

**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** |  | | | **实验日期** |  |

## 实验四：NoSQL和关系数据库的操作比较

### 一、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。

### 二、实验平台

* 操作系统：Linux（建议Ubuntu18.04）；
* Hadoop版本：3.2.0；
* MySQL版本：自定义；
* HBase版本：2.2.1；
* Redis版本：自定义；
* MongoDB版本：自定义；
* JDK版本：openjdk-1.8；
* Java IDE：Eclipse；

### 三、实验步骤

**（一） MySQL数据库操作**148陈元莹

实验前提：安装Mysql Workbench；

Mysql官网：[http://www.mysql.com](http://www.mysql.com/)

在Mysql中新建一个**bigdata**项目

注：每次输入SQL命令之后都要按ctrl键+回车键来执行

**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；**

首先创建表：

create table student(

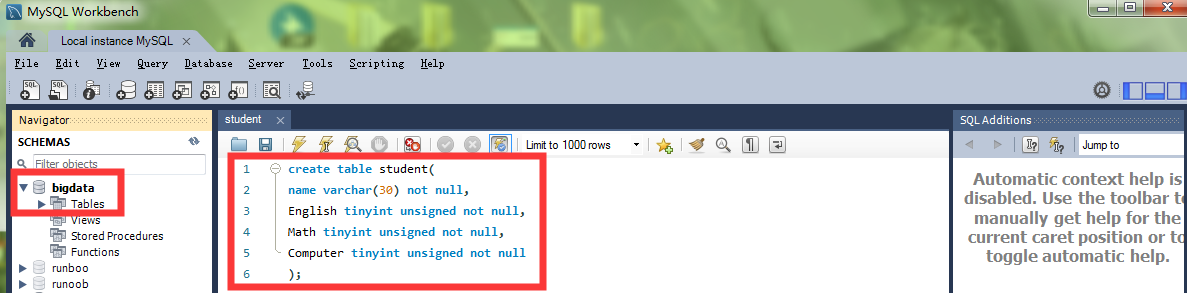
name varchar(30) not null,

English tinyint unsigned not null,

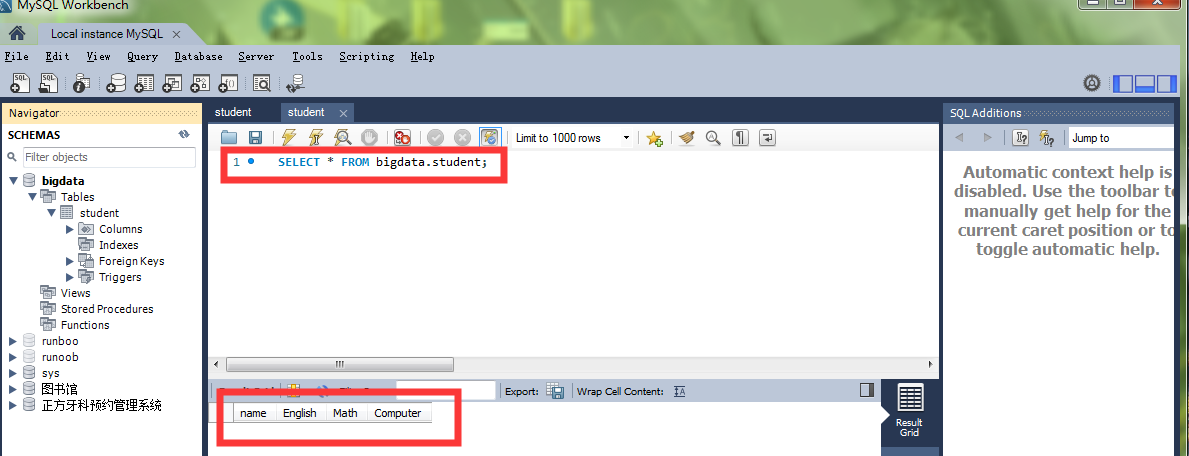
Math tinyint unsigned not null,

Computer tinyint unsigned not null

);



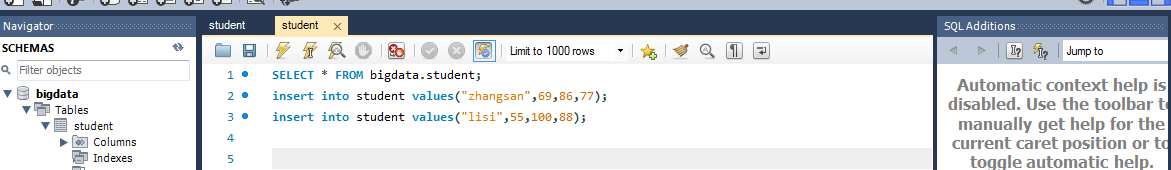
输入命令SELECT \* FROM bigdata.student;查看刚刚创建的学生表



可见，表已创建成功，此时开始添加数据：

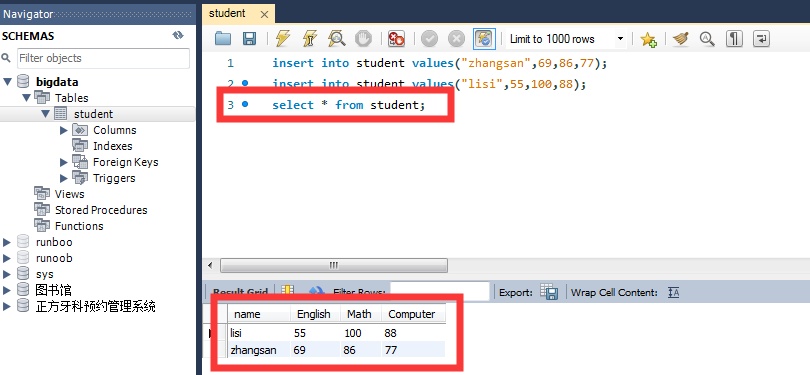
insert into student values("zhangsan",69,86,77);

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



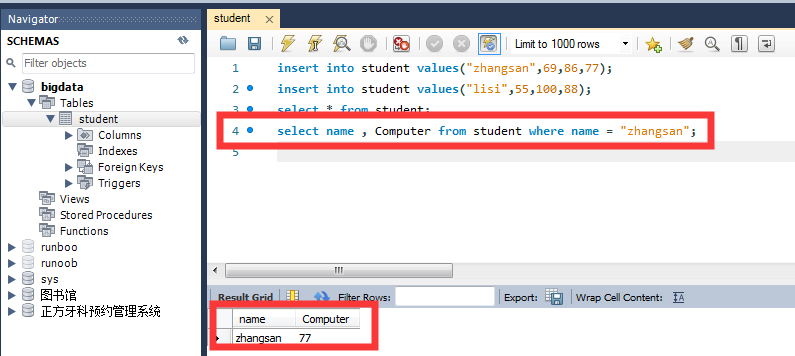
1. **用SQL语句输出Student表中的所有记录；**

SQL语句：select \* from student;



1. **查询zhangsan的Computer成绩；**

SQL语句：select name , Computer from student where name = "zhangsan";



**（4）修改lisi的Math成绩，改为95。**

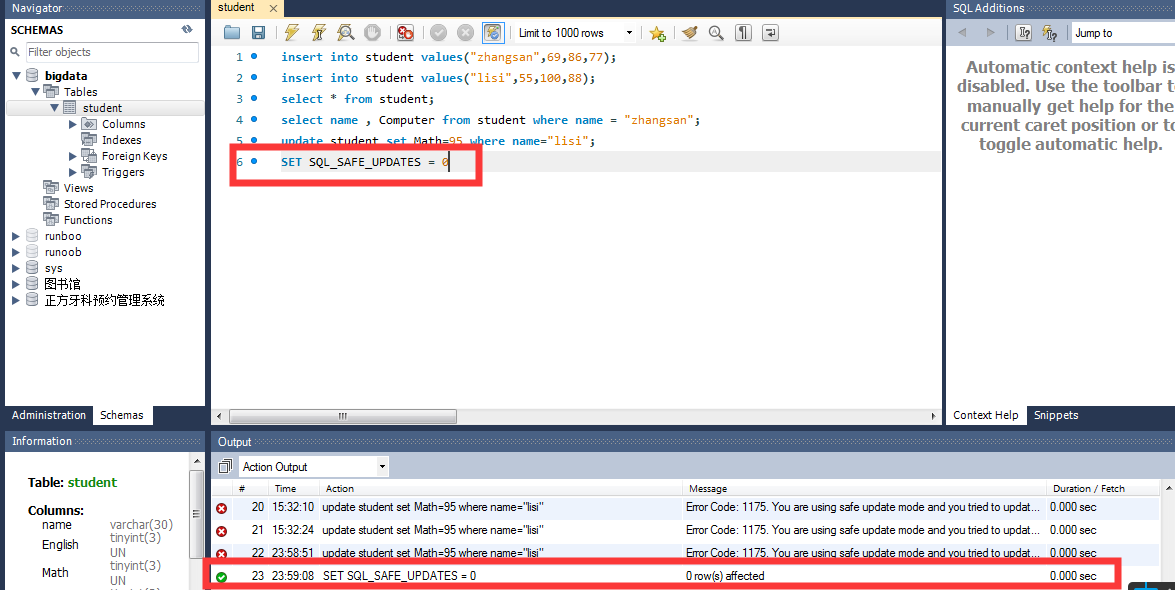
SQL语句：update student set Math=95 where name="lisi";

当执行时报了以下的错误：Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column To disable safe mode, toggle the option in Preferences -> SQL Queries and reconnect.

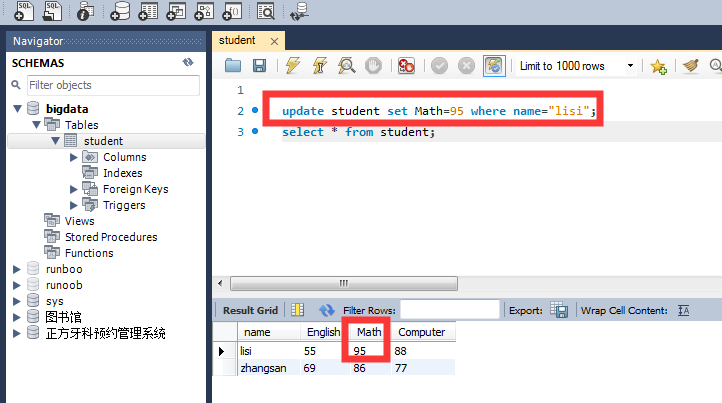


百度了之后发现是这是因为MySql运行在safe-updates模式下，该模式会导致非主键条件下无法执行update或者delete命令；

解决办法：执行命令SET SQL\_SAFE\_UPDATES = 0;修改数据库模式



再次运行SQL语句：update student set Math=95 where name="lisi";



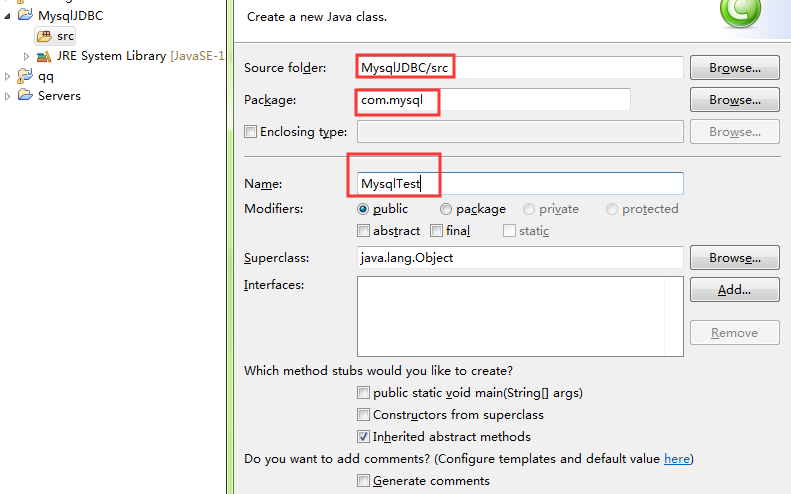
可见lisi的Math成绩已成功修改为95

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** |

新建一个项目MysqlJDBC，然后再建mysql包，在包中新建java文件MysqlTest



去菜鸟教程中：<https://www.runoob.com/java/java-mysql-connect.html>

下载mysql的jar包并导入：



JAVA代码：

package com.mysql;

import java.sql.\*;

public class MysqlTest {

static final String DRIVER = "com.mysql.jdbc.Driver";

static final String DB = "jdbc:mysql://localhost/bigdata?useUnicode=true&characterEncoding=utf-8&useSSL=false" ;

static final String USER = "root";

static final String PASSWD = "ydy19990616";

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

Class.forName(DRIVER);

System.out.println("Connecting to a selected database...");

conn = DriverManager.getConnection(DB, USER, PASSWD);

stmt = conn.createStatement();

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

stmt.executeUpdate(sql);

System.out.println("Inserting records into the table successfully!");

} 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 {

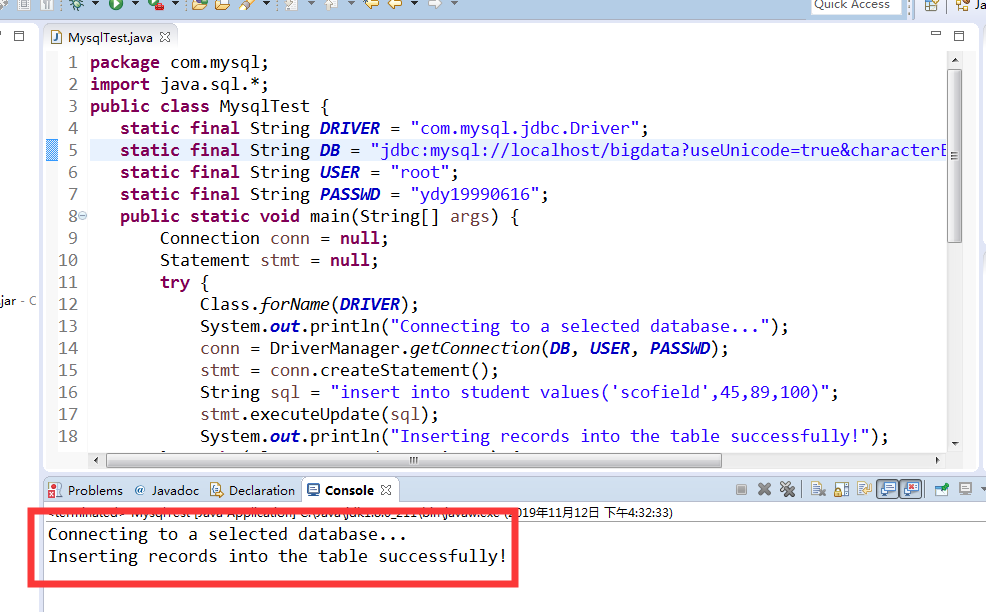
conn.close();

} catch (SQLException e) {

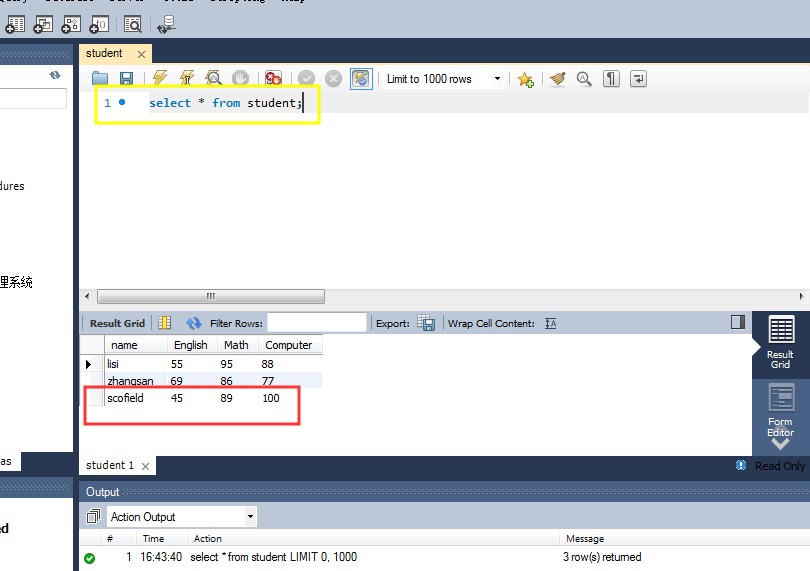
e.printStackTrace();

}}}}

运行成功



在mysql中验证结果，如图数据成功添加：



**（2）获取scofield的English成绩信息**

新建java文件MysqlTest2

JAVA代码：

package com.mysql;

import java.sql.\*;

public class MysqlTest2 {

static final String DRIVER = "com.mysql.jdbc.Driver";

static final String DB = "jdbc:mysql://localhost/bigdata?useUnicode=true&characterEncoding=utf-8&useSSL=false";

static final String USER = "root";

static final String PASSWD = "ydy19990616";

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

try {

Class.forName(DRIVER);

System.out.println("Connecting to a selected database...");

conn = DriverManager.getConnection(DB, USER, PASSWD);

stmt = conn.createStatement();

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

rs = stmt.executeQuery(sql);

System.out.println("name" + "\t\t" + "English");

while (rs.next()) {

System.out.print(rs.getString(1) + "\t\t");

System.out.println(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 {

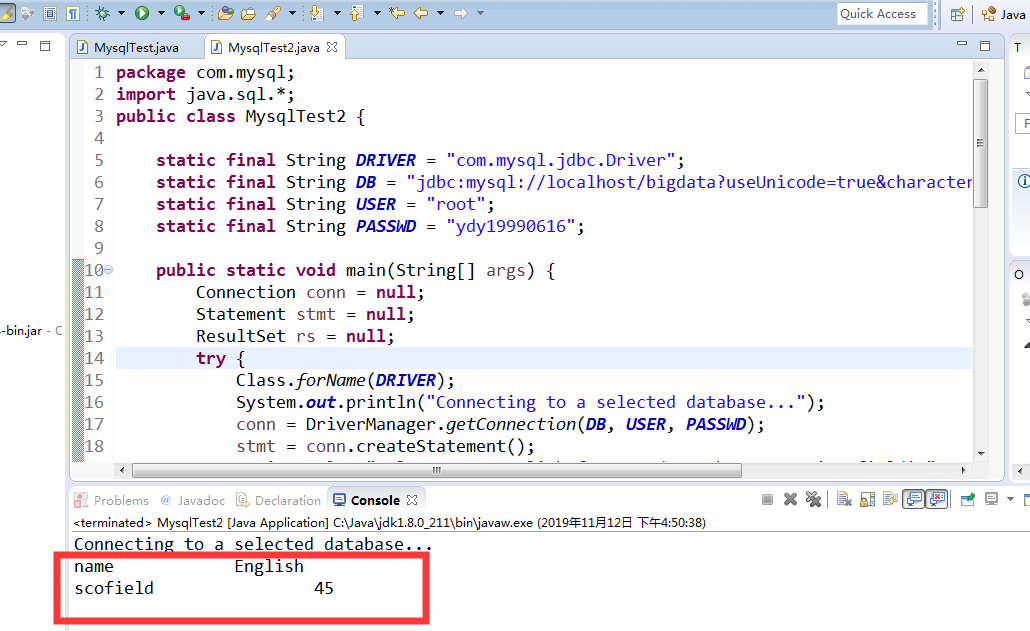
conn.close();

} catch (SQLException e) {

e.printStackTrace();

}}}}

运行成功，获取到scofield的English成绩信息：



**（二）HBase数据库操作**

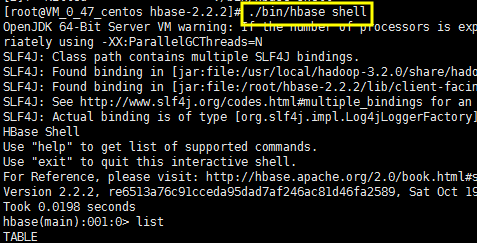
在上次实验中已在云服务器中安装了HBase，因此以下内容在云服务器平台下完成。

**学生表Student**

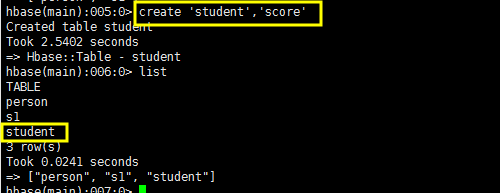
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **score** | | | |
| **English** | **Math** | **Computer** | |
| **zhangsan** | **69** | **86** | | **77** |
| **lisi** | **55** | **100** | | **88** |

1. **根据上面给出的学生表Student的信息，执行如下操作：**
2. **用Hbase Shell命令创建学生表Student；**

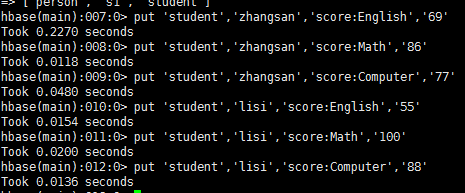
进入hbase-2.2.2路径下，输入命令./bin/hbase shell



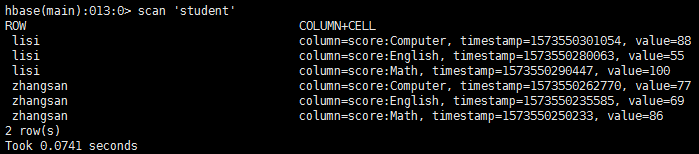
创建 Student 表



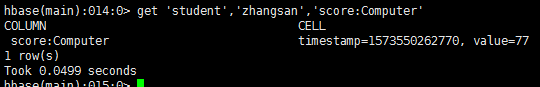
插入数据



1. **用scan命令浏览Student表的相关信息；**

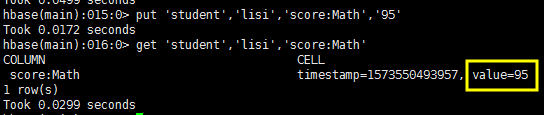


1. **查询zhangsan的Computer成绩；**



**（4）修改lisi的Math成绩，改为95。**

先修改，再用get命令验证结果



**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**

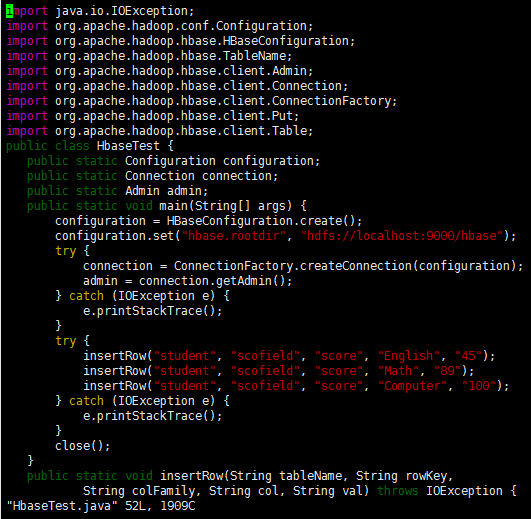
|  |  |  |  |
| --- | --- | --- | --- |
| **scofield** | **45** | **89** | **100** |

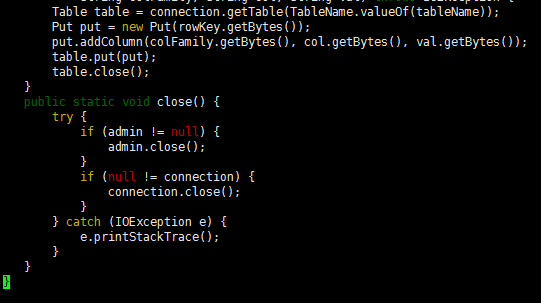
新建一个HbaseTest.java文件,



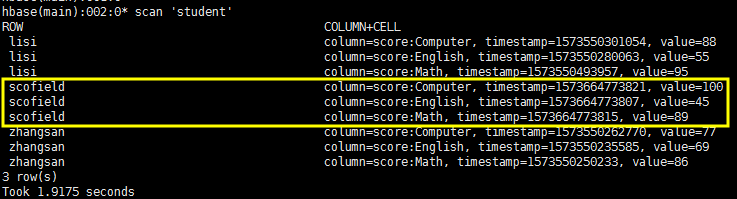


输入代码并编译运行：





hbase shell查看

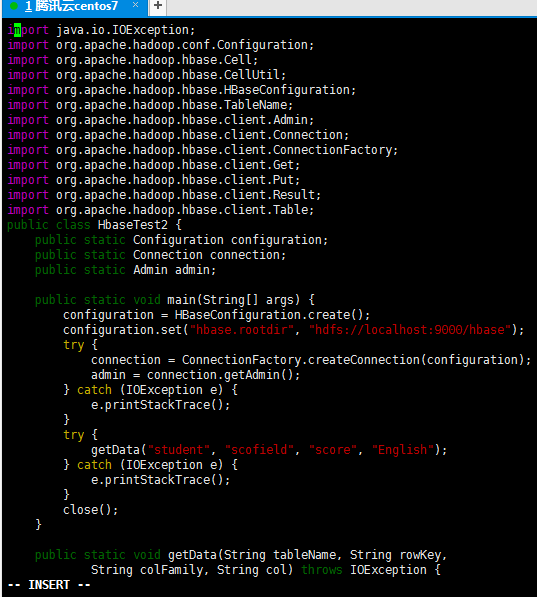


1. **获取scofield的English成绩信息。**

新建一个HbaseTest2.java文件,

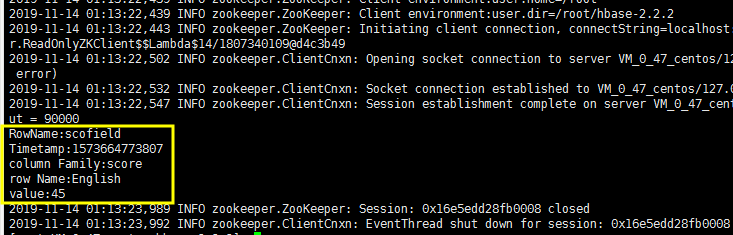


输入代码并编译运行：





运行结果如下所示，成功获取scofield的English成绩信息：



**（三）Redis数据库操作**

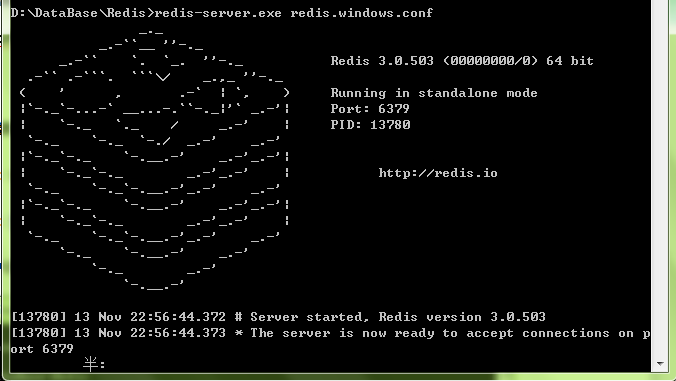
实验前提：安装Redis，

安装教程：<https://www.cnblogs.com/javabg/p/9133206.html>

在cmd中进入Redis的安装目录中

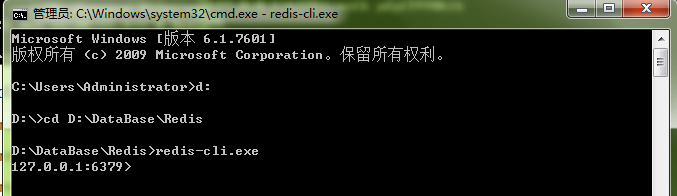


输入redis-server.exe redis.windows.conf启动redis以及配置文件



新打开一个cmd窗口作为客户端

输入命令redis-cli.exe运行redis-cli程序



输入我的redis数据库密码：ydy19990616，接下来就可以开始做题目了

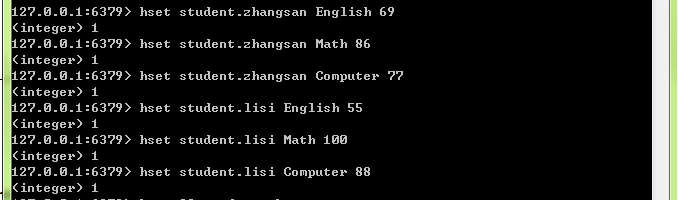


**Student键值对如下：**

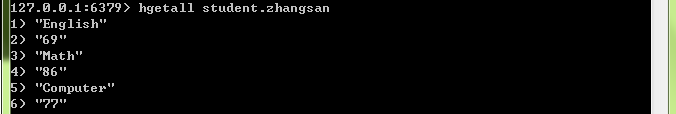
|  |
| --- |
| **zhangsan:｛**  **English: 69**  **Math: 86**  **Computer: 77**  **｝**  **lisi:｛**  **English: 55**  **Math: 100**  **Computer: 88**  **｝** |

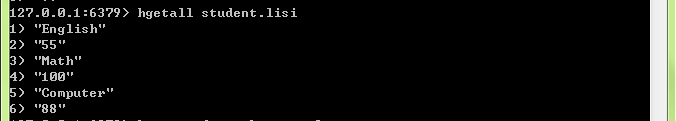
**1. 根据上面给出的键值对，完成如下操作：**

**（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）；**



**（2）用hgetall命令分别输出zhangsan和lisi的成绩信息；**





**（3）用hget命令查询zhangsan的Computer成绩；**



**（4）修改lisi的Math成绩，改为95。**

hset命令修改



hget查询lisi的数学成绩来验证结果



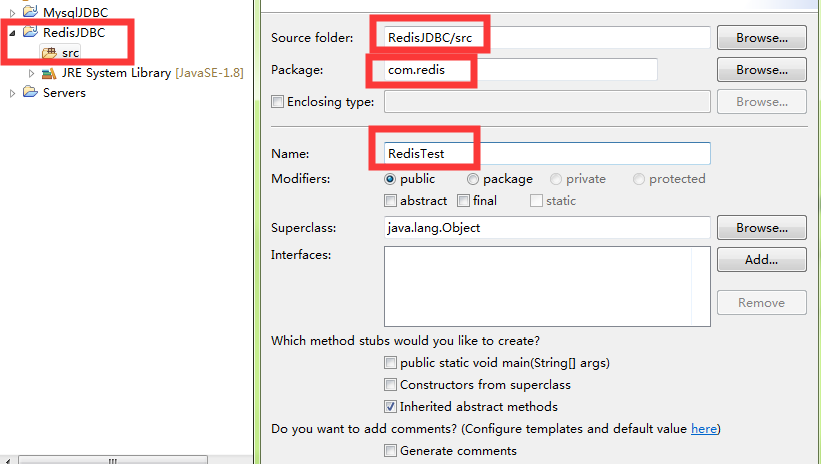
**2.根据上面已经设计出的学生表Student，用Redis的JAVA客户端编程(jedis)，实现如下操作：**

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

**该数据对应的键值对形式如下：**

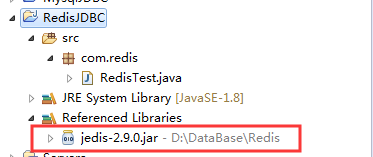
|  |
| --- |
| **scofield:｛**  **English: 45**  **Math: 89**  **Computer: 100**  **｝** |

新建一个项目RedisJDBC，然后再建redis包，在包中新建java文件RedisTest



去菜鸟教程中：<https://www.runoob.com/redis/redis-java.html>

下载redis的jar包并导入：



JAVA代码：

package com.redis;

import java.util.Map;

import redis.clients.jedis.Jedis;

public class RedisTest {

public static void main(String[] args) {

Jedis jedis = new Jedis("localhost");

jedis.auth("ydy19990616");

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");

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

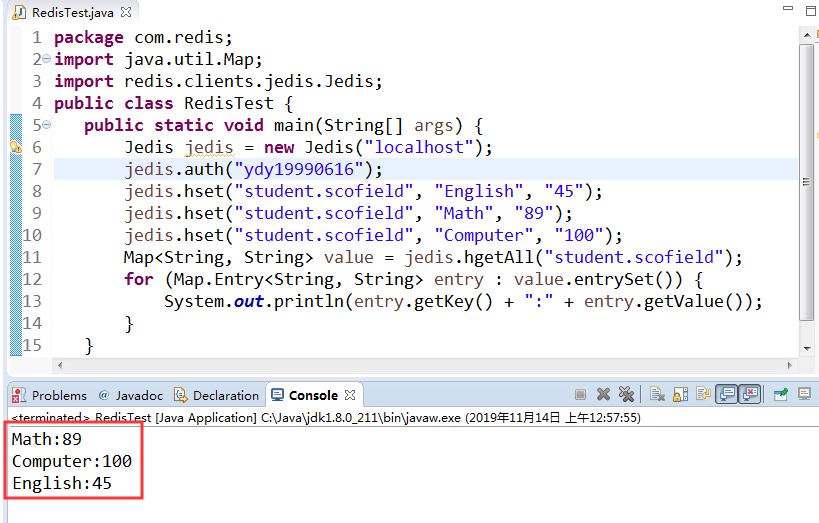
System.out.println(entry.getKey() + ":" + entry.getValue());

}

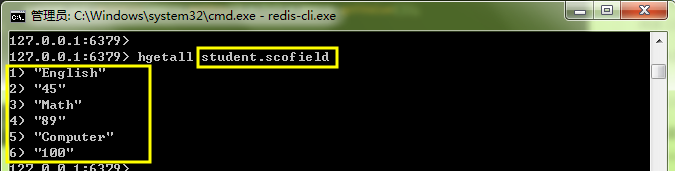
}

}

运行成功：



在redis中验证结果：hgetall student.scofield



**（2）获取scofield的English成绩信息**

新建java文件RedisTest2

JAVA代码：

package com.redis;

import redis.clients.jedis.Jedis;

public class RedisTest2 {

public static void main(String[] args) {

Jedis jedis = new Jedis("localhost");

jedis.auth("ydy19990616");

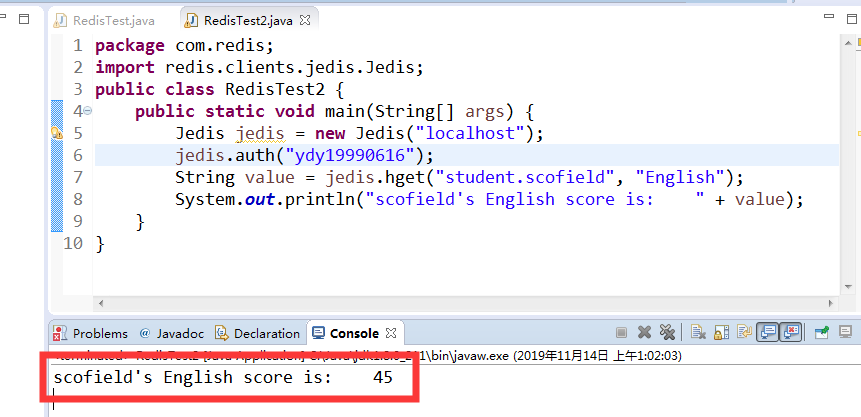
String value = jedis.hget("student.scofield", "English");

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

}

}

运行成功，获取到scofield的English成绩信息：

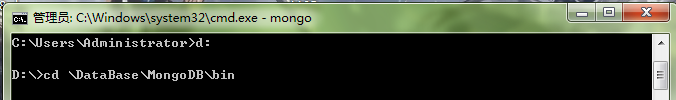


**（四）MongoDB数据库操作**

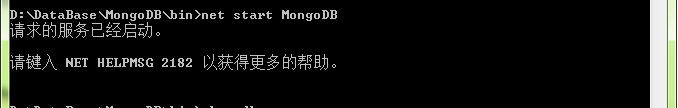
实验前提：安装MongoDB，

安装教程：<https://www.cnblogs.com/zhoulifeng/p/9429597.html>

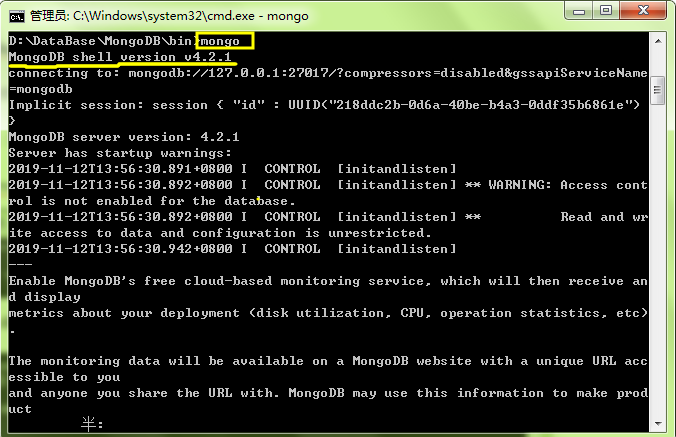
在cmd中进入MongoDB安装路径下的bin目录中



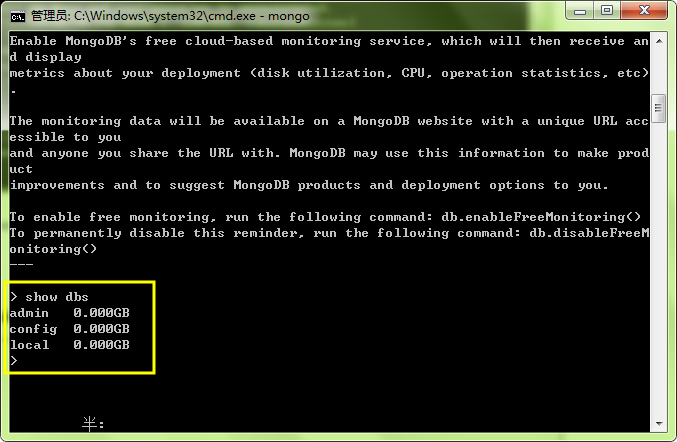
输入命令 net start MongoDB 来启动MongoDB服务



依旧在bin中，输入mongo启动



show dbs  显示当前所有的数据，可以看出现在没有数据内容



接下来就可以开始做实验题目了

**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：**

**（1）用MongoDB Shell设计出student集合；**

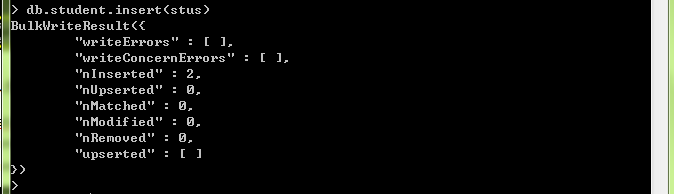
首先输入命令：use student 创建数据库名字，  student就是数据库名字



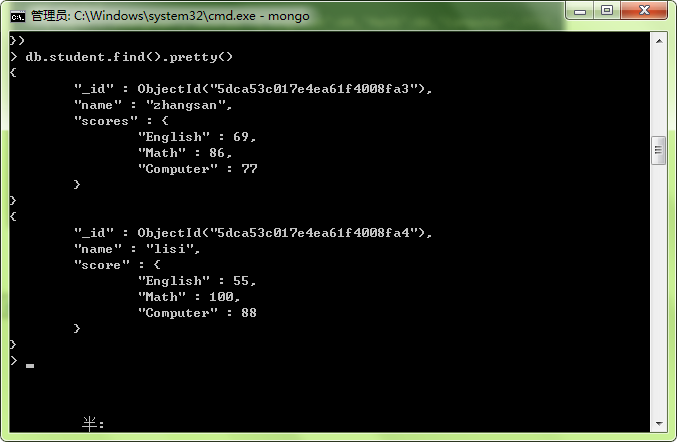
输入给出的Student文档中的数据



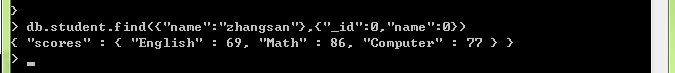
最后输入命令db.student.insert(stus) 将以上数据插入student数据库



**（2）用find()方法输出两个学生的信息；**



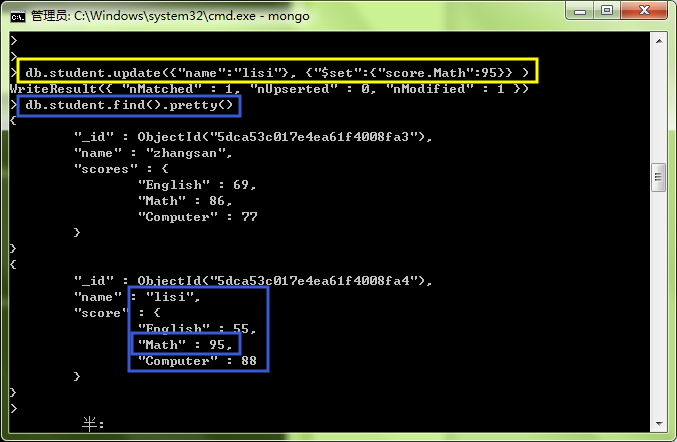
**（3）用find()方法查询zhangsan的所有成绩(只显示score列)；**



**（4）修改lisi的Math成绩，改为95。**

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

然后，再输入命令db.student.find().pretty() 来查看lisi的Math成绩是否成功修改为95



可见，命令执行成功，成功将lisi的数学成绩修改为95

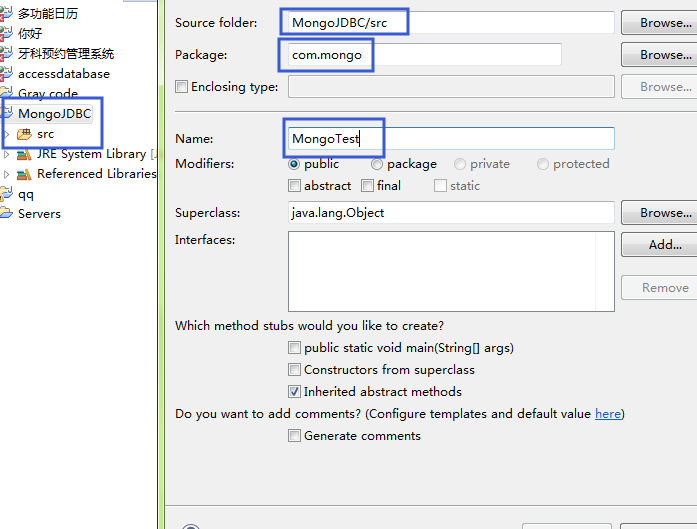
**2.根据上面已经设计出的Student集合，用MongoDB的Java客户端编程，实现如下操作：**

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

**与上述数据对应的文档形式如下：**

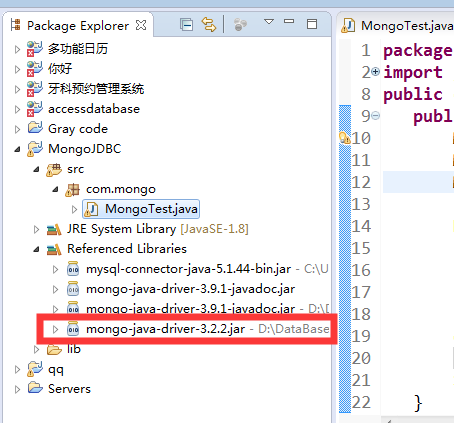
|  |
| --- |
| **{**  **“name”: “scofield”,**  **“score”: {**  **“English”: 45,**  **“Math”: 89,**  **“Computer”: 100**  **}**  **}** |

新建一个项目MongoJDBC，然后再建mongo包，在包中新建java文件MongoTest



去菜鸟教程中：<https://www.runoob.com/mongodb/mongodb-java.html>

下载mongo jar包并导入：



JAVA代码：

package com.mongo;

import java.util.ArrayList;

import java.util.List;

import org.bson.Document;

import com.mongodb.MongoClient;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

public class MongoTest {

public static void main(String[] args) {

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

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

MongoCollection<Document> collection = mongoDatabase

.getCollection("student");

Document document = new Document("name", "scofield").append(

"score",

new Document("English", 45).append("Math", 89).append(

"Computer", 100));

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

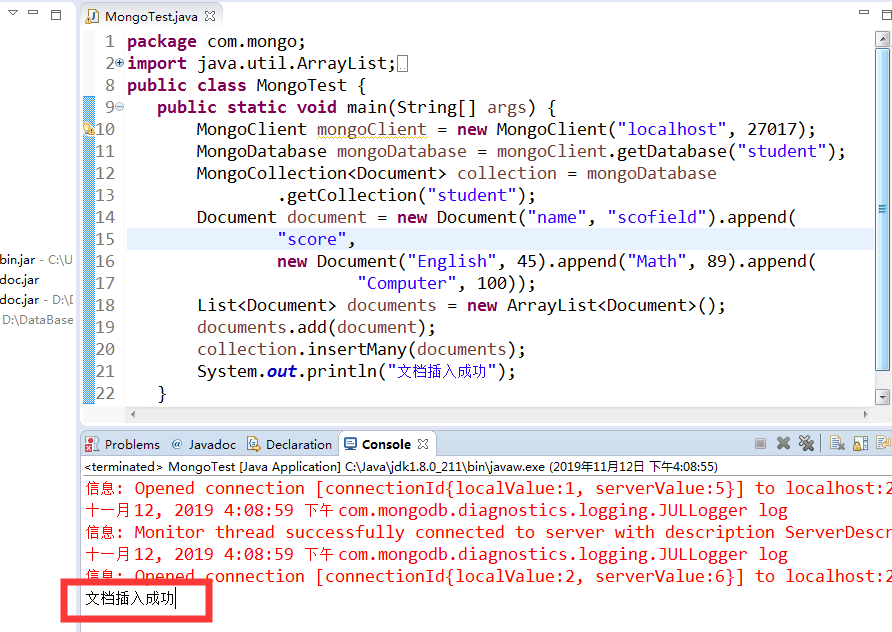
documents.add(document);

collection.insertMany(documents);

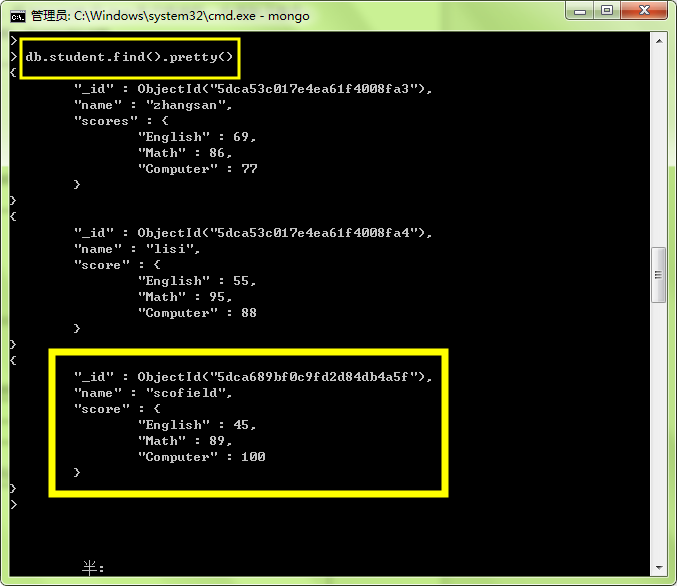
System.out.println("文档插入成功");

}

运行成功：



在mongo中用find（）方法验证结果



**（2）获取scofield的所有成绩信息(只显示score列)**

新建java文件MongoTest2

JAVA代码：

package com.mongo;

import org.bson.Document;

import com.mongodb.MongoClient;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.model.Filters;

import static com.mongodb.client.model.Filters.eq;

public class MongoTest2 {

public static void main(String[] args) {

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

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

MongoCollection<Document> collection =

mongoDatabase.getCollection("student");

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

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

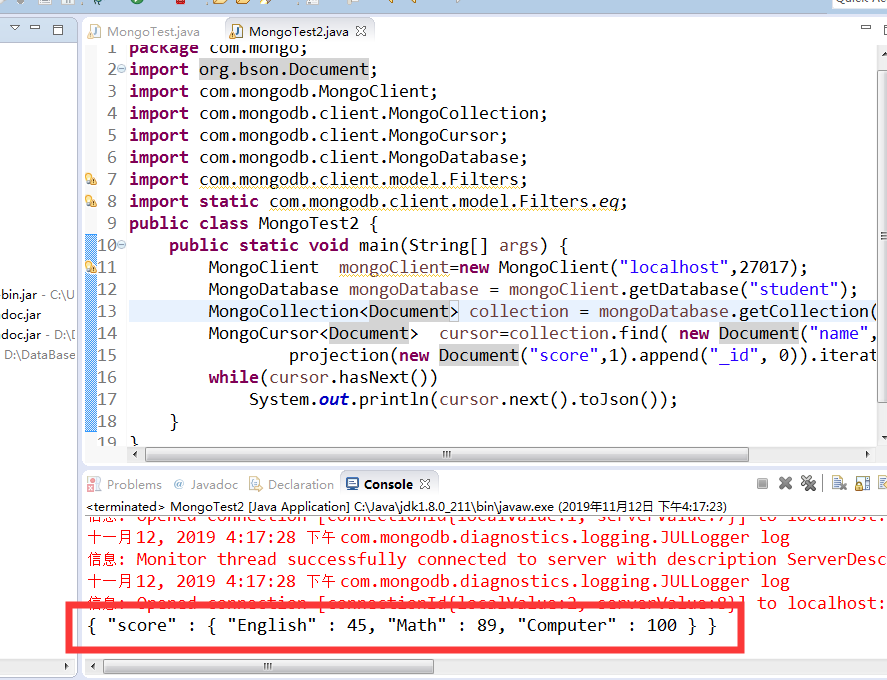
while(cursor.hasNext())

System.out.println(cursor.next().toJson());

}

}

运行成功，获取scofield的所有成绩信息如下：



**总结**

除了Hbase是在云服务器上之外，其它三种数据库实验我都是在windows平台上完成的。本次实验用到的4种数据库中，只有MySQL是关系型数据库，而Hbase、Redis、MongoDB都是NoSQL型的数据库（即非关系型数据库）。

|  |  |
| --- | --- |
| **关系型数据库** | MySQL是最著名的关系型数据库之一，最典型的数据结构是表，由二维表及其之间的联系所组成的一个数据组织，它的优点是：易于维护、使用方便、支持复杂查询。  但随着时代的发展，在数据量如井喷一般爆发的背景下，MySQL在进行海量的数据操作时显得力不从心。固定的表结构，使得其灵活度稍欠；高并发读写需求，对于传统关系型数据库来说，硬盘I/O是一个很大的瓶颈。 |
|  |

|  |  |
| --- | --- |
| **非关系型数据库** | 为了适应海量数据的时代，非关系型数据库在这种背景下应运而生。对比起传统的关系型数据库，它的优点表现在1、格式灵活：存储数据的格式各种类型、支持文档、图像等等，而关系型数据库只支持基础类型；2、速度快：nosql载体不局限于硬盘；3、高扩展性；  4、成本低：nosql数据库部署简单，基本都是开源软件。  缺点：1、不提供sql支持，学习和使用成本较高；2、无事务处理；3、数据结构相对复杂，复杂查询方面稍欠。  在nosql的基础上，左侧的三种数据库在存储数据的种类上又进一步地细分为列式、key-value型以及文档型，与hbase、redis、mongoDB一 一相对。  **列式nosql：**典型应用于分布式的文件系统，以hbase为例，它是以列簇式来进行存储（也就是将同一列数据存在一起）。  **key-value型nosql：**典型应用于内容缓存，处理大量数据的高访问负载；以redis为例，键key是唯一的，指定专属的值value，组成一对键值对。  **文档型nosql：**典型应用于Web应用，以MongoDB为例，它的数据模型类似于key-value的键值对，只不过与redis不同的是数据库并不了解value的内容 |
|  |
|  |
|  |