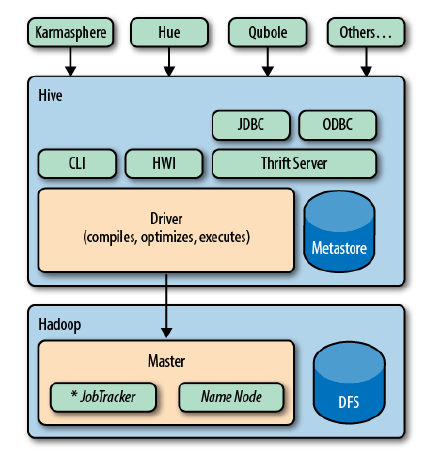
# Hive架构图



# Hive常用命令

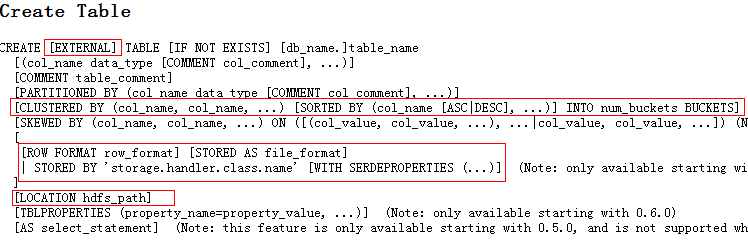
## 数据库操作命令

|  |  |  |
| --- | --- | --- |
| 功能 | 命令 | 备注 |
| 查看当前所有库 | show databases |  |
| 创建库 | create database test |  |
| 删除库 | drop database test |  |
| 使用库 | use test |  |
| 显示库中的表 | Show tables  Show tables like ‘%name%’; |  |
| 查看表结构 | Desc table\_name;  Desc formatted table\_name; |  |
| 查看分区 | show partitions table\_name; |  |
|  |  |  |
|  |  |  |

## HQL-DDL,DML

### 建表

#### 语法



hive建表语法格式

* external 外部表，类似于mysql的csv引擎
* partitioned by 指定分区字段
* clustered by sorted by 可以对表和分区对某个列进行分桶操作，也可以利用sorted by对某个字段进行排序
* row format指定数据行中字段间的分隔符和数据行分隔符
* stored as 指定数据文件格式：textfile sequence rcfile inputformat (自定义的inputformat 类)
* location 指定数据文件存放的hdfs目录

#### 例子

|  |
| --- |
| CREATE EXTERNAL TABLE dbtest.sogou(  ts STRING,  uid STRING,  keyword STRING,  rank INT,  order INT,  url STRING)  COMMENT 'This is the sogou search data of one day'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t' //属性之间以\t分隔  STORED AS TEXTFILE //以文本文件存储  LOCATION '/sogou'; //hadoop上的存储路径 |

### 删表

* 删除内部表时会删除元数据和表数据文件
* 删除外部表（external）时只删除元数据

|  |
| --- |
| drop table table\_name |

### 修改表名

|  |
| --- |
| alter table table\_name  rename to new\_table\_name |

### 增加分区

|  |
| --- |
| alter table page\_view  add partition (dt='2008-08-08', country='us') location '/path/to/us/part080808'  partition (dt='2008-08-09', country='us') location '/path/to/us/part080809'; |

### 删除分区

|  |
| --- |
| alter table page\_view drop partition (dt='2008-08-08', country='us'); |

### 修改字段

|  |
| --- |
| alter table test\_table change filed\_a filed\_b string after filed\_c; |

### 添加字段

|  |
| --- |
| alter table tmp\_h02\_click\_log\_baitiao\_ag\_sum add columns(current\_session\_timelenth\_count bigint comment '页面停留总时长'); |

### 从文件加载数据

|  |
| --- |
| LOAD DATA LOCAL INPATH 'dim\_csl\_rule\_config.txt' OVERWRITE into table test\_db.test\_table; |

注：OVERWRITE表示覆盖，追加不需要此关键字

### hive复制表结构

|  |
| --- |
| CREATE TABLE d\_h02\_click\_log\_baitiao\_ag\_sum LIKE tmp.tmp\_h02\_click\_log\_baitiao\_ag\_sum; |

### 从其它表加载数据

#### 基本的插入

 INSERT OVERWRITE TABLE tablename [PARTITON(partcol1=val1,partclo2=val2)]select\_statement FROM from\_statement

|  |
| --- |
| INSERT OVERWRITE TABLE test\_h02\_click\_log PARTITION(dt) select \*  from stage.s\_h02\_click\_log where dt='2014-01-22' limit 100; |

#### 对多个表进行插入操作

FROM src\_table

INSERT OVERWRITE TABLE tablename1 [PARTITON(partcol1=val1,partclo2=val2)]select\_statement1

INSERT OVERWRITE TABLE tablename2 [PARTITON(partcol1=val1,partclo2=val2)]select\_statement2

|  |
| --- |
| from test\_table  insert overwrite table test\_insert1  select key  insert overwrite table test\_insert2  select value; |

注：OVERWRITE表示覆盖，追加不需要此关键字

### 导出数据到文件

#### 导出到hdfs中

|  |
| --- |
| insert overwrite directory '/tmp/csl\_rule\_cfg' select a.\* from dim.dim\_csl\_rule\_config a; |

#### 导出到本地

|  |
| --- |
| insert overwrite local directory '/tmp/csl\_rule\_cfg' select a.\* from dim.dim\_csl\_rule\_config a; |

#### 导出白条数据

|  |
| --- |
| hive -e "select day\_id,pv,uv,ip\_count,click\_next\_count,second\_bounce\_rate,return\_visit,pg\_type from tmp.tmp\_h02\_click\_log\_baitiao\_ag\_sum where day\_id like '2014-03%';"> /home/jrjt/testan/baitiao.xlsx; |

### 杀死某个任务  不在hive shell中执行

   hadoop job -kill job\_201403041453\_58315

## HQL-SQL

|  |
| --- |
| 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] |

### order by（与传统数据库一致）

order by 会对输入做全局排序，因此只有一个reducer（多个reducer无法保证全局有序）会导致当输入规模较大时，需要较长的计算时间。

order by 和数据库中的Order by 功能一致，按照某一项或几项排序输出。

与数据库中 order by 的区别在于在hive.mapred.mode = strict 模式下 必须指定 limit 否则执行会报错。 原因： 在order by 状态下所有数据会到一台服务器进行reduce操作也即只有一个reduce，如果在数据量大的情况下会出现无法输出结果的情况，如果进行 limit n ，那只有 n \* map number 条记录而已。只有一个reduce也可以处理过来。

### sort by（reduce内的排序）

每个reduce内是如何排序的。当有多个reduce时，不能保证最终结果是排序的。

### distribute by（map的结果按照什么规则分到reduce中）

map的结果根据distribute by指定的属性，将数据分到同一个reducer中。

### cluster By

除了具有Distribute by的功能外，还会对该字段进行排序。因此，常常认为cluster by = distribute by + sort by。

### 根据分区查询

select table\_coulm from table\_name where partition\_name = '2014-02-25';

### mapjoin的使用

应用场景：

1.关联操作中有一张表非常小

2.不等值的链接操作

|  |
| --- |
| select /\*+ mapjoin(A)\*/ f.a,f.b from A t join B f on ( f.a=t.a and f.ftime=20110802) |

## 启动thrift server

hive --service hiveserver

# 列类型

## 原生类型

TINYINT

SMALLINT

INT

BIGINT

BOOLEAN

FLOAT

DOUBLE

STRING

BINARY (Hive 0.8.0以上才可用)

TIMESTAMP (Hive 0.8.0以上才可用)

## 复合类型

arrays: ARRAY<data\_type>

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

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

union: UNIONTYPE<data\_type, data\_type, ...>

## 时间戳

|  |  |
| --- | --- |
| 支持传统的unix时间戳,可选的纳秒级精度 |  |
| 支持的转换 | 整型数值类型：解读为以秒为单位的UNIX时间戳 |
| 浮动点数值类型：解读为以秒和小数精度为单位的UNIX时间戳 |
| 字符串：JDBC兼容的java.sql.Timestamp格式“YYYY-MM-DD HH：MM：SS.fffffffff”（9位小数位精度） |

# 优化

## 排序优化

### Order by

Order by 实现全局排序，一个reduce实现，效率低

### Sort byt

Sort by 实现部分有序，单个reduce输出的结果是有序的，效率高，通常和DISTRIBUTE BY关键字一起使用（DISTRIBUTE BY关键字 可以指定map 到 reduce端的分发key）

### CLUSTER BY

CLUSTER BY col1 等价于DISTRIBUTE BY col1 SORT BY col1.

## 合并小文件

文件数目过多，会给 HDFS 带来压力，并且会影响处理效率，可以通过合并 Map 和 Reduce 的结果文件来消除这样的影响

**hive.merge.mapfiles = true**是否和并 Map 输出文件，默认为 True

**hive.merge.mapredfiles = false**是否合并 Reduce 输出文件，默认为 False

**hive.merge.size.per.task = 256\*1000\*1000**合并文件的大小。

## join优化

* Join查找操作的基本原则：应该将条目少的表/子查询放在 Join 操作符的左边。原因是在 Join 操作的 Reduce 阶段，位于 Join 操作符左边的表的内容会被加载进内存，将条目少的表放在左边，可以有效减少发生内存溢出错误的几率
* Join查找操作中如果存在多个join，且所有参与join的表中其参与join的key都相同，则会将所有的join合并到一个mapred程序中。

## 嵌套SQL并行执行优化

set hive.exec.parallel=true;

set hive.exec.parallel.thread.number=16;

效率可提升至少100%

例如：某job需要11个stage

非并行35分钟

并行8个job执行10分钟

并行16个job执行6分钟