实验 4 NoSQL 和关系数据库的操作比较

19084129-李奕澄

实验目的

1) 理解 4 种数据库的概念及不同点
2) 熟练使用 4 种数据库操作常用的 Shell 命令
3) 熟悉 4 种数据库操作常用的 Java API

实验环境

Ubuntu 18.04
Hadoop 3.1.3
MySQL 版本：5.6
HBase 版本：2.2.2
Redis 版本：5.0.7
MongoDB 版本：4.2.2
JDK 版本：1.8
Java IDE：Eclipase

实验内容

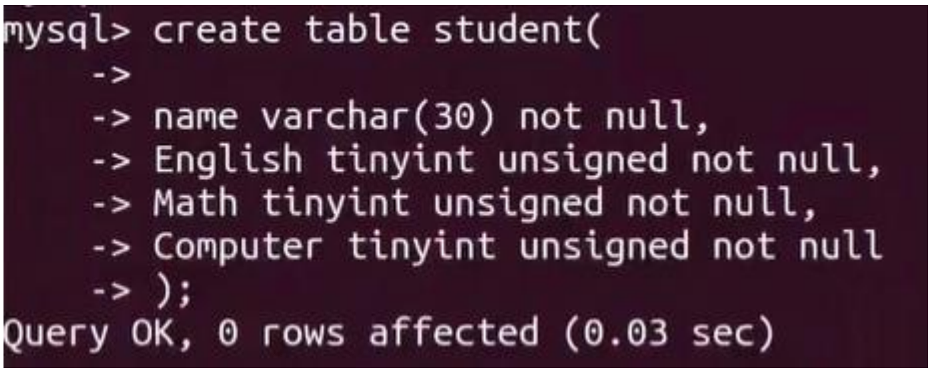
1、MySQL 数据库操作

学生（Student）表

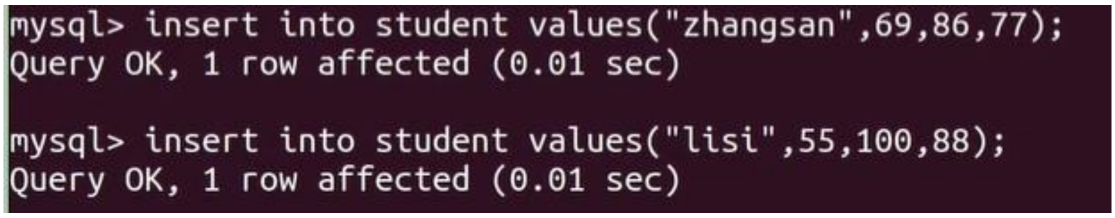
Name	English	Math	Computer
Zhangsan	69	86	77
Lisi	55	100	88

（1） 根据上面给出的 Student 表，在 MySQL 中完成如下操作：

1） 在 MySQL 中创建 Student 表，并录入数据。

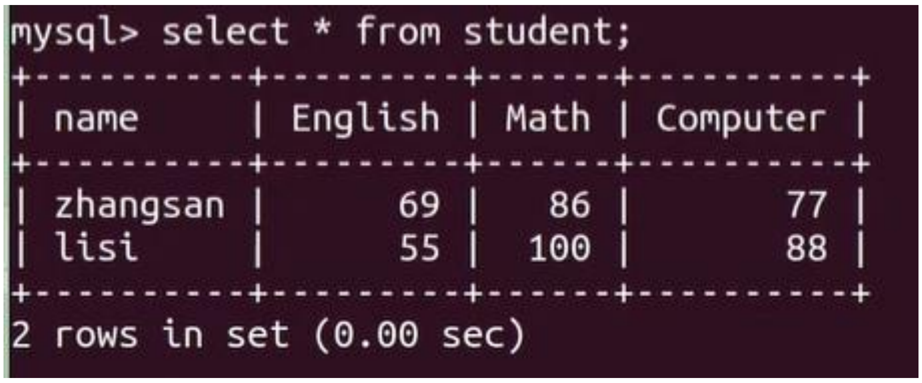


```
mysql> create table student(
    ->
    -> name varchar(30) not null,
    -> English tinyint unsigned not null,
    -> Math tinyint unsigned not null,
    -> Computer tinyint unsigned not null
    -> );
Query OK, 0 rows affected (0.03 sec)
```



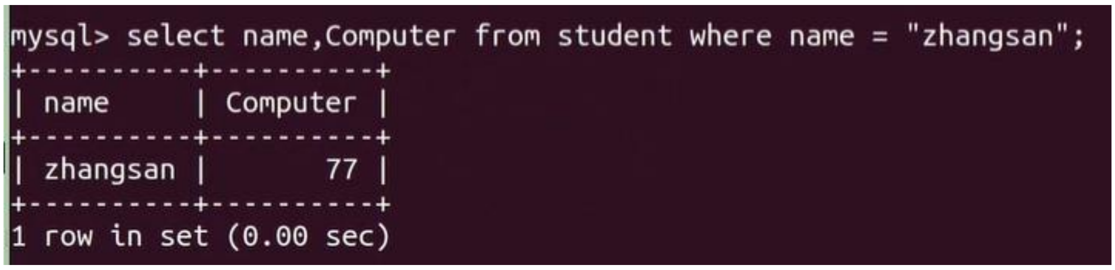
```
mysql> insert into student values("zhangsan",69,86,77);
Query OK, 1 row affected (0.01 sec)

mysql> insert into student values("lisi",55,100,88);
Query OK, 1 row affected (0.01 sec)
```

2） 用 SQL 语句输出 Student 表中的所有记录。



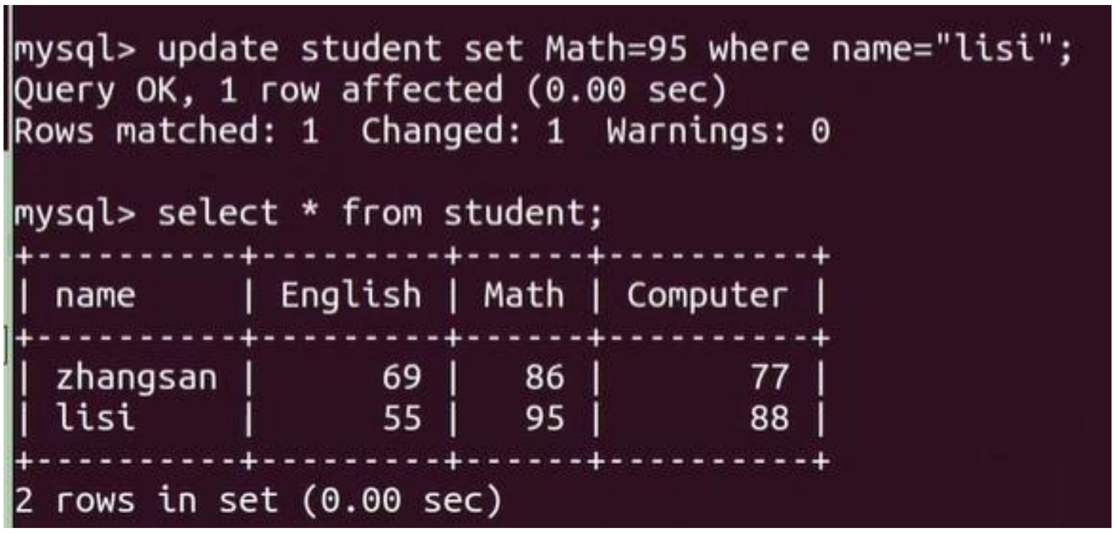
```
mysql> select * from student;
+----------+---------+------+----------+
| name     | English | Math | Computer |
+----------+---------+------+----------+
| zhangsan |      69 |   86 |       77 |
| lisi     |      55 |  100 |       88 |
+----------+---------+------+----------+
2 rows in set (0.00 sec)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

3） 查询 zhangsan 的 Computer 成绩。



```
mysql> select name,Computer from student where name = "zhangsan";
+----------+----------+
| name     | Computer |
+----------+----------+
| zhangsan |       77 |
+----------+----------+
1 row in set (0.00 sec)
```

4） 修改 lisi 的 Math 成绩为 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 * from student;
+----------+---------+------+----------+
| name     | English | Math | Computer |
+----------+---------+------+----------+
| zhangsan |      69 |   86 |       77 |
| lisi     |      55 |   95 |       88 |
+----------+---------+------+----------+
2 rows 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

2） 获取 scofield 的 English 成绩信息。

代码：

```
import java.sql.*;
```

```
public class mysqlTest {

	/**
	 * @param args
	 */
	//JDBC DRIVER and DB
	static final String  DRIVER="com.mysql.jdbc.Driver";
	static final String DB="jdbc:mysql://localhost/test";
	//Database auth
	static final String USER="root";
	static final String PASSWD="123456";

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Connection conn=null;
		Statement stmt=null;
		ResultSet rs=null;
		try {
			//加载驱动程序
			Class.forName(DRIVER);
			System.out.println("Connecting to a selected database...");
			//打开一个连接
			conn=DriverManager.getConnection(DB, USER, PASSWD);
			执行一个插入
			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!");
			//执行一个查询
//			stmt=conn.createStatement();
			sql="select name,English from student where name='scofield' ";
			获得结果集
			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) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}catch (SQLException e) {
			// TODO Auto-generated catch block
```

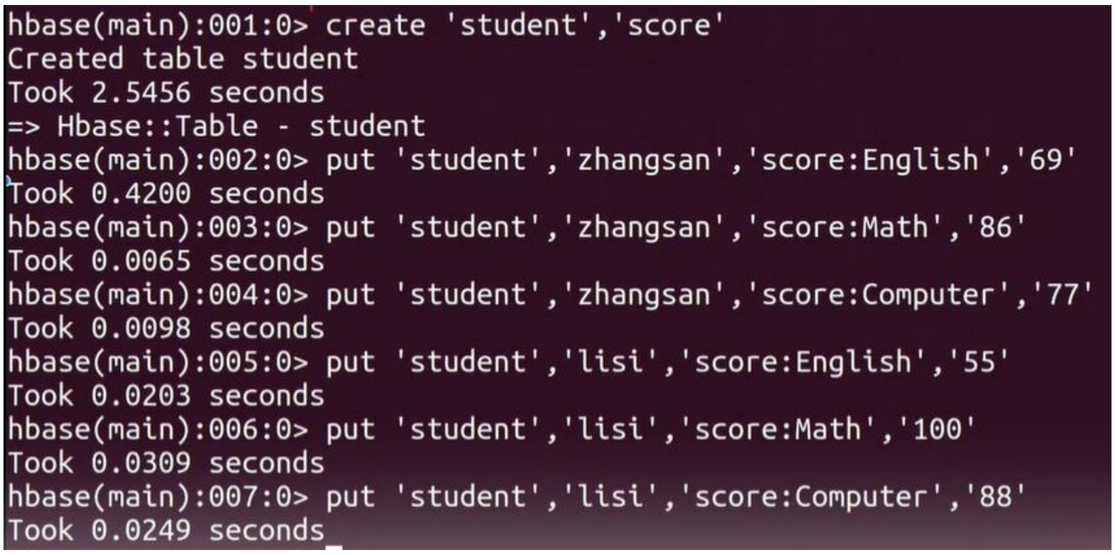
```
                e.printStackTrace();
            }finally
            {
                if(stmt!=null)
                    try {
                        stmt.close();
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                if(conn!=null)
                    try {
                        conn.close();
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
            }
        }
```

2、HBase 数据库操作

Name	Score		
	English	Math	Computer
Zhangsan	69	86	77
Lisi	55	100	88

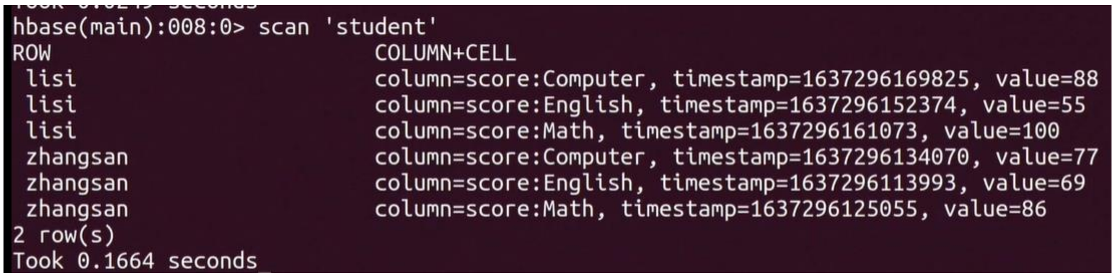
（1） 根据上面给出的 Student 表的信息，执行如下操作：

1） 用 HBase Shell 命令创建学生（Student）表。



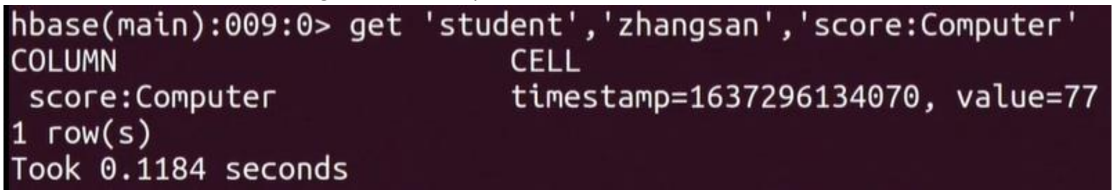
```
hbase(main):001:0> create 'student','score'
Created table student
Took 2.5456 seconds
=> Hbase::Table - student
hbase(main):002:0> put 'student','zhangsan','score:English','69'
Took 0.4200 seconds
hbase(main):003:0> put 'student','zhangsan','score:Math','86'
Took 0.0065 seconds
hbase(main):004:0> put 'student','zhangsan','score:Computer','77'
Took 0.0098 seconds
hbase(main):005:0> put 'student','lisi','score:English','55'
Took 0.0203 seconds
hbase(main):006:0> put 'student','lisi','score:Math','100'
Took 0.0309 seconds
hbase(main):007:0> put 'student','lisi','score:Computer','88'
Took 0.0249 seconds
```

2） 用 scan 命令浏览 Student 表的相关信息。



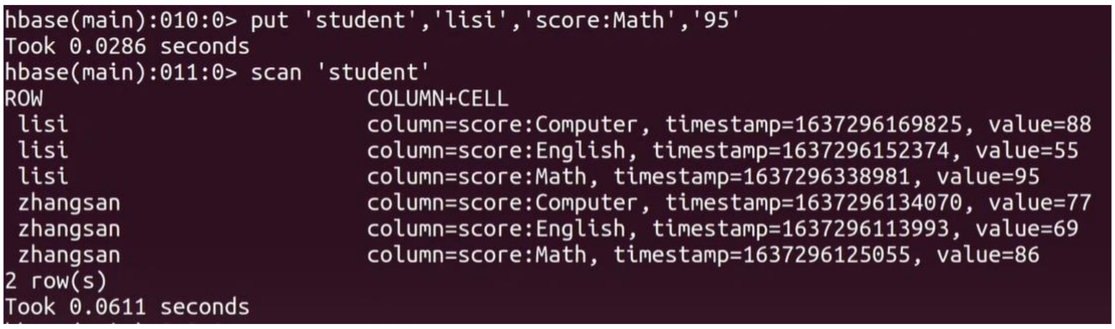
```
hbase(main):008:0> scan 'student'
ROW                           COLUMN+CELL
 lisi                         column=score:Computer, timestamp=1637296169825, value=88
 lisi                         column=score:English, timestamp=1637296152374, value=55
 lisi                         column=score:Math, timestamp=1637296161073, value=100
 zhangsan                     column=score:Computer, timestamp=1637296134070, value=77
 zhangsan                     column=score:English, timestamp=1637296113993, value=69
 zhangsan                     column=score:Math, timestamp=1637296125055, value=86
2 row(s)
Took 0.1664 seconds
```

3） 查询 zhangsan 的 Computer 成绩。



```
hbase(main):009:0> get 'student','zhangsan','score:Computer'
COLUMN                        CELL
 score:Computer               timestamp=1637296134070, value=77
1 row(s)
Took 0.1184 seconds
```

4） 修改 lisi 的 Math 成绩为 95.



```
hbase(main):010:0> put 'student','lisi','score:Math','95'
Took 0.0286 seconds
hbase(main):011:0> scan 'student'
ROW                           COLUMN+CELL
 lisi                         column=score:Computer, timestamp=1637296169825, value=88
 lisi                         column=score:English, timestamp=1637296152374, value=55
 lisi                         column=score:Math, timestamp=1637296338981, value=95
 zhangsan                     column=score:Computer, timestamp=1637296134070, value=77
 zhangsan                     column=score:English, timestamp=1637296113993, value=69
 zhangsan                     column=score:Math, timestamp=1637296125055, value=86
2 row(s)
Took 0.0611 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） 向 Student 表中添加如下所示的一条记录：

Scofield	45	89	100

2）获取 scofield 的 English 成绩信息。

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;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Get;
```

```
public class hbaseTest {

	/**
	 * @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
		init();
		  try {
			  //插入数据
//			insertRow("student","scofield","score","English","45");
//			insertRow("student","scofield","score","Math","89");
//			insertRow("student","scofield","score","Computer","100");
			//查询数据
			getData("student","scofield","score","English");
		} 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 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){
```

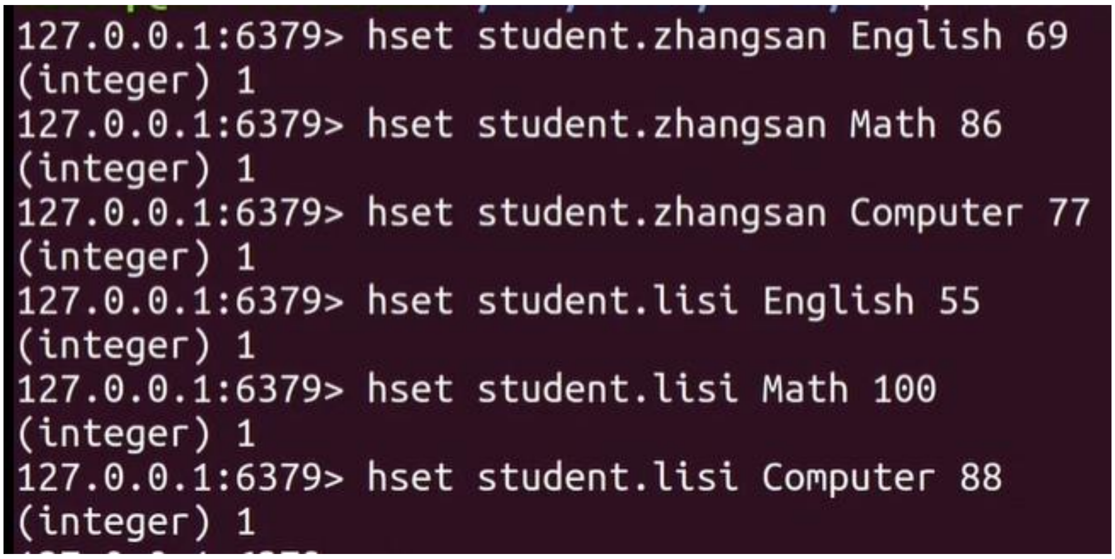
```
                System.out.println("RowName:"+new String(CellUtil.cloneRow(cell))+" ");
                System.out.println("Timetamp:"+cell.getTimestamp()+" ");
                System.out.println("column Family:"+new String(CellUtil.cloneFamily(cell))+" ");
                System.out.println("row Name:"+new String(CellUtil.cloneQualifier(cell))+" ");
                System.out.println("value:"+new String(CellUtil.cloneValue(cell))+" ");
            }
        }

    public static void 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();
            }
      }
}
```
（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
}
```

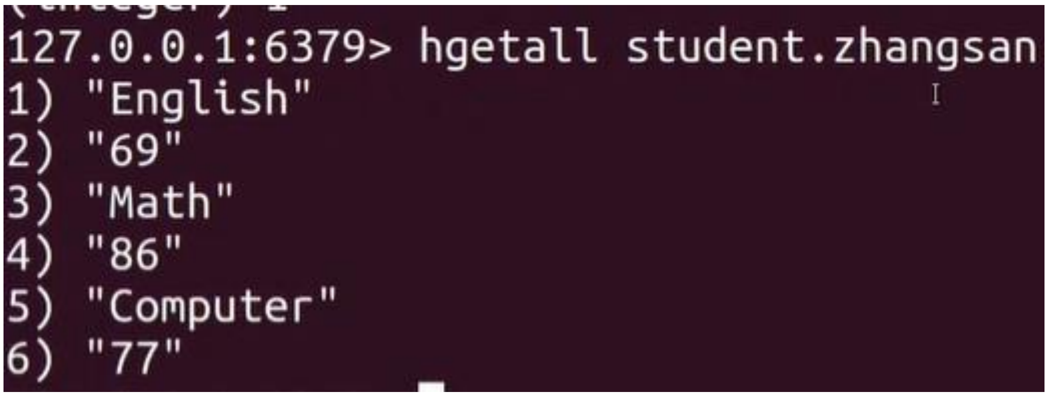
1、根据上面给出的键值对，完成如下操作：

（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）；

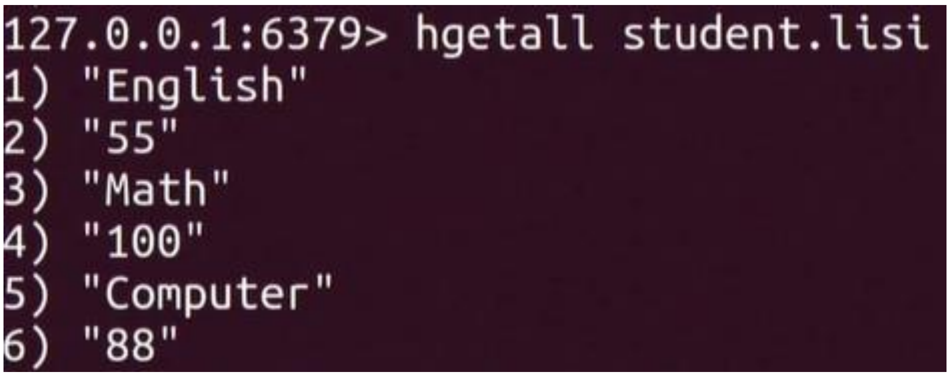


```
127.0.0.1:6379> hset student.zhangsan English 69
(integer) 1
127.0.0.1:6379> hset student.zhangsan Math 86
(integer) 1
127.0.0.1:6379> hset student.zhangsan Computer 77
(integer) 1
127.0.0.1:6379> hset student.lisi English 55
(integer) 1
127.0.0.1:6379> hset student.lisi Math 100
(integer) 1
127.0.0.1:6379> hset student.lisi Computer 88
(integer) 1
```

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

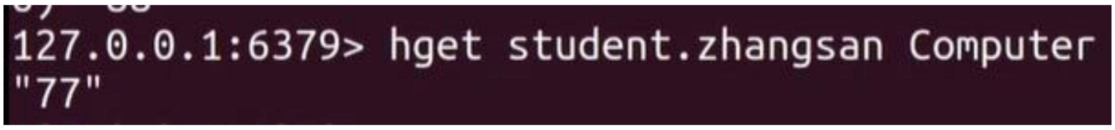


```
127.0.0.1:6379> hgetall student.zhangsan
1) "English"
2) "69"
3) "Math"
4) "86"
5) "Computer"
6) "77"
```



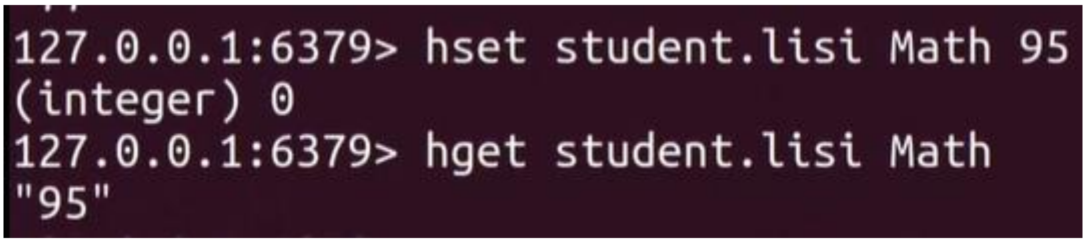
```
127.0.0.1:6379> hgetall student.lisi
1) "English"
2) "55"
3) "Math"
4) "100"
5) "Computer"
6) "88"
```

（3）用 hget 命令查询 zhangsan 的 Computer 成绩；



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

(4）修改 lisi 的 Math 成绩，改为 95。



```
127.0.0.1:6379> hset student.lisi Math 95
(integer) 0
127.0.0.1:6379> hget student.lisi 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
        }
```

（2）获取 scofield 的 English 成绩信息

代码

```
import java.util.Map;
import redis.clients.jedis.Jedis;

public class redisTest {

	/**
	 * @param args
	 */
	public static Jedis jedis;

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		jedis = new Jedis("localhost");
		//插入数据
//		test1();
		//查询数据
		test2();
	}

	public static void test1() {
		// TODO Auto-generated method stub
		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());
		}
	}

	public static void test2() {
		// TODO Auto-generated method stub
		String value=jedis.hget("student.scofield", "English");
		System.out.println("scofield's English score is:     "+value);
	}
}
```

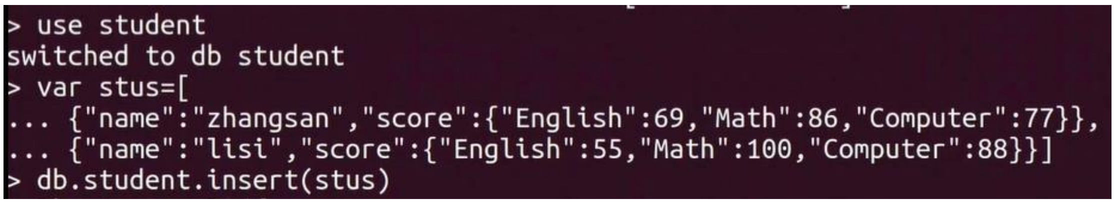
（四）MongoDB 数据库操作

Student 文档如下:

```
{
"name": "zhangsan",
"score": {
		"English": 69,
		"Math": 86,
		"Computer": 77
	}
}
{
"name": "lisi",
"score": {
		"English": 55,
		"Math": 100,
		"Computer": 88
	}
}
```

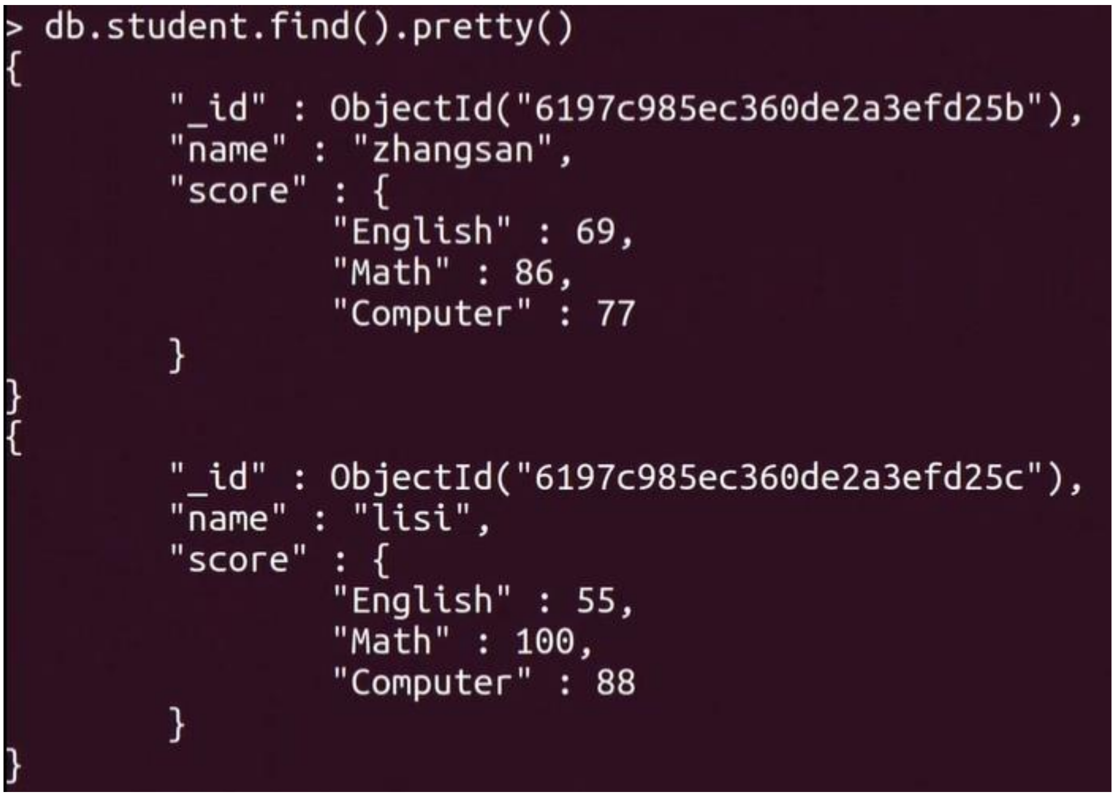
1、 根据上面给出的文档，完成如下操作:

（1） 用 MongoDB Shell 设计出 student 集合;



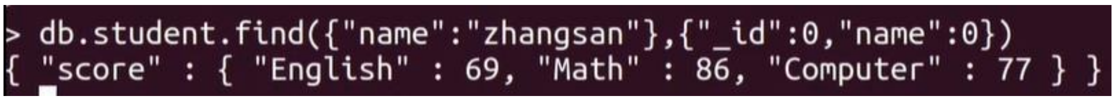
```
> use student
switched to db student
> var stus=[
... {"name":"zhangsan","score":{"English":69,"Math":86,"Computer":77}},
... {"name":"lisi","score":{"English":55,"Math":100,"Computer":88}}]
> db.student.insert(stus)
```

（2）　用 find()方法输出两个学生的信息；



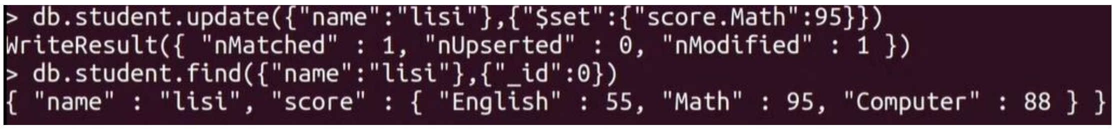
```
> db.student.find().pretty()
{
        "_id" : ObjectId("6197c985ec360de2a3efd25b"),
        "name" : "zhangsan",
        "score" : {
                "English" : 69,
                "Math" : 86,
                "Computer" : 77
        }
}
{
        "_id" : ObjectId("6197c985ec360de2a3efd25c"),
        "name" : "lisi",
        "score" : {
                "English" : 55,
                "Math" : 100,
                "Computer" : 88
        }
}
```

（3）　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})
{ "score" : { "English" : 69, "Math" : 86, "Computer" : 77 } }
```

（4）　修改 lisi 的 Math 成绩，改为 95。



```
> db.student.update({"name":"lisi"},{"$set":{"score.Math":95}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find({"name":"lisi"},{"_id":0})
{ "name" : "lisi", "score" : { "English" : 55, "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
        }
}
```

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

```
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;
import com.mongodb.client.MongoCursor;

public class mongoTest {

	/**
	 * @param args
	 */
	public static MongoClient   mongoClient;
	public static MongoDatabase mongoDatabase;
	public static MongoCollection<Document> collection;

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		init();
		//插入数据
//		test1();
		//查询数据
		test2();

	}

	public static void test1() {
		// TODO Auto-generated method stub
		//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");
	}

	public static void test2() {
		// TODO Auto-generated method stub
		//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());
                }

        public static void init() {
                // TODO Auto-generated method stub
                //实例化一个 mongo 客户端
                  mongoClient=new MongoClient("localhost",27017);
                //实例化一个 mongo 数据库
                mongoDatabase = mongoClient.getDatabase("student");
                //获取数据库中某个集合
                collection = mongoDatabase.getCollection("student");
        }
}
```