# 实验三 **熟悉常用的”HBase”操作**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | B10楼 | 303房； | **实验台号：** |  |
| **实验日期与时间：** | 2023/3/30 | | **评 分：** |  |
| **预习检查纪录：** |  | | **实验教师：** | 焦青松 |

1. **实验目的**

**（1）.理解HBase在Hadoop体系结构的角色。**

**(2).熟练使用HBase操作常用的Shell命令**

**(3).熟悉HBase操作常用的Java API。**

1. **实验平台**

(1).操作系统:Linux

(2).Hadoop版本:3.3.5

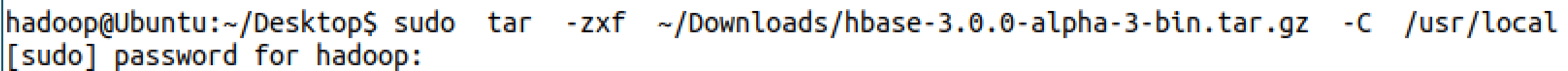
(3).Hbase版本：3.0.0

(4).JDK版本:19.3.0

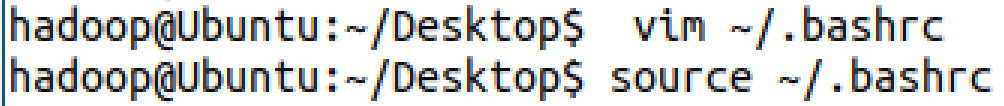
(5).Java IDE:eclipse

1. **实验步骤（包括实验结果/截图）**

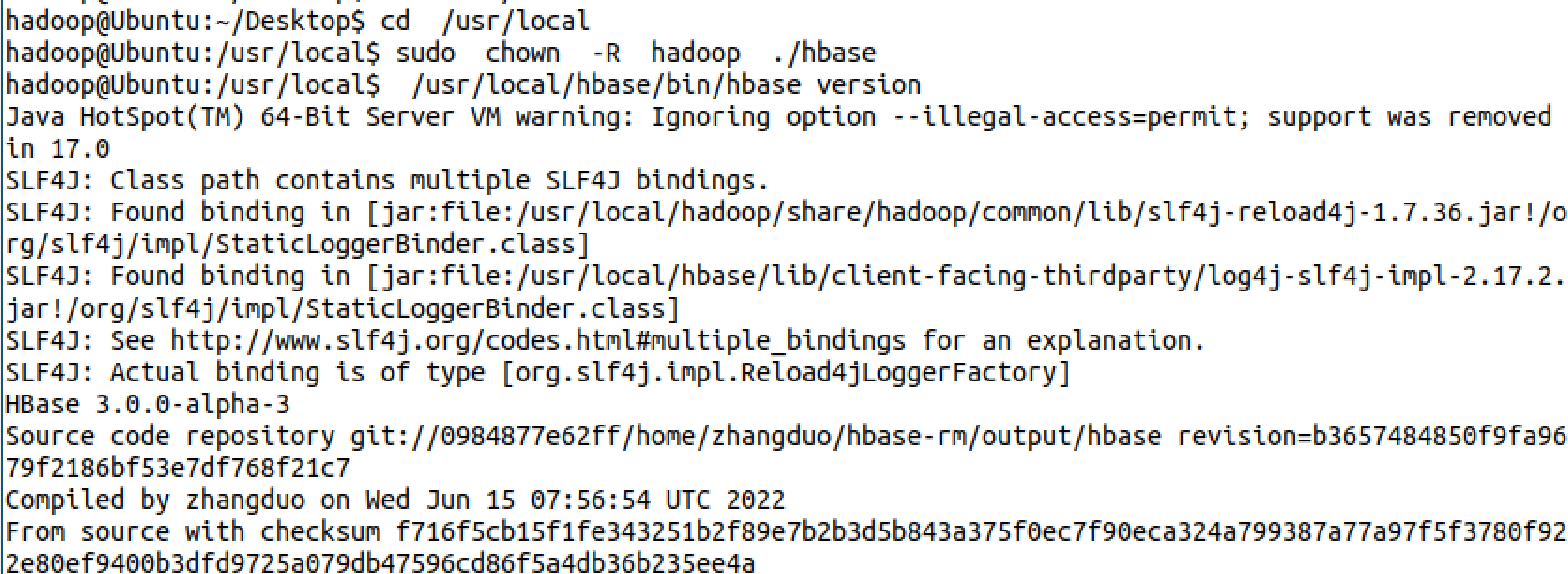
**Hbase的安装**

****

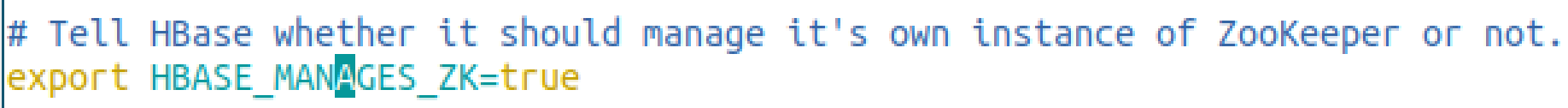
****

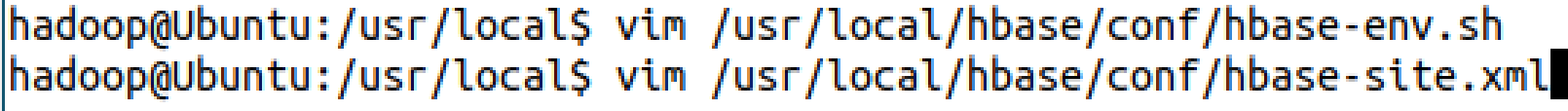
****

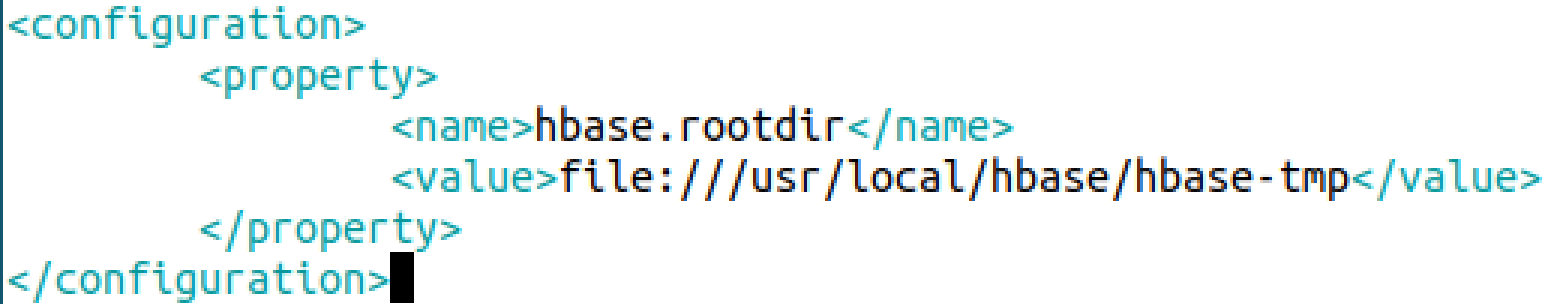
**查看版本**

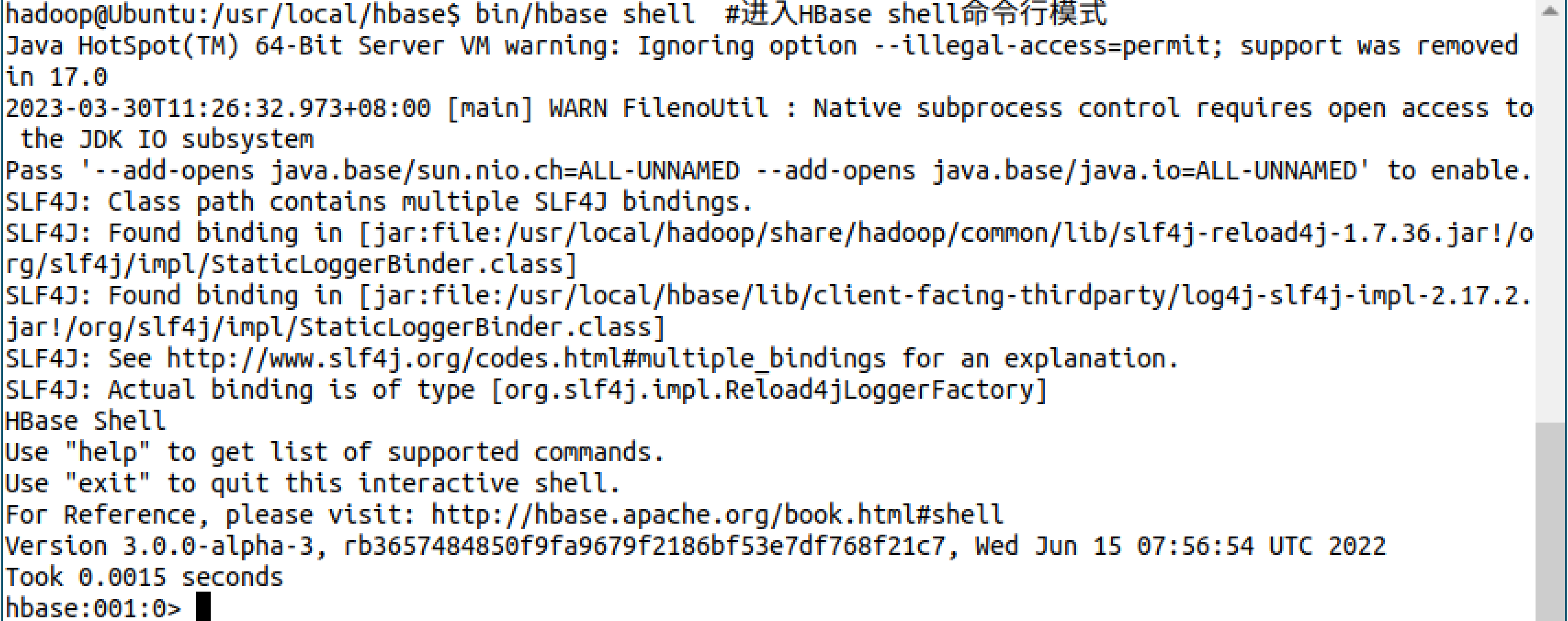
****

**单机模式配置**

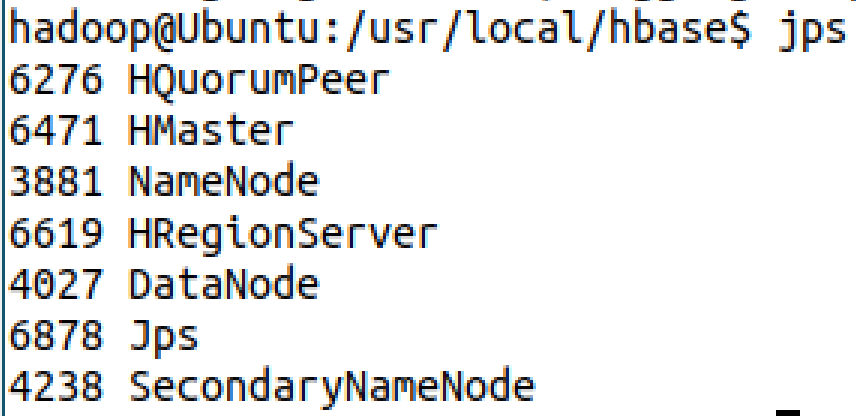
****

****

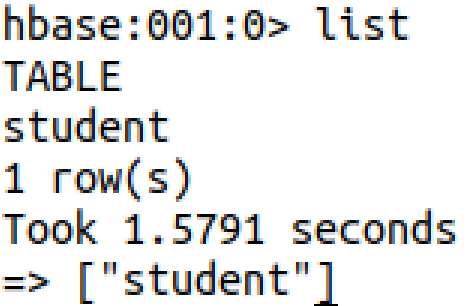
****

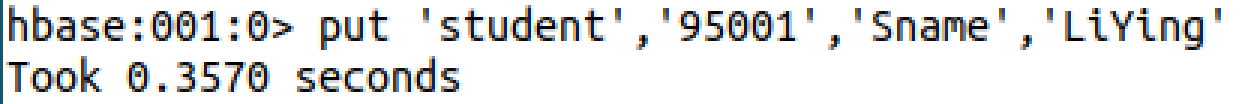
****

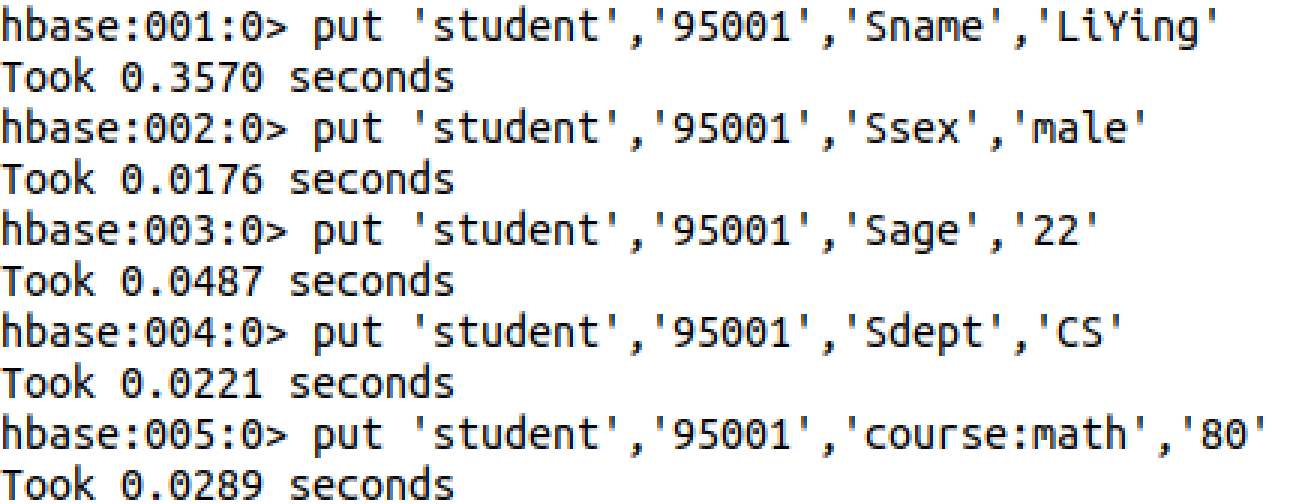
**Hbase配置成功**

****

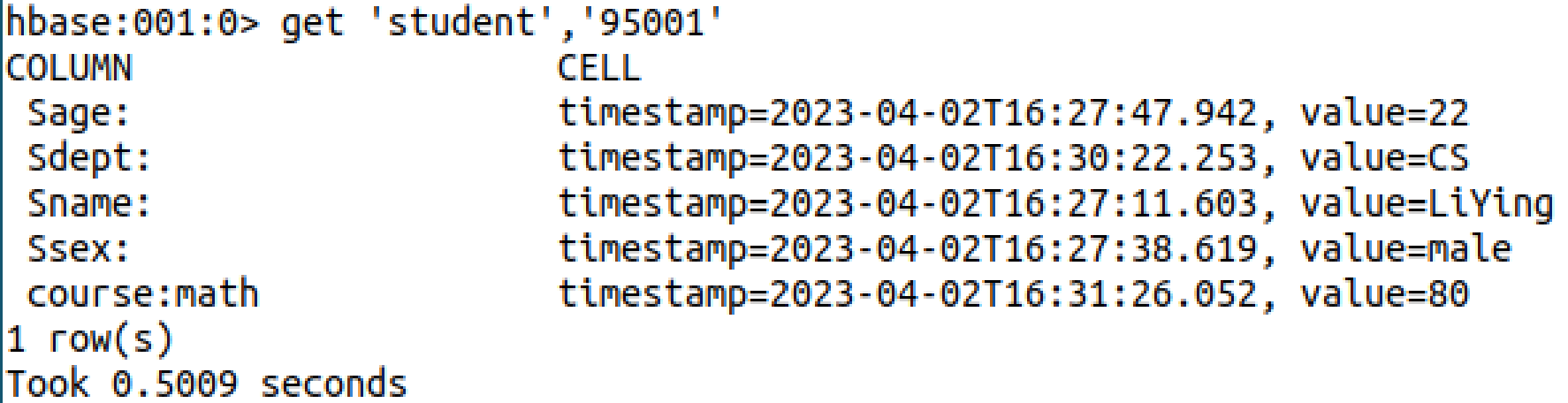
**Hbase创建表成功**

****

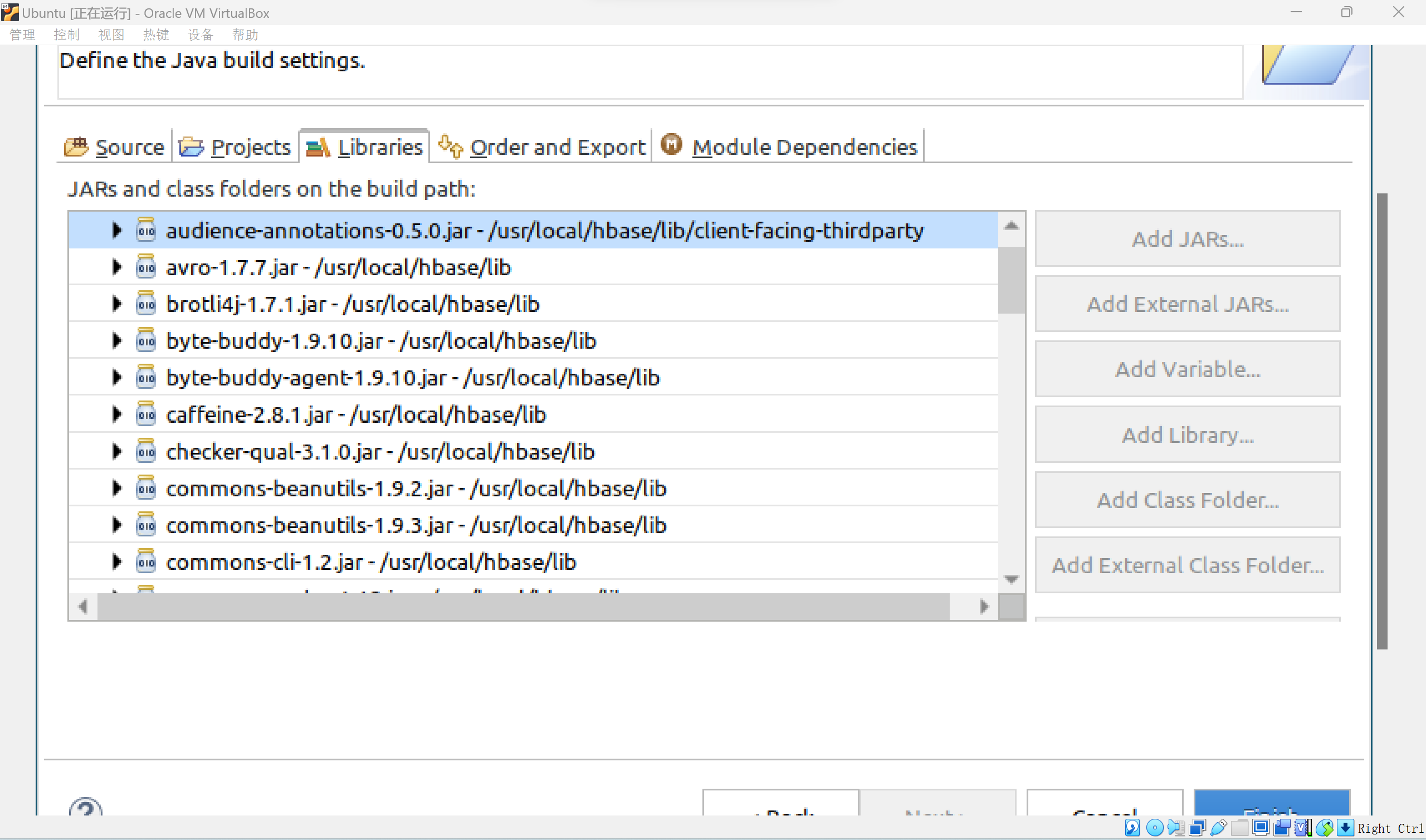
**添加数据**

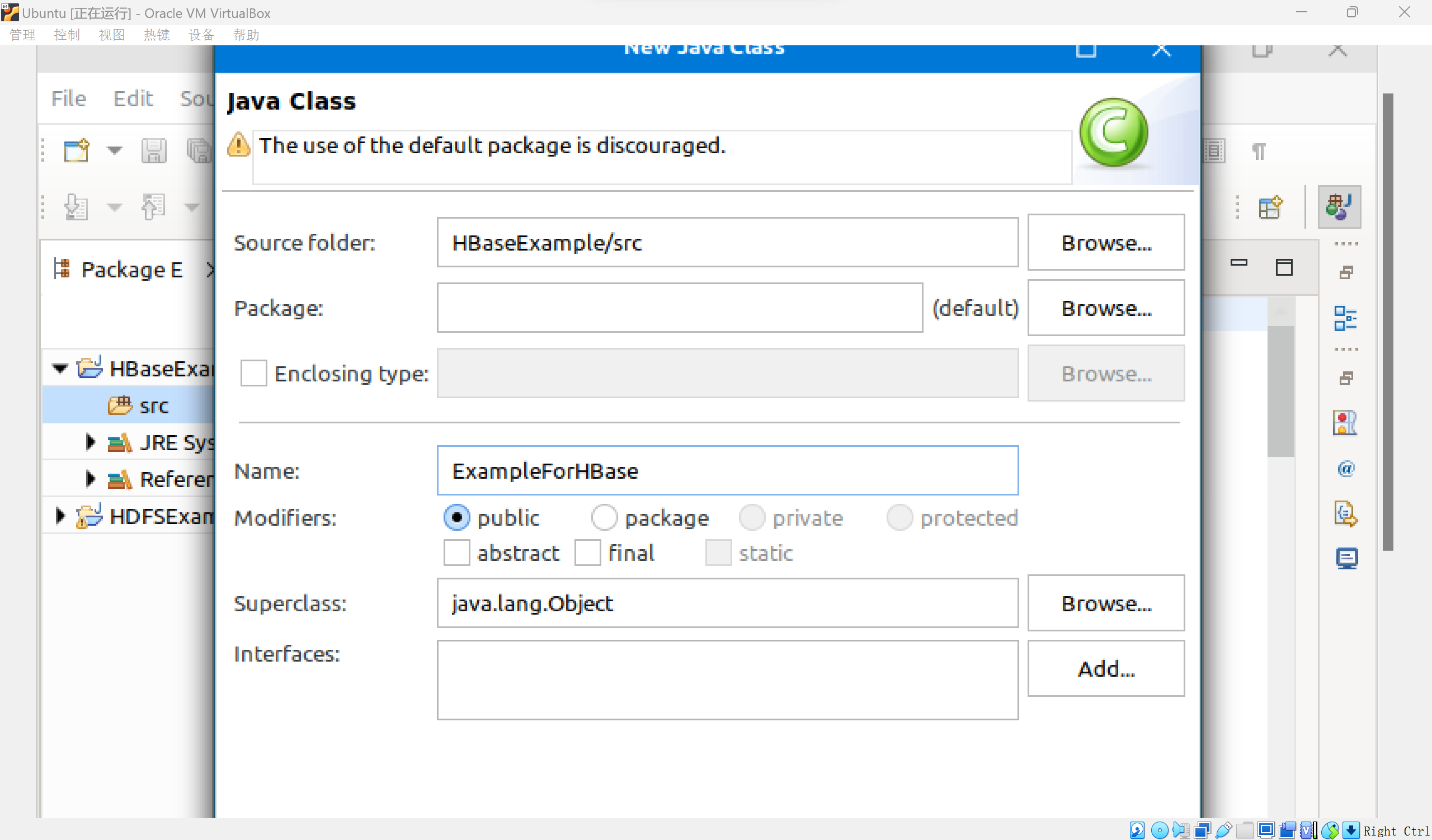
****

**查询数据**

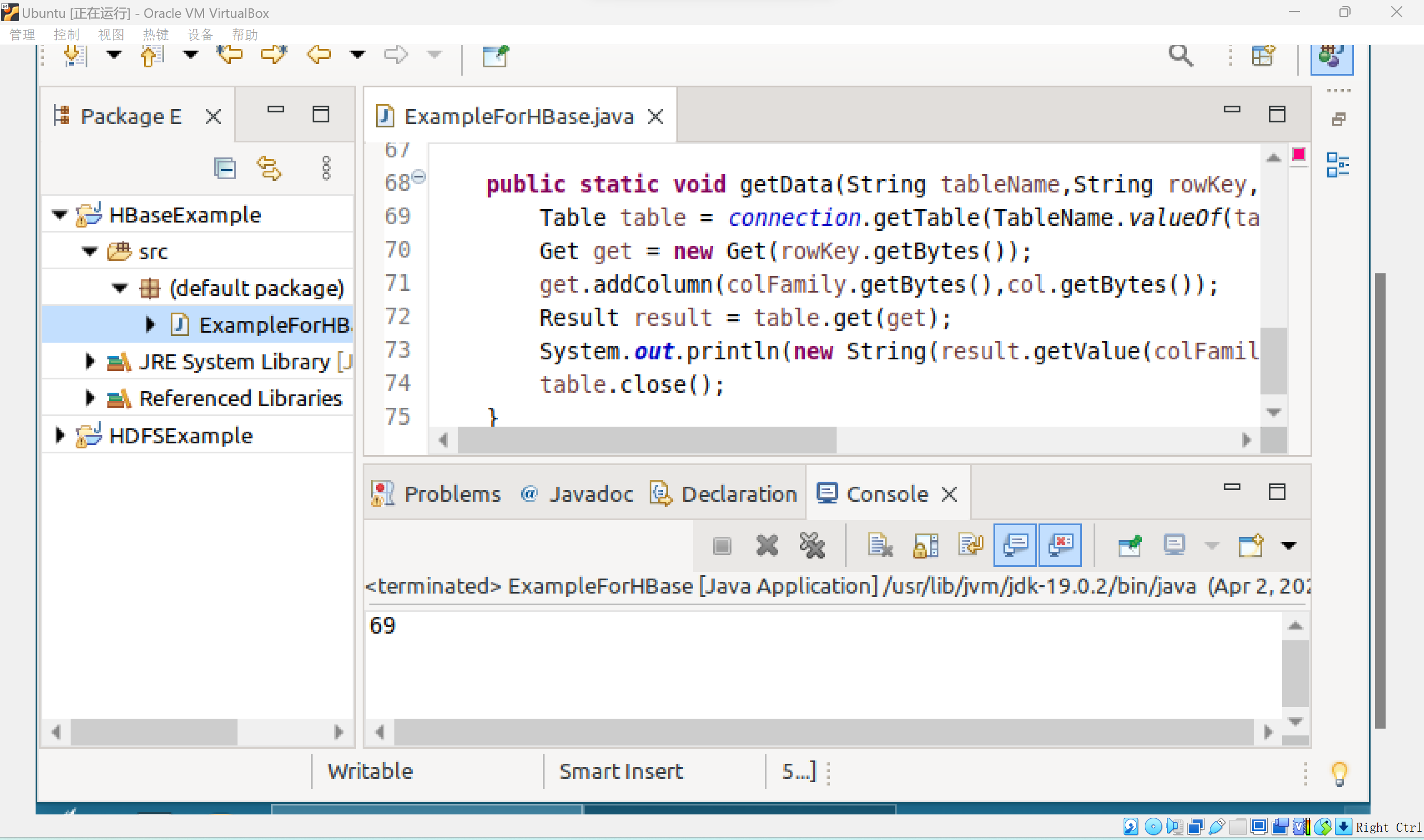
****

**编程实战**

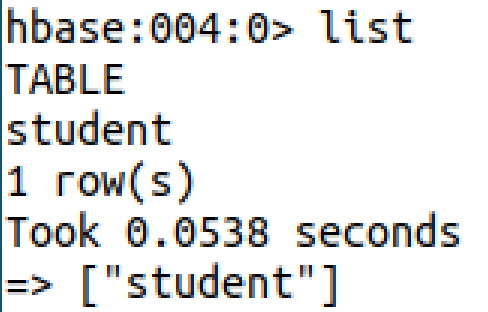
****

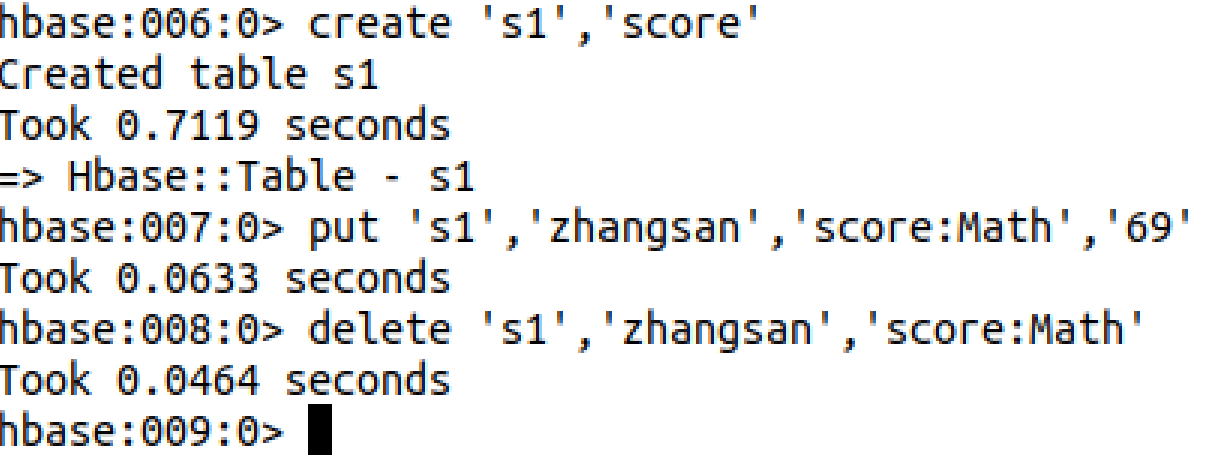
****

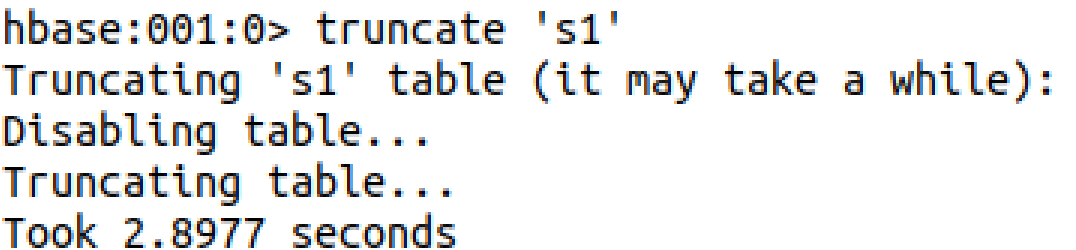
**运行结果为69**

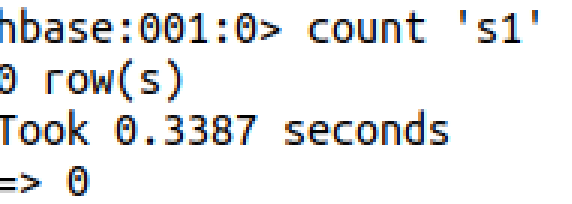
****

**编程实现功能**

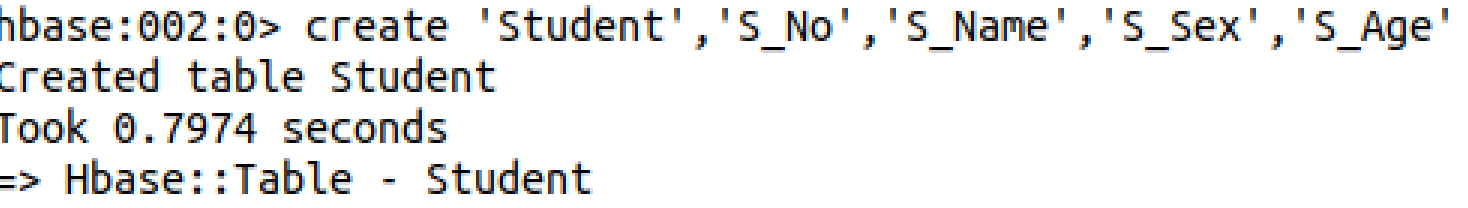
****

****

****

****

**Hbase数据库操作**

****

1. **思考讨论题或体会或对改进实验的建议**

可以使用Intellj IDEA作为项目IDE。

1. **实验源代码**

vim /usr/local/hbase/conf/hbase-env.sh

vim /usr/local/hbase/conf/hbase-site.xml

<configuration>

<property>

<name>hbase.rootdir</name>

<value>file:///usr/local/hbase/hbase-tmp</value>

</property>

</configuration>

cd /usr/local/hbase

bin/start-hbase.sh #启动HBase

bin/hbase shell #进入HBase shell命令行模式

cd /usr/local/hadoop

./sbin/start-dfs.sh

cd /usr/local/hbase

./bin/start-hbase.sh

./bin/hbase shell

hbase>create 'student','Sname','Ssex','Sage','Sdept','course'

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 java.io.IOException;

public class ExampleForHBase {

public static Configuration configuration;

public static Connection connection;

public static Admin admin;

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

init();

createTable("student",new String[]{"score"});

insertData("student","zhangsan","score","English","69");

insertData("student","zhangsan","score","Math","86");

insertData("student","zhangsan","score","Computer","77");

getData("student", "zhangsan", "score","English");

close();

}

public static void init(){

configuration = HBaseConfiguration.create();

configuration.set("hbase.rootdir","hdfs://localhost:9000/hbase");

try{

connection = ConnectionFactory.createConnection(configuration);

admin = connection.getAdmin();

}catch (IOException e){

e.printStackTrace();

}

}

public static void close(){

try{

if(admin != null){

admin.close();

}

if(null != connection){

connection.close();

}

}catch (IOException e){

e.printStackTrace();

}

}

public static void createTable(String myTableName,String[] colFamily) throws IOException {

TableName tableName = TableName.valueOf(myTableName);

if(admin.tableExists(tableName)){

System.out.println("talbe is exists!");

}else {

TableDescriptorBuilder tableDescriptor = TableDescriptorBuilder.newBuilder(tableName);

for(String str:colFamily){

ColumnFamilyDescriptor family =

ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes(str)).build();

tableDescriptor.setColumnFamily(family);

}

admin.createTable(tableDescriptor.build());

}

}

public static void insertData(String tableName,String rowKey,String colFamily,String col,String val) throws IOException {

Table table = connection.getTable(TableName.valueOf(tableName));

Put put = new Put(rowKey.getBytes());

put.addColumn(colFamily.getBytes(),col.getBytes(), val.getBytes());

table.put(put);

table.close();

}

public static void getData(String tableName,String rowKey,String colFamily, String col)throws IOException{

Table table = connection.getTable(TableName.valueOf(tableName));

Get get = new Get(rowKey.getBytes());

get.addColumn(colFamily.getBytes(),col.getBytes());

Result result = table.get(get);

System.out.println(new String(result.getValue(colFamily.getBytes(),col==null?null:col.getBytes())));

table.close();

}

}

public static void listTables() throws IOException {

init();//建立连接

List<TableDescriptor> tableDescriptors = admin.listTableDescriptors();

for(TableDescriptor tableDescriptor : tableDescriptors){

TableName tableName = tableDescriptor.getTableName();

System.out.println("Table:" + tableName);

}

close();//关闭连接

}

//在终端打印出指定的表的所有记录数据

public static void getData(String tableName)throws IOException{

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Scan scan = new Scan();

ResultScanner scanner = table.getScanner(scan);//获取行的遍历器

for (Result result:scanner){

printRecoder(result);

}

close();

}

//打印一条记录的详情

public static void printRecoder(Result result)throws IOException{

for(Cell cell:result.rawCells()){

System.out.print("行健: "+new String(Bytes.toString(cell.getRowArray(),cell.getRowOffset(), cell.getRowLength())));

System.out.print("列簇: "+new String( Bytes.toString(cell.getFamilyArray(),cell.getFamilyOffset(), cell.getFamilyLength()) ));

System.out.print(" 列: "+new String(Bytes.toString(cell.getQualifierArray(),cell.getQualifierOffset(), cell.getQualifierLength())));

System.out.print(" 值: "+new String(Bytes.toString(cell.getValueArray(),cell.getValueOffset(), cell.getValueLength())));

System.out.println("时间戳: "+cell.getTimestamp());

}

}

//向表添加数据

public static void insterRow(String tableName,String rowKey,String colFamily,String col,String val) throws IOException {

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Put put = new Put(rowKey.getBytes());

put.addColumn(colFamily.getBytes(), col.getBytes(), val.getBytes());

table.put(put);

table.close();

close();

}

//删除数据

public static void deleRow(String tableName,String rowKey,String colFamily,String col) throws IOException {

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Delete delete = new Delete(rowKey.getBytes());

//删除指定列族

delete.addFamily(Bytes.toBytes(colFamily));

//删除指定列

delete.addColumn(Bytes.toBytes(colFamily),Bytes.toBytes(col));

table.delete(delete);

table.close();

close();

}

//清空指定的表的所有记录数据

public static void clearRows(String tableName)throws IOException{

init();

TableName tablename = TableName.valueOf(tableName);

admin.disableTable(tablename);

admin.deleteTable(tablename);

TableDescriptorBuilder tableDescriptor = TableDescriptorBuilder.newBuilder(tablename);

admin.createTable(tableDescriptor.build());

close();

}

//统计表的行数

public static void countRows(String tableName)throws IOException{

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Scan scan = new Scan();

ResultScanner scanner = table.getScanner(scan);

int num = 0;

for (Result result = scanner.next();result!=null;result=scanner.next()){

num++;

}

System.out.println("行数:"+ num);

scanner.close();

close();

}

public static void createTable(String tableName,String[] fields) throws IOException {

init();

TableName tablename = TableName.valueOf(tableName);

if(admin.tableExists(tablename)){

System.out.println("table is exists!");

admin.disableTable(tablename);

admin.deleteTable(tablename);//删除原来的表

}

TableDescriptorBuilder tableDescriptor = TableDescriptorBuilder.newBuilder(tablename);

for(String str : fields){

tableDescriptor.setColumnFamily(ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes(str)).build());

}

admin.createTable(tableDescriptor.build());

close();

}

public static void addRecord(String tableName,String row,String[] fields,String[] values) throws IOException {

init();

Table table = connection.getTable(TableName.valueOf(tableName));

for(int i = 0;i != fields.length;i++){

Put put = new Put(row.getBytes());

String[] cols = fields[i].split(":");

put.addColumn(cols[0].getBytes(), cols[1].getBytes(), values[i].getBytes());

table.put(put);

}

table.close();

close();

}

public static void scanColumn(String tableName,String column)throws IOException{

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Scan scan = new Scan();

scan.addFamily(Bytes.toBytes(column));

ResultScanner scanner = table.getScanner(scan);

for (Result result = scanner.next(); result != null; result = scanner.next()){

showCell(result);

}

table.close();

close();

}

//格式化输出

public static void showCell(Result result){

Cell[] cells = result.rawCells();

for(Cell cell:cells){

System.out.println("RowName:"+new String(Bytes.toString(cell.getRowArray(),cell.getRowOffset(), cell.getRowLength()))+" ");

System.out.println("Timetamp:"+cell.getTimestamp()+" ");

System.out.println("column Family:"+new String(Bytes.toString(cell.getFamilyArray(),cell.getFamilyOffset(), cell.getFamilyLength()))+" ");

System.out.println("row Name:"+new String(Bytes.toString(cell.getQualifierArray(),cell.getQualifierOffset(), cell.getQualifierLength()))+" ");

System.out.println("value:"+new String(Bytes.toString(cell.getValueArray(),cell.getValueOffset(), cell.getValueLength()))+" ");

}

}

public static void scanColumn(String tableName,String column)throws IOException{

init();

Table table = connection.getTable(TableName.valueOf(tableName));

Scan scan = new Scan();

scan.addFamily(Bytes.toBytes(column));

ResultScanner scanner = table.getScanner(scan);

for (Result result = scanner.next(); result != null; result = scanner.next()){

showCell(result);

}

table.close();

close();

}

//格式化输出

public static void showCell(Result result){

Cell[] cells = result.rawCells();

for(Cell cell:cells){

System.out.println("RowName:"+new String(Bytes.toString(cell.getRowArray(),cell.getRowOffset(), cell.getRowLength()))+" ");

System.out.println("Timetamp:"+cell.getTimestamp()+" ");

System.out.println("column Family:"+new String(Bytes.toString(cell.getFamilyArray(),cell.getFamilyOffset(), cell.getFamilyLength()))+" ");

System.out.println("row Name:"+new String(Bytes.toString(cell.getQualifierArray(),cell.getQualifierOffset(), cell.getQualifierLength()))+" ");

System.out.println("value:"+new String(Bytes.toString(cell.getValueArray(),cell.getValueOffset(), cell.getValueLength()))+" ");

}

}