福建師範大學
FUJIAN NORMAL UNIVERSITY

计算机与网络空间安全学院学生实验报告

实验课程名称：大数据导论 教师：林鑫泓

实验名称	NoSQL 和关系数据库的操作比较			实验成绩	
学生姓名	叶建行	学 号	116052020005	年级专业班级	2020 级软件工程一班
小组成员	无			实验日期	2022 年 11 月 9

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：Linux（建议 Ubuntu16.04）；
- Hadoop 版本：3.3.1；
- MySQL 版本：5.6；
- HBase 版本：1.1.2；
- Redis 版本：3.0.6；
- MongoDB 版本：3.2.6；
- JDK 版本：1.7 或以上版本；
- Java IDE：idea；
- 。

1.3 实验要求

- 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。

2 实验记录

（一）MySQL 数据库操作

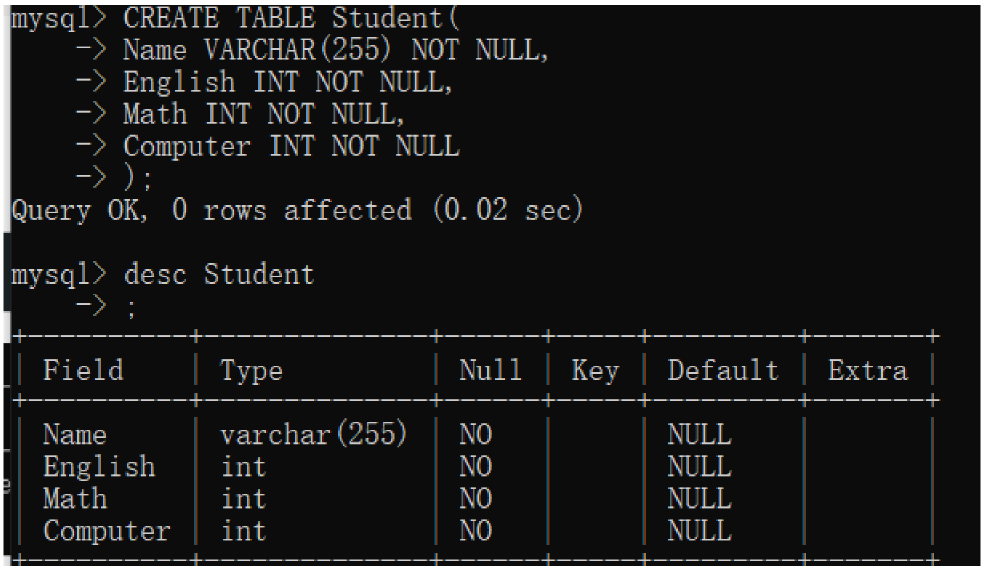
学生表 Student

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：

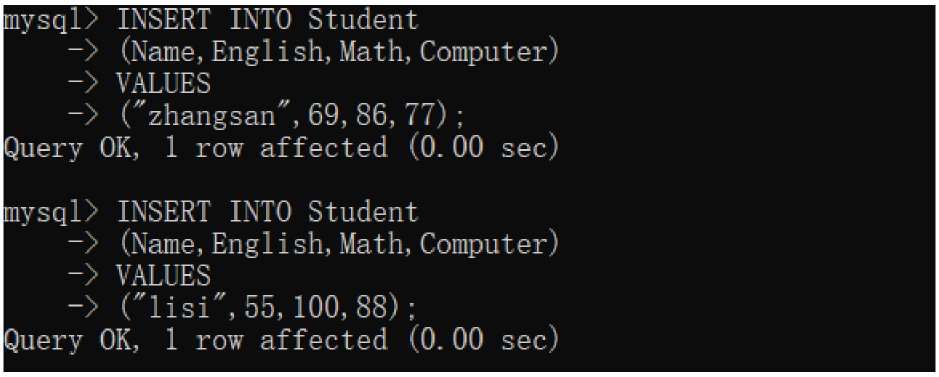
（1）　在 MySQL 中创建 Student 表，并录入数据；

1.　创建 Student 表



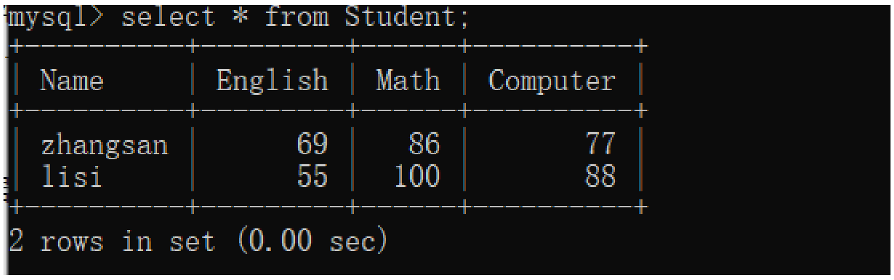
```
mysql> CREATE TABLE Student(
    -> Name VARCHAR(255) NOT NULL,
    -> English INT NOT NULL,
    -> Math INT NOT NULL,
    -> Computer INT NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc Student
    -> ;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Name     | varchar(255) | NO   |     | NULL    |       |
| English  | int          | NO   |     | NULL    |       |
| Math     | int          | NO   |     | NULL    |       |
| Computer | int          | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

2.　插入数据



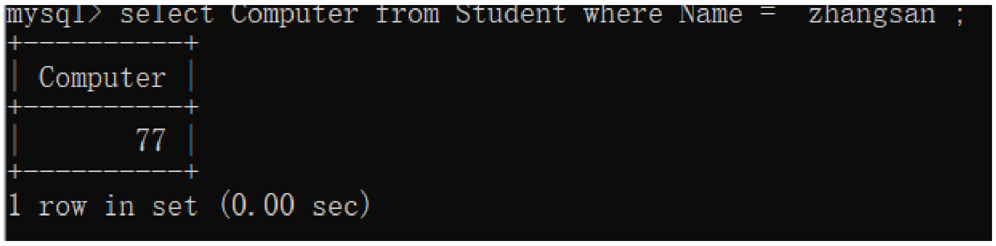
```
mysql> INSERT INTO Student
    -> (Name,English,Math,Computer)
    -> VALUES
    -> ("zhangsan",69,86,77);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Student
    -> (Name,English,Math,Computer)
    -> VALUES
    -> ("lisi",55,100,88);
Query OK, 1 row affected (0.00 sec)
```

（2）　用 SQL 语句输出 Student 表中的所有记录；



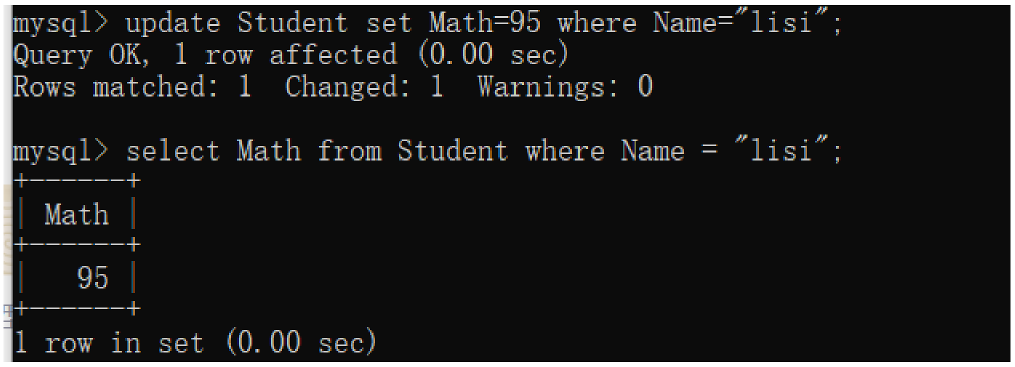
```
mysql> select * from Student;
+----------+---------+------+----------+
| Name     | English | Math | Computer |
+----------+---------+------+----------+
| zhangsan |      69 |   86 |       77 |
| lisi     |      55 |  100 |       88 |
+----------+---------+------+----------+
2 rows in set (0.00 sec)
```

（3） 查询 zhangsan 的 Computer 成绩；



```
mysql> select Computer from Student where Name =  zhangsan ;
+----------+
| Computer |
+----------+
|       77 |
+----------+
1 row in set (0.00 sec)
```

（4） 修改 lisi 的 Math 成绩，改为 95。



```
mysql> update Student set Math=95 where Name="lisi";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select Math from Student where Name = "lisi";
+------+
| Math |
+------+
|   95 |
+------+
1 row in set (0.00 sec)
```

2.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：

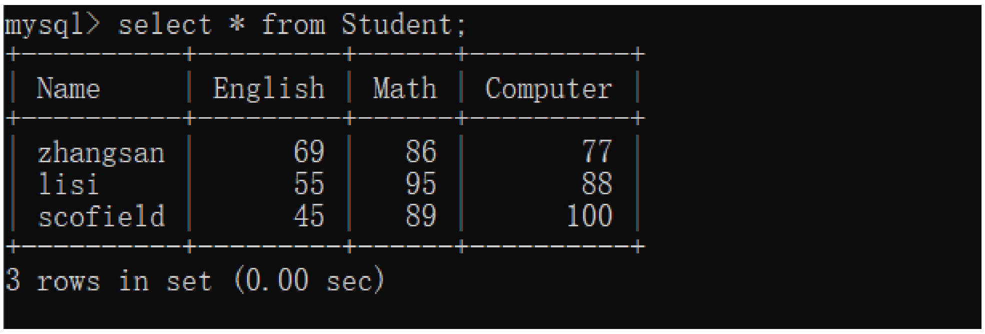
scofield	45	89	100

实验代码

```
  package org.example;

  import java.sql.Connection;
  import java.sql.DriverManager;
  import java.sql.PreparedStatement;
  import java.sql.SQLException;

  public class LinkDatabaseInsert {
      public static void main(String[] args) throws
ClassNotFoundException, SQLException {
          //1.注册数据库的驱动
          Class.forName("com.mysql.jdbc.Driver");
          //2.获取数据库连接（里面内容依次是："jdbc:mysql://主机名:端口号/数
据库名","用户名","登录密码"）
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/hello","root",
"123456");
          //3.需要执行的sql 语句（?是占位符，代表一个参数）
          String sql = "insert into Student(Name,English,Math,Computer)
values(?,?,?,?)";
          //4.获取预处理对象，并依次给参数赋值
```

```
        PreparedStatement statement = connection.prepareCall(sql);
        statement.setString(1,"scofield"); //数据库字段类型是String，就是setString；1代表第一个参数
        statement.setInt(2,45);    //数据库字段类型是int，就是setInt；2代表第二个参数
        statement.setInt(3,89); //数据库字段类型是int，就是setInt；3代表第三个参数
        statement.setInt(4,100); //数据库字段类型是int，就是setInt；4代表第四个参数
        //5.执行sql语句（执行了几条记录，就返回几）
        int i = statement.executeUpdate();
        System.out.println(i);
        //6.关闭jdbc连接
        statement.close();
        connection.close();
    }
}
```

实验结果（表添加了一条数据）



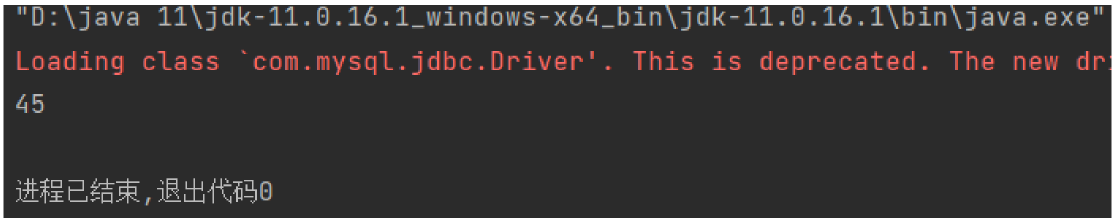
```
mysql> select * from Student;
+----------+---------+------+----------+
| Name     | English | Math | Computer |
+----------+---------+------+----------+
| zhangsan |      69 |   86 |       77 |
| lisi     |      55 |   95 |       88 |
| scofield |      45 |   89 |      100 |
+----------+---------+------+----------+
3 rows in set (0.00 sec)
```

（2）获取 scofield 的 English 成绩信息

实验代码：

```
package org.example;

import java.sql.*;
public class JDBC {
    public static void main(String[] args) {
        //private static final String URL="jdbc:mysql://localhost:3306/数据库名";//jdbc:mysql//服务器地址/数据库名
        //private static final String USER="用户名";//用户名
        //private static final String PASSWORD="密码";//密码
        try {
            //1.加载驱动程序
            //此语句固定，使用MySQL 数据库无需更改，在JSP 中可不加异常处理
            Class.forName("com.mysql.jdbc.Driver");
            //2.获取数据库的连接
            //此语句只需更改端口、数据库名称、用户名及密码，使用MySQL 数据
库格式固定，在JSP 中可不加异常处理
            //可以在括号内使用上述注释的URL、USER、PASSWORD
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/hello","root",
"123456");
            //3.通过数据库的连接操作数据库，实现查找数据
            Statement sql = con.createStatement();
            ResultSet rs = sql.executeQuery("select English from
Student where Name='scofield'");//其后可以加where 语句限制
            while (rs.next()) {
                System.out.println(rs.getString("English"));//表单名，
即表头//数据库设计时，表头使用英文
            }
            con.close();
        }
        catch(Exception E) {
            System.out.println("SQL 异常！！！！");
        }
    }
}
```

实验结果：



```
"D:\java 11\jdk-11.0.16.1_windows-x64_bin\jdk-11.0.16.1\bin\java.exe"
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new dr
45

进程已结束,退出代码0
```

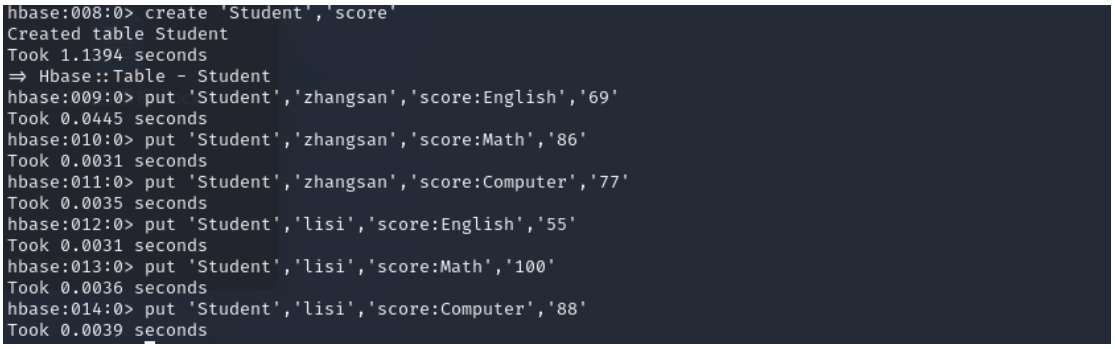
（二）HBase 数据库操作

学生表 Student

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

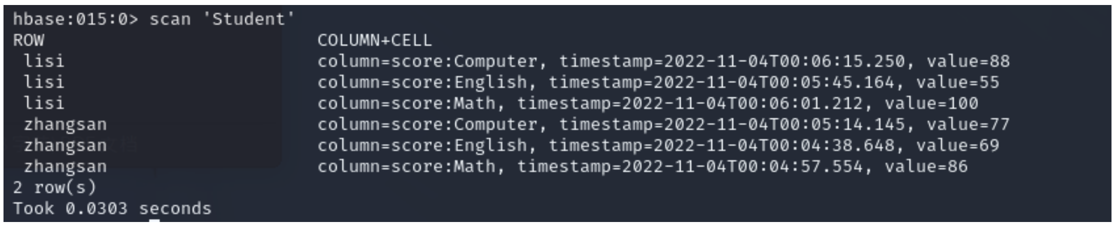
1. 根据上面给出的学生表 Student 的信息，执行如下操作：

（1） 用 Hbase Shell 命令创建学生表 Student；



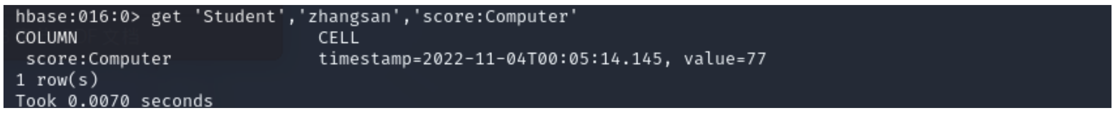
```
hbase:008:0> create 'Student','score'
Created table Student
Took 1.1394 seconds
=> Hbase::Table - Student
hbase:009:0> put 'Student','zhangsan','score:English','69'
Took 0.0445 seconds
hbase:010:0> put 'Student','zhangsan','score:Math','86'
Took 0.0031 seconds
hbase:011:0> put 'Student','zhangsan','score:Computer','77'
Took 0.0035 seconds
hbase:012:0> put 'Student','lisi','score:English','55'
Took 0.0031 seconds
hbase:013:0> put 'Student','lisi','score:Math','100'
Took 0.0036 seconds
hbase:014:0> put 'Student','lisi','score:Computer','88'
Took 0.0039 seconds
```

（2） 用 scan 命令浏览 Student 表的相关信息；



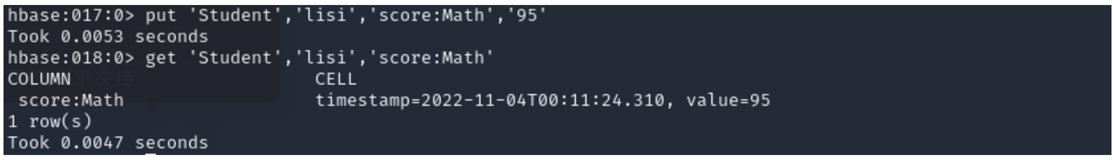
```
hbase:015:0> scan 'Student'
ROW                          COLUMN+CELL
 lisi                        column=score:Computer, timestamp=2022-11-04T00:06:15.250, value=88
 lisi                        column=score:English, timestamp=2022-11-04T00:05:45.164, value=55
 lisi                        column=score:Math, timestamp=2022-11-04T00:06:01.212, value=100
 zhangsan                    column=score:Computer, timestamp=2022-11-04T00:05:14.145, value=77
 zhangsan                    column=score:English, timestamp=2022-11-04T00:04:38.648, value=69
 zhangsan                    column=score:Math, timestamp=2022-11-04T00:04:57.554, value=86
2 row(s)
Took 0.0303 seconds
```

（3） 查询 zhangsan 的 Computer 成绩；



```
hbase:016:0> get 'Student','zhangsan','score:Computer'
COLUMN                       CELL
 score:Computer              timestamp=2022-11-04T00:05:14.145, value=77
1 row(s)
Took 0.0070 seconds
```

（4）修改 lisi 的 Math 成绩，改为 95。



```
hbase:017:0> put 'Student','lisi','score:Math','95'
Took 0.0053 seconds
hbase:018:0> get 'Student','lisi','score:Math'
COLUMN                       CELL
 score:Math                  timestamp=2022-11-04T00:11:24.310, value=95
1 row(s)
Took 0.0047 seconds
```

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

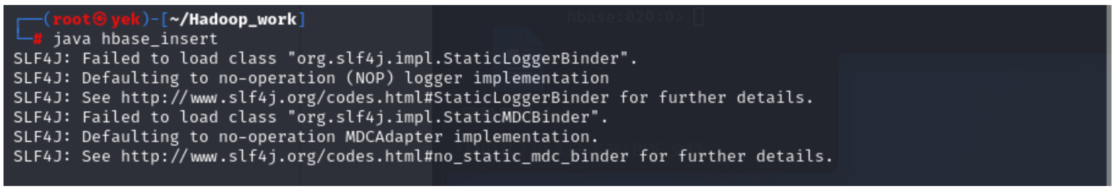
实验代码：

```
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;
public class hbase_insert {
    /**
    * @param args
    */
```

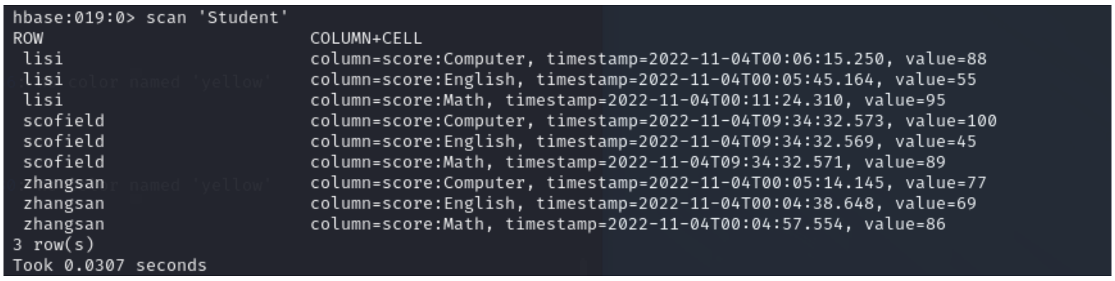
```
      public static Configuration configuration;
      public static Connection connection;
      public static Admin admin;
      public static void main(String[] args) {
          // TODO Auto-generated method stub
          configuration = HBaseConfiguration.create();
          configuration.set("hbase.rootdir","hdfs://localhost:9000/hbase
");
          try{
              connection =
ConnectionFactory.createConnection(configuration);
              admin = connection.getAdmin();
          }catch (IOException e){
              e.printStackTrace();
          }
          try {
              insertRow("Student","scofield","score","English","45");
              insertRow("Student","scofield","score","Math","89");
              insertRow("Student","scofield","score","Computer","100");
          } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
          }
              close();
          }
      public static void insertRow(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 close(){
          try{
              if(admin != null){
              admin.close();
          }
          if(null != connection){
              connection.close();
          }
```

```
        }catch (IOException e){
            e.printStackTrace();
        }
        }
        }
```

实验结果：



```
┌──(root㉿yek)-[~/Hadoop_work]
└─# java hbase_insert
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
```



```
hbase:019:0> scan 'Student'
ROW                            COLUMN+CELL
 lisi                          column=score:Computer, timestamp=2022-11-04T00:06:15.250, value=88
 lisi                          column=score:English, timestamp=2022-11-04T00:05:45.164, value=55
 lisi                          column=score:Math, timestamp=2022-11-04T00:11:24.310, value=95
 scofield                      column=score:Computer, timestamp=2022-11-04T09:34:32.573, value=100
 scofield                      column=score:English, timestamp=2022-11-04T09:34:32.569, value=45
 scofield                      column=score:Math, timestamp=2022-11-04T09:34:32.571, value=89
 zhangsan                      column=score:Computer, timestamp=2022-11-04T00:05:14.145, value=77
 zhangsan                      column=score:English, timestamp=2022-11-04T00:04:38.648, value=69
 zhangsan                      column=score:Math, timestamp=2022-11-04T00:04:57.554, value=86
3 row(s)
Took 0.0307 seconds
```

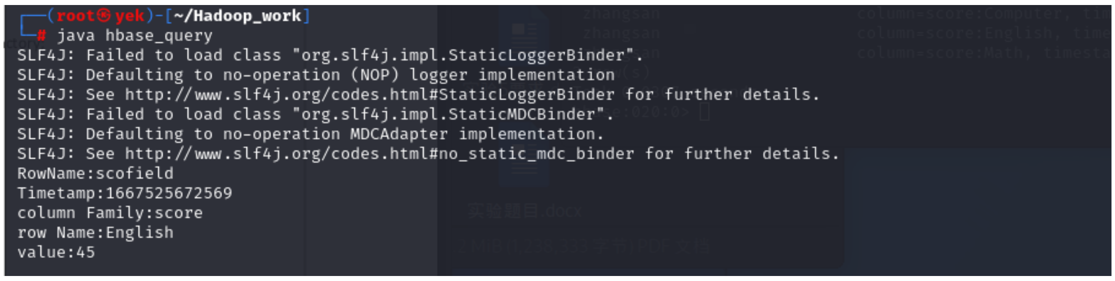
（2）获取 scofield 的 English 成绩信息。

实验代码

```
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.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Table;
public class hbase_query {
    /**
    * @param args
    */
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        configuration = HBaseConfiguration.create();
```

```
          configuration.set("hbase.rootdir","hdfs://localhost:9000/hbase
");
          try{
              connection =
ConnectionFactory.createConnection(configuration);
              admin = connection.getAdmin();
          }catch (IOException e){
              e.printStackTrace();
          }
          try {
              getData("Student","scofield","score","English");
          } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
          }
              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);
          showCell(result);
          table.close();
      }
      public static void showCell(Result result){
          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))+" ");
      }
      }
      public static void close(){
          try{
              if(admin != null){
              admin.close();
          }
          if(null != connection){
              connection.close();
          }
          }catch (IOException e){
              e.printStackTrace();
          }
          }
      }
```

实验结果：



```
┌──(root㉿yek)-[~/Hadoop_work]
└─# java hbase_query
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
RowName:scofield
Timetamp:1667525672569
column Family:score
row Name:English
value:45
```

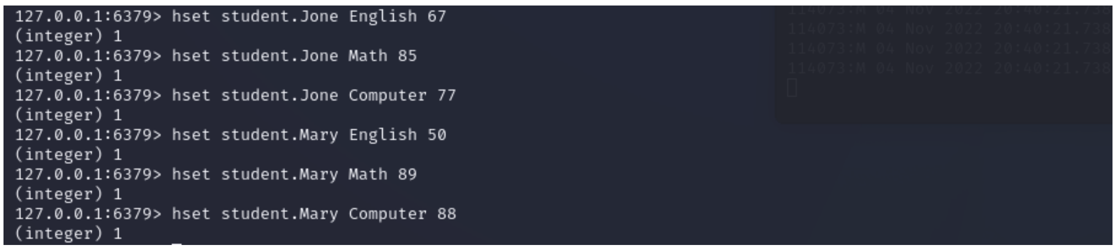
（三）Redis 数据库操作

Student 键值对如下：

```
Jone: {
        English: 67
        Math: 85
        Computer: 77
}
Mary: {
        English: 50
        Math: 89
        Computer: 88
}
```

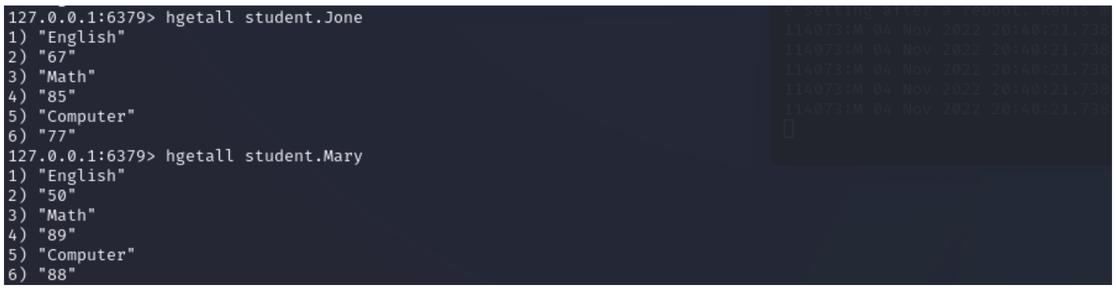
1. 根据上面给出的键值对，完成如下操作：

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



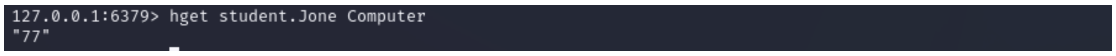
```
127.0.0.1:6379> hset student.Jone English 67
(integer) 1
127.0.0.1:6379> hset student.Jone Math 85
(integer) 1
127.0.0.1:6379> hset student.Jone Computer 77
(integer) 1
127.0.0.1:6379> hset student.Mary English 50
(integer) 1
127.0.0.1:6379> hset student.Mary Math 89
(integer) 1
127.0.0.1:6379> hset student.Mary Computer 88
(integer) 1
```

（2）用 hgetall 命令分别输出 Jone 和 Mary 的成绩信息；



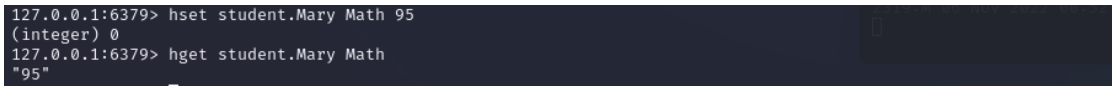
```
127.0.0.1:6379> hgetall student.Jone
1) "English"
2) "67"
3) "Math"
4) "85"
5) "Computer"
6) "77"
127.0.0.1:6379> hgetall student.Mary
1) "English"
2) "50"
3) "Math"
4) "89"
5) "Computer"
6) "88"
```

（3）用 hget 命令查询 Jone 的 Computer 成绩；



```
127.0.0.1:6379> hget student.Jone Computer
"77"
```

（4）修改 Mary 的 Math 成绩，改为 95。



```
127.0.0.1:6379> hset student.Mary Math 95
(integer) 0
127.0.0.1:6379> hget student.Mary Math
"95"
```

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

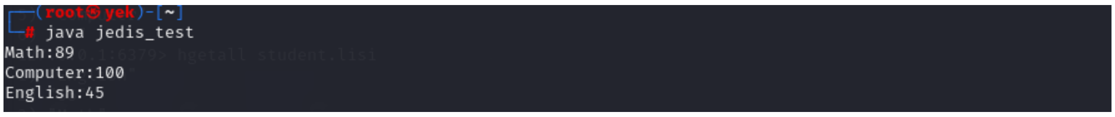
该数据对应的键值对形式如下：

```
scofield: {
                    English: 45
                    Math: 89
                    Computer: 100
}
```

实验代码：

```
import java.util.Map;
import redis.clients.jedis.Jedis;
public class jedis_test {
    /**
    * @param args
    */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        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");
        for(Map.Entry<String, String> entry:value.entrySet())
        {
            System.out.println(entry.getKey()+":"+entry.getValue());
    }
    }
}
```

实验结果：



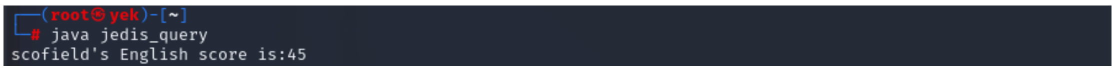
```
┌──(root㉿yek)-[~]
└─# java jedis_test
Math:89
Computer:100
English:45
```

（2）获取 scofield 的 English 成绩信息

实验代码：

```
import java.util.Map;
import redis.clients.jedis.Jedis;
public class jedis_query {
    /**
    * @param args
    */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Jedis jedis = new Jedis("localhost");
        String value=jedis.hget("student.scofield", "English");
        System.out.println("scofield's English score is:"+value);
        }
}
```

实验结果：



```
┌──(root㉿yek)-[~]
└─# java jedis_query
scofield's English score is:45
```

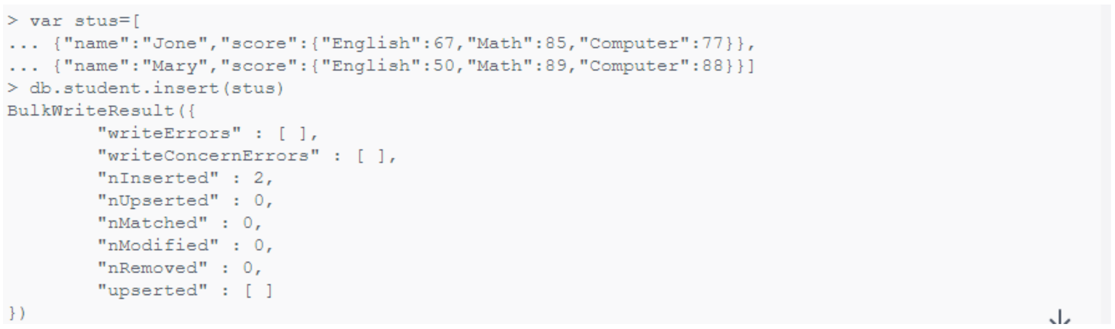
（四）**MongoDB 数据库操作**

Student 文档如下:

```
            {
                  “name”: “Jone”,
                  “score”: {
                       “English”: 67,
                       “Math”: 85,
                       “Computer”: 77
                  }
            }
            {
                  “name”: “Mary”,
                  “score”: {
                       “English”: 50,
                       “Math”: 89,
                       “Computer”: 88
                  }
}
```

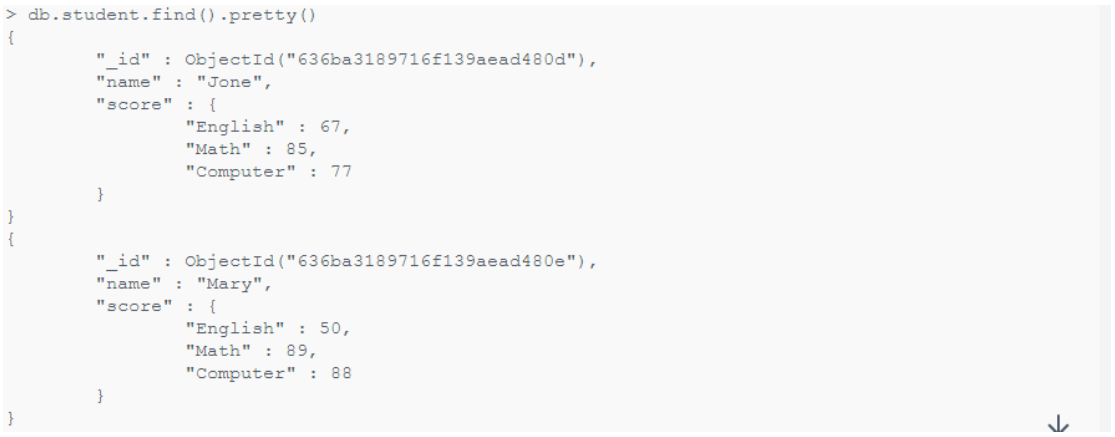
1.根据上面给出的文档，完成如下操作：

（1）用 MongoDB Shell 设计出 student 集合；



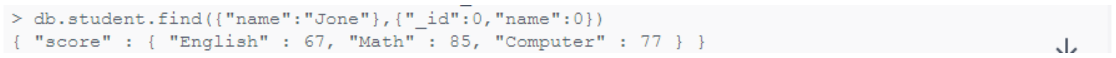
```
> var stus=[
... {"name":"Jone","score":{"English":67,"Math":85,"Computer":77}},
... {"name":"Mary","score":{"English":50,"Math":89,"Computer":88}}]
> db.student.insert(stus)
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

（2）　用 find()方法输出两个学生的信息；



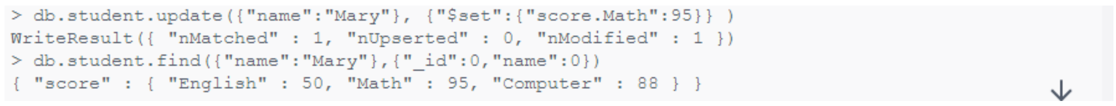
```
> db.student.find().pretty()
{
        "_id" : ObjectId("636ba3189716f139aead480d"),
        "name" : "Jone",
        "score" : {
                "English" : 67,
                "Math" : 85,
                "Computer" : 77
        }
}
{
        "_id" : ObjectId("636ba3189716f139aead480e"),
        "name" : "Mary",
        "score" : {
                "English" : 50,
                "Math" : 89,
                "Computer" : 88
        }
}
```

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



```
> db.student.find({"name":"Jone"},{"_id":0,"name":0})
{ "score" : { "English" : 67, "Math" : 85, "Computer" : 77 } }
```

（4） 修改 Mary 的 Math 成绩，改为 95。



```
> db.student.update({"name":"Mary"}, {"$set":{"score.Math":95}} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find({"name":"Mary"},{"_id":0,"name":0})
{ "score" : { "English" : 50, "Math" : 95, "Computer" : 88 } }
```

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 scofield 数据：English:45 Math:89 Computer:100

与上述数据对应的文档形式如下：

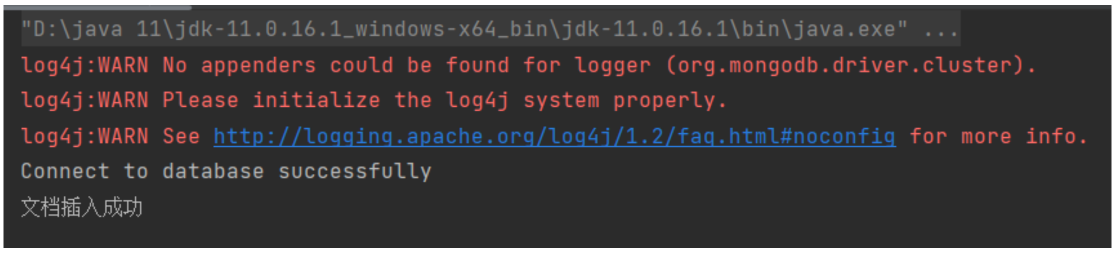
```
{
        "name": "scofield",
        "score": {
            "English": 45,
            "Math": 89,
            "Computer": 100
        }
}
```

实验代码：

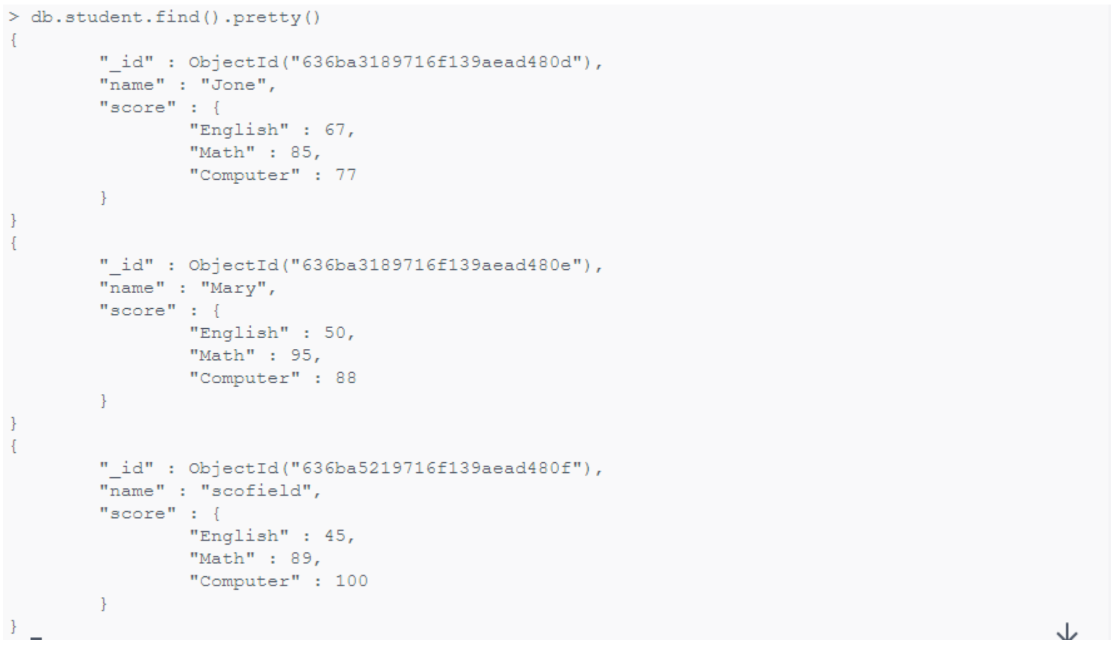
```
package org.example;

import java.util.ArrayList;
import java.util.List;

import com.mongodb.*;
import org.bson.Document;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
public class mongo_learn {
    private static final String MONGO_HOST = "localhost";
    private static final Integer MONGO_PORT = 27021;
    private static final String MONGO_USERNAME = "student";
    private static final String MONGO_PASSWORD = "123456";
    private static final String MONGO_DB_NAME = "student";
    private static final String MONGO_COLLECTION_NAME = "mongo-
collection-test";

    /**
     * @param args
```

```
       */
      public static void main(String[] args) {
  // TODO Auto-generated method stub
  //实例化一个 mongo 客户端
          //连接到MongoDB 服务 如果是远程连接可以替换“localhost”为服务器所在
IP 地址
          //ServerAddress()两个参数分别为 服务器地址 和 端口
          ServerAddress serverAddress = new
ServerAddress(MONGO_HOST,MONGO_PORT);
          List<ServerAddress> addrs = new ArrayList<ServerAddress>();
          addrs.add(serverAddress);
          //MongoCredential.createScramSha1Credential()三个参数分别为 用
户名 数据库名称 密码
          MongoCredential credential =
MongoCredential.createScramSha1Credential(MONGO_USERNAME,  MONGO_DB_NAME,
MONGO_PASSWORD.toCharArray());
          List<MongoCredential> credentials = new
ArrayList<MongoCredential>();
          credentials.add(credential);

          //通过连接认证获取MongoDB 连接
          MongoClient mongoClient = new MongoClient(addrs,credentials);

          //连接到数据库
          MongoDatabase mongoDatabase =
mongoClient.getDatabase(MONGO_DB_NAME);
          System.out.println("Connect to database successfully");

  //获取数据库中某个集合
          MongoCollection<Document> collection =
mongoDatabase.getCollection("student");
  //实例化一个文档,内嵌一个子文档
          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);
          //将文档插入集合中
          collection.insertMany(documents);
          System.out.println("文档插入成功");
      }}
```

实验结果：



```
"D:\java 11\jdk-11.0.16.1_windows-x64_bin\jdk-11.0.16.1\bin\java.exe" ...
log4j:WARN No appenders could be found for logger (org.mongodb.driver.cluster).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connect to database successfully
文档插入成功
```



```
> db.student.find().pretty()
{
        "_id" : ObjectId("636ba3189716f139aead480d"),
        "name" : "Jone",
        "score" : {
                "English" : 67,
                "Math" : 85,
                "Computer" : 77
        }
}
{
        "_id" : ObjectId("636ba3189716f139aead480e"),
        "name" : "Mary",
        "score" : {
                "English" : 50,
                "Math" : 95,
                "Computer" : 88
        }
}
{
        "_id" : ObjectId("636ba5219716f139aead480f"),
        "name" : "scofield",
        "score" : {
                "English" : 45,
                "Math" : 89,
                "Computer" : 100
        }
}
```

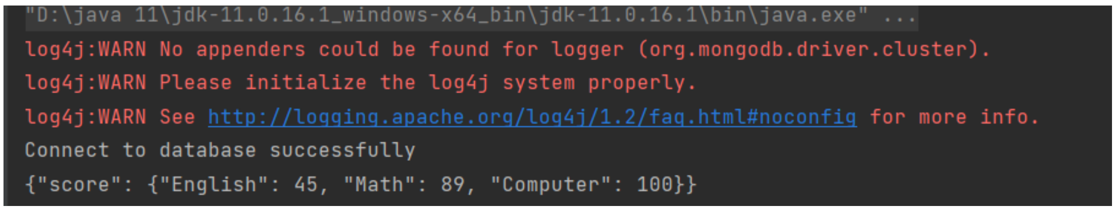
（2）获取 scofield 的所有成绩成绩信息(只显示 score 列)

实验代码：

```java
package org.example;

import java.util.ArrayList;
import java.util.List;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
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 mongo_query {
    private static final String MONGO_HOST = "localhost";
    private static final Integer MONGO_PORT = 27021;
    private static final String MONGO_USERNAME = "student";
    private static final String MONGO_PASSWORD = "123456";
```

```java
     private static final String MONGO_DB_NAME = "student";
     private static final String MONGO_COLLECTION_NAME = "mongo-
collection-test";
     /**
      * @param args
      */
     public static void main(String[] args) {
         // TODO Auto-generated method stub
  //实例化一个 mongo 客户端
         //连接到MongoDB 服务 如果是远程连接可以替换“localhost”为服务器所在
IP 地址
         //ServerAddress()两个参数分别为 服务器地址 和 端口
         ServerAddress serverAddress = new
ServerAddress(MONGO_HOST,MONGO_PORT);
         List<ServerAddress> addrs = new ArrayList<ServerAddress>();
         addrs.add(serverAddress);
         //MongoCredential.createScramSha1Credential()三个参数分别为 用
户名 数据库名称 密码
         MongoCredential credential =
MongoCredential.createScramSha1Credential(MONGO_USERNAME,  MONGO_DB_NAME,
MONGO_PASSWORD.toCharArray());
         List<MongoCredential> credentials = new
ArrayList<MongoCredential>();
         credentials.add(credential);

         //通过连接认证获取MongoDB 连接
         MongoClient mongoClient = new MongoClient(addrs,credentials);

         //连接到数据库
         MongoDatabase mongoDatabase =
mongoClient.getDatabase(MONGO_DB_NAME);
         System.out.println("Connect to database successfully");

  //获取数据库中某个集合
         MongoCollection<Document> collection =
mongoDatabase.getCollection("student");
  //进行数据查找,查询条件为 name=scofield, 对获取的结果集只显示 score 这个域
         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());
     }}
```

实验结果：



```
"D:\java 11\jdk-11.0.16.1_windows-x64_bin\jdk-11.0.16.1\bin\java.exe" ...
log4j:WARN No appenders could be found for logger (org.mongodb.driver.cluster).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connect to database successfully
{"score": {"English": 45, "Math": 89, "Computer": 100}}
```

3 实验总结

练习使用各个 Nosql 数据库的常用命令，加深了我对 Nosql 数据库的理解，以及对 Nosql 数据库结构的认知。通过 hbase Api 进行编程操作，加深 了我对 Nosql 结构的理解