# ------------------环境搭建--------------------------

1. 把Hive移动到/usr/local/cloud/目录下并解压

2. ln -s apache-hive-1.2.0-bin/ hive

3. 修改配置文件

3.1 修改文件名，去掉template

hive-default.xml.template

hive-env.sh.template

hive-exec-log4j.properties.template

hive-log4j.properties.template

3.2 /etc/profile

export HIVE\_HOME=/usr/local/cloud/hive

export PATH=$PATH:$HIVE\_HOME/bin

3.3 hive-env.sh

HADOOP\_HOME=/usr/local/cloud/hadoop

export HIVE\_CONF\_DIR=/usr/local/cloud/hive/conf

4. 启动Hive:

直接运行hive

5. 测试

运行show tables; 有OK出现表示安装成功。

问题解答：

1. “Found class jline.Terminal, but interface was expected”

这个错误表示hive用的jline.jar与hadoop下的jline.jar版本不一致

将hive下的拷贝到hadoop中

# ------------------配置相关--------------------------

0. 配置存储

在$HOME目录下的.hiverc文件中

1. 在提示符中显示数据库名

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

2. 优先使用本地模式执行

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

3. 打印列名

set hive.cli.print.header=true;

4. （非）严格模式

set hive.mapred.mode=strict;

set hive.mapred.mode=nonstrict;

5. 开启动态分区

set hive.exec.dynamic.partition=true;

# --------------动态分区调整--------------------

-- 设置动态分区模式

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

-- 设置总的动态分区个数

set hive.exec.max.dynamic.partitions=300000

-- 设置每个节点上动态分区个数

set hive.exec.max.dynamic.partitions.pernode=10000

9. 设置全局可以产生文件的个数

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

10.开启map site join

set hive.auto.convert.join=true;

11.设置小表大小（字节）

set hive.mapjoin.smalltable.filesize=25000000

12.强制将数据按照桶结构定义来插入数据

set hive.enforce.bucketing=true

13.limit优化

-- 启用limit优化

set hive.limit.optimize.enable=true;

-- 限制从最大多少条数据中进行limit

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

-- 限制最多遍历的文件个数

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

14. 压缩

--开启中间压缩（即map到reduce之间的数据压缩）

set hive.exec.compress.intermediate=true;

-- 开启hadoop中间压缩（即map到reduce之间的数据压缩）

set mapred.compress.map.output=true;

-- 开启hive最终压缩（即reduce输出的数据压缩）

set hive.exec.compress.output=true;

4. 数据仓库的存储地址

hive-default.xml中

<property>

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

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

# ------------------变量定义-------------------------

1、定义

hive --hivevar myname=wufan;

2、获取变量值

${hivevar:myname}

3、显示变量

set myname;

set hivevar:myname;

4、修改（或赋值）变量

set hivevar:myname=shawn;

# ------------------使用变量-------------------------

# 方式一

-- my.sh

#!/bin/bash

tablename="student"

limitcount="8"

hive -S -e "use test; select \* from ${tablename} limit ${limitcount};"

# 方式二

-- my.sh

#! /bin/bash

hive --hivevar tableName=T1 --hivevar tableField1=F1 --hivevar tableField2=F2 -f my.hql

-- my.hql

use mydb;

create table if not exists ${hivevar:tableName}

(

${hivevar:tableField1} string,

${hivevar:tableField2} string

);

# ------------------执行相关-------------------------

1. 执行外部的脚本

hive -f myscript.hql

2. 直接执行脚本

hive -e "select name from user"

3. 静默执行

hive -S myscript.hql

# ------------------数据库相关------------------------

1. 创建数据库

create database if not exists mydb;

2. 展示数据库

show databases;

3. 查看数据库结构

describe database mydb;

4. 数据库切换

use mydb;

5. 删除数据库

drop database if exists mydb cascade;

# ------------------数据表相关------------------------

# ------------------创建数据表------------------------

1. 创建内部表

create table if not exists friends (

name string,

age int

);

create table if not exists friends (

name string,

age int

)

row format delimited

fields terminated by ','

collection items terminated by ':'

map keys terminated by '='

lines terminated by '\n'

stored as textfile;

2. 复制内部表结构

create table if not exists friends2

like friends;

3. 查看表结构

describe mytable;

describe formatted mytable;

4. 创建外部表

create external table if not exists friends (

name string,

age int

)

row format delimited

fields terminated by ','

location '/data/friends';

5. 复制外部表结构

create table if not exists friends3

like friends

location '/data/friends3';

6. 创建分区表

use mydb;

-- 创建内部分区表

create table if not exists mytable (

f1 string,

f2 string

)

partitioned by (cnt string, ct string);

注：分区字段不能和表字段名称一致

-- 创建外部分区表（指定路径）

-- 表目录存放在指定路径下

create external table if not exists mytable2 (

f1 string,

f2 string

)

partitioned by (cnt string, ct string)

row format delimited

fields terminated by ','

location '/data/mytable2';

-- 创建外部分区表（不指定路径）

-- 表目录存放在缺省路径下

create external table if not exists mytable3 (

f1 string,

f2 string

)

partitioned by (cnt string, ct string)

row format delimited

fields terminated by ',';

-- 指定分区路径

alter table mytable3 add if not exists partition (cnt='CA',ct='BJ')

location '/data/mytable3/CA/BJ';

alter table mytable3 add if not exists partition (cnt='CA',ct='SH')

location '/data/mytable3/CA/SH';

7. 显示分区信息

show partitions mytable3;

show partitions mytable3 partition(cnt='CA');

show partitions mytable3 partition(cnt='CA', ct='BJ');

-- 显示分区路径

describe formatted mytable3 partition(cnt='CA', ct='TJ');

8. 分区管理

-- 增加分区

alter table mytable3 add if not exists partition (cnt='CA', ct='TJ')

location '/data/mytable3/CA/TJ';

-- 修改分区

alter table mytable3 partition (cnt='CA', ct='TJ')

set location 'hdfs://127.0.0.1:9000/data/mytable3/newPath/CA/TJ';

-- 删除分区

alter table mytable3 drop if exists partition (cnt='CA', ct='TJ');

# ------------------修改数据表------------------------

-- 修改表名 数据所存储的路径还在mytable3中

alter table mytable3 rename to mytable4;

-- 修改列

alter table mytable4 change f2 f22 string after f1;

-- 增加列

alter table mytable4 add columns (f3 string, f4 string);

-- 修改表存储类型

alter table mytable4 partition (cnt='CA', ct='SH')

set fileformat sequencefile;

-- 隐藏分区数据 (disable)

alter table mytable4

partition (cnt='CA', ct='SH')

enable offline;

# ------------------数据操作------------------------

-- 案例分析

-- 案例一：针对新业务进行大数据存储规划

-- 0. 准备数据表

create external table if not exists mytable (

f1 string,

f2 string,

f3 string

)

partitioned by (cnt string, ct string)

row format delimited

fields terminated by ':'

location '/data/mytable';

-- 1. 加载本地数据

load data local inpath '/home/dev/data/US\_NY\_friends'

overwrite into table mytable

partition (cnt='US', ct='NY');

load data local inpath '/home/dev/data/CA\_BJ\_friends'

overwrite into table mytable

partition (cnt='CA', ct='BJ');

-- 案例二：针对老业务进行大数据存储规划

-- 关注点：1.数据存储的路径（可能为分散的多路径）

2.数据格式不统一（体现在：列分隔符、文件格式如序列化/文本/压缩等、值不统一）

-- 0. 准备数据表

create external table if not exists mytable2 (

f1 string,

f2 string,

f3 string

)

partitioned by (cnt string, ct string)

row format delimited

fields terminated by ':';

alter table mytable2 add if not exists partition (cnt='US', ct='NY')

location '/data/part1/US/NY';

alter table mytable2 add if not exists partition (cnt='CA', ct='BJ')

location '/data/part2/CA/BJ';

-- 通过查询来加载数据

create table if not exists mytable3 (

f1 string,

f2 string,

f3 string

)

partitioned by (cnt string, ct string);

-- 方式一 针对单分区

insert overwrite table mytable3

partition (cnt='CA', ct='BJ')

select \* from mytable2 mt2

where mt2.cnt='CA' and mt2.ct='BJ';

-- 方式二 针对多表多分区

from mytable2 mt2

insert overwrite table mytable3

partition (cnt='CA', ct='BJ')

select f1,f2,f3 where mt2.cnt='CA' and mt2.ct='BJ'

insert overwrite table mytable3

partition (cnt='US', ct='NY')

select f1,f2,f3 where mt2.cnt='US' and mt2.ct='NY';

-- 方式三 动态分区插入

-- drop table if exists mytable3;

-- 设置动态分区模式为非严格模式，这样可以指定全部的分区字段都为动态的

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

insert overwrite table mytable3

partition (cnt, ct)

select mt2.f1,mt2.f2,mt2.f3,mt2.cnt,mt2.ct from mytable2 mt2;

-- 方式四 静态分区插入

insert overwrite table mytable3

partition (cnt='CA',ct)

select mt2.f1,mt2.f2,mt2.f3,mt2.ct

from mytable2 mt2 where mt2.cnt='CA';

注：

1、静态分区要放在动态分区前面

2、静态分区的字段不要出现在select后面

-- 方式五 创建并加载数据（用于内部表）

create table ca\_employees

as

select name,salary from employees

where se.state='CA';

# ------------------导出数据------------------------

-- 方式一

hadoop fs -cp /src\_path /dst\_path

-- 方式二

-- 导出到本地

insert overwrite local directory '/home/dev/data/out'

select f1,f2,f3 from mytable

where cnt='CA';

-- 导出到HDFS

insert overwrite directory '/data/out'

select f1,f2,f3 from mytable

where cnt='CA';

-- 方式三

from mytable mt

insert overwrite directory '/data/out/data\_ca'

select mt.f1,mt.f2,mt.f3 where mt.cnt='CA'

insert overwrite directory '/data/out/data\_us'

select mt.f1,mt.f2,mt.f3 where mt.cnt='US'

# ------------------查询数据------------------------

-- 针对集合字段

select f1, f2[0], f2[1] from mytable;

-- 针对Map字段

select f1, f3["mykey"] from mytable;

-- 针对Struct字段

select f1, f4.p1 from mytable;

-- 嵌套查询

from (

select f1,f2,f3 from mytable1

where cnt='CA'

) mt1

select mt1.f1,mt1.f2,mt1.f3

where mt1.f3>100;

-- case...when...then

select f1, cnt,

case

when cnt='CA' then 'China'

when cnt='US' then 'America'

else 'Unknown'

end as country

from mytable2;

# ------------------浮点比较------------------------

1. 尽量全部统一采用一种类型，如double

2. 当特殊情况下需要进行转换时

select f1,f2 from talbe where f2 > cast(0.2 as float);

3. 和钱相关的避免使用float

# ------------------group by----------------------

-- select后的字段必须是分组字段或要进行聚合的字段

select f1, avg(f2) from mytable where f3 > 100

group by f1;

# ------------------having------------------------

select f1, avg(f2) from mytable where f3 > 100

group by f1

having avg(f2) > 50;

# ------------------inner join---------------------

-- 双表join

select t1.f1,t1.f2,t2.f3

from table1 t1 join table2 t2

on t1.f1 = t2.f1

where t1.f2='CA' and t2.f2='BJ';

-- 多表join

select t1.f1,t1.f2,t2.f3

from table1 t1 join table2 t2

on t1.f1 = t2.f1 join table3 t3 on t1.f1 = t3.f1

where t1.f2='CA' and t2.f2='BJ';

# ------------------left out join---------------------

哪个表数据多，放左边

以左表为基准，打印出匹配左表的数据，右表不匹配则为NULL

# ------------------left semi join---------------------

相当于SQL的in

它的效率要远高于inner join

# ------------------map site join---------------------

map site join是指在map阶段进行表关联

小表会被放入到内存中，来与大表在map阶段完成关联

-- 开启map site join

set hive.auto.convert.join=true;

-- 设置小表大小（字节）

set hive.mapjoin.smalltable.filesize=25000000

# ------------------sort by---------------------

# ------------------distribute by---------------

-- 对每个省份的高考成绩进行降序排列

select \* from 高考成绩

distribute by 省份

sort by 成绩 desc;

# ------------------优化方案---------------------

1.用ON能够优化mapreduce

2.表大小的顺序为从左到右依次增大

3.在语句中增加分区的过滤条件

# ------------------类型转换---------------------

select \* from mytable3

where cast(f2 as float) > 30;

# ----------------------------------------------

# ------------------分桶意义---------------------

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。连接两个在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。

（2）使取样（sampling）更高效。

# ------------------分桶管理---------------------

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。

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

# ------------------数据导入---------------------

1. 利用hive来生成桶数据，就是将现有的表的数据导入到新定义的带有桶的表中。

hive> select \* from users;

0 Nat

2 Joe

3 Kay

4 Ann

-- 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

-- 每个桶对应一个reduce

set hive.enforce.bucketing=true

insert overwrite table bucketed\_users

select \* from users;

# ------------------分桶抽样---------------------

-- 带桶表的抽样

select \* from bucketed\_users

tablesample(bucket 1 out of 4 on id);

-- 不带桶表的抽样

select \* from users

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

# ------------------分桶例子1---------------------

use mydb;

-- 内部分区桶表

create table if not exists user\_bucket (

f1 string,

f2 string,

f3 string

)

partitioned by (cnt string, ct string)

clustered by (f1) into 4 buckets;

-- 外部分区表

create external table if not exists ext\_user\_bucket (

f1 string,

f2 string,

f3 string

)

partitioned by (cnt string, ct string)

clustered by (f1) into 4 buckets

location '/data/ext\_user\_bucket';

-- 设置启用按桶进行数据插入

set hive.enforce.bucketing=true;

from mytable2

insert overwrite table user\_bucket

partition (cnt='CA', ct='BJ')

select f1,f2,f3 where cnt='CA' and ct='BJ'

insert overwrite table user\_bucket

partition (cnt='US', ct='NY')

select f1,f2,f3 where cnt='US' and ct='NY';

-- 将a b c d e f g h八个桶，每隔2个取第1个

-- 结果为 a c e g

select \* from user\_bucket

tablesample (bucket 1 out of 2 on f1);

-- 将a b c d e f g h八个桶，每隔4个取第1个

-- 结果为 a e

select \* from user\_bucket

tablesample (bucket 1 out of 4 on f1);

# ------------------分桶例子2---------------------

总数据为100W条，希望抽样出1000条，分40个桶

bucket x out of y on f1

求，y是多少？

解答：y = 100w / 1000 = 1000 与桶的个数无关

# ------------------视图---------------------

create view if not exists myview as

select \* from mytable

where f1='A' and f2='B';

# ------------------优化---------------------

-- join时，小表放左边，大表放右边

-- 严格模式

set hive.mapred.mode=strict;

-- 1、where中必须含有分区字段

-- 2、order by语句中必须有limit

-- 3、必须通过on来设置关联字段

# ------------------mapper和reducer个数---------------------

# ------------------动态分区---------------------

-- 设置动态分区模式

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

-- 设置总的动态分区个数

set hive.exec.max.dynamic.partitions=300000

-- 设置每个节点上动态分区个数

set hive.exec.max.dynamic.partitions.pernode=10000

# ------------------数据压缩---------------------

-- 开启hive中间压缩（即map到reduce之间的数据压缩）

set hive.exec.compress.intermediate=true;

-- 设定中间压缩方式

<property>

<name>mapred.map.output.compression.codec</name>

<value>org.apache.hadoop.io.compress.SnappyCodec</value>

</property>

-- 开启hadoop中间压缩（即map到reduce之间的数据压缩）

set mapred.compress.map.output=true;

-- 开启hive最终压缩（即reduce输出的数据压缩）

set hive.exec.compress.output=true;

-- 设定最终压缩方式

<property>

<name>mapred.output.compression.codec</name>

<value>org.apache.hadoop.io.compress.SnappyCodec</value>

</property>

# ------------------序列化---------------------

create table if not exists mytable (

f1 string,

f2 string

)

stored as sequencefile;

-- 设定序列化的压缩方式 none record block

<property>

<name>mapred.output.compression.type</name>

<value>block</value>

</property>

# ------------------压缩序列化综合应用---------------

-- 启用中间压缩

set hive.exec.compress.intermediate=true;

-- 启用最终压缩

set hive.exec.compress.output=true;

-- 最终的文件格式（序列化）

set mapred.output.compression.type=block;

-- 中间压缩编码

set mapred.map.output.compression.codec

=org.apache.hadoop.io.compress.SnappyCodec;

-- 最终压缩编码

set mapred.output.compression.codec

=org.apache.hadoop.io.compress.GzipCodec;

# ------------------使用UDF函数---------------

-- 示例一：

中国银行 2.05 2.25 2.85

工商银行 2.05 2.25 2.85

建设银行 2.05 2.25 2.85

-- 将上述利率上浮25个基点

-- 1.创建表并导入数据

use mydb;

create external table if not exists bank (

name string,

f1 double,

f2 double,

f3 double

)

partitioned by (y string,m string)

row format delimited

fields terminated by ' '

location '/data/bank';

load data local inpath '/home/dev/data/bank'

overwrite into table bank partition(y='2015',m='5');

-- 2.编写java代码，实现UDF函数

public class Rate extends UDF {

public double evaluate(double v) {

v = v \* (1 + 0.25);

return v;

}

}

-- 3.上载jar

add jar /home/dev/code/udf.jar;

-- 4.定义udf函数名

create temporary function rate

as 'com.shawn.hive.udf.Rate';

-- 5.使用udf函数

select name,f1,rate(f1) as f11,f2,rate(f2) as f22,f3,rate(f3) as f33

from bank;

-- 示例二：

-- AA表

T440-001 aa bb cc

T440-002 dd ee ff

T440-003 hh ii jj

-- BB表

T440001 11 22 33

T440002 44 55 66

T440003 77 88 99

-- 将上述两表进行关联，得到下面数据：

T440001 aa bb cc 11 22 33

T440002 dd ee ff 44 55 66

T440003 hh ii jj 77 88 99

-- 1.创建表并导入数据

use mydb;

create table if not exists AA (

item string,

f1 string,

f2 string,

f3 string

)

row format delimited

fields terminated by ' '

location '/data/AA';

create table if not exists BB (

item string,

f1 string,

f2 string,

f3 string

)

row format delimited

fields terminated by ' '

location '/data/BB';

load data local inpath '/home/dev/data/d03'

overwrite into table AA;

load data local inpath '/home/dev/data/d04'

overwrite into table BB;

-- 2.编写java代码，实现UDF函数

public class GetItem extends UDF{

public String evaluate(String id) {

id = id.replace("-", "");

return id;

}

}

-- 3.上传jar

add jar /home/dev/code/udf.jar;

-- 4.定义udf函数名

create temporary function getitem

as 'com.shawn.hive.udf.GetItem';

-- 5.使用udf函数

select b.item,a.f1,a.f2,a.f3,b.f1,b.f2,b.f3

from AA a join BB b on getitem(a.item)==b.item;

-- 可以将下面两条放入到.hiverc文件中

-- 使函数一直有效

add jar /home/dev/code/udf.jar;

create temporary function rate

as 'com.shawn.hive.udf.Rate';

# ------------------------------------------------

# --------------自定义hive文件格式serde------------

-- sequencefile

create table if not exists serde1 (

f1 string,

f2 string

)

stored as sequencefile;

create table if not exists serde2 (

f1 string,

f2 string

)

inputformat 'org.apache.hadoop.mapred.TextInputFormat'

outputformat 'org.apache.hadoop.hive.ql.io.HiveSequenceFileOutputFormat';

# ------------------------------------------------

# ------------------实训练习一---------------------

# 每分钟插入一些数据，存放在不同的分区下

-- mycron

\* \* \* \* \* /home/dev/test/my.sh

-- my.sh

#! /bin/bash

source /etc/profile;

# 用于增加的分区名称

date1=`date +%Y-%m-%d-%H-%M`

# 用于删除的分区名词

date2=`date +%Y-%m-%d-%H-%M -d '-5 minutes'`

hive --hivevar date1=${date1} --hivevar date2=${date2} -f /home/dev/test/my.hql

# 删除分区对应的数据目录

hadoop fs -rm -r "/data/mytest/${date2}"

-- my.hql

use mydb;

create external table if not exists mytest (

name string,

age int,

sex string

)

partitioned by (dt string)

row format delimited

fields terminated by ':'

location '/data/mytest';

alter table mytest add if not exists partition (dt='${hivevar:date1}')

location "/data/mytest/${hivevar:date1}";

load data local inpath '/home/dev/data/CA\_BJ\_friends'

into table mytest

partition (dt='${hivevar:date1}');

alter table mytest drop if exists partition (dt='${hivevar:date2}')