数据湖探索

SQL 语法参考

文档版本 01

发布日期 2022-12-07

华为技术有限公司

版权所有 © 华为技术有限公司 2022。 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。

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

1 Spark SQL 语法参考

1.1 批作业 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.autoBroadcastJoinThreshold	209715200	配置执行连接时显示所有工作节点的表的最大字节大小。通过将此值设置为“-1”，可以禁用显示。 **说明** 当前仅支持运行命令**ANALYZE TABLE COMPUTE statistics noscan**的配置单元元存储表，和直接根据数据文件计算统计信息的基于文件的数据源表。
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。

 版权所有 © 华为技术有限公司

名称	默认值	描述
spark.sql.badRecordsPath	-	Bad Records的路径。

1.2 批作业 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 1-2 批作业 SQL 语法

语法分类	功能描述
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
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
	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**

版权所有 © 华为技术有限公司

语法分类	功能描述
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
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

版权所有 © 华为技术有限公司

语法分类	功能描述
	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	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

1.3 数据库

1.3.1 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。

版权所有 © 华为技术有限公司

参数说明

表 1-3 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。
db_comment	数据库描述。
property_name	数据库属性名。
property_value	数据库属性值。

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

1. 队列是使用DLI服务的基础，执行SQL前需要先创建队列。
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;
   ```

1.3.2 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。

版权所有 © 华为技术有限公司

参数说明

表 1-4 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。

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;
   ```

1.3.3 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 1-5 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;
   ```


版权所有 © 华为技术有限公司

1.3.4 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];
```

关键字

无。

参数说明

表 1-6 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.*";
```

1.4 创建 OBS 表

1.4.1 使用 DataSource 语法创建 OBS 表

功能描述

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。

使用说明

- 创建表时不会统计大小。
- 添加数据时会修改大小至0。

版权所有 © 华为技术有限公司

- 如需查看表大小可以通过OBS查看。

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 1-7 参数说明

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，且不能以数字和下划线开头。
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	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。
col_type	字段类型。
col_comment	字段描述。
file_format	OBS表数据存储格式，支持orc，parquet，json，csv，avro类型。
path	数据存储路径。
table_comment	表描述。
select_statement	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。

版权所有 © 华为技术有限公司

表 1-8 OPTIONS 参数描述

参数	描述	默认值
path	指定的表路径，即OBS存储路径。	-
multiLevelDirEnable	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
dataDelegated	是否需要在删除表或分区时，清除path路径下的数据。	false
compression	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，还可以设置以下OPTIONS参数。

表 1-9 CSV 数据格式 OPTIONS 参数说明

参数	描述	默认值
delimiter	数据分隔符。	逗号（即",”）
quote	引用字符。	双引号（即“"”）
escape	转义字符。	反斜杠（即“\”）
multiLine	列数据中是否包含回车符或转行符，true为包含，false为不包含	false
dateFormat	指定CSV文件中date字段的日期格式	yyyy-MM-dd
timestampFormat	指定CSV文件中timestamp字段的日期格式	yyyy-MM-dd HH:mm:ss
mode	指定解析CSV时的模式，有三种模式。 ● PERMISSIVE：宽容模式，遇到错误的字段时，设置该行整行为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。	PERMISSIVE
header	CSV是否包含表头信息，true表示包含表头信息，false为不包含。	false
nullValue	设置代表null的字符，例如，nullValue=“\\N”表示设置\N 代表null。	-
comment	设置代表注释开头的字符，例如，comment='#'表示以#开头的行为注释。	-

版权所有 © 华为技术有限公司

参数	描述	默认值
compression	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。	none
encoding	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。	utf-8

注意事项

- 表名与列名为大小写不敏感，即不区分大小写。
- 表名及列名的描述仅支持字符串常量。
- 创建表时要声明列名及对应的数据类型，数据类型为原生类型。
- 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。
- 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。用户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。用户可以在建表语句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)。
- OBS存储路径必须为OBS上的目录，该目录必须事先创建好，且为空。
- 创建分区表时，PARTITONED 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7000个。
- CTAS建表语句不能指定表的属性，不支持创建分区表。

示例

- 创建名为parquetTable的OBS表。
```
CREATE TABLE parquetTable (name string, id int) USING parquet OPTIONS (path "obs://bucketName/filePath");
```
- 创建名为parquetZstdTable的OBS表，并指定压缩格式为zstd。
```
CREATE TABLE parquetZstdTable (name string, id string) USING parquet OPTIONS (path "obs://bucketName/filePath",compression='zstd');
```
- 以班级号（classNo）为分区字段，创建一张名为student的表，包含姓名（name）与分数（score）两个字段。
```
CREATE TABLE IF NOT EXISTS student(name STRING, score DOUBLE, classNo INT) USING csv OPTIONS (PATH 'obs://bucketName/filePath') PARTITIONED BY (classNo);
```

  说明

  “classNo”为分区字段，在表字段中要放在最后一个，即“student(name STRING, score DOUBLE, classNo INT)”。

- 创建表t1，并将表t2的数据插入到表t1中。
```
CREATE TABLE t1 USING parquet OPTIONS(path 'obs://bucketName/tblPath') AS select * from t2;
```

 版权所有 © 华为技术有限公司

1.4.2 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。

使用说明

- 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。

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。

  比如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。多版本功能其他SQL语法请参考**开启或关闭数据多版本**和**多版本备份恢复数据**章节描述。

  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。

版权所有 © 华为技术有限公司

表 1-10 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。
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。

参数说明

表 1-11 参数说明

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，且不能以数字和下划线开头。
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	字段名称。
col_type	字段类型。
col_comment	字段描述。
row_format	行数据格式。
file_format	OBS表存储格式，支持TEXTFILE, AVRO, ORC, SEQUENCEFILE, RCFILE, PARQUET
table_comment	表描述。
obs_path	OBS存储路径。
key = value	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。

版权所有 © 华为技术有限公司

参数	描述
select_statement	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。

注意事项

- 表名与列名为大小写不敏感，即不区分大小写。
- 表名及列名的描述仅支持字符串常量。
- 创建表时要声明列名及对应的数据类型，数据类型为原生类型。
- 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。
- 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100000个。
- CTAS建表语句不能指定表的属性，不支持创建分区表。

示例

- 创建一张名为student的parquet格式表，该表包含字段id，name，score，其对应的数据类型分别是INT，STRING，FLOAT。
  ```
  CREATE TABLE student (id INT, name STRING, score FLOAT) STORED AS PARQUET LOCATION 'obs://bucketName/filePath';
  ```
- 以班级号（classNo）为分区字段，创建一张名为student的表，包含姓名（name）与分数（score）两个字段。
  ```
  CREATE TABLE IF NOT EXISTS student(name STRING, score DOUBLE) PARTITIONED BY (classNo INT) STORED AS PARQUET LOCATION 'obs://bucketName/filePath';
  ```

  说明

  “classNo”为分区字段，需要在PARTITIONED BY关键字后指定，即“PARTITIONED BY (classNo INT)”，不能放在表名后作为表字段指定。
- 创建表t1，并将表t2的数据插入到表t1中（Hive语法）。
  ```
  CREATE TABLE t1 STORED AS parquet LOCATION 'obs://bucketName/filePath' as select * from t2;
  ```
- 创建表student，并开启数据多版本功能（Hive语法）。
  ```
  CREATE TABLE student (id INT, name STRING, score FLOAT) STORED AS PARQUET LOCATION 'obs://bucketName/filePath' TBLPROPERTIES ("dli.multi.version.enable"="true");
  ```

1.5 创建 DLI 表

1.5.1 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。

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 1-12 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，且不能以数字和下划线开头。
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	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。
col_type	字段类型。
col_comment	字段描述。
file_format	DLI表数据存储格式，支持：parquet格式。
table_comment	表描述。
select_statement	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。

版权所有 © 华为技术有限公司

表 1-13 OPTIONS 参数描述

参数	描述	默认值
multiLevelDirEnable	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	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。	-

注意事项

- 若没有指定分隔符，则默认为逗号（,）。
- 创建分区表时，PARTITONED 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7000个。
- CTAS建表语句不能指定表的属性，不支持创建分区表。

示例

- 创建一张名为src的表，该表包含字段key、value，其对应的数据类型分别是INT、STRING，并指定表的压缩格式为'zstd'。
  ```
  CREATE TABLE src(key INT, value STRING) USING PARQUET OPTIONS(compression = 'zstd');
  ```
- 以班级号（classNo）为分区字段，创建一张名为student的表，包含姓名（name）与分数（score）两个字段，存储格式为parquet。
  ```
  CREATE TABLE student(name STRING, score INT, classNo INT) USING PARQUET OPTIONS('key1' = 'value1') PARTITIONED BY(classNo) ;
  ```

  说明

  “classNo”为分区字段，在表字段中要放在最后一个，即“student(name STRING, score INT, classNo INT)”。
- 创建表t1，并将表t2的数据插入到表t1中。
  ```
  CREATE TABLE t1 USING parquet AS select * from t2;
  ```

1.5.2 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。

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]
```


版权所有 © 华为技术有限公司

```
 STORED AS file_format
 [TBLPROPERTIES (key1=val1, key2=val2, ...)]
 [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：TBLPROPERTIES子句允许用户给表添加key/value的属性。

  比如，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 1-14 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，且不能以数字和下划线开头。
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	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。
col_type	字段类型。
col_comment	字段描述。
row_format	行数据格式。
file_format	DLI表数据存储格式：支持TEXTFILE, AVRO, ORC, SEQUENCEFILE, RCFILE, PARQUET。
table_comment	表描述。

 版权所有 © 华为技术有限公司

参数	描述
select_statement	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。

注意事项

- 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100000个。
- CTAS建表语句不能指定表的属性，不支持创建分区表。

示例

- 创建一张名为src的表，该表包含字段key、value，其对应的数据类型分别是INT、STRING，并可根据需要指定属性。

```
CREATE TABLE src
  (key INT, value STRING)
  STORED AS PARQUET
  TBLPROPERTIES('key1' = 'value1');
```

- 以班级号（classNo）为分区字段，创建一张名为student的表，包含姓名（name）与分数（score）两个字段，并指定表的压缩格式为zstd。

```
CREATE TABLE student
  (name STRING, score INT)
  STORED AS PARQUET
  TBLPROPERTIES(parquet.compression = 'zstd') PARTITIONED BY(classNo INT);
```

- 创建表t1，并将表t2的数据插入到表t1中。

```
CREATE TABLE t1
  STORED AS PARQUET
  AS select * from t2;
```

1.6 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。

版权所有 © 华为技术有限公司

参数说明

表 1-15 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**章节中的示例描述已创建对应的表，如student。
2. 在当前所在数据库下删除名为student的表。

```
DROP TABLE IF EXISTS student;
```

1.7 查看表

1.7.1 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。

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 1-16 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

参数	描述
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**章节中的示例描述已创建对应的表。
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*";
   ```

1.7.2 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;
```

关键字

CREATE TABLE：建表语句。

参数说明

表 1-17 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

1. 参考**创建OBS表**或者**创建DLI表**章节中的示例描述已创建对应的表，如test。
2. 返回test表的建表语句。
   ```
   SHOW CREATE TABLE test;
   ```


版权所有 © 华为技术有限公司

1.7.3 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 1-18 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');
```

1.7.4 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 1-19 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;
```

1.7.5 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 1-20 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')。
  ```

1.7.6 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 1-21 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。

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;
```

1.8 修改表

1.8.1 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 1-22 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

无。

示例

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


版权所有 © 华为技术有限公司

1.8.2 开启或关闭数据多版本

功能描述

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**。

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 1-23 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");
```


版权所有 © 华为技术有限公司

- 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;
  ```

1.9 分区表相关

1.9.1 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 1-24 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。

版权所有 © 华为技术有限公司

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。
- “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。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

- 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';
  ```

 版权所有 © 华为技术有限公司

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';
```

1.9.2 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 1-25 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');
```

 版权所有 © 华为技术有限公司

1.9.3 删除分区

功能描述

删除分区表的一个或多个分区。

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 1-26 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 (dt='2009-09-09',city='xxx')**。

示例

将分区表student的分区dt = '2008-08-08', city = 'xxx'删除。

```
ALTER TABLE student
  DROP
  PARTITION (dt = '2008-08-08', city = 'xxx');
```


版权所有 © 华为技术有限公司

1.9.4 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**。
- 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 1-27 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' ● <partition_filtercondition1> AND|OR <partition_filtercondition2> 例如：start_date < '201911' OR start_date >= '202006' ● （<partition_filtercondition1>)[,partitions (<partition_filtercondition2>), ...] 例如：(start_date <> '202007'), partitions(start_date < '201912')

版权所有 © 华为技术有限公司

示例

将分区表student的分区dt，按照各种筛选过滤条件删除。

```
alter table student drop partitions(start_date < '201911');
alter table student drop partitions(start_date >= '202007');
alter table student drop partitions(start_date BETWEEN '202001' AND '202007');
alter table student drop partitions(start_date < '201912' OR start_date >= '202006');
alter table student drop partitions(start_date > '201912' AND start_date <= '202004');
alter table student drop partitions(start_date != '202007');
alter table student drop partitions(start_date <> '202007');
alter table student drop partitions(start_date <> '202007'), partitions(start_date < '201912');
```

1.9.5 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 1-28 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。

版权所有 © 华为技术有限公司

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';
```

1.9.6 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 1-29 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;
```


版权所有 © 华为技术有限公司

或

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

1.9.7 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 1-30 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;
```


版权所有 © 华为技术有限公司

1.10 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 1-31 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')

  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'。

版权所有 © 华为技术有限公司

  - 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'= "'"。
- 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 1-32 MAXCOLUMNS

可选参数名称	默认值	最大值
MAXCOLUMNS	2000	20000

  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1-33**。

 版权所有 © 华为技术有限公司

表 1-33 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**。

- 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')

 版权所有 © 华为技术有限公司

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**章节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');
```

1.11 插入数据

功能描述

将SELECT查询结果或某条数据插入到表中。

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 ...];
```


版权所有 © 华为技术有限公司

```
values_row:
 : (val1 [, val2, ...])
```

关键字

表 1-34 INSERT 参数

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。
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，因为有一定概率发生并发冲突，导致插入失败。
- 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。
- 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。

 版权所有 © 华为技术有限公司

- 通过配置“spark.sql.shuffle.partitions”参数可以设置表在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**章节中的示例描述创建对应的表。

- 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);
```
- 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);
```

1.12 清空数据

功能描述

清除表的数据。

语法格式

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

关键字

表 1-35 参数

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

 版权所有 © 华为技术有限公司

注意事项

只支持清除表的数据。

示例

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

1.13 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 1-36 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1-9**。

注意事项

- 通过配置“spark.sql.shuffle.partitions”参数可以设置表在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);
  ```
- 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'。

版权所有 © 华为技术有限公司

- 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'= "'"。

示例

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

1.14 多版本备份恢复数据

1.14.1 设置多版本备份数据保留周期

功能描述

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**。

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 1-37 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。
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**。

版权所有 © 华为技术有限公司

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");
```

1.14.2 查看多版本备份数据

功能描述

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**。

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 1-38 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。
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;
```


版权所有 © 华为技术有限公司

- 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');
  ```

1.14.3 恢复多版本备份数据

功能描述

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**。

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**。

参数说明

表 1-39 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。
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**。

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';
  ```

1.14.4 配置多版本过期数据回收站

功能描述

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**。

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 1-40 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：
  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。

版权所有 © 华为技术有限公司

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。

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");
```

1.14.5 清理多版本数据

功能描述

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';
```

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 1-41 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）。

版权所有 © 华为技术有限公司

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

1.15 跨源连接 HBase 表

1.15.1 创建表关联 HBase

功能描述

使用CREATE TABLE命令创建表并关联HBase上已有的表。

前提条件

- 创建表关联HBase之前需要创建跨源连接。
- 请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。
- 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 [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,
  ATTR2 String,
  ATTR3 TYPE)
  USING [HBASE] OPTIONS (
  'ZKHost'='xx',
  'TableName'='TABLE_IN_HBASE',
  'RowKey'='ATTR1:2, ATTR2:10',
  'Cols'='ATTR2:CF1.C1, ATTR3:CF1.C2'
```

关键字

表 1-42 CREATE TABLE 参数描述

参数	描述
USING [HBASE]	指定hbase datasource，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。 ● 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**

版权所有 © 华为技术有限公司

参数	描述
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	通过逗号分隔的表字段与HBase表的列之间的对应关系。其中，冒号前面放置表字段，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。
- Cols与RowKey中的字段加起来的数量必须与表的字段保持一致，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');
```

 版权所有 © 华为技术有限公司

1.15.2 插入数据至 HBase 表

功能描述

INSERT INTO命令将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**SELECT基本语句**。

参数说明

表 1-43 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

- 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。

版权所有 © 华为技术有限公司

- 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);
```

1.15.3 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。

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 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

版权所有 © 华为技术有限公司

- 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)
    ```

  - 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)
      ```

1.16 跨源连接 OpenTSDB 表

版权所有 © 华为技术有限公司

1.16.1 创建表关联 OpenTSDB

功能描述

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

前提条件

创建表关联OpenTSDB之前需要创建跨源连接。

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');
```

关键字

表 1-44 CREATE TABLE 参数描述

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。 ● 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.com:4242',
  'metric' = 'city.temp',
  'tags' = 'city,location');
```

 版权所有 © 华为技术有限公司

1.16.2 插入数据至 OpenTSDB 表

功能描述

使用INSERT INTO命令将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);
```

关键字

表 1-45 INSERT INTO 参数描述

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

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);
```

1.16.3 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;
```


版权所有 © 华为技术有限公司

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;
```

1.17 跨源连接 DWS 表

1.17.1 创建表关联 DWS

功能描述

使用CREATE TABLE命令创建表并关联DWS上已有的表。

前提条件

创建表关联DWS之前需要创建跨源连接。

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');
```


版权所有 © 华为技术有限公司

关键字

表 1-46 CREATE TABLE 参数描述

参数	描述
url	DWS的连接地址，需要先创建跨源连接。 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。
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》>。
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。

版权所有 © 华为技术有限公司

参数	描述
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

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');
```

1.17.2 插入数据至 DWS 表

功能描述

INSERT INTO命令将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] ...;
```


版权所有 © 华为技术有限公司

- 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**SELECT基本语句**。

参数说明

表 1-47 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

- 表必须已经存在。
- 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);
```

1.17.3 查询 DWS 表

SELECT命令用于查询DWS表中的数据。

 版权所有 © 华为技术有限公司

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;
```

1.18 跨源连接 RDS 表

1.18.1 创建表关联 RDS

功能描述

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

前提条件

创建表关联RDS之前需要创建跨源连接。

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');
```


版权所有 © 华为技术有限公司

关键字

表 1-48 CREATE TABLE 参数描述

参数	描述
url	RDS的连接地址，需要先创建跨源连接。 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://192.168.0.193:3306/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》>。
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。

版权所有 © 华为技术有限公司

参数	描述
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

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

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 (
```

 版权所有 © 华为技术有限公司

```
'url'='jdbc:postgresql://to-rds-1174405119-oLRHAGE7.datasource.com:3306/postgreDB',
'driver'='org.postgresql.Driver',
'dbtable'='pg_schema.test1',
'passwdauth' = 'xxx',
'encryption' = 'true');
```

1.18.2 插入数据至 RDS 表

功能描述

INSERT INTO命令将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**SELECT基本语句**。

参数说明

表 1-49 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

- 表必须已经存在。
- 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。

 版权所有 © 华为技术有限公司

- 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);
```

1.18.3 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;
```

1.19 跨源连接 CSS 表

1.19.1 创建表关联 CSS

功能描述

使用CREATE TABLE命令创建表并关联CSS上已有的表。

前提条件

创建表关联CSS之前需要创建跨源连接。

语法格式

```
CREATE TABLE [IF NOT EXISTS] TABLE_NAME(
  FIELDNAME1 FIELDTYPE1,
```


版权所有 © 华为技术有限公司

```
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 1-50 CREATE TABLE 参数描述

参数	描述
es.nodes	CSS的连接地址，需要先创建跨源连接。 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。
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

版权所有 © 华为技术有限公司

参数	描述
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 》>。

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');
```

1.19.2 插入数据至 CSS 表

功能描述

INSERT INTO命令将表中的数据插入到已关联的指定CSS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**SELECT基本语句**。

参数说明

表 1-51 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

- 表必须已经存在。
- 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
```

- 插入数据“1”到表“test”中

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

1.19.3 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。

版权所有 © 华为技术有限公司

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

1.20 跨源连接 DCS 表

1.20.1 创建表关联 DCS

功能描述

使用CREATE TABLE命令创建表并关联DCS上已有的Key。

前提条件

创建表关联DCS之前需要创建跨源连接，绑定队列。

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',
  '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 1-52 CREATE TABLE 参数描述

参数	描述
host	DCS的连接IP，需要先创建跨源连接。 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。

版权所有 © 华为技术有限公司

参数	描述
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》>。
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。
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*:*',
```

 版权所有 © 华为技术有限公司

```
  'key.column' = 'id'
);
```

1.20.2 插入数据至 DCS 表

功能描述

INSERT INTO命令将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 ...];
```

关键字

SELECT对应关键字说明请参考**SELECT基本语句**。

参数说明

表 1-53 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

- 表必须已经存在。
- 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',
```


版权所有 © 华为技术有限公司

```
  'password' = '******',
  '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',
  'port' = '6379',
  'password' = '******',
  '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);
```

1.20.3 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。

 版权所有 © 华为技术有限公司

示例

查询表test_redis中的数据。

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

1.21 跨源连接 DDS 表

1.21.1 创建表关联 DDS

功能描述

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

前提条件

创建表关联DDS之前需要创建跨源连接，绑定队列。

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'
);
```

关键字

表 1-54 CREATE TABLE 参数描述

参数	描述
url	DDS的连接信息，需要先创建跨源连接 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

版权所有 © 华为技术有限公司

参数	描述
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》>。
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。

示例

```
create table 1_datasource_mongo.test_mongo(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');
```

1.21.2 插入数据至 DDS 表

功能描述

INSERT INTO命令将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**SELECT基本语句**。

版权所有 © 华为技术有限公司

参数说明

表 1-55 参数描述

参数	描述
DLI_TABLE	已创建跨源连接的表名称。
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

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);
```

1.21.3 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"，例如：

版权所有 © 华为技术有限公司

示例

查询表test_mongo中的数据。

```
SELECT * FROM test_mongo limit 100;
```

1.22 视图

1.22.1 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。

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;
```

1.22.2 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。

版权所有 © 华为技术有限公司

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;
```

1.23 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;
```

1.24 数据权限管理

1.24.1 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1-56**所示。

表 1-56 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	-

版权所有 © 华为技术有限公司

分类	SQL语句	权限	说明
	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	-
	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	-

版权所有 © 华为技术有限公司

分类	SQL语句	权限	说明
	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）
  - 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）

版权所有 © 华为技术有限公司

- 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）

1.24.2 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;
```

1.24.3 删除角色

功能描述

在当前database或指定database中删除角色。

版权所有 © 华为技术有限公司

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;
```

1.24.4 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;
```

1.24.5 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。

版权所有 © 华为技术有限公司

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;
```

1.24.6 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;
  ```

1.24.7 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。

版权所有 © 华为技术有限公司

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**。
- 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;
```

 版权所有 © 华为技术有限公司

1.24.8 回收权限

功能描述

回收已经授予用户或角色的权限。

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;
```

1.24.9 显示已授权限

功能描述

显示某个用户或角色在resource上已经授予的权限。

语法格式

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

关键字

ROLE：限定后面的role_name是一个角色。

版权所有 © 华为技术有限公司

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：

- 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;
```

显示角色role_name在数据库db1的表tb1上的权限。

```
SHOW GRANT ROLE role_name ON databases.db1.tables.tb1;
```

1.24.10 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;
```

1.25 数据类型

1.25.1 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。

版权所有 © 华为技术有限公司

DLI当前只支持原生数据类型。

1.25.2 原生数据类型

DLI支持原生数据类型，请参见表1-57。

表 1-57 原生数据类型

数据类型	描述	存储空间	范围	OBS表支持情况
INT	有符号整数	4字节	-2147483648 ~ 2147483647	是
STRING	字符串	-	-	是
FLOAT	单精度浮点型	4字节	-	是
DOUBLE	双精度浮点型	8字节	-	是
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
BOOLEAN	布尔类型	1字节	TRUE/FALSE	是
SMALLINT/ SHORT	有符号整数	2字节	-32768~32767	是
TINYINT	有符号整数	1字节	-128~127	是
BIGINT/ LONG	有符号整数	8字节	-9223372036854775808 ~ 92233720368 54775807	是
TIMESTAMP	时间戳，表示日期和时间，格式为原始数据。例如：1621434131222	-	-	是
CHAR	固定长度字符串	-	-	是
VARCHAR	可变长度字符串	-	-	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
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

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。

版权所有 © 华为技术有限公司

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。

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1-58所示。

表 1-58 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。

版权所有 © 华为技术有限公司

显式转换	转换结果
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。

1.25.3 复杂数据类型

Spark SQL支持复杂数据类型，如**表1-59**所示。

表 1-59 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。

版权所有 © 华为技术有限公司

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：

   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
   ```


版权所有 © 华为技术有限公司

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('zhang',23));

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

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

3. 查询结果。

   查询“struct_test”表中的所有数据。

   SELECT * FROM struct_test;

   ```
   8    {"name":"zhang","age":23}
   10    {"name":"wang","age":26}
   9    {"name":"li","age":25}
   ```

   查询“struct_test”表中的**name**和**age**数据。

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

   ```
   8    zhang    23
   10    wang    26
   9    li    25
   ```

1.26 自定义函数

1.26.1 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。

语法格式

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

resource:
  : (JAR|FILE|ARCHIVE)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。

版权所有 © 华为技术有限公司

关键字

- TEMPORARY：所创建的函数仅在当前会话中可用，并且不会持久化到底层元数据库中（如果有的话）。不能为临时函数指定数据库名称。
- USING <resources>：需要加载的资源。可以是JAR、文件或者URI的列表。

示例

创建函数mergeBill。

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

1.26.2 删除函数

功能描述

删除函数。

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;
```

1.26.3 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。

版权所有 © 华为技术有限公司

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;
```

1.26.4 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);
```

其中regex为正则表达式，可以参考如下**表1-60**参数样例。

表 1-60 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。

版权所有 © 华为技术有限公司

```
SHOW FUNCTIONS;
```

1.27 内置函数

1.27.1 数学函数

DLI所支持的数学函数如表1-61所示。

表 1-61 数学函数

函数	返回值	描述
round(DOUBLE a)	DOUBLE	四舍五入。
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。
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(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。
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。
ceil(DOUBLE a), ceiling(DOUBLE a)	BIGINT	将参数向上舍入为最接近的整数。例如：ceil(21.2)，返回22。
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。
exp(DOUBLE a), exp(DECIMAL a)	DOUBLE	返回e的a次方。
ln(DOUBLE a), ln(DECIMAL a)	DOUBLE	返回给定数值的自然对数。
log10(DOUBLE a), log10(DECIMAL a)	DOUBLE	返回给定数值的以10为底自然对数。
log2(DOUBLE a), log2(DECIMAL a)	DOUBLE	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

函数	返回值	描述
log(DOUBLE base, DOUBLE a) log(DECIMAL base, DECIMAL a)	DOUBLE	返回给定底数及指数返回自然对数。
pow(DOUBLE a, DOUBLE p), power(DOUBLE a, DOUBLE p)	DOUBLE	返回a的p次幂。
sqrt(DOUBLE a), sqrt(DECIMAL a)	DOUBLE	返回数值的平方根。
bin(BIGINT a)	STRING	返回二进制格式。
hex(BIGINT a) hex(STRING a)	STRING	将整数或字符转换为十六进制格式。
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。
abs(DOUBLE a)	DOUBLE	取绝对值。
pmod(INT a, INT b), pmod(DOUBLE a, DOUBLE b)	INT or DOUBLE	返回a除b的余数的绝对值。
sin(DOUBLE a), sin(DECIMAL a)	DOUBLE	返回给定角度a的正弦值。
asin(DOUBLE a), asin(DECIMAL a)	DOUBLE	返回给定角度a的反正弦值。
cos(DOUBLE a), cos(DECIMAL a)	DOUBLE	返回给定角度a的余弦值。
acos(DOUBLE a), acos(DECIMAL a)	DOUBLE	返回给定角度a的反余弦值。
tan(DOUBLE a), tan(DECIMAL a)	DOUBLE	返回给定角度a的正切值。
atan(DOUBLE a), atan(DECIMAL a)	DOUBLE	返回给定角度a的反正切值。
degrees(DOUBLE a), degrees(DECIMAL a)	DOUBLE	返回弧度所对应的角度。

版权所有 © 华为技术有限公司

函数	返回值	描述
radians(DOUBLE a), radians(DECIMAL a)	DOUBLE	返回角度所对应的弧度。
positive(INT a), positive(DOUBLE a)	INT or DOUBLE	返回a的值，例如positive(2)，返回2。
negative(INT a), negative(DOUBLE a)	INT or DOUBLE	返回a的相反数，例如negative(2)，返回-2。
sign(DOUBLE a), sign(DECIMAL a)	DOUBLE or INT	返回a所对应的正负号，a为正返回1.0，a为负，返回-1.0，否则则返回0.0。
e()	DOUBLE	返回e的值。
pi()	DOUBLE	返回pi的值。
factorial(INT a)	BIGINT	返回a的阶乘。
cbrt(DOUBLE a)	DOUBLE	返回a的立方根。
shiftleft(TINYINT|SMALLINT|INT a, INT b) shiftleft(BIGINT a, INT b)	INT BIGINT	有符号左边移，将a的二进制数按位左移b位。
shiftright(TINYINT|SMALLINT|INT a, INT b) shiftright(BIGINT a, INT b)	INT BIGINT	有符号右移，将a的二进制数按位右移b位。
shiftrightunsigned(TINYINT|SMALLINT|INT a, INT b), shiftrightunsigned(BIGINT a, INT b)	INT BIGINT	无符号右移，将a的二进制数按位右移b位。
greatest(T v1, T v2, ...)	T	返回列表中的最大值。
least(T v1, T v2, ...)	T	返回列表中的最小值。

 版权所有 © 华为技术有限公司

1.27.2 日期函数

DLI所支持的日期函数如表1-62所示。

表 1-62 日期/时间函数

函数	返回值	描述
from_unixtime(bigint unixtime[, string format])	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')
unix_timestamp()	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。
unix_timestamp(string date)	BIGINT	指定日期参数调用UNIX_TIMESTAMP()，它返回“1970-01-01 00:00:00”到指定日期的秒数。
unix_timestamp(string date, string pattern)	BIGINT	转换pattern格式的日期到UNIX时间戳：unix_timestamp("2009-03-20", "yyyy-MM-dd") = 1237532400
to_date(string timestamp)	STRING	返回时间中的年月日，例如：to_date("1970-01-01 00:00:00") = "1970-01-01"
year(string date)	INT	返回指定日期中的年份。
quarter(string date/timestamp/string)	INT	返回该date/timestamp/string所在的季度，如quarter('2015-04-01')=2。
month(string date)	INT	返回指定时间的月份，范围为1至12月。
day(string date) dayofmonth(string date)	INT	返回指定时间的日期。
hour(string date)	INT	返回指定时间的小时，范围为0到23。
minute(string date)	INT	返回指定时间的分钟，范围为0到59。
second(string date)	INT	返回指定时间的秒，范围为0到59。
weekofyear(string date)	INT	返回指定日期是一年中的第几周，范围为0到53。
datediff(string enddate, string startdate)	INT	两个时间参数的天数之差。
date_add(string startdate, int days)	STRING	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	返回值	描述
date_sub(string startdate, int days)	STRING	给定时间，在此基础上减去指定的时间段。
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，如from_utc_timestamp('1970-01-01 08:00:00','PST') returns 1970-01-01 00:00:00。
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，如to_utc_timestamp('1970-01-01 00:00:00','PST') returns 1970-01-01 08:00:00。
current_date()	DATE	返回当前日期，如2016-07-04。
current_timestamp()	TIMESTAMP	返回当前时间，如2016-07-04 11:18:11.685。
add_months(string start_date, int num_months)	STRING	返回start_date在num_months个月之后的date。
last_day(string date)	STRING	返回date所在月份的最后一天，格式为yyyy-MM-dd，如2015-08-31。
next_day(string start_date, string day_of_week)	STRING	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）。
trunc(string date, string format)	STRING	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。
months_between(string date1, string date2)	DOUBLE	返回date1与date2之间的月份差。
date_format(date/timestamp/string ts, string fmt)	STRING	返回date/timestamp/string的格式化输出，格式支持JAVA的SimpleDateFormat格式，如date_format('2015-04-08', 'y') = '2015'。 其中，y表示Year, 如果是Y，则表示Week year。Week year表示当天所在的周属于的年份，一周从周日开始，周六结束，如果本周跨年，那么这周就算入下一年。

1.27.3 字符串函数

DLI所支持的字符串函数如**表1-63**所示。

版权所有 © 华为技术有限公司

表 1-63 字符串函数

函数	返回值	描述
ascii(string str)	INT	返回字符串中首字符的数字值。
concat(string A, string B...)	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(string SEP, string A, string B...)	STRING	连接多个字符串，字符串之间以指定的分隔符分隔。
encode(string src, string charset)	BINARY	用charset的编码方式对src进行编码。
find_in_set(string str, string strList)	INT	返回字符串str第一次在strlist出现的位置。如果任一参数为NULL，返回NULL。如果第一个参数包含逗号，返回0。
get_json_object(string json_string, 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(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，但str中不存在substr时返回0，注意下标从1开始。
length(string A)	INT	返回字符串的长度。
locate(string substr, string str[, int pos])	INT	返回在下标pos（从1开始）之后，substr在str中出现的最小下标。
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(string str, int len, string pad)	STRING	返回指定长度的字符串，给定字符串str长度小于指定长度len时，由指定字符pad从左侧填补。
ltrim(string A)	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	返回值	描述
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(String format, Obj... args)	STRING	将输入按特定格式打印输出。
regexp_extract(string subject, string pattern, int index)	STRING	通过下标返回正则表达式指定的部分。regexp_extract('foothebar', 'foo(.*?)(bar)', 2) 返回：'bar.'
regexp_replace(string A, string B, string C)	STRING	字符串A中的B字符被C字符替代。
repeat(string str, int n)	STRING	重复N次字符串。
reverse(string A)	STRING	返回倒序字符串。
rpad(string str, int len, string pad)	STRING	返回指定长度的字符串，给定字符串str长度小于指定长度len时，由指定字符pad从右侧填补。
rtrim(string A)	STRING	删除字符串右边的空格，其他的空格保留。
space(int n)	STRING	返回指定数量的空格。
substr(string A, int start) substring(string A, int start)	STRING	从文本字符串A中截取指定的起始位置后的字符。
substr(string A, int start, int len) substring(string A, int start, int len)	STRING	从文本字符串A中截取指定的起始位置后指定长度的字符。
substring_index(string A, string delim, int count)	STRING	返回字符串A在delim出现count次之前的子字符串。

 版权所有 © 华为技术有限公司

函数	返回值	描述
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(string A)	STRING	删除字符串两端的空格，字符之间的空格保留。
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。
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。
levenshtein(string A, string B)	INT	返回两个字符串之间的Levenshtein距离，如levenshtein('kitten','sitting') =3。
soundex(string A)	STRING	从str返回一个soundex字符串，如soundex('Miller')= M460。

1.27.4 聚合函数

聚集函数是从一组输入值计算一个结果。例如使用COUNT函数计算SQL查询语句返回的记录行数。聚合函数如**表1-64**所示。

表 1-64 聚合函数表

函数	返回值类型	描述
count(*), count(expr), count(DISTINCT expr[, expr...])	BIGINT	返回记录条数。
sum(col), sum(DISTINCT col)	DOUBLE	求和。
avg(col), avg(DISTINCT col)	DOUBLE	求平均值。
min(col)	DOUBLE	返回最小值。
max(col)	DOUBLE	返回最大值。
variance(col), var_pop(col)	DOUBLE	返回列的方差。
var_samp(col)	DOUBLE	返回指定列的样本方差。

版权所有 © 华为技术有限公司

函数	返回值类型	描述
stddev_pop(col)	DOUBLE	返回指定列的偏差。
stddev_samp(col)	DOUBLE	返回指定列的样本偏差。
covar_pop(col1, col2)	DOUBLE	返回两列数值协方差。
covar_samp(col1, col2)	DOUBLE	返回两列数值样本协方差。
corr(col1, col2)	DOUBLE	返回两列数值的相关系数。
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(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。

说明

函数如var_pop，stddev_pop，var_samp，stddev_samp，covar_pop，covar_samp，corr，percentile_approx，不支持非数值数据类型，如TimeStamp。

1.27.5 分析窗口函数

窗口函数用于在与当前输入值相关的一组值上执行相关函数计算(包括在GROUP BY中使用的聚集函数，如sum函数、max函数、min函数、count函数、avg函数)，此外分析窗口函数还包括如表1-65中所示的函数。窗口是由一个OVER子句定义的多行记录，窗口函数作用于一个窗口。

表 1-65 函数介绍

函数	返回值	描述
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(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 (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）。

版权所有 © 华为技术有限公司

函数	返回值	描述
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）。
row_number() over (order by col_1[,col_2 ...])	INT	为每一行指派一个唯一的编号。
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。
cume_dist()	DOUBLE	计算某个值在一行中的相对位置。
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)。

1.28 SELECT 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 1-66 SELECT 参数描述

参数	描述
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	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，否则会出错。

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;
```

1.29 过滤 SELECT

1.29.1 WHERE 过滤子句

功能描述

利用WHERE子句过滤查询结果。

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM table_reference
  WHERE where_condition;
```

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：条件过滤关键字，将不满足条件的记录过滤掉，返回满足要求的记录。

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，否则会出错。

示例

将表student中，score在（90，95）之间的记录筛选出来。

```
SELECT * FROM student
  WHERE score > 90 AND score < 95;
```

1.29.2 HAVING 过滤子句

功能描述

利用HAVING子句过滤查询结果。

语法格式

```
SELECT [ALL | DISTINCT] attr_expr_list FROM table_reference
  [WHERE where_condition]
  [GROUP BY col_name_list]
  HAVING having_condition;
```

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。
- HAVING：一般与GROUP BY合用，先通过GROUP BY进行分组，再在HAVING子句中进行过滤，HAVING子句支持算术运算，聚合函数等。

注意事项

- 所查询的表必须是已经存在的，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。

示例

根据字段name对表student进行分组，再按组将score最大值大于95的记录筛选出来。

```
SELECT name, max(score) FROM student
  GROUP BY name
  HAVING max(score) >95;
```

1.30 排序 SELECT

1.30.1 ORDER BY

功能描述

按字段实现查询结果的全局排序。

版权所有 © 华为技术有限公司

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;
```

1.30.2 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;
```


版权所有 © 华为技术有限公司

1.30.3 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;
```

1.30.4 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;
```


版权所有 © 华为技术有限公司

1.31 分组 SELECT

1.31.1 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。

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;
```

1.31.2 用表达式 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。

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。
- 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);
```

1.31.3 GROUP BY 中使用 HAVING 过滤

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。

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;
```

1.31.4 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。

版权所有 © 华为技术有限公司

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。

```
SELECT group_id, job, SUM(salary) FROM group_test
  GROUP BY group_id, job
  WITH ROLLUP;
```

1.31.5 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;
  ```


版权所有 © 华为技术有限公司

- ***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。

```
SELECT group_id, job, SUM(salary) FROM group_test
  GROUP BY group_id, job
  GROUPING SETS (group_id, job);
```

1.32 连接操作 SELECT

1.32.1 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。

版权所有 © 华为技术有限公司

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);
```

1.32.2 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。

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);
```

1.32.3 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。

 版权所有 © 华为技术有限公司

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);
```

1.32.4 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。

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);
```

1.32.5 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;
```


版权所有 © 华为技术有限公司

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;
```

1.32.6 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);
```

1.32.7 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。

版权所有 © 华为技术有限公司

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);
```

1.32.8 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;
```

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);
```

1.33 子查询

版权所有 © 华为技术有限公司

1.33.1 WHERE 嵌套子查询

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');
```

1.33.2 FROM 子句嵌套子查询

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。

版权所有 © 华为技术有限公司

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;
```

1.33.3 HAVING 子句嵌套子查询

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。

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);
```

1.33.4 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。

版权所有 © 华为技术有限公司

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。

```
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;
```

1.34 别名 SELECT

1.34.1 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;
  ```


版权所有 © 华为技术有限公司

1.34.2 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。
- 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";
```

1.35 集合运算 SELECT

1.35.1 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。

版权所有 © 华为技术有限公司

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;
```

1.35.2 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。

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;
```

1.35.3 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

版权所有 © 华为技术有限公司

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;
```

1.36 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

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;
```

1.37 CASE...WHEN

1.37.1 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;
```


版权所有 © 华为技术有限公司

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;
```

1.37.2 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

1.38 OVER 子句

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;
```


版权所有 © 华为技术有限公司

2 Flink SQL 语法参考

2.1 SQL 语法约束与定义

语法约束

- 当前Flink SQL只支持SELECT，FROM，WHERE，UNION，聚合，窗口，流表JOIN以及流流JOIN。
- 数据不能对Source流做insert into操作。
- Sink流不能用来做查询操作。

语法支持范围

- 基础类型： VARCHAR，STRING，BOOLEAN，TINYINT，SMALLINT，INTEGER/INT，BIGINT，REAL/FLOAT，DOUBLE，DECIMAL，DATE，TIME，TIMESTAMP
- Array：使用[]进行引用。例如：
  ```
  insert into temp select CARDINALITY(ARRAY[1,2,3]) FROM OrderA;
  ```

语法定义

```
INSERT INTO stream_name query;
query:
  values
  | {
      select
      | selectWithoutFrom
      | query UNION [ ALL ] query
    }

orderItem:
  expression [ ASC | DESC ]

select:
  SELECT
  { * | projectItem [, projectItem ]* }
  FROM tableExpression [ JOIN tableExpression ]
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
  [ HAVING booleanExpression ]

selectWithoutFrom:
```

 版权所有 © 华为技术有限公司

```
  SELECT [ ALL | DISTINCT ]
  { * | projectItem [, projectItem ]* }

projectItem:
  expression [ [ AS ] columnAlias ]
  | tableAlias . *

tableExpression:
  tableReference

tableReference:
  tablePrimary
  [ [ AS ] alias [ '(' columnAlias [, columnAlias ]* ')' ] ]

tablePrimary:
  [ TABLE ] [ [ catalogName . ] schemaName . ] tableName
  | LATERAL TABLE '(' functionName '(' expression [, expression ]* ')' ')'
  | UNNEST '(' expression ')'

values:
  VALUES expression [, expression ]*

groupItem:
  expression
  | '(' ')'
  | '(' expression [, expression ]* ')'
  | CUBE '(' expression [, expression ]* ')'
  | ROLLUP '(' expression [, expression ]* ')'
  | GROUPING SETS '(' groupItem [, groupItem ]* ')'
```

2.2 流作业 SQL 语法概览

本章节介绍了目前DLI所提供的Flink 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 流作业语法概览

语法分类	功能描述
创建输入流	**DIS输入流**
	DMS输入流
创建输入流	**MRS Kafka输入流**
	开源Kafka输入流
	OBS输入流
创建输出流	**CSS Elasticsearch输出流**
	DCS输出流
	DDS输出流
	DIS输出流
	DMS输出流
	DWS输出流（通过JDBC方式）
	DWS输出流（通过OBS转储方式）

版权所有 © 华为技术有限公司

语法分类	功能描述
创建输出流	MRS HBase输出流
	MRS Kafka输出流
	开源Kafka输出流
	OBS输出流
	RDS输出流
创建输出流	SMN输出流
	文件系统输出流(推荐)
创建中间流	创建中间流
创建维表	创建Redis表
	创建RDS表
自拓展生态	自拓展输入流
	自拓展输出流

2.3 创建输入流

2.3.1 DIS 输入流

功能描述

创建source流从数据接入服务（DIS）获取数据。用户数据从DIS接入，Flink作业从DIS的通道读取数据，作为作业的输入数据。Flink作业可通过DIS的source源将数据从生产者快速移出，进行持续处理，适用于将云服务外数据导入云服务后进行过滤、实时分析、监控报告和转储等场景。

数据接入服务（Data Ingestion Service，简称DIS）为处理或分析流数据的自定义应用程序构建数据流管道，主要解决云服务外的数据实时传输到云服务内的问题。数据接入服务每小时可从数十万种数据源（如IoT数据采集、日志和定位追踪事件、网站点击流、社交媒体源等）中连续捕获、传送和存储数TB数据。DIS的更多信息，请参见。

语法格式

```
CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "dis",
    region = "",
    channel = "",
    partition_count = "",
    encode = "",
    field_delimiter = "",
    offset= "");
```


版权所有 © 华为技术有限公司

关键字

表 2-2 关键字说明

参数	是否必选	说明
type	是	数据源类型，“dis”表示数据源为数据接入服务。
region	是	数据所在的DIS区域。
ak	否	访问密钥ID(Access Key ID)。
sk	否	Secret Access Key，与访问密钥ID结合使用的密钥。
channel	是	数据所在的DIS通道名称。
partition_count	否	数据所在的DIS通道分区数。该参数和partition_range参数不能同时配置。当该参数没有配置的时候默认读取所有partition。
partition_range	否	指定作业从DIS通道读取的分区范围。该参数和partition_count参数不能同时配置。当该参数没有配置的时候默认读取所有partition。 partition_range = "[0:2]"时，表示读取的分区范围是1-3，包括分区1、分区2和分区3。
encode	是	数据编码格式，可选为“csv”、“json”、“xml”、“email”、“blob”和“user_defined”。 ● 若编码格式为“csv”，则需配置“field_delimiter”属性。 ● 若编码格式为“json”，则需配置“json_config”属性。 ● 若编码格式为“xml”，则需配置“xml_config”属性。 ● 若编码格式为“email”，则需配置“email_key”属性。 ● 若编码格式为“blob”，表示不对接收的数据进行解析，流属性仅能有一个且数据格式为ARRAY[TINYINT]。 ● 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
field_delimiter	否	属性分隔符，仅当编码格式为csv时该参数需要填写，例如配置为“,”。
quote	否	可以指定数据格式中的引用符号，在两个引用符号之间的属性分隔符会被当做普通字符处理。 ● 当引用符号为双引号时，请设置quote = "\u005c\u0022"进行转义。 ● 当引用符号为单引号时，则设置quote = "'"。 **说明** ● 目前仅适用于CSV格式。 ● 设置引用符号后，必须保证每个字段中包含0个或者偶数个引用符号，否则会解析失败。

版权所有 © 华为技术有限公司

参数	是否必选	说明
json_config	否	当编码格式为json时，用户需要通过该参数来指定json字段和流定义字段的映射关系，格式为“field1=data_json.field1; field2=data_json.field2; field3=$”，其中field3=$表示field3的内容为整个json串。
xml_config	否	当编码格式为xml时，用户需要通过该参数来指定xml字段和流定义字段的映射关系，格式为“field1=data_xml.field1; field2=data_xml.field2”。
email_key	否	当编码格式为email时，用户需要通过该参数来指定需要提取的信息，需要列出信息的key值，需要与流定义字段一一对应，多个key值时以逗号分隔，例如“Message-ID, Date, Subject, body”，其中由于邮件正文没有关键字，DLI规定其关键字为“body”。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现解码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现解码类的入参，仅支持一个string类型的参数。
offset	否	• 当启动作业后再获取数据，则该参数无效。 • 当获取数据后再启动作业，用户可以根据需求设置该参数的数值。 例如当offset= "100"时，则表示DLI从DIS服务中的第100条数据开始处理。
start_time	否	DIS数据读取起始时间。 • 当该参数配置时则从配置的时间开始读取数据，有效格式为yyyy-MM-dd HH:mm:ss。 • 当没有配置start_time也没配置offset的时候，读取最新数据。 • 当没有配置start_time但配置了offset的时候，则从offset开始读取数据。
enable_checkpoint	否	是否启用checkpoint功能，可配置为true（启用）或者false（停用）， 默认为false。
checkpoint_app_name	否	DIS服务的消费者标识，当不同作业消费相同通道时，需要区分不同的消费者标识，以免checkpoint混淆。
checkpoint_interval	否	DIS源算子做checkpoint的时间间隔，单位秒，默认为60。

版权所有 © 华为技术有限公司

注意事项

在创建Source Stream时可以指定时间模型以便在后续计算中使用，当前DLI支持Processing Time和Event Time两种时间模型，具体使用语法可以参考**配置时间模型**。

示例

- CSV编码格式：从DIS通道读取数据，记录为csv编码，并且以逗号为分隔符。

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT,
  car_timestamp LONG
)
  WITH (
    type = "dis",
    region = "xxx",
    channel = "dliinput",
    encode = "csv",
    field_delimiter = ","
);
```

- JSON编码格式：从DIS通道读取数据，记录为json编码。数据示例：{"car": {"car_id":"ZJA710XC", "car_owner":"coco", "car_age":5, "average_speed":80, "total_miles":15000, "car_timestamp":1526438880}}。

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT,
  car_timestamp LONG
)
  WITH (
    type = "dis",
    region = "xxx",
    channel = "dliinput",
    encode = "json",
    json_config = "car_id=car.car_id;car_owner =car.car_owner;car_age=car.car_age;average_speed
=car.average_speed ;total_miles=car.total_miles;"
);
```

- XML编码格式：从DIS通道读取数据，记录为xml编码。

```
CREATE SOURCE STREAM person_infos (
    pid BIGINT,
    pname STRING,
    page int,
    plocation STRING,
    pbir DATE,
    phealthy BOOLEAN,
    pgrade ARRAY[STRING]
)
  WITH (
    type = "dis",
    region = "xxx",
    channel = "dis-dli-input",
    encode = "xml",
    field_delimiter = ",",
    xml_config =
"pid=person.pid;page=person.page;pname=person.pname;plocation=person.plocation;pbir=person.pbir;
pgrade=person.pgrade;phealthy=person.phealthy"
);
```

xml数据示例如下：

```
<?xml version="1.0" encodeing="utf-8"?>
```


版权所有 © 华为技术有限公司

```
<root>
  <person>
    <pid>362305199010025042</pid>
    <pname>xiaoming</pname>
    <page>28</page>
    <plocation>内蒙古,乌兰察布市,集宁区,商都县</plocation>
    <pbir>1990-10-02</pbir>
    <phealthy>true</phealthy>
    <pgrade>[A,B,C]</pgrade>
  </person>
</root>
```

- EMAIL编码格式：从DIS通道读取数据，每条记录为一封完整邮件。

```
CREATE SOURCE STREAM email_infos (
  Event_ID String,
  Event_Time Date,
  Subject String,
  From_Email String,
  To_EMAIL String,
  CC_EMAIL Array[String],
  BCC_EMAIL String,
  MessageBody String,
  Mime_Version String,
  Content_Type String,
  charset String,
  Content_Transfer_Encoding String
)
  WITH (
    type = "dis",
    region = "xxx",
    channel = "dliinput",
    encode = "email",
    email_key = "Message-ID, Date, Subject, From, To, CC, BCC, Body, Mime-Version, Content-Type,
charset, Content_Transfer_Encoding"
);
```

email数据示例如下：

```
Message-ID: <20090629183903250425 4@sample.com>
Date: Fri, 11 May 2001 09:54:00 -0700 (PDT)
From: zhangsan@sample.com
To: lisi@sample.com, wangwu@sample.com
Subject:  "Hello World"
Cc: lilei@sample.com, hanmei@sample.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: jack@sample.com, lily@sample.com
X-From: Zhang San
X-To: Li Si, Wang Wu
X-cc: Li Lei, Han Mei
X-bcc:
X-Folder: \Li_Si_June2001\Notes Folders\Notes inbox
X-Origin: Lucy
X-FileName: sample.nsf

Dear Associate / Analyst Committee:

Hello World!

Thank you,

Associate / Analyst Program
zhangsan
```

2.3.2 DMS 输入流

分布式消息服务（Distributed Message Service，简称DMS）是一项基于高可用分布式集群技术的消息中间件服务，提供了可靠且可扩展的托管消息队列，用于收发消息

 版权所有 © 华为技术有限公司

和存储消息。分布式消息服务Kafka是一款基于开源社区版Kafka提供的消息队列服务，向用户提供可靠的全托管式的Kafka消息队列。

DLI支持创建输入流从DMS的Kafka获取数据，作为作业的输入数据。创建DMS Kafka输入流的语法与创建开源Apache Kafka输入流一样，具体请参见**开源Kafka输入流**。

2.3.3 MRS Kafka 输入流

功能描述

创建source流从Kafka获取数据，作为作业的输入数据。

Apache Kafka是一个快速、可扩展的、高吞吐、可容错的分布式发布订阅消息系统，具有高吞吐量、内置分区、支持数据副本和容错的特性，适合在大规模消息处理场景中使用。MRS基于Apache Kafka在平台部署并托管了Kafka集群。

前提条件

- Kafka是线下集群，需要通过增强型跨源连接功能将Flink作业与Kafka进行对接。且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "kafka",
    kafka_bootstrap_servers = "",
    kafka_group_id = "",
    kafka_topic = "",
    encode = "json"
  );
```

关键字

表 2-3 关键字说明

参数	是否必选	说明
type	是	数据源类型，“Kafka”表示数据源。
kafka_bootstrap_servers	是	Kafka的连接端口，需要确保能连通（需要通过增强型跨源开通DLI队列和Kafka集群的连接）。
kafka_group_id	否	group id。
kafka_topic	是	读取的Kafka的topic。目前只支持读取单个topic。

版权所有 © 华为技术有限公司

参数	是否必选	说明
encode	是	数据编码格式，可选为“csv”、“json”、“blob”和“user_defined”。 ● 若编码格式为“csv”，则需配置“field_delimiter”属性。 ● 若编码格式为“json”，则需配置“json_config”属性。 ● 当编码格式为"blob"时，表示不对接收的数据进行解析，流属性仅能有一个且为Array[TINYINT]类型。 ● 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现解码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现解码类的入参，仅支持一个string类型的参数。
krb_auth	否	创建跨源认证的认证名。开启kerberos认证时，需配置该参数。 **说明** 请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。
json_config	否	当encode为json时，用户可以通过该参数指定json字段和流属性字段的映射关系。 格式："field1=json_field1;field2=json_field2" 格式说明：field1、field2为创建的表字段名称。json_field1、json_field2为kafka输入数据json串的key字段名称。 具体使用方法可以参考**示例**说明。
field_delimiter	否	当encode为csv时，用于指定csv字段分隔符，默认为逗号。
quote	否	可以指定数据格式中的引用符号，在两个引用符号之间的属性分隔符会被当做普通字符处理。 ● 当引用符号为双引号时，请设置quote = "\u005c\u0022"进行转义。 ● 当引用符号为单引号时，则设置quote = "'"。 **说明** ● 目前仅适用于CSV格式。 ● 设置引用符号后，必须保证每个字段中包含0个或者偶数个引用符号，否则会解析失败。

版权所有 © 华为技术有限公司

参数	是否必选	说明
start_time	否	kafka数据读取起始时间。 当该参数配置时则从配置的时间开始读取数据，有效格式为yyyy-MM-dd HH:mm:ss。start_time要不大于当前时间，若大于当前时间，则不会有数据读取出。
kafka_properties	否	可通过该参数配置kafka的原生属性，格式为"key1=value1;key2=value2"
kafka_certificate_name	否	跨源认证信息名称。跨源认证信息类型为“Kafka_SSL”时，该参数有效。 **说明** • 指定该配置项时，服务仅加载该认证下指定的文件和密码，系统将自动设置到“kafka_properties”属性中。 • Kafka SSL认证需要的其他配置信息，需要用户手动在“kafka_properties”属性中配置。

注意事项

在创建Source Stream时可以指定时间模型以便在后续计算中使用，当前DLI支持Processing Time和Event Time两种时间模型，具体使用语法可以参考**配置时间模型**。

示例

- 从Kafka名称为test的topic中读取数据。

```
CREATE SOURCE STREAM kafka_source (
  name STRING,
  age int
 )
  WITH (
    type = "kafka",
    kafka_bootstrap_servers = "ip1:port1,ip2:port2",
    kafka_group_id = "sourcegroup1",
    kafka_topic = "test",
    encode = "json"
);
```

- 从Kafka读取对象为test的topic，使用json_config将json数据和表字段对应。

  数据编码格式为json且不含嵌套，例如：

```
{"attr1": "lilei", "attr2": 18}
```

  建表语句参考如下：

```
CREATE SOURCE STREAM kafka_source (name STRING, age int)
WITH (
  type = "kafka",
  kafka_bootstrap_servers = "ip1:port1,ip2:port2",
  kafka_group_id = "sourcegroup1",
  kafka_topic = "test",
  encode = "json",
  json_config = "name=attr1;age=attr2"
);
```


版权所有 © 华为技术有限公司

2.3.4 开源 Kafka 输入流

功能描述

创建source流从Kafka获取数据，作为作业的输入数据。

Apache Kafka是一个快速、可扩展的、高吞吐、可容错的分布式发布订阅消息系统，具有高吞吐量、内置分区、支持数据副本和容错的特性，适合在大规模消息处理场景中使用。

前提条件

- Kafka是线下集群，需要通过增强型跨源连接功能将Flink作业与Kafka进行对接。且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "kafka",
    kafka_bootstrap_servers = "",
    kafka_group_id = "",
    kafka_topic = "",
    encode = "json",
    json_config=""
  );
```

关键字

表 2-4 关键字说明

参数	是否必选	说明
type	是	数据源类型，“Kafka”表示数据源。
kafka_bootstrap_servers	是	Kafka的连接端口，需要确保能连通（需要通过增强型跨源开通DLI队列和Kafka集群的连接）。
kafka_group_id	否	group id。
kafka_topic	是	读取的Kafka的topic。目前只支持读取单个topic。

 版权所有 © 华为技术有限公司

参数	是否必选	说明
encode	是	数据编码格式，可选为“csv”、“json”、“blob”和“user_defined”。 • 若编码格式为“csv”，则需配置“field_delimiter”属性。 • 若编码格式为“json”，则需配置“json_config”属性。 • 当编码格式为"blob"时，表示不对接收的数据进行解析，当前表仅能有一个且为Array[TINYINT]类型的表字段。 • 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现解码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现解码类的入参，仅支持一个string类型的参数。
json_config	否	当encode为json时，用户可以通过该参数指定json字段和流属性字段的映射关系。 格式："field1=json_field1;field2=json_field2" 格式说明：field1、field2为创建的表字段名称。json_field1、json_field2为kafka输入数据json串的key字段名称。 具体使用方法可以参考**示例**说明。 **说明** 如果定义的source stream中的属性和json中的属性名称相同，json_configs可以不用配置。
field_delimiter	否	当encode为csv时，用于指定csv字段分隔符，默认为逗号。
quote	否	可以指定数据格式中的引用符号，在两个引用符号之间的属性分隔符会被当做普通字符处理。 • 当引用符号为双引号时，请设置quote = "\u005c\u0022"进行转义。 • 当引用符号为单引号时，则设置quote = "'"。 **说明** • 目前仅适用于CSV格式。 • 设置引用符号后，必须保证每个字段中包含0个或者偶数个引用符号，否则会解析失败。

版权所有 © 华为技术有限公司

参数	是否必选	说明
start_time	否	kafka数据读取起始时间。 当该参数配置时则从配置的时间开始读取数据，有效格式为yyyy-MM-dd HH:mm:ss。start_time要不大于当前时间，若大于当前时间，则不会有数据读取出。 该参数配置后，只会读取Kafka topic在该时间点后产生的数据。
kafka_properties	否	可通过该参数配置kafka的原生属性，格式为"key1=value1;key2=value2"。具体的属性值可以参考Apache Kafka中的描述。
kafka_certificate_name	否	跨源认证信息名称。跨源认证信息类型为“Kafka_SSL”时，该参数有效。 **说明** • 指定该配置项时，服务仅加载该认证下指定的文件和密码，系统将自动设置到“kafka_properties”属性中。 • Kafka SSL认证需要的其他配置信息，需要用户手动在“kafka_properties”属性中配置。

注意事项

在创建Source Stream时可以指定时间模型以便在后续计算中使用，当前DLI支持Processing Time和Event Time两种时间模型，具体使用语法可以参考**配置时间模型**。

示例

- 从Kafka读取对象为test的topic。数据编码格式为json且不含嵌套，例如：{"attr1": "lilei", "attr2": 18}。

```
CREATE SOURCE STREAM kafka_source (name STRING, age int)
WITH (
  type = "kafka",
  kafka_bootstrap_servers = "ip1:port1,ip2:port2",
  kafka_group_id = "sourcegroup1",
  kafka_topic = "test",
  encode = "json",
  json_config = "name=attr1;age=attr2"
);
```

- 从Kafka读取对象为test的topic。数据编码格式为json且包含嵌套。本示例使用了复杂数据类型ROW，ROW使用语法可以参考**数据类型**。

  测试数据参考如下：

```
{
    "id":"1",
    "type2":"online",
    "data":{
        "patient_id":1234,
        "name":"bob1234"
    }
}
```

  则对应建表语句示例为：

```
CREATE SOURCE STREAM kafka_source
(
  id STRING,
```


版权所有 © 华为技术有限公司

```
  type2 STRING,
  data ROW<
    patient_id STRING,
    name STRING>
)
WITH (
  type = "kafka",
  kafka_bootstrap_servers = "ip1:port1,ip2:port2",
  kafka_group_id = "sourcegroup1",
  kafka_topic = "test",
  encode = "json"
);

CREATE SINK STREAM kafka_sink
(
  id STRING,
  type2 STRING,
  patient_id STRING,
  name STRING
)
  WITH (
    type="kafka",
    kafka_bootstrap_servers =  "ip1:port1,ip2:port2",
    kafka_topic = "testsink",
    encode = "csv"
  );

INSERT INTO kafka_sink select id, type2, data.patient_id, data.name from kafka_source;
```

2.3.5 OBS 输入流

功能描述

创建source流从对象存储服务（OBS）获取数据。DLI从OBS上读取用户存储的数据，作为作业的输入数据。适用于大数据分析、原生云应用程序数据、静态网站托管、备份/活跃归档、深度/冷归档等场景。

对象存储服务（Object Storage Service，简称OBS）是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。OBS的更多信息，请参见。

语法格式

```
CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "obs",
    region = "",
    bucket = "",
    object_name = "",
    row_delimiter = "\n",
    field_delimiter = '',
    version_id = ""
  );
```

关键字

表 2-5 关键字说明

参数	是否必选	说明
type	是	数据源类型，“obs”表示数据源为对象存储服务。

版权所有 © 华为技术有限公司

参数	是否必选	说明
region	是	对象存储服务所在区域。
encode	否	数据的编码格式，可以为“csv”或者“json”。默认值为“csv”。
ak	否	访问密钥ID(Access Key ID)。
sk	否	Secret Access Key，与访问密钥ID结合使用的密钥。
bucket	是	数据所在的OBS桶名。
object_name	是	数据所在OBS桶中的对象名。如果对象不在OBS根目录下，则需添加文件夹名，例如：test/test.csv。对象文件格式参考“encode”参数。
row_delimiter	是	行间的分隔符。
field_delimiter	否	属性分隔符。 • 当“encode”参数为csv时，该参数必选。用户可以自定义属性分隔符。 • 当“encode”参数为json时，该参数不需要填写。
quote	否	可以指定数据格式中的引用符号，在两个引用符号之间的属性分隔符会被当做普通字符处理。 • 当引用符号为双引号时，请设置quote = "\u005c\u0022"进行转义。 • 当引用符号为单引号时，则设置quote = "'"。 **说明** • 目前只适用于CSV格式。 • 设置引用符号后，必须保证每个字段中包含0个或者偶数个引用符号，否则会解析失败。
version_id	否	版本号，当obs里的桶或对象有设置版本的时候需填写，否则不用配置该项。

注意事项

在创建Source Stream时可以指定时间模型以便在后续计算中使用，当前DLI支持Processing Time和Event Time两种时间模型，具体使用语法可以参考**配置时间模型**。

示例

- 从OBS的桶读取对象为input.csv的文件，文件以'\n'划行， 以','划列。

  测试输入数据input.csv可以先通过新建input.txt复制如下文本数据，再另存为input.csv格式文件。将input.csv上传到对应OBS桶目录下。例如，当前上传到：“dli-test-obs01”桶目录下。

  ```
  1,2,3,4,1403149534
  5,6,7,8,1403149535
  ```


版权所有 © 华为技术有限公司

创建表参考如下：

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
  WITH (
    type = "obs",
    bucket = "dli-test-obs01",
    region = "xxx",
    object_name = "input.csv",
    row_delimiter = "\n",
    field_delimiter = ","
);
```

- 从OBS的桶读取对象为input.json的文件，文件以'\n'划行。

```
CREATE SOURCE STREAM obs_source (
  str STRING
)
  WITH (
    type = "obs",
    bucket = "obssource",
    region = "xxx",
    encode = "json",
    row_delimiter = "\n",
    object_name = "input.json"
);
```

2.4 创建输出流

2.4.1 MRS OpenTSDB 输出流

功能描述

DLI将Flink作业的输出数据输出到MRS的OpenTSDB中。

前提条件

- 确保MRS的集群已经安装了OpenTSDB。
- 该场景作业需要运行在DLI的独享队列上，因此要与MRS集群建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "opentsdb",
    region = "",
    tsdb_metrics = "",
    tsdb_timestamps = "",
    tsdb_values = "",
    tsdb_tags = "",
    batch_insert_data_num = ""
  )
```


版权所有 © 华为技术有限公司

关键字

表 2-6 关键字说明

参数	是否必选	说明
type	是	输出通道类型，“opentsdb”表示输出到MRS的OpenTSDB。
region	是	MRS服务所在区域。
tsdb_link_address	是	MRS中OpenTSDB实例的服务地址，格式为http://ip:port或者https://ip:port。 **说明** 配置项tsd.https.enabled为true时，需要使用https，注意https暂时不支持证书认证。
tsdb_metrics	是	数据点的metric，支持参数化。
tsdb_timestamps	是	数据点的timestamp，数据类型支持LONG、INT、SHORT和STRING，仅支持指定动态列。
tsdb_values	是	数据点的value，数据类型支持SHORT、INT、LONG、FLOAT、DOUBLE和STRING，支持指定动态列或者常数值。
tsdb_tags	是	数据点的tags，每个tags里面至少一个标签值，最多8个标签值，支持参数化。
batch_insert_data_num	否	表示一次性批量写入的数据量（即数据条数），值必须为正整数，上限为65536，默认值为8。

注意事项

当配置项支持参数化时，表示将记录中的一列或者多列作为该配置项的一部分。例如当配置项设置为car_${car_brand}时，如果一条记录的car_brand列值为BMW，则该配置项在该条记录下为car_BMW。

示例

将流weather_out的数据输出到MRS服务的OpenTSDB中。

```
CREATE SINK STREAM weather_out (
  timestamp_value LONG, /* 时间 */
  temperature FLOAT, /* 温度值 */
  humidity FLOAT, /* 湿度值 */
  location STRING /* 地点 */
)
  WITH (
    type = "opentsdb",
    region = "xxx",
    tsdb_link_address = "https://x.x.x.x:4242",
    tsdb_metrics = "weather",
    tsdb_timestamps = "${timestamp_value}",
    tsdb_values = "${temperature}; ${humidity}",
    tsdb_tags = "location:${location},signify:temperature; location:${location},signify:humidity",
```


版权所有 © 华为技术有限公司

```
  batch_insert_data_num = "10"
);
```

2.4.2 CSS Elasticsearch 输出流

功能描述

DLI将Flink作业的输出数据输出到云搜索服务CSS的Elasticsearch中。Elasticsearch是基于Lucene的当前流行的企业级搜索服务器，具备分布式多用户的能力。其主要功能包括全文检索、结构化搜索、分析、聚合、高亮显示等。能为用户提供实时搜索、稳定可靠的服务。适用于日志分析、站内搜索等场景。

云搜索服务（Cloud Search Service，简称CSS）为DLI提供托管的分布式搜索引擎服务，完全兼容开源Elasticsearch搜索引擎，支持结构化、非结构化文本的多条件检索、统计、报表。

说明

创建CSS集群时如果开启了安全模式，后续将无法关闭。

前提条件

- 该场景作业需要运行在DLI的独享队列上，因此要与云搜索服务建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "es",
    region = "",
    cluster_address = "",
    es_index = "",
    es_type= "",
    es_fields= "",
    batch_insert_data_num= ""
  );
```

关键字

表 2-7 关键字说明

参数	是否必选	说明
type	是	输出通道类型，es表示输出到云搜索服务中。
region	是	数据所在的云搜索服务所在区域。
cluster_address	是	云搜索服务集群的内网访问地址，例如：x.x.x.x:x，多个地址时以逗号分隔。
es_index	是	待插入数据的索引，支持参数化。对应CSS服务中的index。

版权所有 © 华为技术有限公司

参数	是否必选	说明
es_type	是	待插入数据的文档类型，支持参数化。对应CSS服务中的type。 若使用的es版本为6.x，则该值不能以"_"开头。 若使用的es版本为7.x，如果提前预置CSS服务中的type，则es_type需为"_doc"，否则可为符合CSS规范的值。
es_fields	是	待插入数据字段的key，具体形式如："id,f1,f2,f3,f4"，并且保证与sink中数据列一一对应；如果不使用key，而是采用随机的属性字段，则无需使用id关键字，具体形式如："f1,f2,f3,f4,f5"。对应CSS服务中的filed。
batch_insert_data_num	是	表示一次性批量写入的数据量，值必须为正整数，单位为：条。上限为65536，默认值为10。
action	否	当值为add时，表示遇到相同id时，数据被强制覆盖，当值为upsert时，表示遇到相同id时，更新数据（选择upsert时，es_fileds字段中必须指定id），默认值为add。
enable_output_null	否	使用该参数来配置是否输出空字段。当该参数为true表示输出空字段（值为null），若为false表示不输出空字段。默认为false。
max_record_num_cache	否	记录最大缓存数。
es_certificate_name	否	跨源认证信息名称。 若es集群开启安全模式且开启https，则使用证书进行访问，创建的跨源认证类型需要为“CSS”。 若es集群开启安全模式，但关闭https，则使用证书和账号密码进行访问，创建的跨源认证类型需要为"Password"。

注意事项

当配置项支持参数化时，表示将记录中的一列或者多列作为该配置项的一部分。例如当配置项设置为car_${car_brand}时，如果一条记录的car_brand列值为BMW，则该配置项在该条记录下为car_BMW。

示例

将流qualified_cars的数据输出到云搜索服务的集群。

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
  WITH (
    type = "es",
```


版权所有 © 华为技术有限公司

```
  region = "xxx",
  cluster_address = "192.168.0.212:9200",
  es_index = "car",
  es_type = "information",
  es_fields = "id,owner,age,speed,miles",
  batch_insert_data_num = "10"
);
```

2.4.3 DCS 输出流

功能描述

DLI将Flink作业的输出数据输出到分布式缓存服务（DCS）的Redis中。Redis是一种支持Key-Value等多种数据结构的存储系统。可用于缓存、事件发布或订阅、高速队列等场景，提供字符串、哈希、列表、队列、集合结构直接存取，基于内存，可持久化。有关Redis的详细信息，请访问Redis官方网站**https://redis.io/**。

分布式缓存服务（DCS）为DLI提供兼容Redis的即开即用、安全可靠、弹性扩容、便捷管理的在线分布式缓存能力，满足用户高并发及快速数据访问的业务诉求。

前提条件

- 请务必确保您的账户下已在分布式缓存服务（DCS）里创建了Redis类型的缓存实例。
- 该场景作业需要运行在DLI的独享队列上，因此要与DCS集群建立跨源连接，且用户可以根据实际所需设置相应安全组规则。
- 用户通过VPC对等访问DCS实例时，除了满足VPC对等网跨VPC访问的约束之外，还存在如下约束：
  - 当创建DCS实例时使用了172.16.0.0/12~24网段时，DLI队列不能在192.168.1.0/24、192.168.2.0/24、192.168.3.0/24网段。
  - 当创建DCS实例时使用了192.168.0.0/16~24网段时，DLI队列不能在172.31.1.0/24、172.31.2.0/24、172.31.3.0/24网段。
  - 当创建DCS实例时使用了10.0.0.0/8~24网段时，DLI队列不能在172.31.1.0/24、172.31.2.0/24、172.31.3.0/24网段。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
 WITH (
  type = "dcs_redis",
  region = "",
  cluster_address = "",
  password = "",
  value_type= "",key_value= ""
 );
```

 版权所有 © 华为技术有限公司

关键字

表 2-8 关键字说明

参数	是否必选	说明
type	是	输出通道类型，dcs_redis表示输出到分布式缓存服务的Redis存储系统中。
region	是	数据所在的DCS所在区域。
cluster_address	是	Redis实例连接地址。
password	否	Redis实例连接密码，当设置为免密访问时，省略该配置项。
value_type	是	该参数可配置为如下选项或选项的组合： ● 支持指定插入数据类型，包括：string, list, hash, set, zset； ● 支持设置key的过期时间，包括expire, pexpire, expireAt, pexpireAt； ● 支持删除key命令，包括del, hdel； 当需要使用多个命令时，用“;”分隔。
key_value	是	设置具体的key和value，key_value对必须与value_type所指定的类型数相对应，用“;”分隔，且key和value均支持参数化，动态列名采用${列名}表示。

注意事项

- 当配置项支持参数化时，表示将记录中的一列或者多列作为该配置项的一部分。例如当配置项设置为car_${car_brand}时，如果一条记录的car_brand列值为BMW，则该配置项在该条记录下为car_BMW。
- 字符":", ",", ";", "$", "{", "}"已被征用为特殊分隔符，暂时没有提供转义功能，禁止在key和value中作为普通字符使用，否则会影响解析，导致程序异常。

示例

将流qualified_cars的数据输出到DCS服务的Redis类型的缓存实例中。

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed DOUBLE,
  total_miles DOUBLE
)
  WITH (
    type = "dcs_redis",
    cluster_address = "192.168.0.34:6379",
    password = "xxxxxxxx",
    value_type = "string; list; hash; set; zset",
    key_value = "${car_id}_str: ${car_owner}; name_list: ${car_owner}; ${car_id}_hash: {name:${car_owner},
age: ${car_age}}; name_set:   ${car_owner}; math_zset: {${car_owner}:${average_speed}}"
  );
```


版权所有 © 华为技术有限公司

2.4.4 DDS 输出流

功能描述

DLI将作业的输出数据输出到文档数据库服务（DDS）中。

文档数据库服务（Document Database Service，简称DDS）完全兼容MongoDB协议，提供安全、高可用、高可靠、弹性伸缩和易用的数据库服务，同时提供一键部署、弹性扩容、容灾、备份、恢复、监控和告警等功能。

前提条件

- 请务必确保您的账户下已在文档数据库服务（DDS）里创建了DDS实例。

  如何创建DDS实例，请参考中“快速购买文档数据库实例”章节。
- 目前仅支持未开启SSL认证的集群实例，不支持副本集与单节点的类型实例。
- 该场景作业需要运行在DLI的独享队列上，请确保已创建DLI独享队列。
- 确保DLI独享队列与DDS集群建立跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "dds",
    username = "",
    password = "",
    db_url = "",
    field_names = ""
  );
```

关键字

表 2-9 关键字说明

参数	是否必选	说明
type	是	输出通道类型，dds表示输出到文档数据库服务中。
username	是	数据库连接用户名。
password	是	数据库连接密码。
db_url	是	DDS实例的访问地址，形如：ip1:port,ip2:port/database/collection。
field_names	是	待插入数据字段的key，具体形式如："f1,f2,f3"，并且保证与sink中数据列一一对应。
batch_insert_data_num	否	表示一次性批量写入的数据量，值必须为正整数，默认值为10。

版权所有 © 华为技术有限公司

示例

将流qualified_cars 的数据输出到文档数据库collectionTest。

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
  WITH (
    type = "dds",
    region = "xxx",
    db_url = "192.168.0.8:8635,192.168.0.130:8635/dbtest/collectionTest",
    username = "xxxxxxxxxx",
    password =  "xxxxxxxxxx",
    field_names = "car_id,car_owner,car_age,average_speed,total_miles",
    batch_insert_data_num = "10"
  );
```

2.4.5 DIS 输出流

功能描述

DLI将Flink作业的输出数据写入数据接入服务（DIS）中。适用于将数据过滤后导入DIS通道，进行后续处理的场景。

数据接入服务（Data Ingestion Service，简称DIS）为处理或分析流数据的自定义应用程序构建数据流管道，主要解决云服务外的数据实时传输到云服务内的问题。数据接入服务每小时可从数十万种数据源（如IoT数据采集、日志和定位追踪事件、网站点击流、社交媒体源等）中连续捕获、传送和存储数TB数据。DIS的更多信息，请参见。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "dis",
    region = "",
    channel = "",
    partition_key = "",
    encode= "",
    field_delimiter= ""
  );
```

关键字

表 2-10 关键字说明

参数	是否必选	说明
type	是	输出通道类型，dis表示输出到数据接入服务。
region	是	数据所在的DIS所在区域。
ak	否	访问密钥ID(Access Key ID)。
sk	否	Secret Access Key，与访问密钥ID结合使用的密钥。

版权所有 © 华为技术有限公司

参数	是否必选	说明
channel	是	DIS通道。
partition_key	否	数据输出分组主键，多个主键用逗号分隔。当该参数没有配置的时候则随机派发。
encode	是	数据编码格式，可选为“csv”、“json”和“user_defined”。 **说明** • 若编码格式为“csv”，则需配置“field_delimiter”属性。 • 若编码格式为“json”，则需使用“enable_output_null”来配置是否输出空字段，具体见示例。 • 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
field_delimiter	是	属性分隔符。 • 当编码格式为csv时，需要设置属性分隔符，用户可以自定义，如：“，”。 • 当编码格式为json时，则不需要设置属性之间的分隔符。
json_config	否	当编码格式为json时，用户可以通过该参数来指定json字段和流定义字段的映射关系，格式为“field1=data_json.field1;field2=data_json.field2”。
enable_output_null	否	当编码格式为json时，需使用该参数来配置是否输出空字段。 当该参数为“true”表示输出空字段（值为null），若为“false”表示不输出空字段。默认值为“true”。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现编码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现编码类的入参，仅支持一个string类型的参数。

注意事项

无。

版权所有 © 华为技术有限公司

示例

- CSV编码格式：数据输出到DIS通道，使用csv编码，并且以逗号为分隔符，多个分区用car_owner做为key进行分发。数据输出示例："ZJA710XC", "lilei", "BMW", 700000。

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "dis",
    region = "xxx",
    channel = "dlioutput",
    encode = "csv",
    field_delimiter = ","
);
```

- JSON编码格式：数据输出到DIS通道，使用json编码，多个分区用car_owner，car_brand 做为key进行分发,“enableOutputNull”为“true”表示输出空字段（值为null），若为“false”表示不输出空字段。数据示例："car_id ":"ZJA710XC", "car_owner ":"lilei", "car_brand ":"BMW", "car_price ":700000。

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "dis",
    channel = "dlioutput",
    region = "xxx",
    partition_key = "car_owner,car_brand",
    encode = "json",
    enable_output_null = "false"
);
```

2.4.6 DMS 输出流

分布式消息服务（Distributed Message Service，简称DMS）是一项基于高可用分布式集群技术的消息中间件服务，提供了可靠且可扩展的托管消息队列，用于收发消息和存储消息。分布式消息服务Kafka是一款基于开源社区版Kafka提供的消息队列服务，向用户提供可靠的全托管式的Kafka消息队列。

DLI支持将作业的输出数据输出到DMS的Kafka实例中。创建DMS Kafka输出流的语法与创建开源Apache Kafka输出流一样，具体请参见**MRS Kafka输出流**。

2.4.7 DWS 输出流（通过 JDBC 方式）

功能描述

DLI将Flink作业的输出数据输出到数据仓库服务（DWS）中。DWS数据库内核兼容PostgreSQL，PostgreSQL数据库可存储更加复杂类型的数据，支持空间信息服务、多版本并发控制（MVCC）、高并发，适用场景包括位置应用、金融保险、互联网电商等。

数据仓库服务（Data Warehouse Service，简称DWS）是一种基于基础架构和平台的在线数据处理数据库，为用户提供海量数据挖掘和分析服务。DWS的更多信息，请参见。

 版权所有 © 华为技术有限公司

前提条件

- 请务必确保您的账户下已在数据仓库服务（DWS）里创建了DWS集群。
  如何创建DWS集群，请参考《数据仓库服务管理指南》中“创建集群”章节。
- 请确保已创建DWS数据库表。
- 该场景作业需要运行在DLI的独享队列上，因此要与DWS集群建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "rds",
    username = "",
    password = "",
    db_url = "",
    table_name = ""
  );
```

关键字

表 2-11 关键字说明

参数	是否必选	说明
type	是	输出通道类型，rds表示输出到关系型数据库或者数据仓库服务中。
username	是	数据库连接用户名。
password	是	数据库连接密码。
db_url	是	数据库连接地址格式为：postgresql://ip:port/database。
table_name	是	要插入数据的数据库表名。数据库表需事先创建好。

版权所有 © 华为技术有限公司

参数	是否必选	说明

db_columns	否	支持配置输出流属性和数据库表属性的对应关系，需严格按照输出流的属性顺序配置。
		示例：
		`create sink stream a3(student_name  string, student_age int)`
		`  with (`
		`    type = "rds",`
		`    username = "root",`
		`    password = "xxxxxxxx",`
		`    db_url = "postgresql://192.168.0.102:8000/test1",`
		`    db_columns = "name,age",`
		`    table_name = "t1"`
		`  );`
		student_name对应数据库里的name属性，student_age对应数据库里的age属性。
		说明
		• 当不配置db_columns时，若输出流属性个数小于数据库表属性个数，并且数据库多出的属性都是nullable或者有默认值时，这种情况也允许。

primary_key	否	如果想通过主键实时更新表中的数据，需要在创建数据表的时候增加primary_key配置项，如下面例子中的c_timeminute。配置primary_key后，在进行数据写入操作时，如果primary_key存在，则进行更新操作，否则进行插入操作。
		示例：
		`CREATE SINK STREAM test(c_timeminute LONG, c_cnt LONG)`
		`  WITH (`
		`    type = "rds",`
		`    username = "root",`
		`    password = "xxxxxxxx",`
		`    db_url = "postgresql://192.168.0.12:8000/test",`
		`    table_name = "test",`
		`    primary_key = "c_timeminute"`
		`  );`

注意事项

stream_id所定义的流格式需和数据库中的表格式一致。

示例

将流audi_cheaper_than_30w的数据输出到数据库test的audi_cheaper_than_30w表下。

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "rds",
    username = "root",
    password = "xxxxxx",
    db_url = "postgresql://192.168.1.1:8000/test",
    table_name = "audi_cheaper_than_30w"
```


版权所有 © 华为技术有限公司

```
);

insert into audi_cheaper_than_30w select "1","2","3",4;
```

2.4.8 DWS 输出流（通过 OBS 转储方式）

功能描述

创建sink流将Flink作业数据通过OBS转储方式输出到数据仓库服务(DWS)，即Flink作业数据先输出到OBS，然后再从OBS导入到DWS。如何导入OBS数据到DWS具体可参考中“从OBS并行导入数据到集群”章节。

数据仓库服务（Data Warehouse Service，简称DWS）是一种基于基础架构和平台的在线数据处理数据库，为用户提供海量数据挖掘和分析服务。DWS的更多信息，请参见。

注意事项

- 通过OBS转储支持两种中间文件方式：
  - ORC： ORC格式不支持Array数据类型，如果使用ORC格式，需先在DWS中创建外部服务器，具体可参考中“创建外部服务器”章节。
  - CSV： CSV格式默认记录分隔符为换行符，若属性内容中有换行符，建议配置quote，具体参见**表2-12**。
- 如果要写入的表不存在，则会自动创建表。由于DLI SQL类型不支持text，如果存在长文本，建议先在数据库中创建表。
- encode使用orc格式时，创建DWS表时，如果SQL流字段属性定义为String类型，DWS表字段属性不能使用varchar类型，需使用特定的text类型；如果是SQL流字段属性定义为Integer类型，DWS表字段需要使用Integer类型。

前提条件

- 确保已创建OBS桶和文件夹。
- 该场景作业需要运行在DLI的独享队列上，因此要与DWS集群建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
        type = "dws",
        region = "",
        ak = "",
        sk = "",
        encode = "",
        field_delimiter = "",
        quote = "",
        db_obs_server = "",
        obs_dir = "",
        username = "",
        password =  "",
        db_url = "",
        table_name = "",
        max_record_num_per_file = "",
        dump_interval = ""
  );
```

 版权所有 © 华为技术有限公司

关键字

表 2-12 关键字说明

参数	是否必选	说明
type	是	输出通道类型，dws表示输出到数据仓库服务中。
region	是	数据仓库服务所在区域。
ak	是	访问密钥ID(Access Key ID)。
sk	是	Secret Access Key，与访问密钥ID结合使用的密钥。
encode	是	编码方式。当前支持csv和orc两种方式。
field_delimiter	否	属性分隔符。当编码方式为csv时需要配置，建议尽量用不可见字符作为分隔符，如\u0006\u0002。
quote	否	单字节，建议使用不可见字符，如\u0007。
db_obs_server	否	已在数据库中创建的外部服务器，如obs_server。 如果编码方式为orc格式时需指定该参数。
obs_dir	是	中间文件存储目录。格式为{桶名}/{目录名}， 如obs-a1/dir1/subdir。
username	是	数据库连接用户名。
password	是	数据库连接密码。
db_url	是	数据库连接地址。格式为/ip:port/database，如“192.168.1.21:8000/test1”。
table_name	是	数据表名，若表不存在，则自动创建。
max_record_num_per_file	是	每个文件最多存储多少条记录。当文件记录数少于最大值时，该文件会延迟一个转储周期输出。
dump_interval	是	转储周期，单位为秒。
delete_obs_temp_file	否	是否要删除obs上的临时文件，默认为“true”，若设置为“false”，则不会删除obs上的文件，需用户自己清理。
max_dump_file_num	否	执行一次转储操作时最多转储多少文件。 当本次转储操作发现文件数小于最大值，则会延迟一个转储周期输出。

示例

- CSV格式转储。

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
```


版权所有 © 华为技术有限公司

```
 WITH (
   type = "dws",
   region = "xxx",
   ak = "",
   sk = "",
   encode = "csv",
   field_delimiter = "\u0006\u0006\u0002",
   quote = "\u0007",
   obs_dir = "dli-append-2/dws",
   username = "",
   password =  "",
   db_url = "192.168.1.12:8000/test1",
   table_name = "table1",
   max_record_num_per_file = "100",
   dump_interval = "10"
 );
```

- ORC格式转储。

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
  WITH (
    type = "dws",
    region = "xxx",
    ak = "",
    sk = "",
    encode = "orc",
    db_obs_server = "obs_server",
    obs_dir = "dli-append-2/dws",
    username = "",
    password =  "",
    db_url = "192.168.1.12:8000/test1",
    table_name = "table1",
    max_record_num_per_file = "100",
    dump_interval = "10"
  );
```

2.4.9 MRS HBase 输出流

功能描述

DLI将Flink作业的输出数据输出到MRS的HBase中。

前提条件

- 确保您的账户下已在MapReduce服务（MRS）里创建了您配置的集群。DLI支持与开启kerberos的hbase集群对接。
- 该场景作业需要运行在DLI的独享队列上，请确保已创建DLI独享队列。
- 确保DLI独享队列与MRS集群建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。
- **若使用MRS HBase，请在增强型跨源的主机信息中添加MRS集群所有节点的主机ip信息。**

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "mrs_hbase",
    region = "",
```

 版权所有 © 华为技术有限公司

```
  cluster_address = "",
  table_name = "",
  table_columns = "",
  illegal_data_table = "",
  batch_insert_data_num = "",
  action = ""
)
```

关键字

表 2-13 关键字说明

参数	是否必选	说明
type	是	输出通道类型，"mrs_hbase"表示输出到MRS的HBase中。
region	是	MRS服务所在区域。
cluster_address	是	待插入数据表所属集群zookeeper地址，形如：ip1,ip2:port。
table_name	是	待插入数据的表名。 支持参数化，例如当需要某一列或者几列作为表名的一部分时，可表示为”car_pass_inspect_with_age_${car_age}“，其中car_age为列名。
table_columns	是	待插入的列，具体形式如："rowKey,f1:c1,f1:c2,f2:c1"，其中必须指定rowKey，当某列不需要加入数据库时，以第三列为例，可表示为"rowKey,f1:c1,,f2:c1"。
illegal_data_table	否	如果指定该参数，异常数据（比如：rowKey不存在）会写入该表（rowKey为taskNo加下划线加时间戳加六位随机数字，schema为info:data, info:reason），否则会丢弃。
batch_insert_data_num	否	表示一次性批量写入的数据条数，值必须为正整数，上限为1000，默认值为10。
action	否	表示数据是插入还是删除，可选值为add和delete，默认值为add。
krb_auth	否	创建跨源认证的认证名。开启kerberos认证时，需配置该参数，填写对应的跨源认证名称。 **说明** 请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。

注意事项

无。

版权所有 © 华为技术有限公司

示例

将数据输出到MRS的HBase中。

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
  WITH (
    type = "mrs_hbase",
    region = "xxx",
    cluster_address = "192.16.0.88,192.87.3.88:2181",
    table_name = "car_pass_inspect_with_age_${car_age}",
    table_columns = "rowKey,info:owner,,car:speed,car:miles",
    illegal_data_table = "illegal_data",
    batch_insert_data_num = "20",
    action = "add",
    krb_auth = "KRB_AUTH_NAME"
  );
```

2.4.10 MRS Kafka 输出流

功能描述

DLI将Flink作业的输出数据输出到Kafka中。

Apache Kafka是一个快速、可扩展的、高吞吐、可容错的分布式发布订阅消息系统，具有高吞吐量、内置分区、支持数据副本和容错的特性，适合在大规模消息处理场景中使用。MRS基于Apache Kafka在平台部署并托管了Kafka集群。

前提条件

- Kafka是线下集群，需要通过增强型跨源连接功能将Flink作业与Kafka进行对接。且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH(
    type = "kafka",
    kafka_bootstrap_servers = "",
    kafka_topic = "",
    encode = "json"
)
```

关键字

表 2-14 关键字说明

参数	是否必选	说明
type	是	输出通道类型，"kafka"表示输出到Kafka中。
kafka_bootstrap_servers	是	Kafka的连接端口，需要确保能连通（需要通过增强型跨源开通DLI队列和Kafka集群的连接）。

版权所有 © 华为技术有限公司

参数	是否必选	说明
kafka_topic	是	写入的topic。
encode	是	编码格式，当前支持“json”和“user_defined”。 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现编码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现编码类的入参，仅支持一个string类型的参数。
krb_auth	否	创建跨源认证的认证名。开启kerberos认证时，需配置该参数。如果创建的MRS集群未开启kerb认证的集群，请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。
kafka_properties	否	可通过该参数配置kafka的原生属性，格式为"key1=value1;key2=value2"
kafka_certificate_name	否	跨源认证信息名称。跨源认证信息类型为“Kafka_SSL”时，该参数有效。 **说明** • 指定该配置项时，服务仅加载该认证下指定的文件和密码，系统将自动设置到“kafka_properties”属性中。 • Kafka SSL认证需要的其他配置信息，需要用户手动在“kafka_properties”属性中配置。

注意事项

无。

示例

将数据输出到Kafka中。

- 示例一

```
CREATE SINK STREAM kafka_sink (name STRING)
  WITH (
    type="kafka",
    kafka_bootstrap_servers =  "ip1:port1,ip2:port2",
    kafka_topic = "testsink",
    encode = "json"
  );
```

- 示例二

```
CREATE SINK STREAM kafka_sink (
  a1 string,
  a2 string,
  a3 string,
  a4 INT
```


版权所有 © 华为技术有限公司

```
) // 输出字段
  WITH (
    type="kafka",
    kafka_bootstrap_servers =  "192.x.x.x:9093, 192.x.x.x:9093, 192.x.x.x:9093",
    kafka_topic = "testflink", // 写入的topic
    encode = "csv", // 编码格式，支持json/csv
    kafka_certificate_name = "Flink",
    kafka_properties_delimiter = ",",
    kafka_properties = "sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username=\"xxx\" password=\"xxx\";,sasl.mechanism=PLAIN,security.protocol=SASL_SSL"
  );
```

2.4.11 开源 Kafka 输出流

功能描述

DLI将Flink作业的输出数据输出到Kafka中。

Apache Kafka是一个快速、可扩展的、高吞吐、可容错的分布式发布订阅消息系统，具有高吞吐量、内置分区、支持数据副本和容错的特性，适合在大规模消息处理场景中使用。

前提条件

- Kafka是线下集群，需要通过增强型跨源连接功能将Flink作业与Kafka进行对接。且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH(
    type = "kafka",
    kafka_bootstrap_servers = "",
    kafka_topic = "",
    encode = "json"
  )
```

关键字

表 2-15 关键字说明

参数	是否必选	说明
type	是	输出通道类型，"kafka"表示输出到Kafka中。
kafka_bootstrap_servers	是	Kafka的连接端口，需要确保能连通（需要通过增强型跨源开通DLI队列和Kafka集群的连接）。
kafka_topic	是	写入的topic

版权所有 © 华为技术有限公司

参数	是否必选	说明
encode	是	数据编码格式，可选为“csv”、“json”和“user_defined”。 ● 若编码格式为“csv”，则需配置“field_delimiter”属性。 ● 若编码格式为“user_defined”，则需配置“encode_class_name”和“encode_class_parameter”属性。
filed_delimiter	否	当encode为csv时，用于指定各字段分隔符，默认为逗号。
encode_class_name	否	当encode为user_defined时，需配置该参数，指定用户自实现编码类的类名（包含完整包路径），该类需继承类DeserializationSchema。
encode_class_parameter	否	当encode为user_defined时，可以通过配置该参数指定用户自实现编码类的入参，仅支持一个string类型的参数。
kafka_properties	否	可通过该参数配置kafka的原生属性，格式为"key1=value1;key2=value2"
kafka_certificate_name	否	跨源认证信息名称。跨源认证信息类型为“Kafka_SSL”时，该参数有效。 **说明** ● 指定该配置项时，服务仅加载该认证下指定的文件和密码，系统将自动设置到“kafka_properties”属性中。 ● Kafka SSL认证需要的其他配置信息，需要用户手动在“kafka_properties”属性中配置。

注意事项

无。

示例

将流kafka_sink的数据输出到Kafka中。

```
CREATE SINK STREAM kafka_sink (name STRING)
  WITH (
    type="kafka",
    kafka_bootstrap_servers =  "ip1:port1,ip2:port2",
    kafka_topic = "testsink",
    encode = "json"
  );
```

2.4.12 文件系统输出流(推荐）

功能描述

创建sink流将数据输出到分布式文件系统(HDFS)或者对象存储服务（OBS）等文件系统。数据生成后，可直接对生成的目录创建表，通过DLI SQL进行下一步处理分析，并

 版权所有 © 华为技术有限公司

且输出数据目录支持分区表结构。适用于数据转储、大数据分析、备份或活跃归档、深度或冷归档等场景。

对象存储服务（Object Storage Service，简称OBS）是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  [PARTITIONED BY (attr_name (',' attr_name)*]
  WITH (
    type = "filesystem",
    file.path = "obs://bucket/xx",
    encode = "parquet",
    ak = "",
    sk = ""
  );
```

关键字

表 2-16 关键字说明

参数	是否必选	说明
type	是	输出流类型。“type”为“filesystem”，表示输出数据到文件系统。
file.path	是	输出目录，格式为: schema://file.path。 当前schame只支持obs和hdfs。 ● 当schema为obs时，表示输出到对象存储服务OBS。 ● 当schema为hdfs时，表示输出到HDFS。HDFS需要配置代理用户，具体请参考**HDFS代理用户配置**。 示例：hdfs://node-master1sYAx:9820/user/car_infos，其中node-master1sYAx:9820为MRS集群NameNode所在节点信息。
encode	是	输出数据编码格式，当前支持“parquet”格式和“csv”格式。 ● 当schema为obs时，输出数据编码格式仅支持“parquet”格式。 ● 当schema为hdfs时，输出数据编码格式支持“parquet”格式和“csv”格式。
ak	否	输出到OBS时该参数必填。用于访问OBS认证的accessKey，可使用全局变量，屏蔽敏感信息。
sk	否	输出到OBS时该参数必填。用于访问OBS认证的secretKey，可使用全局变量，屏蔽敏感信息。
krb_auth	否	创建跨源认证的认证名。开启kerberos认证时，需配置该参数。如果创建的MRS集群未开启kerb认证的集群，请确保在DLI队列host文件中添加MRS集群master节点的“/etc/hosts”信息。

版权所有 © 华为技术有限公司

参数	是否必选	说明
field_delimiter	否	属性分隔符。 当编码格式为“csv”时，需要设置属性分隔符，用户可以自定义，如：“,”。

注意事项

- 使用文件系统输出流的Flink作业必须开启checkpoint，保证作业的一致性。
- 为了避免数据丢失或者数据被覆盖，开启作业异常自动重启或者手动重启，需要配置为“从checkpoint恢复”。
- checkpoint间隔设置需在输出文件实时性、文件大小和恢复时长之间进行权衡，比如10分钟。
- checkpoint支持如下两种模式：
  - AtLeastOnce：事件至少被处理一次。
  - ExactlyOnce：事件仅被处理一次。
- 使用文件系统输出流写入数据到OBS时，应避免多个作业写同一个目录的情况。
  - OBS对象存储桶的默认行为为覆盖写，可能导致数据丢失。
  - OBS并行文件系统桶的默认行为追加写，可能导致数据混淆。

  因为以上OBS桶类型行为的区别，为避免作业异常重启可能导致的数据异常问题，请根据您的业务需求选择OBS桶类型。

HDFS 代理用户配置

1. 登录MRS管理页面。
2. 选择MRS的HDFS Namenode配置，在“自定义”中添加配置参数。

   其中，core-site值名称“hadoop.proxyuser.myname.hosts”和“hadoop.proxyuser.myname.groups”中的“myname”为传入的krb认证用户名称。

   说明

   需要保证写入HDFS数据路径权限为777。
3. 配置完成后，单击“保存配置”进行保存。

示例

- 示例一：

  该示例将car_info数据，以buyday字段为分区字段，parquet为编码格式，转储数据到OBS。

```
create sink stream car_infos (
  carId string,
  carOwner string,
  average_speed double,
  buyday string
  ) partitioned by (buyday)
  with (
    type = "filesystem",
    file.path = "obs://obs-sink/car_infos",
```


版权所有 © 华为技术有限公司

```
  encode = "parquet",
  ak = "{{myAk}}",
  sk = "{{mySk}}"
);
```

数据最终在OBS中的存储目录结构为：obs://obs-sink/car_infos/buyday=xx/part-x-x。

数据生成后，可通过如下SQL语句建立OBS分区表，用于后续批处理：

a. 创建OBS分区表。

```
create table car_infos (
  carId string,
  carOwner string,
  average_speed double
)
  partitioned by (buyday string)
  stored as parquet
  location 'obs://obs-sink/car_infos';
```

b. 从关联OBS路径中恢复分区信息。

```
alter table car_infos recover partitions;
```

- 示例二

  该示例将car_info数据，以buyday字段为分区字段，csv为编码格式，转储数据到HDFS。

```
create sink stream car_infos (
  carId string,
  carOwner string,
  average_speed double,
  buyday string
  ) partitioned by (buyday)
  with (
    type = "filesystem",
    file.path = "hdfs://node-master1sYAx:9820/user/car_infos",
    encode = "csv",
    field_delimiter = ","
);
```

数据最终在HDFS中的存储目录结构为：/user/car_infos/buyday=xx/part-x-x。

2.4.13 OBS 输出流

功能描述

创建sink流将DLI数据输出到对象存储服务（OBS）。DLI可以将作业分析结果输出到OBS上。适用于大数据分析、原生云应用程序数据、静态网站托管、备份/活跃归档、深度/冷归档等场景。

对象存储服务（Object Storage Service，简称OBS）是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。OBS的更多信息，请参见。

说明

推荐使用《**文件系统输出流（推荐）**》。

前提条件

OBS输出流功能仅支持输出数据到3.0版本以上的桶，请先查看桶信息确认桶的版本。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
```

 版权所有 © 华为技术有限公司

```
        type = "obs",
        region = "",
        encode = "",
        field_delimiter = "",
        row_delimiter = "",
        obs_dir = "",
        file_prefix = "",
        rolling_size = "",
        rolling_interval = "",
        quote = "",
        array_bracket = "",
        append = "",
        max_record_num_per_file = "",
        dump_interval = "",
        dis_notice_channel = ""
)
```

关键字

表 2-17 关键字说明

参数	是否必选	说明
type	是	输出通道类型，“obs”表示输出到对象存储服务。
region	是	对象存储服务所在区域。
ak	否	访问密钥ID(Access Key ID)。
sk	否	Secret Access Key，与访问密钥ID结合使用的密钥。
encode	是	编码方式。当前支持csv/json/orc/avro/avro_merge/parquet格式。
field_delimiter	否	属性分隔符。 仅当编码方式为csv时需要配置，若不配置，默认分隔符为逗号。
row_delimiter	否	行分隔符。当编码格式为csv、json时需要设置。
json_config	否	当编码格式为json时，用户可以通过该参数来指定json字段和流定义字段的映射关系，格式为“field1=data_json.field1;field2=data_json.field2”。
obs_dir	是	文件存储目录。格式为{桶名}/{目录名}， 如obs-a1/dir1/subdir。当编码格式为csv（append为false）、json（append为false）、avro_merge、parquet时，支持参数化。
file_prefix	否	输出文件名前缀。生成的文件会以file_prefix.*x的方式命名，*如file_prefix.1、 file_prefix.2，若没有设置，默认文件前缀为temp。

版权所有 © 华为技术有限公司

参数	是否必选	说明
rolling_size	否	单个文件最大允许大小。 **说明** • rolling_size和rolling_interval必须至少配一样或者都配置。 • 当文件大小超过设置size后，会生成新文件。 • 支持的单位包括KB/MB/GB，若没写单位，表示单位为字节数。 • 当编码格式为orc时不需要设置。
rolling_interval	否	数据保存到对应目录的时间模式。 **说明** • rolling_size和rolling_interval必须至少配一样或者都配置。 • 设置后数据会按照输出时间输出到相应时间目录下。 • 支持的格式为yyyy/MM/dd/HH/mm， 最小单位只到分钟，大小写敏感。例如配置为yyyy/MM/dd/HH， 则数据会写入对应小时这个时间点所产生的目录下，比如2018-09-10 16时产生的数据就会写到{obs_dir}/2018-09-10_16目录下。 • 当rolling_size和rolling_interval都配置时，表示每个时间所对应的目录下，单个文件超过设置大小时，另起新文件。
quote	否	修饰符，仅当编码格式为csv时可配置，配置后会在每个属性前后各加上修饰符，建议使用不可见字符配置，如"\u0007"。
array_bracket	否	数组括号，仅当编码格式为csv时可配置， 可选值为"()", "{}", "[]"， 例如配置了"{}", 则数组输出格式为{a1,a2}。
append	否	值为true或者false，默认为true。 当OBS不支持append模式，且编码格式为csv和json时，可将该参数设置为false。Append为false时需要设置max_record_num_per_file和dump_interval。
max_record_num_per_file	否	文件最大记录数，当编码格式为csv（append为false）、json（append为false）、orc、avro、avro_merge和parquet时需配置，表明一个文件最多存储记录数，当达到最大值，则另起新文件。
dump_interval	否	触发周期， 当编码格式为orc或者配置了DIS通知提醒时需进行配置。 • 在orc编码方式中，该配置表示周期到达时，即使文件记录数未达到最大个数配置，也将文件上传到OBS上。 • 在DIS通知提醒功能中，该配置表示每周期往DIS发送一个通知提醒，表明该目录已写完。
dis_notice_channel	否	OBS目录完成通知通道。表示每周期往DIS通道中发送一条记录，该记录内容为OBS目录路径，表明该目录已书写完毕。

版权所有 © 华为技术有限公司

参数	是否必选	说明
encoded_data	否	当编码格式为json（append为false）、avro_merge和parquet时，可通过配置该参数指定真正需要编码的数据，格式为${field_name}，表示直接将该流字段的内容作为一个完整的记录进行编码。

注意事项

当配置项支持参数化时，表示将记录中的一列或者多列作为该配置项的一部分。例如当配置项设置为car_${car_brand}时，如果一条记录的car_brand列值为BMW，则该配置项在该条记录下为car_BMW。

示例

- 将car_infos数据输出到OBS的obs-sink桶下，输出目录为car_infos， 输出文件以greater_30作为文件名前缀，当单个文件超过100M时新起一个文件，同时数据输出用csv编码，使用逗号作为属性分隔符，换行符作为行分隔符。

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
  WITH (
    type = "obs",
    encode = "csv",
    region = "xxx",
    field_delimiter = ",",
    row_delimiter = "\n",
    obs_dir = "obs-sink/car_infos",
    file_prefix = "greater_30",
    rolling_size = "100m"
);
```

- orc编码格式示例

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
  WITH (
    type = "obs",
    region = "xxx",
    encode = "orc",
    obs_dir = "dli-append-2/obsorc",
    FILE_PREFIX = "es_info",
    max_record_num_per_file = "100000",
    dump_interval = "60"
);
```

- parquet编码示例请参考**文件系统输出流(推荐)**中的示例。

版权所有 © 华为技术有限公司

2.4.14 RDS 输出流

功能描述

DLI将Flink作业的输出数据输出到关系型数据库（RDS）中。目前支持PostgreSQL和MySQL两种数据库。PostgreSQL数据库可存储更加复杂类型的数据，支持空间信息服务、多版本并发控制（MVCC）、高并发，适用场景包括位置应用、金融保险、互联网电商等。MySQL数据库适用于各种WEB应用、电子商务应用、企业应用、移动应用等场景，减少IT部署和维护成本。

关系型数据库（Relational Database Service，简称RDS）是一种基于云计算平台的在线关系型数据库服务。RDS包括如下几种类型的数据库：MySQL、PostgreSQL和Microsoft SQL Server。

前提条件

- 请务必确保您的账户下已在关系型数据库（RDS）里创建了PostgreSQL或MySQL类型的RDS实例。
- 该场景作业需要运行在DLI的独享队列上，因此要与RDS实例建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "rds",
    username = "",
    password = "",
    db_url = "",
    table_name = ""
  );
```

关键字

表 2-18 关键字说明

参数	是否必选	说明
type	是	输出通道类型，rds表示输出到关系型数据库中。
username	是	数据库连接用户名。
password	是	数据库连接密码。
db_url	是	数据库连接地址，格式为："{database_type}://ip:port/database" 目前支持两种数据库连接：MySQL和PostgreSQL ● MySQL: 'mysql://ip:port/database' ● PostgreSQL: 'postgresql://ip:port/database'

版权所有 © 华为技术有限公司

参数	是否必选	说明
table_name	是	要插入数据的数据库表名。

db_columns	否	支持配置输出流属性和数据库表属性的对应关系，需严格按照输出流的属性顺序配置。
		示例：
		`create sink stream a3(student_name  string, student_age int) with (`
		`type = "rds",`
		`username = "root",`
		`password = "xxxxxxxx",`
		`db_url = "mysql://192.168.0.102:8635/test1",`
		`db_columns = "name,age",`
		`table_name = "t1"`
		`);`
		student_name对应数据库里的name属性，student_age对应数据库里的age属性。
		说明
		• 当不配置db_columns时，若输出流属性个数小于数据库表属性个数，并且数据库多出的属性都是nullable或者有默认值时，这种情况也允许。

primary_key	否	如果想通过主键实时更新表中的数据，需要在创建数据表的时候增加primary_key配置项，如下面例子中的c_timeminute。配置primary_key后，在进行数据写入操作时，如果primary_key存在，则进行更新操作，否则进行插入操作。
		示例：
		`CREATE SINK STREAM test(c_timeminute LONG, c_cnt LONG)`
		`WITH (`
		`type = "rds",`
		`username = "root",`
		`password = "xxxxxxxx",`
		`db_url = "mysql://192.168.0.12:8635/test",`
		`table_name = "test",`
		`primary_key = "c_timeminute");`

operation_field	否	该配置项用于指定数据的处理方式，需要配置为${field_name}的形式，field_name的类型必读为string，field_name所代表的真正内容表示为D或者DELETE时，表示删除数据库中该条记录，其余默认插入数据。

注意事项

stream_id所定义的流格式需和数据库中的表格式一致。

示例

将流audi_cheaper_than_30w的数据输出到数据库test的audi_cheaper_than_30w表下。

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
```


版权所有 © 华为技术有限公司

```
  car_price INT
)
  WITH (
    type = "rds",
    username = "root",
    password = "xxxxxx",
    db_url = "mysql://192.168.1.1:8635/test",
    table_name = "audi_cheaper_than_30w"
);
```

2.4.15 SMN 输出流

功能描述

DLI将Flink作业的输出数据输出到消息通知服务（SMN）中。

消息通知服务（Simple Message Notification，简称SMN）为DLI提供可靠的、可扩展的、海量的消息处理服务，它大大简化系统耦合，能够根据用户的需求，向订阅终端主动推送消息。可用于连接云服务、向多个协议推送消息以及集成在产生或使用通知的任何其他应用程序等场景。

SMN的更多信息，请参见。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH(
    type = "smn",
    region = "",
    topic_urn = "",
    urn_column = "",
    message_subject = "",
    message_column = ""
  )
```

关键字

表 2-19 关键字说明

参数	是否必选	说明
type	是	输出通道类型，smn表示输出到消息通知服务中。
region	是	SMN所在区域。
topic_urn	否	SMN服务的主题URN，用于静态主题URN配置。作为消息通知的目标主题，需要提前在SMN服务中创建。 与“urn_column”配置两者至少存在一个，同时配置时，“topic_urn”优先级更高。
urn_column	否	主题URN内容的字段名，用于动态主题URN配置。 与“topic_urn”配置两者至少存在一个，同时配置时，“topic_urn”优先级更高。
message_subject	是	发往SMN服务的消息标题，用户自定义。

版权所有 © 华为技术有限公司

参数	是否必选	说明
message_column	是	输出流的字段名，其内容作为消息的内容，用户自定义。目前只支持默认的文本消息。

注意事项

无。

示例

将流over_speed_warning的数据输出到消息通知服务SMN中。

```
//静态主题配置
CREATE SINK STREAM over_speed_warning (
  over_speed_message STRING /* over speed message */
)
  WITH (
    type = "smn",
    region = "xxx",
    topic_Urn = "xxx",
    message_subject = "message title",
    message_column = "over_speed_message"
  );
//动态主题配置
CREATE SINK STREAM over_speed_warning2 (
    over_speed_message STRING, /* over speed message */
    over_speed_urn STRING
)
  WITH (
    type = "smn",
    region = "xxx",
    urn_column = "over_speed_urn",
    message_subject = "message title",
    message_column = "over_speed_message"
  );
```

2.5 创建中间流

功能描述

中间流用来简化sql逻辑，若sql逻辑比较复杂，可以写多个sql语句，用中间流进行串接。中间流仅为逻辑意义上的流，不会产生数据存储。

语法格式

```
CREATE TEMP STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
```

示例

```
create temp stream a2(attr1 int, attr2 string);
```

2.6 创建维表

版权所有 © 华为技术有限公司

2.6.1 创建 Redis 表

创建Redis表用于与输入流连接。

流表JOIN语法请参见**流表JOIN**。

语法格式

```
CREATE TABLE table_id (key_attr_name STRING(, hash_key_attr_name STRING)?, value_attr_name STRING)
  WITH (
    type = "dcs_redis",
    cluster_address = ""(,password = "")?,
    value_type= "",
    key_column= ""(,hash_key_column="")?);
```

关键字

表 2-20 关键字说明

参数	是否必选	说明
type	是	输出通道类型，dcs_redis表示输出到分布式缓存服务的Redis存储系统中。
cluster_address	是	Redis实例连接地址。
password	否	Redis实例连接密码，当设置为免密访问时，省略该配置项。
value_type	是	指定数据类型。支持的数据类型包括：string, list, hash, set, zset。
key_column	是	指定代表Redis key属性的列名。
hash_key_column	否	当value_type设置为hash时，需要指定本字段作为第二级key属性的列名。
cache_max_num	否	表示最大缓存的查询结果数，默认值为32768。
cache_time	否	表示数据库查询结果在内存中缓存的最大时间。单位为毫秒，默认值为10000，当值为0时表示不缓存。

注意事项

- 不支持Redis集群。
- 请务必确保您的账户下已在分布式缓存服务（DCS）里创建了Redis类型的缓存实例。
- 该场景作业需要运行在DLI的独享队列上，因此要与DCS实例建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

版权所有 © 华为技术有限公司

示例

Redis表用于与输入流连接。

```
CREATE TABLE table_a (attr1 string, attr2 string, attr3 string)
  WITH (
    type = "dcs_redis",
    value_type = "hash",
    key_column = "attr1",
    hash_key_column = "attr2",
    cluster_address = "192.168.1.238:6379",
    password = "xxxxxxxx"
  );
```

2.6.2 创建 RDS 表

创建RDS/DWS表用于与输入流连接。

流表JOIN语法请参见**流表JOIN**。

前提条件

- 请务必确保您的账户下已在关系型数据库（RDS）里创建了PostgreSQL或MySQL类型的RDS实例。
- 该场景作业需要运行在DLI的独享队列上，因此要与RDS实例建立增强型跨源连接，且用户可以根据实际所需设置相应安全组规则。

语法格式

```
CREATE TABLE  table_id (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "rds",
    username = "",
    password = "",
    db_url = "",
    table_name = ""
  );
```

关键字

表 2-21 关键字说明

参数	是否必选	说明
type	是	输出通道类型，rds表示输出到关系型数据库中。
username	是	数据库连接用户名。
password	是	数据库连接密码。

版权所有 © 华为技术有限公司

参数	是否必选	说明
db_url	是	数据库连接地址，格式为："{database_type}://ip:port/database" 目前支持两种数据库连接：MySQL和PostgreSQL ● MySQL: 'mysql://ip:port/database' ● PostgreSQL: 'postgresql://ip:port/database' **说明** 将数据库连接地址设置为DWS数据库地址，即可创建DWS维表。DWS数据库版本大于8.1.0后，无法用开源的postgresql驱动连接，需要用gaussdb驱动进行连接。
table_name	是	用于查询数据的数据库表名。
db_columns	否	流属性和数据库表的字段对应关系。当sink流中流属性字段和数据库表中的流属性字段不完全匹配时，该参数必配。格式为“dbtable_attr1,dbtable_attr2,dbtable_attr3“。
cache_max_num	否	表示最大缓存的查询结果数，默认值为32768。
cache_time	否	表示数据库查询结果在内存中缓存的最大时间。单位为毫秒，默认值为10000，当值为0时表示不缓存。

示例

RDS表用于与输入流连接。

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "dis",
    region = "",
    channel = "dliinput",
    encode = "csv",
    field_delimiter = ","
  );

CREATE TABLE  db_info (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "rds",
    username = "root",
    password = "******",
    db_url = "postgresql://192.168.0.0:2000/test1",
    table_name = "car"
);

CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
```


版权所有 © 华为技术有限公司

```
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
  WITH (
    type = "dis",
    region = "",
    channel = "dlioutput",
    partition_key = "car_owner",
    encode = "csv",
    field_delimiter = ","
  );

INSERT INTO audi_cheaper_than_30w
SELECT a.car_id, b.car_owner, b.car_brand, b.car_price
FROM car_infos as a join db_info as b on a.car_id = b.car_id;
```

说明

将数据库连接地址设置为DWS数据库地址，即可创建DWS维表。DWS数据库版本大于8.1.0后，无法用开源的postgresql驱动连接，需要用gaussdb驱动进行连接。

2.7 自拓展生态

2.7.1 自拓展输入流

用户可通过编写代码实现从想要的云生态或者开源生态获取数据，作为Flink作业的输入数据。

语法格式

```
CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
  WITH (
    type = "user_defined",
    type_class_name = "",
    type_class_parameter = ""
  )
  (TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME
```

关键字

表 2-22 关键字说明

参数	是否必选	说明
type	是	数据源类型，"user_defined"表示数据源为用户自定义数据源。
type_class_name	是	用户实现获取源数据的source类名称，注意包含完整包路径。
type_class_parameter	是	用户自定义source类的入参，仅支持一个string类型的参数。

版权所有 © 华为技术有限公司

注意事项

用户自定义source类需要继承类RichParallelSourceFunction，并指定数据类型为Row
例如定义类MySource：public class MySource extends RichParallelSourceFunction<Row>{}，重点实现其中的open、run和close函数。

依赖pom:

```
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-streaming-java_2.11</artifactId>
   <version>${flink.version}</version>
   <scope>provided</scope>
</dependency>
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-core</artifactId>
   <version>${flink.version}</version>
   <scope>provided</scope>
</dependency>
```

示例

实现每周期产生一条数据（仅包含一个字段，类型为INT，初始值为1，每周期加1），周期时长为60s，通过入参指定。

```
CREATE SOURCE STREAM user_in_data (
   count INT
   )
 WITH (
   type = "user_defined",
   type_class_name = "mySourceSink.MySource",
   type_class_parameter = "60"
    )
    TIMESTAMP BY car_timestamp.rowtime;
```

说明

自定义source类实现，需要将该类打在jar包中，通过sql编辑页上传udf函数按钮上传。

2.7.2 自拓展输出流

用户可通过编写代码实现将DLI处理之后的数据写入指定的云生态或者开源生态。

语法格式

```
CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)* )
 WITH (
   type = "user_defined",
   type_class_name = "",
   type_class_parameter = ""
 );
```


版权所有 © 华为技术有限公司

关键字

表 2-23 关键字说明

参数	是否必选	说明
type	是	数据源类型，"user_defined"表示数据源为用户自定义数据源。
type_class_name	是	用户实现获取源数据的sink类名称，注意包含完整包路径。
type_class_parameter	是	用户自定义sink类的入参，仅支持一个string类型的参数。

注意事项

用户自定义sink类需要继承类RichSinkFunction，并指定数据类型为Row例如定义类MySink：public class MySink extends RichSinkFunction<Row>{}，重点实现其中的open、invoke和close函数。

依赖pom:

```
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-streaming-java_2.11</artifactId>
   <version>${flink.version}</version>
   <scope>provided</scope>
</dependency>
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-core</artifactId>
   <version>${flink.version}</version>
   <scope>provided</scope>
</dependency>
```

示例

实现数据以CSV编码写入DIS通道。

```
CREATE SINK STREAM user_out_data (
   count INT
)
 WITH (
   type = "user_defined",
   type_class_name = "mySourceSink.MySink",
   type_class_parameter = ""
    );
```

说明

自定义sink类实现，需要将该类打在jar包中，通过sql编辑页上传udf函数按钮上传。

版权所有 © 华为技术有限公司

2.8 数据类型

概述

数据类型是数据的一个基本属性，用于区分不同类别的数据。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数据表中。数据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。

大数据平台的Flink SQL与开源社区相同，支持原生数据类型、复杂数据类型和复杂类型嵌套。

原生数据类型

Flink SQL支持原生数据类型，请参见**表2-24**。

表 2-24 原生数据类型

数据类型	描述	存储空间	范围
VARCHAR	可变长度的字符	-	-
BOOLEAN	布尔类型	-	TRUE/FALSE
TINYINT	有符号整数	1字节	-128-127
SMALLINT	有符号整数	2字节	-32768-32767
INT	有符号整数	4字节	-2147483648～2147483647
INTEGER	有符号整数	4字节	-2147483648～2147483647
BIGINT	有符号整数	8字节	-9223372036854775808～9223372036854775807
REAL	单精度浮点型	4字节	-
FLOAT	单精度浮点型	4字节	-
DOUBLE	双精度浮点型	8字节	-
DECIMAL	固定有效位数和小数位数的数据类型	-	-
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 to 9999-12-31

版权所有 © 华为技术有限公司

数据类型	描述	存储空间	范围
TIME	时间类型，以HH:mm:ss表示。 例如20:17:40	-	-
TIMESTAMP(3)	完整日期，包括日期和时间。 例如：1969-07-20 20:17:40	-	-
INTERVAL timeUnit [TO timeUnit]	时间间隔 例如：INTERVAL '1:5' YEAR TO MONTH, INTERVAL '45' DAY	-	-

复杂数据类型

Flink SQL支持复杂数据类型和复杂类型嵌套。复杂数据类型如表2-25所示。

表 2-25 复杂数据类型

数据类型	描述	声明方式	引用方式	构造方式
ARRAY	一组有序字段，所有字段的数据类型必须相同。	ARRAY[TYPE]	变量名[下标]，下标从1开始，例如：v1[1]	Array[value1, value2, ...] as v1
MAP	一组无序的键/值对。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[TYPE，TYPE]	变量名[key]，例如：v1[key]	Map[key, value, key2, value2, key3, value3.....] as v1
ROW	一组命名的字段，字段的数据类型可以不同。	ROW<a1 TYPE1, a2 TYPE2>	变量名.字段名，例如：v1.a1	Row('1',2) as v1

使用示例如下：

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  address ROW<city STRING, province STRING, country STRING>,
  average_speed MAP[STRING, LONG],
  speeds ARRAY[LONG]
)
  WITH (
    type = "dis",
    region = "xxx",
```


版权所有 © 华为技术有限公司

```
  channel = "dliinput",
  encode = "json"
);

CREATE temp STREAM car_speed_infos (
 car_id STRING,
 province STRING,
 average_speed LONG,
 start_speed LONG
);

INSERT INTO car_speed_infos SELECT
  car_id,
  address.province,
  average_speed[address.city],
  speeds[1]
FROM car_infos;
```

复杂类型嵌套

- Json格式增强

  以Source为例进行说明，Sink的使用方法相同。

  - 支持配置Json_schema

    配置了json_schema后，可以不声明DDL中的字段，自动从json_schema中生成。使用示例如下：

    ```
    CREATE SOURCE STREAM data_with_schema WITH (
          type = "dis",
          region = "xxx",
          channel = "dis-in",
          encode = "json",
          json_schema = '{"definitions":{"address":{"type":"object","properties":{"street_address":
    {"type":"string"},"city":{"type":"string"},"state":{"type":"string"}},"required":
    ["street_address","city","state"]}},"type":"object","properties":{"billing_address":{"$ref":"#/
    definitions/address"},"shipping_address":{"$ref":"#/definitions/address"},"optional_address":
    {"oneOf":[{"type":"null"},{"$ref":"#/definitions/address"}]}}}'
        );

        CREATE SINK STREAM buy_infos (
          billing_address_city STRING,
          shipping_address_state string
        ) WITH (
          type = "obs",
          encode = "csv",
          region = "xxx" ,
          field_delimiter = ",",
          row_delimiter = "\n",
          obs_dir = "bucket/car_infos",
          file_prefix = "over",
          rolling_size = "100m"
        );

        insert into buy_infos select billing_address.city, shipping_address.state from
    data_with_schema;
    ```

    示例数据：

    ```
    {
     "billing_address":
      {
       "street_address":"xxx",
       "city":"xxx",
       "state":"xxx"
       },
     "shipping_address":
      {
       "street_address":"xxx",
       "city":"xxx",
    ```


版权所有 © 华为技术有限公司

```
  "state":"xxx"
 }
}
```

- 支持不配置json_schema也不配置json_config。json_config使用可以参考**开源Kafka输入流**样例说明。

  这种情况下默认用ddl中属性名当做json key来进行解析。

  测试示例数据如下，测试数据既包括嵌套json字段，如billing_address、shipping_address，也包括非嵌套的字段id、type2。

```
{
 "id":"1",
 "type2":"online",
 "billing_address":
 {
  "street_address":"xxx",
  "city":"xxx",
  "state":"xxx"
  },
 "shipping_address":
 {
  "street_address":"xxx",
  "city":"xxx",
  "state":"xxx"
 }
}
```

  具体建表和使用示例参考如下：

```
CREATE SOURCE STREAM car_info_data (
      id STRING,
      type2 STRING,
      billing_address Row<street_address string, city string, state string>,
      shipping_address Row<street_address string, city string, state string>,
      optional_address Row<street_address string, city string, state string>
    ) WITH (
      type = "dis",
      region = "xxx",
      channel = "dis-in",
      encode = "json"
    );

   CREATE SINK STREAM buy_infos (
      id STRING,
      type2 STRING,
      billing_address_city STRING,
      shipping_address_state string
    ) WITH (
      type = "obs",
      encode = "csv",
      region = "xxx",
      field_delimiter = ",",
      row_delimiter = "\n",
      obs_dir = "bucket/car_infos",
      file_prefix = "over",
      rolling_size = "100m"
    );

    insert into buy_infos select id, type2, billing_address.city, shipping_address.state from
car_info_data;
```

- Sink序列化支持复杂类型
  - 目前只有CSV、Json两种格式支持复杂类型。
  - Json请参考**•Json格式增强**。
  - 由于CSV没有标准的格式，所以目前暂不支持source解析，只支持sink。
  - 输出格式：尽量和flink原生保持一致。

    Map: {key1=Value1, key2=Value2}

版权所有 © 华为技术有限公司

Row: 平摊用逗号分隔属性，如Row(1, '2') => 1,'2'

2.9 内置函数

2.9.1 数学运算函数

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。

Flink SQL提供的关系运算符，请参见表2-26。

表 2-26 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	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。
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 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 IS DISTINCT FROM B	BOOLEAN	若A与B不相等，则返回TRUE，将空值视为相同。
A IS NOT DISTINCT FROM B	BOOLEAN	若A与B相等，则返回TRUE，将空值视为相同。

版权所有 © 华为技术有限公司

运算符	返回类型	描述
A BETWEEN [ASYMMETRIC | SYMMETRIC] B AND C	BOOLEAN	若A大于或等于B且小于或等于C，则返回TRUE。 • ASYMMETRIC：表示B和C位置相关。 例如：A BETWEEN ASYMMETRIC B AND C 等价于 (A BETWEEN B AND C)。 • SYMMETRIC：表示B和C位置不相关。 例如：A BETWEEN SYMMETRIC B AND C 等价于 (A BETWEEN B AND C) OR (A BETWEEN C AND B)。
A NOT BETWEEN B AND C	BOOLEAN	若A小于B或大于C，则返回TRUE。
A LIKE B [ESCAPE C]	BOOLEAN	若A与模式B匹配，则返回TRUE。 必要时可以定义转义字符C。
A NOT LIKE B [ESCAPE C]	BOOLEAN	若A与模式B不匹配，则返回TRUE。 必要时可以定义转义字符C。
A SIMILAR TO B [ESCAPE C]	BOOLEAN	若A与正则表达式B匹配，则返回TRUE。 必要时可以定义转义字符C。
A NOT SIMILAR TO B [ESCAPE C]	BOOLEAN	若A与正则表达式B不匹配，则返回TRUE。 必要时可以定义转义字符C。
value IN (value [, value]*)	BOOLEAN	若值等于列表中的值，则返回TRUE。
value NOT IN (value [, value]*)	BOOLEAN	若值不等于列表中的每个值，则返回TRUE。

说明

- double、real和float值存在一定的精度差。且我们不建议直接使用等号“=”对两个double类型数据进行比较。用户可以使用两个double类型相减，而后取绝对值的方式判断。当绝对值足够小时，认为两个double数值相等，例如:
  abs(0.9999999999 - 1.0000000000) < 0.000000001 //0.9999999999和1.0000000000为10位精度，而0.000000001为9位精度，此时可以认为0.9999999999和1.0000000000相等。
- 数值类型可与字符串类型进行比较。做大小(>,<,>=,<=)比较时，会默认将字符串转换为数值类型，因此不支持字符串内有除数字字符之外的字符。
- 字符串之间可以进行比较。

逻辑运算符

常用的逻辑操作符有AND、OR和NOT，优先级顺序为：NOT>AND>OR。

运算规则请参见**表2-27**，表中的A和B代表逻辑表达式。

版权所有 © 华为技术有限公司

表 2-27 逻辑运算符

运算符	返回类型	描述
A OR B	BOOLEAN	若A或B为TRUE，则返回TRUE，且支持三值逻辑。
A AND B	BOOLEAN	若A和B为TRUE，则返回TRUE，且支持三值逻辑。
NOT A	BOOLEAN	若A不为TRUE则返回TRUE；若A为UNKNOWN，返回UNKNOWN。
A IS FALSE	BOOLEAN	若A为FALSE则返回TRUE； 若A为UNKNOWN，则返回FALSE。
A IS NOT FALSE	BOOLEAN	若A不为FALSE则返回TRUE；若A为UNKNOWN，则返回TRUE。
A IS TRUE	BOOLEAN	若A为TRUE，则返回TRUE；若A为UNKNOWN，则返回FALSE。
A IS NOT TRUE	BOOLEAN	若A不为TRUE则返回TRUE；若A为UNKNOWN，则返回TRUE。
A IS UNKNOWN	BOOLEAN	若A为UNKNOWN，则返回TRUE。
A IS NOT UNKNOWN	BOOLEAN	若A不为UNKNOWN，则返回TRUE。

说明

逻辑操作符只允许boolean类型参与运算，不支持隐式类型转换。

算术运算符

算术运算符包括双目运算符与单目运算符，这些运算符都将返回数字类型。Flink SQL所支持的算术运算符如**表2-28**所示。

表 2-28 算术运算符

运算符	返回类型	描述
+ numeric	所有数字类型	返回数字。
- numeric	所有数字类型	返回负数。
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。

 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	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。
POWER(A, B)	所有数字类型	返回A数的B次方乘幂。
ABS(numeric)	所有数字类型	返回数值的绝对值。
MOD(A, B)	所有数字类型	返回A除以B的余数（模数）。 返回值只有在A为负数时才为负数。
SQRT(A)	所有数字类型	返回A的平方根。
LN(A)	所有数字类型	返回A的自然对数（基数e）。
LOG10(A)	所有数字类型	返回A的基数10对数。
EXP(A)	所有数字类型	返回e的a次方。
CEIL(A) CEILING(A)	所有数字类型	将参数向上舍入为最接近的整数。例如ceil(21.2)，返回22。
FLOOR(A)	所有数字类型	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。
SIN(A)	所有数字类型	计算给定A的正弦值。
COS(A)	所有数字类型	计算给定A的余弦值。
TAN(A)	所有数字类型	计算给定A的正切值。
COT(A)	所有数字类型	计算给定A的余切值。
ASIN(A)	所有数字类型	计算给定A的反正弦值。
ACOS(A)	所有数字类型	计算给定A的反余弦值。

版权所有 © 华为技术有限公司

运算符	返回类型	描述
ATAN(A)	所有数字类型	计算给定A的反正切值。
DEGREES(A)	所有数字类型	返回弧度所对应的角度。
RADIANS(A)	所有数字类型	返回角度所对应的弧度。
SIGN(A)	所有数字类型	返回a所对应的正负号，a为正返回1，a为负，返回-1，否则则返回0。
ROUND(A, d)	所有数字类型	返回小数部分d位之后数字的四舍五入，d为int型。例如round(21.263,2)，返回21.26。
PI()	所有数字类型	返回pi的值。

说明

字符串类型不能参与算术运算。

2.9.2 字符串函数

DLI常用字符串函数如下所示：

表 2-29 字符串运算符

运算符	返回类型	描述
||	VARCHAR	两个字符串的拼接。
CHAR_LENGTH	INT	返回字符串中的字符数量。
CHARACTER_LENGTH	INT	返回字符串中的字符数量。
CONCAT	VARCHAR	拼接两个或多个字符串值从而组成一个新的字符串。如果任一参数为NULL时，则跳过该参数。
CONCAT_WS	VARCHAR	将每个参数值和第一个参数separator指定的分隔符依次连接到一起组成新的字符串，长度和类型取决于输入值。
HASH_CODE	INT	返回字符串的HASH_CODE()的绝对值。参数除string外，也支持int/bigint/float/double。
INITCAP	VARCHAR	返回字符串，将单词首字母转换为大写，其余为小写。 单词是由非字母、数字、字符分隔的字母、数字、字符序列。

版权所有 © 华为技术有限公司

运算符	返回类型	描述
IS_ALPHA	BOOLEAN	判断字符串是否只包含字母。
IS_DIGITS	BOOLEAN	判断字符串是否只包含数字。
IS_NUMBER	BOOLEAN	判断字符串是否是数值。
IS_URL	BOOLEAN	判断字符串是否是合法的URL地址。
JSON_VALUE	VARCHAR	获取json字符串中指定path的值。
KEY_VALUE	VARCHAR	获取键值对字符串中某一个key对应的值。
LOWER	VARCHAR	返回小写字符的字符串。
LPAD	VARCHAR	将pad字符串拼接到str字符串的的左端，直到新的字符串达到指定长度len为止。
MD5	VARCHAR	返回字符串的MD5值。如果参数为空串（即参数为''）时，则返回空串。
OVERLAY	VARCHAR	用y替换x的子串。从start_position开始，替换length+1个字符。
POSITION	INT	返回目标字符串x在被查询字符串y里第一次出现的位置。如果目标字符串x在被查询字符串y中不存在，返回值为0。
REPLACE	VARCHAR	字符串替换函数，将字符串str1中的所有str2替换成str3。 • str1：原字符。 • str2：目标字符。 • str3：替换字符。
RPAD	VARCHAR	将pad字符串拼接到str字符串的的右端，直到新的字符串达到指定长度len为止。
SHA1	STRING	返回字符串expr的SHA1值。
SHA256	STRING	返回字符串expr的SHA256值。
STRING_TO_ARRAY	ARRAY[STRING]	将字符串value按delimiter分隔为字符串数组。
SUBSTRING	VARCHAR	返回从给定位置开始的A的子字符串。起始位置从1开始。

版权所有 © 华为技术有限公司

运算符	返回类型	描述
TRIM	STRING	从B中除去字符串首尾/首位/末尾的A。默认情况下，首尾的A都被删除。
UPPER	VARCHAR	返回转换为大写字符的字符串。

||

- 功能描述

  两个字符串的拼接。

- 语法

  ```
  VARCHAR VARCHAR a || VARCHAR b
  ```

- 参数说明
  - a：字符串。
  - b：字符串。
- 示例
  - 测试语句

    ```
    SELECT "hello" || "world";
    ```

  - 测试结果

    ```
    "helloworld"
    ```

CHAR_LENGTH

- 功能描述

  返回字符串中的字符数量。

- 语法

  ```
  INT CHAR_LENGTH(a)
  ```

- 参数说明
  - a：字符串。
- 示例
  - 测试语句

    ```
    SELECT  CHAR_LENGTH(var1) as aa FROM T1;
    ```

  - 测试数据和结果

    表 2-30 测试数据和结果

测试数据（var1）	测试结果（aa）
abcde123	8

CHARACTER_LENGTH

- 功能描述

  返回字符串中的字符数量。

版权所有 © 华为技术有限公司

- 语法
  ```
  INT CHARACTER_LENGTH(a)
  ```
- 参数说明
  - a：字符串。
- 示例
  - 测试语句
    ```
    SELECT  CHARACTER_LENGTH(var1) as aa FROM T1;
    ```
  - 测试数据和结果

    表 2-31 测试数据和结果

测试数据（var1）	测试结果（aa）
abcde123	8

CONCAT

- 功能描述

  拼接两个或多个字符串值从而组成一个新的字符串。如果任一参数为NULL时，则跳过该参数。
- 语法
  ```
  VARCHAR CONCAT(VARCHAR var1, VARCHAR var2, ...)
  ```
- 参数说明
  - var1：字符串
  - var2：字符串
- 示例
  - 测试语句
    ```
    SELECT CONCAT("abc", "def", "ghi", "jkl");
    ```
  - 测试结果
    ```
    "abcdefghijkl"
    ```

CONCAT_WS

- 功能描述

  将每个参数值和第一个参数separator指定的分隔符依次连接到一起组成新的字符串，长度和类型取决于输入值。

  说明

  如果separator取值为null，则将separator视作与空串进行拼接。如果其它参数为null，在执行拼接过程中跳过取值为null的参数。
- 语法
  ```
  VARCHAR CONCAT_WS(VARCHAR separator, VARCHAR var1, VARCHAR var2, ...)
  ```
- 参数说明
  - separator：分隔符。
  - var1：字符串。
  - var2：字符串。

版权所有 © 华为技术有限公司

- 示例
  - 测试语句
    ```
    SELECT CONCAT_WS("-", "abc", "def", "ghi", "jkl");
    ```
  - 测试结果
    ```
    "abc-def-ghi-jkl"
    ```

HASH_CODE

- 功能描述

  返回字符串的HASH_CODE()的绝对值。参数除string外，也支持int/bigint/float/double。
- 语法
  ```
  INT HASH_CODE(VARCHAR str)
  ```
- 参数说明
  - str：字符串。
- 示例
  - 测试语句
    ```
    SELECT HASH_CODE("abc");
    ```
  - 测试结果
    ```
    96354
    ```

INITCAP

- 功能描述

  返回字符串，将字符串首字母转换为大写，其余为小写。字符串是由非字母、数字、字符分隔的字母、数字、字符序列。
- 语法
  ```
  VARCHAR INITCAP(a)
  ```
- 参数说明
  - a：字符串。
- 示例
  - 测试语句
    ```
    SELECT INITCAP(var1)as aa FROM T1;
    ```
  - 测试数据和结果

    表 2-32 测试数据和结果

测试数据（var1）	测试结果（aa）
aBCde	Abcde

IS_ALPHA

- 功能描述

  判断字符串是否只包含字母。
- 语法
  ```
  BOOLEAN IS_ALPHA(VARCHAR content)
  ```

 版权所有 © 华为技术有限公司

- 参数说明
  - content：输入字符串。
- 示例
  - 测试语句
    ```
    SELECT IS_ALPHA(content)  AS case_result FROM T1;
    ```
  - 测试数据和结果

    表 2-33 测试数据和结果

测试数据（content）	测试结果（case_result）
Abc	true
abc1#$	false
null	false
""(空字符串)	false

IS_DIGITS

- 功能描述

  判断字符串是否只包含数字。
- 语法
  ```
  BOOLEAN IS_DIGITS(VARCHAR content)
  ```
- 参数说明
  - content：输入字符串。
- 示例
  - 测试语句
    ```
    SELECT IS_DIGITS(content) AS case_result FROM T1;
    ```
  - 测试数据和结果

    表 2-34 测试数据和结果

测试数据（content）	测试结果（case_result）
78	true
78.0	false
78a	false
null	false
"" (空字符串)	false

IS_NUMBER

- 功能描述

  判断字符串是否是数值。

 版权所有 © 华为技术有限公司

- 语法
  ```
  BOOLEAN IS_NUMBER(VARCHAR content)
  ```
- 参数说明
  - content：输入字符串。
- 示例
  - 测试语句
    ```
    SELECT IS_NUMBER(content) AS case_result FROM T1;
    ```
  - 测试数据和结果

    表 2-35 测试数据和结果

测试数据（content）	测试结果（case_result）
78	true
78.0	true
78a	false
null	false
"" (空字符串)	false

IS_URL

- 功能描述

  判断字符串是否是合法的URL地址。
- 语法
  ```
  BOOLEAN IS_URL(VARCHAR content)
  ```
- 参数说明
  - content：输入字符串。
- 示例
  - 测试语句
    ```
    SELECT IS_URL(content) AS case_result FROM T1;
    ```
  - 测试数据和结果

    表 2-36 测试数据和结果

测试数据（content）	测试结果（case_result）
https://www.testweb.com	true
https://www.testweb.com:443	true
www.testweb.com:443	false
null	false
"" (空字符串)	false

版权所有 © 华为技术有限公司

JSON_VALUE

- 功能描述

  获取json字符串中指定path的值。

- 语法

  ```
  VARCHAR JSON_VALUE(VARCHAR content, VARCHAR path)
  ```

- 参数说明
  - content：输入字符串。
  - path：要获取的path路径。

- 示例
  - 测试语句

    ```
    SELECT JSON_VALUE(content, path) AS case_result FROM T1;
    ```

  - 测试数据和结果

    表 2-37 测试数据和结果

测试数据（content，path）		测试结果（case_result）
{ "a1":"v1","a2":7,"a3": 8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}	$	{ "a1":"v1","a2":7,"a3":8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}
{ "a1":"v1","a2":7,"a3": 8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}	$.a1	v1
{ "a1":"v1","a2":7,"a3": 8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}	$.a4	{"a41":"v41","a42": ["v1","v2"]}
{ "a1":"v1","a2":7,"a3": 8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}	$.a4.a42	["v1","v2"]
{ "a1":"v1","a2":7,"a3": 8.0,"a4": {"a41":"v41","a42": ["v1","v2"]}}	$.a4.a42[0]	v1

KEY_VALUE

- 功能描述

  获取键值对字符串中某一个key对应的值。

- 语法

  ```
  VARCHAR KEY_VALUE(VARCHAR content, VARCHAR split1, VARCHAR split2, VARCHAR key_name)
  ```

- 参数说明
  - content：输入字符串。
  - split1：多个键值对分隔符。
  - split2：key/value分隔符。
  - key_name：要获取的键名称。

版权所有 © 华为技术有限公司

- 示例
  - 测试语句
    ```
    SELECT KEY_VALUE(content, split1, split2, key_name)  AS case_result FROM T1;
    ```
  - 测试数据和结果

    表 2-38 测试数据和结果

测试数据（content, split1, split2, key_name）				测试结果（case_result）
k1=v1;k2=v2	;	=	k1	v1
null	;	=	k1	null
k1=v1;k2=v2	null	=	k1	null

LOWER

- 功能描述

  返回小写字符的字符串。
- 语法
  ```
  VARCHAR LOWER(A)
  ```
- 参数说明
  - A：字符串。
- 示例
  - 测试语句
    ```
    SELECT LOWER(var1) AS aa FROM T1;
    ```
  - 测试数据和结果

    表 2-39 测试数据和结果

测试数据（var1）	测试结果（aa）
ABc	abc

LPAD

- 功能描述

  将pad字符串拼接到str字符串的的左端，直到新的字符串达到指定长度len为止。
- 语法
  ```
  VARCHAR LPAD(VARCHAR str, INT len, VARCHAR pad)
  ```
- 参数说明
  - str：拼接前的字符串。
  - len：拼接后的字符串的长度。
  - pad：被拼接的字符串。

 版权所有 © 华为技术有限公司

说明

- 任意参数为null时返回null。
- len为负数时返回为null。
- len不大于str长度时，返回str裁剪为len长度的字符串。

- 示例
  - 测试语句
    ```
    SELECT
      LPAD("adc", 2, "hello"),
      LPAD("adc", -1, "hello"),
      LPAD("adc", 10, "hello");
    ```
  - 测试结果
    ```
    "ad",,"helloheadc"
    ```

MD5

- 功能描述

  返回字符串的MD5值。如果参数为空串（即参数为''）时，则返回空串。
- 语法
  ```
  VARCHAR MD5(VARCHAR str)
  ```
- 参数说明
  - str：字符串
- 示例
  - 测试语句
    ```
    SELECT MD5("abc");
    ```
  - 测试结果
    ```
    "900150983cd24fb0d6963f7d28e17f72"
    ```

OVERLAY

- 功能描述

  用y替换x的子串。从start_position开始，替换length+1个字符。
- 语法
  ```
  VARCHAR OVERLAY ( (VARCHAR x PLACING VARCHAR y FROM INT start_position [ FOR INT
  length ]) )
  ```
- 参数说明
  - x：字符串。
  - y：字符串。
  - start_position：起始位置。
  - length（可选）：字符长度。
- 示例
  - 测试语句：
    ```
    OVERLAY('abcdefg' PLACING 'xyz' FROM 2 FOR 2) AS result FROM T1;
    ```
  - 测试结果：

版权所有 © 华为技术有限公司

表 2-40 测试结果

result
axyzdefg

POSITION

- 功能描述

  返回目标字符串x在被查询字符串y里第一次出现的位置。如果目标字符串x在被查询字符串y中不存在，返回值为0。

- 语法

```
INTEGER POSITION(x IN y)
```

- 参数说明
  - x：字符串。
  - y：字符串。
- 示例
  - 测试语句：

```
POSITION('in' IN 'chin') AS result FROM T1;
```

  - 测试结果

表 2-41 测试结果

result
3

REPLACE

- 功能描述

  字符串替换函数，将字符串str1中的所有str2替换成str3。

- 语法

```
VARCHAR REPLACE(VARCHAR str1, VARCHAR str2, VARCHAR str3)
```

- 参数说明
  - str1：原字符。
  - str2：目标字符。
  - str3：替换字符。
- 示例
  - 测试语句：

```
SELECT
  replace(
    "hello world hello world hello world",
    "world",
    "hello"
  );
```

  - 测试结果

```
"hello hello hello hello hello hello"
```


版权所有 © 华为技术有限公司

RPAD

- 功能描述

  将pad字符串拼接到str字符串的的右端，直到新的字符串达到指定长度len为止。
  - 如果任意参数为null时，则返回null。
  - len为负数时，返回为null。
  - pad为空串，如果len小于str长度，返回str裁剪为len长度的字符串。
- 语法

  ```
  VARCHAR RPAD(VARCHAR str, INT len, VARCHAR pad)
  ```
- 参数说明
  - str：启始的字符串。
  - len：新的字符串的长度。
  - pad：需要重复补充的字符串。
- 示例
  - 测试语句

    ```
    SELECT
      RPAD("adc", 2, "hello"),
      RPAD("adc", -1, "hello"),
      RPAD("adc", 10, "hello");
    ```
  - 测试结果

    ```
    "ad",,"adchellohe"
    ```

SHA1

- 功能描述

  返回字符串expr的SHA1值。
- 语法

  ```
  STRING SHA1(STRING expr)
  ```
- 参数说明
  - expr：字符串。
- 示例
  - 测试语句

    ```
    SELECT SHA1("abc");
    ```
  - 测试结果

    ```
    "a9993e364706816aba3e25717850c26c9cd0d89d"
    ```

SHA256

- 功能描述

  返回字符串expr的SHA256值。
- 语法

  ```
  STRING SHA256(STRING expr)
  ```
- 参数说明
  - expr：字符串。
- 示例
  - 测试语句

    ```
    SELECT SHA256("abc");
    ```

 版权所有 © 华为技术有限公司

  - 测试结果
    ```
    "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
    ```

STRING_TO_ARRAY

- 功能描述

  将字符串value按delimiter分隔为字符串数组。

  说明

  delimiter使用的是java的正则表达式，若使用特殊字符则需要转义。

- 语法
  ```
  ARRAY[String] STRING_TO_ARRAY(STRING value, VARCHAR delimiter)
  ```
- 参数说明
  - value：字符串。
  - delimiter：分隔符。
- 示例
  - 测试语句
    ```
    SELECT
      string_to_array("127.0.0.1", "\\."),
      string_to_array("red-black-white-blue", "-");
    ```
  - 测试结果
    ```
    [127,0,0,1],[red,black,white,blue]
    ```

SUBSTRING

- 功能描述

  返回从指定位置开始的A的子字符串。起始位置从1开始。
  - 如果未指定len，则截取从位置start开始，到字符串结尾的子字符串。
  - 如果指定len，则截取从位置start开始，长度为len的子字符串。

  说明

  start从1开始，start为0时当1看待，为负数时表示从字符串末尾倒序计算位置。

- 语法
  ```
  VARCHAR SUBSTRING(STRING A FROM INT start)
  ```
  或
  ```
  VARCHAR SUBSTRING(STRING A FROM INT start FOR INT len)
  ```
- 参数说明
  - A：指定的字符串。
  - start：在字符串A中开始截取的位置。
  - len：截取的长度。
- 示例
  - 测试语句1
    ```
    SELECT SUBSTRING("123456" FROM 2);
    ```
  - 测试结果1
    ```
    "23456"
    ```
  - 测试语句2
    ```
    SELECT SUBSTRING("123456" FROM 2 FOR 4);
    ```
  - 测试结果2

 版权所有 © 华为技术有限公司

```
"2345"
```

TRIM

- 功能描述

  从B中除去字符串首尾/首位/末尾的A。默认情况下，首尾的A都被删除。

- 语法

  ```
  STRING TRIM( { BOTH | LEADING | TRAILING } STRING a FROM STRING b)
  ```

- 参数说明
  - a：字符串。
  - b：字符串。
- 示例
  - 测试语句

    ```
    SELECT TRIM(BOTH " " FROM "  hello world  ");
    ```

  - 测试结果

    ```
    "hello world"
    ```

UPPER

- 功能描述

  返回转换为大写字符的字符串。

- 语法

  ```
  VARCHAR UPPER(A)
  ```

- 参数说明
  - A：字符串。
- 示例
  - 测试语句

    ```
    SELECT UPPER("hello world");
    ```

  - 测试结果

    ```
    "HELLO WORLD"
    ```

2.9.3 时间函数

Flink SQL所支持的时间函数如**表2-42**所示。

函数说明

表 2-42 时间函数

函数	返回值	描述
DATE string	DATE	将日期字符串以“yyyy-MM-dd”的形式解析为SQL日期。
TIME string	TIME	将时间字符串以“HH:mm:ss”形式解析为SQL时间。
TIMESTAMP string	TIMESTAMP	将时间字符串转换为时间戳，时间字符串格式为：“yyyy-MM-dd HH:mm:ss.fff”。

版权所有 © 华为技术有限公司

函数	返回值	描述
INTERVAL string range	INTERVAL	interval表示时间间隔，有两种类型，一种为“yyyy-MM”，即保存年份和月份，精度到月份，它的Range可以为YEAR或者YEAR To Month；一种为天时间(“dd HH:mm:sss.fff'”)，用来保存天数、小时、分钟、秒和毫秒， 精度最低到毫秒， 他的range可以为DAY TO HOUR，DAY TO MINUTE，DAY TO SECOND或DAY TO milliseconds，比如range为DAY TO SECOND就表示天数、小时、分钟、秒的位置都有效，精度到秒，DAY TO MINUTE就表示精度到分钟。 例如： INTERVAL '10 00:00:00.004' DAY TO milliseconds表示间隔10天4毫秒。 INTERVAL '10' DAY表示间隔10天, INTERVAL '2-10' YEAR TO MONTH表示间隔2年10个月。
CURRENT_DATE	DATE	以UTC时区返回当前SQL日期。
CURRENT_TIME	TIME	以UTC时区返回当前SQL时间。
CURRENT_TIMESTAMP	TIMESTAMP	以UTC时区返回当前SQL时间戳。
LOCALTIME	TIME	返回当前时区的当前SQL时间。
LOCALTIMESTAMP	TIMESTAMP	返回当前时区的当前SQL时间戳。
EXTRACT(timeintervalunit FROM temporal)	INT	提取时间点的一部分或者时间间隔。以int类型返回该部分。 例如：提取日期“2006-06-05”中的日为5，则可以使用：EXTRACT(DAY FROM DATE "2006-06-05")
FLOOR(timepoint TO timeintervalunit)	TIME	向下对齐时间。 例如：FLOOR(TIME '12:44:31' TO MINUTE)按分钟对齐到12:44:00。
CEIL(timepoint TO timeintervalunit)	TIME	向上对齐时间。 例如：CEIL(TIME '12:44:31' TO MINUTE)按分钟对齐到12:45:00。
QUARTER(date)	INT	从SQL日期返回一年的四分之一。

版权所有 © 华为技术有限公司

函数	返回值	描述
(timepoint, temporal) OVERLAPS (timepoint, temporal)	BOOLEAN	确定两个时间间隔是否重叠。时间点和时间被转换成在两个时间点（开始，结束）定义的范围之内，该计算函数是***leftEnd >= rightStart && rightEnd >= leftStart***。当左边结束时间点大于等于右边开始时间点，且右边结束时间点大于等于左边开始时间点，则函数返回true值，否则返回false。 例如： ● 左边的结束时间点是“3:55:00”（2:55:00+1:00:00）大于右边的开始点是“3:30:00”；且右边的结束时间点是“5:30:00”（3:30:00+2:00:00）大于左边开始时间点“2:55:00”，则返回值为true。 (TIME '2:55:00', INTERVAL '1' HOUR) OVERLAPS (TIME '3:30:00', INTERVAL '2' HOUR) 返回值是true。 ● 左边的结束时间点是“10:00:00”小于右边的开始点是“10:15:00”；且右边的结束时间点是“13:15:00”（10:15:00+3:00:00）大于左边开始时间点“9:00:00”，则返回值为false。 (TIME '9:00:00', TIME '10:00:00') OVERLAPS (TIME '10:15:00', INTERVAL '3' HOUR) 返回值是false。
TO_TIMESTAMP(long expr)	TIMESTAMP	将时间戳转换为时间。 该函数入参数据类型仅支持BIGINT，不支持VARCHAR，STRING等其他数据类型。 例如，TO_TIMESTAMP(1628765159000)转换后值为：2021-08-12 18:45:59。
UNIX_TIMESTAMP	BIGINT	返回指定参数的时间戳，时间戳类型为BIGINT类型，单位为“秒”。 支持如下几种使用方法： ● UNIX_TIMESTAMP()：没有参数时，返回当前时间的时间戳。 ● UNIX_TIMESTAMP(STRING datestr)：包含一个参数时，返回参数所表示的时间戳，datestr格式必须为yyyy-MM-dd HH:mm:ss。 ● UNIX_TIMESTAMP(STRING datestr, STRING format)：包含两个参数时，第二个参数可以指定datestr的格式，返回第一个参数所表示的时间戳。

版权所有 © 华为技术有限公司

函数	返回值	描述
UNIX_TIMESTAMP_MS	BIGINT	返回指定参数的时间戳，时间戳类型为BIGINT类型，单位为“毫秒”。 支持如下几种使用方法： ● UNIX_TIMESTAMP_MS()：没有参数时，返回当前时间的时间戳。 ● UNIX_TIMESTAMP_MS(STRING datestr)：包含一个参数时，返回参数所表示的时间戳，datestr格式必须为yyyy-MM-dd HH:mm:ss.SSS。 ● UNIX_TIMESTAMP_MS(STRING datestr, STRING format)：包含两个参数时，第二个参数可以指定datestr的格式，返回第一个参数所表示的时间戳。

注意事项

无。

示例

```
insert into temp SELECT Date '2015-10-11'  FROM  OrderA;//返回日期
insert into temp1 SELECT Time '12:14:50'  FROM  OrderA;//返回时间
insert into temp2 SELECT Timestamp '2015-10-11 12:14:50'  FROM  OrderA;//返回时间戳
```

2.9.4 类型转换函数

语法格式

```
CAST(value AS type)
```

语法说明

类型强制转换。

注意事项

- 若输入为NULL，则返回NULL。
- Flink作业不支持使用CAST将“BIGINT”转换为“TIMESTAMP”，可以使用to_timestamp或者to_localtimestamp进行转换。

示例

将amount值转换成字符串，长度为转换后的实际长度，配置的长度无效。

```
insert into temp select cast(amount as VARCHAR(10)) from source_stream;
```


版权所有 © 华为技术有限公司

常用类型转换函数

表 2-43 常用类型转换函数

函数	说明
cast(v1 as varchar)	将v1转换为字符串类型，v1可以是数值类型，TIMESTAMP/DATE/TIME。
cast (v1 as int)	将v1转换为int, v1可以是数值类型或字符类。
cast(v1 as timestamp)	将v1转换为timestamp类型，v1可以是字符串或DATE/TIME。
cast(v1 as date)	将v1转换为date类型， v1可以是字符串或者TIMESTAMP。

- cast(v1 as varchar)
  - 测试语句：
    ```
    SELECT cast(content as varchar) FROM T1;
    ```
  - 测试数据和结果

    表 2-44 T1

content（INT）	varchar
5	"5"

- cast (v1 as int)
  - 测试语句：
    ```
    SELECT cast(content as int) FROM T1;
    ```
  - 测试数据和结果

    表 2-45 T1

content（STRING）	int
"5"	5

- cast(v1 as timestamp)
  - 测试语句：
    ```
    SELECT cast(content as timestamp) FROM T1;
    ```
  - 测试数据和结果

    表 2-46 T1

content（STRING）	timestamp
"2018-01-01 00:00:01"	1514736001000

版权所有 © 华为技术有限公司

- cast(v1 as date)
  - 测试语句：
    ```
    SELECT cast(content as date) FROM T1;
    ```
  - 测试数据和结果

    表 2-47 T1

content（TIMESTAMP）	date
1514736001000	"2018-01-01"

详细样例代码

```
/** source **/
CREATE
SOURCE STREAM car_infos (cast_int_to_varchar int, cast_String_to_int string,
case_string_to_timestamp string, case_timestamp_to_date timestamp) WITH (
  type = "dis",
  region = "xxxxx",
  channel = "dis-input",
  partition_count = "1",
  encode = "json",
  offset = "13",
  json_config =
"cast_int_to_varchar=cast_int_to_varchar;cast_String_to_int=cast_String_to_int;case_string_to_timestamp=cas
e_string_to_timestamp;case_timestamp_to_date=case_timestamp_to_date"

);
/** sink **/
CREATE
SINK STREAM cars_infos_out (cast_int_to_varchar varchar, cast_String_to_int
int, case_string_to_timestamp timestamp, case_timestamp_to_date date) WITH (
  type = "dis",
  region = "xxxxx",
  channel = "dis-output",
  partition_count = "1",
  encode = "json",
  offset = "4",
  json_config =
"cast_int_to_varchar=cast_int_to_varchar;cast_String_to_int=cast_String_to_int;case_string_to_timestamp=cas
e_string_to_timestamp;case_timestamp_to_date=case_timestamp_to_date",
  enable_output_null="true"
);
/** 统计car的静态信息 **/
INSERT
INTO
  cars_infos_out
SELECT
  cast(cast_int_to_varchar as varchar),
  cast(cast_String_to_int as int),
  cast(case_string_to_timestamp as timestamp),
  cast(case_timestamp_to_date as date)
FROM
  car_infos;
```

返回数据

```
{"case_string_to_timestamp":
1514736001000,"cast_int_to_varchar":"5","case_timestamp_to_date":"2018-01-01","cast_String_to_int":100}
```


版权所有 © 华为技术有限公司

2.9.5 聚合函数

聚合函数是从一组输入值计算一个结果。例如使用COUNT函数计算SQL查询语句返回的记录行数。聚合函数如表2-48所示。

示例数据：表T1

```
|score|
|81   |
|100  |
|60   |
|95   |
|86   |
```

常用聚合函数

表 2-48 常用聚合函数表

函数	返回值类型	描述
COUNT(*)	BIGINT	返回元组个数。
COUNT([ALL] expression...	BIGINT	返回表达式不为NULL的输入行数。对每个值的一个唯一实例使用DISTINCT。
AVG(numeric)	DOUBLE	返回所有输入值的数字的平均值（算术平均值）。
SUM(numeric)	DOUBLE	返回所有输入值之间的数值之和。
MAX(value)	DOUBLE	返回所有输入值的值的最大值。
MIN(value)	DOUBLE	返回所有输入值的值的最小值。
STDDEV_POP(value)	DOUBLE	返回所有输入值之间的数字字段的总体标准偏差。
STDDEV_SAMP(value)	DOUBLE	返回所有输入值之间的数字字段的样本标准偏差。
VAR_POP(value)	DOUBLE	返回所有输入值之间的数字字段的总体方差（总体标准偏差的平方）。
VAR_SAMP(value)	DOUBLE	返回所有输入值之间的数字字段的样本方差（样本标准偏差的平方）。

示例

- COUNT(*)
  - 测试语句：

    ```
    SELECT COUNT(score) FROM T1;
    ```

  - 测试数据和结果

版权所有 © 华为技术有限公司

表 2-49 T1

测试数据（score）	测试结果
81	5
100	
60	
95	
86	

- COUNT([ALL] expression | DISTINCT expression1 [, expression2]*)
  - 测试语句：
    ```
    SELECT COUNT(DISTINCT content ) FROM T1;
    ```
  - 测试数据和结果

表 2-50 T1

content (STRING)	测试结果
"hello1 "	2
"hello2 "	
"hello2"	
null	
86	

- AVG(numeric)
  - 测试语句：
    ```
    SELECT AVG(score) FROM T1;
    ```
  - 测试数据和结果

表 2-51 T1

测试数据（score）	测试结果
81	84.0
100	
60	
95	
86	

- SUM(numeric)
  - 测试语句：

 版权所有 © 华为技术有限公司

```
SELECT SUM(score) FROM T1;
```

- 测试数据和结果

表 2-52 T1

测试数据（score）	测试结果
81	422.0
100	
60	
95	
86	

- MAX(value)
  - 测试语句：
    ```
    SELECT MAX(score) FROM T1;
    ```
  - 测试数据和结果

表 2-53 T1

测试数据（score）	测试结果
81	100.0
100	
60	
95	
86	

- MIN(value)
  - 测试语句：
    ```
    SELECT MIN(score) FROM T1;
    ```
  - 测试数据和结果

表 2-54 T1

测试数据（score）	测试结果
81	60.0
100	
60	
95	
86	

版权所有 © 华为技术有限公司

- STDDEV_POP(value)
  - 测试语句：
    ```
    SELECT STDDEV_POP(score) FROM T1;
    ```
  - 测试数据和结果

    表 2-55 T1

测试数据（score）	测试结果
81	13.0
100	
60	
95	
86	

- STDDEV_SAMP(value)
  - 测试语句：
    ```
    SELECT STDDEV_SAMP(score) FROM T1;
    ```
  - 测试数据和结果

    表 2-56 T1

测试数据（score）	测试结果
81	15.0
100	
60	
95	
86	

- VAR_POP(value)
  - 测试语句：
    ```
    SELECT VAR_POP(score) FROM T1;
    ```
  - 测试数据和结果

    表 2-57 T1

测试数据（score）	测试结果
81	193.0
100	
60	
95	

 版权所有 © 华为技术有限公司

测试数据（score）	测试结果
86	

- VAR_SAMP(value)
  - 测试语句：
    ```
    SELECT VAR_SAMP(score) FROM T1;
    ```
  - 测试数据和结果

    表 2-58 T1

测试数据（score）	测试结果
81	241.0
100	
60	
95	
86	

2.9.6 表值函数

表值函数可以将一行转多行，一列转为多列，仅支持在JOIN LATERAL TABLE中使用。

表 2-59 表值函数表

函数	返回值类型	描述
split_cursor(value, delimiter)	cursor	将字符串value按delimiter分隔为多行字符串。

示例

输入一条记录("student1", "student2, student3")，输出两条记录("student1", "student2") 和 ("student1", "student3") 。

```
create source stream s1(attr1 string, attr2 string) with (.....);
insert into s2 select  attr1, b1 from s1 left join lateral table(split_cursor(attr2, ',')) as T(b1) on true;
```


版权所有 © 华为技术有限公司

2.9.7 其他函数

数组函数

表 2-60 数组函数表

函数	返回值类型	描述
CARDINALITY(ARRAY)	INT	返回数组的元素个数。
ELEMENT(ARRAY)	-	使用单个元素返回数组的唯一元素。 如果数组为空，则返回null。如果数组有多个元素，则抛出异常。

示例：

返回数组的元素个数为3。

```
insert into temp select CARDINALITY(ARRAY[TRUE, TRUE, FALSE]) from source_stream;
```

返回'HELLO WORLD'。

```
insert into temp select ELEMENT(ARRAY['HELLO WORLD']) from source_stream;
```

属性访问函数

表 2-61 属性访问函数表

函数	返回值类型	描述
tableName.compositeType.field	-	选择单个字段，通过名称访问Apache Flink复合类型（如Tuple，POJO等）的字段并返回其值。
tableName.compositeType.*	-	选择所有字段，将Apache Flink复合类型（如Tuple，POJO等）和其所有直接子类型转换为简单表示，其中每个子类型都是单独的字段。

2.10 自定义函数

概述

DLI支持三种自定义函数：

- UDF：自定义函数，支持一个或多个输入参数，返回一个结果值。
- UDTF：自定义表值函数，支持一个或多个输入参数，可返回多行多列。
- UDAF：自定义聚合函数，将多条记录聚合成一个值。

 版权所有 © 华为技术有限公司

说明

自定义函数仅能在独享队列中使用，不支持在共享队列中使用。

POM 依赖

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table_2.11</artifactId>
  <version>1.7.2</version>
  <scope>provided</scope>
</dependency>
<dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-streaming-java_2.11</artifactId>
      <version>1.7.2</version>
      <scope>provided</scope>
</dependency>
```

注意事项

- **暂不支持通过python写UDF、UDTF、UDAF自定义函数。**
- 如果使用IntelliJ IDEA工具对创建的自定义函数进行调试，则需要在IDEA上勾选：include dependencies with "Provided" scope，否则本地调试运行时会加载不到pom文件中的依赖包。

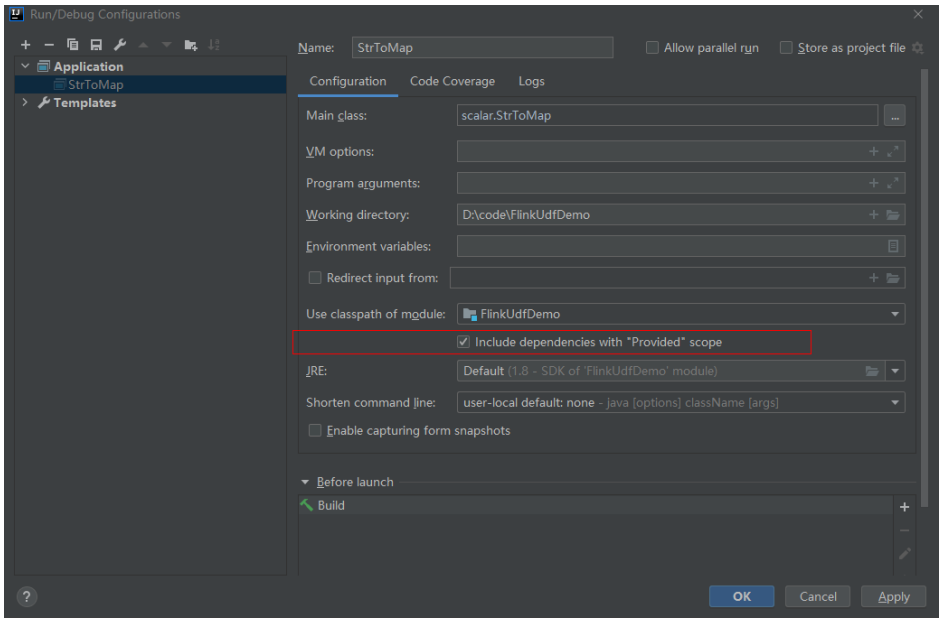
  具体操作以IntelliJ IDEA版本2020.2为例，参考如下：

  a. 在IntelliJ IDEA界面，选择调试的配置文件，单击“Edit Configurations”。

  b. 在“Run/Debug Configurations”界面，勾选：include dependencies with "Provided" scope。

版权所有 © 华为技术有限公司



c. 单击“OK”完成应用配置。

使用方式

1. 编写自定义函数代码。具体的代码样例可以参考**UDF**、**UDTF**或者**UDAF**。
2. 将写好的自定义函数编译并打成JAR包，并上传到OBS上。
3. 在DLI管理控制台的左侧导航栏中，单击“作业管理”>“Flink作业”，在需要编辑的Flink SQL作业对应的“操作”列中，单击“编辑”，进入作业编辑页面。
4. 在“运行参数”页签中，“所属队列”选择专享队列，会出现“UDF Jar”参数，在此处选择存放在OBS上的JAR文件，单击“保存”。

   说明

   在选择自定义函数Jar包之前需要将对应的jar包上传至已创建好的OBS桶中。

   选定JAR包以后，在SQL里添加UDF声明语句，就可以像普通函数一样使用了。

UDF

UDF函数需继承ScalarFunction函数，并实现eval方法。open函数及close函数可选。

编写代码示例

```
import org.apache.flink.table.functions.FunctionContext;
import org.apache.flink.table.functions.ScalarFunction;
public class UdfScalarFunction extends ScalarFunction {
  private int factor = 12;
  public UdfScalarFunction() {
    this.factor = 12;
  }
  /**
   * 初始化操作，可选
   * @param context
   */
  @Override
  public void open(FunctionContext context) {}
  /**
   * 自定义逻辑
   * @param s
   * @return
```

 版权所有 © 华为技术有限公司

```
  */
  public int eval(String s) {
    return s.hashCode() * factor;
  }
  /**
   * 可选
   */
  @Override
  public void close() {}
}
```

使用示例

```
CREATE FUNCTION udf_test AS 'com.xxx.udf.UdfScalarFunction';
INSERT INTO sink_stream select udf_test(attr) FROM source_stream;
```

UDTF

UDTF函数需继承TableFunction函数，并实现eval方法。open函数及close函数可选。如果需要UDTF返回多列，只需要将返回值声明成Tuple或Row即可。若使用Row，需要重载getResultType声明返回的字段类型。

编写代码示例

```
import org.apache.flink.api.common.typeinfo.TypeInformation;
import org.apache.flink.api.common.typeinfo.Types;
import org.apache.flink.table.functions.FunctionContext;
import org.apache.flink.table.functions.TableFunction;
import org.apache.flink.types.Row;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class UdfTableFunction extends TableFunction<Row> {
  private Logger log = LoggerFactory.getLogger(TableFunction.class);
  /**
   * 初始化操作，可选
   * @param context
   */
  @Override
  public void open(FunctionContext context) {}
  public void eval(String str, String split) {
    for (String s : str.split(split)) {
      Row row = new Row(2);
      row.setField(0, s);
      row.setField(1, s.length());
      collect(row);
    }
  }
  /**
   * 函数返回类型声明
   * @return
   */
  @Override
  public TypeInformation<Row> getResultType() {
  return Types.ROW(Types.STRING, Types.INT);
  }
  /**
   * 可选
   */
  @Override
  public void close() {}
 }
```

使用示例

UDTF支持CROSS JOIN和LEFT JOIN，在使用UDTF时需要带上 LATERAL 和TABLE 两个关键字。

版权所有 © 华为技术有限公司

- CROSS JOIN：对于左表的每一行数据，假设UDTF不产生输出，则这一行不进行输出。
- LEFT JOIN：对于左表的每一行数据，假设UDTF不产生输出，这一行仍会输出，UDTF相关字段用null填充。

```
CREATE FUNCTION udtf_test AS 'com.xxx.udf.TableFunction';
// CROSS JOIN
INSERT INTO sink_stream select subValue, length FROM source_stream, LATERAL
TABLE(udtf_test(attr, ',')) as T(subValue, length);
// LEFT JOIN
INSERT INTO sink_stream select subValue, length FROM source_stream LEFT JOIN LATERAL
TABLE(udtf_test(attr, ',')) as T(subValue, length) ON TRUE;
```

UDAF

UDAF函数需继承AggregateFunction函数。首先需要创建一个用来存储计算结果的Accumulator，如示例里的WeightedAvgAccum。

编写代码示例

```
public class WeightedAvgAccum {
public long sum = 0;
public int count = 0;
}
```

```
import org.apache.flink.table.functions.AggregateFunction;
import java.util.Iterator;
/**
* 第一个类型变量为聚合函数返回的类型，第二个类型变量为Accumulator类型
* Weighted Average user-defined aggregate function.
*/
public class UdfAggFunction extends AggregateFunction<Long, WeightedAvgAccum> {
  // 初始化Accumulator
  @Override
  public WeightedAvgAccum createAccumulator() {
    return new WeightedAvgAccum();
  }
  // 返回Accumulator存储的中间计算值
  @Override
  public Long getValue(WeightedAvgAccum acc) {
    if (acc.count == 0) {
       return null;
    } else {
      return acc.sum / acc.count;
 }
}
// 根据输入更新中间计算值
public void accumulate(WeightedAvgAccum acc, long iValue) {
acc.sum += iValue;
acc.count += 1;
}
// Restract撤回操作，和accumulate操作相反
public void retract(WeightedAvgAccum acc, long iValue) {
acc.sum -= iValue;
acc.count -= 1;
}
// 合并多个accumulator值
public void merge(WeightedAvgAccum acc, Iterable<WeightedAvgAccum> it) {
Iterator<WeightedAvgAccum> iter = it.iterator();
while (iter.hasNext()) {
WeightedAvgAccum a = iter.next();
acc.count += a.count;
acc.sum += a.sum;
}
}
// 重置中间计算值
```


版权所有 © 华为技术有限公司

```
public void resetAccumulator(WeightedAvgAccum acc) {
acc.count = 0;
acc.sum = 0L;
}
}
```

使用示例

```
CREATE FUNCTION udaf_test AS 'com.xxx.udf.UdfAggFunction';
INSERT INTO sink_stream SELECT udaf_test(attr2) FROM source_stream GROUP BY attr1;
```

2.11 地理函数

函数说明

基本地理空间几何元素介绍说明如表2-62所示。

表 2-62 基本地理空间几何元素表

地理空间几何元素（统称geometry）	说明	举例
ST_POINT(latitude, longitude)	地理点，包含经度和维度两个信息。	ST_POINT(1.12012, 1.23401)
ST_LINE(array[point1...pointN])	地理线，由多个地理点（ST_POINT）按顺序连接成的折线或直线。	ST_LINE(ARRAY[ST_POINT(1.12, 2.23), ST_POINT(1.13, 2.44), ST_POINT(1.13, 2.44)])
ST_POLYGON(array[point1...point1])	地理多边形，由首尾相同的多个地理点（ST_POINT）按顺序连线围成的封闭多边形区域。	ST_POLYGON(ARRAY[ST_POINT(1.0, 1.0), ST_POINT(2.0, 1.0), ST_POINT(2.0, 2.0), ST_POINT(1.0, 1.0)])
ST_CIRCLE(point, radius)	地理圆形，由圆心地理点（ST_POINT）和半径构成的地理圆形区域。	ST_CIRCLE(ST_POINT(1.0, 1.0), 1.234)

用户可以以基本地理空间几何元素为基础，构造复杂的地理空间几何元素，具体的变换方法见表2-63。

版权所有 © 华为技术有限公司

表 2-63 基于基本地理空间几何元素构造复杂几何元素的变换表

变换方法	说明	举例
ST_BUFFER(geometry, distance)	创建一个环绕包含给定地理空间几何元素的多边形，并以给定距离作为环绕距离，通常使用该函数构造一定宽度的公路范围用于偏航检测。	ST_BUFFER(ST_LINE(ARRAY[ST_POINT(1.12, 2.23), ST_POINT(1.13, 2.44), ST_POINT(1.13, 2.44)]),1.0)
ST_INTERSECTION(geometry, geometry)	创建一个多边形，其范围为给定的两个地理空间几何元素的交叠区域。	ST_INTERSECTION(ST_CIRCLE(ST_POINT(1.0, 1.0), 2.0), ST_CIRCLE(ST_POINT(3.0, 1.0), 1.234))
ST_ENVELOPE(geometry)	创建一个包含给定的地理空间几何元素的最小矩形。	ST_ENVELOPE(ST_CIRCLE(ST_POINT(1.0, 1.0), 2.0))

DLI提供丰富的对地理空间几何元素的操作和位置判断函数，具体的SQL标量函数介绍说明见**表2-64**。

表 2-64 SQL 标量函数表

函数	返回值	说明
ST_DISTANCE(point_1, point_2)	DOUBLE	计算两个地理点之间的欧几里得距离。 示例如下： Select ST_DISTANCE(ST_POINT(x1, y1), ST_POINT(x2, y2)) FROM input
ST_GEODESIC_DISTANCE(point_1, point_2)	DOUBLE	计算两个地理点之间的测地距离，即两个地理点之间地表最短路径距离。 示例如下： Select ST_GEODESIC_DISTANCE(ST_POINT(x1, y1), ST_POINT(x2, y2)) FROM input
ST_PERIMETER(polygon)	DOUBLE	计算多边形的周长。 示例如下： Select ST_PERIMETER(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]) FROM input

版权所有 © 华为技术有限公司

函数	返回值	说明
ST_AREA(polygon)	DOUBLE	计算多边形区域的面积。 示例如下： Select ST_AREA(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]) FROM input
ST_OVERLAPS(polygon_1, polygon_2)	BOOLEAN	判断一个多边形是否与另一个多边形有重叠区域。 示例如下： SELECT ST_OVERLAPS(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]), ST_POLYGON(ARRAY[ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23), ST_POINT(x21, y21)])) FROM input
ST_INTERSECT(line1, line2)	BOOLEAN	检查两条线段是否相互交叉，而非线条所在的直线是否交叉。 示例如下： SELECT ST_INTERSECT(ST_LINE(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12)]), ST_LINE(ARRAY[ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23)])) FROM input
ST_WITHIN(point, polygon)	BOOLEAN	一个点是否包含在几何体（多边形或圆形）内。 示例如下： SELECT ST_WITHIN(ST_POINT(x11, y11), ST_POLYGON(ARRAY[ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23), ST_POINT(x21, y21)])) FROM input
ST_CONTAINS(polygon_1, polygon_2)	BOOLEAN	判断第一个几何体是否包含第二个几何体。 示例如下： SELECT ST_CONTAINS(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]), ST_POLYGON(ARRAY[ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23), ST_POINT(x21, y21)])) FROM input

版权所有 © 华为技术有限公司

函数	返回值	说明
ST_COVERS(polygon_1, polygon_2)	BOOLEAN	第一个几何体是否覆盖第二个几何体。与ST_CONTAINS相似，但在边界重叠情况下ST_COVER判断为TRUE，ST_CONTAINS判断为FALSE。 示例如下： SELECT ST_COVERS(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]), ST_POLYGON([ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23), ST_POINT(x21, y21)])) FROM input
ST_DISJOINT(polygon_1, polygon_2)	BOOLEAN	判断一个多边形是否与另一个多边形不相交（不重叠）。 示例如下： SELECT ST_DISJOINT(ST_POLYGON(ARRAY[ST_POINT(x11, y11), ST_POINT(x12, y12), ST_POINT(x11, y11)]), ST_POLYGON(ARRAY[ST_POINT(x21, y21), ST_POINT(x22, y22), ST_POINT(x23, y23), ST_POINT(x21, y21)])) FROM input

地理函数的基准坐标系标准为全球通用的GPS坐标系标准WGS84，GPS坐标不能直接在百度地图（BD09标准）或者google地图（GCJ02标准）上使用，会有偏移现象，为了在不同地理坐标系之间切换，DLI提供了坐标系转换的一系列函数，并且还提供地理距离与米之间的转换函数。详见**表2-65**。

表 2-65 地理坐标系转换函数与距离单位转换函数表

函数	返回值	说明
WGS84_TO_BD09(geometry)	对应的百度地图坐标系地理空间几何元素	将GPS坐标系下的地理空间几何元素转换成百度地图坐标系下对应的地理空间几何元素。示例如下： WGS84_TO_BD09(ST_CIRCLE(ST_POINT(x, y), r))
WGS84_TO_CJ02(geometry)	对应的Google地图坐标系地理空间几何元素	将GPS坐标系下的地理空间几何元素转换成Google地图坐标系下对应的地理空间几何元素。示例如下： WGS84_TO_CJ02(ST_CIRCLE(ST_POINT(x, y), r))

版权所有 © 华为技术有限公司

函数	返回值	说明
BD09_TO_WGS84(geometry)	对应的GPS坐标系地理空间几何元素	将百度地图坐标系下的地理空间几何元素转换成GPS坐标系下对应的地理空间几何元素。示例如下： BD09_TO_WGS84(ST_CIRCLE(ST_POINT(x, y), r))
BD09_TO_CJ02(geometry)	对应的Google地图坐标系地理空间几何元素	将百度地图坐标系下的地理空间几何元素转换成Google地图坐标系下对应的地理空间几何元素。示例如下： BD09_TO_CJ02(ST_CIRCLE(ST_POINT(x, y), r))
CJ02_TO_WGS84(geometry)	对应的GPS坐标系地理空间几何元素	将Google地图坐标系下的地理空间几何元素转换成GPS坐标系下对应的地理空间几何元素。示例如下： CJ02_TO_WGS84(ST_CIRCLE(ST_POINT(x, y), r))
CJ02_TO_BD09(geometry)	对应的百度地图坐标系地理空间几何元素	将Google地图坐标系下的地理空间几何元素转换成百度地图坐标系下对应的地理空间几何元素。示例如下： CJ02_TO_BD09(ST_CIRCLE(ST_POINT(x, y), r))
DEGREE_TO_METER(distance)	DOUBLE	将地理函数的距离数值转换成以“米”为单位的数值。示例如下（以米为单位计算地理三边形周长）： DEGREE_TO_METER(ST_PERIMETER(ST_POLYGON(ARRAY[ST_POINT(x1,y1), ST_POINT(x2,y2), ST_POINT(x3,y3), ST_POINT(x1,y1)])))

版权所有 © 华为技术有限公司

函数	返回值	说明
METER_TO_DEGREE(numerical_value)	DOUBLE	将以“米”为单位的数值转换成地理函数可计算的距离单位数值。示例如下（画出以指定地理点为圆心，半径1公里的圆）： ST_CIRCLE(ST_POINT(x,y), METER_TO_DEGREE(1000))

DLI还提供了基于窗口的SQL地理聚合函数用于SQL逻辑涉及窗口和聚合的场景。详见表2-66的介绍说明。

表 2-66 时间相关 SQL 地理聚合函数表

函数	说明	举例
AGG_DISTANCE(point)	距离聚合函数，用于计算窗口内所有相邻地理点的距离总和。	SELECT AGG_DISTANCE(ST_POINT(x,y)) FROM input GROUP BY HOP(rowtime, INTERVAL '1' HOUR, INTERVAL '1' DAY)
AVG_SPEED(point)	平均速度聚合函数，用于计算窗口内所有地理点组成的移动轨迹的平均速度，单位为“米/秒”。	SELECT AVG_SPEED(ST_POINT(x,y)) FROM input GROUP BY TUMBLE(proctime, INTERVAL '1' DAY)

注意事项

无。

示例

偏航检测样例：

```
INSERT INTO yaw_warning
SELECT "The car is yawing"
FROM driver_behavior
WHERE NOT ST_WITHIN(ST_POINT(cast(Longitude as DOUBLE), cast(Latitude as DOUBLE)),
ST_BUFFER(ST_LINE(ARRAY[ST_POINT(34.585555,105.725221),ST_POINT(34.586729,105.735974),ST_POINT(34.586492,105.740538),ST_POINT(34.586388,105.741651),ST_POINT(34.586135,105.748712),ST_POINT(34.588691,105.74997)]),0.001));
```


版权所有 © 华为技术有限公司

IP 地理函数

说明

当前仅支持IPV4的IP地址。

表 2-67 IP 地理函数表

函数	返回值	说明
IP_TO_COUNTRY	STRING	获取IP地址所在的国家名称。
IP_TO_PROVINCE	STRING	获取IP地址所在的省份。 用法说明： ● IP_TO_PROVINCE(STRING ip)：返回IP地址所在的省份。 ● IP_TO_PROVINCE(STRING ip, STRING lang)：以指定语言返回IP地址所在的省份。 **说明** ● 当IP无法被解析到省份时，返回该IP所属的国家。当IP无法被解析时，返回“未知”。 ● 函数返回的省份名称均为简称。
IP_TO_CITY	STRING	获取IP地址所在的城市名称。 **说明** 当IP无法被解析到城市时，返回该IP所属的省份或者国家。当IP无法被解析时，返回“未知”。
IP_TO_CITY_GEO	STRING	获取IP地址所在城市的经纬度，格式为“纬度,经度”。 用法说明： IP_TO_CITY_GEO(STRING ip)：返回IP所在城市的经纬度。

2.12 SELECT

SELECT

语法格式

```
SELECT [ ALL | DISTINCT ] { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
  [ HAVING booleanExpression ]
```

语法说明

SELECT语句用于从表中选取数据或者插入常量数据。

注意事项

版权所有 © 华为技术有限公司

- 所查询的表必须是已经存在的表，否则会出错。
- WHERE关键字指定查询的过滤条件，过滤条件中支持算术运算符，关系运算符，逻辑运算符。
- GROUP BY指定分组的字段，可以单字段分组，也可以多字段分组。

示例

找出数量超过3的订单。

```
insert into temp SELECT  * FROM Orders WHERE units > 3;
```

插入一组常量数据。

```
insert into temp select 'Lily’, ‘male’, ‘student’, 17;
```

WHERE 过滤子句

语法格式

```
SELECT  { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
```

语法说明

利用WHERE子句过滤查询结果。

注意事项

- 所查询的表必须是已经存在的，否则会出错。
- WHERE条件过滤，将不满足条件的记录过滤掉，返回满足要求的记录。

示例

找出数量超过3并且小于10的订单。

```
insert into temp SELECT  * FROM Orders
  WHERE units > 3 and units < 10;
```

HAVING 过滤子句

功能描述

利用HAVING子句过滤查询结果。

语法格式

```
SELECT [ ALL | DISTINCT ]  { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
  [ HAVING booleanExpression ]
```

语法说明

HAVING：一般与GROUP BY合用，先通过GROUP BY进行分组，再在HAVING子句中进行过滤，HAVING子句支持算术运算，聚合函数等。

注意事项

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。

版权所有 © 华为技术有限公司

示例

根据字段name对表student进行分组，再按组将score最大值大于95的记录筛选出来。

```
insert into temp SELECT name, max(score) FROM student
  GROUP BY name
  HAVING max(score) >95
```

按列 GROUP BY

功能描述

按列进行分组操作。

语法格式

```
SELECT [ ALL | DISTINCT ]  { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
```

语法说明

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。
- 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。

注意事项

无。

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。

```
insert into temp SELECT name,score, max(score) FROM student
  GROUP BY name,score;
```

用表达式 GROUP BY

功能描述

按表达式对流进行分组操作。

语法格式

```
SELECT [ ALL | DISTINCT ]  { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
```

语法说明

groupItem：可以是单字段，多字段，也可以是字符串函数等调用，不能是聚合函数。

注意事项

无。

示例

版权所有 © 华为技术有限公司

先利用substring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。

```
insert into temp SELECT substring(name,6),count(name) FROM student
  GROUP BY substring(name,6);
```

GROUP BY 中使用 HAVING 过滤

功能描述

利用HAVING子句在表分组后实现过滤。

语法格式

```
SELECT [ ALL | DISTINCT ]  { * | projectItem [, projectItem ]* }
  FROM tableExpression
  [ WHERE booleanExpression ]
  [ GROUP BY { groupItem [, groupItem ]* } ]
  [ HAVING booleanExpression ]
```

语法说明

HAVING：一般与GROUP BY合用，先通过GROUP BY进行分组，再在HAVING子句中进行过滤。

注意事项

- 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中除聚合函数外所使用的字段必须是GROUP BY中出现的字段。
- HAVING子句支持算术运算，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。

```
insert into temp SELECT num, max(price*amount) FROM transactions
  WHERE time > '2016-06-01'
  GROUP BY num
  HAVING max(price*amount)>5000;
```

UNION

语法格式

```
query UNION [ ALL ] query
```

语法说明

UNION返回多个查询结果的并集。

注意事项

- 集合运算是以一定条件将表首尾相接，所以其中每一个SELECT语句返回的列数必须相同，列的类型一定要相同，列名不一定要相同。
- UNION默认是去重的，UNION ALL是不去重的。

示例

版权所有 © 华为技术有限公司

输出Orders1和Orders2的并集，不包含重复记录。

```
insert into temp SELECT  * FROM Orders1
 UNION SELECT  * FROM Orders2;
```

2.13 条件表达式

CASE 表达式

语法格式

```
CASE value WHEN value1 [, value11 ]* THEN result1
 [ WHEN valueN [, valueN1 ]* THEN resultN ]* [ ELSE resultZ ]
 END
```

或

```
CASE WHEN condition1 THEN result1
 [ WHEN conditionN THEN resultN ]* [ ELSE resultZ ]
 END
```

语法说明

- 当value值为value1则返回result1，都不满足则返回resultZ，若没有else语句，则返回null。
- 当condition1为true时返回result1，都不满足则返回resultZ，若没有else语句，则返回null。

注意事项

- 所有result的类型都必须一致。
- 所有condition类型都必须是布尔类型。
- 当没有满足的分支时，若指定else语句，则返回else的值，若没有else语句，则返回null。

示例

当units等于5时返回1，否则返回0。

示例1：

```
insert into temp SELECT  CASE units WHEN 5 THEN 1 ELSE 0 END FROM Orders;
```

示例2：

```
insert into temp SELECT CASE WHEN units = 5 THEN 1 ELSE 0 END FROM Orders;
```

NULLIF 表达式

语法格式

```
NULLIF(value, value)
```

语法说明

如果值相同，则返回NULL。 例如，NULLIF（5，5）返回NULL; NULLIF（5，0）返回5。

注意事项

无。

版权所有 © 华为技术有限公司

示例

当units等于3时返回null，否则返回units。

```
insert into temp SELECT  NULLIF(units, 3) FROM Orders;
```

COALESCE 表达式

语法格式

```
COALESCE(value, value [, value ]* )
```

语法说明

返回从左到右第一个不为NULL的参数的值。

注意事项

所有value的类型都必须一致。

示例

返回5。

```
insert into temp SELECT  COALESCE(NULL, 5) FROM Orders;
```

2.14 窗口

GROUP WINDOW

语法说明

Group Window定义在GROUP BY里，每个分组只输出一条记录，包括以下几种：

说明

- time_attr可以设置processing-time或者event-time。
  - time_attr设置为event-time时参数类型为bigint或者timestamp类型。
  - time_attr设置为processing-time时无需指定类型。
- interval设置窗口周期。

- 分组函数

表 2-68 分组函数表

函数名	说明
TUMBLE(time_attr, interval)	跳跃窗口。
HOP(time_attr, interval, interval)	拓展的跳跃窗口(等价于datastream的滑动窗口)，可以分别设置输出触发周期和窗口周期。
SESSION(time_attr, interval)	会话窗口，interval表示多长时间没有记录则关闭窗口。

版权所有 © 华为技术有限公司

- 窗口函数

  表 2-69 窗口函数表

函数名	说明
TUMBLE_START(time_attr, interval)	返回跳跃窗口开始时间。为UTC时区。
TUMBLE_END(time_attr, interval)	返回跳跃窗口结束时间。为UTC时区。
HOP_START(time_attr, interval, interval)	返回拓展的跳跃窗口开始时间。为UTC时区。
HOP_END(time_attr, interval, interval)	返回拓展的跳跃窗口结束时间。为UTC时区。
SESSION_START(time_attr, interval)	返回会话窗口开始时间。为UTC时区。
SESSION_END(time_attr, interval)	返回会话窗口结束时间。为UTC时区。

示例

```
// 每天计算SUM（金额）（事件时间）。
insert into temp SELECT name,
    TUMBLE_START(ts, INTERVAL '1' DAY) as wStart,
    SUM(amount)
    FROM Orders
    GROUP BY TUMBLE(ts, INTERVAL '1' DAY), name;

// 每天计算SUM（金额）（处理时间）。
insert into temp SELECT name,
    SUM(amount)
    FROM Orders
    GROUP BY TUMBLE(proctime, INTERVAL '1' DAY), name;

// 每个小时计算事件时间中最近24小时的SUM（数量）。
insert into temp SELECT product,
    SUM(amount)
    FROM Orders
    GROUP BY HOP(ts, INTERVAL '1' HOUR, INTERVAL '1' DAY), product;

// 计算每个会话的SUM（数量），间隔12小时的不活动间隙（事件时间）。
insert into temp SELECT name,
    SESSION_START(ts, INTERVAL '12' HOUR) AS sStart,
    SESSION_END(ts, INTERVAL '12' HOUR) AS sEnd,
    SUM(amount)
    FROM Orders
    GROUP BY SESSION(ts, INTERVAL '12' HOUR), name;
```

OVER WINDOW

Over Window与Group Window区别在于Over window每一行都会输出一条记录。

语法格式

```
OVER (
  [PARTITION BY partition_name]
```


版权所有 © 华为技术有限公司

```
 ORDER BY proctime|rowtime(ROWS number PRECEDING) |(RANGE (BETWEEN INTERVAL '1' SECOND
PRECEDING AND CURRENT ROW | UNBOUNDED preceding))
)
```

语法说明

表 2-70 参数说明

参数	参数说明
PARTITION BY	指定分组的主键，每个分组各自进行计算。
ORDER BY	指定数据按processing time或event time作为时间戳。
ROWS	个数窗口。
RANGE	时间窗口。

注意事项

- 同一select里所有聚合函数定义的窗口都必须保持一致。
- 当前Over窗口只支持前向计算(preceding)，不支持following计算。
- 必须指定ORDER BY 按processing time或event time。
- 不支持对常量做聚合操作，如sum(2)。

示例

```
// 计算从规则启动到目前为止的计数及总和(in proctime)
insert into temp SELECT name,
   count(amount) OVER (PARTITION BY name ORDER BY proctime RANGE UNBOUNDED preceding) as
cnt1,
   sum(amount) OVER (PARTITION BY name ORDER BY proctime RANGE UNBOUNDED preceding) as cnt2
   FROM Orders;

// 计算最近四条记录的计数及总和(in proctime)
insert into temp SELECT name,
   count(amount) OVER (PARTITION BY name ORDER BY proctime ROWS BETWEEN 4 PRECEDING AND
CURRENT ROW) as cnt1,
   sum(amount) OVER (PARTITION BY name ORDER BY proctime ROWS BETWEEN 4 PRECEDING AND
CURRENT ROW) as cnt2
   FROM Orders;

// 计算最近60s的计数及总和(in eventtime),基于事件时间处理，事件时间为Orders中的timeattr字段。
insert into temp SELECT name,
   count(amount) OVER (PARTITION BY name ORDER BY timeattr RANGE BETWEEN INTERVAL '60'
SECOND PRECEDING AND CURRENT ROW) as cnt1,
   sum(amount) OVER (PARTITION BY name ORDER BY timeattr RANGE BETWEEN INTERVAL '60' SECOND
PRECEDING AND CURRENT ROW) as cnt2
   FROM Orders;
```

2.15 流表 JOIN

流与表进行连接操作，从表中查询并补全流字段。目前支持连接RDS表和DCS服务的Redis表。通过ON条件描述查询的Key，并补全表结构的Value字段。

RDS表的数据定义语句请参见**创建RDS表**。

版权所有 © 华为技术有限公司

Redis表的数据定义语句请参见**创建Redis表**。

语法格式

```
FROM tableExpression JOIN tableExpression
  ON value11 = value21 [ AND value12 = value22]
```

语法说明

ON条件中只支持表属性等值查询，当存在二级Key时（Redis值类型为HASH情况下），需要AND表达Key和Hash Key等值查询。

注意事项

无。

示例

将车辆信息输入流与车辆价格表做等值连接后，获取车辆价格信息并填入车辆信息输出流后输出。

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_detail_type STRING
)
WITH (
  type = "dis",
  region = "",
  channel = "dliinput",
  partition_count = "1",
  encode = "csv",
  field_delimiter = ","
);

/** 创建数据维表，用于和输入流连接，实现字段回填
  *
  * 根据实际情况修改以下选项：
  * value_type：redis的键值对应值类型，支持STRING、HASH、SET、ZSET、LIST，其中HASH类型需要指定hash_key_column作为二层主键，集合类型将用逗号拼接所有查询出来的值
  * key_column：维表主键对应的列名
  * hash_key_column：当redis的键值对应值类型为HASH时，HASHMAP的KEY对应的列名，当值类型非HASH时，无需指定改配置
  * cluster_address：DCS服务redis集群地址
  * password：DCS服务redis集群密码
  **/
CREATE TABLE car_price_table (
  car_brand STRING,
  car_detail_type STRING,
  car_price STRING
)
WITH (
  type = "dcs_redis",
  value_type = "hash",
  key_column = "car_brand",
  hash_key_column = "car_detail_type",
  cluster_address = "192.168.1.238:6379",
  password = "xxxxxxxx"
);

CREATE SINK STREAM audi_car_owner_info (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
```

 版权所有 © 华为技术有限公司

```
  car_detail_type STRING,
  car_price STRING
)
WITH (
  type = "dis",
  region = "",
  channel = "dlioutput",
  partition_key = "car_owner",
  encode = "csv",
  field_delimiter = ","
);

INSERT INTO audi_car_owner_info
SELECT t1.car_id, t1.car_owner, t2.car_brand, t1.car_detail_type, t2.car_price
FROM car_infos as t1 join car_price_table as t2
ON t2.car_brand = t1.car_brand and t2.car_detail_type = t1.car_detail_type
WHERE t1.car_brand = "audi";
```

2.16 配置时间模型

Flink中主要提供两种时间模型：Processing Time和Event Time。

DLI允许在创建Source Stream和Temp Stream的时候指定时间模型以便在后续计算中使用。

配置 Processing Time

Processing Time是指系统时间，与数据本身的时间戳无关，即在Flink算子内计算完成的时间。

语法格式

```
CREATE SOURCE STREAM stream_name(...) WITH (...)
TIMESTAMP BY proctime.proctime;
CREATE TEMP STREAM stream_name(...)
TIMESTAMP BY proctime.proctime;
```

语法说明

设置Processing Time只需在timestamp by后配置proctime.proctime即可，后续可以直接使用proctime字段。

注意事项

无。

示例

```
CREATE SOURCE STREAM student_scores (
  student_number STRING, /* 学号 */
  student_name STRING, /* 姓名 */
  subject STRING, /* 学科 */
  score INT /* 成绩 */
)
WITH (
  type = "dis",
  region = "",
  channel = "dliinput",
  partition_count = "1",
  encode = "csv",
  field_delimiter=","
)TIMESTAMP BY proctime.proctime;

INSERT INTO score_greate_90
```


版权所有 © 华为技术有限公司

```
SELECT student_name, sum(score) over (order by proctime RANGE UNBOUNDED PRECEDING)
FROM student_scores;
```

配置 Event Time

Event Time是指事件产生的时间，即数据产生时自带时间戳。

语法格式

```
CREATE SOURCE STREAM stream_name(...) WITH (...)
TIMESTAMP BY {attr_name}.rowtime
SET WATERMARK (RANGE {time_interval} | ROWS {literal}, {time_interval});
```

语法说明

设置Event Time需要选定流中的某一个属性来作为时间戳，同时需要设置Watermark策略。

由于网络等原因，有时会导致乱序的产生；对于迟来的数据，需要Watermark来保证一个特定的时间后去触发Window进行计算。Watermark主要是用来处理乱序数据，流处理从事件产生，到发送到DLI服务，中间有一个过程。

Watermark有两种设置策略：

- 按时间周期
  ```
  SET WATERMARK(range interval {time_unit}, interval {time_unit})
  ```
- 按事件个数
  ```
  SET WATERMARK(rows literal, interval {time_unit})
  ```

说明

一个逗号表示一个参数，第一个参数表示Watermark发送周期，第二个参数表示允许最大延迟时间。

注意事项

无。

示例

- time2事件产生时间开始，每10s发送一次watermark，事件最大允许延迟时间为20s。
  ```
  CREATE SOURCE STREAM student_scores (
    student_number STRING, /* 学号 */
    student_name STRING, /* 姓名 */
    subject STRING, /* 学科 */
    score INT, /* 成绩 */
    time2 TIMESTAMP
  )
  WITH (
    type = "dis",
    region = "",
    channel = "dliinput",
    partition_count = "1",
    encode = "csv",
    field_delimiter=","
  )
  TIMESTAMP BY time2.rowtime
  SET WATERMARK (RANGE interval 10 second, interval 20 second);

  INSERT INTO score_greate_90
  SELECT student_name, sum(score) over (order by time2 RANGE UNBOUNDED PRECEDING)
  FROM student_scores;
  ```


版权所有 © 华为技术有限公司

- 每收到10个数据发送一次watermark，事件最大允许延迟时间为20s。

```
CREATE SOURCE STREAM student_scores (
  student_number STRING, /* 学号 */
  student_name STRING, /* 姓名 */
  subject STRING, /* 学科 */
  score INT, /* 成绩 */
  time2 TIMESTAMP
)
WITH (
  type = "dis",
  region = "",
  channel = "dliinput",
  partition_count = "1",
  encode = "csv",
  field_delimiter=","
)
TIMESTAMP BY time2.rowtime
SET WATERMARK (ROWS 10, interval 20 second);

INSERT INTO score_greate_90
SELECT student_name, sum(score) over (order by time2 RANGE UNBOUNDED PRECEDING)
FROM student_scores;
```

2.17 CEP 模式匹配

复杂事件处理（Complex Event Process，简称CEP）用来检测无尽数据流中的复杂模式，拥有从不同的数据行中辨识查找模式的能力。模式匹配是复杂事件处理的一个强大援助。

例子包括受一系列事件驱动的各种业务流程，例如在安全应用中侦测异常行为；在金融应用中查找价格、交易量和其他行为的模式。其他常见的用途如欺诈检测应用和传感器数据的分析等。

语法格式

```
MATCH_RECOGNIZE (
    [ PARTITION BY expression [, expression ]* ]
    [ ORDER BY orderItem [, orderItem ]* ]
    [ MEASURES measureColumn [, measureColumn ]* ]
    [ ONE ROW PER MATCH | ALL ROWS PER MATCH ]
    [ AFTER MATCH
          ( SKIP TO NEXT ROW
          | SKIP PAST LAST ROW
          | SKIP TO FIRST variable
          | SKIP TO LAST variable
          | SKIP TO variable )
    ]
    PATTERN ( pattern )
    [ WITHIN intervalLiteral ]
    DEFINE variable AS condition [, variable AS condition ]*
) MR
```

说明

SQL中的模式匹配是用MATCH_RECOGNIZE子句执行。MATCH_RECOGNIZE子句执行如下任务：

- 使用PARTITION BY 和ORDER BY子句对MATCH_RECOGNIZE子句中的数据进行逻辑分区和排序。
- 使用PATTERN子句来定义要查找的数据行的模式。这些模式使用规则表达式语法。
- 使用DEFINE子句指定PATTERN模式变量所需的逻辑条件。
- 使用MEASURES子句定义度量，这是一些可在SQL查询的其他部分所使用的表达式。

 版权所有 © 华为技术有限公司

语法说明

表 2-71 语法说明

参数	是否必选	说明
PARTITION BY	否	将数据行进行逻辑上的分组。
ORDER BY	否	在分区中对数据行进行逻辑排序。
[ONE ROW | ALL ROWS] PER MATCH	否	为每个匹配选择输出汇总或者明细。 ● ONE ROW PER MATCH：每次检测到完整的匹配后进行汇总输出。 ● ALL ROWS PER MATCH：检测到完整的匹配后会把匹配过程中每条具体记录进行输出。 示例如下： `SELECT * FROM MyTable MATCH_RECOGNIZE ( MEASURES AVG(B.id) as Bid ALL ROWS PER MATCH PATTERN (A B C) DEFINE A AS A.name = 'a', B AS B.name = 'b', C as C.name = 'c' ) MR` 示例说明： 假设MyTable数据格式为(id,name), 有三条数据(1,a) (2,b), (3,c)。 那么ONE ROW PER MATCH会输出B的平均值2。 ALL ROWS PER MATCH会将每条记录及B的平均值输出，也就是输出(1,a, null), (2,b,2), (3,c,2)。
MEASURES	否	定义要输出的度量值。

版权所有 © 华为技术有限公司

参数	是否必选	说明
PATTERN	是	定义要匹配的模式。 ● 连续事件 PATTERN (A B C)即表示检测连续的ABC事件。 ● 逻辑事件PATTERN (A | B)即表示检测A或者B。 ● 修饰符 – *： 0次或多次迭代，如A* 匹配0次或多次A – +： 1次或多次迭代，如A+ 匹配1次或多次A – ?： 0次或1次迭代，如A? 匹配0次或多次A – {n}： n 次迭代 (n > 0)，如A{5} 匹配5次A – {n,}： n 次或更多次迭代 (n >= 0)，如A{5,} 匹配>=5次A – {n,m}： n次至m次 (包括n和m) 迭代 (0 <= n <= m, 0 < m)，如A{3,6} 匹配3至6次A – {,m}： 0次至m次 (包括0和m) 迭代 (m > 0)，如A{,4} 匹配0至4次A
DEFINE	是	定义主要的模式变量条件。
AFTER MATCH SKIP	否	定义在一个匹配找到之后从哪里开始下一轮匹配。 ● SKIP TO NEXT ROW ： 在当前匹配第一行之后的下一行开始下一轮模式匹配 ● SKIP PAST LAST ROW ： 在当前匹配的最后一行之后的下一行开始下一轮匹配 ● SKIP TO FIRST variable：从当前匹配的第一个variable开始下一轮匹配 ● SKIP TO LAST variable：从当前匹配的最后一个variable开始下一轮匹配 ● SKIP TO variable：同SKIP TO LAST variable

CEP 支持的函数

表 2-72 函数说明

函数	说明
MATCH_NUMBER()	说明本次匹配属于第几次匹配。可用在MEASURES和DEFINE子句中。

版权所有 © 华为技术有限公司

函数	说明
CLASSIFIER()	说明当前记录被匹配到PATTERN里的哪个模式变量里。可用在MEASURES和DEFINE子句中。
FIRST()/LAST()	每次匹配里的第一个/最后一个记录。比如PATTERN (A B+ C), FIRST(B.id)代表匹配里的第一个B的id，LAST(B.id)代表匹配里的最后一个B的id。
NEXT()/PREV()	相对偏移，可用在DEFINE里。比如PATTERN (A B+) DEFINE B AS B.price > PREV(B.price)。
RUNNING/ FINAL	RUNNING 表示匹配过程中间值，FINAL表示最终结果值，RUNNING/FINAL一般只在ALL ROWS PER MATCH里才有意义。比如有三条记录(a, 2), (b, 6), (c, 12), 那么RUNNING AVG(A.price)和FINAL AVG(A.price)的值就是(2, 6), (4, 6), (6, 6)。
聚合函数 (COUNT, SUM, AVG, MAX, MIN)	聚合操作，可用在MEASURES和DEFINE子句中。关于聚合函数的详细信息，请参见**聚合函数**。

示例

- 套牌车检测

5分钟内在不同区域的城市道路或者高速道路的摄像头采集到相同牌照的车辆数据，通过对号牌切换特征的模式匹配，实现套牌车检测。

```
INSERT INTO fake_licensed_car
SELECT * FROM camera_license_data MATCH_RECOGNIZE
(
  PARTITION BY car_license_number
  ORDER BY proctime
  MEASURES A.car_license_number as car_license_number, A.camera_zone_number as first_zone,
B.camera_zone_number as second_zone
  ONE ROW PER MATCH
  AFTER MATCH SKIP TO LAST C
  PATTERN (A B+ C)
  WITHIN interval '5' minute
  DEFINE
    B AS B.camera_zone_number <> A.camera_zone_number,
    C AS C.camera_zone_number = A.camera_zone_number
) MR;
```

该规则表示5分钟内在两个不同摄像区域内检测到同一车牌号车辆，为了防止出现误判，即车辆确实从A区域行驶到B区域，检查到B区域后A区域又检测到了该车牌，这种情况则认为是真正的套牌车。

输入数据：

```
浙B88888,zone_A
浙AZ626M,zone_A
浙B88888,zone_A
浙AZ626M,zone_A
浙AZ626M,zone_A
浙B88888,zone_B
浙B88888,zone_B
```


版权所有 © 华为技术有限公司

```
浙AZ626M,zone_B
浙AZ626M,zone_B
浙AZ626M,zone_C
浙B88888,zone_A
浙B88888,zone_A
```

则会输出：

```
浙B88888,zone_A,zone_B
```

2.18 StreamingML

2.18.1 异常检测

异常检测应用场景相当广泛，包括了入侵检测，金融诈骗检测，传感器数据监控，医疗诊断和自然数据检测等。异常检测经典算法包括统计建模方法，基于距离计算方法，线性模型和非线性模型等。

我们采用一种基于随机森林的异常检测方法：

- One-pass算法，O(1)均摊时空复杂度。
- 随机森林结构仅构造一次，模型更新仅仅是节点数据分布值的更新。
- 节点存储多个窗口的数据分布信息，能够检测数据分布变化。
- 异常检测和模型更新在同一个代码框架中完成。

语法格式

```
SRF_UNSUP(ARRAY[字段1, 字段2, ...], '可选参数列表')
```

说明

- 函数输出为[0, 1]区间的DOUBLE值，表示数据的异常打分。
- 字段名必须为一致的数值类型，若字段类型不同，可通过CAST函数转义，例如[a, CAST(b as DOUBLE)]。
- 可选参数列表语法为"key1=value,key2=value2,..."。

参数说明

表 2-73 参数说明

参数	是否必选	说明	默认值
transientThreshold	否	连续transientThreshold个窗口发生数据改变表示发生数据概念迁移。	5
numTrees	否	随机森林中Tree的数量。	15
maxLeafCount	否	Tree最大叶子节点数量。	15
maxTreeHeight	否	Tree最大高度。	12
seed	否	算法使用的随机种子值。	4010
numClusters	否	分类数，默认包含异常和非异常两类。	2

版权所有 © 华为技术有限公司

参数	是否必选	说明	默认值
dataViewMode	否	算法学习模式。 ● history：学习所有历史数据。 ● horizon：仅考虑最近一段时间历史数据，默认为4个窗口。	history

示例

对于数据流MyTable中的c字段运行异常检测算法，当异常分大于0.8时输出异常。

```
SELECT c,
   CASE WHEN SRF_UNSUP(ARRAY[c], "numTrees=15,seed=4010") OVER (ORDER BY proctime RANGE
BETWEEN INTERVAL '99' SECOND PRECEDING AND CURRENT ROW) > 0.8
      THEN 'anomaly'
      ELSE 'not anomaly'
   END
FROM MyTable
```

2.18.2 时间序列预测

流数据处理中经常需要对于时间序列数据进行建模和预测，建模是指提取数据中有用的统计信息和数据特征，预测是指使用模型对未来的数据进行推测。DLI服务提供了一系列随机线性模型，帮助用户在线实时进行模型的建模和预测。

ARIMA (Non-Seasonal)

ARIMA（Auto-Regressive Integrated Moving Average）是时间序列预测中的经典模型，和AR/MA/ARMA模型之间联系紧密。

- AR/MA/ARMA适用于**平稳**序列 (stationary)
  - AR(p)：自回归模型，当前值可以描述为p个之前值的线性组合。利用线性组合的权值即可预测下一个值。
  - MA(q)：移动平均模型，当前值可以描述为序列均值加上q个之前值的白噪声的线性组合。利用线性组合的权值也可预测下一个值。
  - ARMA(p, q)：自回归移动平均模型，综合了AR和MA两个模型的优势，在ARMA模型中，自回归过程负责量化当前数据与前期数据之间的关系，移动平均过程负责解决随机变动项的求解问题，因此，该模型比AR/MA更为有效和常用。
- ARIMA适用于**非平稳**序列 (non-stationary)。ARIMA(p, q, d)中p为自回归项数，q为滑动平均项数，d为使之成为平稳序列所做的差分次数（阶数）。

语法格式

```
AR_PRED(field, degree)：使用AR模型预测新数据。
AR_COEF(field, degree)：返回AR模型的权值。
ARMA_PRED(field, degree)：使用ARMA模型预测新数据。
ARMA_COEF(field, degree)：返回ARMA模型的权值。
ARIMA_PRED(field, degree, derivativeOrder)：使用ARIMA预测新数据。
```


版权所有 © 华为技术有限公司

表 2-74 参数说明

参数	是否必选	说明	默认值
field	是	数据在数据流中的字段名。	-
degree	否	指定使用之前数据项的个数，当前实现中限定p = q = degree。	5
derivativeOrder	否	指定差分次数，通常设置为1或者2。	1

示例

分别使用AR，ARMA，ARIMA结合窗口进行时间序列预测。

```
SELECT b,
   AR_PRED(b) OVER (ORDER BY rowtime ROWS  BETWEEN 5 PRECEDING AND CURRENT ROW) AS ar,
   ARMA_PRED(b) OVER (ORDER BY rowtime ROWS  BETWEEN 5 PRECEDING AND CURRENT ROW) AS
arma,
   ARIMA_PRED(b) OVER (ORDER BY rowtime ROWS  BETWEEN 5 PRECEDING AND CURRENT ROW) AS
arima
FROM MyTable
```

Holt Winters

Holt Winters算法是Exponential smoothing方法中的一种，主要特点是可以捕捉时间序列中的**季节性**趋势。

语法格式

```
HOLT_WINTERS(field, seasonality, forecastOrder)
```

表 2-75 参数说明

参数	是否必选	说明
field	是	数据在数据流中的字段名。
seasonality	是	季节性变化的周期。例如数据点是按天采集，季节性周期是一周，则seasonality为7。
forecastOrder	否	指定需要预测的元素。 当forecastOrder为1，预测下一个元素。 当forecastOrder为2，预测下下一个元素。默认值为1。 使用此参数时需要保证over窗口的大小大于设置的forecastOrder。

示例

使用HOLT WINTERS函数结合窗口进行时间序列预测。

```
SELECT b,
   HOLT_WINTERS(b, 5) OVER (ORDER BY rowtime ROWS  BETWEEN 5 PRECEDING AND CURRENT ROW)
```


版权所有 © 华为技术有限公司

```
AS a1,
   HOLT_WINTERS(b, 5, 2) OVER (ORDER BY rowtime ROWS  BETWEEN 5 PRECEDING AND CURRENT
ROW) AS a2
FROM MyTable
```

2.18.3 实时聚类

聚类算法是非监督算法中非常典型的一类算法，经典的K-Means算法通过提前确定类别数目，计算数据点之间的距离来分类。对于离线静态数据集，我们可以依赖领域中知识来确定类别数目，运行K-Means算法可以取得比较好的聚类效果。但是对于在线实时流数据，数据是在不断变化和演进，类别数目极有可能发生变化，DLI服务提供一种能够应对此类场景，无需提前设定聚类数目，并且低延时的在线聚类算法。

算法大致思想为：定义一种距离函数，两两数据点之间如果距离小于某个阈值，则他们属于同一个类别。若某数据点和多个类别中心点的距离都小于这个阈值，则多个类别会发生合并操作。当数据流中的数据到达，算法会分别计算与所有类别的距离，从而决定此数据作为一个新类别或者归属于某类别。

语法格式

```
CENTROID(ARRAY[field_names], distance_threshold)：加入当前数据点后，该数据点所属分类中心。
CLUSTER_CENTROIDS(ARRAY[field_names], distance_threshold)：加入当前数据点后，所有分类中心。
ALL_POINTS_OF_CLUSTER(ARRAY[field_names], distance_threshold)：加入当前数据点后，该分类所有数据点。
ALL_CLUSTERS_POINTS(ARRAY[field_names], distance_threshold)：加入当前数据点后，所有分类对应的所有数据点。
```

说明

- 聚类算法可以应用在**无界流**中。

参数说明

表 2-76 参数说明

参数	是否必选	说明
field_names	是	数据在数据流中的字段名，多字段以逗号隔开。例如ARRAY[a, b, c]。
distance_threshold	是	距离阈值，当两数据点距离小于阈值时，它们将属于同一个类别。

示例

分别使用四种函数结合窗口来实时计算聚类的相关信息。

```
SELECT
  CENTROID(ARRAY[c,e], 1.0) OVER (ORDER BY proctime RANGE UNBOUNDED PRECEDING) AS centroid,
  CLUSTER_CENTROIDS(ARRAY[c,e], 1.0) OVER (ORDER BY proctime RANGE UNBOUNDED PRECEDING) AS
centroids
FROM MyTable

SELECT
  CENTROID(ARRAY[c,e], 1.0) OVER (ORDER BY proctime RANGE BETWEEN INTERVAL '60' MINUTE
PRECEDING AND CURRENT ROW) AS centroidCE,
  ALL_POINTS_OF_CLUSTER(ARRAY[c,e], 1.0) OVER (ORDER BY proctime RANGE BETWEEN INTERVAL '60'
```


版权所有 © 华为技术有限公司

```
MINUTE PRECEDING AND CURRENT ROW) AS itemList,
  ALL_CLUSTERS_POINTS(ARRAY[c,e], 1.0) OVER (ORDER BY proctime RANGE  BETWEEN INTERVAL '60'
MINUTE PRECEDING AND CURRENT ROW) AS listoflistofpoints
FROM MyTable
```

2.18.4 深度学习模型预测

深度学习已经广泛应用于图像分类、图像识别和语音识别等不同领域，DLI服务中提供了若干函数实现加载深度学习模型并进行预测的能力。

目前可支持的模型包括DeepLearning4j 模型和Keras模型。由于Keras它能够以 TensorFlow、CNTK或者 Theano 作为后端运行，导入来自Keras的神经网络模型，可以借此导入Theano、Tensorflow、Caffe、CNTK等主流学习框架的模型。

语法格式

```
-- 图像分类， 返回预测图像分类的类别id
DL_IMAGE_MAX_PREDICTION_INDEX(field_name, model_path, is_dl4j_model)
DL_IMAGE_MAX_PREDICTION_INDEX(field_name, keras_model_config_path, keras_weights_path) -- 适用于Keras模型

-- 文本分类，返回预测文本分类的类别id
DL_TEXT_MAX_PREDICTION_INDEX(field_name, model_path, is_dl4j_model) -- 采用默认word2vec模型
DL_TEXT_MAX_PREDICTION_INDEX(field_name, word2vec_path, model_path, is_dl4j_model)
```

说明

模型及配置文件等需存储在用户的OBS中，路径格式为"obs://**your_ak**:**your_sk**@obs.**your_obs_region**.xxx.com:443/**your_model_path**"。

参数说明

表 2-77 参数说明

参数	是否必选	说明
field_name	是	数据在数据流中的字段名。 图像分类中field_name类型需声明为ARRAY[TINYINT]。 文本分类中field_name类型需声明为String。
model_path	是	模型存放在OBS上的完整路径，包括模型结构和模型权值。
is_dl4j_model	是	是否是deeplearning4j的模型。 true代表是deeplearning4j，false代表是keras模型。
keras_model_config_path	是	模型结构存放在OBS上的完整路径。在keras中通过model.to_json()可得到模型结构。
keras_weights_path	是	模型权值存放在OBS上的完整路径。在keras中通过model.save_weights(filepath)可得到模型权值。
word2vec_path	是	word2vec模型存放在OBS上的完整路径。

版权所有 © 华为技术有限公司

示例

图片分类预测我们采用Mnist数据集作为流的输入，通过加载预训练的deeplearning4j模型或者keras模型，可以实时预测每张图片代表的数字。

```
CREATE SOURCE STREAM Mnist(
    image Array[TINYINT]
)
SELECT DL_IMAGE_MAX_PREDICTION_INDEX(image, 'your_dl4j_model_path', false) FROM Mnist
SELECT DL_IMAGE_MAX_PREDICTION_INDEX(image, 'your_keras_model_path', true) FROM Mnist
SELECT DL_IMAGE_MAX_PREDICTION_INDEX(image, 'your_keras_model_config_path', 'keras_weights_path')
FROM Mnist
```

文本分类预测我们采用一组新闻标题数据作为流的输入，通过加载预训练的deeplearning4j模型或者keras模型，可以实时预测每个新闻标题所属的类别，比如经济，体育，娱乐等。

```
CREATE SOURCE STREAM News(
    title String
)
SELECT DL_TEXT_MAX_PREDICTION_INDEX(title, 'your_dl4j_word2vec_model_path','your_dl4j_model_path',
false) FROM News
SELECT DL_TEXT_MAX_PREDICTION_INDEX(title,
'your_keras_word2vec_model_path','your_keras_model_path', true) FROM News
SELECT DL_TEXT_MAX_PREDICTION_INDEX(title, 'your_dl4j_model_path', false) FROM New
SELECT DL_TEXT_MAX_PREDICTION_INDEX(title, 'your_keras_model_path', true) FROM New
```

2.19 保留关键字

Flink SQL将一些字符串组合保留为关键字以备将来使用。 如果要使用以下字符串作为字段名称，请确保使用反引号（例如`value`，`count`）。

A

- A
- ABS
- ABSOLUTE
- ACTION
- ADA
- ADD
- ADMIN
- AFTER
- AK
- ALL
- ALLOCATE
- ALLOW
- ALTER
- ALWAYS
- AND
- ANY
- APPEND

版权所有 © 华为技术有限公司

- APP_ID
- ARE
- ARRAY
- ARRAY_BRACKET
- AS
- ASC
- ASENSITIVE
- ASSERTION
- ASSIGNMENT
- ASYMMETRIC
- AT
- AT_LEAST_ONCE
- ATOMIC
- ATTRIBUTE
- ATTRIBUTES
- AUTHORIZATION
- AVG
- AVRO_CONFIG
- AVRO_DATA
- AVRO_SCHEMA

B

- BATCH_INSERT_DATA_NUM
- BEFORE
- BEGIN
- BERNOULLI
- BETWEEN
- BIGINT
- BINARY
- BIT
- BLOB
- BOOL
- BOOLEAN
- BOTH
- BREADTH
- BUCKET
- BY

版权所有 © 华为技术有限公司

C

- C
- CACHE_MAX_NUM
- CACHE_TIME
- CALL
- CALLED
- CARDINALITY
- CASCADE
- CASCADED
- CASE
- CAST
- CATALOG
- CATALOG_NAME
- CEIL
- CEILING
- CENTURY
- CHAIN
- CHANNEL
- CHAR
- CHARACTER
- CHARACTERISTICTS
- CHARACTERS
- CHARACTER_LENGTH
- CHARACTER_SET_CATALOG
- CHARACTER_SET_NAME
- CHARACTER_SET_SCHEMA
- CHAR_LENGTH
- CHECK
- CHECKPOINT_APP_NAME
- CHECKPOINT_INTERVAL
- CHECKPOINTINTERVAL
- CLASS_ORIGIN
- CLOB
- CLOSE
- CLUSTER_ADDRESS
- CLUSTER_ID
- CLUSTER_NAME
- COALESCE

版权所有 © 华为技术有限公司

- COBOL
- COLLATE
- COLLATION
- COLLATION_CATALOG
- COLLATION_NAME
- COLLATION_SCHEMA
- COLLECT
- COLUMN
- COLUMN_NAME
- COLUMN_NAME_MAP
- COMMAND_FUNCTION
- COMMAND_FUNCTION_CODE
- COMMIT
- COMMITTED
- CONDITION
- CONDITION_NUMBER
- CONFIGURATION
- CONFLUENT_CERTIFICATE_NAME
- CONFLUENT_PROPERTIES
- CONFLUENT_SCHEMA_FIELD
- CONFLUENT_URL
- CONNECT
- CONNECTION_NAME
- CONSTRAINT
- CONSTRAINTS
- CONSTRAINT_CATALOG
- CONSTRAINT_NAME
- CONSTRAINT_SCHEMA
- CONSTRUCTOR
- CONTAINS
- CONTINUE
- CONVERT
- CORR
- CORRESPONDING
- COUNT
- COVAR_POP
- COVAR_SAMP
- CREATE

版权所有 © 华为技术有限公司

- CREATE_IF_NOT_EXIST
- CROSS
- CUBE
- CUME_DIST
- CURRENT
- CURRENT_CATALOG
- CURRENT_DATE
- CURRENT_DEFAULT_TRANSFORM_GROUP
- CURRENT_PATH
- CURRENT_ROLE
- CURRENT_SCHEMA
- CURRENT_TIMESTAMP
- CURRENT_TRANSFORM_GROUP_FOR_TYPE
- CURRENT_USER
- CURSOR
- CURSOR_NAME
- CYCLE

D

- DATE
- DATABASE
- DATE
- DATETIME_INTERVAL_CODE
- DATETIME_INTERVAL_PRECISION
- DAY
- DB_COLUMNS
- DB_URL
- DB_OBS_SERVER
- DB_TYPE
- DEALLOCATE
- DEC
- DECADE
- DECIMAL
- DECLARE
- DEFAULTS
- DEFERRABLE
- DEFERRED
- DEFINER
- DEGREE

 版权所有 © 华为技术有限公司

- DELETE
- DELETE_OBS_TEMP_FILE
- DENSE_RANK
- DEPTH
- DEREF
- DERIVED
- DESC
- DESCRIBE
- DESCRIPTION
- DESCRIPTOR
- DETERMINISTIC
- DIAGNOSTICS
- DISALLOW
- DISCONNECT
- DIS_NOTICE_CHANNEL
- DISPATCH
- DISTINCT
- DOMAIN
- DOUBLE
- DOW
- DOY
- DRIVER
- DROP
- DUMP_INTERVAL
- DYNAMIC
- DYNAMIC_FUNCTION
- DYNAMIC_FUNCTION_CODE

E

- EACH
- ELEMENT
- ELSE
- EMAIL_KEY
- ENABLECHECKPOINT
- ENABLE_CHECKPOINT
- ENABLE_OUTPUT_NULL
- ENCODE
- ENCODE_CLASS_NAME
- ENCODE_CLASS_PARAMETER

版权所有 © 华为技术有限公司

- ENCODED_DATA
- END
- ENDPOINT
- END_EXEC
- EPOCH
- EQUALS
- ESCAPE
- ES_FIELDS
- ES_INDEX
- ES_TYPE
- ESTIMATEMEM
- ESTIMATEPARALLELISM
- EXACTLY_ONCE
- EXCEPT
- EXCEPTION
- EXCLUDE
- EXCLUDING
- EXEC
- EXECUTE
- EXISTS
- EXP
- EXPLAIN
- EXTEND
- EXTERNAL
- EXTRACT
- EVERY

F

- FALSE
- FETCH
- FIELD_DELIMITER
- FIELD_NAMES
- FILE_PREFIX
- FILTER
- FINAL
- FIRST
- FIRST_VALUE
- FLOAT
- FLOOR

 版权所有 © 华为技术有限公司

- FOLLOWING
- FOR
- FUNCTION
- FOREIGN
- FORTRAN
- FOUND
- FRAC_SECOND
- FREE
- FROM
- FULL
- FUSION

G

- G
- GENERAL
- GENERATED
- GET
- GLOBAL
- GO
- GOTO
- GRANT
- GRANTED
- GROUP
- GROUPING
- GW_URL

H

- HASH_KEY_COLUMN
- HAVING
- HIERARCHY
- HOLD
- HOUR
- HTTPS_PORT

I

- IDENTITY
- ILLEGAL_DATA_TABLE
- IMMEDIATE
- IMPLEMENTATION

版权所有 © 华为技术有限公司

- IMPORT
- IN
- INCLUDING
- INCREMENT
- INDICATOR
- INITIALLY
- INNER
- INOUT
- INPUT
- INSENSITIVE
- INSERT
- INSTANCE
- INSTANTIABLE
- INT
- INTEGER
- INTERSECT
- INTERSECTION
- INTERVAL
- INTO
- INVOKER
- IN_WITH_SCHEMA
- IS
- ISOLATION

J

- JAVA
- JOIN
- JSON_CONFIG
- JSON_SCHEMA

K

- K
- KAFKA_BOOTSTRAP_SERVERS
- KAFKA_CERTIFICATE_NAME
- KAFKA_GROUP_ID
- KAFKA_PROPERTIES
- KAFKA_PROPERTIES_DELIMITER
- KAFKA_TOPIC
- KEY

版权所有 © 华为技术有限公司

- KEY_COLUMN
- KEY_MEMBER
- KEY_TYPE
- KEY_VALUE
- KRB_AUTH

L

- LABEL
- LANGUAGE
- LARGE
- LAST
- LAST_VALUE
- LATERAL
- LEADING
- LEFT
- LENGTH
- LEVEL
- LIBRARY
- LIKE
- LIMIT
- LONG

M

- M
- MAP
- MATCH
- MATCHED
- MATCHING_COLUMNS
- MATCHING_REGEX
- MAX
- MAXALLOWEDCPU
- MAXALLOWEDMEM
- MAXALLOWEDPARALLELISM
- MAX_DUMP_FILE_NUM
- MAX_RECORD_NUM_CACHE
- MAX_RECORD_NUM_PER_FILE
- MAXVALUE
- MEMBER
- MERGE

版权所有 © 华为技术有限公司

- MESSAGE_COLUMN
- MESSAGE_LENGTH
- MESSAGE_OCTET_LENGTH
- MESSAGE_SUBJECT
- MESSAGE_TEXT
- METHOD
- MICROSECOND
- MILLENNIUM
- MIN
- MINUTE
- MINVALUE
- MOD
- MODIFIES
- MODULE
- MONTH
- MORE
- MS
- MULTISET
- MUMPS

N

- NAME
- NAMES
- NATIONAL
- NATURAL
- NCHAR
- NCLOB
- NESTING
- NEW
- NEXT
- NO
- NONE
- NORMALIZE
- NORMALIZED
- NOT
- NULL
- NULLABLE
- NULLIF
- NULLS

 版权所有 © 华为技术有限公司

- NUMBER
- NUMERIC

O

- OBJECT
- OBJECT_NAME
- OBS_DIR
- OCTETS
- OCTET_LENGTH
- OF
- OFFSET
- OLD
- ON
- ONLY
- OPEN
- OPERATION_FIELD
- OPTION
- OPTIONS
- OR
- ORDER
- ORDERING
- ORDINALITY
- OTHERS
- OUT
- OUTER
- OUTPUT
- OVER
- OVERLAPS
- OVERLAY
- OVERRIDING

P

- PAD
- PARALLELISM
- PARAMETER
- PARAMETER_MODE
- PARAMETER_NAME
- PARAMETER_ORDINAL_POSITION
- PARAMETER_SPECIFIC_CATALOG

版权所有 © 华为技术有限公司

- PARAMETER_SPECIFIC_NAME
- PARAMETER_SPECIFIC_SCHEMA
- PARTIAL
- PARTITION
- PARTITION_COUNT
- PARTITION_KEY
- PARTITION_RANGE
- PASCAL
- PASSTHROUGH
- PASSWORD
- PATH
- PERCENTILE_CONT
- PERCENTILE_DISC
- PERCENT_RANK
- PERSIST_SCHEMA
- PIPELINE_ID
- PLACING
- PLAN
- PLI
- POSITION
- POWER
- PRECEDING
- PRECISION
- PREPARE
- PRESERVE
- PRIMARY
- PRIMARY_KEY
- PRIOR
- PRIVILEGES
- PROCEDURE
- PROCTIME
- PROJECT_ID
- PUBLIC

Q

- QUARTER
- QUOTE

版权所有 © 华为技术有限公司

R

- RANGE
- RANK
- RAW
- READ
- READS
- READ_ONCE
- REAL
- RECURSIVE
- REF
- REFERENCES
- REFERENCING
- REGION
- REGR_AVGX
- REGR_AVGY
- REGR_COUNT
- REGR_INTERCEPT
- REGR_R2
- REGR_SLOPE
- REGR_SXX
- REGR_SXY
- REGR_SYY
- RELATIVE
- RELEASE
- REPEATABLE
- RESET
- RESTART
- RESTRICT
- RESULT
- RETURN
- RETURNED_CARDINALITY
- RETURNED_LENGTH
- RETURNED_OCTET_LENGTH
- RETURNED_SQLSTATE
- RETURNS
- REVOKE
- RIGHT
- ROLE

 版权所有 © 华为技术有限公司

- ROLLBACK
- ROLLING_INTERVAL
- ROLLING_SIZE
- ROLLUP
- ROUTINE
- ROUTINE_CATALOG
- ROUTINE_NAME
- ROUTINE_SCHEMA
- ROW
- ROW_COUNT
- ROW_DELIMITER
- ROW_NUMBER
- ROWS
- ROWTIME

S

- SAVEPOINT
- SCALE
- SCHEMA
- SCHEMA_CASE_SENSITIVE
- SCHEMA_NAME
- SCOPE
- SCOPE_CATALOGS
- SCOPE_NAME
- SCOPE_SCHEMA
- SCROLL
- SEARCH
- SECOND
- SECTION
- SECURITY
- SELECT
- SELF
- SENSITIVE
- SEQUENCE
- SERIALIZABLE
- SERVER
- SERVER_NAME
- SESSION
- SESSION_USER

 版权所有 © 华为技术有限公司

- SET
- SETS
- SIMILAR
- SIMPLE
- SINK
- SIZE
- SK
- SMALLINT
- SOME
- SOURCE
- SPACE
- SPECIFIC
- SPECIFICTYPE
- SPECIFIC_NAME
- SQL
- SQLEXCEPTION
- SQLSTATE
- SQLWARNING
- SQL_TSI_DAY
- SQL_TSI_FRAC_SECOND
- SQL_TSI_HOUR
- SQL_TSI_MICROSECOND
- SQL_TSI_MINUTE
- SQL_TSI_MONTH
- SQL_TSI_QUARTER
- SQL_TSI_SECOND
- SQL_TSI_WEEK
- SQL_TSI_YEAR
- SQRT
- START
- START_TIME
- STATE
- STATEMENT
- STATIC
- STDDEV_POP
- STDDEV_SAMP
- STREAM
- STRING

版权所有 © 华为技术有限公司

- STRUCTURE
- STYLE
- SUBCLASS_ORIGIN
- SUBMULTISET
- SUBSTITUTE
- SUBSTRING
- SUM
- SYMMETRIC
- SYSTEM
- SYSTEM_USER

T

- TABLE
- TABLESAMPLE
- TABLE_COLUMNS
- TABLE_NAME
- TABLE_NAME_MAP
- TEMP
- TEMPORARY
- THEN
- TIES
- TIME
- TIMESTAMP
- TIMESTAMPADD
- TIMESTAMPDIFF
- TIMEZONE_HOUR
- TIMEZONE_MINUTE
- TINYINT
- TO
- TOP_LEVEL_COUNT
- TOPIC
- TOPIC_URN
- TRAILING
- TRANSACTION
- TRANSACTIONAL_TABLE
- TRANSACTIONS_ACTIVE
- TRANSACTIONS_COMMITTED
- TRANSACTIONS_ROLLED_BACK
- TRANSFORM

 版权所有 © 华为技术有限公司

- TRANSFORMS
- TRANSLATE
- TRANSLATION
- TRANX_ID
- TREAT
- TRIGGER
- TRIGGER_CATALOG
- TRIGGER_NAME
- TRIGGER_SCHEMA
- TRIM
- TRUE
- TSDB_LINK_ADDRESS
- TSDB_METRICS
- TSDB_TIMESTAMPS
- TSDB_TAGS
- TSDB_VALUES
- TYPE
- TYPE_CLASS_NAME
- TYPE_CLASS_PARAMETER

U

- UESCAPE
- UNBOUNDED
- UNCOMMITTED
- UNDER
- UNION
- UNIQUE
- UNKNOWN
- UNNAMED
- UNNEST
- UPDATE
- UPPER
- UPSERT
- URN_COLUMN
- USAGE
- USER
- USER_DEFINED_TYPE_CATALOG
- USER_DEFINED_TYPE_CODE
- USER_DEFINED_TYPE_NAME

版权所有 © 华为技术有限公司

- USER_DEFINED_TYPE_SCHEMA
- USERNAME
- USING

V

- VALUE
- VALUES
- VALUE_TYPE
- VARBINARY
- VARCHAR
- VARYING
- VAR_POP
- VAR_SAMP
- VERSION
- VERSION_ID
- VIEW

W

- WATERMARK
- WEEK
- WHEN
- WHENEVER
- WHERE
- WIDTH_BUCKET
- WINDOW
- WITH
- WITHIN
- WITHOUT
- WORK
- WRAPPER
- WRITE

X

- XML
- XML_CONFIG

Y

- YEAR

版权所有 © 华为技术有限公司

Z

- ZONE

 版权所有 © 华为技术有限公司

3 标示符

3.1 aggregate_func

格式

无。

说明

聚合函数。

3.2 alias

格式

无。

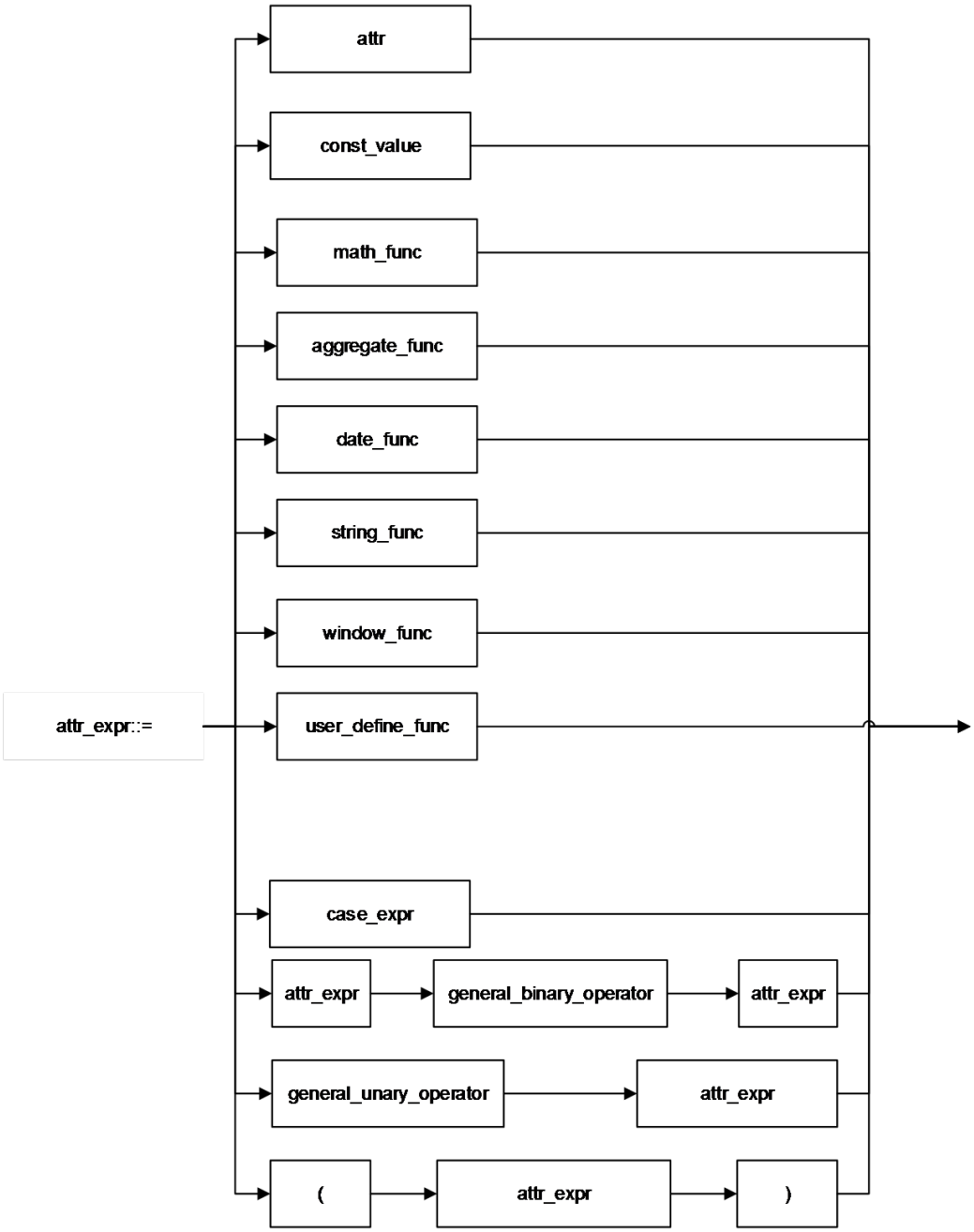
说明

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

3.3 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。

3.4 attr_expr_list

格式

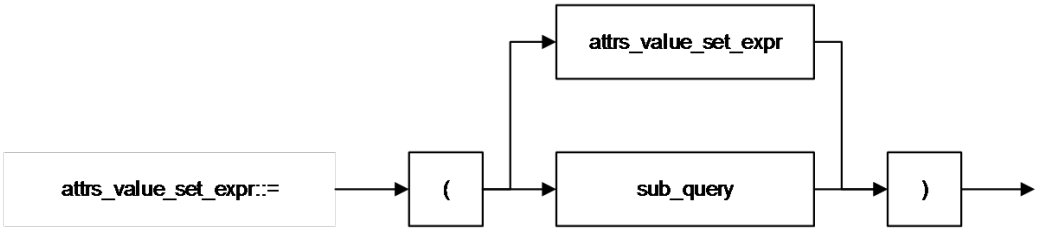
无。

说明

attr_expr列表，以逗号分隔。

3.5 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。

3.6 boolean_expression

格式

无。

说明

返回boolean类型的表达式。

3.7 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。

3.8 col_comment

格式

无。

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。

3.9 col_name

格式

无。

 版权所有 © 华为技术有限公司

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。

3.10 col_name_list

格式

无。

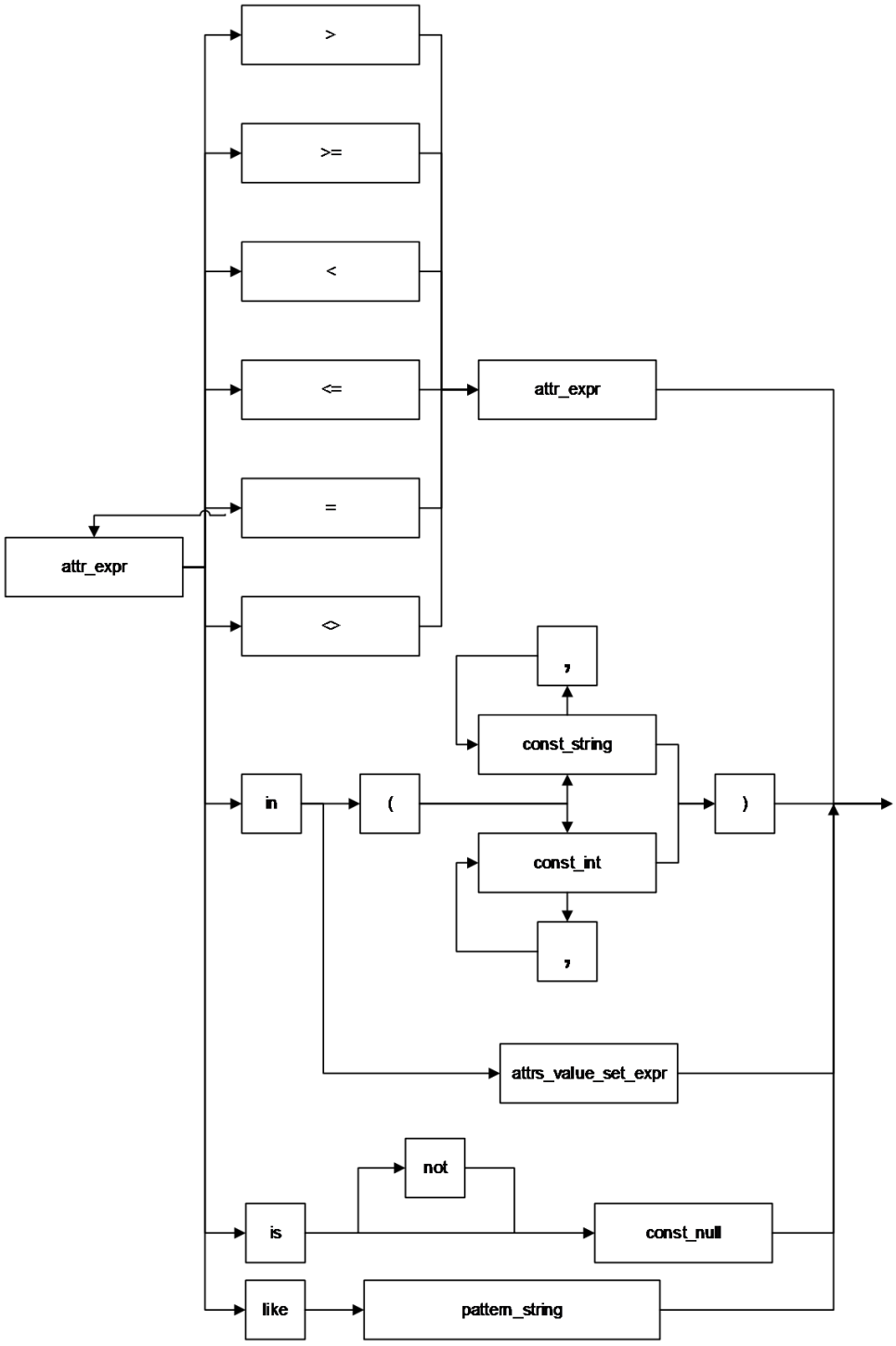
说明

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

3.11 condition

格式



 版权所有 © 华为技术有限公司

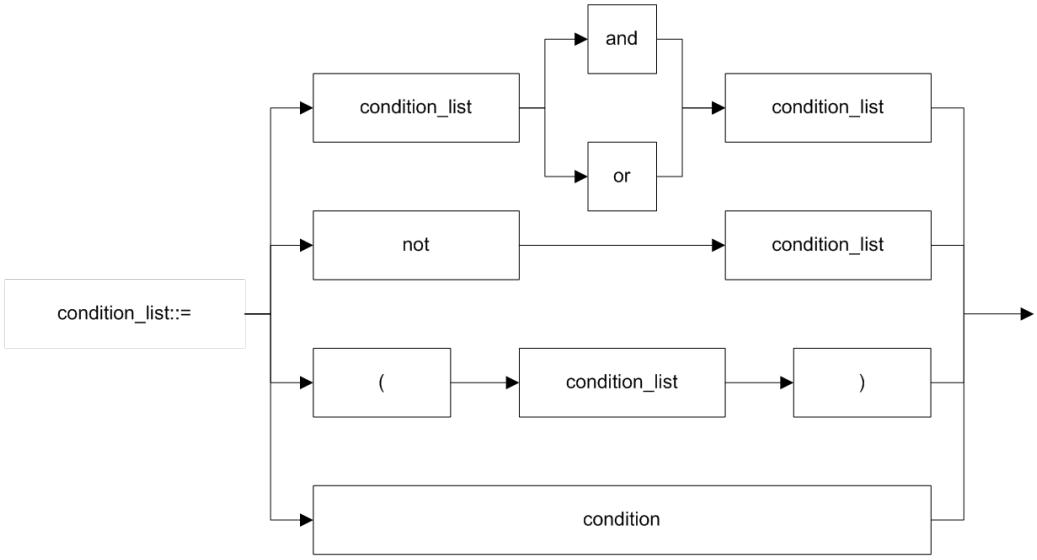
说明

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

3.12 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。

3.13 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

3.14 data_type

格式

无。

说明

数据类型，当前只支持原生数据类型。

3.15 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。

3.16 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。

3.17 else_result_expression

格式

无。

说明

CASE WHEN语句中ELSE语句后的返回结果。

3.18 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。

3.19 file_path

格式

无。

说明

文件路径，该路径是OBS路径。

3.20 function_name

格式

无。

说明

函数名称，仅支持字符串类型。

3.21 groupby_expression

格式

无。

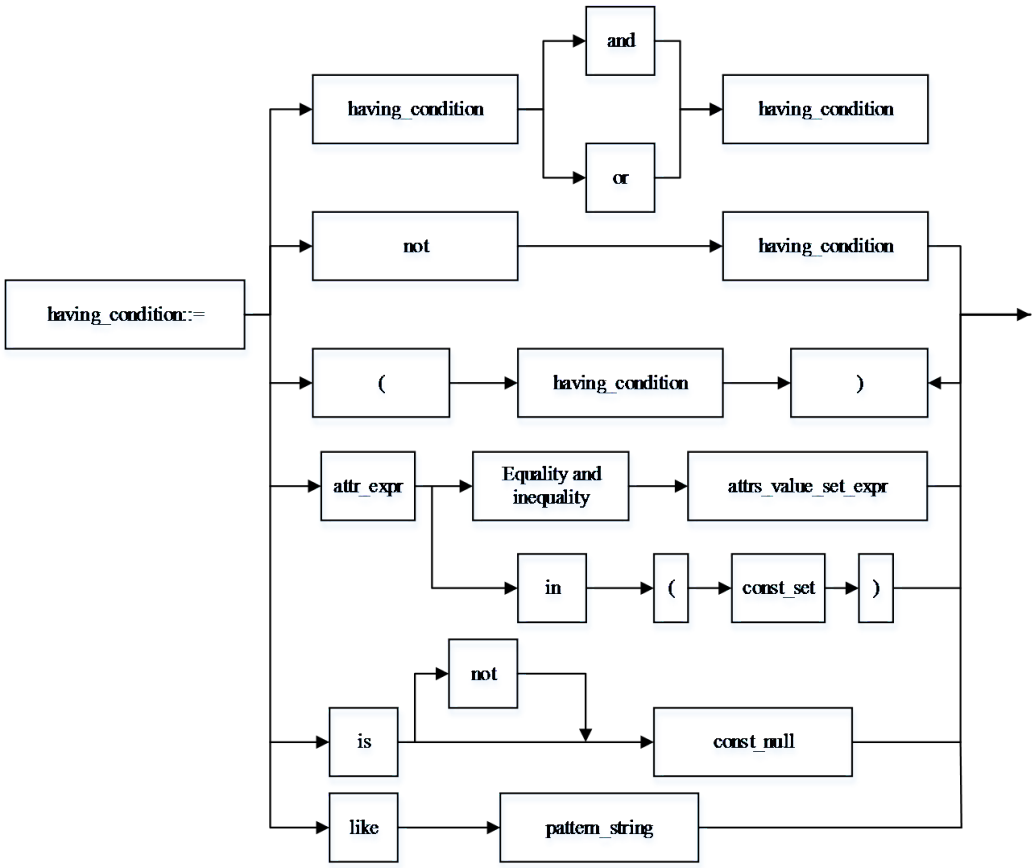
说明

包含GROUP BY的表达式。

 版权所有 © 华为技术有限公司

3.22 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。

3.23 input_expression

格式

无。

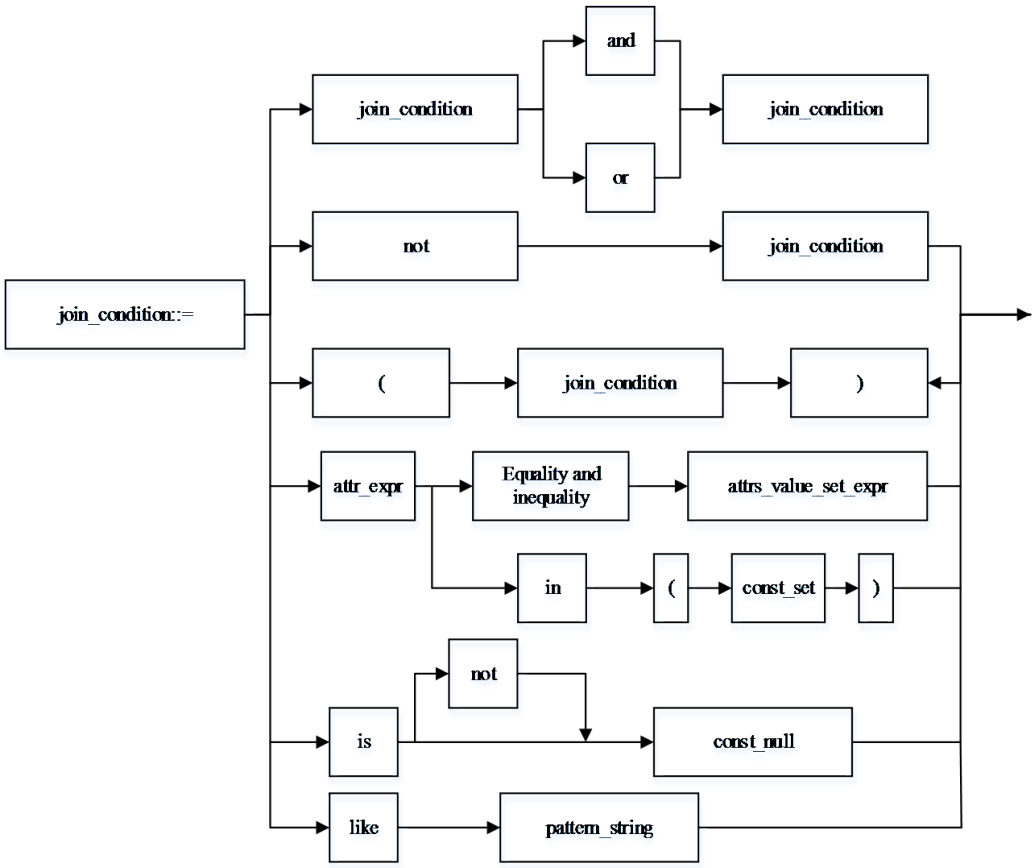
说明

CASE WHEN的输入表达式。

 版权所有 © 华为技术有限公司

3.24 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。
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
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。

3.25 non_equi_join_condition

格式

无。

说明

指不等式join条件。

3.26 number

格式

无。

说明

LIMIT限制输出的行数，只支持INT类型。

3.27 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。

 版权所有 © 华为技术有限公司

3.28 partition_col_value

格式

无。

说明

分区列值，即分区字段的值。

3.29 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。

3.30 property_name

格式

无。

说明

属性名称，仅支持字符串类型。

3.31 property_value

格式

无。

说明

属性值，仅支持字符串类型。

版权所有 © 华为技术有限公司

3.32 regex_expression

格式

无。

说明

模式匹配字符串，支持通配符匹配。

3.33 result_expression

格式

无。

说明

CASE WHEN语句中THEN语句后的返回结果。

3.34 select_statement

格式

无。

说明

SELECT基本语句，即查询语句。

3.35 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。

3.36 sql_containing_cte_name

格式

无。

版权所有 © 华为技术有限公司

说明

包含了cte_name定义的公共表达式的SQL语句。

3.37 sub_query

格式

无。

说明

指子查询。

3.38 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。

3.39 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。

3.40 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。

 版权所有 © 华为技术有限公司

3.41 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。

3.42 when_expression

格式

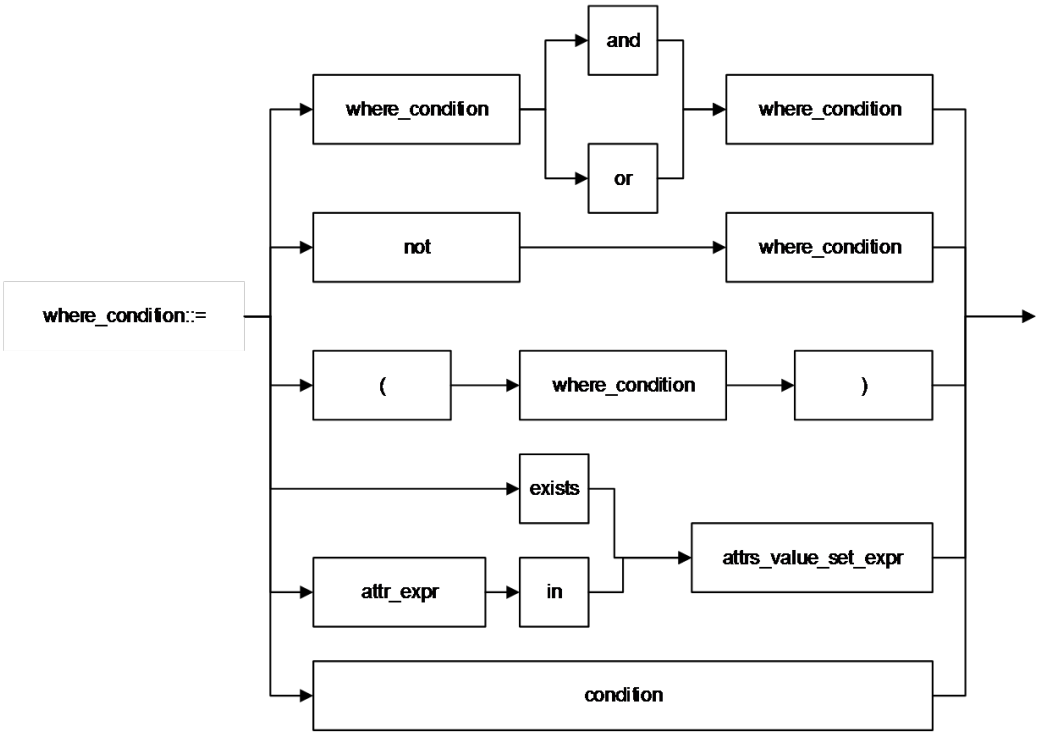
无。

说明

CASE WHEN语句的when表达式，与输入表达式进行匹配。

3.43 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。

3.44 window_function

格式

无。

说明

分析窗口函数，详情请参见**分析窗口函数**。

 版权所有 © 华为技术有限公司

4 运算符

4.1 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4-1。

表 4-1 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。

4.2 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4-2**所示。

表 4-2 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）。

4.3 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4-3，表中的A和B代表逻辑表达式。

表 4-3 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