

**数学与信息学院学生实验报告**

**实验课程名称：** 大数据技术概论 **教师： \_\_**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **NoSQL和关系数据库的操作比较** | | | **实验成绩** |  |
| **学生姓名** |  | **学 号** |  | **年级专业班级** |  |
| **小组成员** | **无** | | | **实验日期** |  |

# 1. 实验目的和要求

## 1.1 实验目的

* 理解四种数据库(MySQL、HBase、Redis和MongoDB)的概念以及不同点；
* 熟练使用四种数据库操作常用的Shell命令；
* 熟悉四种数据库操作常用的Java API。

## 1.2 实验软硬件环境

* 操作系统： Ubuntu19.04；
* Hadoop版本：3.2.0；
* MySQL版本：8.0；
* HBase版本：2.2.1；
* Redis版本：5.05；
* MongoDB版本：3.0.6；
* JDK版本： jdk-13；
* Java IDE：Eclipse；

## 1.3 实验要求

①MySQL数据库操作

②HBase数据库操作

③编程实现相应的功能

④MongoDB数据库操作

# 2.实验记录

## 2.1 MySQL数据库操作

**由于原来电脑的windows系统已经安装配置过mysql，这里直接使用，而且现在的mysql已经不需要向以前一样复杂的安装配置了，只需要在官网下载安装程序，直接一直下一步，在后面勾选安装服务及workbench，安装好后就可以直接打开图形界面使用了。**

学生表Student

|  |  |  |  |
| --- | --- | --- | --- |
| Name | English | Math | Computer |
| zhangsan | 69 | 86 | 77 |
| lisi | 55 | 100 | 88 |

1. 根据上面给出的Student表，在MySQL数据库中完成如下操作：
2. 在MySQL中创建Student表，并录入数据；

use shiyan4;

create table student(

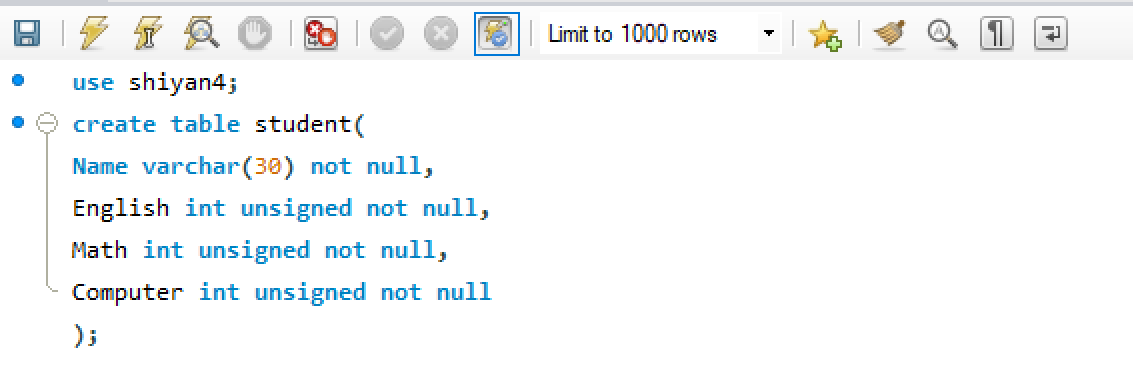
Name varchar(30) not null,

English int unsigned not null,

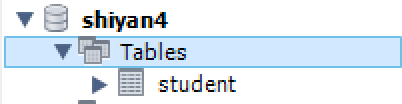
Math int unsigned not null,

Computer int unsigned not null

);



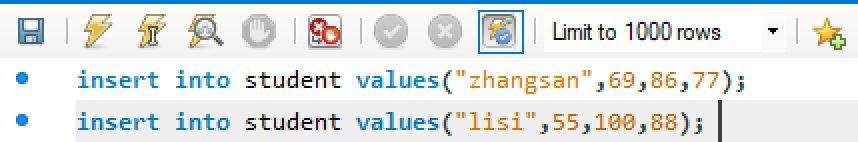
**执行后刷新，可见student表创建成功**



**插入数据**

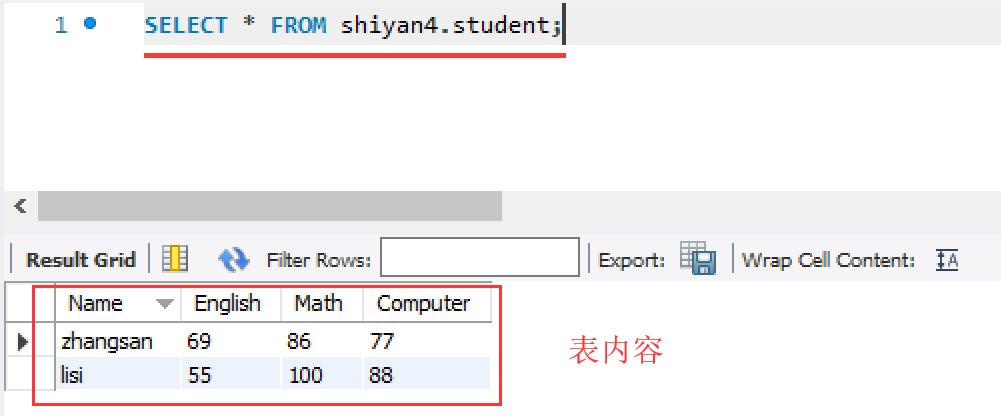
**insert into student values("zhangsan",69,86,77);**

**insert into student values("lisi",55,100,88);**



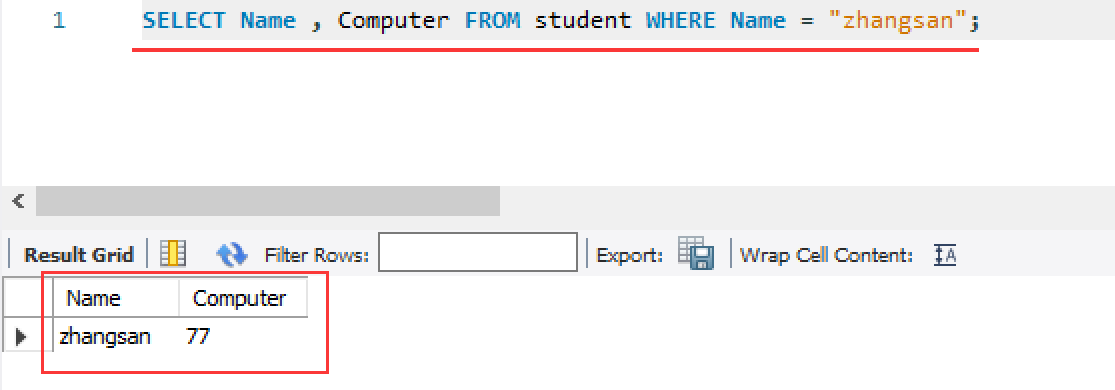
1. 用SQL语句输出Student表中的所有记录；

**SELECT \* FROM shiyan4.student;**



1. 查询zhangsan的Computer成绩；

**SELECTName，Computer FROM student WHERE Name=”zhangsan”**

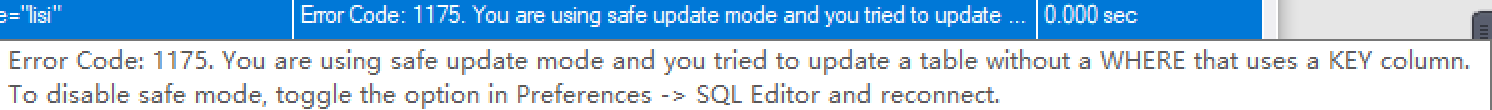


1. 修改lisi的Math成绩，改为95。

**UPDATE student Set Math=95 WHERE Name=”lisi”**



**报错：**



**说明建表时没有设置主键，**

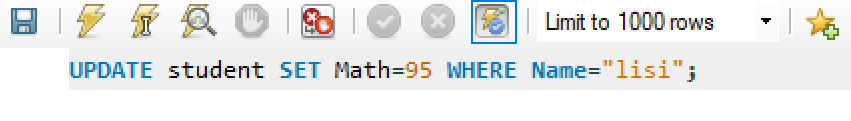
**给表添加主键**

**alter table student add primary key (Name);**



**再执行**

**UPDATE student SET Math=95 WHERE Name="lisi";**

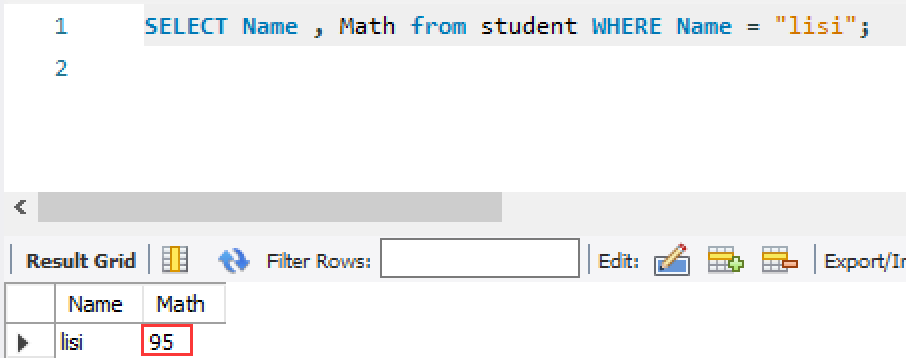


**执行成功**



**使用SELECT语句，查看修改结果：**

**SELECT Name , Math from student WHERE Name = "lisi";**



1. 根据上面已经设计出的Student表，使用MySQL的JAVA客户端编程实现以下操作：

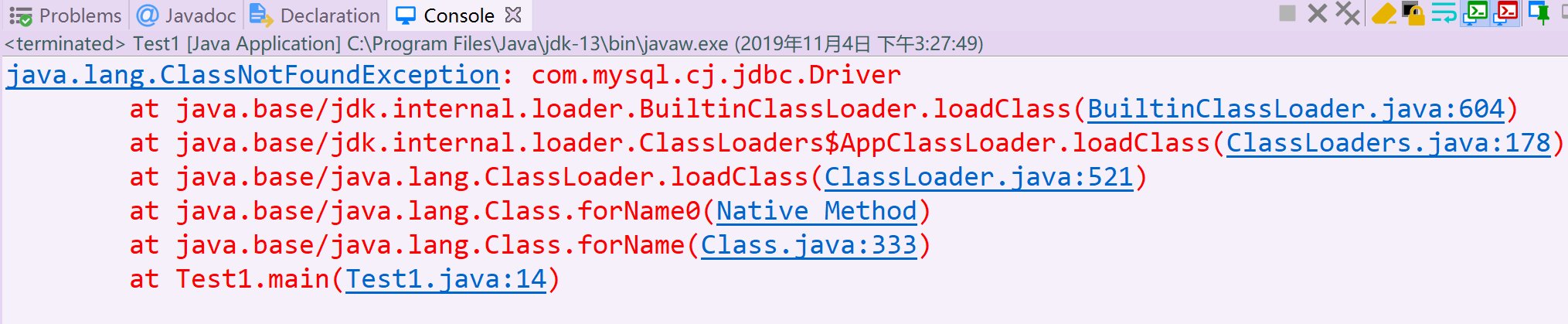
（1）向Student表中添加如下所示的一条记录：

|  |  |  |  |
| --- | --- | --- | --- |
| scofield | 45 | 89 | 100 |

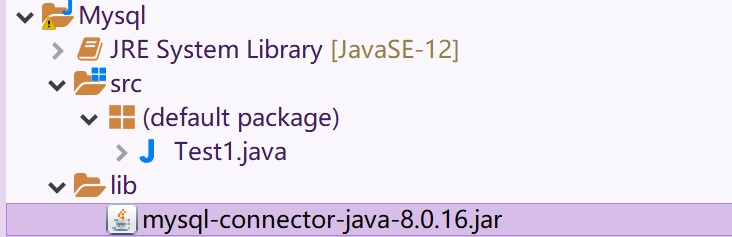
**备注：**

**执行java代码前记得导入mysql的包，我用的是mysql-connector-java-8.0.16**

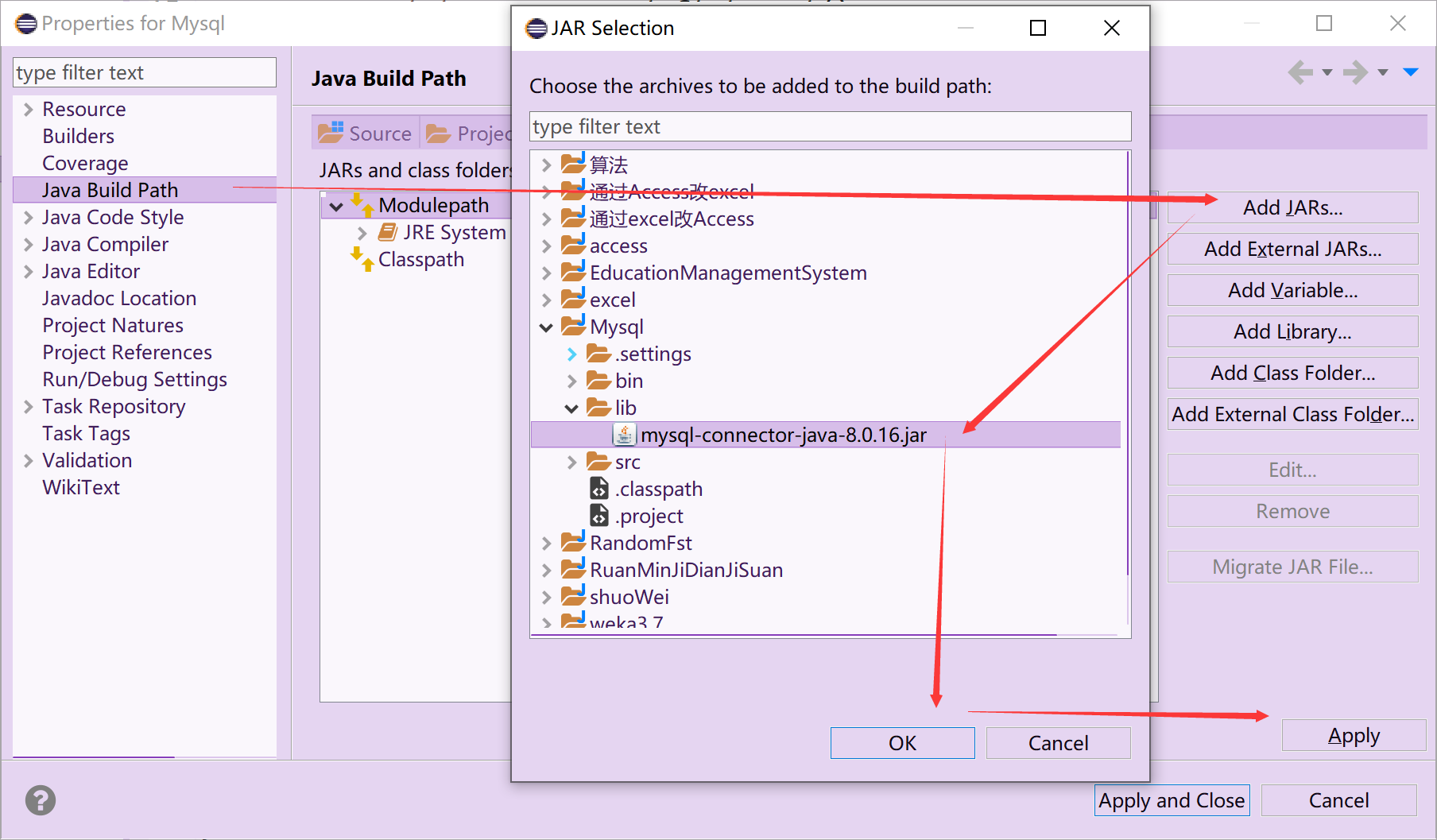
**如果不导包，会报错下面的错误：**



**我这里新建一个lib文件夹，把包（包是自己上网下载的）放进去**



**修改buildpath**



**Java代码：**

**import** java.sql.\*;

**publicclass** Test1 {

**staticfinal** String ***DRIVER*** = "com.mysql.cj.jdbc.Driver";

**staticfinal** String ***DB*** = "jdbc:mysql://localhost:3306/shiyan4?serverTimezone=UTC"; // 数据库

**staticfinal** String ***USER*** = "root";

**staticfinal** String ***PASSWD*** = "lcr123456";

**publicstaticvoid** main(String[] args) {

Connection conn = **null**;

Statement stmt = **null**;

**try** { // 加载驱动程序

Class.*forName*(***DRIVER***);

System.***out***.println("正在连接数据库..."); // 打开一个连接

conn = DriverManager.*getConnection*(***DB***, ***USER***, ***PASSWD***); // 执行一个查询

stmt = conn.createStatement();

String sql = "insert into student values('scofield',45,89,100)";

stmt.executeUpdate(sql);

System.***out***.println("插入成功");

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

} **finally** {

**if** (stmt != **null**)

**try** {

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**if** (conn != **null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

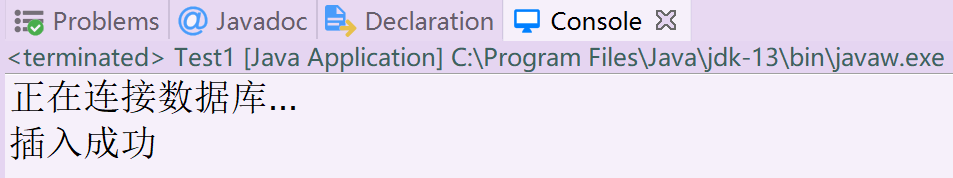
}

}

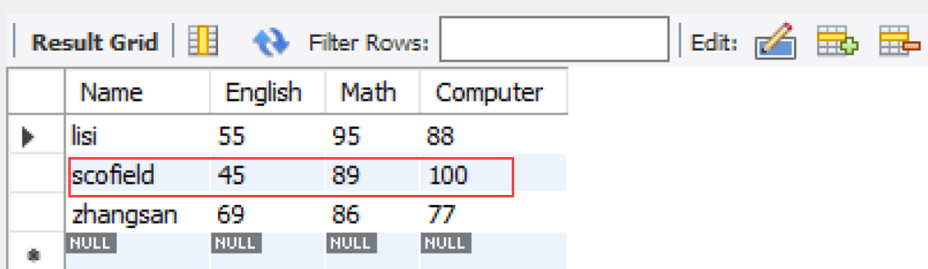
}

**运行结果：**

**控制台：**



**数据库：**



**可见，插入成功**

（2）获取scofield的English成绩信息

**java代码：**

**import** java.sql.\*;

**publicclass** Test2 {

**staticfinal** String ***DRIVER*** = "com.mysql.cj.jdbc.Driver";

**staticfinal** String ***DB*** = "jdbc:mysql://localhost:3306/shiyan4?serverTimezone=UTC"; // 数据库

**staticfinal** String ***USER*** = "root";

**staticfinal** String ***PASSWD*** = "lcr123456";

**publicstaticvoid** main(String[] args) {

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

**try** { // 加载驱动程序

Class.*forName*(***DRIVER***);

System.***out***.println("正在连接数据库..."); // 打开一个连接

conn = DriverManager.*getConnection*(***DB***, ***USER***, ***PASSWD***);// 执行一个查询

stmt = conn.createStatement();

String sql = "select name,English from student where name='scofield' ";

rs = stmt.executeQuery(sql); //执行语句，获得结果

System.***out***.println("name" + " " + "English");

**while** (rs.next()) {

System.***out***.printf("%-10s%-10d",rs.getString(1),rs.getInt(2));

}

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

} **finally** {

**if** (rs != **null**)

**try** {

rs.close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**if** (stmt != **null**)

**try** {

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**if** (conn != **null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

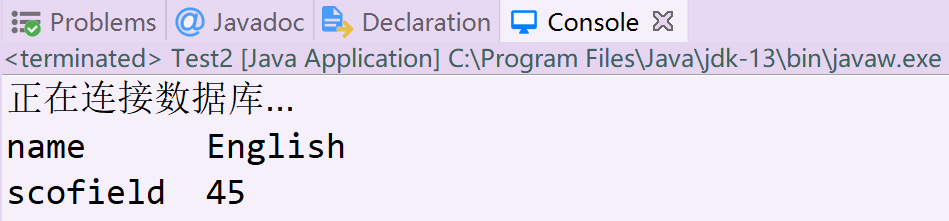
}

}

}

}

**运行结果：**



**使用System.out.printf("%-10s%-10d",rs.getString(1),rs.getInt(2));使得输出结果对齐**

**意思是：掉用printf输出格式，-代表左边对齐，10代表占10位，不够的用空格补全。**

## 2.2 HBase数据库操作

学生表Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | score | | | |
| English | Math | Computer | |
| zhangsan | 69 | 86 | | 77 |
| lisi | 55 | 100 | | 88 |

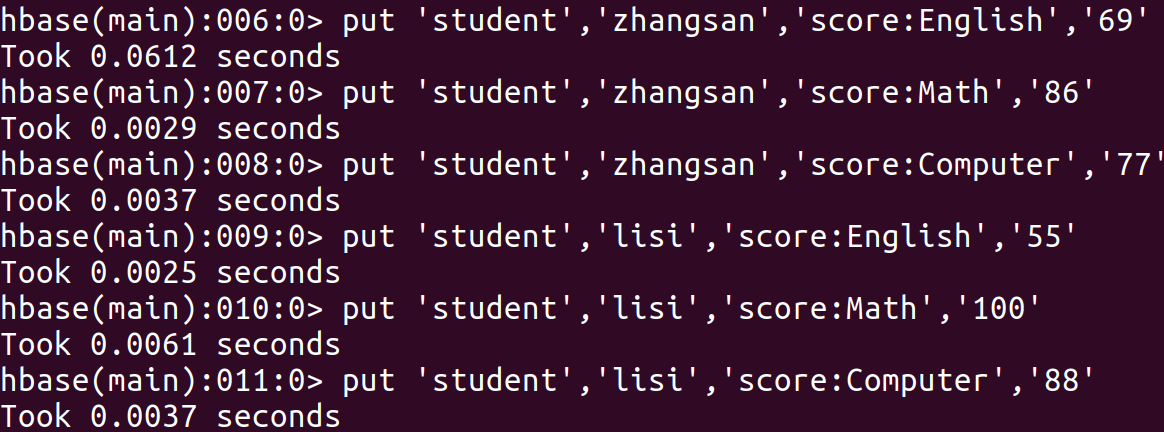
1.根据上面给出的学生表Student的信息，执行如下操作：

1. 用Hbase Shell命令创建学生表Student；

**创建表：create第一个参数位表名，后面为列/列族名**

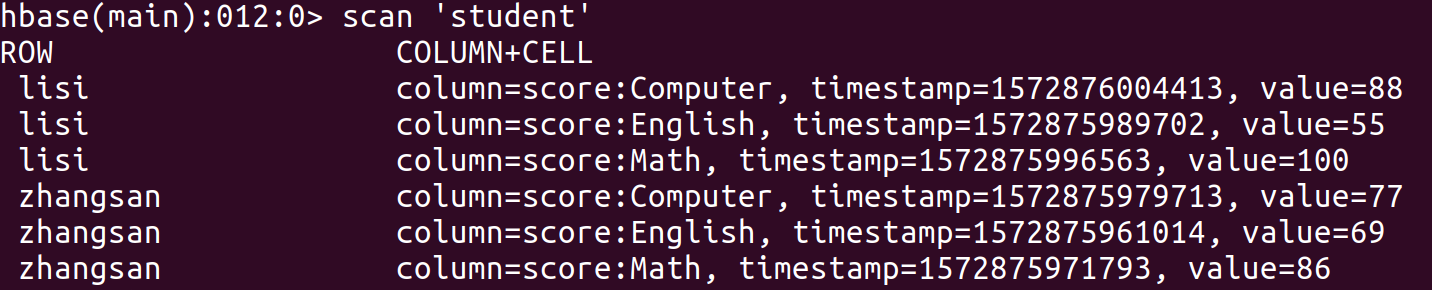


**给表添加数据：put ‘表名’,’行键’,’列名’,’值’**



1. 用scan命令浏览Student表的相关信息；

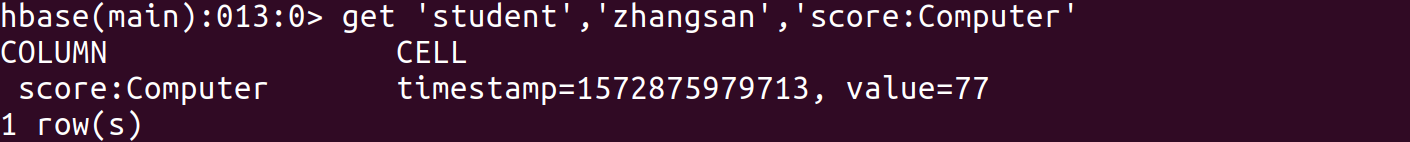
**scan ‘表名’**



**ROW代表行键，column列，timestamp：时间戳（默认使用的是数据添加时的时间，但是也可以使用put方法添加数据的时候手动添加，具体在后面的总结中有详细的描述。）value：值。**

1. 查询zhangsan的Computer成绩；

**get ‘ 表名’,’行键’,’列名’**

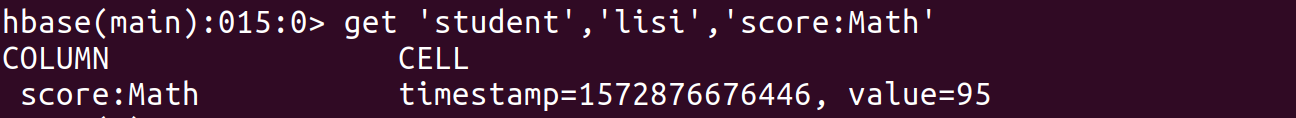


（4）修改lisi的Math成绩，改为95。

**使用Put，如有有数据就会修改，没有就会插入数据。**



**使用get查看修改结果：**



2.根据上面已经设计出的Student表，用HBase API编程实现以下操作：

（1）添加数据：English:45 Math:89 Computer:100

|  |  |  |  |
| --- | --- | --- | --- |
| scofield | 45 | 89 | 100 |

**Java代码：**

**package** hbase;

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configuration;

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

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

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

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

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

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

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

**publicclass** Test4\_1 {

**publicstatic** Configuration *configuration*;

**publicstatic** Connection *connection*;

**publicstatic** Admin *admin*;

**publicstaticvoid** main(String[] args) {

**try** {

*insertRow*("student", "scofield", "score", "English", "45");

*insertRow*("student", "scofield", "score", "Math", "89");

*insertRow*("student", "scofield", "score", "Computer", "100");

} **catch** (IOException e) {

e.printStackTrace();

}

}

**publicstaticvoid** insertRow(String tableName, String key, String colFamily, String col, String val)

**throws** IOException {

*init*();

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

Put put = **new** Put(key.getBytes());

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

table.put(put);

System.***out***.println("插入成功");

table.close();

*close*();

}

**publicstaticvoid** 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();

}

}

**publicstaticvoid** close() {

**try** {

**if** (*admin* != **null**) {

*admin*.close();

}

**if** (**null** != *connection*) {

*connection*.close();

}

} **catch** (IOException e) {

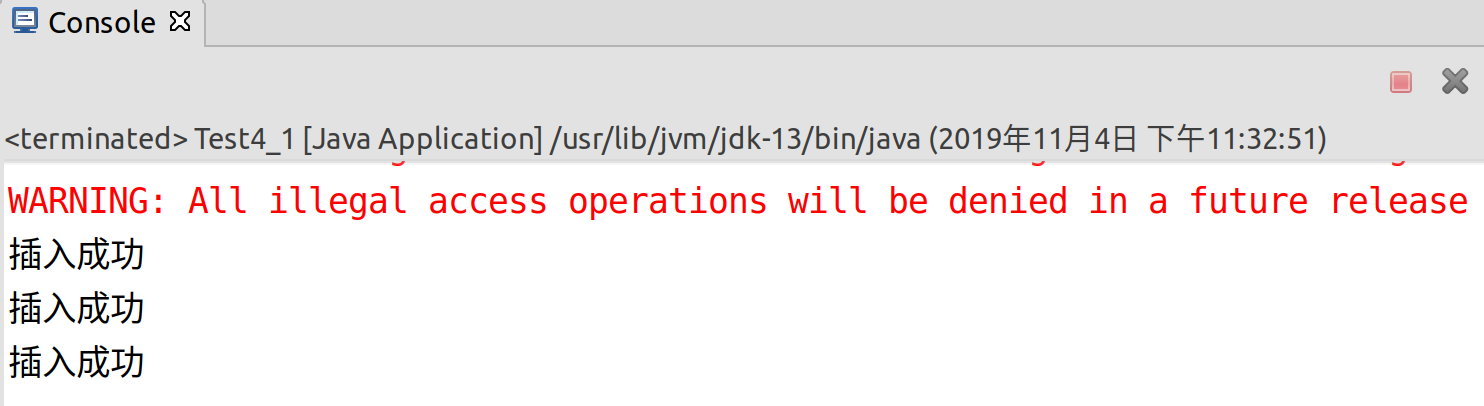
e.printStackTrace();

}

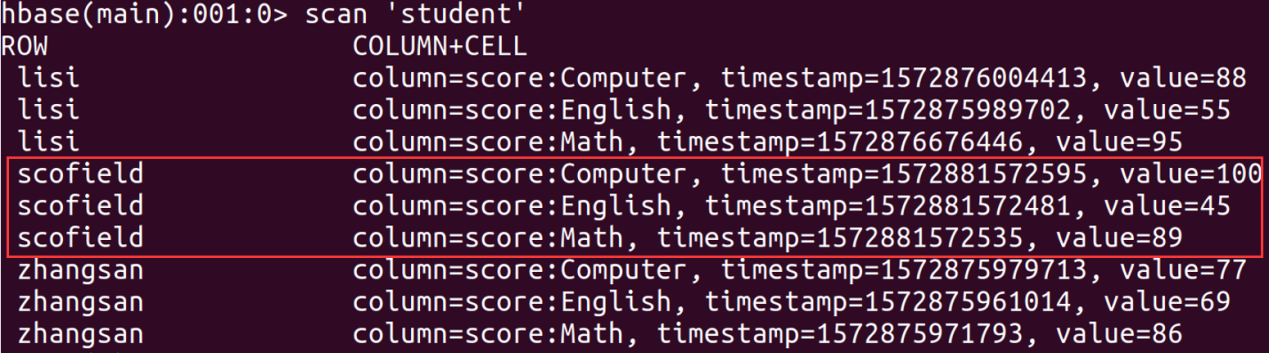
}

}

**运行结果：**



**使用shell，查看表的内容，发现插入成功**



1. 获取scofield的English成绩信息。

**Java代码：**

**package** hbase;

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configuration;

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

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

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

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

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

**publicclass** Test4\_2 {

**publicstatic** Configuration *configuration*;

**publicstatic** Connection *connection*;

**publicstatic** Admin *admin*;

**publicstaticvoid** main(String[] args) {

**try** {

*getData*("student", "scofield", "score", "English");

} **catch** (IOException e) {

e.printStackTrace();

}

}

**publicstaticvoid** getData(String tableName, String rowKey, String colFamily, String col) **throws** IOException {

*init*();

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

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

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

Result result = table.get(get);

Cell[] cells = result.rawCells();

**for** (Cell cell : cells) {

System.***out***.println("RowName:" + **new** String(CellUtil.*cloneRow*(cell)) + " ");

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

System.***out***.println("column Family:" + **new** String(CellUtil.*cloneFamily*(cell)) + " ");

System.***out***.println("row Name:" + **new** String(CellUtil.*cloneQualifier*(cell)) + " ");

System.***out***.println("value:" + **new** String(CellUtil.*cloneValue*(cell)) + " ");

}

table.close();

*close*();

}

**publicstaticvoid** 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();

}

}

**publicstaticvoid** close() {

**try** {

**if** (*admin* != **null**) {

*admin*.close();

}

**if** (**null** != *connection*) {

*connection*.close();

}

} **catch** (IOException e) {

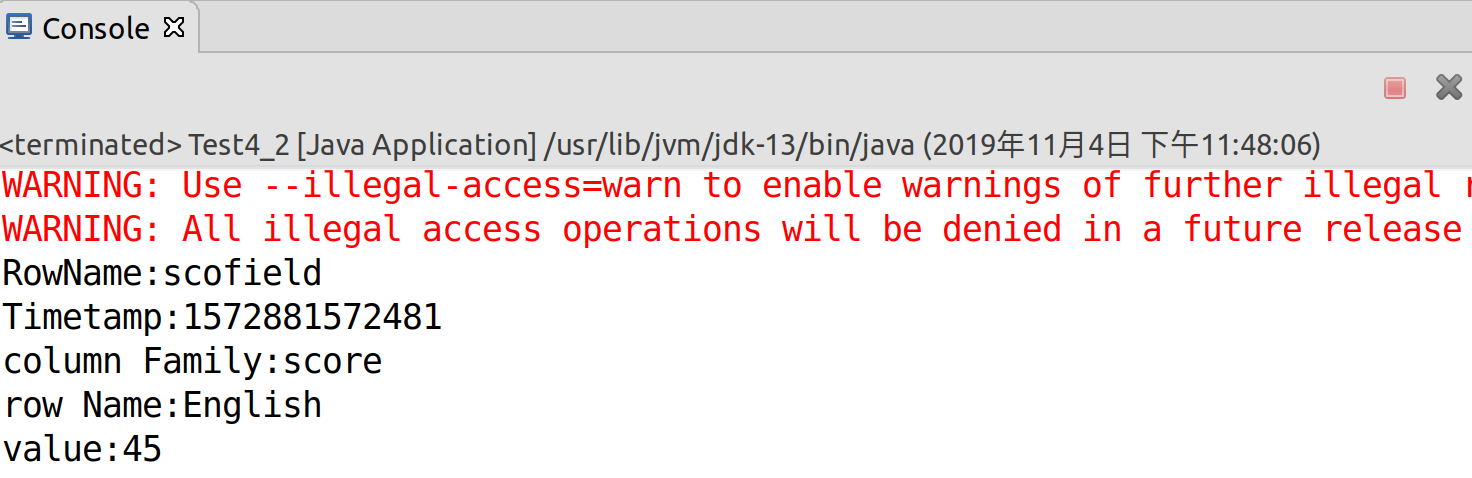
e.printStackTrace();

}

}

}

**运行结果：**





## 2.3 Redis数据库操作

Student键值对如下：

|  |
| --- |
| zhangsan:｛  English: 69  Math: 86  Computer: 77  ｝  lisi:｛  English: 55  Math: 100  Computer: 88  ｝ |

1. 根据上面给出的键值对，完成如下操作：

（1）用Redis的哈希结构设计出学生表Student（键值可以用student.zhangsan和student.lisi来表示两个键值属于同一个表）；

**启动：redis-cli**

**使用hset添加数据， student.代表属于student表，.zhangsan表示行键为zhangsan**

hset student.zhangsan English 69

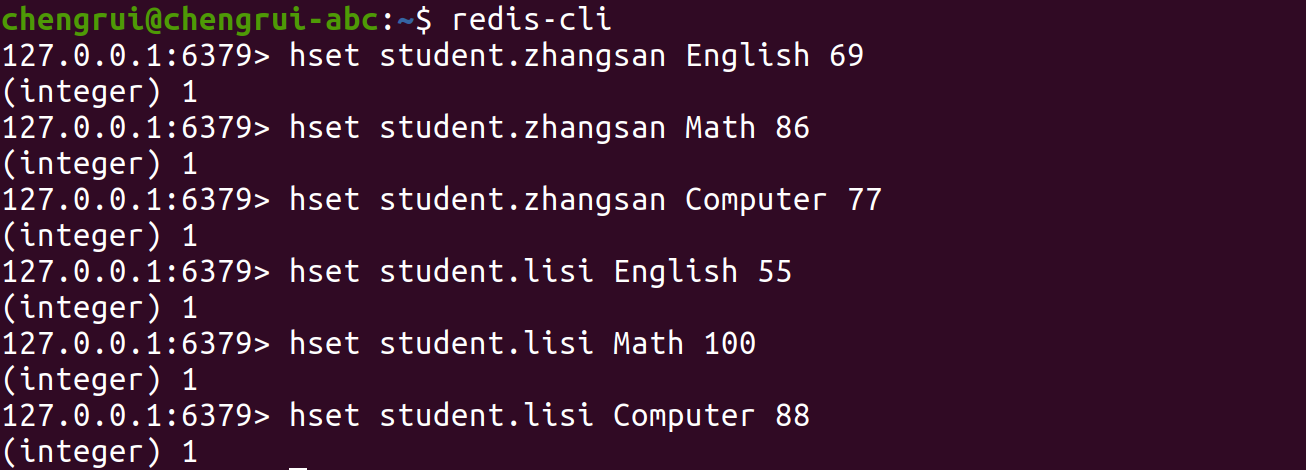
hset student.zhangsan Math 86

hset student.zhangsan Computer 77

hset student.lisi English 55

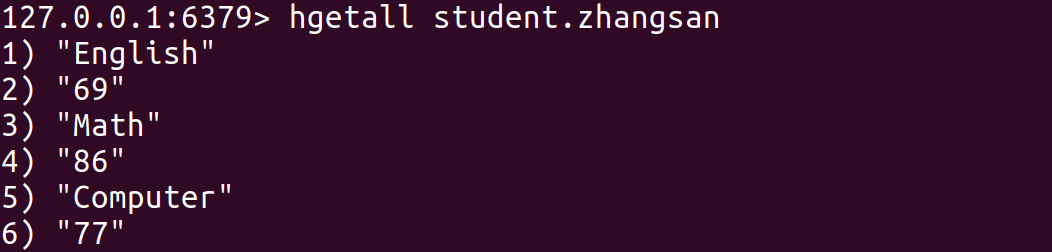
hset student.lisi Math 100

hset student.lisi Computer 88

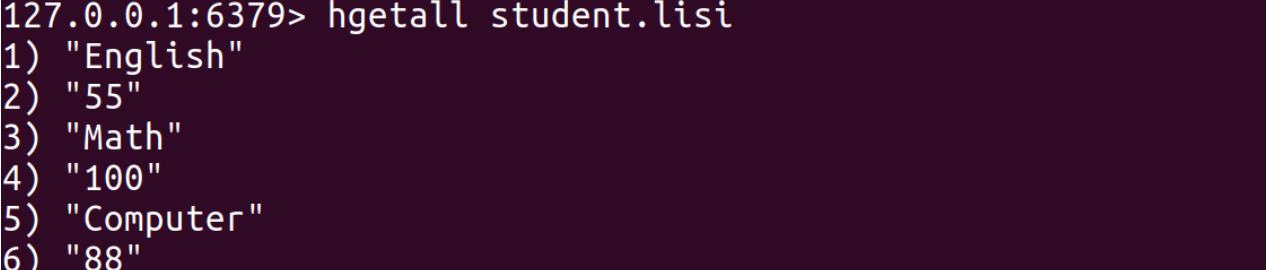


1. 用hgetall命令分别输出zhangsan和lisi的成绩信息；

**hgetall 行键（输出本行的所有信息）**

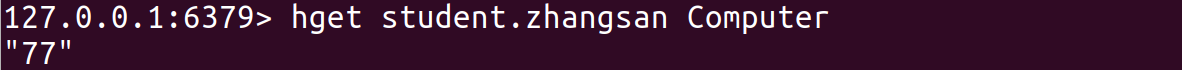


**hgetall student.lisi**



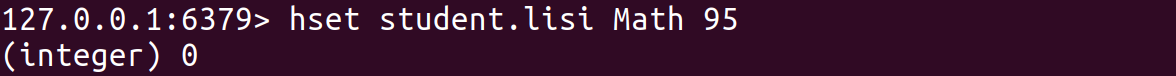
1. 用hget命令查询zhangsan的Computer成绩；

**hget student.zhangsan Computer（hget集合列名）**

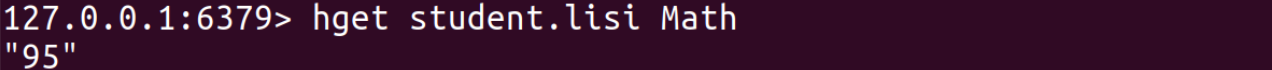


1. 修改lisi的Math成绩，改为95。

**hset student.lisi Math 95**

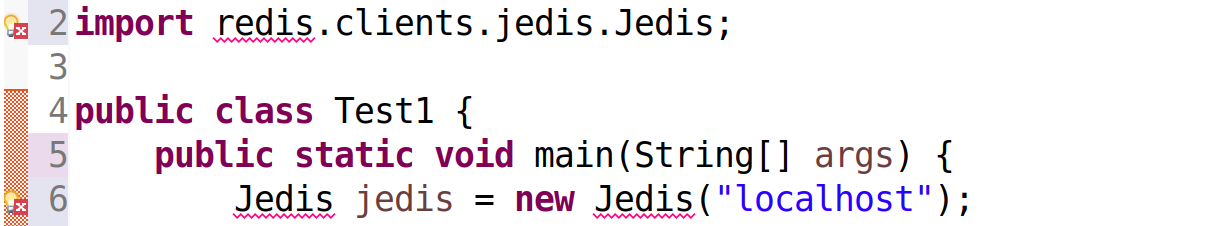


**Hget查看修改将结果**

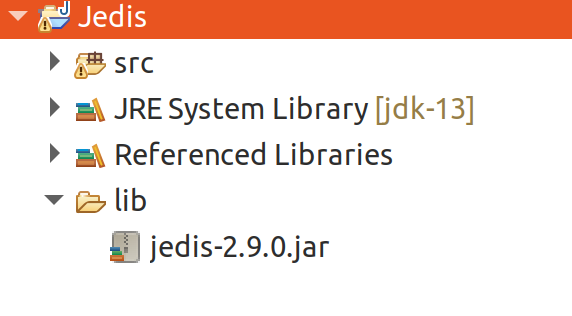


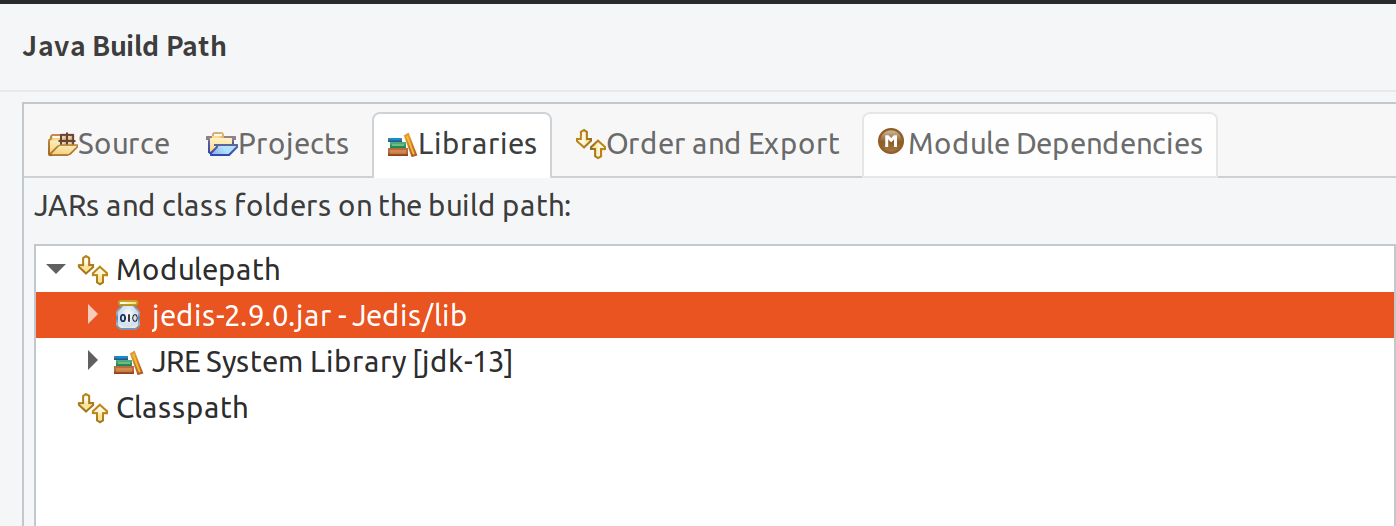
1.根据上面已经设计出的学生表Student，用Redis的JAVA客户端编程(jedis)，实现如下操作：

**一开始会报错，这是因为没有导入包的原因：**



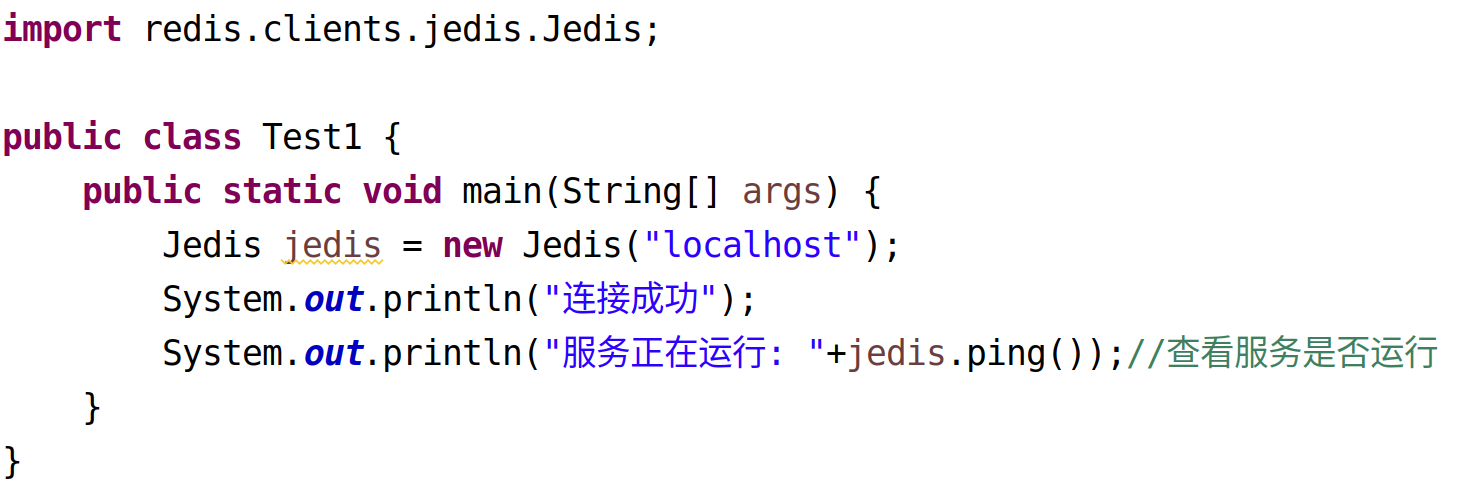
导入包：

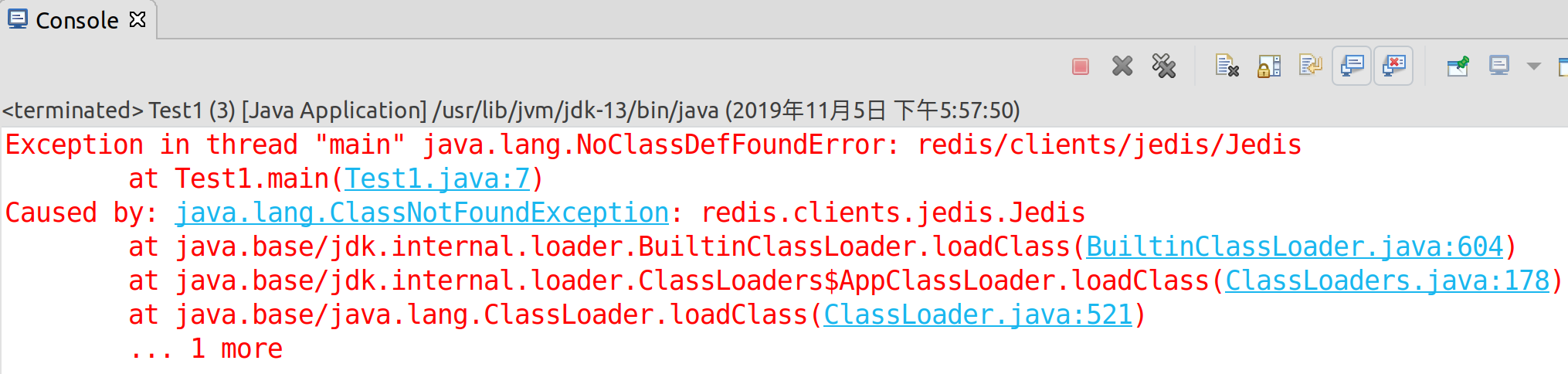




**这里build path错了，不能在modulepath，否则还是会报错。**

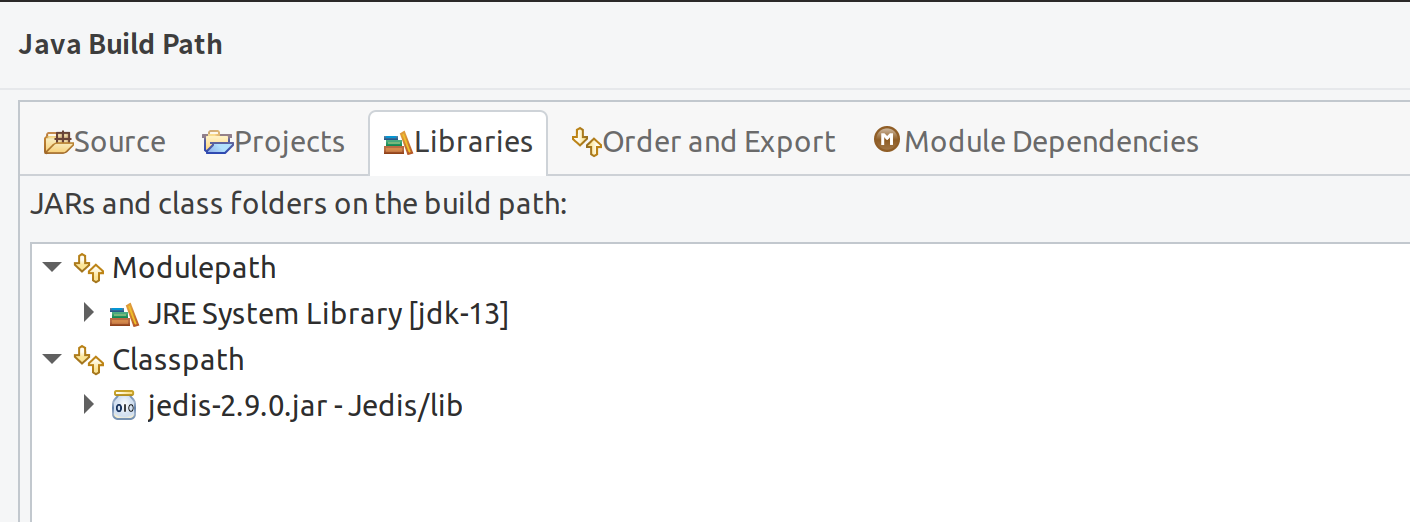
**查看服务是否运行：**



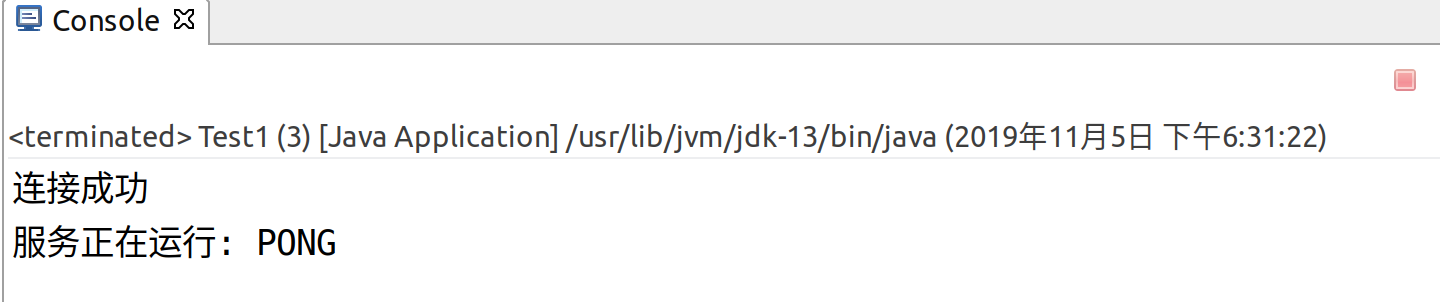


**Build path错了**

**要在classpath中**



**可以了：**



（1）添加数据：English:45 Math:89 Computer:100

该数据对应的键值对形式如下：

|  |
| --- |
| scofield:｛  English: 45  Math: 89  Computer: 100  ｝ |

**java代码：**

**import** java.util.Map;

**import** redis.clients.jedis.\*;

**publicclass** Test1 {

**publicstaticvoid** main(String[] args) {

Jedis jedis = **new** Jedis("localhost");

jedis.hset("student.scofield", "English", "45");

jedis.hset("student.scofield", "Math", "89");

jedis.hset("student.scofield", "Computer", "100");

Map<String, String>value = jedis.hgetAll("student.scofield");

System.***out***.println("信息添加成功：");

System.***out***.println("student.scofield：");

**for** (Map.Entry<String, String>entry : value.entrySet()) {//遍历

//Map.Entry是Map中的一个接口，

//表示一个一个映射项，有key，value

System.***out***.println(entry.getKey() + ":" + entry.getValue());

}

}

}

**运行结果：**



（2）获取scofield的English成绩信息

**Java代码：**

**import** redis.clients.jedis.Jedis;

**publicclass** Test2 {

**publicstaticvoid** main(String[] args) {

Jedis jedis = **new** Jedis("localhost");

String value = jedis.hget("student.scofield", "English");

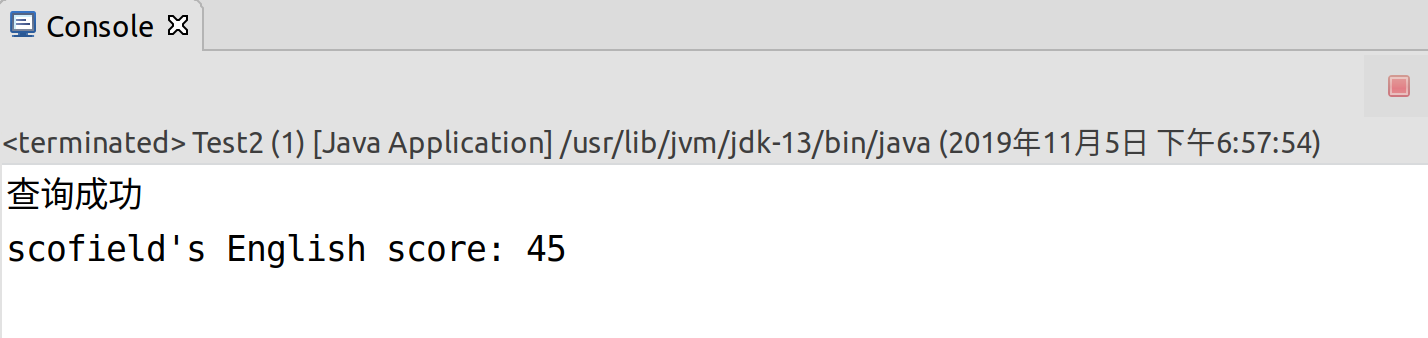
System.***out***.println("查询成功");

System.***out***.println("scofield's English score: " + value);

}

}

**运行结果：**



## 2.4MongoDB数据库操作

Student文档如下:

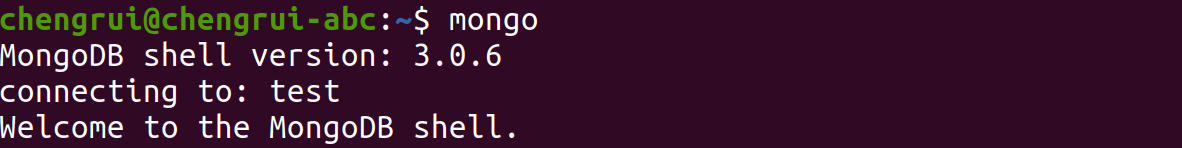
|  |
| --- |
| {  “name”: “zhangsan”,  “score”: {  “English”: 69,  “Math”: 86,  “Computer”: 77  }  }  {  “name”: “lisi”,  “score”: {  “English”: 55,  “Math”: 100,  “Computer”: 88  }  } |

1. 根据上面给出的文档，完成如下操作：

**Mongod开启服务**



**Mogo进入shell**



（1）用MongoDBShell设计出student集合；

**切换到 student 集合：**

**use student**

**定义数组：**

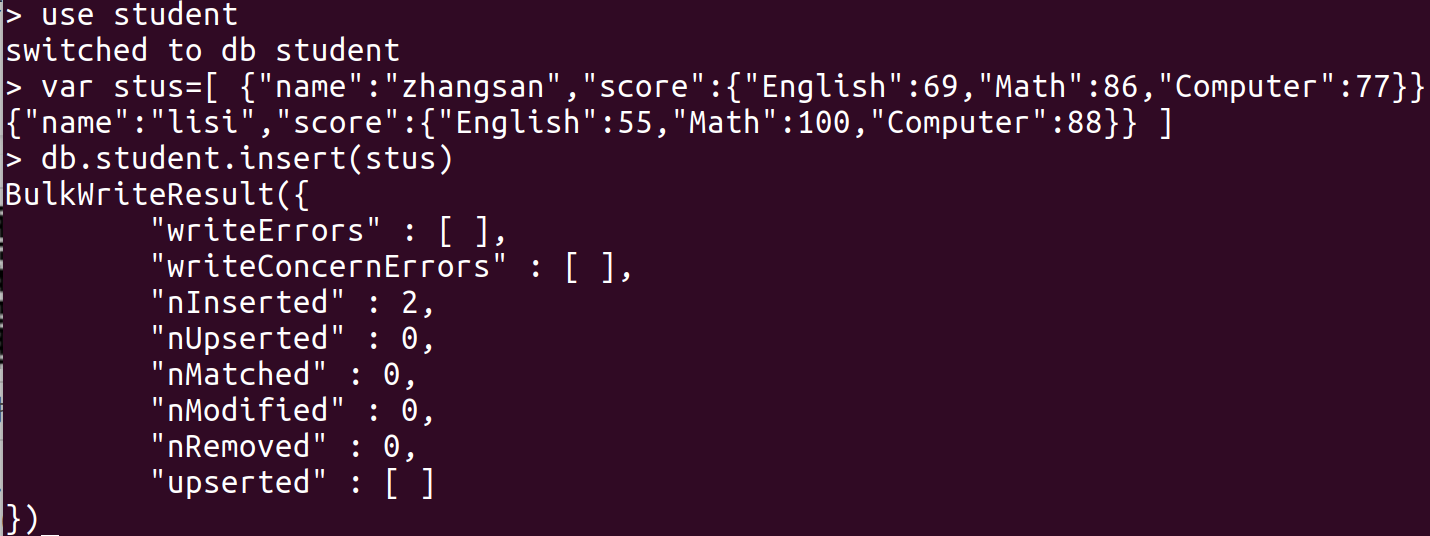
**var stus=[**

**{"name":"zhangsan","scores":{"English":69,"Math":86,"Computer":77}},**

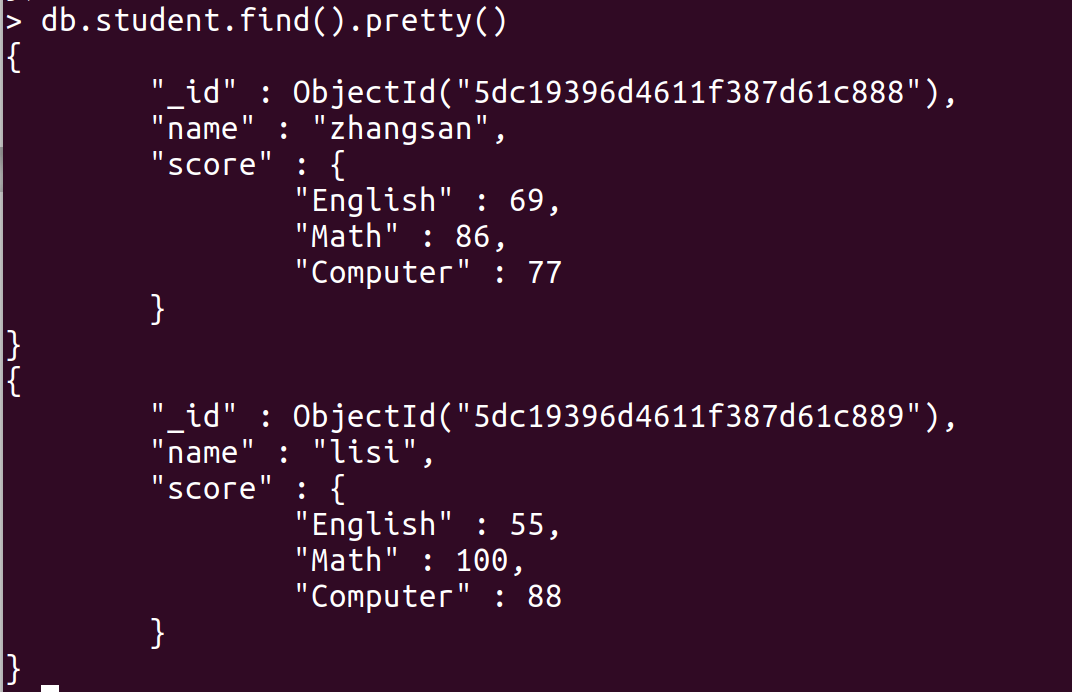
**{"name":"lisi","score":{"English":55,"Math":100,"Computer":88}} ]**

**插入：**

**db.student.insert(stus)**



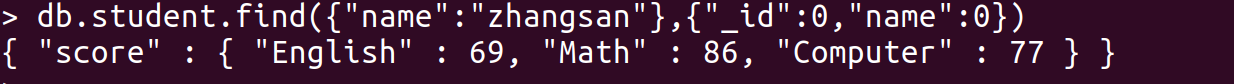
1. 用find()方法输出两个学生的信息；



1. 用find()方法查询zhangsan的所有成绩(只显示score列)；

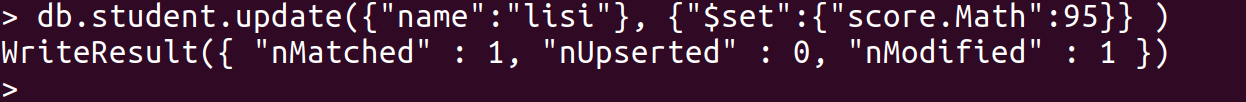
**db.student.find({"name":"zhangsan"},{"\_id":0,"name":0})**

**//\_id":0,"name":0表示不显示id和name，只显示score**

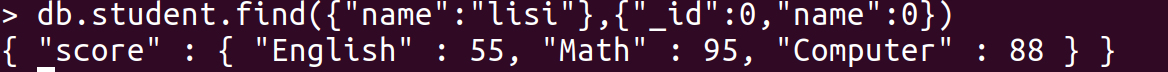


（4）修改lisi的Math成绩，改为95。

**db.student.update({"name":"lisi"}, {"$set":{"score.Math":95}} )**

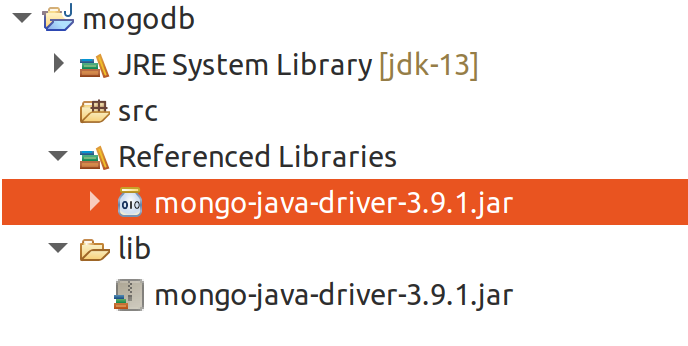


**使用find查看结果**



1. 根据上面已经设计出的Student集合，用MongoDB的Java客户端编程，实现如下操作：

**导入包:**



（1）添加数据：English:45 Math:89Computer:100

与上述数据对应的文档形式如下：

|  |
| --- |
| {  “name”: “scofield”,  “score”: {  “English”: 45,  “Math”: 89,  “Computer”: 100  }  } |

**java代码：**

**package** mogodb;

**import** java.util.ArrayList;

**import** java.util.List;

**import**org.bson.Document;

**import** com.mongodb.MongoClient;

**import** com.mongodb.client.\*;

**publicclass** Test {

**publicstaticvoid** main(String[] args) {

MongoClient mongoClient = **new** MongoClient("localhost", 27017);//客户端

MongoDatabase mongoDatabase = mongoClient.getDatabase("student");//数据库

MongoCollection<Document>collection = mongoDatabase.getCollection("student");//获取集合

//实例化一个文档,内嵌一个子文档

Documentdocument = **new**Document("name", "scofield").append("score",

**new**Document("English", 45).append("Math", 89).append("Computer", 100));

List<Document>documents = **new** ArrayList<Document>();

documents.add(document);

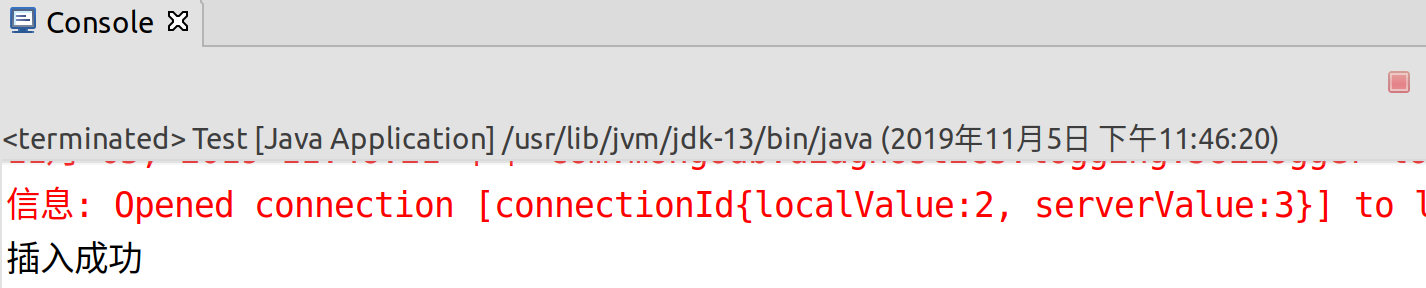
collection.insertMany(documents);// 插入集合

System.***out***.println("插入成功");

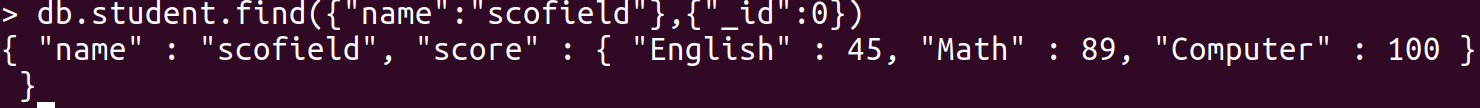
}

}

**运行结果：**



**使用find验证**



1. 获取scofield的所有成绩成绩信息(只显示score列)

**java代码：**

**package** mogodb;

**import** org.bson.Document;

**import** com.mongodb.MongoClient;

**import** com.mongodb.client.\*;

**publicclass** Test2 {

**publicstaticvoid** main(String[] args) {

MongoClient mongoClient = **new** MongoClient("localhost", 27017);

MongoDatabase mongoDatabase = mongoClient.getDatabase("student");

MongoCollection<Document>collection = mongoDatabase.getCollection("student");// 获取集合

// 查找,name=scofield, 只显示 score

MongoCursor<Document>cursor = collection.find(**new** Document("name", "scofield"))

.projection(**new** Document("score", 1).append("\_id", 0)).iterator();

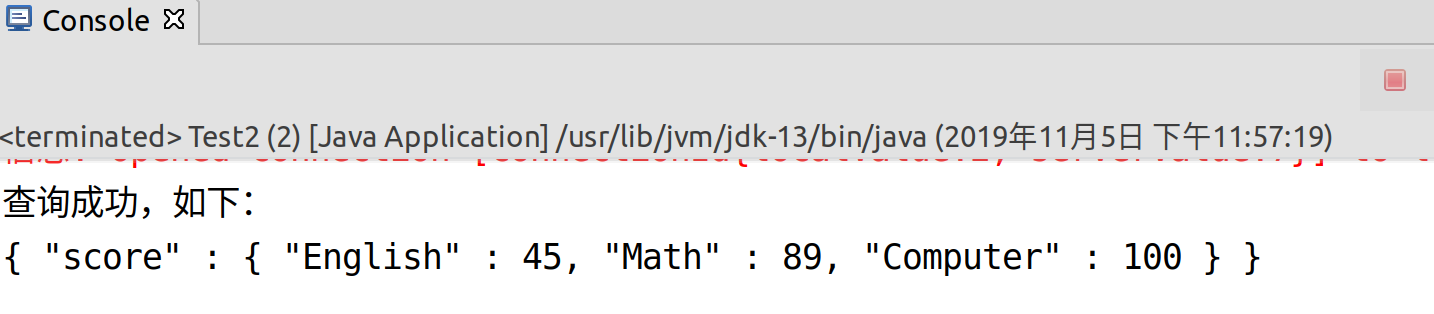
System.***out***.println("查询成功，如下：");

**while** (cursor.hasNext())

System.***out***.println(cursor.next().toJson());//格式化为 JSON 数据格式

}

}

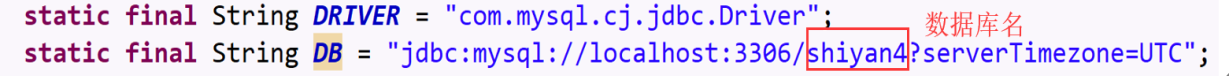


# 实验总结

## 3.1MySQL

（1）

mysql-connector-java-8.0.16包对应代码，DRIVER和DB应该这样写：

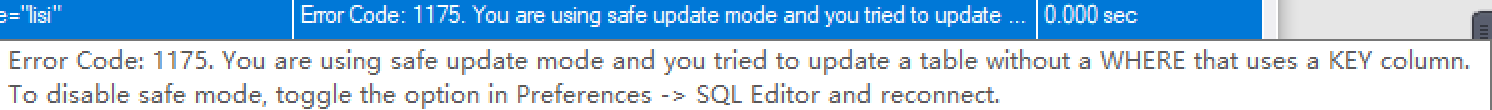


如果写成其他版本的com.mysql.jdbc.driver会报错



（2）

修改表信息时报错：



说明建表时没有设置主键，设置就能解决，alter table student add primary key (Name);

Addprimarykey可以给表添加主键。

（3）

在Java代码输出查询数据库结果对齐：

System.out.printf("%-10s%-10d",rs.getString(1),rs.getInt(2));

意思是：掉用printf输出格式，-代表左边对齐，10代表占10位，不够的用空格补全。

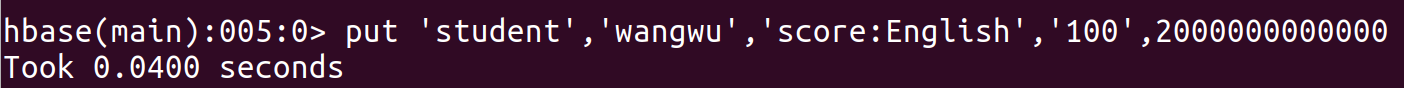
## 3.2HBase

（1）

关于hbase的时间戳的一些思考：

**timestamp：时间戳（默认自动添加，使用的是数据添加时的时间，但是也可以使用put方法添加数据的时候手动添加。）**

**自定义时间戳**：直接在后面加一个参数表示时间。

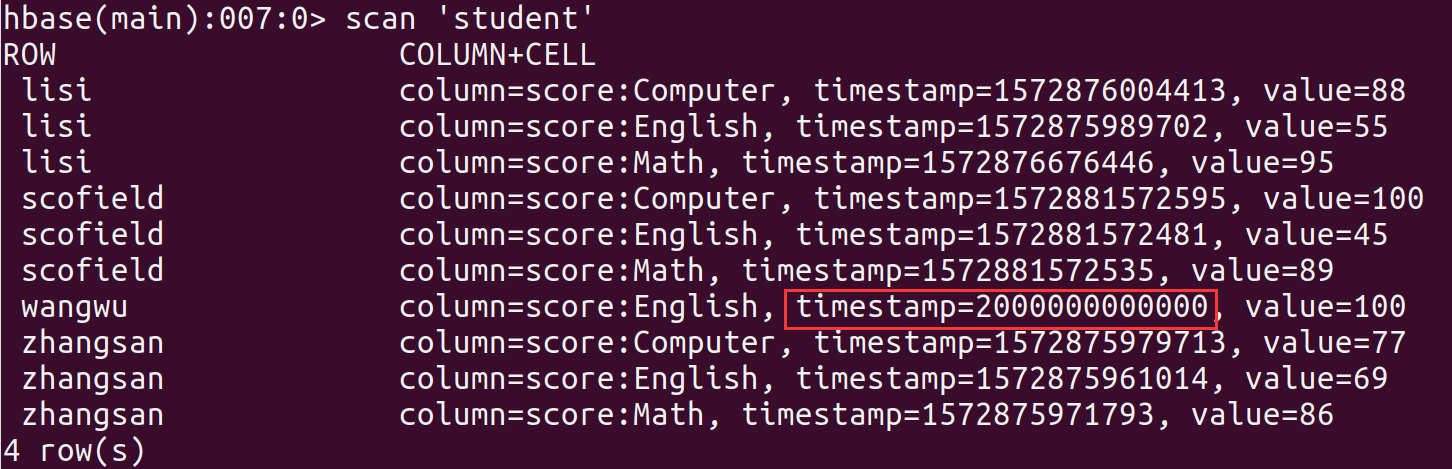


但是自己定义的时间戳，一定要比原来的数据**大**，否者会失效。

**验证一下：**

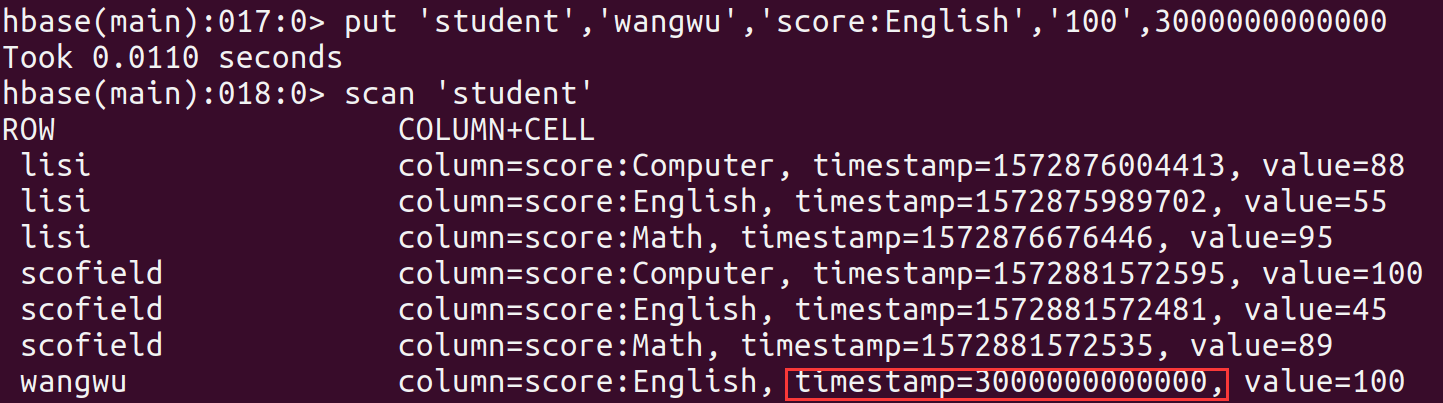
**自定义时间戳：**

**put 'student','wangwu','score:English','100',2000000000000**



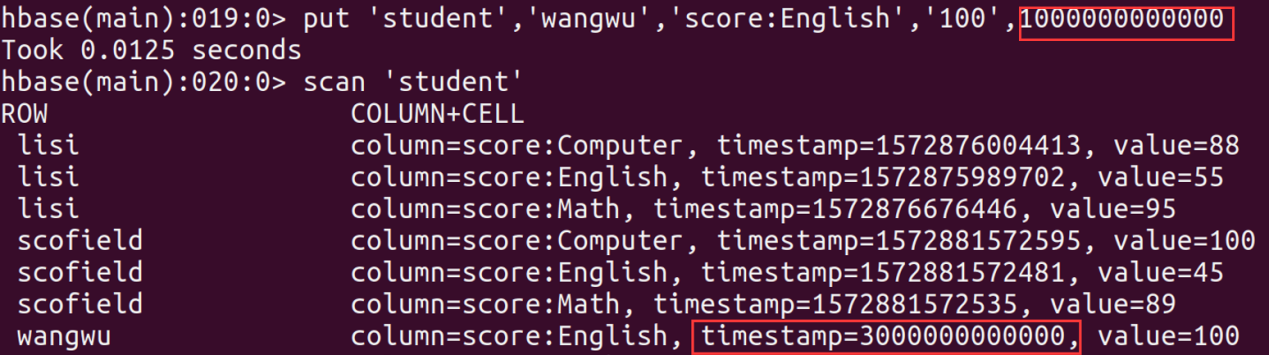
**如果时间戳比原来大：**

**成功修改：**



**比原来小：**

**修改无效：**



（2）

hbase使用Put，如有有数据就会修改，没有就会插入数据。

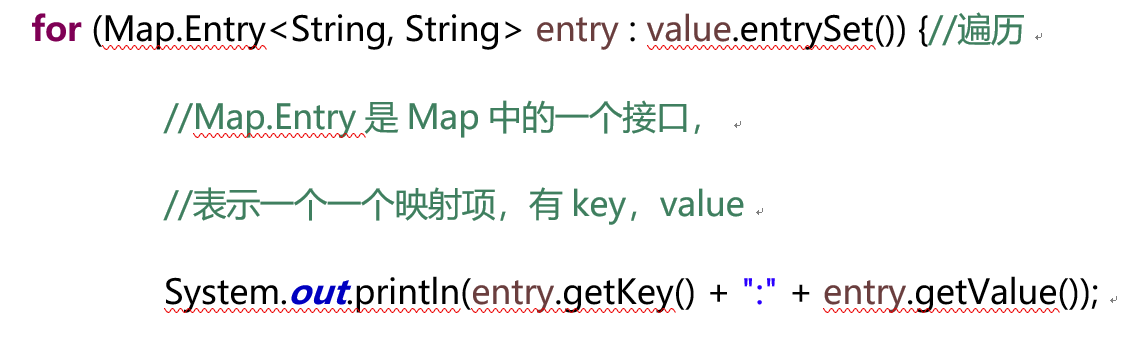
与mongodb类似hset 有数据则修改，无数据就插入。

（3）没有索引，没有查询语句，要通过行键查询。

## 3.3Redis

（1）

**Map.Entry是Map中的一个接口，表示一个一个映射项，有key，value：**



**Java代码中，用来表示，列名：值。**

## 3.4MongoDB

（1）

**使用find查询时:**

**使用0，1可以自定义显示什么，如：**

用find()方法查询zhangsan的所有成绩(只显示score列)；

**db.student.find({"name":"zhangsan"},{"\_id":0,"name":0})**

**//\_id":0,"name":0表示不显示id和name，只显示score**

（2）

**在MongoDB中，**

**表--集合；行--文档；列--字段**

**集合：①就是 MongoDB 文档组，集合存在于数据库中，没有固定的结构，所以在集合可以插入不同格式和类型的数据。**

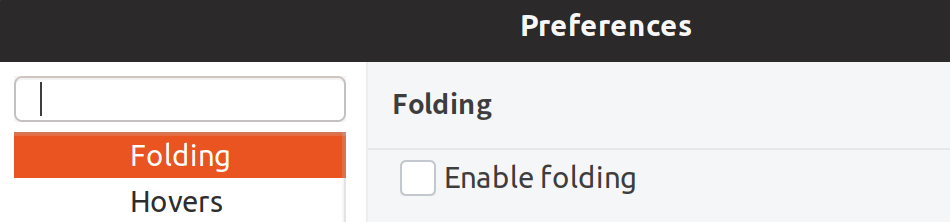
**②当第一个文档插入时，集合就会被创建，所以不需要特地去创建表。**

**文档：是一组键值对，文档不需要设置相同的字段，并且相同的字段不需要相同的数据类型。**

**（3）查询的时候先得到collection，可以看作一个document的链表，结点就是文档，也就是行。**

## 3.5其他

（1）关于怎么从eclipse中**复制出带颜色的代码**



**把勾去掉，复制后就会保留颜色和格式。**

（2）  MongoDb 和Redis  都是NoSQL, 采用结构型数据存储。

（3）**四种**数据库基本操作的**比较**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Mysql | Hbase | Redis | MongoDB |
| 建表 | create table xx(  属性类型  ); | create ‘表名’，’列名’ | Hset添加数据用表名.\*\*，表示同一表 | use 表名 |
| 插入数据 | insert into 表名 values(, , ); | put ‘表名’,’行键’,  ’列名’,’值’ | hset 表名.行键列值 | var 数组=[  {"列名":"值","列名":"值"}  ]  插入：  db.表名.insert(数组) |
| 查询 | select...from...  where… | scan ‘表’、  get ‘表’,‘行’,’列’ | hget集合列名  hgetall集合 | db.表.find({"列":"值"}  ,{显示的约束}) |
| 修改 | update …  set …=…  where… | 使用Put，如有有数据就会修改，没有就会插入数据。 | hset如有有数据就会修改，没有就会插入数据。 | db.表.update(  {"列":"值"},  {"$set":{"列":值}}  ) |