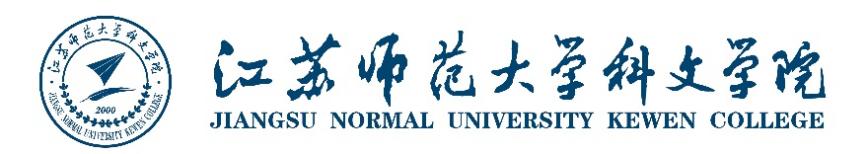
****

**人工智能与软件学院**

实 验 报 告

**课 程 大数据分析与智能决策**

**学 号 7200767121**

**姓 名 彭敏捷**

**班 级 20智科**

**教 师 胡元发**

2023年 4 月至 2023 年 4 月

**一、实验名称**

Hive与HBase的安装及操作

**二、实验目的和要求**

1、Hive安装及建表，查询，导入数据等操作；

2、安装Hbase并完成配置；

3、用Hadoop提供的HBase Shell命令完成数据管理任务：

4、使用Hbase提供的Java API 完成一系列指定编程任务，实现应用程序对数据的管理及存取。

**三、实验环境**

1.VMware Workstation Pro

2.Xshell

1. **实验步骤**
2. Hive安装及建表，查询，导入数据等操作

1.1查看mysql是否安装，如果安装了，卸载mysql

查看 rpm -qa|grep mysql

存在mysql-libs-5.1.73-7.el6.x86\_64

卸载 rpm -e --nodeps mysql-libs-5.1.73-7.el6.x86\_64

1.2安装MySQL

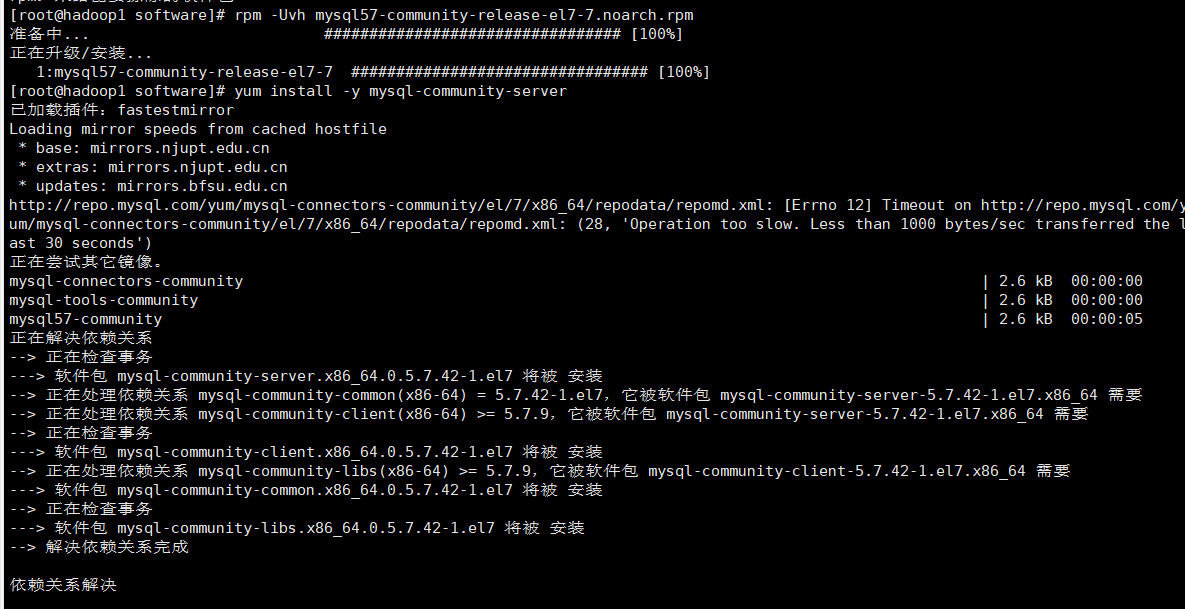
(1)下载MySQL的yum源：

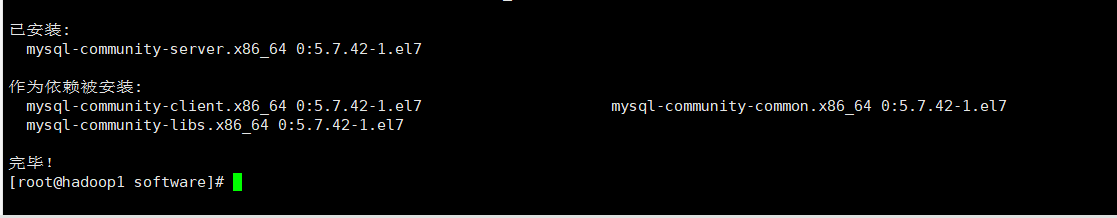
wget <http://dev.mysql.com/get/mysql57-community-release-el7-7.noarch.rpm>



(2)查看下载源中包含的rpm包：

rpm -Uvh mysql57-community-release-el7-7.noarch.rpm

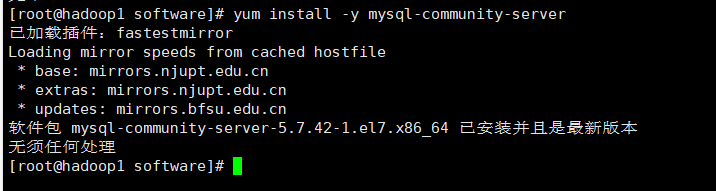


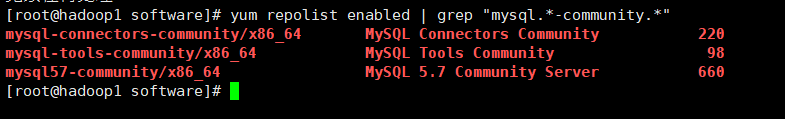


(3)安装MySQL，查看是否安装成功

yum install -y mysql-community-server

yum repolist enabled | grep "mysql.\*-community.\*"

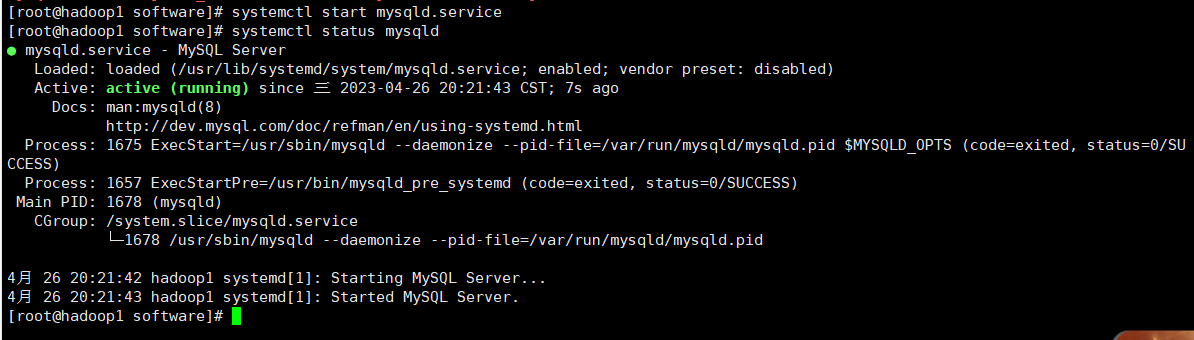




1. 启动mysql服务，查看mysql状态

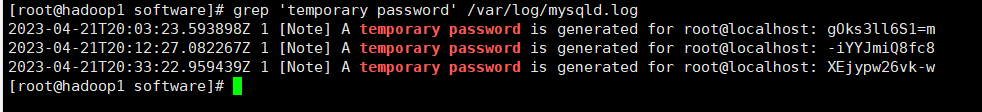
systemctl start mysqld.service

systemctl status mysqld



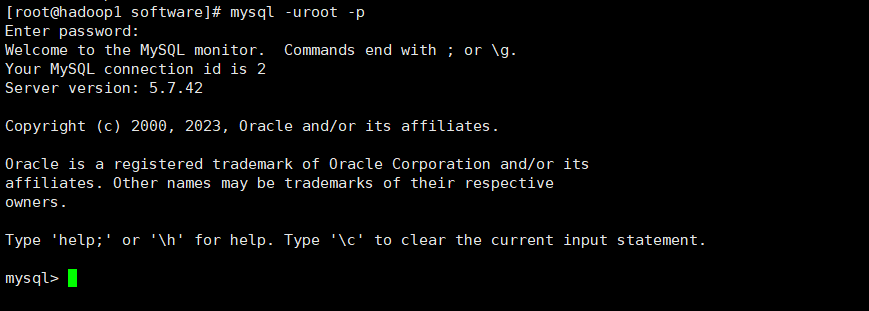
(5)查找MySQL初始密码

grep 'temporary password' /var/log/mysqld.log



(6)登录MySQL:

mysql -uroot -p



(7)设置安全级别

set global validate\_password\_policy=0;

(8)设置密码长度

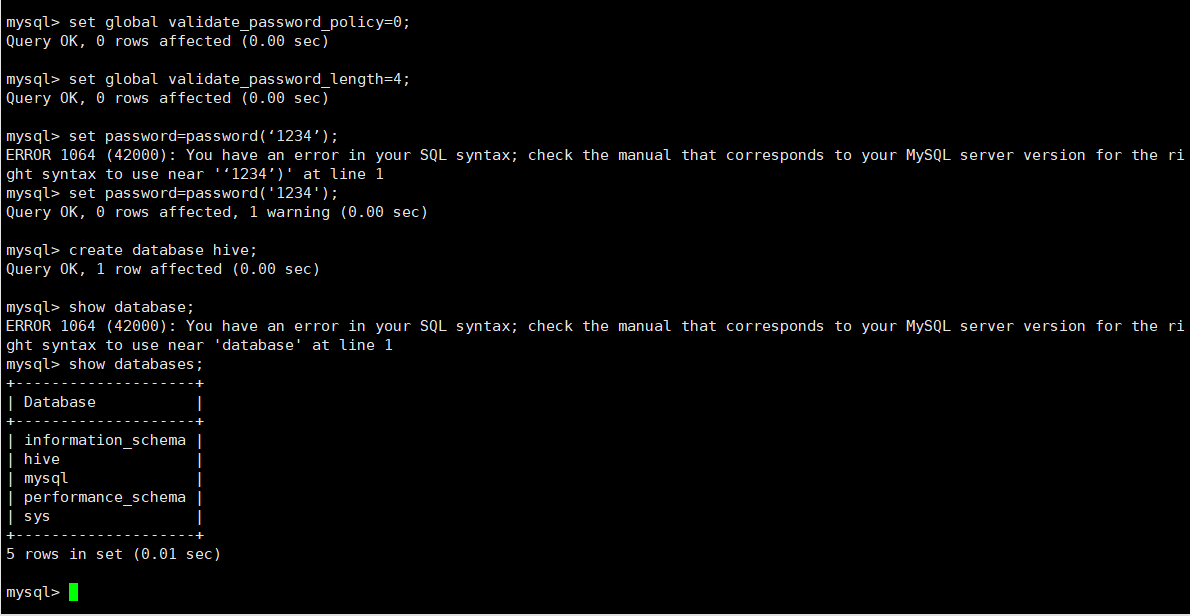
set global validate\_password\_length=4;

(9)设置密码

set password=password(‘1234’);

(10)创建数据库

create database hive;



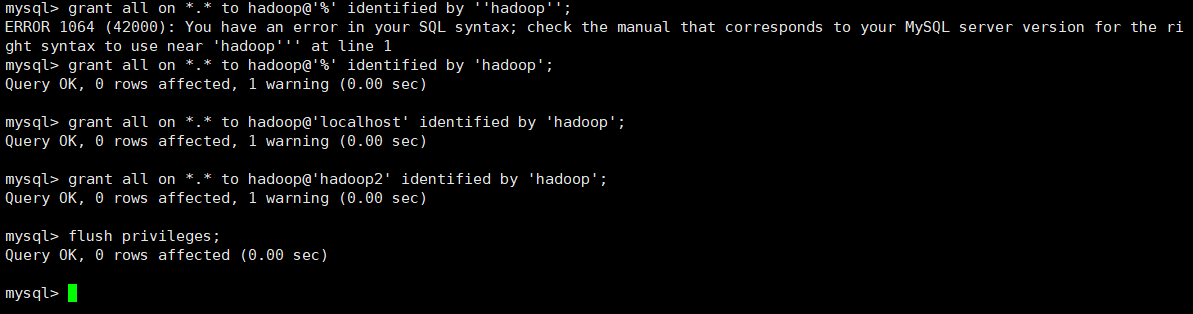
(11)创建MySQL用户，名称为hadoop，设置用户密码

grant all on \*.\* to hadoop@'%' identified by 'hadoop';

grant all on \*.\* to hadoop@'localhost' identified by 'hadoop';

grant all on \*.\* to hadoop@'hadoop2' identified by 'hadoop';

flush privileges;



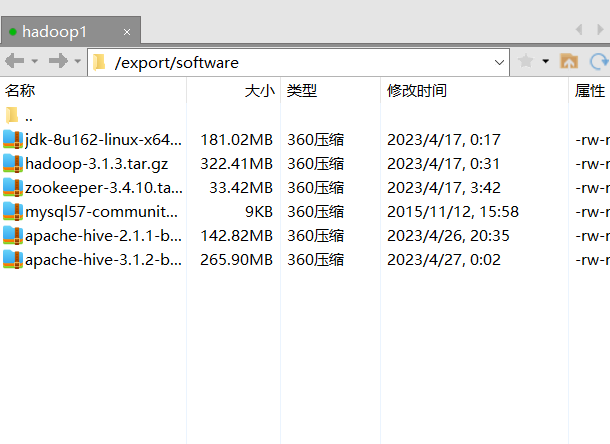
1.3安装Hive

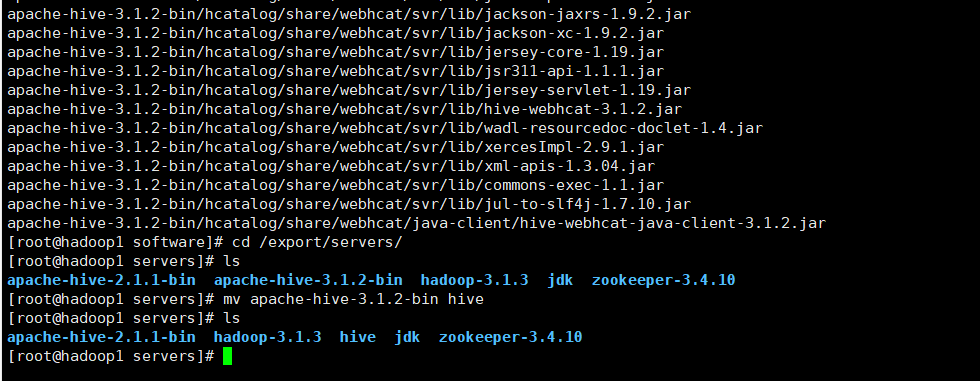
(1)下载安装包，将安装包上传到服务器，并进行解压，修改为hive。（hive和hadoop版本必须同为2.x或同为3.x）

cd /export/software

tar -zxvf apache-hive-3.1.2-bin.tar.gz -C /export/servers

mv apache-hive-3.1.2-bin hive



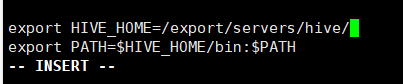


(2)配置Hive环境变量

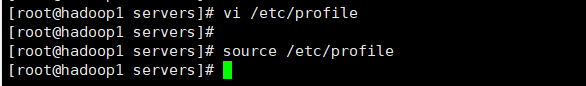
vi /etc/profile

export HIVE\_HOME=/export/servers/hive/

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



配置完成并保存后，刷新配置文件，source /etc/profile



(3)在Hadoop下创建文件夹hive

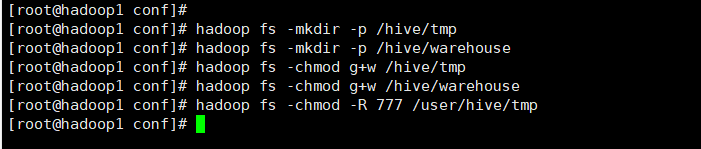
hadoop fs -mkdir -p /hive/tmp

hadoop fs -mkdir -p /hive/warehouse

hadoop fs -chmod g+w /hive/tmp

hadoop fs -chmod g+w /hive/warehouse

hadoop fs -chmod -R 777 /user/hive/tmp



(4)修改Hive配置文件，修改配置文件

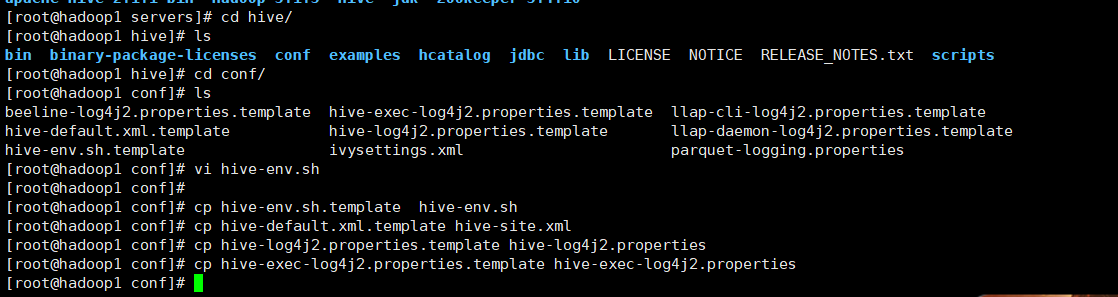
cd /export/servers/hive/conf/

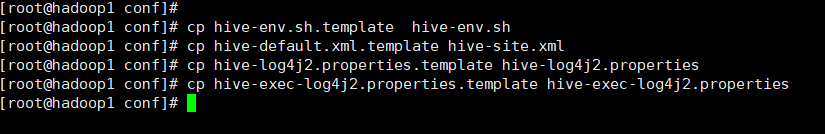
cp hive-env.sh.template hive-env.sh

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

cp hive-log4j2.properties.template hive-log4j2.properties

cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties

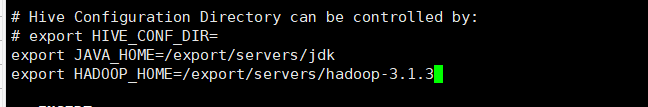




vi hive-env.sh

export JAVA\_HOME=/export/servers/jdk

export HADOOP\_HOME=/export/servers/hadoop-3.1.3



添加hive-site.xml配置文件，配置MySQL相关信息

<configuration>

<property>

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

<value>/hive/tmp</value>

</property>

<property>

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

<value>/hive/warehouse</value>

</property>

<property>

<name>hive.querylog.location</name>

<value>/hive/log</value>

</property>

<!-- 配置MySQL数据库连接信息-->

<property>

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

<value>jdbc:mysql://localhost:3306/metastore?createDatabaseIfNotExist=true&amp;characterEncoding=UTF-8&amp;useSSL=false</value>

</property>

<property>

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

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

</property>

<property>

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

<value>hadoop</value>

</property>

<property>

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

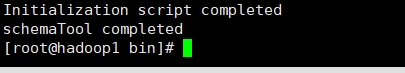
<value>hadoop</value>

</property>

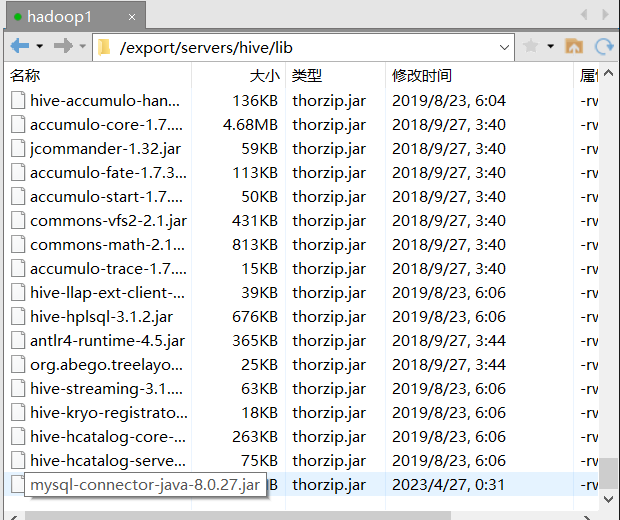
</configuration>

(5)初始化Hive

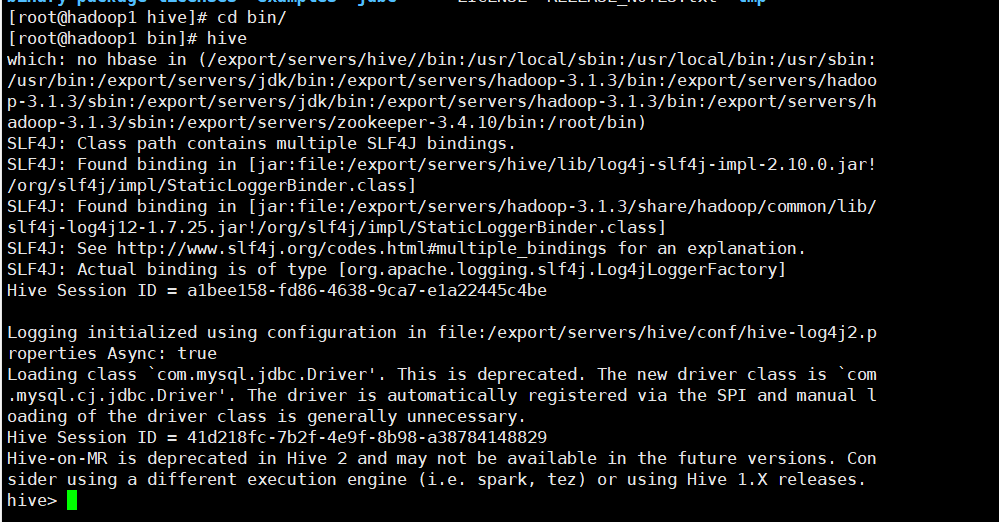
./schematool -dbType mysql -initSchema



(6)下载mysql-connector-java-5.1.46.jar,并复制到Hive安装目录的lib目录下。



(7)启动Hive

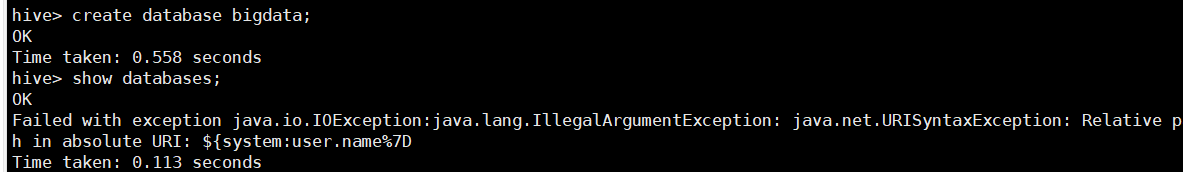


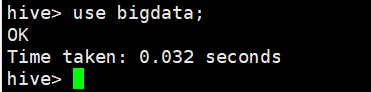
1.4hive建表查询导入数据等操作

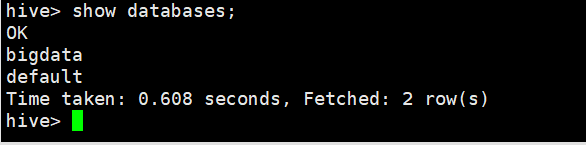
(1)创建数据库create database bigdata;

显示数据库show databases;

使用数据库use bigdata;







1. 创建表

①创建内部表

create table if not exists student(

id int,

name string,

birthday timestamp)

row format delimited

fields terminated by '\t'

lines terminated by '\n'

stored as textfile

location '/hive/warehouse/ student';

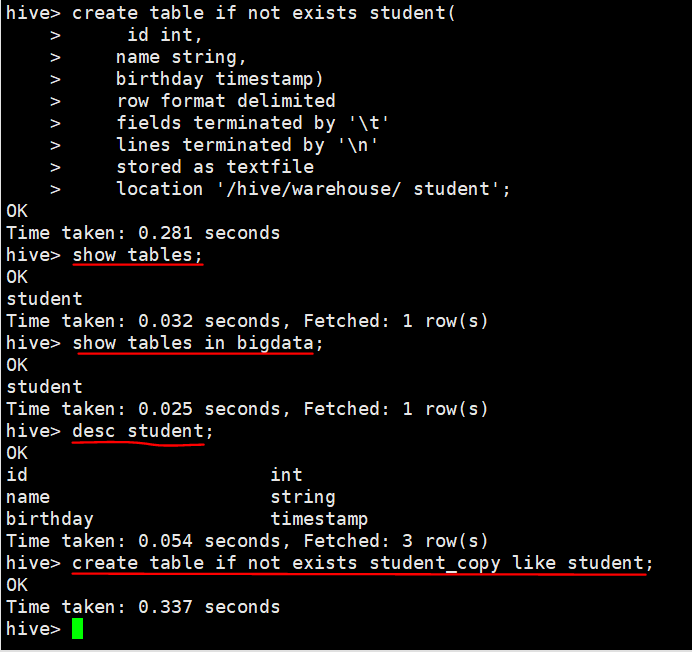
查看当前数据库中表show tables;

查看数据库中的表 show tables in bigdata；

查看数据表的结构信息desc student;

通过复制另一张表的表结构来创建表

create table if not exists student\_copy like student;



②创建外部表，名为student\_external

create external table if not exists student\_external(

id int,

name string,

birthday timestamp)

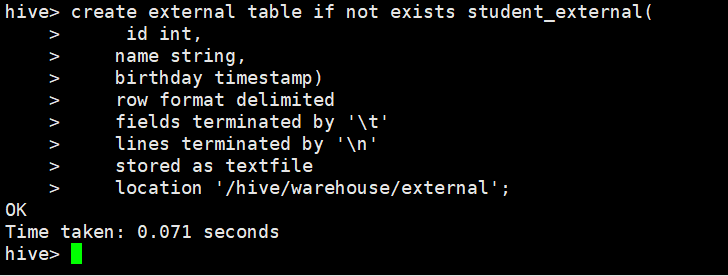
row format delimited

fields terminated by '\t'

lines terminated by '\n'

stored as textfile

location '/hive/warehouse/external';



③创建分区表,名为teacher\_partition

create table teacher\_partition(

id string,

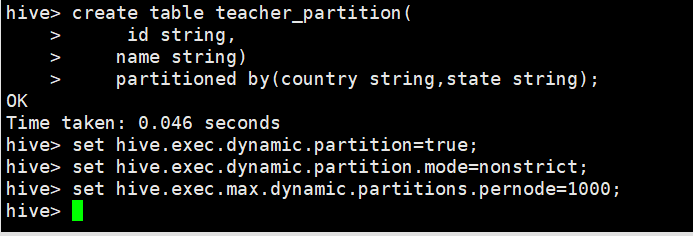
name string)

partitioned by(country string,state string);

set hive.exec.dynamic.partition=true;

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

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



④创建桶表teacher\_bucket

create table teacher\_bucket(

id string,

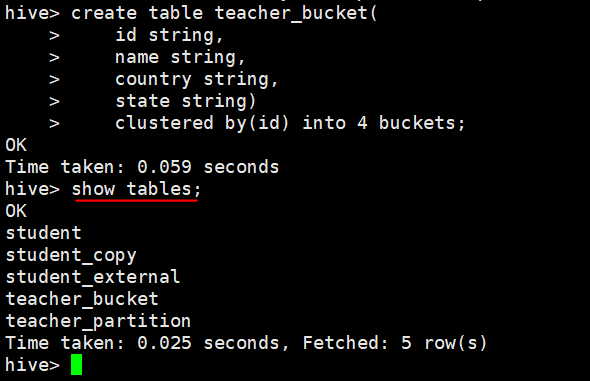
name string,

country string,

state string)

clustered by(id) into 4 buckets;

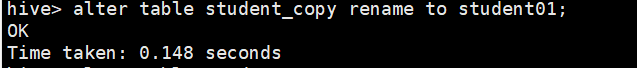
show tables;



1. 修改表alter table

①对表重命名

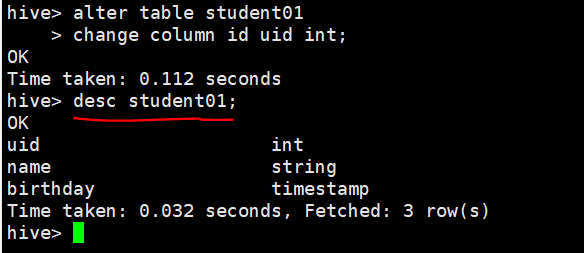
alter table student\_copy rename to student01;



②修改列信息

alter table student01

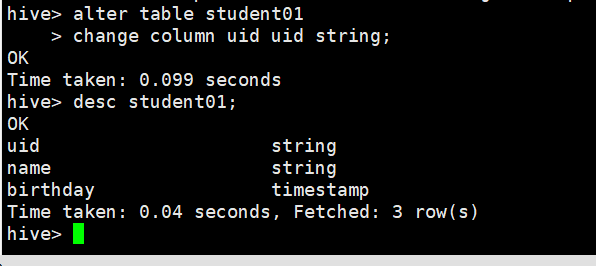
change column id uid int;



③修改字段的类型

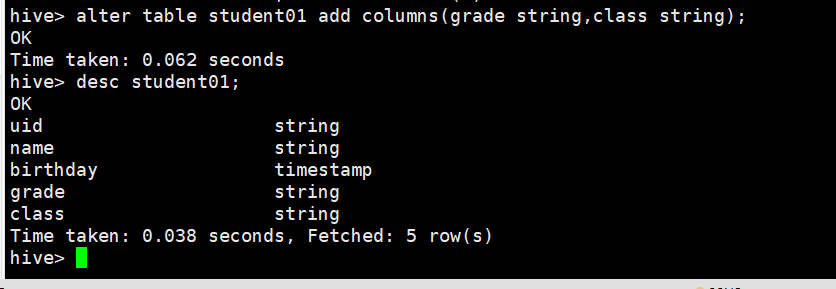
alter table student01

change column uid uid string;



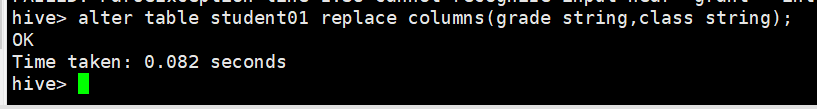
④增加列

alter table student01 add columns(grade string,class string);



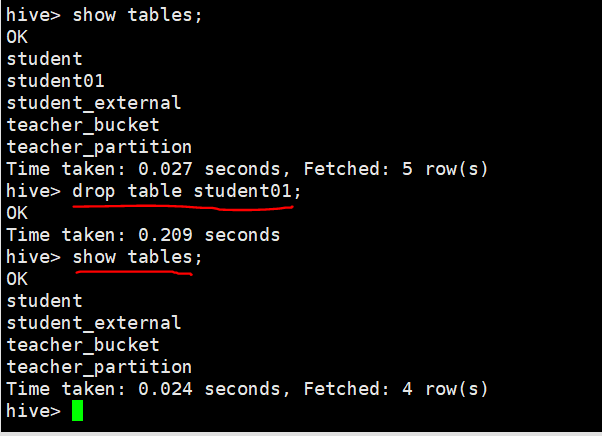
⑤删除或替换列

alter table student01 replace columns(grade string,class string);



⑥删除表

drop table student01;



1. 装载数据

①从本地路径下一次性装大量的数据的方式。以内部表teacher为例。

create table teacher1(id int,name string,country string,state string)

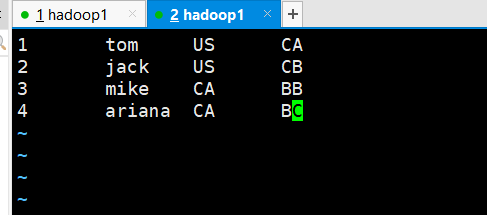
row format delimited fields terminated by '\t'

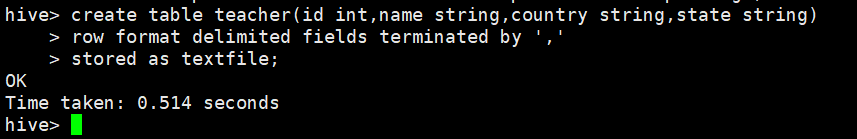
null defined as ' '

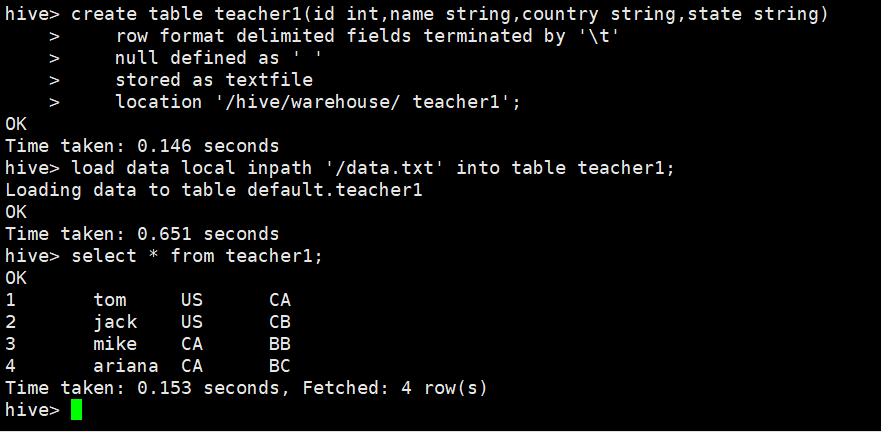
stored as textfile

location '/hive/warehouse/ teacher1';

load data local inpath '/data.txt' into table teacher1;

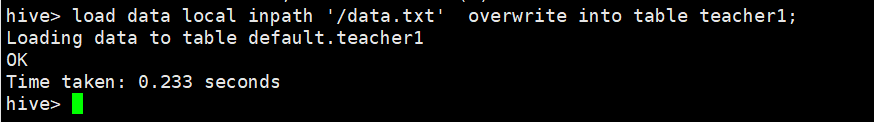






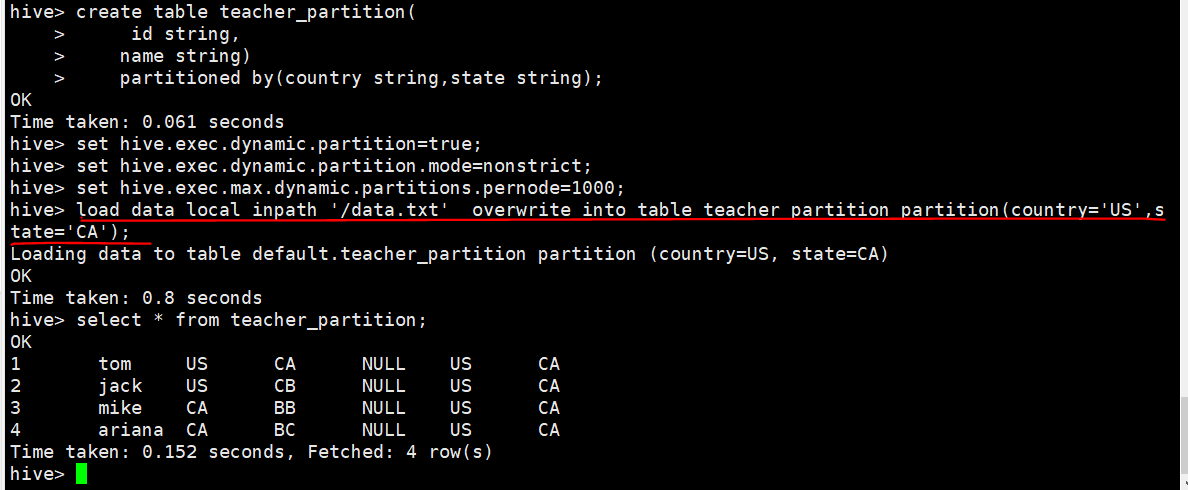
②覆盖表中已有的记录。

load data local inpath '/data.txt' overwrite into table teacher1;



③装载分区表数据.

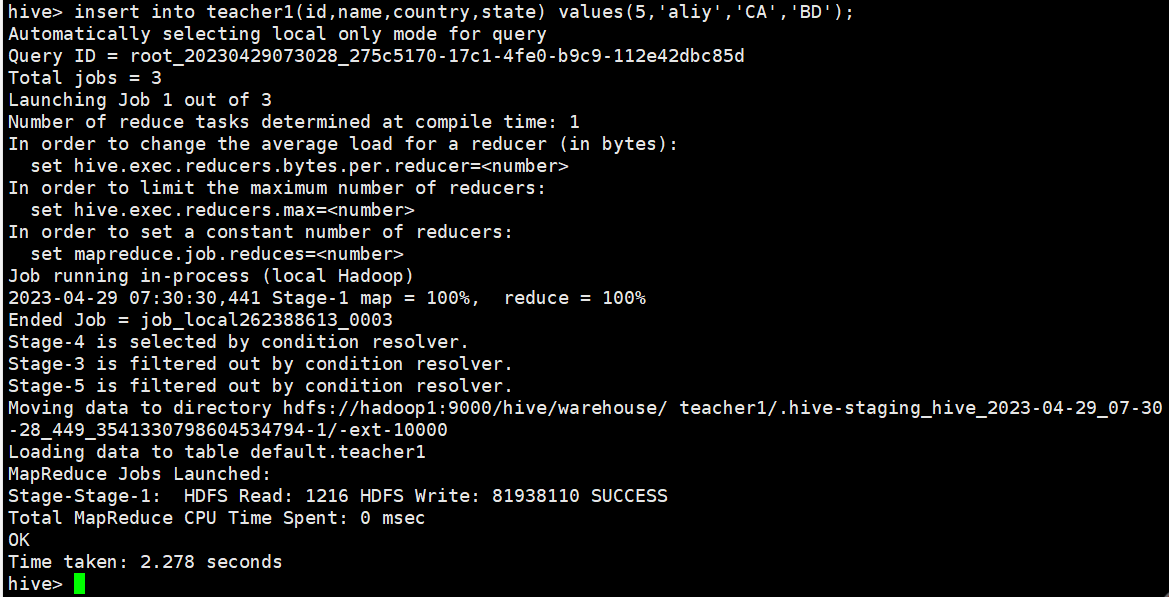
load data local inpath '/data.txt' overwrite into table teacher\_partition partition(country='US',state='CA');



1. 插入数据

①标准SQL插入数据

insert into teacher1(id,name,country,state) values(5,'aliy','CA','BD');



②通过查询语句向表中插入数据

create table teacher01(id int,name string,country string,state string)

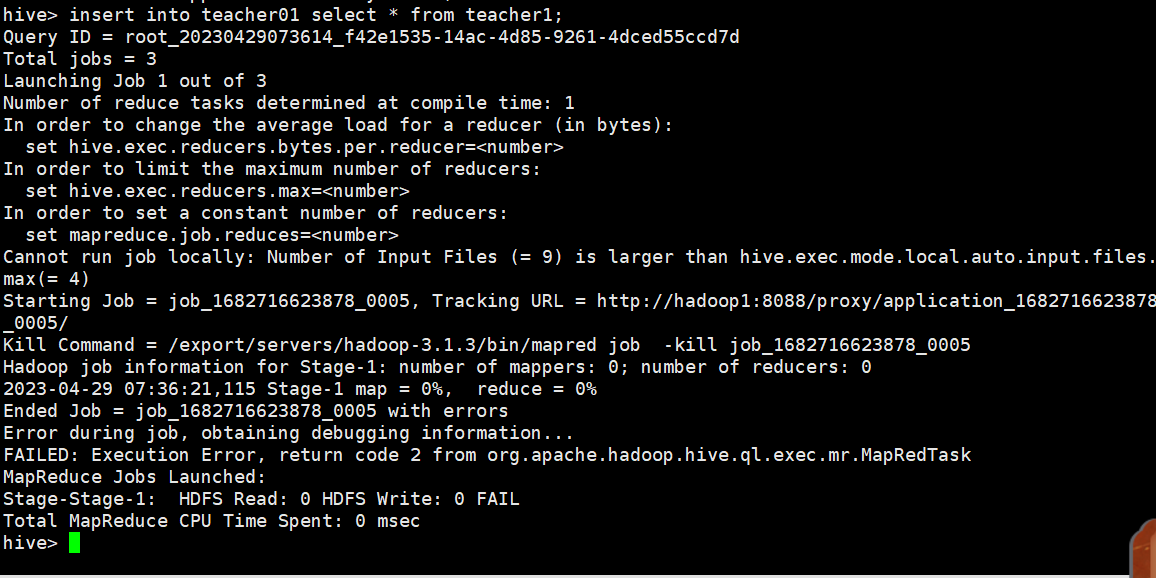
row format delimited fields terminated by '\t'

null defined as ' '

stored as textfile

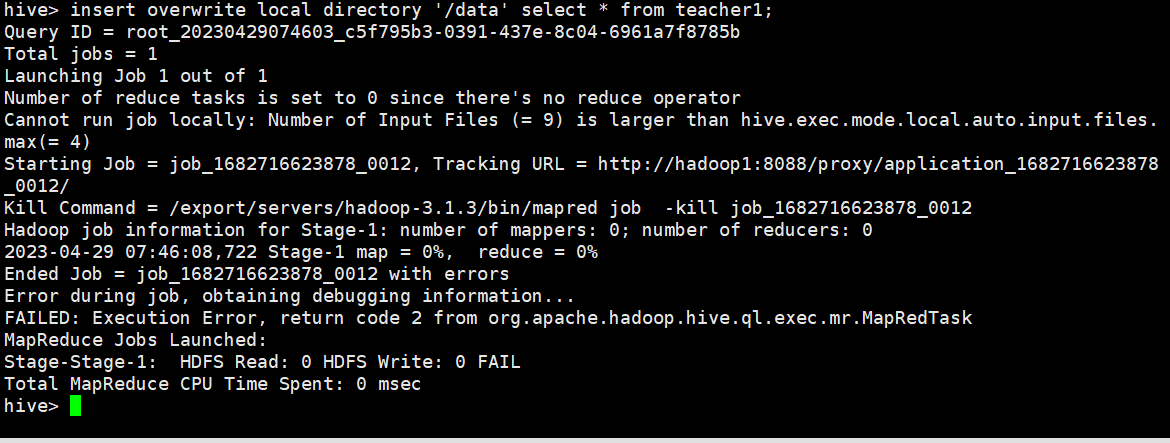
location '/hive/warehouse/ teacher01';

insert into teacher01 select \* from teacher1;



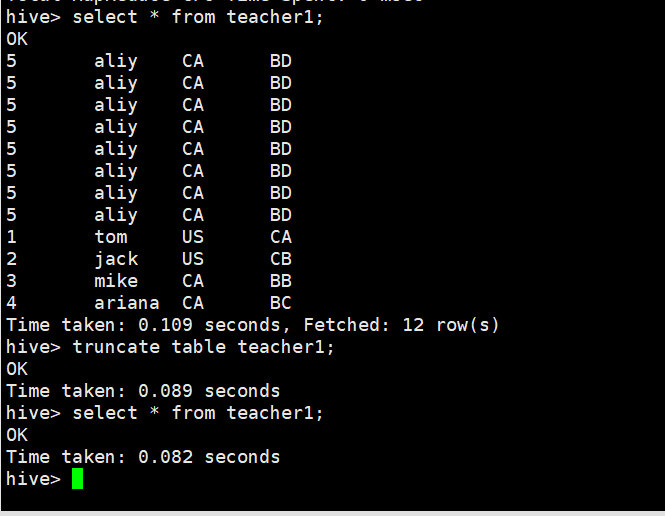
③导出数据

insert overwrite local directory '/data' select \* from teacher1;



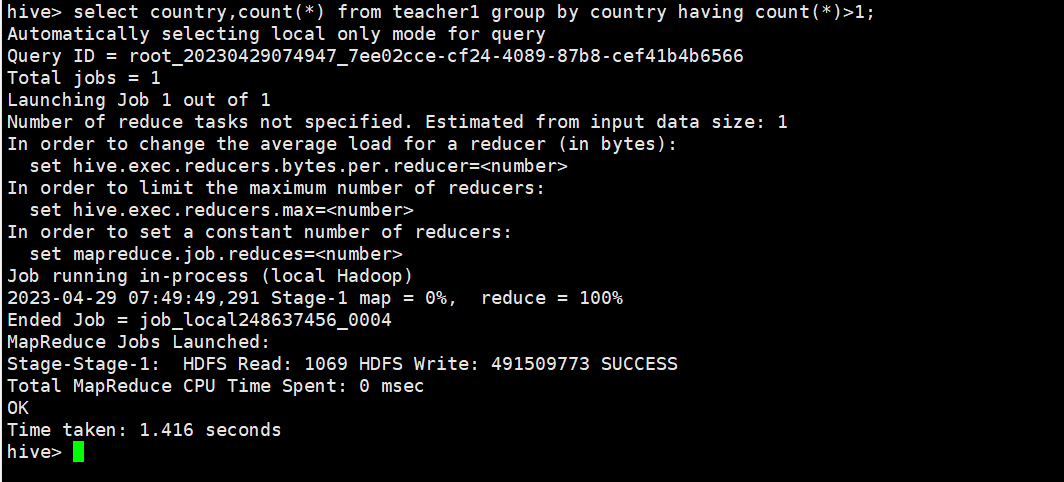
④删除、更新数据

truncate table teacher1;



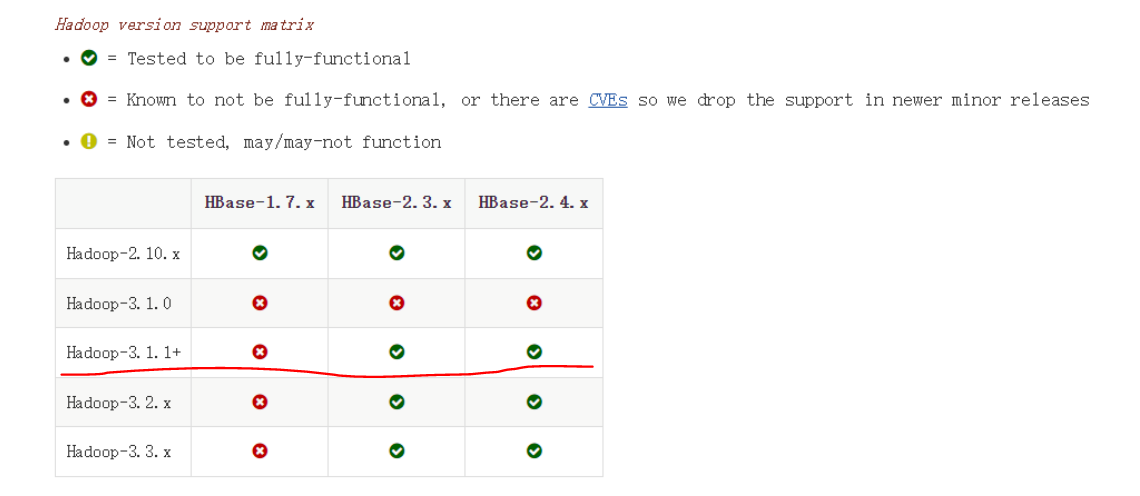
(6)数据查询

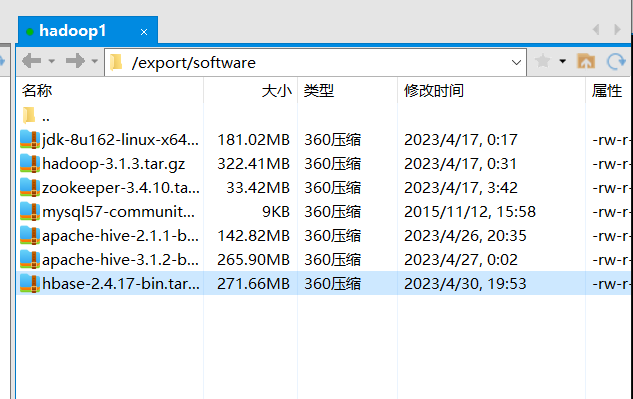
select country,count(\*) from teacher1 group by country having count(\*)>1;

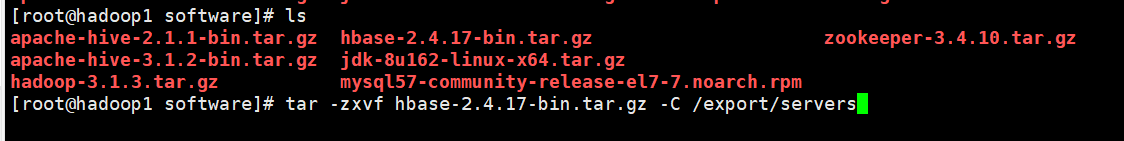


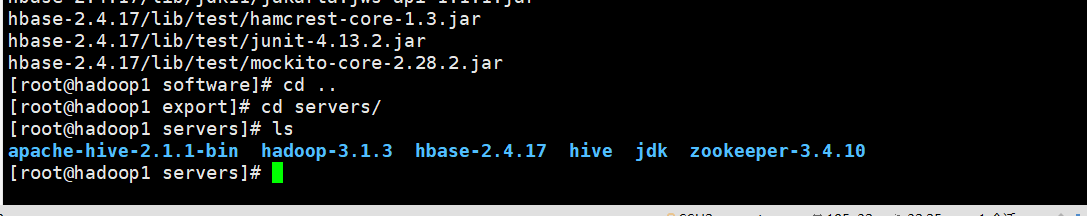
1. 安装Hbase并完成配置
2. 下载安装包，上传到/export/software/并解压安装包到/export/servers。(安装包注意hbase与hadoop的版本兼容)

tar -zxvf hbase-2.4.17-bin.tar.gz -C /export/servers









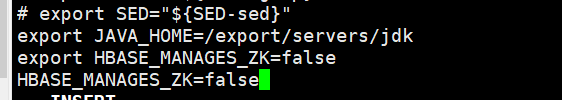
1. 修改配置文件hbase-env.sh、hbase-site.xml、regionservers,路径在HBase解压路径的conf目录下。

①vi hbase-env.sh

export JAVA\_HOME=/export/servers/jdk

export HBASE\_MANAGES\_ZK=false

HBASE\_MANAGES\_ZK=false



②vi hbase-site.xml

<configuration>

<!-- HBase数据目录位置，其中hadoop1是Hadoopmaster机器的机器名-->

<property>

<name>hbase.rootdir</name>

<value>hdfs://hadoop1:9000/HBase</value>

</property>

<!-- 指定hbase是否分布式运行 -->

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

<!-- 默认HMaster HTTP访问端口 -->

<property>

<name>hbase.master.info.port</name>

<value>16010</value>

</property>

<!-- 默认HRegionServer HTTP访问端口 -->

<property>

<name>hbase.regionserver.info.port</name>

<value>16030</value>

</property>

<!-- 指定zookeeper的地址，多个用“,”分割 -->

<property>

<name>hbase.zookeeper.quorum</name>

<value>hadoop1,hadoop2,hadoop3</value>

</property>

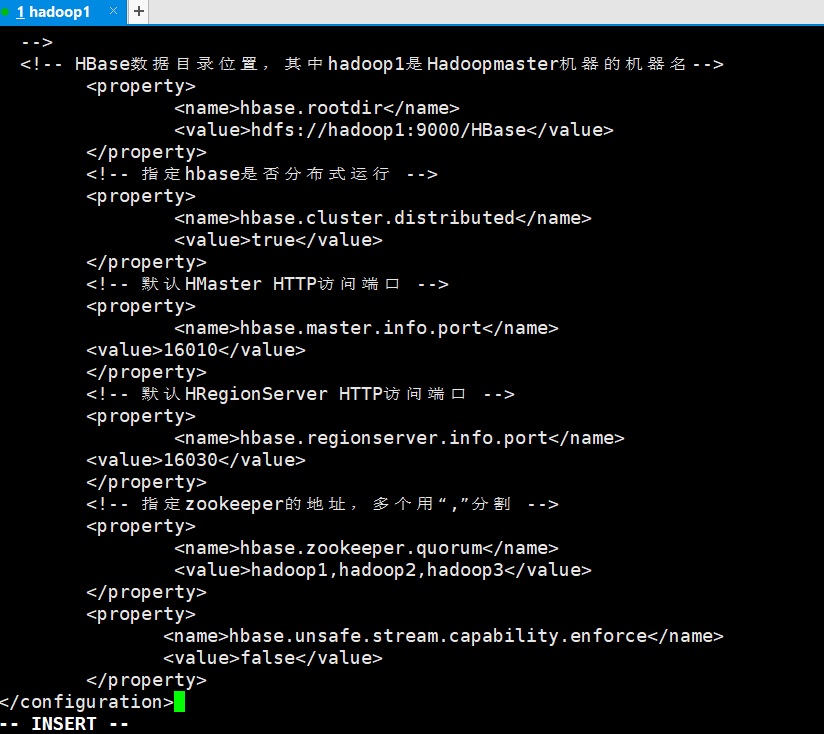
<property>

<name>hbase.unsafe.stream.capability.enforce</name>

<value>false</value>

</property>

</configuration>

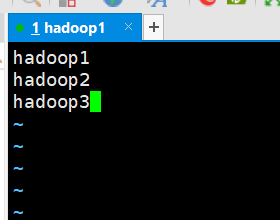


③vi regionservers

hadoop1

hadoop2

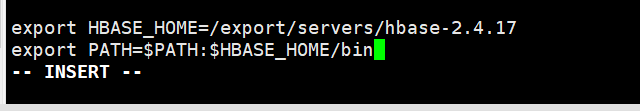
hadoop3



(3)配置系统变量，在/etc/profile文件加入：

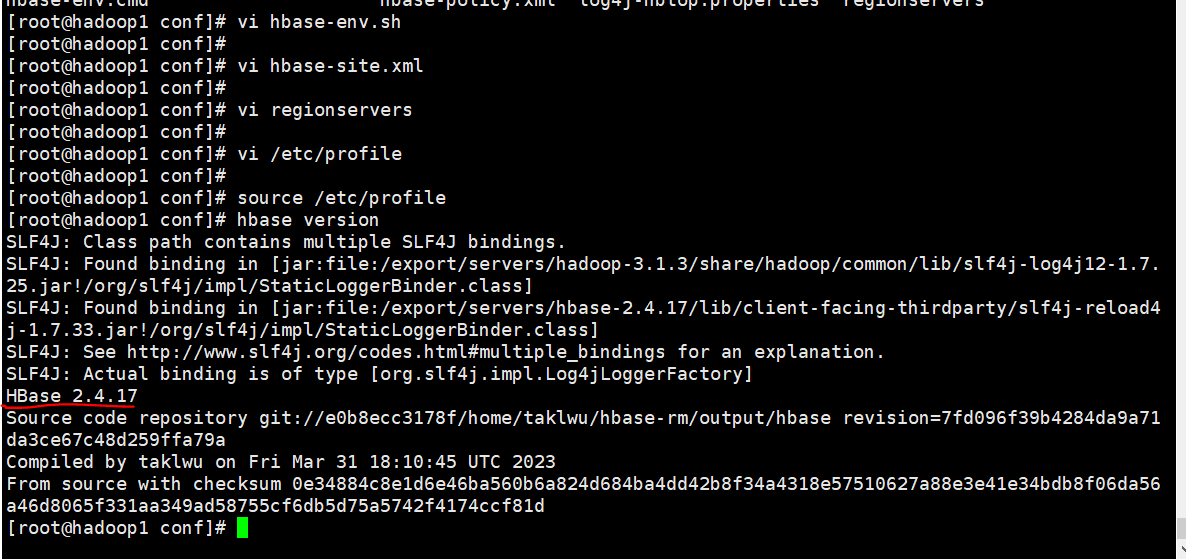
export HBASE\_HOME=/export/servers/hbase-2.4.17

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



(4)系统变量生效 source /etc/profile

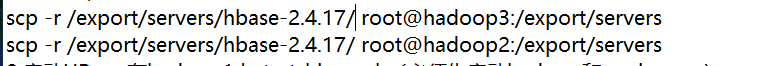
hbase version



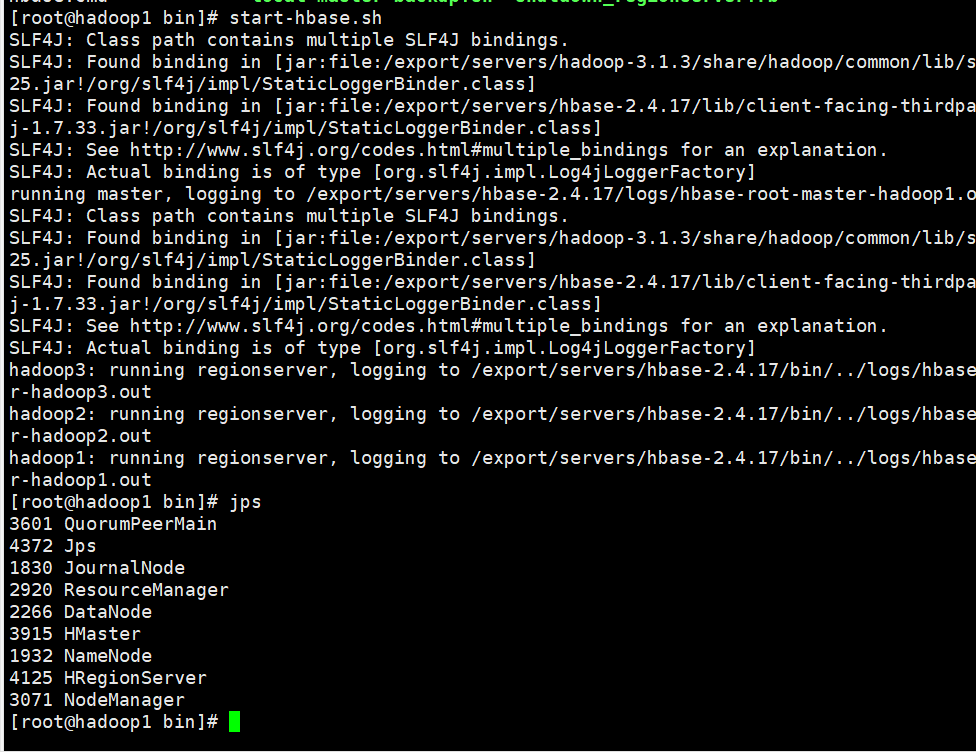
(5)复制安装文件到其他两个机器

scp -r /export/servers/hbase-2.4.17/ root@hadoop3:/export/servers

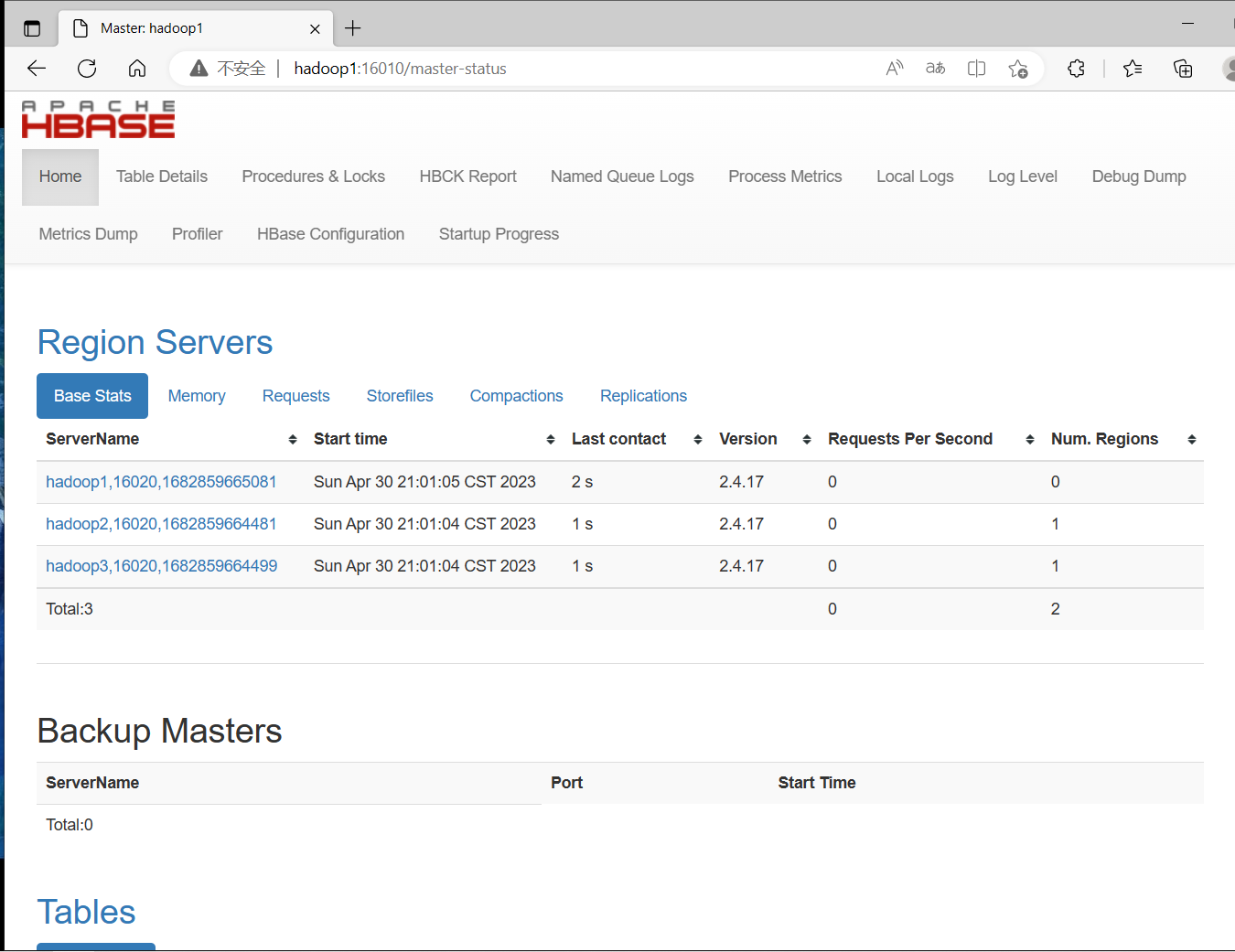
scp -r /export/servers/hbase-2.4.17/ root@hadoop2:/export/servers



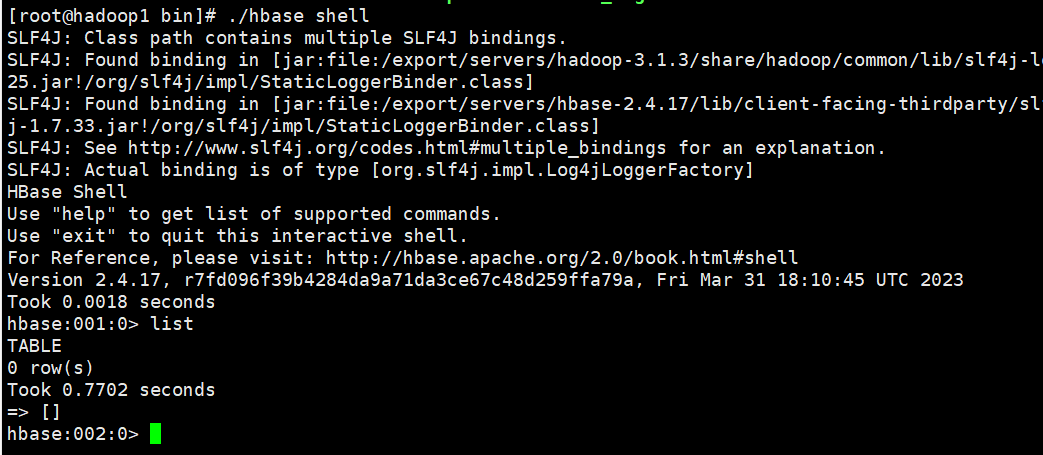
1. 启动HBase,在hadoop1上start-hbase.sh。查看是否启动成功jps。



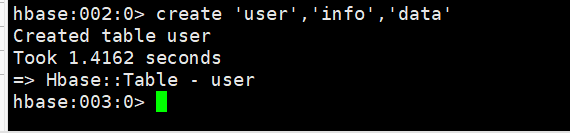
1. 在浏览器输入hadoop1:16010/



1. 用Hadoop提供的HBase Shell命令完成数据管理任务。
2. ./habse shell启动hbase。显示HBase中的表list。



1. 创建表user，包含info、data两个列族。create 'user','info','data'。



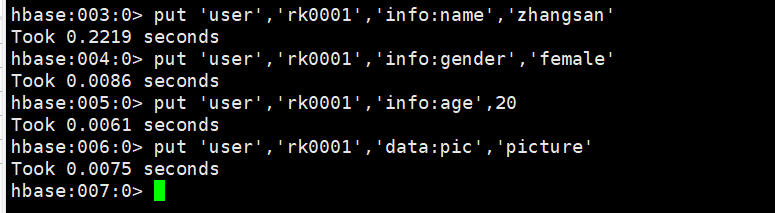
1. 向表user中插入数据。

put 'user','rk0001','info:name','zhangsan'

put 'user','rk0001','info:gender','female'

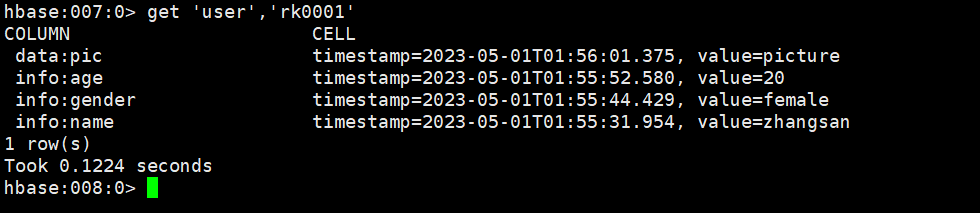
put 'user','rk0001','info:age',20

put 'user','rk0001','data:pic','picture'



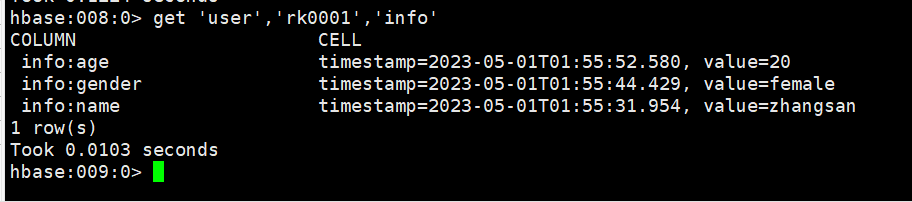
(5)获取user表中RowKey为rk0001的所有信息

get 'user','rk0001'



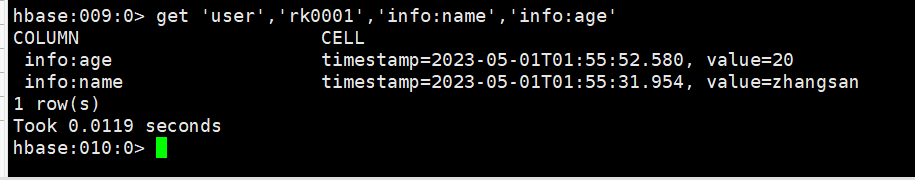
(6)获取user表中RowKey为rk0001和info列族的所有信息

get 'user','rk0001','info'



(7)获取user表中RowKey为rk0001和info列族的name、age列标识符的信息

get 'user','rk0001','info:name','info:age'

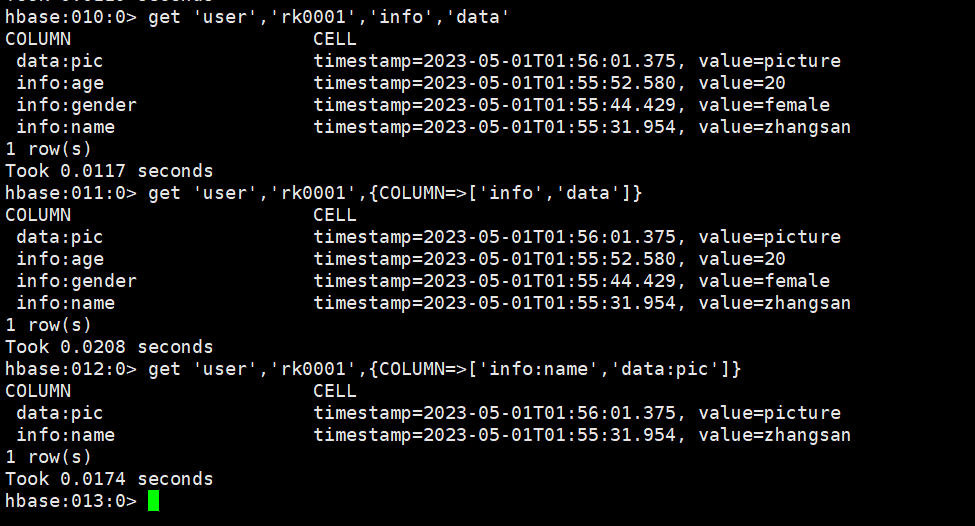


(8)获取user表中RowKey为rk0001,info、data列族的信息

get 'user','rk0001','info','data'

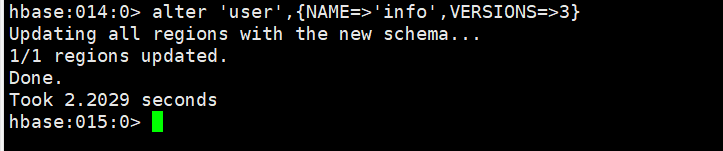
get 'user','rk0001',{COLUMN=>['info','data']}

get 'user','rk0001',{COLUMN=>['info:name','data:pic']}



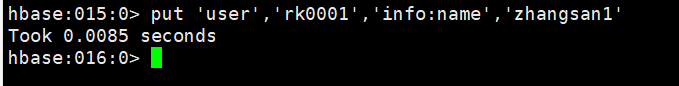
(9)获取user表中RowKey为rk0001,列族为info、版本号最新3个的信息

alter 'user',{NAME=>'info',VERSIONS=>3}



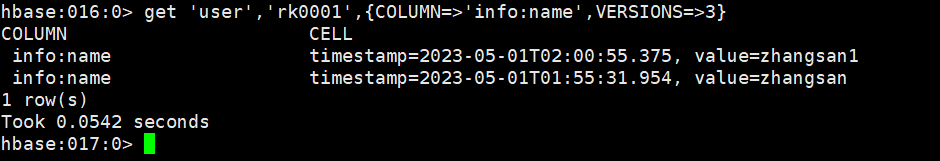
(10)向user表info:name列插入数据

put 'user','rk0001','info:name','zhangsan1'



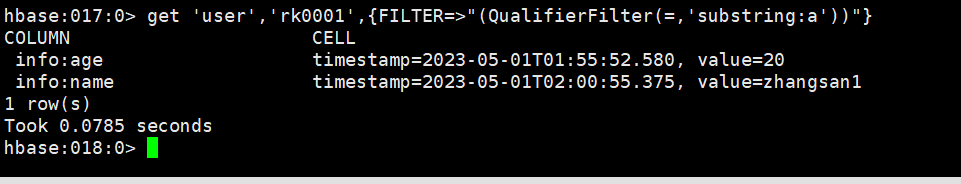
(11)查看user表info:name历史版本

get 'user','rk0001',{COLUMN=>'info:name',VERSIONS=>3}



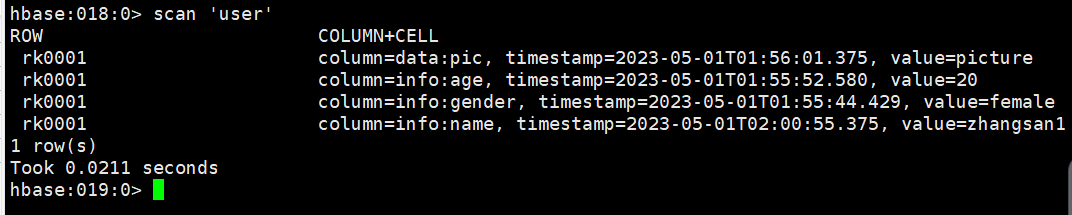
(12)获取user表中RowKey为rk0001、列标识符中含有a的信息

get 'user','rk0001',{FILTER=>"(QualifierFilter(=,'substring:a'))"}



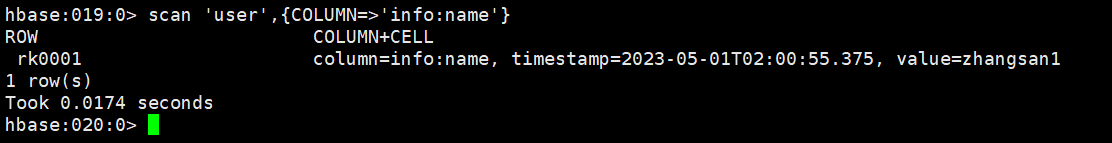
(13)利用scan命令查询user表中的所有信息。

①查询user表中的所有信息。scan 'user'



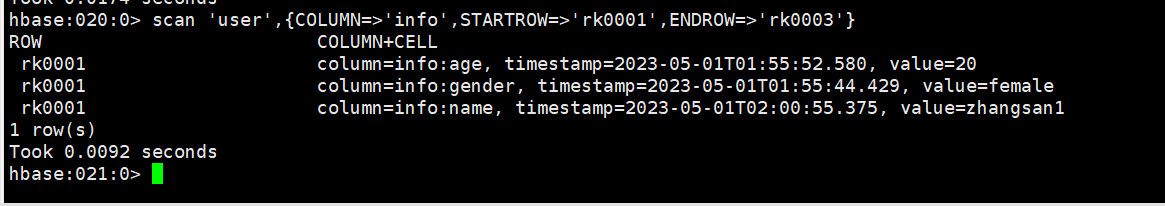
②查询user表中列族为info、列标识符为name的信息

scan 'user',{COLUMN=>'info:name'}



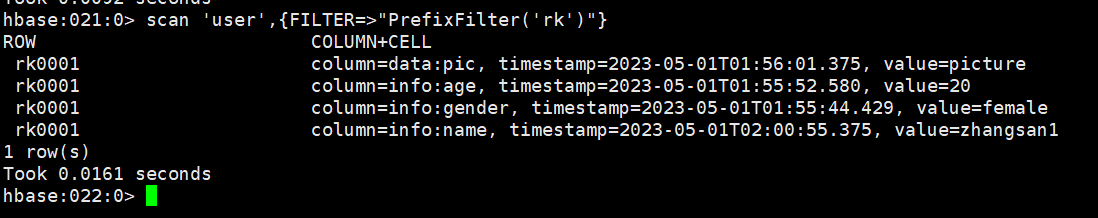
③查询user表中列族为info、RowKey范围(rk0001,rk0003)

scan 'user',{COLUMN=>'info',STARTROW=>'rk0001',ENDROW=>'rk0003'}



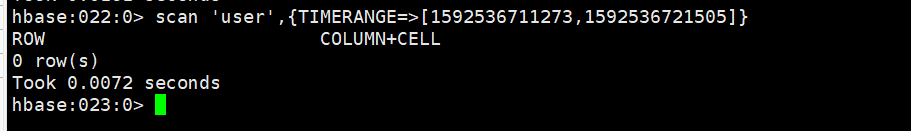
④查询user表中RowKey以rk字符开头的数据

scan 'user',{FILTER=>"PrefixFilter('rk')"}



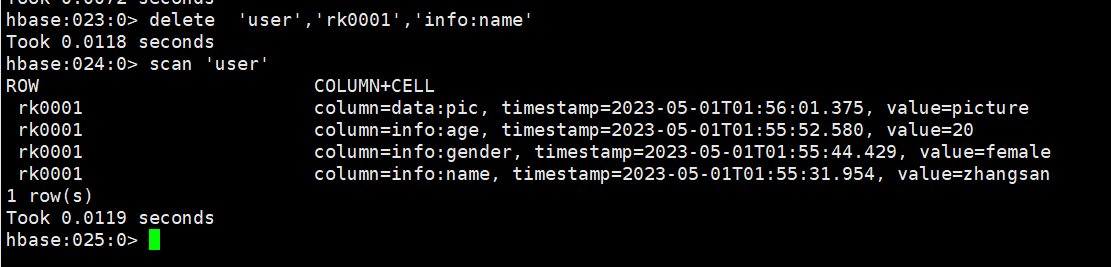
⑤查询user表中指定时间范围的数据

scan 'user',{TIMERANGE=>[1592536711273,1592536721505]}

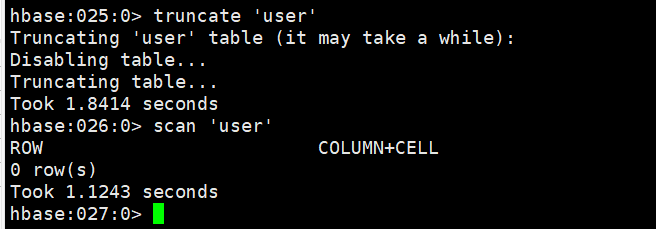


(14)删除数据

①删除user表中RowKey为rk0001、列标识符为info:name的数据。delete 'user','rk0001','info:name'



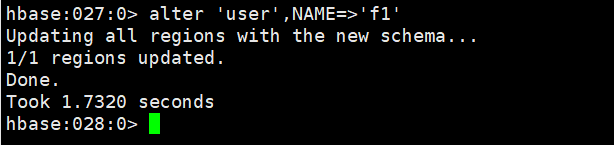
②清空user表数据。truncate 'user'



(15)修改表结构

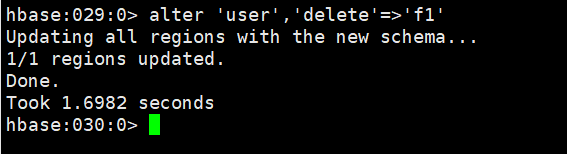
①添加user表f1列族

alter 'user',NAME=>'f1'



②删除user表f1列族

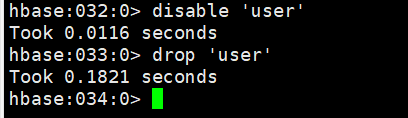
alter 'user','delete'=>'f1'



③删除user表（先停用表）

disable 'user'

drop 'user'



4.使用Hbase提供的Java API 完成一系列指定编程任务，实现应用程序对数据的管理及存取。

（1）导入依赖包，pom.xml文件配置。

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>hbase-api</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>8</maven.compiler.source>

<maven.compiler.target>8</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<!--单元测试依赖-->

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<!--HBase客户端依赖-->

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-client</artifactId>

<version>2.0.0</version>

</dependency>

<!--HBase核心依赖-->

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-common</artifactId>

<version>2.0.0</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

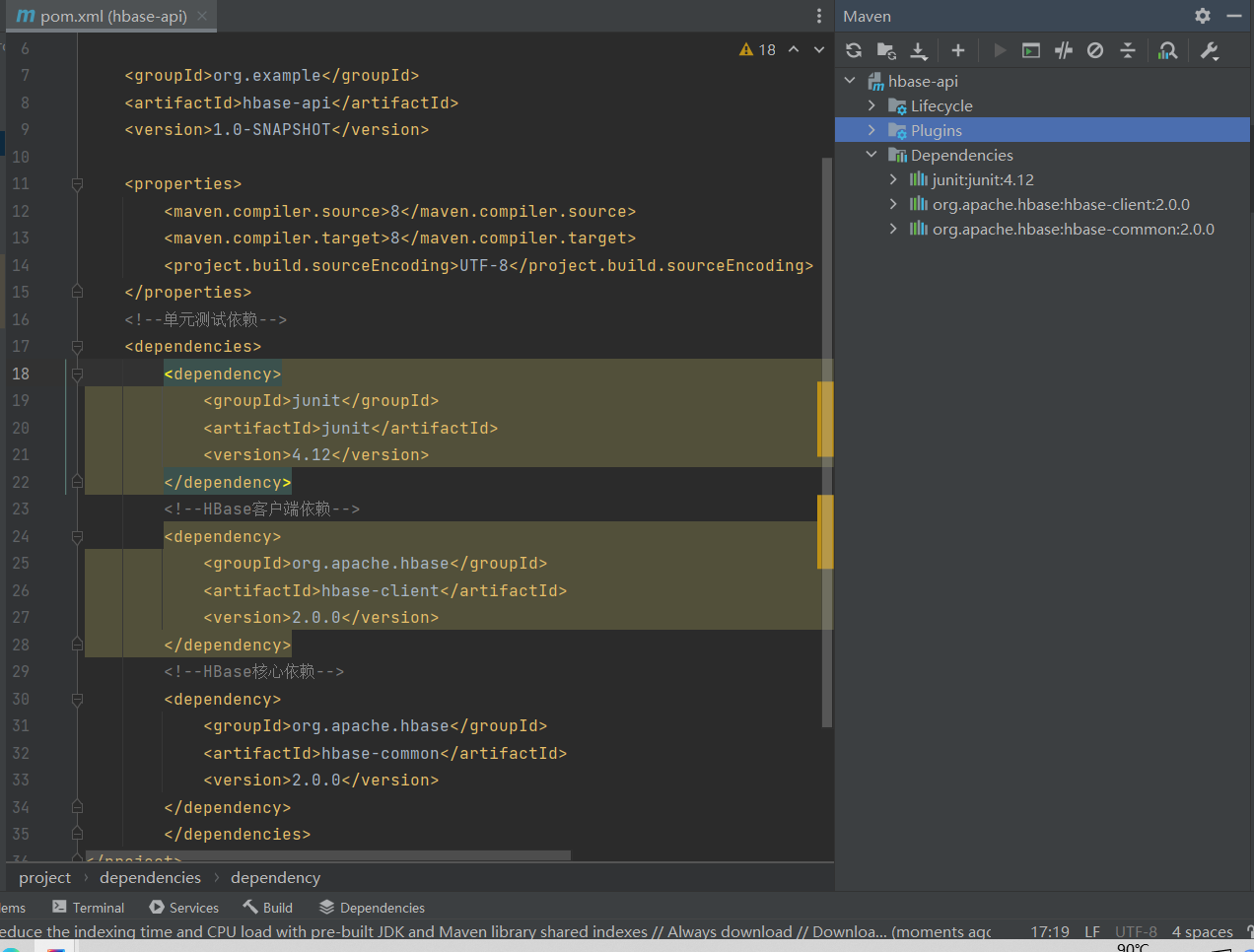
<artifactId>log4j</artifactId>

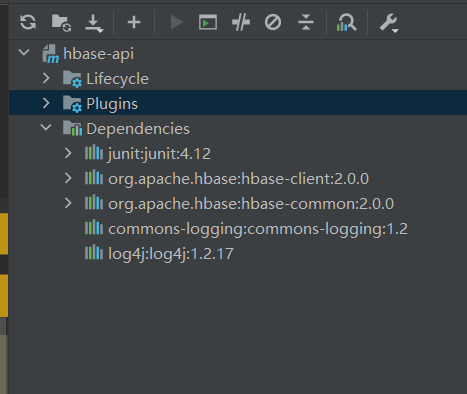
<version>1.2.17</version>

</dependency>

</dependencies>

</project>





1. 新建测试类 HbaseApiDemo

package com.hadoop.hbase;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.\*;

import org.apache.hadoop.hbase.client.\*;

import org.apache.hadoop.hbase.util.Bytes;

import org.junit.Before;

import org.junit.Test;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class HbaseApiDemo{

private Configuration conf=null;

private Connection conn=null;

@Before

public void init() throws IOException{

//获取Configuration对象

conf =HBaseConfiguration.create();

//设置Zookeeper集群地址

conf.set("hbase.zookeeper.quorum", "hadoop1:2181,hadoop2:2181,hadoop3:2181");

//获取连接

conn = ConnectionFactory.createConnection(conf);

}

//创建表

@Test

public void CreateTable() throws Exception {

try{

//获取操作对象

Admin admin=conn.getAdmin();

//构建一个user表

TableDescriptorBuilder t\_user = TableDescriptorBuilder.newBuilder(TableName.valueOf("t\_user"));

//创建列族info

ColumnFamilyDescriptor of = ColumnFamilyDescriptorBuilder.of("info");

t\_user.setColumnFamily(of);

//创建列族data

ColumnFamilyDescriptor of1 = ColumnFamilyDescriptorBuilder.of("data");

t\_user.setColumnFamily(of1);

//构建

TableDescriptor build = t\_user.build();

//创建表

admin.createTable(build);

// 关闭连接

admin.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

}

}

//插入数据

@Test

public void testPut() throws Exception {

//创建table对象，通过table对象来添加数据

Table table = conn.getTable(TableName.valueOf("t\_user"));

//创建一个集合，用于存放Put对象

ArrayList<Put> puts = new ArrayList<Put>();

//构建put对象（kv形式），并指定其行键

Put put01 = new Put(Bytes.toBytes("rk002"));

put01.addColumn(Bytes.toBytes("info"),

Bytes.toBytes("username"), Bytes.toBytes("zhangsan"));

put01.addColumn(Bytes.toBytes("info"),

Bytes.toBytes("password"), Bytes.toBytes("345678"));

Put put02 = new Put("rk003".getBytes());

put02.addColumn(Bytes.toBytes("info"),

Bytes.toBytes("username"), Bytes.toBytes("lisi"));

//把所有的put对象添加到一个集合中

puts.add(put01);

puts.add(put02);

//提交所有的插入数据的记录

table.put(puts);

//关闭

table.close();

conn.close();

}

//查询表

@Test

public void testGet() throws IOException {

Table table = conn.getTable(TableName.valueOf("t\_user"));

//得到用于扫描region的对象

Get get = new Get("rk002".getBytes());

//使用HTable得到resultcanner实现类的对象

Result result1 = table.get(get);

List<Cell> cells = result1.listCells();

for (Cell cell : cells) {

//得到rowkey

System.out.println("行键:" + Bytes.toString(CellUtil.cloneRow(cell)));

//得到列族

System.out.println("列族:" + Bytes.toString(CellUtil.cloneFamily(cell)));

System.out.println("列:" + Bytes.toString(CellUtil.cloneQualifier(cell)));

System.out.println("值:" + Bytes.toString(CellUtil.cloneValue(cell)));

}

}

@Test

public void testScan() throws Exception {

//获取table对象

Table table = conn.getTable(TableName.valueOf("t\_user"));

//获取scan对象

Scan scan = new Scan();

//获取查询的数据

ResultScanner scanner = table.getScanner(scan);

//获取ResultScanner所有数据，返回迭代器

Iterator<Result> iter = scanner.iterator();

//遍历迭代器

while (iter.hasNext()) {

//获取当前每一行结果数据

Result result = iter.next();

//获取当前每一行中所有的cell对象

List<Cell> cells = result.listCells();

//迭代所有的cell

for(Cell c:cells){

//获取行键

byte[] rowArray = c.getRowArray();

//获取列族

byte[] familyArray = c.getFamilyArray();

//获取列族下的列名称

byte[] qualifierArray = c.getQualifierArray();

//列字段的值

byte[] valueArray = c.getValueArray();

//打印rowArray、familyArray、qualifierArray、valueArray

System.out.println("行键:"+new String(rowArray, c.getRowOffset(),

c.getRowLength()));

System.out.print("列族:"+new String(familyArray,c.getFamilyOffset(),

c.getFamilyLength()));

System.out.print(" "+"列:"+ new String(qualifierArray,

c.getQualifierOffset(), c.getQualifierLength()));

System.out.println(" "+"值:"+ new String(valueArray,

c.getValueOffset(), c.getValueLength()));

}

System.out.println("-----------------------");

}

//关闭

table.close();

conn.close();

}

//删除表记录

@Test

public void testDel() throws Exception {

//获取table对象

Table table = conn.getTable(TableName.valueOf("t\_user"));

//获取delete对象,需要一个rowkey

Delete delete = new Delete("rk002".getBytes());

//在delete对象中指定要删除的列族-列名称

delete.addColumn("info".getBytes(), "password".getBytes());

//执行删除操作

table.delete(delete);

//关闭

table.close();

conn.close();

}

//删除表

@Test

public void testDrop() throws Exception {

//获取一个表的管理器

Admin admin = conn.getAdmin();

//删除表时先需要禁用表

admin.disableTable(TableName.valueOf("t\_user"));

admin.deleteTable(TableName.valueOf("t\_user"));

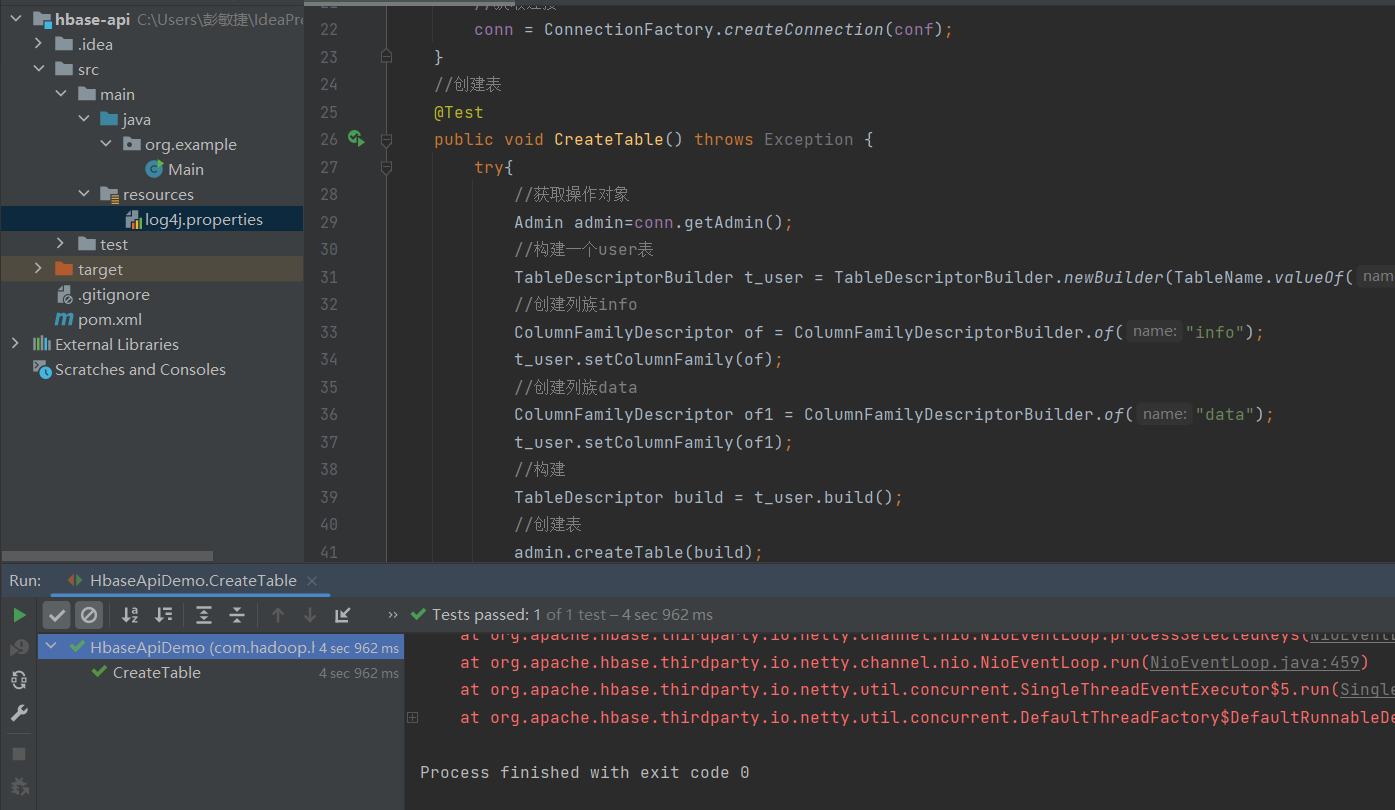
//关闭

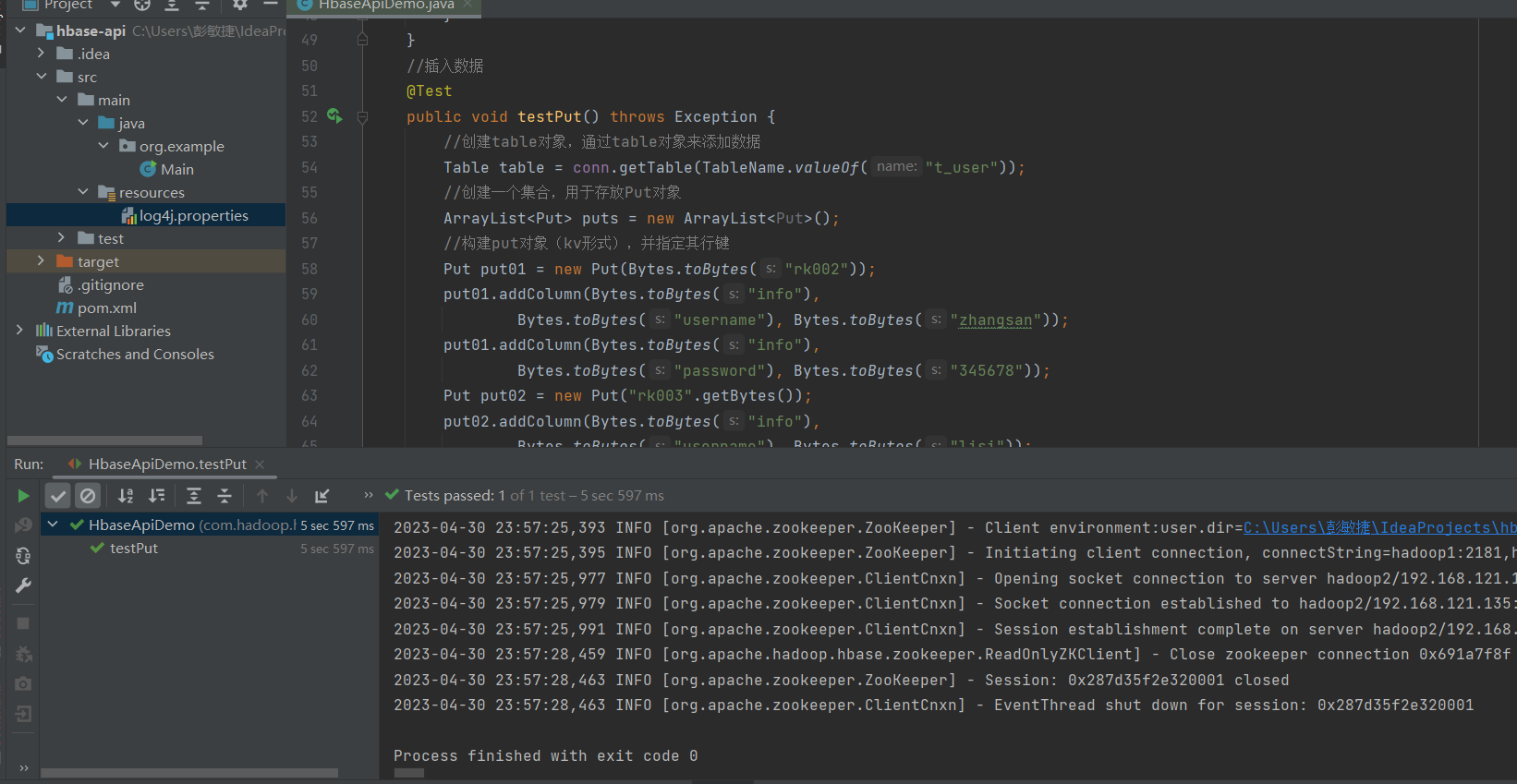
admin.close();

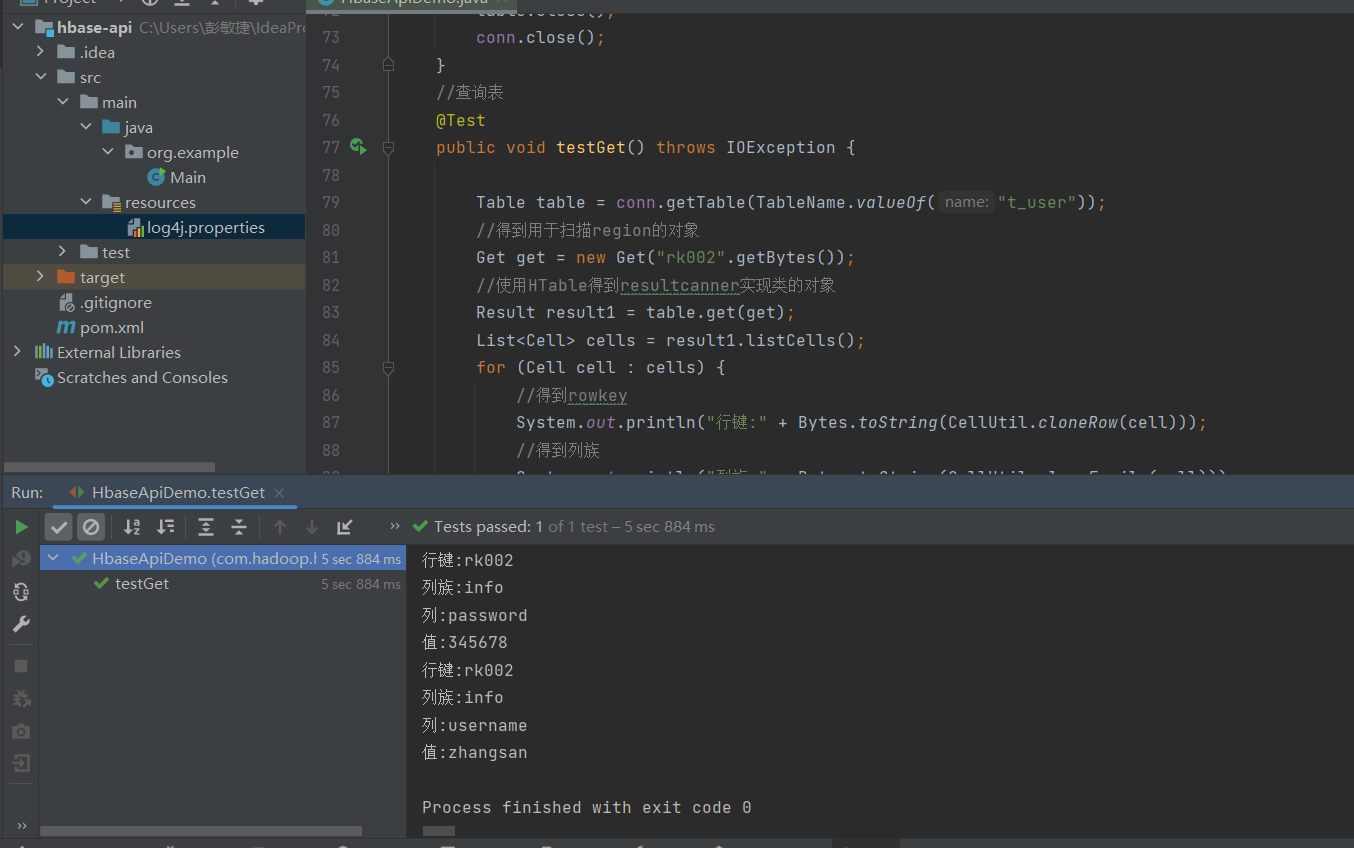
conn.close();

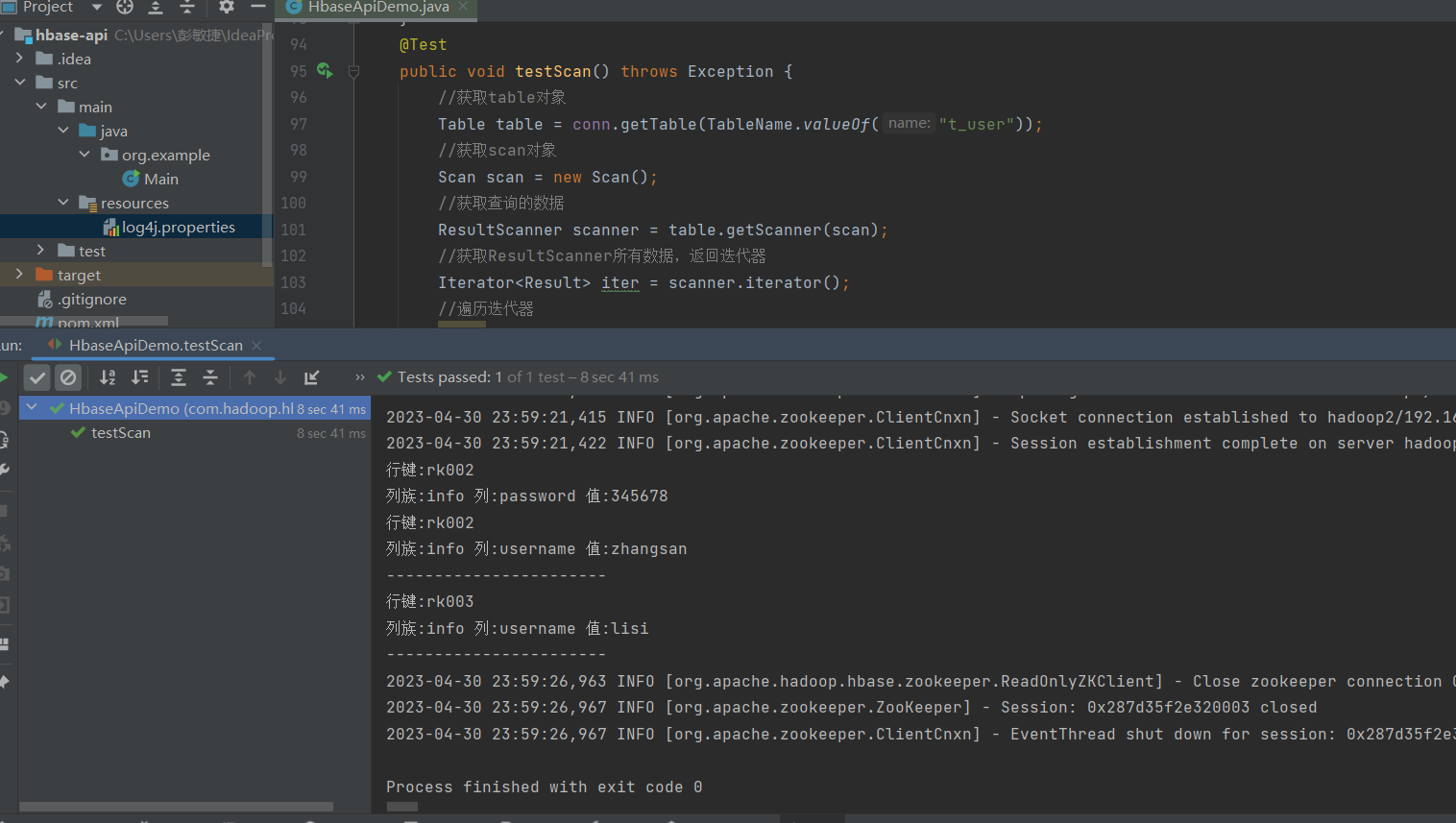
}

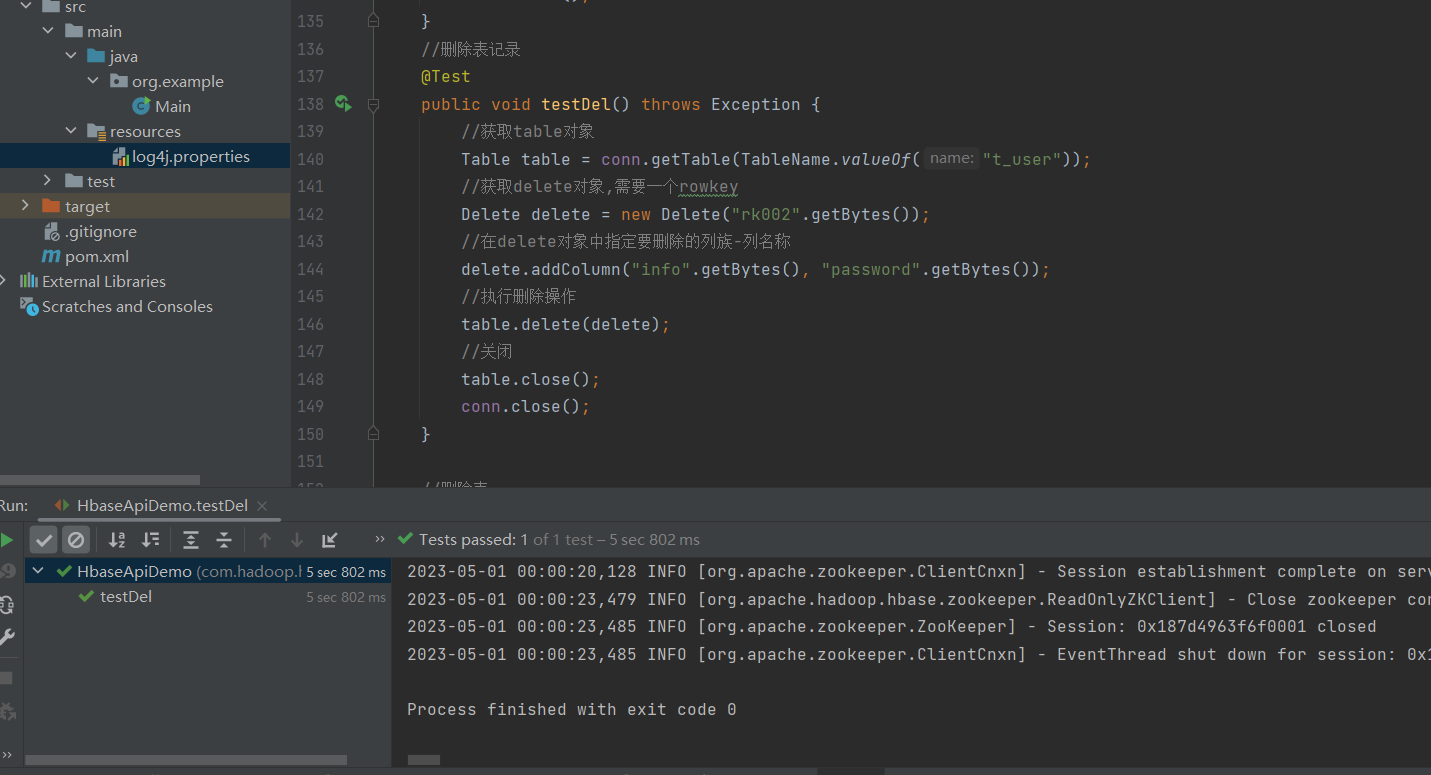
}

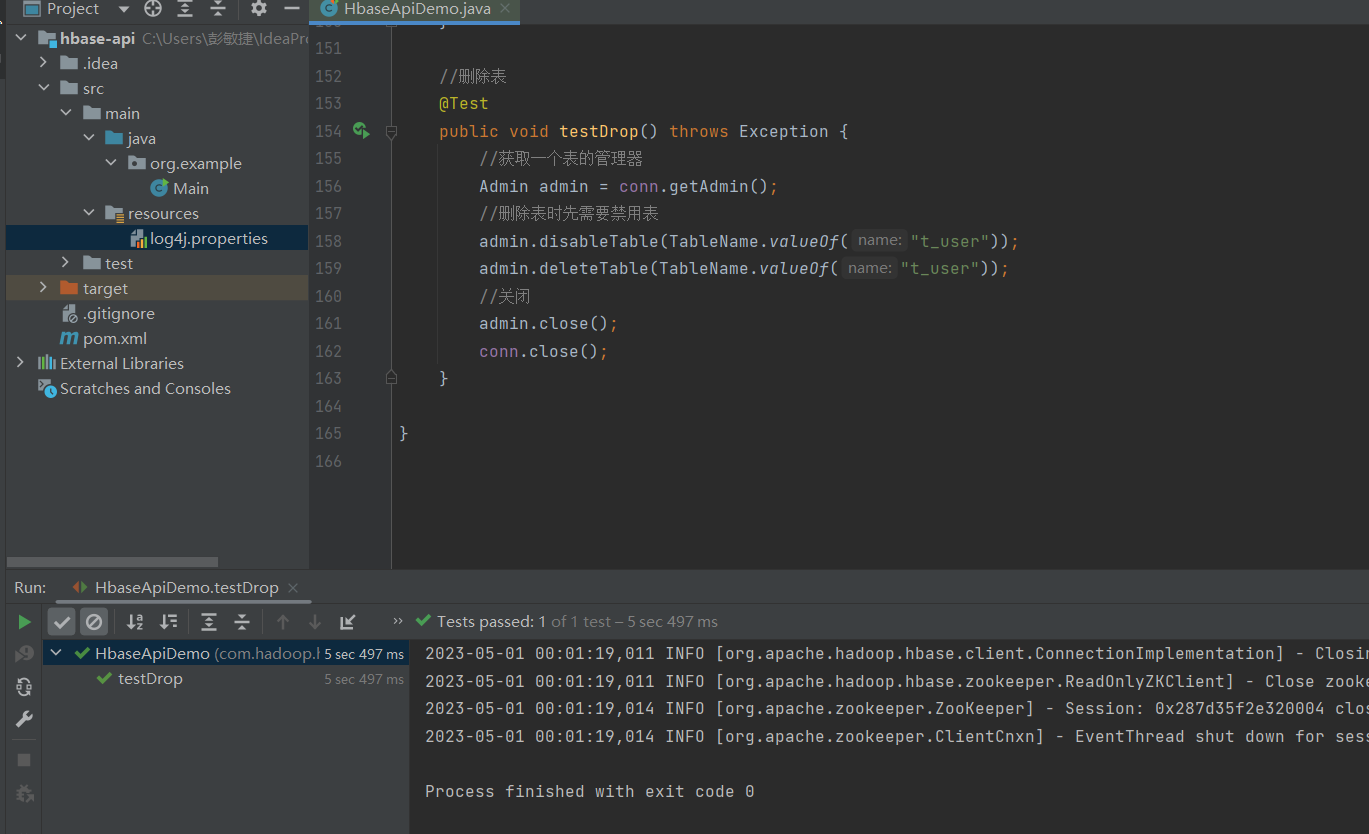




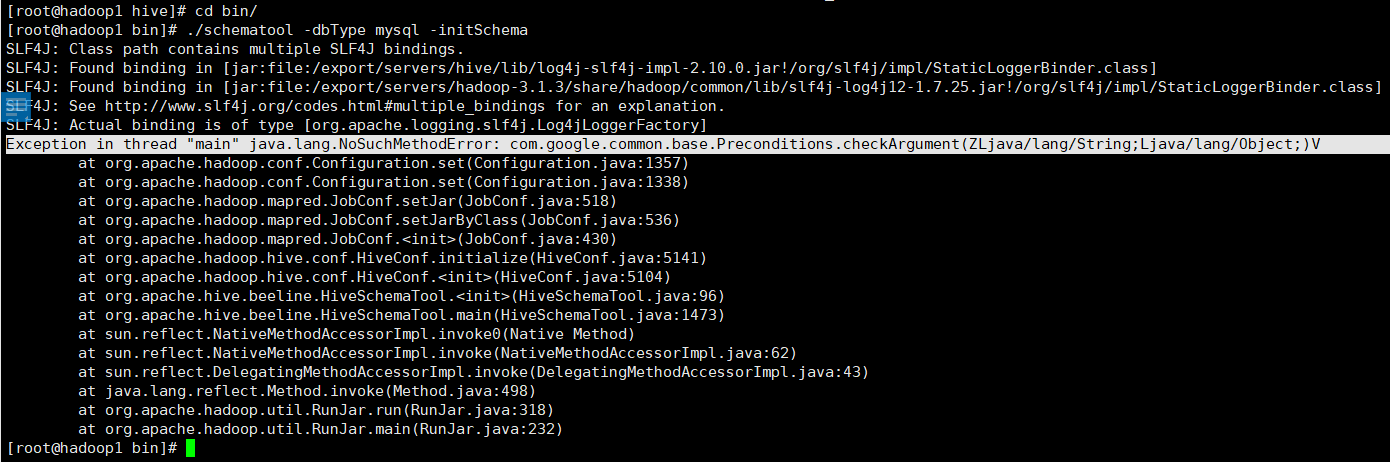






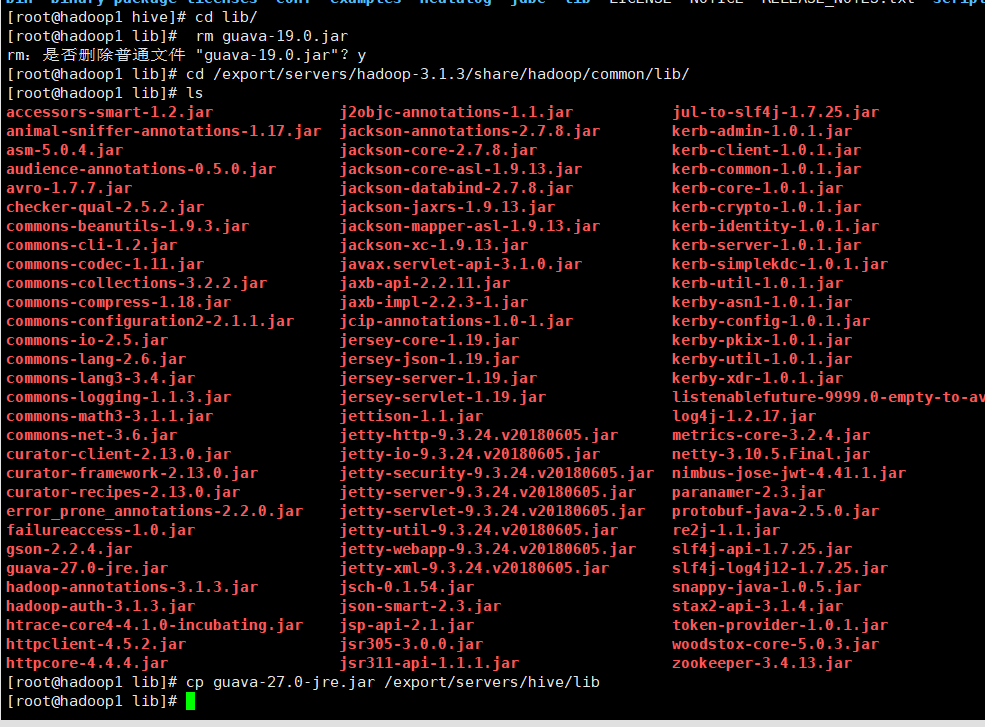


1. **实验总结**
2. hive安装
3. Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V

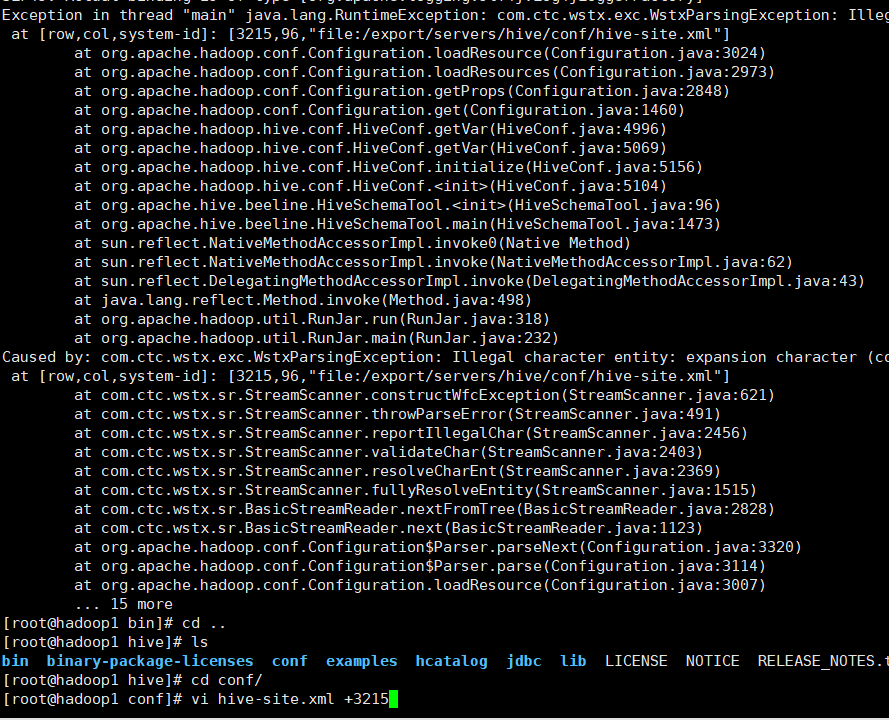


### 原因：[hadoop](https://so.csdn.net/so/search?q=hadoop&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/zhan13253807836/article/details/_blank)和hive的两个guava.jar版本不一致，两个位置分别位于下面两个目录。

解决方法：将高版本复制到低版本目录中，删除低版本。

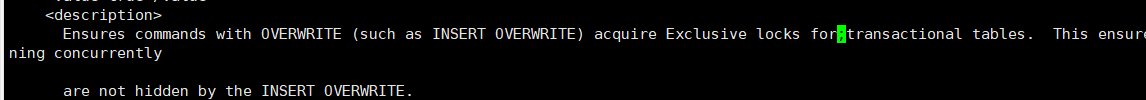


1. Exception in thread "main" java.lang.RuntimeException: com.ctc.wstx.exc.WstxParsingException: Illegal character entity: expansion character (code 0x8 at [row,col,system-id]: [3215,96,"file:.........



原因：上面的第3215行，第96个字符是非法字符。

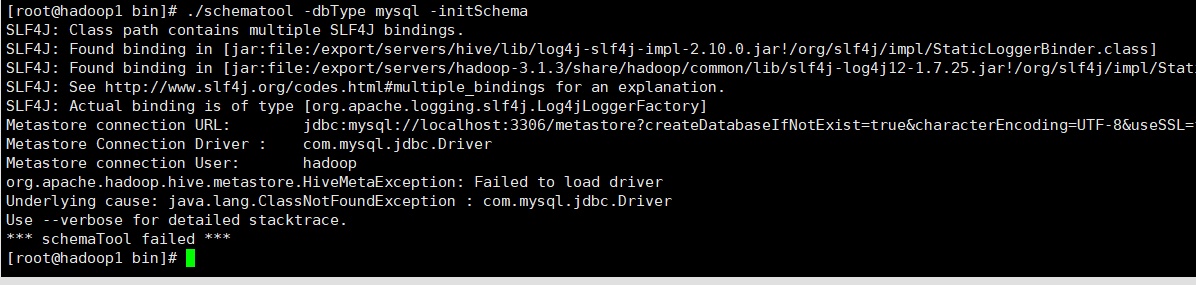
解决方法：for后面的非法字符删除。vi hive-site.xml +3215直接跳转到3215行。

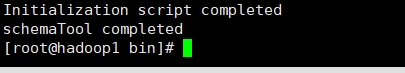


1. hive-site.xml配置出错。

原因：不是直接将配置文件复制到hive-site.xml中，而是在原hive-site.xml中修改。

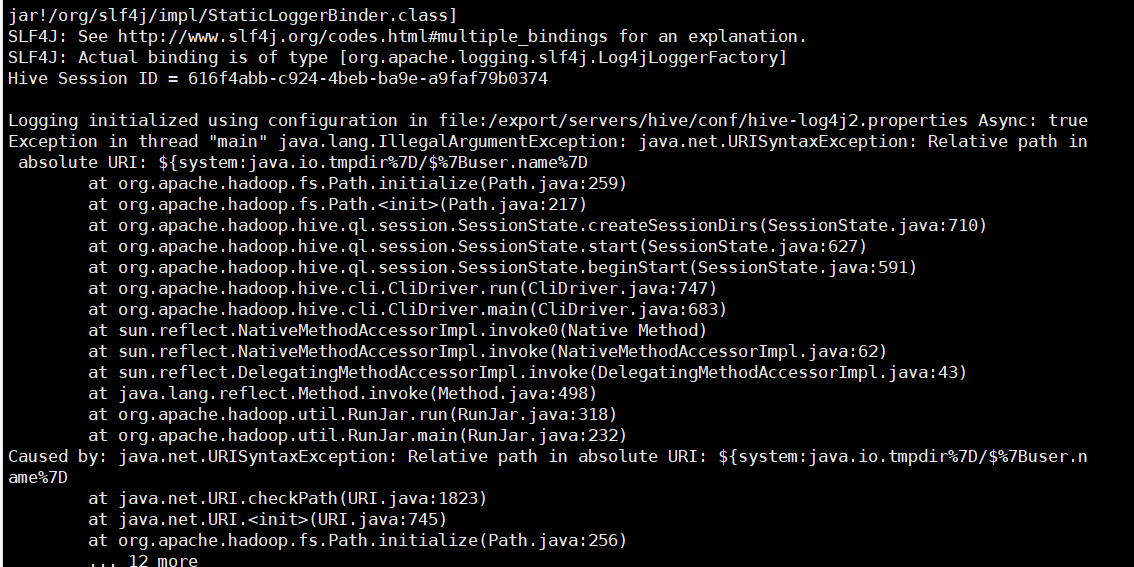
解决方法：按Esc + / +查找文件（例如hive.exec.scratchdir）跳转到相关行，修改value值。





1. Exception in thread "main" java.lang.IllegalArgumentException:  Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D

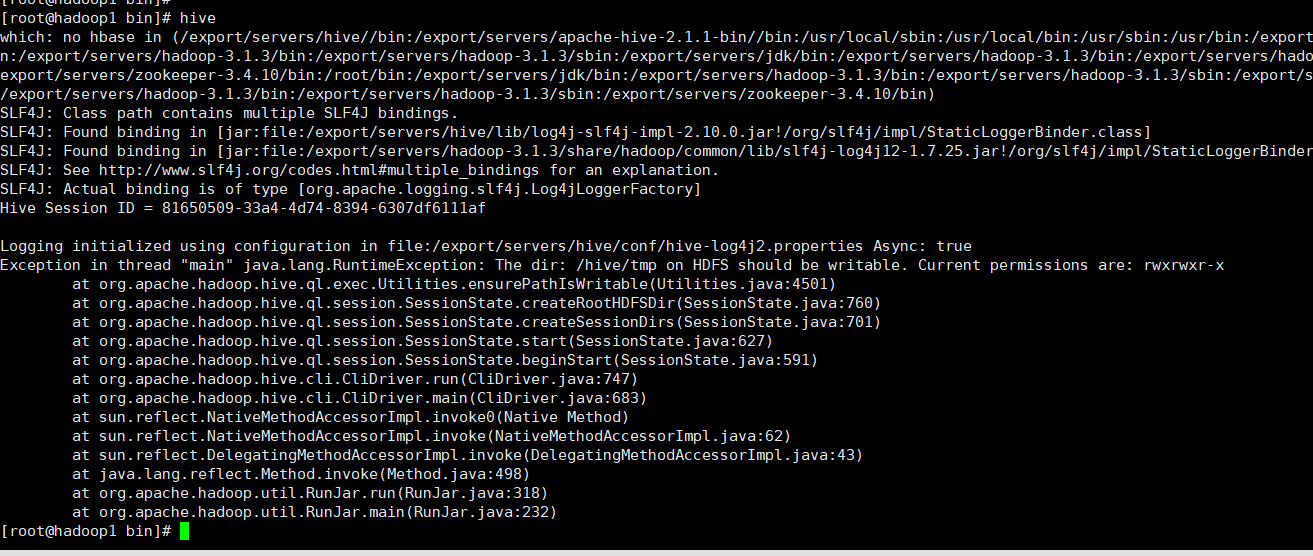
原因：路径出错。



解决方法：



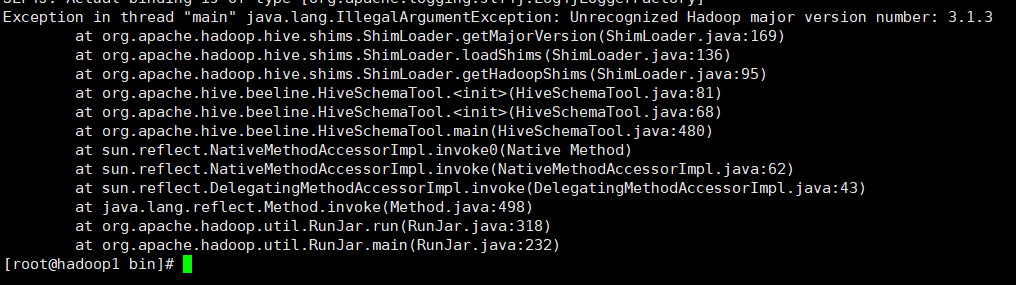
1. The root scratch dir:/hive/tmp on HDFS should be writable. Current permissions are: rwxr-xr-x.



原因：权限错误。

解决方法：hadoop fs -chmod -R 777 /hive/tmp

1. Unrecognized Hadoop major version number: 3.1.3

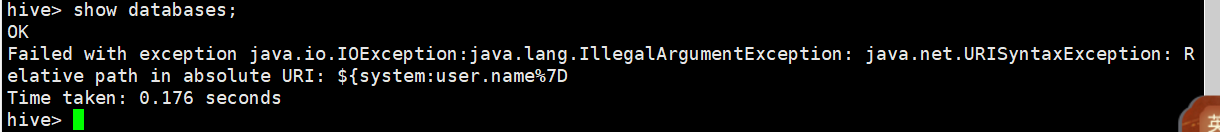


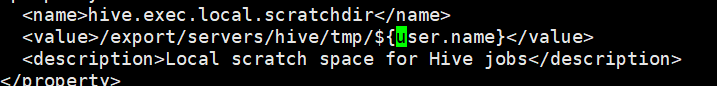
# 原因：hive与[hadoop](https://so.csdn.net/so/search?q=hadoop&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/m0_49945160/article/details/_blank)版本不兼容，必须同为2.x或同为3.x

解决：重新下载hive3.x的安装包。

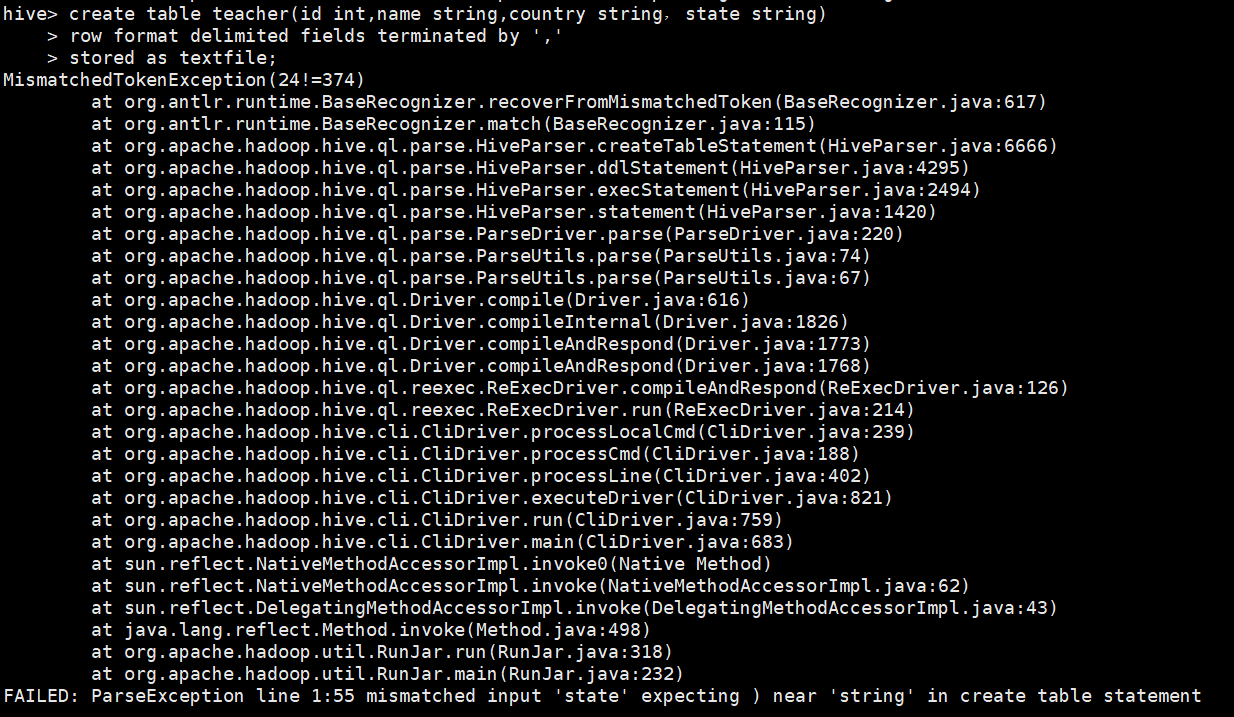
1. Hive数据操作
2. show database;报错。

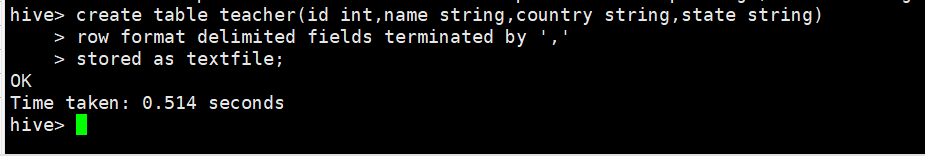
解决：删除” system: ”





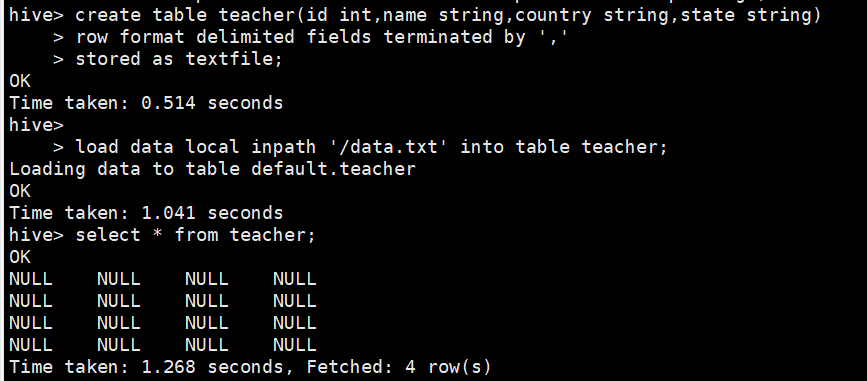
1. 标点符号中英文错误。





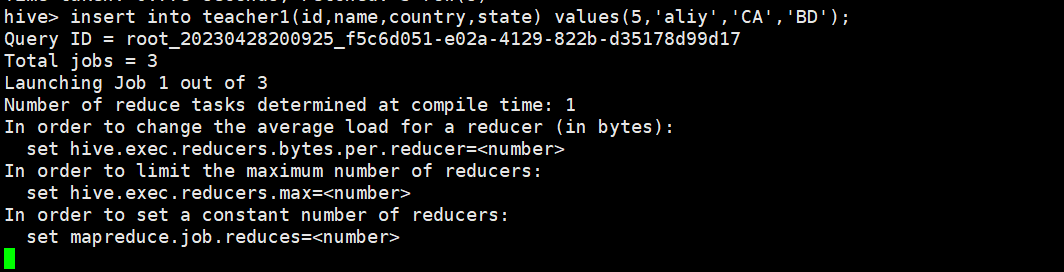
1. 装载数据报错。

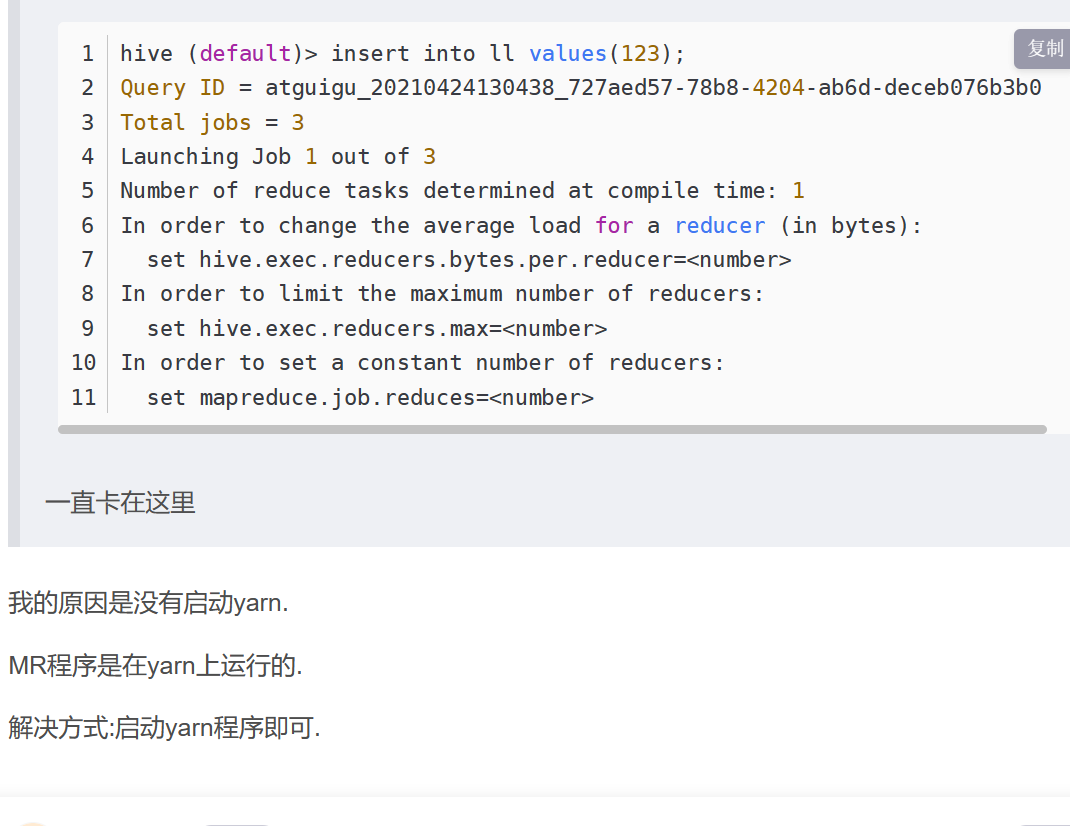
解决：将data.txt中的空格改为tab键

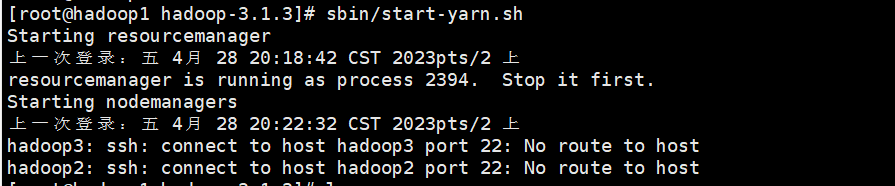


1. 插入数据时卡住不动。

解决：启动yarn。

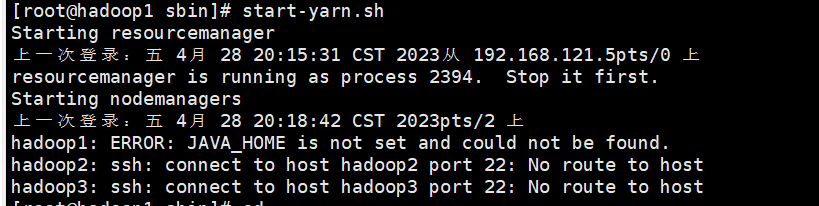






1. JAVA\_HOME配置出错。

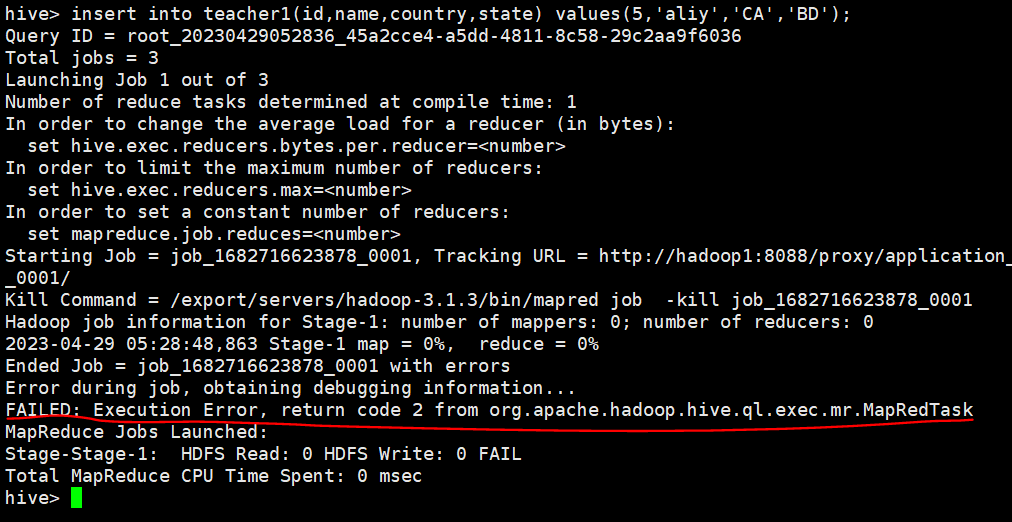
解决：修改JAVA\_HOEM路径。

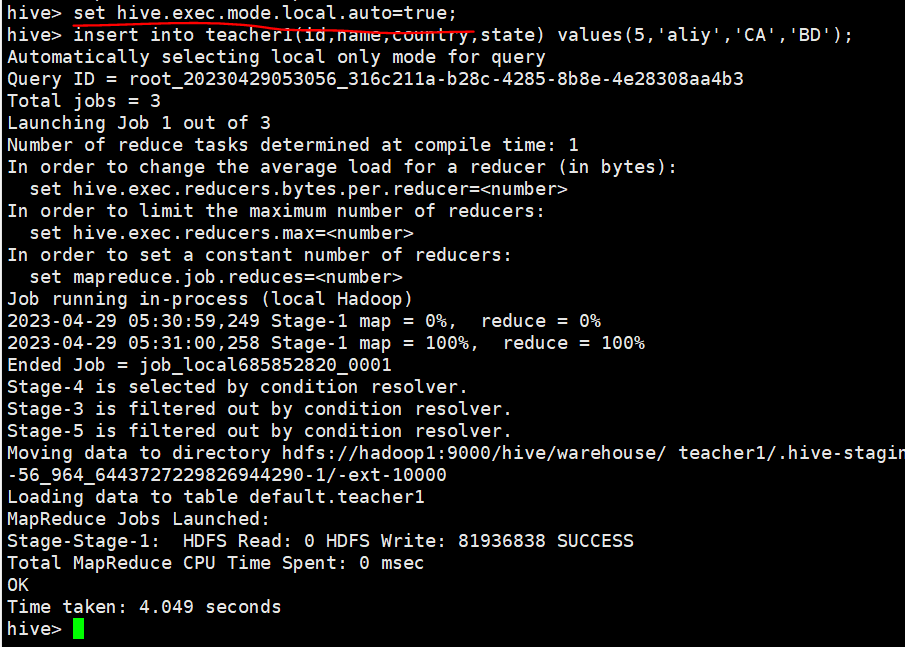




1. return code 2 from org.apache.hadoop.hive.ql.exec.mr.MapRedTask;

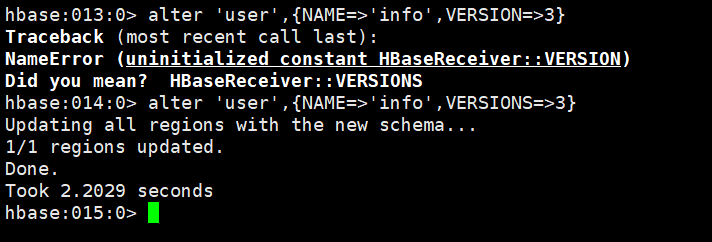
解决：set hive.exec.mode.local.auto=true;



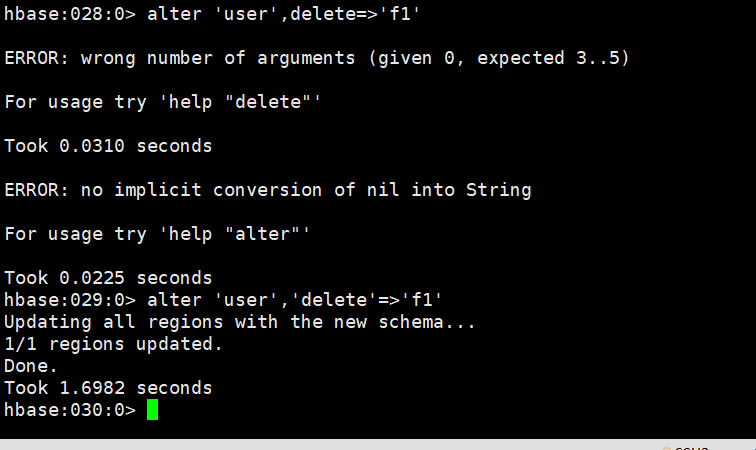


1. Hadoop的HBase Shell命令
2. 获取表报错。(拼写错误）

解决：VERSION改为VERSIONS



1. 删除表报错，漏了标点符号。



1. Hbase的Java API。
2. log4j报错。

解决:在resource下新建log4j.properties，添加内容：

log4j.rootLogger=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

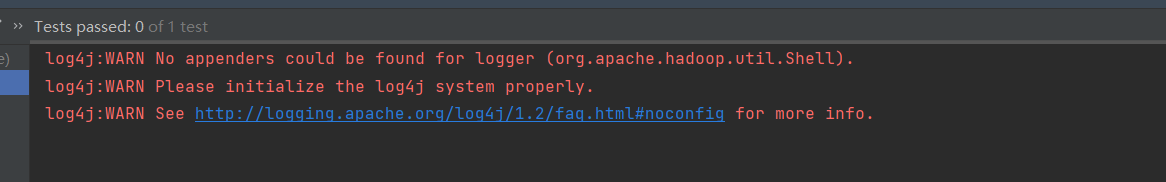
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

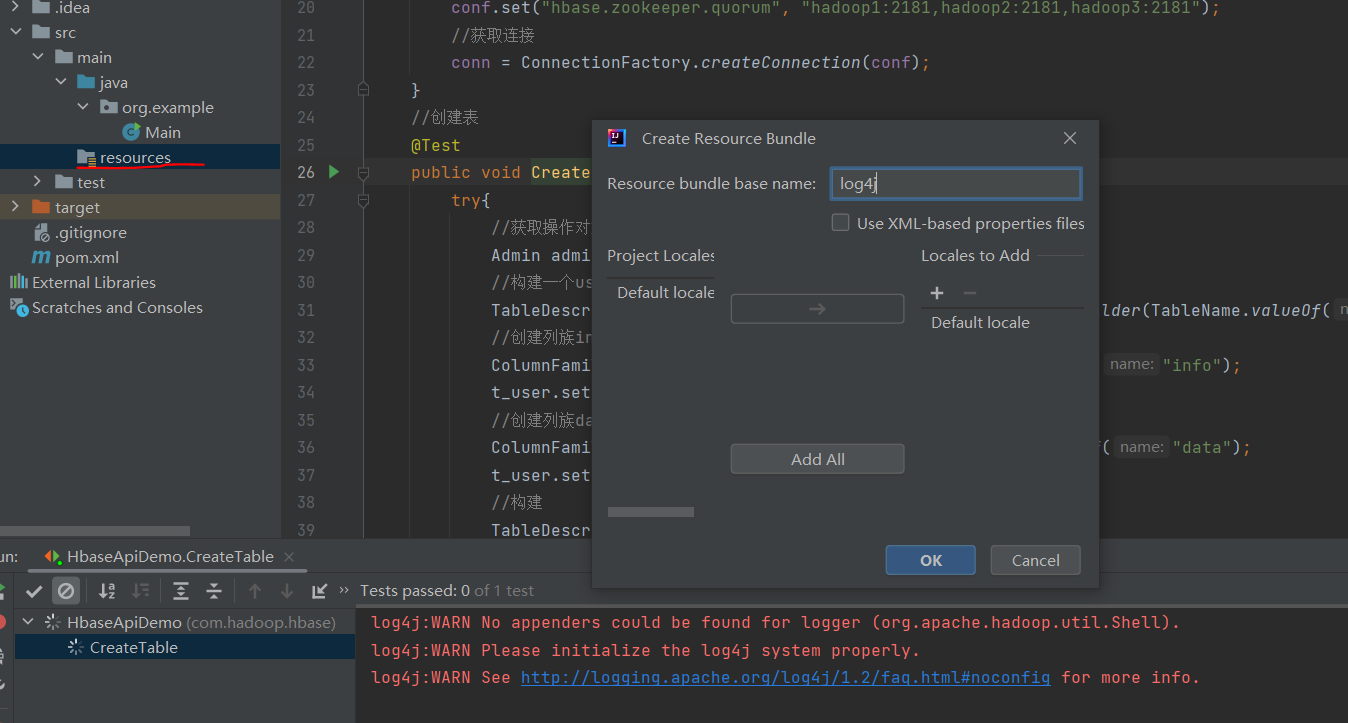
log4j.appender.logfile=org.apache.log4j.FileAppender

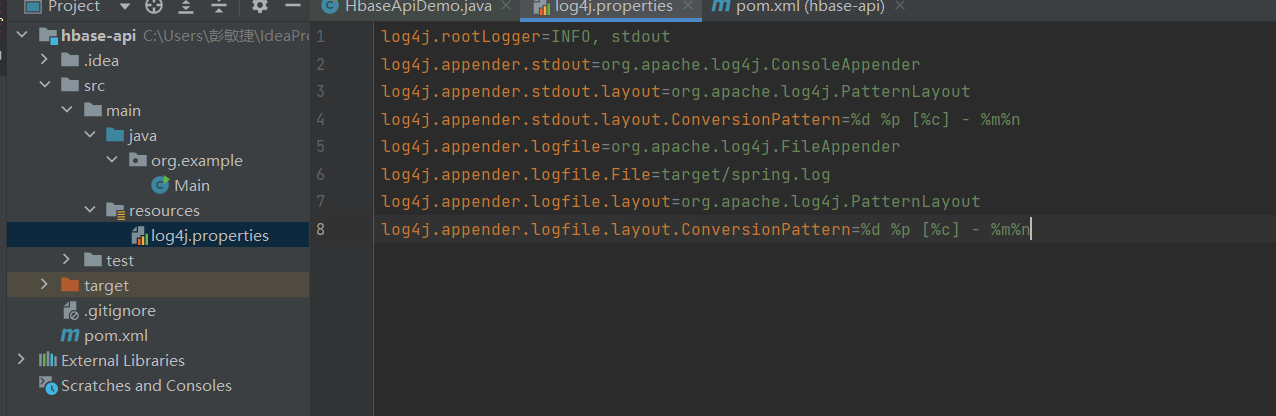
log4j.appender.logfile.File=target/spring.log

log4j.appender.logfile.layout=org.apache.log4j.PatternLayout

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n

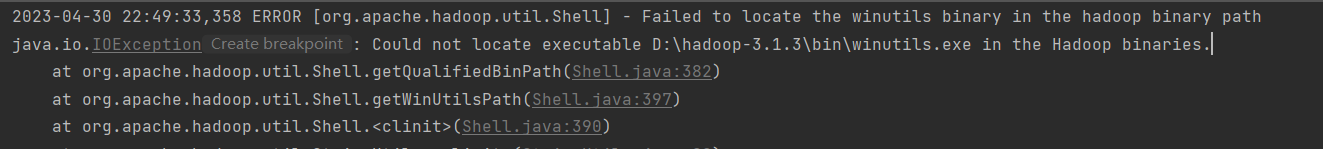


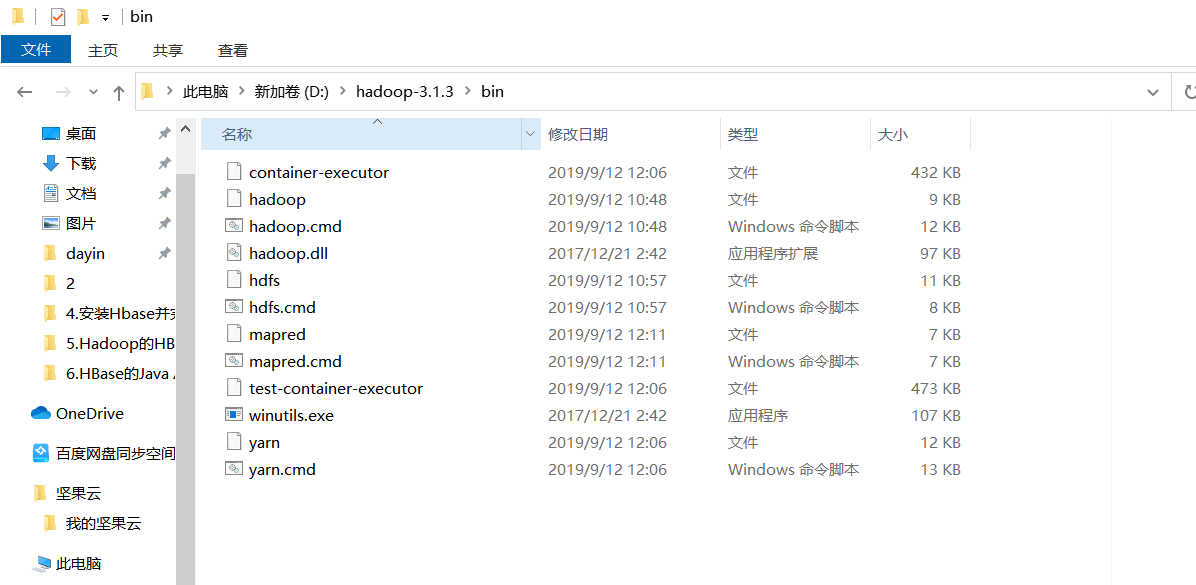




1. hadoop文件中没有winutils。

解决：添加winutils文件到指定目录下。





1. 运行后，一直不停。

原因：没有开启另外两个虚拟机。

解决：开启另外两个虚拟机。

