# 目录

1)—Hive几种数据导入方式

2)—Hive几种导出数据方式

3)—从文件中加载执行语句

4)—hive中所有join

5)—分区、分桶和索引

6)—hive参数设置方法

7)—hive性能调优

8)—hive注意点

9)—数据库创建、表创建、加载数据

# 0-HIVE基础

数据仓库工具

使用类sql语句编写业务逻辑 底层自动转换为MapReduce

# 1.Hive几种数据导入方式

## 1.从本地文件系统中导入数据到hive表

（1）数据准备（/home/sopdm/test.dat）：

1,wyp,25,13188888888

2,test,30,13899999999

3,zs,34,89931412

（2）首先创建表

use sopdm;

drop table if exists sopdm.wyp;

create table if not exists sopdm.wyp(id int,name string,age int,tel string)

row format delimited fields terminated by ','

stored as textfile;

（3）从本地文件系统中导入数据到Hive表

load data local inpath ‘/home/sopdm/test.dat’ into table sopdm.wyp;

（4）可以到wyp表的数据目录下查看，如下命令

dfs -ls /user/sopdm/hive/warehouse/sopdm.db/wyp;

2.从HDFS上导入数据到Hive表

（1）现在hdfs中创建一个input目录存放HDFS文件

hadoop fs -mkdir input; 或 hadoop fs -mkdir /user/sopdm/input;

（2）把本地文件上传到HDFS中,并重命名为test\_hdfs.dat

hadoop fs -put /home/sopdm/test.dat /user/sopdm/input/test\_hdfs.dat;

（3）查看文件

dfs -cat /user/sopdm/input/test\_hdfs.dat;

（4）将内容导入hive表中

--拷贝“本地数据”到“hive”使用：load data local…

--转移“HDFS”到“hive”（必须同一个集群）使用：load data…

load data inpath ‘/user/sopdm/input/test\_hdfs.dat’ into table sopdm.wyp;

3.从别的Hive表中导入数据到Hive表中

create table if not exists sopdm.wyp2(id int,name string,tel string)

row format delimited fields terminated by ','

stored as textfile;

--overwrite是覆盖，into是追加

insert into table sopdm.wyp2

select id,name,tel from sopdm.wyp;

\*\*/ select \* from wyp

--多表插入

--高效方式-查询语句插入多个分区

from sopdm.wyp w

insert overwrite table sopdm.wyp2

select w.id,w.name,w.tel where w.age=25

insert overwrite table sopdm.wyp2

select w.id,w.name,w.tel where w.age=27;

4.创建Hive表的同时导入查询数据

create table sopdm.wyp3

as select id,name,tel,age from sopdm.wyp where age=25;

5.使用sqoop从关系数据库导入数据到Hive表

这个放在sqoop总结中去讲解。

HDFS

mysql

hive

hbase

mongodb

redis KV nosql 9

## 2. Hive几种导出数据方式

1.拷贝文件

如果数据文件恰好是用户需要的格式，那么只需要拷贝文件或文件夹就可以。

hadoop fs –cp source\_path target\_path

2.导出到本地文件系统

--不能使用insert into local directory来导出数据，会报错

--只能使用insert overwrite local directory来导出数据

--hive0.11版本之前，只能使用默认分隔符^A(ascii码是\00001)

insert overwrite local directory '/home/sopdm/wrk'

select id,name,tel,age from sopdm.wyp;

--hive0.11版本之后，可以指定分隔符

insert overwrite local directory '/home/sopdm/wrk'

row format delimited

fields terminated by ','

select id,name,tel,age from sopdm.wyp;

--导出数据到多个输出文件夹

from employees se

insert overwrite local directory ‘/tmp/or\_employees’

select \* se where se.cty=’US’ and se.st=’OR’

insert overwrite local directory ‘/tmp/ca\_employees’

select \* se where se.cty=’US’ and se.st=’CA’

3.导出到HDFS

--比导出文件到本地文件系统少了一个local

insert overwrite directory '/home/sopdm/wrk'

select id,name,tel,age from sopdm.wyp;

--hive0.11版本之后，可以指定分隔符

insert overwritedirectory '/home/sopdm/wrk'

row format delimited

fields terminated by ','

select id,name,tel,age from sopdm.wyp;

4.导出到hive的另一张表

insert into table sopdm.wyp2

partition(age='25')

select id,name,tel,age from sopdm.wyp;

5.使用hive的-e和-f参数命令导出数据

--使用hive的-e参数

hive –e “select \* from wyp” >> /local/wyp.txt

--使用hive的-f参数, wyp.hql中为hql语句

hive –f wyp.hql >> /local/wyp2.txt

6.导出到关系型数据库

放在sqoop中讲。

## 3—从文件中加载执行语句

1. 从文件中执行hql语句

hive -f test.hql;

2. 在hive shell中使用source执行脚本

source test.hql;

## 4—hive中所有join

1. 内连接：inner join

--join优化：在进行join的时候，大表放在最后面

--但是使用 /\*+streamtable(大表名称)\*/ 来标记大表，那么大表放在什么位置都行了

select s.ymd,d.dividend

from stocks s inner join dividends d on s.ymd=d.ymd and s.symbol=d.symbol

where s.symbol=’aapl’;

create table stocks(ymd int,symbol string);

create table dividends(ymd int,symbol string);

2. 外连接：left outer join,right outer join,full outer join

把外连接(outer join)中where语句中的过滤条件，放在on语句是无效的。不过对于内连接有效。

3. 笛卡尔积：join

join的时候不使用on，而使用where的。

--笛卡尔积join很缓慢，可以设置hive.mapred.mode=strict来阻止执行

hive.mapred.mode=strict

4. 左半开连接：left semi-join（hive不支持右半开连接）

左半开连接（left semi-join）会返回左边表的记录，前提是其记录对于右边表满足on语句中的判定条件。

select和where语句中都不能引用右边表的字段。

适用场景：

因为hive不支持in...exists结构，所以要使用左半开连接代替.

注意点：

semi-join比inner join更高效

hive不支持右半开连接.

5. map端join：map-side join

map-side join：hive可以在map端执行连接过程（对于在join时有一个是小表的情况）

使用map-side join，需要配置下：

（1）hive0.7版本之前，需要加/\*+ mapjoin(表名) \*/

select /\*+ mapjoin(d) \*/ s.ymd,d.dividend from stocks s join dividends d

on s.ymd=d.ymd and s.symbol=d.symbol

where s.symbol=’AAPL

（2）hive0.7版本开始，设置hive.auto.convert.join=true

hive.auto.convert.join=true

hive.mapjoin.smalltable.filsize=25000000 --使用这个优化的小表的大小(单位：字节)--注意：右外连接和全外连接不支持这个优化

备注：

(1). hive的join语句，只支持等值连接。

(2). 注：pig提供的交叉生成功能支持“非等值连接”.

(3). hive目前不支持在join 的on子句中使用or。

## 5—分区、分桶和索引

# Hive分区和分桶

存在的目的主要是为了解决大表与大表间的 Join 问题，分桶其实就是把大表化成了“小表”，然后 Map-Side Join 解决之，这是典型的分而治之的思想。在聊 SMB Join 之前，我们还是先复习下相关的基础概念。

## 1、Hive 分区表

在Hive Select查询中一般会扫描整个表内容，会消耗很多时间做没必要的工作。有时候只需要扫描表中关心的一部分数据，因此建表时引入了partition概念。分区表指的是在创建表时指定的partition的分区空间。

Hive可以对数据按照某列或者某些列进行分区管理，所谓分区我们可以拿下面的例子进行解释。

当前互联网应用每天都要存储大量的日志文件，几G、几十G甚至更大都是有可能。存储日志，其中必然有个属性是日志产生的日期。在产生分区时，就可以按照日志产生的日期列进行划分。把每一天的日志当作一个分区。

将数据组织成分区，主要可以提高数据的查询速度。至于用户存储的每一条记录到底放到哪个分区，由用户决定。即用户在加载数据的时候必须显示的指定该部分数据放到哪个分区。

1.1 实现细节

1、一个表可以拥有一个或者多个分区，每个分区以文件夹的形式单独存在表文件夹的目录下。

2、表和列名不区分大小写。 mysql区别

3、分区是以字段的形式在表结构中存在，通过describe table命令可以查看到字段存在， 但是该字段不存放实际的数据内容，仅仅是分区的表示（伪列） 。

## 2、Hive 桶

对于每一个表（table）或者分区， Hive可以进一步组织成桶，也就是说桶是更为细粒度的数据范围划分。Hive也是 针对某一列进行桶的组织。Hive采用对列值哈希，然后除以桶的个数求余的方式决定该条记录存放在哪个桶当中。

把表（或者分区）组织成桶（Bucket）有两个理由：

（1）获得更高的查询处理效率。桶为表加上了额外的结构，Hive 在处理有些查询时能利用这个结构。具体而言，连接两个在（包含连接列的）相同列上划分了桶的表，可以使用 Map 端连接 （Map-side join）高效的实现。比如JOIN操作。对于JOIN操作两个表有一个相同的列，如果对这两个表都进行了桶操作。那么将保存相同列值的桶进行JOIN操作就可以，可以大大较少JOIN的数据量。

（2）使取样（sampling）更高效。在处理大规模数据集时，在开发和修改查询的阶段，如果能在数据集的一小部分数据上试运行查询，会带来很多方便。

1.分区

分区是以字段的形式在表结构中存在，通过describe table命令可以查看到字段存在， 但是该字段不存放实际的数据内容，仅仅是分区的表示（伪列）。

（1）静态分区

create table if not exists wyp12(id int,name string,tel string)

partitioned by(age int)

row format delimited fields terminated by ','

stored as textfile;

动态分区为主

上千万条数据

组合方式动态分区：eg： city，date

--overwrite是覆盖，into是追加

insert into table wyp12

partition(age='26')

select id,name,tel

from sopdm.wyp;

where age=26

insert into table wyp12

partition(age='27')

select id,name,tel

from sopdm.wyp;

where age=27

1.首先新建一个临时表 存储所有数据

2.建分区表 没有建详细分区信息 1次

3.手工插入不同分区的数据信息 N次

（2）动态分区

--设置为true表示开启动态分区功能（默认为false）

set hive.exec.dynamic.partition=true;

--设置为nonstrict,表示允许所有分区都是动态的（默认为strict）

set hive.exec.dynamic.partition.mode=nonstrict;

--insert overwrite是覆盖，insert into是追加

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table sopdm.wyp2

partition(age)

select id,name,tel,age from sopdm.wyp;

2. 分桶

CREATE TABLE bucketed\_user (id INT) name STRING)

CLUSTERED BY (id) INTO 4 BUCKETS;

对于每一个表（table）或者分区， Hive可以进一步组织成桶，也就是说桶是更为细粒度的数据范围划分。Hive也是 针对某一列进行桶的组织。Hive采用对列值哈希，然后除以桶的个数求余的方式决定该条记录存放在哪个桶当中。

把表（或者分区）组织成桶（Bucket）有两个理由：

（1）获得更高的查询处理效率。桶为表加上了额外的结构，Hive 在处理有些查询时能利用这个结构。具体而言，连接两个在（包含连接列的）相同列上划分了桶的表，可以使用 Map 端连接 （Map-side join）高效的实现。比如JOIN操作。对于JOIN操作两个表有一个相同的列，如果对这两个表都进行了桶操作。那么将保存相同列值的桶进行JOIN操作就可以，可以大大较少JOIN的数据量。

（2）使取样（sampling）更高效。在处理大规模数据集时，在开发和修改查询的阶段，如果能在数据集的一小部分数据上试运行查询，会带来很多方便。

3.索引

索引可以加快含有group by语句的查询的计算速度

create index employees\_index on table employees(country)

as 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'

with deferred rebuild

in table employees\_index\_table ;

## hive分区实战

hive引入partition和bucket的概念，中文翻译分别为分区和桶（我觉的不是很合适，但是网上基本都是这么翻译，暂时用这个吧），这两个概念都是把数据划分成块，分区是粗粒度的划分桶是细粒度的划分，这样做为了可以让查询发生在小范围的数据上以提高效率。

分区的作用：使用分区可以加快数据分片的查询速度。

桶的作用：(1)获得更高效的查询处理效率，桶为表加上了额外的结构。

(2)"取样"更高效。在处理大规模数据集时，在开发和修改查询阶段，如果能在数据集的一小部分数据上试进行查询，会带来很多方便。

（1）第一个

create table logs (ts bigint,line string) --ts timestamp line 每一行日志

partitioned by (dt string,country string) -- 分区列 dt 日志产生日期

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

load data local inpath '/input/hive/partitions/file1'

into table logs

partition(dt='2001-01-01',country='GB');

partition(dt='2002-02-02',country='TB');

//看下表的结构

hive> desc logs;

//查看一个表的所有分区

hive> show partitions logs;

SELECT ts, dt, line

FROM logs

WHERE country='GB';

（2）第二个

创建分区表

CREATE TABLE tb\_part\_shop (

shop\_id int,

shop\_name string,

shopkeeper string

) PARTITIONED BY (province\_id int, city\_id int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

描述分区表

>DESCRIBE EXTENDED tb\_part\_shop;

导入分区和数据 加分区

　　ALTER TABLE tb\_part\_shop ADD PARTITION (province\_id = 1001, city\_id = 10001);

　　数据内容

　　33068201 Wal-Mart wal-mart

　　33068202 Carrefour carrefour

　　LOAD DATA LOCAL INPATH '/path/to/data.txt' INTO TABLE tb\_part\_shop PARTITION(province\_id = 1001, city\_id = 10001);

（3）分桶

创建分桶表

CREATE TABLE tb\_bucket\_shop (

shop\_id int,

shop\_name string,

shopkeeper string

) CLUSTERED BY (shop\_id) INTO 4 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

set hive.enforce.bucketing=true;

　　INSERT OVERWRITE TABLE tb\_bucket\_shop SELECT shop\_id, shop\_name, shopkeeper FROM tb\_part\_shop CLUSTER BY shop\_id;

set mapred.reduce.tasks=4;

　　INSERT OVERWRITE TABLE tb\_bucket\_shop SELECT shop\_id, shop\_name, shopkeeper FROM tb\_part\_shop CLUSTER BY shop\_id;

查看表的结构

hive> dfs -ls /user/hive/warehouse/tb\_bucket\_shop

读取数据，看每一个文件的数据：

hive> dfs -cat /user/hive/warehouse/bucketed\_users/\*0\_0;

对桶中的数据进行采样：

hive> SELECT \* FROM bucketed\_users

> TABLESAMPLE(BUCKET 1 OUT OF 4 ON id);

/

## hive分桶实战

create table bucket\_user (id int,name string)

clustered by (id) into 4 buckets;

关键字clustered声明划分桶的列和桶的个数，这里以用户的id来划分桶，划分4个桶。

以下为了简便划分桶的列简称为桶列

hive会计算桶列的hash值再以桶的个数取模来计算某条记录属于那个桶

已经存在的表：

hive> select \* from users;

OK

0 Nat

2 Joe

3 Kay

4 Ann

hive> set hive.enforce.bucketing=true --必须设置这个数据，hive才会按照你设置的桶的个数去生成数据

下面把user的数据导入到bucketed\_users中

insert overwrite table bucketed-users

select \* from users;

然后见证奇迹的时刻：

hive> dfs -ls /user/hive/warehouse/bucketed\_users;

hive> dfs -cat /user/hive/warehouse/bucketed\_users/000000\_0;

0Nat

4Ann

下面来看看利用bucket来对示例数据进行查询

---带桶的表

select \* from bucketed\_users

tablesample(bucket 1 out of 4 on id);

---不带桶的表

select \* from users

tablesample(bucket 1 out of 4 on rand())

## 6—hive参数设置方法

1. 参数声明：在hive的CTL环境中设置（临时）

比如：set hive.fetch.task.conversion=more;

2.命令行设置：使用hadoop命令设置（临时）

hive --hiveconf hive.fetch.task.conversion=more;

3.配置文件：--${HIVE\_HOME}/conf/hive-site.xml里面加入以下配置（长期）

<property>

<name>hive.fetch.task.conversion</name>

<value>more</value>

</property>

4.修改文件$HOME/.hiverc文件（长期，当前用户）

优先级：参数声明 > 命令行参数 > .hiverc文件 > 配置文件设定

## 7—hive性能调优

1.limit限制调整

--因为使用limit语句时候，是先执行整个查询语句，然后再返回部分结果的

set hive.limit.optimize.enable=true;

set hive.limit.row.max.size=10000;

set hive.limit.optimize.limit.file=10;

2.JOIN优化

。。。

3. 本地模式

--hive尝试使用本地模式执行查询，要不然hive会使用MapReduce来执行其他所有的查询

set hive.exec.mode.local.auto=true;

4.并行执行

set hive.exec.parallel=true;

5.严格模式

--对分区表进行查询，在where子句中没有加分区过滤的话，将禁止提交任务(默认：nonstrict)

set hive.mapred.mode=strict;

注：使用严格模式可以禁止3种类型的查询：

（1）对于分区表，不加分区字段过滤条件，不能执行

（2）对于order by语句，必须使用limit语句。

（3）限制笛卡尔积的查询（join的时候不使用on，而使用where的）。

6.调整mapper和reducer个数

set hive.exec.reducers.max=(集群总reduce槽位个数\*1.5)/(执行中的查询的平均个数)

7.JVM重用

set mapred.job.reuse.jvm.num.tasks=10; --10为重用个数

8.索引

索引可以加快含有group by语句的查询的计算速度

9.动态分区调整

--动态分区属性：设置为true表示开启动态分区功能（默认为false）

hive.exec.dynamic.partition=true;

--动态分区属性：设置为nonstrict,表示允许所有分区都是动态的（默认为strict）

--设置为strict，表示必须保证至少有一个分区是静态的

hive.exec.dynamic.partition.mode=strict;

--动态分区属性：每个mapper或reducer可以创建的最大动态分区个数

hive.exec.max.dynamic.partitions.pernode=100;

--动态分区属性：一个动态分区创建语句可以创建的最大动态分区个数

hive.exec.max.dynamic.partitions=1000;

--动态分区属性：全局可以创建的最大文件个数

hive.exec.max.created.files=100000;

--控制DataNode一次可以打开的文件个数

--这个参数必须设置在DataNode的$HADOOP\_HOME/conf/hdfs-site.xml文件中

<property>

<name>dfs.datanode.max.xcievers</name>

<value>8192</value>

</property>

10.推测执行

--目的：是通过加快获取单个task的结果以及进行侦测将执行慢的TaskTracker加入到黑名单的方式来提高整体的任务执行效率

（1）修改 $HADOOP\_HOME/conf/mapred-site.xml文件

<property>

<name>mapred.map.tasks.speculative.execution </name>

<value>true</value>

</property>

<property>

<name>mapred.reduce.tasks.speculative.execution </name>

<value>true</value>

</property>

（2）修改hive配置

set hive.mapred.reduce.tasks.speculative.execution=true;

11.单个MapReduce中多个group by

--多个group by操作组装到单个MapReduce任务中

set hive.multigroupby.singlemr=false;

12.虚拟列

--当hive产生了非预期的或null的时候，可以通过虚拟列进行诊断，判断哪行数据出现问题

INPUT\_\_FILE\_\_NAME （输入文件名）

BLOCK\_\_OFFSET\_\_INSIDE\_\_FILE （块内偏移量）

ROW\_\_OFFSET\_\_INSIDE\_\_BLOCK (行偏移量，需要设置hive.exec.rowoffset=true;启用)

13. 其他参数调优

--开启CLI提示符前打印出当前所在的数据库名

set hive.cli.print.current.db=true;

--让CLI打印出字段名称

hive.cli.print.header=true;

--提高聚合的性能

set hive.map.aggr=true;

--对于简单的不需要聚合的类似SELECT <col> from <table> LIMIT n语句，不需要起MapReduce job，直接通过Fetch task获取数据

set hive.fetch.task.conversion=more;

## 8—hive注意点

--自动补全功能：tab键

--可以把“设置系统属性变量，增加hadoop分布式内存，自定义hive扩展的jar包”放在“$HOME/.hiverc”文件中

--在hive的CLI中执行bash shell命令,只要在命令前加上!，并且以分号(;)结尾就可以

! /bin/echo “what up dog”;

--在hive中使用hadoop的dfs命令

dfs -ls /;

--查看表的存储路径

hive -S -e "set" | grep warehouse;

--开启hadoop回收站功能

--删除的数据被转移到HDFS中的/user/$USER/.Trash目录中

-- fs.trash.interval后面的整数是“回收站检查点”的时间间隔，单位是分钟

fs.trash.interval=1440 --24小时

--注意谓语操作符，“A=B”和“A<=>B”的区别

A=B --A等于B，返回true，反之返回false

A<=>B --如果A和B都为null返回true，其他的等号(=)一样

--修改log级别

--默认的日志存放路径：

/tmp/<user.name>/hive.log

--通过配置可以指定Hive中log4j日志配置文件的存放路径

hive -hiveconf hive.log4j.file=/home/carl/hive-log4j.properties

-hiveconf hive.log4j.exec.file=/home/carl/hive-exec-log4j.properties

--临时

hive --hiveconf hive.root.logger=DEBUG,console

--修改${HIVE\_HOME}/conf/hive-log4j.properties文件（永久）

hive.root.logger=DEBUG,console

--set hiveconf:hive.root.logger=DEBUG,console;是无效的，不能使用的。因为设定log的参数读取在会话建立以前已经完成了。

--case ...when...then句式

select name,salary,

case

when salary < 50000.0 then ‘low’

when salary > 50000.0 and salary < 7000.0 then ‘middle’

else ‘high’

end as bracket

from employees;

--浮点数尽量使用double，对于已经使用float的要使用cast操作符

--和钱相关的都避免使用浮点数

cast(0.2 as float)

--不过将浮点型转换为整数，推荐使用round()或floor()，而不是用cast。

--hive的join语句，只支持等值连接。（注：pig提供的交叉生成功能支持“非等值连接”）

--hive目前不支持在join 的on子句中使用or。

--order by：全局排序

--sort by：局部排序

--如果hive.mapred.mode=strict;时候，因为order by 时间比较长。必须加上limit限制。

--distribute by来保证具有相同属性值的记录会分到同一个reduce中进行处理

select s.ymd,s.symbol,s.price\_close

from stocks s

distribute by s.symbol

sort by s.symbol ASC,s.ymd ASC;

--如果distribute by和sort by语句中涉及到的列完全相同，而且采用的默认的升序排序方式。那么在这种情况下，可以使用cluster by来代替。

--注意：使用distribute by和sort by语句 或者简化版的cluster by语句会剥夺sort by的并行性，然而这样可以实现输出文件的数据是全局排序的。

-- explain dependency语法：以JSON格式输出table或partition信息

explain dependency select \* from wyp;

-- TRUNCATE可以删除HDFS上面相关表存储的数据，但是会保持表和metadata的完整性

truncate table p;

--在开发了hive的udf udaf udtf函数的jar文件后，需要将jar文件放入hive的环境中才可以使用。

add jar path/test.jar;

## 9—数据库创建、表创建、加载数据

--创建数据库

create database if not exists sopdm

comment ‘this is test database’

with dbproperties(‘creator’=’gxw’,’date’=’2014-11-12’) --数据库键值对属性信息

location ‘/my/preferred/directory’;

--查看数据库的描述信息和文件目录位置路径信息

describe database sopdm;

--查看数据库的描述信息和文件目录位置路径信息(加上数据库键值对的属性信息)

describe database extended sopdm;

--删除数据库

drop database if exists sopdm;

--级联删除数据库(当数据库还有表时，级联删除表后在删除数据库),默认是restrict

drop database if exists sopdm cascade;

--修改数据库

--只能修改数据库的键值对属性值。数据库名和数据库所在的目录位置不能修改

alter database sopdm set dmproperties(‘edited-by’=’gaoxianwei’);

--创建表

--其中tblproperties作用：按照键值对的格式为表增加额外的文档说明，也可用来表示数据库连接的必要的元数据信息

--hive会自动增加二个表属性：last\_modified\_by(最后修改表的用户名)，last\_modified\_time(最后一次修改的时间)

create table if not exists sopdm.test1(name string comment ‘姓名’,salary float comment ‘薪水’)

comment ‘这是一个测试的表’

tblproperties(‘creator’=’me’,’created\_at’=’2014-11-13 09:50:33’)

location ‘/user/hive/warehouse/sopdm.db/test1’

--查看和列举表的tblproperties属性信息

show tblproperties table\_name;

--使用like在创建表的时候，拷贝表模式(而无需拷贝数据)

create table if not exists sopdm.test2 like sopdm.test1;

--查看表的详细结构信息（也可以显示表是管理表，还是外部表。还有分区信息）

describe extended sopdm.test1;

--使用formatted信息更多些，可读性更强

describe formatted sopdm.test1;

--创建外部表

--删除表时，表的元数据会被删除掉，但是数据不会被删除

--如果数据被多个工具（如pig等）共享，可以创建外部表

create external table if not exists sopdm.test1(

name string comment ‘姓名’,

salary float comment ‘薪水’)

comment ‘这是一个测试的表’

tblproperties(‘creator’=’me’,’created\_at’=’2014-11-13 09:50:33’)

location ‘/user/hive/warehouse/sopdm.db/test1’

--分区表

create table if not exists sopdm.test1(

name string comment ‘姓名’,

salary float comment ‘薪水’)

comment ‘这是一个测试的表’

partitioned by(country string,state string)

STORED AS rcfile

tblproperties(‘creator’=’me’,’created\_at’=’2014-11-13 09:50:33’)

location ‘/user/hive/warehouse/sopdm.db/test1’

--查看表中存在的所有分区

show partitions table\_name;

--查看表中特定分区

show partitions table\_name partition(country=’US’);

--可以在表载入数据的时候创建分区

load data local inpath ‘${env:HOME/employees}’

into table employees

partition(country=’US’,state=’CA’);

--删除表

drop table if exists table\_name;

--修改表-表重命名

alter table old\_table\_name rename to new\_table\_name;

--增加分区

alter table table\_name add if not exists partition(year=2011,month=1,day=1)

location ‘/logs/2011/01/01’;

--修改分区存储路径

alter table table\_name partition(year=2011,month=1,day=2)

set location ‘/logs/2011/01/02’;

--删除某个分区

alter table table\_name drop if exists partition(year=2011,month=1,day=2);

--修改列信息

alter table table\_name

change column old\_name new\_name int

comment ‘this is comment’

after severity; --字段移到severity字段之后（移动到第一个位置，使用first关键字）

--增加列

alter table table\_name add columns(app\_name string comment ‘application name’);

--删除或者替换列

alter table table\_name replace columns(hms int comment ‘hhh’);

--修改表属性

alter table table\_name set tblproperties(‘notes’=’this is a notes’);

--修改存储属性

alter table table\_name partition(year=2011,month=1,day=1) set fileformat sequencefile;

--指定新的SerDe,并指定SerDe属性

alter table table\_name

set serde “com.example.JSONSerDe”

with serdeproperties(‘prop1’=‘value1’, ‘prop2’=‘value2’);

--增加执行“钩子”——当表中存储的文在hive之外被修改了，就会触发钩子的执行

alter table table\_name touch partition(year=2012,month=1,day=1);

--将分区内的文件打成hadoop压缩包文件，只会降低文件系统中的文件数，减轻NameNode的压力，而不会减少任何的存储空间

--使用unarchive替换archive起到反向操作

alter table table\_name archive partition(year=2012,month=1,day=1);

--防止分区被删除和被查询(使用enable替代disable可以起到反向的操作目的)

alter table table\_name partition(year=2012,month=1,day=1) disable no\_drop;

alter table table\_name partition(year=2012,month=1,day=1) disable offline;

--向管理表中装载数据

-- inpath为一个目录，而且这个路径下不可以包含任何文件夹

load data local inpath ‘${env:HOME}/table\_name’

overwrite into table table\_name

partition(country=’US’);

--通过查询语句向表中插入数据

--overwrite是覆盖，into是追加

insert overwrite table table\_name

partition(country=’US’)

select \* from table\_name2 tn where tn.cnty=’US’

--高效方式-查询语句插入多个分区

from table\_name2 tn

insert overwrite table table\_name

partition(country=’US’,state=’OR’)

select \* where tn.cnty=’US’ and tn.st=’OR’

insert overwrite table table\_name

partition(country=’US’,state=’CA’)

select \* where tn.cnty=’US’ and tn.st=’CA’

--动态插入分区

--hive根据select语句最后2列确定分区字段country和state的值（根据位置）

insert overwrite table table\_name

partition(country,state)

select …,se.cnty,se.st

from employees se;

--动态和静态分区结合

--country为静态分区，state为动态分区（静态分区必须在动态分区之前）

insert overwrite table table\_name

partition(country=‘US’,state)

select …,se.cnty,se.st

from employees se

where se.cnty=’US’;

--单个查询语句中创建表并加载数据

create table table\_name1

as select name,salary,address from table\_name2 where state=’CA’;

--导出数据——拷贝文件

--如果数据文件恰好是用户需要的格式，那么只需要简单的拷贝文件或文件夹就可以。

hadoop fs –cp source\_path target\_path

--导出数据

insert overwrite local directory ‘/tmp/employees’

select name,salary,address from employees se where se.state=’CA’

--导出数据到多个输出文件夹

from employees se

insert overwrite local directory ‘/tmp/or\_employees’

select \* se where se.cty=’US’ and se.st=’OR’

insert overwrite local directory ‘/tmp/ca\_employees’

select \* se where se.cty=’US’ and se.st=’CA’