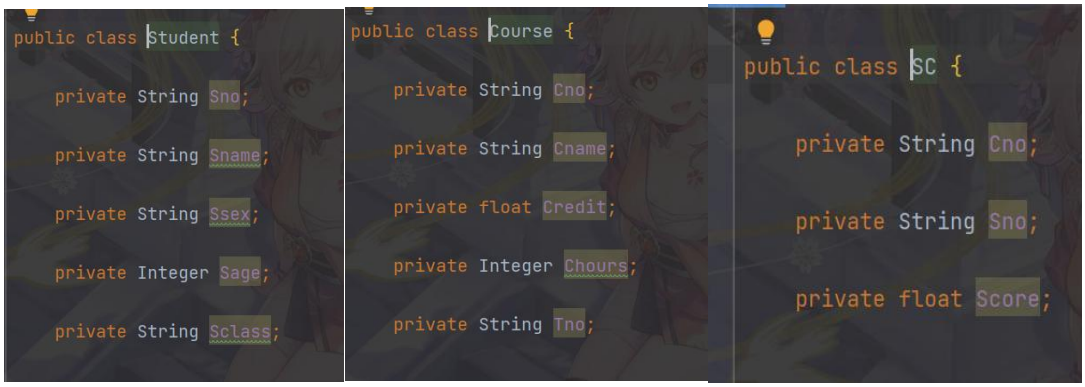
数据库系统实验一

1、实验环境

操作系统：Windows10
编程语言：Java
编译器：jdk1.8
开发环境：Intellij IDEA
数据库：mysql 8.0.32
依赖环境：mysql-connector-java-8.0.32.jar

2、数据结构

定义了三个类 Student、Course、SC 来存储每条记录信息，每个类对应相应表中的一条记录，类中属性对应表中属性。



用三个列表（ArrayList）StudentList、CourseList、SCList 分别存储三个表 Student、Course、SC 的数据，之后将其装入 sct 数据库中的三个表里。

3、设计思路

3.1 生成数据

这里首先定义了一个 JDBCUtils 工具类来连接 mysql 数据库。

然后定义了一个 Data 类

```
    0 个用法
    public  void generateData() {...}
    //将生成的数据插入到数据库中
    0 个用法
    public  void InitDatabase(Connection con) throws SQLException {...}
    //删除所有表中数据
    0 个用法
    public  void deleteData(Connection con) throws SQLException {...}
```

其中 generateData() 方法用来生成数据，然后定义了一个 InitDatabase(Connection con)方法来将生成的数据装入 sct 数据库里的 Student、Course、SC 三个表。生成数据中的 Sname 和 Cname 主要根据的是从网上找到的两个数据集 Chinese_Names.txt 和 courses_zh.txt,里面有足够多的名字。我先将他们读入列表中，然后生成随机数作为下标取其中的名字分别作为 Sname 和 Cname 属性的值。其他字段基本就是用随机数生成的，比较简单，不做赘述。（提前在 mysql 命令行里用 create 语句创建好 sct 数据库和三个表，然后执行 main 函数里的注释代码就可以）。

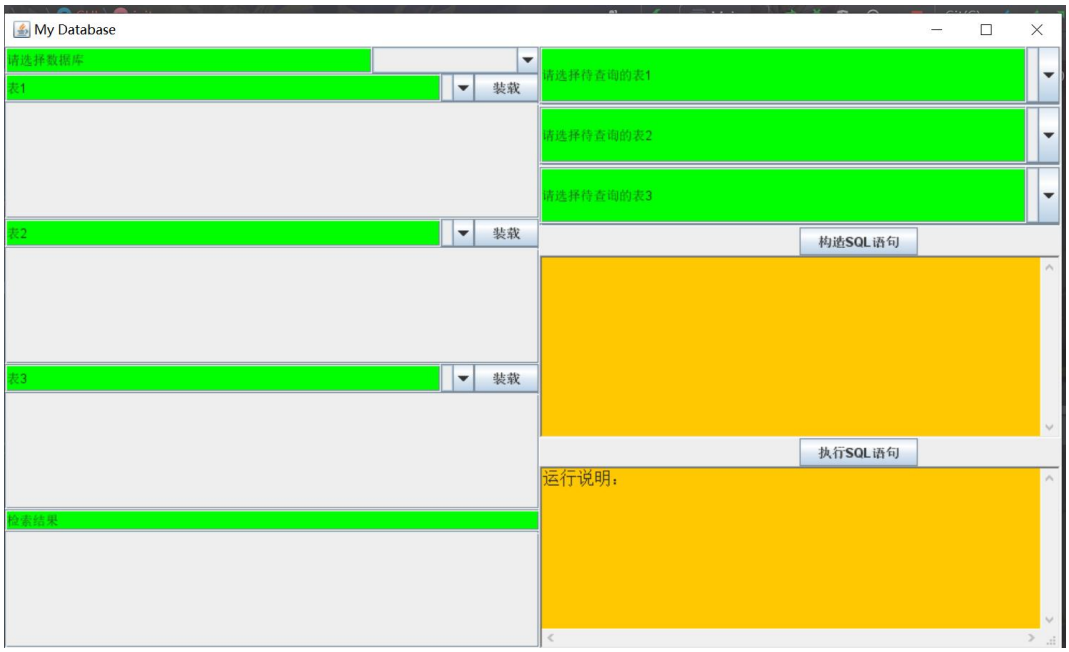
然后 InitDatabase（Connection con）方法中执行 3 条插入语句用来将生成的数据插入数据库中

```
String student_insert_sql = "insert into Student values (?,?,?,?,?)";
String course_insert_sql = "insert into Course values (?,?,?,?,?)";
String sc_insert_sql = "insert into SC values (?,?,?)";
```

deleteData(Connection con)方法用来删除插入数据库三个表里的的所有数据，这里我仅用来测试，之后并未用到。

3.2 GUI 界面编写

基本思路是采用 awt 框架里的 box 容器来装载各个组件，选择用 box 容器是为了方便布局，因为 box 可以选择水平布局和垂直布局，最后的 GUI 界面如下图所示



这个界面的基本思路是一个框架 jframe 里放置一个一个水平布局的大 box，这个大 box 里面再放两个垂直布局的 boxLeft 和 boxRight。

```
Box box = Box.createHorizontalBox();
box.add(boxLeft);
box.add(boxRight);
jframe.add(box);
```

然后编程任务就可以分为左半界面和右半界面的编程。左半界面，主要是用于三个表的装载以及展示检索结果。右半界面，则用于根据所选的表和所选的字段构造 SQL 语句，执行 SQL 语句并展示查询结果，输出相应的运行状态说明信息。

主要的思路就是不断增加组件，然后将组件放到框架里最终达到上图的显示结果。然后为了实现选做功能的可对任何表/任何字段进行检索条件表达的用户交互查询应用程序，我们需要对每一个下拉框和按钮 button 增加鼠标事件监听，当鼠标点击后执行相应的事件监听函数并将结果输出到 GUI 界面上显示。具体鼠标点击后显示效果可见下面第 4 部分运行结果。

3.3 主要算法

总体可以两类，构造 sql 语句算法和执行 sql 语句算法（左半界面的装载其实也是执行 sql 语句），也即检索算法 。

首先是构造 sql 算法，在 makeSQL 按钮的监听事件中实现。

```
makeSQL.addActionListener(new ActionListener() {
```

先创建 4 个 list，分别为 tableNameList、columnNameList、selectedColumnNameList、conditionList。其中 tableNameList 存储选中的三个表的表名，columnNameList 存储所有的表名+列名字段，selectedColumnNameList 存储被选中的表名+列名，conditionList 用来存储文本框里输入的条件。

```
ArrayList<String> tableNameList = new ArrayList<>();
ArrayList<String> columnNameList = new ArrayList<>();//存储所有的表名+列名字段
ArrayList<String> selectedColumnNameList = new ArrayList<>();//存储被选中的表名+列名
ArrayList<String> conditionList = new ArrayList<>();
```

基本构造思路就是将一条 sql 语句拆分为 3 部分，第一部分为 select+str0，第二部分为 from+str1，第三部分为 where+str2，这三部分的具体构造需要依靠上面的 4 个 list。

第一部分这里由于考虑多表联合查询，所有表名可能重复，所以将三个表名重命名为 t1，t2，t3。相应的表名+列名字段则类似为”t1.Sname”。然后如果一个表里属性对应的复选框全不勾选，则默认为 select*，否则为 select 表名+列名字段，例如 select t1.Sname，不同的表名和列名字段中间还需要注意增加”,”。

第二部分比较简单，为 from tableName1 as t1，只需要根据选择的表来生成即可，如果选择了多个表名中间还需要增加”,”即可。

第三部分需要判断文本框里是否有条件，如果都为空，那么第三部分也为空，否则需要根据文本框里的条件生成相应的字符串，这里选择 like+字符串的格式

来生成。

根据三部分内容生成最终的 SQL 语句详见下图。

```
if (str0.toString().equals("")) {
    //一个都不勾默认select *
    if (str2.toString().equals("")) {
        String SQL = "select * " + str0 + "\n" + "from " + str1 + ";";
        sqlText.setText(SQL);
    } else {
        String SQL = "select * " + str0 + "\n" + "from " + str1 + "\n" + "where " + str2 + ";";
        sqlText.setText(SQL);
    }
} else {
    if (str2.toString().equals("")) {
        String SQL = "select  " + str0 + "\n" + "from " + str1 + ";";
        sqlText.setText(SQL);
    } else {
        String SQL = "select  " + str0 + "\n" + "from " + str1 + "\n" + "where " + str2 + ";";
        sqlText.setText(SQL);
    }
}
```

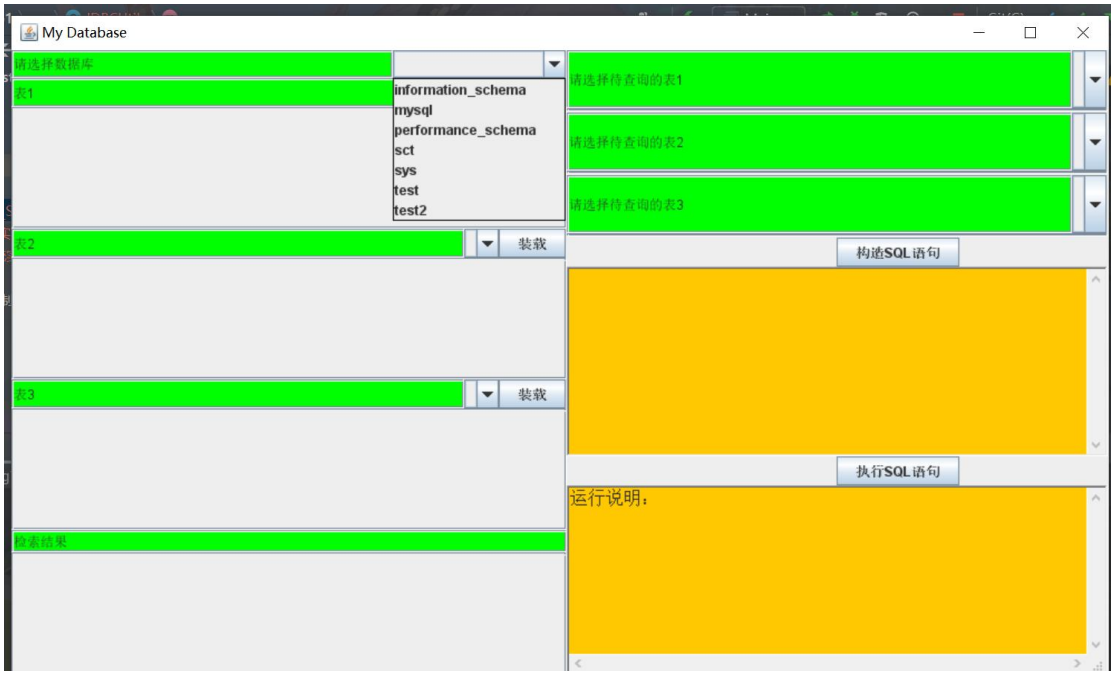
然后是执行 SQL 语句，只需要传入 sql 语句，然后 jdbc 里在选择的数据库执行相应的 sql 语句返回一个存储了属性名和查询结果的二维列表，最后再将二维列表的内容加载到 GUI 界面显示即可。具体实现见下图。

```
public ArrayList<ArrayList<Object>> query(String dbName,String sqlString) throws SQLException {
    ArrayList<ArrayList<Object>> list = new ArrayList<>();
    Connection connection = DriverManager.getConnection( url: URL + dbName, USER, PASSWORD);
    PreparedStatement pr2 = connection.prepareStatement(sqlString);
    ResultSet rs = pr2.executeQuery();
    ResultSetMetaData rs_metaData = rs.getMetaData();
    ArrayList<Object> temp = new ArrayList<>();
    int count = rs_metaData.getColumnCount();
    //二维列表第0行用来存储表头（即列名）
    for (int i = 1; i <= count; i++) {
        temp.add(rs_metaData.getColumnName(i));
    }
    list.add(temp);
    while (rs.next()) {
        ArrayList<Object> temp2 = new ArrayList<>();
        for (int i = 1; i <= count; i++) {
            temp2.add(rs.getObject(i));
        }
        list.add(temp2);
    }
    rs.close();
    pr2.close();
    connection.close();
    return list;
}
```

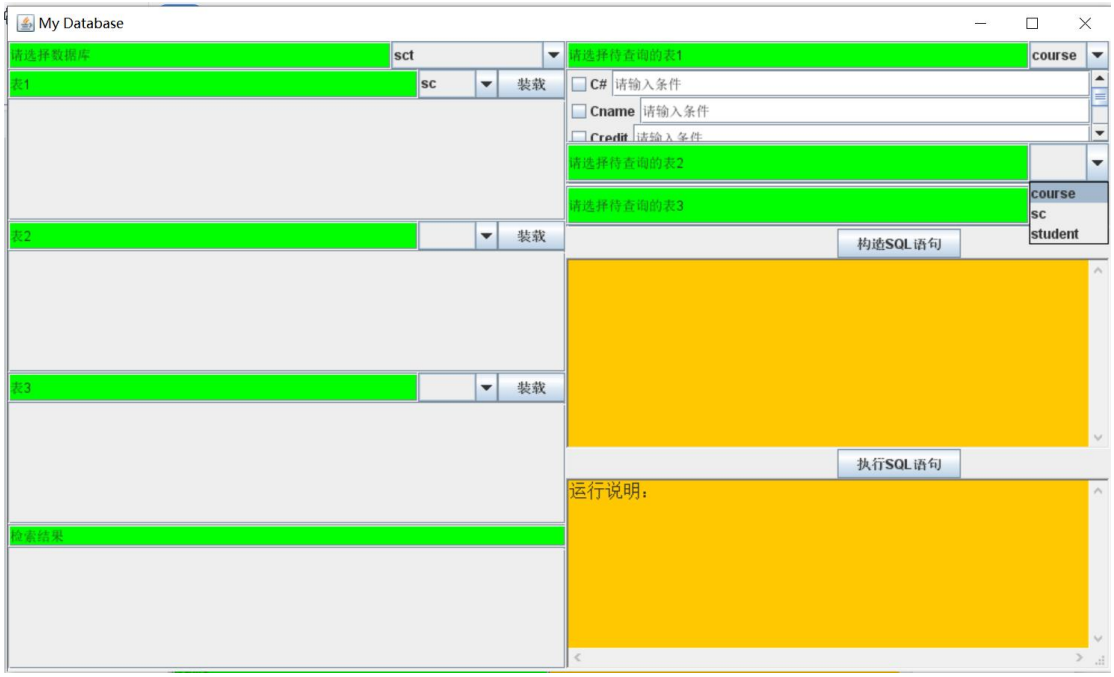
字体大小: 12pt　　重置为 13pt

4、运行结果

可选数据库界面如下

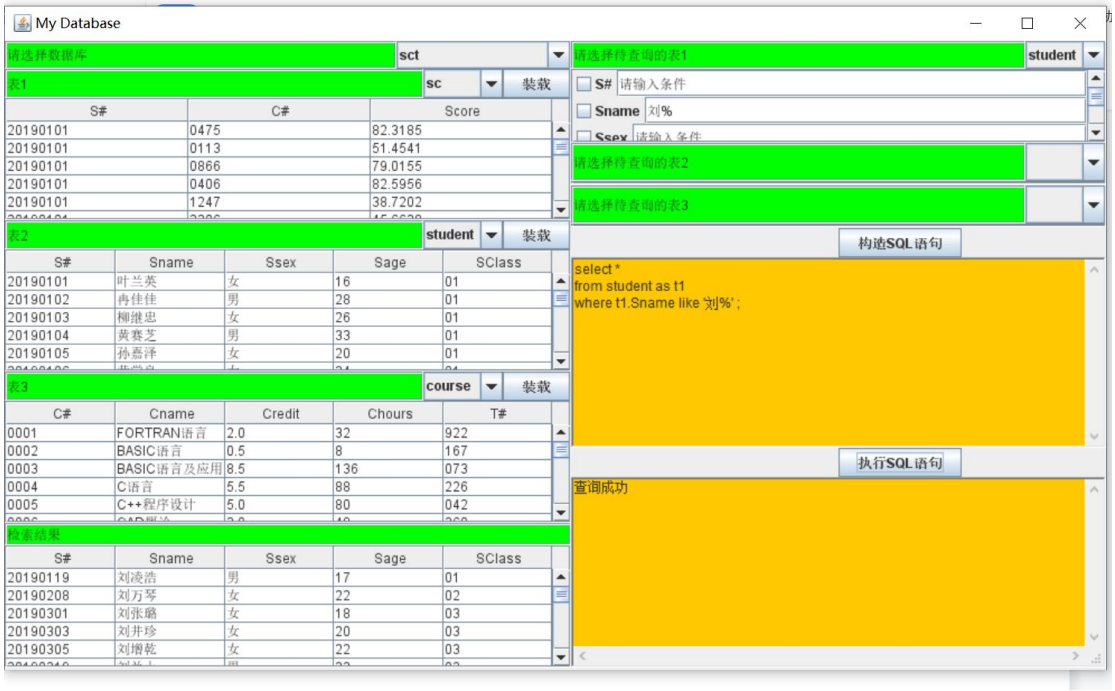


选择数据库后会加载出相应的待选表信息，这里选择 sct 数据库示例。之后在选中表后会加载出相应的待选表的信息。

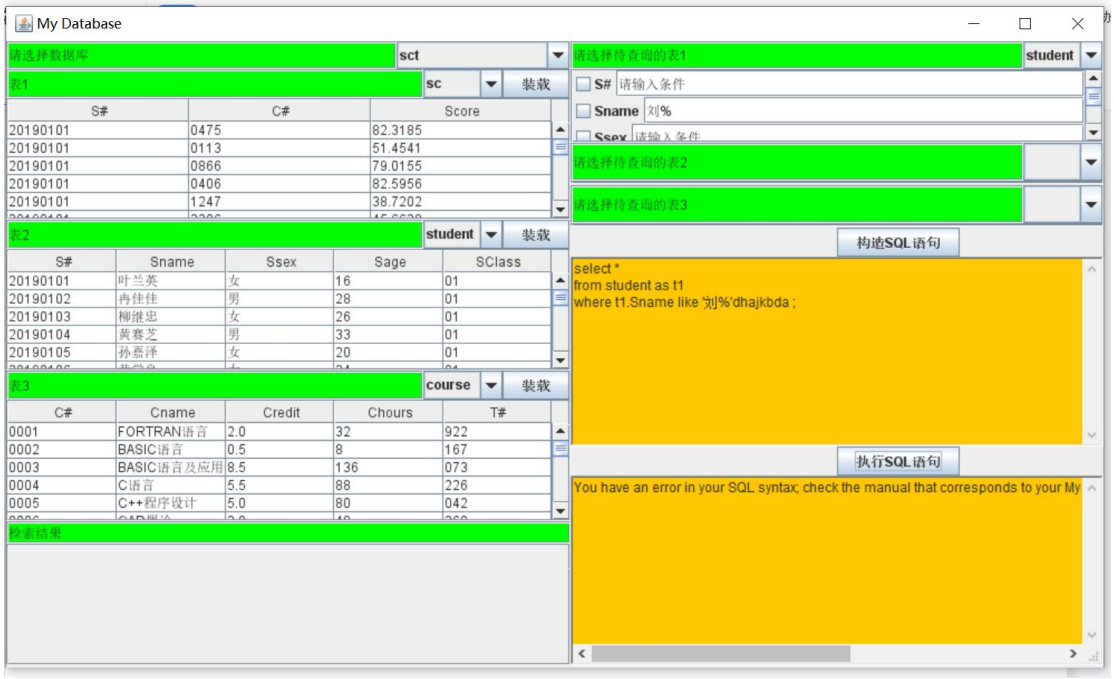


点击左边的三个装载按钮后会将三个选择的三个表的全部内容加载出来。

然后是根据选择的表中属性字段构造 SQL 语句，然后点击执行 SQL 语句按钮，将查询结果装载到界面，并且输出相应的运行消息。



如果运行报错，则给出相应的错误信息



5、实验感悟

总体来说，这次实验学到了很多知识，比如学会了怎么编写 SQL 语句，然后用 jdbc 来执行 sql 语句操作 mysql 数据库，学会了怎么用 awt 和 swing 框架来编写一个相对美观的前端界面。当然实验中也遇到了不少问题，比如界面大小比例失调，某些组件内容无法加载进来，sql 语句执行结果无法加载到界面里，如何根据选择的数据库动态生成相应的带选择的表等等问题，尤其是第一次实验只实现前面要求的功能代码写的实在太烂了，为了实现选做部分功能我几乎将所有的代码重写了一遍，并且所有的组件及相应的监听事件都重新写了一遍，确实花

费了不少的时间。但是好在功夫不负有心人，在查找了诸多资料以及自己的多次尝试下最终成功实现了全部的功能。虽然还有很大的改进空间，比如说界面可以更美观，构造的 sql 语句可以更加多样化等等。但是，总而言之，这个实验，收获满满，成就感满满。