1. HBase Java API代码开发

几个主要HBase API类和数据模型之间的对应关系：

Java类	HBase数据模型	作用
HBaseAdmin	数据库（DataBase）	创建表，删除表，列出表项，使表有效或无效，以及添加或删除表列簇成员
HBaseConfiguration	数据库（DataBase）	配置管理相关
HTable	表（Table）	维护表的信息
HTableDescriptor	表（Table）	管理表和列簇相关
HColumnDescriptor	列簇（Column Family）	维护和管理列簇的信息，例如版本数量，是否压缩等
Put	插入/修改	添加数据
Get	查询	根据单个rowkey进行查询
Delete	删除	删除数据
Result	结果	get操作的结果
Scan	扫描（全表扫描/范围扫描）	扫描数据
ResultScanner	扫描结果	扫描操作的结果数据

1.1. 增删改查基本实现

代码如下：

```
package com.mazh.hbase.core.nx;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

/**
 * 作者：  马中华   https://blog.csdn.net/zhongqi2513
 * 时间：  2017/11/16 16:04
 * 描述：  测试HBase的增删改查
```

```
 */
public class HbaseDemoTest {
    // 声明静态配置
    static Configuration conf = null;

    private static final String ZK_CONNECT_STR =
"bigdata02:2181,bigdata03:2181,bigdata04:2181,bigdata05:2181";

    static {
        conf = HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum", ZK_CONNECT_STR);
    }

    /*
     * 创建表
     * @tableName 表名
     * @family 列簇列表
     */
    public static void creatTable(String tableName, String[] family) throws
Exception {
        HBaseAdmin admin = new HBaseAdmin(conf);
        HTableDescriptor desc = new HTableDescriptor(tableName);
        for (int i = 0; i < family.length; i++) {
            desc.addFamily(new HColumnDescriptor(family[i]));
        }
        if (admin.tableExists(tableName)) {
            System.out.println("table Exists!");
            System.exit(0);
        } else {
            admin.createTable(desc);
            System.out.println("create table Success!");
        }
    }

    /*
     * 为表添加数据（适合知道有多少列簇的固定表）
     * @rowKey rowKey
     * @tableName 表名
     * @column1 第一个列簇列表
     * @value1 第一个列的值的列表
     * @column2 第二个列簇列表
     * @value2 第二个列的值的列表
     */
    public static void addData(
            String rowKey, String tableName, String[] column1, String[] value1,
String[] column2,
            String[] value2) throws IOException {
        // 设置rowkey
        Put put = new Put(Bytes.toBytes(rowKey));

        // HTabel负责跟记录相关的操作如增删改查等//
        HTable table = new HTable(conf, Bytes.toBytes(tableName));

        // 获取所有的列簇
        HColumnDescriptor[] columnFamilies =
table.getTableDescriptor().getColumnFamilies();

        for (int i = 0; i < columnFamilies.length; i++) {
```

```
            // 获取列簇名
            String familyName = columnFamilies[i].getNameAsString();
            // article列簇put数据
            if (familyName.equals("article")) {
                for (int j = 0; j < column1.length; j++) {
                    put.add(Bytes.toBytes(familyName),
Bytes.toBytes(column1[j]), Bytes.toBytes(value1[j]));
                }
            }
            // author列簇put数据
            if (familyName.equals("author")) {
                for (int j = 0; j < column2.length; j++) {
                    put.add(Bytes.toBytes(familyName),
Bytes.toBytes(column2[j]), Bytes.toBytes(value2[j]));
                }
            }
        }
        table.put(put);
        System.out.println("add data Success!");
    }

    /*
     * 根据rwokey查询
     * @rowKey rowKey
     * @tableName 表名
     */
    public static Result getResult(String tableName, String rowKey) throws
IOException {
        Get get = new Get(Bytes.toBytes(rowKey));
        HTable table = new HTable(conf, Bytes.toBytes(tableName));// 获取表
        Result result = table.get(get);
        for (KeyValue kv : result.list()) {
            printKeyValue(kv);
        }
        return result;
    }

    public static void printKeyValue(KeyValue kv) {
        System.out.println("rowkey:" + Bytes.toString(kv.getRow()));
        System.out.println("family:" + Bytes.toString(kv.getFamily()));
        System.out.println("qualifier:" + Bytes.toString(kv.getQualifier()));
        System.out.println("value:" + Bytes.toString(kv.getValue()));
        System.out.println("Timestamp:" + kv.getTimestamp());
        System.out.println("-------------------------------------------");
    }

    public static void printResultScanner(ResultScanner rs) {
        for (Result r : rs) {
            for (KeyValue kv : r.list()) {
                printKeyValue(kv);
            }
        }
    }

    /*
     * 遍历查询hbase表
     * @tableName 表名
     */
```

```
    public static void getResultScann(String tableName) throws IOException {
        Scan scan = new Scan();
        ResultScanner rs = null;
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        try {
            rs = table.getScanner(scan);
            for (Result r : rs) {
                for (KeyValue kv : r.list()) {
                    printKeyValue(kv);
                }
            }
        } finally {
            rs.close();
        }
    }

    /*
     * 遍历查询hbase表
     * @tableName 表名
     * 切记：包括下界，不包括上界
     */
    public static void getResultScann(String tableName, String start_rowkey,
String stop_rowkey) throws IOException {
        Scan scan = new Scan();
        scan.setStartRow(Bytes.toBytes(start_rowkey));
        scan.setStopRow(Bytes.toBytes(stop_rowkey));
        ResultScanner rs = null;
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        try {
            rs = table.getScanner(scan);
            printResultScanner(rs);
        } finally {
            rs.close();
        }
    }

    /*
     * 查询表中的某一列
     * @tableName 表名
     * @rowKey rowKey
     */
    public static void getResultByColumn(String tableName, String rowKey, String
familyName, String columnName) throws IOException {
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        Get get = new Get(Bytes.toBytes(rowKey));

        // 获取指定列簇和列修饰符对应的列
        get.addColumn(Bytes.toBytes(familyName), Bytes.toBytes(columnName));
        Result result = table.get(get);
        for (KeyValue kv : result.list()) {
            printKeyValue(kv);
        }
    }

    /*
     * 更新表中的某一列
     * @tableName 表名
     * @rowKey rowKey
```

```
     * @familyName 列簇名
     * @columnName 列名
     * @value 更新后的值
     */
    public static void updateTable(
            String tableName, String rowKey, String familyName, String
columnName, String value) throws IOException {
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        Put put = new Put(Bytes.toBytes(rowKey));
        put.add(Bytes.toBytes(familyName), Bytes.toBytes(columnName),
Bytes.toBytes(value));
        table.put(put);
        System.out.println("update table Success!");
    }

    /*
     * 查询某列数据的多个版本
     * @tableName 表名
     * @rowKey rowKey
     * @familyName 列簇名
     * @columnName 列名
     */
    public static void getResultByVersion(String tableName, String rowKey,
String familyName, String columnName) throws IOException {
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        Get get = new Get(Bytes.toBytes(rowKey));
        get.addColumn(Bytes.toBytes(familyName), Bytes.toBytes(columnName));
        get.setMaxVersions(5);
        Result result = table.get(get);
        for (KeyValue kv : result.list()) {
            printKeyValue(kv);
        }
    }

    /*
     * 删除指定的列
     * @tableName 表名
     * @rowKey rowKey
     * @familyName 列簇名
     * @columnName 列名
     */
    public static void deleteColumn(
            String tableName, String rowKey, String falilyName, String
columnName) throws IOException {
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        Delete deleteColumn = new Delete(Bytes.toBytes(rowKey));
        deleteColumn.deleteColumns(Bytes.toBytes(falilyName),
Bytes.toBytes(columnName));
        table.delete(deleteColumn);
        System.out.println(falilyName + ":" + columnName + "is deleted!");
    }

    /*
     * 删除指定的列
     * @tableName 表名
     * @rowKey rowKey
     */
```

```
    public static void deleteAllColumn(String tableName, String rowKey) throws
IOException {
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        Delete deleteAll = new Delete(Bytes.toBytes(rowKey));
        table.delete(deleteAll);
        System.out.println("all columns are deleted!");
    }

    /*
     * 删除表
     * @tableName 表名
     */
    public static void deleteTable(String tableName) throws IOException {
        HBaseAdmin admin = new HBaseAdmin(conf);
        admin.disableTable(tableName);
        admin.deleteTable(tableName);
        System.out.println(tableName + "is deleted!");
    }

    public static void main(String[] args) throws Exception {
        // 创建表
        String tableName = "blog";
        String[] family = { "article", "author" };
        creatTable(tableName, family);

        // 为表添加数据
        String[] column1 = { "title", "content", "tag" };
        String[] value1 = {
                "Head First HBase",
                "HBase is the Hadoop database",
                "Hadoop,HBase,NoSQL" };
        String[] column2 = { "name", "nickname" };
        String[] value2 = { "nicholas", "lee" };
        String[] value3 = { "lilaoshi", "malaoshi" };
        addData("rowkey1", "blog", column1, value1, column2, value2);
        addData("rowkey1", "blog", column1, value1, column2, value3);
        addData("rowkey2", "blog", column1, value1, column2, value2);
        addData("rowkey3", "blog", column1, value1, column2, value2);

        // 遍历查询， 根据row key范围遍历查询
//        getResultScann("blog", "rowkey2", "rowkey3");

        // 查询
//        getResult("blog", "rowkey1");

        // 查询某一列的值
//        getResultByColumn("blog", "rowkey1", "author", "name");

        // 更新列
//        updateTable("blog", "rowkey1", "author", "name", "bin");

        // 查询某一列的值
//        getResultByColumn("blog", "rowkey1", "author", "name");

        // 查询某列的多版本
//        getResultByVersion("blog", "rowkey1", "author", "name");

        // 删除一列
```

```
//        deleteColumn("blog", "rowkey1", "author", "nickname");

        // 删除所有列
//        deleteAllColumn("blog", "rowkey1");

        // 删除表
//        deleteTable("blog");
    }
}
```