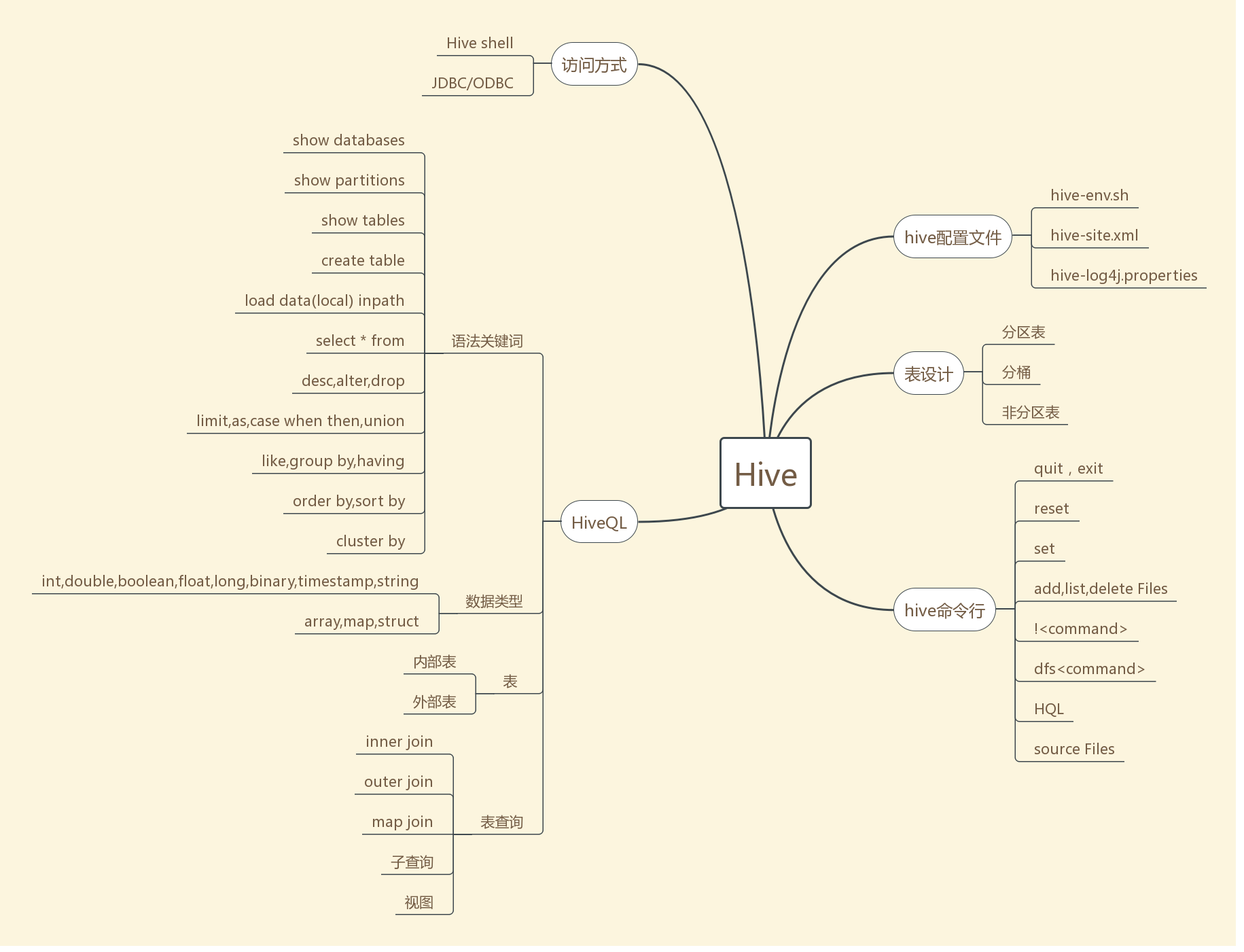
**第12章 Hive**

**12.1 概述**

**1.1 Hive简介**

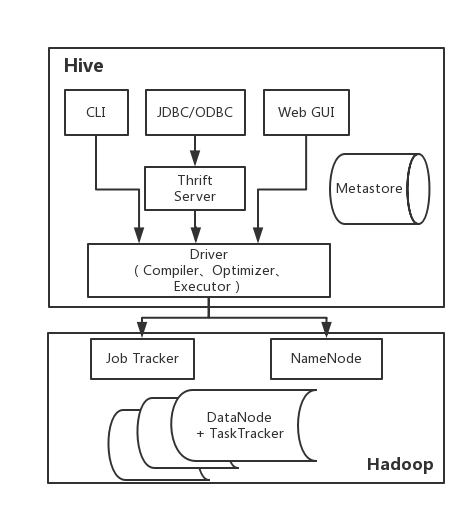
  Hive是Hadoop生态系统中必不可少的一个工具，最初由Fackbook提供，它提供了一种SQL（结构化查询语言）方言，可以查询存储在Hadoop分布式文件系统（HDFS）中的数据或其他Hadoop集成的文件系统，如MapR-FS、Amazon的S3以及像HBase和Cassandra这样的数据库中的数据。Hive设计的目的是让SQL技能良好，但Java技能较弱的分析师可以查询海量数据，2008年Fackbook把Hive项目贡献给Apache。Hive提供了比较完整的SQL功能，自身最大的缺点就是执行速度慢。

**1.2 Hive知识体系**



**12.2 系统架构**

**2.1 Hive架构图**



**2.1.1 用户接口**

  Hive自带以下几种用户接口：

* CLI就是shell命令行；
* JDBC/ODBC是Hive的Java接口，与使用传统数据库JDBC的方式类似，通过Thrift访问MetaStore生成执行计划树；
* Web GUI HWI简单的网页界面，通过浏览器访问

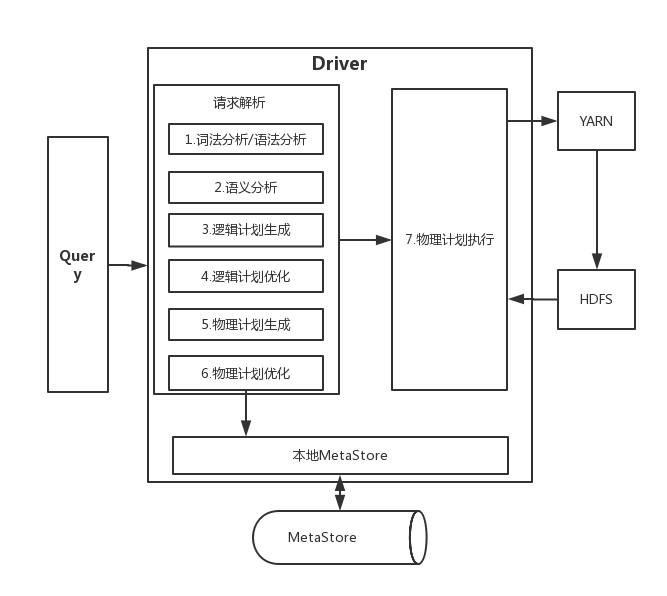
**2.1.2 命令执行**

  所有的命令和查询都会进入到Driver，通过这个模块进行解析编译，对请求的计算进行优化，然后按照执行的步骤执行（通常是启动多个MapReduce任务-JOB来执行）。当需要启动MapReduce任务时，Hive本身不会生成Java MapReduce算法程序，相反，Hive通过一个表示‘JOB执行计划’的XML文件驱动执行内置的、原生的Mapper和Reducer模块。换句话说，这些通用的模块函数类似于微型的语言翻译程序，而驱动计算的‘语言’是以XML形式编码的。Hive通过和JobTracker通信来初始化MapReduce任务，而不必部署在JobTracker所在的管理节点上执行。要处理的数据文件是存储在HDFS中的，而HDFS是由NameNode进行管理的。

**2.1.3 元数据**

  MetaStore默认使用Derby（单用户下使用），通常使用MySQL。Hive会在其中保存表模式和其他系统元数据。其中的元数据包括表信息，列信息，分区信息，以及表的数据所在的目录等。

**2.2 Hive工作原理**



  接收到一个SQL，接下来的处理流程如下：

* 1.词法分析/语法分析

  使用ANTLR将SQL语句解析成抽象语法树-AST；

* 2.语法分析

  从MetaStore获取模式信息，验证SQL语句中库名、表名、列名，以及数据类型的检查和隐式转换，以及Hive提供的函数和用户自定义的函数（UDF/UAF）；

* 3.逻辑计划生成

  生成逻辑计划-算子树；

* 4.逻辑计划优化

  对算子树进行优化，包括列剪枝，分区剪枝，谓词下推等；

* 5.物理计划生成

  将逻辑计划生成包含由MapReduce任务组成的DAG的物理计划；

* 6.物理计划优化

  对生成的MapReduce任务组成的DAG计划树进行优化，组合；

* 6.物理计划执行

  将DAG发送到Hadoop集群进行执行；

* 7.将查询结果返回

**12.3 安装部署**

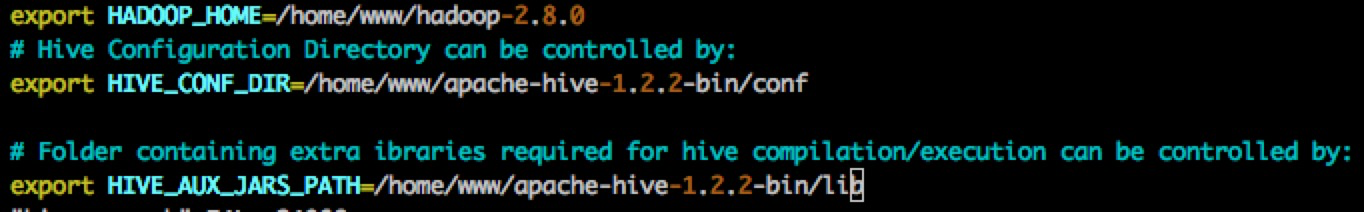
  Hive安装的前提是Java环境，Hadoop环境，MySQL安装完成。

**3.1 安装Hive的版本及配置信息**

* [Hive的版本 hive-1.2.2](http://mirrors.hust.edu.cn/apache/hive/hive-1.2.2/apache-hive-1.2.2-bin.tar.gz)
* hive的配置文件在$HIVE\_HOME/conf下，里面有5个默认的配置文件模板，特别注意还有两个位置的配置文件（如果没有指定$HIVE\_CONF\_DIR的话）/etc/hive/conf & /etc/hive/conf.dist
  + beenline-log4j.properties.template
  + hive-default.xml.template-> hive-site.xml
  + hive-env.sh.template->  [hive-env.sh](http://hive-env.sh)
  + hive-exec-log4j.properties.template
  + hive-log4j.properties.template-> hive-log4j.properties

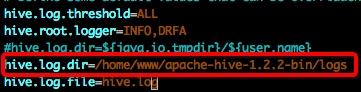
**3.2 修改配置信息**

**1.修改hive-env.sh**



* 配置HADOOP\_HOME和HIVE\_CONF\_DIR

**2.修改hive-log4j.properties**

* 首先mkdir logs创建存放日志文件的目录  
  kdir logs
* 配置  
  

**3.修改hive-site.xml**

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value> jdbc:mysql:*//localhost:3306/hivedb?createDatabaseIfNotExist=true</value>*

<describe>默认的配置元数据是存放在Derby数据库里面的，我们使用MySQL来存储元数据</describe>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>\*\*\*\*\*(根据自己的用户名填)</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>\*\*\*\*\*(根据自己的密码填)</value>

</property>

其他的一些配置项：

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/opt/hive-1.2.1/warehouse</value>

</property>

<property>

<name>hive.exec.scratchdir</name>

<value>/opt/hive-1.2.1/scratchdir</value>

</property>

<property>

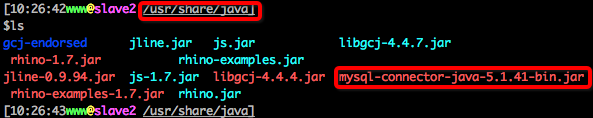
<name>hive.querylog.location</name>

<value>/opt/hive-1.2.1/logs</value>

</property>

* hive.metastore.warehouse.dir：指定hive的数据存储目录，指定的是HDFS上的位置，默认值:/usr/hive/warehouse，为了方便管理也可以在linux上指定一个目录。
* hive.exec.scratchdir：指定hive的临时数据目录，默认位置为：tmp/hive-${[user.name](http://user.name)}。为了方便管理也可以在linux上指定一个目录。

**4.上传JDBC jar包**

  默认使用Derby数据库存放元数据，此时换成了MySQL作为数据库，需要上传MySQL的JDBC驱动包到$HIVE\_HOME/lib/。  
ysql connector jar  


  至此，hive安装部署完成，启动hive shell就可以开启hive的探索之旅了。

**12.4 操作指南**

**4.1 基本操作**

**4.1.1 介绍**

* 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 EXISTS 选型来忽略这个异常。
* 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），Hive创建内部表时，会将数据移动到数据仓库指定的路径；若创建外部表，近记录数据所在的路径，不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除，而外部表只删除元数据，不删除数据。
* LIKE 允许用户复制现有的表结构，但是不复制数据。
* 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确定表的具体的列数据。
* 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SOTORED AS SEQUENCE。
* 有分区的表可以在创建的时候使用PARTITIONED BY语句。一个表可以拥有一个或者多个分区，每一个分区单独存放在一个目录下。而且表和分区都可以对某个列进行CLUSTER BY操作，将若干个列放入到一个桶（BUCKET）中。也可以利用SORT BY对数据进行排序，这样可以为特定应用提高性能。
* 表名和列名不区分大小写，SerDe和属性名区分大小写。表和列的注释是字符串。  
  注：SerDe是Serialize/Deserialize的简称，目的是用于序列化和反序列化。

**4.1.2 语法**

**1、建表**

**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]

| **STORED** **BY** 'storage.handler.class.name' [ **WITH** SERDEPROPERTIES (...) ]

]

[LOCATION hdfs\_path]

[TBLPROPERTIES (property\_name=property\_value, ...)] [**AS** select\_statement] **CREATE** [**EXTERNAL**] **TABLE** [**IF** **NOT** **EXISTS**] table\_name

**LIKE** existing\_table\_name

[LOCATION hdfs\_path]

data\_type

: primitive\_type

| array\_type

| map\_type

| struct\_type

primitive\_type

: TINYINT

| SMALLINT

| INT

| BIGINT

| BOOLEAN

| FLOAT

| **DOUBLE**

| **STRING**

array\_type

: ARRAY < data\_type >

map\_type

: **MAP** < primitive\_type, data\_type >

struct\_type

: **STRUCT** < col\_name : data\_type [**COMMENT** col\_comment], ...>

row\_format

: **DELIMITED** [**FIELDS** **TERMINATED** **BY** char] [COLLECTION ITEMS **TERMINATED** **BY** char]

[**MAP** **KEYS** **TERMINATED** **BY** char] [**LINES** **TERMINATED** **BY** char]

| SERDE serde\_name [**WITH** SERDEPROPERTIES (property\_name=property\_value, property\_name=property\_value, ...)]

file\_format:

: SEQUENCEFILE

| TEXTFILE

| RCFILE (Note: **only** available **starting** **with** 0.6.0)

| INPUTFORMAT input\_format\_classname OUTPUTFORMAT output\_format\_classname

**2、添加分区**

**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, ...)

**3、删除分区**

**ALTER** **TABLE** table\_name *DROP partition\_spec, partition\_spec,...*

**5、重命名表**

**ALTER** **TABLE** table\_name **RENAME** **TO** new\_table\_name

**6、修改列/属性**

**ALTER** **TABLE** table\_name **CHANGE** [**COLUMN**]

col\_old\_name col\_new\_name column\_type [**COMMENT** col\_comment] [**FIRST**|**AFTER** column\_name]

**7、添加/替换列**

**ALTER** **TABLE** table\_name **ADD**|**REPLACE** **COLUMNS** (col\_name data\_type [**COMMENT** col\_comment], ...)

注：**add**是表示新增一字段，位置在所有列后面（**partition**列前）；**replace**则是表示替换表中所有字段。

**8、创建视图**

**CREATE** **VIEW** [**IF** **NOT** **EXISTS**] view\_name [ (column\_name [**COMMENT** column\_comment], ...) ]

[**COMMENT** view\_comment]

[TBLPROPERTIES (property\_name = property\_value, ...)]

**AS** **SELECT** ...

注：视图是只读的，不能用**LOAD**/**INSERT**/**ALTER**。

**9、加载数据**

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

**10、INSERT语法**

Standard syntax:

**INSERT** OVERWRITE **TABLE** tablename1 [**PARTITION** (partcol1=val1, partcol2=val2 ...)] select\_statement1 **FROM** from\_statement

Hive extension (multiple inserts):

**FROM** from\_statement

**INSERT** OVERWRITE **TABLE** tablename1 [**PARTITION** (partcol1=val1, partcol2=val2 ...)] select\_statement1

[**INSERT** OVERWRITE **TABLE** tablename2 [**PARTITION** ...] select\_statement2] ...

Hive extension (dynamic **partition** inserts):

**INSERT** OVERWRITE **TABLE** tablename **PARTITION** (partcol1[=val1], partcol2[=val2] ...) select\_statement **FROM** from\_statement

注：

\* 1 **insert**时，**from**子句既可以放在**select**子句后，也可以放在**insert**子句前。

\* 2 Hive不支持一条一条的用**insert**语句进行插入，也不支持**update**操作。数据是以**load**的方式，加载到建好的表中。数据一旦导入，则不可修改。要么**drop**掉整个表，要么建立新的表，导入新的数据。

**11 WRITE语法**

Standard syntax:

**INSERT** OVERWRITE [LOCAL] **DIRECTORY directory1** SELECT ... FROM ...

Hive **extension** (**multiple inserts):**

FROM from\_statement

**INSERT** OVERWRITE [LOCAL] **DIRECTORY directory1** select\_statement1

[**INSERT** OVERWRITE [LOCAL] **DIRECTORY directory2** select\_statement2] ...

**12 查询**

**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** col\_list]

]

[**LIMIT** **number**]

**13 Hive Join**

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:

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

注：Hive只支持等值连接（equality joins）、外连接（outer joins）和（left/right joins）。Hive不支持非等值的连接，因为非等值的连接很难转化为MapReduce任务。

另外，Hive支持多于两个表的连接。

**4.1.3 基本示例**

* 1、如果一个表已经存在，可以使用IF NOT EXISTS；
* 2、建表示例

**create** **table** **user**

(

**id** int,

cont **string**

)

**row** **format** **delimited**

fileds **terminated** **by** '\005'

注：**terminated** **by** 关于来源的文本数据的字段间隔符。

* 3、如果要将自定义间隔符的文件读入一个表，需要通过创建表的语句来指明输入文件间隔符，然后load data 到这张表。
* 4、load data [local] inpath '/user/admin/a.txt' overwrite into table user;
* 5、使用EXTERNAL创建外表和普通创建管理表的区别是：前者存放元数据，删除后文件系统中的数据不会删除，后者会直接删除文件系统中的数据。
* 6.创建分区分桶表

**create** **table** **if** **not** **exists** page\_view

(

view\_time int,

user\_id bigint,

page\_url **string**,

referrer\_url **string**,

ip **string** **comment** 'ip address'

)**comment** 'page view table'

partitioned **by**(pt **string**,country **string**)

clustered **by**(user\_id) sorted **by**(view\_time) **into** 32 buckets

**row** **format** **delimited**

**fields** **terminated** **by** '\001'

collection items **terminated** **by** '\002'

**map** **keys** **terminated** **by** '\003'

**stored** **as** sequencefile;

* 7、复制一个空表

**create** **table** empty\_key\_value\_store **like** key\_value\_store;

**4.2 函数以及UDF&UDAF&UDTF**

**4.2.1 内置运算符**

**4.2.1.1 关系运算符**

| **运算符** | **类型** | **说明** |
| --- | --- | --- |

**4.3 Hive正则**