实验四　大数据分析平台中Hive表操作

（一）实验目的

1. 理解Hive的体系机构；
2. 熟悉Hive的工作原理；
3. 掌握Hive表的相关操作。

（二）实验要求

1. 完成Hive的DDL操作；
2. 在Hive中实现新建、显示、修改和删除表等操作。

（三）实验环境

1. 大数据分析实验系统（FSDP）；
2. CentOS 6.7；
3. Hadoop 2.7.1；
4. Hive 1.2.1。

（四）实验步骤

1. 熟悉Hive原理；
2. Shell环境下的Hive表操作。

1、熟悉Hive原理

Hive是Hadoop大数据生态圈中的数据仓库，其提供以表格的方式来组织与管理HDFS上的数据、以类SQL的方式来操作表格里的数据，Hive的设计目的是能够以类SQL的方式查询存放在HDFS上的大规模数据集，不必开发专门的MapReduce应用。

Hive本质上相当于一个MapReduce和HDFS的翻译终端，用户提交Hive脚本后，Hive运行时环境会将这些脚本翻译成MapReduce和HDFS操作并向集群提交。

当用户向Hive提交编写的HiveQL后，首先，Hive运行时环境会将这些脚本翻译成MapReduce和HDFS操作；其次，Hive运行时环境使用Hadoop命令行接口向Hadoop集群提交这些MapReduce和HDFS操作；最后，Hadoop集群逐步执行这些MapReduce和HDFS操作，整个过程可概括如下：

(1) 用户编写HiveQL并向Hive运行时环境提交改HiveQL。

(2) Hive运行时环境将该HiveQL翻译成MapReduce和HDFS操作。

(3) Hive运行时环境调用Hadoop命令行接口或程序接口，向Hadoop集群提交翻译后的HiveQL。

(4) Hadoop集群执行HiveQL翻译后的MapReduce-APP或HDFS-APP。

由上述执行过程可知，Hive的核心是其运行时环境，该环境能够将类SQL语句编译成MapReduce。

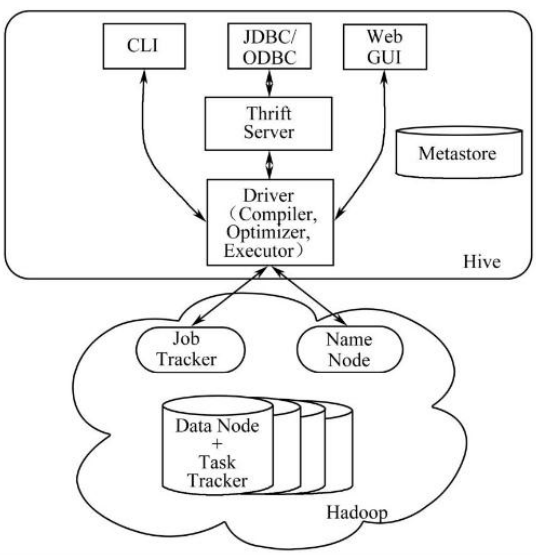
Hive构建在基于静态批处理的Hadoop之上，例如，联机事务处理（OLTP）。Hive查询操作过程严格遵守Hadoop MapReduce的作业执行模型， Hive将用户的HiveQL语句通过解释器转换为MapReduce作业提交到Hadoop集群上，Hadoop监控作业执行流程，然后返回作业执行结果给用户。Hive并非为联机事务处理而设计，Hive并不提供实时的查询和基于行级的数据更新操作。Hive的最佳使用场合是大数据集的批处理作业，例如，网络日志分析。

Hive没有专门的数据存储格式，也没有为数据建立索引，用户可以非常自由地组织Hive中的表，只需要在创建表的告诉Hive中的列分隔符和行分隔符，Hive就可以解析数据。

Hive中所有的数据都存储在HDFS中，Hive包含以下数据模型：表（Table），外部表（External Tabale），分区（Partition），桶（Bucket）。

Hive中Table和数据库中Table在概念上是类似的，每一个Table在Hive中都有一个相应的目录存储数据。

Hive架构与基本组成如下图所示。

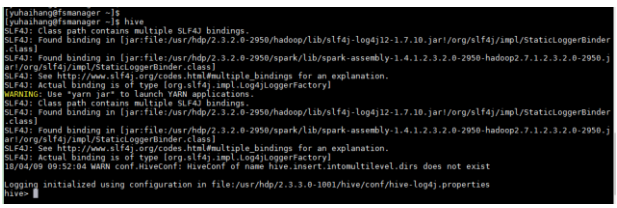


实验四　大数据分析平台中Hive表操作

2、Shell环境下的Hive表操作

(1)启动Hive命令行

```
[test@fsmanager ~]$ hive
```



(2)创建表

默认情况下，新建表的存储格式均为Text类型，字段间默认间隔分隔符为键盘上的Tab键。
创建一个有两个字段的pokes表，其中第一列名为foo，数据类型为INT；第二列名为bar，类型为STRING；

```
hive> create table pokes(foo int,bar string);
```

```
hive> create table pokes(foo int,bar string);
OK
Time taken: 0.596 seconds
hive> describe pokes
    > ;
OK
foo                     int
bar                     string
Time taken: 0.28 seconds, Fetched: 2 row(s)
hive>
```

创建一个有两个实体列和一个（虚拟）分区字段的invites表：

```
hive> create table invites(foo int,bar string)partitioned by(ds string);
```

```
hive> create table invites(foo int,bar string)partitioned by(ds string);
OK
Time taken: 0.135 seconds
hive> describe invites
    > ;
OK
foo                     int
bar                     string
ds                      string

# Partition Information
# col_name              data_type               comment

ds                      string
Time taken: 0.157 seconds, Fetched: 8 row(s)
hive>
```

(3)显示所有的表

```
hive> show tables;
```

```
hive> show tables
    > ;
OK
barrytang_health
chrycl_health
domestic_bs_buyer
ecm_goods
ecm_goods_merged
ecm_goods_merged_result
ecm_goods_statistics
ecm_member
ecm_member_formatted
ecm_member_formatted_result
ecm_payment
```

显示表，也支持正则查询方式，比如显示以s结尾的表。

```
hive> show  tables '.*s';
```

```
hive> show  tables '.*s';
OK
ecm_goods
ecm_goods_statistics
invites
pokes
Time taken: 0.057 seconds, Fetched: 4 row(s)
hive>
```

(4)显示表列

```
hive> describe invites;
```

```
hive> describe invites;
OK
foo                     int
bar                     string
ds                      string

# Partition Information
# col_name              data_type               comment

ds                      string
Time taken: 0.155 seconds, Fetched: 8 row(s)
hive>
```

(5)更改表

① 修改表名

```
hive> alter table person rename to people;
```

```
hive> alter table person rename to people;
OK
Time taken: 0.317 seconds
hive>
```

② 表新增一列

```
hive> alter table pokes add columns (new_col int);
```

```
hive> alter table pokes add columns (new_col int);
OK
Time taken: 0.174 seconds
hive>
```

③ 替换表所有列名（数据不动）

```
hive> alter table invites replace columns (foo1 int,bar1 string,baz1 int);
```

```
hive> alter table invites replace columns (foo1 int,bar1 string,baz1 int);
OK
Time taken: 0.184 seconds
hive>
```

(6)删除表（或列）

```
hive> alter table invites replace columns(foo1 int comment 'only keep the fisrt column');
```

```
hive> alter table invites replace columns(foo1 int comment 'only keep the fisrt column');
OK
Time taken: 0.164 seconds
hive>
```