Hive 基本操作

DDL、DML、DQL

数据定义 - DDL

（1）建表（CREATE）的语法如下：

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...)
  [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [ROW FORMAT row_format]
  [STORED AS file_format]
  [LOCATION hdfs_path]
```

上面的一些关键字解释：
CREATE TABLE 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXIST 选项来忽略这个异常
EXTERNAL 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION）
LIKE 允许用户复制现有的表结构，但是不复制数据
COMMENT 可以为表与字段增加描述
ROW FORMAT 用户在建表的时候可以自定义 SerDe 或者使用自带的 SerDe。如果没有指定 ROW FORMAT 或者 ROW FORMAT DELIMITED，将会使用自带的 SerDe。在建表的时候，用户还需要为表指定列，用户在指定表的列的同时也会指定自定义的 SerDe，Hive 通过 SerDe 确定表的具体的列的数据。
STORED AS 如果文件数据是纯文本，可以使用 STORED AS TEXTFILE。如果数据需要压缩，使用 STORED AS SEQUENCE 。

- **创建简单表**

```
hive> CREATE TABLE xiniu(
        id          INT,
        email       STRING,
        name        STRING);
```

- **创建外部表**

```
hive> CREATE EXTERNAL TABLE xiniu2(
        id              INT,
        email           STRING,
        name              STRING
        )
        LOCATION '/home/hive/external';
```

和简单表相比较，可以发现外部表多了 external 的关键字说明以及 LOCATION 指定外部表存放的路径（如果没有 LOCATION，Hive 将在 HDFS 上的/user/hive/warehouse 文件夹下以外部表的表名创建一个文件夹，并将属于这个表的数据存放在这里）。

- **创建分区表**

为了避免 Hive 在查询时扫描全表，增加没有必要的消耗，因此在建表时加入 partition。

```
hive> CREATE TABLE xiniu3(
        id              INT,
        email           STRING,
        name              STRING
        )
        PARTITIONED BY(sign_date STRING,age INT);
```

可以看到，我们使用了 sign_date 和 age 两个字段作为分区列。但是，我们必须先创建这两个分区，才能够使用。

```
hive> ALTER TABLE xiniu3 add partition(sign_date='20160720',age=20);
```

- **创建 Bucket 表**

Hive 中的 table 可以拆分成 partiton，table 和 partition 又可以进一步通过 CLUSTERED BY 分成更小的文件 bucket，这样使得多个文件可以在 map 上同时启动。 首先需要设置环境变量

```
hive>set hive.enforce.bucketing = true;
hive> CREATE TABLE xiniu4(
        id              INT,
        email           STRING,
        name              STRING,
        age              INT
        )
        PARTITIONED BY (sign_date STRING)
        CLUSTERED BY(id) SORTED BY(age) INTO 5 BUCKETS
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

（2）修改（ALTER）

包括修改表名、添加删除列等操作。

```
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
ALTER TABLE name DROP [COLUMN] column_name
ALTER TABLE name CHANGE column_name new_name new_type
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
```

（3）删除（DROP）

主要是删除表。

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

（4）展示（SHOW）、描述（DESCRIBE）

显示表相关信息。

```
show tables;
show databases;
show functions;
describe [EXTENDED] table_name[DOT col_name]
```

数据操作 - DML

Hive 不支持 insert 语句进行逐条插入，也不支持 update 修改数据。首先需要在 Hive 中建好表，再使用 load 语句将数据导入，数据一旦导入就不可以修改。导入数据的方式可大致分为以下几种：

（1）、从本地文件系统或 HDFS 中导入数据到 Hive 表；
Hive 加载数据到表中时，不做任何转换。加载操作目前只是单纯地复制/移动数据文件到 Hive 表格的对应位置。

其中 `LOCAL` 代表是从本地文件系统还是从 `HDFS` 中取数据。`OVERRITE` 代表覆盖表中之前的数据。

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

（2）、从别的表中查询出相应的数据并导入到 Hive 表中。动态插入分区表时常用此方法。

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)]
```

```
select_statement1 FROM from_statement
```

（3）、在创建表的时候通过从别的表中查询出相应的记录并插入到所创建的表中。
在实际情况中，表的输出结果可能太多，不适于显示在控制台上，这时候，将 Hive 的查询输出结果直接存在一个新的表中是非常方便的，我们称这种情况为 CTAS（create table .. as select）

```
CREATE TABLE tablename AS select_statement1 FROM from_statement
```

数据查询 - DQL

(1)基本的 Select 操作

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
      FROM table_reference
      [WHERE where_condition]
      [GROUP BY col_list]
      [     CLUSTER BY col_list
        | [DISTRIBUTE BY col_list] [SORT BY| ORDER BY col_list]
      ]
     [LIMIT number]
```

一个 SELECT 语句可以是 UNION 查询的一部分，也可以是另一个查询的子查询。
table_reference 表示的是所查询的表。
表名和列名大小写敏感。
使用 ALL 和 DISTINCT 选项区分对重复记录的处理。默认是 ALL，表示查询所有记录。
DISTINCT 表示去掉重复的记录；
LIMIT number 可以限制查询的记录数

简单的查询例子：

```
SELECT * FROM xiniu;
SELECT COUNT(*) FROM XINIU WHERE ID=1;
```

(2)基于 Partition 的查询

一般 SELECT 查询会扫描整个表，使用 PARTITIONED BY 子句建表，查询就可以利用分区剪枝（input pruning）的特性。 Hive 当前的分区剪枝，只有分区断言出现在离 FROM 子句最近的那个 WHERE 子句中，才会启用分区剪枝。
下面是例子，按 SIGN_DATE 分区：

```
select * from xiniu3 where sign_date == '20160720';
```

(3)HAVING 查询

Hive 在 0.7.0 版本中添加了对 HAVING 语句的支持，在旧版本的 Hive 中，使用一个子查询也可以实现相同的效果。

```
SELECT col1 FROM t1 GROUP BY col1 HAVING SUM(col2) > 10
```

等价于

```
SELECT col1 FROM (SELECT col1, SUM(col2) AS col2sum FROM t1 GROUP BY col1) t2 WHERE t2.col2sum > 10
```

(4)Join 查询

Join 的语法如下：

```
join_table:
    table_reference JOIN table_factor [join_condition]
  | table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference join_condition
  | table_reference LEFT SEMI JOIN table_reference join_condition
  | table_reference CROSS JOIN table_reference [join_condition] (as of Hive 0.10)

table_reference:
    table_factor
  | join_table

table_factor:
    tbl_name [alias]
  | table_subquery alias
  | ( table_references )

join_condition:
    ON equality_expression ( AND equality_expression )*

equality_expression:
    expression = expression
```

hive 只支持等连接（equality joins）、外连接（outer joins）、左半连接（left semi joins）。hive 不支持非相等的 join 条件，因为它很难在 map/reduce job 中实现这样的条件。而且，hive 可以 join 两个以上的表。

示例

1. 切换到 hadoop 用户。准备数据文件（如文件夹不存在请自行新建）

cd /hadoop/hive/data/student
vi student.txt

复制以下内容并保存，注意每个属性之间的间隔是空格。

1 2017 adam
2 2017 bob
3 2017 clark
4 2016 dannis
5 2016 emma

2. 切换到 root 用户，确认开启了 ssh 和 mysql 服务
3. 切换到 hadoop 用户，启动 hadoop 进程，启动 hive shell
4. 一般情况下我们都建外表来进行数据分析，因为外表的数据独立，删除表也不会删除源数据。首先建立外表 stu，作为原始数据表

CREATE EXTERNAL TABLE stu (id INT, year INT, name STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';

建表成功如下示：

```
hive> CREATE EXTERNAL TABLE stu (id INT, year INT, name STRING)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';
OK
Time taken: 1.461 seconds
```

5. 并读取数据到该表。注意路径要写文件夹，不要写具体文件！

LOAD DATA LOCAL INPATH '/hadoop/hive/data/student' INTO TABLE stu;

```
hive> LOAD DATA LOCAL INPATH '/hadoop/hive/data/student' INTO TABLE stu;
Copying data from file:/hadoop/hive/data/student
Copying file: file:/hadoop/hive/data/student/student.txt
Loading data to table default.stu
Table default.stu stats: [numFiles=1, numRows=0, totalSize=63, rawDataSize=0]
OK
Time taken: 1.852 seconds
```

查看表数据：

*SELECT * FROM stu;*

6. 开启动态分区

为了方便的使用分区表，需要开启动态分区，进行一些配置。简而言之是为了在导入数据到分区表时，由于部分 column 和所定义表的 partition 匹配，从而自动映射到正确的分区中去，不需要一个一个 partition 的去 add。

设置 hive.exec.dynamic.partition 参数值为 true，默认值为 false

```
SET hive.exec.dynamic.partition=true;
```

动态分区可以允许所有的分区列都是动态分区列，但是要首先设置一个参数 hive.exec.dynamic.partition.mode。它的默认值是 strict，即不允许分区列全部是动态的，这是为了防止用户有可能原意是只在子分区内进行动态建分区，但是由于疏忽忘记为主分区列指定值了，这将导致一个 dml 语句在短时间内创建大量的新的分区（对应大量新的文件夹），对系统性能带来影响。

```
SET hive.exec.dynamic.partition.mode=nonstrict;
```

7. 新建用于分析的表 stu_part

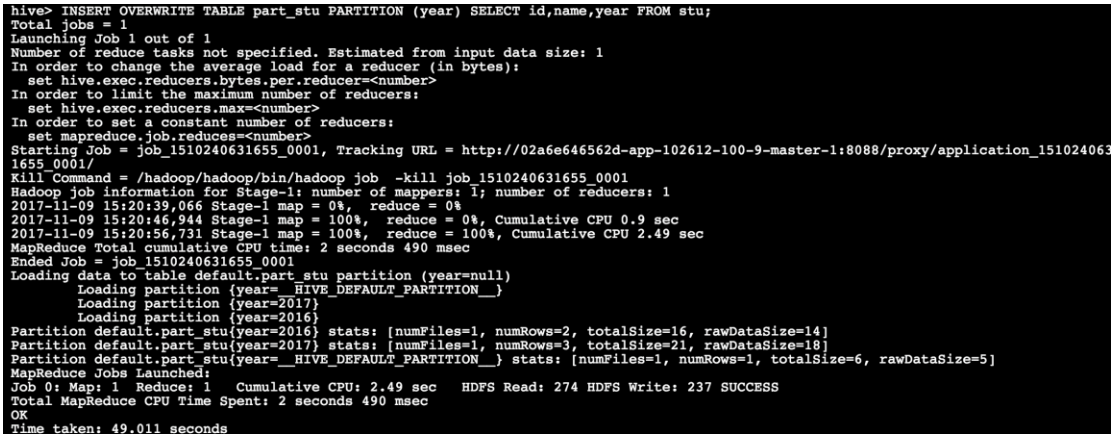
```
CREATE EXTERNAL TABLE part_stu (id INT, name STRING)
PARTITIONED BY (year INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';
```

注意相比 stu 表，stu_part 把 year 列提到 partition 去了。

8. 读取原始表的数据到 stu_part。

year 作为 partition，要放在最后。

```
INSERT OVERWRITE TABLE part_stu PARTITION (year) SELECT id,name,year FROM stu;
```



9. 使用查询语句可以看到分区表已准备完毕。

```
SELECT * FROM part_stu;
SELECT * FROM part_stu WHERE year=2017;
```

作业

https://github.com/jihongfei/DataWarehouseCourse/blob/master/resourse/csv/Crime_Data_2014-2015_without_comma.csv
上为收集到的美国犯罪 2014-2015 统计数据, 以 csv（Comma-Separated Values）格式保存, 最好先用 excel 查看。各列依次为：

日期
发生时刻（1-2359，前两位是小时后两位是分钟）
犯罪类型
受害者年龄
受害者性别
武器代码
武器描述
发生地点
发生地点经纬度

请尝试用 wget 命令将文件下载到本地，并建外表 us_crime，读取该 csv 数据到该表中（若存入 hdfs，load 的时候记得去掉 local 关键字）

回答以下问题：
1. 写出建表语句
2. 14-22，22-06，这两个时间段哪个犯罪起数更多，给出 sql 语句和查询结果
3. （可选）其他发现，如某种类型犯罪的特点