Hive 基本操作

一、实验介绍

1.1 实验内容

hive表的介绍包括：普通表，外部表，分区表，Bucket表，表的命名，表增加、删除分区增加、更新列修改列的名字、类型、位置、注释，增加表的元数据信息等。

1.2 实验知识点

- 数据定义 - DDL
- 数据操作 - DML
- 数据查询 - DQL

1.3 实验环境

- Hive V2.0.0
- hadoop2.4.1
- Xfce终端

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有hadoop基础的用户，熟悉linux基础知识

二、实验步骤

本次讲解 Hive 基本操作分为以下几个要点：

1. **数据定义 - DDL**
2. **数据操作 - DML**
3. **数据查询 - DQL**

三、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：

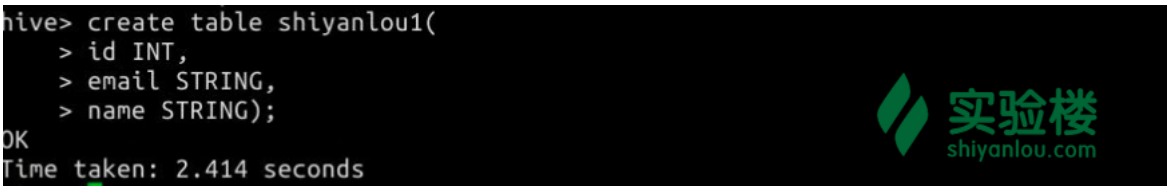
- **CREATE TABLE** 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXIST 选项来忽略这个异常
- **EXTERNAL** 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION）
- **LIKE** 允许用户复制现有的表结构，但是不复制数据
- **COMMENT** 可以为表与字段增加描述
- **ROW FORMAT** 用户在建表的时候可以自定义 SerDe 或者使用自带的 SerDe。如果没有指定 ROW FORMAT 或者 ROW FORMAT DELIMITED，将会使用自带的 SerDe。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 。

（2）建表（CREATE）

- 创建普通表
- 创建外部表
- 创建分区表
- 创建 Bucket 表

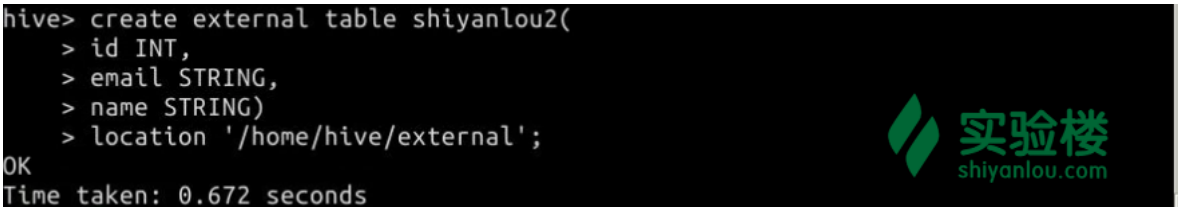
- 创建简单表：

```
hive> CREATE TABLE shiyanlou1(
        id        INT,
        email     STRING,
        name      STRING);
```



- 创建外部表：

```
hive> CREATE EXTERNAL TABLE shiyanlou2(
        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 shiyanlou3(
        id            INT,
        email         STRING,
        name          STRING
        )
        PARTITIONED BY(sign_date STRING,age INT);
```

```
hive> create  table shiyanlou3(
    > id INT,
    > email STRING,
    > name STRING)
    > partitioned by(sign_date STRING,age INT);
OK
Time taken: 2.905 seconds
```

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 shiyanlou3 add partition(sign_date='20160720',age=20);
```

```
hive> alter table shiyanlou3 add partition(sign_date='20160720',age =20);
OK
Time taken: 0.67 seconds
hive>
```

- 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 shiyanlou4(
        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 ',';
```

```
hive> create  table shiyanlou4(
    > 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 ',';
OK
Time taken: 0.42 seconds
```

（3）修改表结构

- 重命名表
- 增加、删除分区
- 增加、更新列
- 修改列的名字、类型、位置、注释
- 增加表的元数据信息
- ...

- 复制一个空表

```
CREATE TABLE empty
LIKE shiyanlou3;
```

```
hive> create  table empty like shiyanlou3;
OK
Time taken: 0.649 seconds
```

- 删除表

```
DROP TABLE [IF EXISTS] table_name [RESTRICT|CASCAD];
```

- 重命名表

```
ALTER TABLE table_name RENAME TO new_table_name
```

- 增加、删除分区

```
# 增加
ALTER TABLE table_name ADD [IF NOT EXISTS] partition_spec [ LOCATION 'location1' ] partition_spec [ LOCATION 'location2' ] ...
partition_spec:
  : PARTITION (partition_col = partition_col_value, partition_col = partiton_col_value, ...)

# 删除
ALTER TABLE table_name DROP
    partition_spec, partition_spec,...
```

- 增加、更新列

```
# ADD 是代表新增一字段，字段位置在所有列后面(partition列前)
# REPLACE 则是表示替换表中所有字段。
ALTER TABLE table_name ADD|REPLACE COLUMNS (col_name data_type [COMMENT col_comment], ...)
```

- 修改列的名字、类型、位置、注释

```
# 这个命令可以允许改变列名、数据类型、注释、列位置或者它们的任意组合
ALTER TABLE table_name CHANGE [COLUMN]
col_old_name col_new_name column_type
[COMMENT col_comment]
[FIRST|AFTER column_name]
```

- 增加表的元数据信息

```
# 用户可以用这个命令向表中增加元数据信息 metadata
ALTER TABLE table_name SET TBLPROPERTIES table_properties table_properties:
    : (property_name = property_value, ...)
```

- 改变文件格式和组织

```
ALTER TABLE table_name SET FILEFORMAT file_format
ALTER TABLE table_name CLUSTERED BY(col_name, col_name, ...)
    [SORTED BY(col_name, ...)] INTO num_buckets BUCKETS
```

- 创建、删除视图

```
# 创建视图
CREATE VIEW [IF NOT EXISTS] view_name [ (column_name [COMMENT column_comment], ...) ][COMMENT view_comment][TBLPROPERT
IES (property_name = property_value, ...)]
AS SELECT ...

# 删除视图
DROP VIEW view_name
```

- 创建、删除函数

```
# 创建函数
CREATE TEMPORARY FUNCTION function_name AS class_name

# 删除函数
DROP TEMPORARY FUNCTION function_name
```

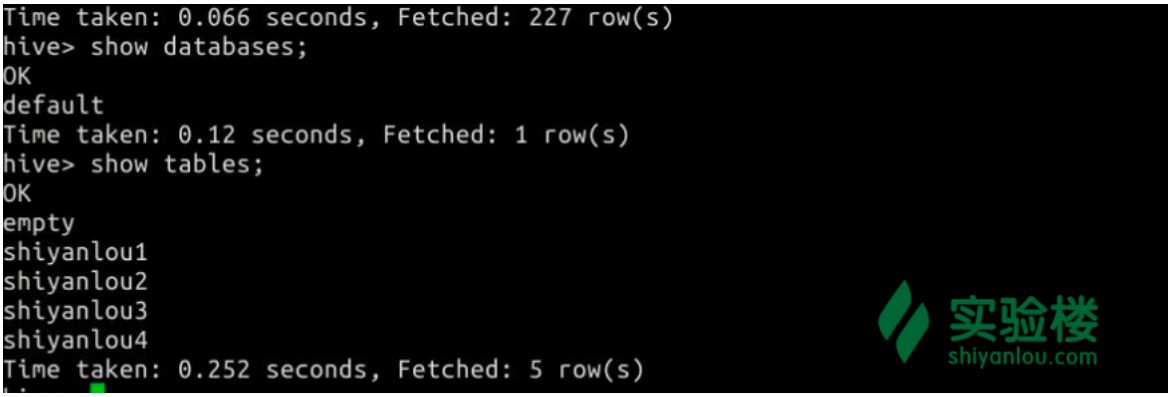

- 展示、描述语句

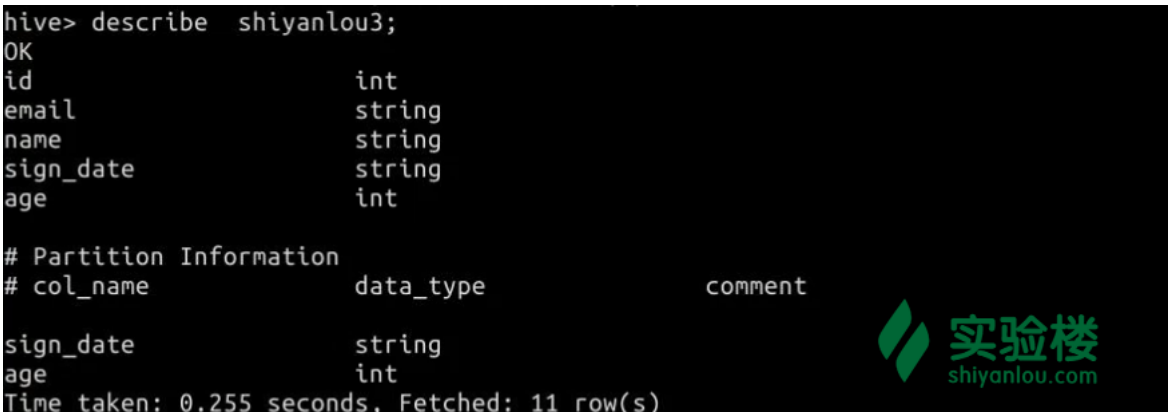
```
# 显示 表
show tables;

# 显示 数据库
show databases;

# 显示 函数
show functions;

# 描述 表/列
describe [EXTENDED] table_name[DOT col_name]
```





四、数据管理操作 - 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。

（1）加载数据到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。

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

- **filepath**
  - 相对路径，例如：project/data1
  - 绝对路径，例如： /user/hive/project/data1
  - 包含模式的完整 URI，例如：hdfs://namenode:9000/user/hive/project/data1
- **LOCAL 关键字**
  - 指定了LOCAL 即本地 load 命令会去查找本地文件系统中的 filepath. 如果发现是相对路径，则路径会被解释为相对于当前用户的当前路径。用户也可以为本地文件指定一个完整的 URI，比如：file:///user/hive/project/data. 此时 load 命令会将 filepath 中的文件复制到目标文件系统中。目标文件系统由表的位置属性决定。被复制的数据文件移动到表的数据对应的位置

- 没有指定LOCAL 如果 filepath 指向的是一个完整的 URI，hive 会直接使用这个 URI. 否则如果没有指定 schema 或者 authority，Hive 会使用在 hadoop 配置文件中定义的 schema 和 authority，fs.default.name 指定了 Namenode 的 URI. 如果路径不是绝对的，Hive 相对于 /user/ 进行解释。 Hive 会将 filepath 中指定的文件内容移动到 table （或者 partition）所指定的路径中。

- **OVERWRITE**
  - 使用 OVERWRITE 关键字，目标表（或者分区）中的内容（如果有）会被删除，然后再将 filepath 指向的文件/目录中的内容添加到表/分区中。如果目标表（分区）已经有一个文件，并且文件名和 filepath 中的文件名冲突，那么现有的文件会被新文件所替代。

- 实例：

```
hive> LOAD DATA LOCAL INPATH './examples/files/kv1.txt' OVERWRITE INTO TABLE pokes;
```

（2）将查询结果插入Hive表

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement
```

- 多插入模式

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select_statement1

INSERT OVERWRITE TABLE tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2
```

- 自动分区模式

```
INSERT OVERWRITE TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement FROM from_statement
```

（3）将查询结果写入HDFS文件系统

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1
  [ROW FORMAT row_format] [STORED AS file_format] (Note: Only available starting with Hive 0.11.0)
  SELECT ... FROM ...
```

- 多插入模式

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1 select_statement1
```

数据写入文件系统时会进行文本序列化，且每列用 `^A` 来区分， `\n` 换行。如果任何一列不是原始类型，那么这些将会被序列化为 JSON 格式。

（4）从SQL获取数据插入Hive表

```
INSERT INTO TABLE tablename [PARTITION (partcol1[=val1], partcol2[=val2] ...)] VALUES values_row [, values_row ...]
```

value_row 应用为（value [, value ...]），其中value为null或是任意有效的SQL 语句。

五、数据查询操作 - DQL

SQL操作：

- 基本的 Select 操作
- 基于 Partition 的查询
- HAVING 查询
- Join 查询

（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 shiyanlou`

（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。

下面是两个例子,page_view根据date分区：

```
SELECT page_views.*
FROM page_views
WHERE page_views.date >= '2008-03-01' AND page_views.date <= '2008-03-31'
```

```
SELECT page_views.*
FROM page_views JOIN dim_users
ON (page_views.user_id = dim_users.id AND page_views.date >= '2008-03-01' AND page_views.date <= '2008-03-31')
```

（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。

六、实验总结

本实验对Hive的三个操作，DDL，DML和DQL进行了最基本的介绍，其中没有详解的地方还有很多，例如：GROUP BY查询，JOIN查询等等，这些内容可以随着对Hive的进一步使用，慢慢地加以学习和理解。

七、参考文档

- 《Hadoop实战 第2版》 (http://book.douban.com/subject/20275953/) 陆嘉恒，机械工业出版社；
- Hadoop Hive sql语法详解 (http://blog.csdn.net/hguisu/article/details/7256833)；
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual)

上一节 (/courses/38/labs/766/document) 下一节 (/courses/38/labs/775/document)