in both directions.');
```

2. 创建存储中间结果（向量）的表pizza_reviews_embeddings和记录最终输出结果的表ai_report。

```
CREATE TABLE public.pizza_reviews_embeddings (
id bigserial NOT NULL,
text_id text NOT NULL,
text_content text NOT NULL, -- it is same as pizza_reviews
```

 版权所有 © 华为云计算技术有限公司

```
model_name text NOT NULL,
ntoken int4 NULL,
nlength int4 NULL,
embedding vector NOT NULL,
CONSTRAINT pizza_reviews_embeddings_pkey PRIMARY KEY (id)
);

CREATE TABLE public.ai_report (
send_message text NULL,
chat_completion jsonb NULL,
final_report text NULL,
create_time timestamptz NULL
);
```

3. 向量化数据并插入表pizza_reviews_embeddings中。

```
WITH tmp AS (
SELECT
tt.id, tt.customer_message,
'text-embedding-3-small'::text as model_name,
ai.openai_embed(customer_message)::vector as embedding
FROM
pizza_reviews  as tt
)
INSERT INTO pizza_reviews_embeddings
(text_id, text_content, model_name, embedding )
SELECT
id, customer_message, model_name, embedding
FROM
tmp;
```

4. 向量化问题并查找相似度最高的3条数据。

```
WITH
business_question AS (
SELECT question
FROM
(values
('why customer do not like our pizza?')
)as t(question)
)
, embedding_question AS (
SELECT
question, ai.openai_embed(question)::vector as embedding
FROM
business_question
)
SELECT
eqt.question,
emt.text_content ,
emt.embedding <-> eqt.embedding as similarity
FROM pizza_reviews_embeddings emt  CROSS JOIN embedding_question eqt
ORDER BY emt.embedding <-> eqt.embedding
LIMIT 3;
```

5. 生成调研报告。

```
WITH embedding_question AS (
SELECT
'why customer dont like our pizza'::text as question, ai.openai_embed('why customer dont like our
pizza')::vector AS embedding
),
reasons AS (
SELECT
eqt.question,
emt.text_content ,
emt.embedding <-> eqt.embedding as similarity
FROM pizza_reviews_embeddings emt  CROSS JOIN embedding_question eqt
ORDER BY
emt.embedding <-> eqt.embedding
LIMIT 5
)
,agg_resons AS (
SELECT
```

 版权所有 © 华为云计算技术有限公司

```
question, jsonb_pretty(jsonb_agg(text_content)) AS reasons
FROM reasons
GROUP BY question
)
,report_needs AS (
SELECT
chr(10)||'// 1. requirements:
// 1.1 generate a business report to answer user question with provided data.
// 1.2 The report should be markdown format and less than 300 words' || chr(10) AS report_needs,
chr(10)||'// 2. data' || chr(10) AS  data_needs,
chr(10)||'// 3. user question'|| chr(10) AS user_question
)
,report AS (
SELECT
report_needs || data_needs ||  reasons  ||user_question || question AS send_message,
ai.openai_chat_complete(
jsonb_build_array(
jsonb_build_object(
'role', 'user', 'content',
report_needs || data_needs ||  reasons  ||user_question || question)
)) AS chat_completion
FROM
agg_resons CROSS JOIN report_needs
)
INSERT INTO ai_report
(send_message, chat_completion, final_report, create_time)
SELECT
send_message,
chat_completion,
now() AS create_time
FROM report;
```

15.4 MCP Server

MCP（Model Context Protocol）是由Anthropic于2024年11月提出的开放协议标准，旨在解决大型语言模型（LLM）与外部系统（如数据库、API）交互的碎片化问题。传统AI集成需为每个工具编写定制化插件，开发效率低且兼容性差。MCP通过标准化接口（类似“AI世界的USB协议”），让LLM动态理解工具功能并执行操作，大幅降低集成成本。

DWS MCP Server通过一系列基于MCP协议实现的工具和资源，让AI模型和agent与DWS集群无缝连接，提供语句执行、元数据查询等功能可以让用户可借助大模型能力，通过自然语言直接操作数据库，达到自然语言向SQL的内置转换，实现一键式后端SQL查询，用户端能够直接接收查询结果。

注意事项

该功能仅9.1.1.200及以上集群版本支持。

DWS MCP 提供的能力

- 元数据查询：通过list_schemas、list_databases、list_tables等工具列出所有模式、数据库或表；get_table_info、get_comment获取表/视图定义或注释。
- 数据库活动监控：通过get_activity工具查询pgxc_stat_activity视图获取近期数据库作业的详细信息。
- 语句执行：通用execute_query工具能够执行指定的语句查询。
- 系统信息：通过system:///{system_path}路径获取例如版本信息（system:///version）等系统信息。

版权所有 © 华为云计算技术有限公司

DWS MCP Server 使用

DWS MCP Server使用的详细方法请参考**mcp-server**。

15.5 大模型特征算子

大模型特征算子以extension形式封装为文件置于DWS系统，通过CREATE EXTENSION命令创建，大模型直接调用封装好的Python UDF函数。**该功能仅9.1.1.200及以上集群版本支持**。

在存算分离架构下，大模型特征算子功能（CREATE EXTENSION语句）仅支持在主VW上执行。

使用示例：建立一个简单的故障预测模型，使用均方根值作为特征值。

步骤1 用户创建EXTENSION，加载大模型特征算子bq_ops。

```
CREATE EXTENSION bq_ops;
```

创建完成后，Python UDF函数自动完成加载，大模型特征算子函数及功能如下表。函数类型均为pythonUDF，返回值类型均为double，反映当前信号的某些特征值。

表 15-6 函数列表

信号特征类型	函数名	算子功能
基础时域特征指标	get_sharpness(signal double precision[], fs int)	尖锐度，例如声音信号反映了中高频成分的多少及其对人耳的刺激程度，数值越大声音听起来越尖锐。
	get_roughness(signal double precision[], fs int)	粗糙度，例如声音信号感知粗糙程度的指标，主要用于描述声音在时间上的快速波动对人耳的影响。
	get_mean_square(signal double precision[])	均方值，反映了振动信号在时间域内的平均能量水平。
	get_rms(signal double precision[])	均方根值(有效值)，一般表征振动信号能量或强度。
	get_var(signal double precision[])	方差值，反映了信号幅值相对于其平均值的离散程度。
	get_pk_pk(signal double precision[])	峰峰值，表征振动信号幅值波动范围的一个重要指标。
	get_shape_factor(signal double precision[])	波形指标，衡量振动信号波形平坦程度的无量纲指标，计算为均方根值与绝对平均值的比值。它表征的是信号幅值分布的集中特性，反映波形与理想正弦波的偏离程度，对信号中的冲击成分和非高斯特性敏感。
	get_crest(signal double precision[])	峰值因数，反映了信号峰值与有效值（RMS值）之间的比例关系，用于描述信号的冲击特性或波形尖锐程度。

版权所有 © 华为云计算技术有限公司

信号特征类型	函数名	算子功能
	get_impulse(signal double precision[])	脉冲因数，反映了信号峰值与平均值之间的比例关系，主要用于评估信号中的冲击特性或瞬时能量集中程度。
	get_clearance(signal double precision[])	裕度，衡量振动信号中极端冲击强度的无量纲指标，计算为峰值与方根幅值的比值。它表征的是信号中最大峰值相对于典型幅值水平的突出程度，对严重局部故障（如严重剥落、断裂）产生的剧烈冲击异常敏感。
	get_skewness(signal double precision[])	偏斜度，反映了信号幅值分布相对于其平均值的偏斜方向和程度。
	get_kurt(signal double precision[])	峭度，通过振动信号幅值分布尖锐程度或尾部特性，反映信号幅值分布相对于正态分布的尖锐程度和异常值（如冲击或瞬态事件）的存在。
	get_gini(signal double precision[])	基尼指数，表征信号能量分布均匀性或集中程度的指标。它最初来源于经济学中的收入不平等度量，在信号处理领域被广泛应用于评估信号的能量分布特性。
频域及声学特征指标	get_spec_ctrd(signal double precision[], fs int)	重心频率，信号功率谱的质量中心，反映了信号中能量的集中位置。
	get_spec_ms(signal double precision[], fs int)	均方频率，信号频谱各频率分量的平方的加权平均，反映信号能量的频率分布特性。
	get_spec_rms(signal double precision[], fs int)	均方根频率，信号频谱的平方和的平方根，用于描述信号频谱的总体分布情况。
	get_spec_var_ctrd(signal double precision[], fs int)	频谱方差，信号频谱分布的方差，反映了信号频率成分的分散程度。
	get_spec_std_ctrd(signal double precision[], fs int)	频谱标准差，频率方差的平方根，用于描述信号频谱分布的离散程度。
	get_pse(signal double precision[], fs int)	谱熵，表征信号频域能量分布复杂度或无序程度的一个重要指标。它基于信息熵的概念，用于描述信号在频域中的能量分布特性。
	get_env_rms(signal double precision[])	包络谱均方根，表征信号包络谱能量水平，反映了信号中调制成分（如故障特征频率）的整体能量强度。

 版权所有 © 华为云计算技术有限公司

信号特征类型	函数名	算子功能
	get_ehr(signal double precision[])	谐波率，用于评估信号中谐波成分的强度，反映信号的周期性特征或非线性特性。
	get_kurt_aver(signal double precision[])	平均峭度，对一段信号或多组信号的峭度值进行平均，用于评估长期内信号的冲击特性变化。
	get_mpf(signal double precision[], fs int)	转速，指在单位时间内的旋转次数，通常以每分钟转数 (RPM) 表示。
带通滤波分段指标	get_bpf_0_500_rms(signal double precision[], fs int)	低频有效值(均方根)，对信号做0-500Hz低通滤波后，计算有效值，描述低频信号的能量或强度。
	get_bpf_500_2000_rms(signal double precision[], fs int)	中频有效值(均方根)，对信号做500-2000Hz带通滤波后，计算有效值，描述中频信号的能量或强度。
	get_bpf_2000_inf_rms(signal double precision[], fs int)	高频有效值(均方根)，对信号做2000-fs/2Hz高通滤波后，计算有效值，描述高频信号的能量或强度。
	get_bpf_0_500_kurt(signal double precision[], fs int)	低频峭度，对信号做0-500Hz低通滤波后，计算峭度指标，描述低频信号中冲击性的强弱。
	get_bpf_500_2000_kurt(signal double precision[], fs int)	中频峭度，对信号做500-2000Hz带通频滤波后，计算峭度指标，描述中频信号中冲击性的强弱。
	get_bpf_2000_inf_kurt(signal double precision[], fs int)	高频峭度，对信号做2000-fs/2Hz高通滤波后，计算峭度指标，描述高频信号中冲击性的强弱。
	get_bpf_0_500_ehr(signal double precision[], fs int)	低频谐波率，对信号做0-500Hz低通滤波后，计算谐噪比，描述低频信号中谐波成分与噪声成分强度比。
	get_bpf_500_2000_ehr(signal double precision[], fs int)	中频谐波率，对信号做500-2000Hz带通滤波后，计算谐噪比，描述中频信号中谐波成分与噪声成分强度比。
	get_bpf_2000_inf_ehr(signal double precision[], fs int)	高频谐波率，对信号做2000-fs/2Hz高通滤波后，计算谐噪比，描述高频信号中谐波成分与噪声成分强度比。

步骤2 创建表bq_col_table，device_code表示当前设备号，measuring_point_code表示测量点编码，date_time表示信号采集日期，high_array列表示当前接收的信号

```
CREATE TABLE bq_col_table(device_code varchar, measuring_point_code text,date_time timestamp with time zone, high_array double precision[]) with (orientation=column, enable_hstore_opt=true);
```

每隔1小时采集10秒钟信号，假设当前入库时数据如下：

 版权所有 © 华为云计算技术有限公司

```
INSERT INTO bq_col_table VALUES
('10098819','3a138131-344a-af96-9e9d-da049656d905','2024-07-13
17:59:59+08:00','{0.527995824813842,-0.62188184261322,-0.332374721765518,-0.139671847224235,-0.308
928370475769,-0.165734529495239,0.137558653950691,-0.923967480659484,-0.398990541696548,0.62027
1801948547,0.366085141897201,-0.873452186584472,-1.00577819347381,-0.581831872463226,0.06752147
52554893,0.789226412773132,-0.643114387989044,-0.779465496540069,0.913703441619873,1.3337252140
0451,-0.0830182060599327,0.621579945087432,1.48476803302764}');
```

步骤3 调用算子get_rms()计算信号均方根值，当前样本信号的振动信号能量强度。上述样例数据结果为：

```
SELECT device_code, measuring_point_code, date_time, get_rms(high_array) FROM bq_col_table;
 device_code |         measuring_point_code         |       date_time        |     get_rms
-------------+--------------------------------------+------------------------+------------------
 10098819    | 3a138131-344a-af96-9e9d-da049656d905 | 2024-07-13 17:59:59+08 | .705324261533061
(1 row)
```

步骤4 重复上述步骤，采集10天信号数据为样本，设定rms的正常取值区间，假设为[0.5, 0.8]。

步骤5 如果有异常信号入库。

```
INSERT INTO bq_col_table VALUES
('10098819','3a138131-344a-af96-9e9d-da049656d905','2024-07-13
07:59:59+08:00','{0.544054210186004,-0.769003570079803,-1.79972970485687,0.659896433353424,1.65061
652660369,-0.221043065190315,-1.83933162689208,-2.58152985572814,-0.627029538154602,2.153721809
3872,2.14685225486755,-0.0429693721234798,-1.21243667602539,-1.02749335765838,-0.52654320001602
1,-0.0408141687512397,1.96406400203704,2.1080584526062,0.257277429103851,-1.36532151699066,-2.31
293749809265,-0.803890943527221,1.13646578788757}');
```

此时库中将存在异常的rms值1.43，据此可以分析该异常信号。

```
SELECT device_code, measuring_point_code, date_time, get_rms(high_array) FROM bq_col_table;
 device_code |         measuring_point_code         |       date_time        |     get_rms
-------------+--------------------------------------+------------------------+------------------
 10098819    | 3a138131-344a-af96-9e9d-da049656d905 | 2024-07-13 17:59:59+08 | .705324261533061
 ......
 10098819    | 3a138131-344a-af96-9e9d-da049656d905 | 2024-07-13 07:59:59+08 | 1.43480152874657
```

----结束

说明

使用上述算子时，会引用第三方库numpy、scipy等，将会导致udf进程的内存占用增大。典型的算子如：get_skewness、get_kurt、get_env_rms、get_ehr、get_kurt_aver、get_bpf_0_500_rms、get_bpf_500_2000_rms、get_bpf_2000_inf_rms、get_bpf_0_500_kurt、get_bpf_500_2000_kurt、get_bpf_2000_inf_kurt、get_bpf_0_500_ehr、get_bpf_500_2000_ehr、get_bpf_2000_inf_ehr。使用时请保证参数**udf_memory_limit**不少于2GB。

 版权所有 © 华为云计算技术有限公司

16 实时数仓

16.1 实时数仓简介

实时数仓在大规模数据查询和分析能力基础上，提供高并发、高性能、低时延、低成本的事务处理能力。适用于**HTAP混合负载场景**和**物联网IoT等实时分析场景**，“一库两用，生产即分析”。

说明

- 如需使用实时数仓能力，请在DWS控制台创建集群时，选择**存算一体1:4云盘规格**，了解更多参见**存算一体规格**。
- 创建DWS集群选择**1:8云盘规格**为标准数仓，**1:4云盘规格**为实时数仓，以下内容描述的“实时数仓”、“标准数仓”即通过以上规格区分。

实时数仓需要支持将insert+upsert+update等操作实时快速入库，数据来源于上游的其他数据库或者应用，同时要求入库后的数据要能及时查询，对于查询的效率要求很高。

目前DWS传统数仓已有的行存表或者列存表都无法同时满足实时入库和实时查询两个诉求。其中行存表实时入库能力强，支持高并发更新，但是磁盘占用高，查询效率低；列存表数据压缩率高，AP查询性能好，但是不能很好的支持并发更新，并发入库存在严重的锁冲突。

为了解决上面的问题，需要在使用列存储格式尽量降低磁盘占用的同时，支持高并发的更新操作入库以及高性能的查询效率。DWS的实时数仓中的HStore表就是针对这种情况设计和实现的，面向对于实时入库和实时查询有较强诉求的场景，同时拥有处理传统TP场景的事务能力。

DWS提供的实时数仓中实现了一种全新的HStore表，可以做到单条或者小批量IUD操作的高并发实时入库，也可以支持大批量的定期入库。数据入库提交后即可查询，无任何时延。支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询，也可以在TPCC这种强事务压力场景下保证数据强一致性。

版权所有 © 华为云计算技术有限公司

说明

- 实时数仓的HStore表仅8.2.0.100及以上集群版本支持。
- 实时数仓为一库两用，生产即分析；适用于交易、分析混合型业务场景。关于如何创建实时数仓请参见**创建DWS存算一体集群**。
- HStore表支持冷热数据管理，具体可参考**冷热数据管理**，该功能仅8.2.0.101及以上集群版本支持。
- HStore表是实时数仓中设计的一种表类型，与SQL参数hstore没有任何关系。

与标准数仓的区别

实时数仓与标准数仓是DWS的两种规格不一样的数仓，在使用上也存在一定差异，具体可参考**表16-1**进行对比分析。

表 16-1 实时数仓与标准数仓的差异

数仓类型	标准数仓（存算一体1:8云盘规格）	实时数仓（存算一体1:4云盘规格）
适用场景	融合分析业务，一体化OLAP分析场景。主要应用于金融、政企、电商、能源等领域。	实时入库+分析混合业务，上游数据实时入库+数据入库后实时高效查询场景。主要用于电商、金融等实时入库要求高的场景。
产品优势	性价比高，使用场景广泛。 支持冷热数据分析，存储、计算弹性伸缩，无限算力、无限容量等。	混合负载，入库性能强。 提供与列存相当的高性能查询效率与高压缩率的数据压缩能力。同时拥有处理传统TP场景的事务能力。
功能特点	支持海量数据离线处理和交互查询，数据规模大、复杂数据挖掘具有很好的性能优势。	支持海量数据高并发的更新操作入库以及高性能的查询效率。在数据规模大、入库并发高、查询要求高的场景下具有很好的性能优势。
SQL语法	SQL语法兼容性高，语法通用，易于使用。	兼容列存语法。
GUC参数	丰富的GUC参数，根据客户业务场景适配最适合客户的数仓环境。	兼容标准数仓GUC参数，同时支持实时数仓调优参数。

技术特点

- 完整的事务一致性

版权所有 © 华为云计算技术有限公司

体现在数据插入或者更新后提交即可见，不存在时延；并发更新后数据保证强一致，不会出现乱序导致的结果预期不一致。

- 查询性能好

  多表关联等复杂AP查询场景下，更完善的分布式查询计划与分布式执行器带来的性能优势，支持复杂的子查询和存储过程。

- 入库快

  彻底解决列存CU锁冲突问题，支持高并发的更新入库操作，典型场景下，并发更新性能是之前的百倍以上。

- 高压缩

  数据在MERGE进入列存主表后，按列存储具有天然的压缩优势，能极大地节省磁盘空间与IO资源。

- 查询加速

  支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询。

行存、列存、HStore 表对比

表 16-2 行存、列存、HStore 表对比

表类型	行存表	列存表	HStore表
数据存储方式	以元组为单位，将每一条数据的所有属性值存储到临近的空间里。	以CU（Compress Unit）为单位，将单个属性的所有值存储到临近的空间里。	数据主要以CU形式存储在列存主表上，对于被更新的列、小批量插入的数据将被序列化后存储到新设计的Delta表上。
数据写入	行存压缩暂未商用，数据按原始状态存储，磁盘空间占用较大。	按列存储时，由于属性值类型相同具有天然的压缩优势。数据写入时能极大节省IO资源与磁盘空间占用。	批量插入的数据直接写入CU，具有与列存一致的压缩优势。 被更新的列、小批量插入的数据会序列化后压缩。同时定期merge到主表CU。
数据更新	数据按行更新，没有CU锁问题，并发更新（update/upsert/delete等）性能好。	即使更新单条数据，也要获取整个CU的锁，基本无法支持并发更新（update/upsert/delete等）。	彻底解决列存更新的CU锁问题，并发更新（update/upsert/delete等）的性能达到行存的60%以上。
数据读取	按行读取，即使只需访问某一列的数据，也需要将一整行的数据取出。查询性能较差。	按列读取时只需访问该列的CU，再加上CU的压缩优势导致需要占用的IO资源更少，读取性能很好。	对于列存主表的数据按列读取，对于被更新的列、小批量插入的数据会反序列化后取出，数据merge到主表后具有与列存一致的数据读取优势。

版权所有 © 华为云计算技术有限公司

表类型	行存表	列存表	HStore表
优点	并发更新性能好。	查询性能好，磁盘占用空间少。	并发更新性能好，数据MERGE后具有与列存一致的查询性能优势与压缩优势。
缺点	占用磁盘空间多，查询性能差。	基本无法支持并发更新。	需要后台常驻线程对HStore表进行merge清理操作。先merge到CU主表再进行清理，与SQL语法中的Merge无关。
适用场景	1. 更新删除操作频繁的TP事务场景。 2. 点查询（基于索引的、返回数据量小的简单查询）。	1. 查询分析为主的AP场景。 2. 数据量大，存入后的更新删除操作少。	1. 实时并发入库场景。 2. 需要支持高并发的更新入库操作以及高性能的查询效率。

16.2 支持与限制

实时数仓兼容所有列存语法，具体情况如下：

表 16-3 支持的语法

语法	是否支持
CREATE TABLE	支持
CREATE TABLE LIKE	支持
DROP TABLE	支持
INSERT	支持
COPY	支持
SELECT	支持
TRUNCATE	支持
EXPLAIN	支持
ANALYZE	支持
VACUUM	支持
ALTER TABLE DROP PARTITION	支持
ALTER TABLE ADD PARTITION	支持

版权所有 © 华为云计算技术有限公司

语法	是否支持
ALTER TABLE SET WITH OPTION	支持
ALTER TABLE DROP COLUMN	支持
ALTER TABLE ADD COLUMN	支持
ALTER TABLE ADD NODELIST	支持
ALTER TABLE CHANGE OWNER	支持
ALTER TABLE RENAME COLUMN	支持
ALTER TABLE TRUNCATE PARTITION	支持
CREATE INDEX	支持
DROP INDEX	支持
DELETE	支持
ALTER TABLE 其他	支持
ALTER INDEX	支持
MERGE	支持
SELECT INTO	支持
UPDATE	支持
CREATE TABLE AS	支持

约束限制

- 当需要使用HStore表时，需要同步修改以下几个参数的默认值，否则会导致HStore表性能严重劣化。

  9.1.1.100集群版本开始推荐的参数修改配置是：autovacuum_max_workers_hstore=3，autovacuum_max_workers=2，autovacuum_max_workers_col=2，autovacuum=true。

- 8.2.1及以上版本支持列存btree索引脏数据清理，频繁更新入库的场景下能有效控制索引空间，提高入库与查询性能。实时场景不应该对列存使用psort索引原因如下：

  a. psort没有索引清理，索引空间会持续膨胀。

  b. psort只对攒批入库的数据进行排序，实时场景性能提升有限。

- 9.1.1.100集群版本hstore表脏页清理依赖列存vacuum和异步排序两个机制协同工作，默认小于1000行的CU被定性为小CU。脏页中小CU会通过异步排序的机制完成合并，非小CU通过列存vacuum重写到新的文件。列存vacuum受阈值参数colvacuum_threshold_scale_factor（该参数需联系技术支持修改）控制。
- 当使用HStore异步排序功能时，异步排序期间可能会阻塞部分数据的DML操作，最大的阻塞粒度为异步排序的行数阈值，因此在反复DML的场景下不建议使用此功能。

 版权所有 © 华为云计算技术有限公司

列存表的 Delta 表差异

表 16-4 HStore 表与列存表的辅助 Delta 表差异

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
表结构	与列存主表的表定义一致。	与主表定义不一致。	不同于主表，与HStore表相同。
功能	用于暂存小批量insert的数据，满阈值后再merge到主表，避免直接insert到主表产生大量小CU。	用于持久化存储update/delete/insert信息。在产生故障后用于恢复内存更新链等管理并发更新的内存结构。	用于持久化存储update/delete/insert信息。在产生故障后用于恢复内存更新链等管理并发更新的内存结构。相比HStore进一步优化。
缺陷	来不及merge导致delta表膨胀，影响查询性能，同时无法解决并发update的锁冲突问题	依赖后台常驻autovacuum来做merge操作。	依赖后台常驻autovacuum来做merge操作。

版权所有 © 华为云计算技术有限公司

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
规格差异	不支持同一CU的并发，适合并发更新较少的情况。	1. 插入和更新限制 ● MERGE INTO操作不支持并发更新同一行或对同一键进行重复更新。 ● 不支持并发update或delete同一行，会触发报错。 2. 索引和查询限制 ● 索引不支持数组条件过滤、IN 表达式过滤、部分索引或表达式索引。 ● 索引不支持失效处理。 3. 表结构和操作限制 ● 在进行分区交换或 relfilenode 操作时，确保交换的表都是 HStore 表。 ● 不支持update修改分布列，不建议update修改分区列（不报错，但性能非常差）。	1. 插入和更新限制 ● MERGE INTO操作不支持并发更新同一行或对同一键进行重复更新。 ● 不支持并发update或delete同一行，会触发报错。 ● hstore_opt不支持跨分区upsert更新。 2. 索引和查询限制 ● 支持bitmap索引。 ● 支持全局字典。 ● bitmap_columns必须在建表时确定，一旦设置后不允许修改。 ● opt 版本不支持 SMP streaming 透传参数。在需要分区剪枝的多表关联查询中，避免使用复制表或设置query_dop。 3. 表结构和操作限制 ● 不支持update修改分布列，不支持update修改分区列。 ● enable_hstore_opt 属性需在创建表时设定，且不可更改。

版权所有 © 华为云计算技术有限公司

数仓类型	列存的delta表	HStore的delta表	HStore Opt的delta表
入库建议	1. 综合入库、查询和空间情况，建议选择hstore_opt表，微批copy无更新入库场景性能要求高的情况下选择hstore表。 2. HStore/HStore Opt共同点： ● update入库性能差，建议修改为upsert。 ● delete入库，确定计划走索引扫描即可，用JDBC batch方式入库最佳。 ● merge into入库建议在单次入库数据量超过100W/dn，且无并发数据保证无重复的情况下使用。 ● 尽量避免对冷分区的数据进行修改和新增。		
点查建议	1. 综合点查场景，建议使用HStore Opt表。 2. HStore/HStore Opt共同点： 在等值过滤条件使用最多且distinct值分布相对均匀的一个列上创建二级分区。 3. HStore的点查建议： ● 使用主键之外的索引加速不能得到预期效果，不建议开启。 ● 如果数据类型多为numeric或者16字节以内的字符串，建议使用turbo加速。 4. HStore Opt的点查建议： ● 除了二级分区之外的等值过滤列，如果过滤条件涉及的列在查询中基本固定，使用cbtree索引，如果不断变化建议使用GIN索引，注意创建索引的列数不要超过5列。 ● 所有涉及等值过滤的字符串列，都可以建表时指定bitmap索引，不限列数，但后续不可修改。 ● 时间范围过滤的列，指定为分区列。 ● 点查返回数据量超过10W/dn的场景，索引扫描的性能提升有限，建议针对这种情况使用guc参数enable_seqscan对比测试，灵活选择。		

16.3 实时数仓 Binlog

16.3.1 订阅实时数仓 Binlog

Binlog 使用介绍

当用户需要捕获数据库事件用于数据增量导出Flink等第三方组件，并协同完成数据加工等任务时，DWS实时数仓中的HStore表提供了Binlog功能，通过消费Binlog数据来实现上下游的数据同步，提高数据加工的效率。

传统的数据比如MySQL数据库等，支持通过Binlog来记录数据库中所有数据的变化，但相比于MySQL的Binlog主要用于数据恢复与主从复制，DWS实时数仓Binlog一般只用于实时场景下的数据同步。同时DWS实时数仓Binlog并不会记录DDL操作，只记录INSERT/DELETE/UPDATE（UPSERT）等DML操作。

版权所有 © 华为云计算技术有限公司

DWS的Binlog的优点如下：

- 表级按需开关：按需给指定表打开或关闭binlog功能，更为灵活。
- 全增量一体消费：支持Flink任务启动后，先全量同步source，再实时消费source端增量。
- 支持消费即清理：对于空间敏感且只关注实时同步与加工的客户，支持消费后即开始异步清理增量，有效减少空间使用。

利用Flink强大的实时处理能力和DWS的Binlog能力，可以快速构建实时数仓，且无需维护其他组件（如kafka），整体架构分层清晰，数据可以高效流动，并且整体任务链路都可以通过Flink SQL来驱动，便于业务人员理解和使用。

约束与限制

1. 当前仅9.1.0.200及以上版本的HStore表支持记录Binlog功能。
2. 使用Binlog的前置条件是必须存在主键约束，并且为HStore表，分布方式只能是Hash分布。
3. Binlog表仅对INSERT/DELETE/UPDATE（UPSERT）等DML操作进行记录，不会记录DDL。
4. 当前Binlog表不支持的操作：INSERT OVERWRITE、修改分布列、给临时表开启Binlog、EXCHANGE/MERGE/SPLIT PARTITION。
5. 当前Binlog表并不限制用户进行以下DDL操作，但进行操作后会导致增量数据与同步点位信息会被清空，需要评估后再执行：

   ADD COLUMN 增加列、DROP COLUMN 删除列、SET TYPE 修改列、TRUNCATE 清空表数据。
6. Binlog表在线或者离线扩容期间会等待Binlog记录的消费，只有Binlog记录消费完毕才可以继续进行接下来的扩缩容步骤，默认等待时间为1小时，可通过GUC参数**binlog_consume_timeout**来设置，如果等待超时或者等待出错都会退出扩缩容过程，认为该表扩缩容失败。
7. VACUUM FULL Binlog表时，会等待Binlog记录的消费，只有Binlog记录消费完毕才可以进行接下来的VACUUM FULL操作，默认等待时间为1小时，可通过GUC参数**binlog_consume_timeout**来设置，如果等待超时或者等待出错都会退出VACUUM FULL过程，认为该表VACUUM FULL失败。且由于需要等待Binlog记录消费完毕，所以即使VACUUM FULL一个分区表，也会对分区的主表上7级锁，阻塞整个表的插入更新或者删除。
8. Binlog表在备份恢复期间，仅会被当做普通HStore表进行备份，恢复后辅助表的增量数据与同步点信息会清空，需要重新开始同步。
9. 支持Binlog时间戳功能，通过设置enable_binlog_timestamp打开，同样只有hstore和hstore_opt两种表支持打开。该约束仅9.1.0.200及以上版本支持。
10. 对于非实时入库场景，不建议开启Binlog功能：大批量入库、定时跑批任务等，会在短时间内产生大量Binlog增量，导致Flink任务内存、计算资源压力过大甚至oom，同时可能导致读Binlog增量超时或者内存报错，影响业务连续性。

 版权所有 © 华为云计算技术有限公司

Binlog 格式与原理

表 16-5 binlog 字段格式

字段名称	字段类型	含义
gs_binlog_sync_point	BIGINT	Binlog系统字段，表示该记录的同步点值，普通GTM模式下，该值唯一且有序。
gs_binlog_event_sequence	BIGINT	Binlog的系统字段，用于表示同一事务内操作的先后顺序。
gs_binlog_event_type	CHAR	Binlog的系统字段，表示当前记录的操作类型。 type可能有以下几种取值： ● 'I'，为INSERT，表示当前Binlog是插入一条新记录。 ● 'd'，为DELETE，表示当前Binlog是删除一条记录。 ● 'B'，为BEFORE_UPDATE，表示当前Binlog是更新前的记录。 ● 'U'，为AFTER_UPDATE，表示当前Binlog是更新后的记录。
gs_binlog_timestamp_us	BIGINT	Binlog的系统字段，表示当前记录入库时的时间戳。 只有开启binlog时间戳功能时会有，没开启binlog时间戳时为空。仅9.1.0.200及以上版本支持。
user_column_1	用户列	用户的自定义数据列。
...	...	...
user_column_n	用户列	用户的自定义数据列。

版权所有 © 华为云计算技术有限公司

说明

- UPDATE（或者UPSERT触发的更新操作）会产生两条Binlog，分别是BEFORE_UPDATE类型与AFTER_UPDATE类型，其中BEFORE_UPDATE主要用于保证Flink等第三方组件做数据加工后加工结果的正确性。
- 对于用户UPDATE/DELETE操作产生的BEFORE_UPDATE以及DELETE类型的Binlog，DWS实时数仓中这两类Binlog类型并不会在操作执行时就反查出所有用户列并填充，保证了入库的性能。
- 当在DWS实时数仓给某个HStore表开启Binlog功能时， 本质就是给HStore表再创建一张辅助表（也通过HStore实现），该辅助表包含gs_binlog_sync_point、gs_binlog_event_event_sequence、gs_binlog_event_type三个系统列以及一个将所有用户列序列化存储到一个字段的value列。
- 当开启binlog时间戳功能时（表级参数enable_binlog_timestamp），辅助表上记录的binlog记录会严格保留至超过TTL才会清理，这会带来数倍的额外空间开销（具体倍数与TTL时间内的更新入库量有关）。当开启普通binlog时（表级参数enable_binlog），辅助表上记录的binlog，只要被下游消费就会允许异步清理，能大幅度减少空间占用。仅9.1.0.200及以上版本支持。对于表上数据变化量较大的表，更推荐开启表级参数enable_binlog，避免产生大量binlog记录后影响空间以及读增量性能。

开启 Binlog

通过在建HStore表时指定表级参数enable_binlog，开启HStore表的Binlog功能。

```
CREATE TABLE hstore_binlog_source (
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true,
    enable_binlog=on,
    binlog_ttl = 259200
);
```

说明

- 对于开启了Binlog的表，并不会立刻给入库的操作记录Binlog，需要再给同步任务注册同步点后，才会开始记录Binlog（开启Flink同步binlog任务后，会自动循环进行获取同步点、获取增量数据、注册同步点操作）。
- binlog_ttl是可选参数，当不设置时将使用默认值86400， 单位为秒，当同步任务注册的同步点超过TTL没有进行增量同步时，该同步点位将被清理。最老的同步点位之前的Binlog（即被所有任务消费了的Binlog）会被异步清理来回收空间。
- 空间开销：对于开启普通binlog的表，如果能保证增量被下游及时消费，那么空间就能被及时清理回收。

通过执行ALTER命令给已有的HStore表开启binlog功能：

```
CREATE TABLE hstore_binlog_source (
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true
);
ALTER TABLE hstore_binlog_source SET (enable_binlog=on);
```

查询 binlog

通过DWS提供的系统函数，可以直接查询目标表在指定DN上binlog信息，以及是否被下游消费完毕等信息。

 版权所有 © 华为云计算技术有限公司

```
-- 模拟Flink调用系统函数获取同步点，参数分别表示 表名、槽位名、是否checkPoint点位，目标DN(为0表示所有DN)。
select * from pg_catalog.pgxc_get_binlog_sync_point('hstore_binlog_source', 'slot1', false, 0);
select * from pg_catalog.pgxc_get_binlog_sync_point('hstore_binlog_source', 'slot1', true, 0);
-- 进行增删改产生增量binlog。
INSERT INTO hstore_binlog_source VALUES(100, 1, 1);
delete hstore_binlog_source where c1 = 100;
INSERT INTO hstore_binlog_source VALUES(200, 1, 1);
update hstore_binlog_source set c2 =2 where c1 = 200;
-- 模拟Flink调用系统函数查询指定CSN区间的Binlog，参数分别表示表名，目标DN(为0表示所有DN),起始CSN点位, 终止CSN点位。
select * from pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
```

```
gaussdb=> SELECT * FROM pgxc_get_binlog_changes('orders_source', 0, 0 , 9999999999);
 gs_binlog_sync_point | gs_binlog_event_sequence | gs_binlog_event_type | gs_binlog_timestamp_us | c1  | c2 | c3
----------------------+--------------------------+----------------------+------------------------+-----+----+----
               886158 |                        2 | I                    |                        | 100 |  1 |  1
               886166 |                        3 | d                    |                        | 100 |  1 |  1
               886190 |                        4 | I                    |                        | 200 |  1 |  1
               886197 |                        5 | B                    |                        | 200 |  1 |  1
               886197 |                        6 | U                    |                        | 200 |  2 |  1
(5 rows)
```

可以看到两次INSERT操作产生了两个gs_binlog_event_type是'I'的记录，DELETE操作产生了type是'd'的记录，UPDATE产生了一行BeforeUpdate的'B'记录以及一条AfterUpdate的'U'记录，分别表示更新前的值以及更新后的值。

通过调用系统函数pgxc_consumed_binlog_records可以查询目标表的binlog是否被所有槽位消费完毕。参数分别表示目标表名以及目标DN（为0表示所有DN）。

```
-- 模拟Flink调用系统函数注册同步点，参数分别表示表名，槽位名，注册的点位，是否属于checkPoint, 点位对应的xmin (获取同步点时会提供)。
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, false, 100);
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, true, 100);
-- 查询表上binlog是否被全部消费，返回 1 表示已经被下游槽位全部消费。
select * from pgxc_consumed_binlog_records('hstore_binlog_source',0);
```

```
postgres=# select * from pgxc_consumed_binlog_records('hstore_binlog_source',0);
 pgxc_consumed_binlog_records
------------------------------
                            1
(1 row)
```

开启 Binlog 时间戳功能

如果需要读取指定时间点之后binlog的功能，通过在建HStore表时指定表级参数enable_binlog_timestamp，开启HStore表的Binlog时间戳功能。仅9.1.0.200及以上版本支持。

```
CREATE TABLE hstore_binlog_source(
    c1  INT PRIMARY KEY,
    c2  INT,
    c3  INT
) WITH (
    ORIENTATION = COLUMN,
    enable_hstore_opt=true,
    enable_binlog_timestamp =on,
    binlog_ttl = 259200
);
```

 版权所有 © 华为云计算技术有限公司

说明

- 对于开启了Binlog时间戳的表，并不会立刻给入库的操作记录Binlog，需要再给同步任务注册同步点后，才会开始记录Binlog（开启Flink同步binlog任务后，会自动循环进行获取同步点、获取增量数据、注册同步点操作）。
- binlog_ttl是可选参数，当不设置时将使用默认值86400， 单位为秒（即默认保留一天），当Binlog记录的时间戳距离现在大于TTL时，会被异步清理。
- 空间开销：对于开启binlog时间戳的表，辅助表上记录的binlog记录会严格保留至超过TTL才会清理，这会带来数倍的额外空间开销（具体倍数与TTL时间内的更新入库量有关）。

查询开启binlog时间戳功能的表上的Binlog：

```
postgres=# select * from pgxc_get_binlog_changes('hstore_binlog_source', 0, 0 , 9999999999);
 gs_binlog_sync_point | gs_binlog_event_sequence | gs_binlog_event_type | gs_binlog_timestamp_us | c1  | c2 | c3
----------------------+--------------------------+----------------------+------------------------+-----+----+----
                10516 |                        2 | I                    |       1731570520900211 | 100 |  1 |  1
                10517 |                        3 | d                    |       1731570520904425 | 100 |  1 |  1
                10518 |                        2 | I                    |       1731570520909055 | 200 |  1 |  1
                10519 |                        3 | B                    |       1731570520914102 | 200 |  1 |  1
                10519 |                        4 | U                    |       1731570520914154 | 200 |  2 |  1
(5 rows)
```

将gs_binlog_timestamp_us从BigInt类型转成可读的时间戳：

```
select to_timestamp(1731569598408661/1000000);
```

```
postgres=#  select to_timestamp(1731570520900211/1000000);
          to_timestamp
--------------------------------
 2024-11-14 15:48:40.900211+08
(1 row)
```

获取各个DN上，目标表指定时间点后的第一条binlog信息（为空表示这个时间点后没有binlog）：

```
select * from pgxc_get_binlog_cursor_by_timestamp('hstore_binlog_source','2024-11-14 15:33:18.40866+08', 0);
```

```
postgres=# select * from pgxc_get_binlog_cursor_by_timestamp('hstore_binlog_source','2024-11-14 15:48:40.900211+08', 0);
 node_name |   node_id   | latest_sync_point | binlog_sync_point | binlog_timestamp_us | binlog_xmin
-----------+-------------+-------------------+-------------------+---------------------+-------------
 dn_2      | -1051926843 |             10532 |             10516 |    1731570520900211 |       10510
 dn_1      | -1300059100 |             10532 |             10518 |    1731570520909055 |       10510
(2 rows)
```

获取开启binlog时间戳功能的表的消费进度：

返回的字段表示最近消费的一条binlog的时间戳，binlog上的最近时间戳，最近消费的一条binlog的CSN点位， binlog上最近CSN点位，未消费的binlog记录数量。

```
-- 模拟Flink调用系统函数注册同步点，参数分别表示表名，槽位名，注册的点位，是否属于checkPoint, 点位对应的xmin (获取同步点时会提供)。
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, false, 100);
select pgxc_register_binlog_sync_point('hstore_binlog_source', 'slot1', 0, 9999999999, true, 100);
-- 查询目标表各个槽位的消费进度。
select * from pgxc_get_binlog_consume_progress('hstore_binlog_source', 0);
```

```
postgres=# select * from pgxc_get_binlog_consume_progress('hstore_binlog_source', 0);
 node_name |   node_id   | slot_name | checkpoint |  latest_consumed_timestamp |     latest_timestamp       | latest_consumed_csn | latest_csn | unconsumed_binlog_count
-----------+-------------+-----------+------------+----------------------------+----------------------------+---------------------+------------+-------------------------
 dn_1      | -1300059100 | slot1     | f          | 2024-11-14 15:48:40+08     | 2024-11-14 15:48:40+08     |               10519 |      10519 |                       0
 dn_1      | -1300059100 | slot1     | t          | 2024-11-14 15:48:40+08     | 2024-11-14 15:48:40+08     |               10519 |      10519 |                       0
 dn_2      | -1051926843 | slot1     | f          | 2024-11-14 15:48:40+08     | 2024-11-14 15:48:40+08     |               10517 |      10517 |                       0
 dn_2      | -1051926843 | slot1     | t          | 2024-11-14 15:48:40+08     | 2024-11-14 15:48:40+08     |               10517 |      10517 |                       0
(4 rows)
```

控制 DML 不产生 Binlog

通过设置会话级参数**enable_generate_binlog**为off，可以控制当前会话的DML，在给开启binlog的表入库时，不产生binlog记录。

版权所有 © 华为云计算技术有限公司

16.3.2 Flink 实时消费 Binlog

注意事项

- 当前仅9.1.0.200及以上的版本HStore表支持记录Binlog功能。
- 目前DWS只有Hstore表支持Binlog功能，表需要包含主键且设置enable_binlog=on或者enable_binlog_timestamp=on。
- 消费的Binlog表名不要带有特殊字符，例如：点（.）、引号（"" 或 ''）等。
- 如果多个任务消费同一张表的Binlog数据，需要保证每个任务的binlogSlotName唯一。
- 为了达到最高的消费速度，建议将任务的并发度和DWS集群DN数设置一致。
- 在使用Flink进行全量同步时， 如果期间出现Flink异常导致全量失败，建议更换binlogSlotName重新进行全量同步，否则会有数据一致性风险。
- 在使用Flink进行增量同步时，如果Flink异常重启，会从上次checkpoint点位开始继续抽取增量，为了避免增量丢失，checkpoint点位的增量会被重复消费。如果要避免Flink异常重启时checkpoint点位重复消费，请开启Flink的有状态重启功能。
- 使用**dws-connector-flink**的Sink能力来写入Binlog数据时，需注意：
  - 如果需要保证DN内的数据写入顺序，则需要设置connectionSize为1。
  - 如果源端有更新主键操作或者需要flink进行聚合计算，将ignoreUpdateBefore设置为false，否则不建议将ignoreUpdateBefore设置为false（默认true）。

Flink 实时消费 Binlog

使用DWS Connector来实时消费Binlog，具体请参见**DWS-Connector**。

如果已使用其他同步工具将全量数据同步到了目标端，后续只想进行增量同步。则可以调用以下系统函数来更新同步点。

```
SELECT * FROM pg_catalog.pgxc_register_full_sync_point('table_name', 'slot_name');
```

源表 DDL

Source端会根据操作类型自动为每行数据设置准确的Flink RowKind类型（INSERT、DELETE、UPDATE_BEFORE、UPDATE_AFTER），这样就能镜像同步表的数据，类似MySQL和Postgres的CDC功能。

```
CREATE TABLE test_binlog_source (
  a int,
  b int,
  c int,
  primary key(a) NOT ENFORCED
) with (
  'connector' = 'dws',
  'url' = 'jdbc:gaussdb://ip:port/gaussdb',
  'binlog' = 'true',
  'tableName' = 'test_binlog_source',
  'binlogSlotName' = 'slot',
  'username'='xxx',
  'password'='xxx')
```


版权所有 © 华为云计算技术有限公司

Binlog 相关参数说明

下表仅涉及消费Binlog时的参数。

表 16-6 消费 Binlog 时的参数

参数	说明	数据类型	默认值
binlog	是否读取Binlog信息	Boolean	false
binlogSlotName	槽位信息，可以理解一个标识。由于可能存在多个Flink任务同时消费同一张表的Binlog信息，所以该场景需要保证每个任务的binlogSlotName不同。	String	Flink映射表的表名
binlogBatchReadSize	批量读取binlog的数据行数	Integer	5000
fullSyncBinlogBatchReadSize	全量读取binlog的数据行数	Integer	50000
binlogReadTimeout	增量消费Binlog数据时超时时间，单位毫秒	Integer	600000
fullSyncBinlogReadTimeout	全量消费Binlog数据时超时时间，单位毫秒	Long	1800000
binlogSleepTime	实时消费不到Binlog数据时休眠时间，单位毫秒。如果连续读取不到Binlog数据，则休眠时间为：binlogSleepTime * 次数，最大为binlogMaxSleepTime。读取到数据后，则重置。	Long	500
binlogMaxSleepTime	实时消费不到Binlog数据时最大休眠时间，单位毫秒。	Long	10000
binlogMaxRetryTimes	消费Binlog数据出错后的重试次数。	Integer	1
binlogRetryInterval	消费binlog数据出错后的重试时间间隔。重试时sleep时间：binlogRetryInterval * (1~binlogMaxRetryTimes) +Random（100）。单位毫秒。	Long	100
binlogParallelNum	消费Binlog数据时线程数，只有任务并发度小于DWS集群DN数时，该参数才有效，即此时一个并发度会消费多个DN上的数据，所以可以考虑设置该参数。	Integer	3
connectionPoolSize	JDBC连接池连接大小。	Integer	5

版权所有 © 华为云计算技术有限公司

参数	说明	数据类型	默认值
needRedistribution	表示是否兼容扩充重分布（需要升级到对应内核版本，如果是低版本则设置为false）；如果设置为true时，flink的restart-strategy不能设置为none。	Boolean	true
newSystemValue	表示读取binlog数据时是否使用新的系统字段（需要升级到对应内核版本，如果是低版本则设置为false）。	Boolean	true
checkNodeChangeInterval	检测节点变化的间隔，只有needRedistribution=true才生效。	Long	10000
connectionSocketTimeout	连接处理超时时间（可以看成客户端执行SQL超时时间），单位毫秒；默认值为0，即不设置超时时间。	Integer	0
binlogIgnoreUpdateBefore	是否过滤Binlog记录中的before_update记录，以及delete记录是否只返回主键信息。该参数仅9.1.0.200及以上版本支持。	Boolean	false
binlogStartTime	设置从某个时间点开始消费Binlog（只能增量消费），格式为yyyy-MM-dd hh:mm:ss且表需要开启enable_binlog_timestamp。 该参数仅9.1.0.200及以上版本支持。	String	无
binlogSyncPointSize	增量读取binlog同步点区间的大小（增量读取binlog时，如果数据量过大可能涉及下盘，可通过调整该参数控制）。 该参数仅9.1.0.200及以上版本支持。	Integer	5000
executeDirectOn	Binlog读取优化，全量同步时能有效避免CN结果集下盘问题，增量同步时能提升3至4倍性能。 该参数仅9.1.0.200及以上版本支持。	Boolean	false
binlogQueueSize	Binlog增量读取时队列长度，当Flink计算压力跟内存压力大时，可调小该参数来实现限流。 该参数仅9.1.0.200及以上版本支持。	Integer	0（表示无限制）

版权所有 © 华为云计算技术有限公司

数据同步示例

- DWS侧：

  说明

  新建binlog表时，**enable_hstore_binlog_table**参数需设置为on，可以通过**show enable_hstore_binlog_table;**命令查询。

  源表（产生binlog）：

  ```
  CREATE TABLE test_binlog_source(a int, b int, c int, primary key(a)) with(orientation=column, enable_hstore_opt=on, enable_binlog=true);
  ```

  目标表：

  ```
  CREATE TABLE test_binlog_sink(a int, b int, c int, primary key(a)) with(orientation=column, enable_hstore_opt=on);
  ```

- Flink侧：

  执行如下命令进行完整数据同步。

  建立源表的映射表：

  ```
  CREATE TABLE test_binlog_source (
    a int,
    b int,
    c int,
    primary key(a) NOT ENFORCED
  ) with (
    'connector' = 'dws',
    'url' = 'jdbc:gaussdb://ip:port/gaussdb',
    'binlog' = 'true',
    'tableName' = 'test_binlog_source',
    'binlogSlotName' = 'slot',
    'executeDirectOn' = 'true',
    'username'='xxx',
    'password'='xxx');
  ```

  建立目标表的映射表：

  ```
  CREATE TABLE test_binlog_sink (
    a int,
    b int,
    c int,
    primary key(a) NOT ENFORCED
  ) with (
    'connector' = 'dws',
    'url' = 'jdbc:gaussdb://ip:port/gaussdb',
    'tableName' = 'test_binlog_sink',
    'ignoreUpdateBefore'='false',
    'connectionSize' = '1',
    'username'='xxx',
    'password'='xxx');

  INSERT INTO test_binlog_sink select * from test_binlog_source;
  ```

使用 Java 程序示例

新建源表和目标表：

```
-- source
create table binlog_test_source(a int, b int, c int, primary key(a)) with(orientation=column, enable_hstore_opt=on, enable_binlog=true);
-- sink
create table binlog_test_sink(a int, b int, c int, primary key(a)) with(orientation=column, enable_hstore_opt=on, enable_binlog=true);
```

demo程序：

```
public class BinlogDemo {
```


版权所有 © 华为云计算技术有限公司

```
    // binlog表的表名
    private static final String BINLOG_TABLE_NAME = "binlog_test_source";

    // binlog表的槽位名
    private static final String BINLOG_SLOT_NAME = "binlog_test_slot";

    // 写入表的表名
    private static final String SINK_TABLE_NAME = "binlog_test_sink";

    public static void main(String[] args) throws Exception {
        DwsConfig dwsConfig = buildDwsConfig();
        DwsClient dwsClient = new DwsClient(dwsConfig);

        TableSchema sourceTableSchema =
dwsClient.getTableSchema(TableName.valueOf(BINLOG_TABLE_NAME));
        TableSchema sinkTableSchema = dwsClient.getTableSchema(TableName.valueOf(SINK_TABLE_NAME));

        // 需要写入哪些列
        List<String> sinkColumns = sinkTableSchema.getColumnNames();

        // 线程池
        DwsConnectionPool dwsConnectionPool = new DwsConnectionPool(dwsConfig);
        // 存放数据的队列
        BlockingQueue<BinlogRecord> queue = new LinkedBlockingQueue<>();
        // 需要同步哪些列
        List<String> sourceColumnNames = sourceTableSchema.getColumnNames();

        BinlogReader binlogReader = new BinlogReader(dwsConfig, queue, sourceColumnNames,
dwsConnectionPool);

        // 启动读取任务
        binlogReader.start();
        binlogReader.getRecords();

        while (binlogReader.isStart()) {
            try {
                while (!queue.isEmpty() && !binlogReader.hasException()) {
                    // 读取数据
                    BinlogRecord record = queue.poll();
                    if (Objects.isNull(record)) {
                        continue;
                    }
                    BinlogRecordType type = BinlogRecordType.toBinlogRecordType(record.getType());
                    List<Object> columnValues = record.getColumnValues();

                    // 写入数据
                    if (BinlogRecordType.INSERT.equals(type) || BinlogRecordType.UPDATE_AFTER.equals(type)) {
                        Operate upsert = dwsClient.write(sinkTableSchema);
                        for (int i = 0; i < sinkColumns.size(); i++) {
                            upsert.setObject(i, columnValues.get(i), false);
                        }
                        upsert.commit();
                    } else if (BinlogRecordType.DELETE.equals(type) ||
BinlogRecordType.UPDATE_BEFORE.equals(type)) {
                        Operate delete = dwsClient.delete(sinkTableSchema);
                        for (int i = 0; i < sinkColumns.size(); i++) {
                            String field = sinkColumns.get(i);
                            if (!sinkTableSchema.isPrimaryKey(field)) {
                                continue;
                            }
                            delete.setObject(i, columnValues.get(i), false);
                        }
                        delete.commit();
                    }
                }
                binlogReader.checkException();
            } catch (Exception e) {
                throw new DwsClientException(ExceptionCode.GET_BINLOG_ERROR, "get binlog  has error", e);
            }
```

 版权所有 © 华为云计算技术有限公司

```
        }
    }

    private static DwsConfig buildDwsConfig() {
        // 初始化一些配置信息(只列举一些必要的配置,具体配置信息请参考文档)
        TableConfig tableConfig = new TableConfig().withBinlog(true)
                .withNewSystemValue(true)
                .withNeedRedistribution(false)
                .withBinlogSlotName(BINLOG_SLOT_NAME);
        return DwsConfig.builder()
                .withUrl("链接信息")
                .withUsername("用户名")
                .withPassword("密码")
                .withBinlogTableName(BINLOG_TABLE_NAME)
                .withTableConfig(BINLOG_TABLE_NAME, tableConfig)
                .build();
    }
}
```

 版权所有 © 华为云计算技术有限公司

17 DWS 系统表和系统视图

17.1 系统表和系统视图概述

系统表是DWS存放结构元数据，是DWS数据库系统运行控制信息的来源，也是数据库系统的核心组成部分。系统表包含集群安装信息以及DWS上运行的各种查询和进程的信息。可以通过查询系统表来收集有关数据库的信息。

系统视图提供了查询系统表和访问数据库内部状态的方法。当用户对数据库中的一张或者多张表的某些字段的组合感兴趣，而又不想每次键入这些查询时，用户就可以定义一个视图来解决这个问题。视图与基本表不同，不是物理上实际存在的，是一个虚表。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。视图每次被引用的时候都会运行一次。

三权分立下，非管理员无权查看系统表和视图。非三权分立下，系统表和系统视图要么只对管理员可见，要么对所有用户可见。下面的系统表和视图有些标识了需要管理员权限，这些系统表和视图只有管理员可以查询。

⚠ 注意

- 禁止对系统表或系统视图进行增删改等操作，手动对系统表或系统视图的修改或破坏可能会导致系统信息不一致，造成系统控制异常甚至出现集群不可用的情况。
- 系统表不支持toast，无法跨页存储，一个页面大小为8K，系统表各个字段长度需小于8K。

17.2 系统表

17.2.1 GS_BLOCKLIST_QUERY

GS_BLOCKLIST_QUERY系统表存储作业黑名单信息和异常信息，该表以unique_sql_id作为唯一索引，进行作业异常信息统计和黑名单记录，可通过与GS_WLM_SESSION_INFO进行关联获取作业的query字段和执行信息。

版权所有 © 华为云计算技术有限公司

说明

- GS_BLOCKLIST_QUERY系统表的schema为dbms_om。
- GS_BLOCKLIST_QUERY系统表仅限在**postgres**数据库中查询，其他数据库中查询会直接报错。
- GS_BLOCKLIST_QUERY系统表包含唯一索引，使用哈希分布方式分布在DN上，分布列为unique_sql_id。
- 通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个unique_sql_id可能会对应一个或多个查询。

DWS同时提供了**GS_BLOCKLIST_QUERY**视图用于查询作业黑名单和异常信息，该视图可直接显示query字段信息，不过因为该视图与GS_WLM_SESSION_INFO为依赖关系，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长的时间。

表 17-1 GS_BLOCKLIST_QUERY 字段

名字	类型	引用	描述
unique_sql_id	bigint	-	基于查询解析树生成的查询ID。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。

17.2.2 GS_BLOCKLIST_SQL

GS_BLOCKLIST_SQL系统表存储作业黑名单信息和异常信息，该表以sql_hash作为唯一索引，进行作业异常信息统计和黑名单记录，可通过与**GS_WLM_SESSION_INFO**进行关联获取作业的query字段和执行信息。

DWS同时提供了**GS_BLOCKLIST_SQL**视图用于查询作业黑名单和异常信息，该视图可直接显示query字段信息，不过因为该视图与GS_WLM_SESSION_INFO为依赖关系，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长的时间。

该系统表仅9.1.0.200及以上集群版本支持。

说明

- GS_BLOCKLIST_SQL系统表的schema为dbms_om。
- GS_BLOCKLIST_SQL系统表仅限在**postgres**数据库中查询，其他数据库中查询会直接报错。
- GS_BLOCKLIST_SQL系统表包含唯一索引，使用哈希分布方式分布在DN上，分布列为sql_hash。
- 通常对于DML语句，在计算sql_hash的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个sql_hash可能会对应一个或多个查询。

表 17-2 GS_BLOCKLIST_SQL 字段

名字	类型	引用	描述
sql_hash	text	-	基于查询解析树生成的sql_hash。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。

17.2.3 GS_OBSSCANINFO

GS_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 17-3 GS_OBSSCANINFO 字段

名字	类型	描述
query_id	bigint	查询标识。
user_id	text	执行该查询的数据库用户。
table_name	text	OBS外表的表名。
file_type	text	底层数据保存的文件格式。
time_stamp	timestamp with time zone	扫描操作开始的时间。
actual_time	double	扫描操作执行时间，单位为秒。
file_scanned	bigint	扫描的文件数量。
data_size	double	扫描的数据量，单位为字节。
billing_info	text	保留字段。

17.2.4 GS_RESPOOL_RESOURCE_HISTORY

GS_RESPOOL_RESOURCE_HISTORY系统表记录资源池监控历史信息，CN和DN上均进行记录。

表 17-4 GS_RESPOOL_RESOURCE_HISTORY 字段

名称	类型	描述
timestamp	timestamp	资源池监控信息持久化时间。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5秒监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为DWS可用CPU，CPU限额管控情况下为关联控制组可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5秒监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_kbytes	bigint	资源池5s监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5秒监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5秒监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5秒监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5秒监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5秒监控周期内网络发送平均速率，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5秒监控周期内网络接收平均速率，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。
obs_io_read_kbytes	bigint	资源池5秒监控周期内OBS IO读字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO读字节数。 ● CN：显示所有DN上资源池OBS IO读字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_write_kbytes	bigint	资源池5秒监控周期内OBS IO写字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO写字节数。 ● CN：显示所有DN上资源池OBS IO写字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double	资源池5秒监控周期内OBS IO读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO读速率。 ● CN：显示所有DN上资源池OBS IO读速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double	资源池5秒监控周期内OBS IO写速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO写速率。 ● CN：显示所有DN上资源池OBS IO写速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。

17.2.5 GS_WLM_INSTANCE_HISTORY

GS_WLM_INSTANCE_HISTORY系统表存储与实例(CN或DN)相关的资源使用相关信息。该系统表里每条记录都是对应时间点某实例资源使用情况，包括：内存、CPU核数、磁盘IO、进程物理IO和进程逻辑IO信息。

表 17-5 GS_WLM_INSTANCE_HISTORY 字段

名称	类型	描述
instancename	text	实例名称。
timestamp	timestamp with time zone	时间戳。
used_cpu	int	实例使用CPU所占用的百分比。
free_mem	int	实例未使用的内存大小，单位MB。
used_mem	int	实例已使用的内存大小，单位MB。
io_await	real	实例所使用磁盘的io_wait值（10秒均值）。
io_util	real	实例所使用磁盘的io_util值（10秒均值）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
disk_read	real	实例所使用磁盘的读速率（10秒均值），单位KB/s。
disk_write	real	实例所使用磁盘的写速率（10秒均值），单位KB/s。
process_read	bigint	实例对应进程从磁盘读数据的读速率(不包括从磁盘pagecache中读取的字节数，10秒均值)，单位KB/s。
process_write	bigint	实例对应进程向磁盘写数据的写速率(不包括向磁盘pagecache中写入的字节数，10秒均值)，单位KB/s。
logical_read	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读字节速率，单位KB/s。
logical_write	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写字节速率，单位KB/s。
read_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读操作次数之和，单位次。
write_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写操作次数之和，单位次。

17.2.6 GS_WLM_OPERATOR_INFO

GS_WLM_OPERATOR_INFO系统表显示执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表中的数据。当设置GUC参数**enable_resource_record**为on时，系统会定时将**GS_WLM_OPERATOR_HISTORY**中的记录导入此系统表，开启此功能会占用系统存储空间并对性能有一定影响，建议在性能定位及监控任务完成后及时关闭。

说明

- GS_WLM_OPERATOR_INFO系统表的schema为dbms_om。
- GS_WLM_OPERATOR_INFO系统表仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。

表 17-6 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值，单位MB。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量，单位MB。默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量，单位MB。默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量，单位MB。默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间，单位毫秒。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间，单位毫秒。
total_cpu_time	bigint	该算子在所有DN上的总执行时间，单位毫秒。
cpu_skew_percent	integer	DN间执行时间的倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

17.2.7 GS_WLM_SESSION_INFO

GS_WLM_SESSION_INFO系统表显示当前CN执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。当设置GUC参数**enable_resource_record**为on时，系统会定时将**GS_WLM_SESSION_HISTORY**中的记录导入此系统表，开启此功能会占用系统存储空间并对性能有一定影响，建议在性能定位及监控任务完成后及时关闭。

不同版本及数据仓库类型下对应的GS_WLM_SESSION_INFO系统表名称不同：

- **存算一体**：
  9.1.0及以下集群版本查询时，需使用GS_WLM_SESSION_INFO系统表名。
  9.1.1及以上集群版本查询时，需使用GS_WLM_SESSION_INFO_V2系统表名。
- **存算分离**：查询时需使用GS_WLM_SESSION_INFO_V3系统表名。

9.1.1.300及以上集群版本中系统会将执行成功且执行时间小于100ms，并且包含sql_hash的短查询进行聚合记录。9.1.2及以上集群版本中，子语句聚合不受100ms执行时间限制，满足其他聚合条件时默认进行聚合。聚合的主要列为datid、dbname、resource_pool、node_group、username、application_name、client_addr、sql_hash、plan_hash、query_band和nodename。

说明

- GS_WLM_SESSION_INFO系统表的schema为dbms_om。
- GS_WLM_SESSION_INFO系统表仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。

GS_WLM_SESSION_INFO系统表的字段同**表17-195**相同，具体字段内容如下：

表 17-7 GS_WLM_SESSION_HISTORY 字段

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
datid	oid	连接后端的数据库OID。	连接后端的数据库OID。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
dbname	text	连接后端的数据库名称。	连接后端的数据库名称。
schemaname	text	模式名。	聚合周期内，同聚合主键第一个语句的schemaname。
nodename	text	语句执行的CN名称。	语句执行的CN名称。
username	text	连接到后端的用户名。	连接到后端的用户名。
application_name	text	连接到后端的应用名。	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。	连接到后端的客户端的IP地址。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。	聚合周期内，同聚合主键第一个语句的客户端的client_hostname。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。	聚合周期内，同聚合主键第一个语句的client_port。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。	聚合周期内，同聚合主键第一个语句的query_band。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。	聚合周期内，同聚合主键所有语句的block_time平均值。
start_time	timestamp with time zone	语句执行的开始时间。	聚合周期内，同聚合主键第一个语句的start_time。
finish_time	timestamp with time zone	语句执行的结束时间。	聚合周期内，同聚合主键最后一个语句的finish_time。
duration	bigint	语句实际执行的时间，单位ms。	聚合周期内，同聚合主键所有语句的duration平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
estimate_total_time	bigint	语句预估执行时间，单位ms。	聚合周期内，同聚合主键所有语句的estimate_total_time平均值。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。	语句执行结束状态，仅聚合执行成功的语句，默认为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。	空。
resource_pool	text	用户使用的资源池。	用户使用的资源池。
control_group	text	语句所使用的Cgroup。	聚合周期内，同聚合主键第一个语句的control_group。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。	聚合周期内，同聚合主键所有语句的estimate_memory平均值。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的min_peak_memory最小值。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的max_peak_memory最大值。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。	聚合周期内，同聚合主键所有语句的average_peak_memory平均值。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。	聚合周期内，同聚合主键所有语句的memory_skew_percent的平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。	空
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的min_spill_size的最小值。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的max_spill_size的最大值。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的average_spill_size的平均值。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。	聚合周期内，同聚合主键所有语句的spill_skew_percent的平均值。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。	聚合周期内，同聚合主键所有语句的min_dn_time的最小值。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。	聚合周期内，同聚合主键所有语句的max_dn_time的最大值。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。	聚合周期内，同聚合主键所有语句的average_dn_time平均值。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。	聚合周期内，同聚合主键所有语句的dntime_skew_percent平均值。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的min_cpu_time最小值。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的max_cpu_time最大值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。	聚合周期内，同聚合主键所有语句的total_cpu_time总量。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。	聚合周期内，同聚合主键所有语句的cpu_skew_percent平均值。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的min_peak_iops最小值。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的max_peak_iops最大值。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的average_peak_iops平均值。
iops_skew_percent	integer	语句在DN间的IO倾斜率。	聚合周期内，同聚合主键所有语句的iops_skew_percent平均值。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警: 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin	聚合周期内，同聚合主键第一个语句的warning。
queryid	bigint	语句执行使用的内部query id。	聚合周期内，同聚合主键第一个语句的queryid。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
query	text	执行的语句，最多可保留64KB长度的字符串。	聚合周期内，同聚合主键第一个语句的query。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。	聚合周期内，同聚合主键第一个语句的query_plan。
node_group	text	语句所属用户对应的逻辑集群。	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。	聚合周期内，同聚合主键第一个语句的pid。
lane	text	语句执行时所在的快慢车道。	聚合周期内，同聚合主键第一个语句的lane。
unique_sql_id	bigint	归一化的Unique SQL ID。	聚合周期内，同聚合主键第一个语句的unique_sql_id。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。	聚合周期内，同聚合主键第一个语句的session_id。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_read_bytes最小值。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_read_bytes最大值。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_read_bytes平均值。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_write_bytes最小值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_write_bytes最大值。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_write_bytes平均值。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。	聚合周期内，同聚合主键所有语句的recv_pkg总量。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。	聚合周期内，同聚合主键所有语句的send_pkg总量。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的recv_bytes总量。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的send_bytes总量。
stmt_type	text	语句对应的查询类型。	聚合周期内，同聚合主键第一个语句的stmt_type。
except_info	text	语句触发的异常规则信息。	空
unique_plan_id	bigint	归一化的Unique plan id。	聚合周期内，同聚合主键第一个语句的unique_plan_id。
sql_hash	text	归一化的sql hash。	归一化的sql hash。
plan_hash	text	归一化的plan hash。	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。	聚合周期内，同聚合主键第一个语句的use_plan_baseline。
outline_name	text	该语句计划对应的outline的名字。	聚合周期内，同聚合主键第一个语句的outline_name。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。	保存导入导出类业务信息的json串具体如下。 1. address：聚合周期内，同聚合主键第一个语句的address。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：聚合周期内，同聚合主键所有语句的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的parse_time平均值。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_hit_ratio总量。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_read_size总量。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_write_size总量。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_size总量。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_time总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_scan_bytes总量。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_remote_read_bytes总量。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_submit_time总量。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_time总量。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_count总量。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_load_time总量。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_conflict_count总量。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_error_count总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。	空
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_rtt总量。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_latency总量。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_1s总量。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_10s总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_count总量。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_retry_count总量。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_rate_limit_count总量。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键第一个语句的gxid。
inserted_rows	bigint	语句插入的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的inserted_rows平均值
deleted_rows	bigint	语句删除的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的deleted_rows平均值
updated_rows	bigint	语句更新的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的updated_rows平均值
returned_rows	bigint	语句查询的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的returned_rows平均值
obs_io_read_bytes	bigint	语句在所有DN上的平均OBS IO读字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_read_bytes平均值
obs_io_write_bytes	bigint	语句在所有DN上的平均OBS IO写字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_write_bytes平均值
calls	integer	默认为1。	聚合周期内，同聚合主键的查询数量。

版权所有 © 华为云计算技术有限公司

17.2.8 GS_WLM_USER_RESOURCE_HISTORY

GS_WLM_USER_RESOURCE_HISTORY系统表存储与用户使用资源相关的信息，该表在CN和DN上均存有数据。该系统表的每条记录都是对应时间点某用户的资源使用情况，包括：内存、CPU核数、存储空间、临时空间、算子落盘空间、逻辑IO流量、逻辑IO次数和逻辑IO速率信息。其中，内存、CPU、IO相关监控项仅记录用户复杂作业的资源使用情况。

GS_WLM_USER_RESOURCE_HISTORY系统表的数据来源于PG_TOTAL_USER_RESOURCE_INFO视图。

表 17-8 GS_WLM_USER_RESOURCE_HISTORY 字段

名称	类型	描述
username	text	用户名。
timestamp	timestamp with time zone	时间戳。
used_memory	int	用户使用的内存大小，单位MB。 • DN：显示当前DN上用户使用的内存大小。 • CN：显示所有DN上用户使用内存的累积和。
total_memory	int	资源池使用的内存大小，单位：MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存max_dynamic_memory。具体的计算公式为： total_memory = max_dynamic_memory * parent_percent * user_percent CN：显示所有DN上用户可用内存上限的累积和。
used_cpu	real	正在使用的CPU核数。
total_cpu	int	该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的存储空间大小，单位KB。
total_space	bigint	可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。
used_temp_space	bigint	已使用的临时存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时存储空间大小，单位KB，值为-1表示未限制最大临时存储空间。
used_spill_space	bigint	已使用的算子落盘存储空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘存储空间大小，单位KB，值为-1表示未限制最大算子落盘存储空间。
read_kbytes	bigint	监控周期内，读操作的字节流量，单位KB。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_kbytes	bigint	监控周期内，写操作的字节流量，单位KB。
read_counts	bigint	监控周期内，读操作的次数，单位次。
write_counts	bigint	监控周期内，写操作的次数，单位次。
read_speed	real	监控周期内，读操作的字节速率，单位KB/s。
write_speed	real	监控周期内，写操作的字节速率，单位KB/s。
send_speed	double	监控周期内，网络发送速率，单位KB/s。
recv_speed	double	监控周期内，网络接收速率，单位KB/s。
obs_io_read_kbytes	bigint	监控周期内，OBS IO读操作的字节流量，单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	监控周期内，OBS IO写操作的字节流量，单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double	监控周期内，OBS IO读操作的字节速率，单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double	监控周期内，OBS IO写操作的字节速率，单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。

17.2.9 PG_AGGREGATE

PG_AGGREGATE系统表存储与聚集函数有关的信息。PG_AGGREGATE里的每条记录都是一条pg_proc里面的记录的扩展。PG_PROC记录承载该聚集的名字、输入和输出数据类型，以及其他一些和普通函数类似的信息。

表 17-9 PG_AGGREGATE 字段

名字	类型	引用	描述
aggfnoid	regproc	PG_PROC.oid	此聚集函数的PG_PROC OID。
aggtransfn	regproc	PG_PROC.oid	转换函数。
aggcollectfn	regproc	PG_PROC.oid	收集函数。
aggfinalfn	regproc	PG_PROC.oid	最终处理函数（如果没有则为0）。
aggsortop	oid	PG_OPERATOR.oid	关联排序操作符（如果没有则为0）。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
aggtranstype	oid	PG_TYPE.oid	此聚集函数的内部转换（状态）数据的数据类型。
agginitval	text	-	转换状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则转换状态值从null开始。
agginitcollect	text	-	收集状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则收集状态值从null开始。

17.2.10 PG_AM

PG_AM系统表存储有关索引访问方法的信息。系统支持的每种索引访问方法都有一行。

表 17-10 PG_AM 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amname	name	-	访问方法的名称。
amstrategies	smallint	-	访问方法的操作符策略个数，或者如果访问方法没有一个固定的操作符策略集则为0。
amsupport	smallint	-	访问方法的支持过程个数。
amcanorder	boolean	-	这种访问方式是否支持通过索引字段值的命令扫描排序。
amcanorderbyop	boolean	-	这种访问方式是否支持通过索引字段上操作符的结果的命令扫描排序。
amcanbackward	boolean	-	访问方式是否支持向后扫描。
amcanunique	boolean	-	访问方式是否支持唯一索引。
amcanmulticol	boolean	-	访问方式是否支持多字段索引。
amoptionalkey	boolean	-	访问方式是否支持第一个索引字段上没有任何约束的扫描。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
amsearcharray	boolean	-	访问方式是否支持ScalarArrayOpExpr搜索。
amsearchnulls	boolean	-	访问方式是否支持IS NULL/NOT NULL搜索。
amstorage	boolean	-	允许索引存储的数据类型与列的数据类型是否不同。
amclusterable	boolean	-	是否允许在一个这种类型的索引上集群。
ampredlocks	boolean	-	是否允许这种类型的一个索引管理细粒度的谓词锁定。
amkeytype	oid	PG_TYPE.oid	存储在索引里数据的类型，如果不是一个固定的类型则为0。
aminsert	regproc	PG_PROC.oid	“插入此行”函数。
ambeginscan	regproc	PG_PROC.oid	“准备索引扫描” 函数。
amgettuple	regproc	PG_PROC.oid	“下一个有效行”函数，如果没有则为0。
amgetbitmap	regproc	PG_PROC.oid	“抓取所有的有效行” 函数，如果没有则为0。
amrescan	regproc	PG_PROC.oid	“（重新）开始索引扫描”函数。
amendscan	regproc	PG_PROC.oid	“索引扫描后清理” 函数。
ammarkpos	regproc	PG_PROC.oid	“标记当前扫描位置”函数。
amrestrpos	regproc	PG_PROC.oid	“恢复已标记的扫描位置”函数。
ammerge	regproc	PG_PROC.oid	“归并多个索引对象”函数。
ambuild	regproc	PG_PROC.oid	“建立新索引”函数。
ambuildempty	regproc	PG_PROC.oid	“建立空索引”函数。
ambulkdelete	regproc	PG_PROC.oid	批量删除函数。
amvacuumcleanup	regproc	PG_PROC.oid	VACUUM后的清理函数。
amcanreturn	regproc	PG_PROC.oid	检查是否索引支持唯一索引扫描的函数，如果没有则为0。
amcostestimate	regproc	PG_PROC.oid	估计一个索引扫描开销的函数。
amoptions	regproc	PG_PROC.oid	用于分析和验证索引的reloptions函数。

 版权所有 © 华为云计算技术有限公司

17.2.11 PG_AMOP

PG_AMOP系统表存储和访问方法操作符族关联的信息。如果一个操作符是一个操作符族中的成员，则在这个表中会占据一行。一个族成员是一个search操作符或一个ordering操作符。一个操作符可以在多个族中出现，但是不能在一个族中的多个搜索位置或多个排序位置中出现。

表 17-11 PG_AMOP 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amopfamily	oid	PG_OPFAMILY.oid	该项的操作符族。
amoplefttype	oid	PG_TYPE.oid	操作符的左输入类型。
amoprighttype	oid	PG_TYPE.oid	操作符的右输入类型。
amopstrategy	smallint	-	操作符策略数。
amoppurpose	"char"	-	操作符目的，s为搜索或o为排序。
amopopr	oid	PG_OPERATOR.oid	该操作符的OID。
amopmethod	oid	PG_AM.oid	使用此操作符族的索引访问方法。
amopsortfamily	oid	PG_OPFAMILY.oid	如果是一个排序操作符，则该项会按照B-Tree操作符族排序；如果是一个搜索操作符，则为0。

search操作符表明这个操作符族的一个索引可以被搜索，找到所有满足WHERE indexed_column operator constant的行。显然，这样的操作符必须返回布尔值，并且它的左输入类型必须匹配索引的字段数据类型。

ordering操作符表明这个操作符族的一个索引可以被扫描，返回以ORDER BY indexed_column operator constant顺序表示的行。这样的操作符可以返回任意可排序的数据类型，它的左输入类型也必须匹配索引的字段数据类型。ORDER BY的确切语义是由amopsortfamily字段指定的，该字段必须为操作符的返回类型引用一个B-Tree操作符族。

17.2.12 PG_AMPROC

PG_AMPROC系统表存储与访问方法操作符族相关联的支持过程的信息。每个属于某个操作符族的支持过程都占有一行。

 版权所有 © 华为云计算技术有限公司

表 17-12 PG_AMPROC 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amprocfamily	oid	PG_OPFAMILY.oid	该项的操作符族。
amproclefttype	oid	PG_TYPE.oid	相关操作符的左输入数据类型。
amprocrighttype	oid	PG_TYPE.oid	相关操作符的右输入数据类型。
amprocnum	smallint	-	支持过程编号。
amproc	regproc	PG_PROC.oid	过程的OID。

amproclefttype和amprocrighttype字段的习惯解释，标识一个特定支持过程所支持的操作符的左右输入类型。对于某些访问方式，匹配支持过程本身的输入数据类型，对其他的则不会匹配。有一个对索引的“缺省”支持过程的概念，amproclefttype和amprocrighttype都等于索引操作符类的opcintype。

17.2.13 PG_ATTRDEF

PG_ATTRDEF系统表存储字段的默认值。

表 17-13 PG_ATTRDEF 字段

名称	类型	描述
adrelid	oid	该字段所属的表。
adnum	smallint	字段编号。
adbin	pg_node_tree	字段缺省值的内部表现形式。
adsrc	text	人类可读的缺省值的内部表现形式。
adbin_on_update	pg_node_tree	字段on_update_expr值的内部表现形式。
adsrc_on_update	text	人类可读的on_update_expr值的内部表现形式。

17.2.14 PG_ATTRIBUTE

PG_ATTRIBUTE系统表存储关于表字段的信息。

 版权所有 © 华为云计算技术有限公司

表 17-14 PG_ATTRIBUTE 字段

名称	类型	描述
attrelid	oid	该字段所属的表。
attname	name	字段名。
atttypid	oid	字段类型。
attstattarget	integer	控制ANALYZE为该字段设置的统计细节的级别。 ● -1，表示未设置，使用全局参数default_statistics_target的值（默认值为100）计算采样大小。 ● 0，表示该列不进行采样。 ● 大于0，表示使用固定值计算采样大小，该值 * 300作为随机抽样的大小。例如，设置为200时，采样大小为200* 300。 ● 小于-1，表示使用百分比方式计算采样大小，采样率=(attstattarget + 1) * (-1) /100。 对于标量数据类型，ATTSTATTARGET既是要收集的“最常用数值”的目标数目，也是要创建的柱状图的目标数量。
attlen	smallint	是本字段类型pg_type.typlen的复制。
attnum	smallint	字段编号。
attndims	integer	如果该字段是数组，该值表示数组的维数，否则是0 。
attcacheoff	integer	在磁盘上总是-1 ，但是如果加载入内存中的行描述器中，它可能会被更新为缓冲在行中字段的偏移量。
atttypmod	integer	记录创建新表时支持的类型特定的数据（比如，varchar字段的最大长度）。它传递给类型相关的输入和长度转换函数当做第三个参数。其值对那些不需要ATTTYPMOD的类型通常为-1。
attbyval	boolean	pg_type.typbyval字段值的复制。
attstorage	"char"	pg_type.typstorage字段值的复制。
attalign	"char"	pg_type.typalign字段值的复制。
attnotnull	boolean	代表一个非空约束。可以改变这个字段来打开或者关闭该约束。
atthasdef	boolean	该字段是否存在缺省值，此时它对应pg_attrdef表里实际定义此值的记录。
attisdropped	boolean	该字段是否已经被删除，不再有效。如果被删除，该字段物理上仍然存在表中，但会被分析器忽略，因此不能再通过SQL访问。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
attislocal	boolean	该字段是否局部定义在对象中。一个字段可以同时是局部定义和继承的。
attcmprmode	tinyint	对某一列指定压缩方式。压缩方式包括： ● ATT_CMPR_NOCOMPRESS ● ATT_CMPR_DELTA ● ATT_CMPR_DICTIONARY ● ATT_CMPR_PREFIX ● ATT_CMPR_NUMSTR
attinhcount	integer	该字段所拥有的直接父表的个数。如果一个字段的父表个数非零，则它就不能被删除或重命名。
attcollation	oid	该列定义的排序规则。
attacl	aclitem[]	列级访问权限控制。
attoptions	text[]	属性级可选项。
attfdwoptions	text[]	属性级外数据选项。
attinitdefval	bytea	存储了此列默认的值表达式。行存表的ADD COLUMN需要使用此字段。
attkvtype	tinyint	该字段的kv_type属性。取值如下： ● 0表示默认值，用于非时序表。 ● 1表示维度属性(TSTAG)，仅用于时序表。 ● 2表示指标属性(TSFIELD)，仅用于时序表。 ● 3表示时间属性(TSTIME)，仅用于时序表。

应用示例

查询指定表中包含的字段名和字段编号。t1和public分别替换为实际的表名和schema名称。

```
SELECT attname,attnum FROM pg_attribute WHERE attrelid=(SELECT pg_class.oid FROM pg_class JOIN pg_namespace ON relnamespace=pg_namespace.oid WHERE relname='t1' and nspname='public') and attnum>0;
    attname      | attnum
-----------------+--------
 product_id      |      1
 product_name    |      2
 product_quantity |      3
(3 rows)
```

17.2.15 PG_AUTHID

PG_AUTHID系统表存储有关数据库认证标识符（角色）的信息。角色把“用户”的概念包含在内。一个用户实际上就是一个rolcanlogin标志被设置的角色。任何角色（不管rolcanlogin设置与否）都能够把其他角色作为成员。

 版权所有 © 华为云计算技术有限公司

在一个集群中只有一份pg_authid，不是每个数据库有一份。需要有系统管理员权限才可以访问此系统表。

表 17-15 PG_AUTHID 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
rolname	name	角色名称。
rolsuper	boolean	角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	角色是否自动继承其所属角色的权限。
rolcreaterole	boolean	角色是否可以创建更多角色。
rolcreatedb	boolean	角色是否可以创建数据库。
rolcatupdate	boolean	角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	角色是否可以登录，即该角色是否能够作为初始会话授权标识符。
rolreplication	boolean	角色是一个复制的角色（适配作用，没有实际的功能）。
rolauditadmin	boolean	审计用户。
rolsystemadmin	boolean	管理员用户。
rolconnlimit	integer	限制单个用户在单个CN上的最大并发连接数。-1表示没有限制。
rolpassword	text	口令(可能是加密的)，如果没有口令，则为NULL。
rolvalidbegin	timestamp with time zone	账户的有效开始时间，如果没有开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	账户的有效结束时间，如果没有结束时间，则为NULL。
rolrespool	name	用户所能够使用的resource pool。
roluseft	boolean	角色是否可以操作外表。
rolparentid	oid	用户所在组用户的OID。
roltabspace	Text	用户永久表存储空间限额。
rolkind	char	特殊用户种类，包括私有用户、逻辑集群管理员、普通用户。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
rolnodegroup	oid	用户所关联的Node Group OID，该Node Group必须是逻辑集群。
roltempspace	Text	用户临时表存储空间限额。
rolspillspace	Text	用户算子落盘空间限额。
rolexcpdata	text	保留字段未使用。
rolauthinfo	text	用户采用LDAP或OneAccess认证时的额外信息。如果是其他认证模式，则为NULL。
rolpwdexpire	integer	用户口令过期时间，在口令未过期时，用户自己修改口令。过期后请管理员修改口令。-1表示没有过期时间限制。
rolpwdtime	timestamp with time zone	口令的创建时间。
roluuid	bigint	角色标识符。该字段仅9.1.0及以上集群版本支持。

17.2.16 PG_AUTH_HISTORY

PG_AUTH_HISTORY系统表记录了角色的认证历史。需要有系统管理员权限才可以访问此系统表。

表 17-16 PG_AUTH_HISTORY 字段

名称	类型	描述
roloid	oid	角色标识。
passwordtime	timestamp with time zone	创建和修改密码的时间。
rolpassword	text	角色密码，使用MD5、SHA256加密或者不加密。

17.2.17 PG_AUTH_MEMBERS

PG_AUTH_MEMBERS系统表存储显示角色之间的成员关系。

表 17-17 PG_AUTH_MEMBERS 字段

名称	类型	描述
roleid	oid	拥有成员的角色ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
member	oid	属于ROLEID角色的成员角色ID。
grantor	oid	赋予此成员关系的角色ID。
admin_option	boolean	如果有权限可以把ROLEID角色的成员关系赋予其他角色，则为真。

17.2.18 PG_BLOCKLISTS

PG_BLOCKLISTS系统表记录了查询过滤规则相关信息。该系统表仅9.1.0.100及以上集群版本支持。

表 17-18 PG_BLOCKLISTS 字段

名称	类型	描述
block_name	name	查询过滤规则名称。
role	oid	查询过滤规则绑定的用户OID。
client_addr	inet	查询过滤规则绑定的客户端IP地址。
application_name	name	查询过滤规则绑定的客户端名称。
unique_sql_id	int8	查询过滤规则匹配的unique_sql_id。
sql_hash	name	查询过滤规则匹配的sql_hash。
block_type	int4	查询过滤规则绑定的语句类型，分别对应SELECT/UPDATE/INSERT/DELETE/MERGE。
partition_num	int4	计划预估扫描节点扫描的最大分区数。
table_num	int4	计划预估扫描的最大表数。
estimate_row	int4	计划预估扫描节点扫描的最大行数。
query_band	text	主动标识的作业类型。
sql	text	查询过滤规则匹配的SQL语句。
created_time	timestamp with time zone	查询过滤规则创建或者修改的时间戳。
resource_pool	name	查询过滤规则拦截到语句所切换到的指定资源池名称。该字段仅9.1.0.200及以上集群版本支持。
max_active_num	integer	查询过滤规则拦截到语句的最大并发数。低于该数值时正常执行，大于等于该数值时会报错拦截。 该字段仅9.1.0.200及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
is_warning	boolean	控制查询过滤规则拦截到语句时的行为。 ● false表示拦截到语句时报错。默认值为false。 ● true表示拦截到语句时产生告警信息。 该字段仅9.1.0.200及以上集群版本支持。

17.2.19 PG_CAST

PG_CAST系统表存储数据类型之间的转化关系。

表 17-19 PG_CAST 字段

名称	类型	描述
castsource	oid	源数据类型的OID。
casttarget	oid	目标数据类型的OID。
castfunc	oid	转化函数的OID。0表示不需要转化函数。
castcontext	"char"	源数据类型和目标数据类型间的转化方式： ● e表示只能进行显式转化（使用CAST或::语法）。 ● i表示只能进行隐式转化。 ● a表示类型间同时支持隐式和显式转化。
castmethod	"char"	转化方法： ● f表示使用castfunc字段中指定的函数进行转化。 ● b表示类型间是二进制强制转化，不使用castfunc。

17.2.20 PG_CLASS

PG_CLASS系统表存储数据库对象信息及其之间的关系。

表 17-20 PG_CLASS 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
relname	name	表、索引、视图等对象的名称。
relnamespace	oid	包含该关系的命名空间的OID。
reltype	oid	与该表的行类型对应的数据类型（索引为零，因为索引没有pg_type记录）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
reloftype	oid	复合类型的OID，0表示其他类型。
relowner	oid	关系所有者。
relam	oid	如果行是索引，即为所用的访问模式（B-tree，hash等）。
relfilenode	oid	该关系在磁盘上的文件的名称，如果没有则为0。
reltablespace	oid	该关系存储所在的表空间。如果为0，则使用该数据库的缺省表空间。如果关系无磁盘文件，该字段无意义。
relpages	double precision	以页（大小为BLCKSZ）为单位的此表在磁盘上的大小，只是优化器使用的一个近似值。
reltuples	double precision	表中行的数目，只是优化器使用的一个估计值。
relallvisible	integer	被标识为全可见的表中的页数。此字段是优化器用来做SQL执行优化使用的。VACUUM、ANALYZE和一些DDL语句（例如，CREATE INDEX）会引起此字段更新。
reltoastrelid	oid	与此表关联的TOAST表的OID ，如果没有则为0。 TOAST表在一个从属表里“离线”存储大字段。
reltoastidxid	oid	对于TOAST表是它的索引的OID，如果不是TOAST表则为0。
reldeltarelid	oid	Delta表的OID。 Delta表附属于列存表。用于存储数据导入过程中的甩尾数据。
reldeltaidx	oid	Delta表的索引表OID。
relcudescrelid	oid	CU描述表的OID。 CU描述表（Desc表）附属于列存表。用于控制表目录中存储数据的可见性。
relcudescidx	oid	CU描述表的索引表OID。
relhasindex	boolean	如果对象是一个表且至少有（或者最近建有）一个索引，则为真。 由CREATE INDEX设置，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，会清理relhasindex字段，将relhasindex值设置为假。
relisshared	boolean	如果该表在整个集群中由所有数据库共享则为真。只有某些系统表（比如pg_database）是共享的。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
relpersistence	"char"	• p表示永久表。 • u表示非日志表。 • t表示本地临时表。 • g表示全局临时表。
relkind	"char"	• r表示普通表。 • i表示索引。 • S表示序列。 • v表示视图。 • c表示复合类型。 • t表示TOAST表。 • f表示外表。 • m表示物化视图。
relnatts	smallint	关系中用户字段数目（除了系统字段以外）。在pg_attribute里肯定有相同数目对应行。
relchecks	smallint	表上检查约束的数目，参见**PG_CONSTRAINT**。
relhasoids	boolean	如果为关系中每行都生成一个OID，则为真。
relhaspkey	boolean	如果该表有一个（或曾有）主键，则为真。
relhasrules	boolean	如果表有规则，则为真。是否有规则可参考系统表**PG_REWRITE**。
relhastriggers	boolean	如果表有（或曾有）触发器，则为真。参见**PG_TRIGGER**。
relhassubclass	boolean	如果表有（或曾有）任何继承的子表，则为真。
relcmprs	tinyint	表示是否启用表的压缩特性。需要特别注意，当且仅当批量插入才会触发压缩，普通的CRUD并不能够触发压缩。 • 0表示其他不支持压缩的表（主要是指系统表，不支持压缩属性的修改操作）。 • 1表示表数据的压缩特性为NOCOMPRESS或者无指定关键字。 • 2表示表数据的压缩特性为COMPRESS。
relhasclusterkey	boolean	是否有局部聚簇存储。
relrowmovement	boolean	针对分区表进行update操作时，是否允许行迁移。 • true：表示允许行迁移。 • false：表示不允许行迁移。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
parttype	"char"	表或者索引是否具有分区表的性质。 • p表示带有分区表性质。 • n表示没有分区表特性。 • v表示该表为HDFS的Value分区表。
relfrozenxid	xid32	该表中所有在此之间的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
relacl	aclitem[]	访问权限。 查询的回显结果为以下形式： rolename=xxxx/yyyy --赋予一个角色的权限 =xxxx/yyyy --赋予public的权限 xxxx表示赋予的权限，yyyy表示授予该权限的角色。权限的参数说明请参见表17-21。
reloptions	text[]	特定的访问方法选项，用"keyword=value"字符串形式表示。
relfrozenxid64	xid	该表中所有在此之前的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。

数据库对象权限使用 ACL（Access Control List）进行控制。DWS使用aclitem来表示一个具体的数据库对象上的权限。

- 对于database和schema，aclitem存储在pg_database.datacl和pg_namespace.nspacl中。
- 对于table、view等其他数据库对象，aclitem存储在pg_class.relacl中。
- 对于列级别的权限，aclitem存储在pg_attribute.attacl中。

例如，test_user=a*r/test_user1，表示test_user这个用户对当前数据库对象拥有INSERT和SELECT权限，其中INSERT权限带有with grant option，表示可以授权其他用户该权限。/test_user1表示这条aclitem权限是test_user1授予的。

下表列出了DWS中所有的数据库对象权限及其缩写。

 版权所有 © 华为云计算技术有限公司

表 17-21 ACL 权限的参数说明

权限	缩写	适用的对象类型
SELECT（读）	r	LARGE OBJECT，SEQUENCE，TABLE (and table-like objects)，table column
UPDATE（写）	w	LARGE OBJECT，SEQUENCE，TABLE，table column
INSERT（插入）	a	TABLE，table column
DELETE	d	TABLE
TRUNCATE	D	TABLE
REFERENCES	x	TABLE，table column
TRIGGER	t	TABLE
EXECUTE	X	FUNCTION，PROCEDURE
USAGE	U	DOMAIN，FOREIGN DATA WRAPPER，FOREIGN SERVER，LANGUAGE，SCHEMA，SEQUENCE，TYPE
CREATE	C	DATABASE，SCHEMA，TABLESPACE
CONNECT	c	DATABASE
TEMPORARY	T	DATABASE
ANALYZE|ANALYSE	A	TABLE
ALTER	L	TABLE
DROP	P	TABLE
VACUUM	v	TABLE
ALL PRIVILEGES	arwdDxtA，vLP	TABLE
给前面权限的授权选项	*	-

应用示例

查看某张表的oid及relfilenode。

```
SELECT oid,relname,relfilenode FROM pg_class WHERE relname = 'table_name';
```

统计行存表数量。

```
SELECT 'row count:'||count(1) as point FROM pg_class WHERE relkind = 'r' and oid > 16384 and
reloptions::text not like '%column%' and reloptions::text not like '%internal_mask%';
```

统计列存表数量。

 版权所有 © 华为云计算技术有限公司

```
SELECT 'column count:'||count(1) as point FROM pg_class WHERE relkind = 'r' and oid > 16384 and
reloptions::text like '%column%';
```

查询数据库中所有表的注释。

```
SELECT relname as tablename,obj_description(relfilenode,'pg_class') as comment FROM pg_class;
```

查看数据库中指定表的权限信息。

```
SELECT relacl FROM pg_class WHERE relname = 'table_name';
```

17.2.21 PG_COLLATION

PG_COLLATION系统表描述可用的排序规则，本质上从一个SQL名字映射到操作系统本地类别。

表 17-22 PG_COLLATION 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
collname	name	-	排序规则名（每个命名空间和编码唯一）。
collnamespace	oid	PG_NAMESPACE.oid	包含该排序规则的命名空间的OID。
collowner	oid	PG_AUTHID.oid	排序规则的所有者。
collencoding	integer	-	排序规则可用的编码，如果适用于任意编码为-1。
collcollate	name	-	排序规则对象的LC_COLLATE。
collctype	name	-	排序规则对象的LC_CTYPE。

17.2.22 PG_CONSTRAINT

PG_CONSTRAINT系统表存储表上的检查约束、主键、唯一约束和外键约束。

表 17-23 PG_CONSTRAINT 字段

名称	类型	描述
conname	name	约束名称（不一定是唯一的）。
connamespace	oid	包含约束的命名空间的OID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
contype	"char"	• c = 检查约束 • f = 外键约束 • p = 主键约束 • u = 唯一约束 • t = 触发器约束
condeferrable	boolean	该约束是否可以推迟。
condeferred	boolean	缺省时该约束是否可以推迟。
convalidated	boolean	约束是否有效。目前，只有外键和CHECK约束可将其设置为FALSE。
conrelid	oid	该约束所在的表；如果不是表约束则为0。
contypid	oid	该约束所在的域；如果不是一个域约束则为0。
conindid	oid	与约束关联的索引ID。
confrelid	oid	如果是外键，则为参考的表；否则为0。
confupdtype	"char"	外键更新动作代码。 • a = 没动作 • r = 限制 • c = 级联 • n =设置为null • d =设置为缺省
confdeltype	"char"	外键删除动作代码。 • a = 没动作 • r = 限制 • c = 级联 • n =设置为null • d =设置为缺省
confmatchtype	"char"	外键匹配类型。 • f = 全部 • p = 部分 • u = 简单（未指定）
conislocal	boolean	是否是为关系创建的本地约束。
coninhcount	integer	约束直接继承父表的数目。继承父表数非零时，不能删除或重命名该约束。
connoinherit	boolean	是否可以被继承。
consoft	boolean	是否为信息约束(Informational Constraint)。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
conopt	boolean	是否使用信息约束优化执行计划。
conkey	smallint[]	如果是表约束，则是约束控制的字段列表。
confkey	smallint[]	如果是一个外键，则是参考的字段的列表。
conpfeqop	oid[]	如果是一个外键，是做PK=FK比较的相等操作符ID的列表。
conppeqop	oid[]	如果是一个外键，是做PK=PK比较的相等操作符ID的列表。
conffeqop	oid[]	如果是一个外键，是做FK=FK比较的相等操作符ID的列表。
conexclop	oid[]	如果是一个排他约束，是列的排他操作符ID列表。
conbin	pg_node_tree	如果是检查约束，则是其表达式的内部形式。
consrc	text	如果是检查约束，则是表达式的人类可读形式。

须知

- 当被引用的对象改变时，consrc不能被更新。例如，它不会跟踪字段的重命名。建议还是使用pg_get_constraintdef()来抽取一个检查约束的定义，而不是依赖这个字段。
- pg_class.relchecks需要和每个关系在此目录中的检查约束数量保持一致。

应用示例

查询指定表是否有主键。

```
CREATE TABLE t1
(
    C_CUSTKEY     BIGINT       ,
    C_NAME        VARCHAR(25)  ,
    C_ADDRESS     VARCHAR(40)  ,
    C_NATIONKEY   INT          ,
    C_PHONE       CHAR(15)     ,
    C_ACCTBAL     DECIMAL(15,2),
    CONSTRAINT C_CUSTKEY_KEY PRIMARY KEY(C_CUSTKEY,C_NAME)
)
DISTRIBUTE BY HASH(C_CUSTKEY,C_NAME);

SELECT conname FROM pg_constraint WHERE conrelid = 't1'::regclass AND contype = 'p';
    conname
---------------
 c_custkey_key
(1 row)
```

 版权所有 © 华为云计算技术有限公司

17.2.23 PG_CONVERSION

PG_CONVERSION系统表描述编码转换信息。

表 17-24 PG_CONVERSION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
conname	name	-	转换名（在一个命名空间里唯一）。
connamespace	oid	PG_NAMESPACE.oid	包含此转换的命名空间的OID。
conowner	oid	PG_AUTHID.oid	编码转换的属主。
conforencoding	integer	-	源编码ID。
contoencoding	integer	-	目的编码ID。
conproc	regproc	PG_PROC.oid	转换过程。
condefault	boolean	-	如果为缺省转换则为真。

17.2.24 PG_DATABASE

PG_DATABASE系统表存储关于可用数据库的信息。

表 17-25 PG_DATABASE 字段

名称	类型	描述
datname	name	数据库名称。
datdba	oid	数据库所有者，通常为其创建者。
encoding	integer	数据库的字符编码方式。 pg_encoding_to_char()可以将此编号转换为编码名称。
datcollate	name	数据库使用的排序顺序。
datctype	name	数据库使用的字符分类。
datistemplate	boolean	是否允许作为模板数据库。
datallowconn	boolean	如果为假，则没有用户可以连接到这个数据库。这个字段用于保护template0数据库不被更改。
datconnlimit	integer	该数据库上允许的最大并发连接数，-1表示无限制。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
datlastsysoid	oid	数据库里最后一个系统OID 。
datfrozenxid	xid32	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。 为保持前向兼容，保留此字段，新增datfrozenxid64用于记录此信息。
dattablespace	oid	数据库的缺省表空间。
datcompatibility	name	数据库兼容模式。 • ORA，表示兼容Oracle数据库。 • TD，表示兼容Teradata数据库。 • MySQL，表示兼容MySQL数据库。
datacl	aclitem[]	访问权限。
datfrozenxid64	xid	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。

17.2.25 PG_DB_ROLE_SETTING

PG_DB_ROLE_SETTING系统表存储数据库运行时每个角色与数据绑定的配置项的默认值 。

表 17-26 PG_DB_ROLE_SETTING 字段

名称	类型	描述
setdatabase	oid	配置项所对应的数据库，如果未指定数据库，则为0。
setrole	oid	配置项所对应的角色，如果未指定角色，则为0。
setconfig	text[]	运行时配置项的默认值。

17.2.26 PG_DB_FILE

PG_DB_FILE系统表存储V3表的文件信息，并对databaseid、tablespaceid、relfilenode和fileid字段创建唯一索引。该系统表的更新由**enable_pg_db_file**参数控制，仅9.1.1.100及以上集群版本支持。

表 17-27 PG_DB_FILE 字段

名称	类型	描述
databaseid	oid	数据库OID。
tablespaceid	oid	表空间OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
relfilenode	oid	表的relfilenode。
fileid	bigint	文件ID。
type	"char"	数据类型（定义了CUDESC、CU、DELTA、ROW，目前只统计CU类型）。
refcount	bigint	用于零拷贝的字段。
filesize	bigint	统计文件大小，单位Byte。
ancestor	bigint	用于零拷贝的字段。
xid	xid	事务ID。

17.2.27 PG_DELAY_DELETE_FILE

PG_DELAY_DELETE_FILE系统表存储LStore延迟删除物理文件的信息，并对databaseid、tablespaceid、relfilenode和fileid字段创建唯一索引。该系统表仅9.1.2及以上集群版本支持。

PG_DELAY_DELETE_FILE系统表的更新由file_deletion_delay_min参数控制。

表 17-28 PG_DELAY_DELETE_FILE 字段

名称	类型	描述
databaseid	bigint	数据库OID。
tablespaceid	oid	表空间OID。
relfilenode	oid	表的relfilenode。
fileid	bigint	文件ID。
type	"char"	数据类型（定义了CUDESC、CU、DELTA及ROW，目前只统计CU类型）。
xid	xid	事务ID。
delete_time	bigint	延迟删除时间间隔。
reserved1	bigint	预留字段。
reserved2	bigint	预留字段。

17.2.28 PG_DEFAULT_ACL

PG_DEFAULT_ACL系统表存储为新建对象设置的初始权限。

 版权所有 © 华为云计算技术有限公司

表 17-29 PG_DEFAULT_ACL 字段

名称	类型	描述
defaclrole	oid	与此权限相关的角色ID。
defaclnamespace	oid	与此权限相关的命名空间，如果没有，则为0。
defaclobjtype	"char"	此权限的对象类型。 • r表示表或视图。 • S表示序列。 • f表示函数。 • T表示类型。
defaclacl	aclitem[]	创建该类型时所拥有的访问权限。

应用示例

查看新建用户role1的初始权限。

```
select * from PG_DEFAULT_ACL;
 defaclrole | defaclnamespace | defaclobjtype |    defaclacl
------------+-----------------+---------------+-----------------
      16820 |           16822 | r             | {role1=r/user1}
```

也可使用如下语句进行转换后更直观的查看。

```
SELECT pg_catalog.pg_get_userbyid(d.defaclrole) AS "Granter",  n.nspname AS "Schema",  CASE d.defaclobjtype WHEN 'r' THEN 'table' WHEN 'S' THEN 'sequence' WHEN 'f' THEN 'function' WHEN 'T' THEN 'type' END AS "Type",  pg_catalog.array_to_string(d.defaclacl, E', ') AS "Access privileges" FROM pg_catalog.pg_default_acl d LEFT JOIN pg_catalog.pg_namespace n ON n.oid = d.defaclnamespace ORDER BY 1, 2, 3;
```

输出结果如下，表示通过用户user1授予用户role1对模式“user1”有读的权限。

```
 Granter | Schema | Type  | Access privileges
---------+--------+-------+-------------------
 user1   | user1  | table | role1=r/user1
(1 row)
```

17.2.29 PG_DEPEND

PG_DEPEND系统表记录数据库对象之间的依赖关系。这些信息允许DROP命令找出哪些其他对象必须由DROP CASCADE删除，或者是在DROP RESTRICT的情况下避免删除。

另请参考**PG_SHDEPEND**，对于记录那些在数据库集群之间共享的对象之间的依赖性关系提供了相似的功能。

表 17-30 PG_DEPEND 字段

名称	类型	引用	描述
classid	oid	**PG_CLASS**.oid	依赖对象所在系统表的OID。
objid	oid	任意OID属性	指定依赖对象的OID。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
objsubid	integer	-	对于表字段，是该属性的字段数（objid和classid引用表本身）。对于所有其他对象类型，此字段是0。
refclassid	oid	PG_CLASS.oid	被引用对象所在的系统表的OID。
refobjid	oid	任意OID属性	指定的被引用对象的OID。
refobjsubid	integer	-	对于表字段，是该字段的字段号（refobjid和refclassid引用表本身）。对于所有其他对象类型，此字段是零。
deptype	"char"	-	定义此依赖关系特定语义的代码。

在所有情况下，一个PG_DEPEND记录表示被引用对象不能在没有删除依赖对象的情况下被删除。但是其中也有几种由deptype定义的情况：

- DEPENDENCY_NORMAL (n)：独立创建的对象之间的一般关系。依赖对象可以在不影响被引用对象的情况下删除。被引用对象只能通过指定CASCADE被删除，这种情况下依赖对象也会被删除。例如：一个表字段对其数据类型有一般依赖关系。
- DEPENDENCY_AUTO (a)：依赖对象可以和被引用对象分别删除，且在被引用对象被删除时应自动被删除（不管是RESTRICT或CASCADE模式）。例如：一个表上的命名约束是该表上的自动依赖关系，因此如果删除了表，它也会被删除。
- DEPENDENCY_INTERNAL (i)：依赖对象作为被引用对象过程的一部分创建，且是其内部实现的一部分。 DROP依赖对象是不会直接允许的（会给用户发出一个针对被引用对象的DROP）。不管是否指定CASCADE，一个被引用对象的DROP将被传播来删除其依赖对象。例如：一个用于强制外键约束的触发器将被设置为内部依赖于其约束的PG_CONSTRAINT项。
- DEPENDENCY_EXTENSION (e)：依赖对象作为被依赖对象extension的一个成员（请参见PG_EXTENSION）。依赖对象可以通过在被依赖对象上DROP EXTENSION删除。在功能上，这种依赖类型和内部依赖的作用相同，其存在只是为了清晰和简化gs_dump。
- DEPENDENCY_PIN (p)：没有依赖对象。这种类型的记录标志着系统本身依赖于被引用对象，因此这个对象绝不能被删除。这种类型的记录只有在initdb的时候创建。有依赖对象的字段都为0。

应用示例

查询名为serial1的数据库对象sequence和哪个表有依赖关系。

1. 先通过系统表PG_CLASS查询序列名为serial1的oid。

```
SELECT oid FROM pg_class WHERE relname ='serial1';
 oid
-------
 17815
(1 row)
```

2. 使用系统表PG_DEPEND根据所查询的序列serial1的oid获取依赖该序列的对象。

```
SELECT * FROM pg_depend WHERE objid ='17815';
 classid | objid | objsubid | refclassid | refobjid | refobjsubid | deptype
---------+-------+----------+------------+----------+-------------+---------
```


版权所有 © 华为云计算技术有限公司

```
   1259 | 17815 |       0 |      2615 |    2200 |          0 | n
   1259 | 17815 |       0 |      1259 |   17812 |          1 | a
(2 rows)
```

3. 根据字段refobjid获取依赖该序列serial1的表的OID，并查询到表名。其结果表示，序列serial1依赖于表customer_address。

```
SELECT relname FROM pg_class where oid='17812';
    relname
------------------
 customer_address
(1 row)
```

17.2.30 PG_DESCRIPTION

PG_DESCRIPTION系统表可以给每个数据库对象存储一个可选的描述（注释）。许多内置的系统对象的描述提供了PG_DESCRIPTION的初始内容。

这个表的功能类似**PG_SHDESCRIPTION**，用于记录整个集群范围内共享对象的注释。

表 17-31 PG_DESCRIPTION 字段

名称	类型	引用	描述
objoid	oid	任意OID属性	描述所属对象的OID。
classoid	oid	**PG_CLASS**.oid	对象显示的系统表的OID。
objsubid	integer	-	对于一个表字段的注释，为字段号（objoid和classoid指向表自身）。对于其他对象类型，为0。
description	text	-	对该对象描述的任意文本。

17.2.31 PG_ENUM

PG_ENUM系统表包含显示每个枚举类型值和标签的记录。给定枚举类型的内部表示实际上是PG_ENUM里面相关行的OID。

表 17-32 PG_ENUM 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
enumtypid	oid	**PG_TYPE**.oid	包含此枚举值的pg_type项的OID。
enumsortorder	real	-	此枚举值在其枚举类型中的排序位置。
enumlabel	name	-	此枚举值的文本标签。

PG_ENUM行的OID值遵循一种特殊规则：OID的数值被保证按照其枚举类型一样的排序顺序排序。即如果两个偶数OID属于同一枚举类型，那么较小的OID必然具有较小

 版权所有 © 华为云计算技术有限公司

enumsortorder值。奇数OID不需要遵循排序顺序。这种规则使得枚举比较例程在很多常见情况下可以避免系统目录查找。创建和修改枚举类型的例程尝试尽可能地为枚举值分配偶数OID。

当一个枚举类型被创建后，其成员会被分配排序顺序位置1到n。但是后面增加的成员可能会分配负值或者分数值的enumsortorder。对于这些值的唯一要求是它们必须被正确的排序且在每个枚举类型中保持唯一。

17.2.32 PG_EXCEPT_RULE

PG_EXCEPT_RULE系统表存储关于异常规则的信息。一个异常规则集合由多个名称相同的异常规则组成。

表 17-33 PG_EXCEPT_RULE

名称	类型	描述
name	name	异常规则集合的名称。
rule	name	该异常规则集中某一个具体规则类型或者触发当前异常规则集时采取的操作。（如blocktime/elapsedtime/spillsize等类型或者触发异常规则后的操作）
value	name	该异常规则对应的规则阈值。如果是触发异常规则后的操作，那么该字段为abort。

17.2.33 PG_EXTERNAL_NAMESPACE

存储EXTERNAL SCHEMA相关的信息。该系统表仅8.3.0及以上版本支持。

表 17-34 PG_EXTERNAL_NAMESPACE 字段

名字	类型	描述
nspid	Oid	External Schema Oid。
srvname	text	外部服务器名称。
source	text	元数据服务类型。
address	text	元数据服务地址。
database	text	元数据服务器数据库。
confpath	text	元数据服务器配置文件路径。
ensoptions	text[]	保留字段，暂时为空。
catalog	text	元数据服务器catalog。

 版权所有 © 华为云计算技术有限公司

示例

查询创建的EXTERNAL SCHEMA ex1：

```
SELECT * FROM pg_external_namespace WHERE nspid = (SELECT oid FROM pg_namespace WHERE
nspname = 'ex1');
```

17.2.34 PG_EXTENSION

PG_EXTENSION系统表存储关于所安装扩展的信息。DWS默认有三十九个扩展，即aio_scheduler、btree_gin、cudesckv、dimsearch、dist_fdw、functional_clog、functional_extension、functional_file、functional_hudi、functional_job、functional_largeobject、functional_memory、functional_other、functional_signal、functional_vacuum、functional_iceberg、gc_fdw、hdfs_fdw、hstore、log_fdw、operational_backup、operational_cgroup、operational_cudesc、operational_other、operational_replication、operational_restoration、operational_stats、operational_xlog、operational_oscache、packages、pgcrypto、pldbgapi、plpgsql、roach_api、tsdb、uuid-ossp、kvstore、pg_buffercache和plpython3u。

表 17-35 PG_EXTENSION

名称	类型	描述
extname	name	扩展名。
extowner	oid	扩展的所有者。
extnamespace	oid	扩展导出对象的命名空间。
extrelocatable	boolean	如果扩展能够重定位到其他schema，则为true。
extversion	text	扩展的版本号。
extconfig	oid[]	扩展的配置信息。
extcondition	text[]	扩展配置信息的过滤条件。

查看当前集群所安装的扩展信息。

```
SELECT * FROM PG_EXTENSION;
        extname         | extowner | extnamespace | extrelocatable | extversion | extconfig | extcondition
------------------------+----------+--------------+----------------+------------+-----------+--------------
 plpgsql                |       10 |           11 | f              | 1.0        |           |
 dist_fdw               |       10 |           11 | t              | 1.0        |           |
 roach_api              |       10 |           11 | t              | 1.0        |           |
 hdfs_fdw               |       10 |           11 | t              | 1.0        |           |
 btree_gin              |       10 |           11 | t              | 1.4        |           |
 gc_fdw                 |       10 |           11 | t              | 1.0        |           |
 log_fdw                |       10 |           11 | t              | 1.0        |           |
 hstore                 |       10 |           11 | t              | 1.1        |           |
 aio_scheduler          |       10 |           11 | t              | 1.0        |           |
 packages               |       10 |           11 | t              | 1.1        |           |
 pldbgapi               |       10 |           11 | t              | 1.0        |           |
 tsdb                   |       10 |           11 | t              | 1.0        |           |
 dimsearch              |       10 |           11 | t              | 1.0        |           |
 uuid-ossp              |       10 |           11 | t              | 1.0        |           |
 pgcrypto               |       10 |           11 | t              | 1.0        |           |
```


版权所有 © 华为云计算技术有限公司

```
operational_xlog        |       10 |          11 | t            | 1.0       |          |
operational_cgroup      |       10 |          11 | t            | 1.0       |          |
operational_backup      |       10 |          11 | t            | 1.0       |          |
operational_cudesc      |       10 |          11 | t            | 1.0       |          |
operational_replication |       10 |          11 | t            | 1.0       |          |
operational_restoration |       10 |          11 | t            | 1.0       |          |
operational_stats       |       10 |          11 | t            | 1.0       |          |
operational_other       |       10 |          11 | t            | 1.0       |          |
operational_oscache     |       10 |          11 | t            | 1.0       |          |
functional_clog         |       10 |          11 | t            | 1.0       |          |
functional_file         |       10 |          11 | t            | 1.0       |          |
functional_hudi         |       10 |          11 | t            | 1.0       |          |
functional_extension    |       10 |          11 | t            | 1.0       |          |
functional_job          |       10 |          11 | t            | 1.0       |          |
functional_largeobject  |       10 |          11 | t            | 1.0       |          |
functional_signal       |       10 |          11 | t            | 1.0       |          |
functional_vacuum       |       10 |          11 | t            | 1.0       |          |
functional_memory       |       10 |          11 | t            | 1.0       |          |
functional_iceberg      |       10 |          11 | t            | 1.0       |          |
functional_other        |       10 |          11 | t            | 1.0       |          |
cudesckv                |       10 |          11 | t            | 1.0       |          |
kvstore                 |       10 |          11 | t            | 1.0       |          |
pg_buffercache          |       10 |          11 | t            | 1.0       |          |
plpython3u              |       10 |          11 | f            | 1.0       |          |
(39 rows)
```

17.2.35 PG_EXTENSION_DATA_SOURCE

PG_EXTENSION_DATA_SOURCE系统表存储外部数据源对象的信息。一个外部数据源对象（Data Source）包含了外部数据库的一些口令编码等信息，主要配合Extension Connector使用。

表 17-36 PG_EXTENSION_DATA_SOURCE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
srcname	name	-	外部数据源对象的名称。
srcowner	oid	PG_AUTHID.oid	外部数据源对象的所有者。
srctype	text	-	外部数据源对象的类型，缺省为空。
srcversion	text	-	外部数据源对象的版本，缺省为空。
srcacl	aclitem[]	-	访问权限。
srcoptions	text[]	-	外部数据源对象的指定选项，使用“keyword=value”格式的字符串。

17.2.36 PG_FILE_RECYCLEBIN

PG_FILE_RECYCLEBIN系统表用于记录回收站中的对象信息，包括主表、辅助表的详细信息，同时记录对象的状态和类型。非重分布业务场景下，回收站中的记录信息过期后至少保留1天。

该系统表仅9.1.0.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-37 PG_FILE_RECYCLEBIN 字段

名字	类型	描述
oid	oid	行标识符。
baserelid	oid	基表对象ID或者父对象.oid（parentrelid）。
relid	oid	表OID，如果是基表与baserelid一致。
groupoid	oid	组ID。
relnamespace	oid	表所在schema。
spcnode	oid	表空间。
dbnode	oid	数据库。
relnode	oid	磁盘上的文件的名称，如果没有则为0。
ownerid	oid	用户ID。
forknumber	integer	文件编号。
status	integer	回收站中对象状态： ● TTL，表示进入回收站默认值。 ● DELETING，表示清理中。 ● DONE，表示已清理。 ● FLASHBACK，表示已闪回。
objtype	integer	对象类型Cudesc，delta，toast，preagg，part，cufile，obspath等不同类型有不同的清理方式。
rcyoperation	char	操作类型。
file_id	bigint	CU文件ID。
ttl	timestamp with time zone	回收站对象到期清理时间。
record_time	timestamp with time zone	加入回收站的时间。
done_time	timestamp with time zone	回收站清理文件的时间。
latency	integer	回收站清理耗时。
filepath	text	物理文件路径。
objname	*name*	对象原始名称。
subobjname	*name*	为分区表时，表示原始分区名。
rbname	*name*	回收站中的uniquenameTb_36uuidView_36uuid。

 版权所有 © 华为云计算技术有限公司

名字	类型	描述
options	text[]	基表的options。
relfrozenxid	xid32	冻结事务ID号。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息
relfrozenxid64	xid	冻结事务ID号。
relmetaversion	xid32	元数据版本信息。
canflashback	boolean	是否可以闪回。

17.2.37 PG_FINE_DR_INFO

PG_FINE_DR_INFO系统表用于记录细粒度容灾备表的回放状态。该系统表仅8.2.0.100及以上集群版本支持。

表 17-38 PG_FINE_DR_INFO 字段

名字	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
relid	oid	细粒度容灾备表的OID。
lastcsn	xid	上一次成功回放时的csn。
lastxmin	xid	上一次成功回放时的xmin。
lastxmax	xid	上一次成功回放时的xmax。
laststarttime	timestamp with time zone	上一次成功回放的开始时间。
lastendtime	timestamp with time zone	上一次成功回放的结束时间。

应用示例

在容灾集群上查看备表回放状态：

```
SELECT * FROM pg_fine_dr_info;
 relid | lastcsn | lastxmin | lastxmax |         laststarttime         |          lastendtime
-------+---------+----------+----------+-------------------------------+-------------------------------
 21132 | 1251610 |  1251609 |  1251611 | 2023-01-04 20:51:58.375136+08 | 2023-01-04 20:51:58.393986+08
(1 row)
```

17.2.38 PG_FOREIGN_DATA_WRAPPER

PG_FOREIGN_DATA_WRAPPER系统表存储外部数据封装器定义。一个外部数据封装器是在外部服务器上驻留外部数据的机制，是可以访问的。

 版权所有 © 华为云计算技术有限公司

表 17-39 PG_FOREIGN_DATA_WRAPPER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
fdwname	name	-	外部数据封装器名。
fdwowner	oid	PG_AUTHID.oid	外部数据封装器的所有者。
fdwhandler	oid	PG_PROC.oid	引用一个负责为外部数据封装器提供扩展例程的处理函数。如果没有提供处理函数则为0。
fdwvalidator	oid	PG_PROC.oid	引用一个验证器函数，这个验证器函数负责验证给予外部数据封装器的选项、外部服务器选项和使用外部数据封装器的用户映射的有效性。如果没有提供验证器函数则为0。
fdwacl	aclitem[]	-	访问权限。
fdwoptions	text[]	-	外部数据封装器指定选项，使用“keyword=value”格式的字符串。

17.2.39 PG_FOREIGN_SERVER

PG_FOREIGN_SERVER系统表存储外部服务器定义。一个外部服务器描述了一个外部数据源，例如一个远程服务器。外部服务器通过外部数据封装器访问。

表 17-40 PG_FOREIGN_SERVER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
srvname	name	-	外部服务器名。
srvowner	oid	PG_AUTHID.oid	外部服务器的所有者。
srvfdw	oid	PG_FOREIGN_DATA_WRAPPER.oid	此外部服务器的外部数据封装器的OID。
srvtype	text	-	服务器的类型（可选）。
srvversion	text	-	服务器的版本（可选）。
srvacl	aclitem[]	-	访问权限。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
srvoptions	text[]	-	外部服务器指定选项，使用“keyword=value”格式的字符串。

17.2.40 PG_FOREIGN_TABLE

PG_FOREIGN_TABLE系统表存储外部表的辅助信息。

表 17-41 PG_FOREIGN_TABLE 字段

名称	类型	描述
ftrelid	oid	外部表的OID。
ftserver	oid	外部表的所在服务器的OID。
ftwriteonly	boolean	外部表是否可写。
ftoptions	text[]	外部表的可选项。
ftreadwriteopts	smallint	外部表的读写属性。 • 1，表示只读外表。 • 2，表示只写外表。 • 3，表示读写外表。

17.2.41 PG_INDEX

PG_INDEX系统表存储索引的一部分信息，其他的信息大多数在PG_CLASS中。

表 17-42 PG_INDEX 字段

名称	类型	描述
indexrelid	oid	此索引的pg_class项的OID。
indrelid	oid	使用该索引的表在pg_class项的OID。
indnatts	smallint	索引中的字段数目。
indisunique	boolean	如果为真，为唯一索引。
indisprimary	boolean	如果为真，该索引为该表的主键。该字段为真时，indisunique也总是为真。
indisexclusion	boolean	如果为真，该索引支持排他约束。
indimmediate	boolean	如果为真，在插入数据时会立即执行唯一性检查。
indisclustered	boolean	如果为真，则该表最后以此索引进行了聚簇。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
indisusable	boolean	如果为真，此索引对INSERT/SELECT可用。
indisvalid	boolean	如果为真，则此索引可以用于查询。如果为假，则该索引可能不完整，仍然必须在INSERT/UPDATE操作时进行更新，但不能安全的被用于查询。如果是唯一索引，则唯一属性也不为真。
indcheckxmin	boolean	如果为真，查询不能使用此索引，直到pg_index此行的xmin低于其快照的TransactionXmin，因为该表可能包含它们可见的不兼容行断开的HOT链。
indisready	boolean	如果为真，表示此索引对插入数据可用。如果为假，在插入或修改数据时忽略此索引。
indkey	int2vector	这是一个包含indnatts值的数组，这些数组值表示此索引所建立的表字段。比如一个值为1 3的意思是第一个字段和第三个字段组成这个索引键。数组中的0表示对应的索引属性是一个表字段上的表达式，而不是一个简单的字段引用。
indcollation	oidvector	索引用到的各列的ID。
indclass	oidvector	对于索引键中的每个字段，该字段都包含要使用的操作符类的OID，详见**PG_OPCLASS**。
indoption	int2vector	存储列前标识位，该标识位是由索引的访问方法定义。
indexprs	pg_node_tree	表达式树（以nodeToString()形式表现）用于那些非简单字段引用的索引属性。它是一个列表，个数与INDKEY中的零值个数相同。如果所有索引属性都是简单的引用，则为空。
indpred	pg_node_tree	部分索引谓词的表达式树（以nodeToString()的形式表现）。如果不是部分索引，则为空。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
indnullstreatment	tinyint	表示唯一索引中NULL值的处理方式，该字段只有当indisunique为真时才起作用。 取值范围： ● 0 表示NULLS DISTINCT，NULL互不相等，NULL值可重复插入。 ● 1 表示NULLS NOT DISTINCT，NULL严格相等，NULL值不可重复插入。 ● 2 表示NULLS IGNORE，在等值比较时忽略NULL值列。若索引列全为NULL，则NULL值可重复插入；若部分索引列为NULL，只有非NULL列不相等，才可插入。 默认值：0 **说明** ● 若当前集群为低版本升级到8.2.0.100版本，对于之前已存在索引，该字段值均为NULL；对于新创建的索引，则根据[NULLS [NOT] DISTINCT | NULLS IGNORE]字段的指定情况确定该字段值，默认值为0。 ● 若当前集群为新装的8.2.0.100版本，对于新创建的索引，则根据[NULLS [NOT] DISTINCT | NULLS IGNORE]字段的指定情况确定该字段值，默认值为0。

17.2.42 PG_INHERITS

PG_INHERITS系统表记录关于表继承层次的信息。数据库里每个直接的子系表都有一条记录。间接的继承可以通过追溯记录链来判断。

表 17-43 PG_INHERITS 字段

名字	类型	引用	描述
inhrelid	oid	PG_CLASS.oid	子表的OID。
inhparent	oid	PG_CLASS.oid	父表的OID。
inhseqno	integer	-	如果一个子表存在多个直系父表（多重继承），这个数字表示此继承字段的排列顺序。计数从1开始。

17.2.43 PG_JOB_INFO

PG_JOB_INFO系统表记录定时任务的执行结果信息。该系统表的schema是dbms_om。

 版权所有 © 华为云计算技术有限公司

表 17-44 dbms_om.pg_job_info 字段

名称	类型	描述
job_id	integer	任务ID。
job_db	oid	任务所在数据库OID。
start_time	timestamp with zone	任务执行开始时间。
status	character(8)	任务执行状态。
end_time	timestamp with zone	任务执行结束时间。
err_msg	text	任务执行出错信息。

17.2.44 PG_JOBS

PG_JOBS系统表存储用户创建的定时任务的任务详细信息，定时任务线程定时轮询pg_jobs系统表中的时间，当任务到期会触发任务的执行。该系统表属于Shared Relation，所有创建的job记录对所有数据库可见。

表 17-45 PG_JOBS 字段

名字	类型	描述
job_id	integer	作业ID，主键，是唯一的（有唯一索引）。
what	text	作业内容。
log_user	oid	创建者的UserID。
priv_user	oid	作业执行者的UserID。
job_db	oid	标识作业执行的数据库OID。
job_nsp	oid	标识作业运行时所在的命名空间OID。
job_node	oid	标识当前作业是在哪个CN上创建和执行。
is_broken	boolean	标识当前作业是否为失效状态。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。

17.2.45 PG_LANGUAGE

PG_LANGUAGE系统表记录了可用于编写函数或存储过程的语言。

表 17-46 PG_LANGUAGE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
lanname	name	-	语言的名称。
lanowner	oid	PG_AUTHID.oid	语言的所有者。
lanispl	boolean	-	内部语言为假（比如SQL），用户定义语言为真。目前，gs_dump仍然使用该字段判断哪些语言需要转储，但可能在将来会被其他机制所取代。
lanpltrusted	boolean	-	如果是可信语言则为真，即系统相信它不会被授予任何正常SQL执行环境之外的权限。只有初始用户可以在用非可信的语言创建函数。
lanplcallfoid	oid	PG_PROC.oid	对于非内部语言，这是指该语言处理器的引用，语言处理器是一个特殊函数，负责执行以某种语言写的所有函数。
laninline	oid	PG_PROC.oid	此字段引用一个负责执行“inline”匿名代码块的函数（DO块）。如果不支持内联块则为0。
lanvalidator	oid	PG_PROC.oid	此字段引用一个语言校验器函数，它负责检查新创建函数的语法和有效性。如果没有提供校验器，则为0。
lanacl	aclitem[]	-	访问权限。

版权所有 © 华为云计算技术有限公司

17.2.46 PG_LARGEOBJECT

PG_LARGEOBJECT系统表保存那些标记着“大对象”的数据。一个大对象是使用其创建时分配的OID标识的。每个大对象都分解成足够小的小段或者“页面”以便以行的形式存储在PG_LARGEOBJECT里。每页的数据定义为LOBLKSIZE。

需要有系统管理员权限才可以访问此系统表。

表 17-47 PG_LARGEOBJECT 字段

名字	类型	引用	描述
loid	oid	PG_LARGEOBJECT_METADATA.oid	包含本页的大对象的标识符。
pageno	integer	-	本页在其大对象数据中的页码，从0开始计算。
data	bytea	-	实际存储在大对象中的数据。这些数据不会超过LOBLKSIZE字节，且可能更小。

PG_LARGEOBJECT的每一行保存一个大对象的一个页面，从该对象内部的字节偏移（pageno * LOBLKSIZE）开始。这种实现允许稀疏存储：页面可能丢失，并且可以比LOBLKSIZE字节少（即使它们不是对象的最后一页）。大对象中丢失的区域会被读为0。

17.2.47 PG_LARGEOBJECT_METADATA

PG_LARGEOBJECT_METADATA系统表存储与大数据相关的元数据。实际的大对象数据存储在PG_LARGEOBJECT里。

表 17-48 PG_LARGEOBJECT_METADATA 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
lomowner	oid	PG_AUTHID.oid	大对象的所有者。
lomacl	aclitem[]	-	访问权限。

17.2.48 PG_MATVIEW

PG_MATVIEW系统表提供获取当前节点的物化视图信息。

版权所有 © 华为云计算技术有限公司

表 17-49 PG_MATVIEW 字段

字段名称	字段类型	描述
mvid	oid	物化视图OID。
build_mode	char	物化视图的build模式。 ● 'd'：代表deferred，表示创建物化视图时需要等到第一次refresh时才会包含数据。 ● 'i'：代表immediate，表示创建物化视图时即包含最新数据。
refresh_method	char	物化视图的刷新方法。 'c' ： 代表COMPLETE，表示完全刷新。
refresh_mode	char	物化视图的刷新模式。 'd'：代表demand，表示按需刷新。
rewrite_enable	boolean	是否支持物化视图的查询重写。
active	boolean	物化视图是否需要刷新。
relnum	Int	物化视图基表个数。
start_time	timestamptz	物化视图第一次定时刷新的时间，为空则第一次刷新时间是当前时间+interval。
interval	interval	物化视图定时刷新时间的间隔。
refresh_time	timestamptz	物化视图的最后一次刷新时间。
refresh_finish_time	timestamptz	物化视图的最后一次刷新结束时间。

17.2.49 PG_MATVIEW_DEPEND

PG_MATVIEW_DEPEND系统表用于查询物化视图包含的基表信息和基表对应的物化视图信息。该视图仅8.2.1.100及以上集群版本支持。

表 17-50 PG_MATVIEW_DEPEND 字段

字段名称	字段类型	描述
matviewid	oid	物化视图OID。
relid	oid	物化视图基表OID。
partids	oidvector	指定分区时候的基表分区OID。
contain_entire_rel	boolean	是否包含整个基表。
matpartid	oid	映射的物化视图的分区OID。

 版权所有 © 华为云计算技术有限公司

字段名称	字段类型	描述
relpartid	oid	映射的基表的分区OID。

17.2.50 PG_MATVIEW_PARTITION

PG_MATVIEW_PARTITION系统表存储分区物化视图每个分区的数据信息。该系统表仅9.1.0.200及以上集群版本支持。

表 17-51 PG_MATVIEW_PARTITION 字段

名称	类型	描述
matpartid	oid	物化视图分区的OID。
active	boolean	分区数据是否为最新数据。 t，表示分区物化视图的数据为最新状态。 f，表示分区物化视图的数据不是最新状态。
refresh_start_time	timestamp with time zone	最后一次刷新的开始时间。
refresh_finish_time	timestamp with time zone	最后一次刷新的结束时间。

17.2.51 PG_NAMESPACE

PG_NAMESPACE系统表存储命名空间，即存储schema相关的信息。

表 17-52 PG_NAMESPACE 字段

名称	类型	描述
nspname	name	命名空间的名称。
nspowner	oid	命名空间的所有者。
nsptimeline	bigint	在DN上创建此命名空间时的时间线。此字段为内部使用，仅在DN上有效。
nspacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。
permspace	bigint	schema永久表空间限额，单位Byte。
usedspace	bigint	schema已用永久表空间大小，单位Byte。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
nsptype	char	区分external schema和普通schema。该参数仅8.3.0及以上版本支持，适配Lakeformation特性。 **说明** 新增nsptype字段，用于区分external schema和普通schema。 • external schema对应的值为“e” • 普通schema对应的值为“i”。

17.2.52 PG_OBJECT

PG_OBJECT系统表存储限定类型对象(object_type中存在的类型)的创建用户、创建时间、最后修改时间和最后analyze时间。

表 17-53 PG_OBJECT 字段

名称	类型	描述
object_oid	oid	对象标识符。
object_type	"char"	对象类型： • r表示表，包括普通表和临时表 • i表示索引 • s表示序列 • v表示视图 • p表示存储过程和函数 • f表示外表
creator	oid	创建用户的标识符。
ctime	timestamp with time zone	对象创建时间。
mtime	timestamp with time zone	对象最后修改时间，默认记录修改行为包括ALTER操作、COMMENT、GRANT/REVOKE和TRUNCATE操作。 **object_mtime_record_mode**参数可以细粒度控制ALTER、COMMENT、GRANT/REVOKE和TRUNCATE操作是否被记录。
last_analyze_time	timestamp with time zone	对象进行最后一次analyze的时间。

 版权所有 © 华为云计算技术有限公司

须知

- 仅针对用户正常操作行为进行记录，无法记录对象升级以前和initdb过程中的行为。
- ctime和mtime的时间记录为本次操作的事务起始时间。
- 由扩容引起的对象修改时间也会被记录。

17.2.53 PG_OBSSCANINFO

PG_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 17-54 PG_OBSSCANINFO 字段

名字	类型	引用	描述
query_id	bigint	-	查询标识。
user_id	text	-	执行该查询的数据库用户。
table_name	text	-	OBS外表的表名。
file_type	text	-	底层数据保存的文件格式。
time_stamp	timestamp with time zone	-	扫描操作开始的时间。
actual_time	double	-	扫描操作执行时间，单位为秒。
file_scanned	bigint	-	扫描的文件数量。
data_size	double	-	扫描的数据量，单位为字节。
billing_info	text	-	保留字段。

17.2.54 PG_OPCLASS

PG_OPCLASS系统表定义索引访问方法操作符类。

每个操作符类为一种特定数据类型和一种特定索引访问方法定义索引字段的语义。一个操作符类本质上指定一个特定的操作符族适用于一个特定的可索引的字段数据类型。索引的字段实际可用的族中的操作符集是接受字段的数据类型作为它们的左边的输入的那个。

表 17-55 PG_OPCLASS 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
opcmethod	oid	PG_AM.oid	操作符类所属的索引访问方法。
opcname	name	-	操作符类的名称。
opcnamespace	oid	PG_NAMESPACE.oid	操作符类所属的命名空间。
opcowner	oid	PG_AUTHID.oid	操作符类所有者。
opcfamily	oid	PG_OPFAMILY.oid	包含此操作符类的操作符族。
opcintype	oid	PG_TYPE.oid	操作符类索引的数据类型。
opcdefault	boolean	-	如果操作符类是opcintype的缺省，则为真。
opckeytype	oid	PG_TYPE.oid	索引数据的类型，如果和opcintype相同则为0。

一个操作符类的opcmethod必须匹配包含它的操作符族的opfmethod。同样，对于任意给定的opcmethod和opcintype的组合，不能有超过一个PG_OPCLASS行有opcdefault为真。

17.2.55 PG_OPERATOR

PG_OPERATOR系统表存储有关操作符的信息。

表 17-56 PG_OPERATOR 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
oprname	name	-	操作符的名称。
oprnamespace	oid	PG_NAMESPACE.oid	包含此操作符的命名空间的OID。
oprowner	oid	PG_AUTHID.oid	操作符所有者。
oprkind	"char"	-	• b=infix =中缀（两边） • l=前缀（左边） • r=后缀（右边）
oprcanmerge	boolean	-	该操作符是否支持合并连接。
oprcanhash	boolean	-	该操作符是否支持Hash连接。

 版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
oprleft	oid	PG_TYPE.oid	左操作数的类型。
oprright	oid	PG_TYPE.oid	右操作数的类型。
oprresult	oid	PG_TYPE.oid	结果类型。
oprcom	oid	PG_OPERATOR.oid	此操作符的交换符（如果存在）。
oprnegate	oid	PG_OPERATOR.oid	此操作符的反转器（如果存在）。
oprcode	regproc	PG_PROC.oid	实现该操作符的函数。
oprrest	regproc	PG_PROC.oid	此操作符的约束选择性计算函数。
oprjoin	regproc	PG_PROC.oid	此操作符的连接选择性计算函数。

17.2.56 PG_OPFAMILY

PG_OPFAMILY系统表定义操作符族。

每个操作符族是操作符和相关支持例程的集合，这些例程实现了为特定索引访问方法指定的语义。此外，按照访问方法指定的某种方式，一个族内的操作符都是“兼容的”。操作符族允许跨数据类型操作符与索引一起使用，并可以推理使用访问方法语义相关内容。

表 17-57 PG_OPFAMILY 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
opfmethod	oid	PG_AM.oid	操作符族使用的索引方法。
opfname	name	-	操作符族的名称。
opfnamespace	oid	PG_NAMESPACE.oid	操作符族的命名空间。
opfowner	oid	PG_AUTHID.oid	操作符族的所有者。

定义一个操作符族的大多数信息不在PG_OPFAMILY，而是在相关的PG_AMOP，PG_AMPROC和PG_OPCLASS中。

版权所有 © 华为云计算技术有限公司

17.2.57 PG_PARTITION

PG_PARTITION系统表存储数据库内所有分区表(partitioned table)、分区(table partition)、分区上toast表和分区索引(index partition)四类对象的信息。分区表索引(partitioned index)的信息不在PG_PARTITION系统表中保存。

表 17-58 PG_PARTITION 字段

名称	类型	描述
relname	name	分区表、分区、分区上toast表和分区索引的名称。
parttype	"char"	对象类型： ● 'r'：partitioned table ● 'p'：table partition ● 'x'：index partition ● 't'：toast table
parentid	oid	当对象为分区表或分区时，此字段表示分区表在PG_CLASS中的OID。 当对象为index partition时，此字段表示所属分区表索引(partitioned index)的OID。
rangenum	integer	保留字段。
intervalnum	integer	保留字段。
partstrategy	"char"	分区表分区策略，现仅支持： 'r'：范围分区。 'v'：数值分区。 'l'：列表分区。
relfilenode	oid	table partition、index partition、分区上toast表的物理存储位置。
reltablespace	oid	table partition、index partition、分区上toast表所属表空间的OID。
relpages	double precision	统计信息：table partition、index partition的数据页数量。
reltuples	double precision	统计信息：table partition、index partition的元组数。
relallvisible	integer	统计信息：table partition、index partition的可见数据页数。
reltoastrelid	oid	table partition所对应toast表的OID。
reltoastidxid	oid	table partition所对应toast表的索引的OID。
indextblid	oid	index partition对应table partition的OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
indisusable	boolean	分区索引是否可用。
reldeltarelid	oid	Delta表的OID。
reldeltaidx	oid	Delta表的索引表的OID。
relcudescrelid	oid	CU描述表的OID。
relcudescidx	oid	CU描述表的索引表的OID。
relfrozenxid	xid32	冻结事务ID号。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
intspnum	integer	间隔分区所属表空间的个数。
partkey	int2vector	分区键的列号。
intervaltablespace	oidvector	间隔分区所属的表空间，间隔分区以round-robin方式落在这些表空间内。
interval	text[]	间隔分区的间隔值。
boundaries	text[]	范围分区和间隔分区的上边界。
transit	text[]	间隔分区的跳转点。
reloptions	text[]	设置partition的存储属性，与pg_class.reloptions的形态一样，用"keyword=value"格式的字符串来表示 ，目前用于在线扩容的信息搜集。
relfrozenxid64	xid	冻结事务ID号。
boundexprs	pg_node_tree	分区边界表达式。 ● 对于范围分区来说是分区上边界表达式。 ● 对于列表分区来说是分区边界枚举值集合。 pg_node_tree数据类型是不可读的，可用如下表达式pg_get_expr把当前字段转换为可读信息。 SELECT pg_get_expr(boundexprs, 0) FROM pg_partition WHERE relname = 'country_202201'; pg_get_expr --- ROW(202201, 'city1'::text), ROW(202201, 'city2'::text) (1 row)
relmetaversion	xid	元数据版本信息，该字段仅9.1.0及以上集群版本支持。

应用示例

查询分区表web_returns_p2的分区信息。

```
CREATE TABLE web_returns_p2
(
```

 版权所有 © 华为云计算技术有限公司

```
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE(wr_returned_date_sk)
(
    PARTITION p2016 START(20161231) END(20191231) EVERY(10000),
    PARTITION p0 END(maxvalue)
);

SELECT oid FROM pg_class WHERE relname ='web_returns_p2';
  oid
-------
 97628

SELECT relname,parttype,parentid,boundaries FROM pg_partition WHERE parentid = '97628';
    relname     | parttype | parentid | boundaries
----------------+----------+----------+------------
 web_returns_p2 | r        |    97628 |
 p2016_0        | p        |    97628 | {20161231}
 p2016_1        | p        |    97628 | {20171231}
 p2016_2        | p        |    97628 | {20181231}
 p2016_3        | p        |    97628 | {20191231}
 p0             | p        |    97628 | {NULL}
(6 rows)
```

17.2.58 PG_PLAN_BASELINE

PG_PLAN_BASELINE系统表记录语句和计划的绑定关系。一个语句只可以绑定一个计划。

表 17-59 PG_PLAN_BASELINE 字段

名称	类型	描述
sql_hash	text	语句的sql_hash。
outline_name	text	该sql_hash对应的sql语句绑定的outline的名字。
plan_baseline	name	该绑定关系的baseline名字。
create_time	timestamp with time zone	计划绑定创建时间。
last_modify_time	timestamp with time zone	最后一次修改时间，即为该语句最后一次绑定计划的时间。
last_execute_time	timestamp with time zone	最后一次被使用时间。
apply_skew_hint	boolean	是否应用outline中的skew hint，该字段9.1.0及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
status	tinyint	绑定状态。该字段9.1.1.200及以上集群版本支持。取值如下： ● 0：preferred，表示优先。 ● 1：approved，表示可用。 ● 2：unapproved，表示不推荐。 ● 3：rejected，表示拒绝。
cost	float8	生成该outline的计划的成本（cost）。该字段9.1.1.200及以上集群版本支持。
normalize_cost	float8	根据TopSQL中该语句duration、cpu和memory使用情况，经过归一化计算的成本（cost）。保留列，暂未使用。该字段9.1.1.200及以上集群版本支持。

17.2.59 PG_PLTEMPLATE

PG_PLTEMPLATE系统表存储过程语言的“模板”信息。

表 17-60 PG_PLTEMPLATE 字段

名称	类型	描述
tmplname	name	该模板所应用的语言的名称。
tmpltrusted	boolean	如果语言被认为是可信的，则为真。
tmpldbacreate	boolean	如果语言是由数据库所有者创建的，则为真。
tmplhandler	text	调用处理器函数的名称。
tmplinline	text	匿名块处理器的名称，如果没有则为NULL。
tmplvalidator	text	校验函数的名称，如果没有则为NULL。
tmpllibrary	text	实现语言的共享库的路径。
tmplacl	aclitem[]	模板的访问权限（未使用）。

17.2.60 PG_PROC

PG_PROC系统表存储函数或过程的信息。

表 17-61 PG_PROC 字段

名称	类型	描述
proname	name	函数名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
pronamespace	oid	此函数所在命名空间的OID。
proowner	oid	函数的所有者。
prolang	oid	实现语言或函数的调用接口。
procost	real	估计执行成本。
prorows	real	结果行估计数。
provariadic	oid	参数元素的数据类型。
protransform	regproc	此函数的简化调用方式。
proisagg	boolean	函数是否为聚集函数。
proiswindow	boolean	函数是否为窗口函数。
prosecdef	boolean	函数是否为一个安全定义器（例如，一个“setuid”函数）。
proleakproof	boolean	函数有无其他影响。如果函数没有对参数进行防泄露处理，则会抛出错误。
proisstrict	boolean	如果任意调用参数为空，函数是否返回空值。这种情况下函数实际上根本不会被调用。非“strict”的函数必须准备处理空值输入。
proretset	boolean	函数是否返回一个集合（即，指定数据类型的多个数值）。
provolatile	"char"	说明该函数的结果是只依赖于它的输入参数，或者还会被外接因素影响。 ● i表示“不可变的”（immutable）函数，对于相同的输入总是输出相同的结果。 ● s表示“稳定的”（stable）函数，对于固定输入其结果在一次扫描里不变。 ● v表示“易变”（volatile）函数，其结果可能在任何时候都变化。
pronargs	smallint	参数个数。
pronargdefaults	smallint	有默认值的参数个数。
prorettype	oid	返回参数类型的OID。
proargtypes	oidvector	函数参数的数据类型的数组。数组里只包括输入参数（包括INOUT参数），因此也表现了函数的调用特征。
proallargtypes	oid[]	函数参数的数据类型的数组。数组里包括所有参数的类型（包括OUT和INOUT参数），如果所有参数都是IN参数，则这个字段就会为空。注意数组下标是以1为起点的，而因为历史原因，proargtypes的下标起点为0。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
proargmodes	"char"[]	函数参数模式的数组。 ● i表示IN参数。 ● o表示OUT参数。 ● b表示INOUT参数。 ● v表示VARIADIC参数。 ● t表示TABLE参数。 如果所有参数都是IN参数，则这个字段为空。注意此数组下标对应的是proallargtypes的位置，而不是proargtypes。
proargnames	text[]	函数参数的名字的数组。没有名字的参数在数组里设置为空字符串。如果没有一个参数有名字，这个字段为空。注意此数组的下标对应proallargtypes而不是proargtypes。
proargdefaults	pg_node_tree	默认值的表达式树。是PRONARGDEFAULTS元素的列表。
prosrc	text	描述函数或存储过程的定义。例如，对于解释型语言来说就是函数的源程序，或者一个链接符号，一个文件名，或者函数和存储过程创建时指定的其他任何函数体内容，具体取决于语言/调用习惯的实现。
probin	text	关于如何调用该函数的附加信息。同样，其含义也是和语言相关的。
proconfig	text[]	函数针对运行时配置变量的本地设置。
proacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。
prodefaultargpos	int2vector	当函数参数较多时，函数具有默认值的入参的位置。
fencedmode	boolean	函数的执行模式，表示函数是在fence还是not fence模式下执行。如果是fence执行模式，函数的执行会在重新fork的进程中执行。默认值是fence。
proshippable	boolean	函数是否可以下推到DN上执行，默认值是false。 ● 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。 ● 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。
propackage	boolean	该函数是否支持重载，主要针对Oracle风格的函数，默认值是false。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
prokind	"char"	标记是函数类型还是存储过程类型。该字段仅9.1.1.100及以上集群版本支持。 ● f，表示类型为函数（Function）。 ● p，表示类型为存储过程（Stored Procedure）。

应用示例

查询指定函数的OID。例如，获取函数justify_days的OID为1295。

```
SELECT oid FROM pg_proc where proname ='justify_days';
 oid
------
 1295
(1 row)
```

查询指定函数是否为聚集函数。例如，查询justify_days函数为非聚集函数。

```
SELECT proisagg FROM pg_proc where proname ='justify_days';
 proisagg
----------
 f
(1 row)
```

17.2.61 PG_PUBLICATION

PG_PUBLICATION系统表存储当前数据库中创建的所有发布。该系统表仅8.2.0.100及以上集群版本支持。

表 17-62 PG_PUBLICATION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
pubname	name	-	发布的名称。
pubowner	oid	PG_AUTHID.oid	发布的拥有者。
puballtables	boolean	-	如果为true，这个发布自动包括数据库中的所有表，包括未来将会创建的任何表。
pubinsert	boolean	-	如果为true，为发布中的表复制INSERT操作。
pubupdate	boolean	-	如果为true，为发布中的表复制UPDATE操作。
pubdelete	boolean	-	如果为true，为发布中的表复制DELETE操作。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
pubtruncate	boolean	-	如果为true，为发布中的表复制TRUNCATE操作。

应用示例

查看所有发布：

```
SELECT * FROM pg_publication;
 pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete | pubtruncate
---------+----------+--------------+-----------+-----------+-----------+-------------
 mypub   |       10 | t            | t         | t         | t         | t
(1 row)
```

17.2.62 PG_PUBLICATION_NAMESPACE

PG_PUBLICATION_NAMESPACE系统表存储当前数据库中的发布和模式之间的映射，这是一种多对多映射。该系统表仅8.2.0.100及以上集群版本支持。

表 17-63 PG_PUBLICATION_NAMESPACE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prpubid	oid	PG_PUBLICATION.oid	映射的发布的OID。
pnnspid	oid	PG_NAMESPACE.oid	映射的模式的OID。

应用示例

查看所有发布和模式的映射：

```
SELECT * FROM pg_publication_namespace;
 pnpubid | pnnspid
---------+---------
   16797 |   16796
(1 row)
```

17.2.63 PG_PUBLICATION_REL

PG_PUBLICATION_REL系统表存储当前数据库中的发布和表之间的映射，这是一种多对多映射。该系统表仅8.2.0.100及以上集群版本支持。

说明

查询时推荐使用视图PG_PUBLICATION_TABLES，可以展现更详细的信息。

版权所有 © 华为云计算技术有限公司

表 17-64 PG_PUBLICATION_REL 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prpubid	oid	PG_PUBLICATION.oid	映射的发布的OID。
prrelid	oid	PG_CLASS.oid	映射的表的OID。

应用示例

查看所有发布和表的映射：

```
postgres=# SELECT * FROM pg_publication_rel;
 prpubid | prrelid
---------+---------
   16797 |   16757
   16797 |   16776
(2 rows)
```

17.2.64 PG_RANGE

PG_RANGE系统表存储关于范围类型的信息。

除了PG_TYPE里类型的记录。

表 17-65 PG_RANGE 字段

名字	类型	引用	描述
rngtypid	oid	PG_TYPE.oid	范围类型的OID。
rngsubtype	oid	PG_TYPE.oid	该范围类型的元素类型（子类型）的OID。
rngcollation	oid	PG_COLLATION.oid	用于范围比较的排序规则的OID，如果没有则为0。
rngsubopc	oid	PG_OPCLASS.oid	用于范围比较的子类型的操作符类的OID。
rngcanonical	regproc	PG_PROC.oid	转换范围类型为规范格式的函数的OID，如果没有则为0。
rngsubdiff	regproc	PG_PROC.oid	返回两个double precision元素值的不同的函数的OID，如果没有则为0。

rngsubopc（如果元素类型是可排序的，则加上rngcollation）决定用于范围类型的排序顺序。当元素类型是rngcanonical时用于离散类型。

17.2.65 PG_REDACTION_COLUMN

PG_REDACTION_COLUMN系统表存储脱敏列的信息。

版权所有 © 华为云计算技术有限公司

表 17-66 PG_REDACTION_COLUMN 字段

名称	类型	描述
object_oid	oid	脱敏对象OID。
column_attrno	smallint	脱敏列attrno。
function_type	integer	脱敏类型。 **说明** 保留字段，仅为向前兼容低版本的脱敏列信息，可取值为0（NONE）、1（FULL）。
function_parameters	text	脱敏类型为partial类型时的参数。（保留字段，无实际意义）
regexp_pattern	text	脱敏类型为regexp时，格式化字符串。（保留字段，无实际意义）
regexp_replace_string	text	脱敏类型为regexp时，替换串。（保留字段，无实际意义）
regexp_position	integer	脱敏类型为regexp时，起始替换位置。（保留字段，无实际意义）
regexp_occurrence	integer	脱敏类型为regexp时，替换次数。（保留字段，无实际意义）
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数。（保留字段，无实际意义）
column_description	text	脱敏列描述信息。
function_expr	pg_node_tree	脱敏函数的内部表现形式。
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。
policy_oid	oid	所属脱敏策略OID。 该字段8.2.1.100及以上集群版本支持，用于查询时直接从系统表元数据检索脱敏列信息。

17.2.66 PG_REDACTION_POLICY

PG_REDACTION_POLICY系统表提供了脱敏对象的信息。

表 17-67 PG_REDACTION_POLICY 字段

名称	类型	描述
object_oid	oid	脱敏对象OID。
policy_name	name	脱敏策略名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
enable	boolean	策略状态（开启、关闭）。 **说明** enable的取值： • true表示开启 • false表示关闭
expression	pg_node_tree	策略生效表达式（针对用户）。
policy_description	text	策略描述信息。
inherited	bool	说明脱敏策略是否“继承”自其他脱敏策略。
policy_order	float4	脱敏策略次序。该字段8.2.1.100及以上集群版本支持。

17.2.67 PG_RELFILENODE_SIZE

PG_RELFILENODE_SIZE系统表存储文件级空间统计信息，表中的每一条记录则对应磁盘上相应的物理文件和该文件的文件大小。

表 17-68 PG_RELFILENODE_SIZE 字段

名称	类型	描述
databaseid	oid	物理文件所属database对应的OID。如果是跨库共享系统表，该值为0。
tablespaceid	oid	物理文件所属表空间对应的OID。
relfilenode	oid	物理文件编号。
backendid	integer	创建物理文件的后台线程号，通常为-1。
type	integer	物理文件的文件类型。 • 0为数据类型。 • 1为FSM文件类型。 • 2为VM文件类型。 • 3为BCM文件类型。 • 大于4为列存表对应列的数据文件和BCM文件大小之和。
filesize	bigint	物理文件的文件大小，单位Byte。

 版权所有 © 华为云计算技术有限公司

17.2.68 PG_RLSPOLICY

PG_RLSPOLICY系统表存储行级访问控制策略的信息。

表 17-69 PG_RLSPOLICY 字段

名称	类型	描述
polname	name	行访问控制策略名称。
polrelid	oid	行访问控制策略的表OID。
polcmd	char	行访问控制策略影响的SQL操作，包括：*(ALL)、r(SELECT)、w(UPDATE)、d(DELETE)。
polpermissive	boolean	行访问控制策略的类型。 **说明** polpermissive的取值： ● true表示PERMISSIVE，表示行访问控制策略是宽容性策略。 ● false表示RESTRICTIVE，表示行访问控制策略是限制性策略。
polroles	oid[]	行访问控制策略影响的数据库用户OID。
polqual	pg_node_tree	行访问控制策略的SQL条件表达式。

17.2.69 PG_RESOURCE_POOL

PG_RESOURCE_POOL系统表提供了数据库资源池的信息。

表 17-70 PG_RESOURCE_POOL 字段

名称	类型	描述
respool_name	name	资源池名称。
mem_percent	integer	内存配置的百分比，0代表资源池内存不管控。
cpu_affinity	bigint	保留字段，无实际意义。
control_group	name	资源池所在的control group名字。
active_statements	integer	资源池上最大的并发数。
max_dop	integer	资源池允许的简单作业最大并发数。-1和0代表不限制。
memory_limit	name	单个查询估算内存上限。
parentid	oid	父资源池OID。
io_limits	integer	保留字段，无实际意义。
io_priority	text	保留字段，无实际意义。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
nodegroup	name	资源池关联的逻辑集群名称，非逻辑集群下为“installation”。
is_foreign	boolean	表示资源池是否用于逻辑集群之外的用户。如果为true，表示资源池用来控制不属于当前资源池的普通用户的资源。
short_acc	boolean	资源池是否开启短查询加速，默认开启。 ● 短查询加速开启，简单查询在快车道管控。 ● 短查询加速关闭，简单查询在慢车道管控。
except_rule	text	资源池关联的异常规则，支持关联多个异常规则，异常规则间用逗号分隔。
weight	integer	资源调度权重，目前仅用于网络调度。

17.2.70 PG_REWRITE

PG_REWRITE系统表存储为表和视图定义的重写规则。

表 17-71 PG_REWRITE 字段

名称	类型	描述
rulename	name	规则名称。
ev_class	oid	使用该规则的表名。
ev_attr	smallint	该规则适用的字段（目前总是为0，表示整个表）。
ev_type	"char"	规则适用的事件类型： ● 1 = SELECT ● 2 = UPDATE ● 3 = INSERT ● 4 = DELETE
ev_enabled	"char"	用于控制复制的触发： ● O = “origin”和“local”模式时触发。 ● D = 禁用触发。 ● R =“replica”时触发。 ● A = 任何模式都会触发。
is_instead	boolean	如果是INSTEAD规则，则为真。
ev_qual	pg_node_tree	规则条件的表达式树（以nodeToString() 形式存在）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
ev_action	pg_node_tree	规则动作的查询树（以nodeToString() 形式存在）。
state_change	timestamp with time zone	ev_enabled字段的刷新时间。该字段仅9.1.0.200及以上集群版本支持。
ev_src	text	规则对应视图的原始创建语句。该字段仅9.1.0.226及以上集群版本支持。

17.2.71 PG_SECLABEL

PG_SECLABEL系统表存储数据对象上的安全标签。

PG_SHSECLABEL的作用类似，只是用于在一个数据库集群内共享的数据库对象的安全标签上。

表 17-72 PG_SECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象的系统目录的OID。
objsubid	integer	-	出现在此对象中的列的序号。
provider	text	-	与此标签相关的标签提供者。
label	text	-	应用于此对象的安全标签。

17.2.72 PG_SHDEPEND

PG_SHDEPEND系统表记录数据库对象和共享对象（比如角色）之间的依赖关系。记录的这些信息使DWS可以确保对象在被删除时没有被其他对象引用。

PG_DEPEND的作用类似，只是它是针对单个数据库中对象之间的依赖。

和大多数其他系统表不同，PG_SHDEPEND在集群的所有数据库之间共享：每个数据库集群只有一个PG_SHDEPEND，并非每个数据库一个。

表 17-73 PG_SHDEPEND 字段

名字	类型	引用	描述
dbid	oid	PG_DATABASE.oid	依赖对象所在的数据库的OID ，如果是共享对象，则为0。
classid	oid	PG_CLASS.oid	依赖对象所在的系统表的OID。
objid	oid	任意OID属性	指定的依赖对象的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
objsubid	integer	-	对于一个表字段，为字段号（objid和classid参考表本身）。对于所有其他对象类型，该字段为0。
refclassid	oid	PG_CLASS.oid	被引用对象所在的系统表的OID(必须是一个共享表)。
refobjid	oid	任意OID属性	指定的被引用对象的OID。
deptype	"char"	-	定义该依赖关系的特定语义的代码见表后说明。
objfile	text	-	用户定义C函数库文件路径。

在任何情况下，一条PG_SHDEPEND记录就表明被引用的对象不能在未删除依赖对象的前提下被删除。但是其中也有几种依赖类型由deptype定义的情况：

- SHARED_DEPENDENCY_OWNER (o)

  被引用的对象（必须是一个角色）是依赖对象的所有者。
- SHARED_DEPENDENCY_ACL (a)

  在依赖对象的ACL（访问控制列表，也就是权限列表）中提到被引用的对象（必须是一个角色）。不会为对象的所有者创建SHARED_DEPENDENCY_ACL，因为所有者将具有SHARED_DEPENDENCY_OWNER记录。
- SHARED_DEPENDENCY_PIN (p)

  没有依赖对象。这类记录标识系统自身依赖于被依赖对象，因此这种对象绝对不能被删除。此类型的记录只能被initdb创建，依赖对象的字段都为0。

17.2.73 PG_SHDESCRIPTION

PG_SHDESCRIPTION系统表存储共享数据库对象的可选注释。可以使用COMMENT命令操作注释的内容，使用gsql的\d命令查看注释内容。

PG_DESCRIPTION提供了类似的功能，它记录了单个数据库中对象的注释。

不同于大多数系统表，PG_SHDESCRIPTION在集群中所有数据库之间共享：每个数据库集群只有一个PG_SHDESCRIPTION，而不是每个数据库一个。

表 17-74 PG_SHDESCRIPTION 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此描述所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象所在的系统目录的OID。
description	text	-	作为对该对象描述的任意文本。

 版权所有 © 华为云计算技术有限公司

17.2.74 PG_SHSECLABEL

PG_SHSECLABEL系统表存储在共享数据库对象上的安全标签。安全标签可以用SECURITY LABEL命令操作。

查看安全标签的简单方法，请参考**PG_SECLABELS**。

PG_SECLABEL的作用类似，只是它是用于单个数据库内部对象的安全标签。

不同于大多数的系统表，PG_SHSECLABEL在一个集群中的所有数据库中共享：每个数据库集群只有一个PG_SHSECLABEL，而不是每个数据库一个。

表 17-75 PG_SHSECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属对象的OID。
classoid	oid	**PG_CLASS**.oid	对象所属系统目录的OID。
provider	text	-	与此标签关联的标签提供者。
label	text	-	应用于该对象的安全标签。

17.2.75 PG_STATISTIC

PG_STATISTIC系统表存储有关该数据库中表和索引列的统计数据。需要有系统管理员权限才可以访问此系统表。

表 17-76 PG_STATISTIC 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
staattnum	smallint	所描述的字段在表中的编号，从1开始。
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	记录全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 • 大于0的值是独立数值的实际数目。 • 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 • 0表示独立数值的数目未知。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则为NULL。 n的取值范围：1～5
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则为NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的方法。 n的取值范围：1～5
stadndistinct	real	记录dn1上字段里唯一的非NULL数据值的数目。 ● 大于0的值是独立数值的实际数目。 ● 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadndistinct=-0.5）。 ● 0表示独立数值的数目未知。
staextinfo	text	统计信息的扩展信息。预留字段。

17.2.76 PG_STATISTIC_DATA

PG_STATISTIC_DATA系统表存储当前集群上表的扩展统计信息数据，包括表达式统计信息数据。该系统表仅8.2.1.200及以上集群版本支持。

表 17-77 PG_STATISTIC_DATA 字段

名称	类型	描述
statoid	oid	扩展统计信息对象的OID。
inherit	boolean	是否统计有继承关系的对象。 ● true表示该表有子表。 ● false表示该表无子表。
ndistinct	pg_ndistinct	N-distinct计数，序列化为pg_ndistinct类型。
dependencies	pg_dependencies	函数依赖统计信息，序列化为pg_dependencies类型。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
mcv	pg_mcv_list	最常见值（MCV）统计信息，序列化为pg_mcv_list类型。
expr	pg_statistic[]	每个表达式的统计信息，序列化为pg_statistic类型的数组。

17.2.77 PG_STATISTIC_EXT

PG_STATISTIC_EXT系统表存储有关该数据库中表的扩展统计数据。收集的扩展统计数据范围由用户指定，需要有系统管理员权限才可以访问此系统表。

表 17-78 PG_STATISTIC_EXT 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	记录全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stadndistinct	real	记录dn1上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadndistinct=-0.5）。 ● 0表示独立数值的数目未知。
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	一个用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
stakey	int2vector	所描述的字段编号的数组。
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则为NULL。 n的取值范围：1～5
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则为NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的方法。 n的取值范围：1～5
staexprs	pg_node_tree	扩展统计信息对应的表达式。

17.2.78 PG_STATISTIC_INFO

PG_STATISTIC_INFO系统表存储当前集群上表的扩展统计信息对象，目前包括表达式统计信息对象。该系统表仅8.2.1.200及以上集群版本支持。

表 17-79 PG_STATISTIC_INFO 字段

名称	类型	描述
relid	oid	扩展统计信息对象中使用的表。
statname	name	扩展统计信息对象的名称。
statnamespace	oid	包含该扩展统计信息对象的命名空间的OID。
statowner	oid	该扩展统计信息对象的所有者OID。
stattarget	integer	用于控制统计信息的详细级别。
keys	int2vector	所描述的字段编号的数组，表示该统计信息对象包括哪些列。
kind	"char"[]	包含统计类型的数组，可能的值有： d：表示N-distinct计数统计信息。 f：表示函数依赖统计信息。 m：表示最常见值（MCV）统计信息。 e：表示表达式统计信息。
exprs	pg_node_tree	扩展统计信息对应的表达式。

 版权所有 © 华为云计算技术有限公司

17.2.79 PG_STAT_OBJECT

PG_STAT_OBJECT系统表存储当前实例上表的统计信息和autovacuum效率信息，并对databaseid、relid和partid字段创建索引。该系统表的更新受**enable_pg_stat_object**参数控制。该系统表仅8.2.1及以上集群版本支持。

表 17-80 PG_STAT_OBJECT 字段

名称	类型	引用	描述
databaseid	oid	**PG_DATABASE**.oid	数据库OID。
relid	oid	**PG_CLASS**.oid	表OID，分区表为主表OID。
partid	oid	**PG_PARTITION**.oid	分区OID，普通表此列为0。
numscans	bigint	-	启动顺序扫描的次数。
tuples_returned	bigint	-	顺序扫描抓取的可见元组条数。
tuples_fetched	bigint	-	抓取的可见元组条数。
tuples_inserted	bigint	-	插入条数。
tuples_updated	bigint	-	更新条数。
tuples_deleted	bigint	-	删除条数。
tuples_hot_updated	bigint	-	HOT更新条数。
n_live_tuples	bigint	-	可见元组数。
last_autovacuum_begin_n_dead_tuple	bigint	-	Autovacuum执行前删除元组数。
n_dead_tuples	bigint	-	Autovacuum成功后删除元组数。
changes_since_analyze	bigint	-	Analyze后最近一次数据修改时间。
blocks_fetched	bigint	-	选中的页面数。
blocks_hit	bigint	-	扫描过的页面数。
cu_mem_hit	bigint	-	CU内存命中次数。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
cu_hdd_sync	bigint	-	从磁盘同步读取CU次数。
cu_hdd_asyn	bigint	-	从磁盘异步读取CU次数。
data_changed_timestamp	timestamp with time zone	-	最近一次数据修改时间。
data_access_timestamp	timestamp with time zone	-	表的最后一次访问时间。
analyze_timestamp	timestamp with time zone	-	最近一次Analyze时间。
analyze_count	bigint	-	Analyze总次数。
autovac_analyze_timestamp	timestamp with time zone	-	最近一次autoanalyze时间。
autovac_analyze_count	bigint	-	Autoanalyze总次数。
vacuum_timestamp	timestamp with time zone	-	最近一次vacuum的时间。
vacuum_count	bigint	-	vacuum总次数。
autovac_vacuum_timestamp	timestamp with time zone	-	最近一次autovacuum时间。
autovac_vacuum_count	bigint	-	Autovacuum总次数。
autovacuum_success_count	bigint	-	成功执行的autovacuum总次数。
last_autovacuum_time_cost	bigint	-	最近一次成功的autovacuum花费时间，单位：微秒。
avg_autovacuum_time_cost	bigint	-	成功执行autovacuum的平均执行时间，单位：微秒。
last_autovacuum_failed_count	bigint	-	从上一次autovacuum成功到现在，autovacuum总失败次数。
last_autovacuum_trigger	smallint	-	最近一次autovacuum触发方式，用于辅助维护人员进行vacuum情况的判断。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
last_autovacuum_oldestxmin	bigint	-	最近一次autovacuum成功执行后的oldestxmin。如果表级oldestxmin特性开启，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值。
last_autovacuum_scan_pages	bigint	-	最近一次autovacuum扫描的页面数（仅针对行存表）。
last_autovacuum_dirty_pages	bigint	-	最近一次autovacuum修改的页面数（仅针对行存表）。
last_autovacuum_clear_deadtuples	bigint	-	最近一次autovacuum清理的deadtuple数（仅针对行存表）。
sum_autovacuum_scan_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计扫描的页面数（仅针对行存表）。
sum_autovacuum_dirty_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计修改的页面数（仅针对行存表）。
sum_autovacuum_clear_deadtuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的deadtuple数（仅针对行存表）。
last_autovacuum_begin_cu_size	bigint	-	最近一次autovacuum前的CU文件大小（仅针对列存表）。
last_autovacuum_cu_size	bigint	-	最近一次autovacuum后的CU文件大小（仅针对列存表）。
last_autovacuum_rewrite_size	bigint	-	最近一次autovacuum重写的列存文件大小（仅针对列存表）。
last_autovacuum_clear_size	bigint	-	最近一次autovacuum清理的列存文件大小（仅针对列存表）。
last_autovacuum_clear_cbtree_tuples	bigint	-	最近一次autovacuum清理的cbtree tuple数（仅针对列存表）。
sum_autovacuum_rewrite_size	bigint	-	从数据库初始化开始到现在，autovacuum累计重写的列存文件大小（仅针对列存表）。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
sum_autovacuum_clear_size	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_cbtree_tuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的cbtree tuple数（仅针对列存表）。
last_autovacuum_csn	bigint	-	如果表级oldestxmin特性打开，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值对应的CSN值。
last_automerge_timestamp	timestamp with time zone	-	上次表发生automerge的时间（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
last_automerge_time_cost	bigint	-	上次表发生automerge的耗时（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
last_automerge_count	bigint	-	上次表automerge的记录数（目前仅针对HStore_opt表），该字段仅9.1.0.100及以上版本支持。
read_sql_count_1d	bigint	-	当日表的累计访问次数。该字段仅9.1.1.100及以上集群版本支持。
extra1	bigint	-	预留字段1。

17.2.80 PG_SUBSCRIPTION

PG_SUBSCRIPTION系统表存储所有现有的订阅。

表 17-81 PG_SUBSCRIPTION 字段

名称	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
subdbid	oid	PG_DATABASE.oid	订阅所在的数据库的OID。
subname	name	-	订阅的名称。
subowner	oid	PG_AUTHID.oid	订阅的拥有者。
subenabled	boolean	-	如果为真，订阅被启用并且应该被复制。
subconninfo	text	-	到发布端数据库的连接信息。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
subslotname	text	-	发布端数据库中复制槽的名称。空表示为NONE。
subpublications	text[]	-	被订阅的发布名称的数组。这些引用的是发布者服务器上的发布。

应用示例

查看所有订阅。

```
SELECT * FROM pg_subscription;
 subdbid | subname | subowner | subenabled |
subconninfo                                | subslotname | subpublications
---------+---------+----------+------------
+--------------------------------------------------------------------------------------------+-------------+-----------------
   15992 | mysub   |       10 | t          | host=1.1.1.1,2.2.2.2 port=10000,20000 dbname=postgres user=repusr1
password=password | mysub       | {mypub}
(1 row)
```

17.2.81 PG_SYNONYM

PG_SYNONYM系统表存储同义词对象名与其他数据库对象名间的映射信息。

表 17-82 PG_SYNONYM 字段

名称	类型	描述
synname	name	同义词名称。
synnamespace	oid	该同义词所在的命名空间的OID。
synowner	oid	同义词的所有者，通常是其创建者OID。
synobjschema	name	关联对象指定的模式名。
synobjname	name	关联对象名。

17.2.82 PG_TABLESPACE

PG_TABLESPACE系统表存储表空间信息。

表 17-83 PG_TABLESPACE 字段

名称	类型	描述
spcname	name	表空间名。
spcowner	oid	表空间的所有者，通常是其创建者。
spcacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
spcoptions	text[]	表空间的选项。
spcmaxsize	text	可使用的最大磁盘空间大小，单位Byte。
relative	boolean	记录表空间指定的存储路径是否为相对路径。

17.2.83 PG_TRIGGER

PG_TRIGGER系统表存储触发器信息。

名称	类型	描述
tgrelid	oid	触发器所在表的OID。
tgname	name	触发器名。
tgfoid	oid	触发器OID。
tgtype	smallint	触发器类型。
tgenabled	"char"	O表示触发器在“origin”和“local”模式下触发。 D表示触发器被禁用。 R表示触发器在“replica”模式下触发。 A表示触发器始终触发。
tgisinternal	boolean	内部触发器标识，如果为true表示内部触发器。
tgconstrrelid	oid	完整性约束引用的表。
tgconstrindid	oid	完整性约束的索引。
tgconstraint	oid	约束触发器在pg_constraint中的OID。
tgdeferrable	boolean	约束触发器是为DEFERRABLE类型。
tginitdeferred	boolean	约束触发器是否为INITIALLY DEFERRED类型。
tgnargs	smallint	触发器函数入参个数。
tgattr	int2vector	当触发器指定列时的列号，未指定则为空数组。
tgargs	bytea	传递给触发器的参数。
tgqual	pg_node_tree	表示触发器的WHEN条件，如果没有则为null。

17.2.84 PG_TDE

PG_TDE系统表存储透明加密密钥信息，该系统表仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-84 PG_TDE 字段

名字	类型	描述
encryptalgo	char	加密算法。支持两种加密算法： ● a：表示aes算法。 ● s：表示国密sm4算法。
dekcipher	text	数据密钥。
cekid	text	主密钥（用于加密数据密钥）。

应用示例

```
SELECT * FROM pg_tde;
encryptalgo|                                dekcipher                                |                cekid
-----------+-------------------------------------------------------------------------------------------------
+--------------------------------------------------------
   a       |
a472d0d8f95a4449e4138b2dd39fdacdd516311a9163c912340229a43da092656a994b7bb0e3cee5bc438aafcb
b99889 |      fb234337-45b3-4cab-8bfe-0086b983c4c3
   s       |
74937c481f70445f90bb0e1831431353cbb4a3203535bf67432455275d6a93bf531310301f690b5274078114e2
431311 |      fb234337-45b3-4cab-8bfe-0086b983c4c3
(2 rows)
```

17.2.85 PG_TS_CONFIG

PG_TS_CONFIG系统表包含表示文本搜索配置的选项。一个配置指定一个特定的文本搜索解析器和一个用于解析器输出类型的字典列表。

解析器在PG_TS_CONFIG记录中显示，但是字典映射的标记是由PG_TS_CONFIG_MAP中的辅助记录定义的。

表 17-85 PG_TS_CONFIG 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
cfgname	name	-	文本搜索配置名。
cfgnamespace	oid	PG_NAMESPACE.oid	此配置所在的命名空间的OID。
cfgowner	oid	PG_AUTHID.oid	配置的所有者。
cfgparser	oid	PG_TS_PARSER.oid	此配置的文本搜索解析器的OID。
cfoptions	text[]	-	分词相关配置选项。

版权所有 © 华为云计算技术有限公司

17.2.86 PG_TS_CONFIG_MAP

PG_TS_CONFIG_MAP系统表包含为每个文本搜索配置的解析器的每种输出符号类型，显示有哪些文本搜索字典可供查询以及以哪种顺序搜索。

表 17-86 PG_TS_CONFIG_MAP 字段

名字	类型	引用	描述
mapcfg	oid	PG_TS_CONFIG.oid	拥有此映射记录的PG_TS_CONFIG的OID。
maptokentype	integer	-	由配置的解析器发出的一个符号类型。
mapseqno	integer	-	查询该项的顺序。
mapdict	oid	PG_TS_DICT.oid	查询的文本搜索字典的OID。

17.2.87 PG_TS_DICT

PG_TS_DICT系统表包含定义文本搜索字典的项。字典取决于文本搜索模板，该模板显示所有需要实现的功能。字典本身提供了用户可设置参数的模板。

即允许字典通过非权限用户创建。参数由文本字符串dictinitoption指定，参数的格式和意义取决于模板。

表 17-87 PG_TS_DICT 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
dictname	name	-	文本搜索字典名。
dictnamespace	oid	PG_NAMESPACE.oid	此字典所在的命名空间的OID。
dictowner	oid	PG_AUTHID.oid	字典的所有者。
dicttemplate	oid	PG_TS_TEMPLATE.oid	此字典的文本搜索模板的OID。
dictinitoption	text	-	模板的初始化选项字符串。

版权所有 © 华为云计算技术有限公司

17.2.88 PG_TS_PARSER

PG_TS_PARSER系统表包含定义文本解析器的项。解析器负责分割输入文本为词位，并且为每个词位分配标记类型。因为解析器必须通过C语言级别的函数实现，所以新解析器必须由数据库系统管理员创建。

表 17-88 PG_TS_PARSER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
prsname	name	-	文本搜索解析器名称。
prsnamespace	oid	PG_NAMESPACE.oid	解析器所在的命名空间的OID。
prsstart	regproc	PG_PROC.oid	解析器的启动函数的OID。
prstoken	regproc	PG_PROC.oid	解析器的下一个标记函数的OID。
prsend	regproc	PG_PROC.oid	解析器的关闭函数的OID。
prsheadline	regproc	PG_PROC.oid	解析器的标题函数的OID。
prslextype	regproc	PG_PROC.oid	解析器的lextype函数的OID。

17.2.89 PG_TS_TEMPLATE

PG_TS_TEMPLATE系统表包含定义文本搜索模板的项。模板是文本搜索字典的类的实现框架。因为模板必须通过C语言级别的函数实现，索引新模板的创建必须由数据库系统管理员创建。

表 17-89 PG_TS_TEMPLATE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
tmplname	name	-	文本搜索模板名。
tmplnamespace	oid	PG_NAMESPACE.oid	模板所属的命名空间的OID。
tmplinit	regproc	PG_PROC.oid	模板的初始化函数的OID。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
tmpllexize	regproc	PG_PROC.oid	模板的lexize函数的OID。

17.2.90 PG_TYPE

PG_TYPE系统表存储数据类型的相关信息。

表 17-90 PG_TYPE 字段

名称	类型	描述
typname	name	数据类型名称。
typnamespace	oid	此类型所在的命名空间的OID。
typowner	oid	此类型的所有者。
typlen	smallint	对于定长类型是该类型内部表现形式的字节数。对于变长类型typlen为负值。 ● -1表示一种“变长”（有长度字属性的数据）。 ● -2表示一个以NULL结尾的C字符串。
typbyval	boolean	指定内部传递这个类型的数值时是传值还是传引用。如果该类型的TYPLEN不是1，2，4，8， TYPBYVAL建议设置为false。变长类型通常是传引用。即使TYPLEN允许传值，TYPBYVAL也可以为false。
typtype	"char"	● b表示基础类型。 ● c表示复合类型（比如，一个表的行类型）。 ● e表示枚举类型。 ● p表示伪类型。 参见typrelid和typbasetype。
typcategory	"char"	数据类型的模糊分类，可用于解析器使用的数据转换依据。
typispreferred	boolean	如果为真，则数据符合TYPCATEGORY所指定的转换规则时进行转换。
typisdefined	boolean	如果定义了类型，则为真。如果是一种尚未定义的类型的占位符，则为假。如果为假，则除了该类型名称，命名空间和OID之外没有可依赖的对象。
typdelim	"char"	分析数组输入时，分隔两个此类型数值的字符。请注意，分隔符是与数组元素数据类型相关联，而不是与数组数据类型相关联。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
typrelid	oid	如果是复合类型（请参见typtype），则此字段指向pg_class中定义该表的行。对于独立的复合类型，pg_class记录并不表示一个表，但是总需要它来查找该类型连接的pg_attribute记录。非复合类型为0。
typelem	oid	如果不为0，则它标识pg_type中的另一行。当前类型可以像一个产生类型为typelem的数组来描述。“true”数组类型是变长的（typlen= -1），但是某些定长（typlen > 0）类型也有非零的typelem（比如name和point）。如果一个定长类型有typelem，则其内部形式必须是typelem数据类型的某个数目的个数值，不能有其他数据。变长数组类型有一个该数组子过程定义的头（文件）。
typarray	oid	如果不为0，则表示在pg_type中有对应的类型记录。
typinput	regproc	输入转换函数（文本格式）。
typoutput	regproc	输出转换函数（文本格式）。
typreceive	regproc	输入转换函数（二进制格式），如果没有则为0。
typsend	regproc	输出转换函数（二进制格式），如果没有则为0。
typmodin	regproc	输入类型修改符函数，如果为0，则不支持。
typmodout	regproc	输出类型修改符函数，如果为0，则不支持。
typanalyze	regproc	自定义的ANALYZE函数，如果使用标准函数，则为0。
typalign	"char"	当存储此类型的数值时要求的对齐方式。适用于磁盘存储以及该值在数据库中的大多数形式。如果数值是连续存储的，比如在磁盘上以完全的裸数据的形式存放时，则先在此类型的数据前填充空白，这样它就可以按照要求的边界存储。对齐引用是该序列中第一个数据的开头。可能的值包含： ● c = char对齐，即不需要对齐。 ● s = short对齐（在大多数机器上是2字节） ● i = int对齐（在大多数机器上是4字节） ● d = double对齐（在大多数机器上是8字节，但不一定是全部） **须知** 对于系统表里使用的类型，在pg_type里定义的尺寸和对齐方式要和编译器在表示表行的结构中布局方式保持一致。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
typstorage	"char"	指明一个变长类型（那些有typlen = -1）是否准备好应付非常规值，以及对这种属性的类型的缺省策略是什么。可能的值包含： ● p：数值总是以简单方式存储。 ● e：数值可以存储在一个"次要"关系中（如果有该关系，请参见pg_class.reltoastrelid）。 ● m：数值可以以内联压缩方式存储。 ● x：数值可以以内联压缩方式或者在"次要"表里存储。 **须知** m域也可以移到从属表里存储，但只是最后的解决方法（首先移动e和x域）。
typnotnull	boolean	表示在某类型上的一个NOTNULL约束。目前只用于域。
typbasetype	oid	如果这是一个衍生类型（请参见typtype），则该标识作为这个类型的基础的类型。如果不是衍生类型则为零。
typtypmod	integer	域使用typtypmod记录要应用于其基础类型上的typmod（如果基础类型不使用typmod，则为-1）。如果此类型不是域，则为-1。
typndims	integer	如果一个域是数组，则typndims是数组维数的数值（即typbasetype是一个数组类型；域的typelem将匹配基本类型的typelem）。除了数组类型的域以外的类型为0。
typcollation	oid	指定类型的排序规则。如果为0，则表示不支持排序规则。
typdefaultbin	pg_node_tree	如果不为NULL，则为该类型缺省表达式的nodeToString() 表现形式。目前这个字段只用于域。
typdefault	text	如果某类型没有相关缺省值，则为NULL。如果typdefaultbin不为NULL，则typdefault必须包含一个typdefaultbin代表的缺省表达式的人类可读版本。如果typdefaultbin为NULL但typdefault不为NULL，typdefault则是该类型缺省值的外部表现形式，可以将其输入到类型的转换器生成一个常量。
typacl	aclitem[]	访问权限。

17.2.91 PG_USER_MAPPING

PG_USER_MAPPING系统表存储从本地用户到远程的映射。

需要有系统管理员权限才可以访问此系统表。普通用户可以使用视图**PG_USER_MAPPINGS**进行查询。

版权所有 © 华为云计算技术有限公司

表 17-91 PG_USER_MAPPING 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
umuser	oid	PG_AUTHID.oid	被映射的本地用户的OID，如果用户映射是公共的则为0。
umserver	oid	PG_FOREIGN_SERVER.oid	包含此映射的外部服务器的OID。
umoptions	text[]	-	用户映射指定选项，使用“keyword=value”格式的字符串。

17.2.92 PG_USER_STATUS

PG_USER_STATUS系统表提供了访问数据库用户的状态。需要有系统管理员权限才可以访问此系统表

表 17-92 PG_USER_STATUS 字段

名称	类型	描述
roloid	oid	角色的标识。
failcount	integer	尝试失败次数。
locktime	timestamp with time zone	角色被锁定的时间点。
rolstatus	smallint	角色的状态。 • 0：正常状态。 • 1：由于登录失败次数超过阈值被锁定了一定的时间。 • 2：被管理员锁定。
permspace	bigint	角色在当前实例上已经使用的永久表存储空间大小，单位为Byte。
tempspace	bigint	角色在当前实例上已经使用的临时表存储空间大小，单位为Byte。

17.2.93 PG_WORKLOAD_ACTION

PG_WORKLOAD_ACTION系统表存储query_band的信息。

 版权所有 © 华为云计算技术有限公司

表 17-93 PG_WORKLOAD_ACTION 字段

名称	类型	描述
qband	name	query band键值对。
class	name	query band关联行为类别。
object	name	query band关联行为。
action	name	query band关联行为表现。

17.2.94 PGXC_CLASS

PGXC_CLASS系统表存储每张表的复制或分布信息。

表 17-94 PGXC_CLASS 字段

名称	类型	描述
pcrelid	oid	表的OID。
pclocatortype	"char"	定位器类型。 ● H：hash ● M：Modulo ● N：Round Robin ● R：Replicate
pchashalgorithm	smallint	使用哈希算法分布元组。
pchashbuckets	smallint	哈希容器的值。
pgroup	name	节点组名称。
redistributed	"char"	表已经完成重分布。
redis_order	integer	重分布的顺序。
pcattnum	int2vector	用作分布键的列标号。
nodeoids	oidvector_extend	表分布的节点OID列表。
options	text	系统内部保留字段，存储扩展状态信息。
node_ids	oidvector_extend	表分布的节点ID列表。该字段仅9.1.1.200及以上集群版本支持。

17.2.95 PGXC_GROUP

PGXC_GROUP系统表存储节点组信息，在存算分离3.0版本中，每个计算组节点称为一个VW（Virtual Warehouse），而在存储KV层，每一个VW会和一个vgroup相对应。

 版权所有 © 华为云计算技术有限公司

表 17-95 PGXC_GROUP 字段

名称	类型	描述
group_name	name	节点组名称。
in_redistribution	"char"	是否需要重分布： ● n，表示NodeGroup没有再进行重分布。 ● y，表示NodeGroup是重分布过程中的源节点组。 ● t，表示NodeGroup是重分布过程中的目的节点组。 ● s，表示NodeGroup不需要重分布，重分布过程将跳过此节点组。
group_members	oidvector_extend	节点组的节点OID列表。
group_buckets	text	分布数据桶的集合。
is_installation	boolean	是否安装子集群。
group_acl	aclitem[]	访问权限。
group_kind	"char"	节点组类型： ● i，表示安装节点组，包含所有DN节点。 ● n，表示普通非逻辑集群节点组。 ● v，表示逻辑集群节点组。 ● e，表示弹性集群节点组 ● r，表示复制表节点组，只能用于创建复制表，可以包含一个或多个逻辑集群节点组。
group_ckpt_csn	xid	节点组最近一次执行增量抽取的CSN。
vgroup_id	xid	节点组对应vgroup的ID标识。
vgroup_bucket_count	oid	节点组对应vgroup的桶数目。
group_ckpt_time	timestamp with time zone	节点组最近一次执行增量抽取的物理时间。
apply_kv_duration	integer	节点组最近一次执行增量抽取中增量扫描耗时(单位为秒)。
ckpt_duration	integer	节点组最近一次执行增量抽取中checkpoint耗时（单位为秒）。

 版权所有 © 华为云计算技术有限公司

17.2.96 PGXC_NODE

PGXC_NODE系统表存储集群节点信息。

表 17-96 PGXC_NODE 字段

名称	类型	描述
node_name	name	节点名称。
node_type	"char"	节点类型。 C：协调节点。 D：数据节点。
node_port	integer	节点的端口号。
node_host	name	节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
node_port1	integer	复制节点的端口号。
node_host1	name	复制节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
hostis_primary	boolean	表明当前节点是否发生主备切换。
nodeis_primary	boolean	在replication表下，是否优选当前节点作为优先执行的节点进行非查询操作。
nodeis_preferred	boolean	在replication表下，是否优选当前节点作为首选的节点进行查询。
node_id	integer	节点标识符。
sctp_port	integer	主节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port	integer	主节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
sctp_port1	integer	备节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port1	integer	备节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
nodeis_central	boolean	当前节点为中心控制节点。

应用示例

查询集群的CN和DN信息：

```
SELECT * FROM pgxc_node;
  node_name   | node_type | node_port | node_host | node_port1 | node_host1 | hostis_primary |
nodeis_primary | nodeis_preferred |    node_id
 | sctp_port | control_port | sctp_port1 | control_port1 | nodeis_central | read_only
--------------+-----------+-----------+-----------+------------+------------+----------------+----------------
```


版权所有 © 华为云计算技术有限公司

```
+------------------+------------
-+-----------+--------------+------------+--------------+----------------+-----------
 datanode1    | D        |    55504 | localhost |    55504 | localhost  | t           | f           | f           |
888802358
 |    55505 |      55507 |        0 |           0 | f           | f
 datanode2    | D        |    55508 | localhost |    55508 | localhost  | t           | f           | f           |
-905831925
 |    55509 |      55511 |        0 |           0 | f           | f
 coordinator1 | C        |    55500 | localhost |    55500 | localhost  | t           | f           | f           |
1938253334
 |        0 |          0 |        0 |           0 | t           | f
 datanode3    | D        |    55542 | localhost |    55542 | localhost  | t           | f           | f           |
-1894792127
 |    57552 |      55544 |        0 |           0 | f           | t
 datanode4    | D        |    55546 | localhost |    55546 | localhost  | t           | f           | f           |
-1307323892
 |    57808 |      55548 |        0 |           0 | f           | t
 datanode5    | D        |    55550 | localhost |    55550 | localhost  | t           | f           | f           |
1797586929
 |    58064 |      55552 |        0 |           0 | f           | t
 datanode6    | D        |    55554 | localhost |    55554 | localhost  | t           | f           | f           |
587455710
 |    58320 |      55556 |        0 |           0 | f           | t
 datanode7    | D        |    55558 | localhost |    55558 | localhost  | t           | f           | f           |
-1685037427
 |    58576 |      55560 |        0 |           0 | f           | t
 datanode8    | D        |    55562 | localhost |    55562 | localhost  | t           | f           | f           |
-993847320
 |    58832 |      55564 |        0 |           0 | f           | t
(9 rows)
```

17.2.97 PLAN_TABLE_DATA

PLAN_TABLE_DATA系统表存储了用户通过执行EXPLAIN PLAN收集到的计划信息。与**PLAN_TABLE**视图不同的是PLAN_TABLE_DATA表存储了所有session和user执行EXPLAIN PLAN收集的计划信息。

表 17-97 PLAN_TABLE_DATA 字段

名称	类型	描述
session_id	text	表示插入该条数据的会话，由服务线程启动时间戳和服务线程ID组成。受非空约束限制。
user_id	oid	用户ID，用于标识触发插入该条数据的用户。受非空约束限制。
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	计划中的节点编号。
operation	varchar2(30)	操作描述。
options	varchar2(255)	操作选项。
object_name	name	操作对应的对象名，来自于用户定义。
object_type	varchar2(30)	对象类型。
object_owner	name	对象所属schema，来自于用户定义。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
projection	varchar2(4000)	操作输出的列信息。

说明

- PLAN_TABLE_DATA中包含了当前节点所有用户、所有会话的数据，仅管理员有访问权限。普通用户可以通过**PLAN_TABLE**视图查看属于自己的数据。
- 对于不活跃（已退出）的会话，其在PLAN_TABLE_DATA中的数据会在一定时间（默认5min）后被gs_clean清理。用户也可以手动执行gs_clean -C选项对表中不活跃的会话数据进行清理。
- PLAN_TABLE_DATA中的数据是用户通过执行EXPLAIN PLAN命令后由系统自动插入表中，因此禁止用户手动对数据进行插入或更新，否则会引起表中的数据混乱。需要对表中数据删除时，建议通过**PLAN_TABLE**视图。
- statement_id、object_name、object_owner和projection字段内容遵循用户定义的大小写存储，其他字段内容采用大写存储。

17.2.98 SNAPSHOT

SNAPSHOT系统表记录每次创建性能视图快照的起止时间，设置enable_wdr_snapshot为on后，该表由后台快照线程创建并维护。需要有系统管理员权限才可以访问此系统表。

须知

- 此系统表的schema是dbms_om。
- 禁止从外部修改或删除此表，否则可能引起视图快照相关功能异常。

表 17-98 dbms_om.snapshot 字段

名称	类型	描述
snapshot_id	name	快照ID（此字段为主键和分布键）。
start_ts	timestamp with time zone	快照开始时间。
end_ts	timestamp with time zone	快照结束时间。

17.2.99 SQL_OUTLINE

SQL_OUTLINE系统表记录语句的计划信息，planmgmt_options设置了相应取值后，会为满足条件的语句生成sql_hash、plan_hash、outline_name并保存计划到此系统表。

 版权所有 © 华为云计算技术有限公司

须知

- 此系统表的schema是dbms_om。
- 禁止从外部修改或删除此表，否则可能引起plan management相关功能异常。
- 此系统表当前不支持自动清理，开启plan management自动提取并保存outline功能后，需关注此系统表空间使用情况，避免占用空间过大导致集群异常。

表 17-99 SQL_OUTLINE 字段

名称	类型	描述
outline_name	text	outline的名字，值以outline_开头。
sql_hash	text	生成该计划的实际sql的sql_hash，值以sql_开头。
plan_hash	text	该计划的plan_hash，值以plan_开头。
create_time	timestamp with time zone	该outline生成的时间。
outline_version	numeric	outline的版本号。 **说明** 不同版本号的outline支持的hint类型不同，具体如下： • 821.001：支持join顺序的hint、join方式的hint和scan方式的hint。 • 821.002：在821.001的基础上，增加了对stream方式的hint（不含AGG重分布分布键hint）。 • 910.001: 在821.002的基础上，增加了对skew方式的hint，支持join和agg两种场景下的skew hint提取。
outline	text	通过计划自动生成的outline字符串。
cost	float8	生成outline的计划成本（cost）。该字段仅9.1.1.200及以上集群版本支持。

17.2.100 TABLES_SNAP_TIMESTAMP

TABLES_SNAP_TIMESTAMP系统表记录每次对每个性能视图创建快照的起止时间，设置enable_wdr_snapshot为on后，该表由后台快照线程创建并维护。需要有系统管理员权限才可以访问此系统表。

表 17-100 dbms_om.tables_snap_timestamp 字段

名称	类型	描述
snapshot_id	name	快照ID（此字段为主键和分布键）。
db_name	text	视图所属数据库名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tablename	text	视图名称。
start_ts	timestamp with time zone	快照开始时间。
end_ts	timestamp with time zone	快照结束时间。

须知

- 此系统表的schema是dbms_om。
- 禁止从外部修改或删除此表，否则可能引起视图快照相关功能异常。

17.2.101 性能视图快照系统表

设置enable_wdr_snapshot为on后，后台快照线程会创建并维护以“SNAP_+视图名称”方式命名的系统表，用于记录各性能视图的快照结果。需要有系统管理员权限才可以访问下列系统表。

- SNAP_**PGXC_OS_RUN_INFO**
- SNAP_**PGXC_WAIT_EVENTS**
- SNAP_**PGXC_INSTR_UNIQUE_SQL**
- SNAP_**PGXC_STAT_BAD_BLOCK**
- SNAP_**PGXC_STAT_BGWRITER**
- SNAP_**PGXC_STAT_REPLICATION**
- SNAP_**PGXC_REPLICATION_SLOTS**
- SNAP_**PGXC_SETTINGS**
- SNAP_**PGXC_INSTANCE_TIME**
- SNAP_**GLOBAL_WORKLOAD_TRANSACTION**
- SNAP_**PGXC_WORKLOAD_SQL_COUNT**
- SNAP_**PGXC_STAT_DATABASE**
- SNAP_**GLOBAL_STAT_DATABASE**
- SNAP_**PGXC_REDO_STAT**
- SNAP_**GLOBAL_REDO_STAT**
- SNAP_**PGXC_REL_IOSTAT**
- SNAP_**GLOBAL_REL_IOSTAT**
- SNAP_**PGXC_TOTAL_MEMORY_DETAIL**
- SNAP_**PGXC_NODE_STAT_RESET_TIME**
- SNAP_**PGXC_SQL_COUNT**
- SNAP_**GLOBAL_TABLE_STAT**

版权所有 © 华为云计算技术有限公司

- SNAP_**GLOBAL_TABLE_CHANGE_STAT**
- SNAP_**GLOBAL_COLUMN_TABLE_IO_STAT**
- SNAP_**GLOBAL_ROW_TABLE_IO_STAT**

此类系统表除增加snapshot_id字段（bigint类型）外，其余的字段定义与对应视图相同，且各表的分布键均为snapshot_id。

例如，SNAP_PGXC_OS_RUN_INFO，用于存储PGXC_OS_RUN_INFO视图的快照，其字段新增了snapshot_id，其余字段含义均与PGXC_OS_RUN_INFO视图相同。

须知

- 以上系统表的schema都是dbms_om。
- 禁止从外部修改或删除以上系统表，否则可能引起视图快照相关功能异常。

17.3 系统视图

17.3.1 ALL_ALL_TABLES

ALL_ALL_TABLES视图存储当前用户所能访问的表或视图。

表 17-101 ALL_ALL_TABLES 字段

名称	类型	描述
owner	name	表或视图的所有者。
table_name	name	表或视图的名称。
tablespace_name	name	表或视图所在的表空间。

17.3.2 ALL_CONSTRAINTS

ALL_CONSTRAINTS视图存储当前用户可访问的约束的信息。

表 17-102 ALL_CONSTRAINTS 字段

名称	类型	描述
constraint_name	character varying(64)	约束名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
constraint_type	text	约束类型。 • c表示检查约束。 • f表示外键约束。 • p表示主键约束。 • u表示唯一约束。
table_name	character varying(64)	约束相关的表名。
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

17.3.3 ALL_CONS_COLUMNS

ALL_CONS_COLUMNS视图存储当前用户可访问的约束字段的信息。

表 17-103 ALL_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名。
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。
position	smallint	表中列的位置。

17.3.4 ALL_COL_COMMENTS

ALL_COL_COMMENTS视图存储当前用户可访问的表或视图中字段的注释信息。

表 17-104 ALL_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
comments	text	注释。

17.3.5 ALL_DEPENDENCIES

ALL_DEPENDENCIES视图存储了当前用户可访问的函数、高级包之间的依赖关系。

须知

因为相关信息的限制，目前DWS中，此表为空表，表内没有任何记录。

表 17-105 ALL_DEPENDENCIES 字段

名称	类型	描述
owner	character varying(30)	对象的所有者。
name	character varying(30)	对象的名称。
type	character varying(17)	对象的类型。
referenced_owner	character varying(30)	引用对象的所有者。
referenced_name	character varying(64)	引用对象的名称。
referenced_type	character varying(17)	引用对象的类型。
referenced_link_name	character varying(128)	引用对象的链接的名称。
schemaid	numeric	当前schema的ID。
dependency_type	character varying(4)	依赖类型（REF或HARD）。

17.3.6 ALL_IND_COLUMNS

ALL_IND_COLUMNS视图存储了当前用户可访问的所有索引的字段信息。

表 17-106 ALL_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
column_name	name	列名。
column_position	smallint	索引中列的位置。

17.3.7 ALL_IND_EXPRESSIONS

ALL_IND_EXPRESSIONS视图存储了当前用户可访问的表达式索引的信息。

表 17-107 ALL_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

17.3.8 ALL_INDEXES

ALL_INDEXES视图存储了当前用户可访问的索引信息。

表 17-108 ALL_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引的名称是否为系统生成。
partitioned	character(3)	表示索引是否具有分区表的性质。

 版权所有 © 华为云计算技术有限公司

17.3.9 ALL_OBJECTS

ALL_OBJECTS视图记录了当前用户可访问的数据库对象。

表 17-109 ALL_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。
object_id	oid	对象的OID。
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。
last_ddl_time	timestamp with time zone	对象的最后修改时间。

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

17.3.10 ALL_PROCEDURES

ALL_PROCEDURES视图存储了当前用户可访问的所有存储过程或函数信息。

表 17-110 ALL_PROCEDURES 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象名称。

17.3.11 ALL_SEQUENCES

ALL_SEQUENCES视图存储当前用户能够访问的所有序列。

表 17-111 ALL_SEQUENCES 字段

名称	类型	描述
sequence_owner	name	序列所有者。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
sequence_name	name	序列的名称。
min_value	bigint	序列最小值。
max_value	bigint	序列最大值。
increment_by	bigint	序列的增量。
cycle_flag	character(1)	表示序列是否是循环序列，取值为Y或N。 ● Y表示是循环序列。 ● N表示不是循环序列。

17.3.12 ALL_SOURCE

ALL_SOURCE视图存储当前用户可访问的存储过程或函数信息，且提供存储过程或函数定义的字段。

表 17-112 ALL_SOURCE 字段

名称	类型	描述
owner	name	对象的所有者。
name	name	对象名称。
type	name	对象的类型。
text	text	对象的定义。

17.3.13 ALL_SYNONYMS

ALL_SYNONYMS视图存储了当前用户可访问的所有同义词信息。

表 17-113 ALL_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者。
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
table_name	text	关联对象名。

17.3.14 ALL_TAB_COLUMNS

ALL_TAB_COLUMNS视图存储了当前用户可访问的表和视图的列的描述信息。

表 17-114 ALL_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	对象创建或增加列时列的序号。
data_length	integer	列的字节长度。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2，bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

17.3.15 ALL_TAB_COMMENTS

ALL_TAB_COMMENTS视图存储当前用户可访问的所有表和视图的注释信息。

 版权所有 © 华为云计算技术有限公司

表 17-115 ALL_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

17.3.16 ALL_TABLES

ALL_TABLES视图存储当前用户可访问的所有表。

表 17-116 ALL_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	表是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

17.3.17 ALL_USERS

ALL_USERS视图存储记录数据库中所有用户，但不对用户信息进行详细的描述。

表 17-117 ALL_USERS 字段

名称	类型	描述
username	name	用户名。
user_id	oid	用户的OID。

 版权所有 © 华为云计算技术有限公司

17.3.18 ALL_VIEWS

ALL_VIEWS视图存储了当前用户可访问的所有视图描述信息。

表 17-118 ALL_VIEWS 字段

名称	类型	描述
owner	name	视图的所有者。
view_name	name	视图的名称。
text_length	integer	视图文本长度。
text	text	视图文本。

17.3.19 DBA_DATA_FILES

DBA_DATA_FILES视图存储关于数据库文件的描述。需要有系统管理员权限才可以访问。

表 17-119 DBA_DATA_FILES 字段

名称	类型	描述
tablespace_name	name	文件所属的表空间的名称。
bytes	double precision	文件的字节长度。

17.3.20 DBA_USERS

DBA_USERS视图存储关于数据库所有用户名信息。需要有系统管理员权限才可以访问。

表 17-120 DBA_USERS 字段

名称	类型	描述
username	character varying(64)	用户名。

17.3.21 DBA_COL_COMMENTS

DBA_COL_COMMENTS视图存储关于数据库中表和视图中字段的注释信息。需要有系统管理员权限才可以访问。

版权所有 © 华为云计算技术有限公司

表 17-121 DBA_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。
comments	text	注释。

17.3.22 DBA_CONSTRAINTS

DBA_CONSTRAINTS视图存储关于数据库表中约束的信息。需要有系统管理员权限才可以访问。

表 17-122 DBA_CONSTRAINTS 字段

名称	类型	描述
constraint_name	character varying(64)	约束名。
constraint_type	text	约束类型。 ● c表示检查约束。 ● f表示外键约束。 ● p表示主键约束。 ● u表示唯一约束。
table_name	character varying(64)	约束相关的表名。
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

17.3.23 DBA_CONS_COLUMNS

DBA_CONS_COLUMNS视图存储关于数据库表中约束字段的信息。需要有系统管理员权限才可以访问。

表 17-123 DBA_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。
position	smallint	表中列的位置。

17.3.24 DBA_IND_COLUMNS

DBA_IND_COLUMNS视图存储关于数据库中所有索引的字段信息。需要有系统管理员权限才可以访问。

表 17-124 DBA_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_name	name	列名。
column_position	smallint	索引中列的位置。

17.3.25 DBA_IND_EXPRESSIONS

DBA_IND_EXPRESSIONS视图存储了数据库中的表达式索引的信息。需要有系统管理员权限才可以访问。

表 17-125 DBA_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

 版权所有 © 华为云计算技术有限公司

17.3.26 DBA_IND_PARTITIONS

DBA_IND_PARTITIONS视图存储数据库中所有索引分区的信息。数据库中每个分区表的每个索引分区（若存在）在DBA_IND_PARTITIONS里都会有一行记录。需要有系统管理员权限才可以访问。

表 17-126 DBA_IND_PARTITIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称。
schema	character varying(64)	索引分区所属分区表索引的模式。
index_name	character varying(64)	索引分区所属分区表索引的名称。
partition_name	character varying(64)	索引分区的名称。
index_partition_usable	boolean	索引分区是否可用。
high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collation、字段数据类型等信息。
def_tablespace_name	name	索引分区的表空间名称。

17.3.27 DBA_INDEXES

DBA_INDEXES视图存储关于数据库下的所有索引信息。需要有系统管理员权限才可以访问。

表 17-127 DBA_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引名称是否为系统生成。
partitioned	character(3)	表示索引是否具有分区表的性质。

17.3.28 DBA_OBJECTS

DBA_OBJECTS视图存储了数据库中所有数据库对象。需要有系统管理员权限才可以访问。

表 17-128 DBA_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。
object_id	oid	对象的OID。
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。
last_ddl_time	timestamp with time zone	对象的最后修改时间。

说明

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

17.3.29 DBA_PART_INDEXES

DBA_PART_INDEXES视图存储数据库中所有分区表索引的信息。需要有系统管理员权限才可以访问。

 版权所有 © 华为云计算技术有限公司

表 17-129 DBA_PART_INDEXES 字段

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称。
schema	character varying(64)	分区表索引的模式。
index_name	character varying(64)	分区表索引的名称。
table_name	character varying(64)	分区表索引所属的分区表名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表索引的索引分区的个数。
def_tablespace_name	name	分区表索引的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

17.3.30 DBA_PART_TABLES

DBA_PART_TABLES视图存储数据中所有分区表的信息。需要有系统管理员权限才可以访问。

表 17-130 DBA_PART_TABLES 字段

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称。
schema	character varying(64)	分区表的模式。
table_name	character varying(64)	分区表的名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表的分区个数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
def_tablespace_name	name	分区表的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

17.3.31 DBA_PROCEDURES

DBA_PROCEDURES视图存储关于数据库下的所有存储过程或函数信息。需要有系统管理员权限才可以访问。

表 17-131 DBA_PROCEDURES 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
object_name	character varying(64)	存储过程或函数名称。
argument_number	smallint	存储过程入参个数。

17.3.32 DBA_SEQUENCES

DBA_SEQUENCES视图存储关于数据库下的所有序列信息。需要有系统管理员权限才可以访问。

表 17-132 DBA_SEQUENCES 字段

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者。
sequence_name	character varying(64)	序列的名称。

17.3.33 DBA_SOURCE

DBA_SOURCE视图存储关于数据库下的所有存储过程或函数信息，且提供存储过程或函数定义的字段。需要有系统管理员权限才可以访问。

表 17-133 DBA_SOURCE 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
name	character varying(64)	存储过程或函数的名称。
text	text	存储过程或函数的定义。

 版权所有 © 华为云计算技术有限公司

17.3.34 DBA_SYNONYMS

DBA_SYNONYMS视图存储关于数据库下的所有同义词信息。需要有系统管理员权限才可以访问。

表 17-134 DBA_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者。
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。
table_name	text	关联对象名。

17.3.35 DBA_TAB_COLUMNS

DBA_TAB_COLUMNS视图存储关于表和视图的字段的信息。数据库里每个表的每个字段都在DBA_TAB_COLUMNS里有一行。需要有系统管理员权限才可以访问。

表 17-135 DBA_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	创建表或视图时列的序号。
data_length	integer	列的字节长度。
comments	text	注释。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2， bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

应用示例

查询指定schema下的所有表字段信息：

```
SELECT * FROM DBA_TAB_COLUMNS WHERE schema = 'public';
```

```
gaussdb=> SELECT * FROM DBA_TAB_COLUMNS WHERE schema = 'public';
  owner   |      table_name       |    column_name    | data_type | column_id | data_length | comments | avg_col_len | nullable | data_precision | data_scale | char_length | schema | kind
----------+-----------------------+-------------------+-----------+-----------+-------------+----------+-------------+----------+----------------+------------+-------------+--------+-------
 dbadmin | customer_address      | ca_address_sk     | int4      |         1 |           4 |          |           4 | n        |                |          0 |           0 | public | table
 dbadmin | hstore_opt_table_demo | t_code            | varchar   |         1 |          20 |          |          11 | y        |                |          0 |          20 | public | table
 dbadmin | test_t5               | age               | int4      |         3 |           4 |          |           4 | y        |                |          0 |           0 | public | table
 dbadmin | hstore_opt_table_demo | t_gisid           | varchar   |         2 |         800 |          |          37 | y        |                |          0 |         800 | public | table
 dbadmin | test_block_rule1      | c1                | int4      |         1 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | test_block_rule2      | c2                | int4      |         2 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | customer_address      | ca_suite_number   | bpchar    |         6 |          -1 |          |          11 | y        |                |          0 |          10 | public | table
 dbadmin | customer              | c_customer_id     | bpchar    |         2 |          -1 |          |             | y        |                |          0 |          16 | public | table
 dbadmin | customer              | c_customer_sk     | int4      |         1 |           4 |          |             | n        |                |          0 |           0 | public | table
 dbadmin | test_t5               | data              | int4      |         2 |           4 |          |           4 | y        |                |          0 |           0 | public | table
 dbadmin | test_block_rule3      | c2                | int4      |         2 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | customer_address      | ca_street_name    | varchar   |         4 |          60 |          |          11 | y        |                |          0 |          60 | public | table
 dbadmin | hstore_opt_table_demo | t_gmid            | varchar   |         4 |          64 |          |          10 | y        |                |          0 |          64 | public | table
 dbadmin | test_block_rule2      | c1                | int4      |         1 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | customer_address      | ca_street_number  | bpchar    |         3 |          -1 |          |          11 | y        |                |          0 |          10 | public | table
 dbadmin | stu_icase_info        | name              | text      |         2 |          -1 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | customer_address      | ca_street_type    | bpchar    |         5 |          -1 |          |          16 | y        |                |          0 |          15 | public | table
 dbadmin | customer_address      | ca_address_id     | bpchar    |         2 |          -1 |          |          17 | n        |                |          0 |          16 | public | table
 dbadmin | hstore_opt_table_demo | t_datatime        | timestamp |         3 |           8 |          |           8 | y        |                |          0 |           0 | public | table
 dbadmin | store_returns         | sr_item_sk        | int4      |         1 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | store_returns         | sr_customer_sk    | int4      |         2 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | test_block_rule3      | c1                | int4      |         1 |           4 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | test_t5               | id                | int4      |         1 |           4 |          |           4 | n        |                |          0 |           0 | public | table
 dbadmin | stu_icase_info        | id                | int8      |         1 |           8 |          |             | y        |                |          0 |           0 | public | table
 dbadmin | test_block_rule1      | c2                | int4      |         2 |           4 |          |             | y        |                |          0 |           0 | public | table
(25 rows)
```

17.3.36 DBA_TAB_COMMENTS

DBA_TAB_COMMENTS视图存储关于数据库下的所有表和视图的注释信息。需要有系统管理员权限才可以访问。

表 17-136 DBA_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

17.3.37 DBA_TAB_PARTITIONS

DBA_TAB_PARTITIONS视图提供数据库中所有分区的信息。

 版权所有 © 华为云计算技术有限公司

表 17-137 DBA_TAB_PARTITIONS 字段

名称	类型	描述
table_owner	character varying(64)	分区所在表的所有者。
schema	character varying(64)	分区表模式。
table_name	character varying(64)	表名。
partition_name	character varying(64)	分区的名称。
high_value	text	范围分区的上边界，或列表分区的边界值集合。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	范围分区的上边界，或列表分区的边界值集合。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collation、字段数据类型等信息。
tablespace_name	name	分区所在表空间的名称。

应用示例

查看分区表的分区信息：

```
DROP TABLE IF EXISTS web_returns_p1;
CREATE TABLE web_returns_p1
(
    wr_returned_date_sk       integer,
    wr_returned_time_sk       integer,
    wr_item_sk                integer NOT NULL,
    wr_refunded_customer_sk   integer
)
WITH (orientation = column)
DISTRIBUTE BY HASH (wr_item_sk)
PARTITION BY RANGE (wr_returned_date_sk)
(
    PARTITION p2016 VALUES LESS THAN(20161231),
    PARTITION p2017 VALUES LESS THAN(20171231),
    PARTITION p2018 VALUES LESS THAN(20181231),
    PARTITION p2019 VALUES LESS THAN(20191231),
    PARTITION p2020 VALUES LESS THAN(maxvalue)
);

SELECT * FROM dba_tab_partitions where table_name='web_returns_p1';
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=> SELECT * FROM dba_tab_partitions where table_name='web_returns_p1';
 table_owner | schema |   table_name   | partition_name | high_value | pretty_high_value | tablespace_name
-------------+--------+----------------+----------------+------------+-------------------+-----------------
 dbadmin     | public | web_returns_p1 | p2016          | 20161231   | 20161231          | cu_obs_tbs
 dbadmin     | public | web_returns_p1 | p2017          | 20171231   | 20171231          | cu_obs_tbs
 dbadmin     | public | web_returns_p1 | p2018          | 20181231   | 20181231          | cu_obs_tbs
 dbadmin     | public | web_returns_p1 | p2019          | 20191231   | 20191231          | cu_obs_tbs
 dbadmin     | public | web_returns_p1 | p2020          | MAXVALUE   | MAXVALUE          | cu_obs_tbs
(5 rows)
```

17.3.38 DBA_TABLES

DBA_TABLES视图存储关于数据库下的所有表信息。需要有系统管理员权限才可以访问。

表 17-138 DBA_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间的名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

17.3.39 DBA_TABLESPACES

DBA_TABLESPACES视图存储有关可用的表空间的信息。需要有系统管理员权限才可以访问。

表 17-139 DBA_TABLESPACES 字段

名称	类型	描述
tablespace_name	character varying(64)	表空间的名称。

版权所有 © 华为云计算技术有限公司

17.3.40 DBA_TRIGGERS

DBA_TRIGGERS视图存储关于数据库内的触发器信息。需要有系统管理员权限才可以访问。

表 17-140 DBA_TRIGGERS 字段

名称	类型	描述
trigger_name	character varying(64)	触发器名称。
table_name	character varying(64)	定义触发器的表的名称。
table_owner	character varying(64)	定义触发器的表的所有者。

17.3.41 DBA_VIEWS

DBA_VIEWS视图存储关于数据库内的视图信息。需要有系统管理员权限才可以访问。

表 17-141 DBA_VIEWS 字段

名称	类型	描述
owner	character varying(64)	视图的所有者。
view_name	character varying(64)	视图的名称。

17.3.42 DUAL

DUAL视图是数据库根据数据字典自动创建的，它只有一个文本字段，且只有一行，用于保存表达式计算结果。任何用户都可以访问它。

表 17-142 DUAL 字段

名称	类型	描述
dummy	text	表达式计算结果。

17.3.43 GET_ALL_TSC_INFO

GET_ALL_TSC_INFO视图用于重新获取所有节点TSC信息。该视图仅8.2.1及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 17-143 GET_ALL_TSC_INFO 字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

17.3.44 GET_TABLESPACE_OBS_FILE

GET_TABLESPACE_OBS_FILE视图用于获取当前节点上一个表空间内的**PG_DB_FILE**系统表上的全部文件信息。该视图仅9.1.1.100及以上集群版本支持。

表 17-144 GET_TABLESPACE_OBS_FILE 字段

名称	类型	描述
db_oid	oid	数据库OID。
table_space_oid	oid	表空间OID。
relfilenode	oid	表的物理目录ID。
file_id	bigint	文件ID。
file_type	text	文件类型。
file_size	bigint	文件大小，单位Byte。
file_path	text	文件绝对路径。

17.3.45 GET_TSC_INFO

GET_TSC_INFO视图用于重新获取当前节点TSC信息。该视图仅8.2.1及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 17-145 GET_TSC_INFO 字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

17.3.46 GLOBAL_COLUMN_TABLE_IO_STAT

GLOBAL_COLUMN_TABLE_IO_STAT视图提供当前数据库所有列存表的IO统计数据。其字段的名称、类型和顺序与GS_COLUMN_TABLE_IO_STAT视图相同，具体的字段请参考**表17-146**。各统计字段为所有节点对应字段之和。

表 17-146 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

17.3.47 GLOBAL_REDO_STAT

GLOBAL_REDO_STAT视图显示集群中所有节点上XLOG重做过程中的统计信息总和。除avgiotim（表示所有节点平均的重做写入时间）外，其余字段名称和

 版权所有 © 华为云计算技术有限公司

PV_REDO_STAT视图相同，但其余字段含义为各节点上PV_REDO_STAT视图同名字段的数值之和。

需要有系统管理员权限才可以访问此视图。

表 17-147 GLOBAL_REDO_STAT 字段

名称	类型	描述
phywrts	bigint	所有节点物理写次数之和。
phyblkwrt	bigint	所有节点物理写块数之和。
writetim	bigint	所有节点物理写消耗时间之和。
avgiotim	bigint	所有节点平均的重做写入时间。
lstiotim	bigint	所有节点上一次写入时间之和。
miniotim	bigint	所有节点最小写入时间之和。
maxiowtm	bigint	所有节点最大写入时间之和。

17.3.48 GLOBAL_REL_IOSTAT

GLOBAL_REL_IOSTAT视图显示集群中所有节点上磁盘读写统计信息的总和。其各字段的名称与GS_REL_IOSTAT视图相同，但含义为各节点上GS_REL_IOSTAT视图同名字段的数值之和。

需要有系统管理员权限才可以访问此视图。

表 17-148 GLOBAL_REL_IOSTAT 字段

名称	类型	描述
phyrds	bigint	所有节点读磁盘次数之和。
phywrts	bigint	所有节点写磁盘次数之和。
phyblkrd	bigint	所有节点读磁盘页数之和。
phyblkwrt	bigint	所有节点写磁盘页数之和。

17.3.49 GLOBAL_ROW_TABLE_IO_STAT

GLOBAL_ROW_TABLE_IO_STAT视图提供当前数据库所有行存表的IO统计数据。其字段的名称、类型和顺序与GS_ROW_TABLE_IO_STAT视图相同，具体的字段请参考表17-149。各统计字段为所有节点对应字段之和。

 版权所有 © 华为云计算技术有限公司

表 17-149 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

17.3.50 GLOBAL_STAT_DATABASE

GLOBAL_STAT_DATABASE视图显示集群中所有节点上数据库的状态和统计信息之和。

- CN上查询GLOBAL_STAT_DATABASE视图，返回的结果除stats_reset字段（当前CN上的状态重置时间）之外，其余字段表示在集群内相关节点上的数值之和。需注意，因GLOBAL_STAT_DATABASE视图中各字段的逻辑含义不同，求和的范围也有所不同。
- 在DN上查询视图GLOBAL_STAT_DATABASE，所得结果与**表17-150**相同。

表 17-150 GLOBAL_STAT_DATABASE 字段

名称	类型	描述	求和范围
datid	oid	数据库OID。	-
datname	name	数据库名。	-
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。	CN
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。	CN
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。	CN
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。	DN

版权所有 © 华为云计算技术有限公司

名称	类型	描述	求和范围
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。	DN
tup_returned	bigint	当前节点上该数据库查询返回的行数。	DN
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。	DN
tup_inserted	bigint	当前节点上该数据库插入的行数。	DN
tup_updated	bigint	当前节点上该数据库更新的行数。	DN
tup_deleted	bigint	当前节点上该数据库删除的行数。	DN
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。请参见PG_STAT_DATABASE_CONFLICTS获取更多信息。	CN和DN
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。	DN
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。	DN
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。	CN和DN
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。	DN
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。	DN
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。	-

17.3.51 GLOBAL_TABLE_CHANGE_STAT

GLOBAL_TABLE_CHANGE_STAT视图显示当前数据库中所有表格（不包括外表）变更情况。表示次数的各字段为实例启动以来的累计值。

版权所有 © 华为云计算技术有限公司

表 17-151 GLOBAL_TABLE_CHANGE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动Vacuum的时间。
vacuum_count	bigint	手动Vacuum的次数。为各CN节点上次数之和。
last_autovacuum	timestamp with time zone	最后一次自动Vacuum的时间。
autovacuum_count	bigint	自动Vacuum的次数。为各CN节点上次数之和。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
analyze_count	bigint	分析（包括手动和自动）的次数。由于analyze会同时在所有节点上进行，该字段为所有CN节点上的最大值。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。为各CN节点上次数之和。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

17.3.52 GLOBAL_TABLE_STAT

GLOBAL_TABLE_STAT视图显示当前数据库中所有表格（不包括外表）的统计信息。除live_tuples和dead_tuples为当前实时值外，其余各统计字段为实例启动以来的累计值。

表 17-152 GLOBAL_TABLE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
distribute_mode	char	表的分布方式，与系统表pgxc_class中的pclocatortype字段含义相同。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
seq_scan	bigint	顺序扫描的次数。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。
index_scan	bigint	索引扫描的次数。
index_tuple_read	bigint	索引扫描的行数。
tuple_inserted	bigint	插入的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
tuple_updated	bigint	更新的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
tuple_deleted	bigint	删除的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
tuple_hot_updated	bigint	热更新的行数。如果是复制表，显示各节点最大值；如果是分布表，显示各节点之和。
live_tuples	bigint	活元组数量。显示各节点最大值；如果是分布表，显示各节点之和。 只适用行存表。
dead_tuples	bigint	死元组数量。显示各节点最大值；如果是分布表，显示各节点之和。 只适用行存表。

17.3.53 GLOBAL_WORKLOAD_SQL_COUNT

GLOBAL_WORKLOAD_SQL_COUNT视图显示集群中所有Workload控制组内SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 17-153 GLOBAL_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

17.3.54 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME

GLOBAL_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

表 17-154 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

 版权所有 © 华为云计算技术有限公司

17.3.55 GLOBAL_WORKLOAD_TRANSACTION

GLOBAL_WORKLOAD_TRANSACTION视图提供集群所有CN上WORKLOAD控制组相关的事务信息的总和。需要有系统管理员权限才可以访问。该视图仅在资源实时监控功能开启，即enable_resource_track为on时有效。

表 17-155 GLOBAL_WORKLOAD_TRANSACTION 字段

名称	类型	描述
workload	name	WORKLOAD控制组名称。
commit_counter	bigint	各CN上提交次数总和。
rollback_counter	bigint	各CN上回滚次数总和。
resp_min	bigint	集群总体最小响应时间。
resp_max	bigint	集群总体最大响应时间。
resp_avg	bigint	各CN上平均响应时间。
resp_total	bigint	各CN上响应时间总和。

17.3.56 GS_ALL_CONTROL_GROUP_INFO

GS_ALL_CONTROL_GROUP_INFO视图显示数据库内所有的控制组信息。

表 17-156 GS_ALL_CONTROL_GROUP_INFO 字段

名称	类型	描述
name	text	控制组的名称。
type	text	控制组的类型。
gid	bigint	控制组ID。
classgid	bigint	Workload所属Class的控制组ID。
class	text	Class控制组。
workload	text	Workload控制组。
shares	bigint	控制组分配的CPU资源配额。
limits	bigint	控制组分配的CPU资源限额。
wdlevel	bigint	Workload控制组层级。
cpucores	text	控制组使用的CPU核的信息。

版权所有 © 华为云计算技术有限公司

17.3.57 GS_BLOCKLIST_QUERY

GS_BLOCKLIST_QUERY视图用于查询作业黑名单信息和异常信息，此视图是由系统表GS_BLOCKLIST_QUERY和GS_WLM_SESSION_INFO关联所得，同时对查询结果进行了去重筛选，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长时间。

说明

- GS_BLOCKLIST_QUERY视图的schema为pg_catalog。
- GS_BLOCKLIST_QUERY视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个unique_sql_id可能会对应一个或多个查询。

表 17-157 GS_BLOCKLIST_QUERY 字段

名字	类型	引用	描述
unique_sql_id	bigint	-	基于查询解析树生成的查询ID。
block_list	boolean	-	查询作业是否属于黑名单。
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。
query	text	-	执行的查询语句。

17.3.58 GS_BLOCKLIST_SQL

GS_BLOCKLIST_SQL视图用于查询作业黑名单信息和异常信息，此视图是由系统表GS_BLOCKLIST_SQL和GS_WLM_SESSION_INFO关联所得，同时对查询结果进行了去重筛选，因此在GS_WLM_SESSION_INFO表较大的情况下，查询可能需要消耗较长时间。

该视图仅9.1.0.200及以上集群版本支持。

说明

- GS_BLOCKLIST_SQL视图存储的schema为pg_catalog。
- GS_BLOCKLIST_SQL视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 通常对于DML语句，在计算sql_hash的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。因此一个sql_hash可能会对应一个或多个查询。

表 17-158 GS_BLOCKLIST_SQL 字段

名字	类型	引用	描述
sql_hash	text	-	基于查询解析树生成的sql_hash。
block_list	boolean	-	查询作业是否属于黑名单。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
except_num	integer	-	查询作业异常次数。
except_time	timestamp	-	查询作业最近一次异常时间。
query	text	-	执行的查询语句。

17.3.59 GS_CLUSTER_RESOURCE_INFO

GS_CLUSTER_RESOURCE_INFO视图显示的是所有DN资源的汇总信息。

表 17-159 GS_CLUSTER_RESOURCE_INFO 字段

名称	类型	描述
min_mem_util	integer	DN最小内存使用率。
max_mem_util	integer	DN最大内存使用率。
min_cpu_util	integer	DN最小CPU使用率。
max_cpu_util	integer	DN最大CPU使用率。
min_io_util	integer	DN最小IO使用率。
max_io_util	integer	DN最大IO使用率。
used_mem_rate	integer	物理节点最大内存使用率。

17.3.60 GS_COLUMN_TABLE_IO_STAT

GS_COLUMN_TABLE_IO_STAT视图显示当前数据库中所有列存表在当前节点上的IO情况。各统计字段为实例启动以来的累计值。

表 17-160 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

17.3.61 GS_LSC_DBSTAT_INFO

GS_LSC_DBSTAT_INFO视图，用于获取当前节点Local RelCache、PartCache的统计信息，例如RelCache和PartCache缓存命中情况及LRU淘汰情况。该视图仅9.1.1.100及以上集群版本支持。

表 17-161 GS_LSC_DBSTAT_INFO 字段

名称	类型	描述
thread_id	bigint	线程号。
thread_name	text	线程名称。
rel_searches	bigint	RelCache查找次数。
rel_hits	bigint	RelCache命中次数。
rel_miss	bigint	RelCache未命中，重新加载次数。
rel_count	bigint	缓存中relation的数量。
rel_swapout_count	bigint	relation被LRU淘汰的数量。
rel_invalidate_count	bigint	RelCache全失效次数。
part_searches	bigint	PartCache查找次数。
part_hits	bigint	PartCache命中次数。
part_miss	bigint	PartCache未命中，重新加载次数。
part_count	bigint	缓存中partition的数量。
part_swapout_count	bigint	partition被LRU淘汰的数量。
part_invalidate_count	bigint	PartCache全失效次数。

应用示例

查询当前视图，统计Local RelCache、PartCache的统计信息。

```
SELECT * FROM GS_LSC_DBSTAT_INFO;
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=> SELECT * FROM GS_LSC_DBSTAT_INFO;
    thread_id     |      thread_name      | rel_searches | rel_hits  | rel_miss | rel_count | rel_swapout_count | rel_invalidate_count | part_searches | part_hits | part_miss | part_count | part_swapout_count | part_invalidate_count
------------------+-----------------------+--------------+-----------+----------+-----------+-------------------+----------------------+---------------+-----------+-----------+------------+--------------------+-----------------------
 281477242769904 | postgres              |        44196 |     20693 |      564 |       206 |                 0 |                    0 |           538 |       269 |        70 |          2 |                  0 |                     0
 281477242775832 | postgres              |        50270 |     38642 |      199 |       187 |                 0 |                    0 |            56 |         0 |         0 |          0 |                  0 |                     0
 281477242777200 | postgres              |          302 |       281 |      211 |       210 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242774008 | postgres              |         7521 |      7313 |      278 |       227 |                 0 |                    0 |            98 |        62 |         6 |          0 |                  0 |                     0
 281477242753944 | postgres              |           62 |        52 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242774920 | postgres              |         5147 |      5093 |      219 |       218 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242748472 | WlParbiter            |       857223 |    731440 |      197 |        22 |                 0 |                    1 |           789 |         0 |         0 |          0 |                  0 |                     1
 281477242761696 | postgres              |          355 |       349 |      200 |       200 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242752120 | postgres              |       126578 |       788 |      209 |        29 |                 0 |                    1 |           798 |         0 |         0 |          0 |                  0 |                     1
 281477242777656 | postgres              |          206 |       197 |      203 |       203 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242756224 | postgres              |      1389046 |   1263243 |      240 |        32 |                 0 |                    1 |           798 |         0 |         0 |          0 |                  0 |                     1
 281477242757136 | postgres              |         1475 |       908 |      369 |        34 |                 0 |                   13 |             0 |         0 |         0 |          0 |                  0 |                    13
 281477242757592 | postgres              |           54 |        51 |      197 |       197 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242774464 | postgres              |           45 |        43 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242754856 | postgres              |         1589 |      1559 |      216 |       216 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242751664 | postgres              |    116174729 | 116048858 |      270 |        48 |                 0 |                    1 |           841 |         0 |         0 |          0 |                  0 |                     1
 281477242750296 | file_recyclebin       |       127148 |      1271 |      198 |        22 |                 0 |                    1 |           841 |         0 |         0 |          0 |                  0 |                     1
 281477242749840 | BucketDispatcher      |           25 |        23 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242750752 | GlobalDeadLockDetector |           25 |        23 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242748928 |                       |          433 |       393 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242747104 | WLMCollectWorker      |     32989566 |  32863608 |      474 |        93 |                 0 |                    1 |         82593 |     81716 |        54 |         34 |                  0 |                     1
 281477242749384 |                       |          432 |       392 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242753488 | postgres              |       126279 |       569 |      196 |        21 |                 0 |                    3 |           784 |         0 |         0 |          0 |                  0 |                     3
 281477242747560 | WLMCalSpaceInfo       |    191961351 | 191835491 |      304 |        38 |                 0 |                    1 |           841 |         0 |         0 |          0 |                  0 |                     1
 281477242748016 | WLMmonitor            |      1336085 |   1210216 |      354 |        37 |                 0 |                    2 |         21219 |     20363 |        17 |          6 |                  0 |                     2
 281477242745736 | WDRSnapshot           |          433 |       393 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242745200 | AuditSr               |          432 |       392 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
 281477242744368 | JobScheduler          |      1348819 |   1222808 |      197 |        22 |                 0 |                    1 |           841 |         0 |         0 |          0 |                  0 |                     1
 281477242743912 | AutoVacLauncher       |       181788 |    181775 |        4 |         4 |                 0 |                    1 |             0 |         0 |         0 |          0 |                  0 |                     1
 281477242744824 | StatCollector         |          433 |       393 |      196 |       196 |                 0 |                    0 |             0 |         0 |         0 |          0 |                  0 |                     0
(30 rows)
```

17.3.62 GS_GET_KVSTORE_DB_COMP_STATS

GS_GET_KVSTORE_DB_COMP_STATS视图用于获取RocksDB内部compaction的统计信息，通过汇总当前集群各节点各RocksDB实例的compaction情况（包括sst文件层级，文件数量，文件大小等），辅助判断是否存在读写压缩瓶颈。该视图仅9.1.2及以上集群版本支持，且需要开启LSM引擎（enable_lsm_engine参数设置为on）才可访问。

表 17-162 GS_GET_KVSTORE_DB_COMP_STATS 字段

名称	类型	描述
node	text	所在DN节点名称。
cf_name	text	所在列族名称。
bucket_id	bigint	所在bucket分区ID。
kvdb_id	bigint	所在RocksDB实例ID，与bucket分区ID相对应。
level	text	L表示文件级别，Sum表示L0到各级别的数据总和，Int（Interval）表示周期内采集的数据和。
files	text	有两个值（a/b），a表示在当前层级的文件数，b表示当前层级正在做压缩的文件数。
size	text	当前层级的文件大小。
score	bigint	压缩得分，当大于1则需要压缩。非L0层级，分数计算公式为（当前层级大小）/（层级的最大大小），对L0层级，分数是当前文件数和压缩触发值的比。
read_gb	bigint	层级N和N+1的压缩中读取数据的字节总数。包括从N层读的数据和N+1层读的数据，单位GB。
rn_gb	bigint	层级N和层级N+1的压缩中读取N层的字节数。
rnp1_gb	bigint	Rnp1(GB): 层级N和N+1的压缩中读取N+1层的字节数。
write_gb	bigint	层级N和N+1的压缩中写数据的字节总数，单位GB。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
wnew_gb	bigint	写入N+1层的新数据的总字节数, 通过 (写入 N+1层的总字节数) - (压缩时从N+1层读取的字节总数)计算。
moved_gb	bigint	压缩时记录N+1层的字节总数，除了更新manifest文件从X层到Y层外，没有任何I/O操作。
w_amp	bigint	(写入N+1层的总字节数) / (从N层读取的总字节数). 这是N层到N+1层的写放大。
rd_mbps	bigint	压缩N到N+1层时的读取速度MB/s，通过（Read（GB）*1024/compaction）周期计算得出。
wr_mbps	bigint	压缩N到N+1层时的写入速度MB/s，见rd_mbps。
comp_sec	bigint	压缩N到N+1层时的花费时间，单位s。
comp_cpu_sec	bigint	压缩N到N+1层 cpu merge 花费时间，单位s。
comp_cnt	bigint	压缩N到N+1层的压缩任务数。
avg_sec	bigint	压缩N到N+1层每个压缩任务的平均耗时。
key_in	text	压缩期间写入的key数。
key_drop	text	压缩期间被丢弃的 Key 数。
rblob_gb	bigint	从blob文件中读取的字节数，单位GB。
wblob_gb	bigint	压缩期间写入表文件的总字节数 ，单位GB。

应用示例

使用GS_GET_KVSTORE_DB_COMP_STATS视图查询RocksDB实例compaction状态。

```
select * from gs_get_kvstore_db_comp_stats('write_group');
   node    | cf_name | bucket_id |      kvdb_id       | level | files |   size   | score | read_gb | rn_gb | rnp1_gb |
write_gb | wnew_gb | moved_gb | w_amp | rd_mbps | wr_mbps | comp_sec | comp_cpu_sec | comp_cnt |
avg_sec | key_in | key_drop | rblob_gb | wblob_gb
-----------+---------+-----------+--------------------+-------+-------+----------+-------+---------+-------+---------
+----------+---------+----------+-------+---------+---------+----------+--------------+----------+---------+--------
+----------+----------+----------
 datanode2 | index   |       15 | 931330329195134991 | L0    | 1/0   | 7.97 KB  |     0 |       0 |     0 |       0
|        0 |       0 |        0 |     1 |       0 |       0 |        4 |            0 |        7 |       0 | 118k   | 118      |        0 |
0
 datanode2 | index   |       15 | 931330329195134991 | L4    | 0/0   | 0.00 KB  |     0 |       0 |     0 |       0
|        0 |       0 |        0 |     0 |       0 |       0 |        0 |            0 |        1 |       0 | 69M    | 69K      |        0 |
0
 datanode2 | index   |       15 | 931330329195134991 | Sum   | 1/0   | 7.97 KB  |     0 |       0 |     0 |       0
|        0 |       0 |        0 |     1 |       0 |       0 |        4 |            0 |        8 |       0 | 187    | 187      |        0 |
0
 datanode2 | cudesc  |       15 | 931330329195134991 | Int   | 0/0   | 0.00 KB  |     0 |       0 |     0 |       0
|        0 |       0 |        0 |     1 |       0 |       0 |        1 |            0 |        2 |       0 | 117    | 117      |        0 |
0
 datanode2 | delta   |       15 | 931330329195134991 | L0    | 0/0   | 0.00 KB  |     0 |       0 |     0 |       0 |
0 |       0 |        0 |     1 |       0 |       0 |        2 |            0 |        4 |       0 | 2      | 2        |        0 |       0
 datanode1 | index   |       14 | 931330329195134990 | Int   | 0/0   | 0.00 KB  |     0 |       0 |     0 |       0 |
0 |       0 |        0 |     1 |       0 |       0 |        1 |            0 |        2 |       0 | 101    | 101      |        0 |       0
```


版权所有 © 华为云计算技术有限公司

```
 datanode1 | delta   |       14 | 931330329195134990 | L0    | 1/0   | 1.04 KB |     0 |       0 |     0 |       0 |
0 |       0 |       0 |     1 |       0 |       0 |       1 |           0 |       3 |       0 | 1       | 1       |       0 |       0
 datanode1 | delta   |       14 | 931330329195134990 | Sum   | 1/0   | 1.04 KB |     0 |       0 |     0 |       0
|       0 |       0 |       0 |     1 |       0 |       0 |       1 |           0 |       3 |       0 | 1       | 1       |       0 |       0
 datanode1 | delta   |       14 | 931330329195134990 | Int   | 0/0   | 0.00 KB |     0 |       0 |     0 |       0 |
0 |       0 |       0 |   993 |       0 |       0 |       0 |           0 |       1 |       0 | 1       | 1       |       0 |       0
 datanode1 | cudesc  |       14 | 931330329195134990 | L0    | 0/0   | 0.00 KB |     0 |       0 |     0 |       0
|       0 |       0 |       0 |     1 |       0 |       0 |       3 |           0 |       4 |       0 | 2       | 2       |       0 |       0
```

17.3.63 GS_GET_KVSTORE_DB_STALL_DETAIL_STATS

GS_GET_KVSTORE_DB_STALL_DETAIL_STATS视图用于获取RocksDB内部写入失速情况的信息，通过汇总当前集群各节点各RocksDB实例的写入减慢或写入暂停情况，辅助判断是否存在写入瓶颈。该视图仅9.1.2及以上集群版本支持，且需要开启LSM引擎（enable_lsm_engine参数设置为on）才可访问。

表 17-163 GS_GET_KVSTORE_DB_STALL_DETAIL_STATS 字段

名称	类型	描述
node	text	当前数据所在DN节点名称。
cf_name	text	当前数据所在列族名称。
bucket_id	bigint	当前bucket分区ID。
kvdb_id	bigint	所在RocksDB实例ID，与bucket分区ID相对应。
cf_l0_file_count_delays_with_ongoing_comp	bigint	正在进行L0压缩时，由l0文件计数限制导致的写入速度减慢次数。
cf_l0_file_count_stop_with_ongoing_comp	bigint	正在进行L0压缩时，由于L0文件计数限制导致写入停止次数。
l0_file_count_limit_delays	bigint	当 level 0 的SST文件达到level0_slowdown_writes_trigger的时候，写入进入失速状态，单位次。
l0_file_count_limit_stops	bigint	当level0的SST文件达到level0_stop_writes_trigger，写入完全停止，单位次。
memtable_limit_delays	bigint	如果 max_write_buffer_number 大于3，并且等待落盘的memtable数量大于或者等于max_write_buffer_number - 1，写入进入失速状态，单位次。
memtable_limit_stops	bigint	当等待落盘的memtable数量大于或等于max_write_buffer_number时，写入会完全停止，以等待落盘完成，单位次。
pending_compaction_bytes_delays	bigint	当预计的等待压缩的字节数达到soft_pending_compaction_bytes，写失速开始，单位次。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
pending_compaction_bytes_stops	bigint	当预计的等待压缩字节数达到hard_pending_compaction_bytes，写入会完全停止，等待压缩，单位次。
total_delays	bigint	发生delays次数和。
total_stops	bigint	发生stops次数和。
cumulative_stall_time	text	累计失速时间。
cumulative_stall_percent	bigint	累计失速百分比。
interval_stall_time	text	间隔周期内失速时间。
interval_stall_percent	bigint	间隔周期内失速百分比。
write_buffer_manager_limit_stops	bigint	由于写缓存限制导致的停止，单位次。

应用示例

使用GS_GET_KVSTORE_DB_STALL_DETAIL_STATS视图查询RocksDB实例后台写入失速的情况。

```
select * from  GS_GET_KVSTORE_DB_STALL_DETAIL_STATS ('write_group');
| cf_name | bucket_id |       kvdb_id        | cf_l0_file_count_delays_with_ongoing_comp |
cf_l0_file_count_stop_with_ongoing_comp | l0_file_count_limit_delaysbigint | l0_file_count_limit_stopsbigint |
memtable_limit_delays | memtable_limit_stops | pending_compactio
n_bytes_delays | pending_compaction_bytes_stops | total_delays | total_stops | cumulative_stall_time |
cumulative_stall_percent | interval_stall_time | interval_stall_percent | write_buffer_manager_limit_stops
-----------+---------+-----------+-------------------+-----------------------------------------
+-----------------------------------------+---------------------------------+--------------------------------
+----------------------+----------------------+------------------
---------------+---------------------------------+--------------+-------------+-----------------------
+--------------------------+--------------------+-------------------------+---------------------------------
   datanode2 | index   |        9 | 931330329192550409 |                                 0
|                                0 |                            0 |                           0 |                0 |
0 |
            0 |                          0 |          0 |          0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                  0 |                          0
   datanode2 | delta   |        9 | 931330329192550409 |                                 0
|                                0 |                            0 |                           0 |                0 |
0 |
            0 |                          0 |          0 |          0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                  0 |                          0
   datanode2 | cudesc  |        9 | 931330329192550409 |                                  0
|                                0 |                            0 |                           0 |                0 |
0 |
            0 |                          0 |          0 |          0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                  0 |                          0
   datanode2 | index   |        1 | 931330329192550401 |                                 0
|                                0 |                            0 |                           0 |                0 |
0 |
            0 |                          0 |          0 |          0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                  0 |                          0
   datanode2 | delta   |        1 | 931330329192550401 |                                 0
|                                0 |                            0 |                           0 |                0 |
0 |
            0 |                          0 |          0 |          0 | 00:00:0.000 H:M:S     |                    0 |
```

 版权所有 © 华为云计算技术有限公司

```
00:00:0.000 H:M:S   |                    0 |                              0
   datanode2 | cudesc  |        1 | 931330329192550401 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
   datanode1 | index   |       14 | 931330329192550414 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
   datanode1 | delta   |       14 | 931330329192550414 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
   datanode1 | cudesc  |       14 | 931330329192550414 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
   datanode1 | index   |        6 | 931330329192550406 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
   datanode1 | delta   |        6 | 931330329192550406 |                                   0
|                              0 |                              0 |                         0 |                0 |
0 |
             0 |                         0 |            0 |           0 | 00:00:0.000 H:M:S     |                    0 |
00:00:0.000 H:M:S   |                    0 |                              0
```

17.3.64 GS_GET_KVSTORE_STATS

GS_GET_KVSTORE_STATS视图用于获取RocksDB内部的统计信息，通过汇总当前集群各节点各RocksDB实例的后台统计信息并直接展示，用于全面查看RocksDB内部的读写情况。该视图仅9.1.2及以上集群版本支持，且需要开启LSM引擎（enable_lsm_engine参数设置为on）才可访问。

表 17-164 GS_GET_KVSTORE_STATS 字段

名称	类型	描述
node	text	当前数据所在DN节点名称。
cf_name	text	当前数据所在列族名称。
bucket_id	bigint	当前bucket分区ID。
kvdb_id	bigint	所在RocksDB实例ID，与bucket分区ID相对应。
stats	text	RocksDB获取的Stats全部数据。

应用示例

使用GS_GET_KVSTORE_STATS视图查询RocksDB实例的后台统计信息。

```
select * from gs_get_kvstore_stats('write_group');
   node    | cf_name | bucket_id |       kvdb_id
|

stats
```


版权所有 © 华为云计算技术有限公司

```
-----------+---------+-----------+--------------------
+-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------
 datanode2 | index   |       15 | 931330329195134991
|


                                      +
       |       |       |             | ** Compaction Stats [index]
**


                 +
       |       |       |             | Level    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB)
Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec)
KeyIn KeyDrop Rblob(GB)
Wblob(GB)

                        +
       |       |       |             |
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------

                                                                                                                                                  +
       |       |       |             |  L0      1/0    7.97 KB   0.2      0.0     0.0      0.0       0.0      0.0
0.0   1.1     0.0     0.0     3.93           0.11         7    0.561     118    118       0.0
0.0

                 +
       |       |       |             |  L4      0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0
0.0   0.3     0.0     0.0     0.62           0.00         1    0.618      69     69       0.0
0.0

                 +
       |       |       |             |  Sum     1/0    7.97 KB   0.0      0.0     0.0      0.0       0.0      0.0
0.0   1.2     0.0     0.0     4.55           0.12         8    0.568     187    187       0.0
0.0

                 +
       |       |       |             |  Int     0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0
0.0   1.1     0.0     0.0     1.13           0.09         2    0.564     117    117       0.0
0.0

                 +
       |       |       |
|


                                      +
       |       |       |             | ** Compaction Stats [index]
**


                 +
       |       |       |             | Priority    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB)
Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec)
KeyIn KeyDrop Rblob(GB)
Wblob(GB)

                        +
       |       |       |             |
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------

                                                                                                                                                  +
```


版权所有 © 华为云计算技术有限公司

```
      |       |       |               | Low      0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0     -0.0
0.0   0.0     0.0     0.0     1.85            0.01         3    0.616     187    187       0.0
0.0

                  +
      |       |       |               | High     0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0
0.0   0.0     0.0     0.0     2.70            0.11         5    0.540     0      0       0.0
0.0

                  +
      |       |       |
|


                                      +
      |       |       |               | Blob file count: 0, total size: 0.0 GB, garbage size: 0.0 GB, space amp:
0.0

                                                                                                                +
      |       |       |
|


                                      +
      |       |       |               | Uptime(secs): 248.6 total, 44.2
interval

                 +
      |       |       |               | Flush(GB): cumulative 0.000, interval
0.000

              +
      |       |       |               | AddFile(GB): cumulative 0.000, interval
0.000

            +
      |       |       |               | AddFile(Total Files): cumulative 0, interval
0

       +
      |       |       |               | AddFile(L0 Files): cumulative 0, interval
0

         +
      |       |       |               | AddFile(Keys): cumulative 0, interval
0

            +
      |       |       |               | Cumulative compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read,
0.00 MB/s read, 4.5
seconds

                                                                                                          +
      |       |       |               | Interval compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read,
0.00 MB/s read, 1.1
seconds

                                                                                                           +
      |       |       |               | Write Stall (count): cf-l0-file-count-limit-delays-with-ongoing-
compaction: 0, cf-l0-file-count-limit-stops-with-ongoing-compaction: 0, l0-file-count-limit-delays: 0, l0-file-
count-limit-stops: 0, memtable-limit-delays: 0, memtable-limit-stops: 0, pending-compaction-bytes-delays:
0, pending-compaction-bytes-stops: 0, total-delays: 0, total-stops: 0, Block cache
LRUCache@0x7f35ab495090#37089 capacity: 1.00 GB usage: 62.17 KB table_size: 1024 occupancy: 40704
```


版权所有 © 华为云计算技术有限公司

```
collections: 1 last_copies: 31 last_secs: 0.00043 secs_since: 75 +
       |       |       |               | Block cache entry stats(count,size,portion): DataBlock(65,38.83
KB,0.00370294%) FilterBlock(20,2.34 KB,0.000223517%) IndexBlock(66,6.75 KB,0.00064373%) Misc(1,0.00
KB,0%)

                                    +
       |       |       |
|


                                     +
       |       |       |               | ** File Read Latency Histogram By Level [index]
**


   +
       |       |       |               | ** Level 0 read latency histogram
(micros):


                +
       |       |       |               | Count: 10 Average: 76223.2000  StdDev:
77239.96


             +
       |       |       |               | Min: 7  Median: 110.0000  Max:
188425


                  +
       |       |       |               | Percentiles: P50: 110.00 P75: 147500.00 P99: 188425.00 P99.9:
188425.00 P99.99:
188425.00

                                                                                                    +
       |       |       |               |
-------------------------------------------------------


                                        +
       |       |       |               | (      6,     10 ]       1  10.000%  10.000%
##


    +
       |       |       |               | (     22,     34 ]       2  20.000%  30.000%
####


     +
       |       |       |               | (     51,     76 ]       1  10.000%  40.000%
##


    +
       |       |       |               | (     76,    110 ]       1  10.000%  50.000%
##


    +
       |       |       |               | (  110000,  170000 ]       4  40.000%  90.000%
########


       +
       |       |       |               | (  170000,  250000 ]       1  10.000% 100.000%
##
```

 版权所有 © 华为云计算技术有限公司

```
     +
       |       |       |
|

                                  +
       |       |       |
|

                                  +
       |       |       |             | ** DB Stats
**

                                +
       |       |       |             | Uptime(secs): 248.6 total, 0.0
interval

                              +
       |       |       |             | Cumulative writes: 0 writes, 0 keys, 0 commit groups, 0.0 writes per
commit group, ingest: 0.00 GB, 0.00
MB/s

                                                                  +
       |       |       |             | Cumulative WAL: 0 writes, 0 syncs, 0.00 writes per sync, written: 0.00
GB, 0.00
MB/s

                                                                        +
       |       |       |             | Cumulative stall: 00:00:0.000 H:M:S, 0.0
percent

          +
       |       |       |             | Interval writes: 0 writes, 0 keys, 0 commit groups, 0.0 writes per
commit group, ingest: 0.00 MB, 0.00
MB/s

                                                                  +
       |       |       |             | Interval WAL: 0 writes, 0 syncs, 0.00 writes per sync, written: 0.00 GB,
0.00
MB/s

                                                                       +
       |       |       |             | Interval stall: 00:00:0.000 H:M:S, 0.0
percent

           +
       |       |       |             | Write Stall (count): write-buffer-manager-limit-stops: 0,
```

17.3.65 GS_GET_KV_LOCK_STATUS

GS_GET_KV_LOCK_STATUS视图用于获取RocksDB内部锁的持有情况，通过汇总当前集群各节点各RocksDB实例内部锁的信息，辅助分析DWS业务锁冲突的相关问题。该视图仅9.1.2及以上集群版本支持，且需要开启LSM引擎（enable_lsm_engine参数设置为on）才可访问。

表 17-165 GS_GET_KV_LOCK_STATUS 字段

名称	类型	描述
node	text	所在节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
kvdb_id	bigint	所在RocksDB实例ID，与bucket分区ID相对应。
cf_id	integer	所在列族ID。
bucket_id	smallint	所在bucket分区ID。
transaction_id	text	关联的事务ID。
is_exclusive	bool	是否持有排他锁。

应用示例

使用GS_GET_KV_LOCK_STATUS视图查询RocksDB内部持有锁情况。

```
postgres=# select * from gs_get_kv_lock_status('write_group');
   node    | kvdb_id | cf_id | bucket_id | transaction_id | is_exclusive
-----------+---------+-------+-----------+----------------+--------------
 datanode1 |   12290 |     2 |         2 | 6              | t
 datanode1 |   12290 |     1 |         2 | 6              | t
 datanode2 |   12291 |     2 |         3 | 6              | t
 datanode2 |   12291 |     1 |         3 | 6              | t
(4 rows)
```

17.3.66 GET_KV_LOCK_STATUS

GET_KV_LOCK_STATUS视图用于获取当前DN上RocksDB内部锁的持有情况。该视图仅9.1.2及以上集群版本支持，且需要开启LSM引擎（enable_lsm_engine参数设置为on）才可访问。

表 17-166 GET_KV_LOCK_STATUS 字段

名称	类型	描述
node	text	所在DN节点名称.
kvdb_id	bigint	所在RocksDB实例ID，与bucket分区ID相对应。
cf_id	integer	所在列族ID。
bucket_id	smallint	所在bucket分区ID。
transaction_id	text	关联的事务ID。
is_exclusive	bool	是否持有排他锁。

应用示例

使用GET_KV_LOCK_STATUS视图查询所在DN上RocksDB内部持有锁情况。

```
SELECT * FROM get_kv_lock_status();
   node    | kvdb_id | cf_id | bucket_id | transaction_id | is_exclusive
-----------+---------+-------+-----------+----------------+--------------
 datanode1 |   12290 |     2 |         2 | 6              | t
 datanode1 |   12290 |     1 |         2 | 6              | t
(2 rows)
```


版权所有 © 华为云计算技术有限公司

17.3.67 GS_LSC_MEMORY_DETAIL

GS_LSC_MEMORY_DETAIL视图，统计当前节点各线程SysCache内存占用情况。该视图仅9.1.1.100及以上集群版本支持。

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）
thread_name	text	线程名称。
context_name	text	MemoryContext名称。
totalsize	bigint	从操作系统分配的总内存。
usedsize	bigint	已使用的内存。

应用示例

查询当前视图，统计各线程SysCache内存占用情况。

```
SELECT * FROM GS_LSC_MEMORY_DETAIL;
```

```
gaussdb=> SELECT * FROM GS_LSC_MEMORY_DETAIL;
             sessid              |   thread_name    |      context_name      | totalsize | usedsize
---------------------------------+------------------+------------------------+-----------+----------
 1780331623.281472742769904.cn_5003 | postgres         | PartCacheMemoryContext |     24576 |     2816
 1780331623.281472742769904.cn_5003 | postgres         | RelCacheMemoryContext  |    122880 |      192
 1780331623.281472742769904.cn_5003 | postgres         | SysCacheMemoryContext  |    122880 |    34704
 1780555430.281472742774008.cn_5003 | postgres         | PartCacheMemoryContext |     24576 |       96
 1780555430.281472742774008.cn_5003 | postgres         | RelCacheMemoryContext  |    122880 |      192
 1780555430.281472742774008.cn_5003 | postgres         | SysCacheMemoryContext  |    122880 |    83296
 1780555945.281472742760784.cn_5003 | postgres         | SysCacheMemoryContext  |     24576 |    14880
 1780555557.281472742774920.cn_5003 | postgres         | SysCacheMemoryContext  |    122880 |    58688
 1780555905.281472742761240.cn_5003 | postgres         | SysCacheMemoryContext  |     57344 |    33616
 1779332982.281472742751664.cn_5003 | postgres         | SysCacheMemoryContext  |     24576 |    18512
 1779332981.281472742747104.cn_5003 | WLMCollectWorker | PartCacheMemoryContext |     57344 |    46512
 1779332981.281472742747104.cn_5003 | WLMCollectWorker | RelCacheMemoryContext  |     66584 |    53160
 1779332981.281472742747104.cn_5003 | WLMCollectWorker | SysCacheMemoryContext  |    122880 |    70192
 1779332981.281472742747560.cn_5003 | WLMCalSpaceInfo  | SysCacheMemoryContext  |     24576 |    19472
 1779332981.281472742748016.cn_5003 | WLMmonitor       | PartCacheMemoryContext |     24576 |     7872
 1779332981.281472742744368.cn_5003 | JobScheduler     | SysCacheMemoryContext  |     24576 |      224
(16 rows)
```

17.3.68 GS_OBS_READ_TRAFFIC

GS_OBS_READ_TRAFFIC视图，统计OBS读流量和平均读带宽，统计结果按10分钟聚集。该视图仅8.2.0及以上集群版本支持。

名称	类型	描述
nodename	TEXT	集群节点。
hostname	TEXT	主机节点。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
traffic_mb	float8	logtime之前10分钟OBS读流量统计。
bandwidth_mb_per_s	float8	平均带宽，单位MB/s。
reqcount	bigint	logtime之前10分钟OBS读次数。
logtime	timestamp with time zone	记录统计信息时刻。

应用示例

查询当前视图，统计OBS读流量和平均读带宽，统计结果按10分钟聚集。

```
select * from gs_obs_read_traffic;
 nodename |     hostname     |    traffic_mb     | bandwidth_mb_per_s | reqcount |        logtime
----------+------------------+-------------------+--------------------+----------+------------------------
 dn_1     | rhel_10_90_45_56 | 101.959338188171 |   5.14830159670447 |       23 | 2022-11-26 09:50:00+08
(1 row)
```

17.3.69 GS_OBS_WRITE_TRAFFIC

GS_OBS_WRITE_TRAFFIC视图，统计OBS写流量和平均写带宽，统计结果按10分钟聚集。该视图仅8.2.0及以上集群版本支持。

名称	类型	描述
nodename	TEXT	集群节点。
hostname	TEXT	主机节点。
traffic_mb	float8	logtime之前10分钟OBS写流量统计。
bandwidth_mb_per_s	float8	平均带宽，单位MB/s。
reqcount	bigint	logtime之前10分钟OBS写次数。
logtime	timestamp with time zone	记录统计信息时刻。

应用示例

查询当前视图，统计OBS写流量和平均写带宽，统计结果按10分钟聚集。

```
select * from gs_obs_write_traffic;
   nodename   |     hostname     |      traffic_mb      | bandwidth_mb_per_s  | reqcount |        logtime
--------------+------------------+----------------------+---------------------+----------+------------------------
 dn_1         | rhel_10_90_45_56 |  .000738143920898438 | .000289970820362525 |       12 | 2022-10-24
16:10:00+08
 dn_1         | rhel_10_90_45_56 |  .000354766845703125 | .000386063466694153 |        7 | 2022-10-24
18:50:00+08
 dn_1         | rhel_10_90_45_56 | 9.34600830078125e-05 | .000143659648687162 |        2 | 2022-11-07
09:20:00+08
```


版权所有 © 华为云计算技术有限公司

```
 dn_1         | rhel_10_90_45_56 | 4.10079956054688e-05 | .000186667253592502 |         1 | 2022-11-07
09:30:00+08
 dn_1         | rhel_10_90_45_56 |     2048.17834663391 |    27.2766632219637 |         2 | 2022-11-22
16:10:00+08
 dn_1         | rhel_10_90_45_56 |     3747.23722648621 |    28.0842938534546 |         4 | 2022-11-22
16:20:00+08
(6 row)
```

17.3.70 GS_INSTR_UNIQUE_SQL

Unique SQL 定义

数据库将接收到的每个SQL的文本字符串，都进行解析并生成内部解析树，遍历解析树并忽略其中的常数值，以一定的算法计算出来一个整数值作为Unique SQL ID，用来唯一标识这一类SQL，Unique SQL ID相同的一类SQL就叫做Unique SQL。

示例

假如，用户先后输入SQL:

```
select * from t1 where id = 1;
select * from t1 where id = 2;
```

那么，这两条SQL的统计信息会汇聚到同一个Unique SQL上：

```
select * from t1 where id = ?;
```

GS_INSTR_UNIQUE_SQL 视图

GS_INSTR_UNIQUE_SQL视图显示当前节点收集的Unique SQL的执行信息，主要包括以下内容：

- Unique SQL ID以及归一化后的SQL文本字符串，归一化后的SQL文本如**示例**中所示。通常对于DML语句，在计算Unique SQL ID的过程中会忽略常量值。但对于DDL、DCL以及设置参数等语句，常量值不可以忽略。
- 执行次数（成功执行的次数），响应时间（数据库内部的SQL执行时间，包括最大、最小和总时间）。
- Cache/IO信息，包含block的物理读、逻辑读次数，仅统计执行成功的SQL在各DN节点上的相关信息。该统计值与查询执行当时所处理的数据量、所使用的内存、是否多次执行、内存管理策略、是否有其他并发查询等因素相关，反映整个查询执行过程中的buffer块物理读和逻辑读次数，不同时间执行可能统计值不同。
- 行活动，包含SELECT语句的结果集返回行数、更新行、插入行、删除行、顺序扫描行、随机扫描行等信息。除结果集返回行数与该SELECT语句的结果集行数一致、且仅在CN上记录外，其他行活动信息均在DN上记录，且统计数值反应的是整个查询执行过程中的行活动，包括对相关系统表、元数据表、数据表等做必要的扫描和修改，与对应数据量以及相关参数设置相关，即统计数值将会大于等于对实际数据表的扫描和修改。
- 时间分布，包含：DB_TIME/CPU_TIME/EXECUTION_TIME/PARSE_TIME/PLAN_TIME/REWRITE_TIME/PL_EXECUTION_TIME/PL_COMPILATION_TIME/NET_SEND_TIME/DATA_IO_TIME，相关定义见**表1**。该信息在CN和DN节点均有统计，视图查询时将汇总展示。
- 软硬解析次数，包含软解析（缓存计划）、硬解析（生成计划）的次数，即如果本次执行的是之前缓存的计划，软解析次数+1，如果本次执行的计划是重新生成的，则硬解析次数+1。该次数在CN和DN节点上都会统计，视图查询时将汇总展示。

 版权所有 © 华为云计算技术有限公司

Unique SQL收集功能存在以下约束：

- 只有执行成功的SQL才会显示其详细的统计信息，否则可能只记录query、node、user等信息。
- 如果开启Unique SQL收集功能，CN节点将对所有接收到的查询进行统计收集，包括工具和用户的查询等。
- 若一条SQL语句内部包含执行多条SQL语句、类似存储过程执行等场景，仅会对最外层SQL生成一条Unique SQL，所有子SQL的统计信息都会汇总到该Unique SQL记录上。
- Unique SQL的响应时间统计中不完全包含NET_SEND_TIME阶段的时间，所以EXECUTION_TIME和elapse_time等时间之间不存在大小比较关系。
- 对于类似begin;...;commit;等形式的事务块，当前不支持统计子句的解析时间（parse_time）。

普通用户访问GS_INSTR_UNIQUE_SQL视图，只能看到该用户相关的Unique SQL信息，管理员用户可以看到当前节点所有的Unique SQL信息。CN和DN上均可查询GS_INSTR_UNIQUE_SQL视图，DN上显示的是本节点内的Unique SQL统计信息，CN上显示的是本节点Unique SQL完整统计信息，即该CN节点会收集其他CN和DN上对应该CN的Unique SQL的执行信息，进行汇总展示。通过查询GS_INSTR_UNIQUE_SQL视图，能够定位由于消耗不同资源导致的Top SQL，为集群性能调优和维护提供依据。

表 17-167 GS_INSTR_UNIQUE_SQL 字段

名称	类型	描述
node_name	name	接收SQL的CN节点名称。
node_id	integer	节点ID，等同于pgxc_node表中node_id。
user_name	name	用户名称。
user_id	oid	用户ID。
unique_sql_id	bigint	归一化的UNIQUE SQL ID。
query	text	归一化的SQL文本。
n_calls	bigint	成功执行次数。
min_elapse_time	bigint	SQL在数据库内的最小运行时间（单位：微秒）。
max_elapse_time	bigint	SQL在数据库内的最大运行时间（单位：微秒）。
total_elapse_time	bigint	SQL在数据库内的总运行时间（单位：微秒）。
n_returned_rows	bigint	行活动-SELECT语句返回的结果集行数。
n_tuples_fetched	bigint	行活动-随机扫描行（列存表/外表不统计）。
n_tuples_returned	bigint	行活动-顺序扫描行（列存表/外表不统计）。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
n_tuples_inserted	bigint	行活动-插入行数。
n_tuples_updated	bigint	行活动-更新行数。
n_tuples_deleted	bigint	行活动-删除行数。
n_blocks_fetched	bigint	buffer的块访问次数，即物理读/IO。
n_blocks_hit	bigint	buffer的块命中次数，即逻辑读/Cache。
n_soft_parse	bigint	软解析次数（缓存计划）。
n_hard_parse	bigint	硬解析次数（生成计划）。
db_time	bigint	有效的DB执行时间，包含等待时间、网络发送时间等，若查询执行涉及到多线程，DB_TIME是多个线程的DB_TIME之和（单位：微秒）。
cpu_time	bigint	CPU的执行时间，不包含sleep时间（单位：微秒）。
execution_time	bigint	查询执行器内的SQL执行时间，DDL语句、以及某些不经过执行器执行的语句（例如Copy语句）不计数（单位：微秒）。
parse_time	bigint	SQL解析时间（单位：微秒）。
plan_time	bigint	SQL生成计划时间（单位：微秒）。
rewrite_time	bigint	SQL重写时间（单位：微秒）。
pl_execution_time	bigint	plpgsql过程化语言函数上的执行时间（单位：微秒）。
pl_compilation_time	bigint	plpgsql过程化语言函数上的编译时间（单位：微秒）。
net_send_time	bigint	网络时间，包含CN向客户端发送数据、DN向CN发送数据等时间（单位：微秒）。
data_io_time	bigint	IO时间，文件IO耗时（单位：微秒）。
first_time	timestamp with time zone	该SQL第一次执行的时间。
last_time	timestamp with time zone	该SQL上一次执行的时间。

 版权所有 © 华为云计算技术有限公司

17.3.71 GS_NODE_STAT_RESET_TIME

GS_NODE_STAT_RESET_TIME视图显示当前节点的统计信息重置时间。

表 17-168 GS_NODE_STAT_RESET_TIME 字段

名称	类型	描述
get_node_stat_reset_time	timestamp with time zone	当前节点的统计信息重置时间。

说明

实例正常运行过程中，内存中的各类统计数值会逐渐累加，以下情况会导致内存中的统计数值被重置为0：

- 实例重启或集群发生了切换；
- 数据库Database被删除（drop）；
- 用户执行了重置操作，如执行pgstat_recv_resetcounter函数会将数据库中的统计计数器清零，执行reset_instr_unique_sql函数会将Unique SQL数据清零。

如果发生了以上事件，DWS会记录统计信息被重置的时间，可通过get_node_stat_reset_time函数查询。

17.3.72 GS_OBS_LATENCY

GS_OBS_LATENCY记录logtime之前10分钟内OBS的平均延迟信息，延迟数据是根据相关OBS的操作进行估算的结果。该视图仅8.2.0及以上集群版本支持。

表 17-169 GS_OBS_LATENCY 字段

名称	类型	描述
nodename	text	集群节点。
hostname	text	主机节点。
latency_ms	double precision	logtime之前10分钟内OBS的平均延迟，单位ms。
reqcount	bigint	logtime之前10分钟内OBS的请求次数。
logtime	timestamp with time zone	记录延迟信息的时刻。

17.3.73 GS_QUERY_MONITOR

GS_QUERY_MONITOR视图显示用户作业运行时的资源使用信息，仅支持在CN上查询，且仅显示主语句监控信息。该视图仅8.2.1.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 17-170 GS_QUERY_MONITOR 视图字段

名称	类型	描述
usename	name	执行该查询的用户名称。
nodename	name	执行该查询的CN名称。
nodegroup	name	执行该查询的集群名称，默认集群显示“installation”。
rpname	name	该查询关联的资源池名称。
priority	name	查询当前优先级，包含Rush/High/Medium/Low四个优先级。
xact_start	timestamp	查询所属事务的开启时间。
query_start	timestamp	查询执行开始时间。
block_time	bigint	作业累积已排队时间，存储过程、多语句可能多次排队。单位：秒。
duration	bigint	作业已运行时间，不包含排队时间。单位：秒。
query_band	text	显示作业标识，可通过GUC参数query_band设置，默认为空。
attribute	text	作业属性： ● Simple：简单作业； ● Complicated：复杂作业。 作业进入资源池管控前无意义，只有进入或已完成资源池管控该字段才有意义。
lane	text	作业排队/执行所处的资源池车道： ● fast：快车道； ● slow：慢车道。 作业进入资源池管控前无意义，只有进入或已完成资源池管控该字段才有意义。
status	text	作业当前状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active：非以上四种状态外的正常状态。 ● unknown：未知状态。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
queue	text	作业排队信息： ● None：作业正在运行； ● Global：作业在CN全局并发队列排队； ● Respool：作业在资源池队列排队； ● CCN：作业在CCN排队。
used_mem	integer	作业在所有DN上内存峰值的最大值，单位：MB。
estimate_mem	integer	作业估算内存，单位：MB。
used_cpu	double precision	作业开始运行至今，占用CPU核数的平均值。
read_speed	integer	作业当前在所有DN上逻辑IO读速率的平均值，单位：KB/s。
write_speed	integer	作业当前在所有DN上逻辑IO写速率的平均值，单位：KB/s。
send_speed	integer	作业开始运行至今，在所有DN上网络发送速率的平均值，单位：KB/s。
recv_speed	integer	作业开始运行至今，在所有DN上网络接收速率的平均值，单位：KB/s。
dn_count	bigint	执行该作业的DN数量。
stream_count	bigint	作业在所有DN上stream线程的数量之和。
pid	bigint	后端线程ID。
lwtid	integer	后台线程的轻量级线程号。
query_id	bigint	查询ID。
unique_sql_id	bigint	归一化的Unique SQL ID。
query	text	正在执行的查询。
obs_io_read_bps	integer	作业当前在所有DN上OBS IO读速率的平均值，单位：KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	integer	作业当前在所有DN上OBS IO写速率的平均值，单位：KB/s。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

17.3.74 GS_QUERY_RESOURCE_INFO

GS_QUERY_RESOURCE_INFO视图显示当前DN节点所有正在运行作业的资源信息。该参数仅9.1.0及以上集群版本支持。

说明

该视图只能在DN节点上查询。仅适用于运维操作定位问题，不建议用户使用。

表 17-171 GS_QUERY_RESOURCE_INFO

名称	类型	描述
node_name	text	实例名称，只包含DN。
user_id	oid	用户ID。
queryid	bigint	语句执行使用的内部query id。
used_mem	int	语句在当前DN使用的内存大小，单位MB。
cpu_time	bigint	语句在当前DN上的CPU时间，单位ms。
used_cpu	double	语句在当前DN使用的CPU个数。
spill_size	bigint	若发生下盘，语句在当前DN下盘数据量，默认为0。单位MB。
read_bytes	bigint	语句在当前DN使用的逻辑读字节数，单位KB。
write_bytes	bigint	语句在当前DN使用的逻辑写字节数，单位KB。
read_count	bigint	语句在当前DN使用的逻辑读次数。
write_count	bigint	语句在当前DN使用的逻辑写次数。
read_speed	int	语句在当前DN使用的逻辑读速率，单位KB/s。
write_speed	int	语句在当前DN使用的逻辑写速率，单位KB/s。
curr_iops	int	语句在当前DN上的每秒IO数值（列存单位是次/s，行存单位是万次/s）。
send_pkg	bigint	语句在当前DN上的通信包发送总量，单位packages。
recv_pkg	bigint	语句在当前DN上的通信包接收总量，单位packages。
send_bytes	bigint	语句在当前DN上的通信流发送数据总量，单位Byte。
recv_bytes	bigint	语句在当前DN上的通信流接收数据总量，单位Byte。
send_speed	int	语句在当前DN使用的网络发送速率，单位KB/s。
recv_speed	int	语句在当前DN使用的网络接收速率，单位KB/s。

版权所有 © 华为云计算技术有限公司

17.3.75 GS_REL_IOSTAT

GS_REL_IOSTAT视图提供当前节点上磁盘读写的统计信息。当前版本中，每次读/写磁盘只读/写一页，所以读写次数与页数相等。

表 17-172 GS_REL_IOSTAT 字段

名称	类型	描述
phyrds	bigint	读磁盘次数。
phywrts	bigint	写磁盘次数。
phyblkrd	bigint	读磁盘页数。
phyblkwrt	bigint	写磁盘页数。

17.3.76 GS_RESPOOL_RUNTIME_INFO

GS_RESPOOL_RUNTIME_INFO视图显示当前CN所有资源池作业运行信息。

表 17-173 GS_RESPOOL_RUNTIME_INFO 字段

名称	类型	描述
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数。
fast_run	int	资源池快车道运行作业数。
fast_wait	int	资源池快车道排队作业数。
slow_run	int	资源池慢车道运行作业数。
slow_wait	int	资源池慢车道排队作业数。

17.3.77 GS_RESPOOL_RESOURCE_INFO

GS_RESPOOL_RESOURCE_INFO视图显示CN上所有资源池作业运行信息以及当前实例（CN/DN）所有资源池资源使用信息。

说明

DN上仅显示当前DN所属逻辑集群的资源池监控信息。

版权所有 © 华为云计算技术有限公司

表 17-174 GS_RESPOOL_RESOURCE_INFO 字段

名称	类型	描述
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，只在CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5秒监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为DWS可用CPU，CPU限额管控情况下为关联控制组可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池当前使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_kbytes	bigint	资源池5秒监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5秒监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5秒监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5秒监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5秒监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5秒监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5秒监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5秒监控周期内网络接收速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_read_kbytes	bigint	资源池5秒监控周期内OBS IO读字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO读字节数。 ● CN：显示所有DN上资源池OBS IO读字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	资源池5秒监控周期内OBS IO写字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO写字节数。 ● CN：显示所有DN上资源池OBS IO写字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double	资源池5秒监控周期内OBS IO读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO读速率。 ● CN：显示所有DN上资源池OBS IO读速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double	资源池5秒监控周期内OBS IO写速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO写速率。 ● CN：显示所有DN上资源池OBS IO写速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。

17.3.78 GS_RESPOOL_MONITOR

GS_RESPOOL_MONITOR视图显示所有资源池作业运行信息及资源使用信息，仅支持CN上查询。该视图仅8.2.1.100及以上集群版本支持。

表 17-175 GS_RESPOOL_MONITOR 字段

名称	类型	描述
rpname	name	资源池名称。
nodegroup	name	资源池所属逻辑集群的名称，默认集群显示“installation”。
cn_count	bigint	集群包含的CN数量，多CN环境下用于判断单CN管控结果是否合理。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
short_acc	boolean	资源池是否开启短查询加速。
session_count	bigint	关联该资源池的会话数量，即关联该资源池的用户发起的会话数量，包含IDLE和ACTIVE会话。
active_count	bigint	关联该资源池的活跃会话数量，即正在执行查询的会话数量。
global_wait	bigint	关联该资源池的所有作业中，因单CN上并发超max_active_statements引起排队的作业数。
fast_run	bigint	关联该资源池的所有作业中，正在资源池快车道运行的作业数。
fast_wait	bigint	关联该资源池的所有作业中，在资源池快车道排队的作业数。
fast_limit	bigint	资源池快车道作业并发上限。
slow_run	bigint	关联该资源池的所有作业中，正在资源池慢车道运行的作业数。
slow_wait	bigint	关联该资源池的所有作业中，在资源池慢车道排队的作业数。
slow_limit	bigint	资源池慢车道作业并发上限。
used_mem	text	资源池在所有DN上已用内存的平均值，显示结果已使用pg_size_pretty格式化。
estimate_mem	text	资源池正在运行的作业估算内存之和，显示结果已使用pg_size_pretty格式化。
mem_limit	text	资源池可用内存的上限，显示结果已使用pg_size_pretty格式化。
query_mem_limit	name	资源池内单个查询可以使用的内存上限，主要用于限制查询估算内存，防止估算内存过大导致异常排队；通过估算内存限制查询实际使用内存，显示结果已使用pg_size_pretty格式化。
used_cpu	double precision	资源池在所有DN上占用CPU核数的平均值；CPU隔离以节点和资源池为单位，单个节点上包含多个DN时，资源池在单节点上占用的CPU核数需要乘以DN数。
cpu_limit	double precision	资源池在所有节点上可用CPU上限的平均值，CPU配额管控情况下为DWS全部可用CPU核数，CPU限额管控情况下为关联控制组的可用CPU核数。
read_speed	text	资源池在所有DN上逻辑IO读速率的平均值，显示结果已使用pg_size_pretty格式化。
write_speed	text	资源池在所有DN上逻辑IO写速率的平均值，显示结果已使用pg_size_pretty格式化。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
send_speed	text	资源池在所有DN上网络发送速率的平均值，显示结果已使用pg_size_pretty格式化。
recv_speed	text	资源池在所有DN上网络接收速率的平均值，显示结果已使用pg_size_pretty格式化。
obs_io_read_bps	text	资源池在所有DN上OBS IO读速率的平均值，显示结果已使用pg_size_pretty格式化。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	text	资源池在所有DN上OBS IO写速率的平均值，显示结果已使用pg_size_pretty格式化。 该字段仅9.1.1.200及以上集群版本支持。

17.3.79 GS_ROW_TABLE_IO_STAT

GS_ROW_TABLE_IO_STAT视图显示当前数据库中所有行存表在当前节点上的IO情况。各统计字段为实例启动以来的累计值。

表 17-176 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

17.3.80 GS_SESSION_CPU_STATISTICS

GS_SESSION_CPU_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理CPU使用的信息。

 版权所有 © 华为云计算技术有限公司

表 17-177 GS_SESSION_CPU_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
username	name	登录到该后端的用户名。
pid	bigint	后端线程ID。
start_time	timestamp with time zone	语句执行的开始时间。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位为ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位为ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位为ms。
query	text	正在执行的语句。
node_group	text	语句所属用户对应的逻辑集群。

17.3.81 GS_SESSION_MEMORY_STATISTICS

GS_SESSION_MEMORY_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理内存使用的信息。

表 17-178 GS_SESSION_MEMORY_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
username	name	登录到该后端的用户名。
pid	bigint	后端线程ID。
start_time	timestamp with time zone	语句执行的开始时间。
min_peak_memory	integer	语句在所有DN上的最小内存峰值大小，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值大小，单位MB。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
query	text	正在执行的语句。
node_group	text	语句所属用户对应的逻辑集群。

17.3.82 GS_SQL_COUNT

GS_SQL_COUNT视图显示数据库当前节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）统计信息，包括执行次数和响应时间（除MERGE INTO语句外，统计其他四类语句的最大、最小、平均和总响应时间，单位为微秒），以及DDL、DML、DCL类型语句的执行次数。

GS_SQL_COUNT视图中对DDL、DML、DCL类型语句分类与SQL语法中略有不同，具体如下：

- CREATE/ALTER/DROP USER，CREATE/ALTER/DROP ROLE等用户相关语句属于DCL类型。
- BEGIN/COMMIT/SET CONSTRAINTS/ROLLBACK/SAVEPOINT/START等事务相关语句属于DCL类型。
- ALTER SYSTEM KILL SESSION等价于SELECT pg_terminate_backend()语句，属于DML类型。

其余语句的分类与SQL语法中定义类似。

普通用户查询GS_SQL_COUNT视图仅能看到该用户当前节点的统计信息。管理员权限用户查询GS_SQL_COUNT视图则能看到所有用户当前节点的统计信息；当集群或该节点重启时，计数会清零，并重新开始计数。计数以节点收到的查询数为准，包括集群内部进行的查询；GS_SQL_COUNT视图涉及的统计信息只在CN上统计，且不统计从其他CN发送过来的SQL。在DN上查询该视图返回结果为空。

表 17-179 GS_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称。
user_name	name	用户名。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
mergeinto_count	bigint	MERGE INTO数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_select_elapse	bigint	SELECT总响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。

17.3.83 GS_STAT_DB_CU

GS_STAT_DB_CU视图查询集群各个节点，每个数据库的CU命中情况。可以通过gs_stat_reset()进行清零。

版权所有 © 华为云计算技术有限公司

表 17-180 GS_STAT_DB_CU 字段

名称	类型	描述
node_name1	text	节点名称。
db_name	text	数据库名称。
mem_hit	bigint	内存命中次数。
hdd_sync_read	bigint	硬盘同步读次数。
hdd_asyn_read	bigint	硬盘异步读次数。

17.3.84 GS_STAT_SESSION_CU

GS_STAT_SESSION_CU视图查询当前集群各个节点，当前运行session的CU命中情况。session退出相应的统计数据会清零。集群重启后，统计数据也会清零。

表 17-181 GS_STAT_SESSION_CU 字段

名称	类型	描述
node_name1	text	节点名称。
mem_hit	integer	内存命中次数。
hdd_sync_read	integer	硬盘同步读次数。
hdd_asyn_read	integer	硬盘异步读次数。

17.3.85 GS_TABLE_CHANGE_STAT

GS_TABLE_CHANGE_STAT视图显示当前数据库中所有表格（不包括外表）在当前节点上的变更情况。表示次数的各字段为实例启动以来的累计值。

表 17-182 GS_TABLE_CHANGE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动vacuum的时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
vacuum_count	bigint	手动Vacuum的次数。
last_autovacuum	timestamp with time zone	最后一次自动vacuum的时间。
autovacuum_count	bigint	自动vacuum的次数。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
analyze_count	bigint	分析（包括手动和自动）的次数。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

17.3.86 GS_TABLE_STAT

GS_TABLE_STAT视图显示当前数据库中所有表格（不包括外表）在当前节点上的统计信息。除live_tuples和dead_tuples为当前实时值外，其余各统计字段为实例启动以来的累计值。

表 17-183 GS_TABLE_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。
index_scan	bigint	索引扫描的次数。
index_tuple_read	bigint	索引扫描的行数。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

17.3.87 GS_TOTAL_NODEGROUP_MEMORY_DETAIL

GS_TOTAL_NODEGROUP_MEMORY_DETAIL视图统计当前数据库逻辑集群使用内存的信息，单位为MB。

表 17-184 GS_TOTAL_NODEGROUP_MEMORY_DETAIL 字段

名称	类型	描述
ngname	text	逻辑集群名称。
memorytype	text	内存类型，包括以下几种： ● ng_total_memory：该逻辑集群的总内存大小。 ● ng_used_memory：该逻辑集群的实际使用内存大小。 ● ng_estimate_memory：该逻辑集群的估算使用内存大小。 ● ng_foreignrp_memsize：该逻辑集群的外部资源池的总内存大小。 ● ng_foreignrp_usedsize：该逻辑集群的外部资源池实际使用内存大小。 ● ng_foreignrp_peaksize：该逻辑集群的外部资源池使用内存的峰值。 ● ng_foreignrp_mempct：该逻辑集群的外部资源池占该逻辑集群总内存大小的百分比。 ● ng_foreignrp_estmsize：该逻辑集群的外部资源池估算使用内存大小。
memorymbytes	integer	内存类型分配内存的大小。

17.3.88 GS_USER_MONITOR

GS_USER_MONITOR视图显示所有用户作业运行信息及资源使用信息，仅支持CN上查询。该视图仅8.2.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-185 GS_USER_MONITOR 字段

名称	类型	描述
usename	name	用户名称。
rpname	name	用户关联的资源池名称。
nodegroup	name	资源池所属逻辑集群的名称，默认集群显示“installation”。
session_count	bigint	该用户发起的会话数量，包含IDLE和ACTIVE会话。
active_count	bigint	该用户发起的活跃会话数量，即正在执行查询的会话数量。
global_wait	bigint	该用户执行的所有作业中，因单CN上并发超max_active_statements引起排队的作业数。
fast_run	bigint	该用户执行的所有作业中，正在资源池快车道运行的作业数。
fast_wait	bigint	该用户执行的所有作业中，在资源池快车道排队的作业数。
slow_run	bigint	该用户执行的所有作业中，正在资源池慢车道运行的作业数。
slow_wait	bigint	该用户执行的所有作业中，在资源池慢车道排队的作业数。
used_mem	bigint	用户在所有DN上已用内存的平均值，单位：MB。
estimate_mem	bigint	用户正在运行的作业估算内存之和，单位：MB。
used_cpu	double precision	用户在所有DN上使用CPU核数的平均值，单个节点上包含多个DN时，用户在单节点上占用的CPU核数需要乘以DN数。
read_speed	bigint	用户在所有DN上逻辑IO读速率的平均值，单位：KB/s。
write_speed	bigint	用户在所有DN上逻辑IO写速率的平均值，单位：KB/s。
send_speed	bigint	用户在所有DN上网络发送速率的平均值，单位：KB/s。
recv_speed	bigint	用户在所有DN上网络接收速率的平均值，单位：KB/s。
used_space	bigint	用户永久表已使用的空间大小，单位：KB。
space_limit	bigint	用户永久表可使用的空间大小上限，单位：KB。-1为不限制。
used_temp_space	bigint	用户临时表已使用的空间大小，单位：KB。
temp_space_limit	bigint	用户临时表可使用的空间大小上限，单位：KB。-1为不限制。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_spill_space	bigint	用户中间结果集落盘已使用的空间大小，单位：KB。
spill_space_limit	bigint	用户中间结果集落盘可使用的空间大小上限，单位：KB。-1为不限制。
obs_io_read_bps	bigint	用户在所有DN上OBS IO读速率的平均值，单位：KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	bigint	用户在所有DN上OBS IO写速率的平均值，单位：KB/s。 该字段仅9.1.1.200及以上集群版本支持。

17.3.89 GS_USER_TRANSACTION

GS_USER_TRANSACTION视图提供查询单CN上用户相关的事务信息。数据库记录每个用户事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 17-186 GS_USER_TRANSACTION 字段

名称	类型	描述
usename	name	用户名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

17.3.90 GS_VIEW_DEPENDENCY

GS_VIEW_DEPENDENCY视图提供查询当前用户可见的所有视图的直接依赖关系。

表 17-187 GS_VIEW_DEPENDENCY 字段

名称	类型	描述
objschema	name	视图空间名称。
objname	name	视图名称。
refobjschema	name	依赖对象的空间名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
refobjname	name	依赖对象的名称。
relobjkind	char	依赖对象的类型。 • r表示依赖对象为表。 • v表示依赖对象为视图。

17.3.91 GS_VIEW_DEPENDENCY_PATH

GS_VIEW_DEPENDENCY_PATH视图提供查询当前用户可见的所有视图的直接依赖关系。如果该视图依赖的基础表存在且各级视图依赖关系正常，通过该视图可以查询自基础表开始的各级视图的依赖关系。

表 17-188 GS_VIEW_DEPENDENCY_PATH 字段

名称	类型	描述
objschema	name	视图空间名称。
objname	name	视图名称。
refobjschema	name	依赖对象的空间名称。
refobjname	name	依赖对象的名称。
path	text	依赖路径。

17.3.92 GS_VIEW_INVALID

GS_VIEW_INVALID视图提供查询当前用户可见的所有不可用的视图。

如果该视图依赖的基础表或函数或同义词存在异常，该视图validtype列显示为“invalid”；如果该视图属于在升级过程中因依赖的系统对象变化，该视图validtype列显示为“invalidInUpgrade”。

表 17-189 GS_VIEW_INVALID 字段

名称	类型	描述
oid	oid	视图OID。
schemaname	name	视图空间名称。
viewname	name	视图名称。
viewowner	name	视图的所有者。
definition	text	视图定义。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
validtype	text	视图有效性标识。
last_invalid_time	timestamp with time zone	视图无效的时间。该字段仅9.1.0.200及以上集群版本支持。

17.3.93 GS_WAIT_EVENTS

GS_WAIT_EVENTS视图显示当前节点上各类等待状态和事件的统计信息。

仅在GUC参数enable_track_wait_event为on的情况下，视图中各统计字段的数值才会被累加。若在运行过程中将enable_track_wait_event设置为off，则不再累加统计数值，但已有数值不受影响。enable_track_wait_event为off，查询该视图返回0行。

表 17-190 GS_WAIT_EVENTS 字段

名称	类型	描述
nodename	name	节点名称。
type	text	事件的类型，包括STATUS，LOCK_EVENT，LWLOCK_EVENT和IO_EVENT四种类型。
event	text	事件名称，可参考PG_THREAD_WAIT_STATUS视图。
wait	bigint	事件发生次数。该字段及以下字段均为进程运行中的累计值。
failed_wait	bigint	等待失败次数。当前版本中只有LOCK和LWLOCK等锁超时或失败才会使用该字段。
total_wait_time	bigint	该事件总持续时间。
avg_wait_time	bigint	该事件平均持续时间。
max_wait_time	bigint	该事件最大等待时间。
min_wait_time	bigint	该事件最小等待时间。

当前版本中，对于type='LOCK_EVENT'，'LWLOCK_EVENT'和'IO_EVENT'的事件，GS_WAIT_EVENTS视图显示范围与PG_THREAD_WAIT_STATUS视图对应事件相同。

对于type='STATUS'的事件GS_WAIT_EVENTS包含的等待状态列如下，其详细含义参见PG_THREAD_WAIT_STATUS视图。

- acquire lwlock
- acquire lock
- wait io

 版权所有 © 华为云计算技术有限公司

- wait pooler get conn
- wait pooler abort conn
- wait pooler clean conn
- wait transaction sync
- wait wal sync
- wait data sync
- wait producer ready
- create index
- analyze
- vacuum
- vacuum full
- gtm connect
- gtm begin trans
- gtm commit trans
- gtm rollback trans
- gtm create sequence
- gtm alter sequence
- gtm get sequence val
- gtm set sequence val
- gtm drop sequence
- gtm rename sequence

17.3.94 GS_WLM_OPERATOR_INFO

本视图显示当前CN上已经完成执行的query语句中的算子执行信息，此系统视图信息来源于系统表dbms_om.**gs_wlm_operator_info**。

说明

- GS_WLM_OPERATOR_INFO视图的schema为pg_catalog。
- GS_WLM_OPERATOR_INFO视图表仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。

表 17-191 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值，单位MB。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量，单位MB。默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量，单位MB。默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量，单位MB。默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间，单位毫秒。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间，单位毫秒。
total_cpu_time	bigint	该算子在所有DN上的总执行时间，单位毫秒。
cpu_skew_percent	integer	DN间执行时间的倾斜率。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

17.3.95 GS_WLM_OPERATOR_HISTORY

GS_WLM_OPERATOR_HISTORY视图显示的是当前用户在当前CN上执行作业结束后的算子的相关记录。

此视图用于从DWS中查询数据，数据库中的数据会定时被清理。当GUC参数**enable_resource_record**为on时，视图中的记录每隔3分钟被转储到系统表**GS_WLM_OPERATOR_INFO**中一次，同时视图中的记录被删除；当GUC参数enable_resource_record为off时，记录在视图中的存留时间达到超期时间后会被删除。记录的数据同**表17-192**。

表 17-192 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值，单位MB。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量，单位MB。默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量，单位MB。默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量，单位MB。默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间，单位毫秒。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间，单位毫秒。
total_cpu_time	bigint	该算子在所有DN上的总执行时间，单位毫秒。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

版权所有 © 华为云计算技术有限公司

17.3.96 GS_WLM_OPERATOR_STATISTICS

GS_WLM_OPERATOR_STATISTICS视图显示当前用户正在执行的作业的算子相关信息。

表 17-193 GS_WLM_OPERATOR_STATISTICS 字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称，算子名称长度最大为127字符（不包含空格等格式化字符）。
start_time	timestamp with time zone	该算子第一次开始执行的时间。
duration	bigint	该算子从开始执行直到结束时总的执行时间(ms)。
status	text	当前算子的执行状态，包括waiting、running和finished。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息，若返回的预估行数超过int64_max时，显示为int64_max。
tuple_processed	bigint	当前算子在所有DN上的返回的元素个数总和，若返回的预估行数超过int64_max时，显示为int64_max。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小逻辑下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大逻辑下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均逻辑下盘数据量(MB)，默认为0。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin
parent_id	integer	该算子节点的父节点id。
exec_count	integer	该算子节点在所有DN上的最大执行次数。
progress	text	该算子的进度信息，第一个算子展示的是作业整体的进度。其他算子展示的是当前算子进度信息。
min_net_size	bigint	该算子在所有DN上的最小网络通信数据量(KB)，主要涉及网络算子。
max_net_size	bigint	该算子在所有DN上的最大网络通信数据量(KB)，主要涉及网络算子。
total_net_size	bigint	该算子在所有DN上的总网络通信数据量(KB)，主要涉及网络算子。
min_read_bytes	bigint	该算子在所有DN上的最小磁盘读取数据量(KB)。
max_read_bytes	bigint	该算子在所有DN上的最大磁盘读取数据量(KB)。
total_read_bytes	bigint	该算子在所有DN上的总磁盘读取数据量(KB)。
min_write_bytes	bigint	该算子在所有DN上的最小磁盘写入数据量(KB)。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_write_bytes	bigint	该算子在所有DN上的最大磁盘写入数据量(KB)。
total_write_bytes	bigint	该算子在所有DN上的总磁盘写入数据量(KB)。

17.3.97 GS_WLM_SESSION_INFO

GS_WLM_SESSION_INFO视图显示当前CN上已经完成执行的query语句的执行信息，此系统视图信息来源于系统表dbms_om.**gs_wlm_session_info**。

说明

- GS_WLM_SESSION_INFO视图的schema为pg_catalog。
- GS_WLM_SESSION_INFO视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 9.1.1.300及以上集群版本中系统会将执行成功且执行时间小于100ms，并且包含sql_hash的短查询进行聚合记录。9.1.2及以上集群版本中，子语句聚合不受100ms执行时间限制，满足其他聚合条件时默认进行聚合。聚合的主要列为datid、dbname、resource_pool、node_group、username、application_name、client_addr、sql_hash、plan_hash、query_band和nodename。

GS_WLM_SESSION_INFO视图的字段同**表17-195**相同，具体字段内容如下：

表 17-194 GS_WLM_SESSION_HISTORY 字段

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
datid	oid	连接后端的数据库OID。	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。	连接后端的数据库名称。
schemaname	text	模式名。	聚合周期内，同聚合主键第一个语句的schemaname。
nodename	text	语句执行的CN名称。	语句执行的CN名称。
username	text	连接到后端的用户名。	连接到后端的用户名。
application_name	text	连接到后端的应用名。	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。	连接到后端的客户端的IP地址。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。	聚合周期内，同聚合主键第一个语句的客户端的client_hostname。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。	聚合周期内，同聚合主键第一个语句的client_port。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。	聚合周期内，同聚合主键第一个语句的query_band。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。	聚合周期内，同聚合主键所有语句的block_time平均值。
start_time	timestamp with time zone	语句执行的开始时间。	聚合周期内，同聚合主键第一个语句的start_time。
finish_time	timestamp with time zone	语句执行的结束时间。	聚合周期内，同聚合主键最后一个语句的finish_time。
duration	bigint	语句实际执行的时间，单位ms。	聚合周期内，同聚合主键所有语句的duration平均值。
estimate_total_time	bigint	语句预估执行时间，单位ms。	聚合周期内，同聚合主键所有语句的estimate_total_time平均值。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。	语句执行结束状态，仅聚合执行成功的语句，默认为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。	空。
resource_pool	text	用户使用的资源池。	用户使用的资源池。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
control_group	text	语句所使用的Cgroup。	聚合周期内，同聚合主键第一个语句的control_group。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。	聚合周期内，同聚合主键所有语句的estimate_memory平均值。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的min_peak_memory最小值。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的max_peak_memory最大值。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。	聚合周期内，同聚合主键所有语句的average_peak_memory平均值。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。	聚合周期内，同聚合主键所有语句的memory_skew_percent的平均值。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。	空
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的min_spill_size的最小值。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的max_spill_size的最大值。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的average_spill_size的平均值。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。	聚合周期内，同聚合主键所有语句的spill_skew_percent的平均值。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。	聚合周期内，同聚合主键所有语句的min_dn_time的最小值。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。	聚合周期内，同聚合主键所有语句的max_dn_time的最大值。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。	聚合周期内，同聚合主键所有语句的average_dn_time平均值。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。	聚合周期内，同聚合主键所有语句的dntime_skew_percent平均值。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的min_cpu_time最小值。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的max_cpu_time最大值。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。	聚合周期内，同聚合主键所有语句的total_cpu_time总量。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。	聚合周期内，同聚合主键所有语句的cpu_skew_percent平均值。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的min_peak_iops最小值。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的max_peak_iops最大值。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的average_peak_iops平均值。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
iops_skew_percent	integer	语句在DN间的IO倾斜率。	聚合周期内，同聚合主键所有语句的iops_skew_percent平均值。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin	聚合周期内，同聚合主键第一个语句的warning。
queryid	bigint	语句执行使用的内部query id。	聚合周期内，同聚合主键第一个语句的queryid。
query	text	执行的语句，最多可保留64KB长度的字符串。	聚合周期内，同聚合主键第一个语句的query。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。	聚合周期内，同聚合主键第一个语句的query_plan。
node_group	text	语句所属用户对应的逻辑集群。	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。	聚合周期内，同聚合主键第一个语句的pid。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
lane	text	语句执行时所在的快慢车道。	聚合周期内，同聚合主键第一个语句的lane。
unique_sql_id	bigint	归一化的Unique SQL ID。	聚合周期内，同聚合主键第一个语句的unique_sql_id。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。	聚合周期内，同聚合主键第一个语句的session_id。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_read_bytes最小值。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_read_bytes最大值。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_read_bytes平均值。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_write_bytes最小值。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_write_bytes最大值。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_write_bytes平均值。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。	聚合周期内，同聚合主键所有语句的recv_pkg总量。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。	聚合周期内，同聚合主键所有语句的send_pkg总量。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的recv_bytes总量。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的send_bytes总量。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
stmt_type	text	语句对应的查询类型。	聚合周期内，同聚合主键第一个语句的stmt_type。
except_info	text	语句触发的异常规则信息。	空
unique_plan_id	bigint	归一化的Unique plan id。	聚合周期内，同聚合主键第一个语句的unique_plan_id。
sql_hash	text	归一化的sql hash。	归一化的sql hash。
plan_hash	text	归一化的plan hash。	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。	聚合周期内，同聚合主键第一个语句的use_plan_baseline。
outline_name	text	该语句计划对应的outline的名字。	聚合周期内，同聚合主键第一个语句的outline_name。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。	保存导入导出类业务信息的json串具体如下。 1. address：聚合周期内，同聚合主键第一个语句的address。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：聚合周期内，同聚合主键所有语句的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的parse_time平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_hit_ratio总量。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_read_size总量。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_write_size总量。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_size总量。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_time总量。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_scan_bytes总量。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_remote_read_bytes总量。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_submit_time总量。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_time总量。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_count总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_load_time总量。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_conflict_count总量。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_error_count总量。
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。	空
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_rtt总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_latency总量。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_1s总量。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_10s总量。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_count总量。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_retry_count总量。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_rate_limit_count总量。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键第一个语句的gxid。
inserted_rows	bigint	语句插入的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的inserted_rows平均值
deleted_rows	bigint	语句删除的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的deleted_rows平均值
updated_rows	bigint	语句更新的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的updated_rows平均值
returned_rows	bigint	语句查询的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的returned_rows平均值

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_read_bytes	bigint	语句在所有DN上的平均OBS IO读字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_read_bytes平均值
obs_io_write_bytes	bigint	语句在所有DN上的平均OBS IO写字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_write_bytes平均值
calls	integer	默认为1。	聚合周期内，同聚合主键的查询数量。

17.3.98 GS_WLM_SESSION_HISTORY

GS_WLM_SESSION_HISTORY视图显示当前用户在当前CN上执行作业结束后的负载管理记录。此视图用于从DWS中查询数据，仅当GUC参数**enable_resource_track**为on时，视图会查询GS_WLM_SESSION_INFO表中3分钟内的数据进行返回。

说明

- GS_WLM_SESSION_HISTORY视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 9.1.1.300及以上集群版本中系统会将执行成功且执行时间小于100ms，并且包含sql_hash的短查询进行聚合记录。9.1.2及以上集群版本中，子语句聚合不受100ms执行时间限制，满足其他聚合条件时默认进行聚合。聚合的主要列为datid、dbname、resource_pool、node_group、username、application_name、client_addr、sql_hash、plan_hash、query_band和nodename。

表 17-195 GS_WLM_SESSION_HISTORY 字段

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
datid	oid	连接后端的数据库OID。	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。	连接后端的数据库名称。
schemaname	text	模式名。	聚合周期内，同聚合主键第一个语句的schemaname。
nodename	text	语句执行的CN名称。	语句执行的CN名称。
username	text	连接到后端的用户名。	连接到后端的用户名。
application_name	text	连接到后端的应用名。	连接到后端的应用名。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。	连接到后端的客户端的IP地址。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。	聚合周期内，同聚合主键第一个语句的客户端的client_hostname。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。	聚合周期内，同聚合主键第一个语句的client_port。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。	聚合周期内，同聚合主键第一个语句的query_band。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。	聚合周期内，同聚合主键所有语句的block_time平均值。
start_time	timestamp with time zone	语句执行的开始时间。	聚合周期内，同聚合主键第一个语句的start_time。
finish_time	timestamp with time zone	语句执行的结束时间。	聚合周期内，同聚合主键最后一个语句的finish_time。
duration	bigint	语句实际执行的时间，单位ms。	聚合周期内，同聚合主键所有语句的duration平均值。
estimate_total_time	bigint	语句预估执行时间，单位ms。	聚合周期内，同聚合主键所有语句的estimate_total_time平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。	语句执行结束状态，仅聚合执行成功的语句，默认为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。	空。
resource_pool	text	用户使用的资源池。	用户使用的资源池。
control_group	text	语句所使用的Cgroup。	聚合周期内，同聚合主键第一个语句的control_group。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。	聚合周期内，同聚合主键所有语句的estimate_memory平均值。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的min_peak_memory最小值。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的max_peak_memory最大值。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。	聚合周期内，同聚合主键所有语句的average_peak_memory平均值。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。	聚合周期内，同聚合主键所有语句的memory_skew_percent的平均值。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。	空

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的min_spill_size的最小值。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的max_spill_size的最大值。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的average_spill_size的平均值。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。	聚合周期内，同聚合主键所有语句的spill_skew_percent的平均值。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。	聚合周期内，同聚合主键所有语句的min_dn_time的最小值。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。	聚合周期内，同聚合主键所有语句的max_dn_time的最大值。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。	聚合周期内，同聚合主键所有语句的average_dn_time平均值。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。	聚合周期内，同聚合主键所有语句的dntime_skew_percent平均值。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的min_cpu_time最小值。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的max_cpu_time最大值。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。	聚合周期内，同聚合主键所有语句的total_cpu_time总量。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。	聚合周期内，同聚合主键所有语句的cpu_skew_percent平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的min_peak_iops最小值。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的max_peak_iops最大值。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的average_peak_iops平均值。
iops_skew_percent	integer	语句在DN间的IO倾斜率。	聚合周期内，同聚合主键所有语句的iops_skew_percent平均值。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin	聚合周期内，同聚合主键第一个语句的warning。
queryid	bigint	语句执行使用的内部query id。	聚合周期内，同聚合主键第一个语句的queryid。
query	text	执行的语句，最多可保留64KB长度的字符串。	聚合周期内，同聚合主键第一个语句的query。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。	聚合周期内，同聚合主键第一个语句的query_plan。
node_group	text	语句所属用户对应的逻辑集群。	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。	聚合周期内，同聚合主键第一个语句的pid。
lane	text	语句执行时所在的快慢车道。	聚合周期内，同聚合主键第一个语句的lane。
unique_sql_id	bigint	归一化的Unique SQL ID。	聚合周期内，同聚合主键第一个语句的unique_sql_id。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。	聚合周期内，同聚合主键第一个语句的session_id。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_read_bytes最小值。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_read_bytes最大值。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_read_bytes平均值。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_write_bytes最小值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_write_bytes最大值。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_write_bytes平均值。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。	聚合周期内，同聚合主键所有语句的recv_pkg总量。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。	聚合周期内，同聚合主键所有语句的send_pkg总量。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的recv_bytes总量。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的send_bytes总量。
stmt_type	text	语句对应的查询类型。	聚合周期内，同聚合主键第一个语句的stmt_type。
except_info	text	语句触发的异常规则信息。	空
unique_plan_id	bigint	归一化的Unique plan id。	聚合周期内，同聚合主键第一个语句的unique_plan_id。
sql_hash	text	归一化的sql hash。	归一化的sql hash。
plan_hash	text	归一化的plan hash。	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。	聚合周期内，同聚合主键第一个语句的use_plan_baseline。
outline_name	text	该语句计划对应的outline的名字。	聚合周期内，同聚合主键第一个语句的outline_name。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。	保存导入导出类业务信息的json串具体如下。 1. address：聚合周期内，同聚合主键第一个语句的address。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：聚合周期内，同聚合主键所有语句的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的parse_time平均值。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_hit_ratio总量。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_read_size总量。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_write_size总量。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_size总量。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_time总量。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_scan_bytes总量。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_remote_read_bytes总量。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_submit_time总量。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_time总量。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_count总量。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_load_time总量。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_conflict_count总量。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_error_count总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。	空
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_rtt总量。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_latency总量。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_1s总量。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_10s总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_count总量。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_retry_count总量。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_rate_limit_count总量。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键第一个语句的gxid。
inserted_rows	bigint	语句插入的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的inserted_rows平均值
deleted_rows	bigint	语句删除的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的deleted_rows平均值
updated_rows	bigint	语句更新的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的updated_rows平均值
returned_rows	bigint	语句查询的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的returned_rows平均值
obs_io_read_bytes	bigint	语句在所有DN上的平均OBS IO读字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_read_bytes平均值
obs_io_write_bytes	bigint	语句在所有DN上的平均OBS IO写字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_write_bytes平均值
calls	integer	默认为1。	聚合周期内，同聚合主键的查询数量。

版权所有 © 华为云计算技术有限公司

17.3.99 GS_WLM_SESSION_STATISTICS

GS_WLM_SESSION_STATISTICS视图显示当前用户在当前CN上正在执行的作业的负载管理记录。

表 17-196 GS_WLM_SESSION_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据OID。
dbname	name	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN节点名称。
username	name	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
pid	bigint	后端线程ID。
block_time	bigint	语句执行前的阻塞时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
duration	bigint	语句已经执行的时间，单位ms。
estimate_total_time	bigint	语句执行预估总时间，单位ms。
estimate_left_time	bigint	语句执行预估剩余时间，单位ms。
enqueue	text	工作负载管理资源状态。
resource_pool	name	用户使用的资源池。
control_group	text	语句所使用的Cgroup。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句在各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在各DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
min_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最小IO读速率，单位KB/s。
max_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最大IO读速率，单位KB/s。
average_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的平均IO读速率，单位KB/s。
min_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最小IO写速率，单位KB/s。
max_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最大IO写速率，单位KB/s。
average_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的平均IO写速率，单位KB/s。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
unique_sql_id	bigint	归一化的Unique SQL ID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
queryid	bigint	语句执行使用的内部query id。
query	text	正在执行的语句。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
stmt_type	text	语句所对应的查询类型。
except_info	text	语句触发的异常规则信息。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。 该字段仅8.3.0.100及以上版本支持。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对3.0表及外表生效。
block_name	text	与语句匹配的对应拦截规则名。该字段仅9.1.0.200及以上集群版本支持。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。
obs_io_read_bps	integer	一个监控周期（5秒）内，语句在所有DN上的平均OBS IO读速率，单位KB/s。该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_write_bps	integer	一个监控周期（5秒）内，语句在所有DN上的平均OBS IO写速率，单位KB/s。该字段仅9.1.1.200及以上集群版本支持。
cudesc_streaming_count	integer	语句所使用的cudesc streaming个数。该字段仅9.1.1.300及以上集群版本支持。

17.3.100 GS_WLM_SQL_ALLOW

GS_WLM_SQL_ALLOW视图显示已经设置的资源管理SQL白名单。

其中白名单包括两部分内容：

- 系统默认的SQL白名单。
- 通过GUC参数wlm_sql_allow_list设置的SQL白名单。

17.3.101 GS_WORKLOAD_SQL_COUNT

GS_WORKLOAD_SQL_COUNT视图显示当前节点上Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 17-197 GS_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

17.3.102 GS_WORKLOAD_SQL_ELAPSE_TIME

GS_WORKLOAD_SQL_ELAPSE_TIME视图显示当前节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

 版权所有 © 华为云计算技术有限公司

表 17-198 GS_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

17.3.103 GS_WORKLOAD_TRANSACTION

GS_WORKLOAD_TRANSACTION视图提供查询单CN上Workload控制组相关的事务信息。数据库记录每个Workload控制组事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 17-199 GS_WORKLOAD_TRANSACTION 字段

名称	类型	描述
workload	name	Workload控制组名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

17.3.104 MPP_TABLES

MPP_TABLES视图显示PGXC_CLASS中的表信息。

表 17-200 MPP_TABLES 字段

名称	类型	描述
schemaname	name	包含表的模式名。
tablename	name	表名。
tableowner	name	表的所有者。
tablespace	name	表所在的表空间。
pgroup	name	节点群的名称。
nodeoids	oidvector_extend	表分布的节点OID列表。

17.3.105 PG_AVAILABLE_EXTENSION_VERSIONS

PG_AVAILABLE_EXTENSION_VERSIONS视图显示数据库中某些特性的扩展版本信息。

表 17-201 PG_AVAILABLE_EXTENSION_VERSIONS 字段

名称	类型	描述
name	name	扩展名。
version	text	版本名。
installed	boolean	如果此扩展的版本当前已经安装，则为真。
superuser	boolean	如果只允许系统管理员安装此扩展，则为真。
relocatable	boolean	如果扩展可以重新加载到另一个模式，则为真。
schema	name	扩展必须安装到的模式名，如果部分或全部可重新定位，则为NULL。
requires	name[]	必备扩展的名称，如果没有则为NULL。
comment	text	扩展的控制文件的注释字符串。

 版权所有 © 华为云计算技术有限公司

17.3.106 PG_AVAILABLE_EXTENSIONS

PG_AVAILABLE_EXTENSIONS视图显示数据库中某些特性的扩展信息。

表 17-202 PG_AVAILABLE_EXTENSIONS 字段

名称	类型	描述
name	name	扩展名。
default_version	text	缺省版本名，如果没有指定则为NULL。
installed_version	text	当前安装的扩展版本，如果未安装则为NULL。
comment	text	扩展控制文件的注释字符串。

17.3.107 PG_BULKLOAD_STATISTICS

在集群任一正常节点上，通过查询PG_BULKLOAD_STATISTICS视图可以获取当前登录节点正在进行的导入导出业务执行情况，其中每一个导入/导出业务对应一条记录。需要有系统管理员权限才可以访问此视图。

表 17-203 PG_BULKLOAD_STATISTICS 字段

名称	类型	描述
node_name	text	节点名称。
db_name	text	数据库名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
session_id	bigint	GDS的会话ID。
direction	text	业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
query	text	查询语句。
address	text	当前导入导出外表的location。
query_start	timestamp with time zone	导入/导出开始时间。
total_bytes	bigint	待处理数据的总大小。 仅GDS普通文件导入时，且该行记录来自CN节点才会显示，否则为空。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
phase	text	当前业务导入导出执行阶段，包括：INITIALIZING、TRANSFER_DATA、RELEASE_RESOURCE。
done_lines	bigint	已传输行数。
done_bytes	bigint	已传输字节数。

17.3.108 PG_COMM_CLIENT_INFO

PG_COMM_CLIENT_INFO视图存储单个节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

表 17-204 PG_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
query_id	bigint	查询ID，对应debug_query_id。
socket	integer	如果是物理连接，显示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，显示sid，显示-1时表示当前连接是物理连接。

17.3.109 PG_COMM_DELAY

PG_COMM_DELAY视图展示单个DN的通信库时延状态。

表 17-205 PG_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称。
remote_name	text	连接对端时延最大的节点名称。
remote_host	text	连接对端IP地址。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
stream_num	integer	当前物理连接使用的stream逻辑连接数量。
min_delay	integer	当前物理连接探测到的最小时延，单位微秒。
average	integer	当前物理连接探测时延的平均值，单位微秒。
max_delay	integer	当前物理连接探测到的最大时延，单位微秒。 **说明** 取值为-1，表示时延探测超时失败，请重新建立节点间连接后再执行查询。

17.3.110 PG_COMM_STATUS

PG_COMM_STATUS视图展示单个DN的通信库状态。

表 17-206 PG_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck	bigint	节点通信库接收包总量。
txpck	bigint	节点通信库发送包总量。
rxkB	bigint	节点通信库接收字节总量，单位KByte。
txkB	bigint	节点通信库发送字节总量，单位KByte。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位KByte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位KByte。
%USED(PM)	integer	postmaster线程实时使用率。
%USED (send)	integer	通信代理发送线程实时使用率。
%USED (recv)	integer	通信代理接收线程实时使用率。
%USED (CQ)	integer	RDMA通信完成事件线程实时使用率。
logic_conn	bigint	节点创建的DN间通信逻辑连接总数。
create_process	bigint	进程创建的子进程总数。该字段仅9.1.1.100及以上集群版本支持。
create_thread	bigint	进程创建的子线程总数。该字段仅9.1.1.100及以上集群版本支持。
create_conn	bigint	进程创建的连接总数，包含集群内、集群外连接。该字段仅9.1.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
accept_conn	bigint	进程接收的连接总数，包含集群内、集群外连接。该字段仅9.1.1.100及以上集群版本支持。

17.3.111 PG_COMM_RECV_STREAM

PG_COMM_RECV_STREAM视图展示单个DN上所有的通信库接收流状态。

表 17-207 PG_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接接收报文正常。 ● HOLD：表示逻辑连接接收报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
recv_bytes	bigint	通信流接收的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均接收速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte。

版权所有 © 华为云计算技术有限公司

17.3.112 PG_COMM_SEND_STREAM

PG_COMM_SEND_STREAM视图展示单个DN上所有的通信库发送流状态。

表 17-208 PG_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态。 ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接发送报文正常。 ● HOLD：表示逻辑连接发送报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

版权所有 © 华为云计算技术有限公司

17.3.113 PG_COMM_QUERY_SPEED

PG_COMM_QUERY_SPEED视图展示单个节点上所有query对应的流量信息。

表 17-209 PG_COMM_QUERY_SPEED 字段

名称	类型	描述
node_name	text	节点名称。
query_id	bigint	通信流对应的debug_query_id编号。
rxkB/s	bigint	查询对应的通信流接收速率，单位Byte/s。
txkB/s	bigint	查询对应的通信流发送速率，单位Byte/s。
rxkB	bigint	查询对应的通信流接收数据总量，单位Byte。
txkB	bigint	查询对应的通信流发送数据总量，单位Byte。
rxpck/s	bigint	查询对应的通信包接收速率，单位packages/s。
txpck/s	bigint	查询对应的通信包发送速率，单位packages/s。
rxpck	bigint	查询对应的通信包接收总量，单位packages。
txpck	bigint	查询对应的通信包发送总量，单位packages。

17.3.114 PG_CONTROL_GROUP_CONFIG

PG_CONTROL_GROUP_CONFIG视图存储系统的控制组配置信息。

表 17-210 PG_CONTROL_GROUP_CONFIG 字段

名称	类型	描述
pg_control_group_config	text	控制组的配置信息。

17.3.115 PG_CURSORS

PG_CURSORS视图列出了当前可用的游标。

表 17-211 PG_CURSORS 字段

名称	类型	描述
name	text	游标名。
statement	text	声明该游标时的查询语句。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
is_holdable	boolean	如果该游标是持久的（就是在声明该游标的事务结束后仍然可以访问该游标）则为TRUE，否则为FALSE。
is_binary	boolean	如果该游标被声明为BINARY则为TRUE，否则为FALSE。
is_scrollable	boolean	如果该游标可以滚动（就是允许以不连续的方式检索）则为TRUE，否则为FALSE。
creation_time	timestamp with time zone	声明该游标的时间戳。

17.3.116 PG_EXT_STATS

PG_EXT_STATS视图提供对存储在PG_STATISTIC_EXT表里面的扩展统计信息的访问。扩展统计信息目前包括多列统计信息。

表 17-212 PG_EXT_STATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
attname	int2vector	PG_STATISTIC_EXT.stakey	统计信息扩展的多列信息。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段组合为空的百分比。
avg_width	integer	-	字段组合记录以字节记的平均宽度。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_distinct	real	-	● 如果大于0，表示字段组合中独立数值的估计数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。 ● 如果等于0，表示独立数值的数目未知。 **说明** 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个字段组合中独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段组合中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段组合的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值均不为NULL。
most_common_freqs	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
most_common_vals_null	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值中至少有一个值为NULL。
most_common_freqs_null	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals_null是NULL，则为NULL。

 版权所有 © 华为云计算技术有限公司

17.3.117 PG_GET_INVALID_BACKENDS

PG_GET_INVALID_BACKENDS视图提供显示CN上连接到当前DN备机的后端线程信息。

表 17-213 PG_GET_INVALID_BACKENDS 字段

名称	类型	描述
pid	bigint	线程ID。
node_name	text	后端线程中连接的节点信息。
dbname	name	当前连接的数据库。
backend_start	timestamp with time zone	后端线程启动的时间。
query	text	后端线程正在执行的查询语句。

17.3.118 PG_GET_SENDERS_CATCHUP_TIME

PG_GET_SENDERS_CATCHUP_TIME视图显示单个DN上当前活跃的主备发送线程的追赶信息。

表 17-214 PG_GET_SENDERS_CATCHUP_TIME 字段

名称	类型	描述
pid	bigint	当前sender的线程ID。
lwpid	integer	当前sender的lwpid。
local_role	text	本地的角色。
peer_role	text	对端的角色。
state	text	当前sender的复制状态。
type	text	当前sender的类型。
catchup_start	timestamp with time zone	catchup启动的时间。
catchup_end	timestamp with time zone	catchup结束的时间。
catchup_type	text	catchup方式为全量还是增量。
catchup_bcm_filename	text	catchup当前执行的bcm文件。
catchup_bcm_finished	integer	catchup已经操作完成的bcm文件数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
catchup_bcm_total	integer	catchup总共需要操作的bcm文件数量。
catchup_percent	text	catchup已经操作完成的百分比。
catchup_remaining_time	text	catchup预估剩余时间。

17.3.119 PG_GLOBAL_TEMP_ATTACHED_PIDS

查看全局临时表在当前节点占有资源的会话信息。该视图仅8.2.1.220及以上集群版本支持。

表 17-215 PG_GLOBAL_TEMP_ATTACHED_PIDS 字段

名称	类型	描述
schemaname	name	模式名。
tablename	name	表名。
pid	bigint	用于表示会话的PID。

17.3.120 PG_GROUP

PG_GROUP视图查看数据库认证角色及角色之间的成员关系。

表 17-216 PG_GROUP 字段

名称	类型	描述
groname	name	组的名称。
grosysid	oid	组的ID。
grolist	oid[]	一个数组，包含这个组里面所有角色的ID。

17.3.121 PG_INDEXES

PG_INDEXES视图提供对数据库中每个索引的有用信息的访问。

表 17-217 PG_INDEXES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表和索引的模式名。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
tablename	name	**PG_CLASS**.relname	此索引所服务的表名。
indexname	name	**PG_CLASS**.relname	索引名。
tablespace	name	**PG_TABLESPACE**.spcname	包含索引的表空间名称。
indexdef	text	-	索引定义（一个重建的CREATE INDEX命令）。

应用示例

查询指定表的索引信息。

```
SELECT * FROM pg_indexes WHERE tablename = 'user_info';
```

```
gaussdb=> SELECT * FROM pg_indexes WHERE tablename = 'user_info';
 schemaname | tablename |      indexname       | tablespace |                                        indexdef
------------+-----------+----------------------+------------+------------------------------------------------------------------------------------------
 public     | user_info | idx_user_info_phone |            | CREATE UNIQUE INDEX idx_user_info_phone ON user_info USING btree (phone) TABLESPACE pg_default
(1 row)
```

查询当前数据库指定模式下所有表的索引信息。

```
SELECT tablename, indexname, indexdef FROM pg_indexes WHERE schemaname = 'public' ORDER BY
tablename,indexname;
```

```
gaussdb=> SELECT tablename, indexname, indexdef FROM pg_indexes WHERE schemaname = 'public' ORDER BY tablename,indexname;
    tablename     |      indexname       |                                        indexdef

------------------+----------------------+------------------------------------------------------------------------------------------
------------------
 customer         | customer_pkey        | CREATE UNIQUE INDEX customer_pkey ON customer USING btree (c_customer_sk) TABLESPACE pg_default
 customer_address | customer_address_idx | CREATE INDEX customer_address_idx ON customer_address USING btree (ca_address_sk) LOCAL(PARTITION ca_address_sk_idx1, PARTITION ca_address_sk_idx2, PARTITION ca_address_sk_idx3, PARTITION ca_address_sk_idx4)  TAB
LESPACE pg_default
 user_info        | idx_user_info_phone  | CREATE UNIQUE INDEX idx_user_info_phone ON user_info USING btree (phone) TABLESPACE pg_default
(3 rows)
```

17.3.122 PG_JOB

PG_JOB视图存储用户创建的定时任务的任务详细信息。

PG_JOB视图用于代替历史版本的PG_JOB系统表，提供对之前版本的前向兼容。原PG_JOB系统表已经变更为PG_JOBS系统表，关于PG_JOBS系统表的描述详见**PG_JOBS**。

表 17-218 PG_JOB 字段

名字	类型	描述
job_id	bigint	作业ID。
current_postgres_pid	bigint	如果当前任务正在被执行，那么此处记录运行此任务的postgres线程ID。默认为-1，表示此任务未被执行或已执行完毕。
log_user	name	创建者的UserName。
priv_user	name	作业执行者的UserName。

名字	类型	描述
dbname	name	标识作业执行的数据库名。
node_name	name	标识当前作业是在哪个CN上创建和执行。
job_status	text	当前任务的执行状态，取值范围：('r', 's', 'f', 'd','p','w','l')，默认为's'，取值含义： ● r=running ● s=successfully finished ● f=job failed ● d=disable ● p=pending ● w=waiting ● l=launching **说明** ● 当用户将定时任务关闭（即job_queue_processes为0时），由于监控job执行的线程不会启动，所以job_status不会根据job的实时状态进行设置，用户不需要关注job_status。 ● 只有当开启定时任务功能（job_queue_processes为非0时），系统才会根据当前job的实时状态刷新job_status的值。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。
nspname	name	作业运行时所在的命名空间的名称。

 版权所有 © 华为云计算技术有限公司

名字	类型	描述
what	text	作业内容。

17.3.123 PG_JOB_PROC

PG_JOB_PROC视图用于代替之前版本的PG_JOB_PROC系统表，提供对之前版本的前向兼容。原PG_JOB_PROC系统表已经和原PG_JOB系统表一同并入当前版本的PG_JOBS系统表，关于PG_JOBS系统表的描述详见**PG_JOBS**。

表 17-219 PG_JOB_PROC 字段

名字	类型	描述
job_id	bigint	作业ID。
what	text	作业内容。

17.3.124 PG_JOB_SINGLE

PG_JOB_SINGLE视图用于显示当前节点的作业信息。

表 17-220 PG_JOB_SINGLE 字段

名字	类型	描述
job_id	bigint	作业ID。
current_postgres_pid	bigint	如果当前任务正在被执行，那么此处记录运行此任务的postgres线程ID。默认为-1，表示此任务未被执行或已执行完毕。
log_user	name	创建者的UserName。
priv_user	name	作业执行者的UserName。
dbname	name	标识作业执行的数据库名。
node_name	name	标识当前作业是在哪个CN上创建和执行。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
job_status	text	当前任务的执行状态，取值范围：('r', 's', 'f', 'd','p','w','l')，默认为's'，取值含义： ● r=running ● s=successfully finished ● f=job failed ● d=disable ● p=pending ● w=waiting ● l=launching **说明** ● 当用户将定时任务关闭（即job_queue_processes为0时），由于监控job执行的线程不会启动，所以job_status不会根据job的实时状态进行设置，用户不需要关注job_status。 ● 只有当开启定时任务功能（job_queue_processes为非0时），系统才会根据当前job的实时状态刷新job_status的值。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。
failure_count	smallint	连续失败计数。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。
nspname	name	作业运行时所在的命名空间的名称。
what	text	作业内容。

 版权所有 © 华为云计算技术有限公司

17.3.125 PG_LIFECYCLE_DATA_DISTRIBUTE

PG_LIFECYCLE_DATA_DISTRIBUTE视图查询OBS多温表中冷热数据分布情况。

表 17-221 PG_LIFECYCLE_DATA_DISTRIBUTE 字段

名称	类型	描述
schemaname	name	模式名。
tablename	name	当前表名。
nodename	name	节点名。
hotpartition	text	该DN节点上的热分区。
coldpartition	text	该DN节点上的冷分区。
switchablepartition	text	该DN节点上的可切分区。
hotdatasize	text	该DN节点上的热分区数据大小。
colddatasize	text	该DN节点上的冷分区数据大小。
switchabledatasize	text	该DN节点上的可切分区数据大小。

17.3.126 PG_LOCKS

PG_LOCKS视图存储各打开事务所持有的锁信息。

表 17-222 PG_LOCKS 字段

名称	类型	引用	描述
locktype	text	-	被锁定对象的类型：relation，extend，page，tuple，transactionid，virtualxid，object，userlock，advisory。
database	oid	PG_DATABASE.oid	被锁定对象所在数据库的OID。 • 如果被锁定的对象是共享对象，则OID为0。 • 如果是一个事务ID，则为NULL。
relation	oid	PG_CLASS.oid	被锁定对象关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。
page	integer	-	关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
tuple	smallint	-	页面里边的行编号，如果对象不是行，则为NULL。
virtualxid	text	-	事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。
transactionid	xid	-	事务的ID，如果对象不是一个事务ID，则为NULL。
classid	oid	PG_CLASS.oid	包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。
objid	oid	-	对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。
objsubid	smallint	-	对于表的某个字段对应为字段编号；对于其他对象类型，该字段为0；如果该对象不是普通数据库对象，则为NULL。
virtualtransaction	text	-	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	-	持有此锁或者等待此锁的服务器线程的逻辑ID。如果锁被一个预备事务持有，则为NULL。
mode	text	-	此线程持有的或者是期望持有的锁模式。更多有关锁模式的内容请参见LOCK章节。
granted	boolean	-	• 如果锁是持有锁，则为TRUE。 • 如果锁是等待锁，则为FALSE。
fastpath	boolean	-	如果通过fast-path获得锁，则为TRUE；如果通过主锁表获得，则为FALSE。
waittime	timestamp with time zone	-	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	-	开始持有锁的时间戳。该字段仅9.1.0.200及以上集群版本支持。

17.3.127 PG_LOCKWAIT_DETAIL

PG_LOCKWAIT_DETAIL视图显示当前实例中锁等待链详细信息。如果存在多级的锁等待关系，会依次将整个锁等待链按照等待顺序显示出来。

该视图8.2.1集群版本中8.2.1.251及以上版本支持；9.1.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-223 PG_LOCKWAIT_DETAIL 字段

名称	类型	描述
level	integer	锁等待链中的层级，以1开始，每显示一层等待关系level会加1。
node_name	name	节点名称，对应pgxc_node表中的node_name列。
lock_wait_hierarchy	text	锁等待链，以节点名称：进程号->等待进程号->等待进程号->...。
lock_type	text	被锁定对象的类型。
database	oid	被锁定对象所在数据库的OID。
relation	oid	被锁定对象关系的OID。
page	integer	关系内部的页面编号。
tuple	smallint	页面的行编号。
virtual_xid	text	事务的虚拟ID。
transaction_id	xid	事务ID。
class_id	oid	包含该对象的系统表的OID。
obj_id	oid	对象在其系统表内的OID。
obj_subid	smallint	对于表的列字段编号。
virtual_transaction	text	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	持有此锁或者等待此锁的线程号。
mode	text	锁级别。
granted	boolean	是否持有锁。
fastpath	boolean	是否通过fastpath机制获得锁。
wait_for_pid	bigint	锁冲突线程的线程号。
conflict_mode	text	锁冲突线程持有的冲突锁级别。
query_id	bigint	查询语句的ID。
query	text	查询语句。
application_name	text	连接到该后端的应用名。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
xact_start	timestamp with time zone	当前事务的启动时间。
query_start	timestamp with time zone	开始当前活跃查询的时间。
state	text	后端当前总体状态。
waittime	timestamp with time zone	开始等待锁的时间戳。
holdtime	timestamp with time zone	开始获取锁的时间戳。

应用示例

步骤1 连接DN节点，开启一个事务，执行查询。

```
BEGIN;
SELECT * FROM t1;
```

步骤2 另一个窗口连接CN，对表t1执行TRUNCATE。

```
TRUNCATE t1;
```

此时TRUNCATE会被阻塞。

步骤3 另开一个窗口连接DN，查询PG_LOCKWAIT_DETAIL视图，获得当前的锁等待情况，帮助诊断阻塞原因。

```
SELECT * FROM PG_LOCKWAIT_DETAIL;
 level | node_name |              lock_wait_hierarchy               | lock_type | database | relation | page | tuple |
virtual_xid | transaction_id | class_id | obj_id | obj_subid | virtual_transaction |       pid       |       mode        |
granted | fastpath |  wait_for_pid   |  conflict_mode  |     query_id      |          query          | application_n
ame |         backend_start         |          xact_start           |          query_start          | state  |
waittime           |          holdtime
-------+-----------+-----------------------------------------------+-----------+----------+----------+------+-------
+-------------+----------------+----------+--------+-----------+---------------------+-----------------
+---------------------+---------+----------+-----------------+-----------------+------------------
+-------------------------+--------------
----+--------------------------------+--------------------------------+--------------------------------+--------
+-------------------------------+-------------------------------
    1 | datanode1 | datanode1:140371308516512                    | relation  |    16564 |    25224 |      |
|            |        3956258 |          |        |           | 15/204              | 140371308516512 | AccessExclusiveLock |
f       | f       | 140371308446552 | AccessShareLock | 77687093572143017 | TRUNCATE t1              |
coordinator1
   | 2025-02-13 15:07:35.168383+08 | 2025-02-13 15:11:23.613945+08 | 2025-02-13 15:11:23.651944+08 |
active | 2025-02-13 15:11:23.65369+08 |
    2 | datanode1 | datanode1:140371308516512 -> 140371308446552 | relation  |    16564 |    25224 |
|      |            |               |          |        |           | 13/18               | 140371308446552 | AccessShareLock     |
t       | f       |                |               |                0 | begin;select * from t1; | gsql
   | 2025-02-13 14:15:58.989993+08 | 2025-02-13 15:10:40.594882+08 | 2025-02-13 15:10:40.594882+08 |
active |                               | 2025-02-13 15:10:40.596243+08
(2 rows)
```

----结束

 版权所有 © 华为云计算技术有限公司

17.3.128 PG_LWLOCKS

PG_LWLOCKS视图提供当前实例正在持有的或等待的轻量级锁信息。该视图仅9.1.0.200及以上集群版本支持。

表 17-224 PG_LWLOCKS 字段

名称	类型	描述
pid	bigint	后端线程ID。
query_id	bigint	查询语句的ID。
lwtid	integer	后端线程的轻量级线程号。
reqlockid	integer	当前线程正在请求的轻量级锁ID。
reqlock	text	reqlockid对应的轻量级锁名称。
heldlocknums	integer	当前线程已经获得的轻量级锁的数量。
heldlockid	integer	当前线程已经获得的轻量级锁ID。
heldlock	text	heldlockid对应的轻量级锁名称。
heldlockmode	text	heldlockid对应的轻量级锁模式。

示例

使用PG_LWLOCKS视图查询当前实例正在持有的或等待的轻量级锁信息：

```
SELECT * FROM pg_lwlocks;
       pid       |     query_id      | lwtid | reqlockid | reqlock | heldlocknums | heldlockid |      heldlock      |
heldlockmode
-----------------+-------------------+-------+-----------+---------+--------------+------------+--------------------
+--------------
 139810224192480 |                 0 | 54842 |           |         |            1 |          7 | WALWriteLock       | Exclusive
 139810224199520 | 78250043526306022 | 54963 |           |         |            1 |     193860 |
BUFFER_POOL_LWLOCK | Exclusive
(2 rows)
```

17.3.129 PG_NODE_ENV

PG_NODE_ENV视图提供获取当前节点的环境变量信息。

表 17-225 PG_NODE_ENV 字段

名称	类型	描述
node_name	text	当前节点名称。
host	text	当前节点的主机名称。
process	integer	当前节点的进程号。
port	integer	当前节点的端口号。
installpath	text	当前节点的安装目录。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
datapath	text	当前节点的数据目录。
log_directory	text	当前节点的日志目录。

17.3.130 PG_OS_THREADS

PG_OS_THREADS视图提供当前节点下所有线程的状态信息。

表 17-226 PG_OS_THREADS 字段

名称	类型	描述
node_name	text	当前节点名称。
pid	bigint	当前节点进程中正在运行的线程号。
lwpid	integer	与pid对应的轻量级线程号。
thread_name	text	与pid对应的线程名称。
creation_time	timestamp with time zone	与pid对应的线程创建的时间。

17.3.131 PG_POOLER_STATUS

PG_POOLER_STATUS视图查询pooler中的缓存连接状态。该视图只能在CN上执行查询，显示本地CN的pooler模块的连接缓存信息。

表 17-227 PG_POOLER_STATUS 字段

名称	类型	描述
database	text	数据库名称。
user_name	text	用户名。
tid	bigint	连接CN的线程ID。
node_oid	bigint	连接的实例节点OID。
node_name	name	连接的实例节点名称。
in_use	boolean	连接是否正被使用。 ● t（true）：表示连接正在使用。 ● f（false）：表示连接没有使用。
fdsock	bigint	对端socket。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
remote_pid	bigint	对端线程号。
session_params	text	由此连接下发的GUC session参数。

应用示例

查看pooler连接池信息：

```
SELECT database,user_name,node_name,in_use,count(*) FROM pg_pooler_status GROUP BY 1, 2, 3 ,4
ORDER BY 5 DESC LIMIT 50;
```

```
gaussdb=> select database,user_name,node_name,in_use,count(*) from pg_pooler_status group by 1, 2, 3 ,4 order by 5 desc limit 50;
 database | user_name |  node_name   | in_use | count
----------+-----------+--------------+--------+-------
 postgres | Ruby      | dn_6003_6004 | t      |     2
 postgres | Ruby      | dn_6005_6006 | t      |     2
 postgres | Ruby      | dn_6001_6002 | t      |     2
 postgres | Ruby      | dn_6001_6002 | f      |     1
 postgres | Ruby      | dn_6005_6006 | f      |     1
 gaussdb  | Ruby      | dn_6001_6002 | f      |     1
 postgres | Ruby      | cn_5002      | f      |     1
 postgres | Ruby      | cn_5001      | f      |     1
 postgres | Ruby      | dn_6003_6004 | f      |     1
 postgres | Ruby      | cn_5002      | t      |     1
(10 rows)
```

17.3.132 PG_PREPARED_STATEMENTS

PG_PREPARED_STATEMENTS视图显示当前会话所有可用的预备语句。

表 17-228 PG_PREPARED_STATEMENTS 字段

名称	类型	描述
name	text	预备语句的标识符。
statement	text	创建该预备语句的查询字符串。对于从SQL创建的预备语句而言是客户端提交的PREPARE语句；对于通过前/后端协议创建的预备语句而言是预备语句自身的文本。
prepare_time	timestamp with time zone	创建该预备语句的时间戳。
parameter_types	regtype[]	该预备语句期望的参数类型，以regtype类型的数组格式出现。与该数组元素相对应的OID可以通过把regtype转换为OID值获取。
from_sql	boolean	• 如果该预备语句是通过PREPARE语句创建的则为true。 • 如果是通过前/后端协议创建的则为false。

 版权所有 © 华为云计算技术有限公司

17.3.133 PG_PREPARED_XACTS

PG_PREPARED_XACTS视图显示当前准备好进行两阶段提交的事务的信息。

表 17-229 PG_PREPARED_XACTS 字段

名称	类型	引用	描述
transaction	xid	-	预备事务的数字事务标识。
gid	text	-	赋予该事务的全局事务标识。
prepared	timestamp with time zone	-	事务准备好提交的时间。
owner	name	PG_AUTHID.rolname	执行该事务的用户名。
database	name	PG_DATABASE.datname	执行该事务所在的数据库名。

17.3.134 PG_PUBLICATION_TABLES

PG_PUBLICATION_TABLES视图显示发布与其所发布的表之间的映射信息。和底层的系统表PG_PUBLICATION_REL不同，这个视图展开了定义为FOR ALL TABLES和FOR ALL TABLES IN SCHEMA的发布，对这类发布来说，每一个可发布的表都有一行。该系统视图仅8.2.0.100及以上集群版本支持。

表 17-230 PG_PUBLICATION_TABLES 字段

名称	类型	描述
pubname	name	发布的名称。
schemaname	name	表的模式的名称。
tablename	name	表的名称。

应用示例

查询所有发布表。

```
SELECT * FROM PG_PUBLICATION_TABLES;
 pubname | schemaname | tablename
---------+------------+-----------
 mypub   | public     | t1
 mypub   | public     | t2
(2 rows)
```

17.3.135 PG_QUERYBAND_ACTION

PG_QUERYBAND_ACTION视图显示query_band关联行为和次序。

版权所有 © 华为云计算技术有限公司

表 17-231 PG_QUERYBAND_ACTION 字段

名称	类型	描述
qband	text	query_band键值对。
respool_id	oid	query_band关联资源池OID。
respool	text	query_band关联资源池名。
priority	text	query_band关联队列内优先级。
qborder	integer	query_band搜索次序。

17.3.136 PG_REPLICATION_SLOTS

PG_REPLICATION_SLOTS视图查看复制节点的信息。

表 17-232 PG_REPLICATION_SLOTS 字段

名称	类型	描述
slot_name	text	复制节点的名称。
plugin	name	逻辑复制槽对应的输出插件名。
slot_type	text	复制节点的类型。
datoid	oid	复制节点的数据库OID。
database	name	复制节点的数据库名称。
active	boolean	复制节点是否为激活状态。
xmin	xid	复制节点事务标识。
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识。
restart_lsn	text	复制节点的Xlog文件信息。
dummy_standby	boolean	复制节点是否为假备。

17.3.137 PG_ROLES

PG_ROLES视图提供访问数据库角色的相关信息。

表 17-233 PG_ROLES 字段

名称	类型	引用	描述
rolname	name	-	角色名。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
rolsuper	boolean	-	该角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	-	该角色是否继承角色的权限。
rolcreaterole	boolean	-	该角色是否可以创建其他的角色。
rolcreatedb	boolean	-	该角色是否可以创建数据库。
rolcatupdate	boolean	-	该角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	-	该角色是否可以登录数据库。
rolreplication	boolean	-	该角色是否可以复制。
rolauditadmin	boolean	-	该角色是否为审计管理员。
rolsystemadmin	boolean	-	该角色是否为系统管理员。
rolconnlimit	integer	-	限制单个用户在单个CN上的最大并发连接数。-1表示无限制。
rolpassword	text	-	不是口令，总是********。
rolvalidbegin	timestamp with time zone	-	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	-	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
rolrespool	name	-	用户所能够使用的resource pool。
rolparentid	oid	PG_AUTHID.rolparentid	用户所在组用户的OID。
roltabspace	text	-	用户永久表存储空间限额。
roltempspace	text	-	用户临时表存储空间限额。
rolspillspace	text	-	用户算子落盘空间限额。
rolconfig	text[]	-	运行时配置变量的会话缺省。
oid	oid	PG_AUTHID.oid	角色的ID。
roluseft	boolean	PG_AUTHID.roluseft	角色是否可以操作外表。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
nodegroup	name	-	角色所关联的逻辑集群名字，如果没有关联逻辑集群，该值为空。

17.3.138 PG_RULES

PG_RULES视图提供对查询重写规则的有用信息访问的接口。

表 17-234 PG_RULES 字段

名称	类型	描述
schemaname	name	包含表的模式名。
tablename	name	规则作用的表名。
rulename	name	规则的名称。
definition	text	规则定义（一个重新构造的创建命令）。

17.3.139 PG_RUNNING_XACTS

PG_RUNNING_XACTS视图显示当前节点运行事务的信息。

表 17-235 PG_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄。
gxid	xid	事务ID号。
state	tinyint	事务状态（3：prepared或者0：starting）。
node	text	节点名称。
xmin	xid	节点上当前数据涉及的最小事务号xmin。
vacuum	boolean	标志当前事务是否是lazy vacuum事务。
timeline	bigint	标志数据库重启次数。
prepare_xid	xid	处于prepared状态的事务的ID号，若不在prepared状态，值为0。
pid	bigint	事务对应的线程ID。
next_xid	xid	CN传给DN的事务ID号。

版权所有 © 华为云计算技术有限公司

17.3.140 PG_SECLABELS

PG_SECLABELS视图提供关于安全标签的信息。

表 17-236 PG_SECLABELS 字段

名称	类型	引用	描述
objoid	oid	任意OID属性	安全标签所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象的系统表的OID。
objsubid	integer	-	对于某个在表字段上的安全标签，为字段编号（引用表本身的objoid和classoid）。对于所有其他对象类型，该字段为0。
objtype	text	-	标签出现的对象的类型。
objnamespace	oid	PG_NAMESPACE.oid	对象的命名空间的OID，如果适用；否则为NULL。
objname	text	-	标签适用的对象名。
provider	text	PG_SECLABEL.provider	与标签相关的标签提供者。
label	text	PG_SECLABEL.label	应用于此对象的安全标签。

17.3.141 PG_SEQUENCES

PG_SEQUENCES视图显示当前用户有权限的序列属性信息。该视图仅9.1.0及以上集群版本支持。

表 17-237 PG_SEQUENCES 字段

名称	类型	描述
schemaname	name	命名空间的名称。
sequencename	name	序列的名称。
sequenceowner	name	序列的所有者。
start_value	bigint	序列的起始值。
min_value	bigint	序列生成的最小值。
max_value	bigint	序列生成的最大值。
increment_by	bigint	序列生成的步长，即每次生成的值增加的量。
cycle	boolean	如果设置为true，则在达到最大值后重新开始从最小值生成序列值；如果设置为false，则在达到最大值后停止生成序列值。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
cache_size	bigint	序列缓存值的大小。
last_value	bigint	序列最近一次生成的值。

17.3.142 PG_SESSION_WLMSTAT

PG_SESSION_WLMSTAT视图显示和当前用户执行作业正在运行时的负载管理相关信息。

表 17-238 PG_SESSION_WLMSTAT 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
datname	name	连接后端的数据库名称。
threadid	bigint	后端线程ID。
processid	integer	后端线程的PID。
usesysid	oid	登录后端的用户OID。
appname	text	连接到后端的应用名。
usename	name	登录到该后端的用户名。
priority	bigint	语句所在Cgroups的优先级，数值越大表示作业优先级越高。 ● 0：表示该作业不受负载管理管控。 ● 1：LOW ● 2：MEDIUM ● 4：HIGH ● 8：RUSH
attribute	text	语句的属性： ● Ordinary：语句发送到数据库后被解析前的默认属性。 ● Simple：简单语句。 ● Complicated：复杂语句。 ● Internal：数据库内部语句。
block_time	bigint	语句当前为止的pending的时间，单位s。
elapsed_time	bigint	语句当前为止的实际执行时间，单位s。
total_cpu_time	bigint	语句在上一时间周期内的DN上CPU使用的总时间，单位s。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
cpu_skew_percent	integer	语句在上一时间周期内的DN上CPU使用的倾斜率。
statement_mem	integer	语句执行所需要的估算内存。
active_points	integer	语句占用的资源池并发点数。
dop_value	integer	语句的从资源池中获取的dop值。
control_group	text	语句当前所使用的Cgroups。
status	text	语句当前的状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active：非以上四种状态外的正常状态。 ● unknown：未知状态。
enqueue	text	语句当前的排队情况，包括： ● Global：全局排队。 ● Respool：资源池排队。 ● CentralQueue：在中心协调节点(CCN)中排队。 ● Transaction：语句处于一个事务块中。 ● StoredProc：语句处于一个存储过程中。 ● None：未在排队。 ● Forced None：事务块语句或存储过程语句由于超出设定的等待时间而强制执行。
resource_pool	name	语句当前所在的资源池。
query	text	该后端的最新查询。如果status是active，此字段显示当前正在执行的查询。所有其他情况表示上一个查询。
is_plana	bool	逻辑集群模式下，语句当前是否占用其他逻辑集群的资源执行。该值默认为f，表示不占用其他逻辑集群的资源执行。
node_group	text	语句所属用户对应的逻辑集群。
lane	text	表示语句查询的快慢车道。 ● fast：快车道。 ● slow：慢车道。 ● none：未管控。

 版权所有 © 华为云计算技术有限公司

17.3.143 PG_SESSION_IOSTAT

PG_SESSION_IOSTAT视图8.1.2版本中已废弃，为兼容历史版本功能保留该视图，当前版本查询无效。可使用**PGXC_WLM_SESSION_STATISTICS**视图查看所有CN上正在执行的作业的负载管理信息。

表 17-239 PG_SESSION_IOSTAT 字段

名称	类型	描述
query_id	bigint	作业ID。
mincurriops	integer	该作业当前io在各DN中的最小值。
maxcurriops	integer	该作业当前io在各DN中的最大值。
minpeakiops	integer	在作业运行时，作业io峰值中，各DN的最小值。
maxpeakiops	integer	在作业运行时，作业io峰值中，各DN的最大值。
io_limits	integer	该作业所设GUC参数io_limits。
io_priority	text	该作业所设GUC参数io_priority。
query	text	作业。
node_group	text	作业所属用户对应的逻辑集群。

17.3.144 PG_SETTINGS

PG_SETTINGS视图显示数据库运行时参数的相关信息。

表 17-240 PG_SETTINGS 字段

名称	类型	描述
name	text	参数名称。
setting	text	参数当前值。
unit	text	参数的隐式结构。
category	text	参数的逻辑组。
short_desc	text	参数的简单描述。
extra_desc	text	参数的详细描述。
context	text	设置参数值的上下文，包括internal，postmaster，sighup，backend，superuser和user。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
vartype	text	参数类型，包括bool，enum，integer，real和string。
source	text	参数的赋值方式。
min_val	text	参数最小值。如果参数类型不是数值型，那么该字段值为null。
max_val	text	参数最大值。如果参数类型不是数值型，那么该字段值为null。
enumvals	text[]	enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。
boot_val	text	数据库启动时参数默认值。
reset_val	text	数据库重置时参数默认值。
sourcefile	text	设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。
sourceline	integer	设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。
sync_to_stream	boolean	标记是否在Stream线程同步设置参数的变更情况。该字段仅9.1.1.300及以上集群版本支持。

17.3.145 PG_SHADOW

PG_SHADOW视图显示了所有在PG_AUTHID中标记了rolcanlogin的角色的属性。

此系统视图的名称源于它不是所有用户可读的，因为包含口令。PG_USER是一个在PG_SHADOW上全局可读的视图，只是把口令域填充成了空白。

表 17-241 PG_SHADOW 字段

名字	类型	引用	描述
usename	name	PG_AUTHID.rolname	用户名。
usesysid	oid	PG_AUTHID.oid	用户的ID。
usecreatedb	boolean	-	用户可以创建数据库。
usesuper	boolean	-	用户是系统管理员。
usecatupd	boolean	-	用户可以更新系统表。即使是系统管理员，如果此字段不为真，也不能更新系统表。

版权所有 © 华为云计算技术有限公司

名字	类型	引用	描述
userepl	boolean	-	用户可以初始化流复制和使系统处于或不处于备份模式。
passwd	text	-	口令（可能是加密的）；如果没有则为null。参阅PG_AUTHID获取加密的口令是如何存储的信息。
valbegin	timestamp with time zone	-	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	-	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	-	用户使用的资源池。
parent	oid	-	父资源池。
spacelimit	text	-	永久表存储空间限额。
tempspacelimit	text	-	临时表存储空间限额。
spillspacelimit	text	-	算子落盘空间限额。
useconfig	text[ ]	-	运行时配置变量的会话缺省。

17.3.146 PG_SHARED_MEMORY_DETAIL

PG_SHARED_MEMORY_DETAIL视图查询所有已产生的共享内存上下文的使用信息。

表 17-242 PG_SHARED_MEMORY_DETAIL 字段

名字	类型	描述
contextname	text	内存上下文的名字。
level	smallint	当前上下文在整体内存上下文中的层级。
parent	text	上级内存上下文。
totalsize	bigint	共享内存总大小，单位Byte。
freesize	bigint	共享内存剩余大小，单位Byte。
usedsize	bigint	共享内存使用大小，单位Byte。

版权所有 © 华为云计算技术有限公司

17.3.147 PG_STATS

PG_STATS视图提供对存储在pg_statistic表里面的单列统计信息的访问。

表 17-243 PG_STATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
attname	name	PG_ATTRIBUTE.attname	字段名。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 • 如果大于0，表示独立数值的实际数目。 • 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 • 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
partname	name	-	分区名。该字段仅8.2.1.200及以上集群版本支持。

17.3.148 PG_STATS_EXT_EXPRS

PG_STATS_EXT_EXPRS视图提供对存储在PG_STATISTIC_DATA表里面的表达式的统计信息。该视图仅8.2.1.200及以上集群版本支持。

表 17-244 PG_STATS_EXT_EXPRS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
statistics_schemaname	name	-	表达式统计信息对象所在模式名。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
statistics_name	name	-	表达式统计信息对象的名称。
statistics_owner	name	-	表达式统计信息对象的所有者。
expr	text	-	表达式
inherited	boolean	-	如果为true，则包含继承的子列；如果为false，则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	● 如果大于0，表示字段中独立数值的估计数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。

17.3.149 PG_STAT_ACTIVITY

PG_STAT_ACTIVITY视图显示和当前用户查询相关的信息。若有管理员权限或预置角色权限可以显示和所有用户查询相关的信息。

表 17-245 PG_STAT_ACTIVITY 字段

名称	类型	描述
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
lwtid	integer	轻量级线程ID。
usesysid	oid	登录该后端的用户OID。
usename	name	登录该后端的用户名。
application_name	text	连接到该后端的应用名。
client_addr	inet	连接到该后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
client_hostname	text	客户端的主机名，此字段是通过client_addr的反向DNS查找得到。此字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁或者等待节点则为t，否则为f。
enqueue	text	语句当前排队状态。可能值是： ● waiting in global queue：表示语句在全局并发队列排队中，主要包含并发数超过单CN配置的max_active_statements。 ● waiting in respool queue：表示语句在资源池排队中，简单作业并发受限，主要是简单作业并发超过快车道并发上限max_dop。 ● waiting in ccn queue：表示作业在CCN排队中，包含全局内存排队和慢车道内存和并发排队，包含以下场景： 1. 全局可用内存超过上限，进行全局内存队列排队。 2. 资源池慢车道并发上限，即资源池并发超过active_statements上限。 3. 资源池慢车道内存上限，即资源池并发作业估算内存超过mem_percent计算的上限。 ● 空或no waiting queue：表示语句正在运行。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	后端当前总体状态。可能值是： • active：后台正在执行查询。 • idle：后台正在等待新的客户端命令。 • idle in transaction：后端在事务中，但事务中没有语句在执行。 • idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 • fastpath function call：后端正在执行一个fast-path函数。 • disabled：如果后端禁用track_activities，则报告此状态。 **说明** 普通用户只能查看到自己账户所对应的会话状态。即其他账户的state信息为空。
resource_pool	name	用户使用的资源池。
stmt_type	text	语句类型。
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

17.3.150 PG_STAT_ALL_INDEXES

PG_STAT_ALL_INDEXES视图显示当前数据库中所有访问特定索引的统计信息。

索引可以通过简单的索引扫描或“位图”索引扫描进行使用。位图扫描中几个索引的输出可以通过AND或者OR规则进行组合， 因此当使用位图扫描的时候，很难将独立对行抓取与特定索引进行组合， 因此，一个位图扫描增加
pg_stat_all_indexes.idx_tup_read使用索引计数，并且增加
pg_stat_all_tables.idx_tup_fetch表计数，但不影响
pg_stat_all_indexes.idx_tup_fetch。

表 17-246 PG_STAT_ALL_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
relname	name	索引的表名。
indexrelname	name	索引名。
idx_scan	bigint	索引上开始的索引扫描数。
idx_tup_read	bigint	通过索引上扫描返回的索引项数。
idx_tup_fetch	bigint	通过使用索引的简单索引扫描抓取的活表行数。

17.3.151 PG_STAT_ALL_TABLES

PG_STAT_ALL_TABLES视图显示当前数据库中与表访问相关的统计信息（包括TOAST表）。

表 17-247 PG_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
vacuum_count	bigint	vacuum次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。
autoanalyze_count	bigint	autoanalyze次数。
last_data_changed	timestamp with time zone	记录该表最后一次数据发生变化的时间（引起数据变化的操作包括INSERT/UPDATE/DELETE、EXCHANGE/TRUNCATE/DROP partition），该列数据仅在本地CN记录。

应用示例

查询表table_test最后一次数据发生变化的时间：

```
SELECT last_data_changed FROM PG_STAT_ALL_TABLES WHERE relname ='user_info';
```

```
gaussdb=> SELECT last_data_changed FROM PG_STAT_ALL_TABLES WHERE relname ='user_info';
       last_data_changed
-------------------------------
 2026-06-04 14:55:34.142767+08
(1 row)
```

17.3.152 PG_STAT_BAD_BLOCK

PG_STAT_BAD_BLOCK视图显示自节点启动后，读取数据时出现Page/CU校验失败的统计信息。

表 17-248 PG_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称。
databaseid	oid	数据库OID。
tablespaceid	oid	表空间OID。
relfilenode	oid	文件对象ID。

版权所有 © 华为云计算技术有限公司

名字	类型	描述
forknum	integer	文件类型。
error_count	integer	出现校验失败的次数。
first_time	timestamp with time zone	第一次出现时间。
last_time	timestamp with time zone	最近一次出现时间。

17.3.153 PG_STAT_BGWRITER

PG_STAT_BGWRITER视图显示关于后端写进程活动的统计信息。

表 17-249 PG_STAT_BGWRITER 字段

名称	类型	描述
checkpoints_timed	bigint	定期执行的检查点数量。
checkpoints_req	bigint	请求执行的检查点数量。
checkpoint_write_time	double precision	检查点期间将文件写入磁盘花费的时间，以毫秒为单位。
checkpoint_sync_time	double precision	检查点期间数据同步到磁盘花费的时间，以毫秒为单位。
buffers_checkpoint	bigint	检查点期间写入缓冲区的数量。
buffers_clean	bigint	后端写进程写的缓冲区数量。
maxwritten_clean	bigint	由于写入缓冲区太多，后端写进程停止清理扫描的次数。
buffers_backend	bigint	后端直接写入的缓冲区数量。
buffers_backend_fsync	bigint	后端需要fsync的次数。
buffers_alloc	bigint	分配的缓冲区数量。
stats_reset	timestamp with time zone	统计重置的时间。

版权所有 © 华为云计算技术有限公司

17.3.154 PG_STAT_DATABASE

PG_STAT_DATABASE视图显示当前节点上每个数据库的状态和统计信息。

表 17-250 PG_STAT_DATABASE 字段

名称	类型	描述
datid	oid	数据库OID。
datname	name	数据库名。
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。
tup_returned	bigint	当前节点上该数据库查询返回的行数。
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。
tup_inserted	bigint	当前节点上该数据库插入的行数。
tup_updated	bigint	当前节点上该数据库更新的行数。
tup_deleted	bigint	当前节点上该数据库删除的行数。
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。可参见PG_STAT_DATABASE_CONFLICTS。
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。

17.3.155 PG_STAT_DATABASE_CONFLICTS

PG_STAT_DATABASE_CONFLICTS视图显示数据库冲突状态的统计信息。

表 17-251 PG_STAT_DATABASE_CONFLICTS 字段

名称	类型	描述
datid	oid	数据库OID。
datname	name	数据库名。
confl_tablespace	bigint	冲突的表空间的数目。
confl_lock	bigint	冲突的锁数目。
confl_snapshot	bigint	冲突的快照数目。
confl_bufferpin	bigint	冲突的缓冲区数目。
confl_deadlock	bigint	冲突的死锁数目。

17.3.156 PG_STAT_GET_MEM_MBYTES_RESERVED

PG_STAT_GET_MEM_MBYTES_RESERVED视图显示线程在内存中保存的当前活动信息。该函数在调用时需要指定线程ID，线程ID的选取请参考**PG_STAT_ACTIVITY**中的pid，线程ID为0时表示选取当前线程ID，例如：

```
SELECT pg_stat_get_mem_mbytes_reserved(0);
```

表 17-252 PG_STAT_GET_MEM_MBYTES_RESERVED 信息

名称	描述
ConnectInfo	连接信息。
ParctlManager	并发管理信息。
GeneralParams	基本参数信息。
GeneralParams RPDATA	基本资源池信息。

 版权所有 © 华为云计算技术有限公司

名称	描述
ExceptionManager	异常管理信息。
CollectInfo	收集信息。
GeneralInfo	基本信息。
ParctlState	并发状态信息。
CPU INFO	CPU信息。
ControlGroup	控制组信息。
IOSTATE	IO状态信息。

17.3.157 PG_STAT_USER_FUNCTIONS

PG_STAT_USER_FUNCTIONS视图显示命名空间中用户自定义函数（函数语言为非内部语言）的状态信息。

表 17-253 PG_STAT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID。
schemaname	name	模式名。
funcname	name	函数名。
calls	bigint	函数被调用的次数。
total_time	double precision	函数的总执行时长。
self_time	double precision	当前线程调用函数的总的时长。

17.3.158 PG_STAT_USER_INDEXES

PG_STAT_USER_INDEXES视图显示数据库中用户自定义普通表和TOAST表的索引状态信息。

表 17-254 PG_STAT_USER_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
indexrelname	name	索引名。
idx_scan	bigint	通过索引扫描的次数。
idx_tup_read	bigint	通过索引上扫描返回的索引条目数量。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。

17.3.159 PG_STAT_USER_TABLES

PG_STAT_USER_TABLES视图显示所有命名空间中用户自定义普通表和TOAST表的状态信息。

表 17-255 PG_STAT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描的次数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum的次数。
analyze_count	bigint	analyze的次数。
autoanalyze_count	bigint	autoanalyze的次数。

17.3.160 PG_STAT_REPLICATION

PG_STAT_REPLICATION视图用于描述日志同步状态信息，如发起端发送日志位置，收端接收日志位置等。

表 17-256 PG_STAT_REPLICATION 字段

名称	类型	描述
pid	bigint	线程的PID。
usesysid	oid	用户系统ID。
usename	name	用户名。
application_name	text	程序名称。
client_addr	inet	客户端地址。
client_hostname	text	客户端名。
client_port	integer	客户端端口号。
backend_start	timestamp with time zone	程序启动时间。
state	text	日志复制的状态（追赶状态，还是一致的流状态）。
sender_sent_location	text	发送端发送日志位置。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
receiver_write_location	text	接收端write日志位置。
receiver_flush_location	text	接收端flush日志位置。
receiver_replay_location	text	接收端replay日志位置。
sync_priority	integer	同步复制的优先级（0表示异步）。
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步）。

17.3.161 PG_STAT_SYS_INDEXES

PG_STAT_SYS_INDEXES视图显示pg_catalog、information_schema模式中所有系统表的索引状态信息。

表 17-257 PG_STAT_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_scan	bigint	通过索引扫描的次数。
idx_tup_read	bigint	通过索引上扫描返回的索引条目数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。

17.3.162 PG_STAT_SYS_TABLES

PG_STAT_SYS_TABLES视图显示pg_catalog、information_schema模式的所有命名空间中系统表的统计信息。

表 17-258 PG_STAT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描的次数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次analyze时间。
last_autoanalyze	timestamp with time zone	最后一次autoanalyze时间。
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum的次数。
analyze_count	bigint	analyze的次数。
autoanalyze_count	bigint	autoanalyze的次数。

 版权所有 © 华为云计算技术有限公司

17.3.163 PG_STAT_XACT_ALL_TABLES

PG_STAT_XACT_ALL_TABLES视图显示命名空间中所有普通表和TOAST表的事务状态信息。

表 17-259 PG_STAT_XACT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在此表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

17.3.164 PG_STAT_XACT_SYS_TABLES

PG_STAT_XACT_SYS_TABLES视图显示命名空间中系统表的事务状态信息。

表 17-260 PG_STAT_XACT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在此表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。
n_tup_ins	bigint	插入行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
n_tup_upd	bigint	更新行数。
n_tup_del	bigint	删除行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

17.3.165 PG_STAT_XACT_USER_FUNCTIONS

PG_STAT_XACT_USER_FUNCTIONS视图显示关于函数执行的统计信息。

表 17-261 PG_STAT_XACT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID。
schemaname	name	模式名。
funcname	name	函数名。
calls	bigint	函数被调用的次数。
total_time	double precision	函数的总执行时长。
self_time	double precision	当前线程调用函数的总的时长。

17.3.166 PG_STAT_XACT_USER_TABLES

PG_STAT_XACT_USER_TABLES视图显示命名空间中用户表的事务状态信息。

表 17-262 PG_STAT_XACT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在该表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的活跃行数。
idx_scan	bigint	在该表上启动的索引扫描数。
idx_tup_fetch	bigint	索引扫描抓取的活跃行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
n_tup_ins	bigint	插入行数。
n_tup_upd	bigint	更新行数。
n_tup_del	bigint	删除行数。
n_tup_hot_upd	bigint	热更新行数（即不需要单独的索引更新）。

17.3.167 PG_STATIO_ALL_INDEXES

PG_STATIO_ALL_INDEXES视图显示当前数据库所有索引的I/O的统计信息。

表 17-263 PG_STATIO_ALL_INDEXES 字段

名称	类型	描述
relid	oid	索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引中模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引缓冲区命中数量。

17.3.168 PG_STATIO_ALL_SEQUENCES

PG_STATIO_ALL_SEQUENCES视图显示当前数据库中相关的序列信息以及指定序列的I/O的统计信息。

表 17-264 PG_STATIO_ALL_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列缓冲区命中数量。

版权所有 © 华为云计算技术有限公司

17.3.169 PG_STATIO_ALL_TABLES

PG_STATIO_ALL_TABLES视图显示当前数据库中所有表的I/O的统计信息（包括TOAST表）。

表 17-265 PG_STATIO_ALL_TABLES 字段

名称	类型	描述
relid	oid	表OID。
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从此表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引命中缓冲区数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

17.3.170 PG_STATIO_SYS_INDEXES

PG_STATIO_SYS_INDEXES视图显示命名空间中所有系统表索引的IO状态信息。

表 17-266 PG_STATIO_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID。
indexrelid	oid	索引的OID。
schemaname	name	索引的模式名。
relname	name	索引的表名。
indexrelname	name	索引名。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引缓冲区命中数量。

版权所有 © 华为云计算技术有限公司

17.3.171 PG_STATIO_SYS_SEQUENCES

PG_STATIO_SYS_SEQUENCES视图显示命名空间中所有系统表为序列的IO状态信息。

表 17-267 PG_STATIO_SYS_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列缓冲区命中数量。

17.3.172 PG_STATIO_SYS_TABLES

PG_STATIO_SYS_TABLES视图显示命名空间中所有系统表的IO状态信息。

表 17-268 PG_STATIO_SYS_TABLES 字段

名称	类型	描述
relid	oid	表OID。
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引命中缓冲区数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

版权所有 © 华为云计算技术有限公司

17.3.173 PG_STATIO_USER_INDEXES

PG_STATIO_USER_INDEXES视图显示命名空间中所有用户关系表索引的IO状态信息。

表 17-269 PG_STATIO_USER_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID。
indexrelid	oid	该索引的OID。
schemaname	name	该索引的模式名。
relname	name	该索引的表名。
indexrelname	name	索引名称。
idx_blks_read	bigint	从索引中读取的磁盘块数。
idx_blks_hit	bigint	索引命中缓冲区数。

17.3.174 PG_STATIO_USER_SEQUENCES

PG_STATIO_USER_SEQUENCES视图显示命名空间中所有用户关系表类型为序列的IO状态信息。

表 17-270 PG_STATIO_USER_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID。
schemaname	name	序列中模式名。
relname	name	序列名。
blks_read	bigint	从序列中读取的磁盘块数。
blks_hit	bigint	序列中缓存命中数。

17.3.175 PG_STATIO_USER_TABLES

PG_STATIO_USER_TABLES视图显示命名空间中所有用户关系表的IO状态信息。

表 17-271 PG_STATIO_USER_TABLES 字段

名称	类型	描述
relid	oid	表OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	表的模式名。
relname	name	表名。
heap_blks_read	bigint	从该表中读取的磁盘块数。
heap_blks_hit	bigint	此表缓冲区命中数。
idx_blks_read	bigint	从表中所有索引读取的磁盘块数。
idx_blks_hit	bigint	表中所有索引缓冲区命中数。
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）。
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）。
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）。
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）。

17.3.176 PG_THREAD_WAIT_STATUS

PG_THREAD_WAIT_STATUS视图检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。

表 17-272 PG_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见**表 17-273**。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
wait_event	text	• 当wait_status为acquire lock、acquire lwlock或wait io时，此列显示具体的锁、轻量级锁或IO等待信息。 • 当线程无其他等待事件且已开启实时算子运行状态监控时，此列显示当前算子在执行计划中的层级及已处理行数。 • 其他情况下，此列为空。

wait_status列的等待状态有以下状态。

表 17-273 等待状态列表

wait_status值	含义
none	未等任意事件。
acquire lock	等待加锁，要么加锁成功，要么加锁等待超时。
acquire lwlock	等待获取轻量级锁。
wait io	等待IO完成。
wait cmd	等待完成读取网络通信包。
wait pooler get conn	等待pooler完成获取连接。
wait pooler abort conn	等待pooler完成终止连接。
wait pooler clean conn	等待pooler完成清理连接。
pooler create conn: [nodename], total N	等待pooler建立连接，当前正在与nodename指定节点建立连接，且仍有N个连接等待建立。
get conn	获取到其他节点的连接。
set cmd: [nodename]	在连接上执行SET/RESET/TRANSACTION BLOCK LEVEL PARA SET/SESSION LEVEL PARA SET，当前正在nodename指定节点上执行。
cancel query	取消某连接上正在执行的SQL语句。
stop query	停止某连接上正在执行的查询。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
wait node: [nodename](plevel), total N, [phase]	等待接收与某节点的连接上的数据，当前正在等待nodename节点plevel线程的数据，且仍有N个连接的数据待返回。如果状态包含phase信息，则可能的阶段状态有： ● begin：表示处于事务开始阶段。 ● commit：表示处于事务提交阶段。 ● rollback：表示处于事务回滚阶段。
wait transaction sync: xid	等待xid指定事务同步。
wait wal sync	等待特定LSN的wal log完成到备机的同步。
wait data sync	等待完成数据页到备机的同步。
wait data sync queue	等待把行存的数据页或列存的CU放入同步队列。
flush data: [nodename](plevel), [phase]	等待向网络中nodename指定节点的plevel对应线程发送数据。如果状态包含phase信息，则可能的阶段状态为wait quota，即当前通信流正在等待quota值。
stream get conn: [nodename], total N	初始化stream flow时，等待与nodename节点的consumer对象建立连接，且当前有N个待建连对象。
wait producer ready: [nodename](plevel), total N	初始化stream flow时，等待每个producer都准备好，当前正在等待nodename节点plevel对应线程的producer对象准备好，且仍有N个producer对象处于等待状态。
synchronize quit	stream plan结束时，等待stream线程组内的线程统一退出。
nodegroup destroy	stream plan结束时，等待销毁stream node group。
wait active statement	等待作业执行，正在资源负载管控中。
wait global queue	等待作业执行，正在全局队列排队。
wait respool queue	等待作业执行，正在资源池上排队。
wait ccn queue	等待作业执行，正在中心协调节点(CCN)中排队。
gtm connect	等待与GTM建连。
gtm get gxid	等待从GTM获取事务xid。
gtm get snapshot	等待从GTM获取事务快照snapshot。
gtm begin trans	等待GTM开始事务。
gtm commit trans	等待GTM提交事务。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
gtm rollback trans	等待GTM执行事务回滚。
gtm create sequence	等待GTM创建sequence。
gtm alter sequence	等待GTM修改sequence。
gtm get sequence val	等待从GTM获取sequence的下一个值。
gtm set sequence val	等待GTM设置sequence的值。
gtm drop sequence	等待GTM删除sequence。
gtm rename sequence	等待GTM重命名sequence。
analyze: [relname], [phase]	当前正在对表relname执行analyze。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的analyze分析操作。
vacuum: [relname], [phase]	当前正在对表relname执行vacuum。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的vacuum清理操作。
vacuum full: [relname]	当前正在对表relname执行vacuum full清理。
create index	当前正在创建索引。
HashJoin - [build hash | write file]	当前是HashJoin算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashJoin算子正在建立哈希表。 ● write file：表示当前HashJoin算子正在将数据写入磁盘。
HashAgg - [build hash | write file]	当前是HashAgg算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashAgg算子正在建立哈希表。 ● write file：表示当前HashAgg算子正在将数据写入磁盘。
HashSetop - [build hash | write file]	当前是HashSetop算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashSetop算子正在建立哈希表。 ● write file：表示当前HashSetop算子正在将数据写入磁盘。
Sort | Sort - write file	当前是Sort算子做排序，write file表示Sort算子正在将数据写入磁盘。

 版权所有 © 华为云计算技术有限公司

wait_status值	含义
Material | Material - write file	当前是Material算子，write file表示Material算子正在将数据写入磁盘。
wait sync consumer next step	Consumer接收端同步等待下一轮迭代。
wait sync producer next step	Producer发送端同步等待下一轮迭代。
wait agent release	正在释放当前agent(8.1.2及以上版本支持)。
wait stream task	stream线程等待被复用(8.1.2及以上版本支持)。

当wait_status为acquire lwlock、acquire lock或者wait io时，表示有等待事件。正在等待获取wait_event列对应类型的轻量级锁、事务锁，或者正在进行IO。

当wait_status不属于上述三种类型时，如果已开启实时算子运行状态监控功能（即参数**enable_operator_realtime_monitor**设置为on，该参数仅9.1.2及以上集群版本支持），wait_event显示当前正在执行的算子、算子在执行计划中的层级以及已处理行数。

实时算子运行状态监控支持HashJoin、Agg、Sort、Material、SetOp、MergeJoin、WindowAgg和各类Scan算子。

其中，wait_status值为acquire lwlock（轻量级锁）时对应的wait_event等待事件类型与描述信息如下。（wait_event为extension时，表示此时的轻量级锁是动态分配的锁，未被监控。）

表 17-274 轻量级锁等待事件列表

wait_event类型	类型描述
ShmemIndexLock	用于保护共享内存中的主索引哈希表。
OidGenLock	用于避免不同线程产生相同的OID。
XidGenLock	用于避免两个事务获得相同的xid。
ProcArrayLock	用于避免并发访问或修改ProcArray共享数组。
SInvalReadLock	用于避免与清理失效消息并发执行。
SInvalWriteLock	用于避免与其他写失效消息、清理失效消息并发执行。
WALInsertLock	用于避免与其他WAL插入操作并发执行。
WALWriteLock	用于避免并发WAL写盘。
ControlFileLock	用于避免pg_control文件的读写并发、写写并发。
CheckpointLock	用于避免多个checkpoint并发执行。
CLogControlLock	用于避免并发访问或者修改Clog控制数据结构。

版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
MultiXactGenLock	用于串行分配唯一MultiXact id。
MultiXactOffsetControlLock	用于避免对pg_multixact/offset的写写并发和读写并发。
MultiXactMemberControlLock	用于避免对pg_multixact/members的写写并发和读写并发。
RelCacheInitLock	用于失效消息场景对init文件进行操作时加锁。
CheckpointerCommLock	用于向checkpointer发起文件刷盘请求场景，需要串行的向请求队列插入请求结构。
TwoPhaseStateLock	用于避免并发访问或者修改两阶段信息共享数组。
TablespaceCreateLock	用于确定tablespace是否已经存在。
BtreeVacuumLock	用于防止vacuum清理B-tree中还在使用的页面。
AutovacuumLock	用于串行化访问autovacuum worker数组。
AutovacuumScheduleLock	用于串行化分配需要vacuum的table。
SyncScanLock	用于确定heap扫描时某个relfilenode的起始位置。
NodeTableLock	用于保护存放CN和DN节点信息的共享结构。
PoolerLock	用于保证两个线程不会同时从连接池里取到相同的连接。
RelationMappingLock	用于等待更新系统表到存储位置之间映射的文件。
AsyncCtlLock	用于避免并发访问或者修改共享通知状态。
AsyncQueueLock	用于避免并发访问或者修改共享通知信息队列。
SerializableXactHashLock	用于避免对于可串行事务共享结构的写写并发和读写并发。
SerializableFinishedListLock	用于避免对于已完成可串行事务共享链表的写写并发和读写并发。
SerializablePredicateLockListLock	用于保护对于可串行事务持有的锁链表。
OldSerXidLock	用于保护记录冲突可串行事务的结构。
FileStatLock	用于保护存储统计文件信息的数据结构。
SyncRepLock	用于在主备复制时保护xlog同步信息。
DataSyncRepLock	用于在主备复制时保护数据页同步信息。
CStoreColspaceCacheLock	用于保护列存表的CU空间分配。
CStoreCUCacheSweepLock	用于列存CU Cache循环淘汰。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
MetaCacheSweepLock	用于元数据循环淘汰。
DfsConnectorCacheLock	用于保护缓存HDFS连接的句柄的全局哈希表。
dummyServerInfoCacheLock	用于保护缓存加速集群连接信息的全局哈希表。
ExtensionConnectorLibLock	用于初始化ODBC连接场景，在加载与卸载特定动态库时进行加锁。
SearchServerLibLock	用于GPU加速场景初始化加载特定动态库时，对读文件操作进行加锁。
DfsUserLoginLock	用于保护HDFS用户信息的全局链表。
DfsSpaceCacheLock	用于控制HDFS表导入时文件ID单调递增。
LsnXlogChkFileLock	用于串行更新特定结构中记录的主备机的xlog flush位置点。
GTMHostInfoLock	用于避免并发访问或者修改GTM主机信息。
ReplicationSlotAllocationLock	用于主备复制时保护主机端的流复制槽的分配。
ReplicationSlotControlLock	用于主备复制时避免并发更新流复制槽状态。
ResourcePoolHashLock	用于避免并发访问或者修改资源池哈希表。
WorkloadStatHashLock	用于避免并发访问或者修改包含CN侧的SQL请求构成的哈希表。
WorkloadIoStatHashLock	用于避免并发访问或者修改用于统计当前DN的IO信息的哈希表。
WorkloadCGroupHashLock	用于避免并发访问或者修改Cgroup信息构成的哈希表。
OBSGetPathLock	用于避免对obs路径的写写并发和读写并发。
WorkloadUserInfoLock	用于避免并发访问或修改负载管理的用户信息哈希表。
WorkloadRecordLock	用于避免并发访问或修改在内存自适应管理时对CN收到请求构成的哈希表。
WorkloadIOUtilLock	用于保护记录iostat，CPU等负载信息的结构。
WorkloadNodeGroupLock	用于避免并发访问或者修改内存中的nodegroup信息构成的哈希表。
JobShmemLock	用于MPP兼容ORACLE定时任务功能中保护定时读取的全局变量。
OBSRuntimeLock	用于获取环境变量，如GAUSSHOME。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
LLVMDumpIRLock	用于导出动态生成函数所对应的汇编语言。
LLVMParseIRLock	用于在查询开始处从IR文件中编译并解析已写好的IR函数。
RPNumberLock	用于加速集群的DN对正在执行计划的任务线程的计数。
ClusterRPLock	用于加速集群的CCN中维护的集群负载数据的并发存取控制。
CriticalCacheBuildLock	用于从共享或者本地缓存初始化文件中加载cache的场景。
WaitCountHashLock	用于保护用户语句计数功能场景中的共享结构。
BufMappingLock	用于保护对共享缓冲映射表的操作。
LockMgrLock	用于保护常规锁结构信息。
PredicateLockMgrLock	用于保护可串行事务锁结构信息。
OperatorRealTLock	用于避免并发访问或者修改记录算子级实时数据的全局结构。
OperatorHistLock	用于避免并发访问或者修改记录算子级历史数据的全局结构。
SessionRealTLock	用于避免并发访问或者修改记录query级实时数据的全局结构。
SessionHistLock	用于避免并发访问或者修改记录query级历史数据的全局结构。
CacheSlotMappingLock	用于保护CU Cache全局信息。
BarrierLock	用于保证当前只有一个线程在创建Barrier。

当wait_status值为wait io时对应的wait_event等待事件类型与描述信息如下。

表 17-275 IO 等待事件列表

wait_event类型	类型描述
BufFileRead	从临时文件中读取数据到指定buffer。
BufFileWrite	向临时文件中写入指定buffer中的内容。
ControlFileRead	读取pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。
ControlFileSync	将pg_control文件持久化到磁盘。数据库初始化时发生。

版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
ControlFileSyncUpdate	将pg_control文件持久化到磁盘。主要在数据库启动、执行checkpoint和主备校验过程中发生。
ControlFileWrite	写入pg_control文件。数据库初始化时发生。
ControlFileWriteUpdate	更新pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。
CopyFileRead	copy文件时读取文件内容。
CopyFileWrite	copy文件时写入文件内容。
DataFileExtend	扩展文件时向文件写入内容。
DataFileFlush	将表数据文件持久化到磁盘。
DataFileImmediateSync	将表数据文件立即持久化到磁盘。
DataFilePrefetch	异步读取表数据文件。
DataFileRead	同步读取表数据文件。
DataFileSync	将表数据文件的修改持久化到磁盘。
DataFileTruncate	表数据文件truncate。
DataFileWrite	向表数据文件写入内容。
LockFileAddToDataDirRead	读取"postmaster.pid"文件。
LockFileAddToDataDirSync	将"postmaster.pid"内容持久化到磁盘。
LockFileAddToDataDirWrite	将pid信息写到"postmaster.pid"文件。
LockFileCreateRead	读取LockFile文件"%s.lock"。
LockFileCreateSync	将LockFile文件"%s.lock"内容持久化到磁盘。
LockFileCreateWRITE	将pid信息写到LockFile文件"%s.lock"。
RelationMapRead	读取系统表到存储位置之间的映射文件。
RelationMapSync	将系统表到存储位置之间的映射文件持久化到磁盘。
RelationMapWrite	写入系统表到存储位置之间的映射文件。
ReplicationSlotRead	读取流复制槽文件。重新启动时发生。
ReplicationSlotRestoreSync	将流复制槽文件持久化到磁盘。重新启动时发生。
ReplicationSlotSync	checkpoint时将流复制槽临时文件持久化到磁盘。
ReplicationSlotWrite	checkpoint时写流复制槽临时文件。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
SLRUFlushSync	将pg_clog、pg_subtrans和pg_multixact文件持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRURead	读取pg_clog、pg_subtrans和pg_multixact文件。
SLRUSync	将脏页写入文件pg_clog、pg_subtrans和pg_multixact并持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRUWrite	写入pg_clog、pg_subtrans和pg_multixact文件。
TimelineHistoryRead	读取timeline history文件。在数据库启动时发生。
TimelineHistorySync	将timeline history文件持久化到磁盘。在数据库启动时发生。
TimelineHistoryWrite	写入timeline history文件。在数据库启动时发生。
TwophaseFileRead	读取pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileSync	将pg_twophase文件持久化到磁盘。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileWrite	写入pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
WALBootstrapSync	将初始化的WAL文件持久化到磁盘。在数据库初始化发生。
WALBootstrapWrite	写入初始化的WAL文件。在数据库初始化发生。
WALCopyRead	读取已存在的WAL文件并进行复制时产生的读操作。在执行归档恢复完后发生。
WALCopySync	将复制的WAL文件持久化到磁盘。在执行归档恢复完后发生。
WALCopyWrite	读取已存在WAL文件并进行复制时产生的写操作。在执行归档恢复完后发生。
WALInitSync	将新初始化的WAL文件持久化磁盘。在日志回收或写日志时发生。
WALInitWrite	将新创建的WAL文件初始化为0。在日志回收或写日志时发生。
WALRead	从xlog日志读取数据。两阶段文件redo相关的操作产生。
WALSyncMethodAssign	将当前打开的所有WAL文件持久化到磁盘。
WALWrite	写入WAL文件。
AuditWrite	审计日志线程写文件。
LogRead	日志线程读文件。

 版权所有 © 华为云计算技术有限公司

wait_event类型	类型描述
LogWrite	日志线程写文件。

当wait_status值为acquire lock（事务锁）时对应的wait_event等待事件类型与描述信息如下。

表 17-276 事务锁等待事件列表

wait_event类型	类型描述
relation	对表加锁。
extend	对表扩展空间时加锁。
partition	对分区表加锁。
partition_seq	对分区表的分区加锁。
page	对表页面加锁。
tuple	对页面上的tuple加锁。
transactionid	对事务ID加锁。
virtualxid	对虚拟事务ID加锁。
object	加对象锁。
cstore_freespace	对列存空闲空间加锁。
userlock	加用户锁。
advisory	加advisory锁。

17.3.177 PG_TABLES

PG_TABLES视图提供了对数据库中每个表访问的有用信息。

表 17-277 PG_TABLES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
tableowner	name	pg_get_userbyid(PG_CLASS.relowner)	表的所有者。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
tablespace	name	PG_TABLESPACE.spcname	包含表的表空间，默认为NULL。
hasindexes	boolean	PG_CLASS.relhasindex	如果表上有索引（或者最近拥有）则为TRUE，否则为FALSE。
hasrules	boolean	PG_CLASS.relhasrules	如果表上有规则，则为TRUE，否则为FALSE。
hastriggers	boolean	PG_CLASS.RELHASTRIGGERS	如果表上有触发器，则为TRUE，否则为FALSE。
tablecreator	name	pg_get_userbyid(PG_OBJECT.creator)	表创建用户，如创建用户已删除，则返回空。
created	timestamp with time zone	PG_OBJECT.ctime	表创建时间。
last_ddl_time	timestamp with time zone	PG_OBJECT.mtime	表最后修改时间。

应用示例

查询指定模式下所有的表。

```
SELECT tablename FROM PG_TABLES WHERE schemaname = 'public';
```

```
gaussdb=> SELECT tablename FROM PG_TABLES WHERE schemaname = 'public';
       tablename
------------------------
 test_block_rule1
 test_block_rule2
 test_block_rule3
 hstore_opt_table_demo
 test_t5
 stu_icase_info
 customer
 store_returns
 customer_address
 web_returns_p1
 user_info
 user_info1
(12 rows)
```


版权所有 © 华为云计算技术有限公司

17.3.178 PG_TDE_INFO

PG_TDE_INFO视图提供了当前集群加密信息。

表 17-278 PG_TDE_INFO 字段

名称	类型	描述
is_encrypt	text	是否加密集群。 ● f：非加密集群。 ● t：加密集群。
g_tde_algo	text	加密算法。 ● AES-CTR-128
remain	text	保留字段。

应用示例

查看当前集群是否加密/查看当前集群的加密算法：

```
SELECT * FROM PG_TDE_INFO;
```

```
gaussdb=> SELECT * FROM PG_TDE_INFO;
 is_encrypt | g_tde_algo  | remain
------------+-------------+--------
 f          | AES-CTR-128 | remain
(1 row)
```

17.3.179 PG_TIMEZONE_ABBREVS

PG_TIMEZONE_ABBREVS视图提供了输入例程能够识别的所有时区缩写。

表 17-279 PG_TIMEZONE_ABBREVS 字段

名称	类型	描述
abbrev	text	时区缩写。
utc_offset	interval	相对于UTC的偏移量。
is_dst	boolean	如果这是一个夏时制时区缩写则为TRUE，否则为FALSE。

17.3.180 PG_TIMEZONE_NAMES

PG_TIMEZONE_NAMES视图提供了所有能够被SET TIMEZONE识别的时区名及其缩写、UTC偏移量等信息。

 版权所有 © 华为云计算技术有限公司

表 17-280 PG_TIMEZONE_NAMES 字段

名称	类型	描述
name	text	时区名。
abbrev	text	时区名缩写。
utc_offset	interval	相对于UTC的偏移量。
is_dst	boolean	如果当前正处于夏令时范围则为TRUE，否则为FALSE。

17.3.181 PG_TOTAL_MEMORY_DETAIL

PG_TOTAL_MEMORY_DETAIL视图显示某个数据库节点内存使用情况。

表 17-281 PG_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存的名称，包括以下几种： ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● max_topsql_memory：TopSQL记录历史作业监控信息允许使用的最大内存。 ● topsql_used_memory：TopSQL记录历史作业监控信息已使用的内存大小。 ● topsql_peak_memory：TopSQL记录历史作业监控信息的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● gpu_max_dynamic_memory：GPU内存最大值。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● gpu_dynamic_peak_memory：GPU内存使用的最大内存。 ● pooler_conn_memory：pooler连接占用内存大小。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
		• pooler_freeconn_memory：pooler空闲连接占用的内存大小。 • storage_compress_memory：列存压缩和解压缩使用的内存大小。 • udf_reserved_memory：为UDF Worker进程预留的内存大小。 • mmap_used_memory：mmap使用的内存大小。
memorymbytes	integer	内存使用的大小，单位MB。

17.3.182 PG_TOTAL_SCHEMA_INFO

PG_TOTAL_SCHEMA_INFO视图显示各个数据库下所有Schema的存储资源使用情况。此视图在参数use_workload_manager为on时才有效。

名称	类型	描述
schemaid	oid	Schema的OID。
schemaname	text	Schema名称。
databaseid	oid	数据库的OID。
databasename	name	数据库名称。
usedspace	bigint	该Schema已使用的永久表存储空间大小，单位Byte。
permspace	bigint	该Schema的永久表存储空间上限值，单位Byte。

17.3.183 PG_TOTAL_USER_RESOURCE_INFO

PG_TOTAL_USER_RESOURCE_INFO视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在参数use_workload_manager为on时才有效。

表 17-282 PG_TOTAL_USER_RESOURCE_INFO 字段

名称	类型	描述
username	name	用户名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_memory	integer	用户使用的内存大小，单位：MB。 ● DN：显示当前DN上用户使用的内存大小。 ● CN：显示所有DN上用户使用内存的累积和。
total_memory	integer	资源池使用的内存大小，单位：MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存max_dynamic_memory。具体的计算公式为： total_memory = max_dynamic_memory * parent_percent * user_percent CN：显示所有DN上用户可用内存上限的累积和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。
total_cpu	integer	在该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑读的字节总数。单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑写的字节总数。单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的次数之和。 DN：实例启动至当前时间为止，该用户逻辑读的次数之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的次数之和。 DN：实例启动至当前时间为止，该用户逻辑写的次数之和。
read_speed	double precision	CN：过去5秒内，该用户在所有DN上逻辑读平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑读平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在所有DN上逻辑写平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑写平均速率。单位KB/s。
send_speed	double precision	CN：过去5秒内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：过去5秒内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络接收平均速率。单位KB/s。
obs_io_read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO读的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO写的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO读平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO读平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO写平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO写平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

17.3.184 PG_USER

PG_USER视图提供了访问数据库用户的信息。

表 17-283 PG_USER 字段

名称	类型	描述
usename	name	用户名。
usesysid	oid	此用户的ID。
usecreatedb	boolean	用户是否可以创建数据库。
usesuper	boolean	用户是否是拥有最高权限的初始系统管理员。
usecatupd	boolean	用户是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
userepl	boolean	用户是否可以复制数据流。
passwd	text	密文存储后的用户口令，始终为********。
valbegin	timestamp with time zone	账户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	账户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	用户所在的资源池。
parent	oid	父用户OID
spacelimit	text	永久表存储空间限额。
tempspacelimit	text	临时表存储空间限额。
spillspacelimit	text	算子落盘空间限额。
useconfig	text[]	运行时配置参数的会话缺省。
nodegroup	name	用户关联的逻辑集群名字，如果该用户没有管理逻辑集群，则该字段为空。

应用示例

查看当前数据库用户列表：

```
SELECT usename  FROM pg_user;
```


版权所有 © 华为云计算技术有限公司

```
gaussdb=> SELECT usename  FROM pg_user;
 usename
---------
 Ruby
 dbadmin
(2 rows)
```

17.3.185 PG_USER_MAPPINGS

PG_USER_MAPPINGS视图提供访问关于用户映射的信息的接口。

这个视图只是一个PG_USER_MAPPING的可读部分的视图化表现，如果用户无权使用它则查询此表时，有些选项字段会显示为空。

表 17-284 PG_USER_MAPPINGS 字段

名字	类型	引用	描述
umid	oid	PG_USER_MAPPING.oid	用户映射的OID。
srvid	oid	PG_FOREIGN_SERVER.oid	包含这个映射的外部服务器的OID。
srvname	name	PG_FOREIGN_SERVER.srvname	外部服务器的名字。
umuser	oid	PG_AUTHID.oid	被映射的本地角色的OID，如果用户映射是公共的则为0。
usename	name	-	被映射的本地用户的名字。
umoptions	text[]	-	如果当前用户是外部服务器的所有者，则为用户映射指定选项， 使用“keyword=value”字符串，否则为null。

17.3.186 PG_VIEWS

PG_VIEWS视图提供访问数据库中每个视图的有用信息。

表 17-285 PG_VIEWS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含视图的模式名。
viewname	name	PG_CLASS.relname	视图的名称。
viewowner	name	PG_AUTHID.rolname	视图的所有者。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
definition	text	-	视图的定义。

应用示例

查询指定模式下的所有视图：

```
SELECT * FROM pg_views WHERE schemaname = 'public';
```

```
gaussdb=> SELECT * FROM pg_views WHERE schemaname = 'public';
 schemaname |         viewname         | viewowner |                    definition
------------+--------------------------+-----------+-------------------------------------------------
 public     | customer_details_view_v1 | dbadmin   | SELECT  * FROM customer WHERE (customer.age < 12);
(1 row)
```

17.3.187 PG_WLM_STATISTICS

PG_WLM_STATISTICS视图显示作业结束后或已被处理异常后的负载管理相关信息。该视图8.1.2版本中已废弃，可使用**PGXC_WLM_SESSION_INFO**视图查看所有CN上执行作业结束后的负载管理记录。

表 17-286 PG_WLM_STATISTICS 字段

名称	类型	描述
statement	text	执行了异常处理的语句。
block_time	bigint	语句执行前的阻塞时间。
elapsed_time	bigint	语句的实际执行时间。
total_cpu_time	bigint	语句执行异常处理时DN上CPU使用的总时间。
qualification_time	bigint	语句检查倾斜率的时间周期。
cpu_skew_percent	integer	语句在执行异常处理时DN上CPU使用的倾斜率。
control_group	text	语句执行异常处理时所使用的Cgroups。
status	text	语句执行异常处理后的状态，包括： ● pending：执行前预备状态。 ● running：执行进行状态。 ● finished：执行正常结束。 ● abort：执行异常终止。
action	text	语句执行的异常处理动作，包括： ● abort：执行终止操作。 ● adjust：执行Cgroups调整操作，目前只有降级操作。 ● finish：正常结束。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
queryid	bigint	语句执行所使用的内部query_id。
threadid	bigint	后端线程ID。

17.3.188 PGXC_BULKLOAD_PROGRESS

PGXC_BULKLOAD_PROGRESS显示导入业务的执行进度，仅支持GDS普通文件导入业务。需要有系统管理员权限才可以访问此视图

表 17-287 PGXC_BULKLOAD_PROGRESS 字段

名称	类型	描述
session_id	bigint	GDS的会话ID。
query_id	bigint	查询ID，对应debug_query_id。
query	text	查询语句。
progress	text	进度百分比。

17.3.189 PGXC_BULKLOAD_INFO

通过CN查看PGXC_BULKLOAD_INFO视图，可获取互联互通、GDS、COPY、\COPY等业务执行结束后的历史统计信息。该视图汇总当前集群上各个节点已经执行结束的导入导出类业务的历史执行情况（包括互联互通集群地址、导入导出业务类型、DN上落盘的最大、最小以及总和的行数与字节数等），从而可以获取导入导出类业务执行的历史信息，辅助进行性能问题排查。

该视图不会记录异常中断的导入导出作业，数据直接从系统表GS_WLM_SESSION_INFO获取，并将loader_status字段解析获取导入导出类业务信息。

需要有系统管理员权限才可以访问此PGXC_BULKLOAD_INFO视图。

表 17-288 PGXC_BULKLOAD_INFO 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。
address	text	互联互通对端集群server的address，非空时即为互联互通业务，源集群会额外获取远端集群端口号。
direction	text	导入导出业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
min_done_lines	json	语句在所有DN上的最小行数。
max_done_lines	json	语句在所有DN上的最大行数。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_done_lines	json	语句在所有DN上的总行数。
min_done_bytes	json	语句在所有DN上的最小字节数。
max_done_bytes	json	语句在所有DN上的最大字节数。
total_done_bytes	json	语句在所有DN上的总字节数。

说明

- 异常中断导入导出类业务不会记录在视图中。
- gds外表与互联互通外表实现机制不同，查询时gds会记录全量，互联互通会记录实际。
- 非全量导入导出外表，带limit时会因limit执行计划的特殊，数据显示为从一个DN上收取，即表现为max值为全部，min值为0。
- 若导入导出表为非分区表时：
  - gds分区表较小的情况下，在一个DN获取完数据后，若其余DN未开始获取数据，将不会在其余DN获取数据，所以在gds from非分区表数据量较小的情况下，min值可能为0，表数据量大时不为0。
  - 互联互通源集群导出非分区表时，所有DN都会被记录，且只会拿取一个DN的数据，所以min值为0。
  - 互联互通远端集群导出复制表时，只有一个DN都会被记录，所以相当于只有一个DN，min值max值相同。
- 导入导出历史监控通过复用历史TopSQL功能实现，同样遵守TopSQL的注意事项、前提条件、操作步骤等，详情请见**历史TopSQL**。
- 因TopSQL记录数据量较大，为提高查询性能建议结合start_time、finish_time等字段按需查询使用，或者降低查询频率。

应用示例

使用PGXC_BULKLOAD_INFO视图查询互联互通导入业务：

```
SELECT * FROM PGXC_BULKLOAD_INFO;
 datid | dbname | schemaname | nodename | username | application_name | client_addr |
client_hostname | client_port | query_band | block_time |          start_time          |          finish_time         |
statu
s |     queryid
|
query
                                                                                                                |             session_id             |     address
|   direction   | min_done_lines | max_done_lines | total_done_lines | min_done_by
tes | max_done_bytes | total_done_bytes
-------+----------+---------------+--------------+----------+------------------+-------------+-----------------
+-------------+-----------+------------+-----------------------------+-----------------------------+-------
---+------------------
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
+---------------------------------------+------------------+--------------+---------------+---------------
+-----------------+------------
----+---------------+------------------
```

 版权所有 © 华为云计算技术有限公司

```
 16134 | postgres | "$user",public | coordinator1 | interconn_user | gsql             |             |                |
-1 |             |           0 | 2023-09-25 10:27:47.184696+08 | 2023-09-25 10:27:48.709665+08 | finish
ed | 72339069014639035 | INSERT INTO interconn_user.lineitem_dest SELECT * FROM
interconn_user.ft_lineitem_local;

                                                                                | 1695608841.140482657154648.coordinator1 |
10.90.45.56:63755 | gds from pipe | 19479           | 20971           | 60175            | 3251258
    | 3500876         | 10038234
 16134 | postgres | "$user",public | coordinator1 | interconn_user | interconnection  | 10.90.45.56 |
|       47668 |             |           0 | 2023-09-25 10:27:47.256095+08 | 2023-09-25 10:27:48.582366+08 | finish
ed | 72339069014639046 | INSERT INTO pg_temp.ft_lineitem_local_72339069014639035_wo SELECT
l_orderkey, l_partkey, l_suppkey, l_linenumber, l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag,
l_linestatus, l_shipdate, l_c
ommitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment FROM public.lineitem; |
1695608867.140482657156768.coordinator1 | 10.90.45.56       | gds to pipe   | 19476           | 20934          |
60175            | 3249308
    | 3489789         | 10038234
(2 rows)
```

17.3.190 PGXC_BULKLOAD_STATISTICS

通过CN查看PGXC_BULKLOAD_STATISTICS视图，可获取GDS、COPY、\COPY等业务执行过程中的实时统计信息。该视图汇总当前集群上各个节点正在执行的导入/导出类业务的实时执行情况，从而可以监控导入导出类业务的实时进度，辅助进行性能问题排查。

PGXC_BULKLOAD_STATISTICS视图与PG_BULKLOAD_STATISTICS视图列定义完全相同。这是由于PGXC_BULKLOAD_STATISTICS视图本质是到集群中各个节点上查询PG_BULKLOAD_STATISTICS视图汇总的结果。

需要有系统管理员权限才可以访问此视图。

表 17-289 PGXC_BULKLOAD_STATISTICS 字段

名称	类型	描述
node_name	text	节点名称。
db_name	text	数据库名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
session_id	bigint	GDS的会话ID。
direction	text	业务类型，取值包括：gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。
query	text	查询语句。
address	text	当前导入导出外表的location。
query_start	timestamp with time zone	导入/导出开始时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_bytes	bigint	待处理数据的总大小。 仅GDS普通文件导入时，且该行记录来自CN节点才会显示，否则为空。
phase	text	当前阶段，包括：INITIALIZING、TRANSFER_DATA、RELEASE_RESOURCE。
done_lines	bigint	已传输行数。
done_bytes	bigint	已传输字节数。

17.3.191 PGXC_COLUMN_TABLE_IO_STAT

PGXC_COLUMN_TABLE_IO_STAT视图提供集群所有CN和DN节点上当前数据库所有列存表的IO统计数据。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与GS_COLUMN_TABLE_IO_STAT视图相同，具体的字段请参考**表17-290**。

表 17-290 GS_COLUMN_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
cu_read	bigint	Compression Unit逻辑读个数。
cu_hit	bigint	Compression Unit命中个数。
cidx_read	bigint	Compression Unit Index逻辑读个数。
cidx_hit	bigint	Compression Unit Index命中个数。

17.3.192 PGXC_COMM_CLIENT_INFO

PGXC_COMM_CLIENT_INFO视图存储所有节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

版权所有 © 华为云计算技术有限公司

表 17-291 PGXC_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
query_id	bigint	查询ID，对应debug_query_id。
socket	integer	如果是物理连接，展示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，展示sid，显示-1时表示当前连接是物理连接。

17.3.193 PGXC_COMM_DELAY

PGXC_COMM_DELAY视图展示所有DN的通信库时延状态。

表 17-292 PGXC_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称。
remote_name	text	连接对端节点的对端时延最大的节点名称。
remote_host	text	连接对端IP的对端地址。
stream_num	integer	当前物理连接使用的stream逻辑连接数量。
min_delay	integer	当前物理连接探测到的最小时延，单位微秒。
average	integer	当前物理连接探测时延的平均值，单位微秒。
max_delay	integer	当前物理连接探测到的最大时延，单位微秒。 **说明** 取值为-1，表示时延探测超时失败，请重新建立节点间连接后再执行查询。

17.3.194 PGXC_COMM_RECV_STREAM

PGXC_COMM_RECV_STREAM视图展示所有DN上的通信库接收流状态。

 版权所有 © 华为云计算技术有限公司

表 17-293 PGXC_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接接收报文正常。 ● HOLD：表示逻辑连接接收报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
recv_bytes	bigint	通信流接收的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均接收速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte。

17.3.195 PGXC_COMM_SEND_STREAM

PGXC_COMM_SEND_STREAM视图展示所有DN上的通信库发送流状态。

 版权所有 © 华为云计算技术有限公司

表 17-294 PGXC_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态： ● UNKNOWN：表示当前逻辑连接状态未知。 ● READY：表示逻辑连接已就绪。 ● RUN：表示逻辑连接发送报文正常。 ● HOLD：表示逻辑连接发送报文等待中。 ● CLOSED：表示关闭逻辑连接。 ● TO_CLOSED：表示将会关闭逻辑连接。 ● WRITING：表示正在写入数据。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

17.3.196 PGXC_COMM_STATUS

PGXC_COMM_STATUS视图展示所有DN的通信库状态。

 版权所有 © 华为云计算技术有限公司

表 17-295 PGXC_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck	bigint	节点通信库接收包总量。
txpck	bigint	节点通信库发送包总量。
rxkB	bigint	节点通信库接收字节总量，单位KByte。
txkB	bigint	节点通信库发送字节总量，单位KByte。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位KByte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位KByte。
%USED(PM)	integer	postmaster线程实时使用率。
%USED (send)	integer	通信代理发送线程实时使用率。
%USED (recv)	integer	通信代理接收线程实时使用率。
%USED (CQ)	integer	RDMA通信完成事件线程实时使用率。
logic_conn	bigint	节点创建的DN间通信逻辑连接总数。
create_process	bigint	进程创建的子进程总数。该字段仅9.1.1.100及以上集群版本支持。
create_thread	bigint	进程创建的子线程总数。该字段仅9.1.1.100及以上集群版本支持。
create_conn	bigint	进程创建的连接总数，包含集群内、集群外连接。该字段仅9.1.1.100及以上集群版本支持。
accept_conn	bigint	进程接收的连接总数，包含集群内、集群外连接。该字段仅9.1.1.100及以上集群版本支持。

17.3.197 PGXC_COMM_QUERY_SPEED

PGXC_COMM_QUERY_SPEED视图展示所有节点上所有query对应的流量信息。

表 17-296 PGXC_COMM_QUERY_SPEED 字段

名称	类型	描述
node_name	text	节点名称。
query_id	bigint	通信流对应的debug_query_id编号。
rxkB/s	bigint	查询对应的通信流接收速率，单位Byte/s。
txkB/s	bigint	查询对应的通信流发送速率，单位Byte/s。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
rxkB	bigint	查询对应的通信流接收数据总量，单位Byte。
txkB	bigint	查询对应的通信流发送数据总量，单位Byte。
rxpck/s	bigint	查询对应的通信包接收速率，单位packages/s。
txpck/s	bigint	查询对应的通信包发送速率，单位packages/s。
rxpck	bigint	查询对应的通信包接收总量，单位packages。
txpck	bigint	查询对应的通信包发送总量，单位packages。

17.3.198 PGXC_DEADLOCK

PGXC_DEADLOCK视图获取导致分布式死锁产生的锁等待信息。

目前，PGXC_DEADLOCK视图只收集locktype为relation、partition、page、tuple和transactionid的锁等待信息。

表 17-297 PGXC_DEADLOCK 字段

名称	类型	描述
locktype	text	被锁定对象的类型。
nodename	name	被锁定对象的节点名称。
dbname	name	被锁定对象的数据库名称。如果被锁定对象是事务，则为NULL。
nspname	name	被锁定对象的命名空间名称。
relname	name	被锁定对象对应的关系名称。如果被锁定对象既不是关系，也不是关系的一部分，则为NULL。
partname	name	被锁定对象对应的分区名称。如果被锁定对象不是分区，则为NULL。
page	integer	被锁定对象对应的页面编号。如果被锁定对象既不是页面，也不是元组，则为NULL。
tuple	smallint	被锁定对象对应的元组编号。如果被锁定对象不是元组，则为NULL。
transactionid	xid	被锁定对象对应的事务ID。如果被锁定对象不是事务，则为NULL。
waitusername	name	等待锁的用户名称。
waitgxid	xid	等待锁的事务ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
waitxactstart	timestamp with time zone	等待锁的事务的开始时间。
waitqueryid	bigint	等待锁的线程的最新查询ID。
waitquery	text	等待锁的线程的最新查询语句。
waitpid	bigint	等待锁的线程ID。
waitmode	text	等待的锁的级别。
holdusername	name	持有锁的用户名称。
holdgxid	xid	持有锁的事务ID。
holdxactstart	timestamp with time zone	持有锁的事务的开始时间。
holdqueryid	bigint	持有锁的线程的最新查询ID。
holdquery	text	持有锁的线程的最新查询语句。
holdpid	bigint	持有锁的线程ID。
holdmode	text	持有锁的级别。
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始持有锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

17.3.199 PGXC_DISK_CACHE_STATS

PGXC_DISK_CACHE_STATS视图记录了文件缓存的使用情况。该系统视图仅9.1.0及以上集群版本支持。

表 17-298 PGXC_DISK_CACHE_STATS 字段

名称	类型	描述
node_name	text	节点名称。
total_read	bigint	访问disk cache的总次数。
local_read	bigint	disk cache读本地磁盘的总次数。
remote_read	bigint	disk cache读远端存储的总次数。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
hit_rate	numeric(5,2)	disk cache的命中率。
cache_size	bigint	disk cache保存的数据总大小，单位kbytes。
fill_rate	numeric(5,2)	disk cache的填充率。

示例

查询每个节点disk cache的命中率：

```
SELECT hit_rate FROM pgxc_disk_cache_stats;
```

```
gaussdb=> SELECT hit_rate FROM pgxc_disk_cache_stats;
 hit_rate
----------
   100.00
   100.00
   100.00
(3 rows)
```

17.3.200 PGXC_DISK_CACHE_ALL_STATS

PGXC_DISK_CACHE_ALL_STATS视图记录文件缓存所有的使用情况。该系统视图仅9.1.0及以上集群版本支持。

表 17-299 PGXC_DISK_CACHE_ALL_STATS 字段

名称	类型	描述
node_name	text	节点名称。
total_read	bigint	访问disk cache的总次数。
local_read	bigint	disk cache访问本地磁盘的总次数。
remote_read	bigint	disk cache访问远端存储的总次数。
hit_rate	numeric(5,2)	disk cache的命中率。
cache_size	bigint	disk cache保存的数据总大小，单位kbytes。
fill_rate	numeric(5,2)	disk cache的填充率。
ttl_rate	numeric(5,2)	TTL空间占整个disk cache空间的比率。该字段仅9.1.1.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
temp_file_size	bigint	临时/冷缓存文件的总大小(kbytes)。
a1in_size	bigint	disk cache中a1in队列保存的数据的总大小，单位kbytes。
a1out_size	bigint	disk cache中a1out队列保存的数据的总大小，单位kbytes。
am_size	bigint	disk cache中am队列保存的数据的总大小，单位kbytes。
ttl_size	bigint	disk cache中ttl队列保存的数据的总大小，单位kbytes。该字段仅9.1.1.100及以上集群版本支持。
a1in_fill_rate	numeric(5,2)	disk cache中a1in队列的填充率。
a1out_fill_rate	numeric(5,2)	disk cache中a1out队列的填充率。
am_fill_rate	numeric(5,2)	disk cache中am队列的填充率。
ttl_fill_rate	numeric(5,2)	disk cache中ttl队列的填充率。该字段仅9.1.1.100及以上集群版本支持。
fd	integer	disk cache正在使用的文件描述符数量。
pin_block_count	bigint	disk cache中被pin住block的数量。该字段仅9.1.0.100及以上集群版本支持。

示例

查询disk cache在每个节点使用的文件描述符的数量：

```
SELECT fd FROM pgxc_disk_cache_all_stats;
```

```
gaussdb=> SELECT fd FROM pgxc_disk_cache_all_stats;
 fd
-----
  15
  94
 107
(3 rows)
```

 版权所有 © 华为云计算技术有限公司

17.3.201 PGXC_DISK_CACHE_PATH_INFO

PGXC_DISK_CACHE_PATH_INFO视图记录了文件缓存所在的硬盘的信息。该系统视图仅9.1.0及以上集群版本支持。

表 17-300 PGXC_DISK_CACHE_PATH_INFO 字段

名称	类型	描述
path_name	text	路径名称。
node_name	text	硬盘所属的节点名称。
cache_size	bigint	硬盘中cache文件所占的总大小，单位bytes。
disk_available	bigint	硬盘的可用空间，单位bytes。
disk_size	bigint	硬盘的总容量，单位bytes。
disk_use_ratio	double precision	硬盘空间的使用率。

示例

查询文件缓存所使用的硬盘的信息：

```
SELECT * FROM pgxc_disk_cache_path_info order by 1;
```

```
gaussdb=> SELECT * FROM pgxc_disk_cache_path_info order by 1;
   path_name    |  node_name   | cache_size | disk_available |  disk_size  |  disk_use_ratio
----------------+--------------+------------+----------------+-------------+-------------------
 dn_6001_6002_0 | dn_6001_6002 |   62944542 |    18055708672 | 21464350720 |  .158804805813385
 dn_6001_6002_1 | dn_6001_6002 |   65042701 |    19186720768 | 21464350720 |  .106112222154372
 dn_6003_6004_0 | dn_6003_6004 |   59797347 |    17853214720 | 21464350720 |  .168238771678065
 dn_6003_6004_1 | dn_6003_6004 |   60846408 |    19344195584 | 21464350720 | .0987756472887152
 dn_6005_6006_0 | dn_6005_6006 |   63993623 |    17934393344 | 21464350720 |   .16445675073278
 dn_6005_6006_1 | dn_6005_6006 |   60846417 |    19173769216 | 21464350720 |  .106715620420127
(6 rows)
```

17.3.202 PGXC_GET_STAT_ALL_TABLES

PGXC_GET_STAT_ALL_TABLES视图获取各表的插入、更新、删除以及脏页率信息。

对于高脏页率的系统表，建议在确认当前没有用户操作该系统表时，再执行VACUUM FULL。

建议对脏页率超过80%的非系统表执行VACUUM FULL，用户也可根据业务场景自行选择是否执行VACUUM FULL。

说明

8.2.0及以上集群版本，查询脏页率推荐使用**PGXC_STAT_TABLE_DIRTY**。

版权所有 © 华为云计算技术有限公司

表 17-301 PGXC_GET_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
relname	name	表名。
schemaname	name	表的模式名。
n_tup_ins	numeric	插入的元组条数。
n_tup_upd	numeric	更新的元组条数。
n_tup_del	numeric	删除的元组条数。
n_live_tup	numeric	live元组的条数。
n_dead_tup	numeric	dead元组的条数。
dirty_page_rate	numeric(5,2)	表的脏页率信息(%)。

应用示例

使用PGXC_GET_STAT_ALL_TABLES视图查询数据库内脏页率大于30%的表：

```
SELECT * FROM PGXC_GET_STAT_ALL_TABLES WHERE dirty_page_rate>30;
```

```
gaussdb=> SELECT * FROM PGXC_GET_STAT_ALL_TABLES WHERE dirty_page_rate>30;
 relid |    relname    | schemaname | n_tup_ins | n_tup_upd | n_tup_del | n_live_tup | n_dead_tup | dirty_page_rate
-------+---------------+------------+-----------+-----------+-----------+------------+------------+-----------------
  1255 | pg_proc       | pg_catalog |      1728 |         0 |      1728 |         18 |       1728 |           98.97
  9001 | pgxc_class    | pg_catalog |       276 |         0 |       216 |         60 |        210 |           77.78
  2836 | pg_toast_1255 | pg_toast   |       384 |         0 |       384 |         36 |         76 |           67.86
  5721 | pg_blocklists | pg_catalog |         6 |        12 |         0 |          6 |         12 |           66.66
  2615 | pg_namespace  | pg_catalog |        15 |         0 |         6 |          9 |          6 |           40.00
  2608 | pg_depend     | pg_catalog |     11304 |         0 |      9288 |      15624 |       9288 |           37.28
(6 rows)
```

17.3.203 PGXC_GET_ALL_TABLES_DIRTY_RATIO

PGXC_GET_ALL_TABLES_DIRTY_RATIO视图获取当前数据库下所有HStore_opt表的cudesc表、delta表以及CU文件的脏页率以及大小信息。该视图仅9.1.1.200及以上集群版本支持。

表 17-302 PGXC_GET_ALL_TABLES_DIRTY_RATIO 字段

名称	类型	描述
database_name	name	表所在数据库的名称。
schema_name	name	表的模式名。
group_name	name	表所在逻辑集群名称。
rel_name	name	表名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
partition_name	name	表的分区名称。
cudesc_min_size	bigint	各个DN上最小的cudesc表大小。
cudesc_max_size	bigint	各个DN上最大的cudesc表大小。
cudesc_avg_size	numeric	各个DN上cudesc表大小的平均值。
cudesc_max_dirty_ratio	bigint	各个DN上cudesc表脏页率的最大值。
delta_min_size	bigint	各个DN上最小的delta表大小。
delta_max_size	bigint	各个DN上最大的delta表大小。
delta_avg_size	numeric	各个DN上delta表大小的平均值。
delta_max_dirty_ratio	bigint	各个DN上delta表脏页率的最大值。
cu_min_size	bigint	各个DN上最小的表cu文件大小。
cu_max_size	bigint	各个DN上最大的表cu文件大小。
cu_avg_size	numeric	各个DN上表cu文件的平均大小。
cu_max_dirty_ratio	bigint	各个DN上表cu文件脏页率的最大值。

应用示例

使用PGXC_GET_ALL_TABLES_DIRTY_RATIO视图查询当前数据库下所有Hstore_opt表的cudesc、delta表的大小与脏页情况：

```
SELECT * FROM PGXC_GET_ALL_TABLES_DIRTY_RATIO;
```

```
gaussdb=> SELECT * FROM PGXC_GET_ALL_TABLES_DIRTY_RATIO;
 database_name | schema_name | group_name |       rel_name       | partition_name | cudesc_min_size | cudesc_max_size | cudesc_avg_size | cudesc_max_dirty_ratio | delta_min_size | delta_max_size | delta_avg_size | delta_max_dirty_ratio | cu_min_size | c
u_max_size | cu_avg_size | cu_max_dirty_ratio
-------------+-------------+------------+----------------------+----------------+-----------------+-----------------+-----------------+------------------------+----------------+----------------+----------------+-----------------------+-------------+--
-------------+-------------+--------------------
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_3_5      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      798720 |
    841216 |      814351 |                    3
 gaussdb     | public      | v3_logical | test_t5                |                |           40960 |           40960 |           40960 |                      0 |          16384 |          16384 |          16384 |                     0 |       24576 |
     24576 |       24576 |                    0
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_7      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      774144 |
    818176 |      790187 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_3_0      |           24576 |           24576 |           24576 |                      0 |          16384 |          16384 |          16384 |                     0 |           0 |
         0 |           0 |                    0
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_5      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      784896 |
    845312 |      814592 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_6      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      772096 |
    820224 |      792064 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_3      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      791552 |
    847872 |      816981 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_1      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      764928 |
    817152 |      788333 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_1_4      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      775168 |
    825344 |      794624 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_2      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      795136 |
    846336 |      817493 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_1_2      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      753664 |
    792576 |      763365 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_1_1      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      796160 |
    857600 |      828565 |                    3
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_1_3      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      793600 |
    846336 |      812885 |                    3
 gaussdb     | dbms_om     |            | gs_wlm_session_info_v2 | p_maximum      |           24576 |           24576 |           24576 |                      0 |          16384 |          16384 |          16384 |                     0 |           0 |
         0 |           0 |                    0
 gaussdb     | public      | v3_logical | hstore_opt_table_demo  | p2024_7_4      |           73728 |           73728 |           73728 |                      0 |          16384 |          16384 |          16384 |                     0 |      774656 |
    813056 |      791381 |                    3
(15 rows)
```

17.3.204 PGXC_GET_STAT_ALL_PARTITIONS

PGXC_GET_STAT_ALL_PARTITIONS视图获取各分区表分区的插入、更新、删除以及脏页率信息。

版权所有 © 华为云计算技术有限公司

该视图的统计信息依赖于ANALYZE，为获取最准确的信息请先对分区表进行ANALYZE。

说明

8.2.0及以上集群版本，查询脏页率推荐使用**PGXC_STAT_TABLE_DIRTY**。

表 17-303 PGXC_GET_STAT_ALL_PARTITIONS 字段

名称	类型	描述
relid	oid	表的OID。
partid	oid	分区的OID。
schemaname	name	表的模式名。
relname	name	表名。
partname	name	分区名。
n_tup_ins	bigint	插入的元组条数。
n_tup_upd	bigint	更新的元组条数。
n_tup_del	bigint	删除的元组条数。
n_live_tup	bigint	live元组的条数。
n_dead_tup	bigint	dead元组的条数。
dirty_page_rate	numeric(5,2)	表的脏页率信息(%)。

应用示例

查询数据库内脏页率大于30%的分区表：

```
SELECT * FROM PGXC_GET_STAT_ALL_PARTITIONS WHERE dirty_page_rate>30;
 relid | partid |   schemaname   |      relname       | partname | n_tup_ins | n_tup_upd | n_tup_del | n_live_tup
| n_dead_tup | dirty_page_rate
-------+--------+----------------+--------------------+----------+-----------+-----------+-----------+-----------
+------------+-----------------
 58320 |  58626 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |         0 |         2
|          100.00
 58430 |  58706 | schema_subquery | store_hash_par_mor | p4       |         1 |         1 |         1 |         0 |
2 |          100.00
 58320 |  58644 | schema_subquery | store_hash_par     | p1       |         3 |         0 |         3 |         0 |         3
|          100.00
 58430 |  58770 | schema_subquery | store_hash_par_mor | p4       |         1 |         1 |         1 |         0 |
2 |          100.00
 58320 |  58643 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |         0 |         2
|          100.00
 58320 |  58625 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |         0 |         2
|          100.00
 58320 |  58579 | schema_subquery | store_hash_par     | p1       |         2 |         0 |         2 |         0 |         2
|          100.00
 58320 |  58619 | schema_subquery | store_hash_par     | p1       |         3 |         0 |         3 |         0 |         3
|          100.00
 58320 |  58627 | schema_subquery | store_hash_par     | p1       |         4 |         0 |         4 |         0 |         4
|          100.00
```


版权所有 © 华为云计算技术有限公司

```
 58320 |  58657 | schema_subquery | store_hash_par     | p1      |       3 |       0 |       3 |       0 |       3
|          100.00
(10 rows)
```

17.3.205 PGXC_GET_TABLE_SKEWNESS

PGXC_GET_TABLE_SKEWNESS视图展示当前库中表的数据分布倾斜情况。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-304 PGXC_GET_TABLE_SKEWNESS 字段

名称	类型	描述
schemaname	name	表所在的模式名。
tablename	name	表名。
totalsize	numeric	表的总大小，单位Byte。
avgsize	numeric(1000, 0)	表大小平均值（totalsize/DN个数，该值为平均分布的理想情况下，表在各DN占用空间大小）。
maxratio	numeric(10,3)	单DN表大小最大值占比（表在各DN占用空间的最大值/avgsize）。
minratio	numeric(10,3)	单DN表大小最小值占比（表在各DN占用空间的最小值/avgsize）。
skewsize	bigint	表分布倾斜值（单DN表大小最大值 - 单DN表大小最小值）。
skewratio	numeric(10,3)	表分布倾斜率（skewsize/avgsize）。
skewstddev	numeric(1000, 0)	表分布标准方差（在表大小一定的情况下，该值越大表明表的整体分布情况越倾斜）。

17.3.206 PGXC_GET_TABLESPACE_OBS_FILE

PGXC_GET_TABLESPACE_OBS_FILE视图用于获取当前集群上一个表空间内的PG_DB_FILE系统表上的全部文件信息。该视图仅9.1.1.100及以上集群版本支持。

表 17-305 PGXC_GET_TABLESPACE_OBS_FILE 字段

名称	类型	描述
node_name	name	节点名称。
db_oid	oid	数据库OID。
table_space_oid	oid	表空间OID。
relfilenode	oid	表的物理目录ID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
file_id	bigint	文件ID。
file_type	text	文件类型。
file_size	bigint	文件大小，单位Byte。
file_path	text	文件绝对路径。

17.3.207 PGXC_GLOBAL_TEMP_ATTACHED_PIDS

查看全局临时表在CN上占有资源的会话信息。该视图仅8.2.1.220及以上集群版本支持。

表 17-306 PG_GLOBAL_TEMP_ATTACHED_PIDS 字段

名称	类型	描述
nodename	name	节点名。
schemaname	name	模式名。
tablename	name	表名。
pid	bigint	用于表示会话的PID。

17.3.208 PGXC_GTM_SNAPSHOT_STATUS

PGXC_GTM_SNAPSHOT_STATUS视图用于查看当前GTM上事务信息。

表 17-307 PGXC_GTM_SNAPSHOT_STATUS 字段

名称	类型	描述
xmin	xid	仍在运行的最小事务号。
xmax	xid	已完成的事务号最大的事务的下一个事务号。
csn	integer	待提交事务的序列号。
oldestxmin	xid	当前最早的活跃事务在其取快照时，所有运行事务号最小的事务。
xcnt	integer	当前活跃的事务个数。
running_xids	text	当前活跃的事务号。

版权所有 © 华为云计算技术有限公司

17.3.209 PGXC_INSTANCE_TIME

PGXC_INSTANCE_TIME视图显示集群中各节点上进程的运行时间信息及各执行阶段所消耗时间，除新增node_name（节点名称）字段外，其余字段内容和PV_INSTANCE_TIME视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-308 PGXC_INSTANCE_TIME 字段

名称	类型	描述
node_name	text	节点名称。
stat_id	integer	类型编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

17.3.210 PGXC_LOCKWAIT_DETAIL

PGXC_LOCKWAIT_DETAIL视图显示集群中每个节点中锁等待链详细信息。如果节点中有多级的锁等待关系，会依次将整个锁等待链按照等待顺序显示出来。

该视图仅8.1.3.200及以上集群版本支持。

表 17-309 PGXC_LOCKWAIT_DETAIL 字段

名称	类型	描述
level	integer	锁等待链中的层级，以1开始，每显示一层等待关系level会加1。
node_name	name	节点名称，对应pgxc_node表中的node_name列。
lock_wait_hierarchy	text	锁等待链，以节点名称：进程号->等待进程号->等待进程号->...。
lock_type	text	被锁定对象的类型。
database	oid	被锁定对象所在数据库的OID。
relation	oid	被锁定对象关系的OID。
page	integer	关系内部的页面编号。
tuple	smallint	页面的行编号。
virtual_xid	text	事务的虚拟ID。
transaction_id	xid	事务ID。
class_id	oid	包含该对象的系统表的OID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
obj_id	oid	对象在其系统表内的OID。
obj_subid	smallint	对于表的列字段编号。
virtual_transaction	text	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	持有此锁或者等待此锁的线程号。
mode	text	锁级别。
granted	boolean	是否持有锁。
fastpath	boolean	是否通过fastpath机制获得锁。
wait_for_pid	bigint	锁冲突线程的线程号。
conflict_mode	text	锁冲突线程持有的冲突锁级别。
query_id	bigint	查询语句的id。
query	text	查询语句。
application_name	text	连接到该后端的应用名。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间。
query_start	timestamp with time zone	开始当前活跃查询的时间。
state	text	后端当前总体状态。
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始获取锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

应用实例

步骤1 连接dn节点，开启一个事务，执行查询：

```
begin;select * from t1;
```

 版权所有 © 华为云计算技术有限公司

步骤2 另一个窗口连接cn，对表t1执行truncate：

```
truncate t1;
```

此时truncate会被阻塞。

步骤3 另开一个窗口连接cn，执行select * from pgxc_lockwait_detail;

```
SELECT * FROM PGXC_LOCKWAIT_DETAIL;
level | node_name |             lock_wait_hierarchy              | lock_type | database |  relation  | page | tuple |
virtual_xid | transaction_id | class_id | obj_id | obj_subid | virtual_transaction |       p
id       |        mode         | granted | fastpath |  wait_for_pid   |  conflict_mode  |     query_id
|                  query                   | application_name |         backend_start         |
xact_start           |         query_start          |        state
-------+-----------+----------------------------------------------+-----------+----------+------------+------+-------
+-------------+----------------+----------+--------+-----------+---------------------+--------
---------+---------------------+---------+----------+-----------------+-----------------+------------------
+-----------------------------------------------+-----------------+------------------------------+--
----------------------------+-----------------------------+--------------------
1 | datanode1 | datanode1:140378619314976                    | relation |    16049 | 2147484411 |      |
|            |         673638 |          |        |           | 19/297              | 1403786
19314976 | AccessExclusiveLock | f       | f        | 140378619263840 | AccessShareLock | 73183493945504391
| TRUNCATE t1                    | coordinator1     | 2023-03-13 12:13:52.530602+08 | 2
023-03-13 14:52:16.1456+08    | 2023-03-13 14:52:16.148693+08 | active
2 | datanode1 | datanode1:140378619314976 -> 140378619263840 | relation |    16049 | 2147484411 |
|      |            |                |          |        |           | 23/16067            | 1403786
19263840 | AccessShareLock     | t       | f        |                |                |                0 | begin;select * from t1;
| gsql             | 2023-03-13 14:19:26.325602+08 | 2
023-03-13 14:52:12.042741+08 | 2023-03-13 14:52:12.042741+08 | idle in transaction
(2 rows)
```

----结束

17.3.211 PGXC_INSTR_UNIQUE_SQL

PGXC_INSTR_UNIQUE_SQL视图展示集群中所有CN节点的Unique SQL的完整统计信息。

需要有系统管理员权限才可以访问此视图，该视图的字段与GS_INSTR_UNIQUE_SQL视图字段相同。具体的字段请参考**表17-310**。

表 17-310 GS_INSTR_UNIQUE_SQL 字段

名称	类型	描述
node_name	name	接收SQL的CN节点名称。
node_id	integer	节点ID，等同于pgxc_node表中node_id。
user_name	name	用户名称。
user_id	oid	用户ID。
unique_sql_id	bigint	归一化的UNIQUE SQL ID。
query	text	归一化的SQL文本。
n_calls	bigint	成功执行次数。
min_elapse_time	bigint	SQL在数据库内的最小运行时间（单位：微秒）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_elapse_time	bigint	SQL在数据库内的最大运行时间（单位：微秒）。
total_elapse_time	bigint	SQL在数据库内的总运行时间（单位：微秒）。
n_returned_rows	bigint	行活动-SELECT语句返回的结果集行数。
n_tuples_fetched	bigint	行活动-随机扫描行（列存表/外表不统计）。
n_tuples_returned	bigint	行活动-顺序扫描行（列存表/外表不统计）。
n_tuples_inserted	bigint	行活动-插入行数。
n_tuples_updated	bigint	行活动-更新行数。
n_tuples_deleted	bigint	行活动-删除行数。
n_blocks_fetched	bigint	buffer的块访问次数，即物理读/IO。
n_blocks_hit	bigint	buffer的块命中次数，即逻辑读/Cache。
n_soft_parse	bigint	软解析次数（缓存计划）。
n_hard_parse	bigint	硬解析次数（生成计划）。
db_time	bigint	有效的DB执行时间，包含等待时间、网络发送时间等，若查询执行涉及到多线程，DB_TIME是多个线程的DB_TIME之和（单位：微秒）。
cpu_time	bigint	CPU的执行时间，不包含sleep时间（单位：微秒）。
execution_time	bigint	查询执行器内的SQL执行时间，DDL语句、以及某些不经过执行器执行的语句（例如Copy语句）不计数（单位：微秒）。
parse_time	bigint	SQL解析时间（单位：微秒）。
plan_time	bigint	SQL生成计划时间（单位：微秒）。
rewrite_time	bigint	SQL重写时间（单位：微秒）。
pl_execution_time	bigint	plpgsql过程化语言函数上的执行时间（单位：微秒）。
pl_compilation_time	bigint	plpgsql过程化语言函数上的编译时间（单位：微秒）。
net_send_time	bigint	网络时间，包含CN向客户端发送数据、DN向CN发送数据等时间（单位：微秒）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
data_io_time	bigint	IO时间，文件IO耗时（单位：微秒）。
first_time	timestamp with time zone	该SQL第一次执行的时间。
last_time	timestamp with time zone	该SQL上一次执行的时间。

17.3.212 PGXC_LOCK_CONFLICTS

PGXC_LOCK_CONFLICTS视图提供集群中有冲突的锁的信息。

当某一个锁正在等待另一个锁，或正在被另一个锁等待，即该锁是有冲突的。

目前，PGXC_LOCK_CONFLICTS视图只收集locktype为relation、partition、page、tuple和transactionid的锁的信息。

表 17-311 PGXC_LOCK_CONFLICTS 字段

名称	类型	描述
locktype	text	被锁定对象的类型。
nodename	name	被锁定对象的节点的名称。
dbname	name	被锁定对象的数据库的名称。如果被锁定对象是事务，则为NULL。
nspname	name	被锁定对象的命名空间的名称。
relname	name	被锁定对象对应的关系的名称。如果被锁定对象既不是关系，也不是关系的一部分，则为NULL。
partname	name	被锁定对象对应的分区的名称。如果被锁定对象不是分区，则为NULL。
page	integer	被锁定对象对应的页面的编号。如果被锁定对象既不是页面，也不是元组，则为NULL。
tuple	smallint	被锁定对象对应的元组的编号。如果被锁定对象不是元组，则为NULL。
transactionid	xid	被锁定对象对应的事务的ID。如果被锁定对象不是事务，则为NULL。
username	name	申请锁的用户的名称。
gxid	xid	申请锁的事务的ID。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
xactstart	timestamp with time zone	申请锁的事务的开始时间。
queryid	bigint	申请锁的线程的最新查询ID。
query	text	申请锁的线程的最新查询语句。
pid	bigint	申请锁的线程的ID。
mode	text	锁的级别。
granted	boolean	• 如果锁已被持有，则为TRUE。 • 如果锁还在等待其他锁，则为FALSE。

17.3.213 PGXC_LWLOCKS

PGXC_LWLOCKS视图提供当前集群中所有实例正在持有的或等待的轻量级锁信息。该视图仅9.1.0.200及以上集群版本支持。

表 17-312 PGXC_LWLOCKS 字段

名称	类型	描述
nodename	name	被锁定对象的节点的名称。
pid	bigint	后端线程ID。
query_id	bigint	查询语句的ID。
lwtid	integer	后端线程的轻量级线程号。
reqlockid	integer	当前线程正在请求的轻量级锁ID。
reqlock	text	reqlockid对应的轻量级锁名称。
heldlocknums	integer	当前线程已经获得的轻量级锁的数量。
heldlockid	integer	当前线程已经获得的轻量级锁ID。
heldlock	text	heldlockid对应的轻量级锁名称。
heldlockmode	text	heldlockid对应的轻量级锁模式。

示例

使用PGXC_LWLOCKS视图查询当前集群中所有实例正在持有的或等待的轻量级锁信息：

```
SELECT * FROM pgxc_lwlocks;
```

```
gaussdb=> SELECT * FROM pgxc_lwlocks;
   nodename   |       pid       | query_id | lwtid | reqlockid | reqlock | heldlocknums | heldlockid |   heldlock    | heldlockmode
--------------+-----------------+----------+-------+-----------+---------+--------------+------------+---------------+--------------
 dn_6001_6002 | 281472857484264 |        0 | 80194 |           |         |            1 |          7 | WALWriteLock | Exclusive
(1 row)
```

 版权所有 © 华为云计算技术有限公司

17.3.214 PGXC_MEMORY_DEBUG_INFO

PGXC_MEMORY_DEBUG_INFO视图显示当前集群每个节点在执行作业时的内存报错信息，便于定位内存报错问题，当执行语句报错提示“memory is temporarily unavailable”时，通过该视图可查询到所有节点的内存报错信息，该报错信息跟日志中显示的内存报错信息相同。该视图8.3.0及以上集群版本支持。

须知

该视图仅显示最近一次报错的集群信息，重复报错信息会进行覆盖，同一个query多次申请内存报错，信息不会进行更新。

表 17-313 PGXC_MEMORY_DEBUG_INFO 字段

名称	类型	描述
node_name	text	实例名称，包含CN和DN。
query_id	bigint	正在申请内存的查询ID。
memory_info	text	当前实例的内存使用情况，主要包含： • process_used_memory：DWS进程所使用的内存大小。 • max_dynamic_memory：最大动态内存。 • dynamic_used_memory：已使用的动态内存。 • dynamic_peak_memory：内存的动态峰值。 • dynamic_used_shrctx：最大动态共享内存上下文。 • dynamic_peak_shrctx：共享内存上下文的动态峰值。 • shared_used_memory：已使用的共享内存。 • cstore_used_memory：列存已使用的内存大小。 • comm_used_memory：通信库已使用的内存大小。 • comm_peak_memory：通信库的内存峰值。 • other_used_memory：其他已使用的内存大小。 • topsql_used_memory：topsql已使用内存大小 • large_storage_memory：列存压缩和解压缩使用的内存大小。 • os_totalmem：操作系统总内存大小。 • os_freeemem：操作系统剩余内存大小。
summary	text	包含实例上作业消耗的总估算内存和消耗的总实际内存大小。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
abnormal_query	text	使用内存异常的线程ID和queryid，包含两种情况： 1. 当前使用内存最大的会话。 2. 估算内存和实际使用内存差别最大的会话。
abnormal_memory	text	使用内存异常的内存块，包含共享内存ctx使用最大的和通用内存ctx使用最大的
top_thread	text	使用内存最多的三个线程信息： context name：表示正在使用内存的内存块。 contextlevel：表示ctx的等级。 sessType：表示ctx顶层节点的类型。 totalsize[274,13,260]MB，表示当前内存ctx的总内存，已释放内存和使用内存大小，单位为MB。
create_time	timestamp with time zone	出现内存不足报错的时间点。

17.3.215 PGXC_NODE_ENV

PGXC_NODE_ENV视图提供获取集群中所有节点的环境变量信息。

表 17-314 PGXC_NODE_ENV 字段

名称	类型	描述
node_name	text	集群中所有节点的名称。
host	text	集群中所有节点的主机名称。
process	integer	集群中所有节点的进程号。
port	integer	集群中所有节点的端口号。
installpath	text	集群中所有节点的安装目录。
datapath	text	集群中所有节点的数据目录。
log_directory	text	集群中所有节点的日志目录。

17.3.216 PGXC_NODE_STAT_RESET_TIME

PGXC_NODE_STAT_RESET_TIME视图显示集群中各节点的统计信息重置时间，除新增node_name（节点名称）字段外，其余字段内容和GS_NODE_STAT_RESET_TIME视图相同。需要有系统管理员权限才可以访问此视图。

版权所有 © 华为云计算技术有限公司

表 17-315 PGXC_NODE_STAT_RESET_TIME 字段

名称	类型	描述
node_name	text	节点名称。
reset_time	timestamp	各节点的统计信息重置时间。

17.3.217 PGXC_OBS_IO_SCHEDULER_STATS

PGXC_OBS_IO_SCHEDULER_STATS视图显示OBS IO Scheduler读/写请求相关的近期实时统计信息。该系统视图仅9.1.0及以上版本支持。

表 17-316 PGXC_OBS_IO_SCHEDULER_STATS 字段

名称	类型	描述
node_name	text	节点名称。
io_type	char	IO类型： ● 'r'表示读。 ● 'w'表示写。 ● 's'表示文件操作。
current_bps	int8	当前带宽速率(KB/s)。
best_bps	int8	近期达到过的最佳带宽速率(KB/s)。
waiting_request_num	int	当前排队的请求数。
mean_request_size	int8	近期已处理请求的平均长度(KB)。
total_token_num	int	总的IO令牌数。
available_token_num	int	可用IO令牌数。
total_worker_num	int	总的工作线程数。
idle_worker_num	int	空闲的工作线程数。

示例

步骤1 查询OBS IO Scheduler在每个节点读请求相关的统计信息。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r       |       26990 |    26990 |                   0 |               215 |              18 |               16
|              12 |              10
 dn_6003_6004 | r       |       21475 |    21475 |                  10 |               190 |              30 |               30
```


版权所有 © 华为云计算技术有限公司

```
|            20 |           20
 dn_6005_6006 | r      |      12384 |    12384 |              36 |             133 |           30 |              27
|            20 |           17
```

查询结果显示，这是当前IO Scheduler在进行读取IO操作时的某个时刻统计信息的快照（snapshot），此时带宽处于上升阶段，current_bps与best_bps相等。以dn_6003_6004为例，可以观察到该DN当前队列中存在排队的请求，total_token_num与available_token_num相等，说明查询视图的时刻IO Scheduler还未开始处理这些请求。

步骤2 等待一段时间后，再次发起查询。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r      |      13228 |    26990 |                   0 |               609 |           18 |              18
|            12 |           12
 dn_6003_6004 | r      |      15717 |    21475 |                   0 |               622 |           30 |              30
|            20 |           20
 dn_6005_6006 | r      |      18041 |    21767 |                   0 |               609 |           30 |              30
|            20 |           20
```

此时队列中已经没有了排队的请求，且available_token_num等于total_token_num，说明IO Scheduler已经处理完所有请求，且没有新的请求需要被处理；但观察到current_bps不为零，是因为统计bps的周期为3秒，此时显示的是3秒前的结果。

步骤3 短暂间隔后再次查询结果如下，current_bps会更新为0。

```
SELECT * FROM pgxc_obs_io_scheduler_stats WHERE io_type = 'r' ORDER BY node_name;

  node_name   | io_type | current_bps | best_bps | waiting_request_num | mean_request_size |
total_token_num | available_token_num | total_worker_num | idle_worker_num
--------------+---------+-------------+----------+---------------------+-------------------+-----------------
+---------------------+------------------+-----------------
 dn_6001_6002 | r      |           0 |    26990 |                   0 |               609 |           18 |              18
|            12 |           12
 dn_6003_6004 | r      |           0 |    21475 |                   0 |               622 |           30 |              30
|            20 |           20
 dn_6005_6006 | r      |           0 |    21767 |                   0 |               609 |           30 |              30
|            20 |           20
```

----结束

17.3.218 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS

PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS视图统计了OBS IO Scheduler不同请求类型（包括读/写/文件操作）下的请求数量、流控信息等内容。该系统视图仅9.1.0及以上版本支持。

其中第一次查询结果显示的是自集群启动以来到查询时刻之间的统计内容，详细字段见下表。

表 17-317 PGXC_OBS_IO_SCHEDULER_PERIODIC_STATS 字段

名称	类型	描述
node_name	name	CN或DN实例的名称，例如，dn_6001_6002。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
io_type	char	IO类型，包括： ● R（读） ● W（写） ● S（文件操作）
recent_throttled_req_num	int	两次查询视图之间的限流次数。
total_throttled_req_num	int	总的限流次数。
last_throttled_dur(s)	int8	距离上次发生限流的时间间隔。
waiting_req_num	int	当前有多少排队的请求数。
mean_tps	numeric(7,2)	两次查询视图之间的平均tps，tps为每秒处理的请求数。
mean_req_size(KB)	int8	两次查询视图之间的请求的平均长度，单位KB。
mean_req_latency(ms)	int8	两次查询视图之间的请求的平均时延，单位ms。
max_req_latency(ms)	int8	两次查询视图之间的请求的最大时延，单位ms。
mean_bps(KB/s)	int8	两次查询视图之间的平均读取请求or写入请求的速度，单位KB/s。
duration(s)	int	两次查询视图之间的时间间隔，单位s。

示例

使用语句SELECT * FROM pgxc_obs_io_scheduler_periodic_stats查询视图内容，结果示例：

```
SELECT * FROM pgxc_obs_io_scheduler_periodic_stats;
```

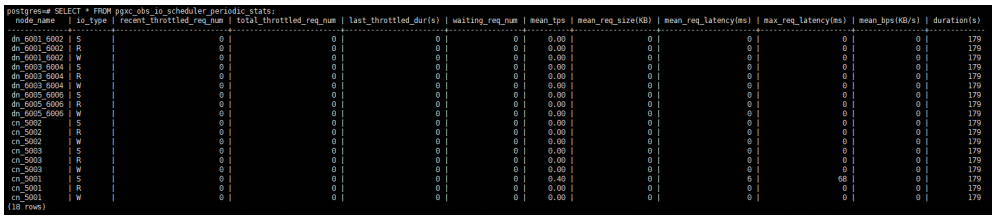
```
postgres=# SELECT * FROM pgxc_obs_io_scheduler_periodic_stats;
  node_name   | io_type | recent_throttled_req_num | total_throttled_req_num | last_throttled_dur(s) | waiting_req_num | mean_tps | mean_req_size(KB) | mean_req_latency(ms) | max_req_latency(ms) | mean_bps(KB/s) | duration(s)
--------------+---------+--------------------------+-------------------------+-----------------------+-----------------+----------+-------------------+----------------------+---------------------+----------------+-------------
dn_6001_6002 | S       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2784
dn_6001_6002 | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2784
dn_6001_6002 | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2784
dn_6003_6004 | S       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
dn_6003_6004 | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
dn_6003_6004 | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
dn_6005_6006 | S       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2785
dn_6005_6006 | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2785
dn_6005_6006 | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2785
cn_5002      | S       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2789
cn_5002      | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2789
cn_5002      | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2789
cn_5003      | S       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
cn_5003      | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
cn_5003      | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2787
cn_5001      | S       |                        0 |                       0 |                     0 |               0 |      .41 |                 5 |                  130 |                   0 |              0 |        2785
cn_5001      | R       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2785
cn_5001      | W       |                        0 |                       0 |                     0 |               0 |     0.00 |                 0 |                    0 |                   0 |              0 |        2785
(18 rows)
```

如果需要显示“mean_tps”数值中小数点前的0，需要在behavior_compat_options参数中设置display_leading_zero（浮点数显示配置项）。

再执行查询视图语句“select * from pgxc_obs_io_scheduler_periodic_stats;” 显示结果如下：

 版权所有 © 华为云计算技术有限公司

```
SELECT * FROM pgxc_obs_io_scheduler_periodic_stats;
```



17.3.219 PGXC_OS_RUN_INFO

PGXC_OS_RUN_INFO视图显示集群中各节点上操作系统运行的状态信息，除新增node_name（节点名称）字段外，其余字段内容和PV_OS_RUN_INFO视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-318 PGXC_OS_RUN_INFO 字段

名称	类型	描述
node_name	text	节点名称。
id	integer	编号。
name	text	操作系统运行状态名称。
value	numeric	操作系统运行状态值。
comments	text	操作系统运行状态注释。
cumulative	boolean	操作系统运行状态的值是否为累加值。

17.3.220 PGXC_OS_THREADS

PGXC_OS_THREADS视图提供当前集群中所有正常节点下的线程状态信息。

表 17-319 PGXC_OS_THREADS 字段

名称	类型	描述
node_name	text	当前集群中所有正常节点的名称。
pid	bigint	当前集群中所有正常节点进程中正在运行的线程号。
lwpid	integer	与pid对应的轻量级线程号。
thread_name	text	与pid对应的线程名称。
creation_time	timestamp with time zone	与pid对应的线程创建的时间。

 版权所有 © 华为云计算技术有限公司

17.3.221 PGXC_POOLER_STATUS

PGXC_POOLER_STATUS视图显示当前集群中各CN节点的pooler中缓存连接状态。该视图只能在CN上执行查询，显示所有CN的pooler模块的连接缓存信息。
PGXC_POOLER_STATUS视图仅8.3.0及以上集群版本支持。

表 17-320 PGXC_POOLER_STATUS 字段

名称	类型	描述
coorname	text	CN节点名称。
database	text	数据库名称。
user_name	text	用户名。
tid	bigint	连接CN的线程ID。
node_oid	bigint	连接的实例节点OID。
node_name	name	连接的实例节点名称。
in_use	boolean	连接是否正被使用。 ● t（true）：表示连接正在使用。 ● f（false）：表示连接没有使用。
fdsock	bigint	对端socket。
remote_pid	bigint	对端线程号。
session_params	text	由此连接下发的GUC会话参数。

17.3.222 PGXC_PREPARED_XACTS

PGXC_PREPARED_XACTS视图显示当前处于prepared阶段的两阶段事务。

表 17-321 PGXC_PREPARED_XACTS 字段

名字	类型	描述
pgxc_prepared_xact	text	查看当前处于prepared阶段的两阶段事务。

17.3.223 PGXC_QUERY_INFO

PGXC_QUERY_INFO视图显示TopSQL经典字段信息。该视图仅9.1.1.100及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

表 17-322 PGXC_QUERY_INFO 字段

名称	类型	描述
dbname	text	连接后端的数据库名称。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。
resource_pool	text	用户使用的资源池。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的PID。
lane	text	语句执行时所在的快慢车道。
unique_sql_id	bigint	归一化的Unique SQL ID。
stmt_type	text	语句对应的查询类型。
except_info	text	语句触发的异常规则信息。
sql_hash	text	归一化的sql hash。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上版本支持。
gxid	xid	语句事务ID号
inserted_rows	bigint	语句插入的行数。
deleted_rows	bigint	语句删除的行数。
updated_rows	bigint	语句更新的行数。
returned_rows	bigint	语句查询的行数。

17.3.224 PGXC_QUERY_SPILL_ACTIVITY

PGXC_QUERY_SPILL_ACTIVITY视图显示各个节点上用户查询及临时文件等内容的详细信息。该视图仅9.1.1.300及以上集群版本支持。

表 17-323 PGXC_QUERY_SPILL_ACTIVITY 字段

名称	类型	描述
coorname	text	当前集群的CN名称。
datname	name	用户会话在后端连接到的数据库名称。
usename	name	登录此后端的用户名。
pid	bigint	后端线程ID。
application_name	text	连接到此后端的应用名。
client_addr	inet	连接到此后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	后端当前总体状态。可能值是： • active：后端正在执行一个查询。 • idle：后端正在等待一个新的客户端命令。 • idle in transaction：后端在事务中，但事务中没有语句在执行。 • idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 • fastpath function call：后端正在执行一个fast-path函数。 • disabled：如果后端禁用track_activities，则报告此状态。 **说明** 只有系统管理员能查看到自己账户所对应的会话状态。其他账户的state信息为空。
resource_pool	name	用户使用的资源池。
query_id	bigint	查询语句的ID。
file_count	bigint	临时文件的数量。
file_size	bigint	临时文件的大小。
nodename	name	节点名称。
modification	timestamp with time zone	文件修改时间。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。

17.3.225 PGXC_RECYCLEBIN_DROP

PGXC_RECYCLEBIN_DROP视图显示回收站里的DROP操作记录，该视图仅9.1.2及以上集群版本支持。

表 17-324 PGXC_RECYCLEBIN_DROP 字段

名称	类型	引用	描述
recyclebin_name	name	PG_FILE_RECYCLEBIN.rbname	回收站中对象的内部名称。
database_name	name	PG_DATABASE.datname	对象所属数据库名称。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
part_name	text	PG_FILE_RECYCLEBIN.subobjname	分区子对象名称。非分区表为NULL。
schema_name	name	PG_NAMESPACE.nspname	原对象所在schema名称。
original_name	name	PG_FILE_RECYCLEBIN.objname	对象未删除前的原始名称。
type	text	-	为固定值“TABLE”。
droptime	timestamp with time zone	PG_FILE_RECYCLEBIN.record_time	DROP操作的记录时间。
expires_at	interval	-	回收站保留TTL（数据生命周期），到期后由后台清理。
can_restore	boolean	-	是否可闪回，true表示可闪回。

17.3.226 PGXC_RECYCLEBIN_TRUNCATE

PGXC_RECYCLEBIN_TRUNCATE视图显示回收站里的TRUNCATE操作记录， 该视图仅9.1.2及以上集群版本支持。

表 17-325 PGXC_RECYCLEBIN_TRUNCATE 字段

名称	类型	引用	描述
recyclebin_name	name	PG_FILE_RECYCLEBIN.rbname	回收站中对象的内部名称。
database_name	name	PG_DATABASE.datname	对象所属数据库名称。
part_name	text	PG_FILE_RECYCLEBIN.subobjname	分区子对象名称；整表TRUNCATE时为NULL。
schema_name	name	PG_NAMESPACE.nspname	原对象所在schema名称。
original_name	name	PG_FILE_RECYCLEBIN.objname	对象原始名称。
type	text	-	为固定值“TABLE”。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
truncatetime	timestamp with time zone	PG_FILE_RECYCLEBIN.record_time	TRUNCATE操作的记录时间。
expires_at	interval	-	回收站保留TTL（数据生命周期），到期后由后台清理。
can_restore	boolean	-	是否可闪回，true表示可闪回。

17.3.227 PGXC_REDO_STAT

PGXC_REDO_STAT视图显示集群中各节点上XLOG重做过程中的统计信息，除新增node_name（节点名称）字段外，其余字段内容和PV_REDO_STAT视图相同。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-326 PGXC_REDO_STAT 字段

名称	类型	描述
node_name	text	节点名称。
phywrts	bigint	物理写次数。
phyblkwrt	bigint	物理写块数。
writetim	bigint	物理写消耗时间。
avgiotim	bigint	平均每次写入时间。
lstiotim	bigint	上一次写入时间。
miniotim	bigint	最小写入时间。
maxiowtm	bigint	最大写入时间。

17.3.228 PGXC_REL_IOSTAT

PGXC_REL_IOSTAT视图显示集群中各节点上磁盘读写的统计信息。需要有系统管理员权限才可以访问此视图。

表 17-327 PGXC_REL_IOSTAT 字段

名称	类型	描述
node_name	text	节点名称。
phyrds	bigint	读磁盘次数。
phywrts	bigint	写磁盘次数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
phyblkrd	bigint	读磁盘页数。
phyblkwrt	bigint	写磁盘页数。

17.3.229 PGXC_REPLICATION_SLOTS

PGXC_REPLICATION_SLOTS视图显示集群中DN上的复制信息，除增加表示节点名称的node_name字段外，其余字段内容和**PG_REPLICATION_SLOTS**视图相同。需要有系统管理员权限才可以访问此视图。

表 17-328 PGXC_REPLICATION_SLOTS 字段

名称	类型	描述
node_name	text	节点名称。
slot_name	text	复制节点的名称。
plugin	name	逻辑复制槽对应的输出插件名。
slot_type	text	复制节点的类型。
datoid	oid	复制节点的数据库OID。
database	name	复制节点的数据库名称。
active	boolean	复制节点是否为激活状态。
xmin	xid	复制节点事务标识。
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识。
restart_lsn	text	复制节点的Xlog文件信息。
dummy_standby	boolean	复制节点是否为假备。

17.3.230 PGXC_RESPOOL_RUNTIME_INFO

PGXC_RESPOOL_RUNTIME_INFO视图显示所有CN上所有资源池作业运行信息。

表 17-329 PGXC_RESPOOL_RUNTIME_INFO 字段

名称	类型	描述
nodename	name	CN名称。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
rpname	name	资源池名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数。
fast_run	int	资源池快车道运行作业数。
fast_wait	int	资源池快车道排队作业数。
slow_run	int	资源池慢车道运行作业数。
slow_wait	int	资源池慢车道排队作业数。

17.3.231 PGXC_RESPOOL_RESOURCE_INFO

PGXC_RESPOOL_RESOURCE_INFO视图显示所有实例上资源池实时监控信息。

说明

- DN上仅显示当前DN所属逻辑集群的资源池监控信息。
- 从8.2.0集群版本开始，新增了内存负反馈机制功能，CCN节点会根据DN节点的实际内存使用情况辅助反向调节语句的估算内存统计值，缓解语句估算内存高估场景，当CCN节点触发通过负反馈机制来减少估算内存统计值来增发作业时，由于CCN节点更新了语句缩减后的估算内存统计值，而CN节点还保留了原先的估算内存统计值，会导致资源池监控视图中的估算内存超过资源池的上限。
- 语句的算子分为重内存算子和非重内存算子，对于语句的内存管控是重内存算子，非重内存算子的内存开销较小；由于非重内存算子的内存开销、线程初始化的开销、表达式的开销等这些内存不做管控，会导致资源池的used_mem会在一定程度上超过mem_limit值。

表 17-330 PGXC_RESPOOL_RESOURCE_INFO 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示"installation"。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。
used_cpu	double	资源池5秒监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为DWS可用CPU，CPU限额管控情况下为关联控制组可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池当前使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5秒监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5秒监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5秒监控周期内逻辑读次数。 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
write_counts	bigint	资源池5秒监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。
read_speed	double	资源池5秒监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5秒监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5秒监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5秒监控周期内网络接收速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。
obs_io_read_kbytes	bigint	资源池5秒监控周期内OBS IO读字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO读字节数。 ● CN：显示所有DN上资源池OBS IO读字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	资源池5秒监控周期内OBS IO写字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO写字节数。 ● CN：显示所有DN上资源池OBS IO写字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_read_bps	double	资源池5秒监控周期内OBS IO读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO读速率。 ● CN：显示所有DN上资源池OBS IO读速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double	资源池5秒监控周期内OBS IO写速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO写速率。 ● CN：显示所有DN上资源池OBS IO写速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。

17.3.232 PGXC_RESPOOL_RESOURCE_HISTORY

PGXC_RESPOOL_RESOURCE_HISTORY视图用于查询所有实例上资源池监控历史信息。

表 17-331 PGXC_RESPOOL_RESOURCE_HISTORY 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
timestamp	timestamp	资源池监控信息持久化时间。
nodegroup	name	资源池所属逻辑集群名称，默认集群显示“installation”。
rpname	name	资源池名称。
cgroup	name	资源池关联控制组名称。
ref_count	int	资源池引用作业数，作业经过资源池不管是否管控都会计数，仅CN上有效。
fast_run	int	资源池快车道运行作业数，只在CN上有效。
fast_wait	int	资源池快车道排队作业数，只在CN上有效。
fast_limit	int	资源池快车道作业并发限制，只在CN上有效。
slow_run	int	资源池慢车道运行作业数，只在CN上有效。
slow_wait	int	资源池慢车道排队作业数，只在CN上有效。
slow_limit	int	资源池慢车道作业并发限制，只在CN上有效。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_cpu	double	资源池5秒监控周期内使用CPU个数平均值，保留小数点后2位。 ● DN：显示当前DN上资源池使用的CPU个数。 ● CN：显示所有DN上资源池使用CPU的累积和。
cpu_limit	int	资源池可用CPU的上限，CPU配额管控情况下为DWS可用CPU，CPU限额管控情况下为关联控制组可用CPU。 ● DN：显示当前DN上资源池可用CPU上限。 ● CN：显示所有DN上资源池可用CPU上限的累积和。
used_mem	int	资源池使用的内存大小，单位MB。 ● DN：显示当前DN上资源池使用的内存大小。 ● CN：显示所有DN上资源池使用内存的累积和。
estimate_mem	int	当前CN上，资源池运行作业的估算内存之和，只在CN上有效。
mem_limit	int	资源池可用内存上限，单位MB。 ● DN：显示当前DN上资源池可用内存上限。 ● CN：显示所有DN上资源池可用内存上限的累积和。
read_kbytes	bigint	资源池5秒监控周期内逻辑读字节数，单位KB。 ● DN：显示当前DN上资源池逻辑读字节数。 ● CN：显示所有DN上资源池逻辑读字节的累积和。
write_kbytes	bigint	资源池5秒监控周期内逻辑写字节数，单位KB。 ● DN：显示当前DN上资源池逻辑写字节数。 ● CN：显示所有DN上资源池逻辑写字节的累积和。
read_counts	bigint	资源池5秒监控周期内逻辑读次数 ● DN：显示当前DN上资源池逻辑读次数。 ● CN：显示所有DN上资源池逻辑读次数的累积和。
write_counts	bigint	资源池5秒监控周期内逻辑写次数。 ● DN：显示当前DN上资源池逻辑写次数。 ● CN：显示所有DN上资源池逻辑写次数的累积和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_speed	double	资源池5秒监控周期内逻辑读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑读速率。 ● CN：显示所有DN上资源池逻辑读速率的累积和。
write_speed	double	资源池5秒监控周期内逻辑写速率平均值，单位KB/s。 ● DN：显示当前DN上资源池逻辑写速率。 ● CN：显示所有DN上资源池逻辑写速率的累积和。
send_speed	double	资源池5秒监控周期内网络发送速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络发送速率。 ● CN：显示所有DN上资源池网络发送速率的累积和。
recv_speed	double	资源池5秒监控周期内网络接收速率平均值，单位KB/s。 ● DN：显示当前DN上资源池网络接收速率。 ● CN：显示所有DN上资源池网络接收速率的累积和。
obs_io_read_kbytes	bigint	资源池5秒监控周期内OBS IO读字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO读字节数。 ● CN：显示所有DN上资源池OBS IO读字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	资源池5秒监控周期内OBS IO写字节数，单位KB。 ● DN：显示当前DN上资源池OBS IO写字节数。 ● CN：显示所有DN上资源池OBS IO写字节的累积和。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double	资源池5秒监控周期内OBS IO读速率的平均值，单位KB/s。 ● DN：显示当前DN上资源池OBS IO读速率。 ● CN：显示所有DN上资源池OBS IO读速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_write_bps	double	资源池5秒监控周期内OBS IO写速率的平均值，单位KB/s。 • DN：显示当前DN上资源池OBS IO写速率。 • CN：显示所有DN上资源池OBS IO写速率的累积和。 该字段仅9.1.1.200及以上集群版本支持。

17.3.233 PGXC_ROW_TABLE_IO_STAT

PGXC_ROW_TABLE_IO_STAT视图提供集群所有CN和DN节点上当前数据库所有行存表的IO统计数据。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与GS_ROW_TABLE_IO_STAT视图相同，具体的字段请参考**表17-332**。

表 17-332 GS_ROW_TABLE_IO_STAT 字段

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
heap_read	bigint	堆逻辑读块数。
heap_hit	bigint	堆命中块数。
idx_read	bigint	索引逻辑读块数。
idx_hit	bigint	索引命中块数。
toast_read	bigint	TOAST表逻辑读块数。
toast_hit	bigint	TOAST表命中块数。
tidx_read	bigint	TOAST表Index逻辑读个数。
tidx_hit	bigint	TOAST表Index命中个数。

17.3.234 PGXC_RUNNING_XACTS

PGXC_RUNNING_XACTS视图主要功能是显示集群中各个节点运行事务的信息，字段内容和**PG_RUNNING_XACTS**相同。

表 17-333 PGXC_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
gxid	xid	事务ID号。
state	tinyint	事务状态（3：prepared或者0：starting）。
node	text	节点名称。
xmin	xid	节点上当前数据涉及的最小事务号xmin。
vacuum	boolean	标志当前事务是否是lazy vacuum事务。
timeline	bigint	标识数据库重启次数。
prepare_xid	xid	处于prepared状态事务的id号，若不在prepared状态，值为0。
pid	bigint	事务对应的线程ID。
next_xid	xid	CN传给DN的事务ID号。

17.3.235 PGXC_SETTINGS

PGXC_SETTINGS视图显示集群中各节点数据库运行时参数的相关信息，除新增node_name（节点名称）字段外，其余字段内容和PG_SETTINGS视图相同。需要有系统管理员权限才可以访问此视图。

表 17-334 PGXC_SETTINGS 字段

名称	类型	描述
node_name	text	节点名称。
name	text	参数名称。
setting	text	参数当前值。
unit	text	参数的隐式结构。
category	text	参数的逻辑组。
short_desc	text	参数的简单描述。
extra_desc	text	参数的详细描述。
context	text	设置参数值的上下文，包括internal，postmaster，sighup，backend，superuser和user。
vartype	text	参数类型，包括bool，enum，integer，real和string。
source	text	参数的赋值方式。
min_val	text	参数最小值。如果参数类型不是数值型，那么该字段值为null。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_val	text	参数最大值。如果参数类型不是数值型，那么该字段值为null。
enumvals	text[]	enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。
boot_val	text	数据库启动时参数默认值。
reset_val	text	数据库重置时参数默认值。
sourcefile	text	设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。
sourceline	integer	设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。
sync_to_stream	布尔值	标记是否在Stream线程同步设置参数的变更情况。该字段仅9.1.1.300及以上集群版本支持。

17.3.236 PGXC_SESSION_WLMSTAT

PGXC_SESSION_WLMSTAT视图显示当前集群中各CN节点用户执行作业正在运行时的负载管理相关信息。

表 17-335 PGXC_SESSION_WLMSTAT 字段

名称	类型	描述
nodename	name	节点名称。
datid	oid	连接后端的数据库OID。
datname	name	连接后端的数据库名称。
threadid	bigint	后端线程ID。
processid	integer	后端线程的PID。
usesysid	oid	登录后端的用户OID。
appname	text	连接到后端的应用名。
usename	name	登录到该后端的用户名。
priority	bigint	语句所在Cgroups的优先级，数值越大表示作业优先级越高。 ● 0：表示该作业不受负载管理管控。 ● 1：LOW ● 2：MEDIUM ● 4：HIGH ● 8：RUSH

版权所有 © 华为云计算技术有限公司

名称	类型	描述
attribute	text	语句的属性： ● Ordinary：语句发送到数据库后被解析前的默认属性。 ● Simple：简单语句。 ● Complicated：复杂语句。 ● Internal：数据库内部语句。
block_time	bigint	语句当前为止的pending的时间，单位s。
elapsed_time	bigint	语句当前为止的实际执行时间，单位s。
total_cpu_time	bigint	语句在上一时间周期内的DN上CPU使用的总时间，单位s。
cpu_skew_percent	integer	语句在上一时间周期内的DN上CPU使用的倾斜率。
statement_mem	integer	语句执行所需要的估算内存。
active_points	integer	语句占用的资源池并发点数。
dop_value	integer	从资源池中获取语句的dop值。
control_group	text	语句当前所使用的Cgroups。
status	text	语句当前的状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active：非以上四种状态外的正常状态。 ● unknown：未知状态。
enqueue	text	语句当前的排队情况，包括： ● Global：全局排队。 ● Respool：资源池排队。 ● CentralQueue：在中心协调节点(CCN)中排队。 ● Transaction：语句处于一个事务块中。 ● StoredProc：语句处于一个存储过程中。 ● None：未在排队。 ● Forced None：事务块语句或存储过程语句由于超出设定的等待时间而强制执行。
resource_pool	name	语句当前所在的资源池。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
query	text	该后端的最新查询。如果status状态是active，此字段显示当前正在执行的查询。所有其他情况表示上一个查询。
is_plana	bool	逻辑集群模式下，语句当前是否占用其他逻辑集群的资源执行。该值默认为f，表示不占用其他逻辑集群的资源执行。
node_group	text	语句所属用户对应的逻辑集群。
lane	text	表示语句查询的快慢车道。 ● fast：快车道。 ● slow：慢车道。 ● none：未管控。

17.3.237 PGXC_STAT_ACTIVITY

PGXC_STAT_ACTIVITY视图显示当前集群下所有CN的当前用户查询相关的信息。

表 17-336 PGXC_STAT_ACTIVITY 字段

名称	类型	描述
coorname	text	当前集群下的CN名称。
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
lwtid	integer	后端线程的轻量级线程号。
usesysid	oid	登录此后端的用户OID。
usename	name	登录此后端的用户名。
application_name	text	连接到此后端的应用名。
client_addr	inet	连接到此后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，此字段是通过client_addr的反向DNS查找得到。仅在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁或者等待节点则为t，否则为f。
enqueue	text	语句当前排队状态。可能值是： ● waiting in global queue：表示语句在全局并发队列排队中。 ● waiting in respool queue：表示语句在资源池排队中，包含以下两类场景： 1. 动态负载开启的情况下，简单作业数量超过了快车道并发上限max_dop。 2. 动态负载关闭的情况下，简单作业数量超过了快车道并发上限max_dop或者复杂作业数量超过了慢车道并发上限。 ● waiting in ccn queue：表示作业在CCN排队中，包含全局内存排队和慢车道内存和并发排队。 ● 空或no waiting queue：表示语句正在运行。
state	text	后端当前总体状态。可能值是： ● active：后端正在执行一个查询。 ● idle：后端正在等待一个新的客户端命令。 ● idle in transaction：后端在事务中，但事务中没有语句在执行。 ● idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 ● fastpath function call：后端正在执行一个fast-path函数。 ● disabled：如果后端禁用track_activities，则报告此状态。 **说明** 只有系统管理员能查看到自己账户所对应的会话状态。其他账户的state信息为空。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
resource_pool	name	用户使用的资源池。
stmt_type	text	用户语句的语句类型。
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

应用实例

查看当前处于阻塞状态的查询语句。

```
SELECT datname,usename,state,query FROM PGXC_STAT_ACTIVITY WHERE waiting = true;
```

查看快照线程的工作状态。

```
SELECT application_name,backend_start,state_change,state,query FROM PGXC_STAT_ACTIVITY WHERE application_name='WDRSnapshot';
```

查看正在运行的查询语句。

```
SELECT datname,usename,state,pid FROM PGXC_STAT_ACTIVITY;
```

```
postgres=# SELECT datname,usename,state,pid FROM PGXC_STAT_ACTIVITY;
 datname  | usename |  state   |       pid
----------+---------+----------+-----------------
 postgres | Ruby    | active   | 139872635409576
 postgres | Ruby    | active   | 139872635412216
 postgres | Ruby    | idle     | 139872635410016
 postgres | Ruby    | active   | 139872635413096
 postgres | Ruby    | active   | 139872635413536
 postgres | Ruby    | active   | 139872635412656
 postgres | Ruby    | idle     | 139872635410896
 postgres | Ruby    | idle     | 139872635413976
 postgres | Ruby    | active   | 139872635414416
 postgres | Ruby    | idle     | 139872635414856
 postgres | Ruby    | idle     | 139872635415296
 postgres | Ruby    | idle     | 139872635707456
```

查看指定数据库postgres上已使用的会话连接数。其中1表示数据库postgres上已使用的会话连接数。

```
SELECT COUNT(*) FROM PGXC_STAT_ACTIVITY WHERE DATNAME='postgres';
```

```
postgres=# SELECT COUNT(*) FROM PGXC_STAT_ACTIVITY WHERE DATNAME='postgres';
 count
-------
    51
(1 row)
```

 版权所有 © 华为云计算技术有限公司

17.3.238 PGXC_STAT_BAD_BLOCK

PGXC_STAT_BAD_BLOCK视图显示集群所有节点从启动后，在读取数据时出现Page/CU校验失败的统计信息。

表 17-337 PGXC_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称。
databaseid	oid	数据库OID。
tablespaceid	oid	表空间OID。
relfilenode	oid	文件对象ID。
forknum	integer	文件类型。
error_count	integer	出现校验失败的次数。
first_time	timestamp with time zone	第一次出现的时间。
last_time	timestamp with time zone	最近一次出现的时间。

17.3.239 PGXC_STAT_BGWRITER

PGXC_STAT_BGWRITER视图显示集群中各节点上后端写进程活动的统计信息，除新增node_name（节点名称）字段外，其余字段内容和**PG_STAT_BGWRITER**视图相同。需要有系统管理员权限才可以访问此视图。

表 17-338 PGXC_STAT_BGWRITER 字段

名称	类型	描述
node_name	text	节点名称。
checkpoints_timed	bigint	定期执行的检查点数量。
checkpoints_req	bigint	请求执行的检查点数量。
checkpoint_write_time	double precision	检查点期间将文件写入磁盘花费的时间，以毫秒为单位。
checkpoint_sync_time	double precision	检查点期间数据同步到磁盘花费的时间，以毫秒为单位。
buffers_checkpoint	bigint	检查点期间写入缓冲区的数量。
buffers_clean	bigint	后端写进程写的缓冲区数量。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
maxwritten_clean	bigint	由于写入缓冲区太多，后端写进程停止清理扫描的次数。
buffers_backend	bigint	后端直接写入的缓冲区数量。
buffers_backend_fsync	bigint	后端需要fsync的次数。
buffers_alloc	bigint	分配的缓冲区数量。
stats_reset	timestamp with time zone	统计重置的时间。

17.3.240 PGXC_STAT_DATABASE

视图PGXC_STAT_DATABASE显示集群中各节点上数据库的状态和统计信息，除新增node_name（节点名称）字段外，其余字段内容和**PG_STAT_DATABASE**视图相同。需要有系统管理员权限才可以访问此视图。

表 17-339 PGXC_STAT_DATABASE 字段

名称	类型	描述
node_name	text	节点名称。
datid	oid	数据库OID。
datname	name	数据库名。
numbackends	integer	当前节点上连接到该数据库的后端数。 这是该视图中唯一一个反映目前状态值的列；所有列均返回自上次重置以来的累积值。
xact_commit	bigint	当前节点上该数据库中已经提交的事务数。
xact_rollback	bigint	当前节点上该数据库中已经回滚的事务数。
blks_read	bigint	当前节点上该数据库中读取的磁盘块的数量。
blks_hit	bigint	当前节点上高速缓存中发现的磁盘块的个数，即缓存中命中的块数（只包括DWS缓冲区高速缓存，不包括文件系统的缓存）。
tup_returned	bigint	当前节点上该数据库查询返回的行数。
tup_fetched	bigint	当前节点上该数据库查询抓取的行数。
tup_inserted	bigint	当前节点上该数据库插入的行数。
tup_updated	bigint	当前节点上该数据库更新的行数。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
tup_deleted	bigint	当前节点上该数据库删除的行数。
conflicts	bigint	当前节点上由于数据库恢复冲突取消的查询数量（只在备用服务器上发生）。可参见PG_STAT_DATABASE_CONFLICTS。
temp_files	bigint	当前节点上该数据库创建的临时文件个数。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不考虑log_temp_files设置。
temp_bytes	bigint	当前节点上该数据库写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。
deadlocks	bigint	当前节点上该数据库中发生的死锁数量。
blk_read_time	double precision	当前节点上该数据库后端读取数据文件块花费的时间，以毫秒计算。
blk_write_time	double precision	当前节点上该数据库后端写入数据文件块花费的时间，以毫秒计算。
stats_reset	timestamp with time zone	当前节点上该数据库统计重置的时间。

17.3.241 PGXC_STAT_OBJECT

PGXC_STAT_OBJECT视图显示集群中所有实例的表的统计信息和autovacuum效率信息。该系统视图仅8.2.1及以上集群版本支持。

表 17-340 PGXC_STAT_OBJECT 字段

名称	类型	引用	描述
nodename	name	-	节点名称。
datname	name	-	表所在数据库名称。
relnamespace	name	-	表所在schema名称。
relname	name	-	表名。
partname	name	-	分区表的分区名。
databaseid	oid	PG_DATABASE.oid	数据库OID。
relid	oid	PG_CLASS.oid	表OID，分区表为主表OID。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
partid	oid	PG_PARTITION.oid	分区OID，普通表此列为0。
numscans	bigint	-	启动顺序扫描的次数。
tuples_returned	bigint	-	顺序扫描抓取的可见元组条数。
tuples_fetched	bigint	-	抓取的可见元组条数。
tuples_inserted	bigint	-	插入条数。
tuples_updated	bigint	-	更新条数。
tuples_deleted	bigint	-	删除条数。
tuples_hot_updated	bigint	-	HOT更新条数。
n_live_tuples	bigint	-	可见元组数。
last_autovacuum_begin_n_dead_tuple	bigint	-	Autovacuum执行前删除元组数。
n_dead_tuples	bigint	-	Autovacuum成功后删除元组数。
changes_since_analyze	bigint	-	Analyze后最近一次数据修改时间。
blocks_fetched	bigint	-	选中的页面数。
blocks_hit	bigint	-	扫描过的页面数。
cu_mem_hit	bigint	-	CU内存命中次数。
cu_hdd_sync	bigint	-	从磁盘同步读取CU次数。
cu_hdd_asyn	bigint	-	从磁盘异步读取CU次数。
data_changed_timestamp	timestamp with time zone	-	最近一次数据修改时间。
data_access_timestamp	timestamp with time zone	-	表的最后一次访问时间。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
analyze_timestamp	timestamp with time zone	-	最近一次Analyze时间。
analyze_count	bigint	-	Analyze总次数。
autovac_analyze_timestamp	timestamp with time zone	-	最近一次autoanalyze时间。
autovac_analyze_count	bigint	-	Autoanalyze总次数。
vacuum_timestamp	timestamp with time zone	-	最近一次vacuum的时间。
vacuum_count	bigint	-	vacuum总次数。
autovac_vacuum_timestamp	timestamp with time zone	-	最近一次autovacuum时间。
autovac_vacuum_count	bigint	-	Autovacuum总次数。
autovacuum_success_count	bigint	-	成功执行的autovacuum总次数。
last_autovacuum_time_cost	bigint	-	最近一次成功的autovacuum花费时间，单位：微秒。
avg_autovacuum_time_cost	bigint	-	成功执行autovacuum的平均执行时间，单位：微秒。
last_autovacuum_failed_count	bigint	-	从上一次autovacuum成功到现在，autovacuum总失败次数。
last_autovacuum_trigger	smallint	-	最近一次autovacuum触发方式，用于辅助维护人员进行vacuum情况的判断。
last_autovacuum_oldestxmin	bigint	-	最近一次autovacuum成功执行后的oldestxmin。如果表级oldestxmin特性开启，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值。
last_autovacuum_scan_pages	bigint	-	最近一次autovacuum扫描的页面数（仅针对行存表）。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
last_autovacuum_dirty_pages	bigint	-	最近一次autovacuum修改的页面数（仅针对行存表）。
last_autovacuum_clear_deadtuples	bigint	-	最近一次autovacuum清理的deadtuple数（仅针对行存表）。
sum_autovacuum_scan_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计扫描的页面数（仅针对行存表）。
sum_autovacuum_dirty_pages	bigint	-	从数据库初始化开始到现在，autovacuum累计修改的页面数（仅针对行存表）。
sum_autovacuum_clear_deadtuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的deadtuple数（仅针对行存表）。
last_autovacuum_begin_cu_size	bigint	-	最近一次autovacuum前的CU文件大小（仅针对列存表）。
last_autovacuum_cu_size	bigint	-	最近一次autovacuum后的CU文件大小（仅针对列存表）。
last_autovacuum_rewrite_size	bigint	-	最近一次autovacuum重写的列存文件大小（仅针对列存表）。
last_autovacuum_clear_size	bigint	-	最近一次autovacuum清理的列存文件大小（仅针对列存表）。
last_autovacuum_clear_cbtree_tuples	bigint	-	最近一次autovacuum清理的cbtree tuple数（仅针对列存表）。
sum_autovacuum_rewrite_size	bigint	-	从数据库初始化开始到现在，autovacuum累计重写的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_size	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的列存文件大小（仅针对列存表）。
sum_autovacuum_clear_cbtree_tuples	bigint	-	从数据库初始化开始到现在，autovacuum累计清理的cbtree tuple数（仅针对列存表）。
last_autovacuum_csn	bigint	-	如果表级oldestxmin特性打开，此字段记录此表最近一次(auto)vacuum使用的oldestxmin值对应的CSN值。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
read_sql_count_1d	bigint	-	当日表的累计访问次数。该字段仅9.1.1.100及以上集群版本支持。
last_reference_timestamp	timestamp with time zone	-	表的最后一次访问时间（该字段仅8.3.0及以上集群版本支持）。 对应PG_STAT_OBJECT中data_changed_time_stamp（最后一次修改时间）和data_access_timestamp（最后一次访问时间）两者中距现在最近的时间。
extra1	bigint	-	预留字段1。
extra2	bigint	-	预留字段2。
extra3	bigint	-	预留字段3。
extra4	bigint	-	预留字段4。

17.3.242 PGXC_STAT_REPLICATION

PGXC_STAT_REPLICATION视图显示集群中各节点上日志同步的状态信息，除新增node_name（节点名称）字段外，其余字段内容和PG_STAT_REPLICATION视图相同。需要有系统管理员权限才可以访问此视图。

表 17-341 PGXC_STAT_REPLICATION 字段

名称	类型	描述
node_name	text	节点名称。
pid	bigint	线程的PID。
usesysid	oid	用户系统ID。
usename	name	用户名。
application_name	text	程序名称。
client_addr	inet	客户端地址。
client_hostname	text	客户端名。
client_port	integer	客户端端口号。
backend_start	timestamp with time zone	程序启动时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
state	text	日志复制的状态（追赶状态，还是一致的流状态）。
sender_sent_location	text	发送端发送日志位置。
receiver_write_location	text	接收端write日志位置。
receiver_flush_location	text	接收端flush日志位置。
receiver_replay_location	text	接收端replay日志位置。
sync_priority	integer	同步复制的优先级（0表示异步）。
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步）。

17.3.243 PGXC_STAT_TABLE_DIRTY

PGXC_STAT_TABLE_DIRTY显示当前集群中所有节点（CN和DN）上全部表的统计信息，并展示表在单节点（单CN级或者单DN级）的脏页率。该视图仅8.1.3及以上集群版本支持。

说明

该视图的统计信息依赖于ANALYZE，为获取最准确的信息请先对表进行ANALYZE。

表 17-342 PGXC_STAT_TABLE_DIRTY 字段

名称	类型	描述
nodename	text	节点名。
schema	name	表的模式名。
tablename	name	表名。
partname	name	分区表的分区名。
last_vacuum	timestampwith time zone	最后一次手动vacuum时间。
last_autovacuum	timestampwith time zone	最后一次autovacuum时间。
last_analyze	timestampwith time zone	最后一次非自动轮询分析的时间。
last_autoanalyze	timestampwith time zone	最后一次自动轮询分析的时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
vacuum_count	bigint	vacuum次数。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。
autoanalyze_count	bigint	自动分析的次数。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	HOT更新的行数。
n_tup_change	bigint	analyze之后改变的行数。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
dirty_rate	bigint	单节点的脏页率（单CN或单DN）。
last_data_changed	timestampwith time zone	记录表最后一次数据变化的时间。该字段在8.2.0及以上集群版本中固定为空，建议通过PG_STAT_ALL_TABLES查询。

使用建议

- 对于高脏页率的系统表，建议在确认当前没有用户操作该系统表时，再执行VACUUM FULL。
- 建议对脏页率超过80%的非系统表执行VACUUM FULL，用户也可根据业务场景自行选择是否执行VACUUM FULL。

使用场景

1. 查询全库所有用户表的整体脏页率：

```
select
    t1.schema,
    t1.tablename,
    t1.total_ins,
    t1.total_upd,
    t1.total_del,
    t1. total_tup_hot_upd,
    t1.total_change,
    t1.total_live,
    t1.total_dead,
    t1.total_dirty_rate,
    t1.max_dirty,
    t2.max_node,
    t1.min_dirty,
    t2.min_node
from
```


版权所有 © 华为云计算技术有限公司

```
    (select
        a.schema,
        a.tablename,
        sum(a.n_tup_ins) as total_ins,
        sum(a.n_tup_upd) as total_upd,
        sum(a.n_tup_del) as total_del,
        sum(a.n_tup_hot_upd) as total_tup_hot_upd,
        sum(a.n_tup_change) as total_change,
        sum(a.n_live_tup) as total_live,
        sum(a.n_dead_tup) as total_dead,
        Round((total_dead / (total_dead + total_live + 0.0001) * 100),2) AS total_dirty_rate,
        max(a.dirty_rate) as max_dirty,
        min(a.dirty_rate) as min_dirty
    from pg_catalog.pgxc_stat_table_dirty a where a.partname is null and a.schema not in
('pg_toast','cstore','gs_logical_cluster','sys','dbms_om','information_schema','pg_catalog','dbms_output','
dbms_random','utl_raw','utl_raw dbms_sql','dbms_lob') group by a.tablename, a.schema
    ) t1,
    (select distinct
    tablename, schema,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate) as min_node,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate desc) as max_node
    from (select * from pg_catalog.pgxc_stat_table_dirty)) t2
where t1.tablename = t2.tablename and t1.schema = t2.schema;
```

2. 查询全库所有表(用户表+系统表)的整体脏页率：

```
select
    t1.schema,
    t1.tablename,
    t1.total_ins,
    t1.total_upd,
    t1.total_del,
    t1. total_tup_hot_upd,
    t1.total_change,
    t1.total_live,
    t1.total_dead,
    t1.total_dirty_rate,
    t1.max_dirty,
    t2.max_node,
    t1.min_dirty,
    t2.min_node
from
    (select
        a.schema,
        a.tablename,
        sum(a.n_tup_ins) as total_ins,
        sum(a.n_tup_upd) as total_upd,
        sum(a.n_tup_del) as total_del,
        sum(a.n_tup_hot_upd) as total_tup_hot_upd,
        sum(a.n_tup_change) as total_change,
        sum(a.n_live_tup) as total_live,
        sum(a.n_dead_tup) as total_dead,
        Round((total_dead / (total_dead + total_live + 0.0001) * 100),2) AS total_dirty_rate,
        max(a.dirty_rate) as max_dirty,
        min(a.dirty_rate) as min_dirty
    from pg_catalog.pgxc_stat_table_dirty a where a.partname is null group by a.tablename, a.schema
    ) t1,
    (select distinct
    tablename, schema,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate) as min_node,
    first_value(nodename) over(partition by tablename, schema order by dirty_rate desc) as max_node
    from (select * from pg_catalog.pgxc_stat_table_dirty)) t2
where t1.tablename = t2.tablename and t1.schema = t2.schema;
```

3. 查询全库系统表信息：

```
select * from pgxc_stat_table_dirty where schema in
('pg_toast','cstore','gs_logical_cluster','sys','dbms_om','information_schema','pg_catalog','dbms_output','
dbms_random','utl_raw','utl_raw dbms_sql','dbms_lob');
```

 版权所有 © 华为云计算技术有限公司

17.3.244 PGXC_STAT_WAL

PGXC_STAT_WAL视图显示当前query的wal日志和数据页的流量信息，该视图仅8.2.0及以上集群版本支持。

表 17-343 PGXC_STAT_WAL 字段

名称	类型	描述
query_id	bigint	当前query的ID。
query_start	timestamp	query起始时间。
global_wal	bigint	当前query在集群产生的wal日志总量，单位是Byte。
global_avg_wal_speed	bigint	当前query在集群产生wal日志的平均速率，单位是Byte/s。
global_datapage	bigint	当前query在集群产生的数据页总量，单位是Byte。
global_avg_datapage_speed	bigint	当前query在集群产生数据页的平均速率，单位是Byte/s。
min_wal_node	Text	当前query产生wal日志量最小的实例组名。
min_wal	bigint	最小node产生的wal日志量，单位是Byte。
max_wal_node	Text	当前query产生wal日志量最大的实例组名。
max_wal	bigint	最大node产生的wal日志量，单位是Byte。
min_datapage_node	Text	当前query产生数据页量最小的实例组名。
min_data_page	bigint	最小node产生的数据页量，单位是Byte。
max_datapage_node	Text	当前query产生数据页量最大的实例组名。
max_data_page	bigint	最大node产生的数据页量，单位是Byte。
avg_wal_per_node	bigint	平均每个node产生的wal日志量，单位是Byte。
avg_datapage_per_node	bigint	平均每个node产生的数据页量，单位是Byte。
query	Text	当前执行的语句。

说明

行存不带索引批量导入时，数据页copy导入会产生logical newpage相关的xlog日志，在xlog量大于默认值时同样会触发流控。

版权所有 © 华为云计算技术有限公司

应用示例

查询集群当前正在执行的语句，以及这些语句分别产生的wal日志的总量和平均速率、数据页总量和平均速率，以及这些信息各自DN上的分布情况。

```
SELECT * FROM PGXC_STAT_WAL;
     query_id      |          query_start          | global_wal | global_avg_wal_speed | global_datapage |
global_avg_datapage_speed | min_wal_node | min_wal  | max_wal_node | max_wal  | min_datapage_node |
min_data_page | max_datapage_node | max_data_page | avg_wal_per_node |
avg_datapage_per_node |                    query
-------------------+-------------------------------+------------+----------------------+-----------------
+--------------------------+--------------+----------+--------------+----------+------------------+---------------
+------------------+---------------+------------------+-
----------------------+-------------------------------------------------
 72620543991351767 | 2022-11-10 16:49:47.743291+08 |    7579052 |              419000 |       284057600
|                15740000 | datanode1    |  7579052 | datanode1    |  7579052 | datanode1         |     284057600
| datanode1         |     284057600 |         7579052 |
          284057600 | insert into mpptest3 select * from mpptest3;
 72620543991351781 | 2022-11-10 16:50:00.616697+08 |   55022176 |            10638000 |             0
|                       0 | datanode1    | 55022176 | datanode1    | 55022176 | datanode1         |             0 |
datanode1         |             0 |        55022176 |
                  0 | insert into mpptest1 select * from mpptest1;
(2 rows)
```

17.3.245 PGXC_SQL_COUNT

通过PGXC_SQL_COUNT视图，可以实时显示集群中各CN节点上SELECT、INSERT、UPDATE、DELETE、MERGE INTO五种SQL、以及DDL、DML、DCL语句的节点级和用户级统计结果，识别当前业务负载较重的query类型，衡量整个集群和单个节点执行某种类型查询的能力。通过对以上几类SQL查询进行计数和响应时间统计，获得指定时刻的统计结果，经计算可以得到指定QPS等统计信息。例如，T1时刻，USER1的SELECT计数结果为X1，T2时刻为X2，则可计算得到该用户SELECT查询的QPS值为(X2-X1)/(T2-T1)。由此，可获得集群用户级QPS曲线图和集群吞吐情况，监测每个用户的业务负载是否发生剧烈变化。如果有剧烈变化，可以定位具体的语句类型(SELECT/INSERT/UPDATE/DELETE/MERGE INTO)。同时观测QPS曲线可以获知问题发生时间点，结合其他工具，定位问题点。能够为集群性能调优、问题定位等提供依据。

PGXC_SQL_COUNT视图的字段与GS_SQL_COUNT一致，详见**表17-179**。

说明

当执行MERGE INTO语句时，若能下推，在DN上收到的是MERGE INTO语句，将在DN节点上进行MERGE INTO类型计数，相应mergeinto_count列计数增加；若不能下推，在DN上收到的是UPDATE或INSERT语句，将在DN节点上进行UPDATE或INSERT类型计数，相应的update_count列或insert_count列计数增加。

17.3.246 PGXC_TABLE_CHANGE_STAT

PGXC_TABLE_CHANGE_STAT视图提供集群所有CN节点上当前数据库所有表的变更情况。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与**GS_TABLE_CHANGE_STAT**视图相同。

表 17-344 PGXC_TABLE_CHANGE_STAT 字段

名称	类型	描述
nodename	name	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
schemaname	name	表的命名空间。
relname	name	表的名称。
last_vacuum	timestamp with time zone	最后一次手动vacuum的时间。
vacuum_count	bigint	手动Vacuum的次数。
last_autovacuum	timestamp with time zone	最后一次自动vacuum的时间。
autovacuum_count	bigint	自动vacuum的次数。
last_analyze	timestamp with time zone	最后一次非自动轮询分析的时间。
analyze_count	bigint	分析（包括手动和自动）的次数。
last_autoanalyze	timestamp with time zone	最后一次自动轮询分析的时间。
autoanalyze_count	bigint	自动分析的次数。
last_change	bigint	最后一次修改（INSERT，UPDATE或DELETE）的时间。

17.3.247 PGXC_TABLE_STAT

PGXC_TABLE_STAT视图提供集群所有CN和DN节点上当前数据库所有表的统计信息。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与**GS_TABLE_STAT**视图相同。

表 17-345 PGXC_TABLE_STAT 字段

名称	类型	描述
nodename	name	节点名称。
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。只统计行存表。如果是分区表，显示各个分区扫描次数的和。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
seq_tuple_read	bigint	顺序扫描的行数。只统计行存表。
index_scan	bigint	索引扫描的次数。只统计行存表。
index_tuple_read	bigint	索引扫描的行数。只统计行存表。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

17.3.248 PGXC_THREAD_WAIT_STATUS

通过CN节点查看PGXC_THREAD_WAIT_STATUS视图，可以查看集群全局各个节点上所有SQL语句产生的线程之间的调用层次关系，以及各个线程的阻塞等待状态，从而更容易定位进程停止响应问题以及类似现象的原因。

PGXC_THREAD_WAIT_STATUS视图和PG_THREAD_WAIT_STATUS视图列定义完全相同，这是由于PGXC_THREAD_WAIT_STATUS视图本质是到集群中各个节点上查询PG_THREAD_WAIT_STATUS视图汇总的结果。

表 17-346 PGXC_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见表17-273。
wait_event	text	• 当wait_status为acquire lock、acquire lwlock或wait io时，此列显示具体的锁、轻量级锁或IO等待信息。 • 当线程无其他等待事件且已开启实时算子运行状态监控时，此列显示当前算子在执行计划中的层级及已处理行数。 • 其他情况下，此列为空。

例如：

在coordinator1执行一条语句之后长时间没有响应。可以创建另外一个连接到coordinator1上，查询coordinator1上的线程状态。

```
 select * from pg_thread_wait_status where query_id > 0;
  node_name   | db_name  | thread_name  | query_id |       tid       | lwtid | ptid  | tlevel | smpid |
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+--------+-------
+----------------------
 coordinator1 | gaussdb | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |     0 | wait node:
datanode4 |
(1 rows)
```

此外，可以查看该语句在全局范围内各个节点上的工作情况。如下所示，每个DN上都没有在等待的阻塞资源，因为读取的数据太多而执行较慢。

```
select * from pgxc_thread_wait_status where query_id=20971544;
  node_name   | db_name  | thread_name  | query_id |       tid       | lwtid | ptid  | tlevel | smpid |
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+--------+-------
+----------------------
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902867994384 | 22735 |       |      0 |     0 | wait
node: datanode3 |
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902838634256 | 22970 | 22735 |      5 |     0 |
synchronize quit     |
 datanode1    | gaussdb | coordinator1 | 20971544 | 139902607947536 | 22972 | 22735 |      5 |     1 |
synchronize quit     |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632156796688 | 22736 |       |      0 |     0 | wait
node: datanode3 |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632030967568 | 22974 | 22736 |      5 |     0 |
synchronize quit     |
 datanode2    | gaussdb | coordinator1 | 20971544 | 140632081299216 | 22975 | 22736 |      5 |     1 |
synchronize quit     |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323627988752 | 22737 |       |      0 |     0 | wait
node: datanode3 |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323523131152 | 22976 | 22737 |      5 |     0 | net
flush data       |
 datanode3    | gaussdb | coordinator1 | 20971544 | 140323548296976 | 22978 | 22737 |      5 |     1 | net
flush data
 datanode4    | gaussdb | coordinator1 | 20971544 | 140103024375568 | 22738 |       |      0 |     0 | wait
node: datanode3
 datanode4    | gaussdb | coordinator1 | 20971544 | 140102919517968 | 22979 | 22738 |      5 |     0 |
synchronize quit     |
```

 版权所有 © 华为云计算技术有限公司

```
 datanode4     | gaussdb | coordinator1 | 20971544 | 140102969849616 | 22980 | 22738 |      5 |      1 |
synchronize quit     |
 coordinator1 | gaussdb | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |      0 | wait node:
datanode4  |
(13 rows)
```

17.3.249 PGXC_TOTAL_MEMORY_DETAIL

PGXC_TOTAL_MEMORY_DETAIL视图显示集群内存使用情况。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-347 PGXC_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存使用的名称。 ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。 ● gpu_max_dynamic_memory：GPU内存最大值。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● gpu_dynamic_peak_memory：GPU内存使用的最大内存。 ● pooler_conn_memory：pooler连接占用内存大小。 ● pooler_freeconn_memory：pooler空闲连接占用的内存大小。 ● storage_compress_memory：列存压缩和解压缩使用的内存大小。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
		• udf_reserved_memory：为UDF Worker进程预留的内存大小。 • mmap_used_memory：mmap使用的内存大小。
memorymbytes	integer	内存使用的大小，单位为MB。

17.3.250 PGXC_TOTAL_SCHEMA_INFO

PGXC_TOTAL_SCHEMA_INFO视图提供了集群所有实例上的Schema空间信息，便于用户获悉集群各个实例上的Schema空间使用情况，仅支持在CN节点上查询。

表 17-348 PGXC_TOTAL_SCHEMA_INFO 字段

名称	类型	描述
schemaname	text	模式名称。
schemaid	oid	模式OID。
databasename	text	数据库名称。
databaseid	oid	数据库OID。
nodename	text	实例名称。
nodegroup	text	节点组名称。
usedspace	bigint	已使用的空间大小，单位为Byte。
permspace	bigint	空间上限值，单位为Byte。

17.3.251 PGXC_TOTAL_SCHEMA_INFO_ANALYZE

PGXC_TOTAL_SCHEMA_INFO_ANALYZE视图提供了集群整体的Schema空间信息，包括：集群空间总值、各实例空间平均值、倾斜率、单实例空间最大值、单实例空间最小值以及最大最小空间所在的实例名，便于用户获悉集群整体的Schema空间使用情况，仅支持在CN节点上查询。

表 17-349 PGXC_TOTAL_SCHEMA_INFO_ANALYZE 字段

名称	类型	描述
schemaname	text	模式名称。
databasename	text	数据库名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
nodegroup	text	节点组名称。
total_value	bigint	该模式的集群空间总值。
avg_value	bigint	该模式的各实例空间平均值。
skew_percent	integer	倾斜率。
extend_info	text	提供信息包括：单实例空间最大值、单实例空间最小值以及最大最小空间所在的实例名。

17.3.252 PGXC_TOTAL_USER_RESOURCE_INFO

PGXC_TOTAL_USER_RESOURCE_INFO视图显示所有实例上用户实时资源消耗信息。该视图仅8.2.0及以上集群版本支持。

表 17-350 PGXC_TOTAL_USER_RESOURCE_INFO 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
username	name	用户名。
used_memory	integer	正在使用的内存大小，单位MB。 DN：显示当前DN上对应用户正在使用的内存大小。 CN：显示所有DN上对应用户正在使用的内存累加和。
total_memory	integer	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 DN：显示当前DN上对应用户可以使用的内存大小。 CN：显示所有DN上对应用户可以使用的内存大小之和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。 DN：显示当前DN上对应用户正在使用的CPU核数。 CN：显示所有DN上对应用户正在使用的CPU核数之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_cpu	integer	用户关联控制组的CPU核数总和。 DN：显示当前DN上对应用户能够使用的CPU核数。 CN：显示所有DN上对应用户能够使用的CPU核数之和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的永久表存储空间大小。 CN：显示所有DN上对应用户已使用的永久表存储空间大小之和。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。 DN：显示当前DN上对应用户可使用的永久表存储空间大小。 CN：显示所有DN上对应用户可使用的永久表存储空间大小之和。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的临时表存储空间大小。 CN：显示所有DN上对应用户已使用的临时表存储空间大小之和。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。 DN：显示当前DN上对应用户可使用的临时表存储空间大小。 CN：显示所有DN上对应用户可使用的临时表存储空间大小之和。
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。 DN：显示当前DN上对应用户已使用的算子落盘空间大小 CN：显示所有DN上对应用户已使用的算子落盘空间大小之和
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。 DN：显示当前DN上对应用户可使用的算子落盘空间大小。 CN：显示所有DN上对应用户可使用的算子落盘空间大小之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑读的字节总数。单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户逻辑写的字节总数。单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑读的次数之和。 DN：实例启动至当前时间为止，该用户逻辑读的次数之和。
write_counts	bigint	CN：过去5秒内，该用户在所有DN上逻辑写的次数之和。 DN：实例启动至当前时间为止，该用户逻辑写的次数之和。
read_speed	double precision	CN：过去5秒内，该用户在单个DN上逻辑读平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑读平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在单个DN上逻辑写平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上逻辑写平均速率。单位KB/s。
send_speed	double precision	CN：过去5秒内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：过去5秒内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上网络接收平均速率。单位KB/s。
obs_io_read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO读的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
obs_io_write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO写的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO读平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO读平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO写平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO写平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。

17.3.253 PGXC_USER_TRANSACTION

PGXC_USER_TRANSACTION视图提供查询所有CN上用户相关的事务信息。需要有系统管理员权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

表 17-351 PGXC_USER_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称。
usename	name	用户名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间。
resp_max	bigint	最大响应时间。
resp_avg	bigint	平均响应时间。
resp_total	bigint	响应时间总和。

17.3.254 PGXC_VARIABLE_INFO

PGXC_VARIABLE_INFO视图用于查询集群中所有节点的xid、oid的状态。

 版权所有 © 华为云计算技术有限公司

表 17-352 PGXC_VARIABLE_INFO 字段

名称	类型	描述
node_name	text	节点名称。
nextOid	oid	该节点下一次生成的OID。
nextXid	xid	该节点下一次生成的事务号。
oldestXid	xid	该节点最早的事务号。
xidVacLimit	xid	强制autovacuum的临界点。
oldestXidDB	oid	该节点datafrozenxid最小的数据库OID。
lastExtendCSNLogpage	integer	最后一次扩展csnlog的页面号。
startExtendCSNLogpage	integer	csnlog扩展的起始页面号。
nextCommitSeqNo	integer	该节点下次生成的csn号。
latestCompletedXid	xid	该节点提交或者回滚后节点上的最新事务号。
startupMaxXid	xid	该节点关机前的最后一个事务号。

17.3.255 PGXC_WAIT_DETAIL

PGXC_WAIT_DETAIL视图显示集群中所有节点SQL的详细等待链信息。该视图仅8.1.3.200及以上集群版本支持。

表 17-353 PGXC_WAIT_DETAIL 字段

名称	类型	描述
level	integer	等待链中的层级，以1开始，每显示一层等待关系level会加1。
lock_wait_hierarchy	text	等待链，以节点名称：进程号->节点名称：等待进程号->节点名称：等待进程号->...。
node_name	text	节点名称。
db_name	text	database名称。
thread_name	text	线程名称。
query_id	bigint	查询语句的id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。
wait_event	text	持有此锁或者在等待此锁的事务的虚拟id。
exec_cn	boolean	是否执行sql语句的cn节点。
wait_node	text	锁级别。
query	text	查询语句。
application_name	text	连接到该后端的应用名。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间。
query_start	timestamp with time zone	开始当前活跃查询的时间。
waiting	boolean	是否正处于等待状态。
state	text	后端当前总体状态。
waittime	timestamp with time zone	开始等待锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。
holdtime	timestamp with time zone	开始获取锁的时间戳。 该字段仅9.1.0.200及以上集群版本支持。

应用实例

步骤1 连接CN节点，开启一个事务，执行update操作：

```
begin;update td set c2=6 where c1=1;
```

步骤2 重开一个窗口连接CN节点，开启另一个事务，执行update操作（注意不要并发更新同一条记录）：

```
begin;update td set c2=6 where c1=7;
```

此时update操作会被阻塞。

 版权所有 © 华为云计算技术有限公司

步骤3 再开一个窗口连接CN节点，创建一个索引：

```
create index c2_key on td(c2);
```

步骤4 执行select * from pgxc_wait_detail;

```
SELECT * FROM PGXC_WAIT_DETAIL;
level |              lock_wait_hierarchy               | node_name | db_name  | thread_name |     query_id
|       tid       | lwtid  | ptid | tlevel | sm
pid | wait_status  | wait_event | exec_cn | wait_node |                query                | application_name |
backend_start         |          xact_st
art           |          query_start          | waiting | state
-------+------------------------------------------------+-----------+----------+-------------+------------------
+-----------------+--------+------+--------+---
----+--------------+------------+---------+-----------+----------------------------------------+-----------------
+-------------------------------+-----------------
-------------+-----------------------------+---------+--------
1 | cn_5001:139870843444360                        | cn_5001   | postgres | workload    | 73183493945299462 |
139870843444360 | 578531 |      |      0 |
0 | wait node    |           | t       |          | WLM fetch collect info from data nodes | workload         |
2023-03-13 13:56:56.611486+08 | 2023-03-14 11:54
:33.562808+08 | 2023-03-13 13:57:00.262736+08 | t       | active
1 | cn_5001:139870843654544                        | cn_5001   | postgres | gsql        | 73183493945299204 |
139870843654544 | 722259 |      |      0 |
0 | wait node    |           | t       |          | update td set c2=6 where c1=1;         | gsql             | 2023-03-14
11:52:05.176588+08 | 2023-03-14 11:52
:19.054727+08 | 2023-03-14 11:53:58.114794+08 | t       | active
1 | cn_5001:139870843655296                        | cn_5001   | postgres | gsql        | 73183493945299218 |
139870843655296 | 722301 |      |      0 |
0 | wait node    |           | t       |          | update td set c2=6 where c1=7;         | gsql             | 2023-03-14
11:52:08.084265+08 | 2023-03-14 11:52
:42.978132+08 | 2023-03-14 11:53:59.459575+08 | t       | active
1 | cn_5001:139870843656424                        | cn_5001   | postgres | gsql        | 73183493945299223 |
139870843656424 | 722344 |      |      0 |
0 | acquire lock | relation    | t       |          | create index c2_key on td(c2);          | gsql             | 2023-03-14
11:52:10.967028+08 | 2023-03-14 11:52
:53.463227+08 | 2023-03-14 11:54:00.25203+08  | t       | active
2 | cn_5001:139870843656424 -> cn_5001:139870843655296 | cn_5001   | postgres | gsql        |
73183493945299218 | 139870843655296 | 722344 |      |       |
|             |           | f       |          | update td set c2=6 where c1=7;         | gsql             | 2023-03-14
11:52:08.084265+08 | 2023-03-14 11:52
:42.978132+08 | 2023-03-14 11:53:59.459575+08 | t       | active
(5 rows)
```

----结束

17.3.256 PGXC_WAIT_EVENTS

PGXC_WAIT_EVENTS视图显示集群中各节点各类等待状态和事件的统计信息，其字段内容和**GS_WAIT_EVENTS**视图相同。需要有系统管理员权限才可以访问此视图。

表 17-354 PGXC_WAIT_EVENTS 字段

名称	类型	描述
nodename	name	节点名称。
type	text	事件的类型，包括STATUS，LOCK_EVENT，LWLOCK_EVENT和IO_EVENT四种类型。
event	text	事件名称，可参考**PG_THREAD_WAIT_STATUS**视图。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
wait	bigint	事件发生次数。该字段及以下字段均为进程运行中的累计值。
failed_wait	bigint	等待失败次数。当前版本中只有LOCK和LWLOCK等锁超时或失败才会使用该字段。
total_wait_time	bigint	该事件总持续时间。
avg_wait_time	bigint	该事件平均持续时间。
max_wait_time	bigint	该事件最大等待时间。
min_wait_time	bigint	该事件最小等待时间。

17.3.257 PGXC_WLM_OPERATOR_HISTORY

PGXC_WLM_OPERATOR_HISTORY视图显示在所有CN上执行作业结束时的算子信息。此视图用于从DWS数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟。

需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。具体的字段请参考**表17-355**。

表 17-355 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值，单位MB。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值，单位MB。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量，单位MB。默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量，单位MB。默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量，单位MB。默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间，单位毫秒。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间，单位毫秒。
total_cpu_time	bigint	该算子在所有DN上的总执行时间，单位毫秒。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

17.3.258 PGXC_WLM_OPERATOR_INFO

PGXC_WLM_OPERATOR_INFO视图显示在所有CN上执行作业结束时的算子信息。此视图的数据直接从系统表**GS_WLM_OPERATOR_INFO**获取。

需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

版权所有 © 华为云计算技术有限公司

说明

PGXC_WLM_OPERATOR_INFO视图表仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。

PGXC_WLM_OPERATOR_INFO视图的字段同**GS_WLM_OPERATOR_INFO**相同，具体字段内容如下：

表 17-356 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
nodename	text	执行语句的CN实例名称。
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值，单位MB。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量，单位MB。默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量，单位MB。默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量，单位MB。默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_cpu_time	bigint	该算子在所有DN上的最小执行时间，单位毫秒。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间，单位毫秒。
total_cpu_time	bigint	该算子在所有DN上的总执行时间，单位毫秒。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin

17.3.259 PGXC_WLM_OPERATOR_STATISTICS

PGXC_WLM_OPERATOR_STATISTICS视图显示在所有CN上正在执行作业的算子信息。系统管理员权限可以查询集群所有用户的作业算子信息，普通用户仅可查询自己的作业算子信息。

PGXC_WLM_OPERATOR_STATISTICS视图的字段同**表17-193**相同，具体字段内容如下：

表 17-357 GS_WLM_OPERATOR_STATISTICS 字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程ID。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称，算子名称长度最大为127字符（不包含空格等格式化字符）。
start_time	timestamp with time zone	该算子第一次开始执行的时间。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
duration	bigint	该算子从开始执行直到结束时总的执行时间(ms)。
status	text	当前算子的执行状态，包括waiting、running和finished。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息，若返回的预估行数超过int64_max时，显示为int64_max。
tuple_processed	bigint	当前算子在所有DN上的返回的元素个数总和，若返回的预估行数超过int64_max时，显示为int64_max。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小逻辑下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大逻辑下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均逻辑下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. The max spill size exceeds the alarm size 256MB 3. The max broadcast size exceeds the alarm size 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict 8. NestLoop in hashjoin
parent_id	integer	该算子节点的父节点id。
exec_count	integer	该算子节点在所有DN上的最大执行次数。
progress	text	该算子的进度信息，第一个算子展示的是作业整体的进度。其他算子展示的是当前算子进度信息。
min_net_size	bigint	该算子在所有DN上的最小网络通信数据量(KB)，主要涉及网络算子。
max_net_size	bigint	该算子在所有DN上的最大网络通信数据量(KB)，主要涉及网络算子。
total_net_size	bigint	该算子在所有DN上的总网络通信数据量(KB)，主要涉及网络算子。
min_read_bytes	bigint	该算子在所有DN上的最小磁盘读取数据量(KB)。
max_read_bytes	bigint	该算子在所有DN上的最大磁盘读取数据量(KB)。
total_read_bytes	bigint	该算子在所有DN上的总磁盘读取数据量(KB)。
min_write_bytes	bigint	该算子在所有DN上的最小磁盘写入数据量(KB)。
max_write_bytes	bigint	该算子在所有DN上的最大磁盘写入数据量(KB)。
total_write_bytes	bigint	该算子在所有DN上的总磁盘写入数据量(KB)。

 版权所有 © 华为云计算技术有限公司

17.3.260 PGXC_WLM_SESSION_INFO

PGXC_WLM_SESSION_INFO视图显示在所有CN上执行作业结束后的负载管理记录。视图信息来源于系统表**GS_WLM_SESSION_INFO**。

说明

- PGXC_WLM_SESSION_INFO视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 9.1.1.300及以上集群版本中系统会将执行成功且执行时间小于100ms，并且包含sql_hash的短查询进行聚合记录。9.1.2及以上集群版本中，子语句聚合不受100ms执行时间限制，满足其他聚合条件时默认进行聚合。聚合的主要列为datid、dbname、resource_pool、node_group、username、application_name、client_addr、sql_hash、plan_hash、query_band和nodename。

PGXC_WLM_SESSION_INFO视图的字段与**GS_WLM_SESSION_HISTORY**相同，具体字段内容如下：

表 17-358 GS_WLM_SESSION_HISTORY 字段

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
datid	oid	连接后端的数据库OID。	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。	连接后端的数据库名称。
schemaname	text	模式名。	聚合周期内，同聚合主键第一个语句的schemaname。
nodename	text	语句执行的CN名称。	语句执行的CN名称。
username	text	连接到后端的用户名。	连接到后端的用户名。
application_name	text	连接到后端的应用名。	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。	连接到后端的客户端的IP地址。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。	聚合周期内，同聚合主键第一个语句的客户端的client_hostname。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。	聚合周期内，同聚合主键第一个语句的client_port。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。	聚合周期内，同聚合主键第一个语句的query_band。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。	聚合周期内，同聚合主键所有语句的block_time平均值。
start_time	timestamp with time zone	语句执行的开始时间。	聚合周期内，同聚合主键第一个语句的start_time。
finish_time	timestamp with time zone	语句执行的结束时间。	聚合周期内，同聚合主键最后一个语句的finish_time。
duration	bigint	语句实际执行的时间，单位ms。	聚合周期内，同聚合主键所有语句的duration平均值。
estimate_total_time	bigint	语句预估执行时间，单位ms。	聚合周期内，同聚合主键所有语句的estimate_total_time平均值。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。	语句执行结束状态，仅聚合执行成功的语句，默认为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。	空。
resource_pool	text	用户使用的资源池。	用户使用的资源池。
control_group	text	语句所使用的Cgroup。	聚合周期内，同聚合主键第一个语句的control_group。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。	聚合周期内，同聚合主键所有语句的estimate_memory平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的min_peak_memory最小值。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的max_peak_memory最大值。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。	聚合周期内，同聚合主键所有语句的average_peak_memory平均值。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。	聚合周期内，同聚合主键所有语句的memory_skew_percent的平均值。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。	空
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的min_spill_size的最小值。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的max_spill_size的最大值。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的average_spill_size的平均值。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。	聚合周期内，同聚合主键所有语句的spill_skew_percent的平均值。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。	聚合周期内，同聚合主键所有语句的min_dn_time的最小值。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。	聚合周期内，同聚合主键所有语句的max_dn_time的最大值。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。	聚合周期内，同聚合主键所有语句的average_dn_time平均值。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。	聚合周期内，同聚合主键所有语句的dntime_skew_percent平均值。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的min_cpu_time最小值。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的max_cpu_time最大值。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。	聚合周期内，同聚合主键所有语句的total_cpu_time总量。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。	聚合周期内，同聚合主键所有语句的cpu_skew_percent平均值。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的min_peak_iops最小值。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的max_peak_iops最大值。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的average_peak_iops平均值。
iops_skew_percent	integer	语句在DN间的IO倾斜率。	聚合周期内，同聚合主键所有语句的iops_skew_percent平均值。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin	聚合周期内，同聚合主键第一个语句的warning。
queryid	bigint	语句执行使用的内部query id。	聚合周期内，同聚合主键第一个语句的queryid。
query	text	执行的语句，最多可保留64KB长度的字符串。	聚合周期内，同聚合主键第一个语句的query。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。	聚合周期内，同聚合主键第一个语句的query_plan。
node_group	text	语句所属用户对应的逻辑集群。	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。	聚合周期内，同聚合主键第一个语句的pid。
lane	text	语句执行时所在的快慢车道。	聚合周期内，同聚合主键第一个语句的lane。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
unique_sql_id	bigint	归一化的Unique SQL ID。	聚合周期内，同聚合主键第一个语句的unique_sql_id。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。	聚合周期内，同聚合主键第一个语句的session_id。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_read_bytes最小值。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_read_bytes最大值。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_read_bytes平均值。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_write_bytes最小值。
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_write_bytes最大值。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_write_bytes平均值。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。	聚合周期内，同聚合主键所有语句的recv_pkg总量。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。	聚合周期内，同聚合主键所有语句的send_pkg总量。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的recv_bytes总量。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的send_bytes总量。
stmt_type	text	语句对应的查询类型。	聚合周期内，同聚合主键第一个语句的stmt_type。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
except_info	text	语句触发的异常规则信息。	空
unique_plan_id	bigint	归一化的Unique plan id。	聚合周期内，同聚合主键第一个语句的unique_plan_id。
sql_hash	text	归一化的sql hash。	归一化的sql hash。
plan_hash	text	归一化的plan hash。	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。	聚合周期内，同聚合主键第一个语句的use_plan_baseline。
outline_name	text	该语句计划对应的outline的名字。	聚合周期内，同聚合主键第一个语句的outline_name。
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。	保存导入导出类业务信息的json串具体如下。 1. address：聚合周期内，同聚合主键第一个语句的address。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：聚合周期内，同聚合主键所有语句的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的parse_time平均值。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_hit_ratio总量。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_read_size总量。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_write_size总量。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_size总量。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_time总量。
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_scan_bytes总量。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_remote_read_bytes总量。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_submit_time总量。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_time总量。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_count总量。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_load_time总量。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_conflict_count总量。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_error_count总量。
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。	空
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_rtt总量。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_latency总量。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_1s总量。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_10s总量。
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_count总量。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_retry_count总量。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_rate_limit_count总量。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键第一个语句的gxid。
inserted_rows	bigint	语句插入的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的inserted_rows平均值
deleted_rows	bigint	语句删除的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的deleted_rows平均值
updated_rows	bigint	语句更新的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的updated_rows平均值
returned_rows	bigint	语句查询的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的returned_rows平均值
obs_io_read_bytes	bigint	语句在所有DN上的平均OBS IO读字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_read_bytes平均值

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_write_bytes	bigint	语句在所有DN上的平均OBS IO写字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_write_bytes平均值
calls	integer	默认为1。	聚合周期内，同聚合主键的查询数量。

17.3.261 PGXC_WLM_SESSION_HISTORY

PGXC_WLM_SESSION_HISTORY视图显示在所有CN上执行作业结束后的负载管理记录。此视图用于从DWS数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟。

说明

- PGXC_WLM_SESSION_HISTORY视图仅支持在**postgres**数据库中查询，其他数据库中查询会直接报错。
- 9.1.1.300及以上集群版本中系统会将执行成功且执行时间小于100ms，并且包含sql_hash的短查询进行聚合记录。9.1.2及以上集群版本中，子语句聚合不受100ms执行时间限制，满足其他聚合条件时默认进行聚合。聚合的主要列为datid、dbname、resource_pool、node_group、username、application_name、client_addr、sql_hash、plan_hash、query_band和nodename。

PGXC_WLM_SESSION_HISTORY视图的字段同**GS_WLM_SESSION_HISTORY**相同，具体字段内容如下：

表 17-359 GS_WLM_SESSION_HISTORY 字段

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
datid	oid	连接后端的数据库OID。	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。	连接后端的数据库名称。
schemaname	text	模式名。	聚合周期内，同聚合主键第一个语句的schemaname。
nodename	text	语句执行的CN名称。	语句执行的CN名称。
username	text	连接到后端的用户名。	连接到后端的用户名。
application_name	text	连接到后端的应用名。	连接到后端的应用名。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。	连接到后端的客户端的IP地址。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。	聚合周期内，同聚合主键第一个语句的客户端的client_hostname。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。	聚合周期内，同聚合主键第一个语句的client_port。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。	聚合周期内，同聚合主键第一个语句的query_band。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。	聚合周期内，同聚合主键所有语句的block_time平均值。
start_time	timestamp with time zone	语句执行的开始时间。	聚合周期内，同聚合主键第一个语句的start_time。
finish_time	timestamp with time zone	语句执行的结束时间。	聚合周期内，同聚合主键最后一个语句的finish_time。
duration	bigint	语句实际执行的时间，单位ms。	聚合周期内，同聚合主键所有语句的duration平均值。
estimate_total_time	bigint	语句预估执行时间，单位ms。	聚合周期内，同聚合主键所有语句的estimate_total_time平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。	语句执行结束状态，仅聚合执行成功的语句，默认为finished。
abort_info	text	语句执行结束状态为aborted时显示异常信息。	空。
resource_pool	text	用户使用的资源池。	用户使用的资源池。
control_group	text	语句所使用的Cgroup。	聚合周期内，同聚合主键第一个语句的control_group。
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。	聚合周期内，同聚合主键所有语句的estimate_memory平均值。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的min_peak_memory最小值。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。	聚合周期内，同聚合主键所有语句的max_peak_memory最大值。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。	聚合周期内，同聚合主键所有语句的average_peak_memory平均值。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。	聚合周期内，同聚合主键所有语句的memory_skew_percent的平均值。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。	空

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的min_spill_size的最小值。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的max_spill_size的最大值。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。	聚合周期内，同聚合主键所有语句的average_spill_size的平均值。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。	聚合周期内，同聚合主键所有语句的spill_skew_percent的平均值。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。	聚合周期内，同聚合主键所有语句的min_dn_time的最小值。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。	聚合周期内，同聚合主键所有语句的max_dn_time的最大值。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。	聚合周期内，同聚合主键所有语句的average_dn_time平均值。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。	聚合周期内，同聚合主键所有语句的dntime_skew_percent平均值。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的min_cpu_time最小值。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。	聚合周期内，同聚合主键所有语句的max_cpu_time最大值。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。	聚合周期内，同聚合主键所有语句的total_cpu_time总量。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。	聚合周期内，同聚合主键所有语句的cpu_skew_percent平均值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的min_peak_iops最小值。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的max_peak_iops最大值。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。	聚合周期内，同聚合主键所有语句的average_peak_iops平均值。
iops_skew_percent	integer	语句在DN间的IO倾斜率。	聚合周期内，同聚合主键所有语句的iops_skew_percent平均值。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin	聚合周期内，同聚合主键第一个语句的warning。
queryid	bigint	语句执行使用的内部query id。	聚合周期内，同聚合主键第一个语句的queryid。
query	text	执行的语句，最多可保留64KB长度的字符串。	聚合周期内，同聚合主键第一个语句的query。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。	聚合周期内，同聚合主键第一个语句的query_plan。
node_group	text	语句所属用户对应的逻辑集群。	语句所属用户对应的逻辑集群。
pid	bigint	语句的后端线程的pid。	聚合周期内，同聚合主键第一个语句的pid。
lane	text	语句执行时所在的快慢车道。	聚合周期内，同聚合主键第一个语句的lane。
unique_sql_id	bigint	归一化的Unique SQL ID。	聚合周期内，同聚合主键第一个语句的unique_sql_id。
session_id	text	在数据库系统中唯一标记一个session，格式：session_start_time.tid.node_name。	聚合周期内，同聚合主键第一个语句的session_id。
min_read_bytes	bigint	语句在所有DN上的最小IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_read_bytes最小值。
max_read_bytes	bigint	语句在所有DN上的最大IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_read_bytes最大值。
average_read_bytes	bigint	语句在所有DN上的平均IO读字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_read_bytes平均值。
min_write_bytes	bigint	语句在所有DN上的最小IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的min_write_bytes最小值。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
max_write_bytes	bigint	语句在所有DN上的最大IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的max_write_bytes最大值。
average_write_bytes	bigint	语句在所有DN上的平均IO写字节数，单位Bytes。	聚合周期内，同聚合主键所有语句的average_write_bytes平均值。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。	聚合周期内，同聚合主键所有语句的recv_pkg总量。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。	聚合周期内，同聚合主键所有语句的send_pkg总量。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的recv_bytes总量。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。	聚合周期内，同聚合主键所有语句的send_bytes总量。
stmt_type	text	语句对应的查询类型。	聚合周期内，同聚合主键第一个语句的stmt_type。
except_info	text	语句触发的异常规则信息。	空
unique_plan_id	bigint	归一化的Unique plan id。	聚合周期内，同聚合主键第一个语句的unique_plan_id。
sql_hash	text	归一化的sql hash。	归一化的sql hash。
plan_hash	text	归一化的plan hash。	归一化的plan hash。
use_plan_baseline	text	当前语句执行是否使用了绑定的计划。如果使用了，则显示pg_plan_baseline中的plan_baseline列的名字。	聚合周期内，同聚合主键第一个语句的use_plan_baseline。
outline_name	text	该语句计划对应的outline的名字。	聚合周期内，同聚合主键第一个语句的outline_name。

版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
loader_status	text	保存导入导出类业务信息的json串具体如下。 1. address：互联互通对端集群的地址，源集群会显示端口号。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：导入导出语句在所有DN上字节数的行数/最小值/最大值/总和。	保存导入导出类业务信息的json串具体如下。 1. address：聚合周期内，同聚合主键第一个语句的address。 2. direction：导入导出业务类型，取值包括gds to file、gds from file、gds to pipe、gds from pipe、copy from、copy to。 3. min/max/total_lines/bytes：聚合周期内，同聚合主键所有语句的行数/最小值/最大值/总和。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的parse_time平均值。
disk_cache_hit_ratio	numeric(5,2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_hit_ratio总量。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_read_size总量。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_disk_write_size总量。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_size总量。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_remote_read_time总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
vfs_scan_bytes	bigint	OBS虚拟文件系统接收到上层请求的扫描的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_scan_bytes总量。
vfs_remote_read_bytes	bigint	OBS虚拟文件系统实际从OBS读取的字节数，单位Bytes。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的vfs_remote_read_bytes总量。
preload_submit_time	bigint	预读流程提交IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_submit_time总量。
preload_wait_time	bigint	预读流程等待IO请求的总时间，单位μs。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_time总量。
preload_wait_count	bigint	预读流程等待IO请求的总次数。该字段仅对存算分离3.0表生效。	聚合周期内，同聚合主键所有语句的preload_wait_count总量。
disk_cache_load_time	bigint	读取磁盘缓存的总时间，单位μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_load_time总量。
disk_cache_conflict_count	bigint	读取磁盘缓存中block产生哈希冲突的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_conflict_count总量。
disk_cache_error_count	bigint	读取磁盘缓存失败的次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的disk_cache_error_count总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
disk_cache_error_code	bigint	读取磁盘缓存失败的错误码，可能产生多个错误码，读取磁盘缓存失败会发起OBS远程读并重写缓存块，错误码类型如下。该字段仅对3.0表及外表生效。 ● 1：磁盘缓存块产生哈希冲突。 ● 2：磁盘缓存块的生成时间晚于OldestXmin事务。 ● 4：磁盘缓存读取缓存文件调用pread系统调用失败。 ● 8：磁盘缓存块的数据版本不匹配。 ● 16：写缓存写入的数据版本与最新版本不匹配。 ● 32：打开缓存块对应的缓存文件失败。 ● 64：磁盘缓存读取数据大小不匹配。 ● 128：磁盘缓存块中记录的csn不匹配。	空
obs_io_req_avg_rtt	bigint	OBS IO请求的平均RTT(Round Trip Time，IO请求往返时间)，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_rtt总量。
obs_io_req_avg_latency	bigint	OBS IO请求的平均延迟，单位为μs。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_avg_latency总量。
obs_io_req_latency_gt_1s	bigint	OBS IO请求延迟超过1s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_1s总量。
obs_io_req_latency_gt_10s	bigint	OBS IO请求延迟超过10s的数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_latency_gt_10s总量。

 版权所有 © 华为云计算技术有限公司

名称	类型	100ms以上长查询记录含义说明	100ms以下短查询聚合记录含义说明
obs_io_req_count	bigint	OBS IO请求的总数量。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_count总量。
obs_io_req_retry_count	bigint	OBS IO请求重试的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_retry_count总量。
obs_io_req_rate_limit_count	bigint	OBS IO请求被流控的总次数。该字段仅对存算分离3.0表及外表生效。	聚合周期内，同聚合主键所有语句的obs_io_req_rate_limit_count总量。
gxid	xid	语句事务ID号。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键第一个语句的gxid。
inserted_rows	bigint	语句插入的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的inserted_rows平均值
deleted_rows	bigint	语句删除的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的deleted_rows平均值
updated_rows	bigint	语句更新的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的updated_rows平均值
returned_rows	bigint	语句查询的行数。该字段仅9.1.1.100及以上集群版本支持。	聚合周期内，同聚合主键所有语句的returned_rows平均值
obs_io_read_bytes	bigint	语句在所有DN上的平均OBS IO读字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_read_bytes平均值
obs_io_write_bytes	bigint	语句在所有DN上的平均OBS IO写字节数，单位Bytes。 该字段仅9.1.1.200及以上集群版本支持。	聚合周期内，同聚合主键所有语句的obs_io_write_bytes平均值
calls	integer	默认为1。	聚合周期内，同聚合主键的查询数量。

 版权所有 © 华为云计算技术有限公司

17.3.262 PGXC_WLM_SESSION_STATISTICS

PGXC_WLM_SESSION_STATISTICS视图显示在所有CN上正在执行的作业的负载管理信息。

表 17-360 PGXC_WLM_SESSION_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据OID。
dbname	name	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN节点名称。
username	name	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标识作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
pid	bigint	后端线程ID。
block_time	bigint	语句执行前的阻塞时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
duration	bigint	语句已经执行的时间，单位ms。
estimate_total_time	bigint	语句执行预估总时间，单位ms。
estimate_left_time	bigint	语句执行预估剩余时间，单位ms。
enqueue	text	工作负载管理资源状态。
resource_pool	name	用户使用的资源池。
control_group	text	语句所使用的Cgroup。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
estimate_memory	integer	语句在单个实例上预估使用的内存，单位MB。该字段只有当GUC参数enable_dynamic_workload为on时才有效。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句在各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All：所有DN均下盘。 [a:b]：数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在各DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
min_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最小IO读速率，单位KB/s。
max_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最大IO读速率，单位KB/s。
average_read_speed	integer	一个监控周期（5秒）内，语句在所有DN上的平均IO读速率，单位KB/s。
min_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最小IO写速率，单位KB/s。
max_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的最大IO写速率，单位KB/s。
average_write_speed	integer	一个监控周期（5秒）内，语句在所有DN上的平均IO写速率，单位KB/s。
recv_pkg	bigint	语句在所有DN上的通信包接收总量，单位packages。
send_pkg	bigint	语句在所有DN上的通信包发送总量，单位packages。
recv_bytes	bigint	语句在所有DN上的通信流接收数据总量，单位Byte。
send_bytes	bigint	语句在所有DN上的通信流发送数据总量，单位Byte。
warning	text	主要显示如下几类告警信息以及SQL自诊断调优相关告警： 1. The max spill size exceeds the alarm size 256MB 2. The max broadcast size exceeds the alarm size 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict 7. NestLoop in hashjoin
unique_sql_id	bigint	归一化的Unique SQL ID。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
queryid	bigint	语句执行使用的内部query id。
query	text	正在执行的语句。
query_plan	text	语句的执行计划。 规格限制： 1. DML语句都会显示执行计划，DDL语句不显示执行计划。 2. 当用户下发PBE(Parse Bind Execute)批处理语句时，为了便于分析语句情况，自8.2.1.100集群版本开始，为批处理的PBE语句的执行计划添加数据绑定次数，显示为“PBE bind times：次数”格式。
node_group	text	语句所属用户对应的逻辑集群。
stmt_type	text	语句所对应的查询类型。
except_info	text	语句触发的异常规则信息。
parse_time	bigint	语句排队前的解析总时间（包含词法语法解析，优化重写和计划生成时间），单位ms。该字段仅8.3.0.100及以上版本支持。
unique_plan_id	bigint	归一化的Unique plan id。
sql_hash	text	归一化的sql hash。
plan_hash	text	归一化的plan hash。
disk_cache_hit_ratio	numeric(5, 2)	磁盘缓存命中率 。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_read_size	bigint	读取磁盘缓存数据的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_disk_write_size	bigint	写入磁盘缓存的数据总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_size	bigint	读取磁盘缓存失败，远程直读OBS的总大小，单位MB。该字段仅对存算分离3.0表及外表生效。
disk_cache_remote_read_time	bigint	读取磁盘缓存失败，远程直读OBS的次数。该字段仅对存算分离3.0表及外表生效。
block_name	text	与语句匹配的对应拦截规则名
obs_io_read_bps	integer	一个监控周期（5秒）内，语句在所有DN上的平均OBS IO读速率，单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	integer	一个监控周期（5秒）内，语句在所有DN上的平均OBS IO写速率，单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
cudesc_streaming_count	integer	语句所使用的cudesc streaming个数。该字段仅9.1.1.300及以上集群版本支持。

17.3.263 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS

PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图展示当前库中表的数据分布倾斜情况。支持快速查看当前库中所有表在各节点的存储空间分布倾斜大小情况。该视图仅8.2.1及以上集群版本支持。

使用建议：

- 使用本视图查询指定表存储分布信息，需要具备指定表的SELECT权限。
- 该函数基于**PG_RELFILENODE_SIZE**系统表上的物理文件存储空间记录，GUC参数use_workload_manager和enable_perm_space需为开启状态（联系技术支持开启）。
- 在大集群大数据量业务场景下进行全库各表所占磁盘空间倾斜率分析时，PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图的查询性能优于gs_table_distribution()函数和PGXC_GET_TABLE_SKEWNESS视图。建议先使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览，再使用gs_table_distribution(schemaname text, tablename text)获取指定表在各个节点所占磁盘空间大小的分布情况。

表 17-361 PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS 字段

名称	类型	描述
schema_name	name	表所在的模式名称。
table_name	name	表名。
total_size	numeric	表在各个节点上的存储空间大小总和值，单位：字节。
avg_size	numeric(1000,0)	表在各个节点上的存储空间大小的平均值，单位：字节。
max_percent	numeric	表在各个节点上的存储空间的最大值占总和值的百分比(%)。
min_percent	numeric	表在各个节点上的存储空间的最小值占总和值的百分比(%)。
skew_percent	numeric	表的倾斜率信息(%)。 倾斜率skew_percent的计算方式：倾斜率(SKEW_PERCENT)= (最大值dnsize - 平均值dnsize) * 100/最大值dnsize

版权所有 © 华为云计算技术有限公司

应用示例

使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览，再使用gs_table_distribution(schemaname text, tablename text)函数获取指定表在各个节点所占磁盘空间大小的分布情况。

步骤1 使用PGXC_WLM_TABLE_DISTRIBUTION_SKEWNESS视图查询表的倾斜状况总览。

```
tpcds_col=# select * from pgxc_wlm_table_distribution_skewness;
```

查询结果如下：

```
tpcds_col=# select * from pgxc_wlm_table_distribution_skewness;
    schema_name     |                   table_name                   | total_size  |  avg_size  | max_percent | min_percent | skew_percent
--------------------+------------------------------------------------+-------------+------------+-------------+-------------+--------------
 pg_catalog         | pg_namespace_oid_index                         |       98304 |      16384 |       16.67 |       16.67 |         0.00
 public             | customer_demographics                          |     4595712 |    1531904 |       33.33 |       33.33 |         0.00
 pg_catalog         | pg_type_oid_index                              |      245760 |      40960 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_partition                                   |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_user_mapping_user_server_index              |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_attrdef                                     |      294912 |      49152 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_global_temp_parent_nsp_index                |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_namespace_nspname_index                     |       98304 |      16384 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_synonym_name_nsp_index                      |       49152 |       8192 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_proc_proname_args_nsp_index                 |     2015232 |     335872 |       16.67 |       16.67 |         0.00
 public             | dbgen_version                                  |       32768 |      10923 |      100.00 |        0.00 |        66.67
 pg_catalog         | pg_amop_opr_fam_index                          |      294912 |      49152 |       16.67 |       16.67 |         0.00
 pg_catalog         | pg_global_temp                                 |           0 |          0 |        0.00 |        0.00 |         0.00
 pg_catalog         | pg_workload_group_name_index                   |           0 |          0 |        0.00 |        0.00 |         0.00
 public             | date_dim                                       |     7159808 |    2386603 |       34.10 |       32.95 |         2.24
 pg_catalog         | pg_operator_oid_index                          |      245760 |      40960 |       16.67 |       16.67 |         0.00
```

显示表dbgen_version的数据倾斜程度较为严重。

步骤2 使用gs_table_distribution(schemaname text, tablename text)函数查询表dbgen_version在各个节点所占磁盘空间大小的分布情况。

```
tpcds_col=# select * from gs_table_distribution('public','dbgen_version');
```

查询结果如下：

```
tpcds_col=# select * from gs_table_distribution('public','dbgen_version');
 schemaname |   tablename    | relkind | relpersistence |   nodename    | dnsize | sessionid
------------+----------------+---------+----------------+---------------+--------+-----------
 public     | dbgen_version | r       | p              | dn_6001_6002 |      0 |
 public     | dbgen_version | r       | p              | dn_6005_6006 |  32768 |
 public     | dbgen_version | r       | p              | dn_6003_6004 |      0 |
(3 rows)
```

显示该表在DN上所占磁盘空间确实存在数据倾斜现象，数据集中在dn_6005_6006上。

----结束

17.3.264 PGXC_WLM_USER_RESOURCE_HISTORY

PGXC_WLM_USER_RESOURCE_HISTORY视图显示所有用户在对应实例上资源消耗的历史信息。该视图仅8.2.0及以上集群版本支持。

表 17-362 PGXC_WLM_USER_RESOURCE_HISTORY 字段

名称	类型	描述
nodename	name	实例名称，包含CN和DN。
username	text	用户名。
timestamp	timestamp with time zone	时间戳。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
used_memory	integer	正在使用的内存大小，单位MB。 DN：显示当前DN上对应用户正在使用的内存大小。 CN：显示所有DN上对应用户正在使用的内存累加和。
total_memory	integer	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 DN：显示当前DN上对应用户可以使用的内存大小。 CN：显示所有DN上对应用户可以使用的内存大小之和。
used_cpu	double precision	正在使用的CPU核数（仅统计非默认资源池上复杂作业的CPU使用情况，且该值为相关控制组的CPU使用统计值）。 DN：显示当前DN上对应用户正在使用的CPU核数。 CN：显示所有DN上对应用户正在使用的CPU核数之和。
total_cpu	integer	用户关联控制组的CPU核数总和。 DN：显示当前DN上对应用户能够使用的CPU核数。 CN：显示所有DN上对应用户能够使用的CPU核数之和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的永久表存储空间大小。 CN：显示所有DN上对应用户已使用的永久表存储空间大小之和。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。 DN：显示当前DN上对应用户可使用的永久表存储空间大小。 CN：显示所有DN上对应用户可使用的永久表存储空间大小之和。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。 DN：显示当前DN上对应用户已使用的临时表存储空间大小。 CN：显示所有DN上对应用户已使用的临时表存储空间大小之和。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。 DN：显示当前DN上对应用户可使用的临时表存储空间大小 CN：显示所有DN上对应用户可使用的临时表存储空间大小之和
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。 DN：显示当前DN上对应用户已使用的算子落盘空间大小。 CN：显示所有DN上对应用户已使用的算子落盘空间大小之和。
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。 DN：显示当前DN上对应用户可使用的算子落盘空间大小。 CN：显示所有DN上对应用户可使用的算子落盘空间大小之和。
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的字节总数，单位KB。 DN：实例启动至当前时间为止，该用户复杂作业read的字节总数，单位KB。
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的字节总数，单位KB。 DN：实例启动至当前时间为止，该用户复杂作业write的字节总数，单位KB。
read_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的次数之和。 DN：实例启动至当前时间为止，该用户复杂作业read的次数之和。
write_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的次数之和。 DN：实例启动至当前时间为止，该用户复杂作业write的次数之和。
read_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业read平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上复杂作业read平均速率。单位KB/s。
write_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业write平均速率。单位KB/s。 DN：过去5秒内，该用户在该DN上复杂作业write平均速率。单位KB/s。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
send_speed	double precision	CN：一个5秒监控周期内，该用户在所有DN上网络发送平均速率之和。单位KB/s。 DN：一个5秒监控周期内，该用户在该DN上网络发送平均速率。单位KB/s。
recv_speed	double precision	CN：一个5秒监控周期内，该用户在所有DN上网络接收平均速率之和。单位KB/s。 DN：一个5秒监控周期内，该用户在该DN上网络接收平均速率。单位KB/s。
obs_io_read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO读的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO读的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上OBS IO写的字节总数。单位KB。 DN：实例启动至当前时间为止，该用户OBS IO写的字节总数。单位KB。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_read_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO读平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO读平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。
obs_io_write_bps	double precision	CN：过去5秒内，该用户在所有DN上OBS IO写平均速率之和。单位KB/s。 DN：过去5秒内，该用户在该DN上OBS IO写平均速率。单位KB/s。 该字段仅9.1.1.200及以上集群版本支持。

17.3.265 PGXC_WLM_WORKLOAD_RECORDS

PGXC_WLM_WORKLOAD_RECORDS视图显示当前用户在每个CN上执行作业时在CN上的状态信息。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。该视图仅在动态负载功能开启，即enable_dynamic_workload为on时有效。

表 17-363 PGXC_WLM_WORKLOAD_RECORDS 字段

名称	类型	描述
node_name	text	作业执行所在的CN的名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
thread_id	bigint	后端线程ID。
processid	integer	线程的lwpid。
timestamp	bigint	语句执行的开始时间。
username	name	登录到该后端的用户名。
memory	integer	语句所需的内存大小。
active_points	integer	语句在资源池上消耗的资源点数。
max_points	integer	资源在资源池上的最大资源数。
priority	integer	作业的优先级。
resource_pool	text	作业所在资源池。
status	text	作业执行的状态，包括： pending：阻塞状态。 running：执行状态。 finished：结束状态。 aborted：终止状态。 unknown：未知状态。
control_group	text	作业所使用的Cgroups。
enqueue	text	作业的排队信息，包括： GLOBAL：全局排队。 RESPOOL：资源池排队。 ACTIVE：不排队。
query	text	正在执行的语句。

17.3.266 PGXC_WORKLOAD_SQL_COUNT

PGXC_WORKLOAD_SQL_COUNT视图显示集群中所有CN节点上的Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-364 PGXC_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。
select_count	bigint	SELECT数量。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
update_count	bigint	UPDATE数量。
insert_count	bigint	INSERT数量。
delete_count	bigint	DELETE数量。
ddl_count	bigint	DDL数量。
dml_count	bigint	DML数量。
dcl_count	bigint	DCL数量。

17.3.267 PGXC_WORKLOAD_SQL_ELAPSE_TIME

PGXC_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有CN节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。

表 17-365 PGXC_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。
total_select_elapse	bigint	SELECT总响应时间。
max_select_elapse	bigint	SELECT最大响应时间。
min_select_elapse	bigint	SELECT最小响应时间。
avg_select_elapse	bigint	SELECT平均响应时间。
total_update_elapse	bigint	UPDATE总响应时间。
max_update_elapse	bigint	UPDATE最大响应时间。
min_update_elapse	bigint	UPDATE最小响应时间。
avg_update_elapse	bigint	UPDATE平均响应时间。
total_insert_elapse	bigint	INSERT总响应时间。
max_insert_elapse	bigint	INSERT最大响应时间。
min_insert_elapse	bigint	INSERT最小响应时间。
avg_insert_elapse	bigint	INSERT平均响应时间。
total_delete_elapse	bigint	DELETE总响应时间。
max_delete_elapse	bigint	DELETE最大响应时间。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
min_delete_elapse	bigint	DELETE最小响应时间。
avg_delete_elapse	bigint	DELETE平均响应时间。

17.3.268 PGXC_WORKLOAD_TRANSACTION

PGXC_WORKLOAD_TRANSACTION视图提供查询所有CN上Workload控制组相关的事务信息。需要有系统管理员权限或预置角色gs_role_read_all_stats权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

表 17-366 PGXC_WORKLOAD_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称。
workload	name	Workload控制组名称。
commit_counter	bigint	提交次数。
rollback_counter	bigint	回滚次数。
resp_min	bigint	最小响应时间，单位微秒。
resp_max	bigint	最大响应时间，单位微秒。
resp_avg	bigint	平均响应时间，单位微秒。
resp_total	bigint	响应时间总和，单位微秒。

17.3.269 PLAN_TABLE

PLAN_TABLE视图显示用户通过执行EXPLAIN PLAN收集到的计划信息。计划信息的生命周期是session级别，session退出后相应的数据将被清除。同时不同session和不同user间的数据是相互隔离的。

表 17-367 PLAN_TABLE 字段

名称	类型	描述
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	查询生成的计划中的每一个执行算子的编号。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
operation	varchar2(30)	计划中算子的操作描述。
options	varchar2(255)	操作选项。
object_name	name	操作对应的对象名，非查询中使用到的对象别名。来自于用户定义。
object_type	varchar2(30)	对象类型。
object_owner	name	对象所属schema，来自于用户定义。
projection	varchar2(4000)	操作输出的列信息。

说明

- object_type取值范围为PG_CLASS中定义的relkind类型（TABLE普通表，INDEX索引，SEQUENCE序列，VIEW视图，FOREIGN TABLE外表，COMPOSITE TYPE复合类型，TOAST表）和计划使用到的rtekind(SUBQUERY, JOIN, FUNCTION, VALUES, CTE, REMOTE_QUERY)。
- object_owner对于RTE来说是计划中使用的对象描述，非用户定义的类型不存在object_owner。
- statement_id、object_name、object_owner、projection字段内容遵循用户定义的大小写存储，其他字段内容采用大写存储。
- 支持用户对PLAN_TABLE进行SELECT和DELETE操作，不支持其他DML操作。

17.3.270 PV_FILE_STAT

PV_FILE_STAT视图通过对数据文件IO的统计，反映数据的IO性能，用于发现IO操作异常等性能问题。

表 17-368 PV_FILE_STAT 字段

名称	类型	描述
filenum	oid	文件标识。
dbid	oid	数据库标识。
spcid	oid	表空间标识。
phyrds	bigint	读物理文件的数目。
phywrts	bigint	写物理文件的数目。
phyblkrd	bigint	读物理文件块的数目。
phyblkwrt	bigint	写物理文件块的数目。
readtim	bigint	读文件的总时长，单位微秒。
writetim	bigint	写文件的总时长，单位微秒。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
avgiotim	bigint	读写文件的平均时长，单位微秒。
lstiotim	bigint	最后一次读文件时长，单位微秒。
miniotim	bigint	读写文件的最小时长，单位微秒。
maxiowtm	bigint	读写文件的最大时长，单位微秒。

17.3.271 PV_INSTANCE_TIME

PV_INSTANCE_TIME视图用于统计进程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

提供当前节点下的各种时间消耗信息，主要分为以下类型：

- DB_TIME: 进程在多核并行计算下的时间开销。
- CPU_TIME：进程在CPU核心上的时间开销。
- EXECUTION_TIME：执行器内的时间开销。
- PARSE_TIME：SQL解析的时间开销。
- PLAN_TIME：生成Plan的时间开销。
- REWRITE_TIME：SQL重写的时间开销。
- PL_EXECUTION_TIME ： plpgsql（存储过程）的执行时间.
- PL_COMPILATION_TIME：plpgsql（存储过程）编译时间。
- NET_SEND_TIME：网络上的时间开销。
- DATA_IO_TIME：IO上的时间开销。

表 17-369 PV_INSTANCE_TIME 字段

名称	类型	描述
stat_id	integer	类型编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

17.3.272 PV_MATVIEW_DETAIL

PV_MATVIEW_DETAIL视图显示物化视图的具体信息。该视图仅9.1.0 200及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-370 PV_MATVIEW_DETAIL 字段

名称	类型	描述
matview	text	物化视图名。
baserel	text	基表名。
partids	oidvector	指定分区时候的分区OID。
contain_entire_rel	boolean	是否以基表的整表建物化视图。
build_mode	text	物化视图的build模式。 ● deferred，表示创建物化视图时需等到第一次REFRESH时才会包含数据。 ● immediate，表示创建物化视图时即包含最新数据。
refresh_mode	text	物化视图的刷新模式。 demand，表示按需刷新。
refresh_method	text	物化视图的刷新方法。 ● complete，表示完全刷新。 ● fast，表示增量刷新。
mapping	text	基表分区和物化视图分区的映射关系。
active	boolean	物化视图的数据状态。 ● true，表示物化视图数据与基表一致，可用于查询重写。 ● false，表示基表数据已变化但物化视图未刷新，数据过期，不可用于查询重写，需刷新后恢复。
rewrite_count	bigint	查询重写命中次数。该字段仅9.1.1.200及以上集群版本支持。
refresh_count	bigint	刷新次数。该字段仅9.1.1.200及以上集群版本支持。
last_refresh_start	timestamp with time zone	最后一次刷新的开始时间。
last_refresh_end	timestamp with time zone	最后一次刷新的结束时间。

17.3.273 PV_MATVIEW_REFRESH_HISTORY

PV_MATVIEW_REFRESH_HISTORY视图显示物化视图的刷新历史。该视图仅9.1.1 300及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-371 PV_MATVIEW_REFRESH_HISTORY 字段

名称	类型	描述
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
matview_name	text	物化视图名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
queryid	bigint	语句执行使用的内部query id。
query	text	刷新执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
matview_refresh_type	text	物化视图的刷新类型，全量刷新或者增量刷新。
inserted_rows	bigint	刷新插入行数。
deleted_rows	bigint	刷新删除行数。
updated_rows	bigint	刷新更新行数。
returned_rows	bigint	刷新返回行数。

 版权所有 © 华为云计算技术有限公司

17.3.274 PV_MATVIEW_REWRITE_HISTORY

PV_MATVIEW_REWRITE_HISTORY视图显示物化视图的查询重写历史信息。该视图仅9.1.1 300及以上集群版本支持。

表 17-372 PV_MATVIEW_REWRITE_HISTORY 字段

名称	类型	描述
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
queryid	bigint	语句执行使用的内部query id。
query	text	刷新执行的语句，最多可保留64KB长度的字符串。
query_plan	text	语句的执行计划。
status	text	语句执行结束状态：正常为finished，异常为aborted。该处记录的语句状态应为数据库服务端执行状态，当服务器端执行成功，结果集返回时报错，该语句应为finished。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
matview	text	查询重写的物化视图名。
rewrite_count	bigint	物化视图查询重写次数。

17.3.275 PV_OS_RUN_INFO

PV_OS_RUN_INFO视图显示当前操作系统运行的状态信息。

 版权所有 © 华为云计算技术有限公司

表 17-373 PV_OS_RUN_INFO 字段

名称	类型	描述
id	integer	编号。
name	text	操作系统运行状态名称。
value	numeric	操作系统运行状态值。
comments	text	操作系统运行状态注释。
cumulative	boolean	操作系统运行状态的值是否为累加值。

17.3.276 PV_REDO_STAT

PV_REDO_STAT视图提供当前节点上XLOG重做过程中的统计信息。

表 17-374 PV_REDO_STAT 字段

名称	类型	描述
phywrts	bigint	物理写次数。
phyblkwrt	bigint	物理写块数。
writetim	bigint	物理写消耗时间。
avgiotim	bigint	平均每次写入时间。
lstiotim	bigint	上一次写入时间。
miniotim	bigint	最小写入时间。
maxiowtm	bigint	最大写入时间。

17.3.277 PV_RUNTIME_ATTSTATS

PV_RUNTIME_ATTSTATS视图显示autoanalyze产生的内存中表级统计信息，各字段含义与**PG_STATS**视图一样。该视图仅8.2.0及以上集群使用。

表 17-375 PV_RUNTIME_ATTSTATS 字段

名称	类型	引用	描述
schemaname	name	**PG_NAMESPACE**.nspname	包含表的模式名。
tablename	name	**PG_CLASS**.relname	表名。
attname	name	**PG_ATTRIBUTE**.attname	字段名。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	● 如果大于0，表示字段中独立数值的估计数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
partname	name	-	分区名。该字段仅8.2.1.200及以上集群版本支持。

17.3.278 PV_RUNTIME_EXPRSTATS

PV_RUNTIME_EXPRSTATS视图显示autoanalyze产生的内存中表达式统计信息。该视图仅8.2.1.200及以上集群版本支持。

表 17-376 PV_RUNTIME_EXPRSTATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
statistics_schemaname	name	-	表达式统计信息对象所在模式名。
statistics_name	name	-	表达式统计信息对象的名称。
statistics_owner	name	-	表达式统计信息对象的所有者。
expr	text	-	表达式
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。

版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
n_distinct	real	-	● 如果大于0，表示字段中独立数值的估计数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。

 版权所有 © 华为云计算技术有限公司

名称	类型	引用	描述
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。
level	text	-	轻量化autoanalyze生成的内存统计信息当前所处的阶段。包含: local, sendlist, global三个阶段。

17.3.279 PV_RUNTIME_RELSTATS

PV_RUNTIME_RELSTATS视图显示autoanalyze产生的内存中表级统计信息，各字段含义与**PG_CLASS**视图一样。该视图仅8.2.0及以上集群使用。

表 17-377 PV_RUNTIME_RELSTATS 字段

名称	类型	描述
nspname	name	模式名。
relname	name	表、索引等对象的名称。
relpages	double precision	以页(大小为BLCKSZ)为单位的此表在磁盘上的大小，只是优化器使用的一个近似值。
reltuples	double precision	表中行的数目，只是优化器使用的一个估计值。
relallvisible	integer	被标识为全可见的表中的页数。此字段是优化器用来做SQL执行优化使用的。
relhasindex	boolean	如果对象是一个表且至少有（或者最近建有）一个索引，则为真。 由CREATE INDEX设置，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，会清理relhasindex字段，将relhasindex值设置为假。
changes	bigint	触发轻量化autoanalyze时，表的历史累计修改条数。
level	text	轻量化autoanalyze生成的内存统计信息当前所处的阶段。包含: local, sendlist, global三个阶段。
partname	name	分区名。该字段仅8.2.1.200及以上集群版本支持。

版权所有 © 华为云计算技术有限公司

17.3.280 PV_SESSION_MEMORY

PV_SESSION_MEMORY视图统计Session级别的内存使用情况，包含执行作业在数据节点上Postgres线程和Stream线程分配的所有内存。

表 17-378 PV_SESSION_MEMORY 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename） **说明** 8.2.1集群版本之前的sessid仅包含线程启动时间+线程标识，8.2.1及以上集群版本sessid新增了节点名称，格式为：timestamp.threadid.nodename
init_mem	integer	当前正在执行作业进入执行器前已分配的内存，单位MB。
used_mem	integer	当前正在执行作业已分配的内存，单位MB。
peak_mem	integer	当前正在执行作业已分配的内存峰值，单位MB。

17.3.281 PV_SESSION_MEMORY_DETAIL

PV_SESSION_MEMORY_DETAIL统计线程的内存使用情况，以MemoryContext节点来统计。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

可通过"select * from pv_session_memctx_detail(threadid, '');"将某个线程所有内存上下文信息记录到“/tmp/dumpmem”目录下的“threadid_timestamp.log”文件中。其中threadid可通过下表sessid中获得。

表 17-379 PV_SESSION_MEMORY_DETAIL 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）。
sesstype	text	线程名称。
contextname	text	内存上下文名称。
level	smallint	当前上下文在整体内存上下文中的层级。
parent	text	父内存上下文名称。
totalsize	bigint	当前内存上下文的内存总数，单位Byte。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
freesize	bigint	当前内存上下文中已释放的内存总数，单位Byte。
usedsize	bigint	当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。

应用示例

查询当前节点上所有MemoryContext的使用情况。

根据sessid定位到该MemoryContext是在哪个线程中创建和使用的，依据totalsize，freesize及usedsize来确认内存的使用情况是否符合预期，预先判断是否可能存在内存泄露的风险。

```
SELECT * FROM pv_session_memory_detail;
                sessid                |        sesstype        |                contextname                | level |
parent            | totalsize | freesize | usedsize
--------------------------------------+------------------------+-------------------------------------------+-------
+-------------------------------+-----------+----------+----------
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TempSmallContextGroup                 |     0 |                           |    646520 |   125184 |      149
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TopMemoryContext                      |     0 |                           |    372672 |    19488 |   353184
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Record information
cache                    |     1 | TopMemoryContext            |     24576 |    13632 |    10944
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Operator class
cache                    |     1 | TopMemoryContext            |     24576 |    15168 |     9408
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | smgr relation
table                    |     1 | TopMemoryContext            |     24576 |    11040 |    13536
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
TransactionAbortContext               |     1 | TopMemoryContext            |     32768 |    32704 |       64
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | Portal
hash                                  |     1 | TopMemoryContext            |     24576 |    16192 |     8384
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
CacheMemoryContext                    |     1 | TopMemoryContext            |   1858848 |    10640 |
1848208
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
InvalidCacheMemoryContext             |     2 | CacheMemoryContext          |   2088960 |   828928 |
1260032
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | WAL record
construction             |     1 | TopMemoryContext            |     49792 |     6528 |    43264
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | LOCAL LOCK
hash                                  |     1 | TopMemoryContext            |     24576 |    15936 |     8640
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector |
Timezones                             |     1 | TopMemoryContext            |     24576 |    15808 |     8768
 1760444600.140326303563096.coordinator1 | GlobalDeadLockDetector | SELF MEMORY
CONTEXT                               |     1 | TopMemoryContext            |     66600 |    17824 |    48776
 1760444600.140326303562480.coordinator1 | file_recyclebin        |
TempSmallContextGroup                 |     0 |                           |    642688 |   129040 |      147
 1760444600.140326303562480.coordinator1 | file_recyclebin        | TopMemoryContext
|     0 |                           |   1437632 |    14256 |  1423376
 1760444600.140326303562480.coordinator1 | file_recyclebin        | Operator class cache
|     1 | TopMemoryContext            |     24576 |    15168 |     9408
 1760444600.140326303562480.coordinator1 | file_recyclebin        | smgr relation table
|     1 | TopMemoryContext            |     24576 |    11040 |    13536
 1760444600.140326303562480.coordinator1 | file_recyclebin        | TransactionAbortContext
|     1 | TopMemoryContext            |     32768 |    32704 |       64
 1760444600.140326303562480.coordinator1 | file_recyclebin        | Portal hash                               |
1 | TopMemoryContext            |     24576 |    16192 |     8384
 1760444600.140326303562480.coordinator1 | file_recyclebin        |
```

 版权所有 © 华为云计算技术有限公司

```
CacheMemoryContext                |    1 | TopMemoryContext          |  1858848 |    12096 |
1846752
```

17.3.282 PV_SESSION_STAT

PV_SESSION_STAT视图以会话线程或AutoVacuum线程为单位，统计会话状态信息。

表 17-380 PV_SESSION_STAT 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）
statid	integer	统计编号。
statname	text	统计会话名称。
statunit	text	统计会话单位。
value	bigint	统计会话值。

17.3.283 PV_SESSION_TIME

PV_SESSION_TIME视图用于统计会话线程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

表 17-381 PV_SESSION_TIME 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识+节点名称（字符串信息为timestamp.threadid.nodename）
stat_id	integer	统计编号。
stat_name	text	运行时间类型名称。
value	bigint	运行时间值。

17.3.284 PV_TOTAL_MEMORY_DETAIL

PV_TOTAL_MEMORY_DETAIL视图统计当前数据库节点使用内存的信息，单位为MB。

表 17-382 PV_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
memorytype	text	内存类型，包括以下几种： ● max_process_memory：DWS集群实例所占用的内存大小。 ● process_used_memory：DWS进程所使用的内存大小。 ● process_peak_memory：DWS进程占用物理内存峰值。 ● shared_used_memory：已使用的共享内存。 ● shmem_mmap_used_memory：共享内存块大小 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_used_memory：已使用的动态内存。 ● comm_used_memory：通信库使用内存大小。 ● topsql_used_memory：TopSQL使用内存大小。 ● storage_compress_memory：列存压缩和解压缩使用的内存大小。 ● scheduler_used_memory：异步IO使用内存大小。 ● disk_cache_used_memory：disk cache使用内存大小。 ● mmap_used_memory：mmap使用的内存大小。 ● prof_used_memory：内存采集功能使用内存大小。 ● cstore_used_memory：列存已使用的内存大小。 ● gpu_dynamic_used_memory：当前GPU可用内存和当前临时GPU内存之和。 ● log_used_memory：日志使用内存大小。 ● jemalloc_metadata：jemalloc元数据使用内存大小。 ● jemalloc_metadata_thp：jemalloc透明大页使用内存大小。 ● total_stack_size：栈内存使用大小。 ● other_used_memory：其他已使用的内存大小。 ● max_shared_memory：最大共享内存。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_dynamic_memory：最大动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● max_shmem_map_memory：最大共享块内存大小。 ● max_cstore_memory：列存所允许使用的最大内存。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
		• max_comm_memory：通信库所允许使用的最大内存。 • comm_peak_memory：通信库内存峰值。 • max_topsql_memory：TopSQL所允许使用的最大内存。 • topsql_peak_memory：TopSQL内存峰值。 • gpu_max_dynamic_memory：GPU内存最大值。 • gpu_dynamic_peak_memory：GPU内存使用的峰值。 • pooler_conn_memory：pooler连接占用内存大小。 • pooler_freeconn_memory：pooler空闲连接占用的内存大小。 • udf_reserved_memory：为UDF Worker进程预留的内存大小。 • scheduler_peak_memory：异步IO内存峰值。 • disk_cache_peak_memory：disk cache内存峰值。 • log_peak_memory：日志内存峰值。 • jemalloc_peak_size：jemalloc分配内存峰值。 • jemalloc_user_active：jemalloc活跃内存值。 • jemalloc_user_allocated：jemalloc分配内存值。 • single_stack_max_size：线程最大栈内存值。
memorymbytes	integer	内存类型分配内存的大小。

17.3.285 REDACTION_COLUMNS

REDACTION_COLUMNS视图展示当前数据库内所有脱敏列信息。

表 17-383 REDACTION_COLUMNS 字段

名称	类型	描述
object_schema	name	脱敏对象的schema。
object_owner	name	脱敏对象的owner。
object_name	name	脱敏对象名称。
column_name	name	脱敏列名称。
function_type	integer	脱敏类型。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
function_parameters	text	脱敏类型为partial类型时的参数（保留字段，无实际意义）。
regexp_pattern	text	脱敏类型为regexp时，pattern串（保留字段，无实际意义）。
regexp_replace_string	text	脱敏类型为regexp时，替换串（保留字段，无实际意义）。
regexp_position	integer	脱敏类型为regexp时，起始替换位置（保留字段，无实际意义）。
regexp_occurrence	integer	脱敏类型为regexp时，替换次数（保留字段，无实际意义）。
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数（保留字段，无实际意义）。
function_info	text	脱敏函数信息。
column_description	text	脱敏列描述信息。
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。
policy_name	name	所属的脱敏策略名称。该字段仅8.2.1.100及以上集群版本支持。

17.3.286 REDACTION_POLICIES

REDACTION_POLICIES视图展示当前数据库内所有脱敏对象信息。

表 17-384 REDACTION_POLICIES 字段

名称	类型	描述
object_schema	name	脱敏对象的schema。
object_owner	name	脱敏对象的owner。
object_name	name	脱敏对象名称。
policy_name	name	脱敏策略名称。
expression	text	策略生效表达式（针对用户）。
enable	boolean	策略状态（开启、关闭）。
policy_description	text	策略描述信息。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
inherited	bool	说明脱敏列是否是“继承”自其他脱敏列。

17.3.287 REMOTE_TABLE_STAT

REMOTE_TABLE_STAT视图提供集群所有DN节点上当前数据库所有表的统计信息。除在每一行前面增加name类型的nodename字段外，其余字段的名称、类型和顺序与GS_TABLE_STAT视图相同。

表 17-385 REMOTE_TABLE_STAT 字段

名称	类型	描述
nodename	name	节点名称。
schemaname	name	表的命名空间。
relname	name	表的名称。
seq_scan	bigint	顺序扫描的次数。只统计行存表。如果是分区表，显示各个分区扫描次数的和。
seq_tuple_read	bigint	顺序扫描的行数。只统计行存表。
index_scan	bigint	索引扫描的次数。只统计行存表。
index_tuple_read	bigint	索引扫描的行数。只统计行存表。
tuple_inserted	bigint	插入的行数。
tuple_updated	bigint	更新的行数。
tuple_deleted	bigint	删除的行数。
tuple_hot_updated	bigint	热更新的行数。
live_tuples	bigint	活元组数量。CN上查询该视图，analyze后显示该表格总的活元组数量，未analyze的情况下显示0。只适用行存表。
dead_tuples	bigint	死元组数量。CN上查询该视图，analyze后显示该表格总的死元组数量，未analyze的情况下显示0。只适用行存表。

17.3.288 SHOW_TSC_INFO

查询当前节点TSC信息。该视图仅8.2.1及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

表 17-386 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

17.3.289 SHOW_ALL_TSC_INFO

查询所有节点TSC信息。该视图仅8.2.1及以上集群版本支持。

表 17-387 返回值字段

名称	类型	描述
node_name	text	节点名称。
tsc_mult	bigint	TSC换算乘数。
tsc_shift	bigint	TSC换算位移数。
tsc_frequency	float8	TSC频率。
tsc_use_frequency	boolean	是否使用TSC频率进行时间换算。
tsc_ready	boolean	是否可以使用TSC频率进行时间换算。
tsc_scalar_error_info	text	获取TSC换算信息的错误信息。
tsc_freq_error_info	text	获取TSC频率的错误信息。

17.3.290 SQL_PLAN_BASELINE

SQL_PLAN_BASELINE视图显示计划的绑定信息，此视图信息来源于系统表**PG_PLAN_BASELINE**。

版权所有 © 华为云计算技术有限公司

表 17-388 PG_PLAN_BASELINE 字段

名称	类型	描述
sql_hash	text	语句的sql_hash。
outline_name	text	该sql_hash对应的sql语句绑定的outline的名称。
plan_baseline	name	该绑定关系的baseline名称。
create_time	timestamp with time zone	计划绑定创建时间。
last_modify_time	timestamp with time zone	最后一次修改时间，即为该语句最后一次绑定计划的时间。
last_execute_time	timestamp with time zone	最后一次被使用时间。
apply_skew_hint	boolean	是否应用outline中的skew hint。该字段9.1.0及以上集群版本支持。
status	tinyint	绑定状态。该字段9.1.1.200及以上集群版本支持。取值如下： ● 0：preferred，表示优先。 ● 1：approved，表示可用。 ● 2：unapproved，表示不推荐。 ● 3：rejected，表示拒绝。
cost	float8	生成该outline的计划的成本（cost）。该字段9.1.1.200及以上集群版本支持。
normalize_cost	float8	根据TopSQL中该语句duration、cpu和memory使用情况，经过归一化计算的成本（cost）。保留列，暂未使用。该字段9.1.1.200及以上集群版本支持。

17.3.291 USER_COL_COMMENTS

USER_COL_COMMENTS视图存储当前用户下表和视图的列注释信息。

表 17-389 DBA_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名。
table_name	character varying(64)	表名或视图名。
owner	character varying(64)	表或视图的所有者。
comments	text	注释。

 版权所有 © 华为云计算技术有限公司

17.3.292 USER_CONSTRAINTS

USER_CONSTRAINTS视图存储当前用户下表中约束的信息。

表 17-390 DBA_CONSTRAINTS 字段

名称	类型	描述
constraint_name	character varying(64)	约束名。
constraint_type	text	约束类型。 ● c表示检查约束。 ● f表示外键约束。 ● p表示主键约束。 ● u表示唯一约束。
table_name	character varying(64)	约束相关的表名。
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）。
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）。

应用示例

查询当前用户下指定表的约束信息。t1替换为实际的表名。

```
SELECT * FROM USER_CONSTRAINTS WHERE table_name='t1';
 constraint_name | constraint_type | table_name | index_owner |  index_name
-----------------+-----------------+------------+-------------+---------------
 c_custkey_key   | p               | t1         | u1          | c_custkey_key
(1 row)
```

17.3.293 USER_CONS_COLUMNS

USER_CONS_COLUMNS视图存储当前用户下表中约束列的信息。

表 17-391 DBA_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名。
column_name	character varying(64)	约束相关的列名。
constraint_name	character varying(64)	约束名。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
position	smallint	表中列的位置。

17.3.294 USER_INDEXES

USER_INDEXES视图存储关于本模式下的索引信息。

表 17-392 DBA_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_name	character varying(64)	索引对应的表名。
uniqueness	text	表示索引是否为唯一索引。
generated	character varying(1)	表示索引名称是否为系统生成。
partitioned	character(3)	表示索引是否具有分区表的性质。

17.3.295 USER_IND_COLUMNS

USER_IND_COLUMNS视图存储当前用户下所有索引的字段信息。

表 17-393 DBA_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_name	name	列名。
column_position	smallint	索引中列的位置。

17.3.296 USER_IND_EXPRESSIONS

USER_IND_EXPRESSIONS视图存储当前用户下基于函数的表达式索引的信息。

 版权所有 © 华为云计算技术有限公司

表 17-394 DBA_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者。
index_name	character varying(64)	索引名。
table_owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_expression	text	定义列的基于函数的索引表达式。
column_position	smallint	索引中列的位置。

17.3.297 USER_IND_PARTITIONS

USER_IND_PARTITIONS视图存储当前用户下的索引分区信息。

表 17-395 DBA_IND_PARTITIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称。
schema	character varying(64)	索引分区所属分区表索引的模式。
index_name	character varying(64)	索引分区所属分区表索引的名称。
partition_name	character varying(64)	索引分区的名称。
index_partition_usable	boolean	索引分区是否可用。
high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	索引分区所对应的表分区的边界（范围分区为上边界，列表分区为边界值集合）。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collation、字段数据类型等信息。
def_tablespace_name	name	索引分区的表空间名称。

版权所有 © 华为云计算技术有限公司

17.3.298 USER_JOBS

USER_JOBS视图为当前用户所属定时任务的详细信息。需要有系统管理员权限才可以访问此系统视图。

表 17-396 USER_JOBS 字段

名字	类型	描述
job	bigint	作业ID。
log_user	name not null	创建者的UserName。
priv_user	name not null	作业执行者的UserName。
dbname	name not null	作业创建数据库名字。
start_date	timestamp without time zone	作业的开始时间。
start_suc	text	作业成功执行的开始时间。
last_date	timestamp without time zone	上次运行开始时间。
last_suc	text	上次成功运行的开始时间。
this_date	timestamp without time zone	正在运行任务的开始时间。
this_suc	text	正在运行任务成功的开始时间。
next_date	timestamp without time zone	任务下次执行时间。
next_suc	text	任务下次成功执行时间。
broken	text	任务状态 如果为Y，不尝试运行此任务。 如果为N，将尝试执行此任务。
status	text	当前任务的执行状态，取值范围：('r', 's', 'f', 'd')，默认为'r'，取值含义如下： ● r=running ● s=successfully finished ● f= job failed ● d=aborted
interval	text	用来计算下次运行时间的时间表达式，如果为null，则表示定时任务只执行一次。

 版权所有 © 华为云计算技术有限公司

名字	类型	描述
failures	smallint	连续失败计数。
what	text	可执行的作业。

17.3.299 USER_OBJECTS

USER_OBJECTS视图描述了当前用户拥有的数据库对象。

表 17-397 DBA_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者。
object_name	name	对象的名称。
object_id	oid	对象的OID。
object_type	name	对象的类型。
namespace	oid	对象所在的命名空间。
created	timestamp with time zone	对象的创建时间。
last_ddl_time	timestamp with time zone	对象的最后修改时间。

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

17.3.300 USER_PART_INDEXES

USER_PART_INDEXES视图存储当前用户下分区表索引的信息。

表 17-398 DBA_PART_INDEXES 字段

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称。
schema	character varying(64)	分区表索引的模式。
index_name	character varying(64)	分区表索引的名称。

版权所有 © 华为云计算技术有限公司

名称	类型	描述
table_name	character varying(64)	分区表索引所属的分区表名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表索引的索引分区的个数。
def_tablespace_name	name	分区表索引的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

17.3.301 USER_PART_TABLES

USER_PART_TABLES视图存储当前用户下分区表的信息。

表 17-399 DBA_PART_TABLES 字段

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称。
schema	character varying(64)	分区表的模式。
table_name	character varying(64)	分区表的名称。
partitioning_type	text	分区表的分区策略。 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）和列表分区（List Partitioning）。
partition_count	bigint	分区表的分区个数。
def_tablespace_name	name	分区表的表空间名称。
partitioning_key_count	integer	分区表的分区键个数。

17.3.302 USER_PROCEDURES

USER_PROCEDURES视图存储关于本模式下的存储过程或函数信息。

 版权所有 © 华为云计算技术有限公司

表 17-400 DBA_PROCEDURES 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
object_name	character varying(64)	存储过程或函数名称。
argument_number	smallint	存储过程入参个数。

17.3.303 USER_SEQUENCES

USER_SEQUENCES视图存储关于本模式下的序列信息。

表 17-401 DBA_SEQUENCES 字段

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者。
sequence_name	character varying(64)	序列的名称。

17.3.304 USER_SOURCE

USER_SOURCE视图存储关于本模式下的存储过程或函数信息，且提供存储过程或函数定义的字段。

表 17-402 DBA_SOURCE 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者。
name	character varying(64)	存储过程或函数的名称。
text	text	存储过程或函数的定义。

17.3.305 USER_SYNONYMS

USER_SYNONYMS视图存储当前用户可访问的同义词信息。

表 17-403 USER_SYNONYMS 字段

名称	类型	描述
schema_name	text	同义词所属模式名。
synonym_name	text	同义词的名称。

 版权所有 © 华为云计算技术有限公司

名称	类型	描述
table_owner	text	关联对象的所有者。
table_schema_name	text	关联对象所属模式名。
table_name	text	关联对象名。

17.3.306 USER_TAB_COLUMNS

USER_TAB_COLUMNS视图存储当前用户可访问的表和视图字段信息。

表 17-404 DBA_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表名或视图名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	创建表或视图时列的序号。
data_length	integer	列的字节长度。
comments	text	注释。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2， bpchar，char类型有效。
schema	character varying(64)	包含该表或视图的命名空间。
kind	text	当前记录所属的种类，如果此列属于表，则此字段显示为table；如果此列属于视图，则此字段显示为view。

版权所有 © 华为云计算技术有限公司

17.3.307 USER_TAB_COMMENTS

USER_TAB_COMMENTS视图存储当前用户所有表和视图的注释信息。

表 17-405 DBA_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者。
table_name	character varying(64)	表或视图的名称。
comments	text	注释。

17.3.308 USER_TAB_PARTITIONS

USER_TAB_PARTITIONS视图存储当前用户下所有分区的信息。当前用户下每个分区表的每个分区在USER_TAB_PARTITIONS中都会有一条记录。

表 17-406 DBA_TAB_PARTITIONS 字段

名称	类型	描述
table_owner	character varying(64)	分区所在表的所有者。
schema	character varying(64)	分区表模式。
table_name	character varying(64)	表名。
partition_name	character varying(64)	分区的名称。
high_value	text	范围分区的上边界，或列表分区的边界值集合。 前向兼容的保留字段，8.1.3集群版本新增pretty_high_value用于记录此信息。
pretty_high_value	text	范围分区的上边界，或列表分区的边界值集合。 查询结果为表分区对应边界表达式的即时反编译输出。该字段的输出比high_value的信息更详细，根据实际使用场景可输出collation、字段数据类型等信息。
tablespace_name	name	分区所在表空间的名称。

17.3.309 USER_TABLES

USER_TABLES视图存储关于当前模式下的表信息。

 版权所有 © 华为云计算技术有限公司

表 17-407 DBA_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
tablespace_name	character varying(64)	表所在的表空间的名称。
status	character varying(8)	当前记录是否有效。
temporary	character(1)	是否为临时表。 ● Y表示是临时表。 ● N表示不是临时表。
dropped	character varying	当前记录是否已删除。 ● YES表示已删除。 ● NO表示未删除。
num_rows	numeric	表的估计行数。

17.3.310 USER_TRIGGERS

USER_TRIGGERS视图存储关于当前用户下的触发器信息。

表 17-408 DBA_TRIGGERS 字段

名称	类型	描述
trigger_name	character varying(64)	触发器名称。
table_name	character varying(64)	定义触发器的表的名称。
table_owner	character varying(64)	定义触发器的表的所有者。

17.3.311 USER_VIEWS

USER_VIEWS视图存储关于当前模式下的所有视图信息。

表 17-409 DBA_VIEWS 字段

名称	类型	描述
owner	character varying(64)	视图的所有者。
view_name	character varying(64)	视图的名称。

 版权所有 © 华为云计算技术有限公司

17.3.312 V$SESSION

V$SESSION视图存储当前会话的所有会话信息。

表 17-410 V$SESSION 字段

名称	类型	描述
sid	bigint	当前活动的后台进程的OID。
serial#	integer	当前活动的后台进程的序号，在DWS中为0。
user#	oid	登录此后台进程的用户的OID。
username	name	登录此后台进程的用户名。

17.3.313 V$SESSION_LONGOPS

V$SESSION_LONGOPS视图存储当前正在执行的操作的进度。

表 17-411 V$SESSION_LONGOPS 字段

名称	类型	描述
sid	bigint	当前正在执行的后台进程的OID。
serial#	integer	当前正在执行的后台进程的序号，在DWS中为0。
sofar	integer	目前完成的工作量，在DWS中为空。
totalwork	integer	工作总量，在DWS中为空。

版权所有 © 华为云计算技术有限公司

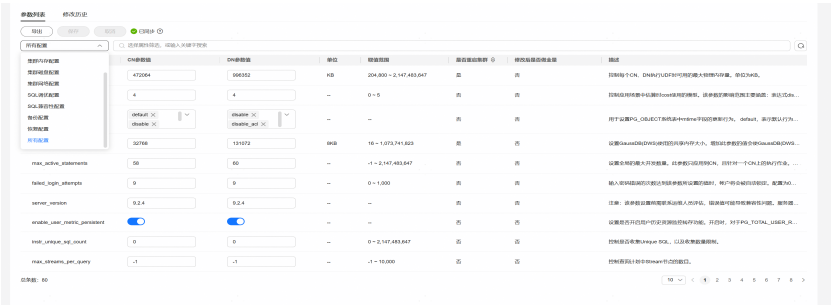
18 DWS 数据库 GUC 参数

18.1 查看 GUC 参数

DWS的GUC参数影响数据库的系统行为，用户可根据业务场景和数据量查看并调整GUC参数取值。

- 查看GUC参数方式一：集群创建成功后，用户可在**DWS控制台**上查看常用的数据库参数。

  图 18-1 查看 GUC 参数

- 查看GUC参数方式二：成功连接集群后，通过SQL命令的方式查看数据库GUC参数。
  - 使用SHOW命令。

    说明

    方式二只能查CN的GUC参数值，DN的GUC参数值可通过方式一：通过管理控制台查看。

    使用如下命令查看单个参数：

    ```
    SHOW server_version;
    ```

    server_version显示数据库版本信息的参数。

    使用如下命令查看所有参数：

    ```
    SHOW ALL;
    ```

  - 使用pg_settings视图。

    使用如下命令查看单个参数：

    ```
    SELECT * FROM pg_settings WHERE NAME='server_version';
    ```

 版权所有 © 华为云计算技术有限公司

使用如下命令查看所有参数：

```
SELECT * FROM pg_settings;
```

18.2 设置 GUC 参数

为确保DWS的最优性能，用户可根据业务需求对数据库中的GUC参数进行调整。

参数类型和值

- DWS的GUC参数类型分为以下五类：
  - SUSET，数据库管理员参数。设置后立即生效，无需重启集群。若在当前会话中设置该类型参数仅当前会话生效。
  - USERSET，普通用户参数。设置后立即生效，无需重启集群。若在当前会话中设置该类型参数仅当前会话生效。
  - POSTMASTER，数据库服务端参数。设置后需要重启集群才能生效，确认修改后系统会提示集群状态为待重启，建议在非业务高峰期手动重启集群，使参数生效。
  - SIGHUP，数据库全局参数。设置后全局生效，无法会话级生效。
  - BACKEND，数据库全局参数。设置后全局生效，无法会话级生效。
- 所有的参数名称不区分大小写。 参数取值有整型、浮点型、字符串、布尔型和枚举型五类。
  - 布尔值可以是（on，off）、（true，false）、（yes，no）或者（1，0），且不区分大小写。
  - 枚举类型的取值由系统表pg_settings的enumvals字段取值所定义。
- 对于有单位的参数，在设置时请指定单位，否则将使用默认的单位。
  - 参数的默认单位由系统表pg_settings的unit字段所定义。
  - 内存单位有：KB（千字节）、MB（兆字节）和GB（吉字节）。
  - 时间单位：ms（毫秒）、s（秒）、min（分钟）、h（小时）和d（天）。

GUC 参数设置

GUC参数设置有两种方式：

- 方式一：集群创建成功后，用户可以登录**DWS控制台**，根据实际需要修改集群的数据库参数。 具体操作请参见**修改DWS集群GUC参数**。
- 方式二：成功连接集群后，通过SQL命令的方式设置SUSET或USERSET类型的参数。

  修改指定数据库，用户，会话级别的参数。
  - 设置数据库级别的参数

    ```
    ALTER DATABASE dbname SET parameter_name TO value;
    ```

    在下次会话中生效。
  - 设置用户级别的参数

    ```
    ALTER USER username SET parameter_name TO value;
    ```

    在下次会话中生效。
  - 设置会话级别的参数

    ```
    SET parameter_name TO value;
    ```

    修改本次会话中的取值。退出会话后，设置将失效。

 版权所有 © 华为云计算技术有限公司

操作步骤

设置参数，以设置explain_perf_mode参数为例。

步骤1 查看explain_perf_mode参数。

```
SHOW explain_perf_mode;
 explain_perf_mode
-------------------
 normal
(1 row)
```

步骤2 设置explain_perf_mode参数。

使用以下任意方式进行设置：

- 设置数据库级别的参数
  ```
  ALTER DATABASE gaussdb SET explain_perf_mode TO pretty;
  ```
  当结果显示为如下信息，则表示设置成功。
  ```
  ALTER DATABASE
  ```
  在下次会话中生效。
- 设置用户级别的参数
  ```
  ALTER USER dbadmin SET explain_perf_mode TO pretty;
  ```
  当结果显示为如下信息，则表示设置成功。
  ```
  ALTER USER
  ```
  在下次会话中生效。
- 设置会话级别的参数
  ```
  SET explain_perf_mode TO pretty;
  ```
  当结果显示为如下信息，则表示设置成功。
  ```
  SET
  ```

步骤3 检查参数设置的正确性。

```
SHOW explain_perf_mode;
 explain_perf_mode
--------------
 pretty
(1 row)
```

----结束

18.3 GUC 使用说明

数据库提供了许多运行参数，配置这些参数可以影响数据库系统的行为。在修改这些参数时请确保已了解对应参数对数据库的影响，否则可能会导致无法预料的结果。

注意事项

- 参数中如果取值范围为字符串，此字符串应遵循操作系统的路径和文件名命名规则。
- 取值范围最大值为INT_MAX的参数，此选项最大值跟所在的操作系统有关。
- 取值范围最大值为DBL_MAX的参数，此选项最大值跟所在的操作系统有关。

18.4 连接和认证

 版权所有 © 华为云计算技术有限公司

18.4.1 连接设置

介绍设置客户端和服务器连接方式相关的参数。

max_connections

参数说明：允许和数据库连接的最大并发连接数。此参数会影响集群的并发能力。

参数类型：POSTMASTER

取值范围：整型

- CN最小值为100，最大值为16384。
- DN最小值为100， 最大值为262143。由于集群内部存在着各种连接，设置时通常达不到最大值，若日志中出现'invalid value for parameter "max_connections"'，需要调小DN的max_connections值。

默认值：CN节点为800，DN节点为5000，如果该默认值超过内核支持的最大值（在执行gs_initdb的时候判断），系统会提示错误。

设置建议：

- CN中此参数建议保持默认值。
- DN中此参数按照如下公式计算：

  dop_limit * 20 * 6 + 24，公式中的dop_limit为集群中每个DN对应的CPU数，计算公式为：dop_limit = 单机器的CPU逻辑核数 / 单机器的DN数。

增大这个参数可能导致DWS要求更多的SystemV共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。

⚠ 注意

max_connections取值的设置受**max_prepared_transactions**的影响，在设置max_connections之前，应确保max_prepared_transactions的值大于或等于max_connections的值，这样可确保每个会话都有一个等待中的预备事务。

application_name

参数说明：连接数据库的客户端程序名称。

参数类型：USERSET

取值范围：字符串。

默认值：gsql

connection_info

参数说明：连接数据库的驱动类型、驱动版本号、当前驱动的部署路径和进程属主用户。（运维类参数，不建议用户设置）

参数类型：USERSET

取值范围：字符串

版权所有 © 华为云计算技术有限公司

默认值：空字符串

说明

- 空字符串，表示当前连接数据库的驱动不支持自动设置connection_info参数或应用程序未设置。
- 驱动连接数据库的时候自行拼接的connection_info参数格式如下：
  ```
  {"driver_name":"ODBC","driver_version": "(GaussDB x.x.x build 39137c2d) compiled at 2022-09-23 15:43:11 commit 3629 last mr 5138 debug","driver_path":"/usr/local/lib/psqlodbcw.so","os_user":"omm"}
  ```
  ODBC，JDBC，gsql连接默认显示driver_name和driver_version，driver_path，os_user，其他接口连接默认显示driver_name和driver_version，driver_path和os_user的显示由用户控制（参见**连接数据库**和**Linux下配置数据源**）。

18.4.2 安全和认证（postgresql.conf）

介绍设置客户端和服务器的安全认证方式的相关参数。

session_timeout

参数说明：表明与服务器建立连接后，不进行任何操作的最长时间。

参数类型：USERSET

取值范围：整型，0-86400，最小单位为秒（s），0表示关闭超时设置。

默认值：10min

注意

- 由于gsql客户端具有自动重连机制，初始化用户本地连接超时后会断开并自动重连。
- pooler连接池与其他CN、DN之间的连接不受session_timeout参数控制。

ssl_renegotiation_limit

参数说明：指定在会话密钥重新协商之前，通过SSL加密通道可以传输的流量。这个重新协商流量限制机制可以减少攻击者针对大量数据使用密码分析法破解密钥的几率，但是也带来较大的性能损失。流量是指发送和接收的流量总和。

参数类型：USERSET

取值范围：整型，0~INT_MAX，单位为KB。0表示禁用重新协商机制。

默认值：0

注意

参数建议保持默认设置，即禁用重协商机制。不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置ssl_renegotiation_limit参数，即使设置也不会生效。

版权所有 © 华为云计算技术有限公司

failed_login_attempts

参数说明：设置允许用户连续登录失败的次数，当账户输入密码错误次数达到该参数的设置值，则当前账户将被锁定，password_lock_time秒后被自动解锁。

参数类型：SIGHUP

取值范围：整型，0~1000

- 0，表示自动锁定功能不生效。
- 正整数，表示当错误密码次数达到failed_login_attempts设定的值时，当前账户将被锁定。

默认值：10

注意

- failed_login_attempts和password_lock_time（该参数默认为1天，需联系技术支持设置）必须都为正数时锁定和解锁功能才能生效。
- failed_login_attempts会与客户端SSL连接模式共同决定用户的密码错误次数。当PGSSLMODE取值是allow或prefer时，客户的一次密码连接请求会生成两次连接请求：一次是尝试SSL连接，另一次是尝试非SSL连接。此时，用户感知到的密码错误次数是failed_login_attempts除以2。

kae_options

参数说明：设置是否开启KAE引擎加解密加速功能，仅9.1.1.200集群版本支持。

该参数9.1.1.300及以上集群版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。

参数类型：USERSET

取值范围：字符串

- none，表示不开启KAE引擎加解密加速功能。
- encryption，表示开启KAE引擎加解密加速功能。

默认值：encryption

18.4.3 通信库参数

本节介绍通信库相关的参数设置及取值范围等内容。

tcp_keepalives_idle

参数说明：在支持TCP_KEEPIDLE套接字选项的系统上，设置发送活跃信号的间隔秒数。不设置发送保持活跃信号，连接就会处于闲置状态。

参数类型：USERSET

取值范围：整型，0~3600，单位为秒（s）。

默认值：0

版权所有 © 华为云计算技术有限公司

⚠ 注意

- 如果操作系统不支持TCP_KEEPIDLE选项 ，该参数值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，该参数将被忽略。

tcp_keepalives_interval

参数说明：在支持TCP_KEEPINTVL套接字选项的操作系统上，以秒数声明在重新传输之间等待响应的时间。

参数类型：USERSET

取值范围：整型，0~180，单位为秒（s）。

默认值：0

⚠ 注意

- 如果操作系统不支持TCP_KEEPINTVL选项，该参数值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，该参数将被忽略。

tcp_keepalives_count

参数说明：在支持TCP_KEEPCNT套接字选项的操作系统上，设置DWS服务端在断开与客户端连接之前可以等待的保持活跃信号个数。

参数类型：USERSET

取值范围：整型，0~100，其中0表示DWS未收到客户端反馈的保持活跃信号则立即断开连接。

默认值：0

⚠ 注意

- 如果操作系统不支持TCP_KEEPCNT选项，该参数值必须为0。
- 在通过Unix域套接字进行连接的操作系统上，该参数将被忽略。

comm_max_datanode

参数说明：通信库支持的最大DN数。

参数类型：USERSET

取值范围：整型，1~8192

默认值：实际DN数

 版权所有 © 华为云计算技术有限公司

⚠ 注意

该参数值的DN数由小调大时立即生效，由大调小时需重启集群生效。

comm_max_stream

参数说明：通信库缓存的最大逻辑连接数据结构量。

参数类型：SIGHUP

取值范围：整型，1~65535

默认值：1024

说明

如果**comm_max_datanode**参数值较小，进程内存充足，可以适当将comm_max_stream值调大。

max_stream_pool

参数说明：设置stream线程池能够容纳stream线程的最大个数。该参数8.1.2及以上版本支持。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，-1和0表示关闭stream线程池。

默认值：

- 新安装集群默认值计算公式为：max_stream_pool=MIN(max_connections, max_process_memory/16/5MB, 1024)
- 旧集群升级到8.3.0.100及以上新集群版本的默认值计算公式为：max_stream_pool=MIN(max_connections, max_process_memory/16/5MB, 1024，旧集群值)，升级时也强制应用新装集群的设置值，当旧值小时取旧值。

说明

- 支持实时减少线程池中stream线程的个数；若增大该参数值，stream线程的个数由业务驱动向上增长。
- 通常情况下不建议调整该参数值，因为stream线程池有自动清理功能。
- 如果stream空闲线程过多造成内存占用，可以调小此值以节约内存。

enable_stream_sync_quit

参数说明：设置stream计划结束时，stream线程是否同步退出。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示stream计划结束后，等待stream线程组内的线程统一退出。
- off表示stream计划结束后，stream线程直接退出，无需等待stream线程组内的线程统一退出。

版权所有 © 华为云计算技术有限公司

默认值：off

enable_connect_standby

参数说明：设置CN连接DN备机。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示CN连接备机。
- off表示CN连接DN主机。

默认值：off

⚠ 注意

- 不建议在日常业务中使用此参数，仅适用于运维操作，更不建议使用gs_guc工具进行全局设置。否则可能会导致数据不一致，结果集错误等问题。
- 已创建临时表的session打开此参数后将造成DN上的临时表数据丢失，后续无法进行临时表相关操作。

comm_quota_size

参数说明：通信库最大可连续发送包总大小。使用1GE网卡时，建议取较小值，推荐设置为20KB~40KB。

参数类型：USERSET

取值范围：整型，0~102400，默认单位为KB。0表示不使用quota机制。

默认值：1MB

comm_usable_memory

参数说明：单个DN内通信库缓存最大可使用内存。

参数类型：SIGHUP

取值范围：整型，102400~1073741823，单位为KB。安装时最小值不得小于1GB。

默认值：max_process_memory/8

⚠ 注意

此参数需根据环境内存及部署方式具体配置，过大会造成OOM，过小会降低通信库性能。

comm_client_bind

参数说明：通信库客户端发起连接时是否使用bind绑定指定IP。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示绑定指定IP。
- off表示不绑定指定IP。

默认值：off

注意

如果集群某一节点存在多个IP处于同一通信网段时，需设置为on。此时将绑定本地listen_addresses指定的IP发起通信，随机端口号不能重复使用，集群并发数量会受到可用随机端口号数量的限制。

comm_no_delay

参数说明：是否使用通信库连接的NO_DELAY属性，重启集群生效。

参数类型：USERSET

取值范围：布尔型

默认值：off

注意

如果集群出现因每秒接收数据包过多导致的丢包时，需设置为off，以便小包合并成大包发送，减少数据包总数。

comm_debug_mode

参数说明：通信库debug模式开关，该参数设置是否打印通信层详细日志，session级别生效。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细debug日志。
- off表示不打印通信库详细debug日志。

默认值：off

注意

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

comm_ackchk_time

参数说明：无数据包接收情况下，该参数设置通信库服务端主动ACK触发时长。

版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，0~20000，单位为毫秒（ms）。取值为0表示关闭此功能。

默认值：2000

comm_timer_mode

参数说明：通信库timer模式开关，该参数设置是否打印通信层各阶段时间桩，session级别生效。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细时间桩日志。
- off表示不打印通信库详细时间桩日志。

默认值：off

注意

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

comm_stat_mode

参数说明：通信库stat模式开关，该参数设置是否打印通信层的统计信息，session级别生效。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库统计信息日志。
- off表示不打印通信库统计信息日志。

默认值：off

注意

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开，打开后及时关闭。

client_connection_check_interval

参数说明：客户端连接状态检测时间间隔。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。0表示不检测客户端连接状态。

默认值：10000

 版权所有 © 华为云计算技术有限公司

⚠ 注意

通过gsql/jdbc/odbc等客户端直连CN执行长查询，在长查询执行期间：

- CN每隔client_connection_check_interval时间检测一次客户端连接状态，若检测到客户端与CN的连接已经断开，则服务端主动终止长查询的执行，释放相关资源，避免集群资源浪费。
- DN每隔client_connection_check_interval时间检测一次CN与DN的连接状态，若检测到CN与DN的连接已经断开，则DN主动终止长查询的执行，释放相关资源，避免集群资源浪费。

conn_recycle_timeout

参数说明：用于控制将CN和其他节点间空闲连接回收到连接池的时间间隔。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0 ~ 3600，单位为秒(s)。0表示关闭空闲连接回收功能。

默认值：30

18.5 资源消耗

18.5.1 内存

介绍与内存相关的参数设置。

须知

本节涉及的参数仅在数据库服务重新启动后生效。

max_process_memory

参数说明：设置一个数据库节点可用的最大物理内存。

参数类型：SIGHUP

取值范围：整型，2*1024*1024～INT_MAX/2，单位为KB。

默认值：非从备DN节点自动适配，一个机器部署多个DN情况下，公式为（物理内存大小）* 0.8 / (1+主DN个数)；一个机器部署单个DN情况下，公式为（物理内存大小）* 0.6；当结果不足2GB时，默认取2GB。从备DN默认为12GB。

设置建议：

⚠ 注意

单机部署的场景仅适用于存量用户。

版权所有 © 华为云计算技术有限公司

- DN上该数值需要根据系统物理内存及单节点部署主DN个数决定。

  一个机器部署多个DN情况下，max_process_memory计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.8 / (n+主DN个数)；一个机器部署单个DN情况下，max_process_memory计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.6。

  该参数目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min_free_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即，max_process_memory=物理内存*0.8/(n+主DN个数)，其中，当集群规模小于256时， n=1；当集群规模大于256且小于512时， n=2；当集群规模超过512时，n=3。
- 不推荐设置为最小阈值。
- CN上该数值内存可设置与DN数值一样。
- RAM：集群规划时分配给集群的最大使用内存。
- DWS从8.2.0版本开始，为了提升内存资源利用率，增大了单机器单DN部署形态的max_process_memory初始值，但当出现集群状态不均衡的情况下，两个主DN节点会同时在一个机器上，如果仍然使用max_process_memory初始值，机器可能出现OOM报错的情况。为此，8.2.0版本开始，max_process_memory参数更改为SIGHUP类型，可以通过手动设置的方式来动态调整；同时，新增max_process_memory_auto_adjust参数，当出现集群状态不均衡的情况下，CM会根据集群状态来动态调整max_process_memory，调整max_process_memory的计算公式为：（物理内存大小 – vm.min_free_kbytes）* 0.8 / 主DN个数。
- DWS从8.2.1版本开始，扩大了max_process_memory动态内存调整的适用范围，由原先仅支持一个机器部署单个DN的集群形态，变更为支持所有部署形态。
  - max_process_memory_auto_adjust为on时，max_process_memory将会在上限和下限两值之间动态调整；其中，max_process_memory下限值的计算公式为：（物理内存大小）* 0.8 / (1+主DN个数)；max_process_memory上限值由GUC参数max_process_memory_balanced设置（max_process_memory_balanced参数设置请联系技术支持工程师）。
  - 集群在均衡模式下，max_process_memory将使用上限值，提高节点整体的内存资源利用率，相对之前版本，内存利用率会提升。
  - 集群在非均衡模式下，max_process_memory将使用下限值，节点整体的内存资源利用率和8.2.1集群之前的版本一致。
  - 升级场景下，该参数为保持前向兼容，系统不会设置max_process_memory_balanced，max_process_memory默认使用升级前设置的值。

max_process_memory_auto_adjust

参数说明：设置是否开启max_process_memory参数的自动调整功能（该参数仅8.2.0及以上集群版本支持）。

参数类型：SIGHUP

取值范围：布尔型

默认值：on

 版权所有 © 华为云计算技术有限公司

shared_buffers

参数说明：设置DWS使用的共享内存大小。增加此参数值会使DWS比系统默认设置需要更多的System V共享内存。

参数类型：POSTMASTER

取值范围：整型，16～1073741823，单位为8KB。

改变BLCKSZ的值会改变最小值。

默认值：CN节点为DN节点值的1/2，DN节点取公式计算：POWER(2,ROUND(LOG(2,max_process_memory/18),0))。如果操作系统支持的共享内存小于32MB，则在初始化数据存储区时会自动调整为操作系统支持的最大值。

设置建议：

由于DWS大部分查询下推，建议DN中此参数设置比CN大。

建议设置shared_buffers值为内存的40%以内。行存列存分开对待。行存设大，列存设小。列存：(单服务器内存/单服务器DN个数)*0.4*0.25。

如果设置较大的shared_buffers需要同时增加checkpoint_segments的值，因为写入大量新增、修改数据需要消耗更多的时间周期。

bulk_write_ring_size

参数说明：数据并行导入使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，16384～INT_MAX，单位为KB。

默认值：2GB

设置建议：建议导入压力大的场景中增加DN中此参数配置。

buffer_ring_ratio

参数说明：设置并行导出时使用环形缓冲区的阈值大小。

参数类型：USERSET

取值范围：整型，1～1000

默认值：250

说明

- 默认值表示阈值为shared_buffers的250/1000即1/4。
- 最小为shared_buffers的1/1000。
- 最大为shared_buffers的大小。

设置建议：导出时出现缓存命中率不符合预期的场景建议在DN中设置此参数。

enable_cstore_ring_buffer

参数说明：设置列存RingBuffer的开关，该参数仅8.2.0及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

默认值：off

设置建议：建议在业务运行一段时间，当CStoreBuffer内已存入大量高频查询的数据后，进行低频大表查询前打开，查询完大表后关闭。

temp_buffers

参数说明：设置每个数据库会话使用的LOCAL临时缓冲区的大小。

参数类型：USERSET

取值范围：整型，100~INT_MAX/2，单位为8KB。

默认值：8MB

说明

- 在每个会话的第一次使用临时表之前可以改变temp_buffers的值，之后的设置将是无效的。
- 一个会话将按照temp_buffers给出的限制，根据需要分配临时缓冲区。如果在一个并不需要大量临时缓冲区的会话里设置一个大的数值，其开销只是一个缓冲区描述符的大小。当缓冲区被使用，就会额外消耗8192字节。

max_prepared_transactions

参数说明：设置可以同时处于“预备”状态的事务的最大数目。增加此参数的值会使DWS比系统默认设置需要更多的System V共享内存。

参数类型：POSTMASTER

取值范围：整型，0~536870911。CN取值为0表示关闭预备事务的特性。

默认值：800

注意

- 当DWS部署为主备双机时，必须确保备机上此参数的设置值大于或等于主机上的设置值，否则无法在备机上进行查询操作。
- 为避免在准备步骤失败，此参数的值不建议小于**max_connections**。

work_mem

参数说明：设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。ORDER BY，DISTINCT和merge joins都要用到排序操作。Hash表在散列连接、散列为基础的聚集、散列为基础的IN子查询处理中都要用到。

对于复杂的查询，可能会同时并发运行好几个排序或者散列操作，每个都可以使用此参数所声明的内存量，不足时会使用临时文件。同样，好几个正在运行的会话可能会同时进行排序操作。因此使用的总内存可能是work_mem的好几倍。

参数类型：USERSET

取值范围：整型，64~INT_MAX，单位为KB。

 版权所有 © 华为云计算技术有限公司

默认值：小规格内存为512MB， 大规格内存为2GB（**max_process_memory**大于等于30GB为大规格内存，否则为小规格内存）。

设置建议：

依据查询特点和并发来确定，一旦work_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。

- 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work_mem=50%内存/10。
- 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work_mem=50%内存/5。
- 对于并发场景，建议work_mem=串行下的work_mem/物理并发数。

须知

开启内存自适应后，收集统计信息后不再需要使用work_mem进行算子内存使用调优，由系统根据当前负载情况，为每个语句生成计划，并估算每个算子的内存使用量和整个语句的内存使用量。系统根据负载情况和整个语句内存使用量进行队列调度，所以多并发场景会出现语句排队的情况。

query_mem

参数说明：设置执行作业所使用的内存。如果设置的query_mem值大于0，在生成执行计划时，优化器会将作业的估算内存调整为该值。

参数类型：USERSET

取值范围：整型，0，或大于32MB的整型，默认单位为KB。如果设置值小于32MB，系统会自动将该参数设置为默认值0，此时优化器不会根据该值调整作业的估算内存。

默认值：0

query_max_mem

参数说明：设置执行作业所能够使用的最大内存。如果设置的query_max_mem值大于0，在生成执行计划时，优化器会根据该值来设置算子的可用内存。当作业执行时所使用内存超过该值时，将报错退出。

参数类型：USERSET

取值范围：整型，0，或大于32MB的整型，单位为KB。如果设置值为小于32MB，系统会自动将该参数设置为默认值0，此时优化器不会根据该值限制作业的内存使用。

默认值：0

agg_max_mem

参数说明：设置执行作业中的Agg算子的聚集列超过5列时，该Agg算子所能够使用的最大内存。当agg_max_mem大于0时生效。（该参数仅8.1.3.200及以上集群版本支持）

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：整型，0，或大于32MB的整型，单位为KB。如果设置值小于32MB，系统会自动将该参数设置为默认值0，此时不会根据该值限制Agg算子的内存使用。

默认值：

- 若当前集群为低版本升级到8.1.3及以上版本，继承升级前参数，默认值为INT_MAX。
- 若当前集群为新装的8.1.3及以上版本，默认值为2GB。

enable_rowagg_memory_control

参数说明：控制行存agg算子内存使用上限。

参数类型：USERSET

取值范围：布尔型

- on表示开启控制行存agg算子内存使用限制。设置为on可避免行存agg内存过度使用导致系统内存不可用，但可能导致agg性能劣化。
- off表示关闭控制行存agg算子内存使用限制。设置为off可能导致系统内存不可用。

默认值：on

maintenance_work_mem

参数说明：设置在维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。该参数的设置会影响VACUUM、VACUUM FULL、CLUSTER、CREATE INDEX的执行效率。

参数类型：USERSET

取值范围：整型，1024~INT_MAX，单位为KB。

默认值：小规格内存为512MB， 大规格内存为2GB（**max_process_memory**大于等于30GB为大规格内存，否则为小规格内存）。

设置建议：

- 建议设置此参数的值等于**work_mem**，可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。
- 当**自动清理**进程运行时，**autovacuum_max_workers**倍数的内存将会被分配，所以此时设置maintenance_work_mem的值应该不小于**work_mem**。
- 如果进行大数据量的cluster等，可以在session中调大该值。

psort_work_mem

参数说明：设置列存表局部排序时，在开始写入临时磁盘文件之前使用的内存大小。该参数适用于带partial cluster key的表插入、带索引的表插入，创建表索引、删除表以及更新表等操作。

参数类型：USERSET

取值范围：整型，64~INT_MAX，单位为KB。

默认值：512MB

 版权所有 © 华为云计算技术有限公司

⚠ 注意

多个正在运行的会话可能会同时进行表的局部排序操作，因此使用的总内存可能是psort_work_mem的好几倍。

max_loaded_cudesc

参数说明：设置列存表在做扫描时，每列缓存cudesc信息的个数。增大设置会提高查询性能，但也会增加内存占用，特别是当列存表的列非常多时。

参数类型：USERSET

取值范围：整型，100~INT_MAX/2

默认值：1024

⚠ 注意

max_loaded_cudesc设置过高时，有可能引起内存分配不足。

max_stack_depth

参数说明：设置DWS执行堆栈的最大安全深度。需要这个安全界限是因为在服务器里，并非所有程序都检查了堆栈深度，只是在可能递归的过程，比如表达式计算这样的过程里面才进行检查。

参数类型：SUSET

设置原则：

- 此参数的最佳设置是等于操作系统内核允许的最大值（就是ulimit -s的设置）。
- 如果设置此参数的值大于实际的内核限制，则一个正在运行的递归函数可能会导致一个独立的服务器进程崩溃。在DWS能够检测内核限制的操作系统上（SLES上），将自动限制设置为一个不安全的值。
- 因为并非所有的操作都能够检测，所以建议用户在此设置一个明确的值。

取值范围：整型，100~INT_MAX，单位为KB。

默认值：2MB

说明

默认值2MB，这个值相对比较小，不容易导致系统崩溃。但是可能会因为该值较小，导致无法执行复杂的函数。

cstore_buffers

参数说明：设置列存和OBS、HDFS外表列存格式（orc、parquet、carbondata）所使用的共享缓冲区的大小。

参数类型：POSTMASTER

取值范围：整型，16384～INT_MAX，单位为KB。

版权所有 © 华为云计算技术有限公司

默认值：CN为32MB，DN取公式计算：
POWER(2,ROUND(LOG(2,max_process_memory/18),0))

设置建议：

列存表使用cstore_buffers设置的共享缓冲区，几乎不用**shared_buffers**。因此在列存表为主的场景中，应减少shared_buffers，增加cstore_buffers。

OBS、HDFS外表使用cstore_buffers设置ORC、Parquet、Carbondata的元数据和数据的缓存，元数据缓存大小为cstore_buffers的1/4，最大不超过2GB，其余缓存空间为列存数据和外表列存格式数据共享使用。

dfs_max_memory

参数说明：设置orc导出时能占用的最大内存。如果导出大宽表时出现内存不足问题，可以调大此参数后重试。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，131072 ~ 10485760，单位为KB。

默认值：256MB

schedule_splits_threshold

参数说明：设置HDFS外表schedule阶段能够在内存中存储的最大文件个数，超过该限制时，将把文件列表下盘处理。

参数类型：USERSET

取值范围：整型，1~INT_MAX

默认值：60000

bulk_read_ring_size

参数说明：并行导出，使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，256~INT_MAX，单位为KB。

默认值：16MB

check_cu_size_threshold

参数说明：列存表插入数据时，如果一个CU中已插入的数据量大于该参数时，开始进行行级大小校验，避免生成非压缩态大于1G的CU。

参数类型：USERSET

取值范围：整型，0~1048576，单位为KB。

默认值：1GB

 版权所有 © 华为云计算技术有限公司

须知

在进行行级大小校验时，如果多次校验不通过，建议先会话级设置该参数为0进行规避。

memory_spread_strategy

参数说明：调整自定义资源池DN内存扩展的策略。建议在内存充裕的业务场景进行调整，该参数设置后会提升查询性能，内存最高可以和默认资源池保持一致，但若业务场景中内存较小的情况下可能出现内存不足报错。该参数仅8.1.3及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none表示不扩展内存
- negative表示按需扩展内存，算子按需扩展，估算的多扩展多，估算的小扩展的小。
- crazy表示直接扩展内存，等同于默认资源池的内存扩展策略，但在业务场景内存较小的情况下可能出现内存不足报错。

默认值：none

enable_global_syscache

参数说明：用于控制是否开启Global SysCache特性。环境中SysCache会消耗大量内存，当现网中SysCache机制出现严重问题无法使用其他手段规避时，打开此开关将SysCache关闭，解决SysCache占用内存高的问题。该参数仅9.1.1.100及以上集群版本支持。

参数类型：POSTMASTER

取值范围：布尔型

- on表示打开Global SysCache特性。
- off表示关闭Global SysCache特性。

默认值：on

说明

新安装集群场景下，该参数默认值为on。升级场景下为保持前向兼容，该参数与升级前集群中该参数的默认值保持一致。

min_batch_rows

参数说明：设置内存攒批的最小行数，如果当前攒批达到该参数设置值就立即写入磁盘，否则先下盘继续攒批。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1000~60000

默认值：10000

 版权所有 © 华为云计算技术有限公司

min_pck_batch_rows

参数说明：用于设置V3 hash分区表pck攒批的行数限制。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，60000~600000

默认值：600000

receive_cudesc_max_memory

参数说明：设置cudesc streaming在CU扫描过程中允许接收cudesc的最大内存。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，4096~1073741823，单位为KB。

默认值：65536

18.5.2 语句磁盘空间管控

介绍与语句磁盘空间管控相关的参数，用于限制语句磁盘空间使用。

sql_use_spacelimit

参数说明：限制单个SQL在单个DN上，触发写盘操作时，所有类型写盘文件的总空间大小，管控的空间包括普通表、临时表以及中间结果集落盘占用的空间。系统管理员用户也受该参数限制。

参数类型：USERSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

默认值：配置sql_use_spacelimit为实例所在磁盘空间总容量的10%。

说明

例如，执行语句中配置参数sql_use_spacelimit=100，当出现单DN写盘超过100kB时，DWS会主动终止该query的运行，并提示用户单DN写盘量超阈值。

```
insert into user1.t1 select * from user2.t1;
ERROR:  The space used on DN (104 kB) has exceeded the sql use space limit (100 kB).
```

建议处理方式：

- 优化语句，减少语句写盘占用空间。
- 如果磁盘空间充足可以适当调大该参数。

temp_file_limit

参数说明：语句执行过程中触发落盘操作时，限制语句中单个线程落盘文件的总空间大小。例如，排序和哈希表使用的临时文件或者游标占用的临时文件。

此设置为会话级别的落盘文件控制。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

版权所有 © 华为云计算技术有限公司

默认值：配置temp_file_limit为实例所在磁盘空间总容量的10%。

须知

SQL查询执行时使用的临时表空间不在此限制。

min_spill_mem

参数说明：控制当前执行语句触发落盘的内存下限，即当前算子需要大于该参数的设置值才能触发落盘。该参数适用于列存agg、join算子，和行存的hash算子。仅9.1.1.300及以上集群版本支持。

参数类型：USERSET

取值范围：整型，64~INT_MAX，单位为KB。

默认值：32MB

说明

新安装的9.1.1.300及以上集群版本中该参数的默认值为32MB。升级场景下该参数的默认值为1MB。

bi_page_reuse_factor

参数说明：行存表批量插入场景下，主备DN使用页复制进行数据同步时，可以复用的旧页面空闲空间的百分比。

参数类型：USERSET

取值范围：整型，0~100，单位为%。其中0表示不对页面进行复用，全部申请新页面。

默认值：70

注意

- 不建议将此值设置为50以下（0除外），如果复用页面的空闲空间较小的情况下，会使主备DN间传输过多的旧页面数据，从而导致批量插入性能下降。
- 不建议将此值设置为90以上，如果此值设置过高，会导致频繁查询空闲页面，但又无法复用旧页面，得不偿失。

18.5.3 内核资源使用

介绍与操作系统内核相关的参数，这些参数是否生效依赖于操作系统的设置。

max_files_per_node

参数说明：限制单个节点上单个SQL打开的文件最大数量。通常情况下，不需要设置此参数。

参数类型：SUSET

版权所有 © 华为云计算技术有限公司

取值范围：整型，-1~INT_MAX，其中-1表示限制最大数量。

默认值：50000

说明

- 新安装9.1.0及以上集群版本中该参数的默认值为50000。
- 升级场景下若原集群已支持max_files_per_node参数，该参数的默认值为保持前向兼容维持原值。
- 升级场景下若原集群未支持max_files_per_node参数，升级后集群中该参数的默认值为-1。
- 执行语句报错“The last file name is [%s] and %d files have already been opened on data node [%s] with a maximum of %d files.”时，请尝试增大max_files_per_node值。

18.5.4 基于开销的清理延迟

基于开销的清理延迟的目的是允许管理员减少VACUUM和ANALYZE语句在并发活动的数据库上的I/O影响。 例如，VACUUM和ANALYZE此类维护语句并不需要迅速完成，且运行时要求此类语句不能严重干扰系统执行其他的数据库操作，管理员就可以使用该功能实现此目的。

须知

有些清理操作会持有关键的锁，这些操作应该尽快结束并释放锁。所以DWS的机制是，在这类操作过程中，基于开销的清理延迟不会发生作用。为了避免在这种情况下的长延时，实际的开销限制取下面两者之间的较大值：

- vacuum_cost_delay*accumulated_balance/vacuum_cost_limit
- vacuum_cost_delay*4

在ANALYZE | ANALYSE和VACUUM语句执行过程中，系统维护一个内部的记数器，跟踪所执行的各种I/O操作的近似开销。如果积累的开销达到了vacuum_cost_limit声明的限制，则执行这个操作的进程将睡眠vacuum_cost_delay指定的时间。然后它会重置记数器然后继续执行。

这个特性是缺省关闭的。要想打开它，把vacuum_cost_delay变量设置为一个非零值。

vacuum_cost_delay

参数说明：指定开销超过vacuum_cost_limit的值时，进程睡眠的时间。

参数类型：USERSET

取值范围：整型，0~100，单位为毫秒（ ms ）。正数值表示打开基于开销的清理延迟特性；0表示关闭基于开销的清理延迟特性。

默认值：0

须知

- 许多系统上，睡眠的有效分辨率是10毫秒。因此把vacuum_cost_delay设置为一个不是10的整数倍的数值与将它设置为下一个10的整数倍作用相同。
- 此参数一般设置较小，常见的设置是10或20毫秒。调整此特性资源占用率时，建议是调整其他参数，而不是该参数。

 版权所有 © 华为云计算技术有限公司

vacuum_cost_page_hit

参数说明：清理一个在共享缓存里找到的缓冲区的预计开销。它代表锁住缓冲池、查找共享的Hash表、扫描页面内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ ms ）。

默认值：1

vacuum_cost_page_miss

参数说明：清理一个要从磁盘上读取的缓冲区的预计开销。它代表锁住缓冲池、查找共享Hash表、从磁盘读取需要的数据块、扫描它的内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ ms ）。

默认值：2

vacuum_cost_page_dirty

参数说明：清理修改一个原先是干净的块的预计开销。它代表把一个脏的磁盘块再次刷新到磁盘上的额外开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ ms ）。

默认值：20

vacuum_cost_limit

参数说明：导致清理进程休眠的开销限制。

参数类型：USERSET

取值范围：整型，1~10000，单位为毫秒（ ms ）。

默认值：200

18.5.5 异步 IO

enable_adio_debug

参数说明：设置是否开启ADIO相关的日志，便于定位ADIO相关问题。运维类参数不建议普通用户设置。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此日志开关。
- off/false表示关闭此日志开关。

默认值：off

 版权所有 © 华为云计算技术有限公司

enable_fast_allocate

参数说明：磁盘空间快速分配开关。只有在XFS文件系统上才能开启该开关。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此功能。
- off/false表示关闭此功能。

默认值：off

prefetch_quantity

参数说明：描述行存储使用ADIO预读取IO量的大小。

参数类型：USERSET

取值范围：整型，128～131072，单位为8KB。

默认值：32MB

backwrite_quantity

参数说明：描述行存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，128～131072，单位为8KB。

默认值：8MB

cstore_prefetch_quantity

参数说明：描述列存储使用ADIO预取IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：32MB

cstore_backwrite_quantity

参数说明：描述列存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

cstore_backwrite_max_threshold

参数说明：描述列存储使用ADIO写入数据库可缓存最大的IO量。

参数类型：USERSET

取值范围：整型，4096～INT_MAX/2，单位为KB。

 版权所有 © 华为云计算技术有限公司

默认值：2GB

fast_extend_file_size

参数说明：描述列存储使用ADIO预扩展磁盘的大小。

参数类型：SUSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

effective_io_concurrency

参数说明：磁盘子系统可以同时有效处理的请求数。对于RAID阵列，此参数应该是阵列中驱动器主轴的数量。

参数类型：USERSET

取值范围：整型，0~1000

默认值：1

disk_cache_dual_write_option

参数说明：控制是否开启缓存双写。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none，表示关闭缓存双写功能。
- all，表示普通v3表以及hstore opt v3表merge时开启缓存双写功能。
- hstore_only，表示只有hstore opt v3表merge时才开启缓存双写功能。

默认值：hstore_only

cu_preload_max_distance

参数说明：用于控制预读CU Group个数的上限。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型， 0~1024，其中0表示关闭预读功能。

默认值：20

cu_preload_count

参数说明：设置预读CU个数的最大值。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型。0~10000，其中0表示关闭预读功能。

默认值：600

 版权所有 © 华为云计算技术有限公司

18.5.6 磁盘缓存

以下参数仅9.1.0及以上集群版本支持。

enable_disk_cache

参数说明：控制是否打开文件缓存。该参数受enable_aio_scheduler和obs_worker_pool_size的前置控制；只有在enable_aio_scheduler=on以及obs_worker_pool_size>=4的前提下，此参数为on才会生效。

参数类型：USERSET

取值范围：布尔型

默认值： off

18.6 并行导入

DWS提供了并行导入功能，以快速、高效地完成大量数据导入。介绍DWS并行导入的相关参数。

raise_errors_if_no_files

参数说明：导入时是否区分“导入文件记录数为空”和“导入文件不存在”。raise_errors_if_no_files=TRUE，则“导入文件不存在”的时候，DWS将上报“文件不存在的”错误。

参数类型：SUSET

取值范围：布尔型

- on表示导入时区分“导入文件记录数为空”和“导入文件不存在”。
- off表示导入时不区分“导入文件记录数为空”和“导入文件不存在”。

默认值：off

partition_max_cache_size

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。通过partition_max_cache_size可指定数据缓存区大小。该值设置过大，将消耗较多系统内存资源；设置过小，将降低列存分区表批量插入性能。

参数类型：USERSET

取值范围：4096~ INT_MAX / 2，最小单位为KB。

默认值：2GB

partition_mem_batch

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。在partition_max_cache_size设置的情况下，通过partition_mem_batch可指定缓存个数。该参数值设置越大，每个分区可用的缓存越小，降低列存分区表批量插入性能；设置越小，每个分区可用的缓存会越大，会消耗较多系统内存资源。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：1~ 65535

默认值：256

gds_debug_mod

参数说明：为了增强对数据并行加载工具General Data Service（以下简称GDS）相关问题的分析定位能力，可以通过此参数选择是否开启GDS的debug功能。参数开启后，将在集群节点对应的日志中输出GDS每次收发的包裹类型、命令交互的对端以及其他交互相关的细节信息，方便记录DWS端状态机的状态跳转，以及目前所处的状态信息。此参数打开会输出额外日志，增加日志IO开销，进而影响性能和日志的信息有效性，因此请仅在定位GDS问题时开启。

参数类型：USERSET

取值范围：布尔型

- on表示开启GDS debug功能。
- off表示不开启GDS debug功能。

默认值：off

max_copy_data_display

参数说明：为copy错误表rawrecord字段长度增加guc管控，该字段为text类型，最大为1G减去8203B（即1073733621B）。该参数仅8.2.1.100及以上版本支持。

设置该参数时，表示能显示的最长字符数，超过该长度结尾显示‘...’。

参数类型：USERSET

取值范围：0~1073733616

默认值：1024

enable_parallel_batch_insert

参数说明：控制行存表和列存表是否开启并发入库功能。该参数仅8.2.0.102及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启并发入库功能。开启并发入库功能可以有效提升行存表和列存表insert、upsert、update和delete语句入库执行速度。
- off表示不开启并发入库功能。

默认值：off

18.7 预写式日志

 版权所有 © 华为云计算技术有限公司

18.7.1 设置

synchronous_commit

参数说明：设置当前事务的同步方式。

参数类型：USERSET

取值范围：枚举类型

- on表示将备机的同步日志刷新到磁盘。
- off表示异步提交。
- local表示为本地提交。
- remote_write表示要备机的同步日志写到磁盘。
- remote_receive表示要备机同步日志接收数据。

默认值：on

⚠ 注意

此参数禁止用户级修改，否则会导致集群发生不可预知异常，需要配置此参数，请联系技术人员确认。

wal_decelerate_policy

参数说明：触发限速后的行为策略，该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- warning表示触发限速条件后仅告警，但不降速。
- decelerate触发限速条件后，根据配速对当前业务进行降速处理。

默认值：warning

说明

warning不影响性能，decelerate会根据配速，对当前业务进行降速处理。

wal_write_speed

参数说明：单DN的每个query每秒最大允许作业触发WAL写入速率（byte/s），该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1024～10240000，单位为KB

默认值：30MB

说明

对于带索引Copy、Delete等大量的作业，会根据配速进行作业速率限制。

版权所有 © 华为云计算技术有限公司

wal_decelerate_trigger_threshold

参数说明：单DN每个query触发wal写入限速的阈值，该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1024～100000000000，单位为KB

默认值：128MB

说明

对于单query产生的xlog量大于该参数设置值时，即触发wal写入限速，对于DDL、少量的DML操作无影响。

commit_delay

参数说明：表示一个已经提交的数据在WAL缓冲区中存放的时间。

参数类型：USERSET

取值范围：整型， 0～100000（微秒），其中0表示无延迟。

默认值：0

须知

- 设置为非 0 值时事务执行commit后不会立即写入WAL中，而仍存放在WAL缓冲区中，等待WalWriter进程周期性写入磁盘。
- 如果系统负载很高，在延迟时间内，其他事务可能已经准备好提交。但如果没有事务准备提交，这个延迟就是在浪费时间。

commit_siblings

参数说明：当一个事务发出提交请求时，如果数据库中正在执行的事务数量大于此参数的值，则该事务将等待一段时间（**commit_delay**的值），否则该事务则直接写入WAL。

参数类型：USERSET

取值范围：整型， 0～1000

默认值：5

wal_compression

参数说明：控制是否对FPI页面进行压缩。

参数类型：USERSET

取值范围：布尔型

- on表示开启FPI压缩。
- off表示关闭FPI压缩。

 版权所有 © 华为云计算技术有限公司

默认值：on

须知

- 当前压缩算法为zlib，暂不支持设置为其他压缩算法。
- 对于通过从低版本升级成为当前版本的集群，此参数默认关闭（off）。如果用户需要，可以通过gs_guc命令打开FPI压缩功能。
- 当前版本若为全新安装版本，此参数默认打开（on）。
- 从低版本升级上来的集群，如果手动开启了此参数，不允许再进行集群回滚操作。

wal_compression_level

参数说明：当打开wal_compression参数时，设置zlib压缩算法的压缩级别。

参数类型：USERSET

取值范围：整型， 0～9

- 0表示不压缩。
- 1表示最低的压缩率。
- 9表示最高的压缩率。

默认值：9

18.7.2 检查点

checkpoint_segments

参数说明：设置checkpoint_timeout周期内所保留的最少WAL日志段文件数量。每个日志文件大小为16MB。

参数类型：SIGHUP

取值范围：整型，1～2147483646

默认值：64

说明

提升此参数可加快大数据的导入速度，但需要结合checkpoint_timeout、shared_buffers这两个参数统一考虑。该参数同时影响WAL日志段文件复用数量，通常情况下pg_xlog文件夹下最大的复用文件个数为2倍的checkpoint_segments个，复用的文件被改名为后续即将使用的WAL日志段文件，不会被真正删除。

18.8 双机复制

 版权所有 © 华为云计算技术有限公司

18.8.1 主服务器

enable_data_replicate

参数说明：当数据库在数据导入行存表时，主机与备机的数据同步方式可以进行选择。

参数类型：USERSET

取值范围：布尔型

- on表示导入数据行存表时主备数据采用数据页的方式进行同步。当replication_type参数为1时，不允许设置为on。
- off表示导入数据行存表时主备数据采用日志（Xlog）方式进行同步。

默认值：on

ha_module_debug

参数说明：用于查看数据复制时具体数据块的复制状态日志。

参数类型：USERSET

取值范围：布尔型

- on表示日志中将打印数据复制时每个数据块的状态。
- off表示日志中不打印数据复制时每个数据块的状态。

默认值：off

18.9 查询规划

18.9.1 优化器方法配置

这些配置参数提供了影响查询优化器选择查询规划的原始方法。如果优化器为特定的查询选择的缺省规划并不是最优的，可以通过使用这些配置参数强制优化器选择一个不同的规划来临时解决这个问题。更好的方法包括调节优化器开销常量、手动运行ANALYZE、增加配置参数**default_statistics_target**的值、增加使用ALTER TABLE SET STATISTICS为指定列增加收集的统计信息。

enable_bitmapscan

参数说明：控制优化器对位图扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_hashagg

参数说明：控制优化器对Hash聚集规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_sortagg

参数说明：控制优化器是否允许agg算子选择sortagg模式。

参数类型：USERSET

取值范围：布尔型

- on表示允许agg算子选择sortagg模式。
- off表示不允许agg算子选择sortagg模式。

默认值：on

enable_mixedagg

参数说明：控制优化器对Mixed Agg聚集规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用，为符合条件的Grouping Sets语句（包括Rollup或Cube）生成Mixed Agg查询计划。
- off表示不使用。

默认值：on

说明

- 新安装的9.1.0.200及以上版本集群中该参数的默认值为on，升级场景该参数的默认值为保持前向兼容维持原值。
- 通常在大数据量场景（单个DN表的数据量在100GB以上），采用Mixed Agg查询计划可以提升语句的执行性能。

  不支持Mixed Agg的场景如下：
  - GROUP BY子句里列的数据类型不支持哈希操作。
  - 聚集函数中包含DISTINCT去重或ORDER BY排序。
  - 使用GROUPING SETS子句时不包含空的分组。

enable_hashjoin

参数说明：控制优化器对Hash连接规划类型的使用。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_indexscan

参数说明：控制优化器对索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_indexonlyscan

参数说明：控制优化器对仅索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_material

参数说明：控制优化器对物化的使用。无法完全消除所有的物化操作，但关闭该参数可以防止优化器插入物化节点。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_mergejoin

参数说明：控制优化器对归并连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

 版权所有 © 华为云计算技术有限公司

默认值：off

enable_nestloop

参数说明：控制优化器是否使用“内表全表扫描嵌套循环连接”。无法完全消除嵌套循环连接，但关闭该参数后，优化器在存在其他连接方法时优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_index_nestloop

参数说明：控制优化器是否使用“内表参数化索引扫描嵌套循环连接”。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

left_join_estimation_enhancement

参数说明：控制left join的行数估计值是否采用优化后的值。该参数仅8.3.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_seqscan

参数说明：控制优化器对顺序扫描规划类型的使用。无法完全消除顺序扫描，但关闭该变量会让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_sort

参数说明：控制优化器使用的排序步骤。无法完全消除明确的排序，但关闭该变量可以让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

max_opt_sort_rows

参数说明：控制ORDER BY子句中最大优化的limit+offset行数。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 0和-1，表示该参数功能不生效。
- 大于0，表示order by子句中limit+offset行数小于该值时，优化生效，大于该值时，优化不生效。优化后耗时减小，但内存消耗可能增大。

默认值：-1

enable_tidscan

参数说明：控制优化器对TID扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_kill_query

参数说明：CASCADE模式删除用户时，会删除此用户拥有的所有对象。此参数标识是否允许在删除用户的时候，取消锁定此用户所属对象的query。

参数类型：SUSET

取值范围：布尔型

- on表示允许取消锁定。
- off表示不允许取消锁定。

默认值：off

 版权所有 © 华为云计算技术有限公司

enforce_oracle_behavior

参数说明：控制正则表达式的规则匹配模式。

参数类型：USERSET

取值范围：布尔型

- on表示正则表达式采用ORACLE格式的匹配规则。
- off表示正则表达式采用POSIX格式的匹配规则。

默认值：on

enable_stream_concurrent_update

参数说明：控制优化器在并发更新场景下对stream的使用，该参数受限于**enable_stream_operator**参数。

参数类型：USERSET

取值范围：布尔型

- on表示允许优化器对update语句生成stream计划。
- off表示优化器对update语句仅能生成非stream计划。

默认值：on

enable_stream_ctescan

参数说明：控制stream计划是否支持ctescan。

参数类型：USERSET

取值范围：布尔型

- on表示stream计划下支持ctescan。
- off表示stream计划下不支持ctescan。

默认值：on

说明

升级场景该参数的默认值为保持前向兼容维持原值。

enable_stream_operator

参数说明：控制优化器对stream的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_stream_recursive

参数说明：控制是否将with recursive关联查询下推DN分布式执行。

该参数在8.2.1.210及以上集群版本中废弃。为了保留前向兼容，可以设置成功，但实际不生效。建议使用**stream_recursive_option**参数。

参数类型：USERSET

取值范围：布尔型

- on表示支持使用with recursive关联查询下推DN分布式执行。
- off表示不支持使用with recursive下推。

默认值：on

stream_recursive_option

参数说明：控制是否将with recursive关联查询下推DN分布式执行。该参数仅8.2.1.210及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- none，表示不支持使用with recursive下推。
- normal，表示在单逻辑集群场景下，支持使用with recursive关联查询下推DN分布式执行。
- nodegroup，表示在多逻辑集群场景下，支持使用with recursive关联查询下推DN分布式执行。
- classic，表示在单逻辑集群场景下，支持使用with recursive关联查询下推DN分布式执行（SELECT常量场景除外），该选项仅9.1.1.300及以上集群版本支持。

默认值：normal

enable_value_redistribute

参数说明：控制是否开启生成value redistribute优化计划，8.2.0及以上集群版本中，该参数针对不带Partition by子句的rank、dense_rank、row_number是否生成value redistribute优化计划生效。

参数类型：USERSET

取值范围：布尔型

- on表示支持使用value redistribute生成优化计划。
- off表示不支持使用value redistribute生成优化计划。

默认值：on

value_redistribute_options

参数说明：控制以下两种查询场景中是否生成value redistribute优化计划。该参数仅9.1.1.100及以上集群版本支持。

场景一：不带Partition by子句的rank、dense_rank、row_number查询场景。

 版权所有 © 华为云计算技术有限公司

场景二：limit x offset y此类topk查询场景。

参数类型：USERSET

取值范围：字符串

- none，表示对所有场景都不使用value redistribute生成优化计划。
- winagg，表示支持对场景一使用value redistribute生成优化计划。
- topk，表示支持对场景二使用value redistribute生成优化计划。

默认值：winagg

说明

场景二topk场景应用value redistribute生成优化计划的限制如下：
1. value_redistribute_options需设置topk，同时开启enable_topk_optimization。
2. 计划支持turbo。
3. limit和offset子句中都是常数。
4. limit + offset满足**late_read_thresholds**中第7个和第8个参数值设置的阈值。

max_recursive_times

参数说明：控制with recursive的最大迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX。

默认值：200

enable_vector_engine

参数说明：控制优化器对向量化执行引擎的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_broadcast

参数说明：控制优化器对stream代价估算时对broadcast分布方式的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_redistribute

参数说明：控制优化器对stream代价估算时对local redistribute和split redistribute分布方式的使用。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_change_hjcost

参数说明：控制优化器在Hash Join代价估算路径选择时，是否使用将内表运行时代价排除在Hash Join节点运行时代价外的估算方式。如果使用，则有利于选择条数少，但运行代价大的表做内表。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_fstream

参数说明：控制优化器下发语句时对stream的使用，该参数只限定于HDFS外表使用。

该参数现在已经废弃。为了保留前向兼容，可以设置成功，但是实际不起任何作用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_hashfilter

参数说明：控制是否允许对包含复制表(包括dual表和常量表)的计划生成hashfilter。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许生成hashfilter。
- off表示任何情况下都不允许生成hashfilter。

默认值：on

 版权所有 © 华为云计算技术有限公司

best_agg_plan

参数说明：对于stream下的Agg操作，优化器会生成三种计划：

1. hashagg+gather(redistribute)+hashagg。
2. redistribute+hashagg(+gather)。
3. hashagg+redistribute+hashagg(+gather)。

本参数用于控制优化器生成哪种hashagg的计划。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为1时，强制生成第一种计划。
- 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
- 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
- 取值为0时，优化器会根据以上三种计划的估算cost选择最优的一种计划生成。

默认值：0

turbo_engine_version

参数说明：对于建表指定turbo存储格式表（表属性中enable_turbo_store参数设置为on），且当查询不涉及merge join或sort agg算子时，执行器可走turbo执行引擎，执行器部分性能可获得成倍性能提升。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为0时，表示turbo执行引擎关闭。
- 取值为1时，表示仅针对单表agg查询场景使用turbo执行引擎。
- 取值为2时，表示仅针对单表agg或多表join关联查询场景使用turbo执行引擎。
- 取值为3时：
  - 9.1.1.300及以上集群版本中对所有列存算子都可使用turbo执行引擎加速。
  - 9.1.1.300以下集群版本中不支持算子如merge join，sort agg等算子（数据量较大且turbo_engine_version取值为3时，merge join，sort agg算子出现的情况较少，因此基本可以实现任意SQL语句的turbo执行引擎加速）。

默认值：3

⚠ 注意

- 开启turbo引擎要求X86服务器支持AVX指令集，ARM服务器支持NEON指令集。
- 8.2.1及以下集群版本升级到9.1.0.226及以上集群版本的场景，该参数的默认值为0。

版权所有 © 华为云计算技术有限公司

enable_bucket_stream_opt

参数说明：对于二级分区表或3.0 hash分布表使用bucket agg和bucket join执行策略。通过避免数据的local重分布或广播，提升SQL语句执行效率。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- true，表示优化器在满足策略生效条件时使用bucket agg和bucket join执行策略生成计划。如使用该优化策略，执行explain末尾会显示字段Bucket Stream: true。
- false，表示不使用bucket agg和bucket join执行策略生成计划。

默认值：true

说明

- 新装9.1.0.200及以上版本集群中该参数的默认值为true，升级场景该参数的默认值为保持前向兼容维持原值。
- bucket agg和bucket join执行策略生效要求当前query可用CPU个数小于等于16，且建表满足以下条件之一：

  1. 二级分区要求分布列与二级分区的secondary_part_column相同，推荐二级分区个数为DN个数*12，支持倍数包括4, 6, 8, 12, 16。

  2. 3.0表要求表为hash分布，且bucketnums/DN个数>10。
- 当计划中local stream代价较低，query可能不会选择bucket agg和bucket join执行策略。

spill_compression

参数说明：控制执行器算子运行数据由于内存不足下盘时的压缩算法。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- 'lz4'，表示使用lz4压缩算法，对于下盘量较小场景性能更优，但占用更多存储空间。
- 'zstd'，表示使用zstd压缩算法，对于下盘量较大场景，IO为主要瓶颈，其性能更优，且存储空间约为lz4的2/3。

默认值：'lz4'

index_selectivity_limit

参数说明：控制列存cbtree索引选择率大于此阈值时，查询优化器会增加索引扫描惩罚代价，以降低其被优先选择的可能性。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0~1。

- 取值为0时，表示参数不生效。
- 取值为其他值时，表示列存cbtree索引选择率百分比，即超过百分之多少会增加惩罚代价。

 版权所有 © 华为云计算技术有限公司

默认值：0.001

index_selectivity_cost

参数说明：控制列存表索引扫描选择率大于选择阈值时cbtree的cost计算，该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，-1，0～10000，负数仅支持设置-1，且不允许设置其他负数。

- 取值为0时，不受索引选择率阈值的影响。
- 取值为-1时，受disable_cost影响。
- 为其他值时，为cbtree cost计算的系数。

默认值：-1

说明

- 9.1.0及以上集群版本中，index_selectivity_cost参数需满足：选择率>index_selectivity_limit的设置值。
- 9.1.0以下集群版本中，index_selectivity_cost参数需满足：选择率>0.001

index_cost_limit

参数说明：控制列存表索引扫描时cbtree的cost计算不生效阈值，该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~2147483647。

- 取值为0时，表示参数不生效。
- 取值为其他值时，表行数少于该值时不受索引选择率阈值的影响。

默认值：0

说明

- 9.1.0及以上集群版本中：选择率为index_selectivity_limit的设置值。
- 9.1.0以下集群版本中：选择率阈值为0.001。

volatile_shipping_version

参数说明：控制volatile函数下推执行的范围。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为3时，在2的基础上，扩展支持InlineCTE在只被引用一次时支持下推，扩展支持复制表UPSERT中使用包含volatile函数时的下推。
- 取值为2时，在1的基础上，扩展支持在复制CTE结果的目标列中包含volatile函数时的下推。
- 取值为1时，扩展支持nextval、uuid_generate_v1、sys_guid、uuid函数出现在语句目标列时的完全下推。

 版权所有 © 华为云计算技术有限公司

- 取值为0时，支持random类函数的完全下推，nextval、uuid_generate_v1函数仅在INSERT含有简单查询语句的部分下推。

默认值：3

agg_redistribute_enhancement

参数说明：当进行Agg操作时，如果包含多个group by列且均不为分布列，进行重分布时会选择某一group by列进行重分布。本参数控制选择重分布列的策略。

参数类型：USERSET

取值范围：布尔型

- on表示会选择估算distinct值最多的一个可重分布列作为重分布列。
- off表示会选择第一个可重分布列为重分布列。

默认值：on

说明

新安装9.1.1.100及以上集群版本中该参数的默认值为on。升级场景下为保持前向兼容该参数的默认值维持原值。

enable_valuepartition_pruning

参数说明：是否对DFS（Distributed File System，分布式文件系统）分区表进行静态/动态优化。

参数类型：USERSET

取值范围：布尔型

- on表示对DFS分区表进行静态/动态优化。
- off表示不对DFS分区表进行静态/动态优化。

默认值：on

expected_computing_nodegroup

参数说明：标识选定的计算Node Group模式或目标计算Node Group。Node Group目前为内部用机制，用户无需设置。

共4种计算Node Group模式，用于关联操作和聚集操作时选定计算Node Group。在每一种模式中，优化器有针对性地选定几个候选计算Node Group，然后根据代价，从中为当前算子挑选更合适的计算Node Group。

参数类型：USERSET

取值范围：字符串

- optimal：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前用户具有COMPUTE权限的所有Node Group包含的所有DN构成的Node Group。
- query：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前查询涉及的所有基表所在Node Group包含的所有DN构成的Node Group。

 版权所有 © 华为云计算技术有限公司

- bind：当前session用户是逻辑集群用户时，候选计算Node Group为当前用户关联的逻辑集群的Node Group；当session用户不是逻辑集群用户时，候选计算Node Group选取规则和参数设置为query时的规则一致。
- Node Group名：
  - **enable_nodegroup_debug**为off时：候选计算Node Group列表包含算子操作对象所在的Node Group和该指定的Node Group。
  - **enable_nodegroup_debug**为on时：候选计算Node Group为指定的Node Group。

默认值：bind

enable_nodegroup_debug

参数说明：控制优化器在多Node Group环境下，是否使用强制弹性计算。Node Group目前为内部用机制，用户无需设置。

该参数只在**expected_computing_nodegroup**被设置为具体Node Group时生效。

参数类型：USERSET

取值范围：布尔型

- on表示强制将计算弹性到expected_computing_nodegroup所指定的Node Group进行计算。
- off表示不强制使用某个Node Group进行计算。

默认值：off

stream_multiple

参数说明：设置优化器计算Stream算子的开销时的加权。

在原代价模型的基础上，最终Stream代价将被乘以此加权参数。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：1

说明

此参数仅对Redistribute和Broadcast类型的Stream有效。

force_bitmapand

参数说明：控制优化器对bitmap and规划类型的强制使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用bitmap and规划类型。
- off表示不使用bitmap and规划类型。

默认值：off

 版权所有 © 华为云计算技术有限公司

qrw_inlist2join_optmode

参数说明：控制是否使用inlist-to-join查询重写。

参数类型：USERSET

取值范围：字符串

- disable：关闭inlist2join查询重写。
- cost_base：基于代价的inlist2join查询重写。
- rule_base：基于规则的inlist2join查询重写，即强制使用inlist2join查询重写。
- 任意正整数：inlist2join查询重写阈值，即list内元素个数大于该阈值，进行inlist2join查询重写。

默认值：disable

inlist_hash_threshold

参数说明：控制inlist元素个数阈值，阈值大于该参数值才会触发inlist哈希优化。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0～10000，其中0表示inlist哈希功能关闭。

默认值：10

enable_inlist_hashing

参数说明：控制是否使用inlist哈希优化。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启inlist哈希优化。
- off表示关闭inlist哈希优化。

默认值：on

setop_optmode

参数说明：控制不带ALL选项的集合操作(UNION/EXCEPT/INTERSECT)语句的各个查询分支语句是否执行去重操作。

参数类型：USERSET

取值范围：枚举型

- disable：查询分支不执行去重操作。
- force：强制查询分支执行去重操作。
- cost：优化器在查询分支去重和不去重这两种执行方式中，选择代价比较小的执行方式。

默认值：cost

 版权所有 © 华为云计算技术有限公司

须知

- 新装9.1.0.200及以上版本集群中该参数的默认值为cost，升级场景该参数的默认值为保持前向兼容维持原值。
- 此参数配置仅在SQL语句的执行计划满足以下条件时生效：
  - SQL语句中的UNION/EXCEPT/INTERSECT操作不带ALL选项。
  - 执行集合操作的各个查询分支在进行集合操作前执行过数据重分布动作。

skew_option

参数说明：控制是否使用优化策略。

参数类型：USERSET

取值范围：字符串

- off：关闭策略。
- normal：采用激进策略。对于不确定是否出现倾斜的场景，认为存在倾斜，并进行相应优化。
- lazy：采用保守策略。对于不确定是否出现倾斜场景，认为不存在倾斜，不进行优化。

默认值：normal

enable_expr_skew_optimization

参数说明：控制是否在倾斜优化策略中使用表达式统计信息。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示在倾斜优化策略中使用表达式统计信息来判断是否存在倾斜。
- off表示在倾斜优化策略中不使用表达式统计信息来判断是否存在倾斜。

默认值：on

prefer_hashjoin_path

参数说明：控制是否优先生成hashjoin路径，以便通过将代价较大的其他路径预剪枝掉的方法来提升整体计划生成时间。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用hashjoin路径提前生成的优化。
- off表示关闭hashjoin路径提前生成的优化。

默认值：on

 版权所有 © 华为云计算技术有限公司

enable_hashfilter_test

参数说明：该参数用于控制是否为基表扫描增加分布列的hashfilter，以便确认结果是否符合预期。

参数类型：USERSET

取值范围：布尔型

- on表示为基表扫描添加分布列的hashfilter。
- off表示不会为基表扫描添加分布列的hashfilter。

默认值：off

须知

该参数仅对hash分布的表有效。

enable_cu_align_8k

参数说明：设置V3表CU是否8K字节对齐。该参数仅9.1.0及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示V3表CU按照8192字节对齐。
- off表示V3表CU按照512字节对齐。

默认值：存算一体集群版本中默认值为on，存算分离集群版本中默认值为off。

enable_cu_batch_insert

参数说明：设置V2表是否开启多列CU批量写功能。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示V2表开启多列CU批量写功能。
- off表示V2表未开启多列CU批量写功能。

默认值：off

enable_topk_optimization

参数说明：控制是否开启TopK排序优化。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启TopK排序优化。
- off表示关闭TopK排序优化。

 版权所有 © 华为云计算技术有限公司

默认值：on

late_read_strategy

参数说明：控制延迟物化策略。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- topk，表示针对同时带有sort和limit的语句进行延迟物化优化。
- none，表示不使用延迟物化优化。

默认值：topk

late_read_thresholds

参数说明：控制延迟物化开启的阈值。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

9.1.1.100之前的版本可设置三个参数：

- 第一个参数，表示延迟物化宽度占基表输出总宽度百分比，取值范围为0~100。
- 第二个参数，表示语句与基表输出行数千分比，取值范围为0~1000。
- 第三个参数，控制是否开启SMP，门槛值为各DN延迟物化的行数，取值范围为INT64MIN~INT64MAX。

9.1.1.100及以上版本增加以下五个参数设置：

- 第四个参数，表示topk采用逆排序优化的排序行数阈值，取值范围为INT64MIN~INT64MAX。
- 第五个参数，表示topk采用逆排序优化的（limit + offset）/ 排序行数百分比阈值，取值范围为0~100。
- 第六个参数，表示topk采用逆排序优化时是否允许有WHERE子句，取值范围为0~1。
- 第七个参数，表示topk采用value redistribute优化计划的（limit + offset）/ 排序行数百分比阈值，取值范围为0~100。
- 第八个参数，表示topk采用value redistribute优化计划的 limit + offset 的行数阈值，取值范围为INT64MIN~INT64MAX。
- 第九个参数，表示topk采用value redistribute优化计划的内存估算阈值，取值范围为0~INTMAX，单位KB。设置为0时，表示topk禁用value redistribute优化计划。该参数仅9.1.1.200及以上集群版本支持。

默认值：

- 9.1.1.200及以上集群版本中为60,1,100,10000,-1,0,15,100000,262144
- 9.1.1.100集群版本中为60,1,100,10000,-1,0,15,100000
- 9.1.1.100之前的集群版本中为60,1,100

 版权所有 © 华为云计算技术有限公司

insert_dop

参数说明：控制insert dop并发数量。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0 ~ 64。

- insert_dop=0时，表示insert并发度跟随子计划中查询语句的并发度。
- insert_dop=1时，表示关闭insert dop功能。
- insert_dop>1时，表示insert的并发度设置为insert_dop与子计划中查询语句的并发度的最小值。

默认值：1

说明

- 如果2.0版本（colversion=2.0）hstore_opt表开启insert dop，需先打开参数enable_insert_column_table_dop。
- insert_dop=0时，即开启自适应模式，并发度跟随子计划中查询语句的并发度，部分场景效果可能不理想。
- 避免设置过大的insert_dop值，可能会导致资源占比过高，建议用户根据实际会话需要进行设置。
- 使用insert_dop时需考虑insert业务并发度和表的列数。并发度越高和表列数越多的场景中，建议insert_dop设置相对越低。如果表列数超过1000列不建议开启insert_dop，避免影响性能。

enable_hstoreopt_auto_bitmap

参数说明：用于控制是否默认在建表时给HStore Opt表自动设置bitmap columns列。该参数仅9.1.0.100及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示默认设置bitmap columns选项。
- off表示默认不设置bitmap columns选项。

默认值：off

enable_hstoreopt_insert_sort

参数说明：用于控制HStore Opt表是否开启入库排序（包括VACUUM FULL）。该参数仅9.1.0.100及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启入库排序。
- off表示在异步排序开启（autovacuum_asyncsort_time_limit大于0）时，关闭入库排序（包括VACUUM FULL)；异步排序关闭时，入库排序（包括VACUUM FULL) 仍然启用。

默认值：on

 版权所有 © 华为云计算技术有限公司

experimental_options

参数说明：控制实验性的优化选项。该参数仅9.1.0.226及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- none：不启用任何优化选项。
- vecsort_opt：控制列存sort是否使用TurboSort优化。
- disable_jsonagg_pushdown：控制是否禁用json_agg函数下推优化。
- vector_window_opt：控制first_value和last_value是否支持向量化引擎。

默认值：none

roaringbitmap_optimize_rule

参数说明：控制是否使用roaringbitmap进行去重优化。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- none，表示不使用roaringbitmap进行去重优化。
- uniq，表示仅对UNIQ函数使用roaringbitmap进行去重优化。

默认值：none

18.9.2 优化器开销常量

介绍优化器开销常量。这里描述的开销可以按照任意标准度量。只关心其相对值，因此以相同的系数缩放它们将不会对优化器的选择产生任何影响。缺省时，它们以抓取顺序页的开销为基本单位。也就是说将seq_page_cost设为1.0，同时其他开销参数以它为基准设置。也可以使用其他基准，比如以毫秒计的实际执行时间。

seq_page_cost

参数说明：设置优化器计算一次顺序磁盘页面抓取的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：1

random_page_cost

参数说明：设置优化器计算一次非顺序抓取磁盘页面的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：4

 版权所有 © 华为云计算技术有限公司

说明

- 虽然服务器允许将random_page_cost设置的比seq_page_cost小，但是物理上实际不受影响。如果所有数据库都位于随机访问内存中时，两者设置为相等很合理。因为在此种情况下，非顺序抓取页并没有副作用。同样，在缓冲率很高的数据库上，应该相对于CPU参数同时降低这两个值，因为获取内存中的页要比通常情况下开销小很多。
- 对于特别表空间中的表和索引，可以通过设置同名的表空间的参数来覆盖这个值。
- 相对于seq_page_cost，减少这个值将导致系统更倾向于使用索引扫描，而增加这个值使得索引扫描开销比较高。可以通过同时增加或减少这两个值来调整磁盘I/O相对于CPU的开销。

cpu_tuple_cost

参数说明：设置优化器计算在一次查询中处理每一行数据的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.01

cpu_index_tuple_cost

参数说明：设置优化器计算在一次索引扫描中处理每条索引的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.005

cpu_operator_cost

参数说明：设置优化器计算一次查询中执行一个操作符或函数的开销。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0.0025

effective_cache_size

参数说明：设置优化器在一次单一的查询中可用的磁盘缓冲区的有效大小。

设置这个参数，还要考虑DWS的共享缓冲区以及内核的磁盘缓冲区。另外，还要考虑预计的在不同表之间的并发查询数目，因为它们将共享可用的空间。

这个参数对DWS分配的共享内存大小没有影响，它也不会使用内核磁盘缓冲，它只用于估算。数值是用磁盘页来计算的，通常每个页面是8192字节。

参数类型：USERSET

取值范围：整型，1～INT_MAX，单位为8KB。

比默认值高的数值可能会导致使用索引扫描，更低的数值可能会导致选择顺序扫描。

默认值：128MB

 版权所有 © 华为云计算技术有限公司

allocate_mem_cost

参数说明：设置优化器计算Hash Join创建Hash表开辟内存空间所需的开销，供Hash join估算不准时调优使用。

参数类型：USERSET

取值范围：浮点型，0～10000。

默认值：0

smp_thread_cost

参数说明：设置优化器计算一个算子并行线程所需的开销，用于在生成query_dop不合适时调优使用。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，1～10000。

默认值：1000

18.9.3 基因查询优化器

介绍基因查询优化器相关的参数。基因查询优化器（GEQO）是一种启发式的查询规划算法。这个算法减少了对复杂查询规划的时间，而且生成规划的开销有时也小于正常的详尽的查询算法。

geqo

参数说明：控制基因查询优化的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

说明

通常情况下在执行过程中不要关闭，geqo_threshold变量提供了更精细的控制GEQO的方法。

geqo_threshold

参数说明：如果执行语句的数量超过设计的FROM的项数，则会使用基因查询优化来执行查询。

参数类型：USERSET

取值范围：整型，2～INT_MAX。

默认值：12

 版权所有 © 华为云计算技术有限公司

说明

- 对于简单的查询，通常使用详尽的搜索方法，当涉及多个表的查询的时候，用GEQO可以更好地管理查询。
- 一个FULL OUTER JOIN构造仅作为一个FROM项。

geqo_effort

参数说明：控制GEQO在规划时间和规划质量之间的平衡。

参数类型：USERSET

取值范围：整型，1～10

默认值：5

说明

- 比默认值大的数值增加了查询规划的时间，但是也增加了选中有效查询的几率。
- geqo_effort实际上并没有直接作用，只是用于计算其他影响GEQO的变量的默认值。如有需求，可以手动设置其他相关参数。

geqo_pool_size

参数说明：控制GEQO使用池的大小，也就是基因全体中的个体数量。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：0

说明

设置为0，表示使用系统自适应方式，DWS会基于geqo_effort和表的个数选取合适的值。若手动指定，必须大于等于2，且有效范围通常介于100到1000之间。

geqo_generations

参数说明：控制GEQO使用的算法的迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：0

说明

如果设置为0，则基于geqo_pool_size选取合适的值。若手动指定，必须大于等于1，且有效范围通常介于100和1000之间。

geqo_selection_bias

参数说明：控制GEQO的选择性偏好，即一个种群中的选择性压力。

参数类型：USERSET

取值范围：浮点型，1.5～2.0

 版权所有 © 华为云计算技术有限公司

默认值：2

说明

数值越大表示选择性压力越高，更优的个体被选中的概率更高。数值越小表示选择性压力越低，允许更多多样性。

geqo_seed

参数说明：控制GEQO使用的随机数生产器的初始化值，用来从顺序连接在一起的查询空间中查找随机路径。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0

说明

不同的值会改变搜索的连接路径，从而影响了所找路径的优劣。

18.9.4 其他优化器选项

default_statistics_target

参数说明：为未通过ALTER TABLE SET STATISTICS设置字段目标的表设置缺省统计目标。此参数设置为正数时，表示统计信息的样本数量；设置为负数时，表示使用百分比的形式设置统计目标。负数转换为对应的百分比，例如-5代表5%。

采样时，系统会将default_statistics_target * 300作为随机抽样的大小。例如，默认值为100时，会随机读取30000个页面再从中随机取30000条数据完成随机抽样。

参数类型：USERSET

取值范围：浮点型，-100～10000。

默认值：100

说明

- 比默认值大的正数数值增加了ANALYZE所需的时间，但是可能会改善优化器的估计质量。
- 调整此参数可能存在性能劣化的风险，如果某个查询劣化，可以考虑：
  1. 恢复默认的统计信息。
  2. 使用plan hint来调整到之前的查询计划。（详细参见**使用Plan Hint进行调优**）
- 当此guc参数设置为负数时，如果计算的采样样本数大于等于总数据量的2%，且用户表的数据量小于1600000时，ANALYZE所需时间相比guc参数为默认值的时间会有所增加。
- autoanalyze不支持临时表采样方式设置采样大小，采样过程使用参数默认值。
- 当强制使用内存方式计算统计信息时，采样大小受maintenance_work_mem参数限制。

random_function_version

参数说明：控制ANALYZE在进行数据采样时选取的random函数版本。该参数仅8.1.2及以上版本支持。

参数类型：USERSET

版权所有 © 华为云计算技术有限公司

取值范围：枚举类型

- 0，表示采用C标准库提供的random函数。
- 1，表示采用优化增强的random函数。

默认值：1

说明

新安装集群中该参数的默认值为1。升级场景下，若目标版本为9.1.1.300及以上集群版本，默认值为1；若目标版本低于9.1.1.300集群版本，则默认值为0。

constraint_exclusion

参数说明：控制查询优化器使用表约束查询的优化。

参数类型：USERSET

取值范围：枚举类型

- on，表示检查所有表的约束。
- off，表示不检查约束。
- partition，表示只检查继承的子表和UNION ALL子查询。

默认值：partition

说明

- 当constraint_exclusion为on时，优化器将查询条件和表的CHECK约束进行比较，在查询条件和约束冲突时跳过对表的扫描。
- constraint_exclusion默认开启时，通常用于实现表分区。若对所有表开启，会增加简单查询的规划开销，对简单查询没有帮助。如果不使用分区表，可以关闭该参数。

cursor_tuple_fraction

参数说明：优化器估计游标获取行数在总行数中的占比。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0.1

说明

比默认值小的值与使用 “fast start” 为游标规划的值相偏离，从而使得前几行恢复的很快而抓取全部的行需要很长的时间。比默认值大的值加大了总的估计的时间。在最大的值1.0处，像正常的查询一样规划游标，只考虑总的估计时间和传送第一行的时间。

from_collapse_limit

参数说明：根据生成的FROM列表的项数来判断优化器是否将把子查询合并到上层查询，如果FROM列表项个数小于等于该参数值，优化器会将子查询合并到上层查询。

参数类型：USERSET

取值范围：整型，1～INT_MAX

 版权所有 © 华为云计算技术有限公司

默认值：8

⚠ 注意

比默认值小的数值会减少规划时间，但可能会降低执行计划的质量。

join_collapse_limit

参数说明：根据得出的列表项数来判断优化器是否执行把除FULL JOINS之外的JOIN构造重写到FROM列表中。

参数类型：USERSET

取值范围：整型，1～INT_MAX。

默认值：8

⚠ 注意

- 设置为1会避免任何JOIN重排。这样就使得查询中指定的连接顺序就是实际的连接顺序。查询优化器并不是总能选取最优的连接顺序，高级用户可以选择暂时把这个变量设置为1，然后指定它们需要的连接顺序。
- 比默认值小的数值会减少规划时间，但可能会降低执行计划的质量。

join_search_mode

参数说明：标识计划路径搜索的方式。

参数类型：USERSET

取值范围：枚举类型

- exhaustive，使用传统动态规划和遗传基因方式进行计划路径搜索。
- heuristic，使用启发式方法进行计划路径搜索。该方式会加快计划生成的性能，但存在忽略掉最优计划的可能性。该设置仅对于指定Drive Hint或连接表个数大于from_collapse_limit的场景生效。

默认值：heuristic

enable_from_collapse_hint

参数说明：标识是否优先以hint生效的方式重写FROM列表，其次再根据from_collapse_limit、join_collapse_limit参数进行重写。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on，表示优先以hint生效的方式重写FROM列表。
- off，表示无差别重写FROM列表。

 版权所有 © 华为云计算技术有限公司

默认值：on

说明

- 该参数启用时，优化器会优先以hint生效的方式重写FROM列表，但hint是否生效在计划生成后才能知道，因此有可能出现该hint不生效的情况。
- 该参数关闭时，回退到8.2.0版本之前的计划，即生成计划时不区分表是否有hint。

plan_mode_seed

参数说明：该参数为调测参数，目前仅支持OPTIMIZE_PLAN和RANDOM_PLAN两种。其中：OPTIMIZE_PLAN表示通过动态规划算法进行代价估算的最优plan，参数值设置为0；RANDOM_PLAN表示随机生成的plan；如果设置为-1，表示用户不指定随机数的种子标识符seed值，由优化器随机生成[1, 2147483647]范围整型值的随机数，并根据随机数生成随机的执行计划；如果用户指定guc参数值为[1, 2147483647]范围的整型值，表示指定的生成随机数的种子标识符seed，优化器需要根据seed值生成随机的执行计划。

参数类型：USERSET

取值范围：整型，-1~ 2147483647

默认值：0

说明

- 当该参数设置为随机执行计划模式时，优化器会生成不同的随机执行计划，该执行计划可能不是最优计划。因此在随机计划模式下，会对查询性能产生影响，所以建议在升级、扩容、缩容等正常业务操作或运维过程中将该参数保持为默认值0。
- 当该参数不为0时，查询指定的plan hint不会生效。

enable_hdfs_predicate_pushdown

参数说明：标识是否启用谓词下推至原生数据层的功能。

参数类型：USERSET

取值范围：布尔型

- on表示启用谓词下推至原生数据层的功能。
- off表示不启用谓词下推至原生数据层的功能。

默认值：on

windowagg_pushdown_enhancement

参数说明：标识是否在聚集场景下启用窗口函数谓词下推增强功能。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示聚集场景启用窗口函数谓词下推增强功能。
- off表示聚集场景不启用窗口函数谓词下推增强功能。

默认值：on

 版权所有 © 华为云计算技术有限公司

implied_quality_optmode

参数说明：标识语句中等值列的条件传递优化策略。

参数类型：USERSET

取值范围：枚举类型

- normal，表示向前兼容8.1.3及之前低集群版本，即推导出的表达式行为优化。
- negative，表示推导出的表达式行为不优化。
- positive，在normal的基础上增加了类型转换表达式的优化。

默认值：normal

enable_random_datanode

参数说明：标识是否允许开启复制表DN随机查找功能，复制表在每个DN存放一份完整数据，随机选取可以缓解节点压力。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启复制表DN随机查找功能。
- off表示不允许开启复制表DN随机查找功能。

默认值：on

hashagg_table_size

参数说明：用于设置执行HASH AGG操作时HASH表的大小。

参数类型：USERSET

取值范围：整型，0～INT_MAX/2

默认值：0

enable_codegen

参数说明：标识是否允许开启代码生成优化，目前代码生成使用的是LLVM优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启代码生成优化。
- off表示不允许开启代码生成优化。

默认值：off

说明

目前LLVM优化仅支持向量化执行引擎特性和SQL on Hadoop特性，在其他场景下建议关闭此参数。

 版权所有 © 华为云计算技术有限公司

codegen_strategy

参数说明：标识在表达式codegen化过程中所使用的代码生成优化策略。

参数类型：USERSET

取值范围：枚举类型

- partial，表示当所计算表达式中即使包含部分未被codegen化的函数时，仍可借助表达式全codegen框架调用LLVM动态编译优化策略。
- pure，表示当所计算表达式整体可被codegen化时，才考虑调用LLVM动态编译优化策略。

默认值：partial

说明

在开启代码生成优化会导致查询性能下降的场景下可以设置此参数为pure，其他场景下建议不改变此参数的默认值partial。

enable_codegen_print

参数说明：标识是否允许在log日志中打印所生成的LLVM IR函数。

参数类型：USERSET

取值范围：布尔型

- on表示允许在log日志中打印IR函数。
- off表示不允许在log日志中打印IR函数。

默认值：off

codegen_cost_threshold

参数说明：由于LLVM编译生成最终的可执行机器码需要一定时间，因此只有当实际执行的代价大于编译生成机器码所需要的代码和优化后的执行代价之和时，利用代码生成才有收益。codegen_cost_threshold标识代价的阈值，当执行估算代价大于该代价时，使用LLVM优化。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：10000

llvm_compile_expr_limit

参数说明：用于标识编译表达式个数的阈值，当表达式个数高于该阈值时，只将阈值之内的表达式使用LLVM编译，高于阈值之外的表达式不使用LLVM编译，同时生成编译告警提示（告警提示需要在执行explain performance前SET analysis_options="on(LLVM_COMPILE)"）。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

默认值：500

 版权所有 © 华为云计算技术有限公司

llvm_compile_time_limit

参数说明：LLVM编译时间在执行器运行时间中的占比超过llvm_compile_time_limit所设置的阈值，说明LLVM编译时间占比过高，生成告警提示（告警提示需要在执行explain performance前SET analysis_options="on(LLVM_COMPILE)"）。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0.2

enable_constraint_optimization

参数说明：设置是否允许HDFS外表使用信息约束（Informational Constraint）优化执行计划。

参数类型：USERSET

取值范围：布尔型

- on表示允许使用信息约束优化执行计划。
- off表示不允许使用信息约束优化执行计划。

默认值：on

enable_bloom_filter

参数说明: 标识是否允许使用BloomFilter优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许使用BloomFilter优化。
- off表示不允许使用BloomFilter优化。

默认值：on

 版权所有 © 华为云计算技术有限公司

说明

适用场景：外表侧同线程包含有HDFS内外表或列存表的HASH JOIN会触发Bloom Filter。

使用限制：

1. JOIN类型仅限于INNER JOIN、SEMI JOIN、RIGHT JOIN、RIGHT SEMI JOIN、RIGHT ANTI JOIN、RIGHT ANTI FULL JOIN。
2. JOIN内表侧关联条件：对于HDFS内外表不能为表达式；对于列存表可以为表达式，但仅限于非JOIN层计算的表达式。
3. JOIN外表侧关联条件必须为简单列关联。
4. JOIN内表侧与外表侧关联条件均为简单列关联时，计划层估算必须可以去除1/3以上的数据（仅针对HDFS内外表）。
5. JOIN不能包含null值关联。
6. 数据类型：
   - HDFS内外表字段类型支持SMALLINT、INTEGER、BIGINT、REAL/FLOAT4、DOUBLE PRECISION/FLOAT8、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、CLOB、TEXT。
   - 列存表字段类型支持SMALLINT、INTEGER、BIGINT、OID、"char"、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、NVARCHAR2(n)、CLOB、TEXT、DATE、TIME、TIMESTAMP、TIMESTAMPTZ，其中字符类型的排序规则必须指定为"C"。

runtime_filter_type

参数说明: 标识使用的runtime filter类型，仅在打开**enable_bloom_filter**时生效。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- All，表示应用所有场景下的runtime filter。
- global_filter，表示应用join场景的global runtime filter以及带有limit的order by场景下的runtime filter。
- Topn_filter，表示应用join场景的local runtime filter以及带有limit的order by场景下的runtime filter。
- Bloom_filter，表示仅应用join场景下的local runtime filter，且满足条件后join会生成bloom filter进行过滤。
- Min_max，表示仅应用join场景下的local runtime filter，且join场景仅会生成min_max过滤器。
- None，表示不使用runtime filter，此时仅对原版bloom filter生效场景具有过滤效果。

默认值：All

 版权所有 © 华为云计算技术有限公司

说明

- 新安装的9.1.1.100及以上集群版本中该参数的默认值为All，从低版本升级到9.1.1.100及以上集群版本场景下，该参数默认值为'min_max,bloom_filter,topn_filter'。
- join场景的local runtime filter：指列存表的HASH JOIN外表侧应用runtime filter的cstore scan算子上无stream算子或只有local stream算子。
- join场景的global runtime filter：指列存表的HASH JOIN外表侧应用runtime filter的cstore scan算子上含有非local stream算子。
- 适用场景：列存表的HASH JOIN外表侧的计划类型以及列存表带有limit的order by计划类型。
- 使用限制：
  - JOIN场景使用限制同**enable_bloom_filter**参数的限制。
  - 带有limit的order by场景下，order by字段类型仅支持SMALLINT、INTEGER、BIGINT、"char"、CHAR(n)/CHARACTER(n)/NCHAR(n)、VARCHAR(n)/CHARACTER VARYING(n)、NVARCHAR2(n)、TEXT、DATE、TIME、TIMESTAMP、TIMESTAMPTZ，其中字符类型其排序规则必须指定为"C"。

runtime_filter_ratio

参数说明: 控制runtime filter在join场景中是否使用bloom filter进行细粒度行级过滤的阈值，仅在**runtime_filter_type**设置值大于等于Bloom_filter时生效。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

默认值：0.01

说明

- 适用场景：列存表的HASH JOIN，且满足：内表estimate_join_rows/外表estimate_join_rows <= runtime_filter_ratio。细粒度行级过滤仅推荐在join场景下内表外表数据量差距明显时使用，不合理的runtime_filter_ratio设置可能会导致join场景性能劣化。
- 使用限制：细粒度行级过滤仅支持join字段类型为SMALLINT、INTEGER、BIGINT、FLOAT类型时使用。

runtime_filter_cost_options

参数说明: 控制在生成计划时，是否根据cost选择生成应用runtime filter计划。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

- apply_partial：build端只要包含probe端某个runtime filter需要的表，就可以生成runtime filter path。
- apply_all：限制build端必须包含probe端可应用的runtime filter需要的所有表，才可以生成runtime filter path。

默认值：''，即生成计划时不考虑根据cost选择应用runtime filter的计划。

说明

若apply_partial和apply_all同时设置，则以apply_all设置生效。

 版权所有 © 华为云计算技术有限公司

enable_extrapolation_stats

参数说明: 标识是否允许基于历史统计信息使用推理估算的逻辑。使用该逻辑对于未及时收集统计信息的表可以增大估算准确的可能性，但也存在错误推理导致估算过大的可能性。

参数类型：USERSET

取值范围：布尔型

- on表示允许基于历史统计信息使用推理估算的逻辑。
- off表示不允许基于历史统计信息使用推理估算的逻辑。

默认值：on

说明

- 新安装集群场景下：默认值为on。
- 升级场景下：若目标版本为9.1.1.300及以上集群版本，默认值为on；若目标版本低于9.1.1.300集群版本，则默认值为off；若设置过该参数，则保持原有设置，以确保前向兼容。

autoanalyze

参数说明: 标识是否允许在生成计划的时候，对于“统计信息完全缺失”或“修改量达到analyze阈值”的表进行统计信息自动收集，当前不支持对外表触发autoanalyze，不支持对带有ON COMMIT [DELETE ROWS|DROP]选项的临时表触发autoanalyze，如需收集，需用户手动执行analyze操作。如果在auto analyze某个表的过程中数据库发生异常，当数据库正常运行之后再执行语句有可能仍提示需要收集此表的统计信息。此时需要用户对该表手动执行一次analyze操作，以同步统计信息数据。

参数类型：USERSET

取值范围：布尔型

- on表示允许自动进行统计信息收集。
- off表示不允许自动进行统计信息收集。

默认值：on

说明

表的修改量达到analyze阈值是指：表的修改量超过autovacuum_analyze_threshold + autovacuum_analyze_scale_factor * reltuples，其中reltuples是pg_class中记录的表的估算行数。

enable_analyze_partition

参数说明: 控制是否支持对表的某个分区收集统计信息。

开启该参数后，可以通过ANALYZE table_name PARTITION（partition_name）收集某个分区的统计信息，查询表的这个分区上的数据时，优化器会选择使用分区统计信息。

该参数开启后，分区表的统计信息获取方式将发生变化。如需开启请联系技术支持。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示支持对表的某个分区收集统计信息。
- off表示不支持对表的某个分区收集统计信息。

默认值：off

analyze_use_dn_correlation

参数说明：设置ANALYZE时CN是否使用DN的相关系数统计信息。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示CN使用DN的相关系数统计信息。
- off表示CN不使用DN的相关系数统计信息。

默认值：on

analyze_predicate_column_threshold

参数说明：控制是否开启谓词列ANALYZE及限定支持的最小列数。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0～10000

- 0，表示关闭谓词列ANALYZE，不会收集谓词列以及对谓词列进行ANALYZE。
- 大于0，表示开启谓词列收集功能，且仅对列数大于等于此值的表进行谓词列ANALYZE。

默认值：0

enable_runtime_analyze_concurrent

参数说明：控制是否支持对一个表进行并发RUNTIME ANALYZE。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示支持并发。
- off表示不支持并发。

默认值：off

说明

新安装的9.1.2及以上集群版本中，该参数的默认值为off。升级场景下该参数的默认值为保持前向兼容维持原值。

analyze_max_columns_count

参数说明：控制ANALYZE能支持最大的列数。该参数仅9.1.0.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，-1～10000

- -1，表示不限制ANALYZE的列数。
- 大于-1，表示将只收集此值之前的列，超过的列不收集。

默认值：-1

query_dop

参数说明: 用户自定义的查询并行度。

参数类型：USERSET

取值范围：整型，-64～64

- [1,64]：打开固定SMP功能，系统会使用固定并行度。
- 0：打开SMP自适应功能，系统会根据资源情况和计划特征动态为每个查询选取[1,8]之间（x86平台），[1,64]之间（鲲鹏平台）的最优的并行度。
- [-64,-1]：打开SMP自适应功能，并限制自适应选取的最大并行度。

默认值：1（64GB及以上内存规格默认值为0）

说明

- 对于短查询为主的TP类业务中，如果不能通过CN轻量化或下发语句进行业务的调优，则生成SMP计划的时间较长，建议设置query_dop=1。对于AP类复杂语句的场景，建议设置query_dop=0。
- 在开启并行查询后，请保证系统CPU、内存、网络、I/O等资源充足，以达到良好效果。
- 为了避免用户设置不合理的过大值造成性能劣化，系统会计算出该DN可用最大CPU核数，并以此来作为query_dop的上限。如果用户设置query_dop超过4并且同时超过该上限，那么系统会重置query_dop为该上限值。
- 对于GDS外表，在dop开启时，GDS服务端连接数会增加，GDS进程连接数过多时，可能会导致GDS性能下降或者连接超时报错等问题。建议合理规划GDS服务器外表，确保GDS机器资源充足。

query_dop_ratio

参数说明: 用于当query_dop取值为0时，设置在系统中给定的最优dop基础上，调整dop的倍数。即最终dop=系统设置dop * query_dop_ratio（最小值为1，最大值为64）。当设置为1时，表示不调整。

参数类型：USERSET

取值范围：浮点型，0-64

默认值：1

query_dop_option

参数说明: 用于当query_dop取值为0时，调整初始dop的大小。该值越小时，初始dop越大。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：字符串，0-1

 版权所有 © 华为云计算技术有限公司

默认值：0.3

debug_group_dop

参数说明：当query_dop取值为0时，针对生成的执行计划划分的以Stream算子为顶点的group，均分配统一的dop并行度。此参数用于人为指定特定group的dop进行性能调优，格式为G1,D1,G2,D2,...，其中：G1,G2为group的ID，可以从日志中获得，D1,D2为指定的dop值，可以为任意正整数。

参数类型：USERSET

取值范围：字符型

默认值：空

说明

该参数仅供内部调优使用，不允许用户进行设置，建议保持默认值。

scan_dop

参数说明：控制基表扫描并行度。

参数类型：USERSET

取值范围：整型，0~64

- 0，表示由系统自动调整的并行度进行表扫描。
- 大于0，表示强制使用指定的并行度进行表扫描。

默认值：0

enable_analyze_check

参数说明: 标识是否允许在生成计划的时候，对于在pg_class中显示reltuples和relpages均为0的表，检查该表是否曾进行过统计信息收集。**该参数8.1.3及以上集群版本中已废弃，为兼容历史版本功能保留该参数，设置不会生效**。

参数类型：SUSET

取值范围：布尔型

- on表示允许检查。
- off表示不允许检查。

默认值：on

enable_sonic_hashagg

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Agg算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Agg算子。
- off表示不使用面向列的hash表设计的Hash Agg算子。

 版权所有 © 华为云计算技术有限公司

默认值：on

说明

- 在开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，查询对应的Hash Agg算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景(**enable_codegen**打开后获得较大性能提升)，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Aggregation”，而未达到该约束条件时，算子名称将显示为“Hash Aggregation”。

enable_sonic_hashjoin

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Join算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Join算子。
- off表示不使用面向列的hash表设计的Hash Join算子。

默认值：on

说明

- 当前开关仅适用于Inner Join的场景。
- 在开启enable_sonic_hashjoin，查询对应的Hash Inner算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashjoin，且查询达到约束条件使用基于面向列的hash表设计的Hash Join算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Join”，而未达到该约束条件时，算子名称将显示为“Hash Join”。

enable_sonic_optspill

参数说明：标识是否优化sonic场景下HashJoin或者HashAgg的下盘文件个数。仅在enable_sonic_hashjoin或者 enable_sonic_hashagg开启情况下生效。

参数类型：USERSET

取值范围：布尔型

- on表示开启下盘文件数优化。
- off表示关闭下盘文件数优化。

默认值：on

说明

满足sonic条件下的HashJoin或者HashAgg算子，在关闭此参数（设置为off）时每列会产生1个下盘文件。开启此参数（设置为on）时如果不同列数据类型相似，只会有1个下盘文件（最多5个文件）。

expand_hashtable_ratio

参数说明：控制Hash Agg和Hash Join算子执行过程中hash表的大小扩大比例。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：浮点型，0, 0.5~10

默认值：0

说明

- 默认值设置为0时，表示hash表大小会根据当前内存进行自适应扩展。
- 默认值设置为0.5~10之间时，显式的是指定hash表扩大的倍数，通常hash表越大性能越好，但会占用更多内存空间，在内存不足场景可能造成数据提前下盘，带来性能劣化。

enable_combine_hash

参数说明：控制hash join/hash agg算子是否使用combine hash算法优化。该参数仅9.1.1.200及以上集群版本支持。

该参数的生效前提为：group by列、join关联列或join probe侧非关联列都为多列。

参数类型：SUSET

取值范围：布尔型

- on表示使用combine hash算法优化。
- off表示不使用combine hash算法优化。

默认值：on

说明

如果在执行计划的Performance相关算子中，显示COMBINE关键字（COMBINE_HASH_AGG、COMBINE_HASH_JOIN_KEYPART或COMBINE_HASH_JOIN_NON_KEYPART），则表明优化生效。

plan_cache_mode

参数说明：标识在prepare语句中，选择生成执行计划的策略。

参数类型：USERSET

取值范围：枚举类型

- auto表示按照默认的方式选择custom plan或者generic plan。
- force_generic_plan表示强制走generic plan。
- force_custom_plan表示强制走custom plan。

默认值：auto

 版权所有 © 华为云计算技术有限公司

说明

- 此参数只对prepare语句生效，一般用在prepare语句中参数化字段存在比较严重的数据倾斜的场景下。
- custom plan是指对于prepare语句，在执行execute的时候，把execute语句中的参数嵌套到语句之后生成的计划。custom plan会根据execute语句中具体的参数生成计划，这种方案的优点是每次都按照具体的参数生成优选计划，执行性能比较好；缺点是每次执行前都需要重新生成计划，存在大量的重复的优化器开销。
- generic plan是指对于prepare语句生成计划，该计划策略会在执行execute语句的时候把参数bind到plan中，然后执行计划。这种方案的优点是每次执行可以省去重复的优化器开销；缺点是当bind参数字段上数据存在倾斜时该计划可能不是最优的，部分bind参数场景下执行性能较差。

wlm_query_accelerate

参数说明：标识在短查询加速打开时，查询是否需要加速。

参数类型：USERSET

取值范围：整型，-1~1

- -1，短查询由快车道管控，长查询由慢车道管控。
- 0，查询不加速，短查询和长查询均由慢车道管控。
- 1，查询加速，短查询和长查询均由快车道管控。

默认值：-1

show_unshippable_warning

参数说明：标识是否将语句不下推的告警打印到客户端。

参数类型：USERSET

取值范围：布尔型

- on表示将语句不下推的原因以WARNING打印到日志和客户端
- off表示仅将语句不下推的原因以LOG打印到日志

默认值：off

hashjoin_spill_strategy

参数说明：选择hashjoin下盘策略。该参数仅对向量化HashJoin生效。

参数类型：USERSET

取值范围：整型，0～6

- 0：当内表较大且无法在数据库可用内存放下所有数据时，会将数据划分成不同的子分区，直到多次划分后无法分开且仍无法在内存放下所有数据时，尝试外表是否可以放到可用内存中建立哈希表。若外表可以放到可用内存中建立哈希表，则执行HashJoin。反之，则执行NestLoop。
- 1：当内表较大且无法在数据库可用内存放下所有数据时，会将数据划分成不同的子分区，直到多次划分后无法分开且仍无法在内存放下所有数据时，尝试外表是否可以放到可用内存中建立哈希表。如果内外表均很大，强制执行HashJoin。

 版权所有 © 华为云计算技术有限公司

- 2：当内表较大，并且多次下盘无法分开时，强制执行HashJoin。
- 3：当内表较大，并且多次下盘无法分开时，尝试外表是否可以放到数据库可用内存建立哈希表。如果内外表均很大，则报错。
- 4：当内表较大，并且多次下盘无法分开时，则报错。
- 5：当内表较大且无法在数据库可用内存放下所有数据时，如果外表数据可以放到内存中，则使用外表建立哈希表执行HashJoin。如果外表数据无法存放到内存中，则将数据划分成不同的子分区，直到内外表多次划分均无法分开时，执行NestLoop。
- 6：当内表较大且无法在数据库可用内存放下所有数据时，如果外表数据可以放到内存中，则使用外表建立哈希表执行HashJoin。如果外表数据无法存放到内存中，则将数据划分成不同的子分区，直到内外表多次划分均无法分开时，强制执行HashJoin。

默认值：1

⚠ 注意

- 9.1.1.300及以上集群版本中（包括新安装及升级场景），该参数的默认值为1；9.1.1.300以下集群版本中，该参数的默认值为0。
- 此参数对于数据distinct值很小且数据量很大的场景，可能出现无法下盘导致使用内存过大产生内存不受控的问题。 取值0时通过尝试内外表交换或者Nestloop可以避免出现此类内存不受控问题。 执行Nestloop可能造成某些场景性能劣化。遇到此种场景，该参数可取值1、2、6强制执行HashJoin。
- 取值0对向量化Full Join不生效，行为与取值1相同。只尝试外表是否可建立哈希表，不执行NestLoop。
- 取值5和6相对于取值0和1的优势是：如果内表数据量大到无法直接放于可用内存中，但是外表可以，则直接使用外表进行Hashjoin，减少后续多次下盘划分数据的时间消耗。当外表数据distinct较少时，使用外表建立哈希表可能导致性能劣化。此时可以将参数取值调整到0或者1。

max_streams_per_query

参数说明：控制查询计划中Stream节点的数目。（该参数仅8.1.1及以上集群版本支持）

参数类型：SUSET

取值范围：整型，-1～10000

- -1，表示查询计划中Stream节点数目无限制。
- 0~10000，表示查询计划中Stream节点数目超过设定值后报错，查询计划不会被执行。

说明

- 此参数只控制DN上的Stream节点，不考虑CN上的Gather节点。
- 此参数不影响Explain查询计划，但是对Explain analyze和Explain performance有影响。

默认值：-1

版权所有 © 华为云计算技术有限公司

enable_agg_limit_opt

参数说明：标识是否对select distinct col from table limit N 场景优化，其中N小于16384时生效, table为列存表。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启该优化。开启后可保证查询结果来自不同DN，且各个DN无需全部建立全量hash表，可显著提升查询性能。
- off表示关闭该优化。

默认值：on

stream_ctescan_pred_threshold

参数说明：当**enable_stream_ctescan**开启、CTE中仅包含单表过滤时，控制CTE中至少包含的过滤条件个数。大于等于该值时使用share scan方式执行；小于该值时将使用inline方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：2

stream_ctescan_max_estimate_mem

参数说明：当**enable_stream_ctescan**开启时，控制CTE的最大估算内存值。该参数需与**stream_ctescan_refcount_threshold**共同使用。当CTE的估算内存大于stream_ctescan_max_estimate_mem且引用次数小于stream_ctescan_refcount_threshold时，将使用inline方式执行；反之，则使用sharescan方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，32768~INT_MAX，单位为KB。

默认值：256MB

stream_ctescan_refcount_threshold

参数说明：当enable_stream_ctescan开启时，控制CTE的最大引用次数。该参数需与**stream_ctescan_max_estimate_mem**共同使用。当CTE的估算内存大于stream_ctescan_max_estimate_mem且引用次数小于stream_ctescan_refcount_threshold时，将使用inline方式执行；反之，则使用share scan方式执行。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX

默认值：4

 版权所有 © 华为云计算技术有限公司

说明

该参数仅在取值大于0时生效，取值为0时仅依赖stream_ctescan_max_estimate_mem控制inline行为。

inlist_rough_check_threshold

参数说明：当**enable_csqual_pushdown**开启时、过滤条件为IN进行rough check条件下推时，控制IN条件中值的个数的最大值。IN过滤条件中值的个数超过该参数时，将获取IN条件中值的最大/最小值进行条件下推。

参数类型：USERSET

取值范围：整型，0～10000

默认值：100

说明

如果IN条件在表的单列分布列上进行，可以在DN进行值的过滤，此时IN条件中值的个数的最大值为inlist_rough_check_threshold的DN倍。

enable_array_optimization

参数说明：控制是否将IN/ANY/ALL条件生成的Array类型拆分为普通表达式执行。此优化将支持向量化执行、rough check剪枝、分区剪枝等多重优化。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示进行Array类型表达式的拆分优化。
- off表示不进行Array类型表达式的拆分优化。

默认值：on

max_skew_num

参数说明：控制优化器允许进行重分布优化的倾斜值个数。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX

默认值：10

enable_dict_plan

参数说明：控制优化器字典编码的总开关，即是否字典编码来加快group by、filter等查询速度。该参数仅8.3.0及以上集群支持。

参数类型：USERSET

取值范围：布尔型

- on表示打开优化器字典总开关。

 版权所有 © 华为云计算技术有限公司

- off表示关闭优化器字典总开关。

默认值：off

dict_plan_distinct_limit

参数说明：用于设置表的某一列的distinct值，小于等于此阈值才会构建字典编码。该参数仅8.3.0及以上集群支持。

参数类型：USERSET

取值范围：0~INT_MAX

默认值：10000

说明

构建字典编码需同时满足dict_plan_distinct_limit和dict_plan_duplicate_ratio参数阈值。

dict_plan_duplicate_ratio

参数说明：用于设置表的某一列的重复率大小，大于等于此阈值才会构建字典编码。字典的典型场景是该列distinct很少，重复率很大。该参数仅8.3.0及以上集群支持。

参数类型：USERSET

取值范围：0.0~100，单位：百分比

默认值：90

说明

构建字典编码需同时满足dict_plan_distinct_limit和dict_plan_duplicate_ratio参数阈值。

enable_cu_predicate_pushdown

参数说明：控制简单过滤条件是否下推到CU来过滤。启用后，能够普遍提升查询性能，尤其是在涉及bitmap_columns列和pck排序列时性能会显著提升。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示简单过滤条件下推到CU来过滤。
- off表示简单过滤条件不下推到CU来过滤。

默认值：on

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- 该参数适用于特定的WHERE条件、IS NULL条件、IN条件等场景。
  1. 支持的列类型：
     1. 整数类型：INT2、INT4、INT8
     2. 日期和时间类型：DATE、TIMESTAMP、TIMESTAMPTZ
     3. 字符串类型：VARCHAR、TEXT
     4. 数值类型：NUMERIC（长度在19以内）
  2. 支持查询条件：该功能支持多种 WHERE 表达式，主要包括：
     1. **IN 表达式**：用于匹配多个值
     2. **IS NULL / IS NOT NULL 条件**：检查列值是否为空或非空
     3. **比较表达式**：如大于（>）、小于（<）、等于（=）、不等于（<>）等条件，用于范围查询和精确匹配。
- 字典列中的简单过滤条件指的是等值（“=”）运算、IN表达式及is (not) null。因为此过滤条件下推到存储层在CU填充VectorBatch时提前过滤，称此过滤为CU Predicate Filter。

force_late_read_mode

参数说明：控制行列混存表（storage_mode = mix) 的读取方式。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- auto，以CUID为单位自适应选择（通常性能最佳）。
- row，强制使用行存模式读取数据。
- col，强制使用列存模式读取数据。

默认值：auto

external_partition_pruning_method

参数说明：控制external schema分区外表查询时采用的分区剪枝方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- partition_list，全量获取lakeformation/hivemetastore的分区信息，然后进行本地过滤。
- partition_folder，不从lakeformation/hivemetastore获取分区信息，直接获取外表在文件系统上的分区目录，根据分区目录进行剪枝。
- url_filter，根据分区查询条件减少从lakeformation/hivemetastore获取的分区信息数量，然后进行本地过滤。

默认值：partition_list

 版权所有 © 华为云计算技术有限公司

info_constraint_options

参数说明：用于设置可创建的信息约束。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none，表示不支持任何信息约束的创建。
- foreign_key，表示支持创建外键约束。
- soft，表示支持创建informational的主键、唯一约束和外键约束。需显式使用创建informational约束的语法。该取值仅9.1.1.100及以上集群版本支持。
- force_soft，表示支持创建informational的主键、唯一约束和外键约束。不显式使用创建informational约束的语句也可创建成功。该取值仅9.1.1.100及以上集群版本支持。

默认值：soft

⚠ 注意

- 9.1.1.100之前的集群版本中该参数的默认值为none，9.1.1.100及以上集群版本中该参数的默认值为soft。
- 9.1.1.100及以上集群版本中设置该参数值为soft或force_soft时：
  - 创建的主键或唯一约束，即不要求其必须包含分布键和分区键，不会自动创建对应的索引，也不会对现有数据进行约束检查。
  - 创建外键时，如果引用分区表时，系统不会报错。

operator_memory_options

参数说明：设置算子内存估算相关参数。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none，表示不进行任何算子内存估算控制。
- count_distinct_sort_agg，表示支持使用sortagg方式估算agg distinct的内存。

默认值：count_distinct_sort_agg

enable_const_numeric_division_opt

参数说明：针对商是**有限小数**的除法运算场景，控制是否自动将类似a/2转换为a*0.5的形式，提升除法计算效率。该参数仅9.1.1.200及以上集群版本支持。

当除法结果为**无限小数**时（例如，a/3 ≈ 0.333...），参数不会生效，仍按常规除法计算。

参数类型：USERSET

取值范围：布尔型

- on表示自动将除法转乘法优化。

版权所有 © 华为云计算技术有限公司

- off表示不进行除法转乘法优化。

默认值：on

说明

新安装的9.1.1.200及以上版本集群中，该参数默认值为on。升级场景为保持前向兼容该参数的默认值为off。

optimizer_options

参数说明：计划生成配置选项。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

- none，表示不进行任何配置。
- subplan_hashjoin_opt，表示子查询场景中优化hashjoin关联顺序。
- material_initplan，表示对initplan进行提前物化。
- skip_expired_stats，表示无表级统计信息时，不使用过期的列级、多列或表达式统计信息。
- force_disable_cost，控制对nestloop路径增加惩罚代价。当**enable_nestloop**为false，设置该选项时，即使在无其他join路径可选的情况下，优化器仍会对nestloop路径增加惩罚代价。该选项仅9.1.1.300及以上集群版本支持。
- stream_add_sort_cost_enhance，控制**cost_model_version**=1时stream增加sort cost的计算方式。该选项仅9.1.1.300及以上集群版本支持。
- keep_window_clause_order，控制是否保持SQL中窗口函数的原始顺序。默认情况下，优化器会对窗口函数重新排序以提升性能。开启此选项后，窗口函数将严格按照SQL中的顺序执行。该选项仅9.1.2及以上集群版本支持。
- histogram_extrapolation，控制当查询条件中的值超出已知数据统计范围时，是否进行外推估算。例如，直方图记录的数据范围是1~100，但查询条件为大于120，开启后优化器会基于已有分布趋势进行合理估算，而非直接给出一个固定的粗略值。该选项仅9.1.2及以上集群版本支持。
- skip_index_correlation，控制优化器在选择索引时是否考虑列之间的相关性。开启后，优化器将跳过相关性计算过程，直接基于其他统计信息选择索引，可加快计划生成速度。适用于列之间相关性较弱或无需精确评估的场景。该选项仅9.1.2及以上集群版本支持。
- cstore_cost_by_cu，控制列存扫描代价是否按CU粒度计算扫描的页面数。开启后先由行级选择率估算CU数量，再乘以每个CU对应的页面数，选择率极小时也至少保留一个CU的页面数，避免代价被低估。该选项仅9.1.2及以上集群版本支持。
- roughcheck_bucketid_cost选项，控制在列存表顺序扫描代价估算中是否增加BucketId RoughCheck过滤效果，仅对3.0表和二级分区表生效。开启后顺序扫描代价估算更准确。该选项仅9.1.2及以上集群版本支持。
- cross_vw_wlm_mem，控制跨VW查询是否受WLM内存机制管控。该选项仅9.1.2及以上集群版本支持。
- boundary_equal_selectivity，控制对“>=、<=、>、<”这类边界条件的行数估算方式。开启后，优化器会针对边界值等于某个具体数值的情况进行更精确的估算，从而生成更优的执行计划。关闭时使用原有估算逻辑。该选项仅9.1.2及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

- limit_scan_cost选项，用于优化分页排序+过滤类查询的执行计划选择。当查询同时带有LIMIT和ORDER BY，并且过滤条件涉及的列不在排序所依赖的索引上时，数据库按索引顺序取数的过程中需要逐行判断过滤条件，实际扫描的数据量可能远大于分页最终需要的数据量。开启该选项后，优化器会根据过滤条件的筛选情况估算实际扫描代价，优先选择整体代价更低的查询计划。该选项仅9.1.2及以上集群版本支持。
- notin_multiplication_sel，控制NOT IN运算行数估算策略。开启后，对not in运算行数估算时，使用乘法来计算多罕见值的比例。关闭时使用原有估算逻辑。该选项仅9.1.2及以上集群版本支持。
- correlated_join_selectivity，控制等值连接中相关列是否使用最小的列选择率替代选择率乘积估算行数。开启后，等值连接中相关列选择率取各列选择率的最小值。关闭时，保持原有选择率相乘逻辑。该选项仅9.1.2及以上集群版本支持。

默认值：subplan_hashjoin_opt

说明

新安装的9.1.1.200及以上版本集群中，该参数默认值为subplan_hashjoin_opt。升级场景为保持前向兼容该参数的默认值为none。

18.10 错误报告和日志

18.10.1 记录日志的位置

log_statement_length_limit

参数说明：控制单独打印的SQL语句的长度。超过设置长度的SQL语句，单独打印到statement-*时间戳*.log中。该参数仅9.1.0.200及以上版本支持。

参数类型：SUSET

取值范围：整型，0~INT_MAX。

默认值：1024

18.10.2 记录日志的时间

client_min_messages

参数说明：控制发送到客户端的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，发送给客户端的消息就越少。

参数类型：USERSET

须知

当client_min_messages和**log_min_messages**取相同值时，其值所代表的级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error。参数的详细信息请参见**表18-1**。

 版权所有 © 华为云计算技术有限公司

默认值：notice

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

参数类型：SUSET

须知

当**client_min_messages**和log_min_messages取相同值log时所代表的消息级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表18-1**。

默认值：warning

log_min_error_statement

参数说明：控制在服务器日志中记录错误的SQL语句。

参数类型：SUSET

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表18-1**。

说明

- 设置为error ，表示导致错误、日志消息、致命错误、panic的语句都将被记录。
- 设置为panic，表示关闭此特性。

默认值：error

log_min_duration_statement

参数说明：当SQL语句的执行时间大于等于该参数设置的时间时，系统会自动把该语句及其实际执行时间记录到数据库日志中。通过该参数可跟踪需要优化的查询语句。对于使用扩展查询协议的客户端，语法分析、绑定、执行各步骤的耗时将被分别记录。

参数类型：SUSET

取值范围：整型，-1 ~ INT_MAX，单位为毫秒（ ms ）。

- -1，表示关闭此参数功能。
- 0，表示记录所有执行的SQL语句及其耗时。
- 大于0，表示根据参数设定时间阈值进行记录。例如，设置为1000时，表示记录执行时间超过1000毫秒的所有SQL语句。

默认值：30min

 版权所有 © 华为云计算技术有限公司

说明

当该参数与**log_statement**同时使用时，已经被log_statement记录的语句文本不会被重复记录。在没有使用syslog情况下，推荐使用log_line_prefix记录PID或会话ID，以便将当前语句与其最终的持续时间消息关联。

backtrace_min_messages

参数说明：控制当产生该设置参数级别相等或更高级别的信息时，会打印函数的堆栈信息到服务器日志文件中。

参数类型：SUSET

取值范围：枚举类型

有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表18-1**。

默认值：panic

注意

该参数用于在实际业务场景中定位问题。由于频繁打印函数栈会增加系统开销并影响稳定性，建议不要将该参数设置为fatal及panic以外的级别。

表18-1解释DWS中使用的消息安全级别。当日志输出到syslog或者eventlog时，DWS进行如表中的转换。

表 18-1 信息严重程度分类

信息严重程度类型	详细说明	系统日志	事件日志
debug[1-5]	报告详细调试信息。	DEBUG	INFORMATION
log	报告对数据库管理员有用的信息， 比如检查点操作统计信息。	INFO	INFORMATION
info	报告用户可能需求的信息， 比如在VACUUM VERBOSE过程中的信息。	INFO	INFORMATION
notice	报告可能对用户有帮助的信息， 比如， 长标识符的截断， 作为主键一部分创建的索引等。	NOTICE	INFORMATION
warning	报告警告信息， 比如在事务块范围之外的COMMIT。	NOTICE	WARNING
error	报告导致当前命令退出的错误。	WARNING	ERROR

版权所有 © 华为云计算技术有限公司

信息严重程度类型	详细说明	系统日志	事件日志
fatal	报告导致当前会话终止的原因。	ERR	ERROR
panic	报告导致整个数据库被关闭的原因。	CRIT	ERROR

plog_merge_age

参数说明：该参数用于控制性能日志数据输出的周期。

参数类型：USERSET

取值范围：0~INT_MAX，单位为毫秒（ ms ）。

- 0，表示当前会话不再输出性能日志数据。
- 大于0，表示当前会话按照指定的时间周期进行输出性能日志数据。该参数设置的越小，输出的日志数据越多，对性能的负面影响越大。

默认值：3s

说明

- 该参数以毫秒为单位，建议在使用过程中设置值为1000的整数倍，即设置值以秒为最小单位。
- 该参数所控制的性能日志文件以prf为扩展名，文件放置在$GAUSSLOG/gs_profile/<node_name> 目录下面，其中node_name是由postgres.conf文件中的pgxc_node_name的值。
- 不建议用户使用该参数。

profile_logging_module

参数说明：用于设置记录性能日志的类型，使用该参数时需确保plog_merge_age参数值非0。该参数属于会话级参数，不建议通过gs_guc工具来设置。仅8.1.3及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

默认值： ALL,on(OBS,HADOOP,REMOTE_DATANODE),off(MD)

设置方法：通过SHOW profile_logging_module;查看支持设置的模块。例如，查询输出结果为：

```
SHOW profile_logging_module;
profile_logging_module
-----------------------------------------------
ALL,on(OBS,HADOOP,REMOTE_DATANODE),off(MD)(1 row)
```

打开MD性能日志，并查看设置结果。ALL表示对所有模块的日志输出进行整体开启或关闭。

```
SET profile_logging_module='on(md)';
SET
```

 版权所有 © 华为云计算技术有限公司

```
SHOW profile_logging_module;
profile_logging_module
-----------------------------------------------
ALL,on(MD,OBS,HADOOP,REMOTE_DATANODE),off()(1 row)
```

18.10.3 记录日志的内容

debug_pretty_print

参数说明：设置此选项对debug_print_parse、debug_print_rewritten和debug_print_plan产生的日志进行缩进，会生成易读但比设置为off时更长的输出格式。

参数类型：USERSET

取值范围：布尔型

- on表示进行缩进。
- off表示不进行缩进。

默认值：on

log_duration

参数说明：控制记录每个已完成SQL语句的执行时间。对使用扩展查询协议的客户端、会记录语法分析、绑定和执行每一步所花费的时间。

参数类型：SUSET

取值范围：布尔型

- 设置为off ，该选项与**log_min_duration_statement**的不同之处在于log_min_duration_statement强制记录查询文本。
- 设置为on并且log_min_duration_statement大于零，记录所有持续时间，但是仅记录超过阈值的语句。这可用于在高负载情况下搜集统计信息。

默认值：on

log_error_verbosity

参数说明：控制服务器日志中每条记录的消息写入的详细度。

参数类型：SUSET

取值范围：枚举类型

- terse输出不包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录。
- verbose输出包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。
- default输出包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录，不包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。

默认值：default

 版权所有 © 华为云计算技术有限公司

log_lock_waits

参数说明：当一个会话的等待获得一个锁的时间超过**deadlock_timeout**的值时，此选项控制在数据库日志中记录此消息。这对于决定锁等待是否会产生一个坏的行为是非常有用的。

参数类型：SUSET

取值范围：布尔型

- on表示记录此信息。
- off表示不记录此信息。

默认值：off

log_statement

参数说明：控制记录SQL语句。对于使用扩展查询协议的客户端，记录接收到执行消息的事件和绑定参数的值（内置单引号要双写）。

参数类型：SUSET

须知

即使log_statement设置为all，包含简单语法错误的语句也不会被记录，因为仅在完成基本的语法分析并确定了语句类型之后才记录日志。在使用扩展查询协议的情况下，在执行阶段之前（语法分析或规划阶段）同样不会记录。将log_min_error_statement设为ERROR或更低才能记录这些语句。

取值范围：枚举类型

- none表示不记录语句。
- ddl表示记录所有的数据定义语句，比如CREATE、ALTER和DROP语句。
- mod表示记录所有DDL语句，还包括数据修改语句INSERT、UPDATE、DELETE、TRUNCATE和COPY FROM 。
- all表示记录所有语句，PREPARE、EXECUTE和EXPLAIN ANALYZE语句也同样被记录。

默认值：none

log_temp_files

参数说明：控制记录临时文件的删除信息。临时文件可以用来排序、哈希及临时查询结果。当一个临时文件被删除时，将会产生一条日志消息。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB

- 正整数表示只记录比log_temp_files设定值大的临时文件的删除信息。
- 0表示记录所有的临时文件的删除信息。
- -1表示不记录任何临时文件的删除信息。

默认值：-1

 版权所有 © 华为云计算技术有限公司

logging_module

参数说明：用于设置或者显示模块日志在服务端的可输出性。该参数属于会话级参数，不建议通过gs_guc工具来设置。

> ⚠ **注意**
>
> **此参数主要用于日志定位，非必要不建议用户配置，否则会导致集群不可预知异常，需要配置此参数，请联系技术人员确认。**

参数类型：USERSET

取值范围：字符串

默认值：所有模块日志在服务端是不输出的，可由SHOW logging_module查看。

设置方法：首先，可以通过SHOW logging_module来查看哪些模块是支持可控制的。例如，查询输出结果为：

```
show logging_module;
logging_module
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
ALL,on(),off(DFS,GUC,HDFS,ORC,SLRU,MEM_CTL,AUTOVAC,ANALYZE,CACHE,ADIO,SSL,GDS,TBLSPC,WLM,SPACE,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_SETOP,OPT_CARD,OPT_SKEW,SMP,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,PARQUET,CARBONDATA,SNAPSHOT,XACT,HANDLE,CLOG,TQUAL,EC,REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ,INSTR,COMM_IPC,COMM_PARAM,CSTORE,JOB,STREAMPOOL,STREAM_CTESCAN)
(1 row)
```

支持可控制的模块使用大写来标识，特殊标识ALL用于对所有模块日志进行设置。可以使用on/off来控制模块日志的输出。设置SSL模块日志为可输出，使用如下命令：

```
set logging_module='on(SSL)';
SET
show
logging_module;

                                                  logging_module
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
ALL,on(SSL),off(DFS,GUC,HDFS,ORC,SLRU,MEM_CTL,AUTOVAC,ANALYZE,CACHE,ADIO,GDS,TBLSPC,WLM,SPACE,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_SETOP,OPT_CARD,OPT_SKEW,SMP,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,PARQUET,CARBONDATA,SNAPSHOT,XACT,HANDLE,CLOG,TQUAL,EC,REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ,INSTR,COMM_IPC,COMM_PARAM,CSTORE,JOB,STREAMPOOL,STREAM_CTESCAN)
(1 row)
```

可以看到模块SSL的日志输出被打开。

ALL标识是相当于一个快捷操作，即对所有模块的日志可输出进行开启或关闭。

```
set logging_module='off(ALL)';
SET
show
logging_module;

                          logging_module
```

```
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------

ALL,on(),off(DFS,GUC,HDFS,ORC,SLRU,MEM_CTL,AUTOVAC,ANALYZE,CACHE,ADIO,SSL,GDS,TBLSPC,WLM,SP
ACE,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,
OPT_SETOP,OPT_CARD,OPT_SKEW,SMP,UDF,COOP_ANALYZE,WLMCP,
ACCELERATE,PLANHINT,PARQUET,CARBONDATA,SNAPSHOT,XACT,HANDLE,CLOG,TQUAL,EC,REMOTE,CN_RE
TRY,PLSQL,TEXTSEARCH,SEQ,INSTR,COMM_IPC,COMM_PARAM,CSTORE,JOB,STREAMPOOL,STREAM_CTESCA
N)
(1 row)

set logging_module='on(ALL)';
SET
show
logging_module;

                      logging_module
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------

ALL,on(DFS,GUC,HDFS,ORC,SLRU,MEM_CTL,AUTOVAC,ANALYZE,CACHE,ADIO,SSL,GDS,TBLSPC,WLM,SPACE,
OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_
SETOP,OPT_CARD,OPT_SKEW,SMP,UDF,COOP_ANALYZE,WLMCP,ACCELE
RATE,PLANHINT,PARQUET,CARBONDATA,SNAPSHOT,XACT,HANDLE,CLOG,TQUAL,EC,REMOTE,CN_RETRY,PLS
QL,TEXTSEARCH,SEQ,INSTR,COMM_IPC,COMM_PARAM,CSTORE,JOB,STREAMPOOL,STREAM_CTESCAN),off()
(1 row)
```

所有模块中，COMM_IPC必须显式的打开/开闭，执行以下命令都可以将该模块的日志打开：

```
set logging_module='on(ALL)';
SET
set logging_module='on(COMM_IPC)';
SET
```

设置成功后，COMM_IPC模块日志不会自动关闭，关闭COMM_IPC模块的日志，必须手动执行关闭命令，以下两条命令都可以将该模块日志关闭：

```
set logging_module='off(ALL)';
SET
set logging_module='off(COMM_IPC)';
SET
```

依赖关系：该参数依赖于**log_min_messages**参数的设置。

enable_unshipping_log

参数说明：用于控制是否打印语句不下推的日志，主要用于定位不下推语句可能导致的性能问题。

参数类型：USERSET

取值范围：布尔型

- on表示打印日志。
- off表示不打印日志。

默认值：on

版权所有 © 华为云计算技术有限公司

log_statement_filter_list

参数说明：控制记录SQL语句。设置错误码的集合，多个错误码通过“，”分隔，例如：GS_001,GS_002。该错误码日志中不打印SQL语句。该参数仅9.1.0.200及以上版本支持。

参数类型：SUSET

取值范围：字符串

默认值：空字符串

18.11 运行时统计

18.11.1 查询和索引统计收集器

查询和索引统计收集器负责收集数据库系统运行中的统计数据，如在一个表和索引上进行了多少次插入与更新操作、磁盘块的数量和元组的数量、每个表上最近一次执行清理和分析操作的时间等。可以通过查询系统视图pg_stats和pg_statistic查看统计数据。以下参数用于设置服务器范围内的统计收集特性。

track_activities

参数说明：控制收集每个会话中当前正在执行命令的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：on

track_counts

参数说明：控制收集数据库活动的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

说明

在autovacuum自动清理进程中选择清理的数据库时，需要数据库的统计数据，故默认值设为on。

默认值：on

track_io_timing

参数说明：控制收集数据库I/O调用时序的统计数据。I/O时序统计数据可以在pg_stat_database中查询。

 版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能，开启时，收集器会重复地去查询当前时间的操作系统，这可能会引起某些平台的重大开销，故默认值设置为off。
- off表示关闭收集功能。

默认值：off

track_functions

参数说明：控制收集函数的调用次数和调用耗时的统计数据。

参数类型：SUSET

须知

当SQL语言函数设置为调用查询的“内联”函数时，不管是否设置此选项，这些SQL语言函数无法被追踪到。

取值范围：枚举类型

- pl表示只追踪过程语言函数。
- all表示追踪SQL和C语言函数。
- none表示关闭函数追踪功能。

默认值：none

update_process_title

参数说明：控制收集每次服务器接收到一个新的SQL语句时产生的进程名称更新的统计数据。

进程名称可以通过ps命令进行查看，在Windows下通过任务管理器查看。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：off

track_thread_wait_status_interval

参数说明：用来定期收集thread状态信息的时间间隔。

参数类型：SUSET

取值范围：整型，0~1440，单位为min。

默认值：30min

 版权所有 © 华为云计算技术有限公司

enable_save_datachanged_timestamp

参数说明：控制是否收集insert/update/delete, exchange/truncate/drop partition操作对表数据改动的时间。

参数类型：USERSET

取值范围：布尔型

- on表示允许收集相关操作对表数据改动的时间。
- off表示禁止收集相关操作对表数据改动的时间。

默认值：on

enable_save_dataaccess_timestamp

参数说明：控制是否记录表的最后一次访问时间。该参数仅8.2.1.210及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许记录表的最后一次访问时间。
- off表示禁止记录表的最后一次访问时间。

默认值：on

说明

新安装的9.1.0.200及以上集群版本中，该参数的默认值为on。升级场景下为保持前向兼容，该参数默认为off。

instr_unique_sql_count

参数说明：控制是否收集整个集群的Unique SQL以及收集数量限制。

参数类型：SIGHUP

取值范围：整型，0～INT_MAX

- 值为0时，表示不收集Unique SQL统计信息。
- 值大于0时，在CN节点上，将会控制收集的Unique SQL数量不超过该设置值。当收集数量达到限制时，不再收集新的Unique SQL，此时可通过reload调大设置值，继续收集新的Unique SQL。

默认值：0

注意

如果新设置值小于原设置值，将会清空对应CN节点已收集的Unique SQL统计信息。需特别注意该清理操作将由资源管理后台线程完成，若GUC参数use_workload_manager为off时清理操作可能失败，可直接使用函数reset_instr_unique_sql进行清理。

版权所有 © 华为云计算技术有限公司

track_sql_count

参数说明：控制对每个会话中当前正在执行的SELECT、INSERT、UPDATE、DELETE、MERGE INTO语句是否进行计数统计，对SELECT、INSERT、UPDATE、DELETE语句的响应时间进行统计，以及对DDL、DML、DCL语句进行计数的统计。

参数类型：SUSET

取值范围：布尔型

- on表示开启统计功能。
- off表示关闭统计功能。

默认值：on

说明

- track_sql_count参数受track_activities约束：
  - track_activities开启而track_sql_count关闭时，如果查询了gs_sql_count、pgxc_sql_count、gs_workload_sql_count、pgxc_workload_sql_count、global_workload_sql_count、gs_workload_sql_elapse_time、pgxc_workload_sql_elapse_time、或global_workload_sql_elapse_time视图，将会有LOG提示track_sql_count是关闭的。
  - track_activities和track_sql_count同时关闭，那么此时将会有两条LOG，分别提示track_activities是关闭的和track_sql_count是关闭的。
  - track_activities关闭而track_sql_count开启，此时将仅有LOG提示track_activities是关闭。
- 当参数关闭时，查询视图的结果为0行。

track_partition_activities

参数说明：控制是否收集数据库中当前正在执行的分区操作相关的信息。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：on

enable_parallel_analyze

参数说明：控制内外表analyze采样时是否使用并行的采样方式。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- true表示内外表analyze采样时使用并行的采样方式。
- false表示内外表analyze采样时不使用并行的采样方式。

默认值：true

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- 开启enable_parallel_analyze的情况下，对外表进行analyze时，尽量避免对目标外表列增加NOT NULL约束条件，防止外表数据源变更时约束失效导致analyze失败；同时目前并行采样不支持物化视图；当发生由于此类原因导致的analyze失败时，可以设置该参数为false，成功执行analyze。
- 目前并行采样仅支持普通列存内表analyze，当内表使用hstore/hstore_opt或声明为复制表时此优化不会生效。
- 目前并行采样仅支持parquet/orc格式存储的外表，当外表为其他格式时此优化不会生效。

parallel_analyze_workers

参数说明：用于设置并行的analyze采样时并发的线程数量。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~64

默认值：10

说明

该参数取值应与集群负载相对应，当集群负载较小时可以根据集群配置适当提高该参数值，进一步增加analyze执行效率。

analyze_sample_multiplier

参数说明：用于设置外表analyze采样的stripe采样率的扩大倍数。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~100，0表示stripe采样率为100%。

默认值：3

tuple_change_sync_threshold

参数说明：设置在DML操作后将修改计数同步给其他CN节点的阈值。该参数仅8.2.1.210及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX

默认值：10000

18.11.2 性能统计

数据库运行过程中可能因锁竞争、磁盘IO过载等问题产生性能瓶颈，通过DWS提供的性能统计参数，可快速定位性能问题。这些参数仅用于辅助数据库管理员进行粗略分析，类似Linux中的操作系统工具getrusage()。

版权所有 © 华为云计算技术有限公司

log_parser_stats

参数说明：控制是否在服务器日志里记录解析器的性能统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

log_planner_stats

参数说明：控制是否在服务器日志里记录查询优化器的性能统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

log_executor_stats

参数说明：控制是否在服务器日志里记录执行器的性能统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

log_statement_stats

参数说明：控制是否在服务器日志里记录整个语句的性能统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

说明

该参数不能和**log_parser_stats**、**log_planner_stats**、**log_executor_stats**参数同时打开。

 版权所有 © 华为云计算技术有限公司

18.12 资源管理

未对数据库资源做控制时，容易出现并发任务抢占资源导致操作系统过载甚至最终崩溃。操作系统过载时，其响应用户任务的速度会变慢甚至无响应；操作系统崩溃时，整个系统将无法对用户提供任何服务。DWS的负载管理功能基于可用资源的多少均衡数据库的负载，以避免数据库系统过载。

space_once_adjust_num

参数说明：空间管控和空间统计功能中，控制慢速构建与细粒度校准操作中每次处理的文件个数阈值。该参数8.1.3及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~INT_MAX

- 0表示不启用慢速构建和细粒度校准功能。

默认值：300

说明

文件个数阈值影响数据库资源，建议合理设置。

default_partition_cache_strategy

参数说明：控制分区缓存的默认策略。该参数8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- cache_each_partition_as_possible表示尽可能地缓存插入的数据，插入时遇到不同分区的数据并不一定将数据刷入CU。
- flush_when_switch_partition表示插入时数据属于不同分区就将数据刷入CU。

默认值：cache_each_partition_as_possible

max_active_statements

参数说明：设置全局的最大并发数量。该参数只在CN上应用且针对一个CN上的执行作业。

数据库管理员需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持最大限度的并发作业，且防止并发执行作业过多，引起系统崩溃。

参数类型：SIGHUP

取值范围：整型，-1 ~ INT_MAX。设置为-1和0表示对最大并发数不做限制。

默认值：60

cgroup_name

参数说明：设置当前使用的Cgroups的名字或者调整当前group下排队的优先级。

版权所有 © 华为云计算技术有限公司

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

参数类型：USERSET

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：字符串

默认值：DefaultClass:Medium

说明

DefaultClass:Medium表示DefaultClass下Timeshare控制组中的Medium控制组。

enable_cgroup_switch

参数说明：是否控制数据库执行语句时根据类型自动切换到TopWD组。

参数类型：USERSET

取值范围：布尔型

- on表示控制数据库执行语句时根据类型自动切换到TopWD组。
- off表示控制数据库执行语句时根据类型不自动切换到TopWD组。

默认值：off

memory_tracking_mode

参数说明：设置记录内存信息的模式。

参数类型：USERSET

取值范围：

- none，不启动内存统计功能。
- normal，仅做内存实时统计，不生成文件。
- executor，生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。
- fullexec，生成文件包含执行层申请过的所有内存上下文信息。

默认值：none

memory_detail_tracking

参数说明：设置需要的线程内分配内存上下文的顺序号以及当前线程所在query的plannodeid。

参数类型：USERSET

取值范围：字符型

默认值：空

 版权所有 © 华为云计算技术有限公司

⚠ 注意

该参数不允许用户进行设置，建议保持默认值。

enable_resource_track

参数说明：设置是否开启资源实时监控功能。该参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开资源监控。
- off表示关闭资源监控。

默认值：on

enable_resource_record

参数说明：设置是否开启资源监控记录归档功能。开启时，对于执行结束的记录，会分别被归档到相应的INFO视图（**GS_WLM_SESSION_INFO**和**GS_WLM_OPERATOR_INFO**）。此参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启资源监控记录归档功能。
- off表示关闭资源监控记录归档功能。

默认值：on

说明

新建集群默认值为on，升级场景该参数的默认值为保持前向兼容维持原值。

enable_track_record_subsql

参数说明：设置是否开启子语句记录归档功能。开启时，存储过程、匿名块内部的子语句会被记录归档到相应的INFO表（**GS_WLM_SESSION_INFO**）。此参数为会话级参数，可在与CN的连接会话中设置生效，仅影响该会话连接中的语句；也可在CN和DN上同时设置，能全局生效。

参数类型：USERSET

取值范围：布尔型

- on表示开启子语句资源监控记录归档功能。
- off表示关闭子语句资源监控记录归档功能。

默认值：on

 版权所有 © 华为云计算技术有限公司

resource_track_subsql_max_depth

参数说明：设置允许记录的最大子语句嵌套层数。此参数为会话级参数，在当前CN连接会话中设置生效，仅影响该会话连接中的语句。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0～INT_MAX

- 值为0时，表示不限制子语句记录层数，记录所有嵌套层级的子语句。
- 值大于0时，表示仅记录指定层数以内的嵌套子语句，超过指定层数的子语句不进行记录归档。

默认值：0

block_rule_cost

参数说明：设置查询过滤器触发的最小代价。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 值为-1时，表示不根据cost判断是否对语句进行拦截，对所有语句进行查询过滤。
- 值为0时，表示对所有cost大于0的语句进行拦截，但特殊类型语句（cost为0）不进行语句拦截。
- 值大于0时，如果语句cost小于block_rule_cost值，则不进行拦截；否则，对语句进行拦截。

默认值：-1

enable_user_metric_persistent

参数说明：设置是否开启用户/资源池历史资源监控转存功能。开启时，对于**PG_TOTAL_USER_RESOURCE_INFO**视图中数据，会定期采样保存到**GS_WLM_USER_RESOURCE_HISTORY**系统表中；对于**GS_RESPOOL_RESOURCE_INFO**视图中数据，会定期采样保存到**GS_RESPOOL_RESOURCE_HISTORY**系统表中。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启用户/资源池历史资源监控转存功能。
- off表示关闭用户/资源池历史资源监控转存功能。

默认值：on

user_metric_retention_time

参数说明：设置用户历史资源监控数据的保存天数。

参数类型：SIGHUP

取值范围：整型，0～3650，单位为天。

 版权所有 © 华为云计算技术有限公司

- 值等于0时，用户历史资源监控数据将永久保存。
- 值大于0时，用户历史资源监控数据将保存对应天数。

默认值：7

resource_track_level

参数说明：设置当前会话的资源监控的等级。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：枚举型

- none，不开启资源监控功能。
- query，开启query级别资源监控功能，开启此功能会把SQL语句的计划信息（类似explain输出信息）记录到TopSQL中。
- perf，开启perf级别资源监控功能，开启此功能会把包含实际执行时间和执行行数的计划信息（类似explain analyze输出信息）记录到TopSQL中。
- operator_realtime, 开启operator级别资源监控功能，开启此功能后会记录实时运行的作业算子信息到TopSQL中，但不会持久化到历史TopSQL中。
- operator，开启operator级别资源监控功能，开启此功能不仅会把包含实际执行时间和执行行数的信息记录到TopSQL中，还会把算子级别执行信息刷新到TopSQL中。

默认值：perf

说明

新安装的9.1.2及以上集群版本中该参数的默认值为perf，升级场景下为保持前向兼容该参数的默认值维持原值。

fast_obs_tablesize_method

参数说明：设置快速计算列存v3和v3 hstore_opt表大小的方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- 0，表示通过list obs文件的方式计算表大小。
- 1，表示wlm后台统计的方式，通过pg_relfilenode_size计算表大小。
- 2，表示若**enable_pg_db_file**是on，则通过遍历**PG_DB_FILE**系统表计算表大小，若enable_pg_db_file是off，则通过cudesc的每个文件的最大偏移和近似估算表大小。

默认值：2

fast_obs_dbsize_method

参数说明：设置快速计算OBS上数据库数据大小的方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：枚举型

- 0，表示通过遍历OBS文件来获得OBS数据库数据大小。
- 1，表示当集群中只有一个数据库时通过OBS桶来获取OBS上数据库数据大小，若存在多个数据则通过遍历OBS文件来获取OBS 数据库数据大小。
- 2，表示如果**enable_pg_db_file**为on，则通过遍历**PG_DB_FILE**系统表来获得OBS数据库数据大小。若enable_pg_db_file为off，则通过遍历OBS文件来获取OBS数据库数据大小。该取值仅9.1.1.200及以上集群版本支持。

默认值：2

说明

新建的9.1.1.200及以上集群版本中该参数的默认值为2。对于升级场景：若手动修改过该参数的默认值，为保持前向兼容维持原值；若未修改过默认值，则升级后该参数的默认值为2。

enable_operator_realtime_monitor

参数说明：设置是否开启实时算子运行状态监控。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启实时算子运行状态监控。
- off表示关闭实时算子运行状态监控。

默认值：on

说明

开启该参数后，可通过**PGXC_THREAD_WAIT_STATUS**和**PG_THREAD_WAIT_STATUS**视图查看当前正在执行的算子信息。当线程不存在其他等待事件时，视图的wait_event字段将显示当前算子在执行计划中的层级以及已处理的行数信息。当线程存在其他实际等待事件时，wait_event字段优先显示对应的等待事件。

resource_track_cost

参数说明：设置对当前会话的语句进行资源监控的最小执行代价。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 值为-1时，不进行资源监控。
- 值大于或等于0时，对执行代价超过该参数值的语句进行资源监控。

默认值：0

说明

新建集群默认值为0，升级场景该参数的默认值为保持前向兼容维持原值。

 版权所有 © 华为云计算技术有限公司

resource_track_duration

参数说明：设置资源监控实时视图（参见**表13-1**）中记录的语句执行结束后进行历史信息转存的最小执行时间。当执行完成的作业，其执行时间不小于此参数值时，作业信息会从实时视图（以statistics为后缀的视图）转存到相应的历史视图（以history为后缀的视图）中。该参数只有当enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：浮点型，0～INT_MAX，单位支持毫秒、秒、分钟、小时、天，默认单位为秒（s）。

- 0，表示资源监控实时视图（**表13-1**）中记录的所有语句都进行历史信息归档。
- 大于0，表示资源监控实时视图（**表13-1**）中记录的语句的执行时间和排队时间之和超过此设置值就会进行历史信息归档。

默认值：0

说明

resource_track_duration参数在升级场景下保持前向兼容。新安装9.1.1.100及以上的9.1.1.xxx集群版本中该参数的默认值为100ms；新安装的9.1.2及以上集群版本中该参数默认值为0。

设置示例如下：

```
SET resource_track_duration='0.1s';
SET resource_track_duration=0.1;
SET resource_track_duration='0.01min'; # 由于该参数的默认单位是秒，且采用四舍五入的方式，因此将0.01min设置为1s，而不是0.6s。
```

resource_track_subsql_duration

参数说明：设置过滤存储过程中子语句的最小执行时间。该参数仅8.2.1及以上集群版本支持。

当存储过程中子语句执行完成时，其执行时间大于此参数设置值时，作业信息会转储到TopSQL归档表中。该参数只有当**enable_track_record_subsql**为on时生效。

参数类型：USERSET

取值范围：浮点型，0～INT_MAX，单位支持毫秒、秒、分钟、小时、天，默认单位为秒（s）。

- 值为0时，存储过程中所有子语句都进行历史信息归档。
- 值大于0时，存储过程中子语句记录的执行时间超过设置值就会进行历史信息归档。

默认值：100ms

说明

resource_track_subsql_duration参数在升级场景下保持前向兼容，新安装的9.1.2及以上集群版本中默认值为100ms，设置示例如下：

```
SET resource_track_subsql_duration='0.1s';
SET resource_track_subsql_duration=0.1;
SET resource_track_subsql_duration='0.01min'; # 由于该参数的默认单位是秒，且采用四舍五入的方式，因此将0.01min设置为1s，而不是0.6s。
```

 版权所有 © 华为云计算技术有限公司

resource_track_application

参数说明：设置TopSQL记录指定客户端下发的语句。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

目前支持设置的客户端包括：gs_rewind、cm_agent、pgxc_clean、gs_clean、gs_running_xacts、OM及wlm。设置多个客户端时，相邻的值需用英文逗号隔开。

默认值：空字符串

query_exception_count_limit

参数说明：设置作业触发异常规则次数的上限。作业触发异常规则次数达到上限后该作业会被自动加入黑名单且禁止再执行，只有移除黑名单后才能恢复运行。

参数类型：USERSET

取值范围：整型，-1～INT_MAX。

- 值为-1时，表示对作业触发异常规则的次数不做限制，即作业触发异常规则次数再多也不会自动将该作业加入黑名单。
- 值大于等于0时，表示作业触发异常规则次数达到阈值时该作业立即被加入黑名单。其中，0和1含义相同，表示只要作业触发异常规则即会将作业加入黑名单。

默认值：-1

disable_memory_protect

参数说明：禁止内存保护功能。当系统内存不足时如果需要查询系统视图，可以先将此参数置为on，禁止内存保护功能，保证视图可以正常查询。该参数只适用于在系统内存不足时进行系统诊断和调试，正常运行时请保持该参数配置为off。

参数类型：USERSET

取值范围：布尔型

- on表示禁止内存保护功能。
- off表示启动内存保护功能。

默认值：off

query_band

参数说明：用于标识当前会话的作业类型，由用户自定义。

参数类型：USERSET

取值范围：字符型

默认值：空

 版权所有 © 华为云计算技术有限公司

wlm_memory_feedback_adjust

参数说明：是否开启动态负载管理中的内存负反馈功能。该参数仅8.2.0及以上集群版本支持。

语句在CN节点上会计算出估算内存，用该值来预占内存资源，语句内存过高估算场景时，语句预占内存资源过大，会引发后续作业排队，导致系统资源利用率下降，内存负反馈机制，会基于DN上的实际使用内存情况，判断如果集群连续一段时间处于高估场景时，会由CCN节点动态降低语句在CN预占的内存资源，将高估的内存腾让出来，给后续作业使用，从一定程度上缓解了语句严重高估导致系统资源利用率下降的问题。

参数类型：SIGHUP

取值范围：字符串

- on表示启用内存负反馈功能。
- off表示关闭内存负反馈功能。
- on()表示启用内存负反馈功能，并且指定触发负反馈需要的时间以及触发负反馈需要的估算内存百分比参数。例如：on(60,50)表示启用内存负反馈功能，触发负反馈机制生效需要连续60秒都是高估场景，还需要语句预占的估算内存总值要超过系统可用内存的50%。默认触发负反馈机制生效的时间长度为50秒，默认触发负反馈生效的最低估算内存总值要超过系统可用内存的40%。

默认值：on

bbox_dump_count

参数说明：在bbox_dump_path定义的路径下，允许存储的DWS所产生core文件最大数。超过此数量，旧的core文件会被删除。此参数只有当enable_bbox_dump为on时才生效。

参数类型：USERSET

取值范围：整型，1～20

默认值：8

说明

在并发产生core文件时，core文件的产生个数可能大于bbox_dump_count。

io_limits

参数说明：该参数8.1.2版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。

参数类型：USERSET

取值范围：整型，0～1073741823

默认值：0

io_priority

参数说明：该参数8.1.2版本中已废弃，为兼容历史版本功能保留该参数，当前版本设置无效。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：枚举型

- None
- Low
- Medium
- High

默认值：None

session_respool

参数说明：当前的session关联的resource pool。

参数类型：USERSET

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：字符串，通过create resource pool所设置的资源池。

默认值：invalid_pool

enable_transaction_parctl

参数说明：是否管控事务块语句和存储过程语句。

参数类型：USERSET

取值范围：布尔型

- on表示对事务块及存储过程语句进行管控。
- off表示对事务块及存储过程语句不进行管控。

默认值：on

session_history_memory

参数说明：设置历史查询视图的内存大小。

参数类型：SIGHUP

取值范围：整型，10240～max_process_memory的50%，单位为KB。

默认值：500MB

说明

新安装的9.1.2及以上集群版本中该参数的默认值为500MB，升级场景下为保持前向兼容该参数的默认值维持原值。

 版权所有 © 华为云计算技术有限公司

topsql_retention_time

参数说明：设置历史TopSQL中gs_wlm_session_info和gs_wlm_operator_info表中数据的保存时间。

参数类型：SIGHUP

取值范围：整型，0~3650，单位为天。

- 值为0时，表示数据永久保存。
- 值大于0时，表示数据能够保存的对应天数。

默认值：30

注意

- 设置此GUC参数启用数据保存功能前，请先清理gs_wlm_session_info和gs_wlm_operator_info表中的数据。
- 新建集群默认值为30，升级场景该参数的默认值为保持前向兼容维持原值。

transaction_pending_time

参数说明：当enable_transaction_parctl为on时，事务块语句和存储过程语句排队的最大时间。

参数类型：USERSET

取值范围：整型，-1～INT_MAX，单位为秒（s）。

- 值为-1或0：事务块语句和存储过程语句无超时判断，排队至资源满足可执行条件。
- 值大于0：事务块语句和存储过程语句排队超过所设数值的时间后，无视当前资源情况强制执行。

默认值：0

注意

此参数仅对存储过程及事务块的内部语句有效，即**PG_SESSION_WLMSTAT**中enqueue字段显示为Transaction或StoredProc的语句才会生效。

enable_concurrency_scaling

参数说明：设置是否打开弹性并发扩展功能。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启弹性并发扩展功能。

 版权所有 © 华为云计算技术有限公司

- off表示关闭弹性并发扩展功能。

默认值：off

concurrency_scaling_max_idle_time

参数说明：用于指定并发扩展创建的弹性逻辑集群的最大空闲时间，如果超过该参数的设置值则进入弹性逻辑集群销毁流程。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~60，单位为分钟（ min ）。其中，0表示该并发扩展创建的弹性逻辑集群不会被销毁，请谨慎设置。

默认值：5

concurrency_scaling_limit_per_main_vw

参数说明：用于限制每个经典逻辑集群所能创建的最大并发扩展弹性逻辑集群的数量。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，0~32

默认值：5

concurrency_scaling_max_vw_active_statements

参数说明：用于指定并发扩展弹性逻辑集群上所能执行的最大并发数。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：整型，-1~INT_MAX。

- 0，表示该并发扩展弹性逻辑集群不执行任何作业，请谨慎设置。
- -1，表示该并发扩展弹性逻辑集群不受并发管控，请谨慎设置。该取值仅9.1.0.200及以上集群版本支持。

默认值：60

concurrency_scaling_max_waiting_statements

参数说明：用于指定全局排队队列中，触发并发扩展弹性逻辑集群创建流程的弹性作业排队数量。该参数仅9.1.0.100及以上集群版本支持。

如果大于等于该参数的设置值且当前已创建的并发扩展逻辑集群未超过concurrency_scaling_limit_per_main_vw设置值，则自动进入创建并发扩展弹性逻辑集群流程。

参数类型：SIGHUP

取值范围：整型，0~INT_MAX。其中0表示未排队也会创建弹性逻辑集群。设置为0时会消耗大量资源，请谨慎设置。

默认值：10

 版权所有 © 华为云计算技术有限公司

vw_offloading_min_cost

参数说明：用于设置作业路由到弹性逻辑集群执行的最小计划代价，大于该参数，作业才被允许路由到弹性逻辑集群执行，小于该参数，作业将会在主逻辑集群执行。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1~INT_MAX。其中-1表示所有作业都将在主逻辑集群执行。

默认值：10000

disk_utilization_threshold

参数说明：磁盘空间管控使用率阈值。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~100。

默认值：90

说明

为保证磁盘空间管控功能正常，该参数的取值应不大于datastorage_threshold_value_check。

disk_utilization_threshold_options

参数说明：用于设置对指定特性启用磁盘空间管控功能。该参数仅9.1.1及以上集群版本支持。

参数类型：USERSET

取值范围：枚举值

- none：表示磁盘空间管控功能关闭。
- vacuum_full：对VACUUM FULL启用磁盘空间管控功能。当启用后，在VACUUM FULL执行前会进行一次检查，如果当前磁盘占用空间 + 执行VACUUM FULL的表的大小 > disk_utilization_threshold指定的阈值时，报错退出。

默认值：vacuum_full

cn_connection_policy

参数说明：用于控制是否启用内部连接优化功能。该功能开启后在使用内部资源管控功能场景下与CCN进行通信时，业务将使用固定的通道与CCN通信。该参数仅9.1.1.200及以上集群版本支持。

参数类型：SIGHUP

取值范围：枚举值

- none：表示内部连接优化功能关闭。
- enable_wlm_cn_comm：表示启用内部连接优化功能，即在使用内部资源管控功能场景下与CCN进行通信时，所有的业务均使用固定通道与CCN进行通信，不占用额外的连接，避免导致CCN上连接数（线程数）过多的问题。

默认值：none

 版权所有 © 华为云计算技术有限公司

18.13 自动清理

系统自动清理进程自动执行VACUUM和ANALYZE命令，回收被标识为删除状态的记录空间，并更新表的统计数据。

autovacuum_compaction_rows_limit

参数说明：小CU的阈值，存活元组数小于这个值的就会被认为是小CU。该参数9.1.1.100及以上集群版本中已废弃。

参数类型：USERSET

取值范围：整型，-1~5000

默认值：2500

说明

- 9.1.0.100以下集群版本，该参数默认值为-1，表示关闭0CU开关。禁止设置该参数，否则可能会导致**主键数据重复**。
- 9.1.0.100版本，该参数默认值为0。
- 9.1.0.200及以上版本，该参数默认值为2500。

autovacuum_asyncsort_rows_limit

参数说明：自动异步排序的行数阈值。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，120000~4200000

默认值：600000

autoanalyze_mode

参数说明：设置autoanalyze的模式。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- normal，表示普通的autoanalyze。
- light，表示轻量化的autoanalyze。

默认值：light

analyze_stats_mode

参数说明：设置ANALYZE计算统计信息的模式。

参数类型：USERSET

取值范围：枚举类型

- memory，表示强制使用内存计算统计信息，不计算多列统计信息。

 版权所有 © 华为云计算技术有限公司

- sample_table，表示强制使用临时采样表计算统计信息，临时表不支持使用该模式。
- dynamic，表示按内存maintenance_work_mem大小自适应选择统计信息计算模式，若**maintenance_work_mem**可放下样本，则使用内存方式，否则使用临时采样表方式。

默认值：dynamic

analyze_sample_mode

参数说明：设置ANALYZE时使用的采样模型。

参数类型：USERSET

取值范围：整型，0～2。

- 0，表示使用默认蓄水池采样模型。
- 1，表示使用优化的蓄水池采样模型。
- 2，表示使用range采样模型。

默认值：0

autovacuum_max_workers

参数说明：设置能同时运行的自动清理行存表线程的最大数量。

参数类型：SIGHUP

取值范围：整型，0～128 。其中0表示不会自动进行行存表的autovacuum。

默认值：2

说明

该参数与autovacuum共同发挥作用，对系统表和用户行存表的清理规则如下：

- autovacuum_max_workers = 0时，autovacuum被彻底关闭，不会对行存表进行清理。
- autovacuum_max_workers > 0和autovacuum = off，只对系统表进行清理。
- autovacuum_max_workers > 0和autovacuum = on，会对所有行存表进行清理。

autovacuum_max_workers_hstore

参数说明：设置同时运行的专用于merge hstore delta表的自动线程的最大数量，不受autovacuum参数控制。

参数类型：SIGHUP

取值范围：整型，0 ~ 128。其中0表示关闭该参数功能。

默认值：3

autovacuum_naptime

参数说明：设置两次自动清理操作的时间间隔。

参数类型：SIGHUP

取值范围：整型，1～2147483 ，单位为秒（s）。

 版权所有 © 华为云计算技术有限公司

默认值：20s

autovacuum_vacuum_cost_delay

参数说明：设置在自动VACUUM操作里使用的开销延迟数值。

参数类型：SIGHUP

取值范围：整型，-1～100，单位为毫秒（ms）。其中-1表示使用常规的vacuum_cost_delay。

默认值：2ms

check_crossvw_write

参数说明：控制是否要开启跨VW写场景的检测。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1、1。

- -1，表示兼容9.0.3版本能力。对于v3表vacuum，只清理所有epoch的非最后一个文件。
- 1，表示检查是否为跨VW写场景。对于v3表vacuum，若检查为非跨VW写的场景，清理所有epoch的非最后一个文件，清理当前epoch的最后一个文件，清理小于当前epoch的最后一个文件。若检查为跨VW写的场景，CN节点会获取所有DN节点的epoch信息包装成epochList下发给元数据VW，v3表vacuum会清理所有epoch的非最后一个文件，清理小于max{epochList}且不在epochList中的epoch最后一个文件。

默认值：1

enable_pg_stat_object

参数说明：控制auto vacuum是否更新**PG_STAT_OBJECT**系统表。该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示auto vacuum更新PG_STAT_OBJECT系统表。
- off表示auto vacuum不更新PG_STAT_OBJECT系统表。

默认值：on

18.14 客户端连接缺省设置

18.14.1 语句行为

介绍SQL语句执行过程的相关默认参数。

 版权所有 © 华为云计算技术有限公司

search_path

参数说明：当一个被引用对象没有指定模式时，此参数设置模式搜索顺序。它的值由一个或多个模式名构成，不同的模式名用逗号隔开。

参数类型：USERSET

- 当前会话存放临时表的模式，可以使用别名pg_temp将它列在搜索路径中，如'pg_temp, public' 。 存放临时表的模式始终会作为第一个被搜索的对象，排在pg_catalog和search_path中所有模式的前面，即具有第一搜索优先级。建议用户不要在search_path中显式设置pg_temp。如果在search_path中指定了pg_temp，但不是在最前面，系统会提示设置无效，pg_temp仍被优先搜索。通过使用别名pg_temp，系统只会在存放临时表的模式中搜索表、视图和数据类型这样的数据库对象，不会在里面搜索函数或运算符这样的数据库对象。
- 系统表所在的模式pg_catalog，总是排在search_path中指定的所有模式前面被搜索，即具有第二搜索优先级（pg_temp具有第一搜索优先级）。建议用户不要在search_path中显式设置pg_catalog。如果在search_path中指定了pg_catalog，但不是在最前面，系统会提示设置无效，pg_catalog仍会被作为第二优先搜索。
- 当没有指定一个特定模式而创建一个对象时，它们被放置到以search_path为命名的第一个有效模式中。当搜索路径为空时，会报错误。
- 通过SQL函数current_schema可以检测当前搜索路径的有效值。这和检测search_path的值不尽相同，因为current_schema显示search_path中首位有效的模式名称。

取值范围：字符串

默认值："$user",public

说明

- $user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。
- 设置为"$user"，public时，支持共享数据库（没有用户具有私有模式和所有共享使用public），用户私有模式和这些功能的组合使用。可以通过改变默认搜索路径来获得其他效果，无论是全局化的还是私有化的。
- 设置为空串（''）的时候，系统会自动转换成一对双引号。
- 设置的内容中包含双引号，系统会认为是不安全字符，会将每个双引号转换成一对双引号。

current_schema

参数说明：设置当前的模式。

参数类型：USERSET

取值范围：字符串

默认值："$user",public

说明

$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

default_tablespace

参数说明：当CREATE命令没有明确声明表空间时，指定所创建对象（表和索引等）的默认表空间。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：字符串，空表示使用默认表空间。

默认值：空

说明

- 值是一个表空间的名字或者一个表示使用当前数据库缺省表空间的空字符串。若指定的是一个非默认表空间，用户必须具有它的CREATE权限，否则尝试创建会失败。
- 临时表不使用此参数，可以用**temp_tablespaces**代替。
- 创建数据库时不使用此参数。默认情况下，一个新的数据库从模板数据库继承表空间配置。

default_storage_nodegroup

参数说明：设置当前的默认建表所在的Node Group，目前只适用普通表。

参数类型：USERSET

取值范围：字符串

- installation，表示默认建表在安装的Node Group上。
- random_node_group，表示默认建表在随机选择的NodeGroup上，该配置8.1.2及以上版本支持，仅用于测试环境。
- roach_group，表示默认建表在所有节点上，该值为roach工具预留，不能用于其他场景。
- 取值为其他字符串，表示默认建表在设置的Node Group上。

默认值：installation

temp_tablespaces

参数说明：当一个CREATE命令没有明确指定一个表空间时，temp_tablespaces指定了创建临时对象（临时表和临时表的索引）所在的表空间。在这些表空间中创建临时文件用来做大型数据的排序工作。

其值是一系列表空间名的列表。如果列表中有多个表空间时，每次临时对象的创建，DWS会在列表中随机选择一个表空间；如果在事务中，连续创建的临时对象被放置在列表里连续的表空间中。如果选择的列表中的元素是一个空串，DWS将自动将当前的数据库设为默认的表空间。

参数类型：USERSET

取值范围：字符串。空字符串表示所有的临时对象仅在当前数据库默认的表空间中创建，请参见**default_tablespace**。

默认值：空

check_function_bodies

参数说明：设置是否在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。为了避免产生问题（比如避免从转储中恢复函数定义时向前引用的问题），偶尔会禁用验证。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。
- off表示在CREATE FUNCTION执行过程中不进行函数体字符串的合法性验证。

默认值：on

default_transaction_isolation

参数说明：设置默认的事务隔离级别。

参数类型：USERSET

取值范围：枚举型

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。
- read uncommitted：读未提交隔离级别，DWS不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其他并发事务提交的修改。
- serializable：事务可序列化，DWS不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

default_transaction_read_only_probe

参数说明：数据库即将只读（磁盘使用率达到90%）时，控制是否终止特殊语句（下盘语句、产生新表或者新物理文件的语句）的执行。该参数由CM模块检测磁盘使用阈值并设置，不建议用户设置。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示终止特殊语句执行。
- off表示不终止特殊语句执行。

默认值：off

default_transaction_deferrable

参数说明：控制每个新事务的默认延迟状态。只读事务或者那些比序列化更加低的隔离级别的事务除外。

DWS不支持可串行化的隔离级别，因此，该参数无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示默认延迟。
- off表示默认不延迟。

默认值：off

 版权所有 © 华为云计算技术有限公司

session_replication_role

参数说明：控制当前会话与复制相关的触发器和规则的行为。

参数类型：SUSET

取值范围：枚举型

- origin，表示从当前会话中复制插入、删除、更新等操作。
- replica，表示从其他地方复制插入、删除、更新等操作到当前会话。
- local，表示函数执行复制时会检测当前登录数据库的角色并采取相应的操作。

默认值：origin

注意

设置此参数会丢弃之前所有缓存的执行计划。

statement_timeout

参数说明：当语句执行时间超过该参数设置的时间（从服务器收到命令时开始计时）时，该语句将会报错并退出执行。

参数类型：USERSET

取值范围：整型，0～2147483647，单位为毫秒（ms）。

默认值：86400000

vacuum_freeze_min_age

参数说明：指定VACUUM在扫描一个表时用于判断是否用FrozenXID替换事务ID的年龄阈值（在同一个事务中）。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

默认值：5000000000

说明

虽然可以将此参数设为0到10亿之间的任意值，但VACUUM会实际有效值范围限制在**autovacuum_freeze_max_age**的50%以内。

vacuum_freeze_table_age

参数说明：指定VACUUM对全表的扫描冻结元组的时间。如果表的pg_class.relfrozenxid字段的值已经达到了参数指定的时间，VACUUM对全表进行扫描。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

默认值：15000000000

版权所有 © 华为云计算技术有限公司

说明

虽然可以将此参数设为零到20亿之间的值，但VACUUM会将实际有效值范围限制在**autovacuum_freeze_max_age**的95%以内。定期的手动VACUUM可以在对此表的反重叠自动清理启动之前运行。

bytea_output

参数说明：设置bytea类型值的输出格式。

参数类型：USERSET

取值范围：枚举型

- hex：将二进制数据编码为每字节2位十六进制数字。
- escape：传统化的PostgreSQL格式。采用以ASCII字符序列表示二进制串的方法，同时将那些无法表示成ASCII字符的二进制串转换成特殊的转义序列。

默认值：hex

xmlbinary

参数说明：设置二进制值是如何在XML中进行编码的。

参数类型：USERSET

取值范围：枚举型

- base64
- hex

默认值：base64

xmloption

参数说明：当XML和字符串值之间进行转换时，设置document或content是否是隐含的。

参数类型：USERSET

取值范围：枚举型

- document：表示HTML格式的文档。
- content：普通的字符串。

默认值：content

gin_pending_list_limit

参数说明：设置当GIN索引启用快速更新**FASTUPDATE**时，Pending List（待处理列表）容量的最大值。

Pending List是GIN索引特有的数据结构，用于临时存储索引更新操作。当GIN索引启用FASTUPDATE参数时，新的索引项不会直接写入主索引结构，而是先存入Pending List，待满足特定条件时再批量合并到主索引中。

当Pending List的容量大于设置值时，会把Pending List中数据批量移动到GIN索引数据结构中以进行清理。单个GIN索引可通过更改索引存储参数覆盖此设置值。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：整型，64～INT_MAX，单位为KB。

默认值：4MB

enable_schema_owner_override

参数说明：控制创建表时是否使用Schema的所有者（Schema Owner）作为新建表的所有者（Table Owner）。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on，表示允许Schema Owner覆盖默认的Table Owner。新建表的所有者设置为所属Schema的所有者。该行为兼容Oracle的对象所有权语义。
- off，表示不允许Schema Owner覆盖Table Owner。新建表的所有者为执行建表操作的用户。遵循PostgreSQL的对象所有权语义。

默认值：on

18.14.2 区域和格式化

介绍时间格式设置的相关参数。

DateStyle

参数说明：设置日期和时间值的显示格式，以及有歧义的输入值的解析规则。

这个变量包含两个独立的部分：输出格式声明（ISO、Postgres、SQL、German）和输入输出的年/月/日顺序（DMY、MDY、YMD）。这两个可以独立设置或者一起设置。关键字Euro和European等价于DMY；关键字US、NonEuro、NonEuropean等价于MDY 。

参数类型：USERSET

取值范围：字符串

默认值：ISO, MDY

说明

gs_initdb会将这个参数初始化成与**lc_time**一致的值。

设置建议：优先推荐使用ISO格式。Postgres、SQL和German均采用字母缩写的形式来表示时区，例如“EST、WST、CST”等。

IntervalStyle

参数说明：设置区间值的显示格式。

参数类型：USERSET

取值范围：枚举型

- sql_standard表示产生与SQL标准规定匹配的输出。

 版权所有 © 华为云计算技术有限公司

- postgres表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为ISO时。
- postgres_verbose表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为non_ISO时。
- iso_8601表示产生与在ISO 8601中定义的“格式与代号”相匹配的输出。
- oracle表示产生于Oracle中与numtodsinterval函数相匹配的输出结果，详细请参考numtodsinterval。

须知

IntervalStyle参数也会影响不明确的间隔输入的说明。

默认值： postgres

TimeZone

参数说明： 设置显示和解释时间类型数值时使用的时区。

参数类型： USERSET

取值范围： 字符串，可查询视图**pg_timezone_names**获得。

默认值： UTC

说明

gs_initdb将设置一个与其系统环境一致的时区值。

timezone_abbreviations

参数说明： 设置服务器接受的时区缩写值。

参数类型： USERSET

取值范围： 字符串，可查询视图pg_timezone_names获得。

默认值： Default

说明

Default表示通用时区的缩写。但也有其他诸如 'Australia' 和 'India' 等用来定义特定的安装。

extra_float_digits

参数说明： 调整浮点值显示的数据位数，浮点类型包括float4、float8 以及几何数据类型。参数值加在标准的数据位数上（FLT_DIG或DBL_DIG中合适的）。

参数类型： USERSET

取值范围： 整型，-15～3

 版权所有 © 华为云计算技术有限公司

说明

- 设置为3，表示包括部分有效的数据位。对转储需要精确恢复的浮点数据尤其有用。
- 设置为负数，表示摒弃不需要的数据位。

默认值：0

client_encoding

参数说明：设置客户端的字符编码类型。

请根据前端业务的情况确定。尽量客户端编码和服务器端编码一致，提高效率。

参数类型：USERSET

取值范围：兼容PostgreSQL所有的字符编码类型。其中UTF8表示使用数据库的字符编码类型。

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。
- 参数建议保持默认值，不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置client_encoding参数，即使设置也不会生效，以保证集群内部通信编码格式一致。

默认值：UTF8

推荐值：SQL_ASCII/UTF8

lc_messages

参数说明：设置信息显示的语言。

可接受的值是与系统相关的；在一些系统上，这个区域范畴并不存在，不过仍然允许设置这个变量，只是不会有任何效果。同样，也有可能是所期望的语言的翻译信息不存在。在这种情况下，用户仍然能看到英文信息。

参数类型：SUSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_monetary

参数说明：设置货币值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

 版权所有 © 华为云计算技术有限公司

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_numeric

参数说明：设置数值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_time

参数说明：设置时间和区域的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

default_text_search_config

参数说明：设置全文检索的配置信息。

如果设置为不存在的文本搜索配置时将会报错。如果default_text_search_config对应的文本搜索配置被删除，需要重新设置default_text_search_config，否则会报设置错误。

- 其被文本搜索函数使用，这些函数并没有一个明确指定的配置。
- 当与环境相匹配的配置文件确定时，gs_initdb会选择一个与环境相对应的设置来初始化配置文件。

参数类型：USERSET

取值范围：字符串

 版权所有 © 华为云计算技术有限公司

说明

DWS支持pg_catalog.english，pg_catalog.simple两种配置。

默认值：pg_catalog.english

18.14.3 其他缺省

主要介绍数据库系统默认的库加载参数。

dynamic_library_path

参数说明：设置数据查找动态加载的共享库文件的路径。当需要打开一个可以动态装载的模块并且在CREATE FUNCTION或LOAD命令里面声明的名字没有目录部分时，系统将搜索这个目录以查找声明的文件。

用于dynamic_library_path的数值必须是一个冒号分隔（Windows下是分号分隔）的绝对路径列表。当一个路径名字以特殊变量$libdir为开头时，会替换为DWS发布提供的模块安装路径。例如：

```
dynamic_library_path = '/usr/local/lib/postgresql:/opt/testgs/lib:$libdir'
```

参数类型：SUSET

取值范围：字符串

说明

设置为空字符串，表示关闭自动路径搜索。

默认值：$libdir

gin_fuzzy_search_limit

参数说明：设置GIN索引返回的集合大小的上限。

参数类型：USERSET

取值范围：整型，0～INT_MAX，0表示没有限制。

默认值：0

18.15 锁管理

在DWS中，并发执行的事务由于竞争资源可能会导致单机死锁或分布式死锁。本节介绍的参数主要管理事务锁的机制。

deadlock_timeout

参数说明：设置死锁超时检测时间，以毫秒为单位。当申请的锁超过设定值时，系统会检查是否产生了死锁。

- 死锁的检查代价是比较高的，服务器不会在每次等待锁的时候都运行这个过程。在系统运行过程中死锁是不经常出现的，因此在检查死锁前只需等待一个相对较短的时间。增加这个值就减少了无用的死锁检查浪费的时间，但是会减慢真正的死锁错误报告的速度。在一个负载过重的服务器上，用户可能需要增大它。这个值的设置应该超过事务持续时间，这样就可以减少在锁释放之前就开始死锁检查的问题。

 版权所有 © 华为云计算技术有限公司

- 设置log_lock_waits时，这个选项也决定了在一个日志消息发出关于锁等待以前要等待的时间。当需要调查锁延迟时，请设置比正常deadlock_timeout更小的值。

参数类型：SUSET

取值范围：整型，1~2147483647，单位为毫秒（ms）。

默认值：1s

ddl_lock_timeout

参数说明：通过该参数单独指定阻塞DDL语句锁等待的时间，当申请的锁等待时间超过设定值时，系统会报错。该参数仅8.1.3.200及以上版本支持。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

- 如果该参数的值等于0，表示该参数不生效。
- 如果该参数的值大于0，DDL锁阻塞时间为该参数的值，其他锁等待时间为lockwait_timeout参数值。

默认值：0

说明

该参数优先级高于lockwait_timeout，只针对AccessExclusiveLock生效。

ddl_select_concurrent_mode

参数说明：通过该参数控制DDL语句和SELECT语句并发的模式。该参数仅8.1.3.320、8.2.1及以上集群版本支持。

参数类型：SUSET

取值范围：字符串

- none：表示该参数不生效，DDL语句和select语句不能并发，保持锁等待状态。
- truncate：表示truncate语句被select语句阻塞时，truncate会中断select语句，优先执行。
- exchange：表示exchange语句被select语句阻塞时，exchange会中断select语句，优先执行。
- vacuum_full：表示vacuum full语句被select语句阻塞时，vacuum full会中断select语句，优先执行。（该参数仅9.1.0.200及以上集群版本支持）
- insert_overwrite：表示insert overwrite语句被select语句阻塞时，insert overwrite会中断select语句，优先执行。（该参数仅9.1.0.200及以上集群版本支持）
- lockstmt，表示添加八级锁被select语句阻塞时，lockstmt会中断select语句，优先执行。如果处于重分布期间select语句会重试。（该参数仅9.1.1.200及以上集群版本支持）

默认值：none

 版权所有 © 华为云计算技术有限公司

说明

- 为了给SELECT语句预留响应信号的时间，当前版本中设置的ddl_lock_timeout的值不足1秒时按照1s处理。
- 与高级别的锁冲突（大于1级），不支持并发（比如autoanalyze_mode=normal时，同时SELECT触发了autoanalyze）。
- 该参数支持与单语句中的select或事务块中的select并发，其他版本中仅支持与单语句的select并发。对于单语句或事务块中的select并发处理，参考**enable_cancel_select_in_txnblock**参数。
- 除none以外的其他几个参数可以组合使用，比如配置为“truncate, exchange”表示truncate和exchange语句被select语句阻塞时，二者会中断select语句，优先执行。

enable_cancel_select_in_txnblock

参数说明：控制在事务块中的select是否可以被中断。

参数类型：USERSET

取值范围：布尔型

- on表示开启，即事务块中的select可以被中断。
- off表示关闭，即事务块中的select不可以被中断。

默认值：on

说明

- 该参数8.2.1.236及以上的8.2.1.2xx集群版本支持；9.1.0.200及以上集群版本支持。
- 该参数用于控制处于事务块中的SELECT是否可以被ddl_select_concurrent_mode中指定的DDL操作中断。
- ddl_select_concurrent_mode参数控制truncate、exchange等DDL语句，enable_cancel_select_in_txnblock参数控制select语句。

lockwait_timeout

参数说明：控制单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

默认值：20min

update_lockwait_timeout

参数说明：允许并发更新参数开启情况下，该参数控制并发更新同一行时单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ms）。

默认值：2min

 版权所有 © 华为云计算技术有限公司

partition_lock_upgrade_timeout

参数说明：分区上的锁级别由允许读的ExclusiveLock升级到读写阻塞的AccessExclusiveLock时，会进行尝试性的锁升级，partition_lock_upgrade_timeout指示了尝试锁升级的超时时间。

- 在分区表上进行MERGE PARTITION和CLUSTER PARTITION操作时，都利用了临时表进行数据重排和文件交换，为了最大程度提高分区上的操作并发度，在数据重排阶段给相关分区加锁ExclusiveLock，在文件交换阶段加锁AccessExclusiveLock。
- 常规加锁方式是等待加锁，直到加锁成功，或者等待时间超过**lockwait_timeout**发生超时失败。
- 在分区表上进行MERGE PARTITION或CLUSTER PARTITION操作时，进入文件交换阶段需要申请加锁AccessExclusiveLock，加锁方式是尝试性加锁，加锁成功了则立即返回，不成功则等待50ms后继续下次尝试，加锁超时时间使用会话级设置参数partition_lock_upgrade_timeout。
- 特殊值：若partition_lock_upgrade_timeout取值-1，表示无限等待，即不停地尝试锁升级，直到加锁成功。

参数类型：USERSET

取值范围：整型，-1 ~ 3000，单位为秒（s）。

默认值：1800

enable_release_scan_lock

参数说明：控制SELECT语句是否在语句执行结束后将一级锁释放。 打开此开关，可以缓解DDL等与事务块中的SELECT锁冲突问题。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启。SELECT语句在语句执行结束后释放一级锁，不会等事务提交才进行释放。
- off表示关闭。

默认值：off

vacuum_full_interruptible

参数说明：通过该参数控制VACUUM FULL语句给其他语句的让锁行为。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启。当VACUUM FULL阻塞其他语句时，VACUUM FULL会中断执行，给其他语句让锁。
- off表示关闭。当VACUUM FULL阻塞其他语句时，VACUUM FULL不会中断执行，其他语句需要等待VACUUM FULL执行完毕放锁后才能执行。

默认值：off

 版权所有 © 华为云计算技术有限公司

18.16 版本和平台兼容性

18.16.1 历史版本兼容性

介绍数据库的向下兼容性和对外兼容性特性的参数控制。数据库系统的向后兼容性能够为旧版本的数据库应用提供支持。本节介绍的参数主要控制数据库的向后兼容性。

array_nulls

参数说明：控制数组输入解析器是否将未用引用的NULL识别为数组的一个NULL元素。

参数类型：USERSET

取值范围：布尔型

- on表示允许向数组中输入空元素。
- off表示向下兼容旧式模式。仍然能够创建包含NULL值的数组。

默认值：on

backslash_quote

参数说明：控制字符串文本中的单引号是否能够用\'表示。

参数类型：USERSET

须知

在字符串文本符合SQL标准的情况下，\没有任何其他含义。这个参数影响的是如何处理不符合标准的字符串文本，包括明确的字符串转义语法是（ E'...' ）。

取值范围：枚举类型

- on表示一直允许使用\'表示。
- off表示拒绝使用\'表示。
- safe_encoding表示仅在客户端字符集编码不会在多字节字符末尾包含\的ASCII值时允许。

默认值：safe_encoding

default_with_oids

参数说明：在没有声明WITH OIDS和WITHOUT OIDS的情况下，这个选项控制在新创建的表中CREATE TABLE和CREATE TABLE AS是否包含一个OID字段。它还决定SELECT INTO创建的表里面是否包含OID 。

不推荐在用户表中使用OID，故默认设置为off。需要带有OID字段的表应该在创建时声明WITH OIDS 。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示在新创建的表中CREATE TABLE和CREATE TABLE AS可以包含一个OID字段。
- off表示在新创建的表中CREATE TABLE和CREATE TABLE AS不可以包含一个OID字段。

默认值：off

enable_parquet_integer_decimal

参数说明：控制parquet外表写decimal(p,s)类型且p<=18时，选择使用INT64类型或FIXED_LEN_BYTE_ARRAY类型存储。

参数类型：USERSET

取值范围：布尔型

- on表示使用INT64类型存储。
- off表示使用FIXED_LEN_BYTE_ARRAY类型存储。

默认值：off

说明

- 此参数支持版本：
  - 9.1.0集群版本：仅9.1.0.222及以上版本支持该参数。
  - 9.1.1集群版本：仅9.1.1.200及以上版本支持该参数。

  为了兼容hive行为（hive使用FIXED_LEN_BYTE_ARRAY），此参数默认使用FIXED_LEN_BYTE_ARRAY类型存储，并保留历史行为兼容性。
- 暂未支持此参数的集群版本中，DWS parquet外表默认将p<=18的decimal存储为INT64类型。
- INT64或FIXED_LEN_BYTE_ARRAY，此处均表示parquet内部定义的类型，不同的外部引擎可能会选择写不同的类型。

escape_string_warning

参数说明：警告在普通字符串中直接使用反斜杠转义。

- 如果需要使用反斜杠作为转义，可以调整为使用转义字符串语法(E'...')来做转义，因为在每个SQL标准中，普通字符串的默认行为现在将反斜杠作为一个普通字符。
- 这个变量可以帮助定位需要改变的代码。

参数类型：USERSET

取值范围：布尔型

默认值：on

lo_compat_privileges

参数说明：控制是否启动对大对象权限检查的向后兼容模式。

参数类型：SUSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

on表示当读取或修改大对象时禁用权限检查，与PostgreSQL 9.0以前的版本兼容。

默认值：off

quote_all_identifiers

参数说明：当数据库生成SQL时，此选项强制引用所有的标识符（包括非关键字）。这将影响到EXPLAIN的输出及函数的结果，例如pg_get_viewdef。详细说明请参见gs_dump的--quote-all-identifiers选项。

参数类型：USERSET

取值范围：布尔型

- on表示打开强制引用。
- off表示关闭强制引用。

默认值：off

sql_inheritance

参数说明：控制继承语义。

参数类型：USERSET

取值范围：布尔型

off表示各种命令不能访问子表，即默认使用ONLY关键字。这是为了兼容7.1之前版本而设置的。

默认值：on

standard_conforming_strings

参数说明：控制普通字符串文本（'...'）中是否按照SQL标准把反斜杠视为普通文本。

- 应用程序通过检查这个参数可以判断字符串文本的处理方式。
- 建议明确使用转义字符串语法（E'...'）来转义字符。

参数类型：USERSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

默认值：on

synchronize_seqscans

参数说明：控制启动同步的顺序扫描。在大约相同的时间内并行扫描读取相同的数据块，共享I/O负载。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示扫描可能从表的中间开始，然后选择"环绕"方式来覆盖所有的行，为了与已经在进行中的扫描活动同步。这可能会造成没有用ORDER BY子句的查询得到行排序造成不可预测的后果。
- off表示确保顺序扫描是从表头开始的。

默认值：on

enable_beta_features

参数说明：控制开启某些功能受限的特性，例如GDS表关联操作。这些特性在历史版本未明确禁止，但某些场景功能上存在问题，不建议用户使用。

参数类型：USERSET

取值范围：布尔型

- on表示开启这些功能受限的特性，保持前向兼容。但某些场景可能存在功能上的问题。
- off表示禁止使用这些特性。

默认值：off

18.16.2 平台和客户端兼容性

数据库系统的对外兼容性，为平台服务多样化客户端提供了关键便利。

transform_null_equals

参数说明：控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。如果expr得出NULL值则返回真，否则返回假。

- 正确的SQL标准兼容的expr = NULL总是返回NULL（未知）。
- Microsoft Access里的过滤表单生成的查询使用expr = NULL来测试空值。打开这个选项，可以使用该接口来访问数据库。

参数类型：USERSET

取值范围：布尔型

- on表示控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。
- off表示不控制，即expr = NULL总是返回NULL（未知）。

默认值：off

说明

新用户经常在涉及NULL的表达式上语义混淆，故默认值设为off。

td_compatible_truncation

参数说明：控制是否开启与Teradata数据库相应兼容的特征。该参数在用户连接上与TD兼容的数据库时，可以将参数设置成为on（即超长字符串自动截断功能启用），该功能启用后，在后续的insert语句中，对目标表中char和varchar类型的列插入超长字符串时，会按照目标表中相应列定义的最大长度对超长字符串进行自动截断。保证数据都能插入目标表中，而不是报错。

 版权所有 © 华为云计算技术有限公司

说明

- 超长字符串自动截断功能不适用于insert语句包含外表的场景。
- 如果向字符集为字节类型编码（SQL_ASCII，LATIN1等）的数据库中插入多字节字符数据（如汉字等），且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

参数类型：USERSET

取值范围：布尔型

- on表示启动超长字符串自动截断功能。
- off表示停止超长字符串自动截断功能。

默认值：off

behavior_compat_options

参数说明：数据库兼容性行为配置项，该参数的值由若干个配置项用逗号隔开构成。不同的兼容性配置项，适用的兼容模式（ORA/TD/MySQL）不同，参见**表18-2**。

参数类型：USERSET

取值范围：字符串

默认值：新安装集群场景下，该参数默认值为check_function_conflicts, check_function_shippable, force_german_dmy, inner_iso_datestyle, correct_convert_tz, disable_srf_condition_expr, cte_mem_exceeds_not_inline_all, ora_void_return_null, job_broken_disable, overwrite_partition_optimization, disable_timetimestamp_comparison, chr_check_illegal_char以防止用户定义错误的函数属性导致严重的问题。

升级场景下保持前向兼容，即与升级前的集群中该参数的默认值保持一致。

说明

- 当前只支持**表18-2**。
- 配置多个兼容性配置项时，相邻配置项用逗号隔开，例如：set behavior_compat_options='end_month_calculate,display_leading_zero';
- 此参数选项中strict_concat_functions和strict_text_concat_td不能同时设置。
- Oracle兼容模式下不建议设置behavior_compat_options= 'return_null_string' ，如果设置了此选项，应避免将查询结果插入到表里。

表 18-2 兼容性配置项

兼容性配置项	兼容性行为控制	适用兼容模式
display_leading_zero	浮点数显示配置项。 • 不设置此配置项时，对于-1~0和0~1之间的小数，不显示小数点前的0。比如，0.25显示为.25。 • 设置此配置项时，对于-1~0和0~1之间的小数，显示小数点前的0。比如，0.25显示为0.25。	ORA TD

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

end_month_calculate	add_months函数计算逻辑配置项。
	假定函数add_months的两个参数分别为param1和param2，param1的月份和param2的月份和为result。
	● 不设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期小，计算结果中的日期字段（Day字段）和param1的日期字段保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-28 00:00:00`
	`(1 row)`
	● 设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期比小，计算结果中的日期字段（Day字段）和result的月末日期保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-31 00:00:00`
	`(1 row)`
		ORA
		TD

compat_analyze_sample	analyze采样行为配置项。 设置此配置项时，会优化analyze的采样行为，主要体现在analyze时全局采样会更精确的控制在3万条左右，更好地控制analyze时Coordinator端的内存消耗，保证analyze性能的稳定性。	ORA TD MySQL
bind_schema_tablespace	绑定模式与同名表空间配置项。 如果存在与模式名schema_name相同的表空间名，那么如果设置search_path为schema_name， default_tablespace也会同步切换到schema_name。	ORA TD MySQL
bind_procedure_searchpath	未指定模式名的数据库对象的搜索路径配置项。 在存储过程中如果不显式指定模式名，会优先在存储过程所属的模式下搜索。 如果找不到，则有两种情况： ● 若不设置此参数，报错退出。 ● 若设置此参数，按照search_path中指定的顺序继续搜索。如果还是找不到，报错退出。	ORA TD MySQL
correct_to_number	控制to_number()结果兼容性的配置项。 若设置此配置项，则to_number()函数结果与PG11保持一致，否则默认与Oracle保持一致。	ORA

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

unbind_divide_bound	控制对整数除法的结果进行范围校验。
	● 不设置此配置项时，将对除法结果进行校验，超出范围则报错。例如，示例中INT_MIN/(-1)会因为超过结果大于INT_MAX而报越界错误：
	`SELECT (-2147483648)::int / (-1)::int;`
	`ERROR:  integer out of range`
	● 若设置此配置项，则不需要对除法结果进行范围校验。例如，示例中INT_MIN/(-1)可以得到输出结果INT_MAX+1：
	`SELECT (-2147483648)::int / (-1)::int;`
	` ?column?`
	`------------`
	` 2147483648`
	`(1 row)`
		ORA
		TD

merge_update_multi	控制行存表merge into匹配多行时是否进行update操作。 若设置此配置项，匹配多行时update不报错，否则默认与Oracle保持一致，报错。	ORA TD
disable_row_update_multi	控制行存表update匹配多行时是否进行update操作。 若设置此配置项，匹配多行时update报错，否则默认可以进行多行匹配更新。	ORA TD

return_null_string	控制函数lpad()、rpad()、repeat()、regexp_split_to_table()和split_part()的结果为空字符串''的显示配置项。
	● 不设置此配置项时，空字符串显示为NULL。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	
	`(1 row)`
	● 设置此配置项时，空字符串显示为''。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	`0`
	`(1 row)`
		ORA

compat_concat_variadic	控制函数concat()和concat_ws()对variadic类型结果兼容性的配置项。 若设置此配置项，当concat函数参数为variadic类型时，保留Oracle和Teradata兼容模式下不同的结果形式；否则默认Oracle和Teradata兼容模式下结果相同，且与Oracle保持一致。	ORA TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
convert_string_digit_to_numeric	控制CHAR类型和INT类型进行二元BOOL运算时类型转换优先级的配置项。 ● 不设置此配置项时，类型转换优先级与PG9.6一致。 ● 设置此配置项时，所有CHAR类型和INT类型的二元BOOL运算均强制转换为NUMERIC类型进行计算。设置此配置项后会被影响的CHAR类型包括BPCHAR、VARCHAR、NVARCHAR2、TEXT四种类型，会被影响的INT类型包括INT1、INT2、INT4、INT8四种类型。 **注意** 此配置项只对二元BOOL运算生效，例如，INT2>TEXT、INT4=BPCHAR，非BOOL运算不会受到影响，该配置项暂不支持INT>'1.1'这类UNKNOWN类型运算的转换。由于该配置项开启后，CHAR类型与INT类型的BOOL运算会优先转换为NUMERIC类型进行计算，因此会影响数据库计算性能，当JOIN列为受影响的类型组合时，还会影响执行计划。	ORA TD MySQL

check_function_conflicts	控制是否检查自定义plpgsql/SQL函数的属性。
	● 不设置此配置项时，不检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性。
	● 设置此配置项时，会检查自定义函数的IMMUTABLE属性，如果函数中含有表，或者是有STABLE/VOLATILE函数时，在执行时会报错。因为函数中如果有表或者STABLE/VOLATILE函数时，与函数定义中的IMMUTABLE属性冲突，即这种场景下，函数的行为非IMMUTABLE。
	例如：设置此参数时，以下场景下会执行报错：
	`CREATE OR replace FUNCTION sql_immutable (INTEGER)`
	`RETURNS INTEGER AS 'SELECT a+$1 from shipping_schema.t4 where a=1;'`
	`LANGUAGE SQL IMMUTABLE`
	`RETURNS NULL`
	`ON NULL INPUT;`
	`select sql_immutable(1);`
	`ERROR:  IMMUTABLE function cannot contain SQL statements with relation or Non-IMMUTABLE function.`
	`CONTEXT:  SQL function "sql_immutable" during startup`
	`referenced column: sql_immutable`
		ORA
		TD
		MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

varray_verification	控制是否校验数组长度以及数组类型长度。用于兼容DWS 8.1.0之前的版本。
	若设置此配置项，不会校验数组长度以及数组类型长度。
	-- 场景1
	CREATE OR REPLACE PROCEDURE varray_verification
	AS
	TYPE org_varray_type IS varray(5) OF VARCHAR2(2);
	v_org_varray org_varray_type;
	BEGIN
	v_org_varray(1) := '111'; --例如赋值已经超过了VARCHAR2(2)的限制，配置该选项后将和历史版本保持一致不进行校验
	END;
	/
	--场景2
	CREATE OR REPLACE PROCEDURE varray_verification_i3_1
	AS
	TYPE org_varray_type IS varray(2) OF NUMBER(2);
	v_org_varray org_varray_type;
	BEGIN
	v_org_varray(3) := 1; --例如赋值已经超过了varray(2)的数组长度限制，配置该选项后将和历史版本保持一致不进行校验
	END;
	/
		ORA
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

strict_concat_functions	控制函数textanycat()和anytextcat()在参数存在空值时，对返回值兼容性的配置项。此参数不能和strict_text_concat_td同时设置。
	MySQL兼容模式下，此参数无影响。
	● 不设置此配置项时，函数textanycat()和anytextcat()的返回值默认与Oracle保持一致。
	● 设置此配置项时，若函数textanycat()和anytextcat()的参数存在空值，则返回值也为空值，保留与Oracle和Teradata兼容模式下不同的结果。
	例如，不设置此配置项时，函数textanycat()和anytextcat()的返回值与Oracle保持一致：
	`SELECT textanycat('gauss', cast(NULL as BOOLEAN));`
	` textanycat`
	`------------`
	` gauss`
	`(1 row)`
	
	`SELECT 'gauss' || cast(NULL as BOOLEAN); --这种情况下，||运算符会被转换为函数textanycat`
	` ?column?`
	`----------`
	` gauss`
	`(1 row)`
	设置此配置项时，保留与Oracle和Teradata兼容模式下不同的结果：
	`SELECT textanycat('gauss', cast(NULL as BOOLEAN));`
	` textanycat`
	`------------`
	
	`(1 row)`
	
	`SELECT 'gauss' || cast(NULL as BOOLEAN); --这种情况下，||运算符会被转换为函数textanycat`
	` ?column?`
	`----------`
	
	`(1 row)`
		ORA
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

strict_text_concat_td	Teradata兼容模式下，控制函数textcat()、textanycat()和anytextcat()在参数存在空值时，对返回值兼容性的配置项。此参数不能和strict_concat_functions同时设置。
	● 不设置此配置项时，Teradata兼容模式下函数textcat()、textanycat()和anytextcat()的返回值与DWS一致。
	● 设置此配置项时，若Teradata兼容模式下函数textcat()、textanycat()和anytextcat()的参数存在空值，则返回值为空值。
	例如，不设置此配置项时，函数textcat()、textanycat()和anytextcat()的返回值与DWS保持一致：
	td_compatibility_db=# SELECT textcat('abc', NULL);
	textcat
	--------- 	
	abc
	(1 row)
	td_compatibility_db=# SELECT 'abc' || NULL; --这种情况下，||运算符会被转换为函数textcat()
	?column?
	---------- 	
	abc
	(1 row)
	设置此配置项时，若函数textcat()、textanycat()和anytextcat()的返回值有空值，则返回NULL：
	td_compatibility_db=# SELECT textcat('abc', NULL);
	textcat
	--------- 	
	
	(1 row)
	td_compatibility_db=# SELECT 'abc' || NULL;
	?column?
	---------- 	
	
	(1 row)
		TD

compat_display_ref_table	设置视图中列的显示格式。 ● 不设置该选项时默认带前缀，即tab.col的格式。 ● 设置该选项时与原始定义一致，原始定义带前缀则显示，否则不显示。 SET behavior_compat_options='compat_display_ref_table'; CREATE OR REPLACE VIEW viewtest2 AS SELECT a.c1, **c2**, a.c3, 0 AS c4 FROM viewtest_tbl a; SELECT pg_get_viewdef('viewtest2'); pg_get_viewdef ---------- SELECT a.c1, **c2**, a.c3, 0 AS c4 FROM viewtest_tbl a; (1 row)	ORA TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
para_support_set_func	列存表中控制函数COALESCE()、NVL()、GREATEST()、LEAST()入参是否支持多结果集表达式。 ● 不设置此配置项时，函数入参包含多结果集表达式时，直接报错不支持。 SELECT COALESCE(regexp_split_to_table(c3,'#'), regexp_split_to_table(c3,'#')) FROM regexp_ext2_tb1 ORDER BY 1 LIMIT 5; ERROR: set-valued function called in context that cannot accept a set ● 设置此配置项时，支持函数入参包含多结果集表达式。 SELECT COALESCE(regexp_split_to_table(c3,'#'), regexp_split_to_table(c3,'#')) FROM regexp_ext2_tb1 ORDER BY 1 LIMIT 5; coalesce ---------- a a a a a (5 rows)	ORA TD
disable_select_truncate_parallel	控制分区表的TRUNCATE等ddl的锁等级。 ● 设置此配置项时，将禁止分区表的不同分区上truncate与DML(如select)的并发，允许分区表上select的FQS(快速下发)。在OLTP场景下分区表上的简单查询较多，并且没有分区表不同分区truncate与DML并发的需求，可以考虑设置此配置项。 ● 不设置此配置项时，分区表上不同分区的select与truncate可以并发进行，同时关闭分区表的FQS（快速下发）来避免可能的不一致问题。	ORA TD MySQL

bpchar_text_without_rtrim	Teradata兼容模式下，设置此参数时，控制bpchar到text转换保留右侧空格，如果实际长度不足bpchar指定的长度，对其进行补空格操作，兼容Teradata对bpchar字符串的处理风格。
	当前不支持“比较字符串时忽略尾部空格”，拼接后结果如果存在尾部空格，进行比较时会对空格敏感。
	例如，设置参数时：
	td_compatibility_db=# select length('a'::char(10)::text);
	length
	-------- 	
	10
	(1 row)
	
	td_compatibility_db=# select length('a'||'a'::char(10));
	length
	-------- 	
	11
	(1 row)
		TD

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

convert_empty_str_to_null_td	Teradata兼容模式下，设置此参数时，控制to_date, to_timestamp和to_number类型转换函数处理空串时，返回null；同时控制to_char函数处理date类型入参时返回的格式。
	例如：
	未设置此参数时：
	td_compatibility_db=# select to_number('');
	 to_number
	----------- 	
	0
	(1 row)
	
	td_compatibility_db=# select to_date('');
	ERROR: the format is not correct
	DETAIL: invalid date length "0", must between 8 and 10.
	CONTEXT: referenced column: to_date
	
	td_compatibility_db=# select to_timestamp('');
	to_timestamp
	------------------------ 	
	0001-01-01 00:00:00 BC
	(1 row)
	
	td_compatibility_db=# select to_char(date '2020-11-16');
	to_char
	------------------------ 	
	2020-11-16 00:00:00+08
	(1 row)
	设置此参数，若to_number, to_date, to_timestamp函数的参数有空串时：
	td_compatibility_db=# select to_number('');
	 to_number
	----------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_date('');
	 to_date
	--------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_timestamp('');
	 to_timestamp
	-------------- 	
	
	(1 row)
	
	td_compatibility_db=# select to_char(date '2020-11-16');
	 to_char
	------------ 	
	2020/11/16
	(1 row)
		TD

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_case_specific	控制字符类型匹配时是否忽略大小写。仅在Teradata兼容模式下生效。 ● 不设置此配置项时，字符类型匹配时，字符的大小写敏感。 ● 设置此配置项时，字符类型匹配时，字符的大小写不敏感。 设置此配置项后会影响的字符类型包括CHAR、TEXT、BPCHAR、VARCHAR、NVARCHAR五种类型，会被影响的操作符包括<、>、=、>=、<=、!=、<>、!=、like、not like、in、not in共12种操作符以及case when、decode 表达式。 **注意** 由于该配置项开启后，字符类型前会增加UPPER函数进而会影响估算逻辑，需要使用增强的估算模型。（建议设置：cost_param=16、cost_model_version = 1、join_num_distinct=-20、qual_num_distinct=200）	TD
enable_interval_to_text	控制interval到text类型的隐式转换功能。 ● 设置此选项时，支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- f (1 row) ● 不设置此选项时，不支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- t (1 row)	ORA TD MySQL
case_insensitive	MySQL兼容模式下，设置此参数，控制locate，strpos，instr字符串函数入参大小写不敏感。 目前默认未设置该参数，即入参大小写敏感。 例如： ● 未设置此选项时，入参大小写敏感。 mysql_compatibility_db=# SELECT LOCATE('sub', 'Substr'); locate -------- 0 (1 row) ● 设置此选项时，入参大小写不敏感。 mysql_compatibility_db=# SELECT LOCATE('sub', 'Substr'); locate -------- 1 (1 row)	MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
inherit_not_null_strict_func	控制函数原有的strict属性，参数为1个的函数可以传递NOT NULL属性的行为。即：对于func(x)，如果func()为strict属性，且x包含NOT NULL约束，则认为func(x)也是包含NOT NULL约束的。 该兼容配置项在某些优化场景，例如：NOT IN优化、COUNT(DISTINCT)优化，会有特定的优化效果，但特定场景可能导致结果错误。 目前默认未设置该参数，保证结果正确，但可能导致性能回退，如果出现问题可设置该参数回退到历史版本行为。	ORA TD MySQL
disable_compat_minmax_expr_mysql	MySQL兼容模式下，控制greatest/least表达式对null入参的处理方式。 默认兼容MySQL。可通过设置此参数，回退到历史版本行为。 ● 不设置此选项时，兼容MySQL行为，入参为null时返回null。 mysql_compatibility_db=# SELECT greatest(1, 2, null), least(1, 2, null); greatest | least ----------+------- | (1 row) ● 设置此选项时，返回非null参数中的最大/小值。 mysql_compatibility_db=# SELECT greatest(1, 2, null), least(1, 2, null); greatest | least ----------+------- 2 | 1 (1 row)	MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

disable_compat_substr_mysql	MySQL兼容模式下，控制substr/substring函数在起始位置pos <= 0时的行为。
	默认兼容MySQL。可通过设置此参数，回退到历史版本行为。
	● 不设置此选项时，兼容MySQL行为，即pos = 0时返回空串，pos < 0时从倒数第 |pos| 个位置开始截取字符。
	mysql_compatibility_db=# SELECT substr('helloworld',0);
	 substr
	-------- 	
	
	(1 row)
	mysql_compatibility_db=# SELECT substring('helloworld',0),substring('helloworld',-2,4);
	 substring | substring
	-----------+-----------
	 | ld
	(1 row)
	● 设置此选项时，pos <= 0时仍然从左侧开始截取字符。此时substr/substring函数行为与ORACLE兼容模式下行为一致。
	mysql_compatibility_db=# SELECT substr('helloworld',0);
	 substr
	------------ 	
	 helloworld
	(1 row)
	mysql_compatibility_db=# SELECT substring('helloworld',0),substring('helloworld',-2,4);
	 substring | substring
	------------+-----------
	 helloworld | h
	(1 row)
		MySQL

disable_compat_trim_mysql	MySQL兼容模式下，控制trim/ltrim/rtrim函数对入参的处理方式。 默认兼容MySQL。可通过设置此参数，回退到历史版本行为。 ● 不设置此选项时，兼容MySQL行为，匹配完整子串。 mysql_compatibility_db=# SELECT trim('{}{name}{}','{}'),trim('xyznamezyx','xyz'); btrim | btrim ---------+---------- {name} | namezyx (1 row) ● 设置此选项时，匹配字符集合中的单个字符。 mysql_compatibility_db=# SELECT trim('{}{name}{}','{}'),trim('xyznamezyx','xyz'); btrim | btrim --------+------- name | name (1 row)	MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
light_object_mtime	控制pg_object系统表mtime字段是否会记录对象行为的操作。 ● 设置此选项时，GRANT/REVOKE/TRUNCATE操作不被mtime记录即不更新mtime字段。 ● 不设置此选项时（默认行为），ALTER操作、COMMENT、GRANT/REVOKE和TRUNCATE均会被mtime记录即更新mtime字段。	ORA TD MyS QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

disable_including_all_mysql	MySQL兼容模式下，控制CREATE TABLE ... LIKE语法是否为INCLUDING_ALL模式。
	默认不设置此参数，即MySQL兼容模式下，CREATE TABLE ... LIKE语法默认为INCLUDING_ALL模式。
	可通过设置此参数，回退到历史版本行为。
	● 不设置此选项，MySQL兼容模式下，CREATE TABLE ... LIKE语法为INCLUDING_ALL模式。
	mysql_compatibility_db=# CREATE TABLE mysql_like(id int, name varchar(10), score int) DISTRIBUTE BY hash(id) COMMENT 'mysql_like';
	CREATE TABLE
	mysql_compatibility_db=# CREATE INDEX index_like ON mysql_like(name);
	CREATE INDEX
	mysql_compatibility_db=# \d+ mysql_like;
	Table "public.mysql_like"
	Column | Type | Modifiers | Storage | Stats target | Description
	--------+----------------------+-----------+----------+--------------+-------------
	id | integer | | plain | |
	name | character varying(10) | | extended | |
	score | integer | | plain | |
	Indexes:
	"index_like" btree (name) TABLESPACE pg_default
	Has OIDs: no
	Distribute By: HASH(id)
	Location Nodes: ALL DATANODES
	Options: orientation=row, compression=no
	
	mysql_compatibility_db=# CREATE TABLE copy_like like mysql_like;
	CREATE TABLE
	mysql_compatibility_db=# \d+ copy_like;
	Table "public.copy_like"
	Column | Type | Modifiers | Storage | Stats target | Description
	--------+----------------------+-----------+----------+--------------+-------------
	id | integer | | plain | |
	name | character varying(10) | | extended | |
	score | integer | | plain | |
	Indexes:
	"copy_like_name_idx" btree (name) TABLESPACE pg_default
	Has OIDs: no
	Distribute By: HASH(id)
	Location Nodes: ALL DATANODES
	Options: orientation=row, compression=no
	● 设置此选项，MySQL兼容模式下，CREATE TABLE ... LIKE语法为空模式。
	mysql_compatibility_db=# SET behavior_compat_options = 'disable_including_all_mysql';
	SET
	mysql_compatibility_db=# CREATE TABLE mysql_copy LIKE mysql_like;
	NOTICE: The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
	HINT: Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
	CREATE TABLE
	mysql_db=# \d+ mysql_copy;
		MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
	Table "public.mysql_copy" Column | Type | Modifiers | Storage | Stats target | Description --------+----------------------+-----------+----------+--------------+------------- id | integer | | plain | | name | character varying(10) | | extended | | score | integer | | plain | | Has OIDs: no Distribute By: ROUND ROBIN Location Nodes: ALL DATANODES Options: orientation=row, compression=no	
cte_onetime_inline	控制非stream计划是否执行inline。 ● 设置此选项时，非stream计划且只被引用一次的CTE执行inline。 ● 不设置此选项时，非stream计划且只被引用一次的CTE不执行inline。	ORA TD MyS QL
skip_first_after_mysql	MySQL兼容模式下，控制是否在ALTER TABLE ADD/MODIFY/CHANGE COLUMN中忽略FIRST/AFTER colname语法。 ● 设置此选项时，将忽略FIRST/AFTER colname语法，执行不报错。 mysql_compatibility_db=# SET behavior_compat_options = 'skip_first_after_mysql'; mysql_compatibility_db=# ALTER TABLE t1 ADD COLUMN b text after a; ALTER TABLE ● 不设置此选项时，将不支持FIRST/AFTER colname语法，执行报错。 mysql_compatibility_db=# SET behavior_compat_options = ''; mysql_compatibility_db=# ALTER TABLE t1 ADD COLUMN b text after a; ERROR: FIRST/AFTER is not yet supported.	MyS QL

enable_division_by_zero_mysql	MySQL兼容模式下，除数为0时，控制除法或取余操作是否报错。（该配置项仅8.1.3.110及以上集群版本支持。）
	● 设置此选项时，除法或取余操作除数为0时，返回NULL。
	compatible_mysql_db=# SET behavior_compat_options = 'enable_division_by_zero_mysql';
	SET
	compatible_mysql_db=# SELECT 1/0 AS test;
	test
	---------- 	
	
	(1 row)
	● 不设置此选项时，除法或取余操作除数为0时，执行报错。
	compatible_mysql_db=# SELECT 1/0;
	ERROR: division by zero
		MyS
		QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
normal_session_id	控制是否生成normal格式的会话ID。 ● 设置此选项时，生成normal格式的会话ID，兼容8.1.3及之前集群版本的会话ID。 SET behavior_compat_options='normal_session_id'; SELECT pg_current_sessionid(); pg_current_sessionid ---------- 1660268184.140594655524608 (1 row) ● 不设置此选项时，生成pretty格式的会话ID。 SET behavior_compat_options=''; SELECT pg_current_sessionid(); pg_current_sessionid ---------- 1660268184.140594655524608.coordinator1 (1 row)	ORA TD MySQL
disable_jsonb_exact_match	控制操作符匹配规则，在对二元操作符进行模糊匹配时是否考虑jsonb类型。 ● 设置此选项时，当对操作符进行模糊匹配时，会在所有备选（即不区分jsonb类型）中匹配，兼容8.1.1之后集群版本的匹配规则。 SET behavior_compat_options='disable_jsonb_exact_match'; select '2022' - '2'::text; ERROR: cannot delete from scalar ● 不设置此选项时，当对操作符进行模糊匹配时，会在不含jsonb类型的备选中匹配，兼容8.1.1之前集群版本的匹配规则。 SET behavior_compat_options=''; select '2022' - '2'::text; ?column? ---------- 2020 (1 row)	ORA TD MySQL
merge_into_with_trigger	控制是否支持对有触发器的表执行MERGE INTO操作。 ● 设置此选项时，可以对有触发器的表执行MERGE INTO操作。注意在MERGE INTO操作执行时，表上的触发器不会被触发执行。 ● 不设置此选项时，对有触发器的表执行MERGE INTO操作时报错。	ORA TD MySQL
add_column_default_v_func	控制alter table add column default expression中expression是否支持volatile类型的函数。 ● 设置此选项时，alter table add column default expression中的expression支持volatile类型的函数。 ● 不设置此选项时，alter table add column default expression中expression不支持volatile类型的函数，如果expression中有volatile的函数，该语句执行会报错。	ORA TD MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_full_group_by_mysql	MySQL兼容模式下，控制查询中GROUP BY后是否可以不体现任何非聚合函数查询字段。 ● 设置此选项时，查询允许GROUP BY后不体现任何非聚合函数查询字段。 SET behavior_compat_options='disable_full_group_by_mysql'; SELECT a,b FROM t1 GROUP BY a; a | b ---+--- 1 | 1 2 | 2 (2 rows) ● 不设置此选项时，查询不允许GROUP BY后不体现任何非聚合函数查询字段，查询报错。 SET behavior_compat_options=''; SELECT a,b FROM t1 GROUP BY a; ERROR: column "t1.b" must appear in the GROUP BY clause or be used in an aggregate function LINE 1: SELECT a,b FROM t1 GROUP BY a; **注意** 该参数需结合full_group_by_mode使用，具体请参考**full_group_by_mode**。 此配置项配置后，若full_group_by_mode为notpadding，对于非GROUP BY后非聚合查询字段需要保证分组后数据一致，否则该列将为随机值。	MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
disable_gc_fdw_filter_partial_pushdown	协同分析外表（类型为gc_fdw）场景下，控制使用过滤条件查询外表数据时过滤条件的下推情况。 ● 设置此选项时，过滤条件中若存在不满足下推条件的因素（如非immutable函数），为了保证结果集，则全部过滤条件不下推，此行为兼容8.2.1版本之前的行为。 --源端集群建表 CREATE TABLE t1(c1 INT, c2 INT, c3 INT) DISTRIBUTE BY HASH(c1); --本地集群建相同结构外表 CREATE SERVER server_remote FOREIGN DATA WRAPPER gc_fdw options(ADDRESS 'address', DBNAME 'dbname', USERNAME 'username', PASSWORD 'password'); CREATE FOREIGN TABLE t1(c1 INT, c2 INT, c3 INT) SERVER server_remote; --打开参数时，条件下推情况 SET behavior_compat_options = 'disable_gc_fdw_filter_partial_pushdown'; EXPLAIN (verbose on,costs off) SELECT * FROM t1 WHERE c1>3 AND c2 <100 AND now() - '20230101' < c3; QUERY PLAN --- --- Streaming (type: GATHER) Output: c1, c2, c3 Node/s: All datanodes -> Foreign Scan on ca_schema.t1 Output: c1, c2, c3 Filter: ((t1.c1 > 3) AND (t1.c2 < 100) AND ((now() - '2023-01-01 00:00:00-08'::timestamp with time zone) < (t1.c3)::interval)) Remote SQL: SELECT c1, c2, c3 FROM ca_schema.t1 (7 rows) ● 不设置此选项时，过滤条件中可下推的部分将下推到源端集群执行，不可下推部分将在本地集群执行，如此可提升外表查询效率。 --关闭参数时，条件下推情况 SET behavior_compat_options = ''; EXPLAIN (verbose on,costs off) SELECT * FROM t1 WHERE c1>3 AND c2 <100 AND now() - '20230101' < c3; QUERY PLAN --- --- Streaming (type: GATHER) Output: c1, c2, c3 Node/s: All datanodes -> Foreign Scan on ca_schema.t1 Output: c1, c2, c3 Filter: ((now() - '2023-01-01 00:00:00-08'::timestamp with time zone) < (t1.c3)::interval) Remote SQL: SELECT c1, c2, c3 FROM ca_schema.t1 WHERE ((c1 > 3)) AND ((c2 < 100)) (7 rows)	ORA TD MyS QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
ignore_unshipped_concurrent_update	并发更新场景下，当前会话的语句不下推时，如果更新的元组被其他会话更新成新元组后，控制当前会话执行UPDATE/DELETE语句时是否忽略处理新的元组。默认不再处理新的元组。 ● 设置此选项时，当前会话执行UPDATE/DELETE语句时忽略处理新的元组，当前UPDATE/DELETE语句执行成功，此行为会导致并发更新场景数据不一致。此行为兼容8.2.1版本之前的行为。 ● 不设置此选项时，当前会话执行UPDATE/DELETE语句检测元组已被更新时，会重新执行当前会话的UPDATE/DELETE语句，以保证数据一致性。语句执行重试次数受**max_query_retry_times**参数控制。	ORA TD MySQL
disable_set_global_var_on_datanode	控制set_config函数是否可以在DN上设置全局变量。 ● 设置此选项时，禁止set_config函数在DN上设置全局变量。此行为默认兼容8.2.1版本之前的行为。 ● 不设置此选项时，允许set_config函数在DN上设置全局变量，这将会使得CN与DN上全局变量值不一致，导致read_global_var函数下推时可能出错。	ORA TD MySQL

variadic_null_check	控制variadic参数是否能传入NULL参数的选项，默认不开启。（该参数仅8.3.0及以上集群版本支持）
	● 设置此选项时，禁止variadic传入NULL参数，传入后会报错。
	`SET behavior_compat_options = 'variadic_null_check';`
	`SELECT format ( 'array', VARIADIC NULL);`
	`ERROR:  VARIADIC parameter must be an array`
	说明
	为了兼容mysql，compat_concat_variadic开启时，对concat函数和concat_ws函数不会生效，仍然可以传入NULL参数。
	● 不设置此选项时，允许variadic传入NULL参数。
	`SET behavior_compat_options = '';`
	`SELECT format ( 'array', VARIADIC NULL);`
	` format`
	`--------`
	` array`
	`(1 row)`
		ORA
		TD
		MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_use_syscol_in_replicate_table	控制复制表在INSERT、UPDATE、MERGE INTO和DELETE时是否可以采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件。该选项默认不设置。 ● 不设置该选项时，如果复制表在INSERT、UPDATE、MERGE INTO和DELETE时采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件，会报如下错误： ERROR: Can not use system column oid/ctid/tableoid/xc_node_id in Replication Table. ● 设置该选项时，可在复制表中使用系统列id/ctid/tableoid/xc_node_id进行INSERT、UPDATE、MERGE INTO和DELETE。 **注意** 复制表在INSERT、UPDATE、MERGE INTO和DELETE时采用oid/ctid/tableoid/xc_node_id作为过滤条件、连接条件和having条件，语句有导致集群core的风险，请慎重考虑配置该选项。	ORA TD MYSQL
enable_force_add_batch	当参数support_batch_bind设置为on时，JDBC的addBatch入库采用多路并行执行策略，该策略要求**enable_fast_query_shipping**和**enable_light_proxy**参数同时为on。 ● 当behavior_compat_options包含选项enable_force_add_batch时，若上述依赖条件不满足，系统将自动选择单条串行执行策略，性能会明显下降。 ● 当behavior_compat_options不包含选项enable_force_add_batch时，若上述依赖条件不满足，系统将报错并提示业务检查相关参数配置。	ORA TD MYSQL
disable_mergesort_without_material	控制当前stream片段是否含有物化算子时stream算子采用merge sort。 ● 设置此选项时，若当前stream片段含有物化算子（material、sort、agg、CteScan），则可采用merge sort，否则不可采用merge sort。 ● 不设置此选项时，是否采用merge sort不需要判断当前stream片段是否含有物化算子。	ORA TD MYSQL
enable_pushdown_groupingset_subquery	当子查询中含有grouping set时，该选项控制是否可将外层查询中仅与该子查询相关的条件下推到子查询中。 ● 当子查询中含有grouping set时，设置此选项，则不可将外层查询的条件下推到子查询中。 ● 当子查询中含有grouping set时，不设置此选项，则将外层查询的条件下推到子查询中。	ORA TD MYSQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_whole_row_var	该参数主要涉及两个场景。1. 控制是否允许表或视图出现在SQL表达式中，包括但不限于查询的目标列表、GROUP BY列表等；2. 控制是否允许非表的record出现的SQL表达式。该选项仅8.3.0及以上集群版本支持。 ● 设置此选项时，允许表或视图出现在SQL表达式中。 SET behavior_compat_options = 'enable_whole_row_var'; SELECT a1 FROM t a1; a1 ---- (0 rows) SELECT t FROM (SELECT 1) as t; t ----- (1) (1 rows) ● 不设置此选项时，不允许表或视图出现在SQL表达式中，如果SQL中出现则报错。 SET behavior_compat_options = ''; SELECT a1 FROM t a1; ERROR: Table or view cannot appear in expression. Table/view name: t, alias: a1. Please check targetList, groupClause etc. SELECT t FROM (SELECT 1) as t; ERROR: Non-table records cannot appear in expression. alias: t. Please check targetList, groupClause etc.	ORA TD MYS QL
enable_unknown_datatype	控制是否允许创建含有unknown类型列的表。该选项仅8.3.0及以上集群版本支持。 ● 设置此选项时，允许创建含有unknown类型列的表。 **注意** unknown类型在数据库中通常用于处理临时数据或中间结果，不适合作为持久化数据的存储类型，不建议用户直接指定表的字段类型为unknown类型，因此也不建议设置该选项。 ● 不设置此选项时，不允许创建含有unknown类型列的表，如果建表SQL含有unknown列，则报错。	ORA TD MYS QL
alter_distribute_key_by_partition	控制ALTER TABLE修改分区表分布列时INSERT INTO是否按分区执行。 ● 设置此选项时，按分区执行INSERT INTO，使用内存降低但性能劣化。 ● 不设置此选项时，将分区表整表进行INSERT INTO，性能较好但使用内存较多。	ORA TD MYS QL
disable_update_returning_check	控制是否禁用涉及多表关联、更新复制表且带returning语句场景。该选项仅8.3.0及以上集群版本支持。 ● 不设置此选项时，更新复制表且带returning语句场景，若涉及多表关联，则报下述错误： ERROR: Unsupported FOR UPDATE replicated table joined with other table. ● 设置此选项，与旧版本前向兼容，但是在更新复制表且带returning语句场景，若涉及多表关联，存在结果集不一致问题。	ORA TD MYS QL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

check_function_shippable	控制是否检查自定义plpgsql/SQL函数的属性。该选项仅8.3.0及以上集群版本支持。
	• 不设置此配置项时，不检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性。
	• 设置此配置项时，会遵循以下逻辑检查自定义函数的IMMUTABLE/STABLE/VOLATILE属性：
	– 设置白名单：对于**DBMS_OUTPUT**的3个函数，跳过check_function_shippable的检测。
	– 如果自定义函数内部有DML语句，外层是IMMUTABLE或者SHIPPABLE，均是下推，故报错。
	– 如果自定义函数外层是shippable，内层是immutable，则通过检查。
	– 如果自定义函数外层是shippable，内层是非immutable且内层是shippable，则通过检查。
	– 如果函数外层是shippable，但是内层非上述的情况，则报错。
	例如：设置此参数时，以下场景下会执行报错：
	`CREATE OR replace function func_ship(a int)`
	`returns int`
	`LANGUAGE plpgsql`
	`NOT FENCED  SHIPPABLE`
	`AS $function$`
	`begin`
	`perform test_ship();`
	`return a;`
	`EXCEPTION WHEN OTHERS THEN`
	`return a;`
	`end $function$;`
	`select func_ship(a) from tt3;`
	`ERROR:  parent function is shippable but child is not immutable or shippable.`
		ORA
		TD
		MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_full_string_agg	控制string_agg(a, delimiter) over (partition by b order by c)场景行为，采用窗口内的全量聚合逻辑还是增量聚合逻辑。该参数8.3.0及以上集群版本支持。 不设置此选项时，采用增量聚合逻辑。设置此选项时，采用窗口内的全量聚合逻辑。默认不设置此选项。 CREATE TABLE string_agg_dn_col(c1 int, c2 text) WITH(orientation = column) distribute by hash(c1); INSERT INTO string_agg_dn_col values(1, 'test'); INSERT INTO string_agg_dn_col values(1, 'haidian'); INSERT INTO string_agg_dn_col values(1, 'nanjing'); SELECT t.c1 AS c1, string_agg(t.c2, ',') OVER(PARTITION BY t.c1 ORDER BY t.c2) AS c2 FROM string_agg_dn_col t ORDER BY c2; c1 | c2 ----+---------------------- 1 | haidian 1 | haidian,nanjing 1 | haidian,nanjing,test (3 rows) SET behavior_compat_options='enable_full_string_agg'; SELECT t.c1 AS c1, string_agg(t.c2, ',') OVER(PARTITION BY t.c1 ORDER BY t.c2) AS c2 FROM string_agg_dn_col t ORDER BY c2; c1 | c2 ----+---------------------- 1 | haidian,nanjing,test 1 | haidian,nanjing,test 1 | haidian,nanjing,test (3 rows)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式

enable_banker_round	控制数值类型舍入行为使用四舍五入还是银行家算法。该选项仅8.3.0及以上集群版本支持。
	受到参数控制的行为包括：
	● INSERT INTO和::xxx指定类型时自动触发的类型转换：所有整数类型（int1，int2，int4，int8）、所有任意精度类型（decimal，numeric，number）和money类型。
	● 对numeric类型的舍入和转换函数：round(xxx.xx,s),cast('xxx.xx',numeric),to_char(xxx.xx,'xxx')。
	● numeric类型的数学计算。
	说明
	银行家算法舍入规则：舍入位后的值大于5时，进位；舍入位后的值小于5时，舍去不进位；舍入位后的值等于5且全为零时，则按照修约前数字奇偶性决定舍入，奇数进位，偶数舍去；舍入位后的值等于5且存在非零数字，则直接进位。
	● 设置选项时，舍入行为采用银行家算法：
	`SET behavior_compat_options = enable_banker_round;`
	`SELECT 1.5::int1,1.5::int2,1.5::int4,1.5::int8,1.5::numeric(10,0),1.115::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 2 | 2 | 2 | 2 | 2 | $1.12`
	`SELECT 0.5::int1,0.5::int2,0.5::int4,0.5::int8,0.5::numeric(10,0),1.105::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 0 | 0 | 0 | 0 | 0 | $1.10`
	`SELECT round(1.05,1),round(1.15,1),cast('1.05',numeric(10,1)),cast('1.15',numeric(10,1)),to_char(1.05,'9D9'),to_char(1.15,'9D9');`
	` round | round | numeric | numeric | to_char | to_char`
	`-------+-------+---------+---------+---------+---------`
	` 1.0 | 1.2 | 1.0 | 1.2 | 1.0 | 1.2`
	● 不设置选项时，舍入行为采用四舍五入：
	`SET behavior_compat_options = '';`
	`SELECT 1.5::int1,1.5::int2,1.5::int4,1.5::int8,1.5::numeric(10,0),1.115::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 2 | 2 | 2 | 2 | 2 | $1.12`
	`SELECT 0.5::int1,0.5::int2,0.5::int4,0.5::int8,0.5::numeric(10,0),1.105::money;`
	` int1 | int2 | int4 | int8 | numeric | money`
	`------+------+------+------+---------+-------`
	` 1 | 1 | 1 | 1 | 1 | $1.11`
	`SELECT round(1.05,1),round(1.15,1),cast('1.05',numeric(10,1)),cast('1.15',numeric(10,1)),to_char(1.05,'9D9'),to_char(1.15,'9D9');`
	` round | round | numeric | numeric | to_char | to_char`
	`-------+-------+---------+---------+---------+---------`
	` 1.1 | 1.2 | 1.1 | 1.2 | 1.1 | 1.2`
		ORA
		TD
		MYS
		QL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
create_partition_local_index	控制在分区表上默认创建的索引是全局（GLOBAL）索引还是本地（LOCAL）索引。该选项仅8.2.1.210及以上集群版本支持。 ● 不设置此选项时，在分区表上默认创建全局（GLOBAL）索引。 SET behavior_compat_options = ''; CREATE INDEX sale_id_idx ON sales(sale_id); ERROR: partitioned table does not support global index HINT: please set behavior_compat_options = 'create_partition_local_index'to create local index by default. ● 设置此选项时，在分区表上默认创建本地（LOCAL）索引。 SET behavior_compat_options = create_partition_local_index; CREATE INDEX sale_id_idx ON sales(sale_id); CREATE INDEX	ORA
enable_int_division_by_truncate	控制整数除法行为结果集输出整数还是浮点数，行为兼容PG或者兼容ORA。 ● 设置此选项时，整数除法结果输出整数，小数位截断，兼容PG行为。 SET behavior_compat_options = 'enable_int_division_by_truncate'; SELECT 8::int8 / 3::int8, 8::int4 / 3::int4, 8::int2 / 3::int2, 8::int1 / 3::int1; ?column? | ?column? | ?column? | ?column? ----------+----------+----------+---------- 2 | 2 | 2 | 2 (1 row) ● 不设置此选项时，整数除法结果输出浮点数，包含小数位，兼容ORA行为。 SET behavior_compat_options = ''; SELECT 8::int8 / 3::int8, 8::int4 / 3::int4, 8::int2 / 3::int2, 8::int1 / 3::int1; ?column? | ?column? | ?column? | ?column? ------------------+------------------+------------------+------------------ 2.66666666666667 | 2.66666666666667 | 2.66666666666667 | 2.66666666666667 (1 row)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
select_into_allow_multi_result	控制存储过程中SELECT INTO是否允许接收多行结果或者没有结果集。 ● 不设置此选项时，存储过程中使用SELECT INTO语句插入多值或者空时，执行报错： CREATE TABLE PersonTmpTable (id int primary key, name varchar(64), age int, city varchar(512) default null, update_time timestamp default null); INSERT INTO PersonTmpTable VALUES(1,'zhangsan', 23, 'wuhan', '2022-12-10 15:39:32'); INSERT INTO PersonTmpTable VALUES(2,'lisi', 11, 'beijing', '2022-12-13 15:39:32'); INSERT INTO PersonTmpTable VALUES(3,'wangwu', 46, 'xian', '2022-12-1 15:39:32'); INSERT INTO PersonTmpTable VALUES(4,'zhaoliu', 46, 'wuhan', '2022-12-31 15:39:32'); SET behavior_compat_options = ''; CREATE OR REPLACE function func_test1 RETURNS int LANGUAGE plpgsql AS $$DECLAREtmp INTEGER;BEGINSELECT id INTO tmp FROM PersonTmpTable;dbms_output.put_line(tmp);return tmp;END;$$;select func_test1(); ERROR: query returned no rows when process INTO. ERROR: query returned 2 rows more than one row. ● 设置此选项，存储过程中使用SELECT INTO语句插入多值或者空时，成功插入第一行或者空。 SET behavior_compat_options = 'select_into_allow_multi_result'; CREATE OR REPLACE function func_test1 RETURNS int LANGUAGE plpgsql AS $$DECLAREtmp INTEGER;BEGINSELECT id INTO tmp FROM PersonTmpTable;dbms_output.put_line(tmp);return tmp;END;$$;select func_test1(); CREATE FUNCTION	ORA TD MYSQL
orderby_null_first	控制ORDER BY排序时NULL值是否默认视为最小值。 ● 设置此选项时，ORDER BY排序时NULL值默认视为最小值。 SET behavior_compat_options = 'orderby_null_first'; SELECT * FROM test ORDER BY a; a | b ---+--- | 1 1 | 2 ● 不设置此选项时，ORDER BY排序时NULL值默认视为最大值。 SET behavior_compat_options = ''; SELECT * FROM test ORDER BY a; a | b ---+--- 1 | 2 | 1	TD
enable_cast_hashjoin	控制在SQL语句的解析阶段，在处理比较操作语句时，针对不支持hash的等值比较操作符，判断是否能转换为一致的数据类型，若能转测，则内部转换比较操作符，使后续join操作能够使用hashjoin以提升查询性能。该选项仅8.3.0及以上集群版本支持。	ORA TD MYSQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
unsupported_set_function_case	控制是否支持case when条件中含有返回为多结果集函数。该选项仅8.2.1.261及以上集群版本支持。新安装的集群版本中默认开启该参数。 ● 设置此选项时，列存不支持case when条件中含有返回为多结果集函数： CREATE TABLE t1(id int, c1 text) with(orientation=column); INSERT INTO t1 values(1, 'a#1'); SET behavior_compat_options = 'unsupported_set_function_case'; SELECT CASE split_part(regexp_split_to_table(c1, E'\,'),'#',1) when 'a' then c1 else null end from t1; ERROR: set-valued function called in context that cannot accept a set ● 不设置此选项时，列存支持case when条件中含有返回为多结果集函数： SET behavior_compat_options = ''; SELECT CASE split_part(regexp_split_to_table(c1, E'\,'),'#',1) when 'a' then c1 else null end from t1; case ------ a#1 (1 row)	ORA TD MYSQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_change_search_path	控制在形成通用计划generic_plan后能否修改搜寻路径。该选项仅9.1.0及以上集群版本支持。 ● 不设置此选项时，当设置新的搜索路径（search_path）并执行EXECUTE语句，数据库将仍然从原表的schema下寻找对应表。 CREATE SCHEMA s1; CREATE TABLE abc(f1 INT); CREATE SCHEMA s2; CREATE TABLE abc(f1 INT); SET search_path = s1; INSERT INTO s1.abc VALUES(123);INSERT INTO s2.abc VALUES(456); SET search_path = s1; PREPARE p1 AS SELECT f1 FROM abc; EXECUTE p1; f1 ----- 123 (1 row) SET search_path = s2; SELECT f1 FROM abc; f1 ----- 456 (1 row) EXECUTE p1; f1 ----- 123 (1 row) ● 设置此选项时，当设置新的搜索路径（search_path）并执行EXECUTE语句，数据库将从新设置的搜索路径中寻找对应表。 SET behavior_compat_options = 'enable_change_search_path'; EXECUTE p1; f1 ----- 456 (1 row) SET search_path = s1; EXECUTE p1; f1 ----- 123 (1 row)	TD
enable_varchar_to_nvarchar2	控制通过DDL语句创建及更新的varchar字段是否自动切换为nvarchar2字段。该选项仅9.1.0及以上集群版本支持。 ● 设置此选项时，通过DDL语句创建/更新的varchar字段自动切换为nvarchar2字段。 ● 不设置此选项时，通过DDL语句创建/更新的varchar字段不会切换为nvarchar2字段。	ORA TD MYSQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
normalize_negative_zero	控制ceil()，round()函数在处理float类型特定值时返回-0与否。 ● 设置此选项时，ceil()处理(-1,0)，round()处理[-0.5, 0)时返回值会返回0。 SET behavior_compat_options='normalize_negative_zero'; SELECT ceil(cast(-0.1 as float)); ceil ------ 0 (1 row) SELECT round(cast(-0.1 as FLOAT)); round ------- 0 (1 row) ● 不设置此选项时，ceil()处理(-1,0)，round()处理[-0.5, 0)时返回值会返回-0。 SET behavior_compat_options = ''; SELECT ceil(cast(-0.1 as FLOAT)); ceil ------ -0 (1 row) SELECT round(cast(-0.1 as FLOAT)); round ------- -0 (1 row)	ORA TD MySQL
disable_client_detection_commit	控制在每次事务提交之前，检测与客户端的连接是否存在。如果不存在，则报错，回滚该事务，防止因断连未感知重复下发导致的数据重复问题。 ● 不设置此选项时，每次事务提交之前检测一次客户端连接是否存在。 ● 设置此选项时，事务提交之前不检测客户端连接是否存在。	ORA TD MySQL
change_illegal_char	控制gds读取UTF8非法字符的显示问题。该选项仅8.3.0.100及以上集群版本支持。 该选项打开时，gds读取到UTF8不兼容的非法字符从“？”变为显示“&#xfffd;”。	MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
row_use_pseudo_name	控制row相关表达式是否对匿名列生成伪列名。该参数仅9.1.0.100及以上集群版本支持。 ● 不设置此选项时，row表达式中若存在对应的真实列名，则使用真实的列名，若本身为匿名列，则生成f1、f2...fn的伪列名。 SELECT row_to_json(row(1,'foo')); row_to_json ---------- {"f1":1,"f2":"foo"} (1 row) CREATE TABLE json_tbl(id INT, x INT, y text) WITH (ORIENTATION = COLUMN); INSERT INTO json_tbl VALUES (1, 1, 'txt1'), (2, 2, 'txt2'), (3, 3, 'txt3'); SELECT to_json(t.*) FROM json_tbl t; to_json ---------- {"id":3,"x":3,"y":"txt3"} {"id":1,"x":1,"y":"txt1"} {"id":2,"x":2,"y":"txt2"} (3 rows) ● 设置此选项时，row表达式中将在列存表条件下生成f1、f2...fn的伪列名。 SET behavior_compat_options='ROW_USE_PSEUDO_NAME'; SELECT to_json(t.*) FROM json_tbl t; to_json ---------- {"f1":3,"f2":3,"f3":"txt3"} {"f1":1,"f2":1,"f3":"txt1"} {"f1":2,"f2":2,"f3":"txt2"} (3 rows)	ORA TD MySQL
enable_trunc_orc_string	控制orc格式外表字段为varchar(n)，但是orc文件字段类型为string，且string长度超过n时，外表查询的行为。 ● 不设置此选项时，查询报错，提示字段超长。 ● 设置此选项时，查询正常，按照varchar(n)定义长度进行截断。	ORA TD MySQL
gds_fill_multi_missing_fields	控制GDS外表容错性参数fill_missing_fields设置为true/on时的行为。在GDS外表fill_missing_fields设置为true/on时，允许数据源文件一行中最后若干字段缺失，处理方式是将这些字段设置为NULL。而在此之前，仅允许数据源文件一行中最后一个字段缺失，否则报错。 ● 设置此选项时，GDS外表允许数据源文件一行最后多个字段缺失。 ● 不设置此选项时，GDS外表允许数据源文件一行最后一个字段缺失。兼容历史行为。	ORA TD MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
force_german_dmy	该选项控制时间日期输出格式german是否必须搭配dmy日期顺序。 设置该选项时，强制要求german格式必须搭配dmy日期顺序，防止在系统中因输出格式冲突，导致日期日月颠倒的数据错误。	ORA TD MySQL
inner_iso_datestyle	在SET datestyle = 'Postgres, MDY' 设置下，CN将输入的时间字符串转换为包含本地时区缩写CST的字符串发送给DN。由于CST可对应多个时区，DN解析时可能错误识别时区，导致CN与DN的数据校验失败。 该场景下设置该选项后，CN发送给DN的时间字符串将统一采用ISO标准格式，确保DN能够准确解析。	ORA TD MySQL

correct_convert_tz	控制convert_tz函数查询结果是否与MySQL行为一致。该选项仅9.1.0.210及以上集群版本支持。
	● 设置此选项时，convert_tz函数在时区转换时返回预期结果，与MySQL行为一致。
	`set behavior_compat_options = 'correct_convert_tz';`
	`select convert_tz('2020-1-1 12:00', '+00:00', 'PRC');`
	`convert_tz`
	`---------------------`
	`2020-01-01 20:00:00`
	`(1 row)`
	● 不设置此选项时，convert_tz函数在时区转换时结果不符合预期，与MySQL行为不一致。
	`set behavior_compat_options = '';`
	`select convert_tz('2020-1-1 12:00', '+00:00', 'PRC');`
	`convert_tz`
	`---------------------`
	`2020-01-01 04:00:00`
	`(1 row)`
		ORA
		TD
		MySQL

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
set_timezone_mysql_style	控制set timezone='+08:00' / '-07:00'等类型时（'+/- 时:分'形式）是否与MySQL行为一致。该选项仅9.1.0.210及以上集群版本支持。 ● 设置此选项时，set timezone='+08:00'表示时区设置为东八区，与MySQL行为一致。 set behavior_compat_options = 'set_timezone_mysql_style'; set timezone='UTC'; select * from t; a ------------------------ 2020-01-01 12:00:00+00 (1 row) set timezone='+08:00'; select * from t; a ------------------------ 2020-01-01 20:00:00+08 (1 row) ● 不设置此选项时，set timezone='+08:00'表示时区设置为西八区，与MySQL行为不一致，为PG行为。 set behavior_compat_options = ''; set timezone='UTC'; select * from t; a ------------------------ 2020-01-01 12:00:00+00 (1 row) set timezone='+08:00'; select * from t; a ------------------------ 2020-01-01 04:00:00-08 (1 row)	ORA TD MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
parquet_int96_is_localtime	控制读取parquet格式外表的int96类型数据（timestamp专用）时，是否将存储的原始时间戳使用数据库时区修正。该选项仅9.1.0.210及以上集群版本支持。 ● 设置此选项时，认为parquet int96类型存储的是本地时间（非UTC+0），读取结果不会再进行时区修正。 set behavior_compat_options = 'parquet_int96_is_localtime'; set time zone 'Asia/Shanghai'; select * from test_parquet_int96; a --------------------- 2025-01-25 00:00:00 (1 row) set time zone 'Atlantic/Cape_Verde'; select * from test_parquet_int96; a --------------------- 2025-01-25 00:00:00 (1 row) ● 不设置此选项时，认为parquet int96类型存储是UTC+0时间，为了兼容spark、hive等引擎，读取结果会使用数据库时区进行修正。 set behavior_compat_options = ''; set time zone 'Asia/Shanghai'; select * from test_parquet_int96; a --------------------- 2025-01-25 08:00:00 (1 row) set time zone 'Atlantic/Cape_Verde'; select * from test_parquet_int96; a --------------------- 2025-01-24 23:00:00 (1 row)	ORA TD MySQL
disable_vfork_create_subprocess	禁止使用vfork实现DWS子进程启动。该选项仅9.1.1.100及以上集群版本支持。 ● 设置此选项时，DWS子进程启动禁止使用vfork优化。 ● 不设置此选项时，DWS子进程启动使用vfork启动，避免父子进程间copy虚拟内存产生的mmap写锁阻塞。	ORA TD MySQL
disable_user_thread_stack	禁止使用用户态内存作为线程栈内存。该选项仅9.1.1.100及以上集群版本支持。 ● 设置此选项时，DWS启动子线程时使用Linux内核态临时内存。 ● 不设置此选项时，DWS启动子线程时使用用户态内存池，避免申请临时栈内存产生的mmap写锁阻塞。	ORA TD MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
turbo_support_nan	对于turbo表，当numeric(p,s) p<=38时，其不允许插入NAN（ Not a Number，非数字 ）。该选项9.1.1.100及以上集群版本支持。 ● 设置此选项时会自动在建表时修改numeric列属性，将p<=38修改至p=39，允许插入NAN。 ● 不设置此选项时，不修改建表属性。	ORA TD MySQL

view_star_expand	在通过pg_get_viewdef或\d+等操作获取视图定义时，该参数控制是否对语句中的*字段进行展开。
	该选项8.2.1集群版本中8.2.1.251及以上版本支持；9.1.1.100及以上集群版本支持。
	● 设置此选项时，语句中的*字段会展示为创建视图时依赖的具体字段。
	`create table t1(a int, b int, c int);`
	`create view v1 as select t1.*, 1 as d from t1;`
	`set behavior_compat_options='on(view_star_expand)';`
	`SET`
	`\d+ v1`
	`View "public.v1"`
	`Column | Type | Modifiers | Storage | Description`
	`--------+---------+-----------+---------+-------------`
	`a | integer | | plain |`
	`b | integer | | plain |`
	`c | integer | | plain |`
	`d | integer | | plain |`
	`View definition:`
	`SELECT t1.a, t1.b, t1.c, 1 AS d`
	`FROM t1;`
	● 不设置此选项时，语句中的*字段按照*字段进行展示，但是实际查找还是创建视图时依赖的具体字段，与之前历史版本保持一致。
		ORA
		TD
		MySQL

disable_srf_condition_expr	控制在case、coalesce、nvl、ifnull条件表达式中是否支持使用返回多结果集的函数。该选项仅9.1.1.100及以上集群版本支持。 ● 设置后不支持在case、coalesce、nvl、ifnull条件表达式中使用返回多结果集的函数。 ● 不设置此选项时，不限制case、coalesce、nvl、ifnull条件表达式中使用返回多结果集的函数。	ORA TD MySQL
enable_pl_with_txnctl	控制是否支持在存储过程中调用commit、rollback管理部分语句的提交和回滚。该选项仅9.1.1.100及以上集群版本支持。 ● 设置此选项时，支持在存储过程中调用commit、rollback进行事务控制。 ● 不设置此选项时，调用带有commit、rollback的存储过程时会报错。	ORA

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
overwrite_partition_optimization	控制指定分区的insert overwrite是否进行单分区的优化。该选项仅9.1.1.100及以上集群版本支持。 ● 设置此选项时，过程中的临时表建为普通表，执行性能较快。 ● 不设置此选项时，过程中的临时表建为分区表，执行性能较慢。	ORA TD MySQL
job_broken_disable	控制Job针对两次任务相同nextRunDate场景下的处理。该选项仅9.1.0.226及以上集群版本支持。 ● 设置此选项时，系统将不会标记此类任务为异常（broken），任务将继续执行。 ● 不设置此选项时，系统将认为任务异常，任务停止执行。	ORA TD MySQL
disable_vectorization_json	控制行存表中的json/jsonb类型是否使用强制向量化。该参数9.1.0.226及以上集群版本支持。 ● 设置此选项时，行存表中的json/jsonb类型不使用强制向量化。 ● 不设置此选项时，行存表中的json/jsonb类型使用强制向量化。	ORA TD MySQL
enable_text_to_unknown	控制insert unknown类型列执行pgxc plan时是否支持将text类型转为unknown类型。该选项仅9.1.1.100及以上集群版本支持。	ORA TD MySQL
cte_mem_exceeds_not_inline_all	控制某个CTE的结果集估算占用内存大小超过stream_ctescan_max_estimate_mem设置的阈值时，只有内存估算超阈值的CTE被inline，而不是整个计划中的所有CTE都进行inline处理。该选项仅9.1.1.100及以上集群版本支持。	ORA TD MySQL
ora_void_return_null	控制ORA兼容模式下CN和DN上的void类型函数返回值是否保持一致。该选项仅9.1.1.100及以上集群版本支持。 ● 设置此选项时，CN和DN上的void类型函数返回值保持一致返回NULL。 ● 不设置此选项时，void类型的函数在CN上返回空串，而在DN上返回NULL。	ORA

版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
enable_vacuum_update_datfrozenxid	控制在VACUUM FULL过程中是否执行更新datfrozenxid64字段的操作。 该选项8.2.1集群版本中8.2.1.200及以上版本支持；9.1.0.210及以上集群版本支持。 设置此选项时需要开启autovacuum，如果autovacuum参数没有开启，即使没有添加enable_vacuum_update_datfrozenxid配置项也会执行更新datfrozenxid64字段的操作。在数据库中表较多时，设置该选项可能会拖慢VACUUM FULL操作在小表上的执行时间。 ● 设置此选项时，VACUUM FULL会执行更新datfrozenxid64字段的操作。 ● 不设置此选项时，默认VACUUM FULL不执行更新datfrozenxid64字段的操作。	ORA TD MySQL

iden_spec_char	控制是否允许用户创建的名字中带有特殊字符。该参数仅9.1.1.200及以上集群版本支持。该功能为Beta特性，如需使用，请联系技术支持。
	● 设置此选项时，创建角色、用户、数据库时， 允许对应数据库对象名称中带有特殊字符。
	`set behavior_compat_options = 'iden_spec_char';`
	`CREATE USER "userName.with@Point" WITH PASSWORD '********';`
	`CREATE USER`
	● 不设置此选项时，不允许用户创建的用户名中带特殊字符。
	`set behavior_compat_options = '';`
	`CREATE ROLE ".@roleName.with@Point" WITH PASSWORD '********';`
	`ERROR:  invalid name: .@roleName.with@Point`
		MySQL

enable_revoke_permission_warning	当回收的权限与现有权限相同时，且回收权限失败，该参数控制系统是否返回提示信息。 该参数9.1.0.226及以上的9.1.0.xxx集群版本支持；9.1.1.200及以上的9.1.1.xxx集群版本支持。 设置此参数时，回收权限失败会显示如下类似提示信息： `WARNING: Relation-level privilege *SELECT* revoke not executed` 不设置此参数时，回收权限失败不会显示提示信息。	ORA TD MySQL
disable_timetimestamp_comparison	设置该选项时time和timestamp数据类型不允许进行隐式类型转换。该选项仅9.1.1.200及以上集群版本支持。 新装集群默认设置该参数，升级场景下默认不设置该参数。	ORA TD MySQL

 版权所有 © 华为云计算技术有限公司

兼容性配置项	兼容性行为控制	适用兼容模式
prefer_alias_in_having_clause	该选项用于控制Having子句中列名的解析顺序。该选项9.1.0.226及以上的9.1.0.xxx集群版本支持；9.1.1.200及以上集群版本支持。 ● 设置该选项时，在Having子句中优先将列名解析为聚集函数的别名，如果聚集函数的别名不存在，则尝试解析为表的列名。 ● 不设置该选项时，在Having子句中优先将列名解析为表的列名，如果表里没有该列名，则尝试解析为聚集函数的别名。 **注意** ● 新装集群和升级场景都不设置该参数。 ● 该选项控制Having子句中列名的解析顺序，设置后会引起包含Having子句的SQL语句结果集变化，请慎重考虑配置该选项。	ORA TD MySQL
ora_char_varchar_keep_space	控制ORA兼容下char与varchar类型转换时是否去除尾部空格。该选项仅9.1.1.300及以上集群版本支持。 ● 设置该选项时，ORA兼容下char类型转varchar类型时保留空格。 ● 不设置该选项时，ORA兼容下char类型转varchar类型时去除空格。	ORA
chr_check_illegal_char	控制chr函数是否对UTF-8字符合法性进行检查。该选项仅9.1.2及以上集群版本支持。 ● 设置该选项时，chr函数会对生成的字符进行UTF-8合法性检查。如果出现非法字符，系统会返回类似报错信息“requested character not valid for encoding”。 ● 不设置该选项时，不进行合法性检查。	ORA TD MySQL
check_distribute_key_encoding	控制是否对哈希分布列的编码合法性进行检查。该选项仅9.1.2及以上集群版本支持。 ● 设置该选项时，系统会检查哈希分布列的编码合法性。如果出现非法字符，系统返回类似报错信息“invalid byte sequence for encoding "UTF8": xxx”。 ● 不设置该选项时，不进行合法性检查。	ORA TD MySQL

enable_matview

参数说明：用于控制CREATE MATERIALIZED VIEW是否能创建物化视图。

参数类型：USERSET

取值范围：布尔型

- on表示CREATE MATERIALIZED VIEW可以创建物化视图。
- off表示CREATE MATERIALIZED VIEW不可以创建物化视图。

 版权所有 © 华为云计算技术有限公司

默认值：on

说明

该参数9.1.1.300及以上集群版本中默认值为on，9.1.1.300以下集群版本中默认值为off。

mv_max_cascade_depth

参数说明：设置嵌套物化视图最大深度。该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：整型， 1～10

默认值：3

matview_incr_refresh_retry_times

参数说明：用于控制物化视图增量刷新失败时的重试次数，所有重试均失败则回退为全量刷新。该参数仅9.1.1.300及以上集群版本支持。

参数类型：USERSET

取值范围：整型，1~100

默认值：1

hive_compat_options

参数说明：用于DWS的MySQL模式下，控制INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时是否将空串转为NULL。该参数仅9.1.0.210及以上集群版本支持。设置示例，参见**INSERT**章节。

参数类型：USERSET

取值范围：枚举型

- empty_str_to_null，表示INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时，会将空串转为NULL。
- 空，表示INSERT INTO VALUES方式插入数据到数值类型字段和INSERT INTO SELECT方式插入数据到字符类型字段时，对空串不做任何处理。

默认值：空

18.17 容错性

当数据库系统发生错误时，以下参数控制服务器处理错误的方式。

exit_on_error

参数说明：控制终止会话。

参数类型：SUSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示任何错误都会终止当前的会话。
- off表示只有FATAL级别的错误才会终止会话。

默认值：off

omit_encoding_error

参数说明：数据库进行字符编码转换，在出现字符编码错误时，若目标端字符集编码为UTF-8，可将有转换错误的被转换字符忽略，并以"?"代替。

参数类型：USERSET

取值范围：布尔型

- on表示有转换错误的字符将被忽略，以"?"代替，打印错误信息到日志中。
- off表示有转换错误的字符不能被转换，打印错误信息到终端。

默认值：off

max_query_retry_times

参数说明：指定SQL语句出错自动重试功能的最大重跑次数，目前支持重跑的错误类型为“Connection reset by peer”、“Lock wait timeout”和“Connection timed out”等，设定为0时关闭重跑功能。

参数类型：USERSET

取值范围：整型，0~20

默认值：6

retry_ecode_list

参数说明：指定SQL语句出错自动重试功能支持的错误类型列表。

参数类型：USERSET

取值范围：字符串

默认值：YY001 YY002 YY003 YY004 YY005 YY006 YY007 YY008 YY009 YY010 YY011 YY012 YY013 YY014 YY015 53200 08006 08000 57P01 XX003 XX009 YY016 CG003 CG004 F0011 F0012 45003 42P30 YY017 YY018 YY019 YY020 YY021 YY022 YY023 29P06 YY024

18.18 连接池参数

当使用连接池访问数据库时，在系统运行过程中，数据库连接是被当作对象存储在内存中的，当用户需要访问数据库时，并非建立一个新的连接，而是从连接池中取出一个已建立的空闲连接来使用。用户使用完毕后，数据库并非将连接关闭，而是将连接放回连接池中，以供下一个请求访问使用。

max_pool_size

参数说明：CN的连接池与其他某个CN/DN的最大连接数。

参数类型：POSTMASTER

 版权所有 © 华为云计算技术有限公司

取值范围：整型，1~65535

默认值：800

注意

- max_pool_size最大可设置为900。设置值若超过900，系统会自动设置为900，防止连接池缓存占用内存过高。
- 该参数在DN上设置不会生效。

persistent_datanode_connections

参数说明：会话是否会释放获得的连接。

参数类型：USERSET

取值范围：布尔型

- off表示会释放获得连接。
- on表示不会释放获得连接。

须知

打开此开关后，会存在会话持有连接但并未运行查询的情况，导致其他查询申请不到连接报错。出现此问题时，需约束会话数量小于等于max_active_statements。

默认值：off

cache_connection

参数说明：是否回收连接池的连接。

参数类型：USERSET

取值范围：布尔型

- on表示回收连接池的连接。
- off表示不回收连接池的连接。

默认值：on

enable_force_reuse_connections

参数说明：会话是否强制重用新的连接。

参数类型：USERSET

取值范围：布尔型

- on表示强制使用新连接。
- off表示使用现有连接。

 版权所有 © 华为云计算技术有限公司

默认值：off

syscache_clean_policy

参数说明：设置DN空闲连接内存及数量清理策略。该参数仅9.1.0.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：字符串

该参数策略由以下三个值组成：

1. 第一个值的取值为0~1，表示DN已使用内存占总可用内存比例。如果已使用内存占用达到该值时，清理1/4的stream线程，继续执行第二个取值的判断。
2. 第二个值的取值为0~1，表示DN上syscache内存占总可用内存比例。如果syscache内存占用达到该值时，继续执行第三个取值的判断。
3. 第三个值的取值为0~INT_MAX，单位为MB。表示空闲线程的syscache内存占用大小。如果空闲线程syscache内存占用达到该值时，则清理该线程占用的syscache。

默认值：0.80,0.30,64

注意

- 设置该参数前，需使用**PV_SESSION_MEMORY_DETAIL**视图及**PV_TOTAL_MEMORY_DETAIL**视图查看内存占用情况，进行评估。
- 设置该参数值时，需遵循设置格式，确保三个值之间以英文逗号分隔，且中间没有空格。
- 如果没有遵循规定的参数值设置格式导致设置失败，会在日志中产生WARNING类型日志，且使用SHOW命令查询该参数时，显示的为上一次设置成功的参数值；如果设置失败重启，该参数会按照默认值进行设置。
- CN上处于Readcommand阶段的线程，超时30秒之后，判断syscache大于256MB清理DN。有以下两个操作：
  1. 由辅助线程监控内存占用，如果整体内存占用到80%时，清理1/4的stream线程，并判断syscache占用是否超过总占用内存的30%，如果大于30%，则清理大于64MB的Readcommand阶段pg线程的syscache。
  2. stream线程在空闲大于30秒，且syscache占用大于64MB，则清理syscache。

18.19 集群事务

介绍集群事务隔离、事务只读、最大prepared事务数、集群维护模式目的参数设置及取值范围等内容。

transaction_isolation

参数说明：设置当前事务的隔离级别。

参数类型：USERSET

取值范围：

 版权所有 © 华为云计算技术有限公司

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
- read uncommitted：读未提交隔离级别，DWS不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其他并发事务提交的修改。
- serializable：事务可序列化，DWS不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

transaction_read_only

参数说明：设置当前事务是只读事务。

参数类型：USERSET

取值范围：布尔型

- on表示设置当前事务为只读事务。
- off表示该事务可以是非只读事务。

默认值：CN节点为off，DN节点为on。

xc_maintenance_mode

参数说明：设置系统进入维护模式。

参数类型：SUSET

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

默认值：off

须知

谨慎打开这个开关，避免引起集群数据不一致。

allow_concurrent_tuple_update

参数说明：设置是否允许并发更新。

参数类型：USERSET

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

默认值：on

 版权所有 © 华为云计算技术有限公司

gtm_backup_barrier

参数说明：指定是否为GTM启动点创建还原点。

参数类型：SUSET

取值范围：布尔型

- on表示创建还原点。
- off表示不创建还原点。

默认值：off

transaction_deferrable

参数说明：指定是否允许一个只读串行事务延迟执行，使其不会执行失败。该参数设置为on时，当一个只读事务发现读取的元组正在被其他事务修改，则延迟该只读事务直到其他事务修改完成。目前，DWS暂时未用到这个参数。与该参数类似的还有一个**default_transaction_deferrable**，设置它来指定一个事务是否允许延迟。

参数类型：USERSET

取值范围：布尔型

- on表示允许执行。
- off表示不允许执行。

默认值：off

enforce_two_phase_commit

参数说明：为了兼容历史版本功能保留该参数，当前版本设置无效。

enable_show_any_tuples

参数说明：该参数只有在只读事务中可用，用于分析。当这个参数被置为on/true时，表中元组的所有版本都会可见。

参数类型：USERSET

取值范围：布尔型

- on/true表示表中元组的所有版本都会可见。
- off/false表示表中元组的所有版本都不可见。

默认值：off

idle_in_transaction_timeout

参数说明：设置允许事务处于idle空闲状态的时间。当事务处于idle状态超过该时间后，会终止本事务。功能仅对直连CN的客户端连接生效，对直连DN或内部连接不生效。该参数仅8.2.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：0 ~ 86400，单位为秒（s）。

默认值：3600

 版权所有 © 华为云计算技术有限公司

18.20 备份容灾参数

cluster_id

参数说明： 在细粒度容灾的主集群上记录本集群的集群ID，其他场景下一般为空。该参数仅8.2.0及以上集群版本支持，属于容灾内部参数，不建议用户修改。

参数类型：SIGHUP

取值范围：字符串。默认情况下为空，如果设置，则为uuid格式。

默认值：空

fine_disaster_main_clusters

参数说明：在细粒度容灾的主集群上记录主集群cluster_id的list。该参数仅8.2.0及以上集群版本支持，属于容灾内部参数，不建议用户修改。

参数类型：SIGHUP

取值范围：字符串。默认情况下为空，如果设置，则list单元为uuid格式，如果有多个单元，使用逗号隔开。

默认值：空

enable_metadata_tracking

参数说明：在细粒度容灾的主集群上，用于追踪细粒度容灾相关对象的元数据（包括DDL、DCL）变更，用于向备集群同步变更的元数据。该参数仅8.2.1及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启元数据变化追踪功能，此后发生元数据变更，可以通过细粒度容灾同步到备集群。
- off表示不开启元数据变化追踪功能。

默认值：off

disaster_cluster_dn_nums

参数说明：用于设置细粒度容灾主备集群DN数，格式是“本地集群DN数，远端集群DB数”，主要用于小集群向大集群同步时的数据分桶。该参数仅8.2.1及以上集群版本支持。

参数类型：SIGHUP

取值范围：字符串。根据主备集群DN情况进行设置，比如DN数比例6:3的容灾，大小集群分别设置为“6,3”和“3,6”。

默认值：0,0

 版权所有 © 华为云计算技术有限公司

fine_dr_consistency_check_mode

参数说明：用于设置细粒度容灾校验主备集群数据一致性的模式，需要保持主备集群一致才能正常校验一致性。该参数仅8.2.1.251及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- none，表示不校验。
- warning，表示若校验主备数据不一致，打印warning日志。
- error，表示若校验主备集群数据不一致，返回报错信息。

默认值：none

18.21 开发人员选项

enable_light_colupdate

参数说明：控制是否使用列存轻量化UPDATE。

参数类型：USERSET

取值范围：布尔型

- on表示开启列存轻量化UPDATE。
- off表示关闭列存轻量化UPDATE。

默认值：off

说明

列存轻量化UPDATE与后台列存AUTOVACUUM并发会小概率报错，可以通过ALTER TABLE设置表级参数enable_column_autovacuum_garbage为off来避免。需要注意的是设置表级参数enable_column_autovacuum_garbage为off会关闭该表的后台列存AUTOVACUUM。

enable_fast_query_shipping

参数说明：控制查询优化器是否使用分布式框架。

参数类型：USERSET

取值范围：布尔型

- on表示执行计划在CN和DN上各自生成。
- off表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。

默认值：on

 版权所有 © 华为云计算技术有限公司

> **⚠ 注意**
>
> 当参数**enable_light_proxy**和enable_fast_query_shipping同时为on，**behavior_compat_options**不包含配置项enable_force_add_batch时，jdbc的addBatch入库采用多路并行执行策略，性能较好。如果enable_fast_query_shipping参数值被修改为off，则jdbc的addBatch入库退回为单条串行执行策略，性能会大幅下降。

enable_column_cluster

参数说明：控制V3 hstore_opt表PCK批量入库及异步排序写入CU时是否跨CU按列聚集，该参数仅9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示V3 hstore_opt表PCK批量入库及异步排序写入CU时，所有CU按照col_id聚集，即col_id相同的CU会顺序写入。
- off表示V3 hstore_opt表PCK批量入库及异步排序写入CU时，所有CU按照cu_id聚集，即cu_id相同的CU会顺序写入。

默认值：on

enable_trigger_shipping

参数说明：控制触发器场景是否允许将触发器下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许将触发器下推到DN执行。
- off表示不允许将触发器下推到DN执行，在CN执行。

默认值：on

enable_remotejoin

参数说明：设置是否允许连接操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许连接操作计划下推到DN执行。
- off表示不允许连接操作计划下推到DN执行。

默认值：on

enable_remotegroup

参数说明：设置是否允许group by与aggregates执行计划下推到DN执行。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示允许group by与aggregates执行计划下推到DN执行。
- off表示不允许group by与aggregates执行计划下推到DN执行。

默认值：on

enable_remotelimit

参数说明：设置是否允许LIMIT子句执行计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许LIMIT子句执行计划下推到DN执行。
- off表示不允许LIMIT子句执行计划下推到DN执行。

默认值：on

enable_limit_stop

参数说明：控制LIMIT语句是否启用early stop优化。对于LIMIT n语句，若使用early stop优化则CN收取到n条数据后会传递请求使DN提前结束执行，适用于复杂查询且带有LIMIT的场景。该参数仅8.1.3.320及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示允许LIMIT语句使用early stop优化。
- off表示不允许LIMIT语句使用early stop优化。

默认值：on

enable_remotesort

参数说明：设置是否允许ORDER BY子句操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许ORDER BY子句操作计划下推到DN执行。
- off表示不允许ORDER BY子句操作计划下推到DN执行。

默认值：on

enable_join_pseudoconst

参数说明：设置是否允许与伪常数进行join。伪常数是指join两侧的变量都等于同一个常量。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示允许与伪常数进行join。
- off表示不允许与伪常数进行join。

默认值：off

cost_model_version

参数说明：控制应用场景中估算时cost使用的模型。该参数的影响范围主要涵盖：表达式distinct估算、HashJoin代价模型、行数估算、重分布时分布键的选择及Aggregate的行数估算等。

参数类型：USERSET

取值范围：0、1、2、3、4

- 0，表示使用原始的cost估算模型。
- 1，表示在0的基础上，使用增强的表达式distinct估算、HashJoin代价模型、行数估算、重分布时分布键的选择及Aggregate的行数估算。
- 2，表示在1的基础上，使用随机性更优的analyze采样算法，以提高统计信息准确性。
- 3，表示在2的基础上，优化大集群场景下的broadcast代价估算，以便优化器选择更优计划。该选项仅8.3.0及以上集群版本支持。
- 4，表示在3的基础上，优化了hashjoin并行化代价、倾斜代价、列存索引有序性代价的代价估算以及coalesce表达式的行数估算，并支持子查询常量输出列进行join时的倾斜优化识别。

默认值：4

debug_assertions

参数说明：控制打开各种断言检查。能够协助调试，当遇到奇怪的问题或者崩溃，请把此参数打开，因为它能暴露编程的错误。要使用这个参数，必须在编译DWS的时候定义宏USE_ASSERT_CHECKING（通过configure选项 --enable-cassert完成）。

参数类型：USERSET

取值范围：布尔型

- on表示打开断言检查。
- off表示不打开断言检查。

说明

当启用断言选项编译DWS时，debug_assertions缺省值为on 。

默认值：off

ignore_checksum_failure

参数说明： 设置读取数据时是否忽略校验信息检查失败（但仍然会告警），继续执行。该参数仅在enable_crc_check为on时有效。继续执行可能导致崩溃，传播或隐藏损坏数据，无法从远程节点恢复数据及其他严重问题。不建议用户修改设置。

参数类型：SUSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示忽略数据校验错误。
- off表示数据校验错误正常报错。

默认值：off

column_default_format

参数说明：设置为不同选项会改变建表语句的结果。如果和用户指定的表级参数有冲突（例如，指定enable_hstore_opt=off），则该参数不生效。该参数9.1.0.200及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none或不设置，表示功能关闭。
- column_hstore_opt，设置该选项时如果指定了列存（包括显式指定orientation=column，或隐式指定）的建表，9.1.1.300及以上实时数仓集群版本下为普通模式hstore_opt表，标准数仓下为批量模式hstore_opt表。
- all_hstore_opt，设置该选项时除指定orientation=row外，默认创建为hstore opt表。该选项仅9.1.1.200及以上集群版本支持。

默认值：column_hstore_opt

说明

新安装的9.1.1.300及以上标准数仓集群版本中，该参数默认值为column_hstore_opt，表示默认创建的列存表都是批量模式hstore_opt表。升级场景下为保持前向兼容该参数的默认值维持原值。

default_table_behavior

参数说明：支持设置默认表的行为类型，该参数仅8.2.1及以上集群版本支持。

参数类型：USERSET

取值范围：column_btree_index、column_high_compress、column_middle_compress、column_low_compress

- column_btree_index，表示列存表默认创建索引为B-Tree。
- column_high_compress，表示列存表默认压缩级别为high。
- column_middle_compress，表示列存表默认压缩级别为middle。
- column_low_compress，表示列存表默认压缩级别为low。

默认值：空字符串

enable_colstore

参数说明：创建表时，当不指定存储方式时，默认创建为列存表。使用时，各节点值需要保持一致。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

默认值：off

 版权所有 © 华为云计算技术有限公司

enable_force_vector_engine

参数说明：对于支持向量化的执行器算子，如果其子节点是非向量化的算子，通过设置此参数为on，强制生成向量化的执行计划。当打开enable_force_vector_engine开关时，无论是行存表、列存表或者是行列混存，如果plantree中不包含不支持向量化的场景，则强制走向量化执行引擎。

参数类型：USERSET

取值范围：布尔型

默认值：off

enable_csqual_pushdown

参数说明：进行查询时，是否要将过滤条件下推，进行Rough Check。

参数类型：USERSET

取值范围：布尔型

- on表示进行查询时，要将过滤条件下推，进行Rough Check。
- off表示进行查询时，不要将过滤条件下推，进行Rough Check。

默认值：on

explain_dna_file

参数说明：指定**explain_perf_mode**为run，导出的csv信息的目标文件。该参数的取值必须为绝对路径加**.csv**格式的文件名。

参数类型：USERSET

取值范围：字符串

默认值：空字符串

explain_perf_mode

参数说明：此参数用来指定explain的显示格式。

参数类型：USERSET

取值范围：normal、pretty、summary、run

- normal：表示使用默认的打印格式。
- pretty：表示使用DWS改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。
- summary：表示在pretty的基础上增加了对打印信息的分析。
- run：表示在summary的基础上将统计的信息输出到csv格式的文件中，以便于进一步分析。

默认值：pretty

join_num_distinct

参数说明：控制应用场景中Join列或表达式的默认distinct值。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：双精度浮点型，大于或等于-100，客户端显示小数时可能会有截断。

- 大于0，表示使用该值作为默认distinct值。
- 大于等于-100且小于0，表示估算默认distinct时使用的百分比。
- 0，表示使用200作为默认distinct值。

默认值：-20

outer_join_max_rows_multipler

参数说明：控制应用场景中外连接估算行数的最大值。

参数类型：USERSET

取值范围：双精度浮点型，0或大于等于1，客户端显示小数时可能会有截断。

- 0，表示不限制outer join的估算行数。
- 大于等于1，表示限制估算行数不超过外连接中外表行数的倍数。

默认值：1.1

qual_num_distinct

参数说明：控制应用场景中过滤列或表达式的默认distinct值。

参数类型：USERSET

取值范围：双精度浮点型，大于等于-100。客户端显示小数时可能会有截断。

- 大于0，表示使用该值作为默认distinct值。
- 大于等于-100且小于0，表示估算默认distinct时使用的百分比。
- 0，表示使用200作为默认distinct值。

默认值：200

trace_notify

参数说明：为LISTEN和NOTIFY命令生成大量调试输出。**client_min_messages**或**log_min_messages**级别必须是DEBUG1或者更低时，才能把这些输出分别发送到客户端或者服务器日志。

参数类型：USERSET

取值范围：布尔型

- on表示打开输出功能。
- off表示关闭输出功能。

默认值：off

trace_sort

参数说明：控制是否在日志中打印排序操作中的资源使用相关信息。这个选项只有在编译DWS的时候定义了TRACE_SORT宏的时候才可用，不过目前TRACE_SORT是由缺省定义的。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

默认值：off

zero_damaged_pages

参数说明：控制检测导致DWS报告错误的损坏的页头，中止当前事务。

参数类型：SUSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

说明

- 设置为on时，系统报告一个警告，把损坏的页面填充为零然后继续处理。该行为会破坏数据，即被损坏页面上的所有行。但是它允许绕开损坏页面然后从表中存在的未损坏页面上继续检索数据行。因此该参数在硬件或者软件错误导致的数据损坏中进行恢复是有作用的。通常不建议该参数设置为on，除非不需要从损坏的页面中恢复数据。
- 对于列存表，会将整个CU跳过然后继续处理。支持的场景包括crc校验失败、magic校验失败以及读取的CU长度错误。

默认值：off

replication_test

参数说明：此参数用于数据复制的内部功能测试。

参数类型：USERSET

取值范围：布尔型

- on表示启用功能测试。
- off表示关闭功能测试。

默认值：off

cost_param

参数说明：该参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

当cost_param & 1 不为0，表示对于求不等值连接选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确，V300R002C00版本开始，已弃用cost_param & 1 不为0时的路径，默认选择更优的估算公式；

当cost_param & 2 不为0，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确；

 版权所有 © 华为云计算技术有限公司

当cost_param & 4 不为0，表示在进行stream节点估算时，选用调试模型，该模型不推荐用户使用。

当cost_param & 16不为0，表示在计算两个及以上过滤条件或Join条件的综合选择率时，采用介于完全相关和完全不相关之间的一种模型，过滤条件较多时倾向于相关性较强的模型。

参数类型：USERSET

取值范围：整型，1～INT_MAX

默认值：16

convert_string_to_digit

参数说明：设置隐式转换优先级，是否优先将字符串转为数字。

参数类型：USERSET

取值范围：布尔型

- on表示优先将字符串转为数字。
- off表示不优先将字符串转为数字。

默认值：on

须知

请谨慎调整该参数，调整该参数会修改内部数据类型转换规则并可能导致不可预期的行为。

nls_timestamp_format

参数说明：设置时间戳默认格式。

参数类型：USERSET

取值范围：字符串

默认值：DD-Mon-YYYY HH:MI:SS.FF AM

enable_partitionwise

参数说明：分区表连接操作是否选择智能算法。

参数类型：USERSET

取值范围：布尔型

- on表示选择智能算法。
- off表示不选择智能算法。

默认值：off

 版权所有 © 华为云计算技术有限公司

enable_partition_dynamic_pruning

参数说明：分区表扫描是否支持动态剪枝。

参数类型：USERSET

取值范围：布尔型

- on表示分区表扫描开启动态剪枝。
- off表示分区表扫描关闭动态剪枝。

默认值：on

max_user_defined_exception

参数说明：异常最大个数，默认值不可更改。

参数类型：USERSET

取值范围：整型

默认值：1000

datanode_strong_sync

参数说明：目前该参数已废弃，无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示启用stream节点间强同步。
- off表示不启用stream节点间强同步。

默认值：off

enable_global_stats

参数说明：标识当前统计信息模式，为便于进行全局统计信息生成计划和单DN统计信息计划对比使用，默认创建为全局统计信息模式。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

- on/true表示全局统计信息。
- off/false表示单DN统计信息。

默认值：on

enable_fast_numeric

参数说明：标识是否开启Numeric类型数据运算优化。Numeric数据运算是较为耗时的操作之一，通过将Numeric转化为int64/int128类型，提高Numeric运算的性能。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on/true表示开启Numeric优化。
- off/false表示关闭Numeric优化。

默认值：on

enable_row_fast_numeric

参数说明：标识行存表numeric数据落盘的格式。

参数类型：USERSET

取值范围：布尔型

- on/true表示行存表numeric落盘格式为bigint格式。
- off/false表示行存表numeric落盘格式为原始格式。

须知

参数值设置为on时，建议同步打开enable_force_vector_engine, 可提升大数据集query的查询性能。但相比于原始格式，大概率会占用更多磁盘空间。以TPC-H测试集为例，大约多占7%空间（不同环境参考值可能有差异）。

默认值：off

rewrite_rule

参数说明：标识开启的可选查询重写规则。有部分查询重写规则是可选的，开启它们并不能总是对查询效率有提升效果。在特定的客户场景中，通过此GUC参数对查询重写规则进行设置，使得查询效率最优。

此参数可以控制查询重写规则的组合，比如有多个重写规则：rule1、rule2、rule3、rule4。可以设置：

```
set rewrite_rule=rule1;          --启用查询重写规则rule1
set rewrite_rule=rule2,rule3;    --启用查询重写规则rule2和rule3
set rewrite_rule=none;           --关闭所有可选查询重写规则
```

参数类型：USERSET

取值范围：字符串

- none：不使用任何可选查询重写规则。
- lazyagg：使用Lazy Agg查询重写规则（消除子查询中的聚集运算）。
- magicset：使用Magic Set查询重写规则（从主查询中下推条件到提升的子链接）。
- uniquecheck：使用Unique Check重写规则（允许目标列不含聚集函数的表达式子链接场景提升，需在子链接按关联列聚集后目标列值唯一才能开启，建议专业调优人员使用）。
- disablerep：使用禁止复制表的子链接提升规则（针对复制表禁止子链接提升）。
- projection_pushdown：使用Projection Pushdown重写规则（子查询中消除父查询不使用的列）。
- or_conversion：使用OR转换重写规则（消除执行效率低下的关联OR条件）。

版权所有 © 华为云计算技术有限公司

- plain_lazyagg: 使用Plain Lazy Agg查询重写规则（消除单子查询中的聚集操作）。该选项仅8.1.3.100及以上集群版本支持。
- eager_magicset: 使用eager_magicset查询重写规则（从主查询中下推条件到子查询）。该选项仅8.2.0及以上集群版本支持。
- casewhen_simplification：使用casewhen语句改写的重写规则，开启后对(case when xxx then const1 else const2)=const1场景进行改写，用于简化CASE WHEN语句，减少冗余计算从而提升查询性能。该选项仅8.3.0及以上集群版本支持。
- outer_join_quality_imply：left outer join和right outer join存在等值关联条件时，外表关联列上的表达式条件下推到内表的关联列上。该选项仅8.3.0及以上集群版本支持。
- inlist_merge：使用inlist_or_inlist查询重写规则，支持基表相同列的or语句合并，开启后对(where a in (list1) or a in (list2))进行合并改写，合并后可支持inlist2join。该选项仅8.3.0及以上集群版本支持。
- subquery_qual_pull_up：对于无法提升的子查询，若子查询与其他表进行关联的关联列在子查询内部有过滤条件，支持从子查询中提取过滤条件并传递到关联条件另一侧。目前只支持不存在类型转换的var op const形式，例如a > 2。打开该开关时，则认为outer_join_quality_imply也是打开的。该选项仅9.1.0及以上集群版本支持。
- join_elimination：开启后，如果查询的表之间额外定义了主键-外键关系，优化器可基于主外键关系，消除查询中不必要的内连接，提升SQL执行性能。该选项仅9.1.0.200及以上集群版本支持。
- count_distinct_rewrite_opt：使用uniq语句改写count (distinct xxx)语句。开启后对distinct列为int类型或查询语句中count distinct语句数量大于等于3的场景进行uniq自动重写，提升SQL性能。该选项仅9.1.1.100及以上集群版本支持。
- selfjoin_elimination：开启后，当SQL包含对结果集重复不敏感的子查询（如exists子查询、in子查询等）时，如果子查询中存在相同表的自连接，且至少包含两个join条件：一个是同一列a的等值比较，另一个是同一列b的不等值比较，且没有其他筛选条件（如where条件或having条件），优化器可消除其中一个表，将自连接转换为count(distinct)聚集函数运算，从而减少join操作进一步提升SQL执行性能。该选项仅9.1.1.100及以上集群版本支持。
- use_ctelist_inline：开启后，优化器将CTE进行inline时，会一次性将所有引用全部进行inline，而不是根据优化器处理的query block层级，逐层inline，这种方式可提升CTE的处理效率。该选项仅9.1.1.100及以上集群版本支持。
- not_distinct_from_opt：开启后，将is not distinct from表达式通过临时改写为等号表达式的方式，使包含该语法的语句能够生成hashjoin或mergejoin计划，从而提升执行性能。该选项仅9.1.0及以上集群版本支持。
- joinquals_ref_subplans：开启后，join的表达式关联条件若出现在下层算子中，能够复用其结果提升语句执行性能。该选项仅9.1.1.100及以上集群版本支持。

默认值：magicset, or_conversion, projection_pushdown, plain_lazyagg, subquery_qual_pull_up, join_elimination, selfjoin_elimination, count_distinct_rewrite_opt, use_ctelist_inline

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- 新安装的9.1.1.100及以上集群版本中该参数的默认值包含重写规则count_distinct_rewrite_opt，升级场景下为保持前向兼容该参数的默认值维持原值。
- 若设置了join_elimination选项，DWS不会对主键-外键关系做主动数据校验，如果用户定义了两表之间的主外键关系，需要自行确保主外键关系的合法性，否则开启此优化可能造成返回的**结果集错误**。具体使用方式参见“**通过主外键约束消除多余的JOIN**”章节。

mv_rewrite_rule

参数说明：标识开启的可选物化视图查询重写规则。

参数类型：USERSET

取值范围：字符串

- none，不使用任何物化视图重写规则。该取值仅8.2.1.100及以上集群版本支持。
- text，使用文本匹配的物化视图重写规则。该取值仅8.2.1.100及以上集群版本支持。
- general，使用结构匹配。该取值仅9.1.0.200及以上集群版本支持。
- predicate，结构匹配重写使用表达式框架，用于过滤条件和关联条件的预处理和正则化。仅在general开启时生效。该取值仅9.1.0.210及以上集群版本支持。
- view_delta，通过主外键关系消除物化视图中的冗余join后与查询进行匹配。该取值仅9.1.0.210及以上集群版本支持。
- single，单表物化视图使用物化视图重写规则。该取值仅9.1.0.222及以上集群版本支持。
- partition，控制分区物化视图重写时是否使用物化视图和基表UNION ALL的方式。该取值仅9.1.2及以上集群版本支持。

默认值：text, general, predicate, view_delta

⚠ 注意

9.1.1.300以下集群版本中general、predicate和view_delta功能受限商用，如需使用请联系技术支持。

mv_computing_resource

参数说明：指定物化视图刷新需要使用的弹性资源池名称。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

默认值：空

 版权所有 © 华为云计算技术有限公司

mv_refresh_parts_per_trans

参数说明：指定分区物化视图每次分批提交的物化视图分区个数，以实现由近及远，最新分区先被刷新提交的目的。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1~INT_MAX。-1表示不进行分批提交。

默认值：-1

mv_direct_query_policy

参数说明：分区物化视图查询时支持使用物化视图和基表UNION ALL的方式，该参数为控制直查策略参数，仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：字符串。

- force_mv，表示直接查询物化视图，无论视图是否失效。
- prefer_mv，表示少量分区失效时，使用可用的分区物化视图与基表进行UNION ALL查询；大量分区失效时，仍使用物化视图进行查询，性能优先。
- prefer_base，表示少量分区失效时，使用可用的分区物化视图与基表进行UNION ALL查询；大量分区失效时，使用基表进行查询。
- force_base，表示强制使用基表进行查询，完全不使用物化视图。

默认值：force_mv

mv_partition_rewrite_threshold

参数说明：控制分区物化视图重写时，使用物化视图与基表进行UNION ALL方式的重写阈值。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1~10000

- 正值，允许的最大失效分区数。
- 负值，允许的最大失效分区比例（例如，取值为-0.3，表示允许最多30%的分区失效）。

默认值：-0.3

mv_refresh_min_interval

参数说明：控制物化视图两次刷新之间的最小时间间隔。距上次刷新完成时间未达到该间隔值，且失效分区数小于**mv_partition_rewrite_threshold**配置的阈值时，跳过本次刷新，以防止频繁刷新。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX，单位为秒（s）。0表示不限制刷新间隔时间。

默认值：0

 版权所有 © 华为云计算技术有限公司

enable_compress_spill

参数说明：标识是否开启下盘压缩功能。

参数类型：USERSET

取值范围：布尔型

- on/true表示开启下盘优化。
- off/false表示关闭下盘优化。

默认值：on

analysis_options

参数说明：设置是否开启对应的功能选项使用其定位功能，包括数据校验，性能统计等。

注意

此参数主要用于日志定位，非必要不建议用户配置，否则会导致集群不可预知异常，需要配置此参数，请联系技术人员确认。

参数类型：USERSET

取值范围：字符串

- LLVM_COMPILE，表示在explain performance显示界面中显示每个线程的codegen编译时间。
- HASH_CONFLICT，表示在DN进程的pg_log目录中的log日志中显示hash表的统计信息，包括hash表大小、hash链长及hash冲突情况。
- STREAM_DATA_CHECK，表示对网络传输前后的数据进行CRC校验。
- TURBO_DATA_CHECK，表示对turbo的ScalarVector和VectorBatch的算子的数据上下文进行校验。该选项仅8.3.0.100及以上集群支持。
- VECTOR_DATA_CHECK，表示对vector的ScalarVector和VectorBatch的算子的数据上下文进行校验。该选项仅9.1.1及以上集群支持。
- KEEP_SAMPLE_DATA，表示以临时表的形式保留每次analyze操作中使用的采样数据。该选项仅9.1.0及以上集群版本支持。
- BLOCK_RULE，表示在explain performance显示界面中显示查询过滤器检查耗时。该选项仅9.1.0.100及以上集群版本支持。
- PARTITION_INSERT_CHECK，表示范围分区表中插入数据时，如果未匹配到任何分区，设置该参数时会在日志中输出分区列的值和分区表当前的分区边界值。该选项9.1.1集群版本中仅9.1.1.300及以上版本支持。

默认值：off(ALL)，不开启任何定位功能。

resource_track_log

参数说明：控制自诊断的日志级别。目前仅对多列统计信息进行控制。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：字符串

- summary：显示简略的诊断信息。
- detail：显示详细的诊断信息。

目前这两个参数值只在显示多列统计信息未收集的告警的情况下有差别，summary不显示未收集多列统计信息的告警，detail会显示这类告警。

默认值：summary

hll_default_log2m

参数说明：该参数可以指定hll数据结构桶的个数。桶的个数会影响hll计算distinct值的精度，桶的个数越多，误差越小。误差范围为：$[-1.04/2^{log2m*1/2},+1.04/2^{log2m*1/2}]$。

参数类型：USERSET

取值范围：整型，10~16。

默认值：11

hll_default_regwidth

参数说明：该参数可以指定hll数据结构每个桶的位数，该值越大，hll所占内存越高。hll_default_regwidth和hll_default_log2m可以决定当前hll能够计算的最大distinct value。具体对应关系可以参见**表1 hll_default_log2m和hll_default_regwidth与当前能计算的最大distinct value值的关系**。

参数类型：USERSET

取值范围：整型，1~5。

默认值：5

表 18-3 hll_default_log2m 和 hll_default_regwidth 与当前能计算的最大 distinct value 值的关系

log2m	regwidth = 1	regwidth = 2	regwidth = 3	regwidth = 4	regwidth = 5
10	7.4e+02	3.0e+03	4.7e+04	1.2e+07	7.9e+11
11	1.5e+03	5.9e+03	9.5e+04	2.4e+07	1.6e+12
12	3.0e+03	1.2e+04	1.9e+05	4.8e+07	3.2e+12
13	5.9e+03	2.4e+04	3.8e+05	9.7e+07	6.3e+12
14	1.2e+04	4.7e+04	7.6e+05	1.9e+08	1.3e+13
15	2.4e+04	9.5e+04	1.5e+06	3.9e+08	2.5e+13

hll_default_expthresh

参数说明：该参数可以用来设置从Explicit模式到Sparse模式的默认阈值大小。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：整型，-1~7。-1表示自动模式，0表示跳过Explicit模式，取1-7表示在基数到达$2^{hll_default_expthresh}$时切换模式。

默认值：-1

hll_default_sparseon

参数说明：该参数可用来指定是否默认开启Sparse模式。

参数类型：USERSET

取值范围：0，1。0表示默认关闭，1表示默认开启。

默认值：1

hll_max_sparse

参数说明：该参数可以用来指定max_sparse的大小。

参数类型：USERSET

取值范围：整型，-1~INT_MAX

默认值：-1

enable_compress_hll

参数说明：该参数可以用来指定是否对hll开启内存优化模式。

参数类型：USERSET

取值范围：布尔型

- on/true表示对hll开启内存优化模式。
- off/false表示不开启内存优化模式。

默认值：off

approx_count_distinct_precision

参数说明：该参数表示HyperLogLog++ (HLL++)算法中分桶个数，可以用来调整approx_count_distinct聚集函数的误差率。桶的个数会影响distinct值估算的精度，桶的个数越多，误差越小。误差范围为：$[-1.04/2^{log2m*1/2},+1.04/2^{log2m*1/2}]$。

参数类型：USERSET

取值范围：整型，10~20。

默认值：17

udf_memory_limit

参数说明：控制每个CN、DN执行UDF时可用的最大物理内存量。

参数类型：POSTMASTER

取值范围：整型，200*1024～**max_process_memory**，单位为KB。

默认值：参数默认值与max_process_memory参数大小相关。具体取值规则如下：

 版权所有 © 华为云计算技术有限公司

- max_process_memory<4GB时，默认值固定为200MB。
- 4GB≤max_process_memory≤2047GB时，默认值为0.05 * max_process_memory。

  若计算结果<1GB， 则以MB为单位，否则以GB为单位。（例如max_process_memory为100GB时，该参数默认值为5GB；max_process_memory为8GB时，该参数默认值为410MB）
- max_process_memory>2047GB时，默认值固定为102GB。

FencedUDFMemoryLimit

参数说明：控制每个fenced udf worker进程使用的虚拟内存。

参数类型：USERSET

设置建议：不建议设置此参数，可用**udf_memory_limit**代替。

取值范围：整数，可带单位（KB，MB，GB）。其中0表示不做内存控制。

默认值：0

enable_pbe_optimization

参数说明：设置优化器是否对以PBE（Parse Bind Execute）形式执行的语句进行查询计划的优化。

参数类型：USERSET

取值范围：布尔型。

- on表示优化器将优化PBE语句的查询计划。
- off表示不使用优化。

默认值：on

enable_light_proxy

参数说明：设置优化器是否对简单查询在CN上优化执行。

参数类型：USERSET

取值范围：布尔型。

- on表示优化器将优化CN上简单查询的执行。
- off表示不使用优化。

默认值：on

⚠ 注意

当参数enable_light_proxy和**enable_fast_query_shipping**同时为on，**behavior_compat_options**不包含配置项enable_force_add_batch时，jdbc的addBatch入库采用多路并行执行策略，性能较好。如果enable_light_proxy参数值被修改为off，则jdbc的addBatch入库退回为单条串行执行策略，性能会大幅下降。

 版权所有 © 华为云计算技术有限公司

enable_parallel_ddl

参数说明：控制多CN对同一数据库对象是否能安全地并发执行DDL操作。

参数类型：USERSET

取值范围：布尔型

- on表示可以安全地并发执行DDL操作，不会出现分布式死锁。
- off表示不能安全地并发执行DDL操作，可能会出现分布式死锁。

默认值：on

gc_fdw_verify_option

参数说明：在协同分析特性中，控制是否启用结果集行数校验逻辑。该参数仅8.1.3.310及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启结果集行数校验逻辑，额外发送select count语句获取预期行数，与实际获取结果集进行比对。
- off表示关闭结果集行数校验逻辑，仅获取所需结果集。

默认值：on

说明

- 该参数开启时性能会有轻微劣化，性能敏感场景可通过关闭该参数提升性能。
- 结果集行数校验失败时会上报异常，可通过设置参数log_min_messages=debug1和logging_module='on(COOP_ANALYZE)'来打开协同分析日志。

show_acce_estimate_detail

参数说明：在DWS集群使用加速集群场景下（即**acceleration_with_compute_pool**设置为on），控制explain命令是否显示用于评估执行计划下推到加速集群的评估信息。评估信息一般用于运维人员在维护工作中使用，因此该参数默认关闭，此外为了避免这些信息干扰正常的explain信息显示，只有在explain命令的verbose选项打开的情况下才显示评估信息。

参数类型：USERSET

取值范围：布尔型

- on表示可以在explain命令的输出中显示评估信息。
- off表示不在explain命令的输出中显示评估信息。

默认值：off

full_group_by_mode

参数说明：结合**behavior_compat_options**中disable_full_group_by_mysql使用，用于控制disable_full_group_by_mysql语法开关打开后的两种不同行为。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：字符串

- nullpadding表示对于非聚集列而言，对该列NULL值进行填充，取该列非NULL值，结果集可能为不同行。
- notpadding表示对于非聚集列而言，不处理NULL值，取该行整行数据，非聚集列结果集为随机的一行。

默认值：notpadding

注意

该参数生效前提为MySQL兼容库下打开disable_full_group_by_mysql，且查询中出现非聚集列场景生效。该参数的两种行为也仅针对查询中的非聚集列生效。

enable_pg_class_visibility_debug

参数说明：控制是否允许输出与pg_class系统表可见性相关的日志，便于定位相关问题。该参数为开发人员专用，不建议普通用户使用。该参数9.1.1.100及以上的9.1.1.xxx集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on/true表示开启此日志开关。
- off/false表示关闭此日志开关。

默认值：off

is_db_oid_consistent

参数说明：用于标识数据库OID是否在所有节点上一致。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- true表示数据库OID在所有节点上一致。
- false表示数据库OID在不同节点中不一致。

默认值：false

注意

DWS标准数仓类型转为存算分离集群时会自动设置此参数，用于判断数据库是否支持创建V3表。该参数仅用于查看，不建议用户设置。

 版权所有 © 华为云计算技术有限公司

enable_cudesc_streaming

参数说明：在存算分离架构下，控制跨逻辑集群访问是否选择cudesc streaming路径（从表所在逻辑集群获取cudesc、delta表数据等信息）。该参数仅9.1.0及以上集群版本支持。

参数类型：SUSET

取值范围：枚举型

- off表示关闭cudesc streaming。
- on表示开启cudesc streaming。
- only_read_on表示只在数据读取时支持cudesc streaming。

默认值： on

force_read_from_rw

参数说明：在存算分离架构下，强制从其他逻辑集群上读取数据（从表所在逻辑集群上读取数据）。该参数仅9.0.0及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

默认值：off

kv_sync_up_timeout

参数说明：在存算分离架构下，设置KV同步等待超时时间。该参数仅9.0.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~2147483647

默认值： 10min

enable_insert_foreign_table_dop

参数说明：控制OBS外表写出时是否启用dop加速。

在外表insert场景，利用SMP线程同时在OBS进行文件写入，提高外表写入性能，充分利用OBS带宽，防止导入大表时超时失败。

每个DN的dop线程数量由query_dop决定，可以设置query_dop控制并行度。该参数仅9.1.0.200及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用外表dop加速。
- off表示禁用外表dop加速。

默认值：off

 版权所有 © 华为云计算技术有限公司

enable_insert_foreign_table_dop_opt

参数说明：控制OBS外表insert dop启用后，是否启用分区重分布优化。导出分区数较多时（导出分区数大于10倍分区数），建议开启，可以减少单分区内的小文件同时提高导出性能。该参数仅9.1.0.200及以上版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示启用分区外表insert dop重分布优化。
- off表示禁用分区外表insert dop重分布优化。

默认值：off

enable_recyclebin

参数说明：控制回收站是否实时开启。该参数仅9.1.0.200以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on表示打开回收站。
- off表示关闭回收站。

默认值： off

说明

- **该功能9.1.2以下集群版本中为Beta特性，如需使用请联系技术支持；9.1.2及以上集群版本中已正式商用。**
- 9.1.1.200及以上集群版本中，该参数的默认值为off。9.1.1.200以下集群版本中该参数的默认值为on。
- 回收站功能仅支持V3列存表（COLVERSION=3.0），且只能是单表语句。不支持行存表、临时表和LSM表。
- 集群升级、扩缩容及重分布期间不支持回收站功能。
- 该参数开启之后：
  - v3表执行DROP TABLE会将表的元数据定义的schema变更为pg_recyclebin，同时相关数据信息会被记录到**PG_FILE_RECYCLEBIN**系统表中。
  - v3表执行TRUNCATE TABLE、ALTER TABLE TRUNCATE PARTITION时会将被TRUNCATE的对象信息保存到**PG_FILE_RECYCLEBIN**中；回收站后台线程会遍历pg_file_recyclebin系统表，对于超过回收站保留时长的对象进行清理，清理后的对象将不能进行闪回。

recyclebin_retention_time

参数说明：设置回收站对象保留的时间，超过该参数设置时间的回收站对象将被自动清理。该参数仅9.1.0.200以上集群版本支持。

参数类型：SUSET

取值范围：整型，1~2147483，单位为秒（s）。

默认值：604800s

 版权所有 © 华为云计算技术有限公司

workload_vw_strategy

参数说明：设置CN的作业负载路由方式。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none，表示按照用户和逻辑集群的绑定关系路由。
- all，表示使用roundrobin方式，将作业路由到绑定的逻辑集群+专用+公用弹性逻辑集群。
- share，表示使用roundrobin方式，将作业路由到所有的公用弹性逻辑集群。
- dedicated，表示使用roundrobin方式，将作业路由到所有的专用弹性逻辑集群。
- elastic，表示使用roundrobin方式，将作业路由到所有的专用+公用的弹性逻辑集群。

默认值：none

enable_hstore_binlog_table

参数说明：用于控制是否可以创建binlog表。

参数类型：SIGHUP

取值范围：布尔型

- on表示可以创建binlog表。
- off表示不可以创建binlog表。

默认值：on

enable_hstore_lightupdate_table

参数说明：用于控制是否可以创建轻量化更新的Hstore_opt表，开启后，在对Hstore_opt表的列进行update更新入库时，不再对非更新列进行反查操作，以此减少入库反查带来的性能和资源开销，进一步提高Hstore_opt表的入库性能。该参数仅9.1.1.100及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔型

- on表示可以创建轻量化更新表。
- off表示不可以创建轻量化更新表。

默认值：on

enable_generate_binlog

参数说明：用于控制当前会话的binlog表上的DML操作是否产生binlog。该参数仅9.1.0.200以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示产生binlog。
- off表示不产生binlog。

默认值：on

binlog_consume_timeout

参数说明：用于控制在线扩缩容binlog表或者vacuum full binlog表时，循环判断binlog记录是否都被消费的超时时间。该参数仅8.3.0.100及以上版本支持。单位秒。

参数类型：SIGHUP

取值范围：整型，0~86400

默认值： 3600

enable_simd_opt

参数说明：用于控制执行引擎是否采用SIMD（单指令多数据流，Single Instruction Multiple Data）对相关计算过程进行性能加速。该参数仅支持加速AVX-512指令集的x86架构的CPU环境中的turbo表，其余场景开启参数无加速效果。仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启执行引擎的SIMD计算加速。
- off表示关闭执行引擎的SIMD计算加速。

默认值：on

说明

新安装的9.1.1.100及以上集群版本中该参数的默认值为on。从低版本升级到9.1.1.100及以上集群版本，默认值为off。

row_group_rows_threshold

参数说明：用于控制行列混存表row模式/mix模式行存的压缩单元行数，即xxx行压缩为一个行存压缩单元row group。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0-10000

- 0表示由DWS自动选择最优值，压缩单元行数为200~2000。
- 大于0时由用户指定压缩单位，压缩单元行数为200~10000，若小于200仍按照200处理。

默认值：0

enable_hstore_nopck_compaction

参数说明：控制未指定PCK的hstore表是否支持小CU合并。该参数仅9.1.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示开启所有hstore表的小CU合并。
- off表示只有指定了PCK列的hstore表支持小CU合并。

默认值：on

data_distribution_check_ratio

参数说明：控制对哈希分布表进行INSERT和MERGE INTO操作时插入数据的哈希分布检查比例（例如，默认值1表示对插入的数据进行1%的哈希分布抽样校验）。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0.0~100。其中0表示不检查。

默认值：1

说明

- 新安装集群场景，该参数默认值为1，表示数据插入哈希分布表时会进行分布正确性的校验。
- 涉及9.1.1.200以下版本的升级场景，该参数默认值为0，表示不进行检查。

enable_shipping_arrayexpr

参数说明：控制是否开启Array相关的表达式计算下推。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启Array表达式下推。
- off表示关闭Array表达式下推。

默认值：off

enable_vectorization_array

参数说明：控制Array相关的表达式计算是否支持向量化引擎。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启Array表达式向量化。
- off表示关闭Array表达式向量化。

默认值：on

 版权所有 © 华为云计算技术有限公司

说明

新安装的9.1.0及以上集群版本，默认值为on；低版本集群升级至9.1.2及以上版本后，默认值为off。

enable_update_optimization

参数说明：控制是否优化以全部主键为更新条件的DELETE/UPDATE语句，仅适用hstore_opt表。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示开启优化。
- off表示关闭优化。

默认值：on

with_recursive_wait_timeout

参数说明：用于设置with recursive算子执行等待超时退出的时间。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX，单位为秒（s）。其中，0表示不设置超时时间。

默认值： 0

dynamic_change_ereport_elevel

参数说明：用于实时调整错误级别或动态打印backtrace。该参数仅9.1.2及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

默认值：default,default,default,0

说明

该参数仅用于开发或维护人员定位问题使用，不建议用户设置。

18.22 审计

18.22.1 审计开关

audit_enabled

参数说明：控制审计进程的开启和关闭。审计进程开启后，将从管道读取后台进程写入的审计信息，并写入审计文件。

参数类型：SIGHUP

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示启动审计功能。
- off表示关闭审计功能。

默认值：on

说明

新安装9.1.1.100及以上集群版本中该参数的CN默认值为on，DN默认值为off。升级场景下为保持前向兼容该参数的默认值维持原值。

audit_space_limit

参数说明：审计文件占用的磁盘空间总量。

参数类型：SIGHUP

取值范围：整型，1024~1073741824，单位为KB。

默认值：1GB

audit_object_name_format

参数说明：控制审计日志object_name字段所显示的对象名的格式。

参数类型：USERSET

取值范围：枚举类型

- single，表示object_name字段显示单个对象名，且为目标对象的名称。
- all，表示object_name字段显示多个对象名称。

默认值：single

说明

当默认值设置为all时，显示多个对象名的场景有：SELECT、DELETE、UPDATE、INSERT、MERGE、CREATE TABLE AS、CREATE VIEW AS、DROP USER ... CASCADE、DROP OWNED BY ... CASCADE、DROP SCHEMA ... CASCADE、DROP TABLE ... CASCADE、DROP FOREIGN TABLE ... CASCADE、DROP VIEW ... CASCADE。

audit_object_details

参数说明：控制审计日志中是否记录object_details字段，该字段为审计语句中的表名、列名以及列使用的类型。该参数仅8.2.1.100及以上集群版本支持。

参数类型：SUSET

取值范围：布尔类型

- on表示审计记录object_details字段。
- off表示审计不记录object_details字段。

默认值：off

 版权所有 © 华为云计算技术有限公司

说明

- 当设置为on时，会对语句中的表名、列名以及列使用的类型进行审计，可能会对性能存在一定的影响，请谨慎开启。
- 当设置为on时，object_details字段记录的场景有：SELECT、DELETE、UPDATE、INSERT、MERGE、CREATE TABLE AS SELECT，GRANT、DECLARE CURSOR，其中对于执行失败的GRANT语句不予记录。

18.22.2 操作审计

security_enable_options

参数说明：该参数决定是否允许安全模式下使用grant_to_public、grant_with_grant_option和foreign_table_options三种功能，可根据实际需求进行配置。该参数仅8.2.0及以上集群版本支持。

参数类型：SIGHUP

取值范围：字符串

- grant_to_public，表示允许安全模式下使用grant to public功能。
- grant_with_grant_option，表示允许安全模式下使用with grant option功能。
- foreign_table_options，表示允许安全模式下使用外表操作功能，不需要显式赋予用户useft权限。

默认值：空

说明

- 新安装集群场景下，该参数默认值为空，表示安全模式下不允许使用grant_to_public、grant_with_grant_option和foreign_table_options中任何一种功能。
- 升级场景下，该参数的默认值保持前向兼容，若用户升级前原版本中GUC参数enable_grant_public和enable_grant_option默认为ON，那么升级后security_enable_options参数的默认值为“grant_to_public, grant_with_grant_option”。

18.23 事务监控

通过设置事务超时预警，可以监控自动回滚的事务并定位其中的语句问题，也可以监控执行时间过长的语句。

transaction_sync_naptime

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_naptime时会主动触发gs_clean进行清理，缩短不一致时的阻塞时长。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒（s）。

默认值：5s

 版权所有 © 华为云计算技术有限公司

说明

若该值设为0，则不会在阻塞达到时长时主动调用gs_clean进行清理，而是靠gs_clean_timeout间隔来调用gs_clean，默认是5分钟。

transaction_sync_timeout

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_timeout时会报错，回滚事务，避免由于sync lock等其他情况长时间进程停止响应造成对系统的阻塞。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒（s）。

默认值：10min

说明

- 若该值设为0，则不会在阻塞超时时报错，回滚事务。
- 该值必须大于gs_clean_timeout，避免DN上由于还未被gs_clean清理的残留事务阻塞超时引起的不必要的事务回滚。

18.24 plan management 管理

planmgmt_options

参数说明：计划管理配置项，该参数的值由若干个配置项用逗号隔开构成。

参数类型：USERSET

取值范围：字符串

- plan_save_mode_outline，表示从计划中推导outline进行保存。(enable_planmgmt_backend为off时，设置该配置项不会生效。)
- plan_save_tblnum_*n*，*n*为整数，取值范围0~65535。表示语句依赖的表个数大等于*n*个时，将生成的generic计划进行保存。
- plan_save_level_topsql，表示把满足topsql条件语句的非FQS的generic计划进行保存。
- plan_save_level_all，表示把所有非FQS的generic计划进行保存。
- enable_plan_baseline，表示为语句使用可用的绑定计划。

默认值：升级场景下保持前向兼容，即与升级前的集群中该参数的默认值保持一致。新安装集群场景下，该参数默认值为空串''，即不开启计划管理功能。

 版权所有 © 华为云计算技术有限公司

说明

- 配置多个兼容性配置项时，相邻配置项用逗号隔开，例如：
  ```
  set planmgmt_options='plan_save_mode_outline,plan_save_tblnum_4,enable_plan_baseline';
  ```
- 此参数选项中plan_save_level_topsql和plan_save_level_all不能同时设置。
- 设置plan_save_mode_outline选项值时，enable_planmgmt_backend参数需设置为on，否则不会生效。
- 该参数的设置是覆盖型的，即最新一次设置会覆盖上次的设置。

enable_generate_plan_hash

参数说明：控制是否为计划生成plan_hash。

参数类型：USERSET

取值范围：布尔型

- on表示开启为计划生成plan_hash。
- off表示关闭为计划生成plan_hash

默认取值：off

hint_option

参数说明：控制绑定outline和手工hint同时存在时hint生效的优先级。

参数类型：USERSET

取值范围：枚举型

- bind_hints_first：当绑定的outline和手工写的hint同时存在时，outline对应版本支持的hint优先手工hint生效。比如：821.001版本的outline支持leading、join、scan hint，那么当SQL语句中存在手工写的这三类hint时，只有outline中的这三类hint生效，手工写的hint不生效。除这三类hint外的手工hint和outline同时生效。
- ignore_manual_hints：表示语句存在绑定outline时，忽略语句中的手动写的hint。
- ignore_bind_hints：表示语句存在绑定outline时，忽略绑定的outline中的hint。

默认取值：bind_hints_first

18.25 GTM 相关参数

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

须知

当client_min_messages和log_min_messages取值相同时，其值所代表的级别不同。

版权所有 © 华为云计算技术有限公司

参数类型：SUSET

取值范围：枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见表18-1。

默认值：warning

enable_multi_sequence_file

参数说明：控制是否开启GTM中的gtm.sequence文件拆分。开启该参数后sequence的持久化操作会按照uuid分组，提高并发性能。该参数仅9.1.1.300及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔型

- on：开启GTM中的gtm.sequence文件拆分。
- off：关闭GTM中的gtm.sequence文件拆分。

默认值：off

说明

- 开启该参数后即开始切换过程，不支持回退，即使再次关闭也无法回退。因为sequence的持久化操作按照uuid分组依赖：
  1. enable_multi_sequence_file参数。
  2. 拆分后的sequence目录是否存在。
- 升级集群场景：禁止在升级观察期开启该参数。如需开启需在升级提交后操作，采用运行期逐步切换的方式，或者在升级提交后开启该参数并重启GTM以便采用启动时全量切换的方式。
- 新建集群场景：启动GTM时直接开启。新建集群不涉及文件切换，因此可以在GTM启动时直接开启该参数。
- 在运行期间切换，需要等待seq操作覆盖所有的分区后才会完成最后的文件切换。如果某些seq分区长期没有被修改，就会长期处于中间状态。处于中间状态不会影响seq操作性能，可以通过创建128个以上的新sequence来激活最后的切换。
- 经过DWS数据验证，10w条seq数据记录的场景下，GTM启动之前设置该参数为on，即使用全量切换，几乎没有额外耗时。

18.26 扩容相关参数

enable_cluster_resize

参数说明：设置是否标识当前会话为扩容重分布会话。

参数类型：SUSET

取值范围：布尔型

- on，标识当前会话为扩容重分布会话，可以执行重分布专有SQL语句。
- off，标识当前会话为非扩容重分布会话，不能执行重分布专有SQL语句。

默认值：off

 版权所有 © 华为云计算技术有限公司

> **⚠ 注意**
>
> 此参数仅用于内部运维场景的扩容重分布会话，禁止在其他业务会话中开启。

enable_auto_insert_into_new_group

参数说明：扩容期间对表执行INSERT OVERWRITE操作时，控制新表是否创建到目标节点组（nodegroup）。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示创建到扩容目标节点组。
- off表示创建到原节点组。

默认值：on

disable_insert_into_newnodes

参数说明：控制扩容期间，新表是否可以创建到目标节点组（nodegroup）。

参数类型：SIGHUP

取值范围：布尔型

- on表示禁止创建到扩容目标节点组，默认创建到原节点组。
- off表示默认创建到目标节点组。

默认值：off

> **⚠ 注意**
>
> 该参数由扩容进程自动控制，不建议用户修改。

18.27 其他选项

server_version

参数说明：报告服务器版本号（字符串形式）。

参数类型：SIGHUP

取值范围：字符串

默认值：9.2.4

server_version_num

参数说明：报告服务器版本号（整数形式）。

参数类型：SIGHUP

 版权所有 © 华为云计算技术有限公司

取值范围：整数，90204~2147483647

默认值：90204

sql_dialect

参数说明：为当前数据库绑定方言插件。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：mysql

默认值：空

说明

需先安装sql_dialect插件，使用ALTER DATABASE *database_name* SET sql_dialect = 'mysql'; 语句设置该参数。

dfs_partition_directory_length

参数说明：在HDFS文件系统上，构造HDFS VALUE分区表的分区目录时，目录名长度的上限值。

参数类型：USERSET

取值范围：92～7999

默认值：512

enable_hadoop_env

参数说明：设置使用Hadoop特性时，是否允许在数据库中创建本地行存表和列存表。DWS集群中，集群安装好后，该参数默认设为off。以支持本地行列存储和跨集群访问Hadoop特性。不推荐用户调整enable_hadoop_env的值。

参数类型：USERSET

取值范围： 布尔型

- on/true，表示使用Hadoop特性时，不允许在数据库中创建本地行存表和列存表。
- off/false，表示使用Hadoop特性时，可以在数据库中创建本地行存表和列存表。

默认值：off

enable_upgrade_merge_lock_mode

参数说明：当该参数设置为on时，通过提升deltamerge内部实现的锁级别，避免和update/delete并发操作时的报错。

参数类型：USERSET

取值范围： 布尔型

- on，提升deltamerge内部实现的锁级别，并发执行deltamerge和update/delete操作时，一个操作先执行，另一个操作被阻塞，在前一个操作完成后，后一个操作再执行。

 版权所有 © 华为云计算技术有限公司

- off，在对HDFS表的delta table的同一行并发执行deltamerge和update/delete操作时，后一个对同一行数据更新的操作会报错退出。

默认值：off

job_queue_processes

参数说明：表示系统可以并发执行的job数目。

参数类型：POSTMASTER

取值范围：0～1000

功能：

- 当job_queue_processes设置为0值，表示不启用定时任务功能，任何job都不会被执行（因为开启定时任务的功能会对系统的性能有影响，有些局点可能不需要定时任务的功能，可以通过设置为0不启用定时任务功能）。
- 当job_queue_processes为大于0时，表示启用定时任务功能且系统能够并发处理的最大任务数。

启用定时任务功能后，job_scheduler线程会在定时时间间隔轮询pg_jobs系统表，系统设置定时任务检查周期默认为1s。

由于并行运行的任务数太多会消耗更多的系统资源，因此需要设置系统并发处理的任务数，当前并发的任务数达到job_queue_processes时，且此时又有任务到期，那么这些任务本次得不到执行而延期到下一轮询周期。因此，建议用户需要根据每个任务的执行时长合理地设置任务的时间间隔（即submit接口中的interval参数），来避免由于任务执行时间太长而导致下个轮询周期无法正常执行。

注：如果同一时间内并行的job数很多，过小的参数值会导致job等待。而过大的参数值则消耗更多的系统资源，建议设置此参数为100，用户可以根据系统资源情况合理调整。

默认值：10

ngram_gram_size

参数说明：ngram解析器分词的长度。

参数类型：USERSET

取值范围：整型 ，1～4

默认值：2

ngram_grapsymbol_ignore

参数说明：ngram解析器是否忽略图形化字符。

参数类型：USERSET

取值范围：布尔型

- on表示忽略图形化字符。
- off表示不忽略图形化字符。

默认值：off

 版权所有 © 华为云计算技术有限公司

ngram_punctuation_ignore

参数说明：ngram解析器是否忽略标点符号。

参数类型：USERSET

取值范围：布尔型

- on表示忽略标点符号。
- off表示不忽略标点符号。

默认值：on

zhparser_multi_duality

参数说明：Zhparser解析器设定是否将长词内的文字自动以二字分词法聚合。

参数类型：USERSET

取值范围：布尔型

- on表示将长词内的文字自动以二字分词法聚合。
- off表示不将长词内的文字自动以二字分词法聚合。

默认值：off

zhparser_multi_short

参数说明：Zhparser解析器分词执行时是否执行针对长词复合切分。

参数类型：USERSET

取值范围：布尔型

- on表示执行针对长词复合切分。
- off表示不执行针对长词复合切分。

默认值：on

zhparser_multi_zall

参数说明：Zhparser解析器是否将全部单字单独显示。

参数类型：USERSET

取值范围：布尔型

- on表示将全部单字单独显示。
- off表示不将全部单字单独显示。

默认值：off

zhparser_multi_zmain

参数说明：Zhparser解析器是否将重要单字单独显示。

参数类型：USERSET

 版权所有 © 华为云计算技术有限公司

取值范围：布尔型

- on表示将重要单字单独显示。
- off表示不将重要单字单独显示。

默认值：off

zhparser_punctuation_ignore

参数说明：Zhparser解析器分词结果是否忽略所有的标点等特殊符号（不会忽略\r和\n）。

参数类型：USERSET

取值范围：布尔型

- on：忽略所有的标点等特殊符号。
- off：不忽略所有的标点等特殊符号。

默认值：on

zhparser_seg_with_duality

参数说明：Zhparser解析器是否将闲散文字自动以二字分词法聚合。

参数类型：USERSET

取值范围：布尔型

- on表示将闲散文字自动以二字分词法聚合。
- off表示不将闲散文字自动以二字分词法聚合。

默认值：off

acceleration_with_compute_pool

参数说明：在查询包含OBS时，通过该参数决定查询是否通过计算资源池进行加速。

参数类型：USERSET

取值范围：布尔型

- on表示包含有OBS的查询在计算资源池可用时，会根据代价评估决定是否通过计算资源池对查询加速。
- off表示任何查询都不会通过计算资源池进行加速。

默认值：off

redact_compat_options

参数说明：设置数据脱敏可算不可见兼容性行为配置项。该参数仅8.1.3.310及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

 版权所有 © 华为云计算技术有限公司

- none，表示未配置兼容项。
- disable_comparison_operator_mask，表示允许不存在暴露原始数据风险的比较操作符绕过脱敏检查，输出实际计算结果。
- relaxed_redact_check，表示对数据脱敏列类型校验进行弱管控，支持修改依赖脱敏的表字段列类型。该配置项仅9.1.0.222及以上的9.1.0.xxx集群版本支持；9.1.1.200及以上集群版本支持。

默认值：none

table_skewness_warning_threshold

参数说明：设置用于表倾斜告警的阈值。

参数类型：USERSET

取值范围：浮点型，0～1

默认值：1

table_skewness_warning_rows

参数说明：设置用于表倾斜告警的行数。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：100000

enable_view_update

参数说明：用于设置是否开启视图更新功能。

参数类型：POSTMASTER

取值范围：布尔型

- on表示启用视图更新功能。
- off表示关闭视图更新功能。

默认值： on

须知

新安装9.1.1.100及以上集群版本中该参数的默认值为on，升级场景下该参数的默认值为保持前向兼容维持原值。

view_independent

参数说明：用于设置是否开启视图与表、函数、同义词的解耦功能。基表恢复后目前已支持自动关联重建。

参数类型：SIGHUP

取值范围：布尔型

 版权所有 © 华为云计算技术有限公司

- on表示启用视图解耦功能，存在视图依赖的表、函数、同义词及其他视图可以单独删除（临时表及临时视图除外），关联视图保留但不可用。
- off表示关闭视图解耦功能，存在视图依赖的表、函数、同义词及其他视图不可以单独删除，仅可使用cascade级联删除。

默认值： off

assign_abort_xid

参数说明：查询时将指定的xid判断为需要abort的事务。

参数类型：USERSET

取值范围：指定xid的字符串

注意

此参数只用于用户误删数据(delete操作)后进行快速恢复。其他场景禁止使用，否则造成事务可见性错误问题。

default_distribution_mode

参数说明：用于设置表的默认分布方式。该参数仅8.1.2及以上版本支持。

参数类型：USERSET

取值范围：枚举类型

- roundrobin，创建表不指定分布方式时，按如下规则选取默认分布方式：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，则选取ROUNDROBIN分布。
- hash，创建表不指定分布方式时，按如下规则选取默认分布方式：
  a. 若建表时包含主键/唯一约束，则选取HASH分布，分布列为主键/唯一约束对应的列。
  b. 若建表时不包含主键/唯一约束，但存在数据类型支持作分布列的列，则选取HASH分布，分布列为第一个数据类型支持作分布列的列。
  c. 若建表时不包含主键/唯一约束，也不存在数据类型支持作分布列的列，选取ROUNDROBIN分布。

默认值：roundrobin

说明

新建8.1.2集群版本默认值为roundrobin，升级到8.1.2集群版本场景该参数的默认值为hash。

object_mtime_record_mode

参数说明：用于设置PG_OBJECT系统表中mtime字段的更新行为。

参数类型：SIGHUP

取值范围：字符串

 版权所有 © 华为云计算技术有限公司

- default，表示默认行为包括ALTER、COMMENT、GRANT/REVOKE和TRUNCATE操作会更新mtime字段。
- disable，表示不更新mtime字段。
- disable_acl，表示GRANT/REVOKE操作不更新mtime字段。
- disable_truncate，表示TRUNCATE操作不更新mtime字段。
- disable_partition，表示分区表相关ALTER操作不更新mtime字段。

默认值：default

max_volatile_tables

参数说明：指定每个session创建的volatile临时表最大个数，其中包括volatile临时表自身及其附属表。该参数仅8.2.0及以上集群版本支持。

参数类型：USERSET

取值范围：整型，0~INT_MAX

默认值：300

query_cache_refresh_time

参数说明：用于控制enable_accelerate_select生效的查询的缓存刷新时间。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：浮点型，0~10000.0，单位为秒（s）。

默认值：60.0

default_cache_distribution_mode

参数说明：用于控制自动创建物化视图的分布方式。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- roundrobin：表的每一行被轮番地发送给各个DN，因此数据会被均匀地分布在各个DN中。
- replication：表的每一行存储在所有DN中，即每个数据节点都有完整的表数据。

默认值：roundrobin

vector_engine_strategy

参数说明：用于控制向量化增强策略。该参数仅8.3.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- force表示向量化增强策略设置为存在不支持向量化场景时，强制回退为行存计划。

 版权所有 © 华为云计算技术有限公司

- improve表示向量化增强策略设置为存在不支持向量化场景时，启用向量化增强，尽可能将计划向量化。

默认值：improve

default_temptable_type

参数说明：用于控制CREATE TABLE创建临时表时在TEMP或TEMPORARY前未指定表类型所创建临时表的类型。该参数仅9.1.0及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- local：表示在未指定类型的情况下创建local型临时表。
- volatile：表示在未指定类型的情况下创建volatile型临时表。

默认值：local

hudi_sync_max_commits

参数说明：设置Hudi同步任务单次同步的最大commits的数量。该参数仅9.1.0.100及以上集群版本支持。

参数类型：USERSET

取值范围：整型，-1~INT_MAX，其中-1和0表示不限制commits的数量。

默认值：-1

foreign_table_default_rw_options

参数说明：控制创建外表时未指定权限时的默认权限。仅9.0.3及以上版本支持。

参数类型：USERSET

取值范围：字符串

- READ_ONLY，表示只读权限。
- WRITE_ONLY，表示只写权限。
- READ_WRITE，表示读写权限。

默认值：READ_ONLY

analyze_options

参数说明：用于控制ANALYZE命令相关特性和功能。该参数仅9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：枚举类型

- none：表示不启用任何ANALYZE的特性和功能选项。
- analyze_on_vw ： 表示ANALYZE支持在弹性VW上执行。
- runtime_v3_table：表示对列存3.0表支持查询触发的Runtime Analyze。

 版权所有 © 华为云计算技术有限公司

- force_sample_new_data，表示ANALYZE支持从hstore opt的delta表中采样最新数据。该取值仅9.1.1.210及以上集群版本支持。
- disable_runtime_v3_part_table，表示对列存3.0分区表禁用查询触发的Runtime Analyze。该选项仅9.1.2及以上集群版本支持。

默认值：analyze_on_vw, force_sample_new_data

allow_system_table_vacuum_full

参数说明：设置是否允许VACUUM FULL系统表。该参数仅9.1.0.226及以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on表示允许VACUUM FULL系统表。
- off表示不允许VACUUM FULL系统表。设置为off时，对指定系统表执行VACUUM FULL时会报错退出。若不指定表对整个数据库执行VACUUM FULL时，会自动跳过所有的系统表。

默认值：off

警告

VACUUM FULL系统表只能离线操作，在线VACUUM FULL系统表除了会锁表，还可能导致一些异常情况并产生报错。

enable_recycle_default_obs_tbs

参数说明：控制是否对default_obs_tbs（冷热表专用OBS表空间）进行残留文件扫描和清理。

DWS冷热表在使用中，可能由于系统故障导致冷热表专用OBS表空间default_obs_tbs生成无法自动清除的残留文件。随着这些文件的不断积累，它们会占用大量磁盘空间，导致磁盘使用率显示不准确，进而影响数据库的正常运维和运行。

开启此功能开关，可以自动清理default_obs_tbs中的残留文件。

该参数仅9.1.1.100及以上集群版本支持。

参数类型：SUSET

取值范围：布尔型

- on表示允许扫描和清理default_obs_tbs。
- off表示不允许扫描和清理default_obs_tbs。

默认值：off

enable_pg_db_file

参数说明：控制V3表写CU文件时文件信息是否记录到PG_DB_FILE系统表，以及V3表的列存vacuum，表大小计算和空间管控等功能中获取文件信息的方式统一改为从PG_DB_FILE系统表遍历获取。该参数仅9.1.1.100及以上集群版本支持。

 版权所有 © 华为云计算技术有限公司

参数类型：USERSET

取值范围：布尔型

- on表示允许V3表写CU文件时文件信息记录到PG_DB_FILE系统表，同时V3表的列存vacuum，表大小计算和空间管控等功能中获取文件信息的方式统一改为从PG_DB_FILE系统表遍历获取。
- off表示不允许V3表写CU文件时文件信息记录到PG_DB_FILE系统表，同时V3表的列存vacuum，表大小计算和空间管控等功能中获取文件信息的方式不变。

默认值：on

说明

- 9.1.1.100及以上新装集群默认为on，低版本集群升级到9.1.1.100及以上集群默认为off。
- 不建议已经打开enable_pg_db_file参数的集群再关闭该参数，否则会导致后续再打开时PG_DB_FILE系统表里的文件信息不准确。
- 若升级后的集群打开enable_pg_db_file，会通过autovacuum在PG_DB_FILE系统表上重建升级前表上的文件信息，因此需要保证列存autovacuum开启（即autovacuum_max_workers_col > 0，colvacuum_threshold_scale_factor > 0），只有重建完成后，对该表的vacuum才可以避免obs list操作。

installation_group_alias

参数说明：用于指定installation nodegroup的别名，在物理集群扩容后，确保CREATE TABLE...to group命令能够正确地指向当前的node group。该参数9.1.1.100及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

默认值：空

说明

该参数支持设置为'all'， 表示CREATE TABLE...to group指定的nodegroup不存在时，默认创建到installation group。

enable_external_column_index_access

参数说明：控制external schema外表字段与实际文件字段的匹配方式。仅支持ORC和PARQUET格式。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：布尔型

- on表示外表字段与文件字段按顺序匹配。
- off表示外表字段与文件字段按名字匹配，无法匹配的字段使用null填充。

默认值：on

max_partition_per_table

参数说明：设置创建分区表的最大分区数。该参数仅9.1.1.200及以上集群版本支持。

参数类型：SIGHUP

 版权所有 © 华为云计算技术有限公司

取值范围：整型，1~32767

默认值：4096

cache_partition_options

参数说明：设置数据批量写入分区表时，系统是否会缓存最近一次分区结果。该参数仅9.1.1.200及以上集群版本支持。

参数类型：SIGHUP

取值范围：枚举类型

- range，表示该功能支持range分区类型的表。
- list，表示该功能支持list分区类型的表。
- none，表示关闭该功能。

默认值：range,list

enable_obs_limiter

参数说明：设置是否开启OBS异步IO限流，开启后限流配置生效。该参数仅9.1.1.200及以上集群版本支持。

参数类型：SIGHUP

取值范围：布尔类型

- on表示开启限流，当存在限流配置时，匹配对应的限流配置进行限流。
- off表示关闭限流，即使存在限流配置，也不进行限流。

默认值：on

hnsw_options

参数说明：设置pgvector hnsw索引相关查询选项。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

- ef_search，表示控制查询参数（查询候选数），取值范围为1~1000。
- max_scan_tuples，表示最大扫描节点数，取值范围为1~INT_MAX。
- mem_scan_multiplier，表示内存动态扩张率，取值范围为1~1000。

默认值：40,20000,1

注意

设置该参数时字符串内包含三个参数值，以逗号作为分隔。不支持单个参数独立设置。

 版权所有 © 华为云计算技术有限公司

ivfflat_options

参数说明：设置pgvector ivfflat索引相关查询选项。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：字符串

- probes，表示扫描簇数，取值范围为1~32768。
- max_probes，表示最大扫描簇数，取值范围为1~32768。

默认值：1,32768

注意

设置该参数时字符串内包含两个参数值，以逗号作为分隔。不支持单个参数独立设置。

ivfflat_iterative_scan

参数说明：设置pgvector ivfflat索引迭代扫描模式，开启会自动扫描更多索引来获取足够的数据量。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- relaxed_order，表示开启，但结果不按实际向量距离排序。
- off，表示不开启。

默认值：off

hnsw_iterative_scan

参数说明：设置pgvector hnsw索引迭代扫描模式，开启会自动扫描更多索引来获取足够的数据量。该参数仅9.1.1.200及以上集群版本支持。

参数类型：USERSET

取值范围：枚举型

- relaxed_order， 表示开启，但结果不按实际向量距离排序。
- strict_order，表示开启，且结果严格按照实际向量距离排序。
- off，表示不开启。

默认值：off

 版权所有 © 华为云计算技术有限公司

19 DWS 开发者术语表

A~E

表 19-1 A~E 术语解释

术语	解释
ACID	在可靠数据库管理系统（DBMS）中，事务（transaction）所应该具有的四个特性：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durability）。
安全环	每一个环都由若干物理机组成，环内的DN形成主、备、从备关系，不向环外延伸。也就是说，环内的任何一个节点的主，或者备，或从备，依然在环内。扩容与缩容时，是以环为最小单位进行的。
Bgwriter	数据库启动时创建的一个后台写线程，此线程用于将数据库中脏页面写入到持久性设备（例如磁盘）中。
bit	比特。计算机处理的最小的信息单位。比特用来表示二进制数字1或0，或者一种逻辑条件真或假。
Bloom Filter	布隆过滤器。由Howard Bloom在1970年提出的二进制向量数据结构，它具有很好的空间和时间效率，被用来检测一个元素是不是集合中的一个成员，这种检测只会对在集合内的数据错判，而不会对不是集合内的数据进行错判，这样每个检测请求返回有“在集合内（可能错误）”和“不在集合内（绝对不在集合内）”两种情况，可见Bloom filter是牺牲了正确率换取时间和空间。
CCN	Central Coordinator ，DWS动态负载管理中心协调节点。负责进行各CN中复杂作业是否可以执行的中心判断、排队和调度，以实现动态负载管理。
CIDR	Classless Inter-Domain Routing，无类域间路由IP编址方案。CIDR摒弃传统的基于类（A类：8，B类：16，C类：24）的地址分配方式，允许使用任意长度的地址前缀，有效提高地址空间的利用率。CIDR表示方法：IP地址/网络ID的位数。比如192.168.23.35/21，其中“21”表示前面地址中的前21位代表网络部分，其余位代表主机部分。

版权所有 © 华为云计算技术有限公司

术语	解释
Cgroups	Control Groups，控制组（DWS中也称之为优先级组）。DWS提供的一种可以限制、记录、隔离进程组所使用的物理资源的机制。
CLI	Command-line Interface，命令行界面。应用程序和用户交互的一种方式，完全基于文本输入和输出。命令通过键盘或类似装置输入，由程序编译并执行。结果是以文本或图形的方式呈现在终端界面。
CM	Cluster Manager，集群管理模块。管理和监控分布式系统中各个功能单元和物理资源的运行情况，确保整个系统的稳定运行。
CMS	Cluster Management Service，集群管理服务。是用于管理集群状态的部件。
CN	Coordinator，负责数据库系统元数据存储、查询任务的分解和部分执行，以及将DN中查询结果汇聚在一起。
CTE	Common Table Expression，公共表达式是一种在SQL查询中定义的临时命名结果集，通常通过 WITH关键字创建。CTE仅存在于单个查询的执行范围内，可被后续的SELECT、INSERT、UPDATE或DELETE语句多次引用。CTE主要作用是简化复杂查询的编写，提升代码可读性和维护性，合理使用CTE，能显著提升复杂SQL的可维护性，尤其在递归或分步处理数据时优势明显。
CU	Compression Unit，压缩单元。列存表的最小存储单位。
core文件	当程序出现内存越界、断言失败或者访问非法内存时，操作系统会中止进程，并将当前内存状态导出到core文件中，以便进一步分析。 core文件包含内存转储，支持全二进制和指定端口格式。core文件名称由字符串core以及操作系统进程ID组成。 core文件不依赖于任何平台。
Core Dump	通常在程序异常终止时，核心转储（Core Dump）、内存转储或系统转储用于记录特定时间计算机程序工作内存的状态。实际上，其他关键程序的状态经常在同一时间进行转储，例如处理器寄存器，包括程序指标和栈指针、内存管理信息、其他处理器和操作系统标记及信息。Core Dump经常用于辅助诊断和纠错计算机程序问题。
DBA	Database Administrator，数据库管理员。指导或执行所有和维护数据库环境相关的操作。
DBLINK	DBLINK是定义一个数据库到另一个数据库路径的对象，通过它可以查询远程数据库对象。
DBMS	Database Management System，数据库管理系统。数据库管理系统是为了访问数据库中的信息而使用的一个管理系统软件。它包含一组程序使用户可以进入、管理、查询数据库中数据。基于真实数据的位置，可以分为内存数据库管理系统和磁盘数据库管理系统。
DCL	Data Control Language，数据控制语言。
DDL	Data Definition Language，数据定义语言。

 版权所有 © 华为云计算技术有限公司

术语	解释
DFS	Distributed File System，分布式文件系统。DFS是一个通用术语，指代任何将文件存储在多个物理节点上并通过网络提供统一访问接口的系统。它不特指具体技术，而是一类系统的统称。 例如HDFS，Hadoop 分布式文件系统，它属于DFS中的一种，专为大数据场景设计，适合高吞吐量、批处理任务。
DML	Data Manipulation Language，数据操纵语言。
DN	Data Node，和CN对应的概念。负责实际执行表数据的存储、查询操作。
ETCD	Editable Text Configuration Daemon，分布式键值存储系统，用于共享配置和服务发现（服务注册和查找）。
ETL	Extract-Transform-Load，描述将数据从来源端经过抽取（extract）、转换（transform）、加载（load）至目的端的过程。
Extension Connector	Extension Connector是DWS提供的功能模块，使用它可以将SQL语句发送到集群外部的Spark，并在当前库中返回执行结果，实现跨集群处理数据。
备份	备份件或者备份过程。指复制并归档计算机数据，当发生数据丢失事件时，可以用该复制并归档的数据来恢复原始数据。
备份和恢复	保护数据库防止由于媒介失效或人为错误造成的数据丢失过程中涉及的一组概念、过程及策略。
备机	DWS双机方案中的一个节点，用于作为主机的备份，在主机异常时，备机会切换到主机状态，以确保能正常提供数据服务。
崩溃	崩溃（或系统崩溃）指计算机或程序（例如软件应用程序或操作系统）异常终止的事件。出现错误后，通常会自动退出。有时出现恶意程序冻结或挂起直到崩溃上报服务记录崩溃的详细信息。对于操作系统内核关键部分的程序，整个计算机可能瘫痪（可能造成致命的系统错误）。
编码	编码是指用代码来表示各组数据资料，使其成为可利用计算机进行处理和分析的信息。用预先规定的方法将文字、数字或其他对象编成数码，或将信息、数据转换成规定的电脉冲信号。
编码技术	呈现计算机软硬件识别的特定字符集数据的技术。
表	表是由行与列组合成的。每一列被当作是一个字段。每个字段中的值代表一种类型的数据。例如，一个表可能有3个字段：姓名、城市和国家。这个表就会有3列：一列代表姓名，一列代表城市，一列代表国家。表中的每一行包含3个字段的内容，姓名字段包含姓名，城市字段包含城市，国家字段包含国家。
表空间	包含表、索引、大对象、长数据等数据的逻辑存储结构。表空间在物理数据和逻辑数据间提供了抽象的一层，为所有的数据库对象分配存储空间。表空间创建好后，创建数据库对象时可以指定该对象所属的表空间。

版权所有 © 华为云计算技术有限公司

术语	解释
并发控制	在多用户环境下同时执行多个事务并保证数据完整性的一个DBMS服务。并发控制是DWS提供的一种多线程管理机制，用来保证多线程环境下在数据库中执行的操作是安全的和一致的。
查询	向数据库发出的信息请求，包含更新、修改、查询或删除信息的请求。
查询操作符	Query Operator，也称为查询迭代算子（Iterator）或查询节点（Query Tree Node）。一个查询的执行可以分解为一个或多个查询操作符，是构成一个查询执行的最基本单位。常见的查询操作符包括表扫描（Scan），表关联（Join），表聚集（Aggregation）等。
查询片段	每一个查询任务都可以分解成为一个或者多个查询片段。每个查询片段由一个或多个查询操作符构成，可独立在节点上运行。通过数据流操作符与其他查询片段块交换数据。
持久性	数据库事务的ACID特性之一。在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。
存储过程	存储过程（StoredProcedure）是在大型数据库系统中，一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名称并设置参数（如果该存储过程带有参数）来执行它。
操作系统	操作系统OS（operating system）由引导程序加载到计算中，对计算机中其他程序进行管理。其他程序叫做应用或应用程序。
从备	Secondary，为了保证集群的高可靠性，主、备间无法正常同步数据时，主节点会将日志同步到从备。如果主节点突然故障不可用，备节点会升主，并且升主成功后从从备节点上同步之前异常期间的日志。
大对象	大对象（Blob）在数据库中指使用二进制方式存储的数据。它通常可以用于存储视频、音频和图像等多媒体数据。
递归查询	递归查询（RECURSIVE Query）是数据库查询中的一种高级技术，主要用于处理层次结构数据或需要逐层迭代计算的场景。它通过递归调用自身来实现对树形结构、图结构或链式关系的遍历和分析。在SELECT语法中，常通过WITH RECURSIVE格式来声明递归查询。
动态负载	DWS动态负载是指基于系统中CPU、I/O、内存等资源的使用情况，自动调节并发作业的运行数量，避免因为系统资源过载导致业务报错或无响应。
段	数据库中，一段指包含一个或多个区域的数据库中的一部分。区域是数据库的最小范围，由单元调用块组成。一个或多个段组成一个表空间。

版权所有 © 华为云计算技术有限公司

F~J

表 19-2 F~J 术语解释

术语	解释
Failover	指当某个节点出现故障时，自动切换到备节点上的过程。反之，从备节点上切换回来的过程称为Failback。
FDW	Foreign Data Wrapper，外部数据封装器。是Postgres提供的一个SQL接口，用于访问远程数据存储中的大数据对象，使DBA可以整合来自不相关数据源的数据，将它们存入数据库中的一个公共模型。
Freeze	在事务ID耗尽时由AutoVacuum Worker进程自动执行的操作。DWS会把事务ID记在行头，在一个事务取得一行时，通过比较行头的事务ID和事务本身的ID判断这行是否可见，而事务ID是一个无符号整数，如果事务ID耗尽，事务ID会跨过整数的界限重新计算，此时原先可见的行就会变成不可见的行，为了避免这个问题，Freeze操作会将行头的事务标记为一个特殊的事务ID，标记了这个特殊的事务ID的行将对所有事务可见，以此避免事务ID耗尽产生的问题。
GDB	GNU工程调试器，可以监控其他程序运行时的内部情况，或者其他程序要崩溃时发生了什么。GDB支持如下四种主要操作（使PDK功能更加强大），辅助查找缺陷。 ● 启动程序，指定可能影响行为的任何因素。 ● 特定条件下，停止程序。 ● 程序停止时，检查发生了什么。 ● 修改程序内容，尝试纠正一个缺陷并继续下一个。
GDS	General Data Service（简称GDS），数据并行加载工具。向DWS导入数据时，需要将此工具部署到源数据所在的服务器上，使DN可以通过该工具获取数据。
GIN索引	Generalized Inverted Index，通用倒排索引。作用为处理索引项为组合值的情况，查询时需要通过索引搜索出出现在组合值中的特定元素值。
GNU	GNU计划，又称革奴计划，是由RichardStallman在1983年9月27日公开发起的。它的目标是创建一套完全自由的操作系统。GNU是“GNU's NotUnix”的递归缩写。Stallman宣布GNU应当发音为Guh-NOO以避免与new这个单词混淆（注：Gnu在英文中原意为非洲牛羚，发音与new相同）。Unix是一种广泛使用的商业操作系统的名称。技术上讲，GNU类似Unix。但是GNU却给了用户自由。
gsql	DWS交互终端。通过gsql能够以交互的方式输入查询，下发查询到DWS，然后查看查询结果。或者，也可以从文件中输入。此外，gsql还提供许多元命令和各种类似shell命令，协助脚本编写及自动化各种任务。
GTM	Global Transaction Manager，全局事务管理器。用于管理事务状态的部件。

版权所有 © 华为云计算技术有限公司

术语	解释
GUC	Grand Unified Configuration，数据库运行参数。配置这些参数可以影响数据库系统的行为。
HA	高可用性（HighAvailability），通过尽量缩短因日常维护操作（计划）和突发的系统崩溃（非计划）所导致的停机时间，以提高系统和应用的可用性。
HBA	host-based authentication，主机认证。主机鉴权允许主机鉴权部分或全部系统用户。适用于系统所有用户或者使用Match指令的子集。该类型鉴权对于管理计算集群以及其他完全同质设备非常有用。总之，服务器上的三个文件以及客户端上的一个文件必须修改，为主机鉴权做准备。
HDFS	Hadoop Distributed File System，Apache Hadoop项目的一个子项目。一个高度容错的分布式文件系统，设计用于在低成本硬件上运行。HDFS提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。
服务器	为客户端提供服务的软硬件的组合。单独使用时，指运行服务器操作系统的计算机，也可以指提供服务的软件或者专用硬件。
高级包	DWS提供的具有一定逻辑和功能的存储过程、函数，这些具备功能的存储过程、函数统称为高级包。
隔离性	数据库事务的ACID特性之一。它是指一个事务内部的操作及使用的数据对其他并发事务是隔离的，并发执行的各个事务之间不能互相干扰。
关系型数据库	创建在关系模型基础上的数据库。关系型数据库借助于集合代数等数学概念和方法来处理数据库中的数据。
归档线程	数据库打开归档功能时启动的一个线程，此线程用于将数据库日志归档到指定的路径。
故障接管	功能对等的系统部件对于故障部件的自动替换过程。系统部件包含处理器、服务器、网络、数据库等。
环境变量	定义进程操作环境某一方面的变量。例如，环境变量可以为主目录，命令搜索路径，使用终端或当前时区。
检查点	将数据库内存中某一时刻的数据存到磁盘的机制。DWS定期将已提交的事务数据和未提交的事务数据存到磁盘，这些数据用来和Redo日志一起在数据库重启和崩溃时恢复数据库。
加密	用于传输数据的功能。通过该功能，可以隐藏信息内容，防止非法使用。
节点	将构成DWS集群环境的各台服务器（物理机或虚拟机）称为集群节点，简称节点。
纠错	系统自动识别软件和数据流上的错误并自动修正错误的能力，提升系统的稳定性和可靠性。
进程	在单个计算机上执行程序的实例。一个进程由一个或多个线程组成。其他进程不能接入某个进程已占用的线程。

版权所有 © 华为云计算技术有限公司

术语	解释
基于时间点恢复	PITR（Point-In-Time Recovery），基于时间点恢复是DWS备份恢复的一个特性，是指在备份数据和WAL日志正常的情况下，数据可以恢复到指定时间点。
记录	在关系型数据库中，每一条记录对应表中的每一行数据。
集群	集群是由一组服务器和其他资源组成的一个单独的系统，可以实现高可用性。有的情况下，可以实现负载均衡及并行处理。

K~O

表 19-3 K~O 术语解释

术语	解释
LLVM	LLVM命名最早源自于底层虚拟机（Low Level Virtual Machine）的缩写。LLVM是构架编译器（compiler）的框架系统，以C++编写而成，用于优化以任意程序语言编写的程序的编译时间（compile-time）、链接时间（link-time）、运行时间（run-time）以及空闲时间（idle-time），对开发者保持开放，并兼容已有脚本。 DWS LLVM动态编译技术可以为每个查询生成定制化的机器码用于替换原本的通用函数。通过减少实际查询时冗余的条件逻辑判断、虚函数调用并提高数据局域性，从而达到提升查询整体性能的目的。
LVS	Linux Virtual Server，虚拟服务器集群系统，用于负责集群的负载均衡。
逻辑复制	数据库主备或两个集群间的数据同步方式。区别于通过物理日志回放方式的物理复制，逻辑复制在两个集群间传输逻辑日志或通过逻辑日志对应的SQL语句实现数据同步。
逻辑日志	数据库修改的日志记录，可直接对应为SQL语句，一般为行级记录。区别于物理日志，物理日志是记录物理页面修改的日志。
逻辑解码	逻辑解码是一种通过对xlog日志的反解实现将数据库表的所有持久更改抽取到一种清晰、易于理解的格式的处理过程。
逻辑复制槽	在逻辑复制的环境下，逻辑复制槽用于防止Xlog被系统或VACUUM回收。DWS中用于记录逻辑解码位置的对象，提供创建、删除、读取、推进等多个SQL接口函数。
MPP	Massive Parallel Processing，大规模并行处理。指利用多个机器构成集群的架构方式，也称为集群（Cluster）系统。
MVCC	Multi-Version Concurrency Control，多版本并发控制。数据库并发控制协议的一种，它的基本算法是一个元组可以有多个版本，不同的查询可以工作在不同的版本上。一个基本的好处是读和写可以不冲突。

版权所有 © 华为云计算技术有限公司

术语	解释
NameNode	NameNode是Hadoop系统中的一个中心服务器，负责管理文件系统的名字空间(namespace)以及客户端对文件的访问。
mmap	memory map，也就是内存映射。mmap是一种内存映射文件的方法，即将一个文件或者其他对象映射到进程的地址空间，文件或设备的内容可以直接作为内存中的一个区域进行访问，而不需要显式地进行读写操作，实现高效文件读写和进程间通信。
Node Group	在DWS集群里Node Group指DN的集合，是集群中的子集群。从性质上可以分为存储子集群Storage Node Group和计算子集群Computing Node Group。存储子集群用来承载本地表的数据存储，而计算子集群用来承载查询的聚集、关联计算。
NULL FIRST	在 SQL中，NULL FIRST是用于显式控制排序时NULL值位置的子句，通常出现在ORDER BY语句中。其核心作用是强制让NULL值的行排在非NULL值的行之前（即优先显示 NULL）。 当在ORDER BY中使用NULL FIRST时，无论排序是升序（ASC）还是降序（DESC），所有NULL值的行都会优先出现在结果集的最前面，非NULL值的行紧随其后（按排序规则排列）。
OLAP	Online Analytical Processing，联机分析处理。是数据仓库系统最主要的应用，专门设计用于支持复杂的分析操作，侧重对决策人员和高层管理人员的决策支持，可以根据分析人员的要求快速、灵活地进行大数据量的复杂查询处理，并且以一种直观而易懂的形式将查询结果提供给决策人员，以便准确掌握企业（公司）的经营状况，了解对象的需求，制定正确的方案。
OM	Operations Management，运维管理模块。提供集群日常运维、配置管理的管理接口、工具。
ORC	Optimized Row Columnar，一种广泛使用的Hadoop系统结构化数据的文件格式，由Hadoop HIVE项目引入。
客户端	连接或请求其他计算机或程序服务的计算机或程序。
空闲空间管理	管理表内空闲空间的机制，通过记录每个表内空闲空间信息，并建立易于查找的数据结构，可以加速对空闲空间进行的操作（例如INSERT）。
跨集群	DWS支持通过外表和Extension Connector访问当前DWS集群外的其他DBMS中的数据，这种行为称为跨集群。
垃圾元组	是指使用DELETE和UPDATE语句删除的元组，DWS在删除元组时只是打个删除标记，由Vacuum线程清理这种垃圾元组。
列	字段的等效概念。在数据库中，表由一列或多列组成。
逻辑节点	一个物理节点上可以安装多个逻辑节点。一个逻辑节点是一个数据库实例。
模式	数据库对象集，包括逻辑结构，例如表、视图、序、存储过程、同义名、索引、集群及数据库连接。
模式文件	用于决定数据库结构的SQL文件。

版权所有 © 华为云计算技术有限公司

P~T

表 19-4 P~T 术语解释

术语	解释
Page	DWS数据库关系对象结构中行存的最小存储单元。一个页（Page）大小默认为8KB。默认情况下，页大小在数据库初始化时确定，并且无法动态更改。
PostgreSQL	PostgreSQL是一个开源的关系数据库管理系统(DBMS)，由全球志愿者团队开发。PostgreSQL不受任何公司或个体所控制，源代码免费使用。
Postgres-XC	一款多节点同步，读写可扩展的PostgreSQL集群数据库。
Postmaster	数据库服务启动时启动的一个线程。用于监听来自集群其他节点或客户端的连接请求。 主机上监听到备机连接请求，并接收后，就会创建一个WAL Sender线程，用于处理与备机的交互。
RHEL	Red Hat Enterprise Linux，红帽企业Linux。
REDO日志	记录对数据库进行操作的日志，这些日志包含重新执行这些操作所需要的信息。当数据库故障时，可以利用REDO日志将数据库恢复到故障前的状态。
SCTP	Stream Control Transmission Protocol，流控制传输协议。是IETF于2000年新定义的一个传输层协议。是提供基于不可靠传输业务的协议之上的可靠的数据报传输协议。SCTP的设计用于通过IP网传输SCN窄带信令消息。
Savepoint	保存点。是一种在关系数据库管理系统中实现子事务（也称为嵌套事务）的方法。在一个长事务中，可以把操作过程分成几部分，前面部分执行成功后，可以建一个保存点，若后面的执行失败，则回滚到这个保存点即可，无需回滚整个事务。保存点对于在数据库应用程序中实现复杂错误恢复很有用。如果在多语句事务中发生错误，则应用程序能够从错误中恢复（通过回滚到保存点）而无需中止整个事务。
Session	数据库系统在接收到应用程序的连接请求时，为该连接创建的一个任务。它被Session Manager管理，完成一些初始化任务，执行用户的所有操作。
Shared-nothing architecture	无共享架构是一种分布式计算架构，这种架构中不存在集中共享CPU、存储的状态，这种架构具有非常强的扩展性。
SIMD	Single Instruction Multiple Data，即单指令多数据，是一种并行计算技术，允许一条指令同时处理多个数据，从而显著提升计算密集型任务的性能。

版权所有 © 华为云计算技术有限公司

术语	解释
SLES	SUSE Linux Enterprise Server，由SUSE提供的企业级Linux操作系统。
SMP	Symmetric Multi-Processing，对称多处理技术，是指在一台计算机上汇集了一组处理器（多CPU），各CPU之间共享内存子系统以及总线结构。操作系统必须支持多任务和多线程处理，以使得SMP系统发挥高效的性能。数据库领域的SMP并行技术，一般指利用多线程技术实现查询的并行执行，以充分利用CPU资源，从而提升查询性能。
SQL	Structure Query Language，结构化查询语言。数据库的标准查询语言。它可以分为数据定义语言（DDL），数据操纵语言（DML）和数据控制语言（DCL）。
SSL	Secure Socket Layer，安全套接层。SSL是Netscape公司率先采用的网络安全协议。它是在传输通信协议（TCP/IP）上实现的一种安全协议，采用公开密钥技术。SSL广泛支持各种类型的网络，同时提供三种基本的安全服务，它们都使用公开密钥技术。SSL支持服务通过网络进行通信而不损害安全性。它在客户端和服务器之间创建一个安全连接。然后通过该连接安全地发送任意数据量。
收敛比	交换机下行带宽与上行带宽的比值。收敛比越高，流量收敛程度越大，丢包越严重。
TCP	Transmission Control Protocol，传输控制协议。用于将数据信息分解成信息包，使之经过IP协议发送；并对利用IP协议接收来的信息包进行校验并将其重新装配成完整的信息。TCP是面向连接的可靠协议，能够确保信息的无误发送。
trace	一种特殊的日志记录方法，用来记录程序执行的信息。程序员使用该信息进行纠错。另外，根据trace日志中信息的类型和内容，有经验的系统管理员、技术支持人员及软件监控工具来诊断软件常见问题。
逃逸字符	在数据库中，逃逸字符（Escape Character）是一种特殊字符，用于转义其他字符，它的主要作用是让数据库系统将原本具有特殊含义的字符当作普通字符处理，让数据库识别并正确处理那些可能与SQL语法冲突的字符（如引号、特殊符号等）。常见的逃逸字符有反斜杠\或双引号''。
全备份	备份整个数据库集群。
全量同步	DWS双机方案中的一种数据同步机制，是指把主机中的所有数据同步给备机。
日志文件	计算机记录自身活动的记录。
事务	数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成，事务必须满足ACID原则。
数据	事实或指令的一种表达形式，适用于人为或自动的通信、解释或处理。数据包含常量、变量、阵列和字符串。
数据重分布	用户改变数据的分布方式后，数据表在节点间重新分布的过程。

 版权所有 © 华为云计算技术有限公司

术语	解释
数据分布	表数据在分布式环境中的分布方式（Distribution），即数据表以何种方式打散存储到各个数据库实例上去。具体的分布方式可以有：散列（Hash）方式，复制方式（Replication）和随机方式（Random）。散列方式根据元组中指定字段的取值算得哈希值，根据节点与哈希值的映射关系获得该元组的目标存储位置。复制方式将元组复制到所有节点上。随机方式将数据随机分布到各节点。
数据分区	数据分区是指在一个数据库实例内部，将表按照划分为多个数据互不重叠的部分（Partition）。具体的分区方式可以有：范围分区（Range），它根据元组中指定字段的取值所处的范围映射到目标存储位置。
数据库	数据库是存储在一起的相关数据的集合，这些数据可以被访问，管理以及更新。同一视图中，数据库可以根据存储内容类型分为以下几类：数目类、全文本类、数字类及图像类。
数据库实例	一个数据库实例是一个DWS进程以及它控制的数据库文件。DWS在一个物理节点上安装多个数据库实例，集群各节点上所安装的GTM、CM、CN、DN统称为实例。一个数据库实例也被称为一个逻辑节点。
数据库双机	DWS提供的高可靠性双机方案。在此方案中，每个DWS逻辑节点标识为主机或备机。在同一时间内，只有一个DWS被标识为主机。双机初次建立时，主机会对每个备机数据做全量同步，然后做增量同步。双机建立之后的运行过程中，主机能接收数据读和写的操作请求，备机只做日志同步。
数据库文件	保存用户数据和数据库系统内部数据的二进制文件。
数据流操作符	负责查询片段间交换数据的操作符。根据数据流的输入、输出关系，可以细分为聚合流（Gather）、广播流（Broadcast）和重分布流（Redistribution）。聚合流将数据从多个查询片段聚合到一个。广播流将数据从一个查询片段向多个传输。重分布流则将多个查询片段的数据，按照一定规则重组后向多个查询片段传输。
数据字典	数据字典是一系列只读的表，用来提供数据库的信息。这些信息包括：数据库设计信息、存储过程信息、用户权限、用户统计数据、数据库进程信息、数据库增长统计数据和数据库性能统计数据。
死锁	为使用同一资源而产生的无法解决的争用状态。
索引	数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。
统计信息	数据库使用统计信息估算查询代价，以查找代价最小的执行计划，统计信息一般是数据库自动收集的，包括表级信息（元组数、页面数等）和列级信息（列的值域分布直方图）。
停用词	在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为Stop Words（停用词）。

 版权所有 © 华为云计算技术有限公司

U~Z

表 19-5 U~Z 术语解释

术语	解释
UNLOGGED表	即非日志表，Unlogged表是一种特殊类型的表，其数据操作不记录预写日志（WAL，Write-Ahead Logging）。这类表在特定场景下能显著提升写入性能，但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动截断，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。 CREATE TABLE语法中，指定UNLOGGED参数，则创建为非日志表。
Vacuum	数据库定期启动的清理垃圾元组的线程，根据配置参数可以同时启动多个。
V2表	指的是建表环节时，执行CREATE TABLE语法中定义的colversion取值为2.0的表，表示列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名，数据存储在本地盘。存算一体场景下，不指定colversion取值时，用户创建的列存表默认为V2表。
V3表	指的是建表环节时，执行CREATE TABLE语法中定义的colversion取值为3.0的表，即存算分离表，表示列存表的每列合并存储在一个文件中，文件名以C1_field.0命名，数据存储在OBS文件系统。存算分离场景下，不指定colversion取值时，用户创建的列存表默认为V3表。
verbose	verbose选项指定显示在屏幕上的处理信息。
WAL	Write-Ahead Logging，预写日志系统。实现事务日志的标准方法，是指对数据文件（表和索引的载体）持久化修改之前必须先持久化相应的日志。
WAL Receiver	数据库复制时备机创建的一个线程的名称。此线程用于从主机接收数据、命令，并反馈确认信息至主机。一个备机只有一个WAL Receiver线程。
WAL Sender	数据库复制过程中，主机接收到备机的连接请求后创建的一个线程的名称。此线程用于发送命令、数据到备机，并从备机接收信息。一个主机可能会有多个WAL Sender线程，每一个WAL Sender线程对应一个备机的一个连接请求。
WAL Writer	数据库启动时创建的一个写Redo日志的线程，用于将内存中的日志写入到持久性设备（如：磁盘）。
WLM	WorkLoad Manager，负载管理。DWS中系统资源控制和分配的模块。
Xlog	表示事务日志，一个逻辑节点中只有一个，不允许创建多个Xlog文件。
xDR	详单。用户面和信令面详单的统称，包括CDR和UFDR、TDR和SDR。

版权所有 © 华为云计算技术有限公司

术语	解释
网络备份	网络备份为各种平台提供一套完整的、灵活的数据保护方案。平台包含Windows、UNIX及Linux。网络备份支持备份、归档、恢复计算机上的文件、文件夹或目录、卷或分区。
谓词	谓词（Predicate）指的是一个逻辑表达式，用于筛选或限定数据。它本质上是一个返回布尔值(TRUE/FALSE/NULL)的条件表达式。像WHERE、CHECK及JOIN子句中都会用到谓词。例如WHERE salary > 5000，“salary > 5000”就是谓词。
谓词列	谓词列指的是在谓词表达式里被引用的列。例如，在salary > 5000这个谓词中，salary就是谓词列。若查询中的谓词列存在索引，查询优化器可能会利用该索引来加速数据的过滤，这种情况被称为“索引扫描”。DWS会收集谓词列的统计信息（9.1.0.100及以上集群版本支持），以此来辅助生成更优的查询计划。
物理节点	一个物理机器称为一个物理节点。
物化视图	物化视图（Materialized View）是数据库中一种特殊类型的数据库对象。它将复杂计算的查询结果预计算并存储在数据库中，实现加速查询的目的。
系统表	存储数据库元信息的表，元信息包括数据库中的用户表、索引、列、函数和数据类型等。
下推	DWS是分布式数据库，其可以利用多DN分布式并行执行查询计划，即将CN中的查询计划下发到各DN中并行执行。这种行为称为下推。与将数据抽取到CN上执行查询的方式相比，下推可以大幅提升查询性能。
压缩	数据压缩，信源编码，或比特率降低涉及使用相比原来较少比特的编码信息。压缩可以是有损或无损。无损压缩通过识别和消除统计冗余降低比特位。无损压缩中没有信息丢失。有损压缩识别并删除次要信息，减少了比特位。减少数据文件大小的方法被普遍称为数据压缩，尽管其正式名称为源编码（数据源的编码，然后将其存储或传输）。
一致性	数据库事务的ACID特性之一。在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
元数据	用来定义数据的数据。主要是描述数据自身信息，包含源、大小、格式或其他数据特征。数据库字段中，元数据用于理解以及诠释数据仓库的内容。
原子性	数据库事务的ACID特性之一。整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚到事务开始前的状态，就像这个事务从来没有执行过一样。
Zhparser	Zhparser是一个针对中文全文检索的扩展插件。它的主要功能是提供中文分词支持，使DWS能够更高效地处理中文文本的搜索和分析。
在线扩容	在线扩容是指DWS扩容重分布过程中支持数据持续入库、查询业务不中断。

 版权所有 © 华为云计算技术有限公司

术语	解释
脏页面	已经被修改且未写入持久性设备的页面。
增量备份	基于上次有效备份之后对文件修改的备份。
增量同步	DWS双机方案中的一种数据同步机制，是指把主机中数据增量同步给备机，即只同步主备间有差异的数据。
主机	DWS数据库双机系统中接收数据读写操作的节点，和所有备机一起协同工作。在同一时间内，双机系统中只有一个节点被标识为主机。
主题词	在标引和检索中用于表达文献主题的规范化的词或词组。
转储文件	转储文件是一种特定类型的trace文件。转储文件为响应事件过程中一次性输出的诊断数据，trace文件指诊断数据的连续输出。
资源池	资源池是DWS提供的一种资源划分的配置机制。通过将用户绑定到资源池，来限定其所执行作业的优先级及能够利用到的资源。
租户	数据库业务用户在给定的计算资源（cpu，内存和io）和存储资源下执行业务，通过资源管理和隔离，达成业务的服务等级协定（SLA）。
最小恢复点	最小恢复点是DWS提供的数据一致性保障手段之一。最小恢复点特性可以在DWS启动时检查出WAL日志和持久化到磁盘的数据的不一致性，并提示用户进行处理。

版权所有 © 华为云计算技术有限公司