# 课堂实验

## 一、基本信息

| 文档编号 | | 文档版本 | 1.0 |
|---|---|---|---|
| 实验名称 | Inceptor SQL 使用方法 | | |
| 所属课程 | 第 11 讲 分布式 SQL 引擎 Inceptor | 认证等级 | 数据工程师（初级） |
| 授课形式 | 上机实验 | 实验批次 | 第 8 次 / 共 11 次 |

## 二、实验目的

- 掌握 Waterdrop 安装和使用方法。
- 掌握 Incepor SQL 基本用法。
- 理解 Inceptor 事务的原子性、一致性和隔离性。
- 理解 Inceptor ORC 表、ORC 事务表、Holodesk 表、分区、分桶的含义。

## 三、实验准备

- 下载并安装 TDH Client。
- 下载并安装 Waterdrop，Inceptor SQL 语句在 Waterdrop 中运行。
- 实验目录与命名规划

（1）本地目录

工作目录：/mnt/disk1/{student_name}

（2）HDFS 目录

工作目录：/tmp/{student_name}

Inceptor 外表目录：/tmp/{student_name}/inceptor_data

（3）Inceptor 数据库名：{db_student_name}

（4）{student_name}为变量，代表学员姓名全拼

## 四、实验内容

### 1、安装 Waterdrop，连接 Inceptor

- 任务：安装 Waterdrop，连接 Inceptor，查看数据库、表等对象。
- 步骤

1. 使用 sz 命令，将文件服务器中的 Waterdrop、Inceptor SDK 和 JDK 下载到客户端

2. 在客户端安装 JDK，并检查 JAVA_HOME 和 PATH 环境变量
3. 解压 Waterdrop，执行 waterdrop.exe，启动 Waterdrop
4. 添加 Inceptor SDK 驱动
5. 新建连接

（1）连接类型：Inceptor Server 2
（2）服务器地址：集群第一个节点 IP
（3）端口：10000
（4）不要安装 Guardian 服务，否则会启动 kerberos 认证，导致连接失败

## 2、验证 Inceptor 事务的 ACID 属性

（1）全局设置

- 任务：设置开启事务，并创建普通 ORC 表。
- 步骤

SQL：

```
// 设置开启事务
1. set transaction.type=inceptor;
// 设置 PLSQL 编译器不检查语义
2. set plsql.compile.dml.check.semantic=false;
// 创建数据库和 ORC 表
3. create database {db_student_name};
4. use {db_student_name};
5. drop table if exists orc_table;
6. create table orc_table(key int, value string) stored as orc;
```

（2）验证事务原子性

- 任务：在事务中同时向 ORC 事务表和 ORC 表插入数据，然后检查 ORC 事务表中是否有数据。因为 ORC 表不支持单条插入，会造成事务回滚，所以 ORC 事务表应该为无数据。
- 步骤

SQL：

```
// 创建 ORC 事务表
1. drop table if exists atomicity_table;
2. create table atomicity_table(key int, value string) clustered by(key) into 8 buckets stored
   as orc tblproperties('transactional'='true');
// 创建存储过程，验证事务原子性
3. create or replace procedure test_atomicity()
   is
   declare
      cnt int:=0;
   begin
      begin transaction;
         insert into atomicity_table values(1,'src1');
```

```
            insert into atomicity_table values(2,'src2');
            insert into orc_table values(1,'test');
        commit;
    exception
        when others then
            rollback;
            dbms_output.put_line('insert failed.');
            dbms_output.put_line('There is nothing be altered.');
    end;
// 执行存储过程
4. begin
        set_env('plsql.catch.hive.exception',true);
        test_atomicity();
    end;
// 查看 ORC 事务表中
5. select * from atomicity_table;
```

（3）验证事务一致性

- 任务：在事务中向 ORC 表插入数据，然后检查 ORC 事务表 consistency_table 中的数据是否被更新。预期结果为数据未更新。
- 步骤

```
SQL：
// 创建 ORC 事务表，并插入两条数据
1. drop table if exists consistency_table;
2. create table consistency_table (key int, value string) clustered by(key) into 8 buckets
   stored as orc tblproperties('transactional'='true');
3. insert into consistency_table values(1,10.5);
4. insert into consistency_table values(2,20.5);
// 创建存储过程，验证事务一致性
5. create or replace procedure test_consistency()
    is
    declare
        cnt int:=0;
    begin
        begin transaction;
            update consistency_table set value = 30.5 where key = 1;
            insert into orc_table values(3,50.5);
            update consistency_table set value = 40.5 where key = 2;
        commit;
    exception
        when others then
            rollback;
            dbms_output.put_line('update failed.');
```

```
            dbms_output.put_line('There is nothing be altered.');
        end;
    // 执行存储过程
    6. begin
            set_env('plsql.catch.hive.exception',true);
            test_consistency();
        end;
    // 查看 ORC 事务表中
    7. select * from consistency_table;
```

（4）验证事务隔离性

- 任务：在第一个事务中 Update ORC 事务表但未提交，在第二个事务中再次 Update 表，最后集中提交事务。预期结果为第二个事务会等待第一个事务先提交。
- 步骤

```
SQL：
    // 创建 ORC 事务表，并插入一条数据
    1. drop table if exists isolation_table;
    2. create table isolation_table(key int, value string) clustered by(key) into 8 buckets stored as
       orc tblproperties('transactional'='true');
    3. insert into isolation_table values(1,100.5);
    // 验证事务隔离性
    4. create or replace procedure test_isolation()
        is
        declare
            cnt int:=0;
        begin
            begin transaction;
                update isolation_table set value='updated1' where key=1;
            begin transaction;
                update isolation_table set value='updated2' where key=1;
            commit;
            commit;
        exception
            when others then
                rollback;
                dbms_output.put_line('update failed.');
                dbms_output.put_line('There is nothing be altered.');
    end;
    // 执行存储过程
    5. begin
            set_env('plsql.catch.hive.exception',true);
            test_isolation();
        end;
```

```
// 查看 ORC 事务表中
6. select * from isolation_table;
```

## 3、创建 ORC 分区分桶表

- 任务：创建范围分区分桶表，并存储为 ORC 格式。
- 步骤

```
SQL：
  // 创建范围分区分桶表，分区键=sj，分桶键=mbbh
  1. create table hq_ais_history_data_orc_bucket (
        cbm string,
        csx int,
        cwjqd int,
        dzdwzz int,
        gjmc string,
        hh string,
        hs double,
        hwlx int,
        hx double,
        hxzt int,
        imobm string,
        mbbh string,
        mdd string,
        txzt int,
        xxlx int,
        xxly int,
        yjddsj string,
        zdjss double,
        zxl int,
        lat double,
        lon double,
        mbsj int
     )
     partitioned by range (sj string) (
        partition values less than ("2014-11-04 23:59:59"),
        partition values less than ("2014-11-05 23:59:59"),
        partition values less than ("2014-11-06 23:59:59"),
        partition values less than ("2014-11-07 23:59:59"),
        partition values less than ("2014-11-08 23:59:59"),
        partition values less than ("2014-11-09 23:59:59"),
        partition values less than ("2014-11-10 23:59:59"),
        partition values less than ("2014-11-11 23:59:59"),
        partition values less than ("2015-08-05 23:59:59")
```

```
)
clustered by (mbbh) into 23 buckets
stored as orc;
```

## 4、Holodesk 表性能测试

- 任务：先创建 Inceptor 外表，再基于外表创建 Holodesk 表，最后比较二者的查询性能。
- 步骤

```
Linux：
  // 登录文件服务器，将 data.csv.zip 复制到集群节点中
  1. cd /mnt/disk1/de_training
  2. scp data.csv.zip 172.16.140.85:/mnt/disk1/{student_name}
  // 登录集群节点，解压缩 data.csv.zip
  3. cd /mnt/disk1/{student_name}
  4. unzip data.csv.zip
  // 执行 TDH Client 的 init.sh 脚本，启动 TDH Client
  5. source {TDH_Client_install_dir}/init.sh
  // 将 Hadoop 当前用户设置为 hdfs，进行访问授权
  6. export HADOOP_USER_NAME=hdfs
  // 将数据文件 data.csv 上传到 HDFS 目录中
  7. hadoop fs -mkdir -p /tmp/{student_name}/inceptor_data
  8. hadoop fs -put data.csv /tmp/{student_name}/inceptor_data
SQL：
  // 在 Inceptor 中创建外表，并查看数据
  1. drop table if exists ext_table;
  2. create external table ext_table(rowkey string, num int, country int, rd int) row format
     delimited fields terminated by ',' location '/tmp/{student_name}/inceptor_data';
  3. select * from ext_table limit 10;
  // 基于外表 ext_table 创建 Holodesk 表
  4. drop table if exists holo_table;
  5. create table holo_table tblproperties("cache"="ram", "holodesk.dimension"="country,
     num") as select * from ext_table distribute by country, num sort by country, num;
  // 比较两种表的查询性能
  6. select avg(rd) from holo_table group by num, country;
  7. select avg(rd) from ext_table group by num, country;
```