大数据hw3报告

单人作业。安装HBase/Hive/Pig，它们的安装指导是以Hadoop文档中伪分布式Hadoop为基础的，有兴趣的同学亦可自行尝试在全分布式Hadoop环境下进行安装。 将Titanic数据导入三个数据库，设计并实现两个对于该数据集实用的查询功能。

报告内容及要求：

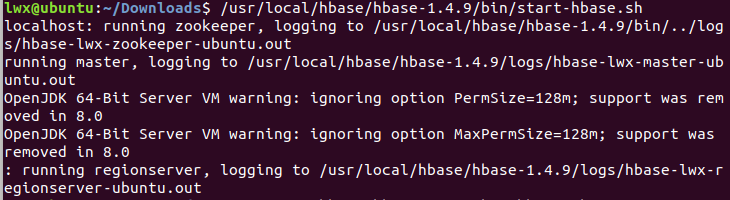
请在报告中详细写明你的实验步骤、技术方法、结果与分析、心得与反馈等，必要时附上相应的代码段或截图。

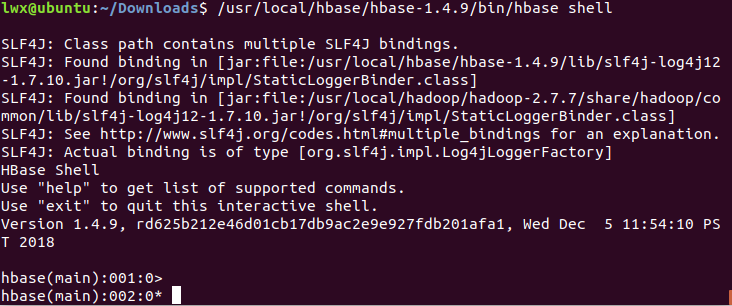
截止时间：请同学们于北京时间2019年5月6日23:59前，提交pdf形式的上机报告，文件名格式为："报告2-张三-1500012345-李四-1600054321.pdf"。

注意事项：请按时提交。每延迟一天当次作业成绩扣除20%，三天以上记为0。

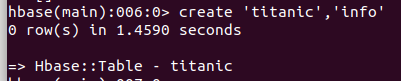
# HBASE

## 实验过程

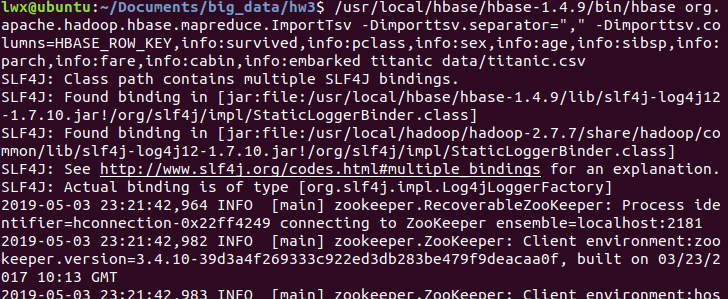


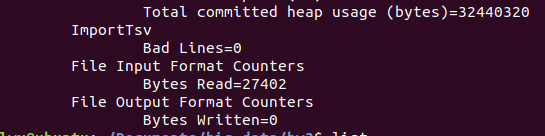


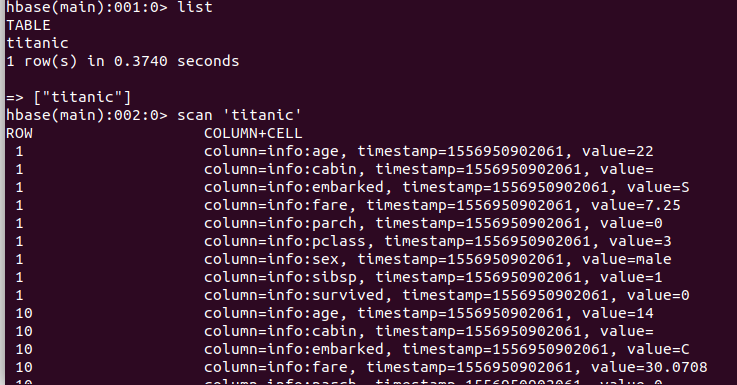
创建表titanic，列族名info



将titanic.csv上传到hdfs中，然后用ImportTsv读取该csv文件，指定不同列的名称，该方法将调用mapreduce将文件读到hbase中的表titanic中

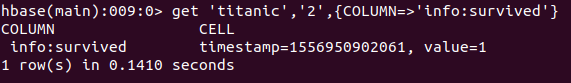


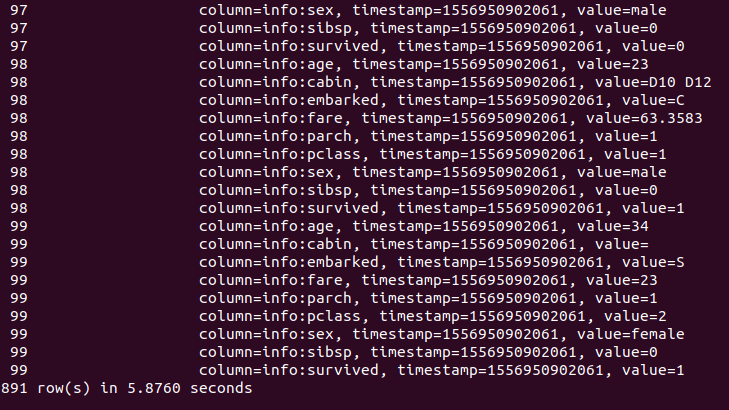




### 查询1

在hbase shell 中查询2号乘客是否幸存





### 查询2

查询幸存总人数。调用java api查询。定义count\_survived,get\_row\_survived等函数。建立连接，逐行获取info:survived列的值，存活为1，否则为0，并将每行的survived值相加，则得到幸存总人数。（代码见附件titanic\_hbase.java）

**static** **int** get\_row\_survived(String table\_name,String row\_key,String col\_family,String col)

**throws** IOException

{

Table table = *connection*.getTable(TableName.*valueOf*(table\_name));

Get get = **new** Get(row\_key.getBytes());

get.addColumn(col\_family.getBytes(),col.getBytes());

Result result = table.get(get);

Cell[] cells = result.rawCells();

**int** if\_survived=0;

**for**(Cell cell:cells)

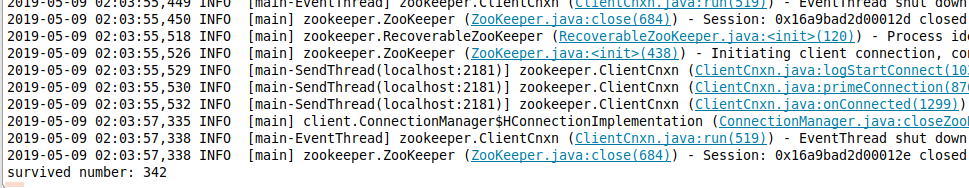
{

String survived=**new** String(CellUtil.*cloneValue*(cell));

if\_survived=Integer.*valueOf*(survived);

}

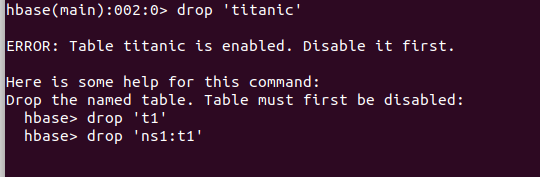
**return** if\_survived;



## 心得

由于之前的hadoop为全分布式版本，克隆原来的虚拟机后修改配置文件为伪分布式并重新格式化，运行时无法启动datanode，这是因为之前的格式化存储了一个VERSION文件，其中包括首次格式化时的datanode的id，而当前的id与其不匹配。所以需要删除tmp文件夹下的所有文件并重新格式化，这样新生成的VERSION文件中的datanode的id与当前id匹配，datanode成功启动

删除某个表需要先disable



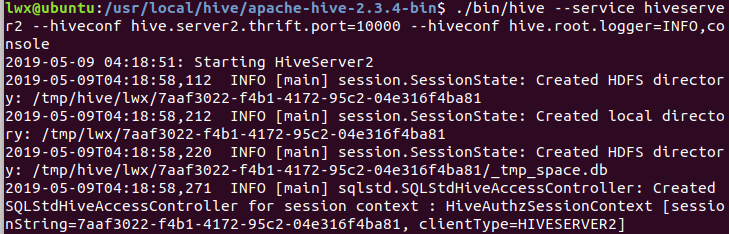
由于ImportTsv通过逗号分隔符来分割csv的一行，人名这一列中存在的逗号将导致匹配错误，显示所有行都是bad line。可以将人名这一列中的逗号预处理为别的符号，本题中这一列并不重要，所以将其删去。

由于hbase shell的增删改查语句很少，只能按行操作，所以查询幸存总人数时使用java api通过自定义的函数搜索数据库。

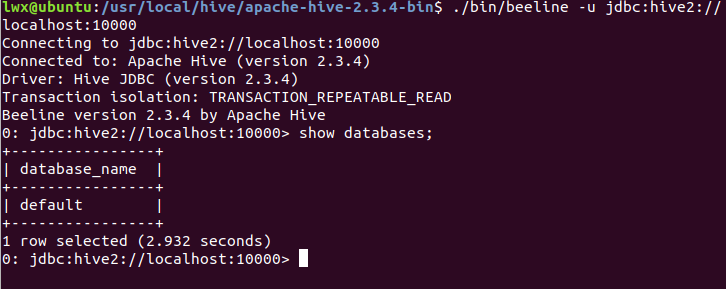
# HIVE

## 实验过程

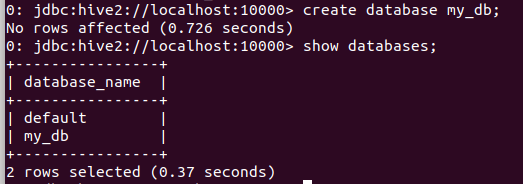
启动HIVE



连接hive并查看所有数据库



创建数据库my\_db



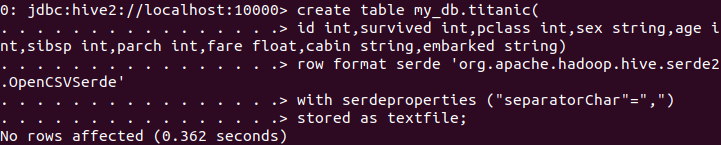
切换到my\_db



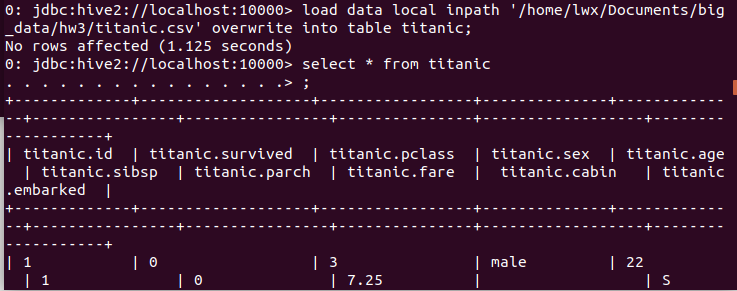
创建一个表test\_table，查看表结构信息



在数据库my\_db中创建表titanic,定义csv格式，分隔符为逗号



从本地文件夹中导入csv到hive的titanic表中



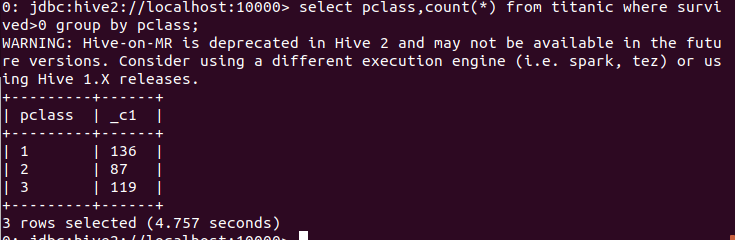
### 查询1

查看存活乘客及其对应的舱位等级



### 查询2

查看一等舱，二等舱，三等舱分别的幸存者总数



## 心得

修改hadoop的core-site.xml时，需要将用户名写入property。但是连接hive数据库时显示没有权限。需要重启hadoop。

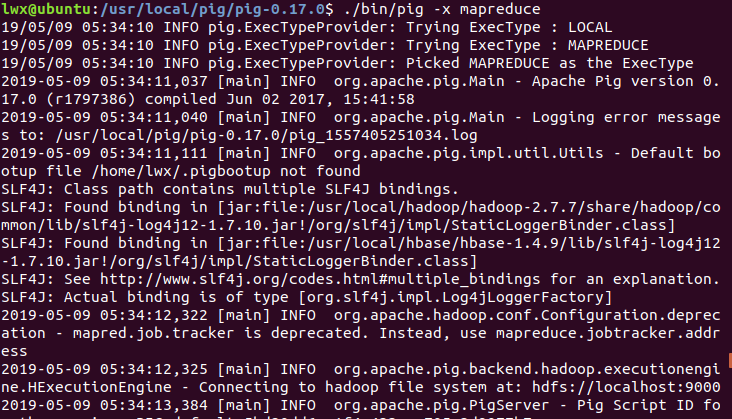
每个命令需要分号结尾

为了导入csv文件到表中，创建表时需要指定表的格式为csv和表的分隔符为逗号

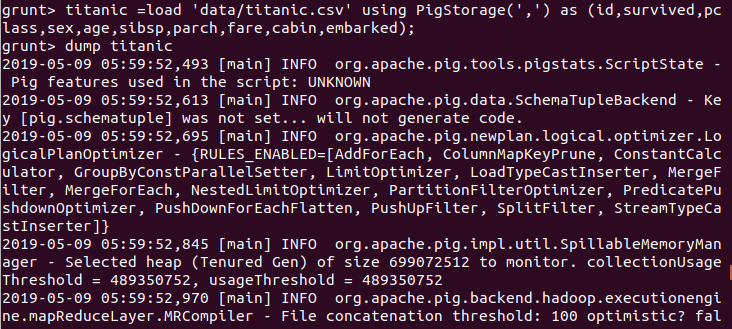
# Pig

## 实验过程

启动pig

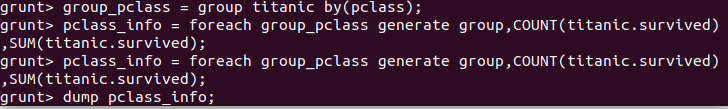


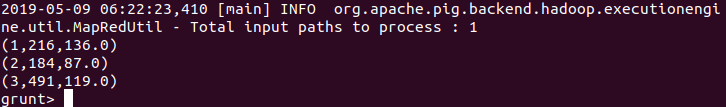
导入titanic



### 查询1

查询每个舱位等级对应的乘客数目和幸存者数目





可见舱位等级越高，存活率越高

### 查询2

查询幸存者和遇难者分别的船票平均值





可见幸存者的票价平均值远高于遇难者，活下来的多是富人。

## 心得

pig是流处理语言，每步一个操作，产生一个新的数据集。

所以pig在实现复杂操作的同时，每一步的命令比较简洁，多步命令结合达到结果。

# 附件：titanic\_hbase.java

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.TableName;

import org.apache.hadoop.hbase.client.HBaseAdmin;

import org.apache.hadoop.hbase.client.HTablePool;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.NavigableMap;

import java.util.Random;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.Cell;

import org.apache.hadoop.hbase.CellScanner;

import org.apache.hadoop.hbase.CellUtil;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.HColumnDescriptor;

import org.apache.hadoop.hbase.HTableDescriptor;

import org.apache.hadoop.hbase.TableName;

import org.apache.hadoop.hbase.client.Admin;

import org.apache.hadoop.hbase.client.Connection;

import org.apache.hadoop.hbase.client.ConnectionFactory;

import org.apache.hadoop.hbase.client.Get;

import org.apache.hadoop.hbase.client.HBaseAdmin;

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

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

import org.apache.hadoop.hbase.client.ResultScanner;

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

import org.apache.hadoop.hbase.client.Table;

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

public class titanic\_hbase

{

    static Configuration configuration;

static Connection connection;

static Admin admin;

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();

}

}

//关闭连接

public static void close()

{

try

{

if(admin != null)

{

admin.close();

}

if(null != connection)

{

connection.close();

}

}catch (IOException e)

{

e.printStackTrace();

}

}

    static void create\_table(String table\_name,String col\_family\_name)throws Exception

    {

     // 配置

Configuration conf = HBaseConfiguration.create();

conf.set("hbase.zookeeper.quorum", "localhost");

Connection connection = ConnectionFactory.createConnection(conf);

// 管理员

Admin hbaseAdmin = connection.getAdmin();

// 表名称

TableName tableName = TableName.valueOf(table\_name);

// 表描述器

HTableDescriptor tableDesc = new HTableDescriptor(tableName);

tableDesc.addFamily(new HColumnDescriptor(col\_family\_name));// 添加列族

// 创建一个表，同步操作

hbaseAdmin.createTable(tableDesc);

System.out.println("创建表" + table\_name + "成功");

    }

     public static void query\_row(String tableNameString,String rowNameString) throws IOException

     {

System.out.println("---------------按行建查询表中数据--------");

Configuration conf = HBaseConfiguration.create();

conf.set("hbase.zookeeper.quorum", "localhost");

Connection connection = ConnectionFactory.createConnection(conf);

//取得数据表对象

Table table=connection.getTable(TableName.valueOf(tableNameString));

//新建一个查询对象作为查询条件

Get get = new Get(rowNameString.getBytes());

//按行查询数据

Result result = table.get(get);

byte[] row = result.getRow();

System.out.println("row key is:" +new String(row));

List<Cell> listCells = result.listCells();

for (Cell cell : listCells)

{

byte[] familyArray = cell.getFamilyArray();

byte[] qualifierArray = cell.getQualifierArray();

byte[] valueArray = cell.getValueArray();

System.out.println("row value is:"+ new String(familyArray) +

new String(qualifierArray) + new String(valueArray));

}

System.out.println("---------------查行键数据结束----------");

     }

     public static void showCell(Result result)

     {

     Cell[] cells = result.rawCells();

     for(Cell cell:cells)

     {

     System.out.println("RowName:"+new String(CellUtil.cloneRow(cell))+" ");

     System.out.println("Timetamp:"+cell.getTimestamp()+" ");

     System.out.println("column Family:"+new String(CellUtil.cloneFamily(cell))+" ");

     System.out.println("row Name:"+new String(CellUtil.cloneQualifier(cell))+" ");

     System.out.println("value:"+new String(CellUtil.cloneValue(cell))+" ");

     }

     }

     static void getData(String tableName,String rowKey,String colFamily,String col)throws IOException

     {

     init();

     Table table = connection.getTable(TableName.valueOf(tableName));

     Get get = new Get(rowKey.getBytes());

     get.addColumn(colFamily.getBytes(),col.getBytes());

     Result result = table.get(get);

     showCell(result);

     table.close();

     close();

     }

     static int get\_row\_survived(String table\_name,String row\_key,String col\_family,String col)

             throws IOException

     {

     Table table = connection.getTable(TableName.valueOf(table\_name));

     Get get = new Get(row\_key.getBytes());

     get.addColumn(col\_family.getBytes(),col.getBytes());

     Result result = table.get(get);

     Cell[] cells = result.rawCells();

     int if\_survived=0;

     for(Cell cell:cells)

     {

//   System.out.println("value:"+new String(CellUtil.cloneValue(cell))+" ");

         String survived=new String(CellUtil.cloneValue(cell));

         if\_survived=Integer.valueOf(survived);

     }

     return if\_survived;

     }

     static int count\_survived(String table\_name)

     {

         init();

         int survived\_cnt=0;

         try

         {

             int row\_num=891;

             String col\_family="info";

             String col="survived";

             for(int row\_id=1;row\_id<=row\_num;++row\_id)

             {

                 String row\_key=String.valueOf(row\_id);

                 int tmp\_survived=get\_row\_survived(table\_name,row\_key,col\_family,col);

                 survived\_cnt+=tmp\_survived;

             }

         }

         catch(Exception e)

         {

             e.printStackTrace();

         }

         close();

         return survived\_cnt;

     }

public static void main(String[] agrs) throws Exception

{

    getData("titanic","1","info","age");

    int survived\_num=count\_survived("titanic");

    System.out.printf("survived number: %d",survived\_num);

}

}