数据仓库服务

实时数仓

文档版本 03

发布日期 2024-01-25

华为云计算技术有限公司

商标声明


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

注意

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

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

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 实时数仓简介

实时数仓需要支持将insert+upsert+update等操作实时快速入库，数据来源于上游的其他数据库或者应用，同时要求入库后的数据要能及时查询，对于查询的效率要求很高。

目前GaussDB(DWS)传统数仓已有的行存表或者列存表都无法同时满足实时入库和实时查询两个诉求。其中行存表实时入库能力强，支持高并发更新，但是磁盘占用高，查询效率低；列存表数据压缩率高，AP查询性能好，但是不能很好的支持并发更新，并发入库存在严重的锁冲突。

为了解决上面的问题，需要在使用列存储格式尽量降低磁盘占用的同时，支持高并发的更新操作入库以及高性能的查询效率。GaussDB(DWS)的实时数仓中的HStore表就是针对这种情况设计和实现的，面向对于实时入库和实时查询有较强诉求的场景，同时拥有处理传统TP场景的事务能力。

GaussDB(DWS)提供的实时数仓中实现了一种全新的HStore表，可以做到单条或者小批量IUD操作的高并发实时入库，也可以支持大批量的定期入库。数据入库提交后即可查询，无任何时延。支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询，也可以在TPCC这种强事务压力场景下保证数据强一致性。

说明

- 实时数仓的HStore表仅8.2.0.100及以上集群版本支持。
- 实时数仓为一库两用，生产即分析；适用于交易、分析混合型业务场景；分为单机、集群两种模式。关于如何创建实时数仓请参见**创建DWS 2.0集群**。
- HStore表支持冷热数据管理，具体可参考**冷热数据管理**，该功能仅8.2.0.101及以上集群版本支持。
- HStore表是实时数仓中设计的一种表类型，与SQL参数hstore没有任何关系。

与标准数仓的区别

实时数仓与标准数仓是GaussDB(DWS)的两种不同类型产品，在使用上也存在一定差异，具体可参考**表1-1**进行对比分析。

表 1-1 实时数仓与标准数仓的差异

数仓类型	标准数仓	实时数仓
适用场景	融合分析业务，一体化OLAP分析场景。主要应用于金融、政企、电商、能源等领域。	实时入库+分析混合业务，上游数据实时入库+数据入库后实时高效查询场景。主要用于电商、金融等实时入库要求高的场景。
产品优势	性价比高，使用场景广泛。 支持冷热数据分析，存储、计算弹性伸缩，无限算力、无限容量等。	混合负载，入库性能强。 提供与列存相当的高性能查询效率与高压缩率的数据压缩能力。同时拥有处理传统TP场景的事务能力。
功能特点	支持海量数据离线处理和交互查询，数据规模大、复杂数据挖掘具有很好的性能优势。	支持海量数据高并发的更新操作入库以及高性能的查询效率。在数据规模大、入库并发高、查询要求高的场景下具有很好的性能优势。
SQL语法	SQL语法兼容性高，语法通用，易于使用。	兼容列存语法。
GUC参数	丰富的GUC参数，根据客户业务场景适配最适合客户的数仓环境。	兼容标准数仓GUC参数，同时支持实时数仓调优参数。

技术特点

- 完整的事务一致性

  体现在数据插入或者更新后提交即可见，不存在时延；并发更新后数据保证强一致，不会出现乱序导致的结果预期不一致。

- 查询性能好

  多表关联等复杂AP查询场景下，更完善的分布式查询计划与分布式执行器带来的性能优势，支持复杂的子查询和存储过程。

- 入库快

  彻底解决列存CU锁冲突问题，支持高并发的更新入库操作，典型场景下，并发更新性能是之前的百倍以上。

- 高压缩

  数据在MERGE进入列存主表后，按列存储具有天然的压缩优势，能极大地节省磁盘空间与IO资源。

- 查询加速

  支持主键等传统索引能力去重和加速点查，也支持分区、多维字典、局部排序等方式进一步加速AP查询。

行存、列存、HStore 表对比

表 1-2 行存、列存、HStore 表对比

表类型	行存表	列存表	HStore表
数据存储方式	以元组为单位，将每一条数据的所有属性值存储到临近的空间里。	以CU（Compress Unit）为单位，将单个属性的所有值存储到临近的空间里。	数据主要以CU形式存储在列存主表上，对于被更新的列、小批量插入的数据将被序列化后存储到新设计的Delta表上。
数据写入	行存压缩暂未商用，数据按原始状态存储，磁盘空间占用较大。	按列存储时，由于属性值类型相同具有天然的压缩优势。数据写入时能极大节省IO资源与磁盘空间占用。	批量插入的数据直接写入CU，具有与列存一致的压缩优势。 被更新的列、小批量插入的数据会序列化后压缩。同时定期merge到主表CU。
数据更新	数据按行更新，没有CU锁问题，并发更新（update/upsert/delete等）性能好。	即使更新单条数据，也要获取整个CU的锁，基本无法支持并发更新（update/upsert/delete等）。	彻底解决列存更新的CU锁问题，并发更新（update/upsert/delete等）的性能达到行存的60%以上。
数据读取	按行读取，即使只需访问某一列的数据，也需要将一整行的数据取出。查询性能较差。	按列读取时只需访问该列的CU，再加上CU的压缩优势导致需要占用的IO资源更少，读取性能很好。	对于列存主表的数据按列读取，对于被更新的列、小批量插入的数据会反序列化后取出，数据merge到主表后具有与列存一致的数据读取优势。
优点	并发更新性能好。	查询性能好，磁盘占用空间少。	并发更新性能好，数据MERGE后具有与列存一致的查询性能优势与压缩优势。
缺点	占用磁盘空间多，查询性能差。	基本无法支持并发更新。	需要后台常驻线程对HStore表进行merge清理操作。先merge到CU主表再进行清理，与SQL语法中的Merge无关。
适用场景	1. 更新删除操作频繁的TP事务场景。 2. 点查询（基于索引的、返回数据量小的简单查询）。	1. 查询分析为主的AP场景。 2. 数据量大，存入后的更新删除操作少。	1. 实时并发入库场景。 2. 需要支持高并发的更新入库操作以及高性能的查询效率。

2 支持与限制

实时数仓兼容所有列存语法，具体情况如下：

表 2-1 支持的语法

语法	是否支持
CREATE TABLE	支持
CREATE TABLE LIKE	支持
DROP TABLE	支持
INSERT	支持
COPY	支持
SELECT	支持
TRUNCATE	支持
EXPLAIN	支持
ANALYZE	支持
VACUUM	支持
ALTER TABLE DROP PARTITION	支持
ALTER TABLE ADD PARTITION	支持
ALTER TABLE SET WITH OPTION	支持
ALTER TABLE DROP COLUMN	支持
ALTER TABLE ADD COLUMN	支持
ALTER TABLE ADD NODELIST	支持
ALTER TABLE CHANGE OWNER	支持
ALTER TABLE RENAME COLUMN	支持
ALTER TABLE TRUNCATE PARTITION	支持

语法	是否支持
CREATE INDEX	支持
DROP INDEX	支持
DELETE	支持
ALTER TABLE 其他	支持
ALTER INDEX	支持
MERGE	支持
SELECT INTO	支持
UPDATE	支持
CREATE TABLE AS	支持

约束限制

1. 当需要使用HStore表时，需要同步修改以下几个参数的默认值，否则会导致HStore表性能严重劣化。

   推荐的参数修改配置是：autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。
2. 当前HStore与列存都不支持使用vacuum清理索引脏数据，在频繁update场景可能会导致索引膨胀，后续版本会支持。

3 实时数仓语法

3.1 CREATE TABLE

功能描述

在当前数据库中创建一个新的空白HStore表，该表由命令执行者所有。

实时数仓提供创建HStore表DDL语句。创建HStore表DDL需要指定enable_hstore为true, 同时需要将orientation属性设置为column。

注意事项

- 创建HStore表的用户需要拥有schema cstore的USAGE权限。
- 表级参数enable_delta与enable_hstore无法同时开启，因为enable_delta用于控制普通列存表的delta开启，会与enable_hstore冲突。
- 每一个HStore表绑定一张delta表，delta表的oid记录在pg_class中reldeltaidx字段（reldelta字段被列存表的delta表使用）。

语法格式

```
CREATE TABLE [ IF NOT EXISTS ] table_name
({ column_name data_type
    | LIKE source_table [like_option [...] ] }
}
    [, ... ])
[ WITH ( {storage_parameter = value}  [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ DISTRIBUTE BY  HASH ( column_name [,...])]
[ TO { GROUP groupname | NODE ( nodename [, ... ] ) } ]
[ PARTITION BY {
        {RANGE (partition_key) ( partition_less_than_item [, ... ] )}
 } [ { ENABLE | DISABLE } ROW MOVEMENT ] ];
其中like选项like_option为：
{ INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES | STORAGE | COMMENTS | PARTITION
| RELOPTIONS | DISTRIBUTION | ALL }
```


列存表的 Delta 表差异

表 3-1 HStore 表与列存表的辅助 Delta 表差异

数仓类型	列存的delta表	HStore的delta表
表结构	与列存主表的表定义一致	与主表表定义不一样。
功能	用于暂存小批量insert的数据，满阈值后再merge到主表，避免直接insert到主表产生大量小CU。	用于持久化存储update/delete/insert信息。在故障拉起后用于恢复内存更新链等管理并发更新的内存结构。
缺陷	来不及merge导致delta表膨胀，影响查询性能，同时无法解决并发update的锁冲突问题	依赖后台常驻autovacuum来做merge操作。

参数说明

- **IF NOT EXISTS**

  指定IF NOT EXISTS时，若不存在同名表，则可以成功创建表。若已存在同名表，创建时不会报错，仅会提示该表已存在并跳过创建。

- **table_name**

  要创建的表名。

  表名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。

- **column_name**

  新表中要创建的字段名。

  字段名长度不超过63个字符，以字母或下划线开头，可包含字母、数字、下划线、$、#。

- **data_type**

  字段的数据类型。

- **LIKE source_table [like_option ...]**

  LIKE子句声明一个表，新表自动从这个表中继承所有字段名及其数据类型。

  新表与原表之间在创建动作完毕之后是完全无关的。在原表做的任何修改都不会传播到新表中，并且也不可能在扫描原表的时候包含新表的数据。

  被复制的列并不使用相同的名字进行融合。如果明确的指定了相同的名字或者在另外一个LIKE子句中，将会报错。

  HStore表只能从HStore表中进行继承。

- **WITH ({ storage_parameter = value } [, ...])**

  这个子句为表指定一个可选的存储参数。

  - ORIENTATION

    指定表数据的存储方式，即时序方式、行存方式、列存方式，该参数设置成功后就不再支持修改。对于HStore表，应当使用列存方式，同时设置enable_hstore为on。

取值范围：

- TIMESERIES，表示表的数据将以时序方式存储。
- COLUMN，表示表的数据将以列存方式存储。
- ROW，表示表的数据将以行方式存储。

默认值：ROW。

- COMPRESSION

  指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。

  取值范围：

  - HStore表和列存表的有效值为YES/NO和/LOW/MIDDLE/HIGH，默认值为LOW。
  - 行存表的有效值为YES/NO，默认值为NO。

- COMPRESSLEVEL

  指定表数据同一压缩级别下的不同压缩水平，它决定了同一压缩级别下表数据的压缩比以及压缩时间。对同一压缩级别进行了更加详细的划分，为用户选择压缩比和压缩时间提供了更多的空间。总体来讲，此值越大，表示同一压缩级别下压缩比越大，压缩时间越长；反之亦然。该参数只对时序表和列存表有效。

  取值范围：0~3

  默认值：0

- MAX_BATCHROW

  指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对时序表和列存表有效。

  取值范围：10000~60000

  默认值60000

- PARTIAL_CLUSTER_ROWS

  指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对时序表和列存表有效。

  取值范围：600000~2147483647

- enable_delta

  指定了在列存表是否开启delta表。对HStore表不能开启该参数。

  默认值：off

- enable_hstore

  指定了是否创建为H-Store表（基于列存表实现）。该参数只对列存表有效。该参数仅8.2.0.100及以上集群版本支持。云原生3.0暂不支持该参数。

  默认值：off

  说明

  打开该参数时必须设置以下GUC参数用于保证H-Store表的清理，推荐值如下：

  autovacuum=true，autovacuum_max_workers=6，autovacuum_max_workers_hstore=3。

- SUB_PARTITION_COUNT

  指定二级分区的个数。该参数用于设置在导入阶段二级分区个数。在建表时进行设置，建表后不支持修改。不建议用户随意设置该默认值，可能会影响导入和查询的性能。

  取值范围：1～1024

  默认值：32

- DELTAROW_THRESHOLD

  指定HStore表导入时小于多少行(SUB_PARTITION_COUNT * DELTAROW_THRESHOLD)的数据进入delta表。

  取值范围：0～60000

  默认值为60000

- COLVERSION

  指定存储格式的版本。 HStore表只支持2.0版本。

  取值范围：

  1.0：列存表的每列以一个单独的文件进行存储，文件名以relfilenode.C1.0、relfilenode.C2.0、relfilenode.C3.0等命名。

  2.0：列存表的每列合并存储在一个文件中，文件名以relfilenode.C1.0命名。

  默认值：2.0

- DISTRIBUTE BY

  指定表如何在节点之间分布或者复制。

  取值范围：

  HASH（column_name ）：对指定的列进行Hash，通过映射，把数据分布到指定DN。

- TO { GROUP groupname | NODE (nodename [, ...]) }

  TO GROUP指定创建表所在的Node Group，目前不支持hdfs表使用。TO NODE主要供内部扩容工具使用，一般用户不应该使用。

- PARTITION BY

  指定HStore表的初始分区。

示例

创建简单的HStore表:

```
CREATE TABLE warehouse_t1
(
   W_WAREHOUSE_SK           INTEGER               NOT NULL,
   W_WAREHOUSE_ID           CHAR(16)              NOT NULL,
   W_WAREHOUSE_NAME         VARCHAR(20)                   ,
   W_WAREHOUSE_SQ_FT        INTEGER                       ,
   W_STREET_NUMBER          CHAR(10)                      ,
   W_STREET_NAME            VARCHAR(60)                   ,
   W_STREET_TYPE            CHAR(15)                      ,
   W_SUITE_NUMBER           CHAR(10)                      ,
   W_CITY                   VARCHAR(60)                   ,
   W_COUNTY                 VARCHAR(30)                   ,
   W_STATE                  CHAR(2)                       ,
   W_ZIP                    CHAR(10)                      ,
   W_COUNTRY                VARCHAR(20)                   ,
   W_GMT_OFFSET             DECIMAL(5,2)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE=ON);

CREATE TABLE warehouse_t2 (LIKE warehouse_t1 INCLUDING ALL);
```


3.2 INSERT

功能描述

往HStore表中插入一行或多行数据。

注意事项

- 当单次插入的数据量大于等于表级参数DELTAROW_THRESHOLD时，数据会直接插入主表生成CU（Compress Unit）。
- 当插入的数据量小于表级参数DELTAROW_THRESHOLD时，会在辅助Delta表上插入一条类型为I的记录，同时将数据序列化存储到这条记录的values字段。
- 插入到Delta表的数据跟主表使用全局统一分配的cuid。
- 插入到delta表的数据依赖autovacuum 来merge到主表CU。

语法格式

```
INSERT [/*+ plan_hint */] [ IGNORE | OVERWRITE ] INTO table_name [ AS alias ] [ ( column_name [, ...] ) ]
    { DEFAULT VALUES
    | VALUES {( { expression | DEFAULT } [, ...] ) }[, ...] | query }
```

参数说明

- **table_name**

  要插入数据的目标表名。

  取值范围：已存在的表名。
- **AS**

  用于给目标表table_name指定别名。alias即为别名的名字。
- **column_name**

  目标表中的字段名。
- **query**

  一个查询语句（SELECT语句），将查询结果作为插入的数据。

示例

创建表reason_t1：

```
-- 创建表reason_t1。
CREATE TABLE reason_t1
(
    TABLE_SK          INTEGER               ,
    TABLE_ID          VARCHAR(20)           ,
    TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE=ON);
```

向表中插入一条记录：

```
INSERT INTO reason_t1(TABLE_SK, TABLE_ID, TABLE_NA) VALUES (1, 'S01', 'StudentA');
```

向表中插入多条记录：

```
INSERT INTO reason_t1 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
SELECT * FROM reason_t1 ORDER BY 1;
 TABLE_SK | TABLE_ID | TABLE_NAME
----------+----------+------------
```

```
     1 |     S01 |  StudentA
     2 |     T01 |  TeacherA
     3 |     T02 |  TeacherB
(3 rows)
```

3.3 DELETE

功能描述

删除HStore表中的数据。

注意事项

- 如果需要删除表上的所有数据，建议使用TRUNCATE语法, 可以有效提高性能同时减少空间膨胀。
- HStore表上的单条Delete操作，会往Delta中插入一条type是D的记录，同时在更新内存更新链用于管理并发。
- HStore表上的批量Delete操作，对于每个CU上的连续delete，会插入一条type是D的记录。
- 对于并发delete场景， 传统列存储格式由于同时操作相同CU时会阻塞所以并发性能较差， 对于HStore表由于不需要阻塞等待，并发delete性能可达到列存的百倍以上。
- 语法完全兼容列存，更多信息可以参考UPDATE语法。

语法格式

```
DELETE FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
   [ USING using_list ]
   [ WHERE condition ]
```

参数说明

- **ONLY**

  如果指定ONLY则只有该表被删除；如果没有声明，则该表和它的所有子表将都被删除。
- **table_name**

  目标表的名字（可以有模式修饰）。

  取值范围：已存在的表名。
- **alias**

  目标表的别名。

  取值范围：字符串，符合标识符命名规范。
- **using_list**

  using子句。
- **condition**

  一个返回boolean值的表达式，用于判断哪些行需要被删除。

示例

创建表reason_t2：


```
CREATE TABLE reason_t2
(
    TABLE_SK          INTEGER               ,
    TABLE_ID          VARCHAR(20)           ,
    TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE=ON);
INSERT INTO reason_t2 VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
```

使用WHERE 条件删除：

```
DELETE FROM reason_t2 WHERE TABLE_SK = 2;
DELETE FROM reason_t2 AS rt2 WHERE rt2.TABLE_SK = 2;
```

使用IN语法删除：

```
DELETE FROM reason_t2 WHERE TABLE_SK in (1,3);
```

3.4 UPDATE

功能描述

更新HStore表上指定的数据。

注意事项

- 与列存一样，当前版本HStore上的UPDATE操作始终先DELETE再INSERT。全局GUC参数可控制打开HStore的轻量化UPDATE，当前版本默认关闭。
- 对于并发UPDATE场景， 传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差， 对于HStore表由于不需要阻塞等待，并发UPDATE性能可达到列存的百倍以上。

语法格式

```
UPDATE [/*+ plan_hint */] [ ONLY ] table_name [ * ] [ [ AS ] alias ]
SET {column_name = { expression | DEFAULT }
    |( column_name [, ...] ) = {( { expression | DEFAULT } [, ...] ) |sub_query }}[, ...]
    [ FROM from_list] [ WHERE condition ];
```

参数说明

- **plan_hint子句**

  以/*+ */的形式在关键字后，用于对指定语句块生成的计划进行hint调优，详细用法请参见**使用Plan Hint进行调优**
- **table_name**

  要更新的表名，可以使用模式修饰。

  取值范围：已存在的表名称。
- **alias**

  目标表的别名。

  取值范围：字符串，符合标识符命名规范。
- **expression**

  赋给字段的值或表达式。
- **DEFAULT**

  用对应字段的缺省值填充该字段。

  如果没有缺省值，则为NULL。

- **from_list**

  一个表的表达式列表，允许在WHERE条件里使用其他表的字段。与在一个SELECT语句的FROM子句里声明表列表类似。

  须知

  目标表绝对不能出现在from_list里，除非在使用一个自连接（此时它必须以from_list的别名出现）。

- **condition**

  一个返回boolean类型结果的表达式。只有这个表达式返回true的行才会被更新。

示例

创建表reason_update：

```
CREATE TABLE reason_update
(
    TABLE_SK          INTEGER               ,
    TABLE_ID          VARCHAR(20)           ,
    TABLE_NA          VARCHAR(20)
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE=ON);
```

向表reason_update中插入数据：

```
INSERT INTO reason_update VALUES (1, 'S01', 'StudentA'),(2, 'T01', 'TeacherA'),(3, 'T02', 'TeacherB');
```

对表reason_update执行UPDATE操作：

```
UPDATE reason_update SET TABLE_NA = 'TeacherD' where TABLE_SK = 3;
```

3.5 UPSERT

功能描述

HStore兼容UPSERT语法，向表中添加一行或多行数据。当出现主键或者唯一约束冲突时更新或者忽略冲突的数据。

注意事项

- 目标表上必须包含主键或者唯一索引才可以执行UPSERT的冲突更新语句。
- 与列存一样，当UPSERT触发更新操作时，当前版本HStore上的更新操作始终先DELETE再INSERT。
- 对于并发UPSERT场景，传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差，对于HStore表由于不需要阻塞等待，并发UPSERT性能可达到列存的百倍以上。

语法格式

表 3-2 UPSERT 语法格式

语法格式	冲突更新	冲突忽略
第一种：不指定索引	INSERT INTO ON DUPLICATE KEY UPDATE	INSERT IGNORE INSERT INTO ON CONFLICT DO NOTHING
第二种：从指定列名或者约束上可以推断唯一约束	INSERT INTO ON CONFLICT(...) DO UPDATE SET INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO UPDATE SET	INSERT INTO ON CONFLICT(...) DO NOTHING INSERT INTO ON CONFLICT ON CONSTRAINT con_name DO NOTHING

参数说明

第一种不指定索引。会在所有主键或唯一索引上检查冲突，有冲突就会忽略或者更新。

第二种指定索引。会从ON CONFLICT子句中指定列名、包含列名的表达式或者约束名上推断主键或者唯一索引。

- 唯一索引推断

  对于第二种语法形式，通过指定列名或者约束名推断主键或者唯一索引。列名可以是单一列名，或者由多个列名组成的表达式，比如column1，column2，column3。

- UPDATE子句

  UPDATE子句可以通过VALUES(colname)或者EXCLUDED.colname引用插入的数据。EXCLUDED表示因冲突原本该排除的数据行。

- WHERE子句
  - 用于在数据冲突时，判断是否满足指定条件。如果满足，则更新冲突数据。否则忽略。
  - 只有第二种语法形式的冲突更新语法可以指定WHERE子句。即 INSERT INTO ON CONFLICT(...) DO UPDATE SET WHERE

示例

创建表reason_upsert并向表中插入数据：

```
CREATE TABLE reason_upsert
(
  a    int primary key,
  b    int,
  c    int
)WITH(ORIENTATION=COLUMN, ENABLE_HSTORE=ON);
INSERT INTO reason_upsert VALUES (1, 2, 3);
```

忽略冲突的数据：

```
INSERT INTO reason_upsert  VALUES (1, 4, 5),(2, 6, 7) ON CONFLICT(a) DO NOTHING;
```

更新冲突的数据：

```
INSERT INTO reason_upsert  VALUES (1, 4, 5),(3, 8, 9) ON CONFLICT(a) DO UPDATE SET b = EXCLUDED.b,
c = EXCLUDED.c;
```

3.6 MERGE INTO

功能描述

通过MERGE INTO语句，将目标表和源表中数据针对关联条件进行匹配，若关联条件匹配时对目标表进行UPDATE，无法匹配时对目标表执行INSERT。此语法可以很方便地用来合并执行UPDATE和INSERT，避免多次执行。

注意事项

对于并发MERGEINTO场景，触发UPDATE时，传统列存储格式由于同时操作相同CU时会产生锁冲突所以并发性能较差，对于HStore表由于不需要阻塞等待，并发MERGE INTO性能可达到列存的百倍以上。

语法格式

```
MERGE INTO table_name [ [ AS ] alias ]
USING { { table_name | view_name } | subquery } [ [ AS ] alias ]
ON ( condition )
[
  WHEN MATCHED THEN
  UPDATE SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
  [ WHERE condition ]
]
[
  WHEN NOT MATCHED THEN
  INSERT { DEFAULT VALUES |
  [ ( column_name [, ...] ) ] VALUES ( { expression | DEFAULT } [, ...] ) [, ...] [ WHERE condition ] }
];
```

参数说明

- **INTO子句**

  指定正在更新或插入的目标表。

  - **talbe_name**

    目标表的表名。

  - **alias**

    目标表的别名。

    取值范围：字符串，符合标识符命名规范。

- **USING子句**

  指定源表，源表可以为表、视图或子查询。

- **ON子句**

  关联条件，用于指定目标表和源表的关联条件。不支持更新关联条件中的字段。ON关联条件可以是ctid, xc_node_id, tableoid这三个系统列。

- **WHEN MATCHED子句**

  当源表和目标表中数据针对关联条件可以匹配上时，选择WHEN MATCHED子句进行UPDATE操作。

  说明

  不支持更新分布列、系统表以及系统列。

- **WHEN NOT MATCHED子句**

  当源表和目标表中数据针对关联条件无法匹配时，选择WHEN NOT MATCHED子句进行INSERT操作。

  说明

  - 不支持INSERT子句中包含多个VALUES。
  - WHEN MATCHED和WHEN NOT MATCHED子句顺序可以交换，可以缺省其中一个，但不能同时缺省。
  - 不支持同时指定两个WHEN MATCHED或WHEN NOT MATCHED子句。

示例

创建进行MERGE INTO的目标：

```
CREATE TABLE target(a int, b int)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE = ON);
INSERT INTO target VALUES(1, 1),(2, 2);
```

创建数据源表：

```
CREATE TABLE source(a int, b int)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE = ON);
INSERT INTO source VALUES(1, 1),(2, 2),(3, 3),(4, 4),(5, 5);
```

执行MERGE INTO操作：

```
MERGE INTO target t
USING source s
ON (t.a = s.a)
WHEN MATCHED THEN
  UPDATE SET t.b = t.b + 1
WHEN NOT MATCHED THEN
  INSERT VALUES (s.a, s.b) WHERE s.b % 2 = 0;
```

3.7 SELECT

功能描述

从HStore表读取数据。

注意事项

- 列存表与HStore表都暂不支持SELECT FOR UPDATE语法。
- 对HStore表执行SELECT查询时，会扫描列存主表CU上的数据、delta表上的I记录中的数据、内存中每行数据更新信息，并将三种信息合并后返回。
- 在通过主键索引或唯一索引查询数据的场景中：

  对于传统列存表，唯一索引会同时存储行存Delta表上的数据位置信息（blocknum，offset）与列存主表的数据位置信息（cuid，offset），数据MERGE到主表后又会插入新的索引元组，索引会持续膨胀。

  对于HStore表，由于实现了全局CUID的统一分配，索引元组中始终只存储（cuid，offset）， 数据MERGE后不会产生新的索引元组。

语法格式

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [/*+ plan_hint */] [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
{ * | {expression [ [ AS ] output_name ]} [, ...] }
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY grouping_element [, ...] ]
```



```
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]
[ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort_expression_clause ] [ NULLS { FIRST | LAST } ]} [, ...] ]
[ { [ LIMIT { count | ALL } ] [ OFFSET start [ ROW | ROWS ] ] } | { LIMIT start, { count | ALL } } ]
```

参数说明

- **DISTINCT [ON (expression [, ...])]**

  从SELECT的结果集中删除所有重复的行，使结果集中的每行都是唯一的。

  ON（expression [, ...]）只保留那些在给出的表达式上运算出相同结果的行集合中的第一行。

- **SELECT列表**

  指定查询表中列名，可以是部分列或者是全部（使用通配符*表示）。

  通过使用子句AS output_name可以为输出字段取个别名，这个别名通常用于输出字段的显示。

- **FROM子句**

  为SELECT声明一个或者多个源表。

  FROM子句涉及的元素如下所示。

- **WHERE子句**

  WHERE子句构成一个行选择表达式，用来缩小SELECT查询的范围。condition是返回值为布尔型的任意表达式，任何不满足该条件的行都不会被检索。

  WHERE子句中可以通过指定"(+)"操作符的方法将表的连接关系转换为外连接。但是不建议用户使用这种用法，因为这并不是SQL的标准语法，在做平台迁移的时候可能面临语法兼容性的问题。同时，使用"(+)"有很多限制：

- **GROUP BY子句**

  将查询结果按某一列或多列的值分组，值相等的为一组。

- **HAVING子句**

  与GROUP BY子句配合用来选择特殊的组。HAVING子句将组的一些属性与一个常数值比较，只有满足HAVING子句中的逻辑表达式的组才会被提取出来。

- **ORDER BY子句**

  对SELECT语句检索得到的数据进行升序或降序排序。对于ORDER BY表达式中包含多列的情况：

示例

创建表reason_select并向表中插入数据：

```
CREATE TABLE reason_select
(
  r_reason_sk integer,
  r_reason_id integer,
  r_reason_desc character(100)
)WITH(ORIENTATION = COLUMN, ENABLE_HSTORE=ON);
INSERT INTO reason_select values(3, 1,'reason 1'),(10, 2,'reason 2'),(4, 3,'reason 3'),(10, 4,'reason 4');
```

执行GROUP BY分组操作：

```
SELECT COUNT(*), r_reason_sk FROM reason_select GROUP BY r_reason_sk;
```

执行HAVING过滤操作：

```
SELECT COUNT(*) c,r_reason_sk FROM reason_select GROUP BY r_reason_sk HAVING c > 1;
```

执行ORDER BY操作：


```
SELECT * FROM reason_select ORDER BY r_reason_sk;
```

3.8 ALTER TABLE

功能描述

修改表，包括修改表的定义、重命名表、重命名表中指定的列、添加/更新多个列、将列存改为HStore表等。

注意事项

- 通过ALTER修改enable_hstore值可以将列存表变成HStore表，或者将HStore修改成列存表。但需要注意enable_delta为on时，无法设置enable_hstore为on。
- 对于部分ALTER操作（修改列类型，分区合并，添加NOT NULL约束，添加主键约束），HStore表需要先将数据MERGE到主表然后再执行原有的ALTER逻辑，这可能会带来额外的性能开销， 性能影响大小与delta表的数据量相关。
- 当需要增加列时， 建议不要与其它类型的ALTER（比如修改列类型等）组合使用，在一条ALTER里只有ADD COLUMN情况下，由于不需要做FULL MERGE, 性能会有很大提升。
- 不支持修改存储参数ORIENTATION。

修改表属性

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> SET （{ENABLE_HSTORE = ON} [, ... ] );
```

将列存表修改成HStore表：

```
CREATE TABLE alter_test(a int, b int) WITH(ORIENTATION = COLUMN);
ALTER TABLE alter_test SET (ENABLE_HSTORE = ON);
```

须知

当需要使用HStore表时，一定要同步修改如下参数的默认值，否则会导致HStore性能严重劣化，推荐的默认配置是

autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。

增加列

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> ADD COLUMN <new_column> <data_type> [ DEFAULT
<default_value>];
```

示例：

创建表alter_test2并对其增加列：

```
CREATE TABLE alter_test2(a int, b int) WITH(ORIENTATION = COLUMN,ENABLE_HSTORE = ON);
ALTER TABLE alter_test ADD COLUMN c int;
```


说明

增加列时不建议在同一个SQL中与其它ALTER组合使用。

重命名

使用语法：

```
ALTER TABLE [ IF EXISTS ] <table_name> RENAME TO <new_table_name>;
```

示例：

创建表alter_test3并对其重命名为alter_new：

```
CREATE TABLE alter_test3(a int, b int) WITH(ORIENTATION = COLUMN,ENABLE_HSTORE = ON);
ALTER TABLE alter_test3 RENAME TO alter_new;
```


4 实时数仓函数

hstore_light_merge(rel_name text)

描述：该函数用于手动对HStore表进行轻量化清理操作，持有目标表的三级锁。

返回值类型：int

示例：

```
SELECT hstore_light_merge('reason_select');
```

hstore_full_merge(rel_name text)

描述：该函数用于手动对HStore表进行全量清理操作。

返回值类型：int

须知

- 执行该操作会强制将DELTA表上的所有可见操作Merge到主表，然后建一张新的空Delta表， 期间持有该表的八级锁。
- 该操作的耗时长度与DELTA表上的数据量有关，务必打开HStore的清理线程，确保HStore表的及时清理。

示例：

```
SELECT hstore_full_merge('reason_select');
```

5 实时数仓 GUC 参数

autovacuum

参数说明：控制是否启动数据库自动清理进程（autovacuum）。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启数据库自动清理进程。
- off表示关闭数据库自动清理进程。

默认值：on

autovacuum_max_workers

参数说明：设置能同时运行的自动清理线程的最大数量，该参数的取值上限与max_connections和job_queue_processes大小有关。

参数类型：SIGHUP

取值范围：整型

- 最小值为0，表示不会自动进行autovacuum。
- 理论最大值为262143，实际最大值为动态值。计算公式为“262143 - max_inner_tool_connections - max_connections - job_queue_processes - 辅助线程数 – autovacuum的launcher线程数 - 1”，其中辅助线程数和autovacuum的launcher线程数由两个宏来指定，当前版本的默认值分别为20和2。

默认值：3

autovacuum_max_workers_hstore

参数说明：设置Autovacuum_max_workers里面，能同时运行的专用于清理hstore的自动清理线程的最大数量。

参数类型：SIGHUP

取值范围：整型

默认值：0

说明

当需要使用hstore表时，一定要同步修改一下几个参数的默认值，否则会导致hstore表性能严重劣化，推荐的修改配置是：

autovacuum_max_workers_hstore=3，autovacuum_max_workers=6，autovacuum=true。

enable_hstore_lightupdate

参数说明：用于控制是否开启对hstore表上的轻量化UPDATE（对Hstore表执行UPDATE时会自动判断是否需要轻量化UPDATE）。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启对hstore表上的轻量化UPDATE。
- off表示关闭对hstore表上的轻量化UPDATE。

默认值：off

enable_hstore_merge_keepgtm

参数说明：用于控制列存、hstore表上的autovacuum中的merge是否在GTM(全局事务管理)占槽位。

参数类型：SIGHUP

取值范围：布尔型

- true表示在GTM（全局事务管理）占槽位。
- false表示在GTM（全局事务管理）不占槽位。

默认值：true

hstore_buffer_size

参数说明：用于控制HStore的CU槽位数量，该槽位用于存储每个CU的更新链，能显著提升更新与查询效率。

为避免占用内存过大，系统会根据内存大小计算出一个槽位值，再与该参数相比取最小的值。

参数类型：POSTMASTER

取值范围：整型，100~10000000

默认值：100000