北京邮电大学计算机学院

2022-2023 学年第一学期实验报告

课程名称： 大数据原理与技术

项目名称： 实验二：HBase 表设计及编程

项目完成人：

姓名： 许振华 **学号：** 2020211721

指导教师： 孙鹏飞

日　　期： 2022 年 11 月 21 日

一、 实验目的

通过根据传统的 SQL 数据库设计 HBase 表，以及使用 JAVA 编程完成相关的数据增、删、查、改的过程，来加深学习的 HBase 的相关概念并学习相关的 JAVA API。

二、 实验内容

本实验内容分为两大部分。

2.1. HBase 表的设计

首先是根据给出的 SQL 数据表设计 HBase 表，结果如下所示：考虑到姓名有可能重复而学号不会，因此将学号作为行键，剩余列族和列所示如下：

行键	学生信息			数学			计算机科学			英语		
学号	姓名	性别	年龄	课程号	学分	成绩	课程号	学分	成绩	课程号	学分	成绩

2.2. 相关编程任务的完成

2.2.1. createTable(String tableName, String[] fields)

创建表，参数 tableName 为表的名称，字符串数组 fields 为存储记录各个域名称的数组。要求当 HBase 已经存在名为 tableName 的表的时候，先删除原有的表，然后再创建新的表。

2.2.2. addRecord(String tableName, String row, String[] fields, String[] values)向表 tableName、行 row（用 S_Name 表示）和字符串数组 fields 指定的单元格中添加对应的数据 values。

2.2.3. scanColumn(String tableName, String column)浏览表 tableName 某一列的数据，如果某一行记录中该列数据不存在，则返回 null。

2.2.4. modifyData(String tableName, String row, String column)修改表 tableName，行 row，列 column 指定的单元格的数据。

2.2.5. deleteRow(String tableName, String row)删除表 tableName 中 row 指定的行的记录。

三、 实验环境

Ubuntu22.04
Openjdk 1.8
Hadoop 3.4.0
Hbase 3.0.0

四、 实验结果

首先设计出相应的表，接着按照变成要求完成编程任务，最后编写测试用例，利用实验所提供的数据完成相关实验，相关测试结果如下：

4.1. createTable(String myTableNaString, String[] colFamilyStrings)

由于测试的时候是多次输入，因此开始测试的时候，已经存在同名表，将会删除并创建新表：

表的信息如下：

```
Run | Debug
public static void main(String[] args) throws IOException {
    String tableNameString = "StuInfo";
    String[] colFamilyStrings = {"SID", "SI", "Ma", "CS", "Eng"};
```

初始化：

```
Start the progress of creating a table named 'StuInfo'.

log4j:WARN No appenders could be found for logger (org.apache.hadoop
.metrics2.lib.MutableMetricsFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig
 for more info.
The table exists and will be deleted!
```

4.2. addRecord(String tableName, String row, String[] fields, String[] values)

向数据库添加数据，数据格式如下：

```
String[] infoFields = {"SI:name", "SI:sex", "SI:age", "Ma:id",
"Ma:credit", "Ma:score", "CS:id", "CS:credit", "CS:score",
"Eng:id", "Eng:credit", "Eng:score"};

String[] info_1 = {"Zhangsan", "male", "23", "123001", "2",
"86", "123002", "5", "", "123003", "3", "69"};
String[] info_2 = {"Mary", "female", "22", "123001", "2", "",
"123002", "5", "77", "123003", "3", "99"};
String[] info_3 = {"Lisi", "male", "24", "123001", "2", "98",
"123002", "5", "95", "123003", "3", ""};
```

添加数据过程：

```
Add record of stu-2015001
Add record of stu-2015002
Add record of stu-2015003
```

4.3. scanColumn(String tableName, String columnFamily)

此用例中，选取列族‘SI’和列‘score’测试：

```
System.out.println("\nScanning the data of colfamily 'SI'...
\n");
scanColumn(tableNameString, "SI");

System.out.println("\nScanning the data of col 'Ma:score'...
\n");
scanColumn(tableNameString, "Ma:score");
```

结果如下：

```
Scanning the data of colfamily 'SI'...

2015001 23      age     SI
2015001 Zhangsan        name    SI
2015001 male    sex     SI
2015002 22      age     SI
2015002 Mary    name    SI
2015002 female  sex     SI
2015003 24      age     SI
2015003 Lisi    name    SI
2015003 male    sex     SI

Scanning the data of col 'Ma:score'...

2015001 86      score   Ma
2015002 null    score   Ma
2015003 98      score   Ma
```

4.4. modifyData(String tableName, String row, String column, String value)

此次用例中，将学号为 2015002 的学生英语成绩修改为了 100，测试代码如下：

```
modifyData(tableNameString, "2015002", "Eng:score", "100");
```

结果：

```
Modifying data of the score of English(2015002)


Check...
2015001 69      score   Eng
2015002 100     score   Eng
2015003 null    score   Eng
```

4.5. deleteRow(String tableName, String row)

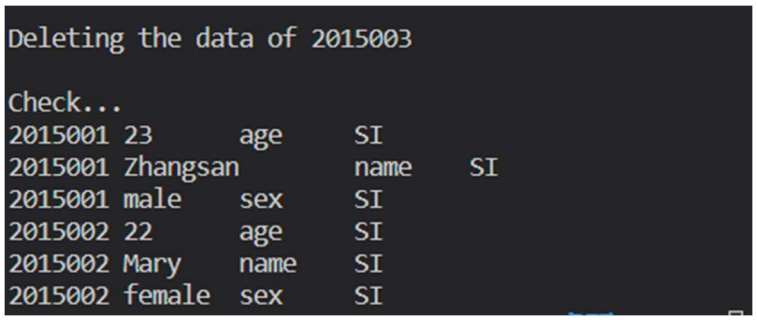
删除行，测试对象为 2015003，测试代码如下：

```
deleteRow(tableNameString, "2015003");
```

之后检查所有信息

```
scanColumn(tableNameString, "SI");
```

结果如下：



综上所述，本次实验完成了要求的全部任务。

五、 附录

（根据实际需要附上实验文档，如：问题分析、设计方案、算法、设计图、主要程序、仿真结果、运行结果、调试心得等，具体内容根据实验要求来定。源代码请附在这里。源代码排版请特别注意，用 5 号字体，行间距为单倍行距。注意节省空间，不要浪费纸张。）

实验所用代码如下：

```
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 TestHBase {

    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;

    public static void main(String[] args) throws IOException {
        String tableNameString = "StuInfo";
        String[] colFamilyStrings = {"SID", "SI", "Ma", "CS", "Eng"};

        String[] infoFields = {"SI:name", "SI:sex", "SI:age", "Ma:id",
"Ma:credit", "Ma:score", "CS:id", "CS:credit", "CS:score", "Eng:id",
"Eng:credit", "Eng:score"};

        String[] info_1 = {"Zhangsan", "male", "23", "123001", "2", "86",
"123002", "5", "", "123003", "3", "69"};
        String[] info_2 = {"Mary", "female", "22", "123001", "2", "", "123002",
"5", "77", "123003", "3", "99"};
        String[] info_3 = {"Lisi", "male", "24", "123001", "2", "98", "123002",
"5", "95", "123003", "3", ""};
```

```
        System.out.println("Start the progress of creating a table named 'StuInfo'.\n");
        createTable(tableNameString, colFamilyStrings);

        System.out.println("Add record of stu-2015001");
        addRecord(tableNameString, "2015001", infoFields, info_1);

        System.out.println("Add record of stu-2015002");
        addRecord(tableNameString, "2015002", infoFields, info_2);

        System.out.println("Add record of stu-2015003");
        addRecord(tableNameString, "2015003", infoFields, info_3);

        System.out.println("\nScanning the data of colfamily 'SI'... \n");
        scanColumn(tableNameString, "SI");

        System.out.println("\nScanning the data of col 'Ma:score'... \n");
        scanColumn(tableNameString, "Ma:score");

        System.out.println("\nModifying data of the score of English(2015002)\n");
        modifyData(tableNameString, "2015002", "Eng:score", "100");
        System.out.println("\nCheck...");
        scanColumn(tableNameString, "Eng:score");

        System.out.println("\nDeleting the data of 2015003");
        deleteRow(tableNameString, "2015003");
        System.out.println("\nCheck...");
        scanColumn(tableNameString, "SI");
    }

    // establish connection
    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();
        }
    }

    // disconnect
```

```
    public static void close() {
        try {
            if (admin != null) {
                admin.close();
            }
            if (null != connection) {
                connection.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // create a table,
    // if it exits, then delete it and create a new one.
    public static void createTable(String myTableNaString, String[]
colFamilyStrings) throws IOException {
        // operate the strings
        // HashMap<String, List<String>> hashMap = new HashMap<>();
        // for(String field : colFamilyStrings){
        //     String[] res = field.split(":");
        //     if(hashMap.get(res[0]) == null){
        //         hashMap.put(res[0], new ArrayList<String>());
        //     }else{
        //         hashMap.get(res[0]).add(res[1]);
        //     }
        // }

        init();
        TableName tableName = TableName.valueOf(myTableNaString);
        if (admin.tableExists(tableName)) {
            System.out.println("The table exists and will be deleted!");
            admin.disableTable(tableName);
            admin.deleteTable(tableName);
        }

        HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);

        // set colFamilies for descriptor
        // Set<String> keySet = hashMap.keySet();
        for (String str : colFamilyStrings) {
            HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);
            hTableDescriptor.addFamily(hColumnDescriptor);
        }
```

```
        // build table descriptor
        //TableDescriptor tableDescriptor = hTableDescriptor.build();

        // create table
        admin.createTable(hTableDescriptor);

        close();
    }

    // add data to an attribute of a row
    public static void insertData(String taleNameString, String rowkeyString,
String colFamilyString, String colString,
            String valString) throws IOException {
        init();
        Table table = connection.getTable(TableName.valueOf(taleNameString));
        Put put = new Put(Bytes.toBytes(rowkeyString));
        put.addColumn(Bytes.toBytes(rowkeyString),  Bytes.toBytes(colString),
Bytes.toBytes(valString));
        table.put(put);
        table.close();
        close();
    }

    // add a record to a table
    public static void addRecord(String tableName, String row, String[]
fields, String[] values) throws IOException {
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Put put = new Put(Bytes.toBytes(row));

        for (int i = 0; i < fields.length; i++) {
            put = new Put(row.getBytes());
            String[] columns = fields[i].split(":");
            if(columns.length == 1){
                put.addColumn(columns[0].getBytes(),           "".getBytes(),
values[i].getBytes());
            }else{
                put.addColumn(columns[0].getBytes(),   columns[1].getBytes(),
values[i].getBytes());
            }
            table.put(put);
        }
        table.close();
```

```
        close();
    }

    // scan the data
    public static void scanData(String tableNameString, String rowkeyString,
String colFamilyString, String colString)
            throws IOException {
        init();

        Table                                table                                =
connection.getTable(TableName.valueOf(tableNameString));
        Get get = new Get(Bytes.toBytes(rowkeyString));
        get.addColumn(Bytes.toBytes(colFamilyString),
Bytes.toBytes(colString));
        Result result = table.get(get);
        System.out.println(new String(
                result.getValue(colFamilyString.getBytes(),    colString    ==
null ? null : colString.getBytes())));
        table.close();

        close();
    }

    // scan a column or a column family
    public  static  void  scanColumn(String  tableName,  String  columnFamily)
throws IOException{
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Scan scan = new Scan();

        ResultScanner resultScanner = table.getScanner(scan);
        String[] splitResult = columnFamily.split(":");

        if(splitResult.length == 1){
            scan.addFamily(columnFamily.getBytes());
        }else{
            scan.addColumn(Bytes.toBytes(splitResult[0]),
Bytes.toBytes(splitResult[1]));
        }
        resultScanner = table.getScanner(scan);
        for(Result r : resultScanner){
            Cell[] cells = r.rawCells();
            if(cells.length == 0) {
                System.out.println("null");
```

```
            }
            for(Cell cell : cells){

                System.out.println(new String(CellUtil.cloneRow(cell)) + "\t"
+    (CellUtil.cloneValue(cell).length    ==    0    ?    "null"    :    new
String(CellUtil.cloneValue(cell)))        +        "\t"        +        new
String(CellUtil.cloneQualifier(cell))+          "\t"          +          new
String(CellUtil.cloneFamily(cell)));
            }
        }
        resultScanner.close();
        close();
    }

    // modify the data
    public static void modifyData(String tableName, String row, String column,
String value) throws IOException{
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Put put = new Put(row.getBytes());
        put.addColumn(column.split(":")[0].getBytes(),
(column.split(":").length      ==      1      ?      "".getBytes()      :
column.split(":")[1].getBytes()), value.getBytes());
        table.put(put);
        close();
    }

    // delete the row
    public  static  void  deleteRow(String  tableName,  String  row)  throws
IOException{
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Delete delete = new Delete(row.getBytes());
        table.delete(delete);
        close();
    }
}
```

输出结果如下：

```
Start the progress of creating a table named 'StuInfo'.

log4j:WARN No appenders could be found for logger
(org.apache.hadoop.metrics2.lib.MutableMetricsFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

```
The table exists and will be deleted!
Add record of stu-2015001
Add record of stu-2015002
Add record of stu-2015003

Scanning the data of colfamily 'SI'...

2015001 23          age         SI
2015001 Zhangsan            name        SI
2015001 male        sex         SI
2015002 22          age         SI
2015002 Mary        name        SI
2015002 female    sex         SI
2015003 24          age         SI
2015003 Lisi        name        SI
2015003 male        sex         SI

Scanning the data of col 'Ma:score'...

2015001 86          score       Ma
2015002 null        score       Ma
2015003 98          score       Ma

Modifying data of the score of English(2015002)


Check...
2015001 69          score       Eng
2015002 100         score       Eng
2015003 null        score       Eng

Deleting the data of 2015003

Check...
2015001 23          age         SI
2015001 Zhangsan            name        SI
2015001 male        sex         SI
2015002 22          age         SI
2015002 Mary        name        SI
2015002 female    sex         SI
```