数据湖探索

Spark SQL 语法参考

文档版本 01

发布日期 2025-08-06

华为技术有限公司

版权所有 © 华为技术有限公司 2025。 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。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

 版权所有 © 华为技术有限公司

目 录

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

1 Spark SQL 常用配置项说明

本章节为您介绍DLI 批作业SQL语法的常用配置项。

表 1-1 常用配置项

名称	默认值	描述
spark.sql.files.maxRecordsPerFile	0	要写入单个文件的最大记录数。如果该值为零或为负，则没有限制。
spark.sql.shuffle.partitions	200	为连接或聚合过滤数据时使用的默认分区数。
spark.sql.dynamicPartitionOverwrite.enabled	false	当前配置设置为“false”时，DLI在覆盖写之前，会删除所有符合条件的分区。例如，分区表中有一个“2021-01”的分区，当使用INSERT OVERWRITE语句向表中写入“2021-02”这个分区的数据时，会把“2021-01”的分区数据也覆盖掉。 当前配置设置为“true”时，DLI不会提前删除分区，而是在运行时覆盖那些有数据写入的分区。
spark.sql.files.maxPartitionBytes	134217728	读取文件时要打包到单个分区中的最大字节数。
spark.sql.badRecordsPath	-	Bad Records的路径。
spark.sql.legacy.correlated.scalar.query.enabled	false	• **该参数设置为true：** – 当子查询中数据不重复的情况下，执行关联子查询，不需要对子查询的结果去重。 – 当子查询中数据重复的情况下，执行关联子查询，会提示异常，必须对子查询的结果做去重处理，比如max(),min()。 • **该参数设置为false：** 不管子查询中数据重复与否，执行关联子查询时，都需要对子查询的结果去重，比如max(),min()，否则提示异常。

 版权所有 © 华为技术有限公司

名称	默认值	描述
spark.sql.keep.distinct.expandThreshold	-	• **参数说明：** 对于包含count(distinct)的多维分析（ with cube ）的查询场景，spark典型的执行计划是将cube使用expand算子来实现，但该操作会导致查询膨胀，为了避免出现查询膨胀，建议执行如下配置： – spark.sql.keep.distinct.expandThreshold：默认值：-1，即使用Spark默认的expand算子。 设置具体数值：即代表定义了查询膨胀的阈值（例如512），超过该阈值count(distinct)使用distinct聚合算子来执行，不再使用expand算子。 – spark.sql.distinct.aggregator.enabled：强制使用distinct聚合算子的开关。配置为true时不再根据spark.sql.keep.distinct.expandThreshold来判断。 • **适用场景：**包含count(distinct)的多维分析（ with cube ）的查询场景，可能包含多个count(distinct)，且包含cube/roll up • **典型场景示例：** `SELECT a1, a2, count(distinct b), count(distinct c) FROM test_distinct group by a1, a2 with cube`
spark.sql.distinct.aggregator.enabled	false	
dli.jobs.sql.resubmit.enable	null	通过设置该参数可以控制在driver故障、队列重启时Spark SQL作业是否重新提交。 • false：禁用作业重试，所有类型的命令都不重新提交，一旦driver故障，作业将标记为失败（ FAILED ）。 • true：启用作业重试，即在driver故障时，所有类型的作业都将重新提交。 **注意** 如果配置为true，在执行INSERT等幂等类型的操作时（例如insert into，load data、update），可能会导致数据一致性问题。即driver故障后作业重试，导致driver故障前已插入的数据被重复写入。

版权所有 © 华为技术有限公司

名称	默认值	描述
spark.sql.dli.job.shareLevel	Queue	该配置项用于设置SQL语句的隔离级别，不同的隔离级别（job, user, project, queue）将决定SQL作业是由独立的Spark Driver和Executor执行，还是共享已经存在的Spark Driver和Executor。 • **job**: – 每个SQL作业都会独立启动一个Spark Driver和一组Executor来执行。 – 适用于需要完全隔离的作业，确保每个作业的执行环境完全独立。 • **user:** – 如果已经有该用户启动的Spark Driver并且该Driver还能继续提交任务，那么新的SQL作业会提交到这个已存在的Driver上执行。 – 如果没有已存在的Driver或者现有Driver无法继续提交任务，则会为该用户新启动一个Spark Driver。 – 适用于同一用户的多个作业需要共享资源的场景。 • **project:** – 如果已经有该项目启动的Spark Driver并且该Driver还能继续提交任务，那么新的SQL作业会提交到这个已存在的Driver上执行。 – 如果没有已存在的Driver或者现有Driver无法继续提交任务，则会为该项目新启动一个Spark Driver。 – 适用于同一项目内的多个作业需要共享资源的场景。 • **queue:** – 如果已经有该队列启动的Spark Driver并且该Driver还能继续提交任务，那么新的SQL作业会提交到这个已存在的Driver上执行。 – 如果没有已存在的Driver或者现有Driver无法继续提交任务，则会为该队列新启动一个Spark Driver。 – 适用于按队列管理资源的场景，可以更细粒度地控制资源分配。 **说明** 上述所有隔离级别所能启动的最多spark driver数量(最大spark driver实例数)以及每个spark driver最多并发执行的SQL数量(单spark driver实例最大并发数)，可以在队列属性中进行设置。 **设置队列属性**

版权所有 © 华为技术有限公司

2 Spark SQL 语法概览

本章节介绍了目前DLI所提供的Spark SQL语法列表。参数说明，示例等详细信息请参考具体的语法说明。

表 2-1 批作业 SQL 语法

语法分类	操作链接
数据库相关语法	创建数据库
	删除数据库
	查看指定数据库
	查看所有数据库
创建OBS表相关语法	使用DataSource语法创建OBS表
	使用Hive语法创建OBS表
创建DLI表相关语法	使用DataSource语法创建DLI表
	使用Hive语法创建DLI表
删除表相关语法	删除表
查看表相关语法	查看所有表
	查看建表语句
	查看表属性
	查看指定表所有列
	查看指定表所有分区
	查看表统计信息
修改表相关语法	添加列
分区表相关语法	添加分区（只支持OBS表）
	重命名分区

版权所有 © 华为技术有限公司

语法分类	操作链接
	删除分区
	修改表分区位置（只支持OBS表）
	更新表分区信息（只支持OBS表）
导入数据相关语法	导入数据
插入数据相关语法	插入数据
清空数据相关语法	清空数据
导出查询结果相关语法	导出查询结果
跨源连接HBase表相关语法	创建表关联HBase
	插入数据至HBase表
	查询HBase表
跨源连接OpenTSDB表相关语法	创建表关联OpenTSDB
	插入数据至OpenTSDB
	查询OpenTSDB表
跨源连接DWS表相关语法	创建表关联DWS
	插入数据至DWS表
	查询DWS表
跨源连接RDS表相关语法	创建表关联RDS
	插入数据至RDS表
	查询RDS表
跨源连接CSS表相关语法	创建表关联CSS
	插入数据至CSS表
	查询CSS表
跨源连接DCS表相关语法	创建表关联DCS
	插入数据至DCS表
	查询DCS表
跨源连接DDS表相关语法	创建表关联DDS
	插入数据至DDS表
	查询DDS表
视图相关语法	创建视图
	删除视图

版权所有 © 华为技术有限公司

语法分类	操作链接
查看计划相关语法	查看计划
数据权限相关语法	创建角色
	删除角色
	绑定角色
	解绑角色
	显示角色
	分配权限
	回收权限
	显示已授权限
	显示所有角色和用户的绑定关系
自定义函数相关语法	创建函数
	删除函数
	显示函数详情
	显示所有函数
数据多版本相关语法（废弃，不推荐使用）	创建OBS表时开启数据多版本 修改表时开启或关闭数据多版本 设置多版本备份数据保留周期 查看多版本备份数据 恢复多版本备份数据 配置多版本过期数据回收站 清理多版本数据

版权所有 © 华为技术有限公司

3 Spark 开源命令支持说明

本章节介绍了目前DLI对开源的Spark SQL语法的支持情况。详细的语法、参数说明，示例等信息请参考**Spark官方文档**。

表 3-1 DLI Spark 开源命令支持说明

功能描述	语法示例	DLI Spark 2.4.5	DLI Spark 3.3.1
创建数据库	CREATE DATABASE testDB;	支持	支持
创建DLI表	create table if not exists testDB.testTable1(id int, age int, money double);	支持	支持
创建OBS表	create table if not exists testDB.testTable2(id int, age int, money double) LOCATION 'obs://bucketName/filePath' partitioned by (dt string) row format delimited fields terminated by ',' STORED as TEXTFILE;	支持	支持
插入测试数据	insert into table testDB.testTable2 values (1, 18, 3.14, "20240101"), (2, 18, 3.15, "20240102");	支持	支持

版权所有 © 华为技术有限公司

功能描述	语法示例	DLI Spark 2.4.5	DLI Spark 3.3.1
修改数据库属性	ALTER DATABASE testDB SET DBPROPERTIES ('Edited-by' = 'John');	不支持	不支持
修改数据库在OBS上的文件存放路径	ALTER DATABASE testDB SET LOCATION 'obs://bucketName/filePath';	不支持	不支持
修改表名	ALTER TABLE testDB.testTable1 RENAME TO testDB.testTable101;	支持	支持
修改表的分区名	ALTER TABLE testDB.testTable2 PARTITION (dt='20240101') RENAME TO PARTITION (dt='20240103'); 只支持OBS表的分区名，且OBS上的文件存储路径不会变。	支持	支持
添加列	ALTER TABLE testDB.testTable1 ADD COLUMNS (name string);	支持	支持
删除列	ALTER TABLE testDB.testTable1 DROP columns (money);	不支持	不支持
修改列名称	ALTER TABLE testDB.testTable1 RENAME COLUMN age TO years_of_age;	不支持	不支持
修改列注释	ALTER TABLE testDB.testTable1 ALTER COLUMN age COMMENT "new comment";	不支持	支持
替换指定列	ALTER TABLE testDB.testTable1 REPLACE COLUMNS (name string, ID int COMMENT 'new comment');	不支持	不支持
自定义表属性	ALTER TABLE testDB.testTable1 SET TBLPROPERTIES ('aaa' = 'bbb');	支持	支持
添加或修改表注释	ALTER TABLE testDB.testTable1 SET TBLPROPERTIES ('comment' = 'test');	支持	不支持
修改表的存储格式	ALTER TABLE testDB.testTable1 SET fileformat csv;	不支持	不支持
删除表属性	ALTER TABLE testDB.testTable1 UNSET TBLPROPERTIES ('test');	支持	支持
删除表的注释	ALTER TABLE testDB.testTable1 UNSET TBLPROPERTIES ('comment');	支持	支持

版权所有 © 华为技术有限公司

功能描述	语法示例	DLI Spark 2.4.5	DLI Spark 3.3.1
展示列信息	DESCRIBE database_name.table_name col_name; 示例：DESCRIBE testDB.testTable1 id;	支持	支持
展示列信息	DESCRIBE table_name table_name.col_name; 示例：DESCRIBE testTable1 testTable1.id; 仅支持查看当前数据库下表的列信息。	支持	支持
返回查询语句的元数据信息	DESCRIBE QUERY SELECT age, sum(age) FROM testDB.testTable1 GROUP BY age;	不支持	支持
返回插入数据的元数据信息	DESCRIBE QUERY VALUES(100, 10, 10000.20D) AS testTable1(id, age, money);	不支持	支持
返回表的元数据信息	DESCRIBE QUERY TABLE testTable1;	不支持	支持
返回表中某列的元数据信息	DESCRIBE FROM testTable1 SELECT age;	不支持	支持
返回表的建表语句	SHOW CREATE TABLE testDB.testTable1 AS SERDE; Spark 3.3.1 执行该语句时，仅适用于查询Hive表的建表语句。	不支持	支持
返回表的建表语句	SHOW CREATE TABLE testDB.testTable1;	支持	支持

版权所有 © 华为技术有限公司

4 数据库相关

4.1 创建数据库

功能描述

创建数据库。

语法格式

```
CREATE [DATABASE | SCHEMA] [IF NOT EXISTS] db_name
  [COMMENT db_comment]
  [WITH DBPROPERTIES (property_name=property_value, ...)];
```

关键字

- IF NOT EXISTS：所需创建的数据库已存在时使用，可避免系统报错。
- COMMENT：对数据库的描述。
- DBPROPERTIES：数据库的属性，且属性名和属性值成对出现。

参数说明

表 4-1 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
db_comment	数据库描述。
property_name	数据库属性名。
property_value	数据库属性值。

 版权所有 © 华为技术有限公司

注意事项

- DATABASE与SCHEMA两者没有区别，可替换使用，建议使用DATABASE。
- “default”为内置数据库，不能创建名为“default”的数据库。

示例

说明

完整的SQL作业提交流程您可以参考《快速入门》中的《**提交SQL作业**》等章节描述。

1. 队列是使用DLI服务的基础，执行SQL前需要先创建队列。具体可以参考《用户指南》中的“**创建队列**”章节。
2. 在DLI管理控制台，单击左侧导航栏中的“SQL编辑器”，可进入SQL作业“SQL编辑器”页面。
3. 在“SQL编辑器”页面右侧的编辑窗口中，输入如下创建数据库的SQL语句，单击“执行”。阅读并同意隐私协议，单击“确定”。

   若testdb数据库不存在，则创建数据库testdb。

   ```
   CREATE DATABASE IF NOT EXISTS testdb;
   ```

4.2 删除数据库

功能描述

删除数据库。

语法格式

```
DROP [DATABASE | SCHEMA] [IF EXISTS] db_name [RESTRICT|CASCADE];
```

关键字

IF EXISTS：所需删除的数据库不存在时使用，可避免系统报错。

注意事项

- DATABASE与SCHEMA两者没有区别，可替换使用，建议使用DATABASE。
- RESTRICT表示如果该database不为空（有表存在），DROP操作会报错，执行失败，RESTRICT是默认逻辑。
- CASCADE表示即使该database不为空（有表存在），DROP也会级联删除下面的所有表，需要谨慎使用该功能。

参数说明

表 4-2 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。

版权所有 © 华为技术有限公司

示例

1. 已参考**示例**中描述创建对应的数据库，如testdb。
2. 若存在testdb数据库，则删除数据库testdb。
   ```
   DROP DATABASE IF EXISTS testdb;
   ```

4.3 查看指定数据库

功能描述

查看指定数据库的相关信息，包括数据库名称、数据库的描述等。

语法格式

```
DESCRIBE DATABASE [EXTENDED] db_name;
```

关键字

EXTENDED：除了显示上述信息外，还会额外显示数据库的属性信息。

参数说明

表 4-3 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。

注意事项

如果所要查看的数据库不存在，则系统报错。

示例

1. 已参考**示例**中描述创建对应的数据库，如testdb。
2. 查看testdb数据库的相关信息。
   ```
   DESCRIBE DATABASE testdb;
   ```

4.4 查看所有数据库

功能描述

查看当前工程下所有的数据库。

语法格式

```
SHOW [DATABASES | SCHEMAS] [LIKE regex_expression];
```

 版权所有 © 华为技术有限公司

关键字

无。

参数说明

表 4-4 参数说明

参数	描述
regex_expression	数据库名称。

注意事项

DATABASES与SCHEMAS是等效的，都将返回所有的数据库名称。

示例

查看当前的所有数据库。

```
SHOW DATABASES;
```

查看当前的所有以test开头的数据库。

```
SHOW DATABASES LIKE "test.*";
```


版权所有 © 华为技术有限公司

5 表相关

5.1 创建 OBS 表

5.1.1 使用 DataSource 语法创建 OBS 表

功能描述

本节介绍使用DataSource语法创建OBS表。

DataSource语法和Hive语法主要区别在于支持的表数据存储格式范围、支持的分区数等有差异，详细请参考语法格式和注意事项说明。

说明

推荐使用OBS并行文件系统进行存储。并行文件系统是一种高性能文件系统，提供毫秒级别访问时延，TB/s级别带宽和百万级别的IOPS，适用于大数据交互式分析场景。

注意事项

- 创建表时不会统计大小。
- 添加数据时会修改大小至0。
- 如需查看表大小可以通过OBS查看。
- CTAS建表语句不能指定表的属性。
- **OBS目录下包含子目录的场景：**

  创建表时，若指定路径为OBS上的目录，且该目录下包含子目录（或嵌套子目录），则子目录下的所有文件类型及其内容也是表内容。

  您需要保证所指定的目录及其子目录下所有文件类型和建表语句中指定的存储格式一致，所有文件内容和表中的字段一致，否则查询将报错。

  您可以在建表语句OPTIONS中设置“multiLevelDirEnable”为true以查询子目录下的内容，此参数默认值为false（注意，此配置项为表属性，请谨慎配置。Hive表不支持此配置项)。
- **关于分区表的使用说明：**

 版权所有 © 华为技术有限公司

- 创建分区表时，PARTITIONED BY中指定分区列必须是表中的列，且必须在Column列表中指定类型。分区列只支持string, boolean, tinyint, smallint, short, int, bigint, long, decimal, float, double, date, timestamp类型。
- 创建分区表时，分区字段必须是表字段的最后一个字段或几个字段，且多分区字段的顺序也必须对应。否则将出错。
- 单表分区数最多允许200000个。
- 2024年1月后新注册使用DLI服务的用户，且使用Spark3.3及以上版本的引擎，在使用DataSource语法创建表时支持使用CTAS创建分区表。

语法格式

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name1 col_type1 [COMMENT col_comment1], ...)]
  USING file_format
  [OPTIONS (path 'obs_path', key1=val1, key2=val2, ...)]
  [PARTITIONED BY (col_name1, col_name2, ...)]
  [COMMENT table_comment]
  [AS select_statement]
```

关键字

- IF NOT EXISTS：指定该关键字以避免表已经存在时报错。
- USING：指定存储格式。
- OPTIONS：指定建表时的属性名与属性值。
- COMMENT：字段或表描述。
- PARTITIONED BY：指定分区字段。
- AS：使用CTAS创建表。

参数说明

表 5-1 参数说明

参数	是否必选	描述
db_name	否	Database名称。 由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	是	Database中的待创建的表名。 由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。 特殊字符需要使用单引号（ '' ）包围起来。 表名对大小写不敏感，即不区分大小写。
col_name	是	以逗号分隔的带数据类型的列名。 列名由字母、数字和下划线（ _ ）组成。不能是纯数字，且至少包含一个字母。 列名为大小写不敏感，即不区分大小写。

 版权所有 © 华为技术有限公司

参数	是否必选	描述
col_type	是	列字段的数据类型。数据类型为原生类型。 请参考**原生数据类型**。
col_comment	否	列字段描述。仅支持字符串常量。
file_format	是	file_format是用于创建表的输入格式。支持orc，parquet，json，csv，avro类型。
path	是	数据文件所在的OBS存储路径，推荐使用OBS并行文件系统存储。 格式：obs://bucketName/tblPath bucketName即桶名称。 tblPath是目录名称。目录后不需要指定文件名。 更多建表时的属性名与属性值请参考**表5-2**。 file_format为csv时表的属性名与属性值请参考**表5-2**和**表5-3**。 当OBS的目录下文件夹与文件同名时，创建OBS表指向的路径会优先指向文件而非文件夹。
table_comment	否	表描述信息。仅支持字符串常量。
select_statement	否	用于CTAS命令，将源表的select查询结果或某条数据插入到新创建的OBS表中。

表 5-2 OPTIONS 参数描述

参数	是否必选	描述
path	否	指定的表路径，即OBS存储路径。
multiLevelDirEnable	否	嵌套子目录场景下，是否迭代查询子目录中的数据。当配置为true时，查询该表时会迭代读取该表路径中所有文件，包含子目录中的文件。 默认值：false
dataDelegated	否	是否需要在删除表或分区时，清除path路径下的数据。 默认值：false
compression	否	指定压缩格式。一般为parquet格式时指定该参数，推荐使用'zstd'压缩格式。

当file_format为csv时，支持设置以下OPTIONS参数。

版权所有 © 华为技术有限公司

表 5-3 CSV 数据格式 OPTIONS 参数说明

参数	是否必选	描述
delimiter	否	数据分隔符。 默认值：逗号（即",”）
quote	否	引用字符。 默认值：双引号（即“"”）
escape	否	转义字符。 默认值：反斜杠（即“\”）
multiLine	否	列数据中是否包含回车符或转行符，true为包含，false为不包含。 默认值：false
dateFormat	否	指定CSV文件中date字段的日期格式。 默认值：yyyy-MM-dd
timestampFormat	否	指定CSV文件中timestamp字段的日期格式。 默认值： yyyy-MM-dd HH:mm:ss
mode	否	指定解析CSV时的模式，有三种模式。默认值：PERMISSIVE ● PERMISSIVE：宽容模式，遇到错误的字段时，设置该字段为Null ● DROPMALFORMED: 遇到错误的字段时，丢弃整行。 ● FAILFAST：报错模式，遇到错误的字段时直接报错。
header	否	CSV是否包含表头信息，true表示包含表头信息，false为不包含。 默认值：false
nullValue	否	设置代表null的字符，例如，nullValue="nl"表示设置nl代表null。
comment	否	设置代表注释开头的字符，例如，comment='#'表示以#开头的行为注释。
compression	否	设置数据的压缩格式。目前支持gzip、bzip2、deflate压缩格式，若不希望压缩，则输入none。 默认值：none
encoding	否	数据的编码格式。支持utf-8，gb2312，gbk三种，如果不填写，则默认为utf-8。 默认值：utf-8

版权所有 © 华为技术有限公司

示例 1：创建 OBS 非分区表

示例说明：创建名为table1的OBS非分区表，使用USING关键字指定该表的存储格式为orc格式。

在您的实际使用中，可以将obs表存储为parquet、json、avro等类型。

```
CREATE TABLE IF NOT EXISTS table1 (
   col_1   STRING,
   col_2   INT)
USING orc
OPTIONS (path 'obs://bucketName/filePath');
```

示例 2：创建 OBS 分区表

示例说明：创建一个名为student的分区表，该分区表使用院系编号（facultyNo）和班级编号（classNo）进行分区。该student表会同时按照不同的院系编号（facultyNo）和不同的班级编号（classNo）分区。

在实际的使用过程中，您可以选择合适的分区字段并将其添加到PARTITIONED BY关键字后的括号内。

```
CREATE TABLE IF NOT EXISTS student (
   Name        STRING,
   facultyNo   INT,
   classNo     INT)
USING csv
OPTIONS (path 'obs://bucketName/filePath')
PARTITIONED BY (facultyNo, classNo);
```

示例 3：使用 CTAS 将源表的全部数据或部分数据创建新的 OBS 非分区表

示例说明：根据**示例1：创建OBS非分区表**中创建的OBS表table1，使用CTAS语法将table1中的数据复制到table1_ctas表中。

在使用CTAS建表的时候，可以忽略被复制的表在建表时所使用的语法，即不论在创建table1时使用的是何种语法，都可以使用DataSource语法的CTAS创建table1_ctas。

此外，本例中table1中OBS表的存储格式为orc，而table1_ctas表的存储格式可以为parquet，即CTAS创建的表存储格式可以不同于原表。

在AS关键字后使用SELECT语句选择需要的数据插入到table1_ctas表中。

SELECT语法为：SELECT <列名称> FROM <表名称> WHERE <相关筛选条件>。

- 示例中使用“SELECT * FROM table1”，'*'表示会从table1中选择所有列，并将table1中所有数据插入到table1_ctas表中。
  ```
  CREATE TABLE IF NOT EXISTS table1_ctas
  USING parquet
  OPTIONS (path 'obs:// bucketName/filePath')
  AS
  SELECT  *
  FROM    table1;
  ```
- 若需要按照自定义方式筛选数据插入table1_ctas中，可以使用如下的SELECT语句“SELECT col_1 FROM table1 WHERE col_1 = 'Ann' ”，这样就可以通过执行SELECT语句从table1中单独选定col_1，并只将其中值等于'Ann'的数据插入到table1_ctas中。
  ```
  CREATE TABLE IF NOT EXISTS table1_ctas
  USING parquet
  OPTIONS (path 'obs:// bucketName/filePath')
  AS
  SELECT  col_1
  ```

 版权所有 © 华为技术有限公司

```
FROM    table1
WHERE   col_1 = 'Ann';
```

示例 4：创建 OBS 非分区表，并自定义列字段数据类型

示例说明：创建名为table2的OBS非分区表，您可以根据业务需求自定义列字段的原生数据类型：

- 与文字字符有关可以使用STRING、CHAR或者VARCHAR。
- 与时间有关的可以使用TIMESTAMP、DATE。
- 与整数有关的可以使用INT、SMALLINT/SHORT、BIGINT/LONG、TINYINT。
- 涉及小数运算可以使用FLOAT、DOUBLE、DECIMAL。
- 若数据只涉及逻辑开关可以使用BOOLEAN类型。

具体使用方法与明细可以参照“数据类型 > 原生数据类型”。

请参考**原生数据类型**。

```
CREATE TABLE IF NOT EXISTS table2 (
   col_01  STRING,
   col_02  CHAR (2),
   col_03  VARCHAR (32),
   col_04  TIMESTAMP,
   col_05  DATE,
   col_06  INT,
   col_07  SMALLINT,
   col_08  BIGINT,
   col_09  TINYINT,
   col_10  FLOAT,
   col_11  DOUBLE,
   col_12  DECIMAL (10, 3),
   col_13  BOOLEAN
)
USING parquet
OPTIONS (path 'obs://bucketName/filePath');
```

示例 5：创建 OBS 分区表，自定义表的 OPTIONS 参数

示例说明：创建OBS表时支持自定义属性名与属性值，OPTIONS参数说明可参考**表5-2**。

本例创建名为table3并以col_2为分区依据的OBS分区表。在OPTIONS中配置path、multiLevelDirEnable、dataDelegated和compression。

- path：OBS存储路径，本例为“obs ://bucketName/filePath”，其中的bucketName为您存储时所使用桶名称，filePath为您实际使用的目录名称；
- 请注意大数据场景建议使用OBS并行文件系统进行存储；
- multiLevelDirEnable：本例设置为true，表示查询该表时会迭代读取表路径中的所有文件和子目录文件，若不需要此项配置可以设置为false或不设置（默认为false）；
- dataDelegated：本例设置为true，表示在删除表或相关分区时，会一并清除该path路径下的所有数据，若不需要此项配置可以设置为false或不设置（默认为false）；
- compression：当创建的OBS表需要压缩时，可以使用compression关键字来配置压缩格式，本例中就使用了zstd压缩格式。

```
CREATE TABLE IF NOT EXISTS table3 (
   col_1   STRING,
```

 版权所有 © 华为技术有限公司

```
    col_2   int
)
USING parquet
PARTITIONED BY (col_2)
OPTIONS (
    path 'obs://bucketName/filePath',
    multiLeveldirenable = true,
    datadelegated = true,
    compression = 'zstd'
);
```

示例 6：创建 OBS 非分区表，自定义表的 OPTIONS 参数

示例说明：CSV表是一种以逗号分隔的纯文本文件格式，用于存储和交换数据。它通常用于简单的数据交换，但是它没有结构化数据的概念，因此不适合存储复杂数据类型。于是当file_format为csv时，支持配置更多的OPTIONS参数（参考**表5-3**）。

本例创建一个名为table4且存储格式为csv非分区表并使用了额外的OPTIONS参数对数据加以约束。

- delimiter：数据分隔符，表示使用逗号（，）作为数据之间的分隔符；
- quote：引用字符，表示使用双引号（”）来表示数据中的引用信息；
- escape：转义字符，表示使用反斜杠（\）作为数据存储时的转义字符；
- multiLine：设置需要存储的列数据中不包含回车符或者换行符；
- dataFormat：表示该csv文件中data字段的指定日期格式为yyyy-MM-dd；
- timestamoFormat：表示该csv文件中会将时间戳格式指定为yyyy-MM-dd HH:mm:ss；
- header：表示该csv表中包含表头信息；
- nullValue：表示设置null来表示csv表中的null值；
- comment：表示该csv表使用斜杠（/）表示注释的开头；
- compression：表示该csv表被压缩，此处csv表支持gzip、bzip2和deflate的压缩格式，若不需要压缩，也可以设置为none；
- encoding：表示该表使用utf-8的数据编码格式，在实际使用中，可以根据您的需求选择utf-8、gb2312和gbk中任一种编码格式，其中默认编码格式为utf-8。

```
CREATE TABLE IF NOT EXISTS table4 (
    col_1 STRING,
    col_2 INT
)
USING csv
OPTIONS (
    path 'obs://bucketName/filePath',
    delimiter       = ',',
    quote           = '#',
    escape          = '|',
    multiline       = false,
    dateFormat      = 'yyyy-MM-dd',
    timestampFormat  = 'yyyy-MM-dd HH:mm:ss',
    mode            = 'failfast',
    header          = true,
    nullValue       = 'null',
    comment         = '*',
    compression     = 'deflate',
    encoding        = 'utf - 8'
);
```

 版权所有 © 华为技术有限公司

常见问题

- **使用default创建DataSource表时提示“xxx dli datasource v2 tables is only supported in spark3.3 or later version.”怎么办?**

  使用default创建DataSource表时引擎版本不低于Spark 3.3.1，如果引擎版本低于Spark 3.3.1则会提示上述错误信息，此时请切换使用Hive语法创建表。详细操作请参考**使用Hive语法创建OBS表**。

- **使用Spark 3.3.1执行jar作业报错 "xxx don`t support dli v1 table."怎么办?**

  该错误提示信息说明使用Spark 3.3.1执行Jar作业时不支持执行与该表相关的操作，请切换使用Hive语法重构表数据结构，例如使用Hive语法[STORED AS file_format] CTAS重新创建表后再执行作业。详细操作请参考**使用Hive语法创建OBS表**。

5.1.2 使用 Hive 语法创建 OBS 表

功能描述

使用Hive语法创建OBS表。DataSource语法和Hive语法主要区别在于支持的表数据存储格式范围、支持的分区数等有差异，详细请参考语法格式和注意事项说明。

说明

推荐使用OBS并行文件系统进行存储。并行文件系统是一种高性能文件系统，提供毫秒级别访问时延，TB/s级别带宽和百万级别的IOPS，适用于大数据交互式分析场景。

注意事项

- 创建表时会统计大小。
- 添加数据时不会修改大小。
- 如需查看表大小可以通过OBS查看。
- CTAS建表语句不能指定表的属性。
- **关于分区表的使用说明：**
  - 创建分区表时，PARTITONED BY中指定分区列必须是不在表中的列，且需要指定数据类型。分区列支持string, boolean, tinyint, smallint, short, int, bigint, long, decimal, float, double, date, timestamp等hive开源支持的类型。
  - 支持指定多个分区字段，分区字段只需在PARTITIONED BY关键字后指定，不能像普通字段一样在表名后指定，否则将出错。
  - 单表分区数最多允许200000个。
  - Spark 3.3及以上版本支持使用Hive语法的CTAS语句创建分区表。
- **关于创建表时设置多字符的分隔符：**
  - 只有指定ROW FORMAT SERDE为org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe时，字段分隔符才支持设置为多字符。
  - 只有Hive OBS表支持在建表时指定多字符的分隔符，Hive DLI表不支持在建表时指定多字符的分隔符。
  - 指定了多字符分隔的表不支持INSERT、IMPORT等写数语句。如需添加数据，请将数据文件直接放到表对应的OBS路径下即可，例如**示例7：创建表并设置多字符的分隔符**中，将数据文件放到obs://bucketName/filePath下。

 版权所有 © 华为技术有限公司

语法格式

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name1 col_type1 [COMMENT col_comment1], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name2 col_type2, [COMMENT col_comment2], ...)]
  [ROW FORMAT row_format]
  [STORED AS file_format]
  LOCATION 'obs_path'
  [TBLPROPERTIES (key = value)]
  [AS select_statement]
row_format:
  : SERDE serde_cls [WITH SERDEPROPERTIES (key1=val1, key2=val2, ...)]
  | DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]]
      [COLLECTION ITEMS TERMINATED BY char]
      [MAP KEYS TERMINATED BY char]
      [LINES TERMINATED BY char]
      [NULL DEFINED AS char]
```

关键字

- EXTERNAL：指创建OBS表。
- IF NOT EXISTS：指定该关键字以避免表已经存在时报错。
- COMMENT：字段或表描述。
- PARTITIONED BY：指定分区字段。
- ROW FORMAT：行数据格式。
- STORED AS：指定所存储的文件格式，当前该关键字只支持指定TEXTFILE, AVRO, ORC, SEQUENCEFILE, RCFILE, PARQUET格式。
- LOCATION：指定OBS的路径。创建OBS表时必须指定此关键字。
- TBLPROPERTIES：TBLPROPERTIES子句允许用户给表添加key/value的属性。
  - （多版本功能已废弃，不推荐使用）开启数据多版本功能，用于表数据的备份与恢复。开启多版本功能后，在进行删除或修改表数据时（insert overwrite或者truncate操作），系统会自动备份历史表数据并保留一定时间，后续您可以对保留周期内的数据进行快速恢复，避免因误操作而丢失数据。多版本功能SQL语法请参考**开启或关闭数据多版本（废弃，不推荐使用）**和**多版本备份恢复数据（废弃，不推荐使用）**。

    创建OBS表时，通过指定**TBLPROPERTIES ("dli.multi.version.enable"="true")**开启DLI数据多版本功能，具体可以参考示例说明。

    表 5-4 TBLPROPERTIES 主要参数说明

key值	value说明
dli.multi.version.enable	• true：开启DLI数据多版本功能。 • false：关闭DLI数据多版本功能。
comment	表描述信息。

 版权所有 © 华为技术有限公司

key值	value说明
orc.compress	orc存储格式表的一个属性，用来指定orc存储的压缩方式。支持取值为： • ZLIB • SNAPPY • NONE
auto.purge	当设置为true时，删除或者覆盖的数据会不经过回收站，直接被删除。

- AS：使用CTAS创建表。
- ROW FORMAT SERDE为org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe时，字段分隔符才支持设置为多字符。使用方法参考**示例7：创建表并设置多字符的分隔符**。

参数说明

表 5-5 参数说明

参数	是否必选	描述
db_name	否	Database名称。 由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	是	Database中的表名。 由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。 特殊字符需要使用单引号（''）包围起来。 表名对大小写不敏感，即不区分大小写。
col_name	是	列字段名称。 列字段由字母、数字和下划线（_）组成。不能是纯数字，且至少包含一个字母。 列名为大小写不敏感，即不区分大小写。
col_type	是	列字段的数据类型。数据类型为原生类型。 请参考**原生数据类型**。
col_comment	否	列字段描述。仅支持字符串常量。
row_format	是	行数据格式。row_format功能只支持textfile类型的表。
file_format	是	OBS表存储格式，支持TEXTFILE, AVRO, ORC, SEQUENCEFILE, RCFILE, PARQUET

版权所有 © 华为技术有限公司

参数	是否必选	描述
table_comment	否	表描述。仅支持字符串常量。
obs_path	是	数据文件所在的OBS存储路径，推荐使用OBS并行文件系统存储。 格式：obs://bucketName/tblPath bucketName即桶名称。 tblPath是目录名称。目录后不需要指定文件名。 当OBS的目录下文件夹与文件同名时，创建OBS表指向的路径会优先指向文件而非文件夹。
key = value	否	设置TBLPROPERTIES具体属性和值。 例如开启DLI数据多版本时，可以设置"dli.multi.version.enable"="true"来开启该功能。
select_statement	否	用于CTAS命令，将源表的select查询结果或某条数据插入到新创建的OBS表中。

示例 1：创建 OBS 非分区表

示例说明：创建名为table1的OBS非分区表，并用STORED AS关键字指定该表的存储格式为orc格式。

在您的实际使用中，可以将OBS表存储为textfile, avro, orc, sequencefile, rcfile, parquet等类型。

```
CREATE TABLE IF NOT EXISTS table1 (
   col_1   STRING,
   col_2   INT
)
STORED AS orc
LOCATION 'obs://bucketName/filePath';
```

示例 2：创建 OBS 分区表

示例说明：创建一个名为student的分区表，该分区表使用院系编号（facultyNo）和班级编号（classNo）进行分区，该student表会同时按照不同的院系编号（facultyNo）和不同的班级编号（classNo）分区。

在实际的使用过程中，您可以选择合适的分区字段并将其添加到PARTITIONED BY关键字后。

```
CREATE TABLE IF NOT EXISTS student(
   id      INT,
   name    STRING
)
STORED AS avro
LOCATION 'obs://bucketName/filePath'
PARTITIONED BY (
   facultyNo   INT,
   classNo     INT
);
```

 版权所有 © 华为技术有限公司

示例 3：使用 CTAS 语句将源表的全部数据或部分数据创建新的 OBS 表

示例说明：根据**示例1：创建OBS非分区表**中创建的OBS表table1，使用CTAS语法将table1中的数据复制到table1_ctas表中。

在使用CTAS建表的时候，可以忽略被复制的表在建表时所使用的语法，即不论在创建table1时使用的是何种语法，都可以使用DataSource语法的CTAS创建table1_ctas。

此外，本例中table1中OBS表的存储格式为orc，而table1_ctas表的存储格式可以为sequencefile或者parquet，即CTAS创建的表存储格式可以不同于原表。

在AS关键字后使用SELECT语句选择需要的数据插入到table1_ctas表中。

SELECT语法为：SELECT <列名称> FROM <表名称> WHERE <相关筛选条件>。

- 示例中使用“SELECT * FROM table1”，'*'表示会从table1中选择所有列，并将table1中所有数据插入到table1_ctas表中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
STORED AS sequencefile
LOCATION 'obs://bucketName/filePath'
AS
SELECT  *
FROM    table1;
```

- 若需要按照自定义方式筛选数据插入table1_ctas中，可以使用如下的SELECT语句“SELECT col_1 FROM table1 WHERE col_1 = 'Ann' ”，这样就可以通过执行SELECT语句从table1中单独选定col_1，并只将其中值等于'Ann'的数据插入到table1_ctas中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
STORED AS parquet
LOCATION  'obs:// bucketName/filePath'
AS
SELECT  col_1
FROM    table1
WHERE   col_1 = 'Ann';
```

示例 4：创建 OBS 非分区表，并自定义列字段数据类型

示例说明：创建名为table2的OBS非分区表，您可以根据业务需求自定义列字段的原生数据类型：

- 与文字字符有关可以使用STRING、CHAR或者VARCHAR。
- 与时间有关的可以使用TIMESTAMP、DATE。
- 与整数有关的可以使用INT、SMALLINT/SHORT、BIGINT/LONG、TINYINT。
- 涉及小数运算可以使用FLOAT、DOUBLE、DECIMAL。
- 若数据只涉及逻辑开关可以使用BOOLEAN类型。

具体使用方法与明细可以参照“数据类型 >原生数据类型”。

请参考**原生数据类型**。

```
CREATE TABLE IF NOT EXISTS table2 (
   col_01  STRING,
   col_02  CHAR (2),
   col_03  VARCHAR (32),
   col_04  TIMESTAMP,
   col_05  DATE,
   col_06  INT,
   col_07  SMALLINT,
   col_08  BIGINT,
   col_09  TINYINT,
```

 版权所有 © 华为技术有限公司

```
  col_10  FLOAT,
  col_11  DOUBLE,
  col_12  DECIMAL (10, 3),
  col_13  BOOLEAN
)
STORED AS parquet
LOCATION 'obs://bucketName/filePath';
```

示例 5：创建 OBS 分区表，自定义表的 TBLPROPERTIES 参数

示例说明：创建名为table3，并以col_3为分区依据的OBS分区表。在TBLPROPERTIES中配置dli.multi.version.enable、comment、orc.compress和auto.purge。

- dli.multi.version.enable：本例配置为true，即代表开启DLI数据多版本功能，用于表数据的备份与恢复。
- comment：表描述信息，comment描述信息支持后续修改。
- orc.compress：指定orc存储的压缩方式，本例定义为ZLIB。
- auto.purge：本例配置为true，即删除或者覆盖的数据会不经过回收站，直接被删除。

```
CREATE TABLE IF NOT EXISTs table3 (
  col_1 STRING,
  col_2 STRING
)
PARTITIONED BY (col_3 DATE)
STORED AS rcfile
LOCATION 'obs://bucketName/filePath'
TBLPROPERTIES (
  dli.multi.version.enable  = true,
  comment                   = 'Created by dli',
  orc.compress              = 'ZLIB',
  auto.purge                = true
);
```

示例 6：创建 textfile 格式的非分区表，并设置 ROW FORMAT

示例说明：创建名为table4的textfile类型的非分区表，并设置ROW FORMAT（ROW FORMAT功能只支持textfile类型的表）。

- FIELDS：字段表格中的列，每个字段有一个名称和数据类型，表中字段之间以'/'分隔。
- COLLECTION ITEMS：集合项指的是一组数据中的元素，可以是数组、列表或集合等，表中集合项以'$'分隔。
- MAP KEYS：映射键是一种键值对的数据结构，用于存储一组相关联的数据，表中Map键以'#'分隔。
- LINES：表格中的行，每一行包含一组字段值，表中行以'\n'结束（注意，只支持用'\n'作为行分隔符）。
- NULL：表示缺少值或未知值的特殊值。在表格中，NULL表示该字段没有值或该值未知。如果数据中存在null值，则用字符串“null”表示。

```
CREATE TABLE IF NOT EXISTS table4 (
  col_1   STRING,
  col_2   INT
)
STORED AS textfile
LOCATION 'obs://bucketName/filePath'
ROW FORMAT
DELIMITED FIELDS TERMINATED   BY '/'
COLLECTION ITEMS TERMINATED   BY '$'
MAP KEYS TERMINATED           BY '#'
```

 版权所有 © 华为技术有限公司

```
LINES TERMINATED            BY '\n'
NULL DEFINED              AS 'null';
```

示例 7：创建表并设置多字符的分隔符

示例说明：创建了一个名为table5的Hive表。表指定序列化和反序列化类ROW FORMAT SERDE，字段之间的分隔符被设置为/#，并且数据以文本文件格式存储。

- 只有指定ROW FORMAT SERDE为org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe时，字段分隔符才支持设置为多字符。
- 只有Hive OBS表支持在建表时指定多字符的分隔符，Hive DLI表不支持在建表时指定多字符的分隔符。
- 指定了多字符分隔的表不支持INSERT、IMPORT等写数语句。如需添加数据，请将数据文件直接放到表对应的OBS路径下即可，本例中，将数据文件放到obs://bucketName/filePath下。
- 本例指定字段分隔符 field.delim'为“/#”。
- ROW FORMAT功能只支持textfile类型的表。

```
CREATE TABLE IF NOT EXISTS table5 (
   col_1   STRING,
   col_2   INT
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe'
WITH SERDEPROPERTIES (
  'field.delim' = '/#'
)
STORED AS textfile
LOCATION 'obs://bucketName/filePath';
```

5.2 创建 DLI 表

5.2.1 使用 DataSource 语法创建 DLI 表

功能描述

使用DataSource语法创建DLI表。DataSource语法和Hive语法主要区别在于支持的表数据存储格式范围、支持的分区数等有差异，详细请参考语法格式和注意事项说明。

注意事项

- CTAS建表语句不能指定表的属性。
- 若没有指定分隔符，则默认为逗号（,）。
- **关于分区表的使用说明：**
  - 创建分区表时，PARTITIONED BY中指定分区列必须是表中的列，且必须在Column列表中指定类型。分区列只支持string, boolean, tinyint, smallint, short, int, bigint, long, decimal, float, double, date, timestamp类型。
  - 创建分区表时，分区字段必须是表字段的最后一个字段或几个字段，且多分区字段的顺序也必须对应。否则将出错。
  - 单表分区数最多允许200000个。
  - 2024年1月后新注册使用DLI服务的用户，且使用Spark3.3及以上版本的引擎，在使用DataSource语法创建表时支持使用CTAS创建分区表。

 版权所有 © 华为技术有限公司

语法格式

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name1 col_type1 [COMMENT col_comment1], ...)]
  USING file_format
  [OPTIONS (key1=val1, key2=val2, ...)]
  [PARTITIONED BY (col_name1, col_name2, ...)]
  [COMMENT table_comment]
  [AS select_statement];
```

关键字

- IF NOT EXISTS：指定该关键字以避免表已经存在时报错。
- USING：指定存储格式。
- OPTIONS：指定建表时的属性名与属性值。
- COMMENT：字段或表描述。
- PARTITIONED BY：指定分区字段。
- AS：使用CTAS创建表。

参数说明

表 5-6 参数描述

参数	是否必选	描述
db_name	否	Database名称。 由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	是	Database中的表名。 由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。 特殊字符需要使用单引号（''）包围起来。 表名对大小写不敏感，即不区分大小写。
col_name	是	以逗号分隔的带数据类型的列名。 列名由字母、数字和下划线（_）组成。不能是纯数字，且至少包含一个字母。 列名为大小写不敏感，即不区分大小写。
col_type	是	列字段的数据类型。数据类型为原生类型。 请参考**原生数据类型**。
col_comment	否	列字段描述。仅支持字符串常量。
file_format	是	DLI表数据存储格式，支持：parquet和orc格式。
table_comment	否	表描述。仅支持字符串常量。

 版权所有 © 华为技术有限公司

参数	是否必选	描述
select_statement	否	用于CTAS命令，将源表的select查询结果或某条数据插入到新创建的DLI表中。

表 5-7 OPTIONS 参数描述

参数	是否必选	描述	默认值
multiLevelDirEnable	否	是否迭代查询子目录中的数据。当配置为true时，查询该表时会迭代读取该表路径中所有文件，包含子目录中的文件。	false
compression	否	指定压缩格式。一般为parquet格式时指定该参数，推荐使用'zstd'压缩格式。	-

示例 1：创建 DLI 非分区表

示例说明：创建名为table1的DLI非分区表，使用USING关键字指定该表的存储格式为orc格式。

在您的实际使用中，还可以将DLI表存储为parquet类型。

```
CREATE TABLE IF NOT EXISTS table1 (
  col_1 STRING,
  col_2 INT)
USING orc;
```

示例 2：创建 DLI 分区表

示例说明：创建一个名为student的分区表，该分区表使用院系编号（facultyNo）和班级编号（classNo）进行分区，该student表会同时按照不同的院系编号（facultyNo）和不同的班级编号（classNo）分区。

在实际的使用过程中，您可以选择合适的分区字段并将其添加到PARTITIONED BY关键字后。

```
CREATE TABLE IF NOT EXISTS student (
  Name       STRING,
  facultyNo  INT,
  classNo    INT
)
USING orc
PARTITIONED BY (facultyNo, classNo);
```

示例 3：使用 CTAS 将源表的全部数据或部分数据创建新的 DLI 表

示例说明：根据**示例1：创建DLI非分区表**中创建的DLI表table1，使用CTAS语法将table1中的数据复制到table1_ctas表中。

在使用CTAS建表的时候，可以忽略被复制的表在建表时所使用的语法，即不论在创建table1时使用的是何种语法，都可以使用DataSource语法的CTAS创建table1_ctas。

 版权所有 © 华为技术有限公司

此外，本例中table1DLI表的存储格式为orc，而table1_ctas表的存储格式可以为orc或者parquet，即CTAS创建的表存储格式可以不同于原表。

在AS关键字后使用SELECT语句选择需要的数据插入到table1_ctas表中。

SELECT语法为：SELECT <列名称> FROM <表名称> WHERE <相关筛选条件>。

- 示例中使用“SELECT * FROM table1”，'*'表示会从table1中选择所有列，并将table1中所有数据插入到table1_ctas表中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
USING parquet
AS
SELECT  *
FROM    table1;
```

- 若需要按照自定义方式筛选数据插入table1_ctas中，可以使用如下的SELECT语句“SELECT col_1 FROM table1 WHERE col_1 = 'Ann' ”，这样就可以通过执行SELECT语句从table1中单独选定col_1，并只将其中值等于'Ann'的数据插入到table1_ctas中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
USING parquet
AS
SELECT  col_1
FROM    table1
WHERE   col_1 = 'Ann';
```

示例 4：创建 DLI 非分区表，并自定义列字段数据类型

示例说明：创建名为table2的DLI非分区表，您可以根据业务需求自定义列子段的原生数据类型：

- 与文字字符有关可以使用STRING、CHAR或者VARCHAR。
- 与时间有关的可以使用TIMESTAMP、DATE。
- 与整数有关的可以使用INT、SMALLINT/SHORT、BIGINT/LONG、TINYINT。
- 涉及小数运算可以使用FLOAT、DOUBLE、DECIMAL。
- 若数据只涉及逻辑开关可以使用BOOLEAN类型。

具体使用方法与明细可以参照“数据类型 >原生数据类型”。

请参考**原生数据类型**。

```
CREATE TABLE IF NOT EXISTS table2 (
   col_01  STRING,
   col_02  CHAR (2),
   col_03  VARCHAR (32),
   col_04  TIMESTAMP,
   col_05  DATE,
   col_06  INT,
   col_07  SMALLINT,
   col_08  BIGINT,
   col_09  TINYINT,
   col_10  FLOAT,
   col_11  DOUBLE,
   col_12  DECIMAL (10, 3),
   col_13  BOOLEAN
)
USING parquet;
```

示例 5：创建 DLI 分区表，自定义表的 OPTIONS 参数

示例说明：创建DLI表时支持自定义属性名与属性值，OPTIONS参数说明可参考**表5-7**。

 版权所有 © 华为技术有限公司

本例创建名为table3并以col_2为分区依据的DLI分区表。在OPTIONS中配置pmultiLevelDirEnable和compression。

- multiLevelDirEnable：本例设置为true，表示查询该表时会迭代读取表路径中的所有文件和子目录文件，若不需要此项配置可以设置为false或不设置（默认为false）；
- compression：当创建的OBS表需要压缩时，可以使用compression关键字来配置压缩格式，本例中就使用了zstd压缩格式。

```
CREATE TABLE IF NOT EXISTs table3 (
   col_1   STRING,
   col_2   int
)
USING parquet
PARTITIONED BY (col_2)
OPTIONS (
   multiLeveldirenable    = true,
   compression            = 'zstd'
);
```

常见问题

- **使用default创建DataSource表时提示“xxx dli datasource v2 tables is only supported in spark3.3 or later version.”怎么办?**

  使用default创建DataSource表时引擎版本不低于Spark 3.3.1，如果引擎版本低于Spark 3.3.1则会提示上述错误信息，此时请切换使用Hive语法创建表。详细操作请参考**使用Hive语法创建DLI表**。

- **使用Spark 3.3.1执行jar作业报错 "xxx don`t support dli v1 table."怎么办?**

  该错误提示信息说明使用Spark 3.3.1执行Jar作业时不支持执行与该表相关的操作，请切换使用Hive语法重构表数据结构，例如使用Hive语法[STORED AS file_format] CTAS重新创建表后再执行作业。详细创建表操作请参考**使用Hive语法创建DLI表**。

5.2.2 使用 Hive 语法创建 DLI 表

功能描述

使用Hive语法创建DLI表。DataSource语法和Hive语法主要区别在于支持的表数据存储格式范围、支持的分区数等有差异，详细请参考语法格式和注意事项说明。

注意事项

- CTAS建表语句不能指定表的属性。
- Hive DLI表不支持在建表时指定多字符的分隔符。
- **关于分区表的使用说明：**
  - 创建分区表时，PARTITONED BY中指定分区列必须是不在表中的列，且需要指定数据类型。分区列支持string, boolean, tinyint, smallint, short, int, bigint, long, decimal, float, double, date, timestamp等hive开源支持的类型。
  - 支持指定多个分区字段，分区字段只需在PARTITIONED BY关键字后指定，不能像普通字段一样在表名后指定，否则将出错。
  - 单表分区数最多允许200000个。
  - Spark 3.3及以上版本支持使用Hive语法的CTAS语句创建分区表。

 版权所有 © 华为技术有限公司

语法格式

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name1 col_type1 [COMMENT col_comment1], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name2 col_type2, [COMMENT col_comment2], ...)]
  [ROW FORMAT row_format]
  STORED AS file_format
  [TBLPROPERTIES (key = value)]
  [AS select_statement];

row_format:
  : SERDE serde_cls [WITH SERDEPROPERTIES (key1=val1, key2=val2, ...)]
  | DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]]
      [COLLECTION ITEMS TERMINATED BY char]
      [MAP KEYS TERMINATED BY char]
      [LINES TERMINATED BY char]
      [NULL DEFINED AS char]
```

关键字

- IF NOT EXISTS：指定该关键字以避免表已经存在时报错。
- COMMENT：字段或表描述。
- PARTITIONED BY：指定分区字段。
- ROW FORMAT：行数据格式。
- STORED AS：指定所存储的文件格式，当前该关键字只支持指定TEXTFILE, AVRO, ORC, SEQUENCEFILE, RCFILE, PARQUET几种格式。创建DLI表时必须指定此关键字。
- TBLPROPERTIES：用于为表添加key/value的属性。
  - 在表存储格式为PARQUET时，可以通过指定TBLPROPERTIES(parquet.compression = 'zstd')来指定表压缩格式为zstd。
- AS：使用CTAS创建表。

参数说明

表 5-8 参数描述

参数	是否必选	描述
db_name	否	Database名称。 由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	是	Database中的表名。 由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。如果特殊字符需要使用单引号（ '' ）包围起来。
col_name	是	列字段名称。 列字段由字母、数字和下划线（ _ ）组成。不能是纯数字，且至少包含一个字母。 列名为大小写不敏感，即不区分大小写。

 版权所有 © 华为技术有限公司

参数	是否必选	描述
col_type	是	列字段的数据类型。数据类型为原生类型。 请参考**原生数据类型**。
col_comment	否	列字段描述。仅支持字符串常量。
row_format	是	行数据格式。row format功能只支持textfile类型的表。
file_format	是	DLI表数据存储格式：支持textfile, avro, orc, sequencefile, rcfile, parquet。
table_comment	否	表描述。仅支持字符串常量。
key = value	否	设置TBLPROPERTIES具体属性和值。 在表存储格式为PARQUET时，可以通过指定TBLPROPERTIES(parquet.compression = 'zstd')来指定表压缩格式为zstd。
select_statement	否	用于CTAS命令，将源表的select查询结果或某条数据插入到新创建的DLI表中。

示例 1：创建 DLI 非分区表

示例说明：创建名为table1的DLI非分区表，并用STORED AS关键字指定该表的存储格式为orc格式。

在您的实际使用中，可以将DLI表存储为textfile, avro, orc, sequencefile, rcfile, parquet等类型。

```
CREATE TABLE IF NOT EXISTS table1 (
  col_1   STRING,
  col_2   INT
)
STORED AS orc;
```

示例 2：创建 DLI 分区表

示例说明：创建一个名为student的分区表，该分区表使用院系编号（facultyNo）和班级编号（classNo）进行分区，该student表会同时按照不同的院系编号（facultyNo）和不同的班级编号（classNo）分区。

在实际的使用过程中，您可以选择合适的分区字段并将其添加到PARTITIONED BY关键字后。

```
CREATE TABLE IF NOT EXISTS student(
  id      int,
  name    STRING
)
STORED AS avro
PARTITIONED BY (
  facultyNo   INT,
  classNo     INT
);
```

 版权所有 © 华为技术有限公司

示例 3：使用 CTAS 语句将源表的全部数据或部分数据创建新的 DLI 表

示例说明：根据**示例1：创建DLI非分区表**中创建的DLI表table1，使用CTAS语法将table1中的数据复制到table1_ctas表中。

在使用CTAS建表的时候，可以忽略被复制的表在建表时所使用的语法，即不论在创建table1时使用的是何种语法，都可以使用DataSource语法的CTAS创建table1_ctas。

本例中table1中DLI表的存储格式为orc，而table1_ctas表的存储格式可以为parquet，即CTAS创建的表存储格式可以不同于原表。

在AS关键字后使用select语句选择需要插入到table1_ctas表中的数据。

SELECT语法为：SELECT <列名称> FROM <表名称> WHERE <相关筛选条件>。

- 示例中使用select * from table1，表示会从table1中选择所有语句，并将这些语句复制到table1_ctas表中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
STORED AS sequencefile
AS
SELECT  *
FROM table1;
```

- 若不需要table1中的全部数据，可以将“AS SELECT * FROM table1”改为“AS SELECT col_1 FROM table1 WHERE col_1 = “Ann”，这样就可以通过执行SELECT语句从table1中单独指定col_1列等于'Ann'的所有行插入到table1_ctas中。

```
CREATE TABLE IF NOT EXISTS table1_ctas
USING parquet
AS
SELECT col_1
FROM table1
WHERE col_1 = 'Ann';
```

示例 4：创建 DLI 非分区表，并自定义列字段数据类型

示例说明：创建名为table2的DLI非分区表，您可以根据业务需求自定义列子段的原生数据类型：

- 与文字字符有关可以使用STRING、CHAR或者VARCHAR。
- 与时间有关的可以使用TIMESTAMP、DATE。
- 与整数有关的可以使用INT、SMALLINT/SHORT、BIGINT/LONG、TINYINT。
- 涉及小数运算可以使用FLOAT、DOUBLE、DECIMAL。
- 若数据只涉及逻辑开关可以使用BOOLEAN类型。

具体使用方法与明细可以参照“数据类型 >原生数据类型”。

请参考**原生数据类型**。

```
CREATE TABLE IF NOT EXISTS table2 (
   col_01  STRING,
   col_02  CHAR (2),
   col_03  VARCHAR (32),
   col_04  TIMESTAMP,
   col_05  DATE,
   col_06  INT,
   col_07  SMALLINT,
   col_08  BIGINT,
   col_09  TINYINT,
   col_10  FLOAT,
   col_11  DOUBLE,
```

 版权所有 © 华为技术有限公司

```
  col_12  DECIMAL (10, 3),
  col_13  BOOLEAN
)
STORED AS parquet;
```

示例 5：创建 DLI 分区表，自定义表的 TBLPROPERTIES 参数

示例说明：本例创建名为table3并以col_3为分区依据的DLI分区表。在TBLPROPERTIES中配置dli.multi.version.enable、comment、orc.compress和auto.purge。

- dli.multi.version.enable：本例配置为true，即代表开启DLI数据多版本功能，用于表数据的备份与恢复。
- comment：表描述信息，TBLPROPERTIES内的描述信息支持后续修改。
- orc.compress：指定orc存储的压缩方式，本例定义为ZLIB。
- auto.purge：本例配置为true，即删除或者覆盖的数据会不经过回收站，直接被删除。

```
CREATE TABLE IF NOT EXISTs table3 (
  col_1 STRING,
  col_2 STRING
)
PARTITIONED BY (col_3 DATE)
STORED AS rcfile
TBLPROPERTIES (
  dli.multi.version.enable    = true,
  comment                     = 'Created by dli',
  orc.compress                = 'ZLIB',
  auto.purge                  = true
);
```

示例 6：创建 textfile 格式的非分区表，并设置 ROW FORMAT

示例说明：本例创建名为table4的textfile类型的非分区表，并设置ROW FORMAT相关格式（ROW FORMAT功能只支持textfile类型的表）。

- 字段（Fields）是表格中的列，每个字段有一个名称和数据类型，表中字段之间以'/'分隔。
- 集合项（COLLECTION ITEMS）指的是一组数据中的元素，可以是数组、列表或集合等，table4中集合项以'$'分隔。
- 映射键（MAP KEYS）是一种键值对的数据结构，用于存储一组相关联的数据，表中Map键以'#'分隔。
- 行（Rows）表格中的行，每一行包含一组字段值，表中行以'\n'结束（注意，只支持用'\n'作为行分隔符）。
- NULL表示缺少值或未知值的特殊值。在表格中，NULL表示该字段没有值或该值未知。如果数据中存在null值，则用字符串“null”表示。

```
CREATE TABLE IF NOT EXISTS table4 (
  col_1   STRING,
  col_2   INT
)
STORED AS TEXTFILE
ROW FORMAT
DELIMITED FIELDS TERMINATED    BY '/'
COLLECTION ITEMS TERMINATED    BY '$'
MAP KEYS TERMINATED            BY '#'
LINES TERMINATED               BY '\n'
NULL DEFINED                   AS 'NULL';
```

 版权所有 © 华为技术有限公司

5.3 删除表

功能描述

删除表。

语法格式

```
DROP TABLE [IF EXISTS] [db_name.]table_name;
```

关键字

- OBS表：仅删除其元数据信息，不删除存放在OBS上的数据。
- DLI表：删除其数据及相应的元数据信息。

参数说明

表 5-9 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。

注意事项

所要删除的表必须是当前数据库下存在的，否则会出错，可以通过添加IF EXISTS来避免出错。

示例

1. 参考**创建OBS表**或者**创建DLI表**中的示例描述创建对应的表。
2. 在当前所在数据库下删除名为test的表。
   ```
   DROP TABLE IF EXISTS test;
   ```

5.4 查看表

5.4.1 查看所有表

功能描述

查看当前数据库下所有的表。显示当前数据库下的所有表及视图。

 版权所有 © 华为技术有限公司

语法格式

```
SHOW TABLES [IN | FROM db_name] [LIKE regex_expression];
```

关键字

FROM/IN：指定数据库名，显示特定数据库下的表及视图。

参数说明

表 5-10 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
regex_expression	数据库下的表名称。

注意事项

无。

示例

1. 参考**创建OBS表**或者**创建DLI表**中的示例描述创建对应的表。
2. 查看当前所在数据库中的所有表与视图。
   ```
   SHOW TABLES;
   ```
3. 查看testdb数据库下所有以test开头的表。
   ```
   SHOW TABLES IN testdb LIKE "test*";
   ```

5.4.2 查看建表语句

功能描述

返回对应表的建表语句。

语法格式

```
SHOW CREATE TABLE table_name;
```

使用Spark 3.3.1版本引擎查看建表语句时请使用以下语法（仅适用于查询Hive表的建表语句）

```
SHOW CREATE TABLE table_name AS SERDE;
```

关键字

CREATE TABLE：建表语句。

 版权所有 © 华为技术有限公司

参数说明

表 5-11 参数说明

参数	描述
table_name	表名称。

注意事项

语句所涉及的表必须存在，否则会出错。

示例

Saprk 2.4.5版本示例：

- 执行以下命令返回测试表testDB01.testTable5的建表语句

  SHOW CREATE TABLE testDB01.testTable5

- 返回test表的建表语句：

```
createtab_stmt
 CREATE TABLE `testDB01`.`testTable5`(`id` INT, `age` INT, `money` DOUBLE)
COMMENT 'test'
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1'
)
STORED AS
  INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
TBLPROPERTIES (
  'hive.serialization.extend.nesting.levels' = 'true',
  'ddlUpdateTime' = '1707202585460'
)
```

Saprk 3.3.1版本示例：

- 执行以下命令返回测试表testDB02.testTable5的建表语句

  SHOW CREATE TABLE testDB02.testTable5 AS SERDE

- 返回test表的建表语句：

```
createtab_stmt
 CREATE TABLE testDB02.testTable5 (
  id INT,
  age INT,
  money DOUBLE)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1')
STORED AS
  INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
TBLPROPERTIES (
  'hive.serialization.extend.nesting.levels' = 'true',
  'transient_lastDdlTime' = '1707201874')
```

 版权所有 © 华为技术有限公司

5.4.3 查看表属性

功能描述

查看表的属性。

语法格式

```
SHOW TBLPROPERTIES table_name [('property_name')];
```

关键字

TBLPROPERTIES：TBLPROPERTIES子句允许用户给表添加key/value的属性。

参数说明

表 5-12 参数说明

参数	描述
table_name	表名称。
property_name	● 命令中不指定property_name时，将返回所有属性及其值； ● 命令中指定property_name时，将返回该特定property_name所对应的值。

注意事项

property_name大小写敏感，不能同时指定多个property_name，否则会出错。

示例

返回test表中属性property_key1的值。

```
SHOW TBLPROPERTIES test ('property_key1');
```

5.4.4 查看指定表所有列

功能描述

查看指定表中的所有列。

语法格式

```
SHOW COLUMNS {FROM | IN} table_name [{FROM | IN} db_name];
```

关键字

- COLUMNS：表中的列。
- FROM/IN：指定数据库，显示指定数据库下的表的列名。FROM和IN没有区别，可替换使用。

 版权所有 © 华为技术有限公司

参数说明

表 5-13 参数说明

参数	描述
table_name	表名称。
db_name	数据库名称。

注意事项

所指定的表必须是数据库中存在的表，否则会出错。

示例

查看student表中的所有列。

```
SHOW COLUMNS IN student;
```

5.4.5 查看指定表所有分区

功能描述

查看指定表的所有分区。

语法格式

```
SHOW PARTITIONS [db_name.]table_name
  [PARTITION partition_specs];
```

关键字

- PARTITIONS：表中的分区。
- PARTITION：分区。

参数说明

表 5-14 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。
table_name	Database中的表名，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。 匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$，如果特殊字符需要使用单引号（''）包围起来。

 版权所有 © 华为技术有限公司

参数	描述
partition_specs	分区信息，key=value形式，key为分区字段，value为分区值。若分区字段为多个字段，可以不包含所有的字段，会显示匹配上的所有分区信息。

注意事项

所要查看分区的表必须存在且是分区表，否则会出错。

示例

- 查看student表下面的所有的分区。
  ```
  SHOW PARTITIONS student;
  ```
- 查看student表中dt='2010-10-10'的分区。
  ```
  SHOW PARTITIONS student PARTITION(dt='2010-10-10');
  ```

5.4.6 查看表统计信息

功能描述

查看表统计信息。返回所有列的列名和列数据类型。

语法格式

```
DESCRIBE [EXTENDED|FORMATTED] [db_name.]table_name;
```

关键字

- EXTENDED：显示表的所有元数据，通常只在debug时用到。
- FORMATTED：使用表格形式显示所有表的元数据。

参数说明

表 5-15 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。
table_name	Database中的表名，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。如果特殊字符需要使用单引号（''）包围起来。

注意事项

若所查看的表不存在，将会出错。

 版权所有 © 华为技术有限公司

示例

查看student表的所有列的列名与列数据类型。

```
DESCRIBE student;
```

5.5 修改表

5.5.1 添加列

功能描述

添加一个或多个新列到表上。

语法格式

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name1 col_type1 [COMMENT
col_comment1], ...);
```

关键字

- ADD COLUMNS：添加列。
- COMMENT：列描述。

参数说明

表 5-16 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以下划线开头。
table_name	表名称。
col_name	列字段名称。
col_type	列字段类型。
col_comment	列描述。

注意事项

该SQL不建议并发执行，并发情况下添加列结果可能互相覆盖。

示例

```
ALTER TABLE t1 ADD COLUMNS (column2 int, column3 string);
```

 版权所有 © 华为技术有限公司

5.5.2 修改列注释

功能描述

修改非分区表或分区表的列注释信息。

语法格式

```
ALTER TABLE [db_name.]table_name CHANGE COLUMN col_name col_name col_type COMMENT 'col_comment';
```

关键字

- CHANGE COLUMN：修改列
- COMMENT：列描述。

参数说明

表 5-17 参数描述

参数	是否必选	描述
db_name	否	Database名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。
table_name	是	表名称。
col_name	是	列字段名称。col_name必须是已存在的列。
col_type	是	列数据类型。本语法不支持修改列数据类型，这里指定的是创建表时指定的列数据类型。
col_comment	是	修改后的列注释信息。注释内容为长度不超过1024字节的有效字符串。

示例

修改表t1中的c1列的注释信息为“the new comment”。

```
ALTER TABLE t1 CHANGE COLUMN c1 c1 STRING COMMENT 'the new comment';
```

5.5.3 开启或关闭数据多版本（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

DLI提供多版本功能，用于数据的备份与恢复。开启多版本功能后，在进行删除或修改表数据时（insert overwrite或者truncate操作），系统会自动备份历史数据并保留一

 版权所有 © 华为技术有限公司

定时间，后续您可以对保留周期内的数据进行快速恢复，避免因误操作丢失数据。其他多版本SQL语法请参考**多版本备份恢复数据（废弃，不推荐使用）**。

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表语法可以参考**使用Hive语法创建OBS表**。

语法格式

- 开启多版本功能
  ```
  ALTER TABLE [db_name.]table_name
  SET TBLPROPERTIES ("dli.multi.version.enable"="true");
  ```
- 关闭多版本功能
  ```
  ALTER TABLE [db_name.]table_name
  UNSET TBLPROPERTIES ("dli.multi.version.enable");
  ```
  开启多版本功能后，在执行insert overwrite或者truncate操作时会自动在OBS存储路径下存储多版本数据。关闭多版本功能后，需要通过如下命令把多版本数据目录回收。
  ```
  RESTORE TABLE [db_name.]table_name TO initial layout;
  ```

关键字

- SET TBLPROPERTIES：设置表属性，开启多版本功能。
- UNSET TBLPROPERTIES：取消表属性，关闭多版本功能。

参数说明

表 5-18 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。
table_name	表名称。

注意事项

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表语法可以参考**使用Hive语法创建OBS表**。

示例

- 修改表test_table，开启多版本功能。
  ```
  ALTER TABLE test_table
  SET TBLPROPERTIES ("dli.multi.version.enable"="true");
  ```
- 修改表test_table，关闭多版本功能。
  ```
  ALTER TABLE test_table
  UNSET TBLPROPERTIES ("dli.multi.version.enable");
  ```
  回退多版本路径。
  ```
  RESTORE TABLE test_table TO initial layout;
  ```

 版权所有 © 华为技术有限公司

5.6 分区相关

5.6.1 添加分区（只支持 OBS 表）

功能描述

创建OBS分区表成功后，OBS表实际还没有生成分区信息。生成分区信息主要有以下两种场景：

- 给OBS分区表插入对应的分区数据，数据插入成功后OBS表才会生成分区元数据信息，后续则可以根据对应分区列进行查询等操作。
- 手工拷贝分区目录和数据到OBS分区表路径下，执行本章节介绍的分区添加命令生成分区元数据信息，后续即可根据对应分区列进行查询等操作。

本章节重点介绍使用**ALTER TABLE**命令添加分区的基本操作和使用说明。

语法格式

```
ALTER TABLE table_name ADD [IF NOT EXISTS]
  PARTITION partition_specs1
  [LOCATION 'obs_path1']
  PARTITION partition_specs2
  [LOCATION 'obs_path2'];
```

关键字

- IF NOT EXISTS：指定该关键字以避免分区重复添加时报错。
- PARTITION：分区。
- LOCATION：分区路径。

参数说明

表 5-19 参数描述

参数	描述
table_name	表名称。
partition_specs	分区字段。
obs_path	OBS存储路径。

注意事项

- 向表中添加分区时，此表和分区列（建表时PARTITIONED BY指定的列）必须已存在，而所要添加的分区不能重复添加，否则将出错。已添加的分区可通过IF NOT EXISTS避免报错。
- 若分区表是按照多个字段进行分区的，添加分区时需要指定所有的分区字段，指定字段的顺序可任意。

版权所有 © 华为技术有限公司

- “partition_specs”中的参数默认带有“( )”。例如：**PARTITION (dt='2009-09-09',city='xxx')**。
- 在添加分区时若指定OBS路径，则该OBS路径必须是已经存在的，否则会出错。
- 若添加多个分区，每组PARTITION partition_specs LOCATION 'obs_path'之间用空格隔开。例如：

  PARTITION partition_specs LOCATION 'obs_path' PARTITION partition_specs LOCATION 'obs_path'。
- 若新增分区指定的路径包含子目录（或嵌套子目录），则子目录下面的所有文件类型及内容也将作为该分区的记录。

  您需要保证该分区目录下所有文件类型和文件内容与表的字段一致，否则查询将报错。

  您可以在建表语句OPTIONS中设置“multiLevelDirEnable”为true以查询子目录下的内容，此参数默认值为false（注意，此配置项为表属性，请谨慎配置。Hive表不支持此配置项)。

示例

- 建OBS表时仅有**一个分区列**，建表成功后添加分区数据。
  a. 先使用DataSource语法创建一个OBS分区表，分区列为external_data，数据存储在obs://bucketName/datapath路径下。
  ```
  create table testobstable(id varchar(128), external_data varchar(16)) using JSON OPTIONS (path 'obs://bucketName/datapath') PARTITIONED by (external_data);
  ```
  b. 拷贝分区数据目录到obs://bucketName/datapath路径下。例如当前拷贝external_data=22的分区目录下所有文件到obs://bucketName/datapath路径下。
  c. 执行添加分区命令，将分区的元数据信息生效。
  ```
  ALTER TABLE testobstable ADD
    PARTITION (external_data='22')
    LOCATION 'obs://bucketName/datapath/external_data=22';
  ```
  d. 添加分区成功后，即可根据分区列进行数据查询等操作。
  ```
  select * from testobstable where external_data='22';
  ```
- 建OBS表时有**多个分区列**，建表成功后添加分区数据。
  a. 先使用DataSource语法创建一个OBS分区表，分区列为external_data和dt，数据存储在obs://bucketName/datapath路径下。
  ```
  create table testobstable(
    id varchar(128),
    external_data varchar(16),
    dt varchar(16)
  ) using JSON OPTIONS (path 'obs://bucketName/datapath') PARTITIONED by (external_data, dt);
  ```
  b. 拷贝分区数据目录到obs://bucketName/datapath路径下。例如拷贝external_data=22及其子目录dt=2021-07-27和目录下文件到obs://bucketName/datapath路径下。
  c. 执行添加分区命令，将分区的元数据信息生效。
  ```
  ALTER TABLE
    testobstable
  ADD
    PARTITION (external_data = '22', dt = '2021-07-27') LOCATION 'obs://bucketName/datapath/external_data=22/dt=2021-07-27';
  ```
  d. 添加分区成功后，即可根据分区列进行数据查询等操作。
  ```
  select * from testobstable where external_data = '22';
  select * from testobstable where external_data = '22' and dt='2021-07-27';
  ```

 版权所有 © 华为技术有限公司

5.6.2 重命名分区（只支持 OBS 表）

功能描述

重命名分区。

语法格式

```
ALTER TABLE table_name
  PARTITION partition_specs
  RENAME TO PARTITION partition_specs;
```

关键字

- PARTITION：分区。
- RENAME：重命名。

参数说明

表 5-20 参数描述

参数	描述
table_name	表名称。
partition_specs	分区字段。

注意事项

- **该命令仅支持操作OBS表，不支持对DLI表进行操作**。
- 所要重命名分区的表和分区必须已存在，否则会出错。新分区名不能与其他分区重名，否则将出错。
- 若分区表是按照多个字段进行分区的，重命名分区时需要指定所有的分区字段，指定字段的顺序可任意。
- “partition_specs”中的参数默认带有“( )”，例如：**PARTITION (dt='2009-09-09',city='xxx')**。

示例

将student表中的分区city='xxx',dt='2008-08-08'重命名为city='xxx',dt='2009-09-09'。

```
ALTER TABLE student
  PARTITION (city='xxx',dt='2008-08-08')
  RENAME TO PARTITION (city='xxx',dt='2009-09-09');
```

5.6.3 删除分区

功能描述

本节操作介绍删除分区表的一个或多个分区。

 版权所有 © 华为技术有限公司

分区表分为两种，OBS表和DLI表。在删除分区时，DLI表和OBS表都支持利用指定条件删除分区表的一个或多个分区。OBS表还支持**按指定筛选条件删除分区**。

注意事项

- 所要删除分区的表必须是已经存在的表，否则会出错。
- 所要删除的分区必须是已经存在的，否则会出错，可通过语句中添加“IF EXISTS”避免该错误。

语法格式

```
ALTER TABLE [db_name.]table_name
  DROP [IF EXISTS]
  PARTITION partition_spec1[,PARTITION partition_spec2,...];
```

关键字

- DROP：删除表分区。
- IF EXISTS：所要删除的分区必须是已经存在的，否则会出错。
- PARTITION：分区。

参数说明

表 5-21 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。
table_name	Database中的表名，由字母、数字和下划线（_）组成。不能是纯数字，且不能以下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。如果特殊字符需要使用单引号（''）包围起来。
partition_specs	分区信息，key=value形式，key为分区字段，value为分区值。若分区字段为多个字段，可以不包含所有的字段，会删除匹配上的所有分区。“partition_specs”中的参数默认带有“( )”，例如：**PARTITION (facultyNo=20, classNo=103);**。

示例

为了便于理解删除分区语句的使用方法，本节示例为您提供源数据，基于源数据提供删除分区的操作示例。

步骤1 使用DataSource语法创建一个OBS表分区表。

创建了一个名为student的OBS分区表，表中有学生学号（id），学生姓名（name），学生院系编号（facultyNo）和学生班级编号（classNo），该表使用学生院系编号（facultyNo）和学生班级编号（classNo）进行分区。

```
create table if not exists student (
id int,
name STRING,
facultyNo int,
```

 版权所有 © 华为技术有限公司

```
classNo INT)
using csv
options (path 'obs://bucketName/filePath')
partitioned by (faculytNo, classNo);
```

步骤2 在表格中插入分区数据。

利用插入数据中的内容，可以插入以下数据

```
INSERT into student
partition (facultyNo = 10, classNo = 101)
values (1010101, "student01"), (1010102, "student02");

INSERT into student
partition (facultyNo = 10, classNo = 102)
values (1010203, "student03"), (1010204, "student04");

INSERT into student
partition (facultyNo = 20, classNo = 101)
values (2010105, "student05"), (2010106, "student06");

INSERT into student
partition (facultyNo = 20, classNo = 102)
values (2010207, "student07"), (2010208, "student08");

INSERT into student
partition (facultyNo = 20, classNo = 103)
values (2010309, "student09"), (2010310, "student10");

INSERT into student
partition (facultyNo = 30, classNo = 101)
values (3010111, "student11"), (3010112, "student12");

INSERT into student
partition (facultyNo = 30, classNo = 102)
values (3010213, "student13"), (3010214, "student14");
```

步骤3 查看分区。

利用查看指定表所有分区中的内容，可以查看相关的分区内容。

示例代码如下：

SHOW partitions student;

表 5-22 表数据示例

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102
facultyNo=20	classNo=103
facultyNo=30	classNo=101
facultyNo=30	classNo=102

步骤4 删除分区。

 版权所有 © 华为技术有限公司

- **示例1：指定多个筛选条件删除分区**

  本示例删除facultyNo为20，classNo为103的分区；

  说明

  如需按指定筛选条件删除分区请参考**指定筛选条件删除分区（只支持OBS表）**。

  示例代码如下：

  ```
  ALTER TABLE student
  DROP IF EXISTS
  PARTITION (facultyNo=20, classNo=103);
  ```

  重新利用第三步中的方法查看表中的分区，可以看到该分区被删除：

  ```
  SHOW partitions student;
  ```

- **示例2：指定单个筛选条件删除分区**

  本示例删除facultyNo为30的分区；在插入数据的过程中可以了解到，facultyNo为30的分区有两个。

  说明

  如需按指定筛选条件删除分区请参考**指定筛选条件删除分区（只支持OBS表）**。

  示例代码如下：

  ```
  ALTER TABLE student
  DROP IF EXISTS
  PARTITION (facultyNo = 30);
  ```

  执行后结果：

  表 5-23 表数据示例

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102
facultyNo=20	classNo=103

----结束

5.6.4 指定筛选条件删除分区（只支持 OBS 表）

功能描述

指定筛选条件删除分区表的一个或多个分区。

注意事项

- 该命令仅支持操作OBS表，不支持对DLI表进行操作。
- 删除操作仅删除表分区元数据，在使用本节语句删除分区时，OBS目录的数据不会自动删除。

版权所有 © 华为技术有限公司

- 所要删除分区的表必须是已经存在的表，否则会出错。
- 所要删除的分区必须是已经存在的，否则会出错，可通过语句中添加“IF EXISTS”避免该错误。

语法格式

```
ALTER TABLE [db_name.]table_name
  DROP [IF EXISTS]
  PARTITIONS partition_filtercondition;
```

关键字

- DROP：删除表分区。
- IF EXISTS：所要删除的分区必须是已经存在的，否则会出错。
- PARTITIONS：分区。

参数说明

表 5-24 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以下划线开头。
table_name	Database中的表名，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。如果特殊字符需要使用单引号（ '' ）包围起来。 **该命令仅支持操作OBS表，不支持对DLI表进行操作。**
partition_filtercondition	分区筛选条件。具体可以为以下格式： <分区列名> <**运算符**> <分区列比较值> 例如：start_date < '201911' • 示例1：<partition_filtercondition1> AND|OR <partition_filtercondition2> 例如：start_date < '201911' OR start_date >= '202006' • 示例2：（<partition_filtercondition1>)[,partitions (<partition_filtercondition2>), ...] 例如：(start_date <> '202007'), partitions(start_date < '201912')

示例

为了便于理解删除分区语句的使用方法，本节示例为您提供源数据，基于源数据提供删除分区的操作示例。

步骤1 使用DataSource语法创建一个OBS表分区表。

创建了一个名为student的OBS分区表，表中有学生学号（id），学生姓名（name），学生院系编号（facultyNo）和学生班级编号（classNo），该表使用学生院系编号（facultyNo）和学生班级编号（classNo）进行分区。

 版权所有 © 华为技术有限公司

```
create table if not exists student (
id int,
name STRING,
facultyNo int,
classNo INT)
using csv
options (path 'path 'obs://bucketName/filePath'')
partitioned by (faculytNo, classNo);
```

步骤2 在表格中插入分区数据。

利用插入数据中的内容，可以插入以下数据

```
INSERT into student
partition (facultyNo = 10, classNo = 101)
values (1010101, "student01"), (1010102, "student02");

INSERT into student
partition (facultyNo = 10, classNo = 102)
values (1010203, "student03"), (1010204, "student04");

INSERT into student
partition (facultyNo = 20, classNo = 101)
values (2010105, "student05"), (2010106, "student06");

INSERT into student
partition (facultyNo = 20, classNo = 102)
values (2010207, "student07"), (2010208, "student08");

INSERT into student
partition (facultyNo = 20, classNo = 103)
values (2010309, "student09"), (2010310, "student10");

INSERT into student
partition (facultyNo = 30, classNo = 101)
values (3010111, "student11"), (3010112, "student12");

INSERT into student
partition (facultyNo = 30, classNo = 102)
values (3010213, "student13"), (3010214, "student14");
```

步骤3 查看分区。

利用查看指定表所有分区中的内容，可以查看相关的分区内容。

示例代码如下：

```
SHOW partitions student;
```

表 5-25 表数据示例

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102
facultyNo=20	classNo=103
facultyNo=30	classNo=101
facultyNo=30	classNo=102

 版权所有 © 华为技术有限公司

步骤4 删除分区。

说明

本节操作介绍按指定筛选条件删除分区。无需指定筛选条件请参考**删除分区**。

本示例与**删除分区**不可混合使用。请区分此处关键字“partitions”与**删除分区**的关键字“partition”。

- **示例1：按指定筛选条件删除分区（该操作仅适用于OBS表），使用AND语句删除分区数据。**

  表 5-26 执行前数据

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102

  执行以下语句删除facultyNo = 20且classNo = 102分区：

```
ALTER TABLE student
DROP IF EXISTS
PARTITIONS (facultyNo = 20 AND classNo = 102);
```

  可以看到该语句会删除同时满足（AND）两个条件的分支的分区。

  表 5-27 执行后数据

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101

- **示例2：按指定筛选条件删除分区（该操作仅适用于OBS表），使用OR语句进行删除。**

  表 5-28 执行前数据

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102

 版权所有 © 华为技术有限公司

执行语句删除满足条件facultyNo = 10或classNo = 101的分区：

```
ALTER TABLE student
DROP IF EXISTS
PARTITIONS (facultyNo = 10),
PARTITIONS (classNo = 101);
```

执行结果：

表 5-29 执行后数据

facultyNo	classNo
facultyNo=20	classNo=102

在上述删除条件的框选下，分区记录中第一条数据既满足院系编号，又满足班级编号，第二条数据满足了院系编号，第三条数据满足了班级编号。

因此执行删除分区语句后只剩余1行分区。

按照方法一，上述执行语句还可以写成：

```
ALTER TABLE student
DROP IF EXISTS
PARTITIONS (facultyNo = 10 OR classNo = 101);
```

- **示例3：按指定筛选条件删除分区（该操作仅适用于OBS表），使用关系运算符语句删除指定分区。**

表 5-30 执行前数据

facultyNo	classNo
facultyNo=10	classNo=101
facultyNo=10	classNo=102
facultyNo=20	classNo=101
facultyNo=20	classNo=102
facultyNo=20	classNo=103

执行删除分区语句，删除classNo大于100小于102的分区：

```
ALTER TABLE student
DROP IF EXISTS
PARTITIONS (classNo BETWEEN 100 AND 102);
```

执行结果：

表 5-31 执行前数据

facultyNo	classNo
facultyNo=20	classNo=103

 版权所有 © 华为技术有限公司

----结束

5.6.5 修改表分区位置（只支持 OBS 表）

功能描述

修改表分区的位置。

语法格式

```
ALTER TABLE table_name
  PARTITION partition_specs
  SET LOCATION obs_path;
```

关键字

- PARTITION：分区。
- LOCATION：分区路径。

参数说明

表 5-32 参数描述

参数	描述
table_name	表名称。
partition_specs	分区字段。
obs_path	OBS存储路径。

注意事项

- 所要修改位置的表分区必须是已经存在的，否则将报错。
- “partition_specs”中的参数默认带有“( )”，例如：**PARTITION (dt='2009-09-09',city='xxx')**。
- 所指定的新的OBS路径必须是已经存在的绝对路径，否则将报错。
- 若新增分区指定的路径包含子目录（或嵌套子目录），则子目录下面的所有文件类型及内容也将作为该分区的记录。用户需要保证该分区目录下所有文件类型和文件内容与表的字段一致，否则查询将报错。

示例

将student表的分区dt='2008-08-08',city='xxx'的OBS路径设置为“obs://bucketName/fileName/student/dt=2008-08-08/city=xxx”。

```
ALTER TABLE student
  PARTITION(dt='2008-08-08',city='xxx')
  SET LOCATION 'obs://bucketName/fileName/student/dt=2008-08-08/city=xxx';
```

 版权所有 © 华为技术有限公司

5.6.6 更新表分区信息（只支持 OBS 表）

功能描述

更新表在元数据库中的分区信息。

语法格式

```
MSCK REPAIR TABLE table_name;
```

或

```
ALTER TABLE table_name RECOVER PARTITIONS;
```

关键字

- PARTITIONS：分区。
- SERDEPROPERTIES：Serde属性。

参数说明

表 5-33 参数描述

参数	描述
table_name	表名称。
partition_specs	分区字段。
obs_path	OBS存储路径。

注意事项

- 该命令的主要应用场景是针对分区表，如当手动在OBS上面添加分区目录时，再通过上述命令将该新增的分区信息刷新到元数据库中，通过“SHOW PARTITIONS table_name”命令查看新增的分区。
- 分区目录名称必须按照指定的格式输入，即“tablepath/partition_column_name=partition_column_value”。

示例

下述两语句都将更新表ptable在元数据库中的分区信息。

```
MSCK REPAIR TABLE ptable;
```

或

```
ALTER TABLE ptable RECOVER PARTITIONS;
```

 版权所有 © 华为技术有限公司

5.6.7 REFRESH TABLE 刷新表元数据

功能描述

Spark为了提高性能会缓存Parquet的元数据信息。当更新了Parquet表时，缓存的元数据信息未更新，导致Spark SQL查询不到新插入的数据作业执行报错，报错信息参考如下：

```
DLI.0002: FileNotFoundException: getFileStatus on  error message
```

该场景下就需要使用REFRESH TABLE来解决该问题。REFRESH TABLE是用于重新整理某个分区的文件，重用之前的表元数据信息，能够检测到表的字段的增加或者减少，主要用于表中元数据未修改，表的数据修改的场景。

语法格式

```
REFRESH TABLE [db_name.]table_name;
```

关键字

无。

参数说明

表 5-34 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以下划线开头。
table_name	表名称。Database中的表名，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以下划线开头。匹配规则为：^(?!_)(?![0-9]+$)[A-Za-z0-9_$]*$。如果特殊字符需要使用单引号（ ' ）包围起来。

注意事项

无。

示例

刷新表test的元数据信息。

```
REFRESH TABLE test;
```

5.7 多版本备份恢复数据（废弃，不推荐使用）

 版权所有 © 华为技术有限公司

5.7.1 设置多版本备份数据保留周期（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

在DLI数据多版本功能开启后，备份数据默认保留7天，您可以通过配置系统参数“dli.multi.version.retention.days”调整保留周期。保留周期外的多版本数据后续在执行insert overwrite或者truncate语句时会自动进行清理。在添加列或者修改分区表时，也可以设置表属性“dli.multi.version.retention.days”调整保留周期。

开启和关闭多版本功能SQL语法请参考**开启或关闭数据多版本（废弃，不推荐使用）**。

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。

语法格式

```
ALTER TABLE [db_name.]table_name
SET TBLPROPERTIES ("dli.multi.version.retention.days"="days");
```

关键字

- TBLPROPERTIES：TBLPROPERTIES子句给表添加key/value的属性。

参数说明

表 5-35 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。
days	多版本中备份数据保留的日期。默认为7天，建议设置在1到7天范围内。

注意事项

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表语法可以参考**使用Hive语法创建OBS表**。

示例

在DLI数据多版本中，设置备份数据保留时间为5天。

```
ALTER TABLE test_table
SET TBLPROPERTIES ("dli.multi.version.retention.days"="5");
```

 版权所有 © 华为技术有限公司

5.7.2 查看多版本备份数据（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

在DLI数据多版本功能开启后，您可以通过**SHOW HISTORY**命令查看表的备份数据。开启和关闭多版本语法请参考**开启或关闭数据多版本（废弃，不推荐使用）**。

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。

语法格式

- 查看某个非分区表的备份数据信息
  ```
  SHOW HISTORY FOR TABLE [db_name.]table_name;
  ```
- 查看指定分区的备份数据信息
  ```
  SHOW HISTORY FOR TABLE [db_name.]table_name PARTITION (column = value, ...);
  ```

关键字

- SHOW HISTORY FOR TABLE：查看备份数据。
- PARTITION：指定分区列。

参数说明

表 5-36 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。
column	分区列名。
value	分区列名对应的值。

注意事项

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表语法可以参考**使用Hive语法创建OBS表**。

示例

- 在DLI数据多版本中，查看表test_table多版本备份数据。
  ```
  SHOW HISTORY FOR TABLE test_table;
  ```
- 在DLI数据多版本中，查看分区表test_table对应dt分区的多版本备份数据。
  ```
  SHOW HISTORY FOR TABLE test_table PARTITION (dt='2021-07-27');
  ```

 版权所有 © 华为技术有限公司

5.7.3 恢复多版本备份数据（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

在DLI数据多版本功能开启后，您可以通过RESTORE TABLE命令恢复表或分区数据到指定版本。开启和关闭多版本语法请参考**开启或关闭数据多版本（废弃，不推荐使用）**。

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。

语法格式

- 恢复非分区表数据到指定版本的备份数据
  ```
  RESTORE TABLE [db_name.]table_name TO VERSION 'version_id';
  ```
- 恢复分区表的单个分区数据为指定版本的备份数据
  ```
  RESTORE TABLE [db_name.]table_name PARTITION (column = value, ...) TO VERSION 'version_id';
  ```

关键字

- RESTORE TABLE：恢复备份数据。
- PARTITION：指定分区列。
- TO VERSION：指定版本号。具体的版本号可以通过**SHOW HISTORY**命令获取，详情请参考**查看多版本备份数据（废弃，不推荐使用）**。

参数说明

表 5-37 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。
column	分区列名。
value	分区列名对应的值。
version_id	指定版本号恢复备份数据。具体的版本号可以通过**SHOW HISTORY**命令获取，详情请参考**查看多版本备份数据（废弃，不推荐使用）**。

注意事项

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。

版权所有 © 华为技术有限公司

示例

- 在DLI数据多版本中，恢复非分区表test_table数据到版本20210930。
```
RESTORE TABLE test_table TO VERSION '20210930';
```
- 在DLI数据多版本中，恢复分区表test_table对应dt分区数据到版本20210930。
```
RESTORE TABLE test_table PARTITION (dt='2021-07-27') TO VERSION '20210930';
```

5.7.4 配置多版本过期数据回收站（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

在DLI数据多版本功能开启后，过期的备份数据后续在执行insert overwrite或者truncate语句时会被系统直接清理。OBS并行文件系统可以通过配置回收站加速删除操作过期的备份数据。通过在表属性添加配置“dli.multi.version.trash.dir”即可开启回收站功能。开启和关闭多版本语法请参考**开启或关闭数据多版本（废弃，不推荐使用）**。

DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。

语法格式

```
ALTER TABLE [db_name.]table_name
SET TBLPROPERTIES ("dli.multi.version.trash.dir"="obs桶多版本回收站目录");
```

关键字

- TBLPROPERTIES：TBLPROPERTIES子句给表添加key/value的属性。

参数说明

表 5-38 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（_）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。
obs桶多版本回收站目录	当前OBS表所在桶下的一个目录，您可以根据需要调整目录路径。比如当前OBS表所在路径为“obs://bucketName/filePath”，OBS表目录下已创建Trash目录，则该回收站目录可以指定为“obs://bucketName/filePath/Trash”。

注意事项

- DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。
- 回收站数据自动清理需要在OBS并行文件系统的桶上配置回收站数据的生命周期规则。具体步骤参考如下：

 版权所有 © 华为技术有限公司

a. 在OBS服务控制台页面左侧选择“并行文件系统”，单击对应的文件系统名称。

b. 在“基础配置”下单击“生命周期规则”，创建或者编辑生命周期规则。

图 5-1 创建生命周期规则

示例

在DLI数据多版本中，通过配置回收站加速删除过期的备份数据，数据回收到OBS的/.Trash目录下。

```
ALTER TABLE test_table
SET TBLPROPERTIES ("dli.multi.version.trash.dir"="/.Trash");
```

5.7.5 清理多版本数据（废弃，不推荐使用）

功能描述

多版本功能计划废弃，不推荐使用。更多相关功能推荐：**Hudi多版本清理操作**、**Hudi Archive操作说明**。

多版本数据保留周期是在表每次执行insert overwrite或者truncate语句时触发，所以当表的多版本数据在保留周期时间外但是后续该表不会再执行insert overwrite或者truncate语句时，多版本保留周期外的数据不会自动清理。可以通过本章节介绍的SQL命令手动进行多版本数据清理。

语法格式

清理多版本保留周期外数据。

```
clear history for table [db_name.]table_name older_than '时间戳';
```

 版权所有 © 华为技术有限公司

关键字

- clear history for table：清理多版本数据。
- older_than：指定清理多版本数据的时间范围。

参数说明

表 5-39 参数说明

参数	描述
db_name	数据库名称，由字母、数字和下划线（ _ ）组成。不能是纯数字，且不能以数字和下划线开头。
table_name	表名称。
时间戳	删除该时间戳时间点之前的多版本数据。时间格式需要为yyyy-MM-dd HH:mm:ss

注意事项

- DLI数据多版本功能当前仅支持通过Hive语法创建的OBS表，具体建表SQL语法可以参考**使用Hive语法创建OBS表**。
- 该命令不会删除当前版本数据。

示例

删除dliTable表在2021-09-25 23:59:59之前生成的多版本数据（多版本生成时会自带一个生成时间时的时间戳）。

```
clear history for table dliTable older_than '2021-09-25 23:59:59';
```

5.8 表生命周期管理

5.8.1 创建表时指定表的生命周期

功能描述

DLI提供了表生命周期管理功能，在创建表时指定表的生命周期。DLI会根据每张表的最后修改时间和表的生命周期来判断是否要回收此表。通过设置表的生命周期，可以帮助您更好的管理数目众多的表，自动清理长期不再使用的数据表，简化数据表的回收流程。同时支持数据恢复设置，避免因误操作丢失数据。

表的回收规则

- 在创建表时通过TBLPROPERTIES指定表的生命周期。
  - **非分区表**

    如果表是非分区表，根据每张表的最后修改时间，经过生命周期时间后判断是否要回收此表。

 版权所有 © 华为技术有限公司

- **分区表**

  如果是分区表，则根据各分区的最后一次表数据被修改的时间（LAST_ACCESS_TIME）判断该分区是否该被回收。分区表的最后一个分区被回收后，该表不会被删除。

  分区表不支持设置分区级的生命周期，仅支持表级别的生命周期管理。

- 生命周期回收为每天定时启动，扫描全量分区。

  生命周期回收为每天定时启动，扫描全量分区的最后一次表数据被修改的时间（LAST_ACCESS_TIME）需要超过生命周期指定的时间才回收。

  假设某个分区表生命周期为1天，该分区数据最后一次被修改的时间是2023年05月20日15时。如果在2023年05月20日15时之前扫描此表（不到一天），则不会回收表分区。如果2023年05月20日回收扫描时发现表分区最后一次表数据被修改的时间（LAST_ACCESS_TIME）超过生命周期指定的时间，则上述分区会被回收。

- 生命周期主要提供定期回收表或分区的功能，每天根据服务的繁忙程度，不定时回收。不能确保表或分区的生命周期到期后，立刻被回收。
- 删除表后，表的所有属性信息全部会删除，包括生命周期。新建同名表后，表的生命周期以新设置的属性为准。

约束限制

- 表生命周期处于公测阶段，如果有需要请联系客服申请开通白名单。
- 使用生命周期前推荐您配置dli_data_clean_agency委托。

  委托权限策略请参考**DLI常见场景的委托权限策略**。

  如果没有配置dli_data_clean_agency委托，系统会默认读取DLI上一代系统委托dli_admin_agency，但是如果您的账户未配置dli_admin_agency，那么无法继续使用当前的表生命周期功能。

  您可以在IAM委托中查看您是否有dli_admin_agency的委托。
- 表生命周期功能支持Hive、DataSource语法创建表、多版本表，暂不支持跨源表、Carbon表。
- 生命周期单位为天，取值为正整数。
- 生命周期只能在表级别设置，不能在分区级设置。为分区表指定的生命周期，适用于该表所有的分区。
- 生命周期设置后，DLI表和OBS表支持数据备份，OBS表的备份目录需要手工设置。且备份目录应选择在并行文件系统上，备份目录必须和原表目录在同一个桶上，备份目录不能与原表相同目录或者子目录同名。

语法格式

- **DataSource语法创建DLI表**

```
CREATE TABLE table_name(name string, id int)
USING parquet
TBLPROPERTIES( "dli.lifecycle.days"=1 );
```

- **Hive语法创建DLI表**

```
CREATE TABLE table_name(name string, id int)
stored as parquet
TBLPROPERTIES( "dli.lifecycle.days"=1 );
```

- **DataSource语法创建OBS表**

```
CREATE TABLE table_name(name string, id int)
USING parquet
OPTIONS (path "obs://dli-test/table_name")
```

 版权所有 © 华为技术有限公司

```
TBLPROPERTIES( "dli.lifecycle.days"=1, "external.table.purge"='true', "dli.lifecycle.trash.dir"='obs://dli-test/Lifecycle-Trash' );
```

- **Hive语法创建OBS表**

```
CREATE TABLE table_name(name string, id int)
STORED AS parquet
LOCATION 'obs://dli-test/table_name'
TBLPROPERTIES( "dli.lifecycle.days"=1, "external.table.purge"='true', "dli.lifecycle.trash.dir"='obs://dli-test/Lifecycle-Trash' );
```

关键字

- TBLPROPERTIES：表的属性，增加表的生命周期功能。
- OPTIONS：新建表的路径，适用于DataSource语法创建OBS表。
- LOCATION:新建表的路径，适用于Hive语法创建OBS表。

参数说明

表 5-40 参数说明

参数名称	是否必选	参数说明
table_name	是	需要设置生命周期的表名。
dli.lifecycle.days	是	设置的生命周期时间，只能为正整数，单位为天。
external.table.purge	否	仅OBS表支持配置该参数。 是否需要在删除表或分区时，清除path路径下的数据。默认不删除。 设置'external.table.purge'='true'时： • 非分区OBS表配置删除文件后，表目录也会删除。 • 分区OBS表自定义分区数据也会删除。
dli.lifecycle.trash.dir	否	仅OBS表支持配置该参数。 设置'external.table.purge'='true'时，清除数据的备份目录，默认七天后删除备份数据。

示例

- **DataSource语法新建test_datasource_lifecycle表，生命周期为100天**

```
CREATE TABLE test_datasource_lifecycle(id int)
USING parquet
TBLPROPERTIES( "dli.lifecycle.days"=100);
```

- **Hive语法新建test_hive_lifecycle表，生命周期为100天。**

```
CREATE TABLE test_hive_lifecycle(id int)
stored as parquet
TBLPROPERTIES( "dli.lifecycle.days"=100);
```

- **DataSource语法新建test_datasource_lifecycle_obs表，生命周期为100天，过期时默认删除数据且数据备份至目录'obs://dli-test/'**。

```
CREATE TABLE test_datasource_lifecycle_obs(name string, id int)
USING parquet
OPTIONS (path "obs://dli-test/xxx")
```

 版权所有 © 华为技术有限公司

```
TBLPROPERTIES( "dli.lifecycle.days"=100, "external.table.purge"='true', "dli.lifecycle.trash.dir"='obs://dli-test/Lifecycle-Trash' );
```

- **Hive语法新建test_hive_lifecycle_obs表，生命周期为100天，过期时默认删除数据且数据备份至目录'obs://dli-test/'**。

```
CREATE TABLE test_hive_lifecycle_obs(name string, id int)
STORED AS parquet
LOCATION 'obs://dli-test/xxx'
TBLPROPERTIES( "dli.lifecycle.days"=100, "external.table.purge"='true', "dli.lifecycle.trash.dir"='obs://dli-test/Lifecycle-Trash' );
```

5.8.2 修改表生命周期的时间

功能描述

修改已存在的分区表或非分区表的生命周期。

当第一次开启生命周期时，会扫描表/分区会扫描路径下的表数据文件，更新表/分区的LAST_ACCESS_TIME，耗时与分区数和文件数相关。

约束限制

- 表生命周期处于公测阶段，如果有需要请联系客服申请开通白名单。
- 表生命周期功能支持Hive、DataSource语法创建表、多版本表，暂不支持跨源表、Carbon表。
- 生命周期单位为天，取值为正整数。
- 生命周期只能在表级别设置，不能在分区级设置。为分区表指定的生命周期，适用于该表所有的分区。

语法格式

```
ALTER TABLE table_name
SET TBLPROPERTIES("dli.lifecycle.days"='N')
```

关键字

TBLPROPERTIES：表的属性增加表的生命周期功能。

参数说明

表 5-41 修改表的生命周期参数说明

参数名称	是否必选	参数说明
table_name	是	需要修改生命周期的表名。
dli.lifecycle.days	是	修改后的生命周期时间，只能为正整数，单位为天。

示例

- 示例1：修改表的生命周期，开启test_lifecycle_exists表生命周期，并将生命周期设为50天。

 版权所有 © 华为技术有限公司

```
alter table test_lifecycle_exists
SET TBLPROPERTIES("dli.lifecycle.days"='50');
```

- 示例2：对已存在且未设置生命周期的分区表或非分区表开启表的生命周期,开启test_lifecycle_exists表生命周期，并将生命周期设为50天。

```
alter table test_lifecycle_exists
SET TBLPROPERTIES(
    "dli.lifecycle.days"='50',
    "dli.table.lifecycle.status"='enable'
);
```

5.8.3 禁止或恢复表的生命周期

功能介绍

禁止或恢复指定表或分区的生命周期。

使用禁止或恢复表的生命周期有以下两种场景：

1. 表或分区表开启了生命周期的功能，该功能可以禁止或恢复表的生命周期，即修改“dli.table.lifecycle.status”的参数值。
2. 表或分区表未开启生命周期的功能，使用禁止或恢复表的生命周期，则会增加“dli.table.lifecycle.status”这一属性。

约束限制

- 表生命周期处于公测阶段，如果有需要请联系客服申请开通白名单。
- 表生命周期功能支持Hive、DataSource语法创建表、多版本表，暂不支持跨源表、Carbon表。
- 生命周期单位为天，取值为正整数。
- 生命周期只能在表级别设置，不能在分区级设置。为分区表指定的生命周期，适用于该表所有的分区。

语法格式

- 该语法在表级别禁止或恢复表的生命周期

```
ALTER TABLE table_name SET TBLPROPERTIES("dli.table.lifecycle.status"={enable|disable});
```

- 该语法可在表或分区表级别禁止或恢复指定表

```
ALTER TABLE table_name [pt_spec] LIFECYCLE {enable|disable};
```

关键字

TBLPROPERTIES：表的属性，增加表的生命周期功能。

参数说明

表 5-42 禁止或恢复生命周期参数说明

参数名称	是否必选	参数说明
table_name	是	待禁止或恢复生命周期的表的名称。

 版权所有 © 华为技术有限公司

参数名称	是否必选	参数说明
pt_spec	否	待禁止或恢复生命周期的表的分区信息。格式为partition_col1=col1_value1, partition_col2=col2_value1...。对于有多级分区的表，必须指明全部的分区值。
enable	否	恢复表或指定分区的生命周期功能 ● 表及其分区重新参与生命周期回收，默认使用当前表及分区上的生命周期配置。 ● 开启表生命周期前可以修改表及分区的生命周期配置，防止开启表生命周期后因使用之前的配置导致数据被误回收。
disable	否	禁止表或指定分区的生命周期功能。 ● 禁止表本身及其所有分区被生命周期回收，优先级高于恢复表分区生命周期。即当使用禁止表或指定分区的生命周期功能时，设置待禁止或恢复生命周期的表的分区信息是无效的。 ● 禁止表的生命周期功能后，表的生命周期配置及其分区的enable和disable标记会被保留。 ● 禁止表的生命周期功能后，仍然可以修改表及分区表的生命周期配置。

示例

- 示例1：禁止表test_lifecycle的生命周期功能。
  ```
  alter table test_lifecycle SET TBLPROPERTIES("dli.table.lifecycle.status"='disable');
  ```
- 示例2：禁止表test_lifecycle中时间为20230520分区的生命周期功能。
  ```
  alter table test_lifecycle partition (dt='20230520') LIFECYCLE 'disable';
  ```

  说明

  - 当设置禁止分区表的生命周期功能后，该表的所有分区的生命周期功能都会被禁止。

 版权所有 © 华为技术有限公司

6 数据相关

6.1 导入数据

功能描述

LOAD DATA可用于导入CSV、Parquet、ORC、JSON、Avro格式的数据，内部将转换成Parquet数据格式进行存储。

语法格式

```
LOAD DATA INPATH 'folder_path' INTO TABLE [db_name.]table_name
  OPTIONS(property_name=property_value, ...);
```

关键字

- INPATH：数据路径。
- OPTIONS：属性列表。

参数说明

表 6-1 参数描述

参数	描述
folder_path	原始数据文件夹或者文件的OBS路径。
db_name	数据库名称。若未指定，则使用当前数据库。
table_name	需要导入数据的DLI表的名称。

以下是可以在导入数据时使用的配置选项：

- DATA_TYPE: 指定导入的数据类型，当前支持CSV、Parquet、ORC、JSON、Avro类型，默认值为“CSV”。
  配置项为OPTIONS('DATA_TYPE'='CSV')

 版权所有 © 华为技术有限公司

导入CSV和JSON文件时，有三种模式可以选择：

- PERMISSIVE：选择PERMISSIVE模式时，如果某一列数据类型与目标表列数据类型不匹配，则该行数据将被设置为null。
- DROPMALFORMED：选择DROPMALFORMED模式时，如果某一列数据类型与目标表列数据类型不匹配，则不导入该行数据。
- FAILFAST：选择FAILFAST模式时，如果某一列类型不匹配，则会抛出异常，导入失败。

模式设置可通过在OPTIONS中添加 OPTIONS('MODE'='PERMISSIVE')进行设置。

- DELIMITER：可以在导入命令中指定分隔符，默认值为“,”。

  配置项为OPTIONS('DELIMITER'=',')。

  对于CSV数据，支持如下所述分隔符：

  - 制表符tab，例如：'DELIMITER'='\t'。
  - 任意的二进制字符，例如：'DELIMITER'='\u0001(^A)'。
  - 单引号（'），单引号必须在双引号（" "）内。例如：'DELIMITER'= "'"。
  - DLI表还支持\001（^A）和\017（^Q），例如：'DELIMITER'='\001(^A)'，'DELIMITER'='\017(^Q)'。
- QUOTECHAR：可以在导入命令中指定引号字符。默认值为"。

  配置项为OPTIONS('QUOTECHAR'='"')
- COMMENTCHAR：可以在导入命令中指定注释字符。在导入操作期间，如果在行的开头遇到注释字符，那么该行将被视为注释，并且不会被导入。默认值为 # 。

  配置项为*OPTIONS('COMMENTCHAR'='#')*
- HEADER：用来表示源文件是否有表头。取值范围为“true”和“false”。“true”表示有表头，“false”表示无表头。默认值为“false”。如果没有表头，可以在导入命令中指定FILEHEADER参数提供表头。

  配置项为*OPTIONS('HEADER'='true')*
- FILEHEADER：如果源文件中没有表头，可在LOAD DATA命令中提供表头。

  OPTIONS('FILEHEADER'='column1,column2')
- ESCAPECHAR：如果用户想在CSV上对Escape字符进行严格验证，可以提供Escape字符。默认值为“\\”。

  配置项为OPTIONS('ESCAPECHAR'='\\')

  说明

  如果在CSV数据中输入ESCAPECHAR，该ESCAPECHAR必须在双引号（" "）内。例如："a \b"。
- MAXCOLUMNS：该可选参数指定了在一行中，CSV解析器解析的最大列数。

  配置项为*OPTIONS('MAXCOLUMNS'='400')*

表 6-2 MAXCOLUMNS

可选参数名称	默认值	最大值
MAXCOLUMNS	2000	20000

 版权所有 © 华为技术有限公司

说明

设置MAXCOLUMNS Option的值后，导入数据会对executor的内存有要求，所以导入数据可能会由于executor内存不足而失败。

- DATEFORMAT：指定列的日期格式。

  OPTIONS('DATEFORMAT'='dateFormat')

  说明

  - 默认值为：yyyy-MM-dd。
  - 日期格式由Java的日期模式字符串指定。在Java的日期和时间模式字符串中，未加单引号(')的字符'A' 到'Z' 和'a' 到'z' 被解释为模式字符，用来表示日期或时间字符串元素。若模式字符使用单引号 (') 引起来，则在解析时只进行文本匹配，而不进行解析。Java模式字符定义请参见**表6-3**。

  表 6-3 日期及时间模式字符定义

模式字符	日期或时间元素	示例
G	纪元标识符	AD
y	年份	1996; 96
M	月份	July; Jul; 07
w	年中的周数	27(该年的第27周)
W	月中的周数	2(该月的第2周)
D	年中的天数	189(该年的第189天)
d	月中的天数	10(该月的第10天)
u	星期中的天数	1 = 星期一, ..., 7 = 星期日
a	am/pm 标记	pm(下午时)
H	24小时数(0-23)	2
h	12小时数(1-12)	12
m	分钟数	30
s	秒数	55
S	毫秒数	978
z	时区	Pacific Standard Time; PST; GMT-08:00

- TIMESTAMPFORMAT：指定列的时间戳格式。

  OPTIONS('TIMESTAMPFORMAT'='timestampFormat')

  说明

  - 默认值为：yyyy-MM-dd HH:mm:ss。
  - 时间戳格式由Java的时间模式字符串指定。Java时间模式字符串定义详见**表3 日期及时间模式字符定义**。

版权所有 © 华为技术有限公司

- MODE：指定导入过程错误记录的处理模式，支持三种选项：PERMISSIVE、DROPMALFORMED和FAILFAST。

  OPTIONS('MODE'='permissive')

  说明

  - PERMISSIVE（默认）：尽可能地解析bad records，如果遇到不能转换的字段，则整行为null
  - DROPMALFORMED：忽略掉无法解析的bad records
  - FAILFAST：遇到无法解析的记录时，抛出异常并使Job失败
- BADRECORDSPATH：指定导入过程中错误记录的存储目录。

  OPTIONS('BADRECORDSPATH'='obs://bucket/path')

  说明

  配置该选项后，MODE不可配，固定为"DROPMALFORMED"，即将能够成功转换的记录导入到目标表，而将转换失败的记录存储到指定错误记录存储目录。

注意事项

- 导入OBS表时，创建OBS表时指定的路径必须是文件夹，若建表路径是文件将导致导入数据失败。
- 仅支持导入位于OBS路径上的原始数据。
- 不建议对同一张表并发导入数据，因为有一定概率发生并发冲突，导致导入失败。
- 导入数据时只能指定一个路径，路径中不能包含逗号。
- 当OBS桶目录下有文件夹和文件同名时，导入数据会优先指向该路径下的文件而非文件夹。
- 导入PARQUET、ORC及JSON类型数据时，必须指定*DATA_TYPE*这一OPTIONS，否则会以默认的“CSV”格式进行解析，从而导致导入的数据格式不正确。
- 导入CSV及JSON类型数据时，如果包含日期及时间列，需要指定*DATEFORMAT*及*TIMESTAMPFORMAT*选项，否则将以默认的日期及时间戳格式进行解析。

示例

说明

导入数据前已参考**创建OBS表**或者**创建DLI表**中的示例描述创建对应的表。

- 可使用下列语句将CSV文件导入到DLI表，“t”为表名。

```
LOAD DATA INPATH 'obs://dli/data.csv' INTO TABLE t
  OPTIONS('DELIMITER'=',' , 'QUOTECHAR'='"','COMMENTCHAR'='#','HEADER'='false');
```

- 可使用下列语句将JSON文件导入到DLI表，“jsontb”为表名。

```
LOAD DATA INPATH 'obs://dli/alltype.json' into table jsontb
  OPTIONS('DATA_TYPE'='json','DATEFORMAT'='yyyy/MM/dd','TIMESTAMPFORMAT'='yyyy/MM/dd HH:mm:ss');
```

6.2 插入数据

功能描述

将SELECT查询结果或某条数据插入到表中。

 版权所有 © 华为技术有限公司

约束限制

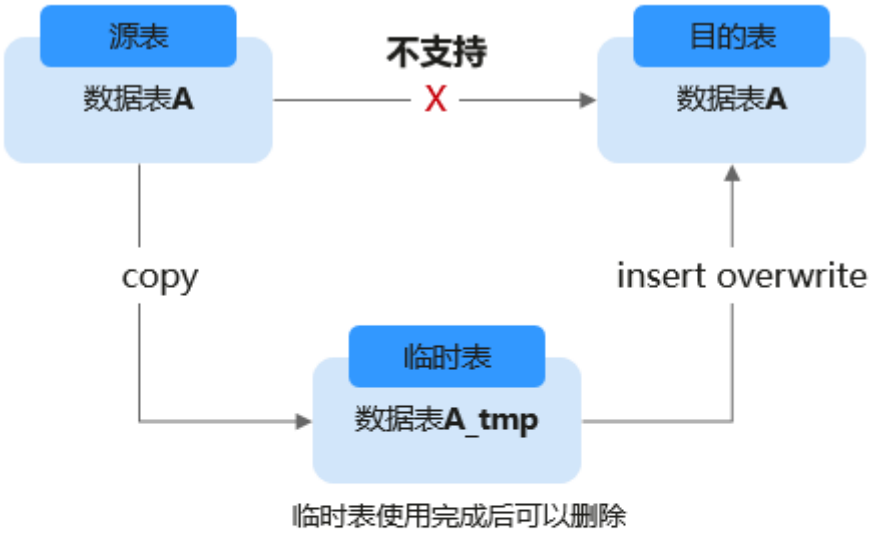
- insert overwrite语法不适用于同一张表内部的“自读自写”场景（包括分区表和非分区表），直接在原表上执行insert overwrite可能会导致数据丢失或数据不一致的风险。

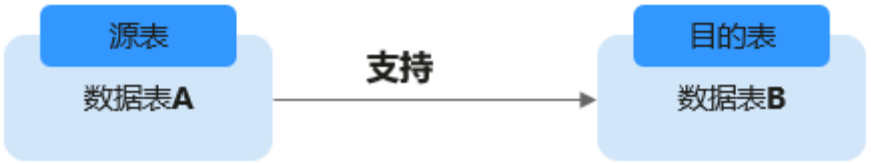
  如果要实现“自读自写”场景数据操作，建议使用一个临时表来处理数据。如图6-1所示。

  "自读自写"即目的的表和数据源表都是同一张表。例如，假设将student表中class_no = 1的学生信息提取出来并覆盖原表，以下语句即为自读自写场景典型语句。

  ```
  INSERT OVERWRITE TABLE student
  SELECT name FROM student WHERE class_no = 1;
  ```

  图 6-1 insert overwrite 自读自写备用方案

- 使用Hive和Datasource（除Hudi外）表在执行数据修改类命令（例如insert into，load data）时由于数据源不支持事务性，在系统故障或队列资源重启后，可能会导致数据重复或数据不一致等问题。

  为了避免这种情况，建议优先选择支持事务性的数据源，如Hudi类型数据源，该类数据源具备ACID（Atomicity、Consistency、Isolation、Durability）能力，有助于确保数据的一致性和准确性。

  了解更多：**执行Insert into后数据重复怎么办?**

版权所有 © 华为技术有限公司

语法格式

- 将SELECT查询结果插入到表中

```
INSERT INTO [TABLE] [db_name.]table_name
  [PARTITION part_spec] select_statement;
INSERT OVERWRITE TABLE [db_name.]table_name
  [PARTITION part_spec] select_statement;
part_spec:
  : (part_col_name1=val1 [, part_col_name2=val2, ...])
```

- 将某条数据插入到表中

```
INSERT INTO [TABLE] [db_name.]table_name
  [PARTITION part_spec] VALUES values_row [, values_row ...];
INSERT OVERWRITE TABLE [db_name.]table_name
  [PARTITION part_spec] VALUES values_row [, values_row ...];
values_row:
  : (val1 [, val2, ...])
```

关键字

表 6-4 INSERT 关键字说明

参数	描述
db_name	需要执行INSERT命令的表所在数据库的名称。
table_name	需要执行INSERT命令的表的名称。
part_spec	指定详细的分区信息。若分区字段为多个字段，需要包含所有的字段，但是可以不包含对应的值，系统会匹配上对应的分区。单表分区数最多允许100000个。
select_statement	源表上的SELECT查询（支持DLI表、OBS表）。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

注意事项

- 表必须已经存在。
- 如果动态分区不需要指定分区，则将“part_spec”作为普通字段放置SELECT语句中。
- 被插入的OBS表在建表时只能指定文件夹路径。
- 源表和目标表的数据类型和列字段个数应该相同，否则插入失败。
- 不建议对同一张表并发插入数据，可能会由于并发冲突导致插入数据结果异常。
- INSERT INTO命令用于将查询的结果追加到目标表中。
- INSERT OVERWRITE命令用于覆盖源表中已有的数据。
- INSERT INTO命令可以并行执行，INSERT OVERWRITE命令只有在分区表下不同的插入到不同静态分区才可以并行。
- INSERT INTO命令和INSERT OVERWRITE命令同时执行，其结果是未知的。
- 在从源表插入数据到目标表的过程中，无法在源表中导入或更新数据。
- 对于Hive分区表的动态INSERT OVERWRITE，支持覆盖涉及到的分区数据，不支持覆盖整表数据。

 版权所有 © 华为技术有限公司

- 如果需要覆盖DataSource表指定分区数据，需要先配置参数：dli.sql.dynamicPartitionOverwrite.enabled=true，再通过“insert overwrite”语句实现，“dli.sql.dynamicPartitionOverwrite.enabled”默认值为“false”，表示覆盖整表数据。例如：
```
insert overwrite table tb1 partition(part1='v1', part2='v2') select * from ...
```

  说明

  在“数据湖探索管理控制台>SQL编辑器”页面，单击编辑窗口右上角“设置”，可配置参数。

- 通过配置“spark.sql.shuffle.partitions”参数可以设置非DLI表在OBS桶中插入的文件个数，同时，为了避免数据倾斜，在INSERT语句后可加上“distribute by rand()”，可以增加处理作业的并发量。例如：
```
insert into table table_target select * from table_source distribute by cast(rand() * N as int);
```

示例

说明

导入数据前已参考**创建OBS表**或者**创建DLI表**中的示例描述创建对应的表。

- 示例1：将SELECT查询结果插入到表中
  - 使用DataSource语法创建一个parquet格式的分区表
```
CREATE TABLE data_source_tab1 (col1 INT, p1 INT, p2 INT)
  USING PARQUET PARTITIONED BY (p1, p2);
```
  - 插入查询结果到分区 (p1 = 3, p2 = 4)中
```
INSERT INTO data_source_tab1 PARTITION (p1 = 3, p2 = 4)
  SELECT id FROM RANGE(1, 3);
```
  - 插入新的查询结果到分区 (p1 = 3, p2 = 4) 中
```
INSERT OVERWRITE TABLE data_source_tab1 PARTITION (p1 = 3, p2 = 4)
  SELECT id FROM RANGE(3, 5);
```
- 示例2：将某条数据插入表中
  - 使用Hive语法创建一个parquet格式的分区表
```
CREATE TABLE hive_serde_tab1 (col1 INT, p1 INT, p2 INT)
  USING HIVE OPTIONS(fileFormat 'PARQUET') PARTITIONED BY (p1, p2);
```
  - 插入两条数据到分区 (p1 = 3, p2 = 4)中
```
INSERT INTO hive_serde_tab1 PARTITION (p1 = 3, p2 = 4)
  VALUES (1), (2);
```
  - 插入新的数据到分区 (p1 = 3, p2 = 4) 中
```
INSERT OVERWRITE TABLE hive_serde_tab1 PARTITION (p1 = 3, p2 = 4)
  VALUES (3), (4);
```

执行 Insert into 后数据重复怎么办?

- **问题现象：**

  使用Hive和Datasource（除Hudi外）表在执行数据修改类命令（例如insert into，load data）时由于数据源不支持事务性，在系统故障或队列资源重启后，可能会导致数据重复或数据不一致等问题。

- **原因分析：**

  在数据的Commit阶段如果出现队列资源重启可能会导致数据已经被修复到正式目录中。如果执行的是Insert into语句，资源重启后触发重试就会有概率导致数据重复写入。

- **解决方案：**

  a. 推荐使用具备ACID能力的Hudi类型数据源。

 版权所有 © 华为技术有限公司

b. 建议尽量使用insert overwrite这样幂等的语法而不是insert into等非幂等语法插入数据。
c. 如果严格需求数据不能重复，建议在insert into后对表数据执行去重操作，防止数据重复。

6.3 复用子查询

功能描述

在执行select查询语句时，通过对同一子查询的结果进行缓存，并在查询的不同部分重复使用，避免重复计算相同的子查询，从而提高查询性能。

语法格式

```
WITH cte_name AS (
    SELECT ...
    FROM table_name
    WHERE ...
)
SELECT ...
FROM cte_name
WHERE ...
```

关键字

表 6-5 关键字说明

参数	描述
cte_name	用户自定义的子查询名称。
table_name	执行子查询命令的表的名称。

示例

- 示例1：基于表sales定义子查询sales_data，查询销售金额大于100的商品，最后查询该子查询的所有数据。

```
WITH sales_data AS (
  SELECT product_name, sales_amount
  FROM sales
  WHERE sales_amount > 100
)
SELECT * FROM sales_data;
```

- 示例2：基于表sales定义子查询sales_data1和sales_data1，其中sales_data1查询销售金额大于100的商品；sales_data2查询销售金额小于20的商品；最后合并两个子查询的数据。

```
WITH sales_data1 AS (
  SELECT product_name, sales_amount
  FROM sales
  WHERE sales_amount > 100
),
sales_data2 AS (
  SELECT product_name, sales_amount
  FROM sales
  WHERE sales_amount < 20
)
```

 版权所有 © 华为技术有限公司

```
SELECT * FROM sales_data1
UNION ALL
SELECT * FROM sales_data2;
```

6.4 清空数据

功能描述

清除DLI表或者OBS表的数据。

语法格式

```
TRUNCATE TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)];
```

关键字

表 6-6 关键字说明

参数	描述
tablename	需要执行Truncate命令的DLI表或者OBS表的名称。
partcol1	需要删除的DLI表或者OBS表的分区名称。

注意事项

只支持清除DLI表或者OBS表的数据。

示例

```
truncate table test PARTITION (class = 'test');
```

 版权所有 © 华为技术有限公司

7 导出查询结果

功能描述

INSERT OVERWRITE DIRECTORY用于将查询结果直接写入到指定的目录，支持按CSV、Parquet、ORC、JSON、Avro格式进行存储。

语法格式

```
INSERT OVERWRITE DIRECTORY path
  USING file_format
  [OPTIONS(key1=value1)]
  select_statement;
```

关键字

- USING：指定所存储格式。
- OPTIONS：导出时的属性列表，为可选项。

参数

表 7-1 INSERT OVERWRITE DIRECTORY 参数描述

参数	描述
path	要将查询结果写入的OBS路径。
file_format	写入的文件格式，支持按CSV、Parquet、ORC、JSON、Avro格式。

说明

file_format为csv时，options参数可以参考表5-3。

注意事项

- 通过配置“spark.sql.shuffle.partitions”参数可以设置非DLI表在OBS桶中插入的文件个数，同时，为了避免数据倾斜，在INSERT语句后可加上“distribute by rand()”，可以增加处理作业的并发量。例如：

 版权所有 © 华为技术有限公司

```
insert into table table_target select * from table_source distribute by cast(rand() * N as int);
```

- 配置项为OPTIONS('DELIMITER=',')时，可以指定分隔符，默认值为“,”。
  对于CSV数据，支持如下所述分隔符：
  - 制表符tab，例如：'DELIMITER'='\t'。
  - 支持通过unicode编码指定分隔符，例如：'DELIMITER'='\u0001'。
  - 单引号（'），单引号必须在双引号（" "）内。例如：'DELIMITER'= "'"。

示例

```
INSERT OVERWRITE DIRECTORY 'obs://bucket/dir'
  USING csv
  OPTIONS(key1=value1)
  select * from db1.tb1;
```

 版权所有 © 华为技术有限公司

8 跨源连接相关

8.1 跨源连接 HBase 表

8.1.1 创建 DLI 表关联 HBase

功能描述

使用CREATE TABLE命令创建DLI表并关联HBase上已有的表。

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

- 创建DLI表关联HBase之前需要创建跨源连接。管理控制台操作请参考**增强型跨源连接**。
- 请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。
  如何添加IP域名映射，请参见《数据湖探索用户指南》中**增强型跨源连接**章节。
- 该语法不支持安全集群。

语法格式

- 单个RowKey

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME (
  ATTR1 TYPE,
  ATTR2 TYPE,
  ATTR3 TYPE)
  USING [CLOUDTABLE | HBASE] OPTIONS (
  'ZKHost'='xx',
  'TableName'='TABLE_IN_HBASE',
  'RowKey'='ATTR1',
  'Cols'='ATTR2:CF1.C1, ATTR3:CF1.C2');
```

- 组合RowKey

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME (
  ATTR1 String,
```


版权所有 © 华为技术有限公司

```
ATTR2 String,
ATTR3 TYPE)
USING [CLOUDTABLE | HBASE] OPTIONS (
'ZKHost'='xx',
'TableName'='TABLE_IN_HBASE',
'RowKey'='ATTR1:2, ATTR2:10',
'Cols'='ATTR2:CF1.C1, ATTR3:CF1.C2'
```

关键字

表 8-1 CREATE TABLE 关键字说明

参数	描述
USING [CLOUDTABLE | HBASE]	指定hbase datasource，"CLOUDTABLE"或"HBASE"二选一，大小写不敏感。
ZKHost	HBase集群的ZK连接地址。 获取ZK连接地址需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 ● 访问CloudTable集群，填写ZK连接地址（内网）。 ● 访问MRS集群，填写ZK所在节点IP与ZK对外端口，格式为："ZK_IP1:ZK_PORT1,ZK_IP2:ZK_PORT2"。 **说明** 访问MRS集群，只支持创建增强型跨源连接并且需要配置主机信息，管理控制台操作请参考**增强型跨源连接**，相关API信息请参考**创建增强型跨源连接**。
TableName	指定在HBase集群中已创建的表名。
RowKey	指定作为rowkey的dli关联表字段，支持单rowkey与组合rowkey。单rowkey支持数值与String类型，不需要指定长度。组合rowkey仅支持String类型定长数据，格式为：属性名1:长度,属性名2:长度。
Cols	通过逗号分隔的DLI表字段与HBase表的列之间的对应关系。其中，冒号前面放置DLI表字段，冒号后面放置HBase表信息，用‘.’分隔HBase表的列族与列名。

注意事项

- 若所要创建的表已经存在将报错，可以通过添加IF NOT EXISTS参数跳过该错误。
- OPTIONS中的所有参数是必选的，参数名称大小写不敏感，但参数值大小写敏感。
- OPTIONS中引号内的值前后不能带空格，空格也会被当做有效值。
- 表名及列名的描述仅支持字符串常量。
- 创建表时要说明列名及对应的数据类型，目前支持的数据类型为：boolean、short、int、long、float、double和string。
- 作为RowKey的字段（如上述语法格式中的ATTR1），其值不能为null，长度要大于0，小于或等于32767。

版权所有 © 华为技术有限公司

- Cols与RowKey中的字段加起来的数量必须与DLI表的字段保持一致，即表中所有的字段都到对应到Cols和RowKey中，但是顺序可以任意。
- 组合Rowkey只支持String类型，在使用组合Rowkey时，每个属性后面必须带上长度。当Rowkey指定的字段只有一个的时候，该字段的类型可以是支持的所有数据类型，并且不需要填写长度。
- 在组合Rowkey的场景中
  - 插入Rowkey数据时，如果某个属性的实际数据的长度比属性作为Rowkey时指定的长度要短，则会在数据后面补'\0'字符；如果某个属性的实际数据的长度比属性作为Rowkey时指定的长度要长，则会在实际插入HBase的时候进行截断。
  - 读取HBase上的Rowkey数据时，如果某个属性的实际数据的长度比属性作为Rowkey时指定的长度要短，则会抛出异常（OutofBoundException）；如果某个属性的实际数据的长度比属性作为Rowkey时指定的长度要长，则会在读取时进行截断。

示例

```
CREATE TABLE test_hbase(
ATTR1 int,
ATTR2 int,
ATTR3 string)
using hbase OPTIONS (
'ZKHost'='to-hbase-1174405101-CE1bDm5B.datasource.com:2181',
'TableName'='HBASE_TABLE',
'RowKey'='ATTR1',
'Cols'='ATTR2:CF1.C1, ATTR3:CF1.C2');
```

8.1.2 插入数据至 HBase 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的hbase表中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

关键字

SELECT对应关键字说明请参考**基本语句**。

 版权所有 © 华为技术有限公司

参数说明

表 8-2 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

注意事项

- DLI表必须已经存在。
- 在“**创建表关联HBase**”章节创建的表中，OPTIONS里的Cols指定的列族如果不存在，insert into执行时会报错。
- 如果插入的(rowkey, 列族, 列)已存在，则执行插入操作时，会覆盖hbase中相同的(rowkey, 列族, 列)。
- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。
- 不支持INSERT OVERWRITE语法。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.1.3 查询 HBase 表

SELECT命令用于查询hbase表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

版权所有 © 华为技术有限公司

注意事项

所查询的表必须是已经存在的表，否则会出错。

示例

查询表中的数据。

```
SELECT * FROM test_hbase limit 100;
```

查询下压

通过hbase进行数据过滤，即HBase Client将过滤条件传给HBase服务端进行处理，HBase服务端只返回用户需要的数据，提高了Spark SQL查询的速度。对于HBase不支持的过滤条件，例如组合Rowkey的查询，直接由Spark SQL进行。

- 支持查询下压的场景
  - 数据类型场景
    - Int
    - boolean
    - short
    - long
    - double
    - string

      说明

      float类型数据不支持查询下压。
  - 过滤条件场景
    - 过滤条件为>,<,>=,<=,=,!=,and,or

      例如：

      ```
      select * from tableName where (column1 >= value1 and column2<= value2) or column3 != value3
      ```
    - 过滤条件为like 和 not like，支持前缀，后缀和包含匹配

      例如：

      ```
      select * from tableName where column1 like "%value" or column2 like "value%" or column3 like "%value%"
      ```
    - 过滤条件为IsNotNull()

      例如：

      ```
      select * from tableName where IsNotNull(column)
      ```
    - 过滤条件为in ,not in

      例如：

      ```
      select * from tableName where column1 in (value1,value2,value3)  and column2 not in (value4,value5,value6)
      ```

 版权所有 © 华为技术有限公司

- 过滤条件为between _ and _
  例如：
  ```
  select * from tableName where column1 between value1 and value2
  ```
- 组合rowkey中的子rowkey过滤
  例如，组合Rowkey为column1+column2+column3，进行子rowkey查询：
  ```
  select * from tableName where column1= value1
  ```

- 不支持查询下压的场景
  - 数据类型场景
    除上述支持的数据类型外，其余复杂数据类型不支持查询下压。
  - 过滤条件场景
    - length，count，max，min，join，groupby，orderby，limit和avg等
    - 过滤条件为列比较
      例如：
      ```
      select * from tableName where column1 > (column2+column3)
      ```

8.2 跨源连接 OpenTSDB 表

8.2.1 创建 DLI 表关联 OpenTSDB

功能描述

使用CREATE TABLE命令创建DLI表并关联OpenTSDB上已有的metric，该语法支持CloudTable服务的OpenTSDB和MRS服务的OpenTSDB。

前提条件

创建DLI表关联OpenTSDB之前需要创建跨源连接。管理控制台操作请参考**增强型跨源连接**。

语法格式

```
CREATE TABLE [IF NOT EXISTS] UQUERY_OPENTSDB_TABLE_NAME
  USING OPENTSDB OPTIONS (
  'host' = 'xx;xx',
  'metric' = 'METRIC_NAME',
  'tags' = 'TAG1,TAG2');
```

 版权所有 © 华为技术有限公司

关键字

表 8-3 CREATE TABLE 关键字描述

参数	描述
host	OpenTSDB连接地址。 获取OpenTSDB连接地址需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 • 访问CloudTable OpenTSDB，填写OpenTSDB链接地址。 • 访问MRS OpenTSDB，若使用增强型跨源连接，填写OpenTSDB所在节点IP与端口，格式为"IP:PORT"，OpenTSDB存在多个节点时，用分号间隔。
metric	所创建的DLI表对应的OpenTSDB中的指标名称。
tags	metric对应的标签，用于归类、过滤、快速检索等操作。可以是1个到8个，以“,”分隔，包括对应metric下所有tagk的值。

注意事项

创建DLI表时，不需要指定timestamp和value字段，系统会根据指定的tags自动构建字段，包含以下字段，其中TAG1和TAG2由tags指定。

- TAG1 String
- TAG2 String
- timestamp Timestamp
- value double

示例

```
CREATE table opentsdb_table
  USING OPENTSDB OPTIONS (
  'host' = 'opentsdb-3xcl8dir15m58z3.cloudtable.com:4242',
  'metric' = 'city.temp',
  'tags' = 'city,location');
```

8.2.2 插入数据至 OpenTSDB 表

功能描述

使用INSERT INTO命令将DLI表中的数据插入到已关联的OpenTSDB metric中。

说明

若OpenTSDB上不存在metric，插入数据时会在OpenTSDB上自动创建一个新的metric。

语法格式

```
INSERT INTO TABLE TABLE_NAME SELECT * FROM DLI_TABLE;
INSERT INTO TABLE TABLE_NAME VALUES(XXX);
```


版权所有 © 华为技术有限公司

关键字

表 8-4 INSERT INTO 关键字说明

参数	描述
TABLE_NAME	所关联的OpenTSDB表名。
DLI_TABLE	创建的DLI表名称。

注意事项

- 插入的数据不能为null；插入的数据相同，会覆盖原数据；插入的数据只有value值不同，也会覆盖原数据。
- 不支持INSERT OVERWRITE语法。
- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。
- 时间戳格式只支持yyyy-MM-dd hh:mm:ss。

示例

```
INSERT INTO TABLE opentsdb_table VALUES('xxx','xxx','2018-05-03 00:00:00',21);
```

8.2.3 查询 OpenTSDB 表

SELECT命令用于查询OpenTSDB表中的数据。

说明

- 若OpenTSDB上不存在metric，查询对应的DLI表会报错。
- 若OpenTSDB开了安全模式，则访问时，需要设置conf:dli.sql.mrs.opentsdb.ssl.enabled=true

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

所查询的表必须是已经存在的表，否则会出错。

示例

查询表opentsdb_table中的数据。

```
SELECT * FROM opentsdb_table limit 100;
```

 版权所有 © 华为技术有限公司

8.3 跨源连接 DWS 表

8.3.1 创建 DLI 表关联 DWS

功能描述

使用CREATE TABLE命令创建DLI表并关联DWS上已有的表。

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

创建DLI表关联DWS之前需要创建跨源连接。管理控制台操作请参考**增强型跨源连接**。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME
  USING JDBC OPTIONS (
  'url'='xx',
  'dbtable'='db_name_in_DWS.table_name_in_DWS',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

关键字

表 8-5 CREATE TABLE 关键字说明

参数	描述
url	DWS的连接地址，需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 创建增强型跨源连接后，可以使用DWS提供的"JDBC连接字符串（内网）"，或者内网地址和内网端口访问，格式为"协议头://内网IP:内网端口/数据库名"，例如："jdbc:postgresql://192.168.0.77:8000/postgres"。 **说明** DWS的连接地址格式为："协议头://访问地址:访问端口/数据库名" 例如： jdbc:postgresql://to-dws-1174405119-ihlUr78j.datasource.com:8000/postgres 如果想要访问DWS中自定义数据库，请在这个连接里将"postgres"修改为对应的数据库名字。
dbtable	指定在DWS关联的表名，或者"模式名.表名"，例如：public.table_name。
user	（已废弃）DWS的用户名。

版权所有 © 华为技术有限公司

参数	描述
password	（已废弃）DWS集群的用户密码。
passwdauth	跨源密码认证名称。跨源认证信息创建方式请参考《数据湖探索用户指南》>《**跨源认证**》。
encryption	使用跨源密码认证时配置为“true”。
partitionColumn	读取数据时，用于设置并发使用的数值型字段。 **说明** • “partitionColumn”、“lowerBound”、“upperBound”、“numPartitions”四个参数必须同时设置，不支持仅设置其中某一个或某几个。 • 为了提升并发读取的性能，建议使用自增列。
lowerBound	partitionColumn设置的字段数据最小值，该值包含在返回结果中。
upperBound	partitionColumn设置的字段数据最大值，该值不包含在返回结果中。
numPartitions	读取数据时并发数。 **说明** 实际读取数据时，会根据“lowerBound”与“upperBound”，平均分配给每个task，获取其中一部分的数据。例如： 'partitionColumn'='id', 'lowerBound'='0', 'upperBound'='100', 'numPartitions'='2' 表示在DLI中会起2个并发task，一个task执行id>=0 and id < 50，另一个task执行id >=50 and id < 100。
fetchsize	读取数据时，每一批次获取数据的记录数，默认值1000。设置越大性能越好，但占用内存越多，该值设置过大会有内存溢出的风险。
batchsize	写入数据时，每一批次写入数据的记录数，默认值1000。设置越大性能越好，但占用内存越多，该值设置过大会有内存溢出的风险。
truncate	执行overwrite时是否不删除原表，直接执行清空表操作，取值范围： • true • false 默认为“false”，即在执行overwrite操作时，先将原表删除再重新建表。
isolationLevel	事务隔离级别，取值范围： • NONE • READ_UNCOMMITTED • READ_COMMITTED • REPEATABLE_READ • SERIALIZABLE 默认值为“READ_UNCOMMITTED”。

版权所有 © 华为技术有限公司

注意事项

创建DWS关联表时，不需要指定关联表的Schema。DLI会自动获取DWS中对应参数"dbtable"中的表的Schema。

示例

```
CREATE TABLE IF NOT EXISTS dli_to_dws
  USING JDBC OPTIONS (
  'url'='jdbc:postgresql://to-dws-1174405119-ih1Ur78j.datasource.com:8000/postgres',
  'dbtable'='test_dws',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

8.3.2 插入数据至 DWS 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的指定DWS表中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-6 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

 版权所有 © 华为技术有限公司

注意事项

- DLI表必须已经存在。
- DLI表在创建时不需要指定Schema信息，Schema信息将使用DWS表的信息。如果select子句中选择的字段数量和类型与DWS表的Schema信息不匹配时，系统将报错。
- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.3.3 查询 DWS 表

SELECT命令用于查询DWS表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

所查询的表必须是已经存在的表，否则会出错。

示例

查询表dli_to_dws中的数据。

```
SELECT * FROM dli_to_dws limit 100;
```

8.4 跨源连接 RDS 表

8.4.1 创建 DLI 表关联 RDS

功能描述

使用CREATE TABLE命令创建DLI表并关联RDS上已有的表。该功能支持访问RDS的MySQL集群和PostGre集群。

 版权所有 © 华为技术有限公司

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

创建DLI表关联RDS之前需要创建跨源连接。管理控制台操作请参考**增强型跨源连接**。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME
  USING JDBC OPTIONS (
  'url'='xx',
  'driver'='DRIVER_NAME',
  'dbtable'='db_name_in_RDS.table_name_in_RDS',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

关键字

表 8-7 CREATE TABLE 关键字说明

参数	描述
url	RDS的连接地址，需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 创建增强型跨源连接后，使用RDS提供的"内网域名"或者内网地址和数据库端口访问，MySQL格式为"协议头://内网IP:内网端口"，PostGre格式为"协议头://内网IP:内网端口/数据库名"。 例如："jdbc:mysql://192.168.0.193:3306"或者"jdbc:postgresql://192.168.0.193:5432/postgres"。
driver	jdbc驱动类名，访问MySQL集群请填写："com.mysql.jdbc.Driver"，访问PostGre集群请填写："org.postgresql.Driver"。
dbtable	• 访问MySQL集群填写"数据库名.表名" **注意** 连接的RDS数据库名不能包含中划线-或^特殊字符，否则会创建表失败。 • 访问PostGre集群填写"模式名.表名" **说明** 模式名即为数据库模式（schema）的名称。数据库中schema是数据库对象集合，包含了表，视图等多种对象。
user	（已废弃）RDS用户名。
password	（已废弃）RDS用户名密码。
passwdauth	跨源密码认证名称。跨源认证信息创建方式请参考《数据湖探索用户指南》>《**跨源认证**》。
encryption	使用跨源密码认证时配置为“true”。

版权所有 © 华为技术有限公司

参数	描述
partitionColumn	读取数据时，用于设置并发使用的数值型字段。 **说明** • “partitionColumn”、“lowerBound”、“upperBound”、“numPartitions”四个参数必须同时设置，不支持仅设置其中某一个或某几个。 • 为了提升并发读取的性能，建议使用自增列。
lowerBound	partitionColumn设置的字段数据最小值，该值包含在返回结果中。
upperBound	partitionColumn设置的字段数据最大值，该值不包含在返回结果中。
numPartitions	读取数据时并发数。 **说明** 实际读取数据时，会根据“lowerBound”与“upperBound”，平均分配给每个task，获取其中一部分的数据。例如： 'partitionColumn'='id', 'lowerBound'='0', 'upperBound'='100', 'numPartitions'='2' 表示在DLI中会起2个并发task，一个task执行id>=0 and id < 50，另一个task执行id >=50 and id < 100。
fetchsize	读取数据时，每一批次获取数据的记录数，默认值1000。设置越大性能越好，但占用内存越多，该值设置过大会有内存溢出的风险。
batchsize	写入数据时，每一批次写入数据的记录数，默认值1000。设置越大性能越好，但占用内存越多，该值设置过大会有内存溢出的风险。
truncate	执行overwrite时是否不删除原表，直接执行清空表操作，取值范围： • true • false 默认为“false”，即在执行overwrite操作时，先将原表删除再重新建表。
isolationLevel	事务隔离级别，取值范围： • NONE • READ_UNCOMMITTED • READ_COMMITTED • REPEATABLE_READ • SERIALIZABLE 默认值为“READ_UNCOMMITTED”。

注意事项

在首次创建RDS关联表时，无需指定关联表的Schema。DLI会自动获取RDS中对应参数"dbtable"中的表的Schema来完成关联表的创建。

 版权所有 © 华为技术有限公司

如果后续需要对RDS表的字段进行修改，关联表将不会自动同步这些更改。此时，您需要重新创建关联表，以确保关联表的Schema与修改后的RDS表保持一致。

示例

访问MySQL

```
CREATE TABLE IF NOT EXISTS dli_to_rds
  USING JDBC OPTIONS (
  'url'='jdbc:mysql://to-rds-117405104-3eAHxnlz.datasource.com:3306',
  'driver'='com.mysql.jdbc.Driver',
  'dbtable'='rds_test.test1',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

访问PostGre

```
CREATE TABLE IF NOT EXISTS dli_to_rds
  USING JDBC OPTIONS (
  'url'='jdbc:postgresql://to-rds-1174405119-oLRHAGE7.datasource.com:5432/postgreDB',
  'driver'='org.postgresql.Driver',
  'dbtable'='pg_schema.test1',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

8.4.2 插入数据至 RDS 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的指定RDS表中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-8 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。

 版权所有 © 华为技术有限公司

参数	描述
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

注意事项

- DLI表必须已经存在。
- DLI表在创建时不需要指定Schema信息，Schema信息将使用RDS表的信息。如果select子句中选择的字段数量和类型与RDS表的Schema信息不匹配时，系统将报错。
- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.4.3 查询 RDS 表

SELECT命令用于查询RDS表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

所查询的表必须是已经存在的表，否则会出错。

示例

查询表test_ct中的数据。

```
SELECT * FROM dli_to_rds limit 100;
```

 版权所有 © 华为技术有限公司

8.5 跨源连接 CSS 表

8.5.1 创建 DLI 表关联 CSS

功能描述

使用CREATE TABLE命令创建DLI表并关联CSS上已有的表。

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

创建DLI表关联CSS之前需要创建跨源连接。管理控制台操作请参考**增强型跨源连接**。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME(
  FIELDNAME1 FIELDTYPE1,
  FIELDNAME2 FIELDTYPE2)
  USING CSS OPTIONS (
  'es.nodes'='xx',
  'resource'='type_path_in_CSS',
  'pushdown'='true',
  'strict'='false',
  'batch.size.entries'= '1000',
  'batch.size.bytes'= '1mb',
  'es.nodes.wan.only' = 'true',
  'es.mapping.id' = 'FIELDNAME');
```

关键字

表 8-9 CREATE TABLE 关键字说明

参数	描述
es.nodes	CSS的连接地址，需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 创建增强型跨源连接后，使用CSS提供的"内网访问地址"，格式为"IP1:PORT1,IP2:PORT2"。
resource	指定在CSS关联的资源名，用"/index/type"指定资源位置（可简单理解index为database，type为table，但绝不等同）。 **说明** • ES 6.X版本中，单个Index只支持唯一type，type名可以自定义。 • ES 7.X版本中，单个Index将使用“_doc”作为type名，不再支持自定义。若访问ES 7.X版本时，该参数只需要填写index即可。

版权所有 © 华为技术有限公司

参数	描述
pushdown	CSS的下压功能是否开启，默认为“true”。包含大量IO传输的表在有where过滤条件的情况下能够开启pushdown降低IO。
strict	CSS的下压是否是严格的，默认为“false”。精确匹配的场景下比pushdown降低更多IO。
batch.size.entries	单次batch插入entry的条数上限，默认为1000。如果单条数据非常大，在bulk存储设置的数据条数前提前到达了单次batch的总数据量上限，则停止存储数据，以batch.size.bytes为准，提交该批次的数据。
batch.size.bytes	单次batch的总数据量上限，默认为1mb。如果单条数据非常小，在bulk存储到总数据量前提前到达了单次batch的条数上限，则停止存储数据，以batch.size.entries为准，提交该批次的数据。
es.nodes.wan.only	是否仅通过域名访问es节点，默认为false。使用css服务提供的原始内网IP地址作为es.nodes时，不需要填写该参数或者配置为false。
es.mapping.id	指定一个字段，其值作为es中Document的id。 **说明** • 相同/index/type下的Document id是唯一的。如果作为Document id的字段存在重复值，则在执行插入es时，重复id的Document将会被覆盖。 • 该特性可以用作容错解决方案。当插入数据执行一半时，DLI作业失败，会有部分数据已经插入到es中，这部分为冗余数据。如果设置了Document id，则在重新执行DLI作业时，会覆盖上一次的冗余数据。
es.net.ssl	连接安全CSS集群，默认值为false
es.certificate.name	连接安全CSS集群，使用的跨源认证信息名称。跨源认证信息创建方式请参考《数据湖探索用户指南》>《**跨源认证**》。

说明

batch.size.entries和batch.size.bytes分别对数据条数和数据量大小进行限制。

示例

```
CREATE TABLE IF NOT EXISTS dli_to_css (doc_id String, name string, age int)
  USING CSS OPTIONS (
 'es.nodes' ='to-css-1174404703-LzwpJEyx.datasource.com:9200',
 'resource' ='/dli_index/dli_type',
 'pushdown' ='false',
 'strict' ='true',
 'es.nodes.wan.only' ='true',
 'es.mapping.id' ='doc_id');
```

8.5.2 插入数据至 CSS 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的指定CSS表中。

 版权所有 © 华为技术有限公司

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-10 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

注意事项

- DLI表必须已经存在。
- DLI表在创建时需要指定Schema信息，如果select子句或者values中字段数量与CSS表的Schema字段数量不匹配时，系统将报错。
- 类型不一致时不一定报错，例如插入int类型数据，但CSS中Schema保存的是文本类型，int类型会被转换成文本类型。
- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

 版权所有 © 华为技术有限公司

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.5.3 查询 CSS 表

SELECT命令用于查询CSS表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

所查询的表必须是已经存在的表，否则会出错。

示例

查询表dli_to_css中的数据。

```
SELECT * FROM dli_to_css limit 100;
```

8.6 跨源连接 DCS 表

8.6.1 创建 DLI 表关联 DCS

功能描述

使用CREATE TABLE命令创建DLI表并关联DCS上已有的Key。

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

创建DLI表关联DCS之前需要创建跨源连接，绑定队列。管理控制台操作请参考**增强型跨源连接**。

语法格式

- 指定Key

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME(
   FIELDNAME1 FIELDTYPE1,
   FIELDNAME2 FIELDTYPE2)
 USING REDIS OPTIONS (
 'host'='xx',
 'port'='xx',
```

 版权所有 © 华为技术有限公司

```
 'passwdauth' = 'xxx',
 'encryption' = 'true',
 'table'='namespace_in_redis:key_in_redis',
 'key.column'= 'FIELDNAME1'
);
```

- 通配key

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME(
   FIELDNAME1 FIELDTYPE1,
   FIELDNAME2 FIELDTYPE2)
 USING REDIS OPTIONS (
 'host'='xx',
 'port'='xx',
 'passwdauth' = 'xxx',
 'encryption' = 'true',
 'keys.pattern'='key*:*',
 'key.column'= 'FIELDNAME1'
);
```

关键字

表 8-11 CREATE TABLE 关键字说明

参数	描述
host	DCS的连接IP，需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 创建增强型跨源连接后，使用DCS提供的"连接地址"。"连接地址"有多个时，选择其中一个即可。 **说明** 访问DCS目前只支持增强型跨源。
port	DCS的连接端口，例如6379。
password	（已废弃）创建DCS集群时填写的密码。访问非安全Redis集群时不需要填写。
passwdauth	跨源密码认证名称。跨源认证信息创建方式请参考《数据湖探索用户指南》>《**跨源认证**》。
encryption	使用跨源密码认证时配置为“true”。
table	对应Redis中的Key或Hash Key。 • 插入redis数据时必填。 • 查询redis数据时与“keys.pattern”参数二选一。
keys.pattern	使用正则表达式匹配多个Key或Hash Key。该参数仅用于查询时使用。查询redis数据时与“table”参数二选一。
key.column	非必填。用于指定schema中的某个字段作为Redis中key的标识。在插入数据时与参数“table”配合使用。
partitions.number	读取数据时，并发task数。
scan.count	每批次读取的数据记录数，默认为100。如果在读取过程中，redis集群中的CPU使用率还有提升空间，可以调大该参数。

版权所有 © 华为技术有限公司

参数	描述
iterator.grouping.size	每批次插入的数据记录数，默认为100。如果在插入过程中，redis集群中的CPU使用率还有提升空间，可以调大该参数。
timeout	连接redis的超时时间，单位ms，默认值2000（2秒超时）。

说明

访问DCS时，不支持复杂类型数据（Array、Struct、Map等）。

可以考虑以下几种方式进行复杂类型数据处理：

- 字段扁平化处理，将下一级的字段展开放在同一层Schema字段中。
- 使用二进制方式进行写入与读取，并通过自定义函数进行编解码。

示例

- 指定table

```
create table test_redis(name string, age int) using redis options(
  'host' = '192.168.4.199',
  'port' = '6379',
  'passwdauth' = 'xxx',
  'encryption' = 'true',
  'table' = 'person'
);
```

- 通配table名

```
create table test_redis_keys_patten(id string, name string, age int) using redis options(
  'host' = '192.168.4.199',
  'port' = '6379',
  'passwdauth' = 'xxx',
  'encryption' = 'true',
  'keys.pattern' = 'p*:*',
  'key.column' = 'id'
);
```

8.6.2 插入数据至 DCS 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的DCS Key中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

 版权所有 © 华为技术有限公司

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-12 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

注意事项

- DLI表必须已经存在。
- DLI表在创建时需要指定Schema信息。
- 如果在建表时指定“key.column”，则在Redis中会以指定字段的值作为Redis Key名称的一部分。例如：

```
create table test_redis(name string, age int) using redis options(
  'host' = '192.168.4.199',
  'port' = '6379',
  'passwdauth' = '******',
  'table' = 'test_with_key_column',
  'key.column' = 'name'
);
insert into test_redis values("James", 35), ("Michael", 22);
```

在redis中将会有2个名为test_with_key_column:James和test_with_key_column:Michael的表：

```
192.168.7.238:6379> keys test_with_key_column:*
1) "test_with_key_column:Michael"
2) "test_with_key_column:James"
192.168.7.238:6379>
```

```
192.168.7.238:6379> hgetall "test_with_key_column:Michael"
1) "age"
2) "22"
192.168.7.238:6379> hgetall "test_with_key_column:James"
1) "age"
2) "35"
192.168.7.238:6379>
```

- 如果在建表时没有指定“key.column”，则在Redis中的key name将会使用uuid。例如：

```
create table test_redis(name string, age int) using redis options(
  'host' = '192.168.7.238',
```

 版权所有 © 华为技术有限公司

```
  'port' = '6379',
  'passwdauth' = '******',
  'table' = 'test_without_key_column'
);
insert into test_redis values("James", 35), ("Michael", 22);
```

在redis中将会有2个以“test_without_key_column:uuid”命名的表：

```
192.168.7.238:6379> keys test_without_key_column:*
1) "test_without_key_column:b0ce581fa0d548e5b2273f4db1df6dcd"
2) "test_without_key_column:1e80aa7175d747ee9a82cce241767b01"
192.168.7.238:6379>
```

```
192.168.7.238:6379> hgetall "test_without_key_column:b0ce581fa0d548e5b2273f4db1df6dcd"
1) "age"
2) "35"
3) "name"
4) "James"
192.168.7.238:6379> hgetall "test_without_key_column:1e80aa7175d747ee9a82cce241767b01"
1) "age"
2) "22"
3) "name"
4) "Michael"
192.168.7.238:6379>
```

示例

```
INSERT INTO test_redis
  VALUES("James", 35), ("Michael", 22);
```

8.6.3 查询 DCS 表

SELECT命令用于查询DCS表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

示例

查询表test_redis中的数据。

```
SELECT * FROM test_redis limit 100;
```

8.7 跨源连接 DDS 表

8.7.1 创建 DLI 表关联 DDS

功能描述

使用CREATE TABLE命令创建DLI表并关联DDS上已有的collection。

说明

Spark跨源开发场景中直接配置跨源认证信息存在密码泄露的风险，优先推荐您使用DLI提供的跨源认证方式。

跨源认证简介及操作方法请参考**跨源认证简介**。

前提条件

创建DLI表关联DDS之前需要创建跨源连接，绑定队列。管理控制台操作请参考**增强型跨源连接**。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME(
   FIELDNAME1 FIELDTYPE1,
   FIELDNAME2 FIELDTYPE2)
 USING MONGO OPTIONS (
 'url'='IP:PORT[,IP:PORT]/[DATABASE][.COLLECTION][AUTH_PROPERTIES]',
 'database'='xx',
 'collection'='xx',
 'passwdauth' = 'xxx',
 'encryption' = 'true'
);
```

说明

文档数据库服务（Document Database Service，简称DDS）完全兼容MongoDB协议，因此语法中使用“using mongo options”。

关键字

表 8-13 CREATE TABLE 关键字说明

参数	描述
url	DDS的连接信息，需要先创建跨源连接，管理控制台操作请参考**增强型跨源连接**。 创建增强型跨源连接后，使用DDS提供的"随机连接地址"，格式为： "IP:PORT[,IP:PORT]/[DATABASE][.COLLECTION][AUTH_PROPERTIES]" 例如："192.168.4.62:8635,192.168.5.134:8635/test?authSource=admin"
database	DDS的数据库名，如果在"url"中同时指定了数据库名，则"url"中的数据库名不生效。
collection	DDS中的collection名，如果在"url"中同时指定了collection，则"url"中的collection不生效。
user	（已废弃）访问DDS集群用户名。
password	（已废弃）访问DDS集群密码
passwdauth	跨源密码认证名称。跨源认证信息创建方式请参考《数据湖探索用户指南》>《**跨源认证**》。
encryption	使用跨源密码认证时配置为“true”。

说明

如果在DDS中已存在collection，则建表可以不指定schema信息，DLI会根据collection中的数据自动生成schema信息。

版权所有 © 华为技术有限公司

示例

```
create table 1_datasource_mongo.test_momgo(id string, name string, age int) using mongo options(
  'url' = '192.168.4.62:8635,192.168.5.134:8635/test?authSource=admin',
  'database' = 'test',
  'collection' = 'test',
  'passwdauth' = 'xxx',
  'encryption' = 'true');
```

8.7.2 插入数据至 DDS 表

功能描述

INSERT INTO命令将DLI表中的数据插入到已关联的指定DDS表中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

- 覆盖插入数据

```
INSERT OVERWRITE TABLE DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-14 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

 版权所有 © 华为技术有限公司

注意事项

DLI表必须已经存在。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.7.3 查询 DDS 表

SELECT命令用于查询DDS表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

如果在建表时没有指定schema信息，则查询出来的结果将会包含"_id"字段用于存放doc中的"_id"。

示例

查询表test_table1中的数据。

```
SELECT * FROM test_table1 limit 100;
```

8.8 跨源连接 Oracle 表

8.8.1 创建 DLI 表关联 Oracle

功能描述

使用CREATE TABLE命令创建DLI表并关联Oracle上已有的表。

前提条件

- 创建DLI表关联Oracle之前需要创建增强型跨源连接。

 版权所有 © 华为技术有限公司

管理控制台操作请参考**增强型跨源连接**。

- 由于仅支持增强型跨源方式连接Oracle，且仅按需专属队列和包周期队列支持增强型跨源。因此仅按需专属队列和包周期队列支持在SQL作业中连接Oracle数据库。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME
  USING ORACLE OPTIONS (
  'url'='xx',
  'driver'='DRIVER_NAME',
  'dbtable'='db_in_oracle.table_in_oracle',
  'user' = 'xxx',
  'password' = 'xxx',
  'resource' = 'obs://rest-authinfo/tools/oracle/driver/ojdbc6.jar'
);
```

关键字

表 8-15 CREATE TABLE 关键字说明

参数	描述
url	Oracle的连接地址。 Oracle url支持以下格式： ● 格式一：jdbc:oracle:thin:@host:port:SID，其中SID是oracle数据库的唯一标识符。 ● 格式二：jdbc:oracle:thin:@//host:port/service_name；这种方式是Oracle推荐的，对于集群来说，每个节点的SID可能不一致，但ServiceName是一致的，包含所有节点。
driver	Oracle驱动类名: oracle.jdbc.driver.OracleDriver
dbtable	指定在Oracle关联的表名，或者"用户名.表名"，例如：public.table_name。
user	Oracle用户名。
password	Oracle用户名密码。
resource	Oracle驱动包的OBS路径。 例如：obs://rest-authinfo/tools/oracle/driver/ojdbc6.jar resource中定义的driver jar包如果被更新，需要重启队列，才会生效。

示例

创建Oracle跨源表

```
CREATE TABLE IF NOT EXISTS oracleTest
  USING ORACLE OPTIONS (
  'url'='jdbc:oracle:thin:@//192.168.168.40:1521/helowin',
  'driver'='oracle.jdbc.driver.OracleDriver',
  'dbtable'='test.Student',
  'user' = 'test',
```


版权所有 © 华为技术有限公司

```
  'password' = 'test',
  'resource' = 'obs://rest-authinfo/tools/oracle/driver/ojdbc6.jar'
);
```

8.8.2 插入数据至 Oracle 表

功能描述

INSERT INTO命令将数据插入到已关联的指定Oracle表中。

语法格式

- 将SELECT查询结果插入到表中：

```
INSERT INTO DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

- 将某条数据插入到表中：

```
INSERT INTO DLI_TABLE
  VALUES values_row [, values_row ...];
```

- 覆盖插入数据

```
INSERT OVERWRITE TABLE DLI_TABLE
  SELECT field1,field2...
  [FROM DLI_TEST]
  [WHERE where_condition]
  [LIMIT num]
  [GROUP BY field]
  [ORDER BY field] ...;
```

关键字

SELECT对应关键字说明请参考**基本语句**。

参数说明

表 8-16 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的DLI表名称。
DLI_TEST	为包含待查询数据的表。
field1,field2...，field	表“DLI_TEST”中的列值，需要匹配表“DLI_TABLE”的列值和类型。
where_condition	查询过滤条件。
num	对查询结果进行限制，num参数仅支持INT类型。
values_row	想要插入到表中的值，列与列之间用逗号分隔。

 版权所有 © 华为技术有限公司

注意事项

DLI表必须已经存在。

示例

- 查询表“user”中的数据插入表“test”中。

```
INSERT INTO test
  SELECT ATTR_EXPR
  FROM user
  WHERE user_name='cyz'
  LIMIT 3
  GROUP BY user_age
```

- 插入数据“1”到表“test”中

```
INSERT INTO test
  VALUES (1);
```

8.8.3 查询 Oracle 表

功能描述

SELECT命令用于查询Oracle表中的数据。

语法格式

```
SELECT * FROM table_name LIMIT number;
```

关键字

LIMIT：对查询结果进行限制，number参数仅支持INT类型。

注意事项

如果在建表时没有指定schema信息，则查询出来的结果将会包含"_id"字段用于存放doc中的"_id"。

示例

查询表test_oracle中的数据。

```
SELECT * FROM test_oracle limit 100;
```

 版权所有 © 华为技术有限公司

9 视图相关

9.1 创建视图

功能描述

创建视图。

语法格式

```
CREATE [OR REPLACE] VIEW view_name AS select_statement;
```

关键字

- CREATE VIEW：基于给定的select语句创建视图，不会将select语句的结果写入磁盘。
- OR REPLACE：指定该关键字后，若视图已经存在将不报错，并根据select语句更新视图的定义。

注意事项

- 所要创建的视图必须是当前数据库下不存在的，否则会报错。当视图存在时，可通过增加OR REPLACE关键字来避免报错。
- 视图中包含的表或视图信息不可被更改，如有更改可能会造成查询失败。
- 如果创建表和创建视图使用的计算引擎不一致，可能会因为varchar类型不兼容，导致视图查询失败。

  例如：使用Spark 3.x版本创建的表，建议您使用Spark 2.x创建相应的视图。

示例

先通过对student表中的id和name数据进行查询，并以该查询结果创建视图student_view。

```
CREATE VIEW student_view AS SELECT id, name FROM student;
```

 版权所有 © 华为技术有限公司

9.2 删除视图

功能描述

删除视图。

语法格式

```
DROP VIEW [IF EXISTS] [db_name.]view_name;
```

关键字

DROP：删除指定视图的元数据。虽然视图和表有很多共同之处，但是DROP TABLE不能用来删除VIEW。

注意事项

所要删除的视图必须是已经存在的，否则会出错，可以通过IF EXISTS来避免该错误。

示例

删除名为student_view的视图。

```
DROP VIEW student_view;
```


版权所有 © 华为技术有限公司

10 查看计划

功能描述

执行该语句将返回该SQL语句的逻辑计划与物理执行计划。

语法格式

```
EXPLAIN [EXTENDED | CODEGEN] statement;
```

关键字

EXTENDED：指定该关键字后，会同时输出逻辑计划与物理执行计划。

CODEGEN：指定该关键字后，若有codegen产生的代码也将输出。

注意事项

无。

示例

返回“SELECT * FROM test”SQL语句的逻辑计划与物理执行计划。

```
EXPLAIN EXTENDED select * from test;
```


版权所有 © 华为技术有限公司

11 数据权限相关

11.1 数据权限列表

DLI中SQL语句与数据库、表、角色相关的权限矩阵如**表11-1**所示。

表 11-1 权限矩阵

分类	SQL语句	权限	说明
Database	DROP DATABASE db1	database.db1的DROP_DATABASE权限	-
	CREATE TABLE tb1(...)	database.db1的CREATE_TABLE权限	-
	CREATE VIEW v1	database.db1的CREATE_VIEW权限	-
	EXPLAIN query	database.db1的EXPLAIN权限	query需要其相应的权限。
Table	SHOW CREATE TABLE tb1	database.db1.tables.tb1的SHOW_CREATE_TABLE权限	-
	DESCRIBE [EXTENDED| FORMATTED] tb1	databases.db1.tables.tb1的DESCRIBE_TABLE权限	-
	DROP TABLE [IF EXISTS] tb1	database.db1.tables.tb1的DROP_TABLE权限	-
	SELECT * FROM tb1	database.db1.tables.tb1的SELECT权限	-
	SELECT count(*) FROM tb1	database.db1.tables.tb1的SELECT权限	-

版权所有 © 华为技术有限公司

分类	SQL语句	权限	说明
	SELECT * FROM view1	database.db1.tables.view1的SELECT权限	-
	SELECT count(*) FROM view1	database.db1.tables.view1的SELECT权限	-
	LOAD DLI TABLE	database.db1.tables.tb1的INSERT_INTO_TABLE权限	-
	INSERT INTO TABLE	database.db1.tables.tb1的INSERT_INTO_TABLE权限	-
	INSERT OVERWRITE TABLE	database.db1.tables.tb1的INSERT_OVERWRITE_TABLE权限	-
	ALTER TABLE ADD COLUMNS	database.db1.tables.tb1的ALTER_TABLE_ADD_COLUMNS权限	-
	ALTER TABLE RENAME	database.db1.tables.tb1的ALTER_TABLE_RENAME权限	-
ROLE&PRIVILEGE	CREATE ROLE	db的CREATE_ROLE权限	-
	DROP ROLE	db的DROP_ROLE权限	-
	SHOW ROLES	db的SHOW_ROLES权限	-
	GRANT ROLES	db的GRANT_ROLE权限	-
	REVOKE ROLES	db的REVOKE_ROLE权限	-
	GRANT PRIVILEGE	db或table的GRANT_PRIVILEGE权限	-
	REVOKE PRIVILEGE	db或table的REVOKE_PRIVILEGE权限	-
	SHOW GRANT	db或table的SHOW_GRANT权限	-

Privilege在进行数据库和表赋权或回收权限时，DLI支持的权限类型如下所示。

- DATABASE上可赋权/回收的权限：
  - DROP_DATABASE（删除数据库）
  - CREATE_TABLE（创建表）
  - CREATE_VIEW（创建视图）
  - EXPLAIN（将SQL语句解释为执行计划）
  - CREATE_ROLE（创建角色）
  - DROP_ROLE（删除角色）

 版权所有 © 华为技术有限公司

  - SHOW_ROLES（显示角色）
  - GRANT_ROLE（绑定角色）
  - REVOKE_ROLE（解除角色绑定）
  - DESCRIBE_TABLE（描述表）
  - DROP_TABLE（删除表）
  - SELECT（查询表）
  - INSERT_INTO_TABLE（插入）
  - INSERT_OVERWRITE_TABLE（重写）
  - GRANT_PRIVILEGE（数据库的赋权）
  - REVOKE_PRIVILEGE（数据库权限的回收）
  - SHOW_PRIVILEGES（查看其他用户具备的数据库权限）
  - ALTER_TABLE_ADD_PARTITION（在分区表中添加分区）
  - ALTER_TABLE_DROP_PARTITION（删除分区表的分区）
  - ALTER_TABLE_RENAME_PARTITION（重命名表分区）
  - ALTER_TABLE_RECOVER_PARTITION（恢复表分区）
  - ALTER_TABLE_SET_LOCATION（设置分区的路径）
  - SHOW_PARTITIONS（显示所有分区）
  - SHOW_CREATE_TABLE（查看建表语句）
- TABLE上可以赋权/回收的权限：
  - DESCRIBE_TABLE（描述表）
  - DROP_TABLE（删除表）
  - SELECT（查询表）
  - INSERT_INTO_TABLE（插入）
  - INSERT_OVERWRITE_TABLE（重写）
  - GRANT_PRIVILEGE（表的赋权）
  - REVOKE_PRIVILEGE（表权限的回收）
  - SHOW_PRIVILEGES（查看其他用户具备的表权限）
  - ALTER_TABLE_ADD_COLUMNS（增加列）
  - ALTER_TABLE_RENAME（重命名表）
  - ALTER_TABLE_ADD_PARTITION（在分区表中添加分区）
  - ALTER_TABLE_DROP_PARTITION（删除分区表的分区）
  - ALTER_TABLE_RENAME_PARTITION（重命名表分区）
  - ALTER_TABLE_RECOVER_PARTITION（恢复表分区）
  - ALTER_TABLE_SET_LOCATION（设置分区的路径）
  - SHOW_PARTITIONS（显示所有分区）
  - SHOW_CREATE_TABLE（查看建表语句）

 版权所有 © 华为技术有限公司

11.2 创建角色

功能描述

- 在当前database或指定database中创建一个新的角色。
- 只有在database上具有CREATE_ROLE权限的用户才能创建角色。例如：管理员用户、database的owner用户和被赋予了CREATE_ROLE权限的其他用户。
- 每个角色必须属于且只能属于一个database。

语法格式

```
CREATE ROLE [db_name].role_name;
```

关键字

无。

注意事项

- 要创建的role_name必须在当前database或指定database中不存在，否则会报错。
- 当未指定“db_name”时，表示在当前database中创建角色。

示例

```
CREATE ROLE role1;
```

11.3 删除角色

功能描述

在当前database或指定database中删除角色。

语法格式

```
DROP ROLE [db_name].role_name;
```

关键字

无。

注意事项

- 要删除的role_name必须在当前database或指定database中存在，否则会报错。
- 当未指定“db_name”时，表示在当前database中删除角色。

示例

```
DROP ROLE role1;
```

 版权所有 © 华为技术有限公司

11.4 绑定角色

功能描述

绑定用户和角色。

语法格式

```
GRANT ([db_name].role_name,...) TO (user_name,...);
```

关键字

无。

注意事项

role_name和username必须存在，否则会报错。

示例

```
GRANT role1 TO user_name1;
```

11.5 解绑角色

功能描述

取消用户和角色的绑定。

语法格式

```
REVOKE ([db_name].role_name,...) FROM (user_name,...);
```

关键字

无。

注意事项

role_name和user_name必须存在，且user_name绑定了该role_name。

示例

取消用户user_name1和role1的绑定。

```
REVOKE role1 FROM user_name1;
```

 版权所有 © 华为技术有限公司

11.6 显示角色

功能描述

显示所有的角色或者显示当前database下绑定到“user_name”的角色。

语法格式

```
SHOW [ALL] ROLES [user_name];
```

关键字

ALL：显示所有的角色。

注意事项

ALL关键字与user_name不可同时存在。

示例

- 显示绑定到该用户的所有角色。
  ```
  SHOW ROLES;
  ```
- 显示project下的所有角色。
  ```
  SHOW ALL ROLES;
  ```

  说明

  只有管理员才有权限执行show all roles语句。
- 显示绑定到用户名为user_name1的所有角色。
  ```
  SHOW ROLES user_name1;
  ```

11.7 分配权限

功能描述

授予用户或角色权限。

语法格式

```
GRANT (privilege,...) ON (resource,..) TO ((ROLE [db_name].role_name) | (USER user_name)),...);
```

关键字

ROLE：限定后面的role_name是一个角色。

USER：限定后面的user_name是一个用户。

注意事项

- privilege必须是可授权限中的一种。且如果赋权对象在resource或上一级resource上已经有对应权限时，则会赋权失败。Privilege支持的权限类型可参见**数据权限列表**。

版权所有 © 华为技术有限公司

- resource可以是queue、database、table、view、column，格式分别为：
  - queue的格式为：queues.queue_name
    queue支持的Privilege权限类型可以参考下表：

操作	说明
DROP_QUEUE	删除队列
SUBMIT_JOB	提交作业
CANCEL_JOB	终止作业
RESTART	重启队列
SCALE_QUEUE	扩缩容队列
GRANT_PRIVILEGE	队列的赋权
REVOKE_PRIVILEGE	队列权限的回收
SHOW_PRIVILEGES	查看其他用户具备的队列权限

  - database的格式为：databases.db_name
    database支持的Privilege权限类型可参见**数据权限列表**。
  - table的格式为：databases.db_name.tables.table_name
    table支持的Privilege权限类型可参见**数据权限列表**。
  - view的格式为：databases.db_name.tables.view_name
    view支持的Privilege权限类型和table一样，具体可以参考**数据权限列表**中table的权限列表描述。
  - column的格式为：
    databases.db_name.tables.table_name.columns.column_name
    column支持的Privilege权限类型仅为：SELECT

示例

给用户user_name1授予数据库db1的删除数据库权限。

```
GRANT DROP_DATABASE ON databases.db1 TO USER user_name1;
```

给用户user_name1授予数据库db1的表tb1的SELECT权限。

```
GRANT SELECT ON databases.db1.tables.tb1 TO USER user_name1;
```

给角色role_name授予数据库db1的表tb1的SELECT权限。

```
GRANT SELECT ON databases.db1.tables.tb1 TO ROLE role_name;
```

11.8 回收权限

功能描述

回收已经授予用户或角色的权限。

版权所有 © 华为技术有限公司

语法格式

```
REVOKE (privilege,...) ON (resource,..) FROM ((ROLE [db_name].role_name) | (USER user_name)),...);
```

关键字

ROLE：限定后面的rol e_name是一个角色。

USER：限定后面的user_name是一个用户。

注意事项

- privilege必须为赋权对象在resource中的已授权限，否则会回收失败。Privilege支持的权限类型可参见**数据权限列表**。
- resource可以是queue、database、table、view、column，格式分别为：
  - queue的格式为：queues.queue_name
  - database的格式为：databases.db_name
  - table的格式为：databases.db_name.tables.table_name
  - view的格式为：databases.db_name.tables.view_name
  - column的格式为：databases.db_name.tables.table_name.columns.column_name

示例

回收用户user_name1对于数据库db1的删除数据库权限。

```
REVOKE DROP_DATABASE ON databases.db1 FROM USER user_name1;
```

回收用户user_name1对于数据库db1的表tb1的SELECT权限。

```
REVOKE SELECT ON databases.db1.tables.tb1 FROM USER user_name1;
```

回收角色role_name对于数据库db1的表tb1的SELECT权限。

```
REVOKE SELECT ON databases.db1.tables.tb1 FROM ROLE role_name;
```

11.9 显示已授权限

功能描述

显示某个用户在resource上已经授予的权限。

语法格式

```
SHOW GRANT USER user_name ON resource;
```

关键字

USER：限定后面的user_name是一个用户。

注意事项

resource可以是queue、database、table、column、view，格式分别为：

 版权所有 © 华为技术有限公司

- queue的格式为：queues.queue_name
- database的格式为：databases.db_name
- table的格式为：databases.db_name.tables.table_name
- column的格式为：databases.db_name.tables.table_name.columns.column_name
- view的格式为：databases.db_name.tables.view_name

示例

显示用户user_name1在数据库db1上的权限。

```
SHOW GRANT USER user_name1 ON databases.db1;
```

11.10 显示所有角色和用户的绑定关系

功能描述

在当前database显示角色与某用户的绑定关系。

语法格式

```
SHOW PRINCIPALS ROLE;
```

关键字

无。

注意事项

变量ROLE必须存在。

示例

```
SHOW PRINCIPALS role1;
```

 版权所有 © 华为技术有限公司

12 数据类型

12.1 概述

数据类型是数据的一个基本属性，用于区分不同类型的数据。不同的数据类型所占的存储空间不同，能够进行的操作也不相同。

数据库中的数据存储在表中。表中的每一列都定义了数据类型，用户存储数据时，须遵从这些数据类型的属性，否则可能会出错。

DLI当前只支持原生数据类型。

12.2 原生数据类型

DLI支持原生数据类型，请参见**表12-1**。

表 12-1 原生数据类型

数据类型	描述	存储空间	范围	OBS表支持情况	DLI表支持情况
INT	有符号整数	4字节	-2147483648～2147483647	是	是
STRING	字符串	-	-	是	是
FLOAT	单精度浮点型	4字节	-	是	是
DOUBLE	双精度浮点型	8字节	-	是	是

版权所有 © 华为技术有限公司

数据类型	描述	存储空间	范围	OBS表支持情况	DLI表支持情况
DECIMAL(precision,scale)	10进制精确数字类型。固定有效位数和小数位数的数据类型，例如：3.5 ● precision：表示最多可以表示多少位的数字。 ● scale：表示小数部分的位数。	-	1<=precision<=38 0<=scale<=38 若不指定precision和scale，则默认为decimal（38,38）。	是	是
BOOLEAN	布尔类型	1字节	TRUE/FALSE	是	是
SMALLINT/SHORT	有符号整数	2字节	-32768~32767	是	是
TINYINT	有符号整数	1字节	-128~127	是	否
BIGINT/LONG	有符号整数	8字节	-9223372036854775808 ~ 9223372036854775807	是	是
TIMESTAMP	时间戳，表示日期和时间，格式为原始数据。例如：1621434131222	-	-	是	是
CHAR	固定长度字符串	-	-	是	是
VARCHAR	可变长度字符串	-	-	是	是
DATE	日期类型，描述了特定的年月日，以yyyy-mm-dd格式表示，例如：2014-05-29	-	DATE类型不包含时间，所表示日期的范围为0000-01-01~9999-12-31。	是	是

 版权所有 © 华为技术有限公司

说明

- VARCHAR和CHAR在DLI实际存储是STRING型，因此超出长度的字符串不会被截断。
- FLOAT类型在DLI实际存储是DOUBLE型。

INT

有符号整数，存储空间为4字节，-2147483648～2147483647，在NULL情况下，默认值为0。

STRING

字符串类型。

FLOAT

单精度浮点型，存储空间为4字节，在NULL情况下，采用计算值默认值为0。

由于浮点类型的数据在计算机中的存储方式的限制，在比较两个浮点类型的数据是否相等时，因存在精度问题，不能直接采用“a==b”的方式进行比较，建议使用“(a-b)的绝对值<=EPSILON”这种方式进行比较，EPSILON为允许的误差范围，一般为1.19209290E-07F。若两个浮点数的差值的绝对值在这个范围内就认为相等。

DOUBLE

双精度浮点型，存储空间为8字节，在NULL情况下，采用计算值默认值为0。

由于浮点类型的数据在计算机中的存储方式的限制，在比较两个浮点类型的数据是否相等时，因存在精度问题，不能直接采用“a==b”的方式进行比较，建议使用“(a-b)的绝对值<=EPSILON”这种方式进行比较，EPSILON为允许的误差范围，一般为2.2204460492503131E-16。若两个浮点数的差值的绝对值在这个范围内就认为相等。

DECIMAL

Decimal(p,s)表示数值中共有p位数，其中整数p-s位，小数s位。p表示可储存的最大十进制数的位数总数，小数点左右两侧都包括在内。有效位数p必须是1至最大有效位数38之间的值。s表示小数点右侧所能储存的最大十进制数的位数。小数位数必须是从0到p的值。只有在指定了有效位数时，才能指定小数位数。因此，$0 \le s \le p$。例如：decimal(10,6)，表示数值中共有10位数，其中整数占4位，小数占6位。

BOOLEAN

布尔类型，包括TRUE与FALSE。

SMALLINT/SHORT

有符号整数，存储空间为2字节，范围为-32768～32767。当为NULL情况下，采用计算值默认为0。

TINYINT

有符号整数，存储空间为1字节，范围为-128～127。当为NULL情况下，采用计算值默认为0。

 版权所有 © 华为技术有限公司

BIGINT/LONG

有符号整数，存储空间为8字节，范围为-9223372036854775808～9223372036854775807，不支持科学计数法。当为NULL情况下。采用计算值默认为0。

TIMESTAMP

支持传统的UNIX TIMESTAMP，提供达到微秒级别精度的选择。TIMESTAMP是以指定时间和UNIX epoch（UNIX epoch时间为1970年1月1日00:00:00）之间的秒数差定义的。可以向TIMESTAMP隐性转换的数据类型有STRING（必须具有"yyyy-MM-dd HH:mm:SS[.ffffff]"格式。小数点后精度可选）。

CHAR

CHAR的长度是固定的，使用指定长度的固定长度表示字符串。DLI中实际存储为STRING类型。

VARCHAR

VARCHAR生成时会带有一个长度指定数，用来定义字符串中的最大字符数。如果一个向VARCHAR转换的STRING型中的字符个数超过了长度指定数，那么这个STRING会被自动缩短。和STRING类型一样，VARCHAR末尾的空格数是有意义的，会影响比较结果。DLI中实际存储为STRING类型。

DATE

DATE类型只能和DATE、TIMESTAMP和STRING进行显式转换（cast），具体如**表12-2**所示。

表 12-2 cast 函数转换

显式转换	转换结果
cast(date as date)	相同DATE值。
cast(timestamp as date)	根据本地时区从TIMESTAMP得出年/月/日，将其作为DATE值返回。
cast(string as date)	如果字符串的形式是“yyyy-MM-dd”，将对应年/月/日作为DATE值返回。如果字符串不具有这种形式，返回空。
cast(date as timestamp)	根据本地时区生成并返回对应DATE的年/月/日零点的TIMESTAMP值。
cast(date as string)	根据DATE的年/月/日值生成并返回“yyyy-MM-dd”格式的字符串。

12.3 复杂数据类型

Spark SQL支持复杂数据类型，如**表12-3**所示。

 版权所有 © 华为技术有限公司

表 12-3 复杂数据类型

数据类型	描述	使用格式
ARRAY	一组有序字段，使用指定的值构造ARRAY数组。可以为任意类型，要求所有字段的数据类型必须相同。	array(<value>,<value>[, ...]) 具体使用示例详见：**ARRAY示例**。
MAP	一组无序的键/值对，使用给定的Key和Value对生成MAP。键的类型必须是原生数据类型，值的类型可以是原生数据类型或复杂数据类型。同一个MAP键的类型必须相同，值的类型也必须相同。	map(K <key1>, V <value1>, K <key2>, V <value2>[, ...]) 具体使用示例详见：**MAP示例**。
STRUCT	一组命名的字段，字段的数据类型可以不同。	struct(<value1>,<value2>[, ...]) 具体使用示例详见：**STRUCT示例**。

使用限制

- 创建含有复杂数据类型字段的表时，该表存储格式不支持CSV（txt）。
- 如果表中含有复杂数据类型字段时，该表不支持CSV（txt）格式的文件数据导入。
- MAP数据类型建表必须指定schema，且不支持date、short、timestamp数据类型。
- 对于JSON格式OBS表，MAP的键类型只支持STRING类型。
- 由于MAP类型的键不能为NULL，MAP键不支持对插入数据进行可能出现NULL值类型之间的隐式转换，如：STRING类型转换为其他原生类型、FLOAT类型转换为TIMESTAMP类型、其他原生类型转换为DECIMAL类型等。
- STRUCT数据类型不支持double，boolean数据类型。

ARRAY 示例

创建表“array_test”，将“id”参数定义为“ARRAY<INT>”数据类型，“name”参数定义为“STRING”数据类型。建表成功后插入测试数据到“array_test”中。操作如下：

1. 创建表。

   CREATE TABLE array_test(name STRING, id ARRAY < INT >) USING PARQUET;

2. 插入测试数据。

   INSERT INTO array_test VALUES ('test',array(1,2,3,4));

   INSERT INTO array_test VALUES ('test2',array(4,5,6,7))

   INSERT INTO array_test VALUES ('test3',array(7,8,9,0));

3. 查询结果。

   查“array_test”表中的所有数据：

 版权所有 © 华为技术有限公司

SELECT * FROM array_test;

```
test3    [7,8,9,0]
test2    [4,5,6,7]
test     [1,2,3,4]
```

查“array_test”表中id数组第0个元素的数据。

SELECT id[0] FROM array_test;

```
7
4
1
```

MAP 示例

创建表“map_test”，将“score”参数定义为“map<STRING,INT>)”数据类型（键为STRING类型，值为INT类型）。建表成功后插入测试数据至“map_test”中。操作如下：

1. 创建表。

   CREATE TABLE map_test(id STRING, score map<STRING,INT>) USING PARQUET;

2. 插入测试数据。

   INSERT INTO map_test VALUES ('test4',map('math',70,'chemistry',84));

   INSERT INTO map_test VALUES ('test5',map('math',85,'chemistry',97));

   INSERT INTO map_test VALUES ('test6',map('math',88,'chemistry',80));

3. 查询结果。

   查询“map_test”表里的所有数据。

   SELECT * FROM map_test;

   ```
   test6    {"chemistry":80,"math":88}
   test5    {"chemistry":97,"math":85}
   test4    {"chemistry":84,"math":70}
   ```

   查询“map_test”表中的数学成绩。

   SELECT id, score['Math'] FROM map_test;

   ```
   test6    88
   test5    85
   test4    70
   ```

STRUCT 示例

创建表“struct_test”，将info定义为“STRUCT<name:STRING, age:INT>”数据类型（由name和age构成的字段，其中name为STRING类型，age为INT类型）。建表成功后插入测试数据至“struct_test”表中。操作如下：

1. 创建表。

   CREATE TABLE struct_test(id INT, info STRUCT<name:STRING,age:INT>) USING PARQUET;

2. 插入测试数据。

   INSERT INTO struct_test VALUES (8, struct('user1',23));

   INSERT INTO struct_test VALUES (9, struct('user2,25));

   INSERT INTO struct_test VALUES (10, struct('user3',26));

3. 查询结果。

   查询“struct_test”表中的所有数据。

 版权所有 © 华为技术有限公司

SELECT * FROM struct_test;

```
8{"name":"user1,"age":23}
10{"name":"user2,"age":26}
9{"name":"user3,"age":25}
```

查询“struct_test”表中的**name**和**age**数据。

SELECT id,info.name,info.age FROM struct_test;

```
8       user1   23
10      user2   26
9    user3    25
```

 版权所有 © 华为技术有限公司

13 自定义函数

13.1 创建函数

功能描述

DLI支持创建使用UDF和UDTF等自定义函数应用于Spark作业开发当中。

具体使用自定义函数端到端的开发指导可以参考：**Spark SQL作业使用UDF**和**Spark SQL作业使用UDTF**。

语法格式

```
CREATE FUNCTION [db_name.]function_name AS class_name
  [USING resource,...]

resource:
  : JAR file_uri
```

或

```
CREATE OR REPLACE FUNCTION [db_name.]function_name AS class_name
  [USING resource,...]

resource:
  : JAR file_uri
```

注意事项

- 如果在数据库中存在同名的函数，系统将会报错。
- 只支持Hive语法创建函数。
- 请注意避免该场景：如果创建的自定义函数F1指定类C1，程序包名JAR1，创建自定义函数F2也指定类C1，程序包JAR2，因为F2和F1使用相同的类名，导致功能相互冲突，影响作业执行。
- 如需使用UDF热加载功能请提交工单开通白名单。

关键字

- USING <resources>：需要加载的资源。可以是JAR、文件或者URI的列表。

 版权所有 © 华为技术有限公司

- OR REPLACE：支持自定义函数热加载功能。
  - 如果创建自定义函数时**不携带OR REPLACE**，则需要注意以下场景：

表 13-1 不携带 OR REPLACE 场景说明

序号	场景说明	场景举例	生效机制	操作影响
场景一	修改了原有程序包类的实现逻辑，重新创建的函数指定的JAR包名和类名保持和原有一致。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. 后续对J1包中函数实现做了逻辑修改，重新执行创建函数F2，指定类名C1，Jar包名J1。 **说明** 注意，如果步骤2继续使用函数名F1，则会因为函数名重复导致创建失败。这时可以考虑使用**OR REPLACE**，或者替换所有作业中的函数F1为F2。	需要重启Spark SQL队列后新创建的自定义函数F2生效	1. 需要重启Spark SQL队列，影响当前运行的作业。 2. 重启队列后，影响F1原有功能，F1的功能变为和F2一样。
场景二	在原有程序包类的基础上新增了类，新创建的函数指定为新增的类，包名不变。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. J1的Jar包中新增了类C2，C1保持不变，新创建自定义函数F2，指定类名C2，Jar包名J1。	需要重启Spark SQL队列后新创建的自定义函数F2生效	1. 需要重启Spark SQL队列，影响当前运行的作业。 2. 重启队列后，F1的功能不变。

 版权所有 © 华为技术有限公司

序号	场景说明	场景举例	生效机制	操作影响
场景三	原有程序包类的实现逻辑不变，重新打包程序包名。新创建的函数指定新JAR包名，类名保持不变。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. 重新打包Jar包为J2，功能逻辑不变。新创建的自定义函数F2，指定类名C1，Jar包名J2。	新创建的自定义函数F2立即生效	无影响。

- 如果创建自定义函数**携带OR REPLACE**，表示需要对已有的函数内容进行功能替换并实时生效。

⚠ 注意

- **该功能开启当前需要提交工单开通白名单。**
- 如果要在所有SQL队列上立即生效，需要分别选择SQL队列执行一遍：CREATE OR REPLACE xxx FUNCTION ...，否则没有执行的队列可能延迟0-12小时生效。
- 如果当前运行的作业中使用自定义函数F1，该F1函数指定类名C1，程序包名J1，作业运行了一半后，重新修改J1程序包逻辑，CREATE OR REPLACE FUNCTION F1后，后续作业使用新的F1函数逻辑，但是已执行完一半的作业使用旧的函数逻辑。

版权所有 © 华为技术有限公司

表 13-2 携带 OR REPLACE 场景说明

序号	场景说明	场景举例	生效机制	操作影响
场景一	修改了原有程序包类的实现逻辑，重新创建的函数指定的JAR包名和类名保持和原有一致。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. 后续对J1包中函数实现做了逻辑修改，重新执行创建函数F1，指定类名C1，Jar包名J1。	立即生效	F1的功能变成新修改的逻辑。

 版权所有 © 华为技术有限公司

序号	场景说明	场景举例	生效机制	操作影响
场景二	在原有程序包类的基础上新增了类，新创建的函数指定为新增的类，包名不变。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. J1的Jar包中新增了类C2，C1保持不变，新创建自定义函数F2，指定类名C2，Jar包名J1。因为是第一次创建指定C2的函数，这时该F2函数不能立即生效。	两种生效方式 • CREATE OR REPLACE *F2* FUNCTION创建语句再执行一次生效 • 重启Spark SQL队列后生效	如果是选择重启Spark SQL队列，则会导致当前运行的作业受影响。

版权所有 © 华为技术有限公司

序号	场景说明	场景举例	生效机制	操作影响
场景三	修改了原有程序包类的类名，新创建的函数指定为新类名，包名不变。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. J1的Jar包中修改原C1为C2，C1已经不存在，新创建自定义函数F2，指定类名C2，Jar包名J1。	立即生效	F1功能失效，F2功能立即生效。
场景四	原有程序包类的实现逻辑不变，重新打包程序包名。新创建的函数指定新JAR包名，类名保持不变。	1. 在Spark SQL队列下已创建自定义函数F1，指定类名C1，Jar包名J1。 2. 重新打包Jar包为J2，功能逻辑不变。新创建的自定义函数F2，指定类名C1，Jar包名J2。	立即生效	无影响。

示例

创建函数mergeBill。

```
CREATE FUNCTION mergeBill AS 'com.xxx.hiveudf.MergeBill'
  using jar 'obs://onlyci-7/udf/MergeBill.jar';
```

13.2 删除函数

功能描述

删除函数。

 版权所有 © 华为技术有限公司

语法格式

```
DROP [TEMPORARY] FUNCTION [IF EXISTS] [db_name.] function_name;
```

关键字

- TEMPORARY：所删除的函数是否为临时函数。
- IF EXISTS：所删除的函数不存在时使用，可避免系统报错。

注意事项

- 删除一个已存在的函数。如果要删除的函数不存在，则系统报错。
- 只支持HIVE语法。

示例

删除函数mergeBill。

```
DROP FUNCTION mergeBill;
```

13.3 显示函数详情

功能描述

查看指定函数的相关信息。

语法格式

```
DESCRIBE FUNCTION [EXTENDED] [db_name.] function_name;
```

关键字

EXTENDED：显示扩展使用信息。

注意事项

返回已有函数的元数据（实现类和用法），如果函数不存在，则系统报错。

示例

查看函数mergeBill的相关信息。

```
DESCRIBE FUNCTION mergeBill;
```

13.4 显示所有函数

功能描述

查看当前工程下所有的函数。

语法格式

```
SHOW [USER|SYSTEM|ALL] FUNCTIONS ([LIKE] regex | [db_name.] function_name);
```

 版权所有 © 华为技术有限公司

其中regex为正则表达式，可以参考如下表13-3参数样例。

表 13-3 regex 参数举例说明

regex表达式	匹配含义
'xpath*'	表示匹配所有xpath开头的函数名。 例如：SHOW FUNCTIONS LIKE 'xpath*' ; 表示可以匹配到：xpath、xpath_int、xpath_string等等xpath开头的函数。
'x[a-z]+'	表示匹配以x开头，后面是a到z范围的一个到多个字符的函数名。如可以匹配到：xpath、xtest等。
'x.*h'	匹配以x开头，h结尾，中间为一个或多个字符的函数名。如可以匹配到：xpath、xtesth等。

其他更多正则表达式的使用，可参考官网说明。

关键字

LIKE：此限定符仅为兼容性而使用，没有任何实际作用。

注意事项

显示与给定正则表达式或函数名匹配的函数。如果未提供正则表达式或名称，则显示所有函数。如果声明了USER或SYSTEM，那么将分别显示用户定义的Spark SQL函数和系统定义的Spark SQL函数。

示例

查看当前的所有函数。

```
SHOW FUNCTIONS;
```

 版权所有 © 华为技术有限公司

14 内置函数

14.1 日期函数

14.1.1 日期函数概览

DLI所支持的日期函数如表14-1所示。

表 14-1 日期/时间函数

函数	命令格式	返回值	功能简介
add_months	add_months(string start_date, int num_months)	STRING	返回start_date在num_months个月之后的date。
current_date	current_date()	DATE	返回当前日期，如2016-07-04。
current_timestamp	current_timestamp()	TIMESTAMP	返回TIMESTAMP类型的时间戳。
date_add	date_add(string startdate, int days)	STRING或DATE	给定时间，在此基础上加上指定的时间段。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
dateadd	dateadd(string date, bigint delta, string datepart)	STRING或DATE	dateadd函数用于按照指定的单位datepart和幅度delta修改date的值。 date：必填。日期值，string类型。 使用的时间格式为yyyy-mm-dd hh:mi:ss，例如2021-08-28 00:00:00。 delta：必填。修改幅度，BIGINT类型。 datepart：必填。指定修改的单位，STRING类型常量。
date_sub	date_sub(string startdate, int days)	STRING	给定时间，在此基础上减去指定的时间段。
date_format	date_format(string date, string format)	STRING	用于将date按照format指定的格式转换为字符串。
datediff	datediff(string date1, string date2)	BIGINT	datediff函数用于计算两个时间date1、date2的日期差值。
datediff1	datediff1(string date1, string date2, string datepart)	BIGINT	datediff1函数用于计算两个时间date1、date2的差值，将差值以指定的时间单位datepart表示。
datepart	datepart（string date，string datepart）	BIGINT	取日期中符合指定时间单位的字段值。
datetrunc	datetrunc (string date, string datepart)	STRING	datetrunc函数用于计算将日期date按照datepart指定的时间单位进行截取后的日期值。 date：必填。格式为yyyy-mm-dd或yyyy-mm-dd hh:mi:ss。 datepart：必填。STRING类型常量。支持扩展的日期格式。
day/dayofmonth	day(string date)、dayofmonth(string date)	INT	返回指定时间的日期。
from_unixtime	from_unixtime(bigint unixtime)	STRING	将时间戳转换为时间格式，格式为“yyyy-MM-dd HH:mm:ss”或“yyyyMMddHHmmss.uuuuuu”。 例如：select FROM_UNIXTIME(1608135036,'yyyy-MM-dd HH:mm:ss')

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
from_utc_timestamp	from_utc_timestamp(string timestamp, string timezone)	TIMESTAMP	将UTC的时间戳转化为timezone所对应的时间戳。
getdate	getdate()	STRING	获取当前系统时间。
hour	hour(string date)	INT	返回指定时间的小时，范围为0到23。
isdate	isdate(string date , string format)	BOOLEAN	date：必填。STRING类型。会隐式转换为STRING类型后参与运算。 format：必填。STRING类型常量，不支持日期扩展格式。如果format中出现多余的格式串，则只取第一个格式串对应的日期数值，其余的会被视为分隔符。例如isdate("1234-yyyy", "yyyy-yyyy")，会返回True。
last_day	last_day(string date)	DATE	返回date所在月份的最后一天，格式为yyyy-MM-dd，如2015-08-31。
lastday	lastday(string date)	STRING	lastday函数用于返回date所在月的最后一天，返回值STRING类型，格式为yyyy-mm-dd hh:mi:ss。
minute	minute(string date)	INT	返回指定时间的分钟，范围为0到59。
month	month(string date)	INT	返回指定时间的月份，范围为1至12月。
months_between	months_between(string date1, string date2)	DOUBLE	返回date1与date2之间的月份差。
next_day	next_day(string start_date, string day_of_week)	DATE	返回start_date之后最接近day_of_week的日期，格式为yyyy-MM-dd，day_of_week表示一周内的星期（如Monday、FRIDAY）。
quarter	quarter(string date)	INT	返回该date/timestamp/string所在的季度，如quarter('2015-04-01')=2。
second	second(string date)	INT	返回指定时间的秒，范围为0到59。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
to_char	to_char(string date, string format)	STRING	将日期按照指定格式转换为字符串。
to_date	to_date(string timestamp)	DATE	返回时间中的年月日，例如：to_date("1970-01-01 00:00:00") = "1970-01-01"
to_date1	to_date1(string date, string format)	STRING	返回STRING类型，格式为yyyy-mm-dd hh:mi:ss。date或format值为NULL时，返回NULL。 将指定格式的字符串转换为日期值。
to_utc_timestamp	to_utc_timestamp(string timestamp, string timezone)	TIMESTAMP	将timezone所对应的时间戳转换为UTC的时间戳。
trunc	trunc(string date, string format)	DTAE	将date按照特定的格式进行清零操作，支持格式为MONTH/MON/MM, YEAR/YYYY/YY，如trunc('2015-03-17', 'MM') = 2015-03-01。
unix_timestamp	unix_timestamp(string timestamp, string pattern)	BIGINT	如果不带参数的调用，返回一个Unix时间戳（从“1970-01-01 00:00:00”到现在的秒数）为无符号整数。
weekday	weekday (string date)	INT	返回日期值是当前周的第几天。
weekofyear	weekofyear(string date)	INT	返回指定日期是一年中的第几周，范围为0到53。
year	year(string date)	INT	返回指定日期中的年份。

14.1.2 add_months

add_months函数用于计算日期值增加指定月数后的日期。即start_date在num_months个月之后的date。

命令格式

```
add_months(string start_date, int num_months)
```


版权所有 © 华为技术有限公司

参数说明

表 14-2 参数说明

参数	是否必选	参数类型	说明
start_date	是	DATE或STRING	代表起始日期。 支持以下格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
num_months	是	INT	代表需要增加月的数量。

返回值说明

返回开始日期startdate增加num_months个月后的日期，返回值格式为yyyy-mm-dd。

返回值date类型的日期值。

说明

- startdate非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- startdate为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- startdate值为NULL时，返回报错。
- num_months值为NULL时，返回NULL。

示例代码

返回2023-05-26。

```
select add_months('2023-02-26',3);
```

返回2023-05-14。

```
select add_months('2023-02-14 21:30:00',3);
```

返回NULL。

```
select add_months('20230815',3);
```

返回NULL。

```
select add_months('2023-08-15 20:00:00',null);
```

14.1.3 current_date

current_date函数用于返回当前日期值。返回值格式为yyyy-mm-dd。

相似函数：**getdate**，getdate函数用于返回当前系统时间。返回值格式为yyyy-mm-dd hh:mi:ss。

命令格式

```
current_date()
```

 版权所有 © 华为技术有限公司

参数说明

无

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd

示例代码

返回2023-08-16。

```
select current_date();
```

14.1.4 current_timestamp

CURRENT_TIMESTAMP函数用于返回当前时间戳。

命令格式

```
current_timestamp()
```

参数说明

无

返回值说明

返回TIMESTAMP类型的时间戳。

示例代码

返回1692002816300。

```
select current_timestamp();
```

14.1.5 date_add

date_add函数用于计算按照days幅度递增startdate日期的天数。

如需要获取当前日期基础上指定变动幅度的日期，可结合**current_date**或**getdate**函数共同使用。

请注意date_add函数与**date_sub**函数逻辑反。

命令格式

```
date_add(string startdate, int days)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-3 参数说明

参数	是否必选	参数类型	说明
start_date	是	DATE 或 STRING	代表起始日期。 支持以下格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
days	是	BIGINT	代表需要增加天的数量。 • days大于0，则为增加天数。 • days小于0，则减去天数。 • days等于0，即加0天，日期不变。 • days值为NULL时，返回NULL。

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd。

说明

- startdate非DATE或STRING类型时，返回报错，错误信息：data type mismatch；
- startdate为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL；
- startdate值为NULL时，返回NULL。
- days值为NULL时，返回NULL。

示例代码

返回2023-03-01。加1天，结果超出当年2月份的最后1天，实际值为下个月的第1天。

```
select date_add('2023-02-28 00:00:00', 1);
```

返回2023-02-27。减1天。

```
select date_add(date '2023-02-28', -1);
```

返回2023-03-20。

```
select date_add('2023-02-28 00:00:00', 20);
```

假设当前时间为2023-08-14 16:00:00，返回2023-08-13。

```
select date_add(getdate(),-1);
```

返回NULL。

```
select date_add('2023-02-28 00:00:00', null);
```

 版权所有 © 华为技术有限公司

14.1.6 dateadd

dateadd函数用于按照指定的单位datepart和幅度delta修改date的值。

如需要获取当前日期基础上指定变动幅度的日期，可结合current_date或getdate函数共同使用。

命令格式

```
dateadd(string date, bigint delta, string datepart)
```

参数说明

表 14-4 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表起始日期。 支持以下格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
delta	是	BIGINT	代表修改幅度。
datepart	是	STRING	代表修改的时间单位。 参数datepart支持扩展的日期格式：年-year、月-month或-mon、日-day和小时-hour。 • yyyy代表年份。 • MM代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

返回值说明

返回STRING类型的日期值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。
- delta或datepart值为NULL时，返回NULL。

版权所有 © 华为技术有限公司

示例代码

返回2023-08-15 17:00:00。加1天。

```
select dateadd( '2023-08-14 17:00:00', 1, 'dd');
```

返回2025-04-14 17:00:00。加20个月，月份溢出，年份加1。

```
select dateadd('2023-08-14 17:00:00', 20, 'mm');
```

返回2023-09-14 17:00:00。

```
select dateadd('2023-08-14 17:00:00', 1, 'mm');
```

返回2023-09-14。

```
select dateadd('2023-08-14', 1, 'mm');
```

假设当前时间为2023-08-14 17:00:00，返回2023-08-13 17:00:00。

```
select dateadd(getdate(),-1,'dd');
```

返回NULL。

```
select dateadd(date '2023-08-14', 1, null);
```

14.1.7 date_sub

date_sub函数按照days幅度递减startdate日期的天数。

如需要获取当前日期基础上指定变动幅度的日期，可结合**current_date**或**getdate**函数共同使用。

请注意date_sub函数与**date_add**函数逻辑反。

命令格式

```
date_sub(string startdate, int days)
```

参数说明

表 14-5 参数说明

参数	是否必选	参数类型	说明
start_date	是	DATE 或 STRING	代表起始日期。 支持以下格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

 版权所有 © 华为技术有限公司

参数	是否必选	参数类型	说明
days	是	BIGINT	代表需要减少的天数。 ● 当days大于0时，从startdate中减少指定的天数。 ● 当days小于0时，向startdate中增加指定的天数。 ● 当days等于0时，即加0天，日期不变。 ● 当days值为NULL时，返回NULL。

返回值说明

返回DATE类型的日期值。

说明

- startdate非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- startdate为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。
- format值为NULL时，返回NULL。

示例代码

返回2023-08-12。减2天。

```
select date_sub('2023-08-14 17:00:00', 2);
```

返回2023-08-15。增1天。

```
select date_sub(date'2023-08-14', -1);
```

假设当前时间为2023-08-14 17:00:00，返回2023-08-13。

```
select date_sub(getdate(),1);
```

返回NULL。

```
select date_sub('2023-08-14 17:00:00', null);
```

14.1.8 date_format

date_format函数用于将date按照format指定的格式转换为字符串。

命令格式

```
date_format(string date, string format)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-6 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表待转换的日期。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
format	是	STRING	代表需要转换的格式。 格式为代表年月日时分秒的时间单位与任意字符的组合，其中： • yyyy代表年份。 • MM代表月份。 • dd代表天。 • HH代表24小时制时。 • hh代表12小时制时。 • mm代表分钟。 • ss代表秒。

返回值说明

按指定类型返回STRING类型的日期。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。
- format值为NULL时，返回NULL。

示例代码

返回2023-08-14。

```
select date_format('2023-08-14','yyyy-MM-dd');
```

返回2023-08。

```
select date_format('2023-08-14','yyyy-MM')
```

返回20230814。

```
select date_format('2023-08-14','yyyyMMdd')
```

 版权所有 © 华为技术有限公司

14.1.9 datediff

datediff函数用于计算两个时间date1、date2的日期差值。

相似函数：**datediff1**，datediff1函数用于计算两个时间date1、date2的差值，将差值以指定的时间单位datepart表示。

命令格式

```
datediff(string date1, string date2)
```

参数说明

表 14-7 参数说明

参数	是否必选	参数类型	说明
date1	是	DATE 或 STRING	计算两个时间date1、date2的日期差值中的被减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
date2	是	DATE 或 STRING	计算两个时间date1、date2的日期差值的减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回BIGINT类型。

说明

- date1、date2非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date1、date2为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- 如果date1小于date2，返回值为负数。
- date1或date2值为NULL时，返回NULL。

示例代码

返回10。

```
select datediff('2023-06-30 00:00:00', '2023-06-20 00:00:00');
```

返回11。

```
select datediff(date '2023-05-21', date '2023-05-10');
```

 版权所有 © 华为技术有限公司

返回NULL。

```
select datediff(date '2023-05-21', null);
```

14.1.10 datediff1

datediff1函数用于计算两个时间date1、date2的差值，将差值以指定的时间单位datepart表示。

相似函数：**datediff**，datediff函数用于计算两个时间date1、date2的日期差值，不支持指定返回的时间单位。

命令格式

```
datediff1(string date1, string date2, string datepart)
```

参数说明

表 14-8 参数说明

参数	是否必选	参数类型	说明
date1	是	DATE 或 STRING	计算两个时间date1、date2的日期差值中的被减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
date2	是	DATE 或 STRING	计算两个时间date1、date2的日期差值的减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
datepart	是	STRING	代表需要返回的时间单位。 参数datepart支持扩展的日期格式：年-year、月-month或-mon、日-day和小时-hour。 • yyyy代表年份。 • MM代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

版权所有 © 华为技术有限公司

返回值说明

返回BIGINT类型。

说明

- date1、date2非DATE或STRING类型时，返回报错，错误信息：data type mismatch；
- date1、date2为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL；
- 如果date1小于date2，返回值为负数。
- date1或date2值为NULL时，返回NULL。
- datepart值为NULL时，返回NULL。

示例代码

返回14400。

```
select datediff1('2023-06-30 00:00:00', '2023-06-20 00:00:00', 'mi');
```

返回10。

```
select datediff1(date '2023-06-21', date '2023-06-11', 'dd');
```

返回NULL。

```
select datediff1(date '2023-05-21', date '2023-05-10', null);
```

返回NULL。

```
select datediff1(date '2023-05-21', null, 'dd');
```

14.1.11 datepart

datepart函数用于计算日期date中符合指定时间单位datepart的值。

命令格式

```
datepart（string date，string datepart）
```

参数说明

表 14-9 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表起始日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

 版权所有 © 华为技术有限公司

参数	是否必选	参数类型	说明
datepart	是	STRING	代表需要返回的时间单位。 参数datepart支持扩展的日期格式：年-year、月-month或-mon、日-day和小时-hour。 • yyyy代表年份。 • MM代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

返回值说明

返回BIGINT类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch；
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL；
- datepart值为NULL时，返回NULL。
- datepart值为NULL时，返回NULL。

示例代码

返回2023。

```
select datepart(date '2023-08-14 17:00:00', 'yyyy');
```

返回2023。

```
select datepart('2023-08-14 17:00:00', 'yyyy');
```

返回59。

```
select datepart('2023-08-14 17:59:59', 'mi')
```

返回NULL。

```
select datepart(date '2023-08-14', null);
```

14.1.12 datetrunc

datetrunc函数用于计算将日期date按照datepart指定的时间单位进行截取后的日期值。

截取datepart之前的部分，除截取的部分外自动填充为默认值。可参考**示例代码**。

命令格式

```
datetrunc (string date, string datepart)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-10 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表起始日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
datepart	是	STRING	代表需要返回的时间单位。 参数datepart支持扩展的日期格式：年-year、月-month或-mon、日-day和小时-hour。 • yyyy代表年份。 • MM代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

返回值说明

返回DATE或STRING类型的日期值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch；
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL；
- datepart值为NULL时，返回NULL。
- datepart值为时、分、秒时，按照天截取返回

示例代码

静态数据示例

返回2023-01-01 00:00:00。

```
select datetrunc('2023-08-14 17:00:00', 'yyyy');
```

返回2023-08-01 00:00:00。

```
select datetrunc('2023-08-14 17:00:00', 'month');
```

返回2023-08-14。

```
select datetrunc('2023-08-14 17:00:00', 'DD');
```

返回2023-01-01。

 版权所有 © 华为技术有限公司

```
select datetrunc('2023-08-14', 'yyyy');
```

返回2023-08-14 17:00:00。

```
select datetrunc('2023-08-14 17:11:11', 'hh');
```

返回NULL。

```
select datetrunc('2023-08-14', null);
```

14.1.13 day/dayofmonth

day函数用于返回指定日期的天。

命令格式

```
day(string date)、dayofmonth(string date)
```

参数说明

表 14-11 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回1。

```
select day('2023-08-01');
```

返回NULL。

```
select day('20230816');
```

返回NULL。

```
select day(null);
```

 版权所有 © 华为技术有限公司

14.1.14 from_unixtime

from_unixtime函数用于计算将数字型的UNIX值代表的时间戳转换为日期值。

命令格式

```
from_unixtime(bigint unixtime)
```

参数说明

表 14-12 参数说明

参数	是否必选	参数类型	说明
unixtime	是	BIGINT	UNIX格式的时间戳。代表需要转换的时间戳 此处参数应填正常UNIX格式时间戳前十位。

返回值说明

返回STRING类型的日期值，格式为yyyy-mm-dd hh:mi:ss。

说明

unixtime值为NULL时，返回NULL。

示例代码

返回2023-08-16 09:39:57。

```
select from_unixtime(1692149997);
```

返回NULL。

```
select from_unixtime(NULL);
```

表数据示例

```
select unixdate, from_unixtime(unixdate) as timestamp_from_unixtime from database_t;
输出：
+-----------------+-----------------------------+
| unixdate        | timestamp_from_unixtime     |
+-----------------+-----------------------------+
| 1690944759224   | 2023-08-02 10:52:39         |
| 1690944999811   | 2023-08-02 10:56:39         |
| 1690945005458   | 2023-08-02 10:56:45         |
| 1690945011542   | 2023-08-02 10:56:51         |
| 1690945023151   | 2023-08-02 10:57:03         |
+-----------------+-----------------------------+
```

14.1.15 from_utc_timestamp

from_utc_timestamp函数用于计算将UTC的时间戳转化为timezone所对应的UNIX格式的时间戳。

 版权所有 © 华为技术有限公司

命令格式

```
from_utc_timestamp(string timestamp, string timezone)
```

参数说明

表 14-13 参数说明

参数	是否必选	参数类型	说明
timestamp	是	DATE STRING TINYINT SMALLINT INT BIGINT	代表待转换的时间。 DATE或STRING类型的日期值，或TINYINT、SMALLINT、INT或BIGINT类型的时间戳。 格式： yyyy-mm-dd yyyy-mm-dd hh:mi:ss yyyy-mm-dd hh:mi:ss.ff3
timezone	是	STRING	代表需要转换的目标时区。

返回值说明

返回TIMESTAMP类型的时间戳。

说明

- timestamp非DATE或STRING类型时，返回报错，错误信息：data type mismatch；
- timestamp为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL；
- timestamp值为NULL时，返回NULL。
- timezone值为NULL时，返回NULL。

示例代码

返回1691978400000（代表2023-08-14 10:00:00）。

```
select from_utc_timestamp('2023-08-14 17:00:00','PST');
```

返回1691917200000（代表2023-08-13 17:00:00）。

```
select from_utc_timestamp(date '2023-08-14 00:00:00','PST');
```

返回NULL。

```
select from_utc_timestamp('2023-08-13',null);
```

14.1.16 getdate

getdate函数用于返回当前系统时间。返回值格式为yyyy-mm-dd hh:mi:ss。

相似函数：**current_date**，current_date函数用于返回当前日期值。返回值格式为yyyy-mm-dd。

 版权所有 © 华为技术有限公司

命令格式

```
getdate()
```

参数说明

无

返回值说明

返回STRING类型的日期值，格式为yyyy-mm-dd hh:mi:ss。

示例代码

假设当前时间为2023-08-10 10:54:00，返回2023-08-10 10:54:00。

```
select getdate();
```

14.1.17 hour

hour函数用于返回指定时间的小时，范围为0到23。

命令格式

```
hour(string date)
```

参数说明

表 14-14 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型的值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回10。

 版权所有 © 华为技术有限公司

```
select hour('2023-08-10 10:54:00');
```

返回12。

```
select hour('12:00:00');
```

返回NULL。

```
select hour('20230810105600');
```

返回NULL。

```
select hour(null);
```

14.1.18 isdate

isdate函数用于判断一个日期字符串能否根据指定的格式转换为一个日期值。

命令格式

```
isdate(string date , string format)
```

参数说明

表 14-15 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要判断的字符串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，会隐式转换为STRING类型后参与运算 格式为任意字符串。
format	是	STRING	代表需要转换的目标日期格式。 STRING类型常量，不支持日期扩展格式。 format:格式为代表年月日时分秒的时间单位与任意字符的组合，其中： • yyyy代表年份。 • mm代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

返回值说明

返回BOOLEAN类型的值。

 版权所有 © 华为技术有限公司

说明

date或format值为NULL时，返回NULL。

示例代码

返回true。

```
select isdate('2023-08-10','yyyy-mm-dd');
```

返回false。

```
select isdate(123456789,'yyyy-mm-dd');
```

14.1.19 last_day

last_day函数用于返回date所在月份的最后一天。

相似函数：**lastday**，lastday函数用于返回date所在月的最后一天，截取到天，时分秒部分为00:00:00。

命令格式

```
last_day(string date)
```

参数说明

表 14-16 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回2023-08-31。

```
select last_day('2023-08-15');
```

 版权所有 © 华为技术有限公司

返回2023-08-31。

```
select last_day('2023-08-10 10:54:00');
```

返回NULL。

```
select last_day('20230810');
```

14.1.20 lastday

lastday函数用于返回date所在月的最后一天，截取到天，时分秒部分为00:00:00。

相似函数：**last_day**，last_day函数用于返回date所在月份的最后一天。返回值格式为：yyyy-mm-dd。

命令格式

```
lastday(string date)
```

参数说明

表 14-17 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回STRING类型的日期值。格式为yyyy-mm-dd hh:mi:ss。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回2023-08-31。

```
select lastday('2023-08-10');
```

返回2023-08-31 00:00:00。

```
select lastday ('2023-08-10 10:54:00');
```

返回NULL。

```
select lastday (null);
```

 版权所有 © 华为技术有限公司

14.1.21 minute

minute函数用于返回指定时间的分钟，范围为0到59。

命令格式

```
minute(string date)
```

参数说明

表 14-18 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回54。

```
select minute('2023-08-10 10:54:00');
```

返回54。

```
select minute('10:54:00');
```

返回NULL。

```
select minute('20230810105400');
```

返回NULL。

```
select minute(null);
```

14.1.22 month

month函数用于返回指定时间的月份，范围为1至12月。

 版权所有 © 华为技术有限公司

命令格式

```
month(string date)
```

参数说明

表 14-19 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 date取值为STRING类型格式时，至少要包含yyyy-mm-dd且不含多余的字符串。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回8。

```
select month('2023-08-10 10:54:00');
```

返回NULL。

```
select month('20230810');
```

返回NULL。

```
select month(null);
```

14.1.23 months_between

months_between函数用于返回date1与date2之间的月份差。

命令格式

```
months_between(string date1, string date2)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-20 参数说明

参数	是否必选	参数类型	说明
date1	是	DATE 或 STRING	代表被减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
date2	是	DATE 或 STRING	代表减数。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回DOUBLE类型的值。

说明

- date1、date2非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date1、date2为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- 当date1晚于date2时，返回值为正。当date2晚于date1时，返回值为负。
- 当date1和date2分别对应两个月的最后一天，返回整数月；否则计算方式为date1减去date2的天数除以31天。
- date1或date2值为NULL时，返回NULL。

示例代码

返回0.0563172。

```
select months_between('2023-08-16 10:54:00', '2023-08-14 17:00:00');
```

返回0.06451613。

```
select months_between('2023-08-16','2023-08-14');
```

返回NULL。

```
select months_between('2023-08-16',null);
```

14.1.24 next_day

next_day函数用于返回start_date之后最接近day_of_week的日期。

命令格式

```
next_day(string start_date, string day_of_week)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-21 参数说明

参数	是否必选	参数类型	说明
start_date	是	DATE 或 STRING	代表需要处理的日期。 start_date取值为STRING类型格式时，至少要包含yyyy-mm-dd且不含多余的字符串。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
day_of_week	是	STRING	一个星期前2个或3个字母，或者一个星期的全名。例如TU代表星期二。

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd。

说明

- start_date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- start_date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- start_date值为NULL时，返回NULL。
- day_of_week值为NULL时，返回NULL。

示例代码

返回2023-08-22。

```
select next_day('2023-08-16','TU');
```

返回2023-08-22。

```
select next_day('2023-08-16 10:54:00','TU');
```

返回2023-08-23。

```
select next_day('2023-08-16 10:54:00','WE');
```

返回NULL。

```
select next_day('20230816','TU');
```

返回NULL。

```
select next_day('20230816 20:00:00',null);
```

14.1.25 quarter

quarter函数用于返回该date所在的季度，范围为1~4。

 版权所有 © 华为技术有限公司

命令格式

```
quarter(string date)
```

参数说明

表 14-22 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回3。

```
select quarter('2023-08-16 10:54:00');
```

返回3。

```
select quarter('2023-08-16');
```

返回NULL。

```
select quarter(null);
```

14.1.26 second

second函数用于返回指定时间的秒，范围为0到59。

命令格式

```
second(string date)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-23 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回36。

```
select second('2023-08-16 10:54:36');
```

返回36。

```
select second('10:54:36');
```

返回NULL。

```
 select second('20230816105436');
```

返回NULL。

```
select second(null);
```

14.1.27 to_char

to_char函数用于将日期按照指定格式转换为字符串。

命令格式

```
to_char(string date, string format)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-24 参数说明

参数	是否必选	参数类型	说明
date	是	DATE 或 STRING	代表需要处理的日期。 格式为： ● yyyy-mm-dd ● yyyy-mm-dd hh:mi:ss ● yyyy-mm-dd hh:mi:ss.ff3
format	是	STRING	代表需要转换的目标日期格式。 STRING类型常量，不支持日期扩展格式。 format:格式为代表年月日时分秒的时间单位与任意字符的组合，其中： ● yyyy代表年份。 ● mm代表月份。 ● dd代表天。 ● hh代表小时。 ● mi代表分钟。 ● ss代表秒。

返回值说明

返回STRING类型。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- format值为NULL时，返回NULL。

示例代码

返回静态数据示例2023-08*16。

```
select to_char('2023-08-16 10:54:36', '静态数据示例yyyy-mm*dd');
```

返回20230816。

```
select to_char('2023-08-16 10:54:36', 'yyyymmdd');
```

返回NULL。

```
select to_char('静态数据示例2023-08-16', '静态数据示例yyyy-mm*dd');
```

返回NULL。

```
select to_char('20230816', 'yyyy');
```

 版权所有 © 华为技术有限公司

返回NULL。

```
select to_char('2023-08-16 10:54:36', null);
```

14.1.28 to_date

to_date函数用于返回时间中的年月日。

相似函数：**to_date1**，to_date1函数用于将指定格式的字符串转换为日期值，支持指定转换的日期格式。

命令格式

```
to_date(string timestamp)
```

参数说明

表 14-25 参数说明

参数	是否必选	参数类型	说明
timestamp	是	DATE STRING	代表待处理的时间。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd。

说明

- timestamp非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- timestamp为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。

示例代码

返回2023-08-16。

```
select to_date('2023-08-16 10:54:36');
```

返回NULL。

```
select to_date(null);
```

14.1.29 to_date1

to_date1函数用于将指定格式的字符串转换为日期值。

相似函数：**to_date**，to_date函数用于返回时间中的年月日，不支持指定转换的日期格式。

 版权所有 © 华为技术有限公司

命令格式

```
to_date1(string date, string format)
```

参数说明

表 14-26 参数说明

参数	是否必选	参数类型	说明
date	是	STRING	要转换的字符串。 格式： ● yyyy-mm-dd ● yyyy-mm-dd hh:mi:ss ● yyyy-mm-dd hh:mi:ss.ff3
format	是	STRING	代表需要转换的日期格式。 STRING类型常量，不支持日期扩展格式。 format:格式为代表年月日时分秒的时间单位与任意字符的组合，其中： ● yyyy代表年份。 ● mm代表月份。 ● dd代表天。 ● hh代表小时。 ● mi代表分钟。 ● ss代表秒。

返回值说明

返回STRING类型的日期值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。
- format值为NULL时，返回yyyy-mm-dd格式的日期值。

示例代码

返回2023-08-16 10:54:36

```
select to_date1('2023-08-16 10:54:36','yyyy-mm-dd hh:mi:ss');
```

返回2023-08-16 00:00:00。

```
select to_date1('2023-08-16','yyyy-mm-dd');
```

返回NULL。

 版权所有 © 华为技术有限公司

```
select to_date1(null);
```

返回2023-08-16。

```
select to_date1('2023-08-16 10:54:36');
```

14.1.30 to_utc_timestamp

to_utc_timestamp函数用于将timezone所对应的时间戳转换为UTC的时间戳。

命令格式

```
to_utc_timestamp(string timestamp, string timezone)
```

参数说明

表 14-27 参数说明

参数	是否必选	参数类型	说明
timestamp	是	DATE STRING TINYINT SMALLINT INT BIGINT	代表待处理的时间。 DATE或STRING类型的日期值，或TINYINT、SMALLINT、INT或BIGINT类型的时间戳。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
timezone	是	STRING	代表需要转换的目标时区。

返回值说明

返回BIGINT类型值。

说明

- timestamp非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- timestamp为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- timestamp值为NULL时，返回NULL。
- timezone值为NULL时，返回NULL。

示例代码

返回1692028800000。

```
select to_utc_timestamp('2023-08-14 17:00:00','PST');
```

返回null。

```
select to_utc_timestamp(null);
```

 版权所有 © 华为技术有限公司

14.1.31 trunc

trunc函数用于将date按照特定的格式进行清零操作。

清零操作即返回默认值，年、月、日的默认值为01，时、分、秒、毫秒 的默认值为00。

命令格式

```
trunc(string date, string format)
```

参数说明

表 14-28 参数说明

参数	是否必选	参数类型	说明
date	是	DATE或STRING	需要处理的日期。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
format	是	STRING	代表需要转换的目标日期格式。 format:格式为代表年月日时分秒的时间单位与任意字符的组合，其中： • yyyy代表年份。 • MM代表月份。

返回值说明

返回DATE类型的日期值，格式为yyyy-mm-dd。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。
- format值为NULL时，返回NULL。

示例代码

返回2023-08-01。

```
select trunc('2023-08-16', 'MM');
```

返回2023-08-01。

```
select trunc('2023-08-16 10:54:36', 'MM');
```

返回NULL。

 版权所有 © 华为技术有限公司

```
select trunc(null, 'MM');
```

14.1.32 unix_timestamp

unix_timestamp函数用于将日期值转化为数字型的UNIX格式的日期值。

函数返回值将返回正常UNIX格式时间戳前十位。

命令格式

```
unix_timestamp(string timestamp, string pattern)
```

参数说明

表 14-29 参数说明

参数	是否必选	参数类型	说明
timestamp	否	DATE或STRING	代表待转换的日期值。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3
pattern	否	STRING	代表需要转换的格式。 pattern为空时，默认为yyyy-MM-dd hh:mm:ss格式。 format:格式为代表年月日时分秒的时间单位与任意字符的组合，其中： • yyyy代表年份。 • MM代表月份。 • dd代表天。 • hh代表小时。 • mi代表分钟。 • ss代表秒。

返回值说明

返回BIGINT类型的值。

说明

- timestamp值为NULL时，返回NULL。
- timestamp和pattern都为空时，返回从“1970-01-01 00:00:00”到现在的秒数代表的时间戳。

示例代码

返回1692149997。

 版权所有 © 华为技术有限公司

```
select unix_timestamp('2023-08-16 09:39:57')
```

假设当前系统时间为2023-08-16 10:23:16，返回1692152596。

```
select unix_timestamp();
```

返回1692115200（即2023-08-16 00:00:00）。

```
select unix_timestamp("2023-08-16 10:56:45", "yyyy-MM-dd");
```

表数据示例

```
select timestamp1, unix_timestamp(timestamp1) as date1_unix_timestamp, timestamp2,
unix_timestamp(datetime1) as date2_unix_timestamp, timestamp3, unix_timestamp(timestamp1) as
date3_unix_timestamp from database_t;输出：
+------------+------------------------+----------------------+---------------------- --
+-----------------------------------+---------------------------+
| timestamp1| date1_unix_timestamp | timestamp2              | date2_unix_timestamp |
timestamp3                          | date3_unix_timestamp      |
+------------+------------------------+----------------------+------------------------
+-----------------------------------+---------------------------+
| 2023-08-02 | 1690905600000          | 2023-08-02 11:09:14 | 1690945754793          | 2023-01-11
00:00:00.123456789 | 1673366400000              |
| 2023-08-03 | 1690992000000          | 2023-08-02 11:09:31 | 1690945771994          | 2023-02-11
00:00:00.123456789 | 1676044800000              |
| 2023-08-04 | 1691078400000          | 2023-08-02 11:09:41 | 1690945781270          | 2023-03-11
00:00:00.123456789 | 1678464000000              |
| 2023-08-05 | 1691164800000          | 2023-08-02 11:09:48 | 1690945788874          | 2023-04-11
00:00:00.123456789 | 1681142400000              |
| 2023-08-06 | 1691251200000          | 2023-08-02 11:09:59 | 1690945799099          | 2023-05-11
00:00:00.123456789 | 1683734400000              |
+------------+------------------------+----------------------+------------------------
+-----------------------------------+---------------------------+
```

14.1.33 weekday

weekday函数用于返回日期值是当前周的第几天。

命令格式

```
weekday (string date)
```

参数说明

表 14-30 参数说明

参数	是否必选	参数类型	说明
date	是	DATE或STRING	需要处理的日期。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型的值。

 版权所有 © 华为技术有限公司

说明

- 周一作为一周的第一天，返回值为0。其他日期依次递增，周日返回6。
- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回2。

```
select weekday ('2023-08-16 10:54:36');
```

返回NULL。

```
select weekday (null);
```

14.1.34 weekofyear

weekofyear函数用于返回指定日期是一年中的第几周，范围为0到53。

命令格式

```
weekofyear(string date)
```

参数说明

表 14-31 参数说明

参数	是否必选	参数类型	说明
date	是	DATE或STRING	需要处理的日期。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型的值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回33。

```
select weekofyear('2023-08-16 10:54:36');
```

 版权所有 © 华为技术有限公司

返回NULL。

```
select weekofyear('20230816');
```

返回NULL。

```
select weekofyear(null);
```

14.1.35 year

year函数用于返回指定日期中的年份。

命令格式

```
year(string date)
```

参数说明

表 14-32 参数说明

参数	是否必选	参数类型	说明
date	是	DATE或STRING	需要处理的日期。 格式： • yyyy-mm-dd • yyyy-mm-dd hh:mi:ss • yyyy-mm-dd hh:mi:ss.ff3

返回值说明

返回INT类型的值。

说明

- date非DATE或STRING类型时，返回报错，错误信息：data type mismatch。
- date为DATE或STRING类型，但不符合日期值的入参格式时，返回NULL。
- date值为NULL时，返回NULL。

示例代码

返回2023。

```
select year('2023-08-16 10:54:36');
```

返回NULL。

```
select year('23-01-01');
```

返回NULL。

```
select year('2023/08/16');
```

返回NULL。

```
select year(null);
```

 版权所有 © 华为技术有限公司

14.2 字符串函数

14.2.1 字符串函数概览

DLI所支持的字符函数如**字符串函数**所示。

表 14-33 字符串函数

函数	命令格式	返回值	功能简介
ascii	ascii(string <str>)	BIGINT	返回字符串中首字符的数字值。
concat	concat(array<T> <a>, array<T> <b>[,...]) , concat(string <str1>, string <str2>[,...])	ARRAY或STRING	连接多个字符串，合并为一个字符串，可以接受任意数量的输入字符串。
concat_ws	concat_ws(string <separator>, string <str1>, string <str2>[,...]) , concat_ws(string <separator>, array<string> <a>)	ARRAY或STRUCT	连接多个字符串，字符串之间以指定的分隔符分隔。
char_matchcount	char_matchcount(string <str1>, string <str2>)	BIGINT	计算str1中有多少个字符出现在str2中。
encode	encode(string <str>, string <charset>)	BINARY	将str按照charset格式进行编码。
find_in_set	find_in_set(string <str1>, string <str2>)	BIGINT	查找字符串str1在以逗号（,）分隔的字符串str2中的位置，从1开始计数。
get_json_object	get_json_object(string <json>, string <path>)	STRING	根据所给路径对json对象进行解析，当json对象非法时将返回NULL。
instr	instr(string <str>, string <substr>)	INT	返回substr在str中最早出现的下标。当参数中出现NULL时，返回NULL，当str中不存在substr时返回0，注意下标从1开始。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
instr1	instr1(string <str1>, string <str2>[, bigint <start_position>[, bigint <nth_appearance>]])	BIGINT	instr1函数用于计算子串str2在字符串str1中的位置。
initcap	initcap(string A)	STRING	将文本字符串转换成首字母大写其余字母小写的形式。
keyvalue	keyvalue(string <str>,[string <split1>,string <split2>,] string <key>)	STRING	用于计算将字符串str按照split1进行切分，并按split2将每组变成Key-Value对，返回key所对应的Value。
length	length(string <str>)	BIGINT	返回字符串的长度。
lengthb	lengthb(string <str>)	STRING	计算字符串str以字节为单位的长度。
levenshtein	levenshtein(string A, string B)	INT	返回两个字符串之间的Levenshtein距离，如levenshtein('kitten','sitting') =3。
locate	locate(string <substr>, string <str>[, bigint <start_pos>])	BIGINT	用于在str中查找substr的位置。
lower/lcase	lower(string A) ，lcase(string A)	STRING	将文本字符串转换成字母全部小写的形式。
lpad	lpad(string <str1>, int <length>, string <str2>)	STRING	用于返回指定长度的字符串，给定字符串str1长度小于指定长度length时，由指定字符str2从左侧填补。
ltrim	ltrim([<trimChars>,] string <str>)	STRING	删除字符串左边的空格，其他的空格保留。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
parse_url	parse_url(string urlString, string partToExtract [, string keyToExtract])	STRING	返回给定URL的指定部分，partToExtract的有效值包括HOST，PATH， QUERY， REF，PROTOCOL， AUTHORITY，FILE和USERINFO。 例如：parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'HOST') 返回'facebook.com'.。 当第二个参数为QUERY时，可以使用第三个参数提取特定参数的值，例如： parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'QUERY', 'k1') 返回'v1'。
printf	printf(String format, Obj... args)	STRING	将输入按特定格式打印输出。
regexp_count	regexp_count(string <source>, string <pattern>[, bigint <start_position>])	BIGINT	用于计算source中从start_position位置开始，匹配指定pattern的子串数。
regexp_extract	regexp_extract(string <source>, string <pattern>[, bigint <groupid>])	STRING	用于将字符串source按照pattern的分组规则进行字符串匹配，返回第groupid个组匹配到的字符串内容。
replace	replace(string <str>, string <old>, string <new>)	STRING	将字符串中与指定字符串匹配的子串替换为另一字符串。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
regexp_replace	• 适用于Spark2.4.5：regexp_replace(string <source>, string <pattern>, string <replace_string>) • 适用于Spark3.3.1：regexp_replace(string <source>, string <pattern>, string <replace_string>[, bigint <occurrence>])	STRING	• 适用于Spark2.4.5：用于将source字符串中第occurrence次匹配pattern的子串，以及之后匹配pattern的子串，全都替换成指定字符串replace_string后，返回结果字符 • 适用于Spark3.3.1：用于将source字符串中第occurrence次匹配pattern的子串，以及之后匹配pattern的子串，全都替换成指定字符串replace_string后，返回结果字符
regexp_replace1	regexp_replace1(string <source>, string <pattern>, string <replace_string>[, bigint <occurrence>])	STRING	将source字符串中第occurrence次匹配pattern的子串，替换成指定字符串replace_string后，返回结果字符串。
regexp_instr	regexp_instr(string <source>, string <pattern>[,bigint <start_position>[, bigint <occurrence>[, bigint <return_option>]]])	BIGINT	用于计算字符串source从start_position开始，与pattern第occurrence次匹配的子串的起始或结束位置。
regexp_substr	regexp_substr(string <source>, string <pattern>[, bigint <start_position>[, bigint <occurrence>]])	STRING	用于计算从start_position位置开始，source中第occurrence次匹配指定pattern的子串。
repeat	repeat(string <str>, bigint <n>)	STRING	重复N次字符串。
reverse	reverse(string <str>)	STRING	返回倒序字符串。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
rpad	rpad(string <str1>, int <length>, string <str2>)	STRING	用于将字符串str2将字符串str1向右补足到length位。
rtrim	rtrim([<trimChars>,]string <str>)， rtrim(trailing [<trimChars>] from <str>)	STRING	删除字符串右边的空格，其他的空格保留。
soundex	soundex(string <str>)	STRING	从str返回一个soundex字符串，如soundex('Miller')= M460。
space	space(bigint <n>)	STRING	返回指定数量的空格。
substr/substring	substr(string <str>, bigint <start_position>[, bigint <length>]) ，substring(string <str>, bigint <start_position>[, bigint <length>])	STRING	用于返回字符串str从start_position开始，长度为length的子串。
substring_index	substring_index(string <str>, string <separator>, int <count>)	STRING	用于截取字符串str第count个分隔符之前的字符串。如果count为正，则从左边开始截取。如果count为负，则从右边开始截取。
split_part	split_part(string <str>, string <separator>, bigint <start>[, bigint <end>])	STRING	用于依照分隔符separator拆分字符串str，返回从start部分到end部分的子串（闭区间）。
translate	translate(string|char|varchar input, string|char|varchar from, string|char|varchar to)	STRING	将input字符串中的所出现的字符或者字符串from用字符或者字符串to替换。例如：将abcde中的bcd替换成BCD，translate("abcde", "bcd", "BCD")
trim	trim([<trimChars>,]string <str>)， trim([BOTH] [<trimChars>] from <str>)	STRING	删除字符串两端的空格，字符之间的空格保留。
upper/ucase	upper(string A) ，ucase(string A)	STRING	将文本字符串转换成字母全部大写的形式。

版权所有 © 华为技术有限公司

14.2.2 ascii

ascii函数用于返回字符串str第一个字符的ASCII码。

命令格式

```
ascii(string <str>)
```

参数说明

表 14-34 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会自动转换为STRING类型后参与运算。 例如“ABC”

返回值说明

返回BIGINT的值。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str值为NULL时，返回NULL。

示例代码

- 返回字符串ABC第一个字符的ASCII码。命令示例如下。

  返回97。

  ```
  select ascii('ABC');
  ```

- 输入参数为NULL。命令示例如下。

  返回NULL。

  ```
  select ascii(null);
  ```

14.2.3 concat

concat函数用于拼接数组或字符串。

命令格式

输入为ARRAY数组：将多个ARRAY数组中的所有元素连接在一起，生成一个新的ARRAY数组。

```
concat(array<T> <a>, array<T> <b>[,...])
```

输入为字符串：将多个字符串连接在一起，生成一个新的字符串。

 版权所有 © 华为技术有限公司

```
concat(string <str1>, string <str2>[,...])
```

参数说明

- 输入为ARRAY数组

  表 14-35 参数说明

参数	是否必选	参数类型	说明
a、b	是	STRING	ARRAY数组。 array<T>中的T指代ARRAY数组元素的数据类型，数组中的元素可以为任意类型。 a和b中元素的数据类型必须一致。数组中的元素为NULL值时会参与运算。

- 输入为字符串

  表 14-36 参数说明

参数	是否必选	参数类型	说明
str1、str2	是	STRING	字符串。 如果输入参数为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会自动转换为STRING类型后参与运算，其他类型会返回报错。

返回值说明

返回ARRAY数组或STRING的值。

说明

- 返回ARRAY类型。如果任一输入ARRAY数组为NULL，返回结果为NULL。
- 返回STRING类型。如果没有参数或任一参数为NULL，返回结果为NULL。

示例代码

- 连接ARRAY数组array(1, 2)和array(2, -2)。命令示例如下。

  返回[1, 2, 2, -2]。

```
select concat(array(1, 2), array(2, -2));
```

- 任一ARRAY数组为NULL。命令示例如下。

  返回NULL。

```
select concat(array(10, 20), null);
```

- 连接字符串ABC和DEF。命令示例如下。

  返回ABCDEF。

 版权所有 © 华为技术有限公司

```
select concat('ABC','DEF');
```

- 输入为空。命令示例如下。

  返回NULL。

```
select concat();
```

- 任一字符串输入为NULL。命令示例如下。

  返回NULL。

```
select concat('abc', 'def', null);
```

14.2.4 concat_ws

concat_ws函数用于连接多个字符串，字符串之间以指定的分隔符分隔。

命令格式

```
concat_ws(string <separator>, string <str1>, string <str2>[,...])
```

或

```
concat_ws(string <separator>, array<string> <a>)
```

返回将参数中的所有字符串或ARRAY数组中的元素按照指定的分隔符连接在一起的结果。

参数说明

表 14-37 参数说明

参数	是否必选	参数类型	说明
separator	是	STRING	STRING类型的分隔符。
str1、str2	是	STRING	至少要指定2个字符串。 STRING类型。如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
a	是	ARRAY	数组中的元素为STRING类型。

返回值说明

返回STRING类型或STRUCT类型的值。

说明

- str1或str2非STRING、BIGINT、DECIMAL、DOUBLE或DATETIME类型时，返回报错。
- 如果参数（待拼接字符）为NULL，则会忽略这个参数
- 如果没有输入参数（待拼接字符）返回NULL。

示例代码

- 将字符串ABC和DEF通过:连接。命令示例如下。

  返回ABC:DEF。

```
select concat_ws(':','ABC','DEF');
```

 版权所有 © 华为技术有限公司

- 任一输入参数为NULL。命令示例如下。

  返回avg:18。

  ```
  select concat_ws(':','avg',null,'18');
  ```

- 将ARRAY数组array('name', 'lilei')中的元素通过:连接。命令示例如下。

  返回name:lilei。

  ```
  select concat_ws(':',array('name', 'lilei'));
  ```

14.2.5 char_matchcount

char_matchcount函数用于计算str1中有多少个字符出现在str2中。

命令格式

```
char_matchcount(string <str1>, string <str2>)
```

参数说明

表 14-38 参数说明

参数	是否必选	参数类型	说明
str1、str2	是	STRING	待计算的字符串str1、str2。

返回值说明

返回BIGINT类型。

说明

str1或str2值为NULL时，返回NULL。

示例代码

返回3。

```
select char_matchcount('abcz','abcde');
```

返回NULL。

```
select char_matchcount(null,'abcde');
```

14.2.6 encode

encode函数用于使用charset的编码方式对str进行编码。

命令格式

encode(string <str>, string <charset>)

 版权所有 © 华为技术有限公司

参数说明

表 14-39 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	至少要指定2个字符串。 STRING类型。如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
charset	是	STRING	编码格式。 取值范围为：UTF-8、UTF-16、UTF-16LE、UTF-16BE、ISO-8859-1、US-ASCII。

返回值说明

返回BINARY类型的值。

说明

str或charset值为NULL时，返回NULL。

示例代码

- 将字符串abc按照UTF-8格式编码。命令示例如下。

  返回abc。

  ```
  select encode("abc", "UTF-8");
  ```

- 任一输入参数为NULL。命令示例如下。

  返回结果为NULL。

  ```
  select encode("abc", null);
  ```

14.2.7 find_in_set

find_in_set函数用于查找字符串str1在以逗号（,）分隔的字符串str2中的位置，从1开始计数。

命令格式

```
find_in_set(string <str1>, string <str2>)
```

参数说明

表 14-40 参数说明

参数	是否必选	参数类型	说明
str1	是	STRING	待查找的字符串。
str2	是	STRING	以逗号（,）分隔的字符串。

 版权所有 © 华为技术有限公司

返回值说明

返回BIGINT类型的值。

说明

- 当str2中无法匹配到str1或str1中包含逗号（,）时，返回0。
- 当str1或str2值为NULL时，返回NULL。

示例代码

- 查找字符串ab在字符串abc,123,ab,c中的位置。命令示例如下。
  返回3。
  ```
  select find_in_set('ab', 'abc,123,ab,c');
  ```
- 查找字符串hi在字符串abc,123,ab,c中的位置。命令示例如下。
  返回0。
  ```
  select find_in_set('hi', 'abc,123,ab,c');
  ```
- 任一输入参数为NULL。命令示例如下。
  返回NULL。
  ```
  select find_in_set(null, 'abc,123,ab,c');
  ```

14.2.8 get_json_object

get_json_object函数用于根据所给路径对json对象进行解析，当json对象非法时将返回NULL。

命令格式

```
get_json_object(string <json>, string <path>)
```

参数说明

表 14-41 参数说明

参数	是否必选	参数类型	说明
json	是	STRING	标准的JSON格式对象，格式为{Key:Value,Key:Value,...}
path	是	STRING	表示在json中的path，以$开头。不同字符的含义如下： • $表示根节点。 • .表示子节点。 • []表示[number]表示数组下标，从0开始。 • *表示Wildcard for []，返回整个数组。*不支持转义。

 版权所有 © 华为技术有限公司

返回值说明

返回STRING类型的值。

说明

- 如果json为空或非法的json格式，返回NULL。
- 如果json合法，path也存在，则返回对应字符串。

示例代码

- 提取JSON对象src_json.json中的信息。命令示例如下。

```
jsonString = {"store": {"fruit":[{"weight":8,"type":"apple"},{"weight":9,"type":"pear"}], "bicycle":
{"price":19.95,"color":"red"} }, "email":"amy@only_for_json_udf_test.net", "owner":"Tony" }
```

提取owner字段信息，返回Tony。

```
select get_json_object(jsonString, '$.owner');
```

提取store.fruit字段第一个数组信息，返回{"weight":8,"type":"apple"}。

```
select get_json_object(jsonString, '$.store.fruit[0]');
```

提取不存在的字段信息，返回NULL。

```
select get_json_object(jsonString, '$.non_exist_key');
```

- 提取数组型JSON对象的信息。命令示例如下。

返回22。

```
select get_json_object('{"array":[["a",11],["b",22],["c",33]]}','$.array[1][1]');
```

返回["h00","h11","h22"]。

```
select get_json_object('{"a":"b","c":{"d":"e","f":"g","h":["h00","h11","h22"]},"i":"j"}','$.c.h[*]');
```

返回["h00","h11","h22"]。

```
select get_json_object('{"a":"b","c":{"d":"e","f":"g","h":["h00","h11","h22"]},"i":"j"}','$.c.h');
```

返回h11。

```
select get_json_object('{"a":"b","c":{"d":"e","f":"g","h":["h00","h11","h22"]},"i":"j"}','$.c.h[1]');
```

- 提取带有.的JSON对象中的信息。命令示例如下。

创建一张表。

```
create table json_table (id string, json string);
```

向表中插入数据，Key带 "."

```
insert into table json_table (id, json) values ("1", "{\"city1\":{\"region\":{\"rid\":6}}}");
```

向表中插入数据，Key不带 "."

```
insert into table json_table (id, json) values ("2", "{\"city1\":{\"region\":{\"rid\":7}}}");
```

取rid的值，查询key为city1，返回6。由于包含.，只能用['']来解析。

```
select get_json_object(json, "$['city1'].region['id']") from json_table where id =1;
```

取rid的值，查询key为city1，返回7。查询方法有如下两种。

```
select get_json_object(json, "$['city1'].region['id']") from json_table where id =2;
select get_json_object(json, "$.city1.region['id']") from json_table where id =2;
```

- JSON输入为空或非法格式。命令示例如下。

返回NULL。

```
select get_json_object('','$.array[2]');
```

返回NULL。

```
select get_json_object('"array":["a",1],"b":["c",3]','$.array[1][1]');
```

- JSON字符串涉及转义。命令示例如下。

 版权所有 © 华为技术有限公司

返回"3"。

```
select get_json_object('{"a":"\\"3\\"","b":"6"}', '$.a');
```

返回'3'。

```
select get_json_object('{"a":"\'3\'","b":"6"}', '$.a');
```

- 一个JSON对象中可以出现相同的Key，可以成功解析。

  返回1。

  ```
  select get_json_object('{"b":"1","b":"2"}', '$.b');
  ```

- 输出结果按照JSON字符串的原始排序方式输出。

  返回{"b":"3","a":"4"}。

  ```
  select get_json_object('{"b":"3","a":"4"}', '$');
  ```

14.2.9 instr

instr函数用于返回substr在str中最早出现的下标。

当参数中出现NULL时，返回NULL，当str中不存在substr时返回0，注意下标从1开始。

相似函数：**instr1**，instr1函数用于计算子串str2在字符串str1中的位置，instr1函数支持指定起始搜索位置和匹配次数。

命令格式

```
instr(string <str>, string <substr>)
```

参数说明

表 14-42 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待搜索的目标字符串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。
substr	是	STRING	待匹配的子串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。

返回值说明

返回BIGINT类型的值。

 版权所有 © 华为技术有限公司

说明

- 如果在str1中未找到str2，则返回0。
- 如果str2为空串，则总能匹配成功，例如select instr('abc','');会返回1。
- str1或str2值为NULL时，返回NULL。

示例代码

- 计算字符b在字符串abc中的位置。命令示例如下。
  返回2。
  ```
  select instr('abc', 'b');
  ```
- 任一输入参数为NULL。命令示例如下。
  返回NULL。
  ```
  select instr('abc', null)
  ```

14.2.10 instr1

instr1函数用于计算子串str2在字符串str1中的位置。

相似函数：**instr**，instr函数用于返回substr在str中最早出现的下标。但是instr不支持指定起始搜索位置和匹配次数。

命令格式

```
instr1(string <str1>, string <str2>[, bigint <start_position>[, bigint <nth_appearance>]])
```

参数说明

表 14-43 参数说明

参数	是否必选	参数类型	说明
str1	是	STRING	待搜索的目标字符串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。
str2	是	STRING	待匹配的子串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。
start_position	否	BIGINT	表示从str1的第几个字符开始搜索，默认起始位置是第一个字符位置1。 不支持指定负数。
nth_appearance	否	BIGINT	表示str2在str1中第nth_appearance次匹配的位置。 如果nth_appearance为其他类型或小于等于0，则返回报错。

 版权所有 © 华为技术有限公司

返回值说明

返回BIGINT类型。

说明

- 如果在str1中未找到str2，则返回0。
- 如果str2为空串，则总能匹配成功。
- str1、str2、start_position或nth_appearance值为NULL时，返回NULL。

示例代码

返回 10

```
select instr1('Tech on the net', 'h', 5, 1);
```

返回2。

```
select instr1('abc', 'b');
```

返回NULL。

```
select instr('abc', null);
```

14.2.11 initcap

initcap函数用于将文本字符串转换成首字母大写其余字母小写的形式。

命令格式

```
initcap(string A)
```

参数说明

表 14-44 参数说明

参数	是否必选	参数类型	说明
A	是	STRING	待转换的文本字符串。

返回值说明

返回一个STRING类型字符串，字符串中每个单词首字母大写，其余变为小写。

示例代码

返回Dli Sql

```
SELECT initcap("dLI sql");
```

 版权所有 © 华为技术有限公司

14.2.12 keyvalue

keyvalue函数用于计算将字符串str按照split1进行切分，并按split2将每组变成Key-Value对，返回key所对应的Value。

命令格式

```
keyvalue(string <str>,[string <split1>,string <split2>,] string <key>)
```

参数说明

表 14-45 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待拆分的字符串。
split1、split2	否	STRING	用于作为分隔符的字符串，按照指定的两个分隔符拆分源字符串。如果表达式中没有指定这两项，默认split1为";"，split2为":"。当某个被split1拆分后的字符串中有多个split2时，返回结果未定义。
key	否	BIGINT	将字符串按照split1和split2拆分后，返回key值对应的Value。

返回值说明

返回STRING类型。

说明

- split1或split2值为NULL时，返回NULL。
- str或key值为NULL或没有匹配的key时，返回NULL。
- 如果有多个Key-Value匹配，返回第一个匹配上的key对应的Value。

示例代码

返回2。

```
select keyvalue('a:1;b:2', 'b');
```

返回2。

```
select keyvalue("\;abc:1\;def:2","\;",":","def");
```

14.2.13 length

length函数用于返回字符串的长度。

相似函数：**lengthb**，lengthb函数用于计算字符串str以字节为单位的长度，返回STRING类型的值。

 版权所有 © 华为技术有限公司

命令格式

```
length(string <str>)
```

参数说明

表 14-46 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待搜索的目标字符串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。

返回值说明

返回BIGINT类型的值。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str值为NULL时，返回NULL。

示例代码

- 计算字符串abc的长度。命令示例如下。
  返回3。
  ```
  select length('abc');
  ```
- 输入参数为NULL。命令示例如下。
  返回NULL。
  ```
  select length(null);
  ```

14.2.14 lengthb

lengthb函数用于计算字符串str以字节为单位的长度。

相似函数：**length**，length函数用于返回字符串的长度，返回BIGINT类型的值。

命令格式

```
lengthb(string <str>)
```

参数说明

表 14-47 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	输入的字符串。

 版权所有 © 华为技术有限公司

返回值说明

返回STRING类型的值。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str值为NULL时，返回NULL。

示例代码

返回5。

```
select lengthb('hello');
```

返回NULL。

```
select lengthb(null);
```

14.2.15 levenshtein

levenshtein函数用于返回两个字符串之间的Levenshtein距离，如levenshtein('kitten','sitting') =3。

说明

Levenshtein距离，是编辑距离的一种。指两个字串之间，由一个转成另一个所需的最少编辑操作次数。

命令格式

```
levenshtein(string A, string B)
```

参数说明

表 14-48 参数说明

参数	是否必选	参数类型	说明
A、B	是	STRING	计算Levenshtein距离需要输入的字符串。

返回值说明

返回INT类型的值。

示例代码

返回3

```
SELECT levenshtein('kitten','sitting');
```

 版权所有 © 华为技术有限公司

14.2.16 locate

locate函数用于在str中查找substr的位置。您可以通过start_pos指定开始查找的位置，从1开始计数。

命令格式

```
locate(string <substr>, string <str>[, bigint <start_pos>])
```

参数说明

表 14-49 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待搜索的目标字符串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。
substr	是	STRING	待匹配的子串。 如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算，其他类型会返回报错。
start_pos	否	BIGINT	指定查找的起始位置。

返回值说明

返回为BIGINT类型。

说明

- str中无法匹配到substr时，返回0。
- str或substr值为NULL时，返回NULL。
- start_pos值为NULL时，返回0。

示例代码

- 查找字符串ab在字符串abhiab中的位置。命令示例如下。
  返回1。
  ```
  select locate('ab', 'abhiab');
  ```
  返回5。
  ```
  select locate('ab', 'abhiab', 2);
  ```
  返回0。
  ```
  select locate('ab', 'abhiab', null);
  ```
- 查找字符串hi在字符串hanmeimei and lilei中的位置。命令示例如下。
  返回0。

 版权所有 © 华为技术有限公司

```
select locate('hi', 'hanmeimei and lilei');
```

14.2.17 lower/lcase

lower函数用于将文本字符串转换成字母全部小写的形式。

命令格式

```
lower(string A) / lcase(string A)
```

参数说明

表 14-50 参数说明

参数	是否必选	参数类型	说明
A	是	STRING	待转换的文本字符串。

返回值说明

返回为STRING类型的值。

说明

- 入参非 STRING、BIGINT、DOUBLE、DECIMAL 或 DATETIME 类型时，返回报错。
- 入参值为NULL时，返回NULL。

示例代码

将字符串中的大写字符转换为小写字符。命令示例如下。

返回 abc。

```
select lower('ABC');
```

输入参数为NULL。命令示例如下。

返回NULL。

```
select lower(null);
```

14.2.18 lpad

locate函数用于返回指定长度的字符串，给定字符串str1长度小于指定长度length时，由指定字符str2从左侧填补。

命令格式

```
lpad(string <str1>, int <length>, string <str2>)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-51 参数说明

参数	是否必选	参数类型	说明
str1	是	STRING	待向左补位的字符串。
length	是	STRING	向左补位位数。
str2	否	STRING	用于补位的字符串。

返回值说明

返回STRING类型的值。

说明

- 如果length小于str1的位数，则返回str1从左开始截取length位的字符串。
- 如果length为0，则返回空串。
- 如果没有输入参数或任一输入参数值为NULL，返回NULL。

示例代码

- 用字符串ZZ将字符串abcdefgh向左补足到10位。命令示例如下。
  返回ZZabcdefgh。
  ```
  select lpad('abcdefgh', 10, 'ZZ');
  ```
- 用字符串ZZ将字符串abcdefgh向左补足到5位。命令示例如下。
  返回abcde。
  ```
  select lpad('abcdefgh', 5, 'ZZ');
  ```
- length为0。命令示例如下。
  返回空串。
  ```
  select lpad('abcdefgh', 0, 'ZZ');
  ```
- 任一输入参数为NULL。命令示例如下。
  返回NULL。
  ```
  select lpad(null ,0, 'ZZ');
  ```

14.2.19 ltrim

ltrim函数用于从str的左端去除字符：

- 如果未指定trimChars，则默认去除空格字符。
- 如果指定了trimChars，则以trimChars中包含的字符作为一个集合，从str的左端去除尽可能长的所有字符都在集合trimChars中的子串。

相似函数：

- **rtrim**，rtrim函数用于从str的右端去除字符。
- **trim**，trim函数用于从str的左右两端去除字符。

 版权所有 © 华为技术有限公司

命令格式

```
ltrim([<trimChars>,] string <str>)
```

参数说明

表 14-52 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待去除左端字符的字符串。如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
trimChars	是	STRING	待去除的字符。

返回值说明

返回为STRING类型。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str或trimChars值为NULL时，返回NULL。

示例代码

- 去除字符串" abc"的左边空格。命令示例如下。

  返回字符串abc 。

  ```
  select ltrim('     abc');
  ```

  等效于如下语句。

  ```
  select trim(leading from '     abc');
  ```

  leading代表去除字符串前面的空格
- 输入参数为NULL。命令示例如下。

  返回NULL。

  ```
  select ltrim(null);
  select ltrim('xy', null);
  select ltrim(null, 'xy');
  ```
- 去除字符串yxlucyxx左端所有字符都在集合xy中的子串。

  返回lucyxx，只要左端遇到x或者y就会被去掉。

  ```
  select ltrim( 'xy','yxlucyxx');
  ```

  等效于如下语句。

  ```
  select trim(leading 'xy' from 'yxlucyxx');
  ```

14.2.20 parse_url

parse_url函数用于返回给定URL的指定部分，partToExtract的有效值包括HOST，PATH， QUERY， REF， PROTOCOL， AUTHORITY，FILE和USERINFO。

 版权所有 © 华为技术有限公司

例如：parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'HOST') 返回 'facebook.com'.。

当第二个参数为QUERY时，可以使用第三个参数提取特定参数的值，例如：parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'QUERY', 'k1') 返回'v1'。

命令格式

```
parse_url(string urlString, string partToExtract [, string keyToExtract])
```

参数说明

表 14-53 参数说明

参数	是否必选	参数类型	说明
urlString	是	STRING	URL链接。无效URL链接会返回报错。
partToExtract	是	STRING	取值包含：HOST、PATH、QUERY、REF、PROTOCOL、AUTHORITY、FILE和USERINFO，不区分大小写。
keyToExtract	否	STRING	当part取值为QUERY时，根据key值取出对应的Value值。

返回值说明

返回STRING类型。返回规则如下：

说明

- urlString、partToExtract或keyToExtract值为NULL时，返回NULL。
- partToExtract 取值不符合要求时，返回报错。

示例代码

返回example.com。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'HOST');
```

返回/over/there/index.dtb。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'PATH');
```

返回animal。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'QUERY', 'type');
```

返回nose。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'REF');
```

返回file。

 版权所有 © 华为技术有限公司

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'PROTOCOL');
```

返回 username@example.com:8042。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'AUTHORITY');
```

返回username。

```
select parse_url('file://username@example.com:666/over/there/index.dtb?
type=animal&name=narwhal#nose', 'USERINFO');
```

14.2.21 printf

printf函数用于将输入按特定格式打印输出。

命令格式

```
printf(String format, Obj... args)
```

参数说明

表 14-54 参数说明

参数	是否必选	参数类型	说明
format	是	STRING	用于定义输出格式
Obj	否	STRING	其他输入参数。

返回值说明

返回STRING类型的值。

将Obj中的参数填入format后打印输出。

示例代码;

返回字符串：姓名：user1，年龄：20，性别：女，籍贯：城市1。

```
SELECT printf('姓名：%s，年龄：%d，性别：%s，籍贯：%s', "user1", 20, "女", "城市1");
```

14.2.22 regexp_count

regexp_count函数用于计算source中从start_position位置开始，匹配指定pattern的子串数。

命令格式

```
regexp_count(string <source>, string <pattern>[, bigint <start_position>])
```

 版权所有 © 华为技术有限公司

参数说明

表 14-55 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	待搜索的字符串，其他类型会返回报错。
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。pattern为空串或其他类型时返回报错。
start_position	否	BIGINT	BIGINT类型常量，必须大于0。其他类型或值小于等于0时返回报错。不指定时默认为1，表示从source的第一个字符开始匹配。

返回值说明

返回BIGINT类型的值。

说明

- 如果没有匹配成功，返回0。
- source、pattern值为NULL时，返回NULL。

示例代码

返回4。

```
select regexp_count('ab0a1a2b3c', '[0-9]');
```

返回3。

```
select regexp_count('ab0a1a2b3c', '[0-9]', 4);
```

返回 null。

```
select regexp_count('ab0a1a2b3c', null);
```

14.2.23 regexp_extract

REGEXP_EXTRACT函数用于将字符串source按照pattern的分组规则进行字符串匹配，返回第groupid个组匹配到的字符串内容。

命令格式

regexp_extract(string <source>, string <pattern>[, bigint <groupid>])

 版权所有 © 华为技术有限公司

参数说明

表 14-56 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	待拆分的字符串。
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。
groupid	否	BIGINT	BIGINT类型常量，必须大于等于0。

返回值说明

返回STRING类型。

说明

- 如果pattern为空串或pattern中没有分组，返回报错。
- groupid非BIGINT类型或小于0时，返回报错。
- 不指定时默认为1，表示返回第一个组。
- 如果groupid等于0，则返回满足整个pattern的子串。
- source、pattern或groupid值为NULL时，返回NULL。

示例代码

将 basketball 按照 bas(.*?)(ball) 拆分。返回ket。

```
select regexp_extract('basketball', 'bas(.*?)(ball)');
```

返回 basketball 。

```
select regexp_extract('basketball', 'bas(.*?)(ball)',0);
```

返回99。在DLI上提交正则计算的SQL，需要使用两个"\"作为转义字符。

```
select regexp_extract('8d99d8', '8d(\\d+)d8');
```

返回【你好】。

```
select regexp_extract('【你好】hello', '([^\\x{00}-\\x{ff}]+)');
```

返回你好。

```
select regexp_extract('【你好】hello', '([\\x{4e00}-\\x{9fa5}]+)');
```

14.2.24 replace

replace函数用于用new字符串替换str字符串中与old字符串完全重合的部分并返回替换后的str。

如果没有重合的字符串，返回原str。

命令格式

```
replace(string <str>, string <old>, string <new>)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-57 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待替换的字符串。
old	是	STRING	待比较的字符串。
new	是	STRING	替换后的字符串。

返回值说明

返回STRING类型的值。

说明

如果任一输入参数值为NULL，返回NULL。

示例代码

返回AA123AA。

```
select replace('abc123abc','abc','AA');
```

返回NULL。

```
select replace('abc123abc',null,'AA');
```

14.2.25 regexp_replace

根据Spark版本不同，regexp_replace函数的功能略有差异：

- Spark2.4.5版本及以前版本：regexp_replace函数用于将source字符串中匹配pattern的子串替换成指定字符串replace_string后，返回结果字符串。
- Spark3.1.1版本：regexp_replace函数用于将source字符串中第occurrence次匹配pattern的子串，以及之后匹配pattern的子串，全都替换成指定字符串replace_string后，返回结果字符串。

相似函数：**regexp_replace1**，regexp_replace1函数用于将source字符串中第occurrence次匹配pattern的子串，替换成指定字符串replace_string后，返回结果字符串。但egexp_replace1函数仅适用于Spark2.4.5版本及以前版本。

即Spark2.4.5版适用的函数：regexp_replace1函数支持指定occurrence，但是regexp_replace函数不支持指定occurrence。

命令格式

- spark2.4.5及以前版本
  ```
  regexp_replace(string <source>, string <pattern>, string <replace_string>)
  ```
- spark3.1.1版本
  ```
  regexp_replace(string <source>, string <pattern>, string <replace_string>[, bigint <occurrence>])
  ```

 版权所有 © 华为技术有限公司

参数说明

表 14-58 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	待替换的字符。
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。更多正则表达式编写规范，请参见正则表达式规范。pattern为空串时返回报错。
replace_string	是	STRING	将匹配pattern的字符串替换后的字符串。
occurrence	否	BIGINT	必须大于等于1，表示将第occurrence次匹配的字符串替换为replace_string，为1时表示替换所有匹配的子串。为其他类型或小于1时，返回报错。默认值为1。 **说明** 该字段仅Spark3.1.1版本的功能适用。

返回值说明

返回STRING类型的值。

说明

- 如果pattern为空串或pattern中没有分组，返回报错。
- 当引用不存在的组时，不进行替换。
- 如果replace_string值为NULL且pattern有匹配，返回NULL。
- 如果replace_string值为NULL但pattern不匹配，返回NULL。
- source、pattern或occurrence值为NULL时，返回NULL。

示例代码

- 适用于spark2.4.5及以前版本示例

  返回 num-num。

  ```
  SELECT regexp_replace('100-200', '(\\d+)', 'num');
  ```

- 适用于spark3.1.1版本示例。

  返回 2222。

  ```
  select regexp_replace('abcd', '[a-z]', '2');
  ```

  返回 2222。

  ```
  select regexp_replace('abcd', '[a-z]', '2', 1);
  ```

  返回 a222。

  ```
  select regexp_replace('abcd', '[a-z]', '2', 2);
  ```

  返回 ab22。

  ```
  select regexp_replace('abcd', '[a-z]', '2', 3);
  ```

 版权所有 © 华为技术有限公司

返回 abc2。

```
select regexp_replace('abcd', '[a-z]', '2', 4);
```

14.2.26 regexp_replace1

regexp_replace1函数用于将source字符串中第occurrence次匹配pattern的子串，替换成指定字符串replace_string后，返回结果字符串。

说明

regexp_replace1函数只适用于Spark 2.4.5及之前的版本。

相似函数：**regexp_replace**，regexp_replace函数针对不同的Spark版本，功能略有差异，请参考**regexp_replace**查看详细的功能说明。

命令格式

```
regexp_replace1(string <source>, string <pattern>, string <replace_string>[, bigint <occurrence>])
```

参数说明

表 14-59 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	待替换的字符
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。更多正则表达式编写规范，请参见正则表达式规范。pattern为空串时返回报错。
replace_string	是	STRING	将匹配pattern的字符串替换后的字符串。
occurrence	否	BIGINT	必须大于等于1，表示将第occurrence次匹配的字符串替换为replace_string，为1时表示替换所有匹配的子串。为其他类型或小于1时，返回报错。默认值为1。

返回值说明

返回STRING类型的值。

说明

- 当引用不存在的组时，不进行替换。
- 如果replace_string值为NULL且pattern有匹配，返回NULL。
- 如果replace_string值为NULL但pattern不匹配，返回NULL。
- source、pattern或occurrence值为NULL时，返回NULL。

 版权所有 © 华为技术有限公司

示例代码

返回 2222。

```
select regexp_replace1('abcd', '[a-z]', '2');
```

返回 2bcd。

```
select regexp_replace1('abcd', '[a-z]', '2', 1);
```

返回 a2cd。

```
select regexp_replace1('abcd', '[a-z]', '2', 2);
```

返回 ab2d。

```
select regexp_replace1('abcd', '[a-z]', '2', 3);
```

返回 abc2。

```
select regexp_replace1('abcd', '[a-z]', '2', 4);
```

14.2.27 regexp_instr

regexp_instr函数用于计算字符串source从start_position开始，与pattern第occurrence次匹配的子串的起始或结束位置。

命令格式

```
regexp_instr(string <source>, string <pattern>[,bigint <start_position>[, bigint <occurrence>[, bigint <return_option>]]])
```

参数说明

表 14-60 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	源字符串。
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。pattern为空串时返回报错。
start_position	否	BIGINT	BIGINT类型常量。搜索的开始位置。不指定时默认值为1。
occurrence	否	BIGINT	BIGINT类型常量。指定匹配次数，不指定时默认值为1，表示搜索第一次出现的位置。
return_option	否	BIGINT	BIGINT类型常量。指定返回的位置。值为0或1，不指定时默认值为0，其他类型或不允许的值会返回报错。0表示返回匹配的开始位置，1表示返回匹配的结束位置。

 版权所有 © 华为技术有限公司

返回值说明

返回BIGINT类型。return_option指定匹配的子串在source中的开始或结束位置。

说明

- 如果pattern为空串，返回报错。
- start_position或occurrence非BIGINT类型或小于等于0时，返回报错。
- source、pattern、start_position、occurrence或return_option值为NULL时，返回NULL

示例代码

返回6。

```
select regexp_instr('a1b2c3d4', '[0-9]', 3, 2);
```

返回NULL。

```
select regexp_instr('a1b2c3d4', null, 3, 2);
```

14.2.28 regexp_substr

regexp_substr函数用于计算从start_position位置开始，source中第occurrence次匹配指定pattern的子串。

命令格式

```
regexp_substr(string <source>, string <pattern>[, bigint <start_position>[, bigint <occurrence>]])
```

参数说明

表 14-61 参数说明

参数	是否必选	参数类型	说明
source	是	STRING	待搜索的字符串。
pattern	是	STRING	STRING类型常量或正则表达式。待匹配的模型。
start_position	否	BIGINT	起始位置，必须大于0。不指定时默认为1，表示从source的第一个字符开始匹配。
occurrence	否	BIGINT	BIGINT常量，必须大于0。不指定时默认为1，表示返回第一次匹配的子串。

返回值说明

返回STRING类型的值。

 版权所有 © 华为技术有限公司

说明

- 如果pattern为空串，返回报错。
- 没有匹配时，返回NULL。
- start_position或occurrence非BIGINT类型或小于等于0时，返回报错。
- source、pattern、start_position、occurrence或return_option值为NULL时，返回NULL。

示例代码

返回a。

```
select regexp_substr('a1b2c3', '[a-z]');
```

返回b。

```
select regexp_substr('a1b2c3', '[a-z]', 2, 1);
```

返回c。

```
select regexp_substr('a1b2c3', '[a-z]', 2, 2);
```

返回NULL。

```
select regexp_substr('a1b2c3', null);
```

14.2.29 repeat

repeat函数用于返回将str重复n次后的字符串。

命令格式

```
repeat(string <str>, bigint <n>)
```

参数说明

表 14-62 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算。
n	是	BIGINT	重复的数字n。

返回值说明

返回STRING类型。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- n为空时，返回报错。
- str或n值为NULL时，返回NULL。

 版权所有 © 华为技术有限公司

示例代码

将字符‘123’重复2次，返回 123123。

```
SELECT repeat('123', 2);
```

14.2.30 reverse

reverse函数用于返回倒序字符串。

命令格式

```
reverse(string <str>)
```

参数说明

表 14-63 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	如果输入为BIGINT、DOUBLE、DECIMAL或DATETIME类型，则会隐式转换为STRING类型后参与运算。

返回值说明

返回STRING类型。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str值为NULL时，返回NULL。

示例代码

返回 LQS krapS。

```
SELECT reverse('Spark SQL');
```

返回[3,4,1,2]。

```
SELECT reverse(array(2, 1, 4, 3));
```

14.2.31 rpad

rpad函数用于将字符串str2将字符串str1向右补足到length位。

命令格式

```
rpad(string <str1>, int <length>, string <str2>)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-64 参数说明

参数	是否必选	参数类型	说明
str1	是	STRING	待向右补位的字符串。
length	是	INT	向右补位位数。
str2	是	STRING	用于补位的字符串。

返回值说明

返回STRING类型。

说明

- 如果length小于str1的位数，则返回str1从左开始截取length位的字符串。
- 如果length为0，则返回空串。
- 如果没有输入参数或任一输入参数值为NULL，返回NULL。

示例代码

返回 hi???。

```
SELECT rpad('hi', 5, '??');
```

返回 h。

```
SELECT rpad('hi', 1, '??');
```

14.2.32 rtrim

rtrim函数用于从str的右端去除字符：

- 如果未指定trimChars，则默认去除空格字符。
- 如果指定了trimChars，则以trimChars中包含的字符作为一个集合，从str的右端去除尽可能长的所有字符都在集合trimChars中的子串。

相似函数：

- **ltrim**，ltrim函数用于从str的左端去除字符。
- **trim**，trim函数用于从str的左右两端去除字符。

命令格式

```
rtrim([<trimChars>, ]string <str>)
```

或

```
rtrim(trailing [<trimChars>] from <str>)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-65 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待去除右端字符的字符串。如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
trimChars	是	STRING	待去除的字符。

返回值说明

返回为STRING类型的值。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str或trimChars值为NULL时，返回NULL。

示例代码

- 去除字符串 yxabcxx 的右边空格。命令示例如下。

  返回字符串 yxabcxx。

  ```
  select rtrim('yxabcxx     ');
  ```

  等效于如下语句。

  ```
  select trim(trailing from ' yxabcxx ');
  ```

- 去除字符串yxabcxx右端所有字符都在集合xy中的子串。

  返回yxabc，只要右端遇到x或者y就会被去掉。

  ```
  select rtrim('xy', 'yxabcxx');
  ```

  等效于如下语句。

  ```
  select trim(trailing 'xy' from 'yxabcxx');
  ```

- 输入参数为NULL。命令示例如下。

  返回NULL。

  ```
  select rtrim(null);
  select ltrim('yxabcxx', 'null');
  ```

14.2.33 soundex

soundex函数用于从str返回一个soundex字符串，如soundex('Miller')= M460。

命令格式

```
soundex(string <str>)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-66 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待转换的字符串。

返回值说明

返回STRING类型的值。

说明

str值为NULL时，返回NULL。

示例代码

返回M460

```
SELECT soundex('Miller');
```

14.2.34 space

space函数用于返回指定数量的空格。

命令格式

```
space(bigint <n>)
```

参数说明

表 14-67 参数说明

参数	是否必选	参数类型	说明
n	是	BIGINT	用于指定空格数量。

返回值说明

返回STRING类型。

说明

- n为空时，返回报错。
- n值为NULL时，返回NULL。

示例代码

返回6。

```
select length(space(6));
```

 版权所有 © 华为技术有限公司

14.2.35 substr/substring

substr、substring函数用于返回字符串str从start_position开始，长度为length的子串。

命令格式

```
substr(string <str>, bigint <start_position>[, bigint <length>])
```

或

```
substring(string <str>, bigint <start_position>[, bigint <length>])
```

参数说明

表 14-68 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
start_position	是	BIGINT	表示起始位置。默认起始位置为1。 如果start_position取值为正，则从左边开始。如果start_position取值为负，则从右边开始。
length	否	BIGINT	表示子串的长度值。值必须大于0。

返回值说明

返回STRING类型的值。

说明

- str非STRING、BIGINT、DECIMAL、DOUBLE或DATETIME类型时，返回报错。
- length非BIGINT类型或值小于等于0时，返回报错。
- 当length被省略时，返回到str结尾的子串。
- str、start_position或length值为NULL时，返回NULL。

示例代码

返回 k SQL。

```
SELECT substr('Spark SQL', 5);
```

返回 SQL。

```
SELECT substr('Spark SQL', -3);
```

返回 k。

```
SELECT substr('Spark SQL', 5, 1);
```

 版权所有 © 华为技术有限公司

14.2.36 substring_index

substring_index函数用于截取字符串str第count个分隔符之前的字符串。如果count为正，则从左边开始截取。如果count为负，则从右边开始截取。

命令格式

```
substring_index(string <str>, string <separator>, int <count>)
```

参数说明

表 14-69 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待截取的字符串。
separator	是	STRING	STRING类型的分隔符。
count	否	INT	指定分隔符位置。

返回值说明

返回STRING类型。

说明

如果任一输入参数值为NULL，返回NULL。

示例代码

返回 hello.world。

```
SELECT substring_index('hello.world.people', '.', 2);
```

返回world.people。

```
select substring_index('hello.world.people', '.', -2);
```

14.2.37 split_part

split_part函数用于依照分隔符separator拆分字符串str，返回从start部分到end部分的子串（闭区间）。

命令格式

```
split_part(string <str>, string <separator>, bigint <start>[, bigint <end>])
```

 版权所有 © 华为技术有限公司

参数说明

表 14-70 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待拆分的字符串。
separator	是	STRING	STRING类型常量。拆分用的分隔符，可以是一个字符，也可以是一个字符串。
start	是	STRING	BIGINT类型常量，必须大于0。表示返回段的开始编号（从1开始）。
end	否	BIGINT	BIGINT类型常量，大于等于start。表示返回段的截止编号，可省略，缺省时表示和start取值相等，返回start指定的段。

返回值说明

返回STRING类型的值。

说明

- 如果start的值大于切分后实际的分段数，例如字符串拆分完有4个片段，start大于4，返回空串。
- 如果separator不存在于str中，且start指定为1，返回整个str。如果str为空串，则输出空串。
- 如果separator为空串，则返回原字符串str。
- 如果end大于片段个数，返回从start开始的子串。
- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- start或end非BIGINT类型常量时，返回报错。
- 除separator外，如果任一参数值为NULL，返回NULL。

示例代码

返回aa。

```
select split_part('aa,bb,cc,dd', ',', 1);
```

返回aa,bb。

```
select split_part('aa,bb,cc,dd', ',', 1, 2);
```

返回空串。

```
select split_part('aa,bb,cc,dd', ',', 10);
```

返回aa,bb,cc,dd。

```
select split_part('aa,bb,cc,dd', ':', 1);
```

返回空串。

```
select split_part('aa,bb,cc,dd', ':', 2);
```

返回aa,bb,cc,dd。

 版权所有 © 华为技术有限公司

```
select split_part('aa,bb,cc,dd', '', 1);
```

返回bb,cc,dd。

```
select split_part('aa,bb,cc,dd', ',', 2, 6);
```

返回NULL。

```
select split_part('aa,bb,cc,dd', ',', null);
```

14.2.38 translate

translate函数用于将input字符串中的所出现的字符或者字符串from用字符或者字符串to替换。

例如：将abcde中的bcd替换成BCD。

```
translate("abcde", "bcd", "BCD")
```

命令格式

```
translate(string|char|varchar input, string|char|varchar from, string|char|varchar to)
```

参数说明

表 14-71 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待截取的字符串。
separator	是	STRING	STRING类型的分隔符。
count	否	INT	指定分隔符位置。

返回值说明

返回STRING类型的值。

说明

如果任一输入参数值为NULL，返回NULL。

示例代码

返回 A1B2C3。

```
SELECT translate('AaBbCc', 'abc', '123');
```

14.2.39 trim

trim函数用于从str的左右两端去除字符：

- 如果未指定trimChars，则默认去除空格字符。
- 如果指定了trimChars，则以trimChars中包含的字符作为一个集合，从str的左右两端去除尽可能长的所有字符都在集合trimChars中的子串。

 版权所有 © 华为技术有限公司

相似函数：

- **ltrim**，ltrim函数用于从str的左端去除字符。
- **rtrim**，rtrim函数用于从str的右端去除字符。

命令格式

```
trim([<trimChars>,]string <str>)
```

或

```
trim([BOTH] [<trimChars>] from <str>)
```

参数说明

表 14-72 参数说明

参数	是否必选	参数类型	说明
str	是	STRING	待去除左右两端字符的字符串。 如果输入为BIGINT、DECIMAL、DOUBLE或DATETIME类型，则会隐式转换为STRING类型后参与运算。
trimChars	否	STRING	待去除的字符。

返回值说明

返回为STRING类型的值。

说明

- str非STRING、BIGINT、DOUBLE、DECIMAL或DATETIME类型时，返回报错。
- str或trimChars值为NULL时，返回NULL。

示例代码

- 去除字符串 yxabcxx 的左右空格。命令示例如下。
  返回字符串yxabcxx。
  ```
  select trim(' yxabcxx ');
  ```
  等效于如下语句。
  ```
  select trim(both from ' yxabcxx ');
  select trim(from ' yxabcxx ');
  ```
- 去除字符串yxabcxx左右两端所有字符都在集合xy中的子串。
  返回abc，只要左右两端遇到x或者y就会被去掉。
  ```
  select trim('xy', 'yxabcxx');
  ```
  等效于如下语句。
  ```
  select trim(both 'xy' from 'yxabcxx');
  ```
- 输入参数为NULL。命令示例如下。
  返回NULL。

 版权所有 © 华为技术有限公司

```
select trim(null);
select trim(null, 'yxabcxx');
```

14.2.40 upper/ucase

upper函数用于从将文本字符串转换成字母全部大写的形式。

命令格式

```
upper(string A)
```

或

```
ucase(string A)
```

参数说明

表 14-73 参数说明

参数	是否必选	参数类型	说明
A	是	STRING	待转换的文本字符串。

返回值说明

返回STRING类型。

说明

- 入参非 STRING、BIGINT、DOUBLE、DECIMAL 或 DATETIME 类型时，返回报错。
- 入参值为NULL时，返回NULL。

示例代码

将字符串中的小写字符转换为大写字符。命令示例如下。

返回ABC。

```
select upper('abc');
```

输入参数为NULL。命令示例如下。

返回NULL。

```
select upper(null);
```

14.3 数学函数

14.3.1 数学函数概览

DLI所支持的数学函数如**数学函数**所示。

表 14-74 数学函数

函数	命令格式	返回值	功能简介
abs	abs(DOUBLE a)	DOUBLE或INT	取绝对值。
acos	acos(DOUBLE a)	DOUBLE	返回给定角度a的反余弦值。
asin	asin(DOUBLE a)	DOUBLE	返回给定角度a的反正弦值。
atan	atan(DOUBLE a)	DOUBLE	返回给定角度a的反正切值。
bin	bin(BIGINT a)	STRING	返回二进制格式。
bround	bround(DOUBLE a)	DOUBLE	HALF_EVEN模式四舍五入，与传统四舍五入方式的区别在于，对数字5进行操作时，由前一位数字来决定，前一位数字为奇数，增加一位，前一位数字为偶数，舍弃一位。例如：bround(7.5)=8.0，bround(6.5)=6.0
bround	bround(DOUBLE a, INT d)	DOUBLE	保留小数点后d位，d位之后数字以HALF_EVEN模式四舍五入。与传统四舍五入方式的区别在于，对数字5进行操作时，由前一位数字来决定，前一位数字为奇数，增加一位，前一位数字为偶数，舍弃一位。例如：bround(8.25, 1) = 8.2, bround(8.35, 1) = 8.4。
cbrt	cbrt(DOUBLE a)	DOUBLE	返回a的立方根。
ceil	ceil(DOUBLE a)	DECIMAL	将参数向上舍入为最接近的整数。例如：ceil(21.2)，返回22。
conv	conv(BIGINT num, INT from_base, INT to_base), conv(STRING num, INT from_base, INT to_base)	STRING	进制转换，将from_base进制下的num转化为to_base进制下面的数。例如：将5从十进制转换为四进制，conv(5,10,4)=11。
cos	cos(DOUBLE a)	DOUBLE	返回给定角度a的余弦值。
cot1	cot1(DOUBLE a)	DOUBLE或DECIMAL类型	计算number的余切函数，输入为弧度值。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
degress	degrees(DOUBLE a)	DOUBLE	返回弧度所对应的角度。
e	e()	DOUBLE	返回e的值。
exp	exp(DOUBLE a)	DOUBLE	返回e的a次方。
factorial	factorial(INT a)	BIGINT	返回a的阶乘。
floor	floor(DOUBLE a)	BIGINT	对给定数据进行向下舍入最接近的整数。例如：floor(21.2)，返回21。
greatest	greatest(T v1, T v2, ...)	DOUBLE	返回列表中的最大值。
hex	hex(BIGINT a) hex(STRING a)	STRING	将整数或字符转换为十六进制格式。
least	least(T v1, T v2, ...)	DOUBLE	返回列表中的最小值。
ln	ln(DOUBLE a)	DOUBLE	返回给定数值的自然对数。
log	log(DOUBLE base, DOUBLE a)	DOUBLE	返回给定底数及指数返回自然对数。
log10	log10(DOUBLE a)	DOUBLE	返回给定数值的以10为底自然对数。
log2	log2(DOUBLE a)	DOUBLE	返回给定数值的以2为底自然对数。
median	median(colname)	DOUBLE或DECIMAL	计算中位数。
negative	negative(INT a)	DECIMAL或INT	返回a的相反数，例如negative(2)，返回-2。
percentile	percentile(colname,DOUBLE p)	DOUBLE或ARRAY	计算精确百分位数，适用于小数据量。先对指定列升序排列，然后取精确的第p位百分数。p必须在0和1之间。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
percentile_approx	percentile_approx(colname,DOUBLE p)	DOUBLE或ARRAY	计算近似百分位数，适用于大数据量。先对指定列升序排列，然后取第p位百分数对应的值。
pi	pi()	DOUBLE	返回pi的值。
pmod	pmod(INT a, INT b)	DECIMAL或INT	返回a除b的余数的绝对值。
positive	positive(INT a)	DECIMAL、DOUBLE或INT	返回a的值，例如positive(2)，返回2。
pow	pow(DOUBLE a, DOUBLE p), power(DOUBLE a, DOUBLE p)	DOUBLE	返回a的p次幂。
radians	radians(DOUBLE a)	DOUBLE	返回角度所对应的弧度。
rand	rand(INT seed)	DOUBLE	返回大于或等于0且小于1的平均分布随机数。如果指定种子seed，则会得到一个稳定的随机数序列。
round	round(DOUBLE a)	DOUBLE	四舍五入。
round	round(DOUBLE a, INT d)	DOUBLE	小数部分d位之后数字四舍五入，例如round(21.263,2)，返回21.26。
shiftleft	shiftleft(BIGINT a, INT b)	INT	有符号左边移，将a的二进制数按位左移b位。
shiftright	shiftright(BIGINT a, INT b)	INT	有符号右移，将a的二进制数按位右移b位。
shiftrightunsigned	shiftrightunsigned(BIGINT a, INT b)	INT	无符号右移，将a的二进制数按位右移b位。
sign	sign(DOUBLE a)	DOUBLE	返回a所对应的正负号，a为正返回1.0，a为负，返回-1.0，否则返回0.0。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
sin	sin(DOUBLE a)	DOUBLE	返回给定角度a的正弦值。
sqrt	sqrt(DOUBLE a)	DOUBLE	返回数值的平方根。
tan	tan(DOUBLE a)	DOUBLE	返回给定角度a的正切值。

14.3.2 abs

abs函数用于计算入参的绝对值。

命令格式

```
abs(DOUBLE a)
```

参数说明

表 14-75 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE或INT类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回NULL。

```
select abs(null);
```

返回1。

```
select abs(-1);
```

返回3.1415926。

```
select abs(-3.1415926);
```


版权所有 © 华为技术有限公司

14.3.3 acos

acos函数用于返回给定角度a的反余弦值。

命令格式

```
acos(DOUBLE a)
```

参数说明

表 14-76 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a取值范围为[-1,1]，a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型，值在0~π之间。

说明

- a的值不在[-1,1]范围内时，返回NaN。
- a为NULL，则返回NULL。

示例代码

返回3.141592653589793。

```
select acos(-1);
```

返回0。

```
select acos(1);
```

返回NULL。

```
select acos(null);
```

返回NAN。

```
select acos(10);
```

14.3.4 asin

asin函数用于返回给定角度a的反正弦值。

命令格式

```
asin(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-77 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a取值范围为[-1,1]，a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型，值在-π/2~π/2之间。

说明

- a的值不在[-1,1]范围内时，返回NaN。
- a为NULL，则返回NULL。

示例代码

返回1.5707963267948966。

```
select asin(1);
```

返回0.6435011087932844。

```
select asin(0.6);
```

返回NULL。

```
select asin(null);
```

返回NAN。

```
select asin(10);
```

14.3.5 atan

atan函数用于返回给定角度a的反正切值。

命令格式

```
atan(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-78 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型，值在-π/2~π/2之间。

说明

- a的值不在[-1,1]范围内时，返回NaN。
- a为NULL，则返回NULL。

示例代码

返回0.7853981633974483。

```
select atan(1);
```

返回0.5404195002705842。

```
select atan(0.6);
```

返回NULL。

```
select atan(null);
```

14.3.6 bin

bin函数用于返回a的二进制格式。

命令格式

```
bin(BIGINT a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-79 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式为整数格式。 参数a非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。

返回值说明

返回STRING类型。

说明

a值为NULL时，返回NULL。

示例代码

返回1。

```
select bin(1);
```

返回NULL。

```
select bin(null);
```

返回1000。

```
select bin(8);
```

返回1000。

```
select bin(8.123456);
```

14.3.7 bround

bruond函数用于返回一个数值，该数值是按照指定d位小数进行四舍五入运算的结果。

命令格式

```
bround(DOUBLE a, INT d)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-80 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 代表需要被四舍五入的值。 该命令与传统四舍五入方式的区别在于，对数字5进行操作时，由前一位数字来决定，前一位数字为奇数，增加一位，前一位数字为偶数，舍弃一位。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。
d	否	DOUBLE、BIGINT、DECIMAL、STRING类型。	代表需要四舍五入到的位数。 参数d非INT类型时，会隐式转换为INT类型后参与运算。

返回值说明

返回DOUBLE类型。

说明

a或d值为NULL时，返回NULL。

示例代码

返回123.4。

```
select bround(123.45,1);
```

返回123.6。

```
select bround(123.55,1);
```

返回NULL。

```
select bround(null);
```

返回123.457。

```
select bround(123.456789,3.123456);
```

14.3.8 cbrt

cbrt函数用返回a的立方根。

命令格式

```
cbrt(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-81 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型。

说明

a为NULL，则返回NULL。

示例代码

返回3。

```
select cbrt(27);
```

返回3.3019272488946267。

```
select cbrt(36);
```

返回NULL。

```
select cbrt(null);
```

14.3.9 ceil

ceil函数用于对a进行向上舍入最接近的整数。

命令格式

```
ceil(DOUBLE a)
```

参数说明

表 14-82 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

 版权所有 © 华为技术有限公司

返回值说明

返回DECIMAL类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回2。

```
select ceil(1.3);
```

返回-1。

```
select ceil(-1.3);
```

返回NULL。

```
select ceil(null);
```

14.3.10 conv

conv函数用于进制转换，将from_base进制下的num转化为to_base进制下面的数。

命令格式

```
conv(BIGINT num, INT from_base, INT to_base)
```

参数说明

表 14-83 参数说明

参数	是否必选	参数类型	说明
num	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	需要进行转换进制的数。 参数num格式为浮点数格式、整数格式、字符串格式。
from_base	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	被转换的进制from_base。 参数from_base格式为浮点数格式、整数格式、字符串格式。
to_base	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	转化至的进制to_base。 参数to_base格式为浮点数格式、整数格式、字符串格式。

 版权所有 © 华为技术有限公司

返回值说明

返回STRING类型。

说明

- num、from_base或to_base值为NULL时，返回NULL。
- 转换过程以64位精度工作，溢出时返回NULL。
- num如果输入的是小数，会转为整数值后进行进制转换，小数部分会被舍弃。

示例代码

-返回8。

```
select conv('1000', 2, 10);
```

返回B。

```
select conv('1011', 2, 16);
```

返回703710。

```
select conv('ABCDE', 16, 10);
```

返回27。

```
select conv(1000.123456, 3.123456, 10.123456);
```

返回18446744073709551589。

```
select conv(-1000.123456, 3.123456, 10.123456);
```

返回NULL。

```
select conv('1100', null, 10);
```

14.3.11 cos

cos函数用于计算a的余弦值，输入为弧度

命令格式

```
cos(DOUBLE a)
```

参数说明

表 14-84 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

 版权所有 © 华为技术有限公司

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回-0.9999999999999986

```
select cos(3.1415926);
```

返回NULL。

```
select cos(null);
```

14.3.12 cot1

cot1函数用于计算a的余切值，输入为弧度。

命令格式

```
cot1(DOUBLE a)
```

参数说明

表 14-85 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE或DECIMAL类型。

说明

a为NULL，则返回NULL。

示例代码

返回1.0000000000000002。

```
select cot1(pi()/4);
```

返回NULL。

```
select cot1(null);
```

 版权所有 © 华为技术有限公司

14.3.13 degrees

degress函数用于计算返回弧度所对应的角度。

命令格式

```
degrees(DOUBLE a)
```

参数说明

表 14-86 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回90.0。

```
select degrees(1.5707963267948966);
```

返回0。

```
select degrees(0);
```

返回NULL。

```
select degrees(null);
```

14.3.14 e

e函数用于计算返回e的值。

命令格式

```
e()
```

返回值说明

返回DOUBLE类型的值。

 版权所有 © 华为技术有限公司

示例代码

返回2.718281828459045。

```
select e();
```

14.3.15 exp

abs函数用于计算返回e的a次方的值。

命令格式

```
exp(DOUBLE a)
```

参数说明

表 14-87 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回7.38905609893065。

```
select exp(2);
```

返回20.085536923187668。

```
select exp(3);
```

返回NULL。

```
select exp(null);
```

14.3.16 factorial

factorial函数用于返回a的阶乘。

命令格式

```
factorial(INT a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-88 参数说明

参数	是否必选	参数类型	说明
a	是	BIGINT、INT、SMALLINT、TINYINT类型。	参数a的格式为整数格式。 参数a非INT类型时，会隐式转换为INT类型后参与运算。 字符串会转为对应的ASCII码。

返回值说明

返回BIGINT类型。

说明

- a值为0时，返回1。
- a值为NULL或[0,20]之外的值，返回NULL。

示例代码

返回720。

```
select factorial(6);
```

返回1。

```
select factorial(1);
```

返回120。

```
select factorial(5.123456);
```

返回NULL。

```
select factorial(null);
```

返回NULL。

```
select factorial(21);
```

14.3.17 floor

floor函数用于对a进行向下舍入最接近的整数。

命令格式

```
floor(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-89 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回BIGINT类型。

说明

a为NULL，则返回NULL。

示例代码

返回1。

```
select floor(1.2);
```

返回-2。

```
select floor(-1.2);
```

返回NULL。

```
select floor(null);
```

14.3.18 greatest

greatest函数用于返回列表中的最大值。

命令格式

```
greatest(T v1, T v2, ...)
```

参数说明

表 14-90 参数说明

参数	是否必选	参数类型	说明
v1	是	DOUBLE、BIGINT、DECIMAL类型。	参数v1的格式包括浮点数格式、整数格式。

 版权所有 © 华为技术有限公司

参数	是否必选	参数类型	说明
v2	是	DOUBLE、BIGINT、DECIMAL类型。	参数v2的格式包括浮点数格式、整数格式。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回4.0。

```
select greatest(1,2.0,3,4.0);
```

返回NULL。

```
select greatest(null);
```

14.3.19 hex

hex函数用于将整数或字符转换为十六进制格式。

命令格式

```
hex(BIGINT a)
```

参数说明

表 14-91 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。 字符串会转为对应的ASCII码。

返回值说明

返回STRING类型。

 版权所有 © 华为技术有限公司

说明

- a值为0时，返回0。
- a值为NULL时，返回NULL。

示例代码

返回0。

```
select hex(0);
```

返回61。

```
select hex('a');
```

返回10。

```
select hex(16);
```

返回31。

```
select hex('1');
```

返回3136。

```
select hex('16');
```

返回NULL。

```
select hex(null);
```

14.3.20 least

least函数用于返回列表中的最小值。

命令格式

```
least(T v1, T v2, ...)
```

参数说明

表 14-92 参数说明

参数	是否必选	参数类型	说明
v1	是	DOUBLE、BIGINT、DECIMAL类型。	参数v1的格式包括浮点数格式、整数格式。
v2	是	DOUBLE、BIGINT、DECIMAL类型。	参数v2的格式包括浮点数格式、整数格式。

 版权所有 © 华为技术有限公司

返回值说明

返回DOUBLE类型的值。

说明

- v1、v2...为String类型时，返回报错。
- 所有参数都为NULL时，返回NULL。

示例代码

返回1.0。

```
select least(1,2.0,3,4.0);
```

返回NULL。

```
select least(null);
```

14.3.21 ln

ln函数用于返回给定数值的自然对数。

命令格式

```
ln(DOUBLE a)
```

参数说明

表 14-93 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

- a值为负数或0时，返回NULL。
- a值为NULL时，返回NULL。

示例代码

返回1.144729868791239。

```
select ln(3.1415926);
```

 版权所有 © 华为技术有限公司

返回1。

```
select ln(2.718281828459045);
```

返回NULL。

```
select ln(null);
```

14.3.22 log

log函数根据给定底数及指数返回自然对数。

命令格式

```
log(DOUBLE base, DOUBLE a)
```

参数说明

表 14-94 参数说明

参数	是否必选	参数类型	说明
base	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数base的格式包括浮点数格式、整数格式、字符串格式。 参数base非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

- base或a为NULL时，返回NULL。
- base或a为负数或0时，返回NULL。
- 如果base为1（会引发一个除零行为），会返回NULL。

示例代码

返回2。

```
select log(2, 4);
```

返回NULL。

```
select log(2, null);
```

 版权所有 © 华为技术有限公司

返回NULL。

```
select log(null, 4);
```

14.3.23 log10

log10函数用于返回给定数值的以10为底自然对数。

命令格式

```
log10(DOUBLE a)
```

参数说明

表 14-95 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a值为0、负数或NULL时，返回NULL。

示例代码

返回NULL。

```
select log10(null);
```

返回NULL。

```
select log10(0);
```

返回0.9542425094393249。

```
select log10(9);
```

返回1。

```
select log10(10);
```

14.3.24 log2

log2函数用于返回给定数值的以2为底自然对数。

 版权所有 © 华为技术有限公司

命令格式

```
log2(DOUBLE a)
```

参数说明

表 14-96 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a值为0、负数或NULL时，返回NULL。

示例代码

返回NULL。

```
select log2(null);
```

返回NULL。

```
select log2(0);
```

返回3.1699250014423126。

```
select log2(9);
```

返回4。

```
select log2(16);
```

14.3.25 median

median函数用于计算入参的中位数。

命令格式

```
median(colname)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-97 参数说明

参数	是否必选	参数类型	说明
colname	是	DOUBLE、DECIMAL、STRING、BIGINT类型。	代表需要排序的列名。 列中元素为DOUBLE类型。 当列中元素非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE或DECIMAL类型。

说明

列名不存在时，返回报错。

示例代码

假设列int_test中的元素为1、2、3、4，类型为INT类型。

返回2.5。

```
select median(int_test) FROM int_test;
```

14.3.26 negative

negative函数用于返回a的相反数。

命令格式

```
negative(INT a)
```

参数说明

表 14-98 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。

 版权所有 © 华为技术有限公司

返回值说明

返回DECIMAL或INT类型。

说明

a为NULL，则返回NULL。

示例代码

返回-1。

```
SELECT negative(1);
```

返回3。

```
SELECT negative(-3);
```

14.3.27 percentile

percentile函数用于计算精确百分位数，适用于小数据量。先对指定列升序排列，然后取第p位百分数的精确值。

命令格式

```
percentile(colname,DOUBLE p)
```

参数说明

表 14-99 参数说明

参数	是否必选	参数类型	说明
colname	是	STRING类型	代表需要排序的列名。 列中元素只能为整数类型。
p	是	DOUBLE类型	p的范围为0-1。参数p的格式包括浮点数格式。

返回值说明

返回DOUBLE或ARRAY类型。

说明

- 列名不存在时，返回报错。
- p为NULL或在[0,1]之外时，返回报错。

示例代码

假设列int_test中的元素为1、2、3、4，类型为INT类型。

返回3.0999999999999996。

```
select percentile(int_test,0.7) FROM int_test;
```

 版权所有 © 华为技术有限公司

返回3.997。

```
select percentile(int_test,0.999) FROM int_test;
```

返回2.5。

```
select percentile(int_test,0.5) FROM int_test;
```

返回[1.3,1.9,2.5,2.8,3.7]。

```
select percentile (int_test,ARRAY(0.1,0.3,0.5,0.6,0.9)) FROM int_test;
```

14.3.28 percentile_approx

percentile_approx函数用于计算近似百分位数，适用于大数据量。先对指定列升序排列，然后取第p位百分数最靠近的值。

命令格式

```
percentile_approx (colname,DOUBLE p)
```

参数说明

表 14-100 参数说明

参数	是否必选	参数类型	说明
colname	是	STRING类型	代表需要排序的列名。 列中元素为DOUBLE类型。当列中元素非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。
p	是	DOUBLE类型	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数p的范围为0-1。参数p的格式包括浮点数格式。

返回值说明

返回DOUBLE类型或ARRAY类型的值。

说明

- 列名不存在时，返回报错。
- p为NULL或在[0,1]之外时，返回报错。

示例代码

假设列int_test中的元素为1、2、3、4，类型为INT类型。

返回3。

```
select percentile_approx(int_test,0.7) FROM int_test;
```

返回3。

 版权所有 © 华为技术有限公司

```
select percentile_approx(int_test,0.75) FROM int_test;
```

返回2。

```
select percentile_approx(int_test,0.5) FROM int_test;
```

返回[1,2,2,3,4]。

```
select percentile_approx (int_test,ARRAY(0.1,0.3,0.5,0.6,0.9)) FROM int_test;
```

14.3.29 pi

pi函数用于返回π的值。

命令格式

```
pi()
```

返回值说明

返回DOUBLE类型的值。

示例代码

返回3.141592653589793。

```
select pi();
```

14.3.30 pmod

pmod函数用于返回a除b的余数的绝对值。

命令格式

pmod(INT a, INT b)

参数说明

表 14-101 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。
b	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数b的格式包括浮点数格式、整数格式、字符串格式。

 版权所有 © 华为技术有限公司

返回值说明

返回DECIMAL或INT类型。

说明

- a或b为NULL，则返回NULL。
- b为0时，返回NULL

示例代码

返回2。

```
select pmod(2,5);
```

返回3。

```
select pmod(-2,5)（解析：-2=5*(-1)...3）;
```

返回NULL。

```
select pmod(5,0);
```

返回1。

```
select pmod(5,2);
```

返回0.877。

```
select pmod(5.123,2.123);
```

14.3.31 positive

positive函数用于返回a的值。

命令格式

```
positive(INT a)
```

参数说明

表 14-102 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。

返回值说明

返回 DECIMAL、DOUBLE或INT类型。

 版权所有 © 华为技术有限公司

说明

a为NULL，则返回NULL。

示例代码

返回3。

```
SELECT positive(3);
```

返回-3。

```
SELECT positive(-3);
```

返回123。

```
SELECT positive('123');
```

14.3.32 pow

pow函数用于计算返回a的p次幂。

命令格式

```
pow(DOUBLE a, DOUBLE p),
power(DOUBLE a, DOUBLE p)
```

参数说明

表 14-103 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。
p	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数p的格式包括浮点数格式、整数格式、字符串格式。 参数p非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a、p为NULL，则返回NULL。

 版权所有 © 华为技术有限公司

示例代码

返回16。

```
select pow(2, 4);
```

返回NULL。

```
select pow(2, null);
```

返回17.429460393524256。

```
select pow(2, 4.123456);
```

14.3.33 radians

radians函数用于返回角度所对应的弧度。

命令格式

```
radians(DOUBLE a)
```

参数说明

表 14-104 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回1.0471975511965976。

```
select radians(60);
```

返回0。

```
select radians(0);
```

返回NULL。

```
select radians(null);
```

 版权所有 © 华为技术有限公司

14.3.34 rand

rand函数用于返回大于或等于0且小于1的平均分布随机数。

命令格式

```
rand(INT seed)
```

参数说明

表 14-105 参数说明

参数	是否必选	参数类型	说明
seed	否	INT类型。	参数seed的格式包括浮点数格式、整数格式、字符串格式。 如果指定种子seed，在相同运行环境下，将会得到一个稳定的随机数序列。

返回值说明

返回DOUBLE类型的值。

示例代码

返回0.3668915240363728。

```
select rand();
```

返回0.25738143505962285。

```
select rand(3);
```

14.3.35 round

round函数用于计算a的四舍五入到d位的值。

命令格式

```
round(DOUBLE a, INT d)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-106 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	代表需要被四舍五入的值。 参数a的格式包括浮点数格式、整数格式、字符串格式。
d	否	INT类型。	默认值：0。 代表需要四舍五入到的位数。 参数d非INT类型时，会隐式转换为INT类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

- d为负数时，返回报错。
- a或d值为NULL时，返回NULL。

示例代码

返回123.0。

```
select round(123.321);
```

返回123.4。

```
select round(123.396, 1);
```

返回NULL。

```
select round(null);
```

返回123.321。

```
select round(123.321, 4);
```

返回123.3。

```
select round(123.321,1.33333);
```

返回123.3。

```
select round(123.321,1.33333);
```

14.3.36 shiftleft

shiftleft函数用于有符号左移，将a的二进制数按位左移b位。

 版权所有 © 华为技术有限公司

命令格式

shiftleft(BIGINT a, BIGINT b)

参数说明

表 14-107 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 当参数a非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。
b	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 当参数b非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。

返回值说明

返回INT类型。

说明

a或b值为NULL时，返回NULL。

示例代码

返回8。

```
select shiftleft(1,3);
```

返回48。

```
select shiftleft(6,3);
```

返回48。

```
select shiftleft(6.123456,3.123456);
```

返回NULL。

```
select shiftleft(null,3);
```

14.3.37 shiftright

shiftright函数用于有符号右移，将a的二进制数按位右移b位。

 版权所有 © 华为技术有限公司

命令格式

```
shiftright(BIGINT a, BIGINT b)
```

参数说明

表 14-108 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 当参数a非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。
b	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数b的格式包括浮点数格式、整数格式、字符串格式。 当参数b非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。

返回值说明

返回INT类型。

说明

a或b值为NULL时，返回NULL。

示例代码

返回2。

```
select shiftright(16,3);
```

返回4。

```
select shiftright(36,3);
```

返回4。

```
select shiftright(36.123456,3.123456);
```

返回NULL。

```
select shiftright(null,3);
```

14.3.38 shiftrightunsigned

shiftrightunsigned函数用于无符号右移，将a的二进制数按位右移b位。

命令格式

```
shiftrightunsigned(BIGINT a, BIGINT b)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-109 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 当参数a非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。
b	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数b的格式包括浮点数格式、整数格式、字符串格式。 当参数b非BIGINT类型时，会隐式转换为BIGINT类型后参与运算。

返回值说明

返回INT类型。

说明

a或b值为NULL时，返回NULL。

示例代码

返回2。

```
select shiftrightunsigned(16,3);
```

返回536870910。

```
select shiftrightunsigned(-16,3);
```

返回2。

```
select shiftrightunsigned(16.123456,3.123456);
```

返回NULL。

```
select shiftrightunsigned(null,3);
```

14.3.39 sign

sign函数用于返回a所对应的正负号。

命令格式

```
sign(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-110 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。

返回值说明

返回DOUBLE类型。

说明

- a值为正数时，返回1。
- a值为负数时，返回-1。
- a值为0时，返回0。
- a值为NULL时，返回NULL。

示例代码

返回-1。

```
select sign(-3);
```

返回1。

```
select sign(3);
```

返回0。

```
select sign(0);
```

返回1。

```
select sign(3.1415926);
```

返回NULL。

```
select sign(null);
```

14.3.40 sin

sin函数用于计算a的正弦值，输入为弧度。

命令格式

```
sin(DOUBLE a)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-111 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回1。

```
select sin(pi()/2);
```

返回NULL。

```
select sin(null);
```

14.3.41 sqrt

sqrt函数用于返回数值的平方根。

命令格式

```
sqrt(DOUBLE a)
```

参数说明

表 14-112 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

 版权所有 © 华为技术有限公司

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回2.8284271247461903。

```
select sqrt(8);
```

返回4。

```
select sqrt(16);
```

返回NULL。

```
select sqrt(null);
```

14.3.42 tan

tan函数用于计算a的正切值，输入为弧度。

命令格式

```
tan(DOUBLE a)
```

参数说明

表 14-113 参数说明

参数	是否必选	参数类型	说明
a	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	参数a的格式包括浮点数格式、整数格式、字符串格式。 参数a非DOUBLE类型时，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

返回0.9999999999999999。

```
select tan(pi()/4);
```

 版权所有 © 华为技术有限公司

返回NULL。

```
select tan(null);
```

14.4 聚合函数

14.4.1 聚合函数概览

DLI所支持的聚合函数如**聚合函数表**所示。

表 14-114 聚合函数表

函数	命令格式	返回值	功能简介
avg	avg(col), avg(DISTINCT col)	DOUBLE	求平均值。
corr	corr(col1, col2)	DOUBLE	返回两列数值的相关系数。
count	count([distinct|all] <colname>)	BIGINT	返回记录条数。
covar_pop	covar_pop(col1, col2)	DOUBLE	返回两列数值协方差。
covar_samp	covar_samp(col1, col2)	DOUBLE	返回两列数值样本协方差。
max	max(col)	DOUBLE	返回最大值。
min	min(col)	DOUBLE	返回最小值。
percentile	percentile(BIGINT col, p)	DOUBLE	返回数值区域的百分比数值点。0<=P<=1,否则返回NULL,不支持浮点型数值。
percentile_approx	percentile_approx(DOUBLE col, p [, B])	DOUBLE	返回组内数字列近似的第p位百分数（包括浮点数），p值在[0,1]之间。参数B控制近似的精确度，B值越大，近似度越高，默认值为10000。当列中非重复值的数量小于B时，返回精确的百分数。
stddev_pop	stddev_pop(col)	DOUBLE	返回指定列的偏差。
stddev_samp	stddev_samp(col)	DOUBLE	返回指定列的样本偏差。
sum	sum(col), sum(DISTINCT col)	DOUBLE	求和。
variance/var_pop	variance(col), var_pop(col)	DOUBLE	返回列的方差。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
var_samp	var_samp(col)	DOUBLE	返回指定列的样本方差。

14.4.2 avg

avg函数用于计算求平均值。

命令格式

```
avg(col), avg(DISTINCT col)
```

参数说明

表 14-115 参数说明

参数	是否必选	参数类型	说明
col	是	所有数据类型	列值支持所有数据类型，可以转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

如果col值为NULL时，该列不参与计算。

示例代码

- 计算所有仓库的平均商品数（items）。命令示例如下：
  ```
  select avg(items) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  100.0
  ```
- 与group by配合使用，计算每个仓库中所有商品的平均库存。命令示例如下：
  ```
  select warehouseId, avg(items) from warehourse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1    155
  city2    101
  city3    194
  ```

14.4.3 corr

corr函数用于返回两列数值的相关系数。

 版权所有 © 华为技术有限公司

命令格式

```
corr(col1, col2)
```

参数说明

表 14-116 参数说明

参数	是否必选	参数类型	说明
col1	是	DOUBLE、BIGINT、INT、SMALLINT、TINYINT、FLOAT、DECIMAL类型	数据类型为数值的列。其他类型返回NULL。
col2	是	DOUBLE、BIGINT、INT、SMALLINT、TINYINT、FLOAT、DECIMAL类型	数据类型为数值的列。其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）和价格（price）的相关系数。命令示例如下：
  ```
  select corr(items,price) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  1.242355
  ```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）和价格（price）的相关系数。命令示例如下：
  ```
  select warehouseId, corr(items,price) from warehourse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1  0.43124
  city2  0.53344
  city3  0.73425
  ```

14.4.4 count

count函数用于返回记录条数。

 版权所有 © 华为技术有限公司

命令格式

```
count([distinct|all] <colname>)
```

参数说明

表 14-117 参数说明

参数	是否必选	说明
distinct或all	否	表示在计数时是否去除重复记录，默认为all，即计算全部记录。 如果指定distinct，则只计算唯一值数量。
colname	是	列值可以为任意类型。 colname可以为*，即count(*)，返回所有行数。

返回值说明

返回BIGINT类型。

说明

colname值为NULL时，该行不参与计算。

示例代码

- 计算所有仓库表中的记录数。命令示例如下：
  ```
  select count(*) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  10
  ```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，计算各仓库（warehouseId）的商品数。命令示例如下：
  ```
  select warehouseId, count(*) from warehouse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1   6
  city2   5
  city3   6
  ```
  示例3：通过distinct去重，计算仓库数量。命令示例如下：
  ```
  select count(distinct warehouseId) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
   3
  ```

14.4.5 covar_pop

covar_pop函数用于返回两列数值协方差。

命令格式

```
covar_pop(col1, col2)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-118 参数说明

参数	是否必选	说明
col1	是	数据类型为数值的列。其他类型返回NULL。
col2	是	数据类型为数值的列。其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）和价格（price）的协方差。命令示例如下：
  ```
  select covar_pop(items，price) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  1.242355
  ```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）和价格（price）的协方差。命令示例如下：
  ```
  select warehouseId, covar_pop(items，price) from warehourse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1    1.13124
  city2    1.13344
  city3    1.53425
  ```

14.4.6 covar_samp

covar_samp函数用于返回两列数值样本协方差。

命令格式

```
covar_samp(col1, col2)
```

参数说明

表 14-119 参数说明

参数	是否必选	说明
col1	是	数据类型为数值的列。其他类型返回NULL。
col2	是	数据类型为数值的列。其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

 版权所有 © 华为技术有限公司

示例代码

- 计算所有商品库存（items）和价格（price）的样本协方差。命令示例如下：
```
select covar_samp(items,price) from warehouse;
```
返回结果如下：
```
_c0
1.242355
```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）和价格（price）的样本协方差。命令示例如下：
```
select warehouseId, covar_samp(items,price) from warehourse group by warehouseId;
```
返回结果如下：
```
warehouseId _c1
city1    1.03124
city2    1.03344
city3    1.33425
```

14.4.7 max

max函数用于返回最大值。

命令格式

```
max(col)
```

参数说明

表 14-120 参数说明

参数	是否必选	参数类型	说明
col	是	除BOOLEAN外的任意类型。	列值可以为除BOOLEAN外的任意类型。

返回值说明

返回DOUBLE类型的值。

说明

返回值的类型与col类型相同。返回规则如下：

- col值为NULL时，该行不参与计算。
- col为BOOLEAN类型时，不允许参与运算。

示例代码

- 计算所有商品的最高库存（items）。命令示例如下：
```
select max(items) from warehouse;
```
返回结果如下：
```
_c0
900
```
- 与group by配合使用，求每个仓库的最高库存。命令示例如下：
```
select warehouseId, max(items) from warehouse group by warehouseId;
```

 版权所有 © 华为技术有限公司

返回结果如下：

```
warehouseId _c1
city1    200
city2    300
city3    400
```

14.4.8 min

min函数用于返回最小值。

命令格式

```
min(col)
```

参数说明

表 14-121 参数说明

参数	是否必选	参数类型	说明
col	是	除BOOLEAN外的任意类型。	列值可以为除BOOLEAN外的任意类型。

返回值说明

返回DOUBLE类型的值。

说明

返回值的类型与col类型相同。返回规则如下：

- col值为NULL时，该行不参与计算。
- col为BOOLEAN类型时，不允许参与运算。

示例代码

- 计算所有商品的最低库存（items）。命令示例如下：
  ```
  select min(items) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  600
  ```
- 与group by配合使用，求每个仓库的最低库存。命令示例如下：
  ```
  select warehouseId, min(items) from warehouse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1    15
  city2    10
  city3    19
  ```

14.4.9 percentile

percentile函数用于返回数值区域的百分比数值点。

 版权所有 © 华为技术有限公司

命令格式

```
percentile(BIGINT col, p)
```

参数说明

表 14-122 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。其他类型返回NULL。
p	是	0<=P<=1,否则返回NULL。

返回值说明

返回DOUBLE类型的值。

说明

0<=P<=1,否则返回NULL。

示例代码

- 计算所有商品库存（items）的 0.5 百分位。命令示例如下：
  ```
  select percentile(items,0.5) from warehouse;
  ```
  返回结果如下：
  ```
  +------------+
  | _c0        |
  +------------+
  | 500.6      |
  +------------+
  ```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的 0.5 百分位。命令示例如下：
  ```
  select warehouseId, percentile(items, 0.5) from warehourse group by warehouseId;
  ```
  返回结果如下：
  ```
  +------------+------------+
  | warehouseId| _c1        |
  +------------+------------+
  | city1    | 499.6      |
  | city2    | 354.8      |
  | city3    | 565.7      |
  +------------+------------+
  ```

14.4.10 percentile_approx

percentile_approx函数用于返回组内数字列近似的第p位百分数（包括浮点数）。

命令格式

```
percentile_approx(DOUBLE col, p [, B])
```

 版权所有 © 华为技术有限公司

参数说明

表 14-123 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。其他类型返回NULL。
p	是	0<=P<=1,否则返回NULL。
B	是	参数B控制近似的精确度，B值越大，近似度越高，默认值为10000。当列中非重复值的数量小于B时，返回精确的百分数。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）的 0.5 百分位，精确度100。命令示例如下：
```
select PERCENTILE_APPROX(items,0.5，100) from warehouse;
```
返回结果如下：
```
+------------+
| _c0        |
+------------+
| 521        |
+------------+
```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的 0.5 百分位，精确度100。命令示例如下：
```
select warehouseId, PERCENTILE_APPROX(items, 0.5, 100) from warehourse group by warehouseId;
```
返回结果如下：
```
+------------+------------+
| warehouseId| _c1        |
+------------+------------+
| city1      | 499      |
| city2      | 354      |
| city3      | 565      |
+------------+------------+
```

14.4.11 stddev_pop

stddev_pop函数用于返回指定列的偏差。

命令格式

```
stddev_pop(col)
```

 版权所有 © 华为技术有限公司

参数说明

表 14-124 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）的偏差。命令示例如下：
  ```
  select stddev_pop(items) from warehouse;
  ```
  返回结果如下：
  ```
  _c0
  1.342355
  ```
- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的偏差。命令示例如下：
  ```
  select warehouseId, stddev_pop(items) from warehourse group by warehouseId;
  ```
  返回结果如下：
  ```
  warehouseId _c1
  city1    1.23124
  city2    1.23344
  city3    1.43425
  ```

14.4.12 stddev_samp

stddev_samp函数用于返回指定列的样本偏差。

命令格式

```
stddev_samp(col)
```

参数说明

表 14-125 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

 版权所有 © 华为技术有限公司

示例代码

- 计算所有商品库存（items）的样本偏差。命令示例如下：

```
select stddev_samp(items) from warehouse;
```

返回结果如下：

```
+------------+
| _c0        |
+------------+
| 1.342355   |
+------------+
```

- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的样本偏差。命令示例如下：

```
select warehouseId, stddev_samp(items) from warehourse group by warehouseId;
```

返回结果如下：

```
+------------+------------+
| warehouseId| _c1        |
+------------+------------+
| city1    | 1.23124   |
| city2    | 1.23344   |
| city3    | 1.43425   |
+------------+------------+
```

14.4.13 sum

sum函数用于计算求和。

命令格式

```
sum(col),
sum(DISTINCT col)
```

参数说明

表 14-126 参数说明

参数	是否必选	说明
col	是	列值支持所有数据类型，可以转换为DOUBLE类型后参与运算。 列值可以为DOUBLE、DECIMAL或BIGINT类型。 如果输入为STRING类型，会隐式转换为DOUBLE类型后参与运算。

返回值说明

返回DOUBLE类型的值。

说明

如果col值为NULL时，该行不参与计算。

示例代码

- 计算所有仓库的商品（items）总和。命令示例如下：

 版权所有 © 华为技术有限公司

```
select sum(items) from warehouse;
```

返回结果如下：

```
_c0
55357
```

- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，计算各仓库商品的总数（items）总和。命令示例如下：

```
select warehouseId, sum(items) from warehouse group by warehouseId;
```

返回结果如下：

```
warehouseId| _c1
city1    15500
city2    10175
city3    19400
```

14.4.14 variance/var_pop

variance/var_pop函数用于返回列的方差。

命令格式

```
variance(col),
var_pop(col)
```

参数说明

表 14-127 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。 参数为其他类型的列返回NULL。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）的方差。命令示例如下：

```
select variance(items) from warehouse;
--等效于如下语句。
select var_pop(items) from warehouse;
```

返回结果如下：

```
_c0
203.42352
```

- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的方差。命令示例如下：

```
select warehouseId, variance(items) from warehourse group by warehouseId;
--等效于如下语句。
select warehouseId, var_pop(items) from warehourse group by warehouseId;
```

返回结果如下：

```
warehouseId _c1
city1    19.23124
```

 版权所有 © 华为技术有限公司

```
city2    17.23344
city3    12.43425
```

14.4.15 var_samp

var_samp函数用于返回指定列的样本方差。

命令格式

```
var_samp(col)
```

参数说明

表 14-128 参数说明

参数	是否必选	说明
col	是	数据类型为数值的列。 其他类型返回NULL。

返回值说明

返回DOUBLE类型的值。

示例代码

- 计算所有商品库存（items）的样本方差。命令示例如下：

  ```
  select var_samp(items) from warehouse;
  ```

  返回结果如下：

  ```
  _c0
  294.342355
  ```

- 与group by配合使用，对所有商品按照仓库（warehouseId）进行分组，并计算同组商品库存（items）的样本方差。命令示例如下：

  ```
  select warehouseId, var_samp(items) from warehourse group by warehouseId;
  ```

  返回结果如下：

  ```
  warehouseId _c1
  city1    18.23124
  city2    16.23344
  city3    11.43425
  ```

14.5 分析窗口函数

14.5.1 分析窗口函数概览

DLI所支持的分析窗口函数如**分析窗口函数介绍**所示。

 版权所有 © 华为技术有限公司

表 14-129 分析窗口函数介绍

函数	命令格式	返回值	功能简介
cume_dist	cume_dist()	DOUBLE	用于求累计分布，相当于求分区中大于等于或小于等于当前行的数据在分区中的占比。
first_value	first_value(col)	参数的数据类型	返回结果集中某列第一条数据的值。
last_value	last_value(col)	参数的数据类型	返回结果集中某列最后一条数据的值。
lag	lag (col,n,DEFAULT)	参数的数据类型	用于统计窗口内往上第n行值。第一个参数为列名，第二个参数为往上第n行（可选，默认为1），第三个参数为默认值（当往上第n行为NULL时候，取默认值，如不指定，则为NULL）。
lead	lead (col,n,DEFAULT)	参数的数据类型	用于统计窗口内往下第n行值。第一个参数为列名，第二个参数为往下第n行（可选，默认为1），第三个参数为默认值（当往下第n行为NULL时候，取默认值，如不指定，则为NULL）。
percent_rank	percent_rank()	DOUBLE	为窗口的ORDER BY子句所指定列中值的返回秩，但以介于0和1之间的小数形式表示，计算方法为 (RANK - 1)/(- 1)。
rank	rank()	INT	计算一个值在一组值中的排位。如果出现并列的情况，RANK函数会在排名序列中留出空位。
row_number	row_number() over (order by col_1[,col_2 ...])	INT	为每一行指派一个唯一的编号。

14.5.2 cume_dist

cume_dist函数用于求累计分布，相当于求分区中大于等于或小于等于当前行的数据在分区中的占比。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

版权所有 © 华为技术有限公司

命令格式

```
cume_dist() over([partition_clause] [orderby_clause])
```

参数说明

表 14-130 参数说明

参数	是否必选	说明
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表salary，并添加数据，命令示例如下：

```
CREATE EXTERNAL TABLE salary (
dept STRING, -- 部门名称
userid string, -- 员工ID
sal INT -- 薪水
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
d1,user1,1000
d1,user2,2000
d1,user3,3000
d2,user4,4000
d2,user5,5000
```

- 统计小于等于当前薪水的人数占比

```
select dept, userid, sal,
      cume_dist() over(order by sal) as cume1
from salary;
--  结果：
d1 user1 1000 0.2
d1 user2 2000 0.4
d1 user3 3000 0.6
d2 user4 4000 0.8
d2 user5 5000 1.0
```

- 按部门分组统计小于等于当前薪水的人数的比例

```
select dept, userid, sal,
      cume_dist() over (partition by dept order by sal) as cume2
from salary;
-- 结果：
d1 user1 1000 0.3333333333333333
d1 user2 2000 0.6666666666666666
d1 user3 3000 1.0
```

 版权所有 © 华为技术有限公司

```
d2 user4 4000 0.5
d2 user5 5000 1.0
```

- 按照sal降序排序后，结果就是统计 大于等于 当前薪水的人数的比例

```
select dept, userid, sal,
      cume_dist() over(order by sal desc) as cume3
from salary;
-- 结果：
d2 user5 5000 0.2
d2 user4 4000 0.4
d1 user3 3000 0.6
d1 user2 2000 0.8
d1 user1 1000 1.0
select dept, userid, sal,
      cume_dist() over(partition by dept order by sal desc) as cume4
from salary;
-- 结果：
d1 user3 3000 0.3333333333333333
d1 user2 2000 0.6666666666666666
d1 user1 1000 1.0
d2 user5 5000 0.5
d2 user4 4000 1.0
```

14.5.3 first_value

first_value函数用于取当前行所对应窗口的第一条数据的值。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
first_value(<expr>[, <ignore_nulls>]) over ([partition_clause] [orderby_clause] [frame_clause])
```

参数说明

表 14-131 参数说明

参数	是否必选	说明
expr	是	待计算返回结果的表达式。
ignore_nulls	否	BOOLEAN类型，表示是否忽略NULL值。默认值为False。 当参数的值为True时，返回窗口中第一条非NULL的值。
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。
frame_clause	否	用于确定数据边界。

 版权所有 © 华为技术有限公司

返回值说明

参数的数据类型。

示例代码

示例数据

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。创建表logs，并添加数据，命令示例如下：

```
create table logs(
 cookieid string,
 createtime string,
 url string
)
STORED AS parquet;
```

添加数据如下：

```
cookie1 2015-04-10 10:00:02 url2
cookie1 2015-04-10 10:00:00 url1
cookie1 2015-04-10 10:03:04 url3
cookie1 2015-04-10 10:50:05 url6
cookie1 2015-04-10 11:00:00 url7
cookie1 2015-04-10 10:10:00 url4
cookie1 2015-04-10 10:50:01 url5
cookie2 2015-04-10 10:00:02 url22
cookie2 2015-04-10 10:00:00 url11
cookie2 2015-04-10 10:03:04 url33
cookie2 2015-04-10 10:50:05 url66
cookie2 2015-04-10 11:00:00 url77
cookie2 2015-04-10 10:10:00 url44
cookie2 2015-04-10 10:50:01 url55
```

示例：将所有记录根据cookieid分组，并按createtime升序排列，返回每组中的第一行数据。命令示例如下

```
SELECT cookieid, createtime, url,
      FIRST_VALUE(url) OVER (PARTITION BY cookieid ORDER BY createtime) AS first
FROM logs;
```

返回结果如下：

```
cookieid createtime         url  first
cookie1 2015-04-10 10:00:00 url1 url1
cookie1 2015-04-10 10:00:02 url2 url1
cookie1 2015-04-10 10:03:04 url3 url1
cookie1 2015-04-10 10:10:00 url4 url1
cookie1 2015-04-10 10:50:01 url5 url1
cookie1 2015-04-10 10:50:05 url6 url1
cookie1 2015-04-10 11:00:00 url7 url1
cookie2 2015-04-10 10:00:00 url11 url11
cookie2 2015-04-10 10:00:02 url22 url11
cookie2 2015-04-10 10:03:04 url33 url11
cookie2 2015-04-10 10:10:00 url44 url11
cookie2 2015-04-10 10:50:01 url55 url11
cookie2 2015-04-10 10:50:05 url66 url11
cookie2 2015-04-10 11:00:00 url77 url11
```

14.5.4 last_value

last_value函数用于取当前行所对应窗口的最后一条数据的值。

使用限制

窗口函数的使用限制如下：

 版权所有 © 华为技术有限公司

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
last_value(<expr>[, <ignore_nulls>]) over ([partition_clause] [orderby_clause] [frame_clause])
```

参数说明

表 14-132 参数说明

参数	是否必选	说明
expr	是	待计算返回结果的表达式。
ignore_nulls	否	BOOLEAN类型，表示是否忽略NULL值。默认值为False。 当参数的值为True时，返回窗口中第一条非NULL的值。
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。
frame_clause	否	用于确定数据边界。

返回值说明

参数的数据类型。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表logs，并添加数据，命令示例如下：

```
create table logs(
 cookieid string,
 createtime string,
 url string
)
STORED AS parquet;
```

添加数据如下：

```
cookie1 2015-04-10 10:00:02 url2
cookie1 2015-04-10 10:00:00 url1
cookie1 2015-04-10 10:03:04 url3
cookie1 2015-04-10 10:50:05 url6
cookie1 2015-04-10 11:00:00 url7
cookie1 2015-04-10 10:10:00 url4
cookie1 2015-04-10 10:50:01 url5
cookie2 2015-04-10 10:00:02 url22
cookie2 2015-04-10 10:00:00 url11
cookie2 2015-04-10 10:03:04 url33
cookie2 2015-04-10 10:50:05 url66
cookie2 2015-04-10 11:00:00 url77
```

 版权所有 © 华为技术有限公司

```
cookie2 2015-04-10 10:10:00 url44
cookie2 2015-04-10 10:50:01 url55
```

示例：将所有记录根据cookieid分组，并按createtime升序排列，返回每组中的最后一行数据。命令示例如下

```
SELECT cookieid, createtime, url,
    LAST_VALUE(url) OVER(PARTITION BY cookieid ORDER BY createtime) AS last
FROM logs;

-- 返回结果：
cookieid createtime         url last
cookie1 2015-04-10 10:00:00 url1 url1
cookie1 2015-04-10 10:00:02 url2 url2
cookie1 2015-04-10 10:03:04 url3 url3
cookie1 2015-04-10 10:10:00 url4 url4
cookie1 2015-04-10 10:50:01 url5 url5
cookie1 2015-04-10 10:50:05 url6 url6
cookie1 2015-04-10 11:00:00 url7 url7
cookie2 2015-04-10 10:00:00 url11 url11
cookie2 2015-04-10 10:00:02 url22 url22
cookie2 2015-04-10 10:03:04 url33 url33
cookie2 2015-04-10 10:10:00 url44 url44
cookie2 2015-04-10 10:50:01 url55 url55
cookie2 2015-04-10 10:50:05 url66 url66
cookie2 2015-04-10 11:00:00 url77 url77
```

说明

截止到当前行的最后一个值，其实就是它本身。

14.5.5 lag

lag函数用于用于统计窗口内往上第n行值。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
lag(<expr>[, bigint <offset>[, <default>]]) over([partition_clause] orderby_clause)
```

参数说明

表 14-133 参数说明

参数	是否必选	说明
expr	是	待计算返回结果的表达式。
offset	否	偏移量，BIGINT类型常量，取值大于等于0。值为0时表示当前行，为1时表示前一行，以此类推。默认值为1。输入值为STRING类型、DOUBLE类型则隐式转换为BIGINT类型后进行运算。

 版权所有 © 华为技术有限公司

参数	是否必选	说明
default	是	常量，默认值为NULL。 当offset指定的范围越界时的缺省值，需要与expr对应的数据类型相同。如果expr非常量，则基于当前行进行求值。
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

参数的数据类型。

示例代码

示例数据

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。创建表logs，并添加数据，命令示例如下：

```
create table logs(
 cookieid string,
 createtime string,
 url string
)
STORED AS parquet;
```

添加数据如下：

```
cookie1 2015-04-10 10:00:02 url2
cookie1 2015-04-10 10:00:00 url1
cookie1 2015-04-10 10:03:04 url3
cookie1 2015-04-10 10:50:05 url6
cookie1 2015-04-10 11:00:00 url7
cookie1 2015-04-10 10:10:00 url4
cookie1 2015-04-10 10:50:01 url5
cookie2 2015-04-10 10:00:02 url22
cookie2 2015-04-10 10:00:00 url11
cookie2 2015-04-10 10:03:04 url33
cookie2 2015-04-10 10:50:05 url66
cookie2 2015-04-10 11:00:00 url77
cookie2 2015-04-10 10:10:00 url44
cookie2 2015-04-10 10:50:01 url55
```

将所有记录根据cookieid分组，并按createtime升序排列，返回窗口内往上第2行的值。命令示例如下

示例1：

```
SELECT cookieid, createtime, url,
      LAG(createtime, 2) OVER (PARTITION BY cookieid ORDER BY createtime) AS last_2_time
FROM logs;
-- 返回结果：
cookieid createtime         url  last_2_time
cookie1 2015-04-10 10:00:00 url1 NULL
cookie1 2015-04-10 10:00:02 url2 NULL
cookie1 2015-04-10 10:03:04 url3 2015-04-10 10:00:00
cookie1 2015-04-10 10:10:00 url4 2015-04-10 10:00:02
cookie1 2015-04-10 10:50:01 url5 2015-04-10 10:03:04
```

 版权所有 © 华为技术有限公司

```
cookie1 2015-04-10 10:50:05 url6 2015-04-10 10:10:00
cookie1 2015-04-10 11:00:00 url7 2015-04-10 10:50:01
cookie2 2015-04-10 10:00:00 url11 NULL
cookie2 2015-04-10 10:00:02 url22 NULL
cookie2 2015-04-10 10:03:04 url33 2015-04-10 10:00:00
cookie2 2015-04-10 10:10:00 url44 2015-04-10 10:00:02
cookie2 2015-04-10 10:50:01 url55 2015-04-10 10:03:04
cookie2 2015-04-10 10:50:05 url66 2015-04-10 10:10:00
cookie2 2015-04-10 11:00:00 url77 2015-04-10 10:50:01
```

说明

说明：因为没有设置默认值，当没有上两行时显示为NULL。

示例2：

```
SELECT cookieid, createtime, url,
       LAG(createtime,1,'1970-01-01 00:00:00') OVER (PARTITION BY cookieid ORDER BY createtime) AS
last_1_time
FROM cookie4;
-- 结果：
cookieid createtime          url last_1_time
cookie1 2015-04-10 10:00:00 url1 1970-01-01 00:00:00 (显示默认值)
cookie1 2015-04-10 10:00:02 url2 2015-04-10 10:00:00
cookie1 2015-04-10 10:03:04 url3 2015-04-10 10:00:02
cookie1 2015-04-10 10:10:00 url4 2015-04-10 10:03:04
cookie1 2015-04-10 10:50:01 url5 2015-04-10 10:10:00
cookie1 2015-04-10 10:50:05 url6 2015-04-10 10:50:01
cookie1 2015-04-10 11:00:00 url7 2015-04-10 10:50:05
cookie2 2015-04-10 10:00:00 url11 1970-01-01 00:00:00 (显示默认值)
cookie2 2015-04-10 10:00:02 url22 2015-04-10 10:00:00
cookie2 2015-04-10 10:03:04 url33 2015-04-10 10:00:02
cookie2 2015-04-10 10:10:00 url44 2015-04-10 10:03:04
cookie2 2015-04-10 10:50:01 url55 2015-04-10 10:10:00
cookie2 2015-04-10 10:50:05 url66 2015-04-10 10:50:01
cookie2 2015-04-10 11:00:00 url77 2015-04-10 10:50:05
```

14.5.6 lead

lead函数用于用于统计窗口内往下第n行值。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
lead(<expr>[, bigint <offset>[, <default>]]) over([partition_clause] orderby_clause)
```

参数说明

表 14-134 参数说明

参数	是否必选	说明
expr	是	待计算返回结果的表达式。

 版权所有 © 华为技术有限公司

参数	是否必选	说明
offset	否	偏移量，BIGINT类型常量，取值大于等于0。值为0时表示当前行，为1时表示前一行，以此类推。默认值为1。输入值为STRING类型、DOUBLE类型则隐式转换为BIGINT类型后进行运算。
default	是	常量，默认值为NULL。 当offset指定的范围越界时的缺省值，需要与expr对应的数据类型相同。如果expr非常量，则基于当前行进行求值。
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

参数的数据类型。

示例代码

示例数据

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。创建表logs，并添加数据，命令示例如下：

```
create table logs(
 cookieid string,
 createtime string,
 url string
)
STORED AS parquet;
```

添加数据如下：

```
cookie1 2015-04-10 10:00:02 url2
cookie1 2015-04-10 10:00:00 url1
cookie1 2015-04-10 10:03:04 url3
cookie1 2015-04-10 10:50:05 url6
cookie1 2015-04-10 11:00:00 url7
cookie1 2015-04-10 10:10:00 url4
cookie1 2015-04-10 10:50:01 url5
cookie2 2015-04-10 10:00:02 url22
cookie2 2015-04-10 10:00:00 url11
cookie2 2015-04-10 10:03:04 url33
cookie2 2015-04-10 10:50:05 url66
cookie2 2015-04-10 11:00:00 url77
cookie2 2015-04-10 10:10:00 url44
cookie2 2015-04-10 10:50:01 url55
```

将所有记录根据cookieid分组，并按createtime升序排列，返回窗口内往下第2行和第1行的值。命令示例如下

```
SELECT cookieid, createtime, url,
      LEAD(createtime, 2) OVER(PARTITION BY cookieid ORDER BY createtime) AS next_2_time,
      LEAD(createtime, 1, '1970-01-01 00:00:00') OVER(PARTITION BY cookieid ORDER BY createtime) AS
next_1_time
FROM logs;
```

 版权所有 © 华为技术有限公司

```
-- 返回结果：
cookieid createtime         url  next_2_time          next_1_time
cookie1 2015-04-10 10:00:00 url1 2015-04-10 10:03:04  2015-04-10 10:00:02
cookie1 2015-04-10 10:00:02 url2 2015-04-10 10:10:00  2015-04-10 10:03:04
cookie1 2015-04-10 10:03:04 url3 2015-04-10 10:50:01  2015-04-10 10:10:00
cookie1 2015-04-10 10:10:00 url4 2015-04-10 10:50:05  2015-04-10 10:50:01
cookie1 2015-04-10 10:50:01 url5 2015-04-10 11:00:00  2015-04-10 10:50:05
cookie1 2015-04-10 10:50:05 url6 NULL                 2015-04-10 11:00:00
cookie1 2015-04-10 11:00:00 url7 NULL                 1970-01-01 00:00:00
cookie2 2015-04-10 10:00:00 url11 2015-04-10 10:03:04 2015-04-10 10:00:02
cookie2 2015-04-10 10:00:02 url22 2015-04-10 10:10:00 2015-04-10 10:03:04
cookie2 2015-04-10 10:03:04 url33 2015-04-10 10:50:01 2015-04-10 10:10:00
cookie2 2015-04-10 10:10:00 url44 2015-04-10 10:50:05 2015-04-10 10:50:01
cookie2 2015-04-10 10:50:01 url55 2015-04-10 11:00:00 2015-04-10 10:50:05
cookie2 2015-04-10 10:50:05 url66 NULL                 2015-04-10 11:00:00
cookie2 2015-04-10 11:00:00 url77 NULL                 1970-01-01 00:00:00
```

14.5.7 percent_rank

percent_rank函数为窗口的ORDER BY子句所指定列中值的返回值，但以介于0和1之间的小数形式表示，计算方法为 (分组内当前行的RANK值-1)/(分组内总行数-1)。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
percent_rank() over([partition_clause] [orderby_clause])
```

参数说明

表 14-135 参数说明

参数	是否必选	说明
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

返回DOUBLE类型的值。

示例代码

示例数据

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。创建表salary，并添加数据，命令示例如下：

 版权所有 © 华为技术有限公司

```
CREATE EXTERNAL TABLE salary (
dept STRING, -- 部门名称
userid string, -- 员工ID
sal INT -- 薪水
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
d1,user1,1000
d1,user2,2000
d1,user3,3000
d2,user4,4000
d2,user5,5000
```

示例：计算员工薪水在部门内的百分比排名。

```
select dept, userid, sal,
      percent_rank() over(partition by dept order by sal) as pr2
from salary;
-- 结果分析：
d1 user1 1000 0.0    -- (1-1)/(3-1)=0.0
d1 user2 2000 0.5    -- (2-1)/(3-1)=0.5
d1 user3 3000 1.0    -- (3-1)/(3-1)=1.0
d2 user4 4000 0.0    -- (1-1)/(2-1)=0.0
d2 user5 5000 1.0    -- (2-1)/(2-1)=1.0
```

14.5.8 rank

rank函数用于计算一个值在一组值中的排位。如果出现并列的情况，RANK函数会在排名序列中留出空位。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
rank() over ([partition_clause] [orderby_clause])
```

参数说明

表 14-136 参数说明

参数	是否必选	说明
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

返回INT类型的值。

 版权所有 © 华为技术有限公司

说明

a为NULL，则返回NULL。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表logs，并添加数据，命令示例如下：

```
CREATE TABLE logs (
cookieid string,
createtime string,
pv INT
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
cookie1 2015-04-10 1
cookie1 2015-04-11 5
cookie1 2015-04-12 7
cookie1 2015-04-13 3
cookie1 2015-04-14 2
cookie1 2015-04-15 4
cookie1 2015-04-16 4
cookie2 2015-04-10 2
cookie2 2015-04-11 3
cookie2 2015-04-12 5
cookie2 2015-04-13 6
cookie2 2015-04-14 3
cookie2 2015-04-15 9
cookie2 2015-04-16 7
```

示例：将所有记录根据cookieid分组，并按pv降序排列，返回组内每行的序号。命令示例如下:

```
select cookieid, createtime, pv,
      rank() over(partition by cookieid order by pv desc) as rank
from logs
where cookieid = 'cookie1';
-- 结果：
cookie1 2015-04-12 7 1
cookie1 2015-04-11 5 2
cookie1 2015-04-16 4 3 (并列第三)
cookie1 2015-04-15 4 3
cookie1 2015-04-13 3 5 (跳过4，从5开始)
cookie1 2015-04-14 2 6
cookie1 2015-04-10 1 7
```

14.5.9 row_number

row_number函数用于计算行号。从1开始递增。

使用限制

窗口函数的使用限制如下：

- 窗口函数只能出现在select语句中。
- 窗口函数中不能嵌套使用窗口函数和聚合函数。
- 窗口函数不能和同级别的聚合函数一起使用。

命令格式

```
row_number() over([partition_clause] [orderby_clause])
```

参数说明

表 14-137 参数说明

参数	是否必选	说明
partition_clause	否	指定分区。分区列的值相同的行被视为在同一个窗口内。
orderby_clause	否	指定数据在一个窗口内如何排序。

返回值说明

返回DOUBLE类型的值。

说明

a为NULL，则返回NULL。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表logs，并添加数据，命令示例如下：

```
CREATE TABLE logs (
cookieid string,
createtime string,
pv INT
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
cookie1 2015-04-10 1
cookie1 2015-04-11 5
cookie1 2015-04-12 7
cookie1 2015-04-13 3
cookie1 2015-04-14 2
cookie1 2015-04-15 4
cookie1 2015-04-16 4
cookie2 2015-04-10 2
cookie2 2015-04-11 3
cookie2 2015-04-12 5
cookie2 2015-04-13 6
cookie2 2015-04-14 3
cookie2 2015-04-15 9
cookie2 2015-04-16 7
```

示例：将所有记录根据cookieid分组，并按pv降序排列，返回组内每行的序号。命令示例如下:

```
select cookieid, createtime, pv,
      row_number() over (partition by cookieid order by pv desc) as index
from logs;

-- 返回结果：
cookie1 2015-04-12 7 1
cookie1 2015-04-11 5 2
cookie1 2015-04-16 4 3
cookie1 2015-04-15 4 4
cookie1 2015-04-13 3 5
cookie1 2015-04-14 2 6
```

 版权所有 © 华为技术有限公司

```
cookie1 2015-04-10 1 7
cookie2 2015-04-15 9 1
cookie2 2015-04-16 7 2
cookie2 2015-04-13 6 3
cookie2 2015-04-12 5 4
cookie2 2015-04-11 3 5
cookie2 2015-04-14 3 6
cookie2 2015-04-10 2 7
```

14.6 其他函数

14.6.1 函数概览

DLI提供了的decode1、javahash、max_pt等函数的说明如下。

表 14-138 其他新增函数说明

函数	命令格式	返回值	功能简介
decode1	decode1(<expression>, <search>, <result>[, <search>, <result>]...[, <default>])	参数的数据类型	实现if-then-else分支选择的功能。
javahash	javahash(string a)	STRING	返回hash值。
max_pt	max_pt(<table_full_name>)	STRING	返回分区表的一级分区中有数据的分区的最大值，按字母排序，且读取该分区下对应的数据。
ordinal	ordinal(bigint <nth>, <var1>, <var2>[,...])	DOUBLE或DATETIME	将输入变量按从小到大排序后，返回nth指定位置的值。
trans_array	trans_array (<num_keys>, <separator>, <key1>,<key2>, …,<col1>,<col2>, <col3>) as (<key1>,<key2>,..,<col1>, <col2>)	参数的数据类型	将一行数据转为多行的UDTF，将列中存储的以固定分隔符格式分隔的数组转为多行。
trunc_numeric	trunc_numeric(<number>[, bigint<decimal_places>])	DOUBLE或DECIMAL类型	将输入值number截取到指定小数点位置。

版权所有 © 华为技术有限公司

函数	命令格式	返回值	功能简介
url_decode	url_decode(string <input>[, string <encoding>])	STRING	将字符串从application/x-www-form-urlencoded MIME格式转为常规字符。
url_encode	url_encode(string <input>[, string <encoding>])	STRING	将字符串编码为application/x-www-form-urlencoded MIME格式。

14.6.2 decode1

decode1函数实现if-then-else分支选择的功能。

命令格式

```
decode1(<expression>, <search>, <result>[, <search>, <result>]...[, <default>])
```

参数说明

表 14-139 参数说明

参数	是否必选	参数类型	说明
expression	是	所有数据类型。	要比较的表达式。
search	是	与expression一致。	与expression进行比较的搜索项。
result	是	所有数据类型。	search和expression的值匹配时的返回值。
default	否	与result一致。	如果所有的搜索项都不匹配，则返回default值，如果未指定，则返回NULL。

返回值说明

result 和 default 为返回值，支持返回所有的数据类型。

版权所有 © 华为技术有限公司

说明

- 如果匹配，返回result。
- 如果没有匹配，返回default。
- 如果没有指定default，返回NULL。
- 如果search选项有重复且匹配时，会返回第一个值。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表salary，并添加数据，命令示例如下：

```
CREATE EXTERNAL TABLE salary (
dept_id STRING, -- 部门
userid string, -- 员工ID
sal INT
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
d1,user1,1000
d1,user2,2000
d1,user3,3000
d2,user4,4000
d2,user5,5000
```

示例

返回每个部门的名称

当 dept_id 的值为d1时，返回DLI；值为d2时，返回MRS；其他场景返回Others。

```
select dept, decode1(dept, 'd1', 'DLI', 'd2', 'MRS', 'Others') as result from sale_detail;
```

返回结果：

```
d1 DLI
d2 MRS
d3 Others
d4 Others
d5 Others
```

14.6.3 javahash

javahash函数用于返回a的hash值。

命令格式

```
javahash(string a)
```

参数说明

表 14-140 参数说明

参数	是否必选	参数类型	说明
a	是	STRING类型。	需要返回hash值的数据。

 版权所有 © 华为技术有限公司

返回值说明

返回STRING类型的值。

说明

返回hash值，如果a为null，返回报错。

示例代码

返回 48690

```
select javahash("123");
```

返回 123

```
select javahash(123);
```

14.6.4 max_pt

max_pt函数用于返回分区表的一级分区中有数据的分区的最大值，按字母排序，且读取该分区下对应的数据。

命令格式

```
max_pt(<table_full_name>)
```

参数说明

表 14-141 参数说明

参数	是否必选	参数类型	说明
table_full_name	是	STRING类型。	指定表名。必须对表有读权限。

返回值说明

返回STRING类型的值。

说明

- 返回最大的一级分区的值。
- 如果只是用alter table的方式新加了一个分区，但是此分区中并无任何数据，则此分区不会作为返回值。

示例代码

例如 table1 是分区表，该表对应的分区为20120801和20120802，且都有数据。则以下语句中max_pt返回值为‘20120802’。DLI SQL语句会读出pt=‘20120802’分区下的数据。

命令示例如下。

```
select * from tablel where pt = max_pt('dbname.table1');
```

 版权所有 © 华为技术有限公司

等效于如下语句。

```
select * from table1 where pt = (select max(pt) from dbname.table1);
```

14.6.5 ordinal

ordinal函数用于将输入变量按从小到大排序后，返回nth指定位置的值。。

命令格式

```
ordinal(bigint <nth>, <var1>, <var2>[,...])
```

参数说明

表 14-142 参数说明

参数	是否必选	参数类型	说明
nth	是	BIGINT类型。	指定要返回的位置值。
var	是	BIGINT、DOUBLE、DATETIME或STRING类型。	待排序的值。

返回值说明

DOUBLE或DECIMAL类型。

说明

- 排在第nth位的值，当不存在隐式转换时返回值同输入参数数据类型。
- 当有类型转换时，DOUBLE、BIGINT、STRING之间的转换返回DOUBLE类型；STRING、DATETIME之间的转换返回DATETIME类型。不允许其他的隐式转换。
- NULL为最小值。

示例代码

返回2。

```
select ordinal(3, 1, 3, 2, 5, 2, 4, 9);
```

14.6.6 trans_array

trans_array函数用于将一行数据转为多行的UDTF，将列中存储的以固定分隔符格式分隔的数组转为多行。

使用限制

- 所有作为key的列必须位于在前面，而要转置的列必须放在后面。

 版权所有 © 华为技术有限公司

- 在一个select中只能有一个UDTF，不可以再出现其他的列。
- 不可以与group by、cluster by、distribute by、sort by一起使用。

命令格式

```
trans_array (<num_keys>, <separator>, <key1>,<key2>,…,<col1>,<col2>,<col3>) as (<key1>,<key2>,...,<col1>,<col2>)
```

参数说明

表 14-143 参数说明

参数	是否必选	参数类型	说明
num_keys	是	BIGINT类型。	BIGINT类型常量，值必须>=0。在转为多行时作为转置key的列的个数。
separator	是	STRING类型。	STRING类型常量，用于将字符串拆分成多个元素的分隔符。为空时返回报错。
keys	是	STRING类型。	转置时作为key的列， 个数由num_keys指定。如果num_keys指定所有的列都作为key（即num_keys等于所有列的个数），则只返回一行。
cols	是	STRING类型。	要转为行的数组，keys之后的所有列视为要转置的数组，必须为STRING类型。

返回值说明

参数的数据类型。

说明

- 返回转置后的行，新的列名由as指定。
- 作为key的列类型保持不变，其余所有的列是STRING类型。
- 拆分成的行数以个数多的数组为准，不足的补NULL。

示例代码

为便于理解函数的使用方法，本文为您提供源数据，基于源数据提供函数相关示例。
创建表salary，并添加数据，命令示例如下：

```
CREATE EXTERNAL TABLE salary (
dept_id STRING, -- 部门
user_id string, -- 员工ID
sal INT -- 薪水
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
stored as textfile;
```

添加数据如下：

```
d1,user1/user4,1000/6000
d1,user2/user5,2000/7000
d1,user3/user6,3000
d2,user4/user7,4000
d2,user5/user8,5000/8000
```

 版权所有 © 华为技术有限公司

执行SQL

```
select trans_array(1, "/", dept_id, user_id, sal) as (dept_id, user_id, sal) from salary;
```

返回结果如下：

```
d1,user1,1000
d1,user4,6000
d1,user2,2000
d1,user5,7000
d1,user3,3000
d1,user6,NULL
d2,user4,4000
d2,user7,NULL
d2,user5,5000
d2,user8,8000
```

14.6.7 trunc_numeric

trunc_numeric函数用于将输入值number截取到指定小数点位置。

命令格式

```
trunc_numeric(<number>[, bigint<decimal_places>])
```

参数说明

表 14-144 参数说明

参数	是否必选	参数类型	说明
number	是	DOUBLE、BIGINT、DECIMAL、STRING类型。	需要截取的数据。
decimal_places	否	BIGINT类型。	默认为0，截取位置的小数点位。

返回值说明

返回DOUBLE或DECIMAL类型。

说明

返回规则如下：

- number为DOUBLE、DECIMAL类型时会返回相应的类型。
- number为STRING、BIGINT类型时，返回DOUBLE类型。
- decimal_places非BIGINT类型时，返回报错。
- number值为NULL时，返回NULL。

示例代码

返回 3.141。

 版权所有 © 华为技术有限公司

```
select trunc_numeric(3.1415926, 3);
```

返回 3。

```
select trunc_numeric(3.1415926);
```

报错。

```
select trunc_numeric(3.1415926, 3.1);
```

14.6.8 url_decode

url_decode函数用于将字符串从application/x-www-form-urlencoded MIME格式转为常规字符。

命令格式

```
url_decode(string <input>[, string <encoding>])
```

参数说明

表 14-145 参数说明

参数	是否必选	参数类型	说明
input	是	STRING类型。	要输入的字符串。
encoding	否	STRING类型。	指定编码格式，支持GBK或UTF-8等标准编码格式，不输入默认为UTF-8。

返回值说明

返回STRING类型的值。

说明

STRING类型UTF-8编码的字符串。

示例代码

返回 Example for URL_DECODE:// dsf(fasfs)。

```
select url_decode('Example+for+url_decode+%3A%2F%2F+dsf%28fasfs%29', 'GBK');
```

14.6.9 url_encode

url_encode函数用于将字符串编码为application/x-www-form-urlencoded MIME格式。

命令格式

url_encode(string <input>[, string <encoding>])

 版权所有 © 华为技术有限公司

参数说明

表 14-146 参数说明

参数	是否必选	参数类型	说明
input	是	STRING类型。	要输入的字符串。
encoding	否	STRING类型。	指定编码格式，支持GBK或UTF-8等标准编码格式，不输入默认为UTF-8。

返回值说明

返回STRING类型的值。

说明

input或encoding值为NULL时，返回NULL。

示例代码

返回Example+for+url_encode+%3A%2F%2F+dsf%28fasfs%29。

```
select url_encode('Example for url_encode:// dsf(fasfs)', 'GBK');
```

 版权所有 © 华为技术有限公司

15 SELECT

15.1 基本语句

功能描述

基本的查询语句，返回查询结果。

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM table_reference
  [WHERE where_condition]
  [GROUP BY col_name_list]
  [ORDER BY col_name_list][ASC | DESC]
  [CLUSTER BY col_name_list | DISTRIBUTE BY col_name_list]
  [SORT BY col_name_list]]
  [LIMIT number];
```

关键字

表 15-1 SELECT 关键字说明

参数	描述
ALL	ALL关键字用于返回数据库所有匹配的行，包括重复的行。ALL关键字的后面只能跟*，否则执行语句会出错。 ALL是SQL语句的默认行为，通常不会被明确写出，如果不指定ALL或DISTINCT，查询结果将包含所有的行，即使是重复的行数据也将被返回。
DISTINCT	在SELECT语句中使用DISTINCT关键字时，系统会在查询结果中去除重复的数据，确保结果的唯一性。
WHERE	指定查询的过滤条件，支持算术运算符、关系运算符和逻辑运算符。
where_condition	过滤条件。
GROUP BY	指定分组的字段，支持单字段及多字段分组。

 版权所有 © 华为技术有限公司

参数	描述
col_name_list	字段列表。
ORDER BY	对查询结果进行排序。
ASC/DESC	ASC为升序，DESC为降序，默认为ASC。
CLUSTER BY	为分桶且排序，按照分桶字段先进行分桶，再在每个桶中依据该字段进行排序，即当DISTRIBUTE BY的字段与SORT BY的字段相同且排序为降序时，两者的作用与CLUSTER BY等效。
DISTRIBUTE BY	指定分桶字段，不进行排序。
SORT BY	将会在桶内进行排序。
LIMIT	对查询结果进行限制，number参数仅支持INT类型。

注意事项

- 所查询的表必须是已经存在的表，否则提示查询错误。
- 在DLI管理控制台提交SQL语句读取binary类型的数据进行展示时，会对binary数据进行Base64转换。
- 在where子句中使用not in时，必须保证子查询结果集不包含任何null值。一旦存在null，整个查询结果将为空集。解决方案请参考**常见问题：not in子查询中有null值，查询结果为空怎么办?**

  使用DLI提供的SQL防御规则功能可以配置Not in<Subquery>的拦截规则，在作业运行时及时提醒您修改Not in<Subquery>语句。了解**SQL防御规则配置方法**。

示例

将表student中，name为Mike的数据记录查询出来，并根据字段score升序排序。

```
SELECT * FROM student
  WHERE name = 'Mike'
  ORDER BY score;
```

常见问题：not in 子查询中有 null 值，查询结果为空怎么办?

- **问题概述：**SQL语法中只要not in子查询中有一个null值，整个查询结果就会是空集（无数据返回）。
- **示例：**如下示例中期望的结果是3，但是反馈的是null。

  ```
  -- 表A
  id
  ---
  1
  2
  3

  -- 表B
  value
  -----
  1
  2
  ```

  执行以下SQL语句:

 版权所有 © 华为技术有限公司

```
SELECT * FROM A
WHERE id NOT IN (SELECT value FROM B);
```

- **原因分析：**

  id NOT IN (1, 2, NULL)

  等价于

  id <> 1 AND id <> 2 AND id <> NULL

  id <> NULL 的结果是 UNKNOWN（SQL 的三值逻辑），整个表达式为 FALSE

- **解决方案**：

  为了避免NULL导致空结果可以使用以下方案：

  - 方案1：用NOT EXISTS替代NOT IN

    ```
    SELECT * FROM A
    WHERE NOT EXISTS (
      SELECT 1 FROM B WHERE B.value = A.id
    );
    ```

  - 方案2：在子查询中过滤掉NULL

    ```
    SELECT * FROM A
    WHERE id NOT IN (
      SELECT value FROM B WHERE value IS NOT NULL
    );
    ```

15.2 排序

15.2.1 ORDER BY

功能描述

按字段实现查询结果的全局排序。

语法格式

```
SELECT attr_expr_list FROM table_reference
  ORDER BY col_name
  [ASC | DESC] [,col_name [ASC | DESC],...];
```

关键字

- ASC/DESC：ASC为升序，DESC为降序，默认为ASC。
- ORDER BY：对全局进行单列或多列排序。与GROUP BY一起使用时，ORDER BY后面可以跟聚合函数。

注意事项

所排序的表必须是已经存在的，否则会出错。

示例

根据字段score对表student进行升序排序，并返回排序后的结果。

```
SELECT * FROM student
  ORDER BY score;
```

 版权所有 © 华为技术有限公司

15.2.2 SORT BY

功能描述

按字段实现表的局部排序。

语法格式

```
SELECT attr_expr_list FROM table_reference
  SORT BY col_name
  [ASC | DESC] [,col_name [ASC | DESC],...];
```

关键字

- ASC/DESC：ASC为升序，DESC为降序，默认为ASC。
- SORT BY：一般与GROUP BY一起使用，为PARTITION进行单列或多列的局部排序。

注意事项

所排序的表必须是已经存在的，否则会出错。

示例

根据字段score对表student在Reducer中进行升序排序。

```
SELECT * FROM student
  SORT BY score;
```

15.2.3 CLUSTER BY

功能描述

按字段实现表的分桶及桶内排序。

语法格式

```
SELECT attr_expr_list FROM table_reference
  CLUSTER BY col_name [,col_name ,...];
```

关键字

CLUSTER BY：根据指定的字段进行分桶，支持单字段及多字段，并在桶内进行排序。

注意事项

所排序的表必须是已经存在的，否则会出错。

示例

根据字段score对表student进行分桶并进行桶内局部降序排序。

```
SELECT * FROM student
  CLUSTER BY score;
```

 版权所有 © 华为技术有限公司

15.2.4 DISTRIBUTE BY

功能描述

按字段实现表的分桶。

语法格式

```
SELECT attr_expr_list FROM table_reference
  DISTRIBUTE BY col_name [,col_name ,...];
```

关键字

DISTRIBUTE BY：根据指定的字段进行分桶，支持单字段及多字段，不会在桶内进行排序。与SORT BY配合使用即为分桶后的排序。

注意事项

所排序的表必须是已经存在的，否则会出错。

举例

根据字段score对表student进行分桶。

```
SELECT * FROM student
  DISTRIBUTE BY score;
```

15.3 分组

15.3.1 按列 GROUP BY

功能描述

按列对表进行分组操作。

语法格式

```
SELECT attr_expr_list FROM table_reference
  GROUP BY col_name_list;
```

关键字

GROUP BY：按列可分为单列GROUP BY与多列GROUP BY。

- 单列GROUP BY：指GROUP BY子句中仅包含一列，col_name_list中包含的字段必须出现在attr_expr_list的字段内，attr_expr_list中可以使用多个聚合函数，比如count()，sum()，聚合函数中可以包含其他字段。
- 多列GROUP BY：指GROUP BY子句中不止一列，查询语句将按照GROUP BY的所有字段分组，所有字段都相同的记录将被放在同一组中，同样，GROUP BY中出现的字段必须在attr_expr_list的字段内，attr_expr_list也可以使用聚合函数。

 版权所有 © 华为技术有限公司

注意事项

所要分组的表必须是已经存在的表，否则会出错。

示例

根据score及name两个字段对表student进行分组，并返回分组结果。

```
SELECT score, count(name) FROM student
  GROUP BY score,name;
```

15.3.2 按表达式 GROUP BY

功能描述

按表达式对表进行分组操作。

语法格式

```
SELECT attr_expr_list FROM table_reference
  GROUP BY groupby_expression [, groupby_expression, ...];
```

关键字

groupby_expression：可以是单字段，多字段，也可以是聚合函数，字符串函数等。

注意事项

- 所要分组的表必须是已经存在的表，否则会出错。
- 同单列分组，GROUP BY中出现的字段必须包含在attr_expr_list的字段中，表达式支持内置函数，自定义函数等。

示例

先利用substr函数取字段name的子字符串，并按照该子字符串进行分组，返回每个子字符串及对应的记录数。

```
SELECT substr(name,6),count(name) FROM student
  GROUP BY substr(name,6);
```

15.3.3 GROUP BY 中使用 HAVING

功能描述

利用HAVING子句在表分组后实现过滤。

语法格式

```
SELECT attr_expr_list FROM table_reference
  GROUP BY groupby_expression[, groupby_expression… ]
  HAVING having_expression;
```

关键字

groupby_expression：可以是单字段，多字段，也可以是聚合函数，字符串函数等。

 版权所有 © 华为技术有限公司

注意事项

- 所要分组的表必须是已经存在的表，否则会出错。
- 如果过滤条件受GROUP BY的查询结果影响，则不能用WHERE子句进行过滤，而要用HAVING子句进行过滤。HAVING与GROUP BY合用，先通过GROUP BY进行分组，再在HAVING子句中进行过滤，HAVING子句中可支持算术运算，聚合函数等。

示例

先依据num对表transactions进行分组，再利用HAVING子句对查询结果进行过滤，price与amount乘积的最大值大于5000的记录将被筛选出来，返回对应的num及price与amount乘积的最大值。

```
SELECT num, max(price*amount) FROM transactions
  WHERE time > '2016-06-01'
  GROUP BY num
  HAVING max(price*amount)>5000;
```

15.3.4 ROLLUP

功能描述

ROLLUP生成聚合行、超聚合行和总计行。可以实现从右到左递减多级的统计，显示统计某一层次结构的聚合。

语法格式

```
SELECT attr_expr_list FROM table_reference
  GROUP BY col_name_list
  WITH ROLLUP;
```

关键字

ROLLUP：为GROUP BY的扩展，例如：***SELECT a, b, c, SUM(expression) FROM table GROUP BY a, b, c WITH ROLLUP;***将转换成以下四条查询：

- (a, b, c)组合小计
  ```
  SELECT a, b, c, sum(expression) FROM table
    GROUP BY a, b, c;
  ```
- (a, b)组合小计
  ```
  SELECT a, b, NULL, sum(expression) FROM table
    GROUP BY a, b;
  ```
- (a)组合小计
  ```
  SELECT a, NULL, NULL, sum(expression) FROM table
    GROUP BY a;
  ```
- 总计
  ```
  SELECT NULL, NULL, NULL, sum(expression) FROM table;
  ```

注意事项

所要分组的表必须是已经存在的表，否则会出错。

示例

根据group_id与job两个字段生成聚合行、超聚合行和总计行，返回每种聚合情况下的salary总和。

 版权所有 © 华为技术有限公司

```
SELECT group_id, job, SUM(salary) FROM group_test
  GROUP BY group_id, job
  WITH ROLLUP;
```

15.3.5 GROUPING SETS

功能描述

GROUPING SETS生成交叉表格行，可以实现GROUP BY字段的交叉统计。

语法格式

```
SELECT attr_expr_list FROM table_reference
  GROUP BY col_name_list
  GROUPING SETS(col_name_list);
```

关键字

GROUPING SETS：为对GROUP BY的扩展，例如

- ***SELECT a, b, sum(expression) FROM table GROUP BY a, b GROUPING SETS((a,b));***

  将转换为以下一条查询：

  ```
  SELECT a, b, sum(expression) FROM table
    GROUP BY a, b;
  ```

- ***SELECT a, b, sum(expression) FROM table GROUP BY a, b GROUPING SETS(a,b);***

  将转换为以下两条查询：

  ```
  SELECT a, NULL, sum(expression) FROM table GROUP BY a;
  UNION
  SELECT NULL, b, sum(expression) FROM table GROUP BY b;
  ```

- ***SELECT a, b, sum(expression) FROM table GROUP BY a, b GROUPING SETS((a,b), a);***

  将转换为以下两条查询：

  ```
  SELECT a, b, sum(expression) FROM table GROUP BY a, b;
  UNION
  SELECT a, NULL, sum(expression) FROM table GROUP BY a;
  ```

- ***SELECT a, b, sum(expression) FROM table GROUP BY a, b GROUPING SETS((a,b), a, b, ());***

  将转换为以下四条查询：

  ```
  SELECT a, b, sum(expression) FROM table GROUP BY a, b;
  UNION
  SELECT a, NULL, sum(expression) FROM table GROUP BY a, NULL;
  UNION
  SELECT NULL, b, sum(expression) FROM table GROUP BY NULL, b;
  UNION
  SELECT NULL, NULL, sum(expression) FROM table;
  ```

注意事项

- 所要分组的表必须是已经存在的表，否则会出错。
- 不同于ROLLUP，GROUPING SETS目前仅支持一种格式。

示例

根据group_id与job两个字段生成交叉表格行，返回每种聚合情况下的salary总和。

 版权所有 © 华为技术有限公司

```
SELECT group_id, job, SUM(salary) FROM group_test
  GROUP BY group_id, job
  GROUPING SETS (group_id, job);
```

15.4 连接

15.4.1 内连接

功能描述

仅将两个表中满足连接条件的行组合起来作为结果集。

语法格式

```
SELECT attr_expr_list FROM table_reference
  {JOIN | INNER JOIN} table_reference ON join_condition;
```

关键字

JOIN/INNER JOIN：只显示参与连接的表中满足JOIN条件的记录。

注意事项

- 所要进行JOIN连接的表必须是已经存在的表，否则会出错。
- 在一次查询中可以连接两个以上的表。

示例

通过将student_info与course_info两张表中的课程编号匹配建立JOIN连接，来查看学生姓名及所选课程名称。

```
SELECT student_info.name, course_info.courseName FROM student_info
  JOIN course_info ON (student_info.courseId = course_info.courseId);
```

15.4.2 左外连接

功能描述

根据左表的记录去匹配右表，返回所有左表记录，没有匹配值的记录的返回NULL。

语法格式

```
SELECT attr_expr_list FROM table_reference
  LEFT OUTER JOIN table_reference ON join_condition;
```

关键字

LEFT OUTER JOIN：返回左表的所有记录，没有匹配值的记录将返回NULL。

注意事项

所要进行JOIN连接的表必须是已经存在的表，否则会出错。

 版权所有 © 华为技术有限公司

示例

左外连接时利用student_info表中的courseId与course_info中的courseId进行匹配，返回已经选课的学生姓名及所选的课程名称，没有匹配值的右表记录将返回NULL。

```
SELECT student_info.name, course_info.courseName FROM student_info
  LEFT OUTER JOIN course_info ON (student_info.courseId = course_info.courseId);
```

15.4.3 右外连接

功能描述

根据右表的记录去匹配左表，返回所有右表记录，没有匹配值的记录返回NULL。

语法格式

```
SELECT attr_expr_list FROM table_reference
  RIGHT OUTER JOIN table_reference ON join_condition;
```

关键字

RIGHT OUTER JOIN：返回右表的所有记录，没有匹配值的记录将返回NULL。

注意事项

所要进行JOIN连接的表必须是已经存在的表，否则会出错。

示例

右外连接和左外连接相似，但是会将右边表（这里的course_info)中的所有记录返回，没有匹配值的左表记录将返回NULL。

```
SELECT student_info.name, course_info.courseName FROM student_info
  RIGHT OUTER JOIN course_info ON (student_info.courseId = course_info.courseId);
```

15.4.4 全外连接

功能描述

根据左表与右表的所有记录进行匹配，没有匹配值的记录返回NULL。

语法格式

```
SELECT attr_expr_list FROM table_reference
  FULL OUTER JOIN table_reference ON join_condition;
```

关键字

FULL OUTER JOIN：根据左表与右表的所有记录进行匹配，没有匹配值的记录返回NULL。

注意事项

所要进行JOIN连接的表必须是已经存在的表，否则会出错。

 版权所有 © 华为技术有限公司

示例

利用全外连接可以将两张表中的所有记录返回，没有匹配值的左表及右表记录将返回NULL。

```
SELECT student_info.name, course_info.courseName FROM student_info
  FULL OUTER JOIN course_info ON (student_info.courseId = course_info.courseId);
```

15.4.5 隐式连接

功能描述

与内连接功能相同，返回两表中满足WHERE条件的结果集，但不用JOIN显示指定连接条件。

语法格式

```
SELECT table_reference.col_name, table_reference.col_name, ... FROM table_reference, table_reference
  WHERE table_reference.col_name = table_reference.col_name;
```

关键字

WHERE：隐式连接利用WHERE条件实现类似JOIN...ON...的连接，返回匹配的记录。**语法格式**中仅给出等式条件下的WHERE条件过滤，同时也支持不等式WHERE条件过滤。

注意事项

- 所要进行JOIN连接的表必须是已经存在的表，否则会出错。
- 隐式JOIN的命令中不含有JOIN...ON...关键词，而是通过WHERE子句作为连接条件将两张表连接。

示例

返回courseId匹配的学生姓名及课程名称。

```
SELECT student_info.name, course_info.courseName FROM student_info,course_info
  WHERE student_info.courseId = course_info.courseId;
```

15.4.6 笛卡尔连接

功能描述

笛卡尔连接把第一个表的每一条记录和第二个表的所有记录相连接，如果第一个表的记录数为m, 第二个表的记录数为n,则会产生m*n条记录数。

语法格式

```
SELECT attr_expr_list FROM table_reference
  CROSS JOIN table_reference ON join_condition;
```

关键字

join_condition：连接条件，如果该条件恒成立（比如1=1），该连接就是笛卡尔连接。所以，笛卡尔连接输出的记录条数等于被连接表的各记录条数的乘积，若需要进行笛

 版权所有 © 华为技术有限公司

卡尔积连接，需使用专门的关键词CROSS JOIN。CROSS JOIN是求笛卡尔积的标准方式。

注意事项

所要进行JOIN连接的表必须是已经存在的表，否则会出错。

示例

返回student_info与course_info两张表中学生姓名与课程名称的所有组合。

```
SELECT student_info.name, course_info.courseName FROM student_info
  CROSS JOIN course_info ON (1 = 1);
```

15.4.7 左半连接

功能描述

左半连接用来查看左表中符合JOIN条件的记录。

语法格式

```
SELECT attr_expr_list FROM table_reference
  LEFT SEMI JOIN table_reference ON join_condition;
```

关键字

LEFT SEMI JOIN：只显示左表中的记录。可通过在LEFT SEMI JOIN， WHERE...IN和WHERE EXISTS中嵌套子查询来实现。左半连接与左外连接的区别是，左半连接将返回左表中符合JOIN条件的记录，而左外连接将返回左表所有的记录，匹配不上JOIN条件的记录将返回NULL值。

注意事项

- 所要进行JOIN连接的表必须是已经存在的表，否则会出错。
- 此处的attr_expr_list中所涉及的字段只能是左表中的字段，否则会出错。

示例

返回选课学生的姓名及其所选的课程编号。

```
SELECT student_info.name, student_info.courseId FROM student_info
  LEFT SEMI JOIN course_info ON (student_info.courseId = course_info.courseId);
```

15.4.8 不等值连接

功能描述

不等值连接中，多张表通过不相等的连接值进行连接，并返回满足条件的结果集。

语法格式

```
SELECT attr_expr_list FROM table_reference
  JOIN table reference ON non_equi_join_condition;
```

 版权所有 © 华为技术有限公司

关键字

non_equi_join_condition：与join_condition类似，只是join条件均为不等式条件。

注意事项

所要进行JOIN连接的表必须是已经存在的表，否则会出错。

示例

返回student_info_1与student_info_2两张表中的所有学生姓名对组合，但不包含相同姓名的姓名对。

```
SELECT student_info_1.name, student_info_2.name FROM student_info_1
  JOIN student_info_2 ON (student_info_1. name <> student_info_2. name);
```

15.5 子句

15.5.1 FROM

功能描述

在FROM子句中嵌套子查询，子查询的结果作为中间过渡表，进而作为外部SELECT语句的数据源。

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM (sub_query) [alias];
```

关键字

- ALL：返回重复的行。为默认选项。其后只能跟*，否则会出错。
- DISTINCT：从结果集移除重复的行。

注意事项

- 所要查询的表必须是已经存在的表，否则会出错。
- FROM嵌套子查询中，子查询必须要取别名，且别名的命名要早于别名的使用，否则会出错。建议别名不要重名。
- FROM后所跟的子查询结果必须带上前面所取的别名，否则会出错。

示例

返回选了course_info表中课程的学生姓名，并利用DISTINCT关键字进行去重。

```
SELECT DISTINCT name FROM (SELECT name FROM student_info
  JOIN course_info ON student_info.courseId = course_info.courseId) temp;
```

 版权所有 © 华为技术有限公司

15.5.2 OVER

功能描述

窗口函数与OVER语句一起使用。OVER语句用于对数据进行分组，并对组内元素进行排序。窗口函数用于给组内的值生成序号。

语法格式

```
SELECT window_func(args) OVER
  ([PARTITION BY col_name, col_name, ...]
   [ORDER BY col_name, col_name, ...]
   [ROWS | RANGE BETWEEN (CURRENT ROW | (UNBOUNDED |[num]) PRECEDING)
  AND (CURRENT ROW | ( UNBOUNDED | [num]) FOLLOWING)]);
```

关键字

- PARTITION BY：可以用一个或多个键分区。和GROUP BY子句类似，PARTITION BY将表按分区键分区，每个分区是一个窗口，窗口函数作用于各个分区。单表分区数最多允许7000个。
- ORDER BY：决定窗口函数求值的顺序。可以用一个或多个键排序。通过ASC或DESC决定升序或降序。窗口由WINDOW子句指定。如果不指定，默认窗口等同于ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW，即窗口从表或分区（如果OVER子句中用PARTITION BY分区）的初始处到当前行。
- WINDOW：通过指定一个行区间来定义窗口。
- CURRENT ROW：表示当前行。
- num PRECEDING：定义窗口的下限，即窗口从当前行向前数num行处开始。
- UNBOUNDED PRECEDING：表示窗口没有下限。
- num FOLLOWING：定义窗口的上限，即窗口从当前行向后数num行处结束。
- UNBOUNDED FOLLOWING：表示窗口没有上限。
- ROWS BETWEEN…和RANGE BETWEEN…的区别：
  - ROW为物理窗口，即根据ORDER BY子句排序后，取前N行及后N行的数据计算（与当前行的值无关，只与排序后的行号相关）。
  - RANGE为逻辑窗口，即指定当前行对应值的范围取值，列数不固定，只要行值在范围内，对应列都包含在内。
- 窗口有以下多种场景，如
  - 窗口只包含当前行。
    ```
    ROWS BETWEEN CURRENT ROW AND CURRENT ROW
    ```
  - 窗口从当前行向前数3行开始，到当前行向后数5行结束。
    ```
    ROWS BETWEEN 3 PRECEDING AND 5 FOLLOWING
    ```
  - 窗口从表或分区的开头开始，到当前行结束。
    ```
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ```
  - 窗口从当前行开始，到表或分区的结尾结束。
    ```
    ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
    ```
  - 窗口从表或分区的开头开始，到表或分区的结尾结束。
    ```
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    ```

 版权所有 © 华为技术有限公司

注意事项

OVER子句包括：PARTITION BY子句、ORDER BY子句和WINDOW子句，可组合使用。OVER子句为空表示窗口为整张表。

示例

上述语句窗口从表或分区的开头开始，到当前行结束，对over_test表按照id字段进行排序，并返回排序好后的id及id所对应的序号。

```
SELECT id, count(id) OVER (ORDER BY id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM over_test;
```

15.5.3 WHERE

功能描述

在WHERE子句中嵌套子查询，利用子查询的结果作为过滤条件。

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM table_reference
  WHERE {col_name operator (sub_query) | [NOT] EXISTS sub_query};
```

关键字

- ALL：返回重复的行。为默认选项。其后只能跟*，否则会出错。
- DISTINCT：从结果集移除重复的行。
- WHERE：WHERE子句嵌套将利用子查询的结果作为过滤条件。
- operator：包含关系运算符中的等式与不等式操作符及IN，NOT IN，EXISTS，NOT EXISTS操作符。
  - 当operator为IN或者NOT IN时，子查询的返回结果必须是单列。
  - 当operator为EXISTS或者NOT EXISTS时，子查询中一定要包含WHERE条件过滤。当子查询中有字段与外部查询相同时，需要在该字段前加上表名。

注意事项

所要查询的表必须是已经存在的表，否则会出错。

示例

先通过子查询在course_info中找到Biology所对应的课程编号，再在student_info表中找到选了该课程编号的学生姓名。

```
SELECT name FROM student_info
  WHERE courseId = (SELECT courseId FROM course_info WHERE courseName = 'Biology');
```

15.5.4 HAVING

功能描述

在HAVING子句中嵌套子查询，子查询结果将作为HAVING子句的一部分。

 版权所有 © 华为技术有限公司

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM table_reference
  GROUP BY groupby_expression
  HAVING aggregate_func(col_name) operator (sub_query);
```

关键字

- ALL：返回重复的行。为默认选项。其后只能跟*，否则会出错。
- DISTINCT：从结果集移除重复的行。
- groupby_expression：可以是单字段，多字段，也可以是聚合函数，字符串函数等。
- operator：此操作符包含等式操作符与不等式操作符，及IN，NOT IN操作符。

注意事项

- 所要查询的表必须是已经存在的表，否则会出错。
- 此处的sub_query与聚合函数的位置不能左右互换。

示例

对表student_info按字段name进行分组，计算每组中记录数，若其记录数等于子查询中表course_info的记录数，返回表student_info中字段name等于表course_info字段name的记录数。

```
SELECT name FROM student_info
  GROUP BY name
  HAVING count(name) = (SELECT count(*) FROM course_info);
```

15.5.5 多层嵌套子查询

功能描述

多层嵌套子查询，即在子查询中嵌套子查询。

语法格式

```
SELECT attr_expr FROM ( SELECT attr_expr FROM ( SELECT attr_expr FROM... ... ) [alias] ) [alias];
```

关键字

- ALL：返回重复的行。为默认选项。其后只能跟*，否则会出错。
- DISTINCT：从结果集移除重复的行。

注意事项

- 所要查询的表必须是已经存在的表，否则会出错。
- 在嵌套查询中必须指定子查询的别名，否则会出错。
- 别名的命名必须在别名的使用之前，否则会出错，建议别名不要重名。

示例

通过三次子查询，最终返回user_info中的name字段。

 版权所有 © 华为技术有限公司

```
SELECT name FROM ( SELECT name, acc_num FROM ( SELECT name, acc_num, password FROM ( SELECT name, acc_num, password, bank_acc FROM user_info) a ) b ) c;
```

15.6 别名 SELECT

15.6.1 表别名

功能描述

给表或者子查询结果起别名。

语法格式

```
SELECT attr_expr_list FROM table_reference [AS] alias;
```

关键字

- table_reference：可以是表，视图或者子查询。
- AS：可用于连接table_reference和alias，是否添加此关键字不会影响命令执行结果。

注意事项

- 所要查询的表必须是已经存在的，否则会出错。
- 别名的命名必须在别名的使用之前，否则会出错。此外，建议不要重名。

示例

- 给表simple_table起为n的别名，并利用n.name访问simple_table中的name字段。
  ```
  SELECT n.score FROM simple_table n WHERE n.name = "leilei";
  ```
- 将子查询的结果命令为m，并利用SELECT * FROM m返回子查询中的所有结果。
  ```
  SELECT * FROM (SELECT * FROM simple_table WHERE score > 90) AS m;
  ```

15.6.2 列别名

功能描述

给列起别名。

语法格式

```
SELECT attr_expr [AS] alias, attr_expr [AS] alias, ... FROM table_reference;
```

关键字

- alias：用于对attr_expr中的字段名称起别名。
- AS：是否添加此关键字不会影响结果。

注意事项

- 所要查询的表必须是已经存在的，否则会出错。

 版权所有 © 华为技术有限公司

- 别名的命名必须在别名的使用之前，否则会出错。此外，建议不要重名。

示例

先通过子查询SELECT name AS n FROM simple_table WHERE score > 90获得结果，在子查询中给name起的别名n可直接用于外部SELECT语句。

```
SELECT n FROM (SELECT name AS n FROM simple_table WHERE score > 90) m WHERE n = "xiaoming";
```

15.7 集合运算 SELECT

15.7.1 UNION

功能描述

UNION返回多个查询结果的并集。

语法格式

```
select_statement UNION [ALL] select_statement;
```

关键字

UNION：集合运算，以一定条件将表首尾相接，其中每一个SELECT语句返回的列数必须相同，列的类型和列名不一定要相同。

注意事项

- UNION默认是去重的，UNION ALL是不去重的。
- 不能在多个集合运算间（UNION，INTERSECT，EXCEPT）加括号，否则会出错。

示例

返回“SELECT * FROM student _1”查询结果与“SELECT * FROM student _2”查询结果的并集，不包含重复记录。

```
SELECT * FROM student_1 UNION SELECT * FROM student_2;
```

15.7.2 INTERSECT

功能描述

INTERSECT返回多个查询结果的交集。

语法格式

```
select_statement INTERSECT select_statement;
```

关键字

INTERSECT：返回多个查询结果的交集，且每一个SELECT语句返回的列数必须相同，列的类型和列名不一定要相同。INTERSECT默认去重。

 版权所有 © 华为技术有限公司

注意事项

不能在多个集合运算间（UNION，INTERSECT，EXCEPT）加括号，否则会出错

示例

返回“SELECT * FROM student _1”查询结果与“SELECT * FROM student _2”查询结果的交集，不包含重复记录。

```
SELECT * FROM student _1 INTERSECT SELECT * FROM student _2;
```

15.7.3 EXCEPT

功能描述

返回两个查询结果的差集。

语法格式

```
select_statement EXCEPT select_statement;
```

关键字

EXCEPT：做集合减法。A EXCEPT B将A中所有和B重合的记录扣除，然后返回去重后的A中剩下的记录，EXCEPT默认不去重。与UNION相同，每一个SELECT语句返回的列数必须相同，列的类型和列名不一定要相同。

注意事项

不能在多个集合运算间（UNION，INTERSECT，EXCEPT）加括号，否则会出错

示例

先将“SELECT * FROM student_1”查询结果减去“SELECT * FROM student_2”结果中的重合部分，然后返回剩下来的记录。

```
SELECT * FROM student_1 EXCEPT SELECT * FROM student_2;
```

15.8 WITH...AS

功能描述

通过用WITH...AS定义公共表达式（CTE）来简化查询，提高可阅读性和易维护性。

语法格式

```
WITH cte_name AS (select_statement) sql_containing_cte_name;
```

关键字

- cte_name：公共表达式的名字，不允许重名。
- select_statement：完整的SELECT语句。
- sql_containing_cte_name：包含了刚刚定义的公共表达式的SQL语句

 版权所有 © 华为技术有限公司

注意事项

- 定义了一个CTE后必须马上使用，否则这个CTE定义将失效。
- 可以通过一次WITH定义多个CTE，中间用逗号连接，后定义的CTE可以引用已经定义的CTE。

示例

将“SELECT courseId FROM course_info WHERE courseName = 'Biology'”定义为公共表达式nv，然后在后续的查询中直接利用nv代替该SELECT语句。

```
WITH nv AS (SELECT courseId FROM course_info WHERE courseName = 'Biology') SELECT DISTINCT courseId FROM nv;
```

15.9 CASE...WHEN

15.9.1 简单 CASE 函数

功能描述

依据input_expression与when_expression的匹配结果跳转到相应的result_expression。

语法格式

```
CASE input_expression WHEN when_expression THEN result_expression [...n] [ELSE else_result_expression] END;
```

关键字

CASE：简单CASE函数中支持子查询，但须注意input_expression与when_expression是可匹配的。

注意事项

如果没有取值为TRUE的input_expression = when_expression，则当指定ELSE子句时，DLI将返回else_result_expression；当没有指定ELSE子句时，返回NULL值。

示例

返回表student中的字段name及与id相匹配的字符。匹配规则如下：

- id为1则返回'a'；
- id为2则返回'b'；
- id为3则返回'c'；
- 否则返回NULL。

```
SELECT name, CASE id WHEN 1 THEN 'a' WHEN 2 THEN 'b' WHEN 3 THEN 'c' ELSE NULL END FROM student;
```

 版权所有 © 华为技术有限公司

15.9.2 CASE 搜索函数

功能描述

按指定顺序为每个WHEN子句的boolean_expression求值。返回第一个取值为TRUE的boolean_expression的result_expression。

语法格式

```
CASE WHEN boolean_expression THEN result_expression [...n] [ELSE else_result_expression] END;
```

关键字

boolean_expression：可以包含子查询，但整个boolean_expression表达式返回值只能是布尔类型。

注意事项

如果没有取值为TRUE的Boolean_expression，则当指定ELSE子句时，DLI将返回else_result_expression；当没有指定ELSE子句时，返回NULL值。

示例

对表student进行查询，返回字段name及与score对应的结果，score大于等于90返回EXCELLENT，score在（80,90）之间的返回GOOD，否则返回BAD。

```
SELECT name, CASE WHEN score >= 90 THEN 'EXCELLENT' WHEN 80 < score AND score < 90 THEN 'GOOD' ELSE 'BAD' END AS level FROM student;
```

 版权所有 © 华为技术有限公司

16 标示符

16.1 aggregate_func

格式

无。

说明

聚合函数。

aggregate_func通常在数据库查询中用于对一组值进行计算并返回单个结果。

16.2 alias

格式

无。

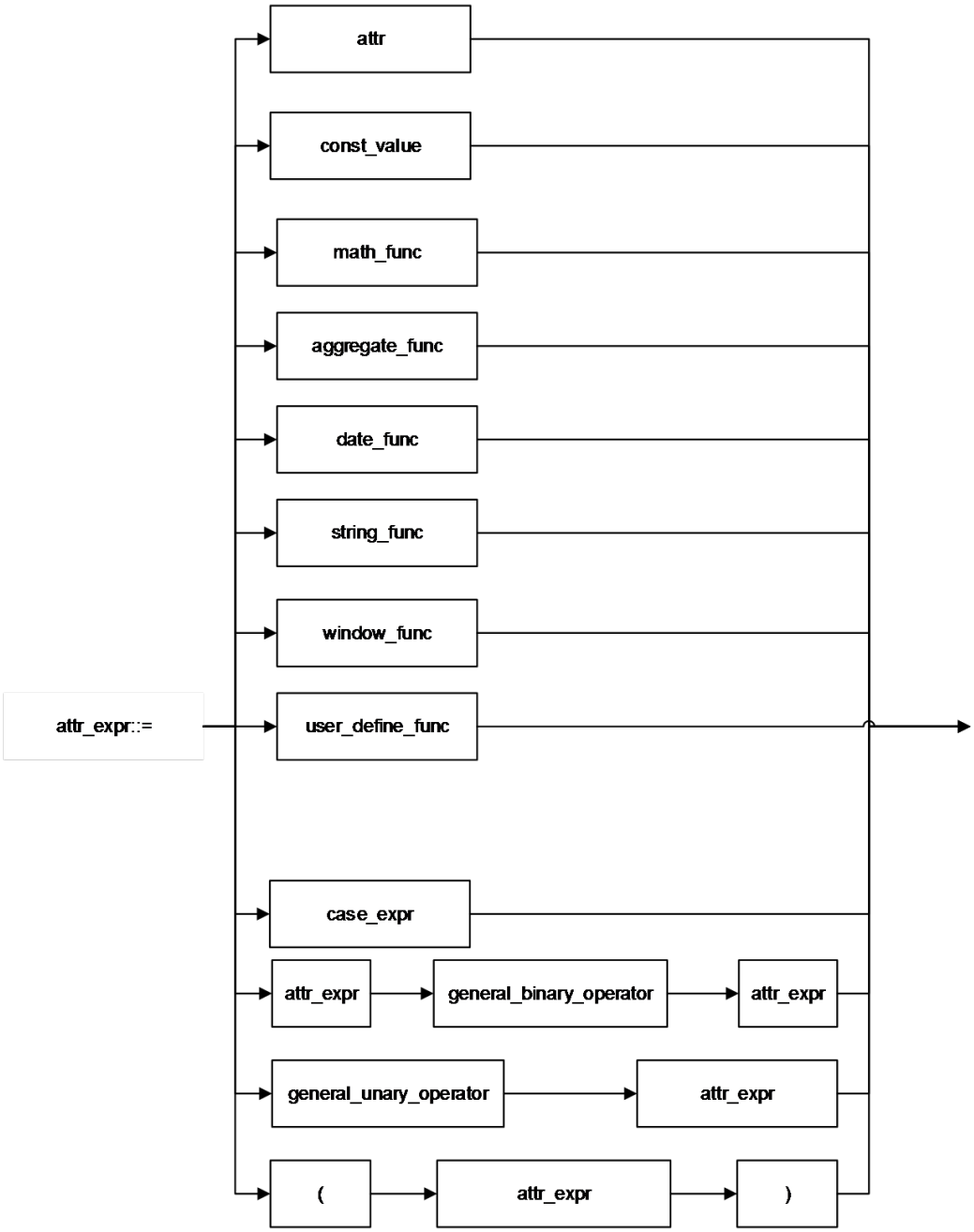
说明

别名，可给字段、表、视图、子查询起别名，仅支持字符串类型。

版权所有 © 华为技术有限公司

16.3 attr_expr

格式



说明

语法	描述
attr_expr	属性表达式。

版权所有 © 华为技术有限公司

语法	描述
attr	表的字段，与col_name相同。
const_value	常量值。
case_expr	case表达式。
math_func	数学函数。
date_func	日期函数。
string_func	字符串函数。
aggregate_func	聚合函数。
window_func	分析窗口函数。
user_define_func	用户自定义函数。
general_binary_operator	普通二元操作符。
general_unary_operator	普通一元操作符。
(	指定子属性表达式开始。
)	指定子属性表达式结束。

16.4 attr_expr_list

格式

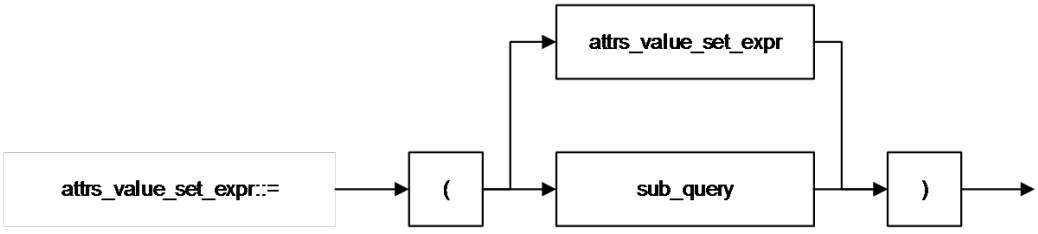
无。

说明

attr_expr列表，以逗号分隔。

16.5 attrs_value_set_expr

格式



 版权所有 © 华为技术有限公司

说明

语法	描述
attrs_value_set_expr	属性值集合。
sub_query	子查询语句。
(	指定子查询表达式开始。
)	指定子查询表达式结束。

16.6 boolean_expression

格式

无。

说明

返回boolean类型的表达式。

16.7 class_name

格式

无。

说明

函数所依赖的类名，注意类名需要包含类所在包的完整路径。

16.8 col

格式

无。

说明

函数调用时的形参，一般即为字段名称，与col_name相同。

16.9 col_comment

格式

无。

 版权所有 © 华为技术有限公司

说明

对列（字段）的描述，仅支持字符串类型，描述长度不能超过256字节。

16.10 col_name

格式

无。

说明

列名，即字段名称，仅支持字符串类型，名称长度不能超过128个字节。

16.11 col_name_list

格式

无。

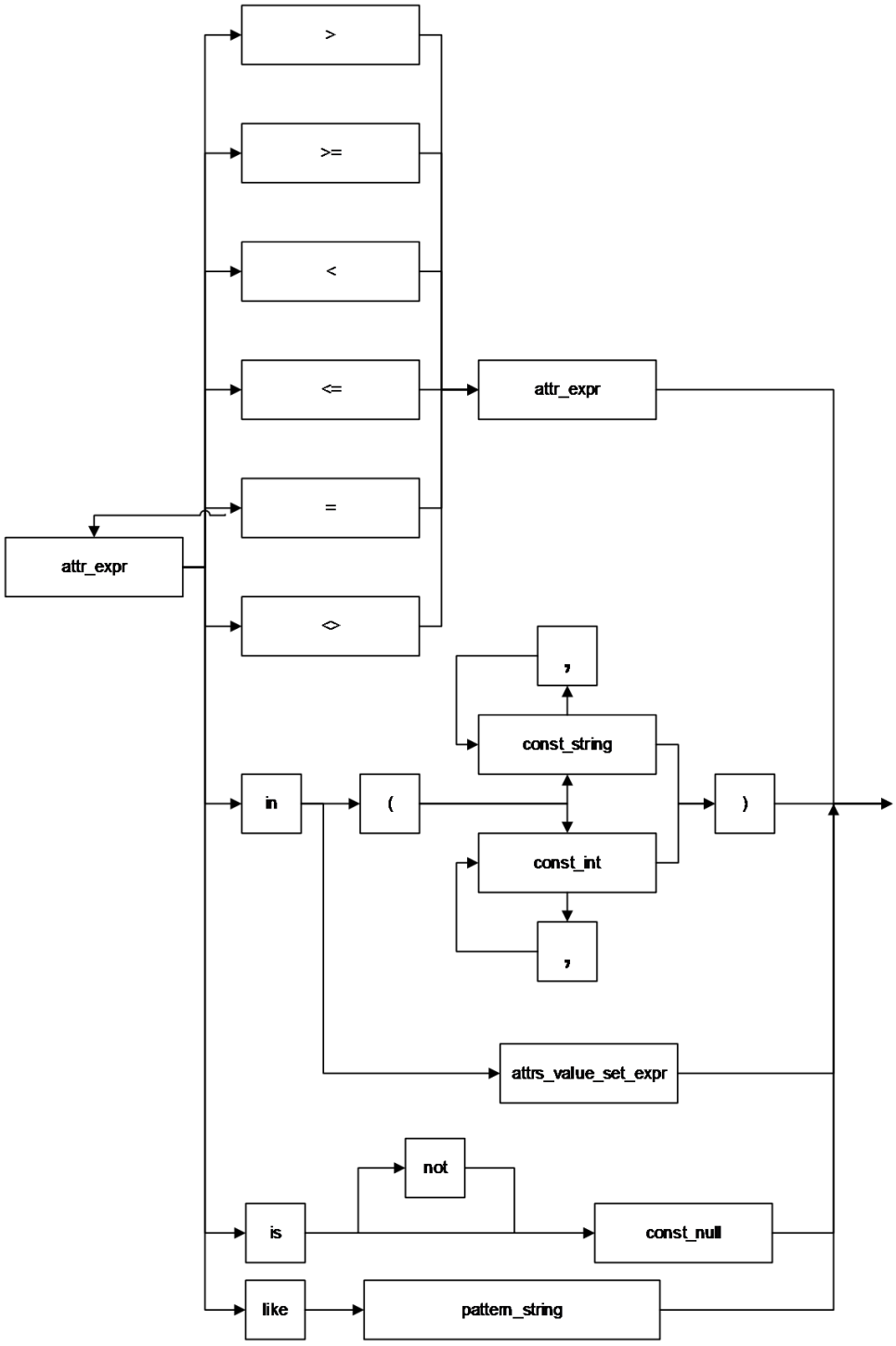
说明

字段列表，可由一个或多个col_name构成，多个col_name之间用逗号分隔。

 版权所有 © 华为技术有限公司

16.12 condition

格式



版权所有 © 华为技术有限公司

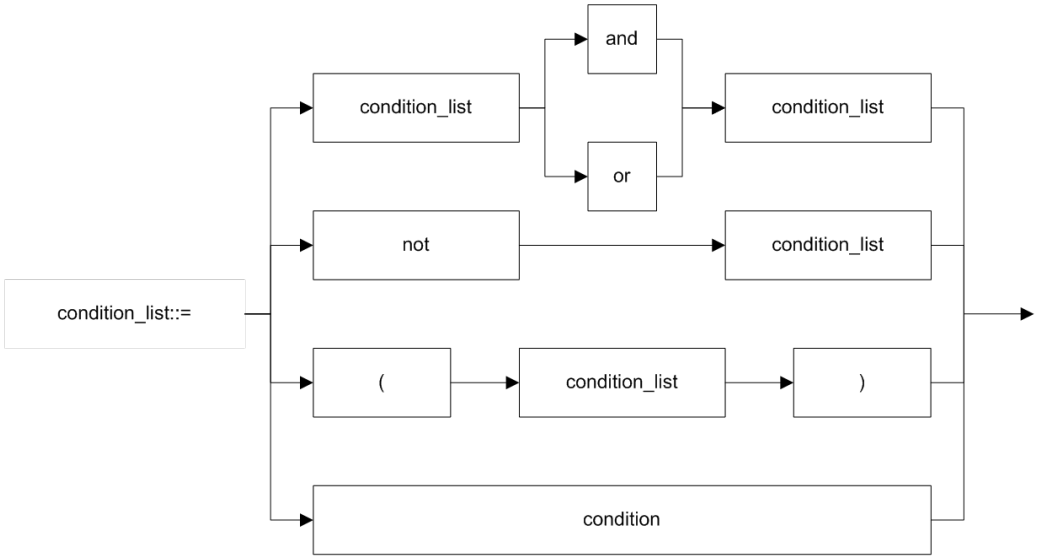
说明

语法	描述
condition	逻辑判断条件。
>	关系运算符：大于。
>=	关系运算符：大于等于。
<	关系运算符：小于。
<=	关系运算符：小于等于。
=	关系运算符：等于。
<>	关系运算符：不等于。
is	关系运算符：是。
is not	关系运算符：不是。
const_null	常量：空值。
like	关系运算符：用于通配符匹配。
pattern_string	模式匹配字符串，支持通配符匹配。WHERE LIKE条件过滤时，支持SQL通配符中“%”与“_”，“%”代表一个或多个字符，“_”仅代表一个字符。
attr_expr	属性表达式。
attrs_value_set_expr	属性值集合。
in	关键字，用于判断属性是否在一个集合中。
const_string	字符串常量。
const_int	整型常量。
(	指定常量集合开始。
)	指定常量集合结束。
,	逗号分隔符。

 版权所有 © 华为技术有限公司

16.13 condition_list

格式



说明

语法	描述
condition_list	逻辑判断条件列表。
and	逻辑运算符：与。
or	逻辑运算符：或。
not	逻辑运算符：非。
(	子逻辑判断条件开始。
)	子逻辑判断条件结束。
condition	逻辑判断条件。

16.14 cte_name

格式

无。

说明

公共表达式的名字。

版权所有 © 华为技术有限公司

16.15 data_type

格式

无。

说明

数据类型，当前只支持原生数据类型。

16.16 db_comment

格式

无。

说明

对数据库的描述，仅支持字符串类型，描述长度不能超过256字节。

16.17 db_name

格式

无。

说明

数据库名称，仅支持字符串类型，名称长度不能超过128字节。

16.18 else_result_expression

格式

无。

说明

CASE WHEN语句中ELSE语句后的返回结果。

16.19 file_format

格式

| AVRO

 版权所有 © 华为技术有限公司

| CSV

| JSON

| ORC

| PARQUET

说明

- 目前包含以上6种格式。
- 指定数据格式的方式有两种，一种是USING，可指定以上6种数据格式，另一种是STORED AS，只能指定ORC和PARQUET。
- ORC对RCFile做了优化，可以提供一种高效的方法来存储Hive数据。
- PARQUET是面向分析型业务的列式存储格式。

16.20 file_path

格式

无。

说明

文件路径，该路径是OBS路径。

16.21 function_name

格式

无。

说明

函数名称，仅支持字符串类型。

16.22 groupby_expression

格式

无。

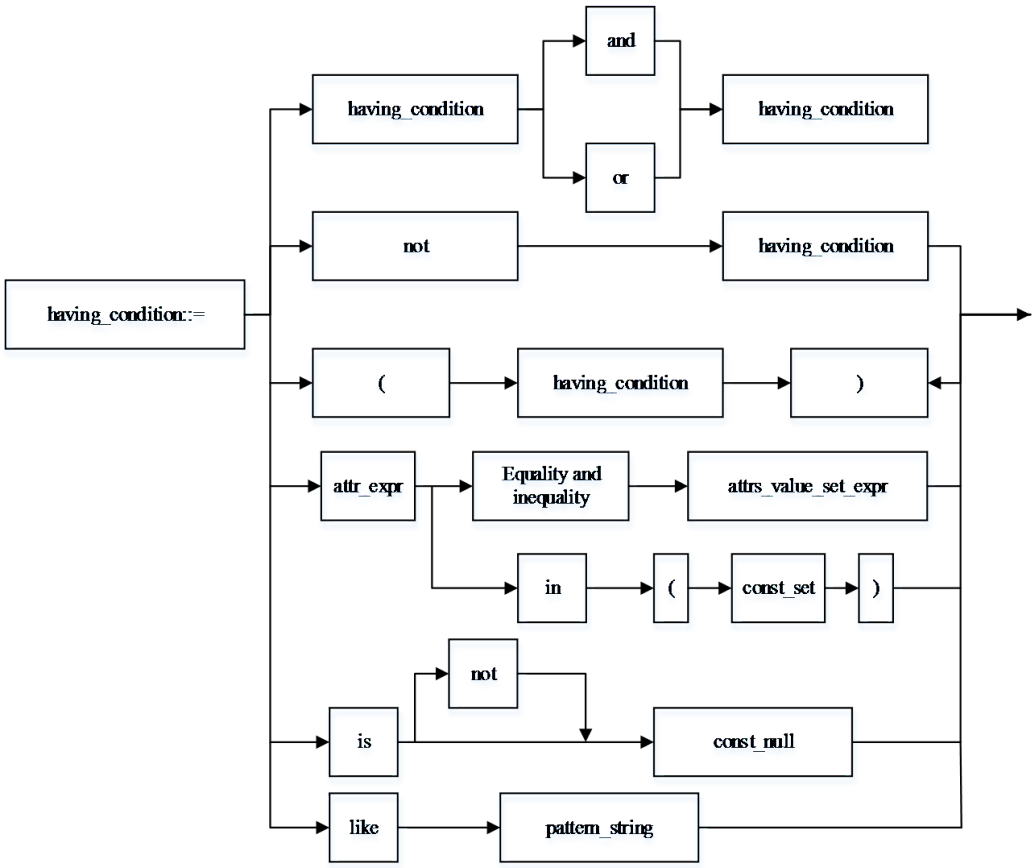
说明

包含GROUP BY的表达式。

 版权所有 © 华为技术有限公司

16.23 having_condition

格式



说明

语法	描述
having_condition	having逻辑判断条件。
and	逻辑运算符：与。
or	逻辑运算符：或。
not	逻辑运算符：非。
(	子逻辑判断条件开始。
)	子逻辑判断条件结束。
condition	逻辑判断条件。
const_set	常量集合，元素间逗号分隔。
in	关键字，用于判断属性是否在一个集合中。

版权所有 © 华为技术有限公司

语法	描述
attrs_value_set_expr	属性值集合。
attr_expr	属性表达式。
Equality and inequality	等式与不等式，详情请参见**关系运算符**。
pattern_string	模式匹配字符串，支持通配符匹配。WHERE LIKE条件过滤时，支持SQL通配符中“%”与“_”，“%”代表一个或多个字符，“_”仅代表一个字符。
like	关系运算符：用于通配符匹配。

16.24 hdfs_path

格式

无。

说明

HDFS的路径，如“hdfs:///tmp”。

16.25 input_expression

格式

无。

说明

CASE WHEN的输入表达式。

16.26 input_format_classname

格式

无。

说明

指定输入格式的类名，如***org.apache.hadoop.mapred.TextInputFormat***。

 版权所有 © 华为技术有限公司

16.27 jar_path

格式

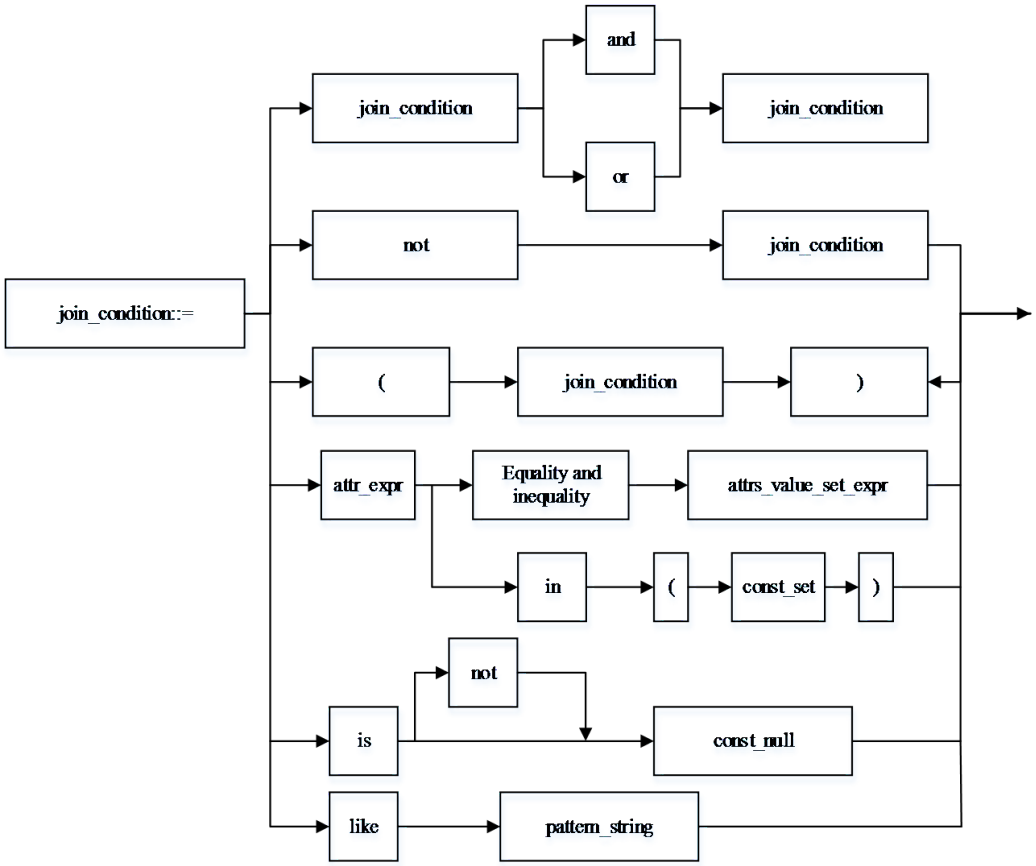
无。

说明

jar包路径，该路径可以是本地路径也可以是HDFS路径。

16.28 join_condition

格式



说明

语法	描述
join_condition	join逻辑判断条件。
and	逻辑运算符：与。

 版权所有 © 华为技术有限公司

语法	描述
or	逻辑运算符：或。
not	逻辑运算符：非。
(	子逻辑判断条件开始。
)	子逻辑判断条件结束。
condition	逻辑判断条件。
const_set	常量集合，元素间逗号分隔。
in	关键字，用于判断属性是否在一个集合中。
atrrs_value_set_expr	属性值集合。
attr_expr	属性表达式。
Equality and inequality	等式与不等式，详情请参见**关系运算符**。
pattern_string	模式匹配字符串，支持通配符匹配。WHERE LIKE条件过滤时，支持SQL通配符中“%”与“_”，“%”代表一个或多个字符，“_”仅代表一个字符。

16.29 non_equi_join_condition

格式

无。

说明

指不等式join条件。

16.30 number

格式

无。

说明

LIMIT限制输出的行数，只支持INT类型。

 版权所有 © 华为技术有限公司

16.31 num_buckets

格式

无。

说明

分桶的个数，仅支持INT类型。

16.32 output_format_classname

格式

无。

说明

指定输出格式的类名，如 ***org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat。***

16.33 partition_col_name

格式

无。

说明

分区列名，即分区字段名称，仅支持字符串类型。

16.34 partition_col_value

格式

无。

说明

分区列值，即分区字段的值。

 版权所有 © 华为技术有限公司

16.35 partition_specs

格式

partition_specs : (partition_col_name = partition_col_value, partition_col_name = partition_col_value, ...);

说明

表的分区列表，以key=value的形式表现，key为partition_col_name ，value为partition_col_value ，若存在多个分区字段，每组key=value之间用逗号分隔。

16.36 property_name

格式

无。

说明

属性名称，仅支持字符串类型。

16.37 property_value

格式

无。

说明

属性值，仅支持字符串类型。

16.38 regex_expression

格式

无。

说明

模式匹配字符串，支持通配符匹配。

 版权所有 © 华为技术有限公司

16.39 result_expression

格式

无。

说明

CASE WHEN语句中THEN语句后的返回结果。

16.40 row_format

格式

ROW FORMAT DELIMITED

[FIELDS TERMINATED BY separator]

[COLLECTION ITEMS TERMINATED BY separator]

[MAP KEYS TERMINATED BY separator] [LINES TERMINATED BY separator]

[NULL DEFINED AS separator]

| SERDE serde_name [WITH SERDEPROPERTIES (property_name=property_value, property_name=property_value, ...)]

说明

- separator指语法中的分隔符或替代符，仅支持CHAR类型。
- FIELDS TERMINATED BY指定表中字段级别的分隔符，仅支持CHAR类型。
- COLLECTION ITEMS TERMINATED BY指定集合级别的分隔符，仅支持CHAR类型
- MAP KEY TERMINATED BY仅用于指定MAP类型中的key与vaule之间的分隔符号，仅支持CHAR类型。
- LINES TERMINATED BY指定行与行之间的分隔符，目前只支持“\n”。
- 使用NULL DEFINED AS子句可以指定NULL的格式。
- SERDE serde_name [WITH SERDEPROPERTIES (property_name=property_value, property_name=property_value, ...)]可利用以下语句实现NULL值转换为空字符串。

  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' with serdeproperties('serialization.null.format' = '')

16.41 select_statement

格式

无。

 版权所有 © 华为技术有限公司

说明

SELECT基本语句，即查询语句。

16.42 separator

格式

无。

说明

分隔符，仅支持CHAR类型，支持用户自定义，如逗号、分号、冒号等。

16.43 serde_name

格式

无。

说明

指定serde的名称。

16.44 sql_containing_cte_name

格式

无。

说明

包含了cte_name定义的公共表达式的SQL语句。

16.45 sub_query

格式

无。

说明

指子查询。

 版权所有 © 华为技术有限公司

16.46 table_comment

格式

无。

说明

对表的描述，仅支持字符串类型，描述长度不能超过256字节。

16.47 table_name

格式

无。

说明

表名称，支持字符串类型和“$”符号，名称长度不能超过128字节。

16.48 table_properties

格式

无。

说明

表的属性列表，以key=value的形式表示，key为property_name，value为property_value，列表中每组key=value之间用逗号分隔。

16.49 table_reference

格式

无。

说明

表或视图的名称，仅支持字符串类型，也可为子查询，当为子查询时，必须加别名。

16.50 view_name

格式

无。

 版权所有 © 华为技术有限公司

说明

视图名称，仅支持字符串类型，名称长度不能超过128个字节。

16.51 view_properties

格式

无。

说明

视图的属性列表，以key=value的形式表示，key为property_name，value为property_value，列表中每组key=value之间用逗号分隔。

16.52 when_expression

格式

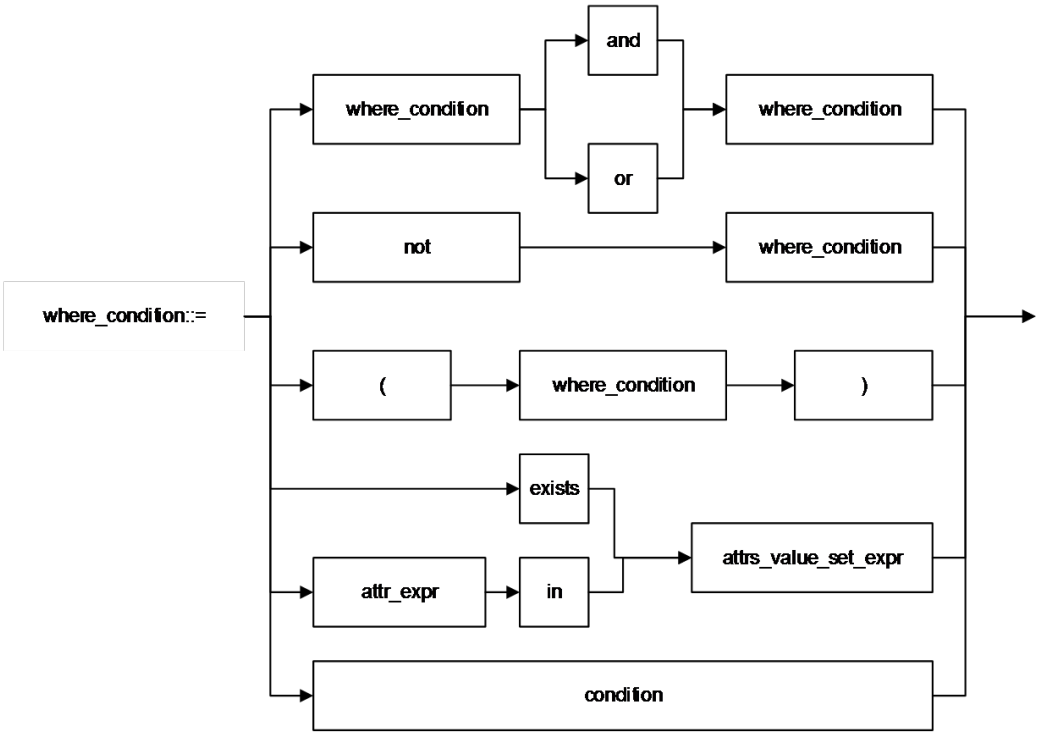
无。

说明

CASE WHEN语句的when表达式，与输入表达式进行匹配。

16.53 where_condition

格式



 版权所有 © 华为技术有限公司

说明

语法	描述
where_condition	where逻辑判断条件。
and	逻辑运算符：与。
or	逻辑运算符：或。
not	逻辑运算符：非。
(	子逻辑判断条件开始。
)	子逻辑判断条件结束。
condition	逻辑判断条件。
exists	关键字，用于判断是否存在一个不为空的集合，若exists后面跟的为子查询，子查询中须包含逻辑判断条件。
in	关键字，用于判断属性是否在一个集合中。
attrs_value_set_expr	属性值集合。
attr_expr	属性表达式。

16.54 window_function

格式

无。

说明

分析窗口函数。

 版权所有 © 华为技术有限公司

17 运算符

17.1 关系运算符

所有数据类型都可用关系运算符进行比较，并返回一个BOOLEAN类型的值。

关系运算符均为双目操作符，被比较的两个数据类型必须是相同的数据类型或者是可以进行隐式转换的类型。

DLI提供的关系运算符，请参见表17-1。

表 17-1 关系运算符

运算符	返回类型	描述
A = B	BOOLEAN	若A与B相等，返回TRUE，否则返回FALSE。用于做赋值操作。
A == B	BOOLEAN	若A与B相等，返回TRUE，否则返回FALSE。不能用于赋值操作。
A <=> B	BOOLEAN	若A与B相等，返回TRUE，否则返FALSE，若A与B都为NULL则返回TRUE，A与B其中一个为NULL则返回FALSE。
A <> B	BOOLEAN	若A与B不相等，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL，该种运算符为标准SQL语法。
A != B	BOOLEAN	与<>逻辑操作符相同，该种运算符为SQL Server语法。
A < B	BOOLEAN	若A小于B，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A <= B	BOOLEAN	若A小于或者等于B，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A > B	BOOLEAN	若A大于B，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。

版权所有 © 华为技术有限公司

运算符	返回类型	描述
A >= B	BOOLEAN	若A大于或者等于B，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A BETWEEN B AND C	BOOLEAN	若A大于等于B且小于等于C则返回TRUE，否则返回FALSE。若A、B、C三者中存在NULL，则返回NULL。
A NOT BETWEEN B AND C	BOOLEAN	若A小于B或大于C则返回TRUE，否则返回FALSE。若A、B、C三者中存在NULL，则返回NULL。
A IS NULL	BOOLEAN	若A为NULL则返回TRUE，否则返回FALSE。
A IS NOT NULL	BOOLEAN	若A不为NULL，则返回TRUE，否则返回FALSE。
A LIKE B	BOOLEAN	若字符串A与字符串B相匹配则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A NOT LIKE B	BOOLEAN	若字符串A与字符串B不相匹配则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A RLIKE B	BOOLEAN	JAVA的LIKE操作，若A或其子字符串与B相匹配，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A REGEXP B	BOOLEAN	与A RLIKE B结果相同。

17.2 算术运算符

算术运算符包括双目运算与单目运算，这些运算符都将返回数字类型。DLI所支持的算术运算符如表17-2所示。

表 17-2 算术运算符

运算符	返回类型	描述
A + B	所有数字类型	A和B相加。结果数据类型与操作数据类型相关，例如一个整数类型数据加上一个浮点类型数据，结果数值为浮点类型数据。
A–B	所有数字类型	A和B相减。结果数据类型与操作数据类型相关。
A * B	所有数字类型	A和B相乘。结果数据类型与操作数据类型相关。
A / B	所有数字类型	A和B相除。结果是一个double（双精度）类型的数值。
A % B	所有数字类型	A对B取余数，结果数据之类与操作数据类型相关。

 版权所有 © 华为技术有限公司

运算符	返回类型	描述
A & B	所有数字类型	查看两个参数的二进制表示法的值，并执行按位”与”操作。两个表达式的一位均为1时，则结果的该位为1。否则，结果的该位为0。
A | B	所有数字类型	查看两个参数的二进制表示法的值，并执行按位”或”操作。只要任一表达式的一位为1，则结果的该位为 1。否则，结果的该位为0。
A ^ B	所有数字类型	查看两个参数的二进制表示法的值，并执行按位”异或”操作。当且仅当只有一个表达式的某位上为1时，结果的该位才为1。否则结果的该位为0。
~A	所有数字类型	对一个表达式执行按位”非”操作（取反）。

17.3 逻辑运算符

常用的逻辑操作符有AND、OR和NOT，它们的运算结果有三个值，分别为TRUE、FALSE和NULL，其中NULL代表未知。优先级顺序为：NOT>AND>OR。

运算规则请参见**表17-3**，表中的A和B代表逻辑表达式。

表 17-3 逻辑运算符

运算符	返回类型	描述
A AND B	BOOLEAN	若A与B都为TRUE则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。
A OR B	BOOLEAN	若A或B为TRUE，则返回TRUE，否则返回FALSE。若A或B为NULL，则返回NULL。一个为TRUE，另一个为NULL时，返回TRUE。
NOT A	BOOLEAN	若A为FALSE则返回TRUE，若A为NULL则返回NULL，否则返回FALSE。
! A	BOOLEAN	与NOT A相同。
A IN (val1, val2, ...)	BOOLEAN	若A与(val1, val2, ...)中任意值相等则返回TRUE，否则返回FALSE。
A NOT IN (val1, val2, ...)	BOOLEAN	若A与(val1, val2, ...)中任意值都不相等则返回TRUE，否则返回FALSE。
EXISTS (subquery)	BOOLEAN	若子查询返回结果至少包含一行则返回TRUE，否则返回FALSE。

 版权所有 © 华为技术有限公司

运算符	返回类型	描述
NOT EXISTS (subquery)	BOOLEAN	若子查询返回结果一行都不包含则返回TRUE，否则返回FALSE。

 版权所有 © 华为技术有限公司