



https://www.yiibai.com/hive/hive\_data\_types.html

https://www.cnblogs.com/biehongli/p/7699578.html

https://blog.csdn.net/u013980127/article/details/52604882

Hive只在一个节点上安装即可

1.上传tar包

2.解压

tar -zxvf hive-0.9.0.tar.gz -C /cloud/

3.配置mysql metastore（切换到root用户）

配置HIVE\_HOME环境变量

rpm -qa | grep mysql

rpm -e mysql-libs-5.1.66-2.el6\_3.i686 --nodeps

rpm -ivh MySQL-server-5.1.73-1.glibc23.i386.rpm

rpm -ivh MySQL-client-5.1.73-1.glibc23.i386.rpm

修改mysql的密码

/usr/bin/mysql\_secure\_installation

（注意：删除匿名用户，允许用户远程连接）

登陆mysql

mysql -u root -p

4.配置hive

cp hive-default.xml.template hive-site.xml

修改hive-site.xml（删除所有内容，只留一个<property></property>）

添加如下内容：

<property>

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

<value>jdbc:mysql://weekend01:3306/hive?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

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

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

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

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

<value>root</value>

<description>username to use against metastore database</description>

</property>

<property>

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

<value>root</value>

<description>password to use against metastore database</description>

</property>

5.安装hive和mysq完成后，将mysql的连接jar包拷贝到$HIVE\_HOME/lib目录下

如果出现没有权限的问题，在mysql授权(在安装mysql的机器上执行)

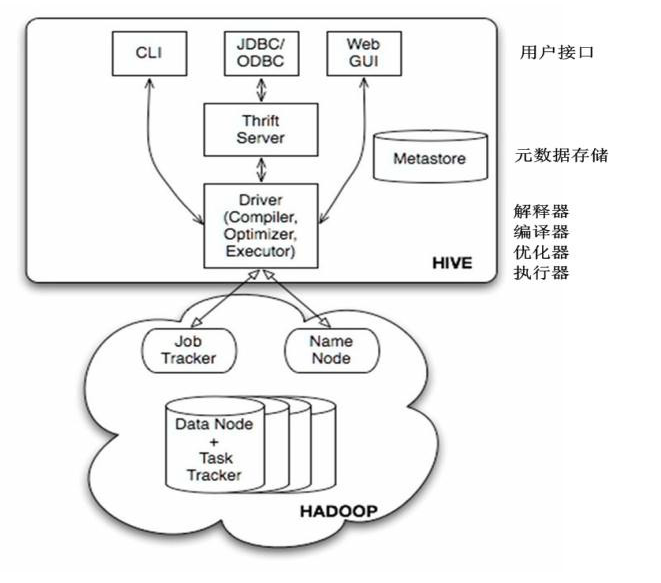
mysql -uroot -p

#(执行下面的语句 \*.\*:所有库下的所有表 %：任何IP地址或主机都可以连接)

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123' WITH GRANT OPTION;

FLUSH PRIVILEGES;

Hive是基于Hadoop的一个数据仓库，可以将结构化的数据文件映射为一张表，并提供类sql查询功能，Hive底层将sql语句转化为mapreduce任务运行。相对于用java代码编写mapreduce来说，Hive的优势明显：快速开发，人员成本低，可扩展性（自由扩展集群规模），延展性（支持自定义函数）。



原子数据类型：TINYINT SMALLINT INT BIGINT FLOAT DOUBLE BOOLEAN STRING

复杂数据类型：STRUCT MAP ARRAY

建表语句：

DDL：

创建内部表：

create table mytable(

id int,

name string)

row format delimited fields terminated by '\t' stored as textfile;

强制删除，移动到hdfs回收站。Drop database a\_db cascade；

常见外部表：关键字 external

create external table mytable2(

id int,

name string)

row format delimited

fields terminated by '\t'

location '/user/hive/warehouse/mytable2';

创建分区表：分区字段要写在partiton by（）

create table mytable3(

id int,

name string)

partitioned by(sex string) row format delimited fields terminated by '\t'stored as textfile;

静态分区插入数据

load data local inpath '/root/hivedata/boy.txt' overwrite into table mytable3 partition(sex='boy');

增加分区：

alter table mytable3 add partition (sex='unknown') location '/user/hive/warehouse/mytable3/sex=unknown';

删除分区：alter table mytable3 drop if exists partition(sex='unknown');

分区表默认为静态分区，可转换为自动套分区

set hive.exec.dynamic.partition=true;

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

给分区表灌入数据：

insert into table mytable3 partition (sex) select id,name,'boy' from student\_mdf;

查询表分区：show partitions mytable3;

查询分区表数据：select \* from mytable3;

查询表结构：desc mytable3;

DML:

重命名表：alter table student rename to student\_mdf

增加列：alter table student\_mdf add columns (sex string);

修改列名：alter table student\_mdf change sex gender string;

替换列结构：alter table student\_mdf replace columns (id string, name string);

装载数据：（本地数据）load data local inpath '/home/lym/zs.txt' overwrite into student\_mdf;

（HDFS数据）load data inpath '/zs.txt' into table student\_mdf;

插入一条数据：insert into table student\_mdf values('1','zhangsan');

创建表接收查询结果：create table mytable5 as select id, name from mytable3;

导出数据：（导出到本地）insert overwrite local directory '/root/hivedata/mytable5.txt' select \* from mytable5;

（导出到HDFS）

insert overwrite directory 'hdfs://master:9000/user/hive/warehouse/mytable5\_load' select \* from mytable5;

数据查询：

select \* from mytable3; 查询全表

select uid,uname from student; 查询学生表中的学生姓名与学号字段

select uname,count(\*) from student group by uname; 统计学生表中每个名字的个数

常用的功能还有 having、order by、sort by、distribute by、cluster by；等等

关联查询中有

内连接：将符合两边连接条件的数据查询出来

select \* from t\_a a inner join t\_b b on a.id=b.id;

左外连接：以左表数据为匹配标准，右边若匹配不上则数据显示null

select \* from t\_a a left join t\_b b on a.id=b.id;

右外连接：与左外连接相反

select \* from t\_a a right join t\_b b on a.id=b.id;

左半连接：左半连接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 \* from t\_a a left semi join t\_b b on a.id=b.id;

全连接(full outer join)：

select \* from t\_a a full join t\_b b on a.id=b.id;

in/exists关键字(1.2.1之后新特性)：效果等同于left semi join

select \* from t\_a a where a.id in (select id from t\_b);

select \* from t\_a a where exists (select \* from t\_b b where a.id = b.id);

shell操作Hive指令：

-e：从命令行执行指定的HQL

-f：执行HQL脚本

-v：输出执行的HQL语句到控制台

内置函数

查看内置函数：show functions;

显示函数的详细信息：DESC FUNCTION abs;

重要常用内置函数：

sum()--求和

count()--求数据量

avg()--求平均值

distinct--去重

min--求最小值

max--求最大值

自定义函数：

1.先开发一个简单的Java类，org.apache.hadoop.hive.ql.exec.UDF，重载evaluate方法

import org.apache.hadoop.hive.ql.exec.UDF;

public final class AddUdf extends UDF {

public Integer evaluate(Integer a, Integer b) {

if (null == a || null == b) {

return null;

} return a + b;

}

public Double evaluate(Double a, Double b) {

if (a == null || b == null)

return null;

return a + b;}

}

2.打成jar包上传到服务器

3.将jar包添加到hive add jar /home/lan/jar/addudf.jar;

4.创建临时函数与开发好的class关联起来

CREATE TEMPORARY FUNCTION add\_example AS 'org.day0914.AddUdf';

5.使用自定义函数 SELECT add\_example(scores.math, scores.art) FROM scores;

销毁临时函数：DROP TEMPORARY FUNCTION add\_example;

JDBC程序来创建数据库

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveCreateDb {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");

Statement stmt = con.createStatement();

stmt.executeQuery("CREATE DATABASE userdb");

System.out.println(“Database userdb created successfully.”);

con.close();

}

}

set hive.cli.print.header=true;

CREATE TABLE page\_view(viewTime INT, userid BIGINT,

page\_url STRING, referrer\_url STRING,

ip STRING COMMENT 'IP Address of the User')

COMMENT 'This is the page view table'

PARTITIONED BY(dt STRING, country STRING)

ROW FORMAT DELIMITED //按行分

FIELDS TERMINATED BY '\001' //以by后面的字符分块

STORED AS SEQUENCEFILE; TEXTFILE //存储格式

//sequencefile

create table tab\_ip\_seq(id int,name string,ip string,country string)

row format delimited

fields terminated by ','

stored as sequencefile;

insert overwrite table tab\_ip\_seq select \* from tab\_ext;

//create & load

create table tab\_ip(id int,name string,ip string,country string)

row format delimited

fields terminated by ','

stored as textfile;

load data local inpath '/home/hadoop/ip.txt' into table tab\_ext; //导入文件

//或者直接hadoop -put

//external

CREATE EXTERNAL TABLE tab\_ip\_ext(id int, name string,

ip STRING,

country STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/external/hive';

// CTAS 用于创建一些临时表存储中间结果

CREATE TABLE tab\_ip\_ctas

AS

SELECT id new\_id, name new\_name, ip new\_ip,country new\_country

FROM tab\_ip\_ext

SORT BY new\_id;

//insert from select 用于向临时表中追加中间结果数据

create table tab\_ip\_like like tab\_ip;

insert overwrite table tab\_ip\_like

select \* from tab\_ip;

//PARTITION

create table tab\_ip\_part(id int,name string,ip string,country string)

partitioned by (part\_flag string)

row format delimited fields terminated by ',';

load data local inpath '/home/hadoop/ip.txt' overwrite into table tab\_ip\_part

partition(part\_flag='part1');

load data local inpath '/home/hadoop/ip\_part2.txt' overwrite into table tab\_ip\_part

partition(part\_flag='part2');

select \* from tab\_ip\_part;

select \* from tab\_ip\_part where part\_flag='part2';

select count(\*) from tab\_ip\_part where part\_flag='part2';

alter table tab\_ip change id id\_alter string;

ALTER TABLE tab\_cts ADD PARTITION (partCol = 'dt') location '/external/hive/dt';

show partitions tab\_ip\_part;

//write to hdfs

insert overwrite local directory '/home/hadoop/hivetemp/test.txt' select \* from tab\_ip\_part where part\_flag='part1';

insert overwrite directory '/hiveout.txt' select \* from tab\_ip\_part where part\_flag='part1';

//array

create table tab\_array(a array<int>,b array<string>)

row format delimited

fields terminated by '\t'

collection items terminated by ',';

示例数据

tobenbrone,laihama,woshishui 13866987898,13287654321

abc,iloveyou,itcast 13866987898,13287654321

select a[0] from tab\_array;

select \* from tab\_array where array\_contains(b,'word');

insert into table tab\_array select array(0),array(name,ip) from tab\_ext t;

//map

create table tab\_map(name string,info map<string,string>)

row format delimited

fields terminated by '\t'

collection items terminated by ';'

map keys terminated by ':';

示例数据：

fengjie age:18;size:36A;addr:usa

furong age:28;size:39C;addr:beijing;weight:180KG

load data local inpath '/home/hadoop/hivetemp/tab\_map.txt' overwrite into table tab\_map;

insert into table tab\_map select name,map('name',name,'ip',ip) from tab\_ext;

//struct

create table tab\_struct(name string,info struct<age:int,tel:string,addr:string>)

row format delimited

fields terminated by '\t'

collection items terminated by ','

load data local inpath '/home/hadoop/hivetemp/tab\_st.txt' overwrite into table tab\_struct;

insert into table tab\_struct select name,named\_struct('age',id,'tel',name,'addr',country) from tab\_ext;

//cli shell

hive -S -e 'select country,count(\*) from tab\_ext' > /home/hadoop/hivetemp/e.txt

有了这种执行机制，就使得我们可以利用脚本语言（bash shell,python）进行hql语句的批量执行

select \* from tab\_ext sort by id desc limit 5;

select a.ip,b.book from tab\_ext a join tab\_ip\_book b on(a.name=b.name);

//UDF

select if(id=1,first,no-first),name from tab\_ext;

hive>add jar /home/hadoop/myudf.jar;

hive>CREATE TEMPORARY FUNCTION my\_lower AS 'org.dht.Lower';

select my\_upper(name) from tab\_ext;

//CLUSTER create table tab\_ip\_cluster(id int,name string,ip string,country string)

clustered by(id) into 3 buckets;

load data local inpath '/home/hadoop/ip.txt' overwrite into table tab\_ip\_cluster;

set hive.enforce.bucketing=true;

insert into table tab\_ip\_cluster select \* from tab\_ip;

select \* from tab\_ip\_cluster tablesample(bucket 2 out of 3 on id);

创建带桶的数据表

#设置变量,设置分桶为true, 设置reduce数量是分桶的数量个数

set hive.enforce.bucketing = true;

set mapreduce.job.reduces=4;

hive> create table if not exists tb\_stud(id int,name string,age int)

> partitioned by(clus string)

> clustered by(id) sorted by(age) into 2 buckets #分桶，根据id进行分桶，分成2个桶。

> row format delimited

> fields terminated by ',';

OK

Time taken: 0.194 seconds

hive> load data local inpath '/home/hadoop/data\_hadoop/tb\_clustered' overwrite into table tb\_stud partition (clus='20171211');

Loading data to table test.tb\_stud partition (clus=20171211)

Partition test.tb\_stud{clus=20171211} stats: [numFiles=1, numRows=0, totalSize=38, rawDataSize=0]

OK

Time taken: 0.594 seconds