**实 验 报 告 册**

2023学年 春季 学期

专业班级： 计科6班

学 号： 2020204331

姓 名： 王博文

指导教师： 纪俊

青岛大学计算机科学技术学院

|  |  |  |  |
| --- | --- | --- | --- |
| 实验项目 | HBase安装部署、使用与编程 | 日 期 期 | 2023.5.16 |
| 实验环境 | LINUX | 成绩评定 |  |
| 实验目的和要求：  **搭建HBase，在HBase的Shell中创建数据库、增加数据记录、查询记录、删除记录、删除数据库，之后通过Java编程实现上述操作**  **要求：每位同学独立完成，详细记录实验过程所遇到的问题与解决的问题，以及最终实验结果** | | | |
| 实验内容或结果：  **第一步建表**  **描述这个表**  **添加数据**  **删除数据**    **查询数据**  **Scan命令**    **删除表**    **查询表格的历史数据：**  **此步骤需要先创建一个表，然后对数据先进行插入然后再多次进行修改**    **上面框选的就是91001的历史username**  **用hbase查看当前的表格**    Teacher是上面用来查看历史数据所创建的表格  Student则是刚刚代码生成的表格    **采用java代码进行操作**  **代码截取于https://dblab.xmu.edu.cn/blog/2442/**  **源代码并没有删除和打印表格，需要自己编写**  **import org.apache.hadoop.conf.Configuration;**  **import org.apache.hadoop.hbase.\*;**  **import org.apache.hadoop.hbase.client.\*;**  **import org.apache.hadoop.hbase.util.Bytes;**    **import java.io.IOException;**  **public class ExampleForHBase {**  **public static Configuration configuration;**  **public static Connection connection;**  **public static Admin admin;**  **public static void main(String[] args)throws IOException{**  **init();**  **// 创建表**  **createTable("student",new String[]{"score"});**  **// 添加数据**  **insertData("student","syj","score","Math","120");**  **insertData("student","syj","score","CS","120");**  **insertData("student","scs","score","Math","125");**  **insertData("student","scs","score","OS","81");**  **// 打印表**  **printTable("student");**  **// 查看数据**  **getData("student", "syj", "score","Math");**  **getData("student", "scs", "score","Math");**  **// 删除数据**  **deleteData("student","scs", "score","Math","OS");**    **printTable("student");**  **// 删除表**  **deleteTable("student");**  **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 deleteTable(String tableName) throws IOException {**  **init();**  **TableName tn = TableName.valueOf(tableName);**  **if (admin.tableExists(tn)) {**  **admin.disableTable(tn);**  **admin.deleteTable(tn);**  **}**  **System.out.println("删除成功");**    **close();**  **}**  **public static void createTable(String myTableName,String[] colFamily) throws IOException {**  **TableName tableName = TableName.valueOf(myTableName);**  **if(admin.tableExists(tableName)){**  **System.out.println("talbe is exists!");**  **}else {**  **TableDescriptorBuilder tableDescriptor = TableDescriptorBuilder.newBuilder(tableName);**  **for(String str:colFamily){**  **ColumnFamilyDescriptor family =**  **ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes(str)).build();**  **tableDescriptor.setColumnFamily(family);**  **}**  **admin.createTable(tableDescriptor.build());**  **}**  **System.out.println("创建成功");**    **}**    **public static void insertData(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();**  **System.out.println("添加成功");**  **}**  **public static void deleteData(String tableName,String rowKey,String colFamily, String col,String col2)throws IOException{**  **Table table = connection.getTable(TableName.valueOf(tableName));**  **Delete delete =new Delete(rowKey.getBytes());**  **table.delete(delete);**  **table.close();**  **System.out.println("删除成功");**  **}**  **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);**  **System.out.println("查找成功：");**  **System.out.println(new String(result.getValue(colFamily.getBytes(),col==null?null:col.getBytes())));**  **table.close();**  **}**  **public static void printTable(String tableName) throws IOException {**  **Table table = connection.getTable(TableName.valueOf(tableName));**  **System.out.println("表格：");**  **Scan scan = new Scan();**  **ResultScanner scanner = table.getScanner(scan);**  **for (Result result : scanner) {**  **byte[] rowKey = result.getRow();**  **System.out.print("row key: " + Bytes.toString(rowKey) + "\t");**  **Cell[] cells = result.rawCells();**  **for (Cell cell : cells) {**  **byte[] family = CellUtil.cloneFamily(cell);**  **byte[] qualifier = CellUtil.cloneQualifier(cell);**  **byte[] value = CellUtil.cloneValue(cell);**  **System.out.print(Bytes.toString(family) + ":" + Bytes.toString(qualifier) + "=" + Bytes.toString(value) + "\t");**  **}**  **System.out.println();**  **}**  **scanner.close();**  **table.close();**  **}}**  **运行结果：**  IMG_256  IMG_256 | | | |
| 思考或体会：  **实验所产生的问题：**  **这次的实验问题实在是太多了。**  **由于我们小组的成员并不是一个宿舍的，有别的成员住在学校的另一边，所以我们一般都在上课的时候进行试验，因此进度上会有些缓慢。**   1. **单机Hadoop的安装**   **在最开始的时候，我们组员并没有意识到版本的问题会有多大的影响，我最初的hadoop是按照网上的教程进行安装，其中在初始化也就也是hdfs namenode -format时，发现进程缺少，少过Datanode也少过Namenode等等，我开始上网搜索一个一个解决，一般的原因都是因为初始化次数过多，hadoop找不到你新建的东西，此时就需要把tmp文件和log文件删除然后再次重新初始化。**   1. **Hadoop集群的安装** 2. **我们组员一共有四个人，我是Master，在没上这门课的时候我的电脑就是双系统，有自己的用户名字（Tony-Wang），其余的组员都是按照林子雨也就是下面这个链接的教程安装的**[Hadoop3.1.3安装教程\_单机/伪分布式配置\_Hadoop3.1.3/Ubuntu18.04(16.04)\_厦大数据库实验室博客 (xmu.edu.cn)](https://dblab.xmu.edu.cn/blog/2441/)。**他们的用户名都是hadoop，在我们连接好后进行ssh无密码登录的时候返现了问题，教程一样且ip也对，却无法连接，经过长时间的排查返现是用户名的问题。假使我的名字是hadoop那么就可以直接ssh Slave1链接，但我有我自己的名字，在连接的时候就需要写ssh hadoop@Slave1才可成功链接。** 3. **在配置单机的时候由于我自己下的java和他们下的java版本不一样，导致了也无法运行Hadoop，所以我把他们的java的路径及名称都改成了我的路径和名称，正因如此才可进行圣经的wordcount。** 4. **Hbase的搭建**   **这个可谓是最让我感到难过的一次搭建，我们小组本来的速度很快，但就是在这个实验上拌了很大的一个跟头。**   1. **还是那个历史遗留问题，我是自己下载的hadoop，我的版本是3.3.1，而他们的版本是3.1.3。由于hbase对hadoop有很强的依赖性，但起初我们并没有意识到，下载的hbase2.2.2并不适配我的3.3.1（网上找到的hbase2.3.4虽然适配我的3.3.1但没有详细的教程，所以按照2.2.2装的，不成功）,无奈只能重新从hadoop安装，也就是第一步做起。期间用了各种方法，由于自己的疏忽，导致在编辑配置文件的时候没有把全部的Master改成符合我主机名字的形式，在加上自己心智的不坚定，导致我原有的主机名字也叫我改成了hadoop，最终我的名字跟教程上的名字一摸一样，没有了自己的特色。** 2. **从头开始，我把我自己下的hadoop和java全部换成了跟小组成员一样的且遵循**[(26条消息) 《大数据基础教程、实验和案例教程---林子雨版》分布式模式的HBase配置\_林子雨hbase\_笔写心城的博客-CSDN博客](https://blog.csdn.net/qq_39940390/article/details/94477049)**教程来搭建，虽然流程上还是很曲折的，但按照这个大方向，我们醉成成功搭建并做起了实验。** 3. **时间来到了下一次的实验，我如往常一样打开hadoop，但是弹出了**  [Cannot execute /usr/bin/hadoop/libexec/hdfs-config.sh](https://www.cnblogs.com/carlohu/p/10972201.html)**这一串报错。**[Cannot execute /usr/bin/hadoop/libexec/hdfs-config.sh - 风笑夜 - 博客园 (cnblogs.com)](https://www.cnblogs.com/carlohu/p/10972201.html)虽然不知道问题是什么，但是按照此网站中的**unset HADOOP\_HOME指令得以继续实验，下课以后，我们手动关闭了hbase，但他会卡住，最终会我们采用了直接关机，这也为下次实验埋下了伏笔。**  1. **果然，在第二次实验的开头我们又花了好长一段时间找问题并修改问题。这是我使用的网站**[(26条消息) stop-hbase.sh关闭不了，一直处于等待状态。（已解决）\_hbase停止不了\_柿子镭的博客-CSDN博客](https://blog.csdn.net/weixin_45890771/article/details/122205099) 2. **编写java程序的时候对照着林子雨的网站自己学习，上网搜索hbase对应的java语句应该如何编写。**  总结：一个小组的成员应该按照一个教程继续做下去，且不要认为某些地方不一样不耽误后续的使用。 **（虽然我觉得这样真的有点限制住同学们，我一直认为解决问题的方法绝不只是循规蹈矩，按照教程一路走下去，但…………人的耐心也是很重要的因素，我不能拖着整个小组的速度，看着别的小组按照这个方法成功以后还慢条斯理的解决自己的问题。倘若这是我自己进行的实验，我会极力反对改我用户名，重装hadoop等一些操作，可能结果还是推倒重来，毕竟实力就在这里且网上的教程也确实不够全面，但我还是会进行尝试。）** | | | |
| 备注：  无 | | | |