腾讯云数据仓库 TCHouse-P

数据接入

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

数据接入
外表读写 COS 使用说明

最近更新时间：2025-03-05 15:08:32

概述

腾讯云数据仓库 TCHouse-P 支持通过创建外表的方式读写 HDFS/COS/融合桶上的外表数据文件进行数据分析。本文主要介绍其使用方法。在使用之前确保已有 HDFS/COS/融合桶的用户账户并且网络能够联通。

创建 hdfs_fdw 插件

```
CREATE EXTENSION hdfs_fdw;
```

创建 SERVER

根据实际情况选择以下三种 Server 中的一种创建。

创建 COS Server

```
CREATE SERVER $cos_server名
FOREIGN DATA WRAPPER hdfs_fdw
OPTIONS (
  address 'cos://$bucketname',
  appid '$appid',
  access_keyid '$ak',
  secret_accesskey '$sk',
  region '$region',
  client_type 'cos'
);
```

创建 HDFS Server

```
CREATE SERVER $hdfs_server名
FOREIGN DATA WRAPPER hdfs_fdw
OPTIONS (
  address 'ofs://xxxxxx.chdfs.ap-guangzhou.myqcloud.com',
  appid '$appid',
  client_type 'hdfs'
);
```

创建融合桶 Server

```
CREATE SERVER $cosn_server名
FOREIGN DATA WRAPPER hdfs_fdw
OPTIONS (
  address 'cosn://$bucketname',
  appid '$appid',
  access_keyid '$ak',
  secret_accesskey '$sk',
  region '$region',
  client_type 'cosn'
);
```

创建 USER MAPPING

根据实际情况选择以下三种 User Mapping 中的一种创建。

```
CREATE USER MAPPING FOR $TDSQL-A用户 SERVER $cos_server名; -- 前面创建的
COS Server 的
CREATE USER MAPPING FOR $TDSQL-A用户 SERVER $hdfs_server名; -- 前面创建的
HDFS Server 的
CREATE USER MAPPING FOR $TDSQL-A用户 SERVER $cosn_server名; -- 前面创建的融
合桶 Server 的
```

pg 读 COS

创建非分区外表

表指定文件格式需要和远程读取的文件格式一致。

创建 CSV 格式的表

```
CREATE FOREIGN TABLE test_csv(id int, name TEXT)
SERVER $ServerName
OPTIONS (
  FORMAT 'csv',
  FOLDERNAME '$数据目录/',
  distribute 'shard'
);
```

创建 TEXT 格式的表

```
CREATE FOREIGN TABLE test_text(id int, name TEXT)
SERVER $ServerName
OPTIONS (
  FORMAT 'text',
  DELIMITER '$列分隔符',
  FOLDERNAME '$数据目录/',
  distribute 'shard'
);
```

创建 ORC 格式的表

```
CREATE FOREIGN TABLE orc_table(a bigint, b text, c float)
SERVER $ServerName
OPTIONS (
  FORMAT 'orc',
  FOLDERNAME '$数据目录/',
  distribute 'shard'
);
```

创建分区外表

当 Hive 建表语句为以下时：

```
CREATE FOREIGN TABLE login_logs_orc(l_id text,l_loginName text,l_date
text,year text,month text)
SERVER cosn_server
OPTIONS (
  FORMAT 'orc',
  FOLDERNAME '$数据目录/',
  distribute 'shard',
  PARTITION 'year,month'
);
```

查询外表并将数据导入内部表

```
insert into 内部表 select * from 外部表;
```

pg 写 COS

创建写的外部表

根据实际情况选择以下三种中的一种创建：

```
create foreign table f_csv(f1 int, f2 int)
server cos_server
options(FORMAT 'csv', DELIMITER ',', FOLDERNAME '/csv/', distribute
'shard') WRITE ONLY;

create foreign table f_txt(f1 int, f2 int)
server cos_server
options(FORMAT 'text', DELIMITER ',', FOLDERNAME '/txt/', distribute
'shard') WRITE ONLY;

create foreign table f_orc(f1 int, f2 int)
server cos_server
options(FORMAT 'orc', FOLDERNAME '/orc/', distribute 'shard') WRITE
ONLY;
```

pg 导入 COS

根据实际情况选择以下三种中的一种：

```
insert into f_csv select * from 内部表;
insert into f_txt select * from 内部表;
insert into f_orc select * from 内部表;
```

使用外表同步 EMR 数据

最近更新时间：2024-03-12 11:27:31

背景说明

在数据仓库的建设中，通常我们使用 Hive 处理原始数据（PB 级别），进行耗时较长的 ETL 工作，再将结果数据（TB 级别）交由准实时的计算引擎（例如腾讯云数据仓库 TCHouse-P）对接 BI 工具，保证报表的准实时展现。
本文介绍了如何将 EMR 上 Hive 的数据通过 COS 导入到腾讯云数据仓库 TCHouse-P的过程。

操作步骤

> **注意**
> - 腾讯云数据仓库 TCHouse-P不支持 ORC、Parquet 等格式，仅支持 CSV 等文本格式及其对应的 GZIP 压缩格式。
> - 腾讯云数据仓库 TCHouse-P 侧导入 COS 数据的效率与文件的个数有一定关系，建议个数为腾讯云数据仓库 TCHouse-P 计算节点个数的 N 倍。

1. 开启 EMR 读写对象存储能力
   首先需要保证 EMR 具备读写 COS 的能力，可在创建 EMR 时，勾选**开启**对象存储。

2. 创建 Hive 本地表并写入数据

```
create table hive_local_table(c1 int, c2 string, c3 int, c4 string);
insert into hive_local_table values(1001, 'c2', 99, 'c4'),(1002, 'c2',
100, 'c4'),(1003, 'c2', 101, 'c4'),(1004, 'c2', 100, 'c4'),(1005,
'c2', 101, 'c4')
```

3. 创建 Hive COS 外表

```
create table hive_cos_table(c1 int, c2 string, c3 int, c4 string)
row format delimited fields terminated by ','
LINES TERMINATED BY '\n'
stored as textfile location 'cosn://{bucket_name}/{dir_name}';
```

详细信息可以参考 EMR 文档 使用 Hive 在 COS/CHDFS 中创建库表 。

4. 将本地数据导入 COS

```
insert into hive_cos_table select * from hive_local_table;
```

成功写入后，可以在对应的 COS 目录下看到文件。

5. 在腾讯云数据仓库 TCHouse-P 侧创建 COS 外表

```
CREATE READABLE EXTERNAL TABLE  snova_cos_table (c1 int, c2
varchar(32), c3 int, c4 varchar(32))
LOCATION('cos:// {BUCKET}-{APPID}.cos.{REGION}.myqcloud.com/{PREFIX}
secretKey=**** secretId=***')
FORMAT 'csv';
```

详细内容可以参见 使用外表高速导入或导出 COS 数据 。

6. 在腾讯云数据仓库 TCHouse-P 侧创建本地表并导入数据

```
create table snova_local_table(c1 int, c2 text, c3 int, c4 text);
insert into snova_local_table select * from snova_cos_table;
```

使用 rule 规则实现云数据仓库 PostgreSQL upsert 操作

最近更新时间：2023-09-05 11:53:25

背景说明

腾讯云数据仓库 TCHouse-P 底层是基于 greenplum6 来构建，postgresql 内核为9.4版本，目前并不能很好支持 postgresql 的 `insert .. on conflict` 特性，所以对于 upsert 场景需要采用额外的方式来进行处理，这里提供一种利用 postgresql rule 特性来进行 upsert 的方法。

规则介绍

PostgreSQL 规则系统允许在更新、插入、删除时执行一个其它的预定义动作。简单的说，规则就是在指定表上执行指定动作的时候，将导致一些额外的动作被执行。另外，一个 `INSTEAD` 规则可以用另外一个命令取代特定的命令，或者完全不执行该命令。规则还可以用于实现表视图。规则实际上只是一个命令转换机制，或者说命令宏。这种转换发生在命令开始执行之前。
详细信息可参考 rule 使用手册 。

upsert rule

如果需要实现 upsert 的操作，那么需要这样一条规则：当进行 insert 操作时，判断是否已经有相应的记录，如果存在记录则改为进行 update 操作，如果不存在记录则进行正常 insert 操作。
下面以一个数据库实例来进行说明：
创建一个测试数据库。

```
CREATE TABLE my_test(
   id integer,
   num1 integer,
   num2 decimal,
   str1 varchar(20),
   str2 text,
   PRIMARY KEY(id)
) distributed by (id);
```

然后给表增加 rule 规则。

```
create rule r1 as on insert to my_test where exists (select 1 from
my_test t1 where t1.id=NEW.id limit 1) do instead update my_test set
num1=NEW.num1,num2=NEW.num2,str1=NEW.str1,str2=NEW.str2 where id=NEW.id;
```

这条 rule 命令的含义就是针对 insert 操作，如果新的 insert 语句的 id 是存在，那么就直接用新 insert 里面的值 update 原来的数据，语句中的 NEW.XXX，即新 insert 语句的值，操作完成后可以看到。数据表中存在 rule 规则，接着进行 insert 操作，如果 id 存在，那么不会因为主键约束报错，而是进行 update 操作。

```
\d my_test
                                    Table "public.my_test"
 Column |          Type           | Collation | Nullable |
Default
--------+-------------------------+-----------+----------+----------------
---------------------
 id     | integer                 |           | not null |
nextval('my_test_id_seq'::regclass)
 num1   | integer                 |           |          |
 num2   | numeric                 |           |          |
 str1   | character varying(20)   |           |          |
 str2   | text                    |           |          |
Indexes:
    "my_test_pkey" PRIMARY KEY, btree (id)
Rules:
    r1 AS
    ON INSERT TO my_test
   WHERE (EXISTS ( SELECT 1
     FROM my_test my_test_1
     WHERE my_test_1.id = new.id
     LIMIT 1)) DO INSTEAD  UPDATE my_test SET num1 = new.num1, num2 =
new.num2, str1 = new.str1, str2 = new.str2
```

使用注意

rule 规则使用存在一定局限，如下所示：

1. 因为 exists 对于 insert 批量插入处理不完善，如果语句没有设置唯一约束或者主键约束，可能在 insert 批量插入时产生重复数据，应尽量避免使用批量 insert，使用时也要避免需要判断 upsert 的字段不重复，或者对需要判断的字段增加唯一约束。例如下面这种批量操作，如果 id 没有主键约束，那么可能执行后存在重复数据。

   ```
   insert into my_test
   (id,num1,num2,str1,str2)values(1,2,1.0,'111','555'),
   (1,3,2.0,'111','666');
   ```

2. rule 不支持 COPY 语句，COPY 语句效果和 insert 批处理类似，都可能导致重复数据。
3. 在设置 update 规则时，如果设置了 rule 用法，但是 insert 语句没有传 num1 和 num2 字段，这两个字段更新后会为空值，导致原数据丢失。

```
update my_test set
num1=NEW.num1,num2=NEW.num2,str1=NEW.str1,str2=NEW.str2
```