XLSX2SQLite实验报告

1 程序功能

该程序为命令行程序，用于读取特定`xlsx`文件，找到指定的页，然后在本地的`SQLite`数据库中建立指定名称的表。

- `xlsx`里的第一行作为表中的字段名。
- 字段类型由`xlsx`数据推断得到。
  - 整数的为`int`类型。
  - 浮点数位`real`类型。
  - 其他的为`char`类型，大小为所有行中最大的长度。
- 程序为该表建立一个`PK`,以表达行号。程序输出表的结构和行的数量。

程序的使用方法：

- 进入文档对应目录`3160104463_刘家豪_XLSX2SQLite`.
- 在当前目录下命令行操作`java -jar X2S.jar para1, para2, para3, para4`.
- 其中参数依次为
  - 数据库名
  - `xlsx`文件名
  - `Excel`中的页名，当该参数省略时，取第一个页。
  - 数据库表名，当这个参数省略时，用页名或当第三个参数也省略时用`xlxs`文件名前缀作为表名。

2 实验原理

这个程序的实现过程可分为三个部分：

- `Excel`文件的读取
- `Table`的建立
- `Data`的插入

2.1 Excel文件的读取

对于`Excel`文件的读取，采用`poi`实现。

考虑到需要根据`Excel`文件中数据完成`SQLite`数据库中表的数据类型的推断，以及进行`Excel`文件数据读取，花费时间较多。故决定采用一次读取的方式。

在读取`Excel`文件时，返回`List<String[]>`,同时，为了便于实现数据推断，在返回值的第二处添加额外的`String[]`用来记录，每个属性所对应的数据类型是`NUMERIC`或是其他。在后期处理时，我们可以根据`.`来判断`NUMERIC`下的是`INT`还是`REAL`.对于`CHAR`，通过查询整个`List<String[]>`获取长度。

2.1.1 页的查找

```
if(tableName.length() != 0) {
    sheet = workbook.getSheet(tableName);
    if(sheet == null) {
        throw new Exception("该Excel文件中无此页");
    }
}else {
    sheet = workbook.getSheetAt(0);
}
```

当输入对应页名，但无法查找到对应页，抛出异常。当无页名输入，采用默认页，即第一页。

2.1.2 属性名及其类型的获取

```
for(int i = firstCellNum; i < lastCellNum; i++) {
    Cell attrCell = attrTag.getCell(i);
    attr[i] = getCellValue(attrCell);
    Cell typeCell = typetag.getCell(i);
    switch(typeCell.getCellType()) {
    case NUMERIC:
        type[i] = "NUMERIC";
        break;
    default:
        type[i] = "CHAR";
    }
}
```

遍历第一行及第二行获取对应的属性名以及大致的类型范围，为后续对属性类型推测做准备。

2.1.3 数据项的获取

```
for(int i = firstRowNum + 1; i <= lastRowNum; i++) {
    Row row = sheet.getRow(i);
    String [] record = new String[attrTag.getPhysicalNumberOfCells()];
    for(int k =  firstCellNum; k < lastCellNum; k++) {
        Cell cell = row.getCell(k);
        record[k] = getCellValue(cell);
    }
    content.add(record);
}
```

依次遍历每行的每列，且将对应数据转换为字符串，进行存储，以便返回。其中最为重要的是，将整数转回为字符串返回，若是采用`NUMERIC`类型直接返回，会出现`int`变成`double`的现象。通过转换成字符串，然后根据是否含有`.`来判断类型是`int`还是`real`

2.2 数据库表的操作

2.2.1 数据库连接

采用`JDBC`对数据库进行连接。

```
public static Connection createConnection(String db) throws SQLException,
ClassNotFoundException{
    Class.forName(Class_Name);
    return DriverManager.getConnection(db);
}
```

2.2.2 数据库表建立

通过对`List<String[]>`的解析获取不同属性的数据类型。

```
for(int i = 0; i < column; i++) {
    if(attr.get(1)[i].equals("NUMERIC")) {
        if(attr.get(2)[i].contains(".")) {
            type[i] = "REAL";
        }else {
            type[i] = "INT";
        }
    }
    else if(attr.get(1)[i].equals("CHAR")) {
        int length = 0;
        for(int k = 2; k < attr.size(); k++) {
            if(attr.get(k)[i].length() > length) {
                length = attr.get(k)[i].length();
            }
```

```
        }
        String tmp = "CHAR(" + length + ")";
                    type[i] = tmp;
    }
}
```

通过判断字符串中是否含有`.`来判断`NUMERIC`的具体类型；若类型是`CHAR`则遍历整列，找到最长的取为`CHAR`的长度。

```
String deletesql = "DROP TABLE IF EXISTS " + tableName + ";";
String createsql = "CREATE TABLE IF NOT EXISTS " + tableName + " " +
                   "(ID INT PRIMARY KEY NOT NULL," +
                    table +");";
stmt.executeUpdate(deletesql);
stmt.executeUpdate(createsql);
```

分别定义删除已存在表和创建对应表的`sql`语句，然后，执行该语句，完成数据库表的建立。

2.2.3 数据的插入

```
conn.setAutoCommit(false);
```

为了提高数据插入的速度，关闭了自动提交。与此同时，采用批处理的方式，进一步提高插入速度。

```
for(int i = 2; i < attr.size(); i++) {
    ps.setInt(1, i - 1);
    for(int j = 0; j < attr.get(i).length; j++) {
        if(type[j].equals("INT")) {
            ps.setInt(j + 2, Integer.parseInt(attr.get(i)[j]));
        }else if(type[j].equals("REAL")) {
            ps.setFloat(j + 2, Float.parseFloat(attr.get(i)[j]));
        }else {
            ps.setString(j + 2, attr.get(i)[j]);
        }
    }
    ps.addBatch();
    if(++cnt % bitchSize == 0) {
        ps.executeBatch();
        conn.commit();
    }
}
```

同时为了提高安全性，采用`prepareStatement()`的方式，进行提交。

2.3 命令行参数

鉴于命令行参数可能存在缺省的情况，故在读取相应参数时，进行多重判断以达到相应效果。

```
if(args.length < 2) {
    try {
        throw new Exception("参数输入过少");
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
String db_url = args[0];
String excelFile = args[1];
String sheetName = "";
String tableName = "";
if(args.length == 2) {
    tableName = excelFile.substring(0, excelFile.indexOf("."));
}
else if(args.length == 3) {
    sheetName = args[2];
    tableName = args[2];
}else if(args.length == 4) {
    sheetName = args[2];
    tableName = args[3];
}
```

3 实验结果

省略两个参数:

```
C:\Users\liuji\Desktop\3160104463_刘家豪_XLSX2SQLite>java -jar X2S.jar X2S.db test.xlsx
TableName:      test
ID                      INT
姓名                    CHAR(3)
学号                    INT
校区                    CHAR(3)
The number of Line is:  14400
```

省略一个参数:

```
C:\Users\liuji\Desktop\3160104463_刘家豪_XLSX2SQLite>java -jar X2S.jar X2S.db test.xlsx Exp
TableName:      Exp
ID                      INT
涡轮                    REAL
电磁                    REAL
流量                    REAL
转子                    INT
压差                    INT
标签                    CHAR(15)
The number of Line is:  17100
```

不省略参数:

```
C:\Users\liuji\Desktop\3160104463_刘家豪_XLSX2SQLite>java -jar X2S.jar X2S.db test.xlsx Student stu
TableName:      stu
ID                      INT
姓名                    CHAR(4)
性别                    CHAR(1)
省份                    CHAR(8)
类型                    CHAR(4)
班级                    CHAR(7)
The number of Line is:  339
```

创建表的语句:

```
sqlite> .tables
Exp   stu   test
sqlite> .schema Exp
CREATE TABLE Exp (ID INT PRIMARY KEY    NOT NULL,涡轮    REAL, 电磁      REAL, 流量      REAL, 转子  INT, 压差    INT, 标签      CHAR(15));
sqlite> .schema stu
CREATE TABLE stu (ID INT PRIMARY KEY    NOT NULL,姓名    CHAR(4), 性别   CHAR(1), 省份   CHAR(8), 类型    CHAR(4), 班级
CHAR(7));
sqlite> .schema test
CREATE TABLE test (ID INT PRIMARY KEY   NOT NULL,姓名    CHAR(3), 学号   INT, 校区      CHAR(3));
```

查询部分内容:

```
sqlite> .excel
sqlite> select * from Exp
   ...> where ID > 10 and ID < 50;
```

1	11	0.61	0.591	9.82	590	10	establish
2	12	1.51	1.537	49.47	1500	54	authority
3	13	2.54	2.603	50.97	2500	87	major
4	14	3.54	3.634	50.91	3500	138	issue
5	15	4.53	4.634	98.06	4500	179	labour
6	16	0.61	0.591	9.82	590	10	occur
7	17	1.51	1.537	49.47	1500	54	economic
8	18	2.54	2.603	50.97	2500	87	involve
9	19	3.54	3.634	50.91	3500	138	percent
10	20	4.53	4.634	98.06	4500	179	interpretation
11	21	0.61	0.591	9.82	590	10	consistent
12	22	1.51	1.537	49.47	1500	54	income
13	23	2.54	2.603	50.97	2500	87	structure
14	24	3.54	3.634	50.91	3500	138	legal
15	25	4.53	4.634	98.06	4500	179	concept
16	26	0.61	0.591	9.82	590	10	formula
17	27	1.51	1.537	49.47	1500	54	section