

实验报告

**大数据处理技术**

报 告 人 姓 名 刘飞鸿

学 号 S2010W0748

学 科 专 业 电子信息

报告提交日期 2020年11月25日

# 实验 5 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 Ubuntu16.04）；
  + Hadoop 版本：2.7.1；
  + 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：Eclipse；

## 三、实验步骤

#### （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：
2. 在 MySQL 中创建 Student 表，并录入数据；
3. 用 SQL 语句输出 Student 表中的所有记录；
4. 查询 zhangsan 的 Computer 成绩；
5. 修改 lisi 的 Math 成绩，改为 95。
6. 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：
7. 向 Student 表中添加如下所示的一条记录：

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

1. 获取 scofield 的 English 成绩信息

#### （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：
2. 用 Hbase Shell 命令创建学生表 Student；
3. 用 scan 命令浏览 Student 表的相关信息；
4. 查询 zhangsan 的 Computer 成绩；
5. 修改 lisi 的 Math 成绩，改为 95。
6. 根据上面已经设计出的 Student 表，用 HBase API 编程实现以下操作：
7. 添加数据：English:45 Math:89 Computer:100

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

1. 获取 scofield 的 English 成绩信息。

#### （三）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）；

* 1. 用 hgetall 命令分别输出 zhangsan 和 lisi 的成绩信息；
  2. 用 hget 命令查询 zhangsan 的 Computer 成绩；
  3. 修改 lisi 的 Math 成绩，改为 95。

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：
2. 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

｝

1. 获取 scofield 的 English 成绩信息

#### （四）MongoDB 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：
   1. 用 MongoDB Shell 设计出 student 集合；
   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。
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：
3. 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

}

}

1. 获取 scofield 的所有成绩成绩信息(只显示 score 列)

## 四、实验报告

## MySQL 数据库操作

## 安装MySQL:

## 配置MySQL:

## 初始化配置:

## 不设置验证，设置密码为1234，不移除匿名用户，允许远程登陆，不移除测试数据库，重新加载权限表。

## 检查mysql服务状态:

## 根据上面给出的 Student 表，在 MySQL 数据库中完成如下操作：

## 在 MySQL 中创建 Student 表，并录入数据；

## 创建数据库并查看:

## 使用数据库:

## 

## 创建表并查看:

## 

## 录入数据并查看:

## 用 SQL 语句输出 Student 表中的所有记录；

## 查询 zhangsan 的 Computer 成绩；

## 

## 修改 lisi 的 Math 成绩，改为 95。

## 使用 MySQL 的 JAVA 客户端编程实现以下操作

## 给root用户开放授权远程访问:

## 向 Student 表中添加如下所示的一条记录

## 下载JDBC: 到www.mvnrepository.com网站中，搜索mysql，点击列表第一个，下载版本8.0.11，下载jar包

## 创建jdbc项目:

## 在项目里添加jar包:

## 

## 此时项目文件夹下出现这个jar包

## 书写TestInsert这个类如下:

## 

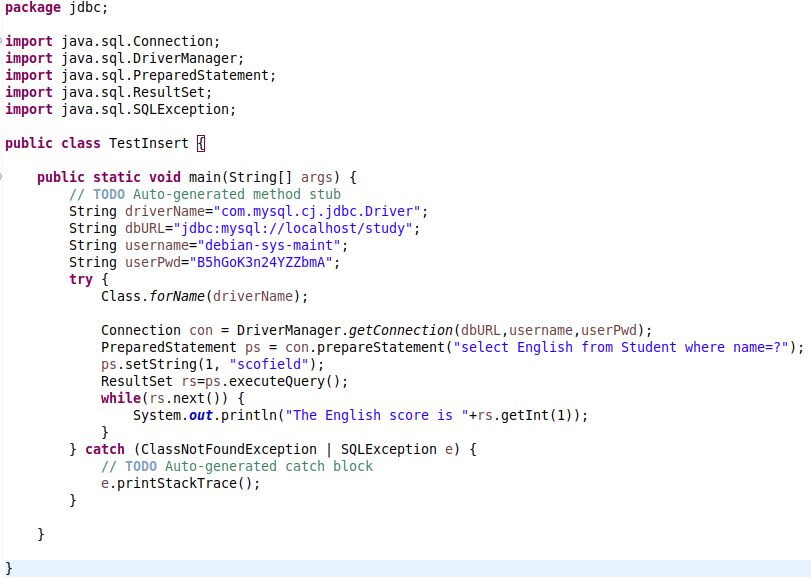
## 运行如下：这里并无输出打印值，但没报错，说明执行正确

## 

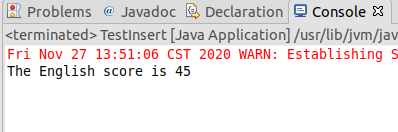
## 在shell端验证如下，成功！

## 

* + 获取 scofield 的 English 成绩信息
    1. 代码如下: //使用被授权的debian-sys-maint用户查询



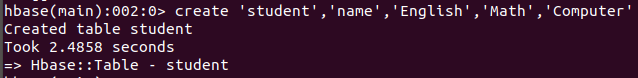
* + 1. 运行结果如下：

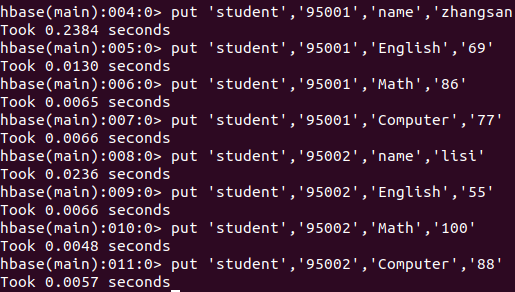


## Hbase数据库操作 //安装与配置在上个实验报告已阐述，这里略去

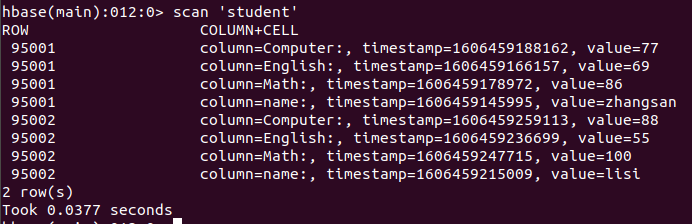
**根据上面给出的学生表 Student 的信息，执行如下操作：**

* + 用 Hbase Shell 命令创建学生表 Student；

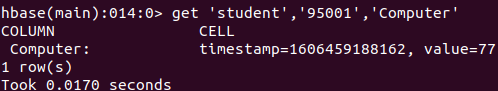
****

****

* + 用 scan 命令浏览 Student 表的相关信息；

****

* + 查询 zhangsan 的 Computer 成绩；

****

* + 修改 lisi 的 Math 成绩，改为 95。 //**包含查看验证**

## 

## 利用HBASE API完成以下操作:

* + 添加数据：English:45 Math:89 Computer:100

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

## 书写的代码如下:

|  |
| --- |
| package hdfs\_inputSteam;import org.apache.hadoop.conf.Configuration;import org.apache.hadoop.hbase.HBaseConfiguration;import org.apache.hadoop.hbase.HColumnDescriptor;import org.apache.hadoop.hbase.HTableDescriptor;import org.apache.hadoop.hbase.TableName;import org.apache.hadoop.hbase.client.\*;import java.io.IOException;public class CreateTable {public static Configuration configuration;public static Connection connection;public static Admin admin;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(), null, values[i].getBytes());table.put(put);}table.close();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 main(String[] args) {String[] fields = {"name:","English:","Math:","Computer:"};String[] values = {"scofield","45","89","100"};try {addRecord("student", "95003",fields,values);} catch (IOException e) {e.printStackTrace();}}} |

## 运行的结果如下: //可以看到事务成功建立了,完成后进程的关闭也显而易见

## 

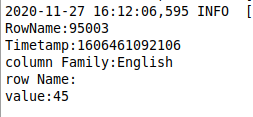
## 对运行结果进行验证如下:

## 

* + 获取 scofield 的 English 成绩信息。
    1. 代码如下：

|  |
| --- |
| package hdfs\_inputSteam;  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.\*;  import org.apache.hadoop.hbase.util.Bytes;  import java.io.IOException;  public class ScanColumn {  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  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();  }  public static void showCell(Result result) {  Cell[] cells = result.rawCells();  for (Cell cell : cells) {  String s=new String(CellUtil.cloneRow(cell));  if(s.equals("95003")) {  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 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();  }  }  // 关闭连接  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  try {  scanColumn("student", "English");  } catch (IOException e) {  e.printStackTrace();  }  }  } |

1. 运行的结果如下: //**结果与插入的数据抑制，正确！**

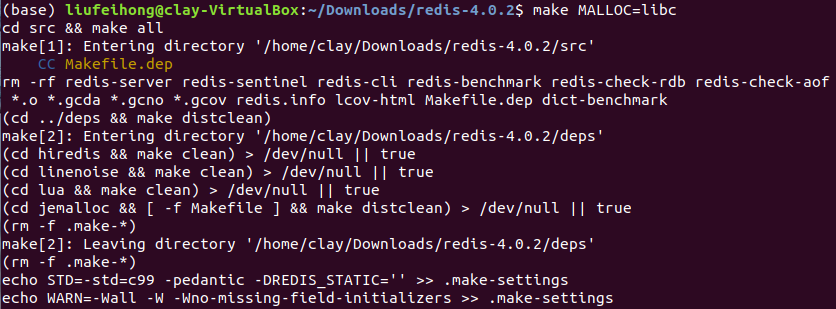


## Redis数据库操作

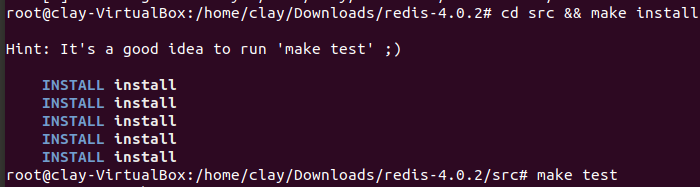
## 下载REDIS的源码并解压到安装目录，再切换到REDIS源代码目录

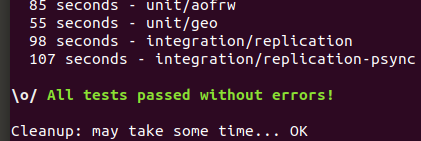
|  |
| --- |
| wget http://download.redis.io/releases/redis-4.0.2.tar.gztar -xf redis-4.0.2.tar.gz -C ~/cd redis-4.0.2/ |

1. 编译源代码



1. 编译安装并验证





1. 创建配置文件



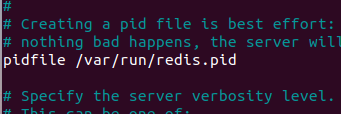
1. 修改配置文件的内容



* 1. 是否以后台daemon方式运行，默认不是后台运行



* 1. redis的PID文件路径

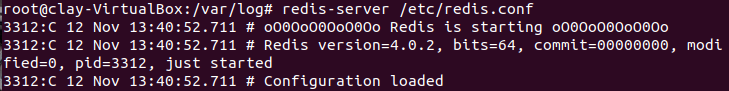


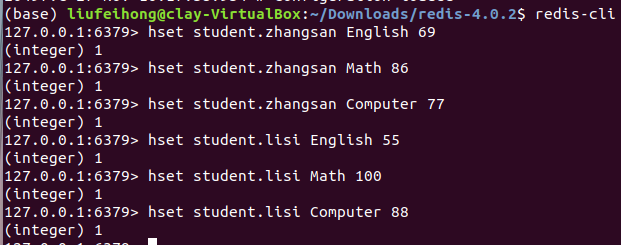
* 1. 定义log文件位置，模式log信息定向到stdout，输出到/dev/null



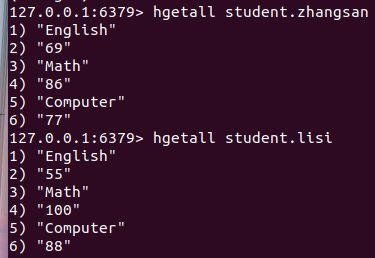
* 1. 本地数据库存放路径，默认为./，编译安装默认存在在/usr/local/bin下



1. 启动Redis:
   1. 启动Redis服务器: 
   2. 连接Redis服务器:  
      
2. Shell端编程
   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）；



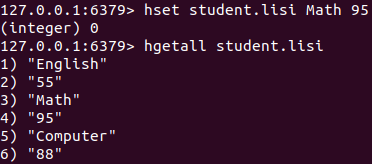
* 1. 用 hgetall 命令分别输出 zhangsan 和 lisi 的成绩信息；



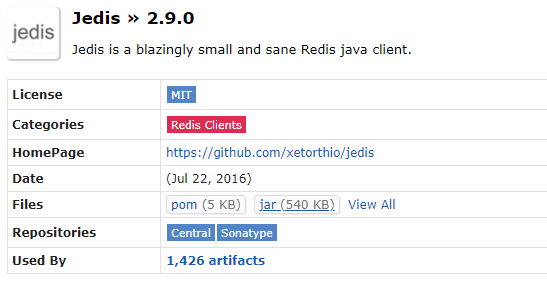
* 1. 用 hget 命令查询 zhangsan 的 Computer 成绩；

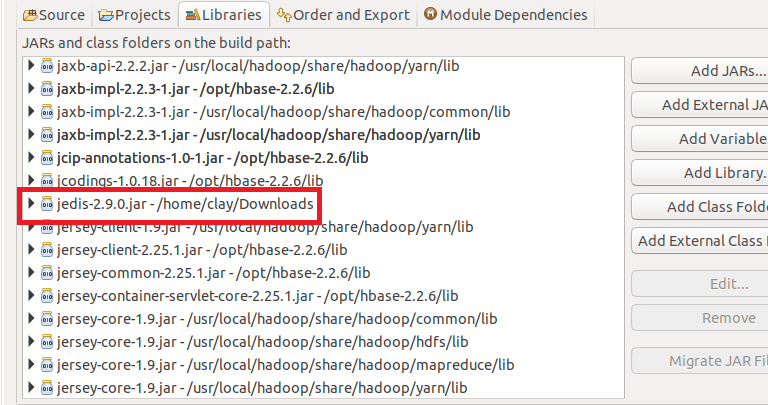


* 1. 修改 lisi 的 Math 成绩，改为 95。



1. Java端如下:
   1. 下载jar包并在工程中导入jar包





添加数据：English:45 Math:89 Computer:100

该数据对应的键值对形式如下：

scofield:｛

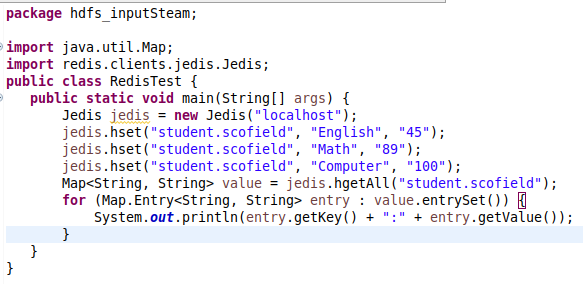
English: 45

Math: 89

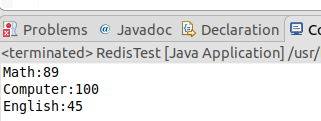
Computer: 100

｝

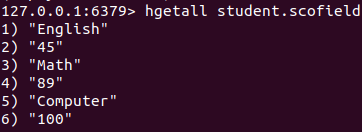
1. 书写的代码如下：



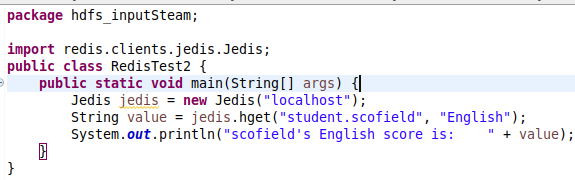
1. Eclipse运行的结果如下:



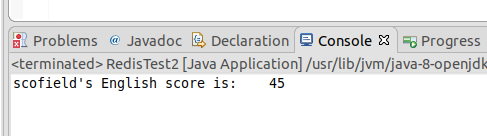
1. 验证如下:



1. 获取 scofield 的 English 成绩信息’
   * 书写的代码如下:



* + - 1. 查询结果如下,正确！



* **MongoDB数据库操作**

1. 通过WGET下载Linux版本的MongoDB

|  |
| --- |
| wget https://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-rhel62-3.4.3.tgz |

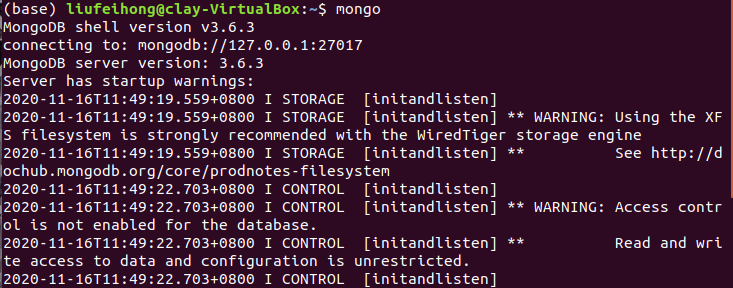
1. 将解压后的文件移动到最终的安装目录

|  |
| --- |
| mv mongodb-linux-x86\_64-rhel62-3.4.3/ /usr/local/mongodb |

1. 在data文件夹下创建mongodb.conf配置文件,并在里面输入以下内容

|  |
| --- |
| #端口号port = 27017  #数据目录  dbpath = /usr/local/mongodb/data/db  #日志目录  logpath = /usr/local/mongodb/data/logs/mongodb.log  #设置后台运行  fork = true  #日志输出方式  logappend = true  #开启认证  #auth = true |

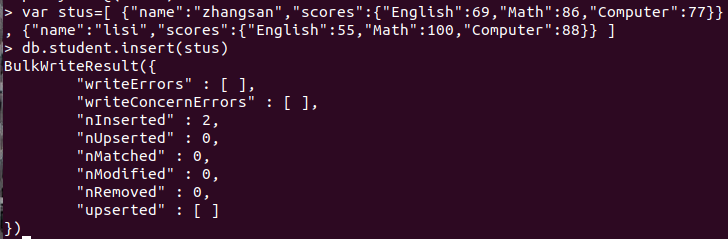
1. 启动mongo服务器



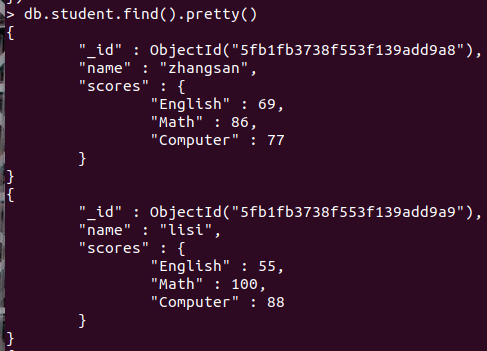
根据上面给出的文档，完成如下操作：

* 1. 用 MongoDB Shell 设计出 student 集合；





* 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)；



* 1. 修改 lisi 的 Math 成绩，改为 95。



并加以验证如下



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,

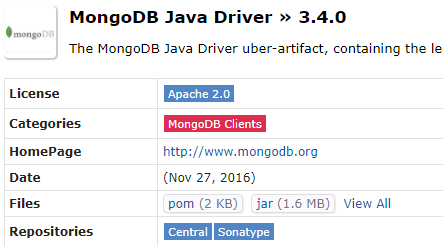
“Math”: 89,

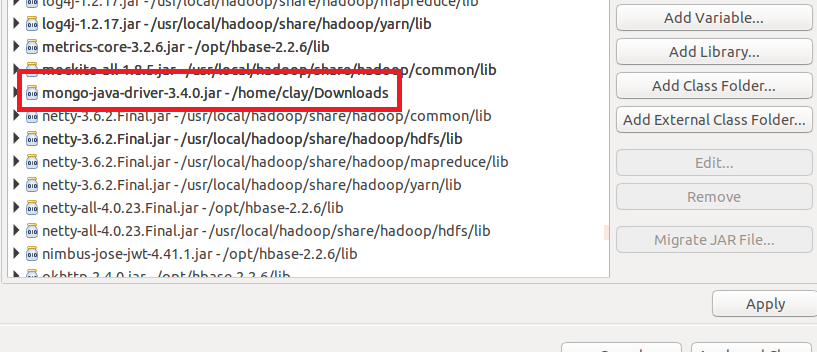
“Computer”: 100

}

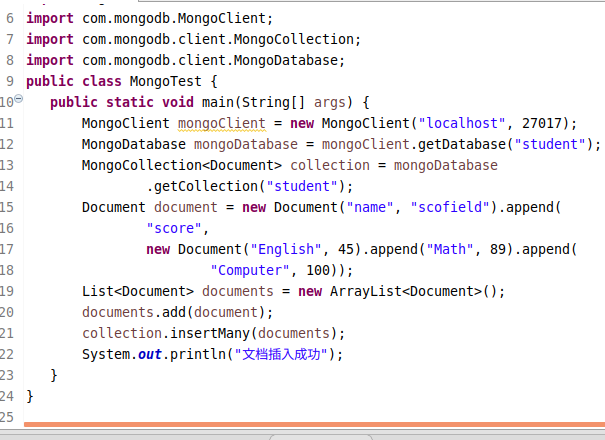
}

* + 1. 下载jar包并在工程中导入jar包

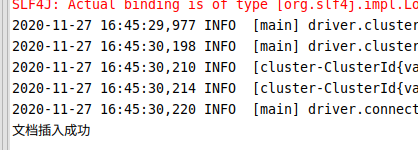




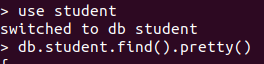
* + 1. 书写的代码如下:

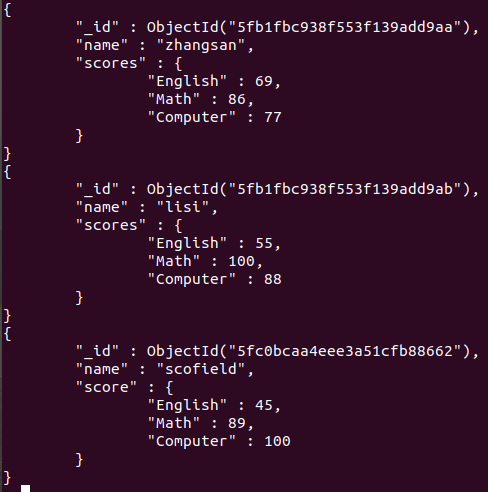


* + 1. 输出的结果如下: //**显示文档插入成功**

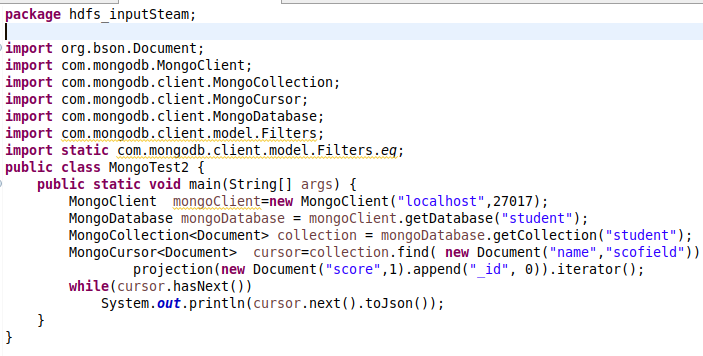


* + 1. 验证的结果如下：





* 1. 获取 scofield 的所有成绩成绩信息(只显示 score 列)
     1. 书写的代码如下:



* + 1. 运行的结果如下,成功！

