# Hive使用和开发文档

## 一Hive使用文档

如果使用uddp的uhive请先看http://docs.ucloud.cn/uddp/index.html

### 1.1 Hive简介

　Hive是基于Hadoop的一个数据仓库工具，对 HDFS 海量数据库中的数据进行查询、统计、更新等操作，并提供完整的SQL查询功能，可以将SQL语句转换为MapReduce任务进行运行。

Hive主要特点如下：

* 通过HQL语言非常容易的完成海量数据数据分析
* 支持JSON，CSV，TEXTFILE， SEQUENCEFILE等多种存储格式
* 多种客户端连接方式，支持JDBC、Thrfit等接口，并且Hiveserver2提供了对客户端的并发支持。

目前UDDP提供的Hive版本是hive-0.12.0-cdh5.1.0，Hadoop的版本是hadoop-2.3.0-cdh5.1.0，我们下面以此版本为基础做介绍。

### 1.2 Hive基本架构

#### 1.2.1 Hive架构

Metastore

Hiveserver2

Hiveserver

DB

Beeline CLI

Hive CLI

JDBC/ODBC

Python/PHP

Nodejs ….

如上图所示，Hive架构主要包括如下组件：Hive CLI（command line interface）、Beeline CLI、 JDBC/ODBC/Python/PHP/Nodejs、HiveServer/Hiveserver2、Metastore，这些组件我可以分为两大类：客户端组件和服务端组件。

* **Metastore**：元数据服务组件，这个组件存储hive的元数据，hive的元数据存储在关系数据库里，hive支持的关系数据库有derby、mysql、PostgreSQL、Oracle。元数据对于hive十分重要，因此hive支持把metastore服务独立出来，安装到远程的服务器集群里，从而解耦hive服务和metastore服务，保证hive运行的健壮性。
* **Hiveserver/Hiveserver2**：Hiveserver2相对于Hiveserver提供了多客户端并发执行的支持，首先两者提供了Thrift server，用来支持JDBC/ODBC/PHP/Python等其他客户端的支持，其次二者提供**Driver组件**，该组件包括Complier、Optimizer和Executor，它的作用是将我们写的HiveQL（类SQL）语句进行解析、编译优化，生成执行计划，然后调用底层的mapreduce计算框架。
* **Hive CLI**：hive的命令行接口
* **Beeline CLI**:Hiveserver2支持的命令行接口
* **JDBC/ODBC/Python/PHP/Nodejs：其他客户端的支持**

#### 1.2.2 Hive数据模型

1.Hive数据库

类似传统数据库的DataBase，默认的dafault

简单示例命令行 hive > create database test\_database;

2.Hive内部表

    Hive的内部表与数据库中的Table在概念上是类似，**删除表时，元数据与数据都会被删除。**

内部表简单示例：  
    创建数据文件：test.txt

上传文件到：hdfs dfs -put test.txt /test/1

创建表：create table test (key string)  
    加载数据：LOAD DATA INPATH ‘/test/1/test.txt’ INTO TABLE test

查看数据：select \* from test;

select count(\*) from test   
    删除表：drop table test

3.Hive**外部表**

外部表指向已经在HDFS中存在的数据，可以创建Partition。它和内部表在元数据的组织上是相同的，而实际数据的存储则有较大的差异。内部表的创建过程和数据加载过程这两个过程可以分别独立完成，也可以在同一个语句中完成，在加载数据的过程中，实际数据会被移动到数据仓库目录中；之后对数据对访问将会直接在数据仓库目录中完成。删除表时，表中的数据和元数据将会被同时删除。而外部表只有一个过程，加载数据和创建表同时完成（CREATE EXTERNAL TABLE ……LOCATION），实际数据是存储在LOCATION后面指定的 HDFS 路径中，并不会移动到数据仓库目录中。**当删除一个External Table时，仅删除该链接。**

外部表简单示例：

创建数据文件：hehe.txt

上传到hdfs：hdfs dfs -put hehe.txt /test/1

创建表：create external table hehe (key string)  
    加载数据：LOAD DATA INPATH ‘/test/1/hehe.txt’ INTO TABLE hehe

查看数据：select \* from hehe;

select count(\*) from hehe;  
    删除表：drop table hehe

4.Hive**分区**

   Hive根据分区列（如日期）的值对表进行粗略划分，把表组织成分区及子分区，以此加快数据分片的查询速度。**在文件系统级别，分区只是表目录下嵌套的子目录**

分区表简单示例：

创建分区表：create table table\_name (key string) partitioned by (dt string)  
     删除分区表：drop table table\_name

增加一个分区，分区字段是partition\_date:

alter table table\_name add partition(partition\_date = 20150102);

删除一个分区:

alter table table\_name drop partition(partition\_date = 20150102);

5.Hive**桶**

   表或分区可以进一步分为桶，它会为数据提供额外的结构以获得更高效的查询处理。物理上，每个桶就是表/分区目录里的一个文件。在map端连接时，若两个表以相同方式划分桶且桶个数是倍数关系，则处理左表某个桶的mapper知道右表相匹配的行在哪个桶内，可以提高连接效率；另外，当桶中数据根据一个或多个列排序时，桶的连接变成了高效的合并排序。

创建桶的语法：

    create table test\_bucket\_table (key string) clustered by (key) into 20 buckets( Hive根据对id取哈希值再mod桶数的结果，把记录随机分散到不同的桶内）

**（6）Hive的视图**

视图与传统数据库的视图类似。视图是只读的，它基于的基本表，如果改变，数据增加不会影响视图的呈现；如果删除，会出现问题。如果不指定视图的列，会根据select语句后的生成。

    示例：create view test\_view as select \* from test

备注：

1. Hive DDL的详细操作请参考官方文档：<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>

2.Hive DML的详细操作请参考官方文档：

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML

### 1.3 Hive客户端环境配置（tar包安装，其他方式的安装请参考uddp文档）

#### 1.3.1 JDK的安装配置（1.7以及以上版本）

首先进行JDK安装（以jdk-7u55为例）。

1.下载JDK安装包(jdk-7u55-linux-x64.tar.gz),创建JDK安装目录，并将安装包解压至该目录：

mkdir –p /usr/java

2.建立软链接：

ln -s /usr/java/jdk1.7.0\_55 /usr/java/java

3.配置环境变量。在/etc/profile文件结尾添加：

export JAVA\_HOME=/usr/java/java

export JRE\_HOME=**${**JAVA\_HOME**}**/jre

export CLASSPATH=.:**${**JAVA\_HOME**}**/lib:**${**JRE\_HOME**}**/lib

export PATH=**${**JAVA\_HOME**}**/bin:$PATH

4.然后，下载jdk1.7版本对应的jce(UnlimitedJCEPolicyJDK7.zip)，将解压后的jar包copy到/usr/java/java/jre/lib/security：

yum install unzip -y

unzip UnlimitedJCEPolicyJDK7.zip

cp jce/\*.jar /usr/java/jdk1.7.0\_55/jre/lib/security

#### 1.3.2 Hadoop客户端的安装配置

1.下载对应的hadoop版本

wget <http://archive.cloudera.com/cdh5/cdh/5/hadoop-2.3.0-cdh5.1.0.tar.gz>

2.解压

tar –zxvf hadoop-2.3.0-cdh5.1.0.tar.gz

3.配置hadoop客户端

在管理控制台MapReduce管理页面下载客户端配置文件包，解压后将以下四个配置文件复制到/root/hadoop-2.0.0-cdh5.1.0/etc/hadoop 目录下：

core-site.xml

hadoop-env.sh

hdfs-site.xml

mapred-site.xml

**注解：**

1.把配置文件hadoop-env.sh中的HADOOP\_MAPRED\_HOME，修改为mapreduce1的安装目录。如：

export HADOOP\_MAPRED\_HOME=/root/hadoop-2.3.0-cdh5.1.0/share/hadoop/mapreduce1

2. 把配置文件hadoop-env.sh中JAVA\_HOME修改为jdk的安装目录

export JAVA\_HOME=/usr/java/java

3.添加native包

需要把编译好的native包添加到HADOOP\_HOME/lib目录下

4.处理$HADOOP\_HOME/bin目录

1) cp $HADOOP\_HOME/bin/hdfs $HADOOP\_HOME/bin-mapreduce1

2)mv $HADOOP\_HOME/bin $HADOOP\_HOME/bin.bak

3)mv $HADOOP\_HOME/bin-mapreduce1 $HADOOP\_HOME/bin

#### 1.3.3 Hive客户端的安装配置

1.下载对应的hive版本

wget <http://archive.cloudera.com/cdh5/cdh/5/hive-0.12.0-cdh5.1.0.tar.gz>

2.解压

tar –zxvf hive-0.12.0-cdh5.1.0

3.配置hive客户端

1）配置文件在hive解压目录hive-0.12.0-cdh5.1.0/conf下面

在hive-env.sh 文件中添加JAVA\_HOME和HADOOP\_MAPRED\_HOME

2）新增hive-site.xml，在文件中添加配置

<property>

<name>hive.metastore.uris</name>

<value>thrift://IP:9083</value>

</property>

<property>

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

<value>/hive/MapReduceID/HiveID/hive-${user.name}</value>

</property>

注解 请将IP替换成自己Hive实例的ip，MapReduceID修改为自己的MapReduce实例id(如：umr-frees5),HiveID修改为自己hive对应的Principal id(如：uhivecqxseb)。

3）

修改HADOOP\_HOME/etc/hadoop/core-site.xml文件,把下面配置注销或者删除掉。

<!--

<property>

<name>io.compression.codecs</name> <value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec,com.hadoop.compression.lzo.LzoCodec,com.hadoop.compression.lzo.LzopCodec,org.apache.hadoop.io.compress.SnappyCodec</value>

</property>

-->

4）修改/etc/hosts

如果你的hosts文件中有127.0.0.1 10-10-50-79请删除掉，10-10-50-79是您的主机名称

#### 1.3.4 kerberos的客户端的安装配置

请参考我们的uddp帮助文档http://docs.ucloud.cn/uddp/index.html

#### 1.3.5 添加环境变量到.bashrc

export HADOOP\_HOME=/root/hadoop-2.3.0-cdh5.1.0

export HIVE\_HOME=/root/hive-0.12.0-cdh5.1.0

export HADOOP\_MAPRED\_HOME=/root/hadoop-2.3.0-cdh5.1.0/share/hadoop/mapreduce1

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

备注：

1.需要把上面HADOOP\_HOME和HIVE\_HOME路径改为自己hadoop和hive的安装路径

2.更新 .bashrc

source .bashrc

## 二Hive开发文档

### 2.1 hive命令行使用简介

请参考官方文档：

<https://cwiki.apache.org/confluence/display/Hive/HiveClient>

### 2.2 beeline命令行使用简介

请参考官方文档：

<https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients>

### 2.3 JDBC客户端

Hive JDBC接口遵循标准的JAVA JDBC驱动标准，提供了JAVA程序访问hive的方法。

#### 2.3.1 JDBC访问Hiveserver

1. 示例代码HiveJdbcClient.java

org.apache.hadoop.hive.jdbc.HiveDriver是hiveserver的dirvername，hiveserver的访问地址是jdbc:hive://Ip:10000/default，示例代码如下：

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveJdbcClient {

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

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

try {

Class.forName(driverName);

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

System.exit(1);

}

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

Statement stmt = con.createStatement();

String tableName = "testHiveDriverTable";

stmt.executeQuery("drop table " + tableName);

ResultSet res = stmt.executeQuery("create table " + tableName + " (key int, value string)");

// show tables

String sql = "show tables '" + tableName + "'";

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

if (res.next()) {

System.out.println(res.getString(1));

}

// describe table

sql = "describe " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(res.getString(1) + "\t" + res.getString(2));

}

// load data into table

// NOTE: filepath has to be local to the hive server

// NOTE: /tmp/a.txt is a ctrl-A separated file with two fields per line

String filepath = "/umr-u4x3iv/uhivecqxseb/warehouse/a.txt";

sql = "load data inpath '" + filepath + "' into table " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

// select \* query

sql = "select \* from " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(String.valueOf(res.getInt(1)) + "\t" + res.getString(2));

}

// regular hive query

sql = "select count(1) from " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(res.getString(1));

}

}

}

2编译 javac HiveJdbcClient.java

3.运行脚本test.sh

#!/bin/bash

HADOOP\_MAPREDUCE\_HOME=/data/hadoop-2.3.0-cdh5.1.0/share/hadoop/mapreduce1

HIVE\_HOME=/data/hive-0.12.0-cdh5.1.0

echo -e '1\x01foo' > /tmp/a.txt

echo -e '2\x01bar' >> /tmp/a.txt

hdfs dfs -put /tmp/a.txt /umr-u4x3iv/uhivecqxseb/warehouse/

CLASSPATH=.:$HIVE\_HOME/conf

for i in ${HADOOP\_MAPREDUCE\_HOME}/lib/hadoop-\*.jar ; do

CLASSPATH=$CLASSPATH:$i

done

for i in ${HIVE\_HOME}/lib/\*.jar ; do

CLASSPATH=$CLASSPATH:$i

done

java -cp $CLASSPATH HiveJdbcClient

备注：1. Connection con = DriverManager.getConnection("jdbc:hive://ip:10000/default", "", ""); ip修改为自己hiveserver的ip

2. /umr-u4x3iv/uhivecqxseb/warehouse/修改为自己hdfs存储目录

3. HADOOP\_MAPREDUCE\_HOME、HIVE\_HOME要修改为对应自己的hadoop、hive安装目录

#### 2.3.2 JDBC访问Hiveserver2

1. 示例代码Hive2JdbcClient.java

org.apache.hive.jdbc.HiveDriver是hiveserver2的dirvername，hiveserver2的访问地址是"jdbc:hive2://ip:10001/default，示例代码如下：

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class Hive2JdbcClient {

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

/\*\*

\* @param args

\* @throws SQLException

\*/

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

try {

Class.forName(driverName);

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

System.exit(1);

}

//replace "hive" here with the name of the user the queries should run as

Connection con = DriverManager.getConnection("jdbc:hive2://ip:10001/default", "", "");

Statement stmt = con.createStatement();

String tableName = "testHive2DriverTable";

stmt.execute("drop table if exists " + tableName);

stmt.execute("create table " + tableName + " (key int, value string)");

// show tables

String sql = "show tables '" + tableName + "'";

System.out.println("Running: " + sql);

ResultSet res = stmt.executeQuery(sql);

if (res.next()) {

System.out.println(res.getString(1));

}

// describe table

sql = "describe " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(res.getString(1) + "\t" + res.getString(2));

}

// load data into table

// NOTE: filepath has to be local to the hive server

// NOTE: /tmp/a.txt is a ctrl-A separated file with two fields per line

String filepath = "/umr-u4x3iv/uhivecqxseb/warehouse/b.txt";

sql = "load data inpath '" + filepath + "' into table " + tableName;

System.out.println("Running: " + sql);

stmt.execute(sql);

// select \* query

sql = "select \* from " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(String.valueOf(res.getInt(1)) + "\t" + res.getString(2));

}

// regular hive query

sql = "select count(1) from " + tableName;

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

while (res.next()) {

System.out.println(res.getString(1));

}

}

}

2编译 javac Hive2JdbcClient.java

3.运行脚本test.sh

#!/bin/bash

HADOOP\_MAPREDUCE\_HOME=/data/hadoop-2.3.0-cdh5.1.0/share/hadoop/mapreduce1

HIVE\_HOME=/data/hive-0.12.0-cdh5.1.0

echo -e '1\x01foo' > /tmp/b.txt

echo -e '2\x01bar' >> /tmp/b.txt

hdfs dfs -put /tmp/b.txt /umr-u4x3iv/uhivecqxseb/warehouse/

CLASSPATH=.:$HIVE\_HOME/conf

for i in ${HADOOP\_MAPREDUCE\_HOME}/lib/hadoop-\*.jar ; do

CLASSPATH=$CLASSPATH:$i

done

for i in ${HIVE\_HOME}/lib/\*.jar ; do

CLASSPATH=$CLASSPATH:$i

done

java -cp $CLASSPATH Hive2JdbcClient

备注：1. Connection con = DriverManager.getConnection("jdbc:hive2://ip:10001/default", "", "");

ip修改为自己hiveserver的ip

2. /umr-u4x3iv/uhivecqxseb/warehouse/修改为自己hdfs存储目录

3. HADOOP\_MAPREDUCE\_HOME、HIVE\_HOME要修改为对应自己的hadoop、hive安装目

### 2.4 python客户端

#### 2.4.1 Hiveserver的python客户端

示例代码如下：

#!/usr/bin/env python

import sys

sys.path.append("/data/hive-0.12.0-cdh5.1.0/lib/py")

from hive\_service import ThriftHive

from hive\_service.ttypes import HiveServerException

from thrift import Thrift

from thrift.transport import TSocket

from thrift.transport import TTransport

from thrift.protocol import TBinaryProtocol

try:

transport = TSocket.TSocket('ip', 10000)

transport = TTransport.TBufferedTransport(transport)

protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = ThriftHive.Client(protocol)

transport.open()

client.execute("SELECT \* FROM hehe")

while (1):

row = client.fetchOne()

if (row == None):

break

print row

client.execute("SELECT \* FROM hehe")

print client.fetchAll()

transport.close()

except Thrift.TException, tx:

print '%s' % (tx.message)

备注：1.需要添加到hive py目录到python系统路径下：

sys.path.append("/data/hive-0.12.0-cdh5.1.0/lib/py")

2 transport = TSocket.TSocket('ip', 10000)修改为hivesever的ip

3. client.execute("SELECT \* FROM hehe") 这里的hehe表格需要先创建好

#### 2.4.2 Hiveserver2的python客户端

支持Hiveserver2 Python客户端的地址在

<https://github.com/BradRuderman/pyhs2> 包含所有依赖的包，这个driver需要Python2.6或者以上版本。Hiveserver2使用python客户端的过程如下：

1.下载pyhs2

git clone <https://github.com/BradRuderman/pyhs2.git>

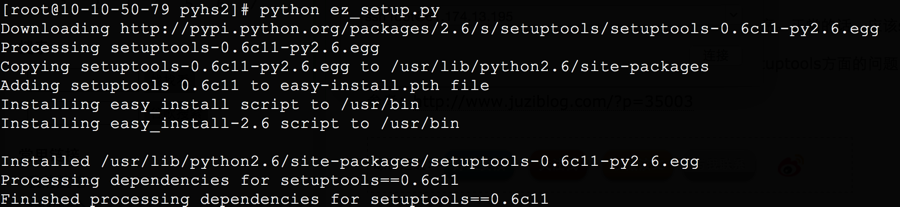
2.安装依赖

1)yum install gcc-c++ cyrus-sasl-\* python-devel

2)安装setuptools

wget -q [http://peak.telecommunity.com/dist/ez\_setup.py ./](http://peak.telecommunity.com/dist/ez_setup.py%20./)

python ez\_setup.py



如果上面方式安装失败需要手动下载setuptools-0.6c11.tar.gz安装包安装。

3编译安装pyhs2

进入pyhs2目录

python setup.py bulid

python setup.py install

4.示例代码：即pyhs2下example.py ，下面的host后面的localhost要改为自己hiveserver的ip端口号port由10000改为10001， 下面执行语句中的table修改为hive中已经有的table名称cur.execute("select \* from table")

import pyhs2

with pyhs2.connect(host='localhost',

port=10001,

authMechanism="PLAIN",

user='root',

password='123',

database='default') as conn:

with conn.cursor() as cur:

#Show databases

print cur.getDatabases()

#Execute query

cur.execute("select \* from table")

#Return column info from query

print cur.getSchema()

#Fetch table results

for i in cur.fetch():

print i

### 2.5 其他客户端的使用

HiveServer的其他客户端使用请参考：

https://cwiki.apache.org/confluence/display/Hive/HiveClient#HiveClient-ThriftJavaClient

HiveServer2的其他客户端使用请参考：

https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-RubyClient

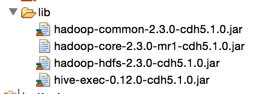
### 2.6常用增强特性

#### 2.6.1 支持用户自定义函数 (UDF)

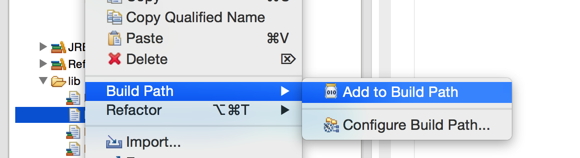
**下面简介eclipse编写自定义hive udf函数**

1  在eclipse中新建java project   hiveudf

2 在hiveudf工作中添加一个文件夹lib，lib包需要拷贝如下四个jar包，



3.把lib目录下的所有jar包 Add to build Path



4. 创建一个UDFLower 类并且继承hive的UDF类

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

import org.apache.hadoop.io.Text;

public class UDFLower extends UDF{

    public Text evaluate(final Text s){

        if (null == s){

            return null;

        }

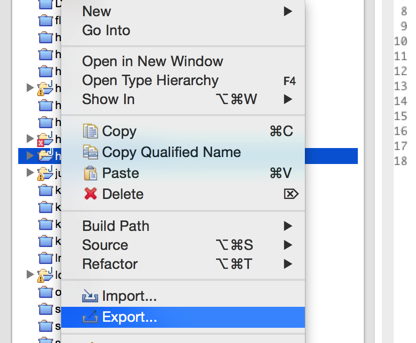
        return new Text(s.toString().toLowerCase());

    }

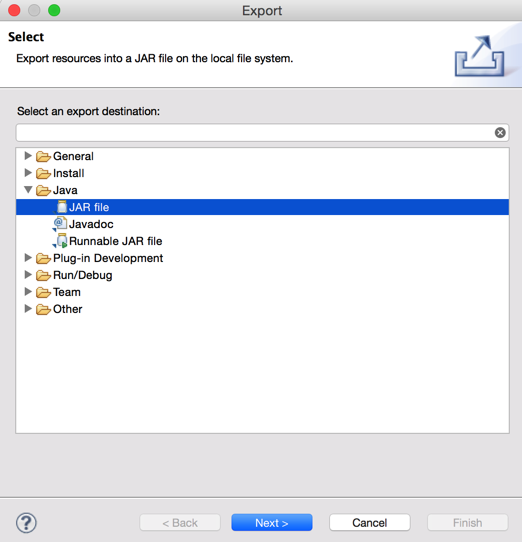
}

5  编译输出打包文件为 testhiveudf.jar

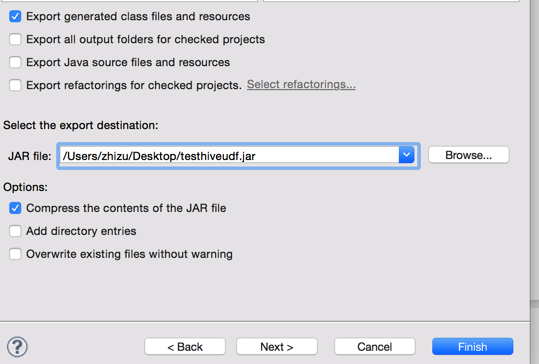
1）



2）



3)点击next，把jar包命名



4）点击finish即可把jar包导出到指定的位置

6先 将testhiveudf.jar上传linux机器（hadoop客户端所在机器），然后再上传到hdfs目录。

hdfs dfs –put testhiveudf.jar /test/jar

7 打开hive命令行测试（或者beeline客户端）,把自定义jar add 到path

   hive> add jar hdfs://UCloudcluster/test/jar/testhiveudf.jar;

8.创建udf函数

hive> create temporary function my\_lower as 'UDFLower';

  // UDFLower'表示你的类名称，如果有包名需要加上包名报名.类名。

9.创建测试数据

hive> create table hehe (name string);

导入数据文件test.txt

test.txt文件内容为

HELLO

WORLD

HEHE

1）上传数据到hdfs

hdfs dfs -put test.txt /test/1

2)load数据到数据库表中

hive> load data inpath '/test/1/test.txt' into table hehe;

10.测试数据

hive> select name from hehe;

hive>select my\_lower(name) from hehe;

#### 2.6.2 支持json格式数据

hive解析json格式的数据是一个常用操作，我们已经把hive依赖的json jar包**hive-json-serde-0.2.jar放在服务端的系统路径下，如果通过beeline无需在通过add jar加载对应的jar包。如果你使用的hive命令行接口需要add jar，需要做如下操作：**

hive> add jar /data/hive-json-serde-0.2.jar;

具体操作过程如下：

1.创建hive表格

如果使用hive命令行接口，在操作前需要 add jar

CREATE TABLE test\_json(id BIGINT,text STRING)ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.JsonSerde' STORED AS TEXTFILE

2.上传数据json格式的数据到hdfs

test.json

{"id":1234544353,"text":"efds"}

hdfs dfs -put test.json /umr-u4x3iv/hive

3.加载数据

LOAD DATA INPATH "/umr-u4x3iv/hive/test.json" OVERWRITE INTO TABLE test\_json;

4.执行查询：返回如下说明使用json文件解析成功



#### 2.6.3 正则匹配使用

如果使用hive命令接口需要add hive-contrib-0.12.0-cdh5.1.0.jar jar包，并且把路径修改为自己的路径

hive> add jar /data/hive-0.12.0-cdh5.1.0/lib/hive-contrib-0.12.0-cdh5.1.0.jar;

如果使用beeline直接进行下面测试操作：

1.测试数据nginx\_log

180.173.250.74 168.34.34

2.测试表格

CREATE TABLE logs(

         host STRING,

         identity STRING)

           ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'

            WITH SERDEPROPERTIES (

            "input.regex" = "([^ ]\*) ([^ ]\*)",

             "output.format.string" = "%1$s %2$s"

  )STORED AS TEXTFILE;

备注：创建表格的时候需要指定ROW FORMAT SERDE，SERDEPROPERTIES中的 input.regex和output.format.string

3.加载数据

LOAD DATA INPATH "/umr-testtest/hive/nginx\_log" INTO TABLE logs;

4.测试数据

select \* from logs;

select \* from logs group by host;

### 2.8 sqoop导入导出数据

sqoop导入导出mysql中的数据，确保mysql中的数据可以被远程用户访问。要不会包权限错误。详细使用请参考官方文档：

http://sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html

#### 2.8.1 sqoop的安装配置（tar包安装）

1.下载sqoop

wget http://archive.cloudera.com/cdh5/cdh/5/[sqoop-1.4.4-cdh5.1.0.tar.gz](http://archive.cloudera.com/cdh5/cdh/5/sqoop-1.4.4-cdh5.1.0.tar.gz)

2.解压

tar -zxvf sqoop-1.4.4-cdh5.1.0.tar.gz

3.加入环境变量中去vim ~/.bashrc

export SQOOP\_HOME=/data/sqoop-1.4.4-cdh5.1.0

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

4.配置sqoop环境变量（SQOOP\_HOME是sqoop的安装目录）

1) cp $SQOOP\_HOME/conf/sqoop-env-template.sh $SQOOP\_HOME/conf/sqoop-env.sh

2) 在sqoop-env.sh修改配置

#Set path to where bin/hadoop is available

export HADOOP\_COMMON\_HOME=/data/hadoop-2.3.0-cdh5.1.0

#Set path to where hadoop-\*-core.jar is available

export HADOOP\_MAPRED\_HOME=/data/hadoop-2.3.0-cdh5.1.0/share/hadoop/mapreduce1

#Set the path to where bin/hive is available

export HIVE\_HOME=/data/hive-0.12.0-cdh5.1.0

5.下载mysql驱动并且拷贝SQOOP\_HOME/lib目录下

mysql-connector-java-5.1.35-bin.jar

4.修改/etc/hosts

把/etc/hosts文件中的127.0.0.1 hostname这个配置删除掉

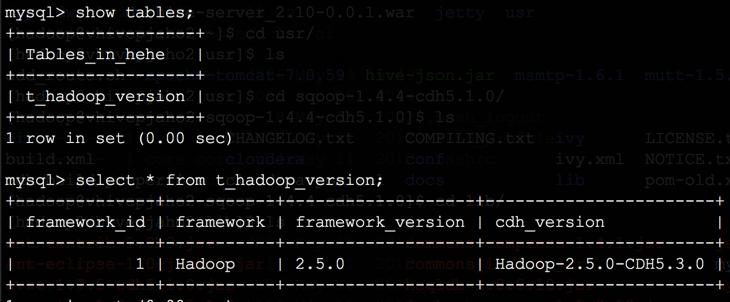
#### 2.8.2 sqoop从mysql导入数据到hdfs

1.mysql信息如下：

以MySQL 为例：

* 10.10.50.79
* database: hehe
* 用户：test 密码：test

表格信息如下：



2.执行语句

sqoop import --connect jdbc:mysql://10.10.50.79/hehe --username test --password test --table t\_hadoop\_version --target-dir /umr-u4x3iv/sqoop-import-hdfs

备注1)10.10.50.79是mysql的ip

2)--username 访问数据的用户名称

3)--password 访问数据的密码

4)--table t\_hadoop\_version数据表名称

5) --target-dir是数据中数据导入到hdfs中目录

3.查看结果



#### 2.8.3 sqoop从mysql导入数据hive

1.mysql信息如**2.8.2**

**2.执行语句**

sqoop import --connect jdbc:mysql://10.10.50.79/hehe --username test --password test --table t\_hadoop\_version --warehouse-dir /umr-u4x3iv/uhivecqxseb/warehouse --hive-import --create-hive-table

备注1)10.10.50.79是mysql的ip

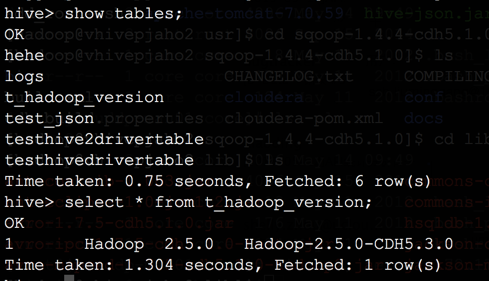
2)--username 访问数据的用户名称

3)--password 访问数据的密码

4)--table t\_hadoop\_version数据表名称

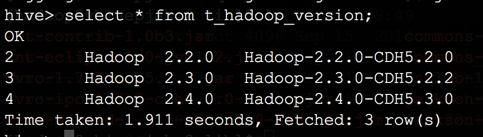
5) --warehouse-dir是hive数据创库目录

3.查看结果，如何所示mysql中的表格t\_hadoop\_version已经导入hive中



#### 2.8.4 sqoop从hive导出数据mysql

1.hive表格中数据如下所示：



2.执行语句

sqoop export --connect jdbc:mysql://10.10.50.79/hehe --username test --password test --table t\_hadoop\_version --export-dir /umr-u4x3iv/uhivecqxseb/warehouse/t\_hadoop\_version --fields-terminated-by "\t"

备注1)10.10.50.79是mysql的ip

2)--username 访问数据的用户名称

3)--password 访问数据的密码

4)--table t\_hadoop\_version数据表名称

5) --export-dir是要导出的数据目录

6）--fields-terminated-by数据分割方式

3.查看结果,之前mysql数据库中的结果+hive数据创库的结果，显示如下所示：



sqoop的详细使用请参考官方文档：

http://sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html