**16. Hive查询**

**实验目的**

（1）了解Hive的SQL基本语法

（2）掌握Hive多种查询方式

**实验原理**

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HQL语句通过解释器转换为MapReduce作业提交到Hadoop集群上，Hadoop监控作业执行过程，然后返回作业执行结果给用户。

如下图Hive执行流程大致步骤为：

（1）用户提交查询等任务给Driver。

（2）编译器获得该用户的任务Plan。

（3）编译器Compiler根据用户任务去MetaStore中获取需要的Hive的元数据信息。

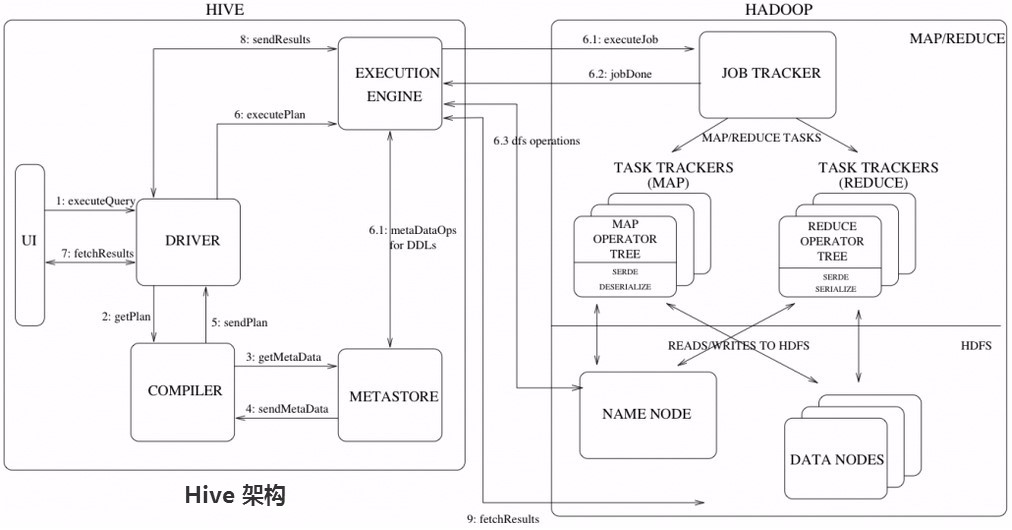
（4）编译器Compiler得到元数据信息，对任务进行编译，先将HiveQL转换为抽象语法树，然后将抽象语法树转换成查询块，将查询块转化为逻辑的查询计划，重写逻辑查询计划，将逻辑计划转化为物理的计划（MapReduce），最后选择最佳的策略。

（5）将最终的计划提交给Driver。

（6）Driver将计划Plan转交给ExecutionEngine去执行，获取元数据信息，提交给JobTracker或者SourceManager执行该任务，任务会直接读取HDFS中文件进行相应的操作。

（7）获取执行的结果。

（8）取得并返回执行结果。

[](http://192.168.1.241/doc/exper/28/img/01.jpg)

Hive的入口是Driver，执行的SQL语句首先提交到Driver驱动，然后调用Compiler解释驱动，最终解释成MapReduce任务执行，最后将结果返回。

（编译流程）

一条SQL进入Hive经过上述过程，使得一个编译过程变成了一个作业。

（1）首先，Driver会输入一个字符串SQL，然后经过Parser变成AST，这个变成AST的过程是通过Antlr来完成的，也就是Anltr根据语法文件来将SQL变成AST。

（2）AST进入Semantic Analyzer（核心）变成QB（QueryBlock）。一个最简的查询块，通常来讲，一个From子句会生成一个QB，生成QB是一个递归过程，生成的QB经过GenLogical Plan过程，变成了一个有向无环图。

（3）OP DAG经过逻辑优化器，对这个图上的边或者结点进行调整，顺序修订，变成了一个优化后的有向无环图。这些优化过程包括谓词下推（Predicate Push Down），分区剪裁（Partition Prunner），关联排序（Join Reorder）等等。

（4）经过了逻辑优化，这个有向无环图还要能够执行。所以有了生成物理执行计划的过程。Gen Tasks。Ｈive的作法通常是碰到需要分发的地方，切上一刀，生成一道MapReduce作业。如Group By切一刀，Join切一刀，Distribute By切一刀，Distinct切一刀。这么很多刀砍下去之后，这个逻辑有向无环图就被切成了很多个子图，每个子图构成一个结点。这些结点又连成了一个执行计划图，也就是Task Tree。

（5）对于Task Tree的优化，比如基于输入选择执行路径，增加备份作业等。可以由Physical Optimizer来完成。经过Physical Optimizer，每一个结点就是一个MapReduce作业或者本地作业，就可以执行了。

这就是一个SQL如何变成MapReduce作业的过程。

**实验内容**

（1）掌握Hive的普通查询、别名查询、限定查询与多表联合查询。

（2）掌握Hive的多表插入、多目录输出以及使用Shell脚本查看Hive中的表。

**实验步骤**

（1）启动Hadoop、MySQL、Hive。首先检查Hadoop相关进程，是否已经启动。若未启动，则启动Hadoop。然后执行启动以下命令，开启Mysql库，用于存放Hive的元数据。启动Mysql后，在终端命令行界面，直接输入Hive命令，启动Hive命令行。

（2）打开一个新的命令行终端，切换到~/hive-data目录，如不存在需提前创建hive-data文件夹。

从~/resources/hive-data中复制buyer\_log和buyer\_favorite到hive-data目录

[hfut@master ~]$ cp ~/resources/hive-data/buyer\_log ~/hive-data/

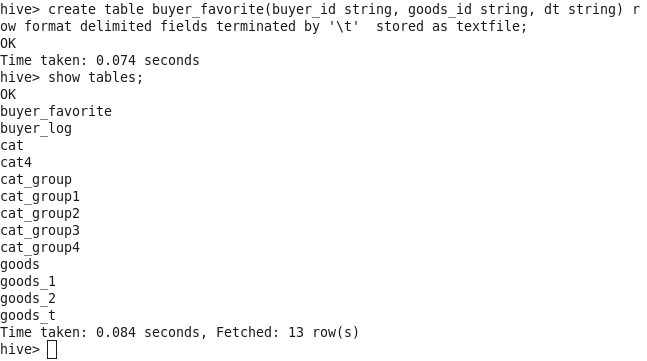
[hfut@master ~]$ cp ~/resources/hive-data/buyer\_favorite ~/hive-data/

（3）在hive命令行，创建买家行为日志表，名为buyer\_log，包含ID（id）、用户ID（buyer\_id）、时间（dt）、 地点（ip）、操作类型（opt\_type）5个字段，字符类型为string，以'\t'为分隔符。

hive> create table buyer\_log(id string, buyer\_id string, dt string, ip string, opt\_type string) row format delimited fields terminated by '\t' stored as textfile;

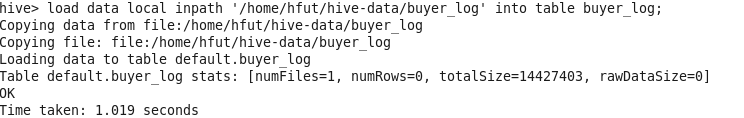
创建买家收藏表，名为buyer\_favorite，用户ID（buyer\_id）、商品ID（goods\_id）、时间（dt）3个字段，字符类型为string，以'\t'为分隔符。

hive> create table buyer\_favorite(buyer\_id string, goods\_id string, dt string) row format delimited fields terminated by '\t' stored as textfile;

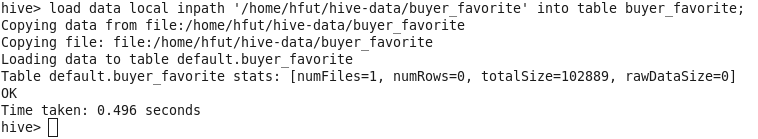


（4）将本地~/hive-data/下的表buyer\_log中数据导入到Hive中的buyer\_log表中，表buyer\_favorite中数据导入到Hive中的buyer\_favorite表中。

hive> load data local inpath '/home/hfut/hive-data/buyer\_log' into table buyer\_log;

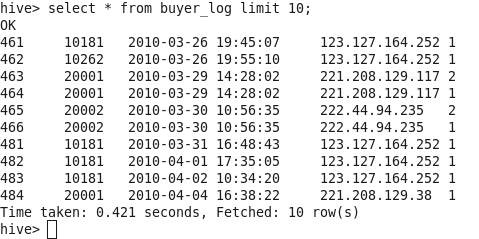


hive> load data local inpath '/home/hfut/hive-data/buyer\_favorite' into table buyer\_favorite;



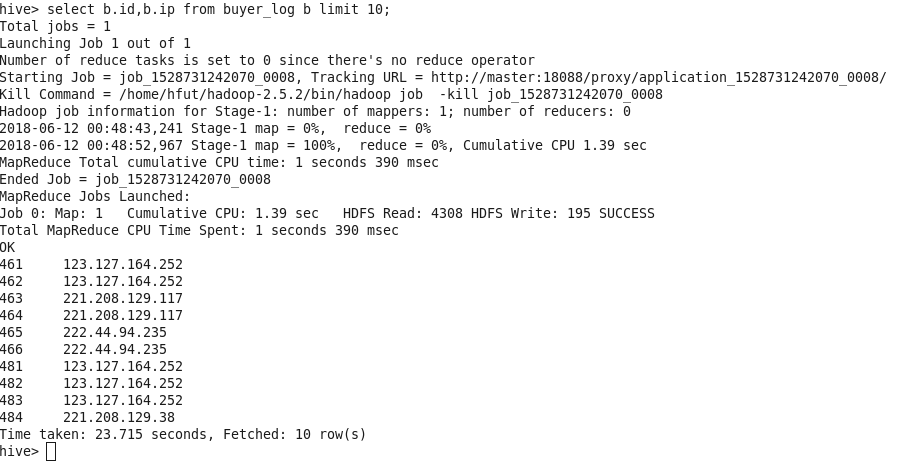
（5）普通查询，例如查询buyer\_log表中全部字段，数据量大时应避免查询全部数据。（limit 10为限制查询10条数据）

hive> select \* from buyer\_log limit 10;



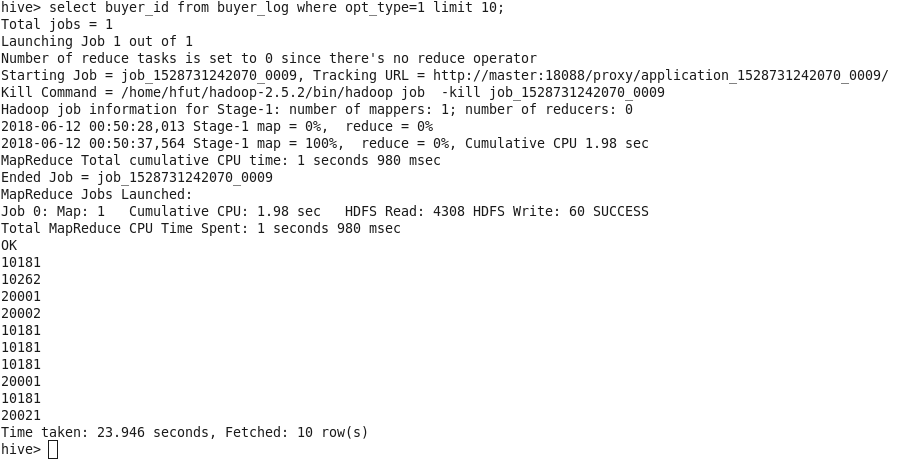
（6）别名查询，例如查询表buyer\_log中id和ip字段，当多表连接字段较多时，常常使用别名。（limit 10为限制查询10条数据）

hive> select b.id, b.ip from buyer\_log b limit 10;



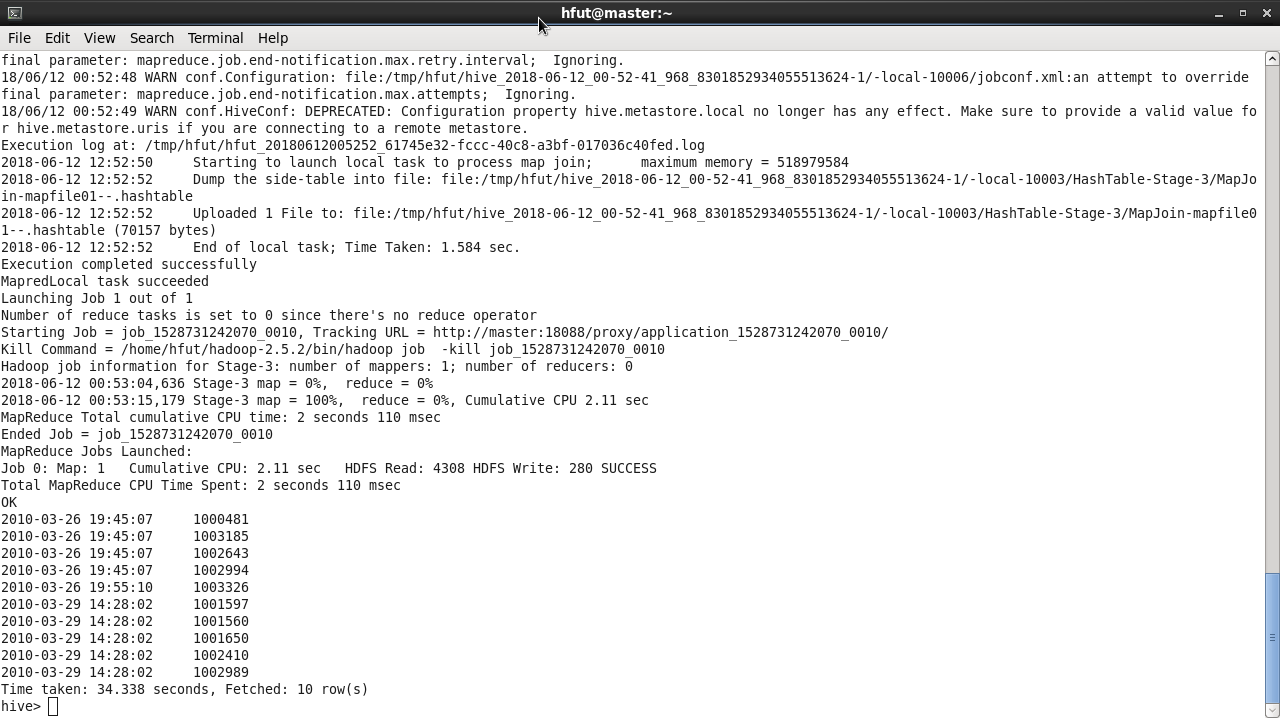
（7）限定查询（where），例如查询buyer\_log表中opt\_type=1的用户ID(buyer\_id)。（limit 10为限制查询10条数据）

hive> select buyer\_id from buyer\_log where opt\_type=1 limit 10;



（8）两表或多表联合查询，例如通过用户ID(buyer\_id)连接表buyer\_log和表buyer\_favorite，查询表buyer\_log的dt字段和表buyer\_favorite的goods\_id字段，多表联合查询可以按需求查询多个表中不同字段，生产中常用。（limit 10为限制查询10条数据）

hive> select l.dt, f.goods\_id from buyer\_log l, buyer\_favorite f where l.buyer\_id = f.buyer\_id limit 10;



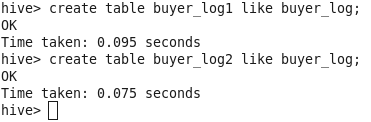
（9）多表插入，多表插入指的是在同一条语句中，把读取的同一份数据插入到不同的表中。只需要扫描一遍数据即可完成所有表的插入操作， 效率很高。

例：我们使用买家行为日志buyer\_log表作为插入表，创建buyer\_log1和buyer\_log2两表作为被插入表。

创建buyer\_log1和buyer\_log2。

hive> create table buyer\_log1 like buyer\_log;

hive> create table buyer\_log2 like buyer\_log;

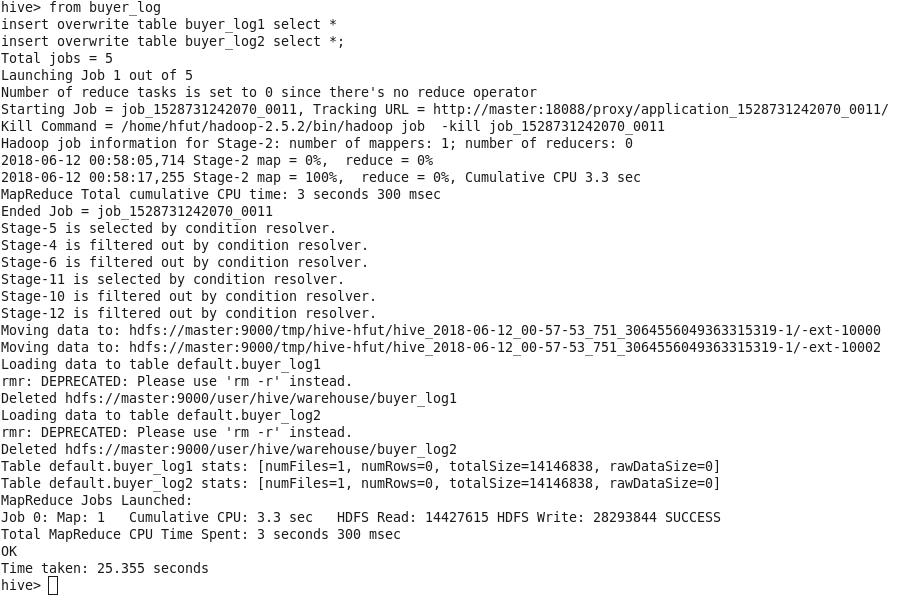


将buyer\_log表中数据插入到buyer\_log1和buyer\_log2。

hive> from buyer\_log

insert overwrite table buyer\_log1 select \*

insert overwrite table buyer\_log2 select \*;

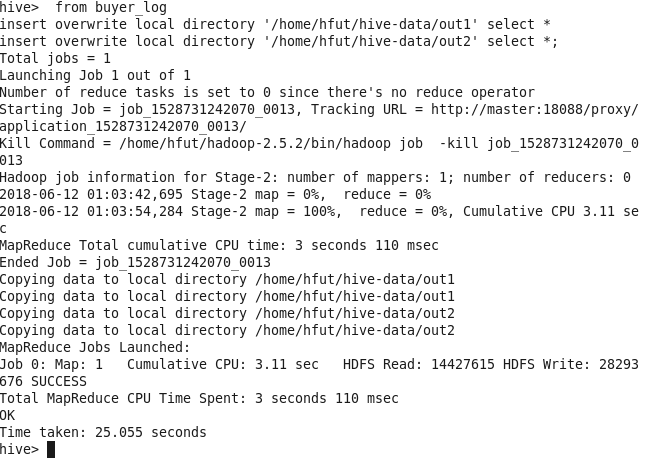


（10）多目录输出文件，将同一文件输出到本地不同文件夹中，提高效率，可以避免重复操作from，将买家行为日志buyer\_log表导出到本地‘/home/hfut/hive-data/out1’和‘/home/hfut/hive-data/out2’中

hive> from buyer\_log

insert overwrite local directory '/home/hfut/hive-data/out1' select \*

insert overwrite local directory '/home/hfut/hive-data/out2' select \*;

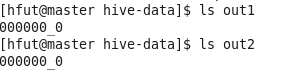


在本地切换到/home/hfut/hive-data/中，查询输出文件。

[hfut@master ~]$ cd /home/hfut/hive-data

[hfut@master hive-data]$ ls out1

[hfut@master hive-data]$ ls out2



(11) 使用shell脚本调用Hive查询语句。

切换目录到本地目录/home/hfut/hive-data下，使用gedit编写一个shell脚本，名为sh1，使其功能实现查询Hive中所有表。

[hfut@master ~] cd /home/hfut/hive-data

[hfut@master hive-data] gedit sh1

在sh1中，输入以下脚本，并保存退出

hive -e 'show tables;'

编写完成，赋予其执行权限。

[hfut@master hive-data] chmod +x sh1

执行shell脚本 。

[hfut@master hive-data] ./sh1



采用shell脚本来执行一些Hive查询语句可以简化很多的开发工作，可以利用Linux自身的一些工具，实现定时的job任务。