1 Hive数据导入方式

Hive常见的数据导入方式共有四中：1）从本地文件系统导入数据到Hive表；2）从HDFS上导入数据到Hive表；3）从别的表中查询出相应的数据并导入到Hive表中；4）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。

1.1 从本地文件系统中导入数据到Hive表

```
>hive> create table test
     > (id int, name string,
     > age int, tel string)
     > ROW FORMAT DELIMITED
     > FIELDS TERMINATED BY '\t'
     > STORED AS TEXTFILE;
OK
Time taken: 2.832 seconds
```

这个表很简单，只有四个字段。本地文件系统里面有个/home/work/test.txt文件,文件中各列使用’\t’分隔，可以通过如下语句将test.txt里的数据导入到表test里：

```
hive> load data local inpath 'test.txt' into table test;
Copying data from file:/home/test/test.txt
Copying file: file:/home/test/test.txt
Loading data to table default.test
Table default.test stats:
[num_partitions: 0, num_files: 1, num_rows: 0, total_size: 67]
OK

Time taken: 5.967 seconds
```

1.2 从HSDFS上导入数据到Hive表

从本地文件系统中将数据导入到Hive表的过程中，其实是先将数据临时复制到HDFS的一个目录下（典型的情况是复制到上传用户的HDFS home目录下,比如/home/test/），然后再将数据从那个临时目录下移动（注意，这里说的是移动，不是复制！）到对应的Hive表的数据目录里面。既然如此，那么Hive肯定支持将数据直接从HDFS上的一个目录移动到相应Hive表的数据目录下，假设有下面这个文件/home/test/add.txt，具体的操作如下：

```
hive> load data inpath '/home/test/add.txt' into table test;
Loading data to table default.test
Table default.wyp stats:
[num_partitions: 0, num_files: 2, num_rows: 0, total_size: 215]
OK
Time taken: 0.47 seconds
```

注意load data inpath ‘/home/test/add.txt’ into table test;里面是没有local这个单词的，这个是和

1中的区别。

1.3 从别的表中查询出相应的数据并导入到Hive表中

假设Hive中有test1表，其建表语句如下所示：

```
hive> create table test1(
    > id int, name string, tel string)
    > partitioned by (age int)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t'
    > STORED AS TEXTFILE;
```

将test表中的查询结果插入到test中：

```
hive> insert into table test1
    > partition (age='25')
    > select id, name, tel
    > from test;
```

如果目标表（test）中不存在分区字段，可以去掉partition (age='25')语句。当然，我们也可以在select语句里面通过使用分区值来动态指明分区：

```
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> insert into table test1
    > partition (age)
    > select id, name, tel, age
    > from test;
```

这种方法叫做动态分区插入，但是Hive中默认是关闭的，所以在使用前需要先把hive.exec.dynamic.partition.mode设置为nonstrict。当然，Hive也支持insert overwrite方式来插入数据，从字面我们就可以看出，overwrite是覆盖的意思，是的，执行完这条语句的时候，相应数据目录下的数据将会被覆盖！而insert into则不会，注意两者之间的区别。例子如下：

```
hive> insert overwrite table test2
    > partition (age)
    > select id, name, tel, age
    > from test;
```

更厉害的是，Hive还支持多表插入，什么意思呢？在Hive中，我们可以把insert语句倒过来，把from放在最前面，它的执行效果和放在后面是一样的，如下：

```
hive> show create table test3;
OK
CREATE  TABLE test3(
    id int,
    name string)
Time taken: 0.277 seconds, Fetched: 18 row(s)

hive> from test
    > insert into table test2
    > partition(age)
    > select id, name, tel, age
    > insert into table test3
    > select id, name
    > where age>'25';
```

可以在同一个查询中使用多个insert子句，这样的好处是我们只需要扫描一遍源表就可以生成多个不相交的输出。

1.4 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）如下：

```
hive> create table test4
    > as
    > select id, name, tel
    > from test;
```

数据就插入到test4表中去了，CTAS操作是原子的，因此如果select查询由于某种原因而失败，新表是不会创建的！

2 Hive数据导出方式

根据导出的地方不一样，Hive共有三中数据导出方式：（1）、导出到本地文件系统；（2）、导出到HDFS中；（3）、导出到Hive的另一个表中。

2.1 导出到本地文件系统

```
hive> insert overwrite local directory '/home/work/test'
    > row format delimited #定义各个域之间的分隔符
    > fields terminated by '\t'
    > select * from test;
```

这条HQL的执行需要启用Mapreduce完成，运行完这条语句之后，将会在本地文件系统的/home/work/test/目录下生成文件，这个文件是Reduce产生的结果（这里生成的文件名是000000_0）。

2.2 导出到HDFS中

和导入数据到本地文件系统一样的简单，去掉**directory**之前的**local**就可以了。

```
hive> insert overwrite directory '/home/work/test.txt'
    > select * from test;
```

将会在HDFS的/home/work/hdfs目录下保存导出来的数据。注意，和导出文件到本地文件系统的HQL少一个local，数据的存放路径就不一样了。

2.3 导出到Hive的另一个表中

```
hive> insert into table new_test
       > partition (age='25')
       > select id, name, tel
       > from test
```

2.4 其它

其实，我们还可以用hive的-e和-f参数来导出数据。其中-e 表示后面直接接带双引号的sql语句；而-f是接一个文件，文件的内容为一个sql语句。

```
$ hive -e "select * from test" > test.txt
$ cat test.sql
select * from test
$ hive -f test.sql > test.txt
```

3 Hive中order by,sort by, distrbute by 和cluster by的区别

3.1 order by

Hive中的order by和数据库中的order by 功能一致，按照某一项或者几项排序输出，可以指定是升序或者是降序排序。它保证全局有序，但是进行order by的时候是将所有的数据全部发送到一个Reduce中，所以在大数据量的情况下可能不能接受，最后这个操作将会产生一个文件。

3.2 sort by

sort by只能保证在同一个reduce中的数据可以按指定字段排序。使用sort by 你可以指定执行的reduce个数 （set mapreduce.job.reduce=） 这样可以输出更多的数据。对输出的数据再执行归并排序，即可以得到全部结果。需要注意的是，N个Reduce处理的数据范围是可以重叠的，所以最后排序完的N个文件之间数据范围是有重叠的。

3.3 distribute by

按照指定的字段将数据划分到不同的输出reduce中，这使用在本博客的《Hive：解决Hive创建文

件数过多的问题》有详细的介绍。这个可以保证每个Reduce处理的数据范围不重叠，每个分区内的数据是没有排序的。

3.4 cluster by

cluster by 除了具有 distribute by 的功能外还兼具 sort by 的功能。 所以最终的结果是每个Reduce处理的数据范围不重叠，而且每个Reduce内的数据是排序的，而且可以打到全局有序的结果。