###### 试述Hadoop生态系统以及每个部分的具体功能。（10分）

1. HDFS,分布式文件系统，提供大数据文件存储功能
2. MapReduce，提供分布式并行编程模型支持
3. YARN，集群资源调度
4. HBase，分布式数据库
5. Hive，数据集管理工具
6. Flume，日志采集系统
7. Sqoop，hadoop与关系数据库交换组件
8. Mahout，机器学习领域的算法实现
9. Zookeeper，提供分布式系统的基础服务
10. Ambari，hadoop集群的web工具

###### 试述HDFS中名称节点运行的基本原理以及只设置唯一的名称节点的局限性表现在哪些方面。（10分）

1. 命名空间的局限性：名称节点是保存在内存中的，因此名称节点能够容纳对象（文件、块）的个数会受到内存空间大小的限制。
2. 性能的瓶颈：整个分布式文件系统的吞吐量受限于单个名称节点的吞吐量。
3. 隔离问题：由于集群中只有一个名称节点，只有一个命名空间，因此无法对不同应用程序进行隔离。
4. 集群的可用性：一旦这个唯一的名称节点发生故障，会导致整个集群变得不可用。

###### 试述HBase各功能组件及其作用。（10分）

1. Client，HBASE功能的使用者
2. Zookeeper，提供集群管理功能，管理每个region服务器
3. Master，管理用户丢表的操作，管理region服务器，实现负载均衡
4. Region，响应客户读写请求

###### （1）试述Hlog的工作原理。（5分）

HBase系统为每个Region服务器配置了一个HLog文件，它是一种预写式日志（Write Ahead Log），用户更新数据必须首先写入日志后，才能写入MemStore缓存，并且，直到MemStore缓存内容对应的日志已经写入磁盘，该缓存内容才能被刷写到磁盘。

（2）在HBase中，每个Region服务器维护一个Hlog，而不是为每个Region都单独维护一个Hlog。请说明这种做法的优点和缺点。（5分）

优点： 多个Region对象的更新操作所发生的日志修改，只需要不断把日志记录追加到单个日志文件中，不需要同时打开、写入到多个日志文件中。

缺点：如果一个Region服务器发生故障，为了恢复其上次的Region对象，需要将Region服务器上的对象，需要将Region服务器上的HLog按照其所属的Region对象进行拆分，然后分发到其他Region服务器上执行恢复操作。

###### 试述CAP理论的具体含义并举例说明不同产品在设计时是如何运用CAP理论的。（10分）

一致性（Consistency）可用性（Availability）分区容错性（Partition tolerance）

(1) CA: 优先保证一致性和可用性，放弃分区容错。 这也意味着放弃系统的扩展性，系统不再是分布式的，有违设计的初衷。

(2) CP: 优先保证一致性和分区容错性，放弃可用性。在数据一致性要求比较高的场合(譬如:zookeeper,Hbase) 是比较常见的做法，一旦发生网络故障或者消息丢失，就会牺牲用户体验，等恢复之后用户才逐渐能访问。

(3) AP: 优先保证可用性和分区容错性，放弃一致性。NoSQL中的Cassandra 就是这种架构。跟CP一样，放弃一致性不是说一致性就不保证了，而是逐渐的变得一致。

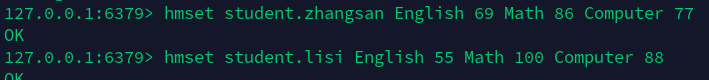
###### 六、Redis数据库操作。（10分）

Student键值对如下：

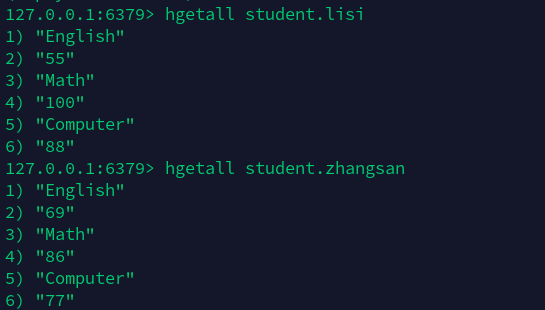
|  |
| --- |
| zhangsan:｛  English: 69  Math: 86  Computer: 77  ｝  lisi:｛  English: 55  Math: 100  Computer: 88  ｝ |

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）；



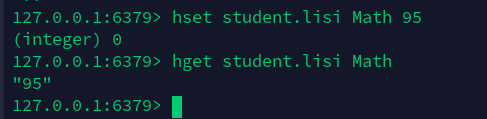
1. 用hgetall命令分别输出zhangsan和lisi的成绩信息；



1. 用hget命令查询zhangsan的Computer成绩；



1. 修改lisi的Math成绩，改为95。



七、MongoDB数据库操作。

Student文档如下:

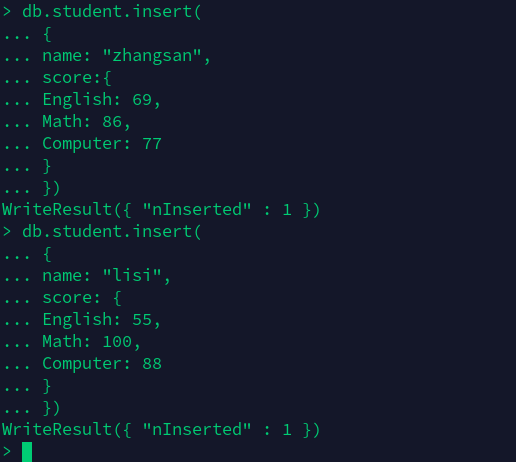
|  |
| --- |
| {  “name”: “zhangsan”,  “score”: {  “English”: 69,  “Math”: 86,  “Computer”: 77  }  }  {  “name”: “lisi”,  “score”: {  “English”: 55,  “Math”: 100,  “Computer”: 88  }  } |

1.根据上面给出的文档，完成如下操作：（20分）

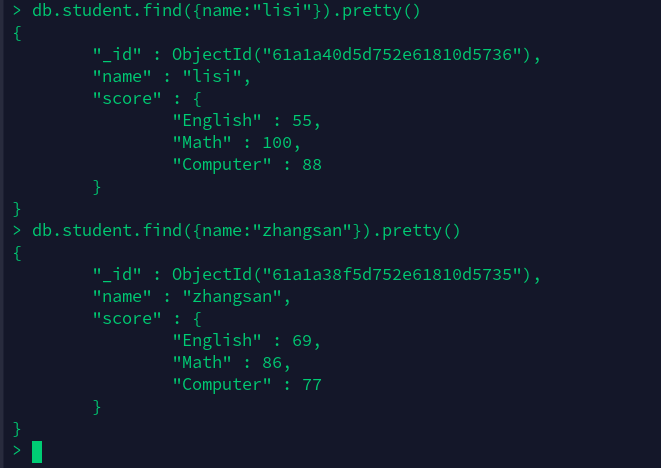
（1）用MongoDB Shell设计出student集合；

use student



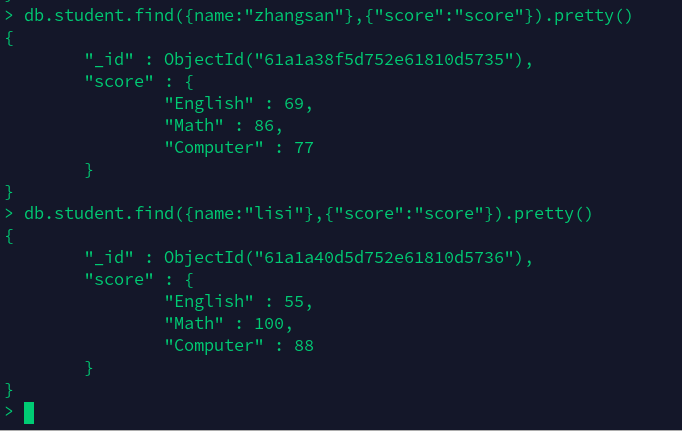


1. 用find()方法输出两个学生的信息；



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

db.student.find({name:”zhangsan”},{“score”:1});



1. 修改lisi的Math成绩，改为95。

mongodb有两种更新操作，分别是$set和$unset，set表示添加或修改，unset表示删除

db.student.update({name:”lisi”},{$set:{math:95}});

db.student.update({name=”lisi”,{$unset:{math:””}}});



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为45 ，Math为89，Computer为100。（10分）

与上述数据对应的文档形式如下：

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

static String uri = "mongodb://java\_client:8218630@localhost:27017/student";

public static void main(String[] args) {

try (MongoClient mongoClient = MongoClients.create(uri)) {

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

MongoCollection<Document> collection = database.getCollection("student");

// Document doc = collection.find(eq("name", "scofield")).first();

// System.out.println(doc.toJson());

Document score = new Document();

score.put("English",45);

score.put("Math",89);

score.put("Computer",89);

Document doc = new Document();

doc.put("name","scofield");

doc.put("score",score);

collection.insertOne(doc);

}

}

（2）获取scofield的所有成绩信息(只显示score列)。（10分）

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

MongoCollection<Document> collection = database.getCollection("student");

Document doc = collection.find(eq("name", "scofield"),).first();

System.out.println(doc.toJson());