第7单HBase分布式数据库实训

(源自:https://biglab.site)

(版本:Ver1.0-20231120)

实训1使用HBase Shell命令创建学生表并插入数据

实训要点

1. 掌握HBase中表的创建（create）、启用（enable）、删除（drop）、修改（alter）等管理命令的使用方法。
2. 掌握HBase中表数据的添加（put）、获取（get、扫描（scan）、删除（delete）、统计（count）等数据操作命令的使用方法。

需求说明

为了更好地管理数据，每个数据表的设计都是需要经过仔细思考的。某高校学院为了便于管理学生信息，决定将学生的基本信息和学生成绩信息合并在一起。现已为该需求设计了一张表，为student表，表结构如表7-9所示。请根据student表的表结构，使用HBase创建该表，并将学生数据插入student表中，以便后续对学生信息进行查询、分析。

student 表的表结构：

RowKey	**address**			**score**		
province	city	street	Java	Python	Web	
Tom	chongqing	hechuan	yiwudadao	85	82	93
Jack	guangdong	guangzhou	kaitaidadao	87	83	80

实现思路及步骤

1. 启动HBase服务，进入HBase Shell界面。
2. 使用“list”命令查看当前HBase中的表，若没有则建立新表，若有则先删除表。根据student表的表结构创建student表，包含两个列族address和score。
3. 查看student表的描述信息，同时使得该表处于启用状态。
4. 向 student 表中添加表《student 表的表结构》所示的数据。

作业要求

1. 在Hive中创建myname数据库，在myname库中创建student表，截图建表语句和运行结果；
2. 在Hive中查看student表的描述信息，同时使得该表处于启用状态，截图脚本和运行结果；
3. 在Hive中添加表《student 表的表结构》所示的数据，截图脚本和运行结果；

实训参考

脚本参考

以下在master主机或slave从机的hbase shell命令行中执行：

```
#若表存在，先删除
drop 'student'
#创建student表(包含address和score两个列族)
create 'student','address','score'
#查看student表中描述信息
describe 'student'
#先查看表是否处于启用状态
is_enabled 'student'
#如果不是，用下面的命令让其处于启动状态
enable 'student'
#向表中添加数据
put 'student','Tom','address:province','chongqing'
put 'student','Tom','address:city','hechuan'
put 'student','Tom','address:street','yiwudadao'
put 'student','Tom','score:Java',85
put 'student','Tom','score:Python',82
put 'student','Tom','score:Web',93
put 'student','Jack','address:province','guangdong'
put 'student','Jack','address:city','guangzhou'
put 'student','Jack','address:street','kaitaidadao'
put 'student','Jack','score:Java',87
put 'student','Jack','score:Python',83
put 'student','Jack','score:Web',80

```

运行结果

```
hbase:066:0> #若表存在，先删除
drop 'student'hbase:067:0> drop 'student'
Took 0.7120 seconds



hbase:068:0> #创建student表(包含address和score两个列族)
create 'student','address','score'
```

```
hbase:069:0> create 'student','address','score'
#查看student表中描述信息
describe 'student'
#先查看表是否处于启用状态
is_enabled 'student'
#如果不是，用下面的命令让其处于启动状态
enable 'student'
#向表中添加数据
put 'student','Tom','address:province','chongqing'
put 'student','Tom','address:city','hechuan'
put 'student','Tom','address:street','yiwudadao'
put 'student','Tom','score:Java',85
put 'student','Tom','score:Python',82
put 'student','Tom','score:Web',93
put 'student','Jack','address:province','guangdong'
put 'student','Jack','address:city','guangzhou'
put 'student','Jack','address:street','kaitaidadao'
put 'student','Jack','score:Java',87
put 'student','Jack','score:Python',83
put 'student','Jack','score:Web',80
    Created table student
Took 1.1720 seconds


=> Hbase::Table - student
hbase:070:0> #查看student表中描述信息
hbase:071:0> describe 'student'
Table student is ENABLED


student


COLUMN FAMILIES DESCRIPTION


{NAME => 'address', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS =>
'1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE',
COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE =>
'
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}



{NAME => 'score', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS =>
'1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE',
COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE =>
'tr
ue', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}



2 row(s)
Quota is disabled
```

```
Took 0.0714 seconds


hbase:072:0> #先查看表是否处于启用状态
hbase:073:0> is_enabled 'student'
true


Took 0.0085 seconds


=> true
hbase:074:0> #如果不是，用下面的命令让其处于启动状态
hbase:075:0> enable 'student'
Took 0.0543 seconds


hbase:076:0> #向表中添加数据
hbase:077:0> put 'student','Tom','address:province','chongqing'
Took 0.1186 seconds


hbase:078:0> put 'student','Tom','address:city','hechuan'
Took 0.0065 seconds


hbase:079:0> put 'student','Tom','address:street','yiwudadao'
Took 0.0247 seconds


hbase:080:0> put 'student','Tom','score:Java',85
Took 0.0213 seconds


hbase:081:0> put 'student','Tom','score:Python',82
Took 0.0083 seconds


hbase:082:0> put 'student','Tom','score:Web',93
Took 0.0062 seconds


hbase:083:0> put 'student','Jack','address:province','guangdong'
Took 0.0102 seconds


hbase:084:0> put 'student','Jack','address:city','guangzhou'
Took 0.0094 seconds


hbase:085:0> put 'student','Jack','address:street','kaitaidadao'
Took 0.0073 seconds


hbase:086:0> put 'student','Jack','score:Java',87
```

```
Took 0.0362 seconds


hbase:087:0> put 'student','Jack','score:Python',83
Took 0.0113 seconds


hbase:088:0> put 'student','Jack','score:Web',80
Took 0.0086 seconds


hbase:089:0>
```

实训2使用HBase Java API对HBase数据库的表进行操作

实训要点

1. 掌握通过Java API操作HBase分布式数据库的方法。
2. 掌握使用IntelliJ IDEA创建工程并导入HBase依赖包的方法。

需求说明

用HBase 存储社交网站站内短信信息，要求记录发送者、时间、内容、接收者，如表7-10所示。使用HBase创建info _ table表，将表7-10中的数据插入info _ table表中，并验证数据是否插入成功，并查询指定发送者发出的所有信息和指定接受者接收的所有信息。

社交网站站内短信信息:

发送者	时间	内容	接收者	
Lilei	202106191530	今天上哪门课程呀?	Hanmeimei	
Hanmeimei	202106191531	大数据技术原理及应用。	Lilei	
Lilei	202106191531	是吗?	Hanmeimei	
Hanmeimei	202106191532	是的，这门课程还挺难的。	Lilei	
Lilei	202106191532	那我们要怎么学习这门课程呢?	Hanmeimei	
Lilei	202106191555	还在吗?	Hanmeimei	

实现思路及步骤

1. 创建Java工程HBaseDemo，并导入依赖包。
2. 创建名为HBaseOperation的Java类。
3. 定义init（）和close（）方法，完成与HBase数据库的连接和关闭。
4. 创建info _ table表。
5. 向info _ table表中插入表7-10中的数据。
6. 查询info _ table 表中的数据以验证是否插入成功，再使用get（）方法查看发送者发送的所有信息和接收者接收的所有信息。

实训参考

数据表准备

以下在master主机或slave从机的hbase shell命令行中执行：

```
#若表存在，先删除
drop 'students'
```

结果

```
hbase:089:0> #若表存在，先删除
drop 'students'hbase:090:0> drop 'students'

ERROR: Table students is enabled. Disable it first.

For usage try 'help "drop"'

Took 0.0212 seconds


hbase:091:0>
```

源码参考

```
package chap7_hbase;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

public class HBaseApi {
    public static Configuration configuration;  //管理HBase的配置信息
    public static Connection connection;  //管理HBase的连接
    public static Admin admin;   //管理HBase数据库表信息
    public static void main(String[] args)throws IOException {
        init();
        System.out.println(connection);
        createTable("info_table",new String[]{"发送者","接收者"});
        insertData("info_table","Lilei","发送者","时间","202106191530");
        insertData("info_table","Lilei","发送者","内容","今天上哪门课程呀？");
        insertData("info_table","Hanmeimei","发送者","时间","202106191531");
        insertData("info_table","Hanmeimei","发送者","内容","大数据技术原理及应用");
        insertData("info_table","Lilei","发送者","时间","202106191531");
        insertData("info_table","Lilei","发送者","内容","是吗？");
        insertData("info_table","Hanmeimei","发送者","时间","202106191532");
```

```
        insertData("info_table","Hanmeimei","发送者","内容","是的，这门课程还挺难的。");
        insertData("info_table","Lilei","发送者","时间","202106191532");
        insertData("info_table","Lilei","发送者","内容","那我们要怎么学习这门课程呢？");
        insertData("info_table","Lilei","发送者","时间","202106191555");
        insertData("info_table","Lilei","发送者","内容","还在吗");
        insertData("info_table","Hanmeimei","接收者","时间","202106191530");
        insertData("info_table","Hanmeimei","接收者","内容","今天上哪门课程呀？");
        insertData("info_table","Lilei","接收者","时间","202106191531");
        insertData("info_table","Lilei","接收者","内容","大数据技术原理及应用？");
        insertData("info_table","Hanmeimei","接收者","时间","202106191531");
        insertData("info_table","Hanmeimei","接收者","内容","是吗？");
        insertData("info_table","Lilei","接收者","时间","202106191532");
        insertData("info_table","Lilei","接收者","内容","是的，这门课程还挺难的？");
        insertData("info_table","Hanmeimei","接收者","时间","202106191532");
        insertData("info_table","Hanmeimei","接收者","内容","那我们要怎么学习这门课程呢？");
        insertData("info_table","Hanmeimei","接收者","时间","202106191555");
        insertData("info_table","Hanmeimei","接收者","内容","还在吗");
        close();
    }
    //建立连接
    public static void init(){
        configuration  = HBaseConfiguration.create();

configuration.set("hbase.rootdir","hdfs://master:8020/data/hbase_db");
        configuration.set("hbase.zookeeper.quorum","master,slave1,slave2");
        try{
            connection = ConnectionFactory.createConnection(configuration);
            admin = connection.getAdmin();
        }catch (IOException e){
            e.printStackTrace();
        }
    }
    // 关闭连接
    public static void close() throws IOException {
        try{
            if(admin != null){
                admin.close();
            }
            if(null != connection){
                connection.close();
            }
        }catch (IOException e){
            e.printStackTrace();
        }
    }
    /**
     *
     * @param myTableName 表名
     * @param colFamily 列族
```

```
     * @throws IOException
     */
    //创建表
    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());
        }
    }
    //
    /**
     * 添加数据
     * @param tableName 表名
     * @param rowKey    行键
     * @param colFamily 列族
     * @param col       列限定符
     * @param val       数据
     * @throws IOException
     */

    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();
    }

}
```

运行结果

```
。。。。。
2023/11/20 12:41:40,275- ZooKeeper: Client environment:os.arch=amd64
2023/11/20 12:41:40,275- ZooKeeper: Client environment:os.version=10.0
2023/11/20 12:41:40,275- ZooKeeper: Client environment:user.name=yuxm
2023/11/20 12:41:40,275- ZooKeeper: Client
environment:user.home=C:\Users\yuxm
2023/11/20 12:41:40,275- ZooKeeper: Client
environment:user.dir=D:\yuxm\wordc2
```

```
2023/11/20 12:41:40,276- ZooKeeper: Initiating client connection, connectString=master:2181,slave1:2181,slave2:2181 sessionTimeout=90000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$8/802055745@46d7a6a5
2023/11/20 12:41:40,520- ClientCnxn: Opening socket connection to server master/192.168.128.130:2181. Will not attempt to authenticate using SASL (unknown error)
2023/11/20 12:41:40,520- ClientCnxn: Socket connection established to master/192.168.128.130:2181, initiating session
2023/11/20 12:41:40,530- ClientCnxn: Session establishment complete on server master/192.168.128.130:2181, sessionid = 0x10000544c9a0004, negotiated timeout = 40000
hconnection-0x68267da0
2023/11/20 12:41:42,603- HBaseAdmin: Operation: CREATE, Table Name: default:info_table, procId: 133 completed
2023/11/20 12:41:42,852- ConnectionImplementation: Closing master protocol: MasterService

Process finished with exit code 0
```

的hbase shell里查看结果

```
hbase:091:0> list
TABLE
info_table
phone_log
student
students
4 row(s)
Took 0.0095 seconds
=> ["info_table", "phone_log", "student", "students"]
hbase:092:0> scan 'students'
ROW                                                    COLUMN+CELL
 George                                                column=score:Bigdata, timestamp=2023-11-16T13:40:27.942, value=69
 George                                                column=score:JavaWeb, timestamp=2023-11-16T13:40:27.968, value=77

 George                                                column=score:Python, timestamp=2023-11-16T13:40:27.963, value=86
1 row(s)
Took 0.1525 seconds
hbase:093:0>
```