**MapReduce操作HBase数据**

**实验环境（普开实验“数据库技术(NoSQL)”对应章节虚拟机）**

虚拟机数量：3

系统版本：Centos 7.5

Hadoop版本：Apache Hadoop 2.7.3

Zookeeper版本：Apache Zookeeper 3.4.10

Eclipse版本：Neon.3 4.6.3

HBase版本：Apache HBase 1.2.6

1. **HBase编程：读取HBase表数据**
2. **实验描述（参照普开实验“数据库技术(NoSQL)”第15章）**

使用MapReduce批量将HBase表中数据导入到HDFS上，学习本实验将掌握通过MapReduce编程来连接并读取HBase数据表，学习HBase数据库在分布式计算中的应用。

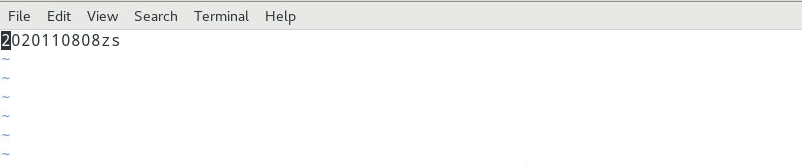
1. **主要步骤：**

修改主机名：主机名格式：学号+自己姓名简称（未修改扣分）





/etc/hostname文件内容如下



* 启动Hadoop集群

在master运行：



* 启动Zookeeper集群

需要在master、slave01、slave02分别运行：



* 启动HBase集群

在master运行：



* 进入HBase Shell创建实验用表

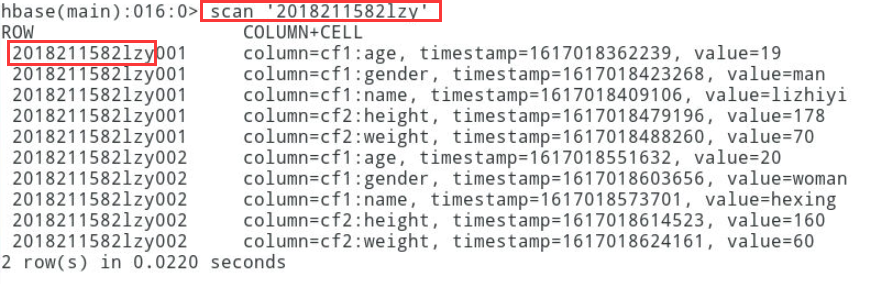
输入hbase shell进入hbase交互式环境：



数据库表格设计要求：（未按要求设计扣分）

1. 表格命名：学号+姓名（如截图1）
2. 列族数量：至少两列
3. 某一列族下至少有两个子列
4. 行数不限定，字段名不限定
5. ROW命名：学号+姓名+编号（如截图1）

【截图1：数据库表格】（截图需要包含标记信息，未按要求扣分）



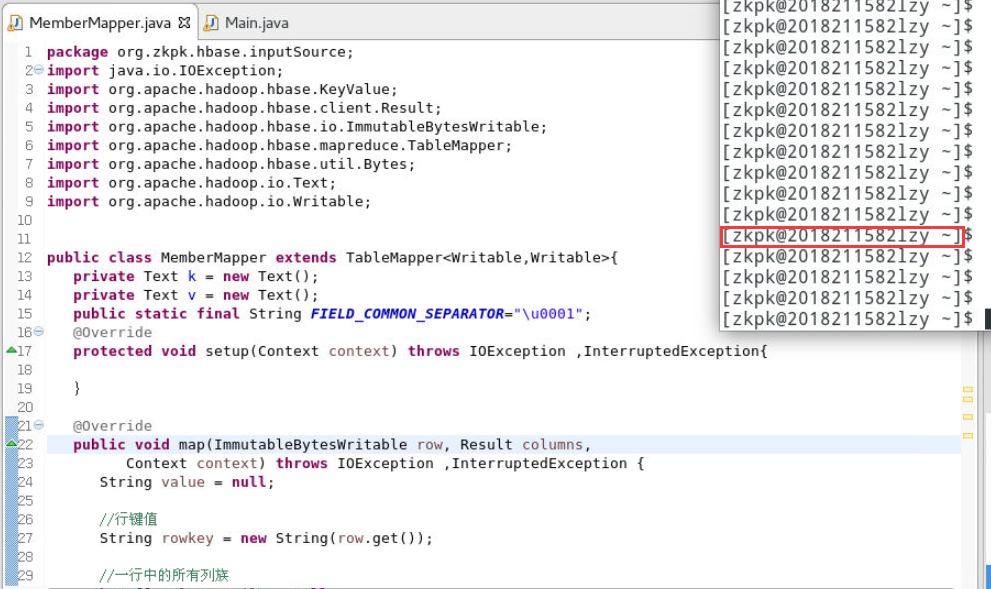
**说明：**这里我的设置的表名为2018211582lzy，列族数量两个，分别为cf1和cf2，其中cf1列族包含3个子列，分别为age、gender和name，cf2列族包含2个子列，分别为weight和height，这里我填充了部分数据，如图所展示。

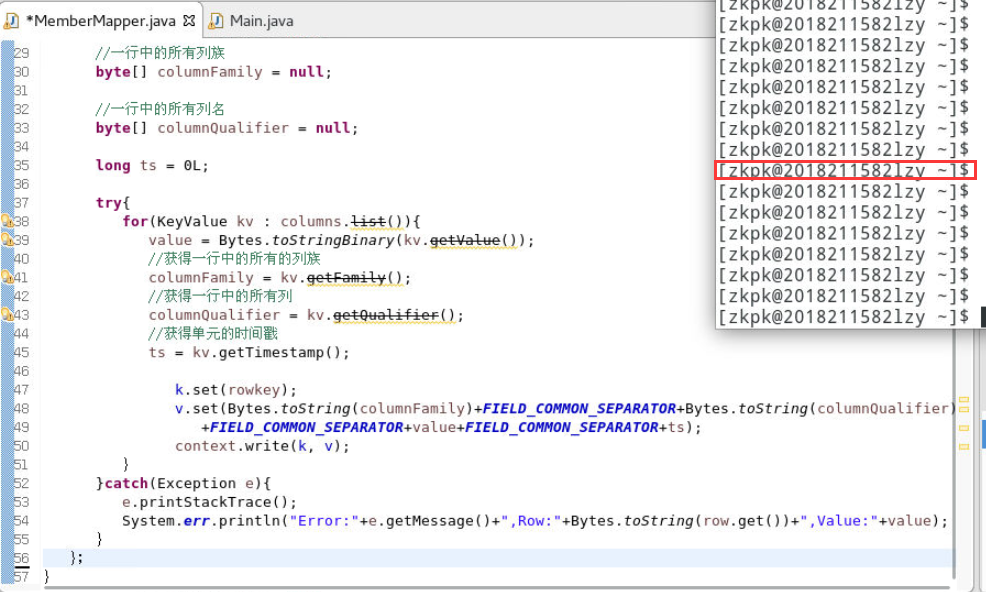
* 编写代码

【截图2：完整Mapper代码】

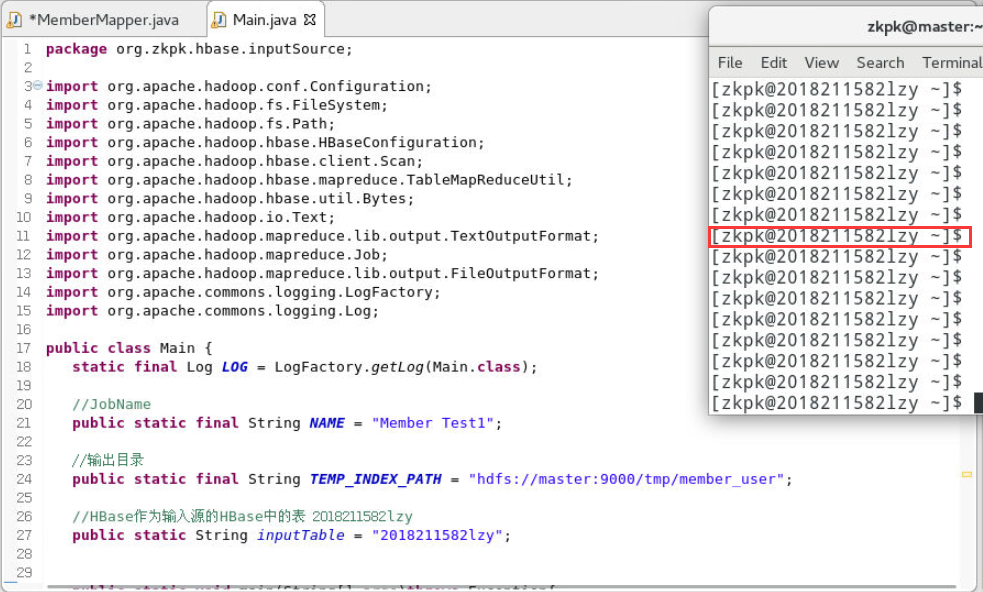
（截图需要包含标记信息，未按要求扣分，代码不完整扣分）

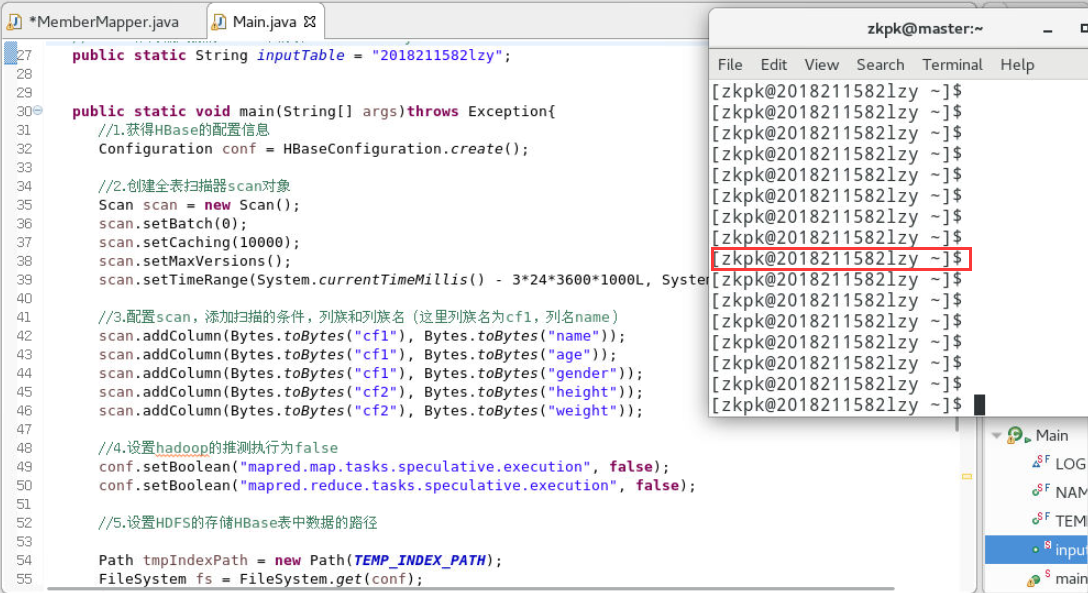
**MemberMapper.java:**

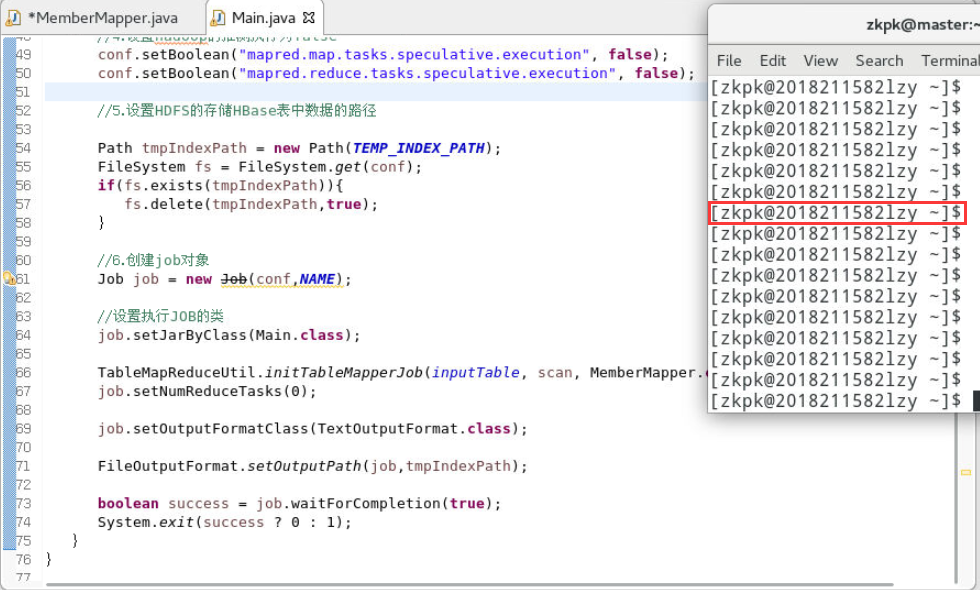




**Main.java:**



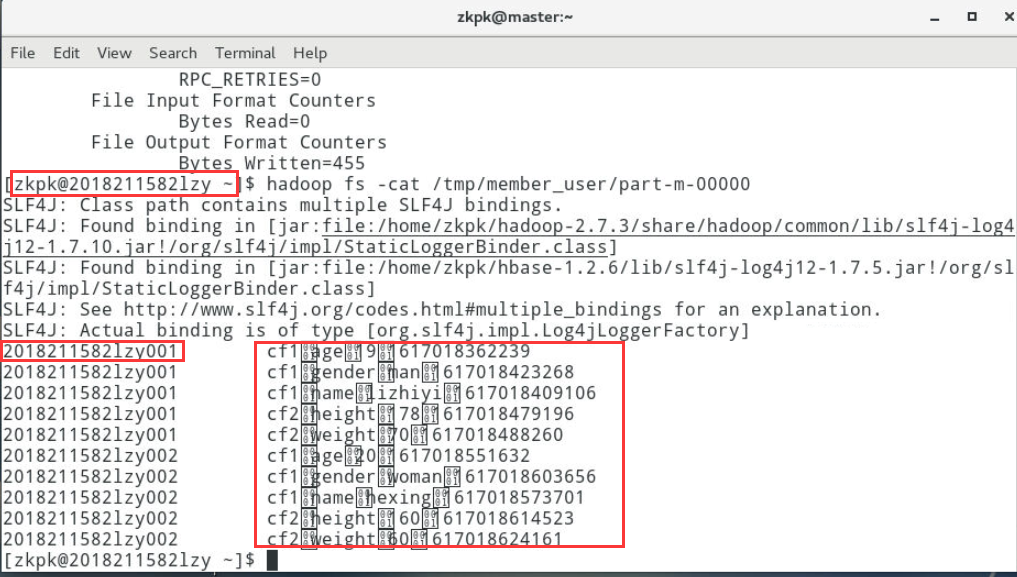




* 打包程序
* 运行程序，查看结果

【截图3：结果截图】（截图需要包含标记信息，未按要求扣分）

**运行结果：**

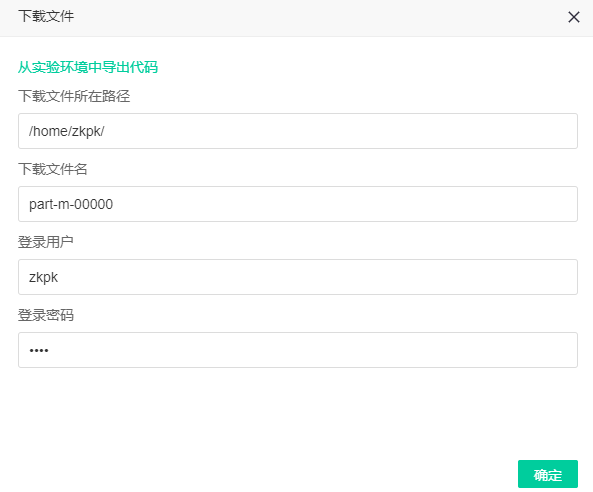


**提示：**需要把”part-m-00000”文件保存到自己电脑上（实验二需要这个数据）

* 拉取文件命令：



* 把”part-m-00000”文件从虚拟机下载到自己电脑：



1. **HBase编程：存储数据至HBase表**
2. **实验描述（参照普开实验“数据库技术(NoSQL)”第16章）**

使用MapReduce批量将HDFS上的数据导入到HBase表中，学习本实验能快速掌握HBase数据库在分布式计算中的应用，理解Java API读取HBase数据等相关内容。

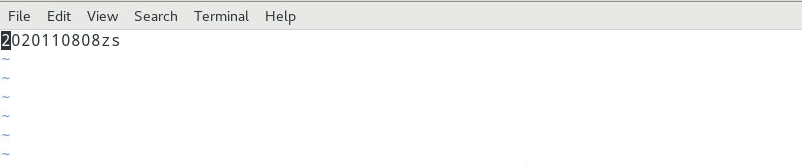
1. **主要步骤**

修改主机名：主机名格式：学号+自己姓名简称（未修改扣分）





/etc/hostname文件内容如下



* 启动Hadoop集群

在master运行：



* 启动Zookeeper集群

需要在master、slave01、slave02分别运行：



* 启动HBase集群

在master运行：



* 进入HBase Shell创建实验用表

输入hbase shell进入hbase交互式环境：



表格要求与实验一一致

由于表格不存在，需要手动创建（不需要插入数据）

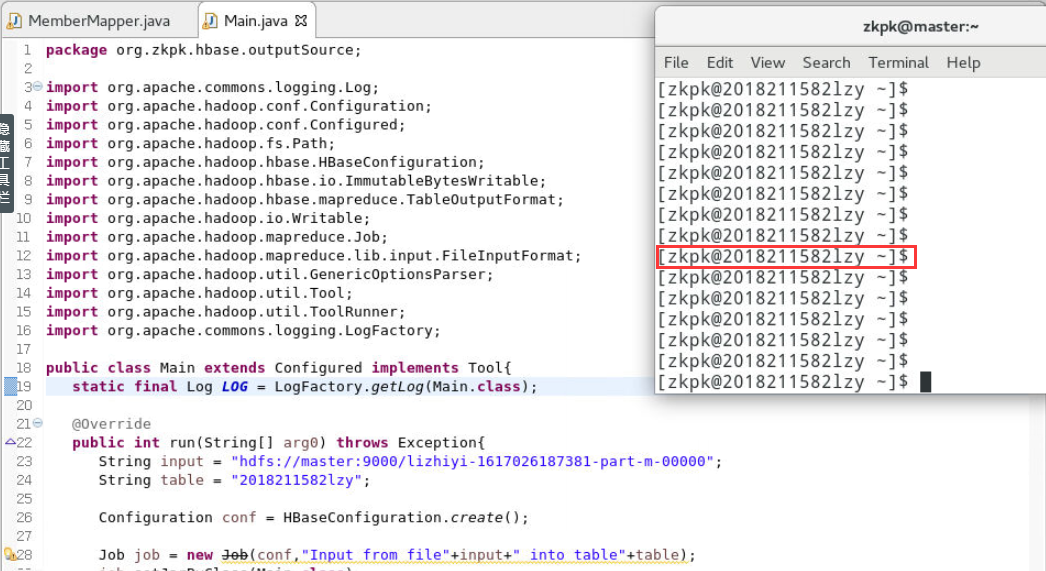


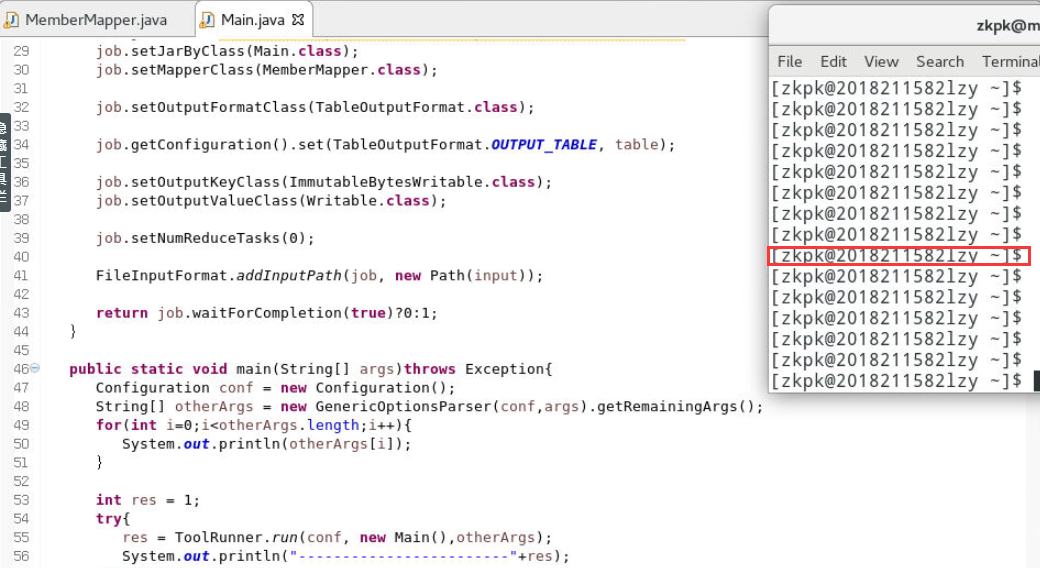
* 打开Eclipse
* 编写代码

【截图4：截取完整Mapper.java代码】

（截图需要包含标记信息，未按要求扣分，代码不完整扣分）

**Main.java:**







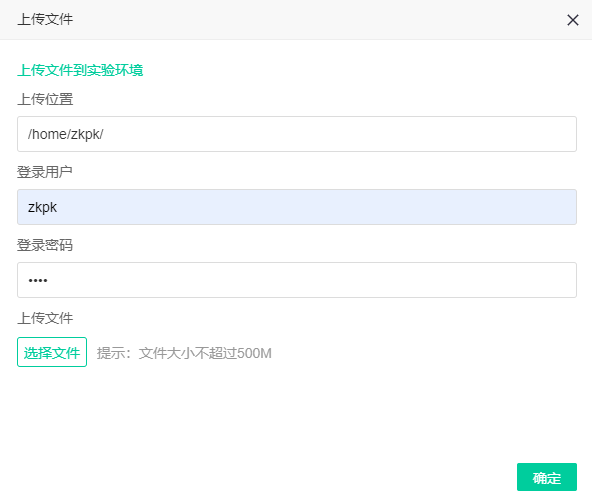
**MemberMappper.java:**





* 打包程序
* 上传实验一下载的“part-m-0000”文件(如果忘记下载，需要手动创建)

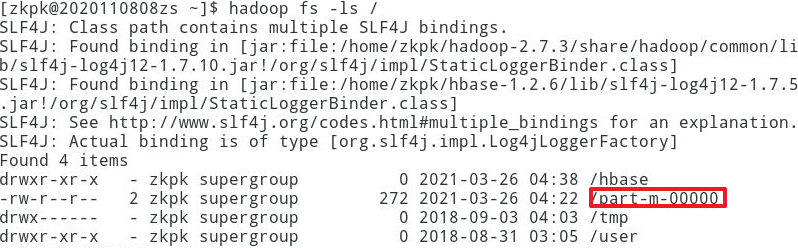
上传文件：



上传“part-m-0000”到hdfs命令：

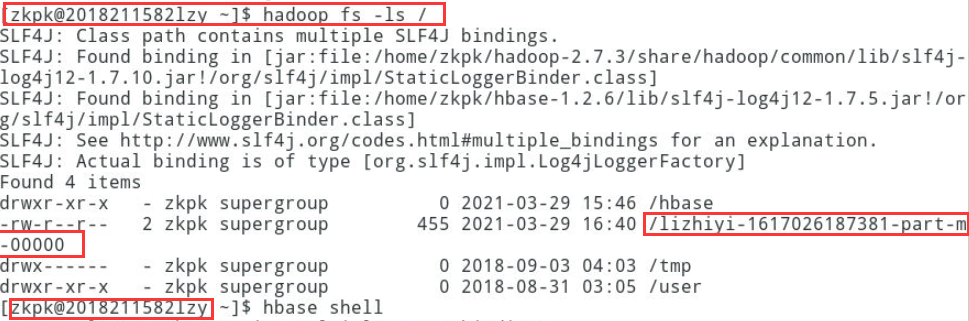


查看是否上传成功：



* 运行jar包，查看结果

【截图5：结果截图】（截图需要包含标记信息，未按要求扣分）







**附 录**

**实验一:**

**Main.java**

**package** org.zkpk.hbase.inputSource;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.FileSystem;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.hbase.HBaseConfiguration;

**import** org.apache.hadoop.hbase.client.Scan;

**import** org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;

**import** org.apache.hadoop.hbase.util.Bytes;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**import** org.apache.commons.logging.LogFactory;

**import** org.apache.commons.logging.Log;

**public** **class** Main {

**static** **final** Log LOG = LogFactory.getLog(Main.**class**);

*//JobName*

**public** **static** **final** String NAME = "Member Test1";

*//输出目录*

**public** **static** **final** String TEMP\_INDEX\_PATH = "hdfs://master:9000/tmp/member\_user";

*//HBase作为输入源的HBase中的表 2018211582lzy*

**public** **static** String inputTable = "2018211582lzy";

**public** **static** **void** main(String[] args)**throws** Exception{

*//1.获得HBase的配置信息*

Configuration conf = HBaseConfiguration.create();

*//2.创建全表扫描器scan对象*

Scan scan = **new** Scan();

scan.setBatch(0);

scan.setCaching(10000);

scan.setMaxVersions();

scan.setTimeRange(System.currentTimeMillis() - 3\*24\*3600\*1000L, System.currentTimeMillis());

*//3.配置scan，添加扫描的条件，列族和列族名（这里列族名为cf1，列名name）*

scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("name"));

scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("age"));

scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("gender"));

scan.addColumn(Bytes.toBytes("cf2"), Bytes.toBytes("height"));

scan.addColumn(Bytes.toBytes("cf2"), Bytes.toBytes("weight"));

*//4.设置hadoop的推测执行为false*

conf.setBoolean("mapred.map.tasks.speculative.execution", **false**);

conf.setBoolean("mapred.reduce.tasks.speculative.execution", **false**);

*//5.设置HDFS的存储HBase表中数据的路径*

Path tmpIndexPath = **new** Path(TEMP\_INDEX\_PATH);

FileSystem fs = FileSystem.get(conf);

**if**(fs.exists(tmpIndexPath)){

fs.delete(tmpIndexPath,**true**);

}

*//6.创建job对象*

Job job = **new** Job(conf,NAME);

*//设置执行JOB的类*

job.setJarByClass(Main.**class**);

TableMapReduceUtil.initTableMapperJob(inputTable, scan, MemberMapper.**class**, Text.**class**, Text.**class**, job);

job.setNumReduceTasks(0);

job.setOutputFormatClass(TextOutputFormat.**class**);

FileOutputFormat.setOutputPath(job,tmpIndexPath);

**boolean** success = job.waitForCompletion(**true**);

System.exit(success ? 0 : 1);

}

}

**MemberMapper.java**

**package** org.zkpk.hbase.inputSource;

**import** java.io.IOException;

**import** org.apache.hadoop.hbase.KeyValue;

**import** org.apache.hadoop.hbase.client.Result;

**import** org.apache.hadoop.hbase.io.ImmutableBytesWritable;

**import** org.apache.hadoop.hbase.mapreduce.TableMapper;

**import** org.apache.hadoop.hbase.util.Bytes;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.io.Writable;

**public** **class** MemberMapper **extends** TableMapper<Writable,Writable>{

**private** Text k = **new** Text();

**private** Text v = **new** Text();

**public** **static** **final** String FIELD\_COMMON\_SEPARATOR="";

@Override

**protected** **void** setup(Context context) **throws** IOException ,InterruptedException{

}

@Override

**public** **void** map(ImmutableBytesWritable row, Result columns,

Context context) **throws** IOException ,InterruptedException {

String value = **null**;

*//行键值*

String rowkey = **new** String(row.get());

*//一行中的所有列族*

**byte**[] columnFamily = **null**;

*//一行中的所有列名*

**byte**[] columnQualifier = **null**;

**long** ts = 0L;

**try**{

**for**(KeyValue kv : columns.list()){

value = Bytes.toStringBinary(kv.getValue());

*//获得一行中的所有的列族*

columnFamily = kv.getFamily();

*//获得一行中的所有列*

columnQualifier = kv.getQualifier();

*//获得单元的时间戳*

ts = kv.getTimestamp();

k.set(rowkey);

v.set(Bytes.toString(columnFamily)+FIELD\_COMMON\_SEPARATOR+Bytes.toString(columnQualifier)

+FIELD\_COMMON\_SEPARATOR+value+FIELD\_COMMON\_SEPARATOR+ts);

context.write(k, v);

}

}**catch**(Exception e){

e.printStackTrace();

System.err.println("Error:"+e.getMessage()+",Row:"+Bytes.toString(row.get())+",Value:"+value);

}

};

}

**实验二:**

**Main.java**

**package** org.zkpk.hbase.outputSource;

**import** org.apache.commons.logging.Log;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.conf.Configured;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.hbase.HBaseConfiguration;

**import** org.apache.hadoop.hbase.io.ImmutableBytesWritable;

**import** org.apache.hadoop.hbase.mapreduce.TableOutputFormat;

**import** org.apache.hadoop.io.Writable;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.util.GenericOptionsParser;

**import** org.apache.hadoop.util.Tool;

**import** org.apache.hadoop.util.ToolRunner;

**import** org.apache.commons.logging.LogFactory;

**public** **class** Main **extends** Configured **implements** Tool{

**static** **final** Log LOG = LogFactory.getLog(Main.**class**);

@Override

**public** **int** run(String[] arg0) **throws** Exception{

String input = "hdfs://master:9000/lizhiyi-1617026187381-part-m-00000";

String table = "2018211582lzy";

Configuration conf = HBaseConfiguration.create();

Job job = **new** Job(conf,"Input from file"+input+" into table"+table);

job.setJarByClass(Main.**class**);

job.setMapperClass(MemberMapper.**class**);

job.setOutputFormatClass(TableOutputFormat.**class**);

job.getConfiguration().set(TableOutputFormat.OUTPUT\_TABLE, table);

job.setOutputKeyClass(ImmutableBytesWritable.**class**);

job.setOutputValueClass(Writable.**class**);

job.setNumReduceTasks(0);

FileInputFormat.addInputPath(job, **new** Path(input));

**return** job.waitForCompletion(**true**)?0:1;

}

**public** **static** **void** main(String[] args)**throws** Exception{

Configuration conf = **new** Configuration();

String[] otherArgs = **new** GenericOptionsParser(conf,args).getRemainingArgs();

**for**(**int** i=0;i<otherArgs.length;i++){

System.out.println(otherArgs[i]);

}

**int** res = 1;

**try**{

res = ToolRunner.run(conf, **new** Main(),otherArgs);

System.out.println("------------------------"+res);

}**catch**(Exception e){

e.printStackTrace();

}

System.exit(res);

}

}

**MemberMapper.java**

**package** org.zkpk.hbase.outputSource;

**import** org.apache.hadoop.hbase.client.Put;

**import** org.apache.hadoop.hbase.io.ImmutableBytesWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Mapper;

**public** **class** MemberMapper **extends** Mapper<LongWritable,Text,

ImmutableBytesWritable,Put>{

*//列族*

**private** **byte**[] family = **null**;

*//列名*

**private** **byte**[] qualifier = **null**;

*//单元格存放的值*

**private** **byte**[] val = **null**;

*//行键*

**private** String rowkey = **null**;

*//时间戳*

**private** **long** ts = System.currentTimeMillis();

**protected** **void** map(LongWritable key,Text value,Context context)

**throws** java.io.IOException , InterruptedException{

**try**{

String lineString = value.toString();

String[] arr = lineString.split("**\t**",-1);

**if**(arr.length==2){

rowkey = arr[0];

String[] vals = arr[1].split("",-1);

**if**(vals.length==4){

family = vals[0].getBytes();

qualifier = vals[1].getBytes();

val = vals[2].getBytes();

ts = Long.parseLong(vals[3]);

Put put = **new** Put(rowkey.getBytes(),ts);

put.add(family, qualifier, val);

context.write(**new** ImmutableBytesWritable(rowkey.getBytes()), put);

}

}

}**catch**(Exception e){

e.printStackTrace();

}

}

}