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

```
hbase(main):002:0> describe 'student'
Table student is ENABLED
student
COLUMN FAMILIES DESCRIPTION
{NAME => 'Sage', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_
BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'fals
e', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLIC
ATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_ME
MORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'f
alse', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'Sdept', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION
_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'fal
se', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLI
CATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_M
EMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => '
false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'Sname', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION
_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'fal
se', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLI
CATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_M
```

添加数据

```
Took 0.2972 seconds
hbase(main):003:0> put 'student','95001','Sname','LiYing'
Took 0.3161 seconds
```

删除数据

```
Took 0.0235 seconds
hbase(main):005:0>   delete 'student','95001','Ssex'
Took 0.0882 seconds
hbase(main):006:0>   deleteall 'student','95001'
Took 0.0283 seconds
```

查询数据

```
Took 0.0262 seconds
hbase(main):012:0>   get 'student','95001'
COLUMN                CELL
 Sname:               timestamp=1685262147451, value=LiYing
 course:math          timestamp=1685262152014, value=80
1 row(s)
```

Scan 命令

```
hbase(main):013:0>   scan 'student'
ROW                   COLUMN+CELL
 95001                column=Sname:, timestamp=1685262147451, value=LiYing
 95001                column=course:math, timestamp=1685262152014, value=80
1 row(s)
```

删除表

```
Took 0.0545 seconds
hbase(main):014:0> disable 'student'

Took 4.5613 seconds
hbase(main):015:0> drop 'student'
Took 0.7957 seconds
```

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

```
Took 0.7957 seconds
hbase(main):016:0>   create 'teacher',{NAME=>'username',VERSIONS=>5}
Created table teacher
Took 2.2688 seconds
=> Hbase::Table - teacher
hbase(main):017:0> put 'teacher','91001','username','Mary'
Took 0.0444 seconds
hbase(main):018:0> put 'teacher','91001','username','Mary1'
Took 0.0307 seconds
hbase(main):019:0> put 'teacher','91001','username','Mary2'
Took 0.0183 seconds
hbase(main):020:0> put 'teacher','91001','username','Mary3'
Took 0.0179 seconds
hbase(main):021:0> put 'teacher','91001','username','Mary4'
Took 0.0146 seconds
hbase(main):022:0> put 'teacher','91001','username','Mary5'  get 'teacher','91001'
>'username',VERSIONS=>5}
SyntaxError: (hbase):22: syntax error, unexpected tIDENTIFIER
put 'teacher','91001','username','Mary5'  get 'teacher','91001',{COLUMN=>'username'
S=>5}


hbase(main):023:0>   get 'teacher','91001',{COLUMN=>'username',VERSIONS=>5}
COLUMN                CELL
 username:            timestamp=1685262670889, value=Mary4
 username:            timestamp=1685262670855, value=Mary3
 username:            timestamp=1685262670822, value=Mary2
 username:            timestamp=1685262670774, value=Mary1
 username:            timestamp=1685262670729, value=Mary
1 row(s)
Took 0.0319 seconds
hbase(main):024:0>
```

上面框选的就是 91001 的历史 username

用 hbase 查看当前的表格

```
hbase(main):001:0> list
TABLE
student
teacher
2 row(s)
Took 0.3303 seconds
=> ["student", "teacher"]
hbase(main):002:0>
```

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

```
hbase(main):003:0> scan 'student'
ROW                    COLUMN+CELL
 zhangsan              column=score:Computer, timestamp=1685264139391, value=77
 zhangsan              column=score:English, timestamp=1685264139349, value=69
 zhangsan              column=score:Math, timestamp=1685264139367, value=86
1 row(s)
Took 0.2233 seconds
```

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.get
Bytes())));
        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：

```
创建成功
添加成功
添加成功
添加成功
添加成功
表格:
row key: scs    score:Math=125  score:OS=81
row key: syj    score:CS=120    score:Math=120
查找成功:
120
查找成功:
125
删除成功
表格:
row key: syj    score:CS=120    score:Math=120
2023-05-30 10:47:14,686 INFO  [main] client.HBaseAdmi
删除成功
```

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。

一、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。

二、Hadoop 集群的安装

1. 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)。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。
2. 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。

三、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** 这一串报错。[Cannot execute /usr/bin/hadoop/libexec/hdfs-config.sh - 风笑夜 - 博客园 (cnblogs.com)](#)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。

4. 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](#)

5. 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