# 第二天笔记



## Hive 查看SQL解析计划：

```
// extended 可选，可以打印更多细节
explain  select  a.id
        ,a.name
        ,a.clazz
        ,t1.sum_score
from(
    select  id
            ,sum(score) as sum_score
    from score
```

```
    group by id
)t1 right join (
    select  id
            ,name
            ,'文科一班' as clazz
    from students
    where clazz = '文科一班'
) a
on t1.id = a.id
order by t1.sum_score desc
limit 10;
```

## Hive建表

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  // 定义字段名，字段类型
  [(col_name data_type [COMMENT col_comment], ...)]
  // 给表加上注解
  [COMMENT table_comment]
  // 分区
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  // 分桶
  [CLUSTERED BY (col_name, col_name, ...)
  // 设置排序字段 升序、降序
  [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [
    // 指定设置行、列分隔符
   [ROW FORMAT row_format]
   // 指定Hive储存格式：textFile、rcFile、SequenceFile 默认为：textFile
   [STORED AS file_format]

   | STORED BY 'storage.handler.class.name' [ WITH SERDEPROPERTIES (...) ]
(Note:  only available starting with 0.6.0)
  ]
  // 指定储存位置
  [LOCATION hdfs_path]
  // 跟外部表配合使用，比如：映射HBase表，然后可以使用HQL对hbase数据进行查询，当然速度比较慢
  [TBLPROPERTIES (property_name=property_value, ...)]  (Note:  only available
starting with 0.6.0)
  [AS select_statement]  (Note: this feature is only available starting with
0.5.0.)
```

### 建表1：全部使用默认建表方式

```
create table students
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','; // 必选，指定列分隔符
```

**建表2：指定location （这种方式也比较常用）**

```
create table students2
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/input1'; // 指定Hive表的数据的存储位置，一般在数据已经上传到HDFS，想要直接使用，会指定Location，通常Locaion会跟外部表一起使用，内部表一般使用默认的location
```

**建表3：指定存储格式**

```
create table students3
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS rcfile; // 指定储存格式为rcfile，inputFormat:RCFileInputFormat,outputFormat:RCFileOutputFormat，如果不指定，默认为textfile，注意：除textfile以外，其他的存储格式的数据都不能直接加载，需要使用从表加载的方式。
```

**建表4：create table xxxx as select_statement(SQL语句) (这种方式比较常用)**

```
create table students4 as select * from students2;
```

**建表5：create table xxxx like table_name 只想建表，不需要加载数据**

```
create table students5 like students;
```

## Hive加载数据

**1、使用**`hdfs dfs -put '本地数据' 'hive表对应的HDFS目录下'`

**2、使用 load data inpath**

> 下列命令需要在hive shell里执行

```
// 将HDFS上的/input1目录下面的数据 移动至 students表对应的HDFS目录下，注意是 移动、移动、移动
load data inpath '/input1/students.txt' into table students;
```

```
// 清空表
truncate table students;
// 加上 local 关键字 可以将Linux本地目录下的文件 上传到 hive表对应HDFS 目录下 原文件不会被删除
load data local inpath '/usr/local/soft/data/students.txt' into table students;
// overwrite 覆盖加载
load data local inpath '/usr/local/soft/data/students.txt' overwrite into table students;
```

**3、create table xxx as SQL语句**

**4、insert into table xxxx SQL语句 （没有as）**

```
// 将 students表的数据插入到students2 这是复制 不是移动 students表中的表中的数据不会丢失
insert into table students2 select * from students;

// 覆盖插入 把into 换成 overwrite
insert overwrite table students2 select * from students;
```

## Hive 内部表（Managed tables）vs 外部表（External tables）

**建表：**

```
// 内部表
create table students_internal
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/input2';

// 外部表
create external table students_external
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/input3';
```

**加载数据：**

```
hive> dfs -put /usr/local/soft/data/students.txt /input2/;
hive> dfs -put /usr/local/soft/data/students.txt /input3/;
```

**删除表：**

```
hive> drop table students_internal;
Moved: 'hdfs://master:9000/input2' to trash at:
hdfs://master:9000/user/root/.Trash/Current
OK
Time taken: 0.474 seconds
hive> drop table students_external;
OK
Time taken: 0.09 seconds
hive>
```

> 可以看出，删除内部表的时候，表中的数据（HDFS上的文件）会被同表的元数据一起删除
>
> 删除外部表的时候，只会删除表的元数据，不会删除表中的数据（HDFS上的文件）
>
> 一般在公司中，使用外部表多一点，因为数据可以需要被多个程序使用，避免误删，通常外部表会结合location一起使用
>
> 外部表还可以将其他数据源中的数据 映射到 hive中，比如说：hbase，ElasticSearch.....
>
> 设计外部表的初衷就是 让 表的元数据 与 数据 解耦
>
> Managed tables are Hive owned tables where the entire lifecycle of the tables' data are managed and controlled by Hive. External tables are tables where Hive has loose coupling with the data.
>
> All the write operations to the Managed tables are performed using Hive SQL commands. If a Managed table or partition is dropped, the data and metadata associated with that table or partition are deleted. The transactional semantics (ACID) are also supported only on Managed tables.

## Hive 分区

> 分区表实际上是在表的目录下在以分区命名，建子目录
>
> 作用：进行分区裁剪，避免全表扫描，减少MapReduce处理的数据量，提高效率
>
> 一般在公司的hive中，所有的表基本上都是分区表，通常按日期分区、地域分区
>
> 分区表在使用的时候记得加上分区字段
>
> 分区也不是越多越好，一般不超过3级，根据实际业务衡量

**建立分区表：**

```
create external table students_pt1
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
PARTITIONED BY(pt string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/student/input1';
```

**增加一个分区：**

```
alter table students_pt1 add partition(pt='20220218');
```

**删除一个分区：**

```
alter table students_pt1 drop partition(pt='20220216');
```

**查看某个表的所有分区**

```
show partitions students_pt1; // 推荐这种方式（直接从元数据中获取分区信息）

select distinct pt from students_pt; // 不推荐
```

**往分区中插入数据：**

```
insert into table students_pt1 partition(pt='20220218') select * from students;

load data local inpath '/usr/local/soft/data/students.txt' into table
students_pt1 partition(pt='20220217');
```

**查询某个分区的数据：**

```
// 全表扫描，不推荐，效率低
select count(*) from students_pt1;

// 使用where条件进行分区裁剪，避免了全表扫描，效率高
select count(*) from students_pt1 where pt='20220218';

// 也可以在where条件中使用非等值判断
select count(*) from students_pt1 where pt<='20210112' and pt>='20210110';
```

## Hive动态分区

> 有的时候我们原始表中的数据里面包含了 "日期字段 dt"，我们需要根据dt中不同的日期，分为不同的分区，将原始表改造成分区表。
>
> hive默认不开启动态分区
>
> 动态分区：根据数据中某几列的不同的取值 划分 不同的分区

**开启Hive的动态分区支持**

```
# 表示开启动态分区
hive> set hive.exec.dynamic.partition=true;
# 表示动态分区模式：strict（需要配合静态分区一起使用）、nostrict
# strict： insert into table students_pt partition(dt='anhui',pt) select
......,pt from students;
hive> set hive.exec.dynamic.partition.mode=nostrict;
# 表示支持的最大的分区数量为1000，可以根据业务自己调整
hive> set hive.exec.max.dynamic.partitions.pernode=1000;
```

**建立原始表并加载数据**

```
create table students_dt
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string,
    dt string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

**建立分区表并加载数据**

```
create table students_dt_p
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
PARTITIONED BY(dt string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

**使用动态分区插入数据**

```
// 分区字段需要放在 select 的最后，如果有多个分区字段 同理，它是按位置匹配，不是按名字匹配
insert into table students_dt_p partition(dt) select id,name,age,gender,clazz,dt
from students_dt;
// 比如下面这条语句会使用age作为分区字段，而不会使用student_dt中的dt作为分区字段
insert into table students_dt_p partition(dt) select id,name,age,gender,dt,age
from students_dt;
```

**多级分区**

```
create table students_year_month
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string,
    year string,
    month string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

create table students_year_month_pt
(
    id bigint,
    name string,
    age int,
    gender string,
```

```
    clazz string
)
PARTITIONED BY(year string,month string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

insert into table students_year_month_pt partition(year,month) select
id,name,age,gender,clazz,year,month from students_year_month;
```

> 自己尝试一下多级分区
>
> 上单讲分区：https://developer.aliyun.com/article/81775

## Hive分桶

> 分桶实际上是对文件（数据）的进一步切分
>
> Hive默认关闭分桶
>
> 作用：在往分桶表中插入数据的时候，会根据 clustered by 指定的字段 进行hash分区 对指定的buckets个数 进行取余，进而可以将数据分割成buckets个数个文件，以达到数据均匀分布，可以解决Map端的"数据倾斜"问题，方便我们取抽样数据，提高Map join效率
>
> 分桶字段 需要根据业务进行设定

### 开启分桶开关

```
hive> set hive.enforce.bucketing=true;
```

### 建立分桶表

```
create table students_buks
(
    id bigint,
    name string,
    age int,
    gender string,
    clazz string
)
CLUSTERED BY (clazz) into 12 BUCKETS
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

### 往分桶表中插入数据

```
// 直接使用load data 并不能将数据打散
load data local inpath '/usr/local/soft/data/students.txt' into table
students_buks;

// 需要使用下面这种方式插入数据，才能使分桶表真正发挥作用
insert into students_buks select * from students;
```

> https://zhuanlan.zhihu.com/p/93728864 Hive分桶表的使用场景以及优缺点分析

## Hive JDBC

### 启动hiveserver2

```
hive --service hiveserver2 &

或者
hiveserver2 &
```

### 新建maven项目并添加两个依赖

```
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.7.6</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hive/hive-jdbc -->
    <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-jdbc</artifactId>
        <version>1.2.1</version>
    </dependency>
```

### 编写JDBC代码

```
import java.sql.*;

public class HiveJDBC {
    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
        Class.forName("org.apache.hive.jdbc.HiveDriver");
        Connection conn =
DriverManager.getConnection("jdbc:hive2://master:10000/test3");
        Statement stat = conn.createStatement();
        ResultSet rs = stat.executeQuery("select * from students limit 10");
        while (rs.next()) {
            int id = rs.getInt(1);
            String name = rs.getString(2);
            int age = rs.getInt(3);
            String gender = rs.getString(4);
            String clazz = rs.getString(5);
            System.out.println(id + "," + name + "," + age + "," + gender + ","
+ clazz);
        }
        rs.close();
        stat.close();
        conn.close();
    }
}
```