

合肥工业大学

计算机与信息学院

《软件工程师综合训练》报告

选题题目： 某高校科研管理系统

学号姓名： 2018211958 孙淼

专业班级： 计算机科学与技术 18-2 班

指导老师： 李心科

2021 年 10 月 10 日

一、概述

1.1 课题背景

某高校科研管理系统，基本功能要求：

- (1) 实现部门、职务、职称等基本信息的管理；
- (2) 实现教师信息的管理；
- (3) 实现可以科研项目的申报、审批管理；
- (4) 实现科研项目的验收管理；
- (5) 创建默认，并绑定到科研项目的验收标志，使其默认值为“未验收”；
- (6) 创建触发器，验收项目时自动修改项目的验收标志为“验收通过”；
- (7) 创建存储过程统计个院系科研项目的申报和完成数量；
- (8) 具有数据备份和数据恢复功能。

1.2 设计任务与要求

- ①通过调查研究和运用 Internet，收集和调查有关资料、最新技术信息。
- ②理解和掌握数据库系统的需求分析过程和基本方法
- ③基本掌握撰写设计报告的基本步骤和写作方法。
- ④根据课题的要求基本理解和掌握 E-R 图的设计方法和关系模式的转换。
- ⑤根据 ER 图生成数据库表，进行数据库的逻辑设计。
- ⑥数据库完整性、安全性保障措施。
- ⑦数据库的物理设计。
- ⑧数据库实施维护计划。
- ⑨应用系统的设计和实现。

1.3 开发环境与工具

数据库平台：MS SQL Server

开发平台：J2EE Eclipse

软件架构：C/S

二、需求分析

随着信息化管理在高校的推行,各高校越来越重视对科研信息的管理,但由于各高校对科研工作量化的差异,使得在科研管理上出现各自为政的尴尬局面.

因此,根据高校近年来科研管理的实际情况,按照学校实际需求,开发一个基于C/S模式的科研管理系统,使学校的科研工作管理能够方便、规范、快捷地进行。高校科研范围涉及各级各类课题、论文、教材、学术交流等。科研管理系统的实现除了满足学院对所有科研人员的科研成果进行查询、汇总、分析之外,也对院领导的决策提供依据。

2.1 用户需求分析

系统的用户包括有一般科研人员(教师)、院系科研管理人员。

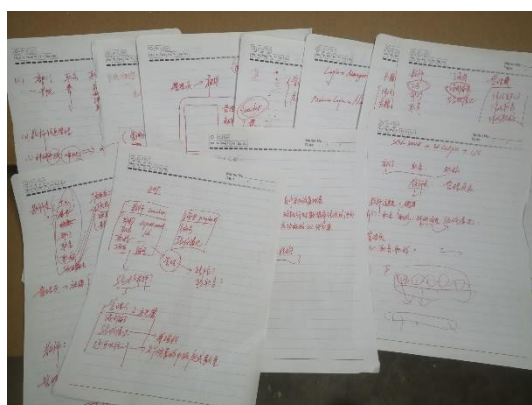
一般科研人员需求:一般科研人员登陆后,希望系统能够提供管理自己的科研成果(包括论文、著作等)科研项目和科研经费使用的平台;希望能够查看自己或他人的科研成果信息。其中,个人科研成果数据需提交院系科研秘书审核通过后才能成为有效数据。

院系科研管理人员需求:科研管理人员或科研秘书希望系统能够接收一般科研人员的成果信息和项目申报信息等,科研秘书能够对成果信息和项目申报信息的真实性和准确性进行审核;希望系统能够进行学术交流信息的管理和实验室管理;希望系统能够查询统计所有科研信息并生成相关报表。

2.2 功能需求分析

系统的开发是为了更加高效地管理科研信息,使得整个科研管理工作流程更加规范。所开发的系统既要方便学校科学技术处的监督和管理,又要有利于学院领导或学院内部教师之间的协作交流,从学院的实际情况出发,把现有的科研基础数据进行整合。根据学院可能的用户需求。

这部分,在后续的介绍中我会伴随着模块代码进行介绍。为了后续软件设计的流畅,我们花费了一天时间高强度调研和分析目前的现有科研管理系统的特征,迭代了很多版本,如下图所示:



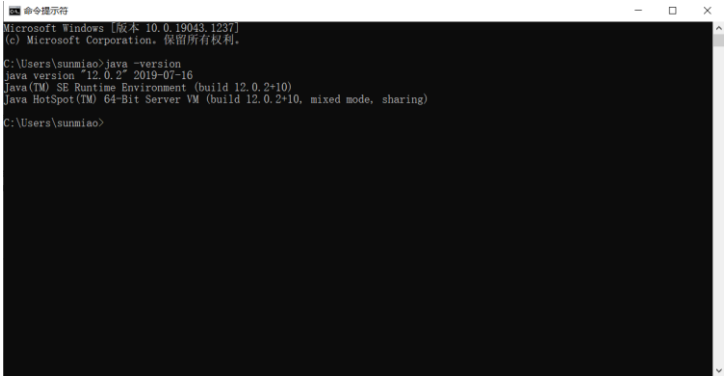
三、数据库设计

要设计数据库，首先需要完成数据库与 JDBC 的连接，这一部分也是不错的探索，记载如下：

预备工作：

要将 SQL server 与 JDK 环境连接，首先需要，配置电脑环境：

按照 SQL server 的要求，需要安装 JDBC 驱动系统，cmd 可以看到我的 JRE 是 12；



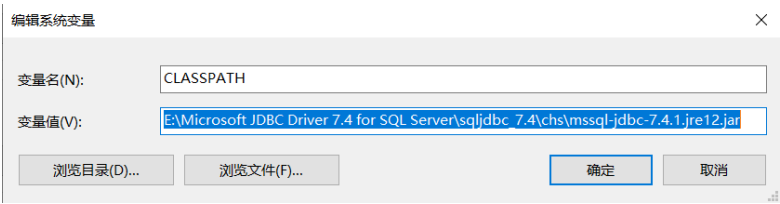
那么根据 JDBC 的 docs 的要求，我的 Java Runtime Environment 应该是自 Microsoft JDBC Driver 7.4 for SQL Server

自 Microsoft JDBC Driver 7.4 for SQL Server 起，支持 Java 开发工具包 (JDK) 12.0 和 Java 运行时环境 (JRE) 12.0。

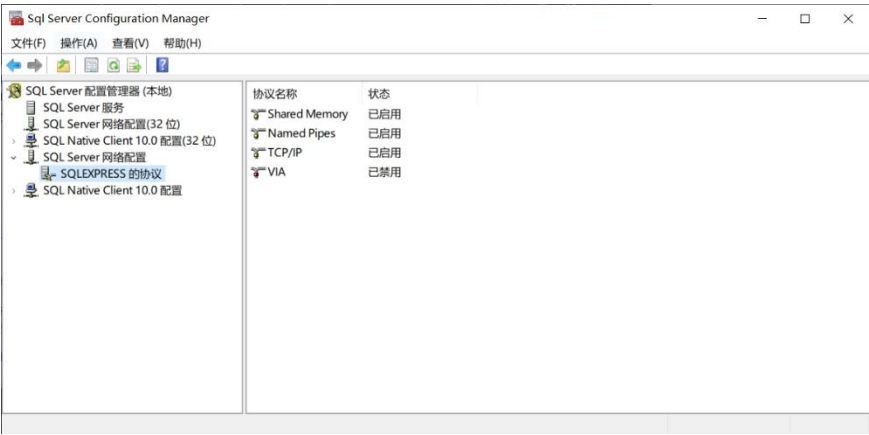
详细对照一下 Driver 压缩包里面的三个文件，应该是 mssql-jdbc-7.4.1.jre12.jar 对应 Java12

JAR	JDBC 版本法规遵从性	推荐的 Java 版本	说明
mssql-jdbc-7.4.1.jre8.jar	4.2	8	需要 Java Runtime Environment (JRE) 1.8。使用 JRE 1.7 或更低版本会引发异常。 7.4 中的新功能包括：JDK 12 支持、NTLM 身份验证和 useFmtOnly。
mssql-jdbc-7.4.1.jre11.jar	4.3	11	需要 Java 运行时环境 (JRE) 11.0。使用 JRE 10.0 或更低版本会引发异常。 7.4 中的新功能包括：JDK 12 支持、NTLM 身份验证和 useFmtOnly。
mssql-jdbc-7.4.1.jre12.jar	4.3	12	需要 Java Runtime Environment (JRE) 12.0。使用 JRE 11.0 或更低版本会引发异常。 7.4 中的新功能包括：JDK 12 支持、NTLM 身份验证和 useFmtOnly。

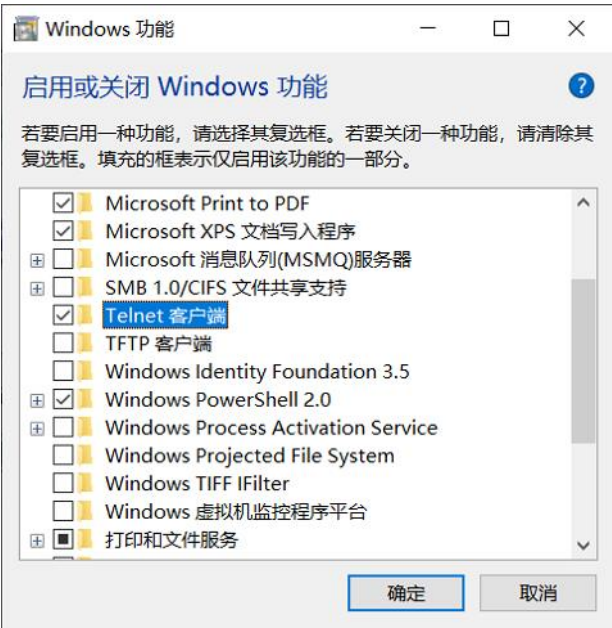
于是在系统环境里配置一下路径
到此驱动已经配置完成



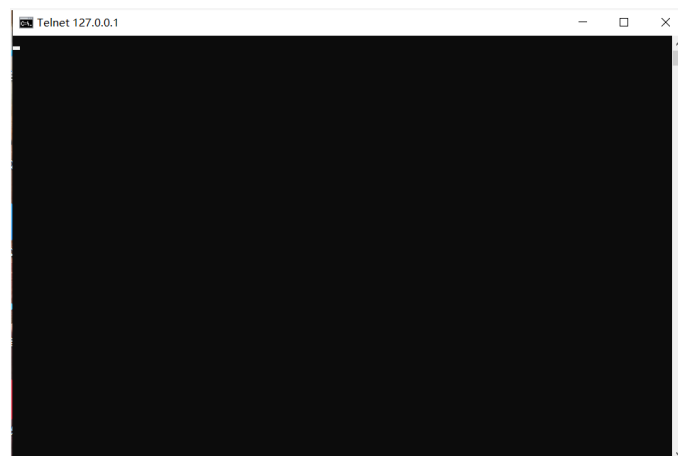
然后安装好 SQL server，之前学《数据库系统》的时候已经配置。
WIN+R 配置一下 Sql Server Configuration Manager，主要是配置 TCP/IP 协议中的 IP 地址项：IP1 和 IPALL 都加上 1433TCP 端口，IP1 和 IP10 的 IP 地址改为 127.0.0.1（本机），然后其余 IP 都将“已启用”打开为“是”，即可。
配置完成后，重启一下 SQL server 服务即可生效。



然后在 windows 功能中启用 windows 功能中的 Telnet 客户端。

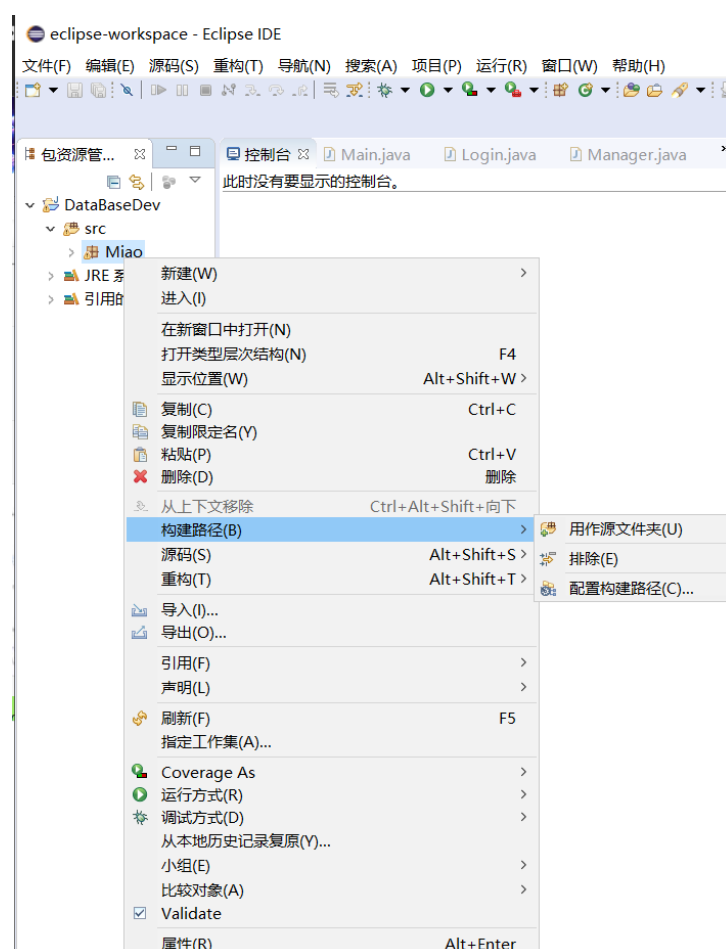


然后在 cmd 中验证一下，Telnet 127.0.0.1 成功。



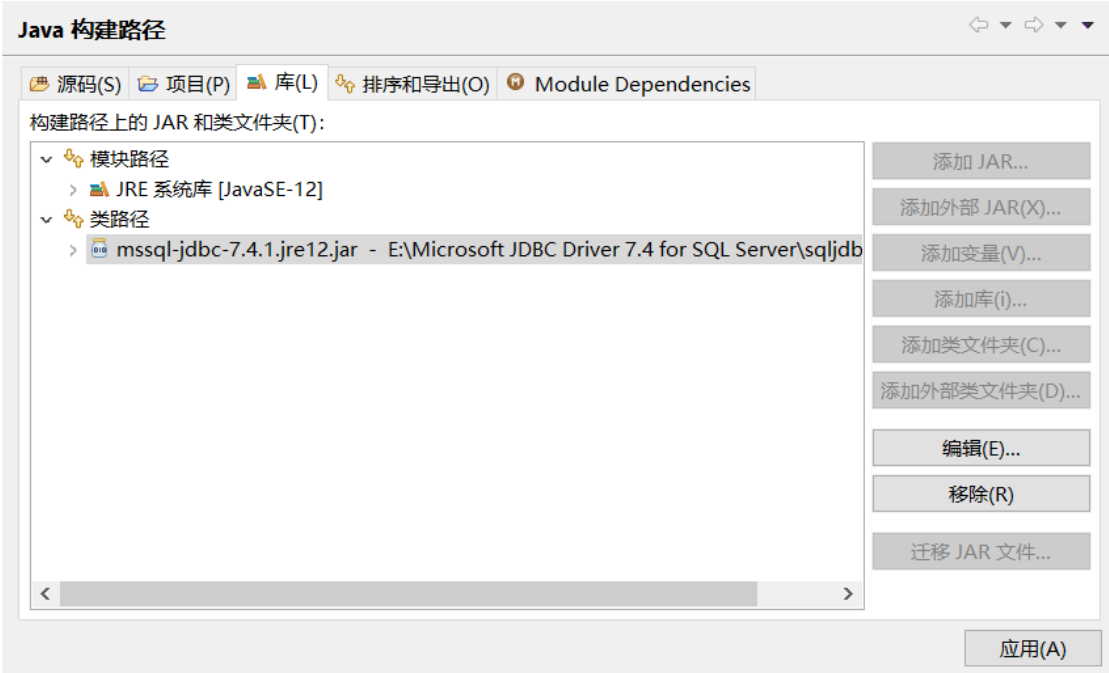
考虑到我用的是 Eclipse, 且每个 IDE 的供应商都提供了在 IDE 中设置 classpath 的不同方法。仅在操作系统中设置 classpath 将无法正常工作, 还需要在 IDE 的 classpath 中添加 sqljdbc.jar。

方法也很简单, 如下:

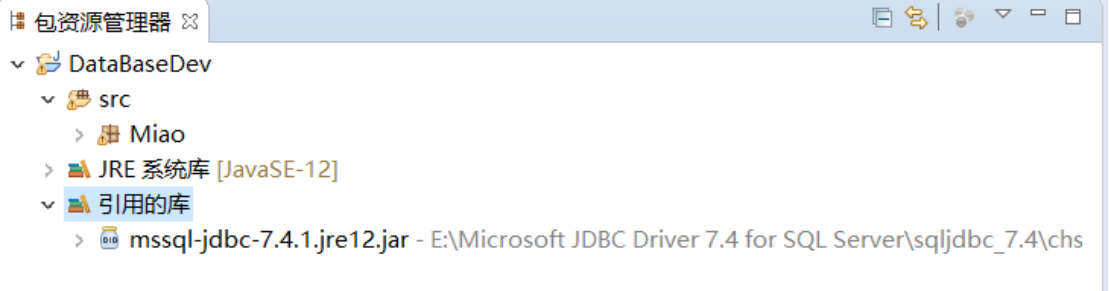


然后在这个包的配置中添加这个库

完成后如下：



在包资源管理器中如下：



到此，数据库相关的配置就完成了，下面介绍数据库的设计：

首先是后端，我选择的是 MS SQL Sever，考虑到高校科研管理系统的复杂性，我们对其进行了一定程度上的简化，首先是这个数据库总体上是如下所示：

左边是 Project 表，有三个字段，分别是：

proNo：项目号

proName：项目名

proCK：项目验收情况

teaNo：教师号

其中 proNo（项目号）是主键，另外还有外键 teaNo（教师号）

右边是 Teacher 表，有六个字段，分别是：

teaNo：教师号

teaName: 教师名

teaDept: 教师部门

teaDuty: 教师职务

teaTitle: 教师职称

teaGender: 教师性别

其中 teaNo (教师号) 是主键

然后暂时创建了三个用户，分为两种类型：

(1) 管理员 adminView, 密码是 123456

(2) 教师孙淼, 密码 123456

(3) 教师李东清, 密码 123456

对于管理员类型用户，我们给予他的权限是：

Update 更新数据库：也就是可以对科研项目进行验收，将其验收标志由默认的 0（代表未验收）验收为 1（代表已验收）；

Select 查找数据库：也就是可以看到所有的教师信息和科研信息，也可以通过主键 proNo（科研号）进行定向查询，或是通过主键 teaNo（教师号）进行定向查询；

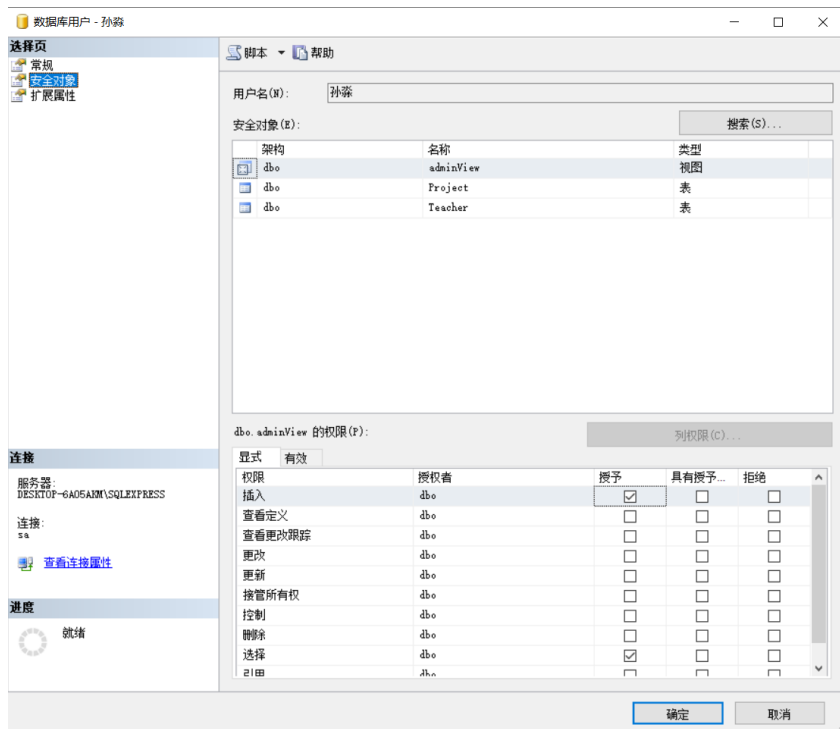
对于教师类型用户，我们给予他的权限是：

Insert 插入数据库，也就是可以申报新的科研项目，并且根据课设要求，其验收标志默认为 0（代表未验收）；

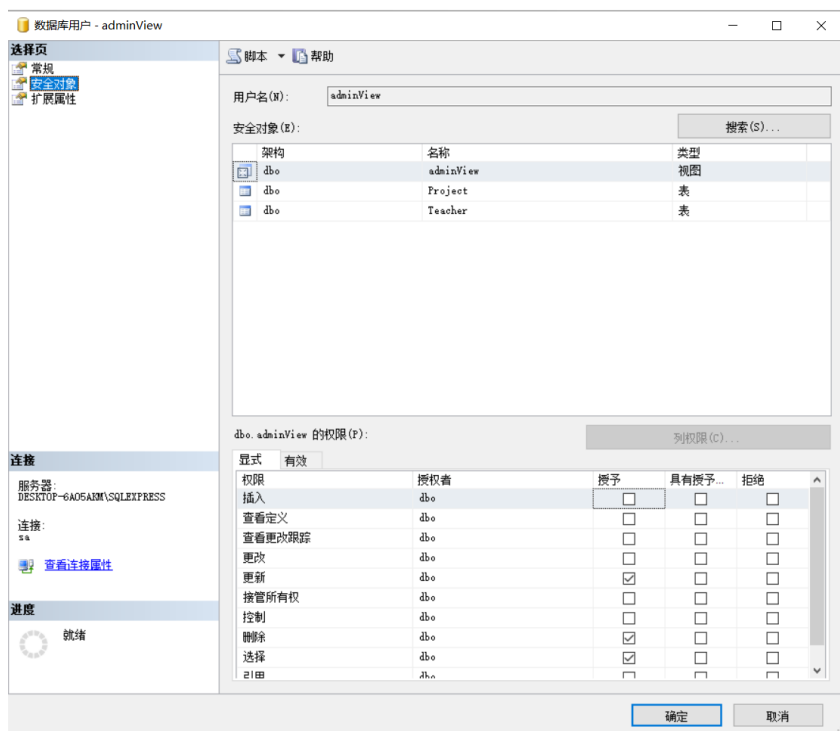
Select 查找数据库：也就是可以看到所有的教师信息和科研信息，也可以通过主键 proNo（科研号）进行定向查询，或是通过主键 teaNo（教师号）进行定向查询；

根据上面的设计，我们对数据库中不同的用户类型进行了不同的权限限制，如下：

对于教师类型用户“孙淼”，我只授予其对于两张表的，插入，选择权限。



对于管理员类型用户 adminView，我只授予其对于两张表的，更新，删除，选择权限。



接下来是根据数据库的结构，对课设的各种要求的 SQL 语句实现，如下：

1. --insert into teacher(teaNo,teaName,teaDept,teaDuty,teaTitle)
2. --values('2018211970','李东清','计算机院','学生','助教')
3. --insert into teacher(teaNo,teaName,teaDept,teaDuty,teaTitle)
4. --values('2018211958','孙淼','计算机院','学生','助教')

```
5. --select * from teacher
6.
7. --insert into project(proNo,proName,proCK,teaNo)
8. --values('0001','东的项目 1','0','2018211970')
9. --insert into project(proNo,proName,proCK,teaNo)
10. --values('0002','东的项目 2','1','2018211970')
11. --insert into project(proNo,proName,proCK,teaNo)
12. --values('0003','森的项目 1','0','2018211958')
13. --insert into project(proNo,proName,proCK,teaNo)
14. --values('0004','森的项目 2','1','2018211958')
15. --select * from project
16.
17. --select * from adminView
18.
19. /*update project
20. set proCK=0*/
21.
22. --定义触发器:
23. --planA
24. /*
25. create trigger audit
26. on project
27. after update
28. as
29. if(update(proCK))
30. begin
31. declare @newproCK nchar(1),@oldproCK nchar(1)
32. select @newproCK=proCK from inserted
33. select @oldproCK=proCK from deleted
34. if(@newproCK>@oldproCK)
35. print'审批通过'
36. else
37. print'项目已审批, 不能重复审批'
38. end
39. */
40.
41. --planB
42. /*
43.
44. */
45.
46. --更新试例:
47. /*
48. update project
```

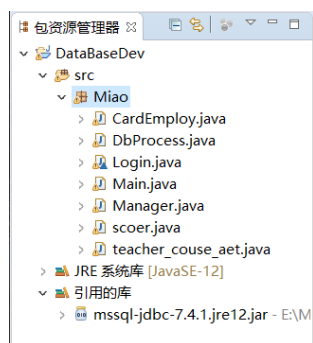
```

49. set proCK='1'where proNo='0003'
50. */
51.
52. --定义存储过程:
53.
54. create procedure proStats
55. as
56. begin
57.     select
58.         (select COUNT(proNo) FROM project where proCK='1') finCount,
59.         (select COUNT(proNo) from project ) decCount;
60. end
61.
62.
63. --执行存储过程:
64. -- exec dbo.proStats
65.
66. --默认值:
67. /*
68. alter table project
69. add constraint proCK default '0' for proCK
70. */
71. /*insert into project(proNo,proName,teaNo)
72. values('0005','东的项目 1','2018211970')*/
73.
74. --select * from adminView

```

包括:

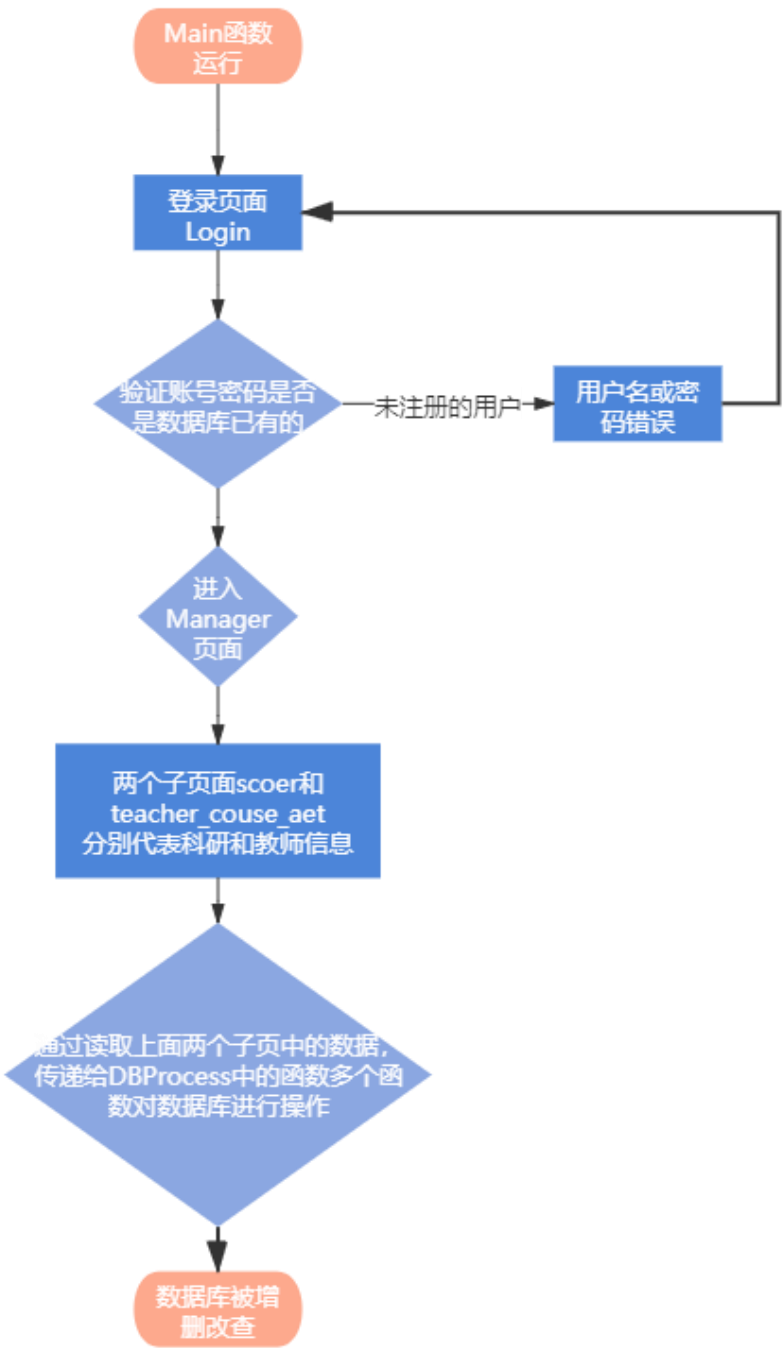
- 创建默认，并绑定到科研项目的验收标志，使其默认值为“未验收”；
- 创建触发器，验收项目时自动修改项目的验收标志为“验收通过”；
- 创建存储过程统计个院系科研项目的申报和完成数量；



四、系统实现

以上就是数据库部分，接下来介绍一下前端的设计，我是用 Java 实现的，程序树结构如上图所示：

软件系统中各个类的关系图大概如下所示：



下面对各个类中一些关键的代码进行介绍：首先是 Main 类，主函数，功能是加载数据库引擎，由于是本机地址和限定的端口，所以在代码中都要有体现，核心代码如下：

```
1. String JDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";// SQL 数据库引擎
2. String connectDB = "jdbc:sqlserver://127.0.0.1:1433;DatabaseName=test";// 数据源 DatabaseName 是已经创建的数据库的名字 不是表的名字
3.
4. try {
5.     Class.forName(JDriver);// 加载数据库引擎，返回给定字符串名的类
6. } catch (ClassNotFoundException e) {
7.     // e.printStackTrace();
8.     System.out.println("加载数据库引擎失败");
9.     System.exit(0);
10. }
11.
12. System.out.println("数据库驱动成功");
13.
14. Login login = new Login();
```

然后是其启动的登录页面类 Login，主要是一些控件，需要注意的是对登录名和密码的其他情况要进行一些判断，验收的时候李老师也指出这里应该还存在更多的可能，详细的和我们介绍了一个真正的高校科研管理系统要考虑的层次之复杂，让我收获很多：

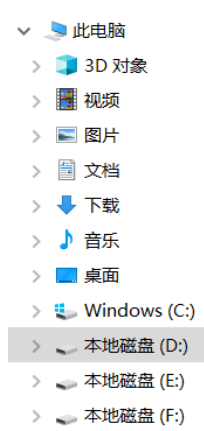
```
1.         else if(jTFuser.getText().isEmpty()&&jTFpassword.getText().isEmpty()
2.         )
3.         {
4.             JOptionPane.showMessageDialog(null,"请输入用户名和密码！","提示消息",JOptionPane.WARNING_MESSAGE);
5.         }else if(jTFuser.getText().isEmpty())
6.         {
7.             JOptionPane.showMessageDialog(null,"请输入用户名！","提示消息",JOptionPane.WARNING_MESSAGE);
8.         }else if(jTFpassword.getText().isEmpty())
9.         {
10.             JOptionPane.showMessageDialog(null,"请输入密码！","提示消息",JOptionPane.WARNING_MESSAGE);
11.         }else
12.         {
```

```

12.     JOptionPane.showMessageDialog(null, "用户名或者密码错误! \n 请重新输入", "提示
      消息", JOptionPane.ERROR_MESSAGE);
13.     //清空输入框
14.     clear();
15. }

```

然后是登录之后进入的科研系统主页 Manager 类，根据科研要求，调研了目前高校的一些信息系统的板式，我们选择了树形子目录结构（windows 中的文件系统结构），所以 Manager 实际上是一个各个子页面树节点的汇总，也就是下图所示的形式的模仿：



对于这部分，其核心代码如下：

```

1.  // 给树的各个结点赋值
2.  root = new DefaultMutableTreeNode("高校科研管理系统");
3.
4.  t1 = new DefaultMutableTreeNode("科研处");
5.  //t1_1 = new DefaultMutableTreeNode("科研项目管理");
6.  t1_2 = new DefaultMutableTreeNode("项目信息管理");
7.  //t1_3 = new DefaultMutableTreeNode("学生选课结果表");
8.  //t1_4 = new DefaultMutableTreeNode("删除员工资料");
9.  //t1_5 = new DefaultMutableTreeNode("查询全体员工");
10.
11. t2 = new DefaultMutableTreeNode("人事处");
12. //t2_1 = new DefaultMutableTreeNode("教师信息管理");
13. t2_2 = new DefaultMutableTreeNode("教师信息管理");

```

需要注意的是，在这里还有一个类是面板类 CardEmploy，其调用两个子类 scoer 和 teacher_couse_aet 用于显示数据库中的值和各种控件，进而读取控件中输入的值来生成 SQL 语句，进而完成对数据库的操作；

```
1. // 实例化 CardEmploy 面板 并加到 jsplitpane 的边
2. ae = new CardEmploy();
```

然后就是 CardEmploy 类的调用的两个子类，其中 scoer 类是科研信息管理子页面，这个类的主要功能是在其中获取控件上用户输入的值，将这些值读取之后传递给 DBProcess 类，并调用 DBProcess 类中的函数，继而实现对数据库的操作，这一部分的核心代码在于对数据库的四种操作的核心代码（并在 DBProcess 给出响应之后显示在控件上）：

- 对全部科研信息的查询

```
1. public void queryAllProcess()
2. {
3.     dbProcess.pro_queryAllProcess();
4.
5.     studentVector.clear();
6.
7.     sum = dbProcess.list.size() / 4 + "";
8.     checked = dbProcess.list_sum.size() + "";
9.     dbProcess.list_sum.clear();
10.
11.    jTFcourse.setText(checked + "/" + sum);
12.
13.    while (!dbProcess.list.isEmpty()) {
14.
15.        //System.out.print(dbProcess.list.poll());
16.        Vector v = new Vector();
17.        v.add(dbProcess.list.poll());
18.        v.add(dbProcess.list.poll());
19.        v.add(dbProcess.list.poll());
20.        v.add(dbProcess.list.poll());
21.        studentVector.add(v);
22.    }
23.    //
24.    studentJTable.updateUI();
25.
26.    System.out.println("读取完毕");
27.
28.    dbProcess.disconnect();
29. }
```

- 通过主键对科研部分信息的查询

```

1. public void queryProcess(String sQueryField)
2. {
3.     temp_proNo = sQueryField;
4.
5.     dbProcess.pro_queryProcess();
6.
7.     studentVector.clear();
8.
9.     while (!dbProcess.list.isEmpty()) {
10.
11.         //System.out.print(dbProcess.list.poll());
12.         Vector v = new Vector();
13.         v.add(dbProcess.list.poll());
14.         v.add(dbProcess.list.poll());
15.         v.add(dbProcess.list.poll());
16.         v.add(dbProcess.list.poll());
17.         studentVector.add(v);
18.     }
19.
20.     studentJTable.updateUI();
21.
22.     dbProcess.disconnect();
23. }

```

- 当用户是教师类型时，通过对数据库的插入操作完成项目的申报功能

```

1. public void insertProcess()
2. {
3.     System.out.println("准备申报");
4.
5.     temp_proNo = jTFSNo.getText().trim();
6.     temp_proName = jTFSName.getText().trim();
7.     temp_proCK = jTFcourse.getText().trim();
8.     temp_teaNo = jTFscoer.getText().trim();
9.
10.    dbProcess.pro_insertProcess();
11.
12.    System.out.println("申报成功");
13.
14.    studentVector.clear();
15.
16.    dbProcess.disconnect();
17. }

```


- 当用户是管理员类型时，通过对数据库的更新操作完成对项目的验收功能

```
1. public void updateProcess()
2. {
3.     temp_proNo = jTFSNo.getText().trim();
4.     temp_proCK = jTFcourse.getText().trim();
5.
6.     dbProcess.pro_updataProcess();
7.
8.     studentVector.clear();
9.
10.    dbProcess.disconnect();
11. }
```

对于教师信息的面板，我设计了查询和定向查询两种功能。

对于全部查询，其核心代码如下：

```
1. public void queryAllProcess()
2. {
3.     dbProcess.tea_queryAllProcess();
4.
5.     studentVector.clear();
6.
7.     while (!dbProcess.list.isEmpty()) {
8.
9.         Vector v = new Vector();
10.        v.add(dbProcess.list.poll());
11.        v.add(dbProcess.list.poll());
12.        v.add(dbProcess.list.poll());
13.        v.add(dbProcess.list.poll());
14.        v.add(dbProcess.list.poll());
15.        v.add(dbProcess.list.poll());
16.        studentVector.add(v);
17.    }
18.
19.    studentJTable.updateUI();
20.
21.    System.out.println("读取完毕");
22.
23.    dbProcess.disconnect();
24. }
```

对于通过主键教师号进行定向查询，核心代码如下：

```
1. public void queryProcess(String jTFQueryField)
2. {
3.
4.     temp_teaNo = jTFQueryField;
5.
6.     dbProcess.tea_queryProcess();
7.
8.     studentVector.clear();
9.
10.    while (!dbProcess.list.isEmpty()) {
11.
12.        //System.out.print(dbProcess.list.poll());
13.        Vector v = new Vector();
14.        v.add(dbProcess.list.poll());
15.        v.add(dbProcess.list.poll());
16.        v.add(dbProcess.list.poll());
17.        v.add(dbProcess.list.poll());
18.        v.add(dbProcess.list.poll());
19.        v.add(dbProcess.list.poll());
20.        studentVector.add(v);
21.    }
22.
23.    studentJTable.updateUI();
24.    dbProcess.disconnect();
25. }
```

对于最关键的 DBProcess 类，其才是实际上对数据库进行操作的类，通过获取两个“面板”类的控件上的值，根据用户需求生成或者填补产生 SQL 语句，对后端的数据库进行操作，其主要的函数也就是对应着上面两种面板里的六种函数，增改查，其核心代码如下：

- 对科研项目的全部查询的核心代码行

```
1. Statement stmt = con.createStatement();
2. ResultSet rs = stmt.executeQuery("SELECT * FROM Project");
3.
4. while (rs.next()) {
5.     list.add(rs.getString("proNo"));
6.     list.add(rs.getString("proName"));
```

```

7.    list.add(rs.getString("proCK"));
8.    list.add(rs.getString("teaNo"));
9. }

```

- 对教师信息的全部查询的核心代码行

```

1. Statement stmt = con.createStatement();
2. ResultSet rs = stmt.executeQuery("SELECT * FROM Teacher");
3.
4. while (rs.next()) {
5.     list.add(rs.getString("teaNo"));
6.     list.add(rs.getString("teaName"));
7.     list.add(rs.getString("teaDept"));
8.     list.add(rs.getString("teaDuty"));
9.     list.add(rs.getString("teaTitle"));
10.    list.add(rs.getString("teaGender"));
11. }

```

- 对科研信息进行通过项目号进行定向查询的核心代码行

```

1. String sql = "select * from Project where proNo = ";
2.
3. sql = sql + "'" + singel_query + "'";
4.
5.     Statement stmt = con.createStatement();
6.
7. ResultSet rs = stmt.executeQuery(sql);
8.
9.     while (rs.next()) {
10.         list.add(rs.getString("proNo"));
11.         list.add(rs.getString("proName"));
12.         list.add(rs.getString("proCK"));
13.         list.add(rs.getString("teaN

```

- 对教师信息进行通过项目号进行定向查询的核心代码行

```

1. String sql = "select * from Teacher where teaNo = ";
2.
3. sql = sql + "'" + singel_query + "'";
4.     Statement stmt = con.createStatement();
5.
6. ResultSet rs = stmt.executeQuery(sql);
7.
8.     while (rs.next()) {

```

```

9.         list.add(rs.getString("teaNo"));
10.        list.add(rs.getString("teaName"));
11.        list.add(rs.getString("teaDept"));
12.        list.add(rs.getString("teaDuty"));
13.        list.add(rs.getString("teaTitle"));
14.        list.add(rs.getString("teaGender"));
15.    }

```

- 对科研信息进行更新操作进而实现管理员对科研项目审批功能的实现

```

1. String sql = "update Project set proCK = '";
2. sql = sql + single_proCK + "'where proNo = '";
3. sql = sql + single_proNo + "'";
4.
5.     int rs = stmt.executeUpdate(sql);

```

- 对科研信息进行插入操作进而实现教师对科研项目进行申报功能的实现

```

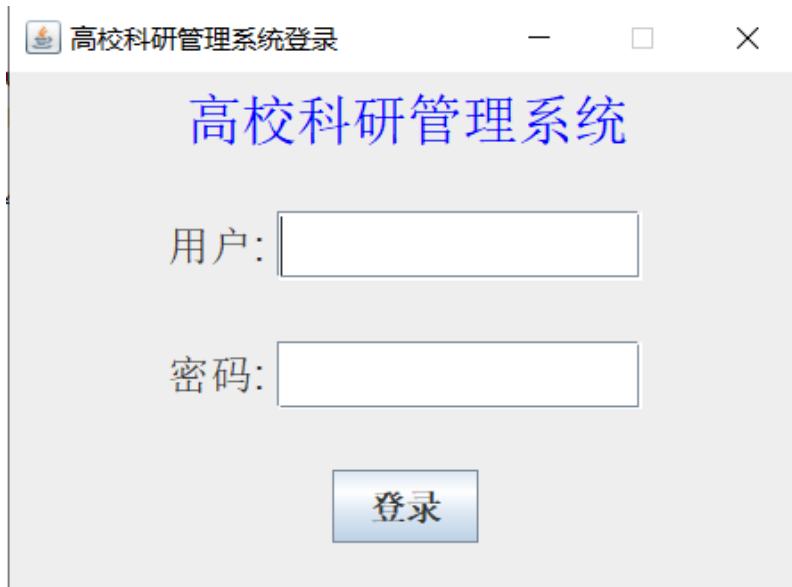
1. String sql = "insert into Project (proNo, proName, proCK, teaNo) values ('";
2. sql = sql + single_proNo + "', '";
3. sql = sql + single_proName + "', '";
4. sql = sql + "0" + "', '";
5. sql = sql + single_teaNo + "')";
6.
7.     int rs = stmt.executeUpdate(sql);

```

这次数据库学习中学到了许多，并且也懂得了很多，比如，数据库的索引是一个表中所包含的值得列表，注明了表中包含各种值得行所在的存储位置。创建索引，我最大的感受就是能大量的节约时间，特别是当表中数据很大的时候，规则，约束，默认值则一起保证了数据的完整性。规则是数据库中堆存数在标的列或用户名数据类型的值得规定和限制，约束定义了关于列中允许的规则，默认值就是用户输入记录是没有指定具体数据自动插入的数据。有个美国数据库设计专家说：“键，到处都是键，除了键之外，什么也没有”，

遇见许多困难的时候要多多去查阅资料和多去看前人所做的实例来解决自己的问题，同时也学到了许多解决问题的方法。

最后就是系统演示了，首先运行 Main 主函数，主函数调用 Login() 打开了登录页面：



高校科研管理系统登录

高校科研管理系统

用户:

密码:

登录

如果忘记输入密码，账号，或是账号密码不正确，会报错：



高校科研管理系统登录

高校科研管理系统

用户: adminView

密码: 12345

登录

提示消息

用户名或者密码错误！
请重新输入

确定

然后可以登录一下管理员账号，adminView，123456，看一下管理员的功能是否成功：



高校科研管理系统登录

高校科研管理系统

用户: adminView

密码: 123456

登录

成功登录，进入管理员页面，项目信息管理子页面如下：

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名称	验收情况	教师号
-----	------	------	-----

请通过项目号查找

项目号 项目名称
验收情况 教师号

教师信息管理子页面如下：

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

教师信息

教师号	教师名	部门	职务	职称	性别
-----	-----	----	----	----	----

请通过教师号查询

教师号

教师号 教师名 部门
职务 职称 性别

然后可以在项目信息管理页面查询所有科研项目信息：

并且可以看到“验收情况”栏中是已验收的项目数和项目总数的比值 4/8。

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名	验收情况	教师号
00001	东的项目1	1	2018211970
00002	1	1	2018211958
00003	森的项目1	1	2018211958
00004	admin修改项目名字和CK...	1	2018211958
00005	孙森用户通过基表project...	0	2018211958
00006	李东清用户通过基表proje...	0	2018211970
00007	最后一次测试	0	2018211958
00008	验收给老师看	0	2018211958

请通过项目号查找

项目号 项目名 验收情况
4/8 教师号

然后可以进行单项查询，通过主键 proNo（项目号）进行定向查询：

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名	验收情况	教师号
00001	东的项目1	1	2018211970

请通过项目号查找

项目号 项目名 验收情况
4/8 教师号

还可以对项目进行验收，我们对未验收的 00005 号项目进行验收看看：
在下方的栏中输入要验收的项目号和项目验收情况，然后点击“更新”

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名称	验收情况	教师号
00001	东的项目1	1	2018211970
00002	1	1	2018211958
00003	森的项目1	1	2018211958
00004	admin修改项目名称和CK...	1	2018211958
00005	孙森用户通过基表project...	0	2018211958
00006	李东清用户通过基表proje...	0	2018211970
00007	最后一次测试	0	2018211958
00008	验收给老师看	0	2018211958

请通过项目号查找

项目号 项目名称
验收情况 教师号

然后再查询所有项目，就可以看到 00005 号项目的验收情况已经从 0 变为了 1
总的验收情况也从 4/8 变成了 5/8

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名称	验收情况	教师号
00001	东的项目1	1	2018211970
00002	1	1	2018211958
00003	森的项目1	1	2018211958
00004	admin修改项目名称和CK...	1	2018211958
00005	孙森用户通过基表project...	1	2018211958
00006	李东清用户通过基表proje...	0	2018211970
00007	最后一次测试	0	2018211958
00008	验收给老师看	0	2018211958

请通过项目号查找

项目号 项目名称
验收情况 教师号

然后管理员也可以对教师信息进行全部查询：

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

教师信息

教师号	教师名	部门	职务	职称	性别
2018211958	孙森	计算机与信息学院	码农	助教	男
2018211970	李东清	计算机与信息学院	码农	助教	男

请通过教师号查询

教师号

查询

查询所有记录

教师号

教师名

部门

职务

职称

性别

插入

更新

删除当前记录

以及通过主键教师号的定向查询：

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

教师信息

教师号	教师名	部门	职务	职称	性别
2018211958	孙森	计算机与信息学院	码农	助教	男

请通过教师号查询

教师号

查询

查询所有记录

教师号

教师名

部门

职务

职称

性别

插入

更新

删除当前记录

到此，管理员的功能就展示完毕，然后登陆一下教师账号：

高校科研管理系统登录

高校科研管理系统

用户: 孙淼

密码: 123456

登录

查询功能都是一样的，就不展示了，直接展示其的项目申报功能（管理员所没有的功能）：

首先查询所有项目，可以看到项目 00008 是已经被申报的,00009 没有被申报的，我们如果申报 00008 会被 SQL server 阻止，这个是数据库的主键的内部逻辑，不需要我实现，我们直接申报 00009 号项目，教师号是已经存在的 2018211958，通过外键可以直接连接到教师名字孙淼上，而不需要我写到控件里面，根据课设要求，项目的验收情况是默认，所以我不输入：

高校科研管理系统

高校科研管理系统

科研处

项目信息管理

人事处

教师信息管理

科研项目信息

项目号	项目名	验收情况	教师号
00001	东的项目1	1	2018211970
00002	1	1	2018211958
00003	森的项目1	1	2018211958
00004	admin修改项目名字和CK...	1	2018211958
00005	孙淼用户通过基表project...	1	2018211958
00006	李东清用户通过基表proje...	0	2018211970
00007	最后一次测试	0	2018211958
00008	验收给老师看	0	2018211958

请通过项目号查找

查询

查询所有记录

项目号 00009

项目名 国庆节快乐

验收情况

教师号 2018211958

插入

更新

删除当前记录

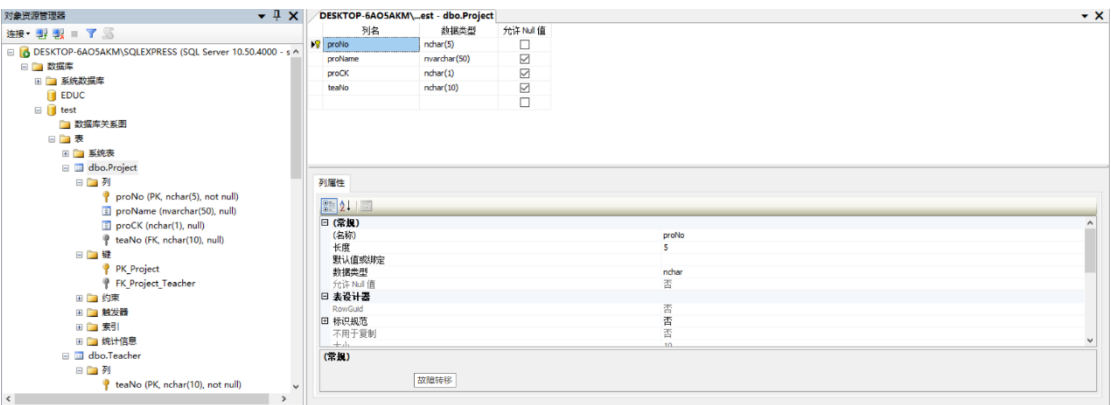
删除所有记录

点击插入之后，查询全部，就可以看到这个项目已经被插入了，并且验收情况是0（默认）：

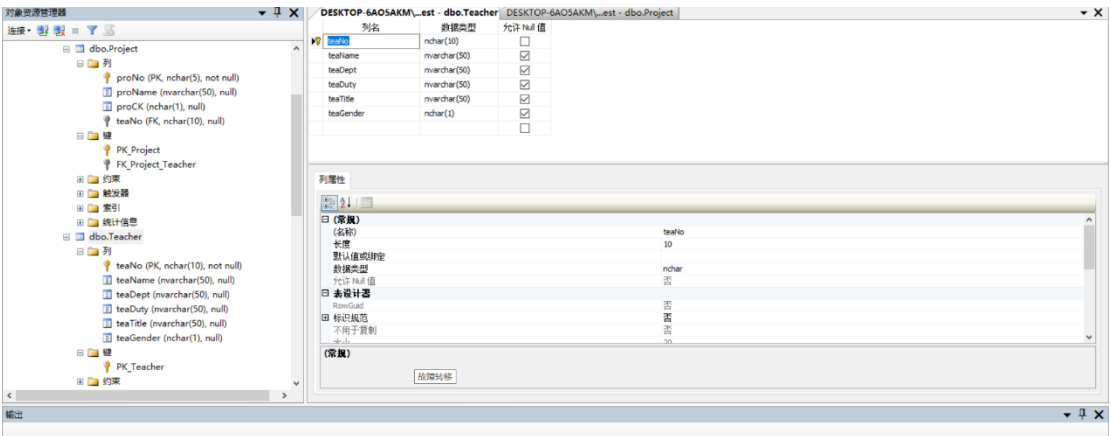


对于数据库部分，我们设计了两个表和一个视图。

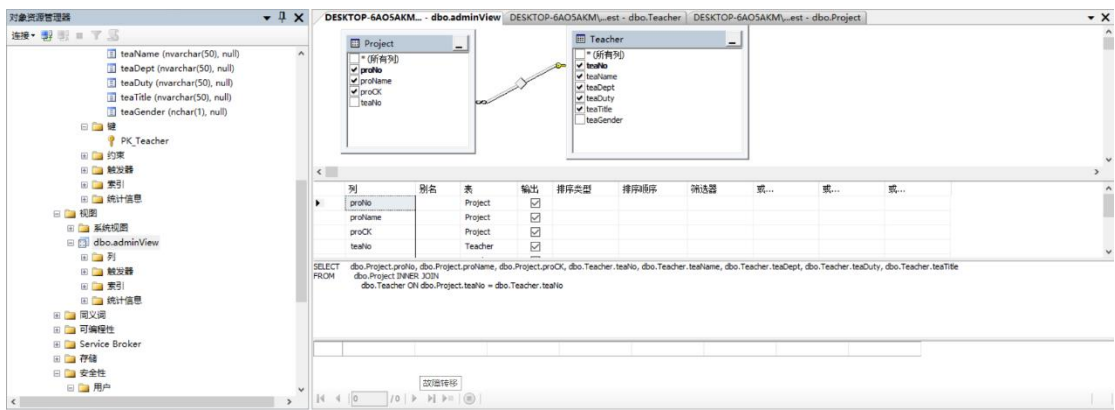
Project 表如下：



Teacher 表如下：



adminView 视图如下：



五、验收情况及改进

验收情况：情况良好，通过。并得到李老师肯定，后续在李老师指点下对实验内容进行了一定程度上的升级和反思。

1) 由于对高校科研系统有一定的了解，所以选择了这个作为题目，但是随着项目的深入，我们发现项目越往下进行下去，遇到的困难和问题越多。就如李老师所说，如果真的要完成一个标准的、可投入实际使用的科研管理系统，我们需要考虑的东西会很多：高校科研管理所设计的事务非常繁杂，涉及到校科研处、各个系的科研管理以及个人对科研信息的查询统计等。所设计的系统既要有利于科研处的监督管理又要有利于各个系及教师的分工协作。系统功能主要包括校科研处科研管理、院系(或部门)科研管理、个人科研信息查询、校级科研项目整体信息管理。可以划分为科研信息维护、科研信息查询与统计、科研信息报表、科研信息分析、校级课题申报、审批、合同签订、结题、其它处理等操作；其中科研信息应细分为学术论文、著作、科研项目、科研成果、科研奖励等信息。

2) 我和配合的队友李东清也严格按照实验要求进行了需求分析等步骤，具体情况可以见上面的历程介绍：①通过调查研究和运用 Internet，收集和调查有关资料、最新技术信息。②理解和掌握数据库系统的需求分析过程和基本方法③基本掌握撰写设计报告的基本步骤和写作方法。④根据课题的要求基本理解和掌握 E-R 图的设计方法和关系模式的转换。⑤根据 ER 图生成数据库表，进行数据库的逻辑设计。⑥数据库完整性、安全性保证措施。⑦数据库的物理设计。⑧数据库实施维护计划。⑨应用系统的设计和实现。

3) 一些关键的技术，比如数据库表的创建：如果严格按照要求来的话，我们后续又进行了思考和讨论，结果如下：

根据系统需求分析,将系统的实现直接定位在更好的适合各层次人员的需求和操作,系统的所有初始化数据均采用界面录入的方法,由各类人员分工完成。根据我校的实际情况,笔者将科研人员基本信息、论文信息、科研项目信息、成果信息、获奖情况信息等作为系统的初始化数据,为个人和院系查询、统计、打印,科研处操作、加工、管理、集成等工作提供基本的数据平台。主要数据表格设计如下:

科研人员信息表(职工号,姓名,性别,职称,出生日期,所学专业,现从事专业,学历,学位,工作时间,所在部门,备注)。

论文信息表(序号,论文名称,刊物名称,刊物类别,刊物级别,主办单位,作者,职工号,发表时间,刊号,备注)。

论著信息表(序号,著作名称,图书编号,著作类别,出版社,学科类别,作者,职工号,出版时间,版次,备注)。

科研项目表(序号,项目编号,项目名称,项目来源,项目类别,完成单位,负责人,课题组成员,立项时间,拟定期限,经费金额,是否鉴定,鉴定时间,鉴定单位,备注)。

科研成果表(序号,成果名称,成果类别,成果经费,完成时间,负责人,完成单位,是否投入生产,经济效益,备注)。

科研奖励表(序号,获奖人员,项目名称,奖励名称,授予单位,时间,备注)。

校级项目管理表(项目编号,项目名称,课题名称,负责人,所在部门,职务或职称,联系电话,申报表,申报时间,申报经费,是否批准,批准日期,批准经费,合同时间,合同内容,是否结题,结题时间,鉴定技术负责人,项目成果效益,备注)。

校级鉴定专家表(序号,姓名,性别,出生日期,工作单位,学历,学位,所学专业,现从事专业,职称职务,备注)。

在上述表格中,加下划线字段代表主键,加双划线字段代表外键。其中科研人员信息表之所以选择(职工号,姓名)作为主键,考虑到姓名经常作为查询、操作条件,并在相关表中作为科技人员的重要描述和限制;为了避免数据录入的无序状况,数据库表之间建立了严格的参照完整性,并且对所有表都分配了操作权限,并将错误信息及时反馈给操作人员,科研人员信息表、论文信息表、论著信息表

通过“职工号”建立参照完整性关系，在录入论文、论著时，作者必须是本单位职工，如果科研人员信息表不存在该员工，便给出错误提示，提示检查是否是合法职工，如果是，便可以在科研人员信息表中添加该员工信息后录入论文、论著信息，否则，拒绝录入，“职工号”又为组合查询提供了连接条件支持；同时将科研项目表、科研成果表、校级项目管理表中“负责人”和科研人员信息表中的“姓名”建立主键、外键参照完整性关系，校级课题的申报负责人也必须是本单位某部门的职工；获奖人员，必须是科研人员信息表中某一职工；将校级鉴定专家表和校级项目管理表通过“鉴定技术负责人”、“姓名”建立参照完整性管理，限定了鉴定技术负责人必须是本校专家表中的专业人员；所有表的主键设置都加上“姓名”字段，目的是为了符合人们的常规习惯，常以某个专业人员姓名作为操作、查询条件，科研人员信息表与其他相关表格建立参照完整性为了限制科研、论文、论著、成果、奖励等信息的科学合法性，同时有利于统计和查询；只有完善的参照完整性，才能使得数据库中数据具有一致性和互操作性。

4) 对于数据库安全设计角度：

网络的安全非常重要，从数据库级、服务器级和应用程序级综合考虑，在科研管理系统中要根据不同用户，设置不同的权限、不同的初始化菜单。在本系统中，数据库的安全主要通过数据库的存取控制机制实现的。首先定义各类管理人员的操作权限即角色，其次定义数据库登录，最后依据用户权限表将登录分配为相应的角色。由于人员的复杂性，角色分为三个层次：科研处级别、二级部门级别、普通教师级别。用户属于某一角色，即使用户职务调动时，对用户增删，不影响其他用户的操作，只是角色中用户数量的变化。建立专门的系统使用权限表来记录用户和权限，同时也兼容部门设置信息，对不同部门分配相应的权限和账号，科研处是最高管理级别部门。

为了进一步保护数据库，建立备份数据库服务器，定期进行数据库备份和复制，本系统数据库主要采用 SQL Server 数据库管理工具进行设计，它具有强大的管理、安全、登陆、网络支持功能，且可以和其它数据库进行转换，是一种网络数据库管理软件，符合设计要求。

最主要的收获还是这是第一次从整体考虑，建立严格的参照完整性和科学数据库结构，将存储过程、触发器应用到数据库设计中，加大了后台的功能和效率，

对数据库的安全进行分级、分类设置和管理，提高了数据库的安全访问能力；它的数据冗余少，具有更高的数据一致性、可操作性和安全性，结合前台开发工具和平台，可以实现强大的功能设计和系统实现。

附录(源代码)

CardEmploy.java

```
1. package Miao;
2.
3. import javax.swing.*;
4. import java.awt.*;
5.
6. public class CardEmploy extends Panel{
7.
8.     CardLayout c;
9.     //查询表
10.    //DatabaseCourseDesign selE;
11.    //添加表
12.    scoer addE;
13.
14.    //Teachercahar teaE;
15.
16.    teacher_couse_aet kaikeE;
17.
18.    //couse_result resE;
19.
20.    //修改员工信息表
21.    /*
22.    ReviseEmploy revE;
23.
24.    //删除员工表格
25.    DelEmploy delE;
26.
27.    //所有员工信息
28.    AllEmploy allE;
29.
30.    //人事变动信息
31.    Examine exaE;
32.
33.    //历史记录界面
34.    History His;
35.    */
36.    public CardEmploy()
```

```
37.     {
38.         //查询员工表
39.         //selE = new DatabaseCourseDesign();
40.
41.         //添加员工表
42.         addE = new scoer();
43.
44.         //teaE = new Teachercahar();
45.
46.         kaikeE =new teacher_couse_aet();
47.
48.         //resE = new couse_result();
49.         //修改员工信息
50.         /*
51.         revE = new ReviseEmploy();
52.
53.         //删除员工表格
54.
55.
56.         //所有员工信息
57.         allE =new AllEmploy();
58.
59.         //人事变动信息
60.         exaE = new Examine();
61.         //历史记录界面
62.         His = new History();
63.         */
64.
65.         JPanel jp = new JPanel();
66.
67.         //定义 cardemploy 面板 为卡片布局
68.         //把各个面板加入到 C 的卡片布局中
69.         c = new CardLayout();
70.         this.setLayout(c);
71.         //this.add(selE,"1");
72.         this.add(addE,"2");
73.         //this.add(resE,"3");
74.         //this.add(teaE,"4");
75.         this.add(kaikeE,"5");
76.         /*
77.         this.add(exaE,"6");
78.         this.add(His,"7");
79.         */
80.     }
```


81. }

DbProcess. java

```
1. package Miao;
2.
3. import java.sql.*;
4. import java.util.ArrayList;
5. import java.util.LinkedList;
6. import java.util.List;
7. import java.util.Queue;
8. import java.util.Vector;
9.
10. import javax.swing.JOptionPane;
11.
12.
13. public class DbProcess{
14.     Connection connection = null;
15.     public ResultSet rs = null;
16.
17.
18.     public Queue<String> list = new LinkedList<String>();
19.     public Queue<String> list_sum = new LinkedList<String>();
20.
21.     //
22.     Connection con;// 连接数据库对象
23.     Statement stmt;// 创建 SQL 命令对象
24.     //
25.
26.     //Statement stmt;
27.
28.     //mysql 数据库 url
29.     //String userMySql="root";
30.     //String passwordMySql="mzc277171";
31.
32.     String JDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";/
/ SQL 数据库引擎
33.     String connectDB = "jdbc:sqlserver://127.0.0.1:1433;DatabaseName=
test";// 数据源 DatabaseName 是已经创建的数据库的名字 不是表的名字
34.
35.
36.     //String userMySql = "com.microsoft.sqlserver.jdbc.SQLServerDrive
r";// SQL 数据库引擎
37.     //String passwordMySql = "jdbc:sqlserver://127.0.0.1:1433;Databas
eName=test";// 数据源 DatabaseName 是已经创建的数据库的名字 不是表的名字
38.
```

```

39.     //String urlMySQL = "jdbc:mysql://localhost:3306/education?user="
    +userMySQL+"&password="+passwordMySQL + "&useUnicode=true&characterEn
    coding=gbk";
40.     public DbProcess() {
41.
42.         try {
43.             Class.forName(JDriver);// 加载数据库引擎, 返回给定字符串名的
            类
44.         } catch (ClassNotFoundException e) {
45.             // e.printStackTrace();
46.             System.out.println("加载数据库引擎失败");
47.             System.exit(0);
48.         }
49.
50.
51. //     try {
52. //         //mysql 数据库设置驱动程序类型
53. //         Class.forName("com.mysql.jdbc.Driver");
54. //         System.out.println("mysql 数据库驱动加载成功");
55. //     }
56. //     catch(java.lang.ClassNotFoundException e) {
57. //         e.printStackTrace();
58. //     }
59. }
60.
61.
62. public void pro_queryAllProcess(){
63.
64.     try {
65.         if(Login.identify.equals("adminView")) {
66.
67.             String user = "adminView";
68.             String password = "123456";
69.
70.             Connection con = DriverManager.getConnection(connectD
            B, user, password);// 连接数据库对象
71.             System.out.println("admin 连接数据库成功,进入到 DbProcess
            阶段");
72.
73.             Statement stmt = con.createStatement();
74.             ResultSet sum = stmt.executeQuery("SELECT * FROM Proj
            ect where proCK = '1'");
75.
76.             while (sum.next()) {

```

```
77.                list_sum.add(sum.getString("proCK"));
78.            }
79.
80.
81.                ResultSet rs = stmt.executeQuery("SELECT * FROM Proje
ct");
82.
83.
84.                while (rs.next()) {
85.                    list.add(rs.getString("proNo"));
86.                    list.add(rs.getString("proName"));
87.                    list.add(rs.getString("proCK"));
88.                    list.add(rs.getString("teaNo"));
89.                }
90.                System.out.println("全部查询—完成");
91.            }
92.            if(Login.identify.equals("孙淼")) {
93.
94.                String user = "孙淼";
95.                String password = "123456";
96.
97.                Connection con = DriverManager.getConnection(connectD
B, user, password);// 连接数据库对象
98.                System.out.println("admin 连接数据库成功,进入到 DbProcess
阶段");
99.
100.                Statement stmt = con.createStatement();
101.                ResultSet rs = stmt.executeQuery("SELECT * FROM Pr
oject");
102.
103.                while (rs.next()) {
104.                    list.add(rs.getString("proNo"));
105.                    list.add(rs.getString("proName"));
106.                    list.add(rs.getString("proCK"));
107.                    list.add(rs.getString("teaNo"));
108.                }
109.                System.out.println("全部查询—完成");
110.            }
111.        }
112.        catch(Exception e){
113.            e.printStackTrace();
114.        }
115.    }
116.
```



```
155.         list.add(rs.getString("teaName"));
156.         list.add(rs.getString("teaDept"));
157.         list.add(rs.getString("teaDuty"));
158.         list.add(rs.getString("teaTitle"));
159.         list.add(rs.getString("teaGender"));
160.     }
161.     System.out.println("全部查询—完成");
162. }
163. }
164.     catch(Exception e){
165.         e.printStackTrace();
166.     }
167. }
168.
169.     public void disconnect(){
170.         try{
171.             if(connection != null){
172.                 connection.close();
173.                 connection = null;
174.             }
175.         }
176.         catch(Exception e){
177.             e.printStackTrace();
178.         }
179.     }
180.
181.
182. // public ResultSet executeQuery(String sql) {
183. //     try {
184. //         System.out.println("executeQuery(). sql = " + sql);
185. //         PreparedStatement pstmt = connection.prepareStatement(s
186. //             ql);
187. //         // 执行查询
188. //         rs = pstmt.executeQuery();
189. //     }
190. //     catch(SQLException ex) {
191. //         ex.printStackTrace();
192. //     }
193. //     return rs;
194. // }
195. //插入
196. //executeUpdate 的返回值是一个整数, 指示受影响的行数(即更新计数)。
197. //executeUpdate 用于执行 INSERT、UPDATE 或 DELETE 语句
```



```
240.         list.add(rs.getString("proCK"));
241.         list.add(rs.getString("teaNo"));
242.     }
243.     System.out.println("读取完毕");
244.
245.     // 将查询获得的记录数据, 转换成适合生成 JTable 的数据形式
246. }
247. else if(Login.identify.equals("孙淼")) {
248.     // 建立查询条件
249.     String sql = "select * from Project where proNo = ";
250.
251.     sql = sql + "'" + singel_query + "'";
252.
253.     String user = "孙淼";
254.     String password = "123456";
255.
256.     Connection con = DriverManager.getConnection(connectDB, user, password);// 连接数据库对象
257.     System.out.println("只根据项目号—查询完成");
258.
259.     Statement stmt = con.createStatement();
260.
261.     ResultSet rs = stmt.executeQuery(sql);
262.
263.     while (rs.next()) {
264.         list.add(rs.getString("proNo"));
265.         list.add(rs.getString("proName"));
266.         list.add(rs.getString("proCK"));
267.         list.add(rs.getString("teaNo"));
268.     }
269.     System.out.println("读取完毕");
270.
271.     // 将查询获得的记录数据, 转换成适合生成 JTable 的数据形式
272. }
273.
274. }catch(SQLException sqle){
275.     System.out.println("sqle = " + sqle);
276.     JOptionPane.showMessageDialog(null,
277.         "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
278. }catch(Exception e){
279.     System.out.println("e = " + e);
280.     JOptionPane.showMessageDialog(null,
281.         "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
282. }
```

```
283.     }
284.
285.     public void tea_queryProcess()
286.     {
287.         String singel_query = teacher_couse_aet.temp_teaNo;
288.
289.         try{
290.             if(Login.identify.equals("adminView")) {
291.                 // 建立查询条件
292.                 String sql = "select * from Teacher where teaNo = ";
293.
294.                 sql = sql + "'" + singel_query + "'";
295.
296.                 String user = "adminView";
297.                 String password = "123456";
298.
299.                 Connection con = DriverManager.getConnection(connectDB, user, password);// 连接数据库对象
300.                 System.out.println("只根据项目号—查询完成");
301.
302.                 Statement stmt = con.createStatement();
303.
304.                 ResultSet rs = stmt.executeQuery(sql);
305.
306.                 while (rs.next()) {
307.                     list.add(rs.getString("teaNo"));
308.                     list.add(rs.getString("teaName"));
309.                     list.add(rs.getString("teaDept"));
310.                     list.add(rs.getString("teaDuty"));
311.                     list.add(rs.getString("teaTitle"));
312.                     list.add(rs.getString("teaGender"));
313.                 }
314.                 System.out.println("读取完毕");
315.
316.                 // 将查询获得的记录数据，转换成适合生成 JTable 的数据形式
317.             }
318.
319.             else if(Login.identify.equals("孙淼")) {
320.                 // 建立查询条件
321.                 String sql = "select * from Teacher where teaNo = ";
322.
323.                 sql = sql + "'" + singel_query + "'";
324.
325.                 String user = "孙淼";
```



```
326.         String password = "123456";
327.
328.         Connection con = DriverManager.getConnection(connectDB, us
er, password);// 连接数据库对象
329.         System.out.println("只根据项目号—查询完成");
330.
331.         Statement stmt = con.createStatement();
332.
333.         ResultSet rs = stmt.executeQuery(sql);
334.
335.         while (rs.next()) {
336.             list.add(rs.getString("teaNo"));
337.             list.add(rs.getString("teaName"));
338.             list.add(rs.getString("teaDept"));
339.             list.add(rs.getString("teaDuty"));
340.             list.add(rs.getString("teaTitle"));
341.             list.add(rs.getString("teaGender"));
342.         }
343.         System.out.println("读取完毕");
344.
345.         // 将查询获得的记录数据, 转换成适合生成 JTable 的数据形式
346.     }
347.
348.     }catch(SQLException sqle){
349.         System.out.println("sqle = " + sqle);
350.         JOptionPane.showMessageDialog(null,
351.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
352.     }catch(Exception e){
353.         System.out.println("e = " + e);
354.         JOptionPane.showMessageDialog(null,
355.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
356.     }
357. }
358.
359. public void pro_updataProcess() {
360.
361.     String single_proNo = scoer.temp_proNo;
362.     String single_proCK = scoer.temp_proCK;
363.
364.     try{
365.         if(Login.identify.equals("adminView")) {
366.
367.             String sql = "update Project set proCK = '";
368.             sql = sql + single_proCK + "'where proNo = '";
```

```
369.         sql = sql + single_proNo + "";
370.
371.         String user = "adminView";
372.         String password = "123456";
373.
374.         Connection con = DriverManager.getConnection(connectDB, user, password); // 连接数据库对象
375.
376.         Statement stmt = con.createStatement();
377.
378.         int rs = stmt.executeUpdate(sql);
379.         //System.out.println("");
380.
381.         //         while (rs > 0) {
382.         //             list.add(rs.getString("proNo"));
383.         //             list.add(rs.getString("proName"));
384.         //             list.add(rs.getString("proCK"));
385.         //             list.add(rs.getString("teaNo"));
386.         //             rs--;
387.         //         }
388.         System.out.println("项目验收完毕");
389.
390.         // 将查询获得的记录数据, 转换成适合生成 JTable 的数据形式
391.     }
392.
393.     }catch(SQLException sqle){
394.         System.out.println("sqle = " + sqle);
395.         JOptionPane.showMessageDialog(null,
396.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
397.     }catch(Exception e){
398.         System.out.println("e = " + e);
399.         JOptionPane.showMessageDialog(null,
400.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
401.     }
402.
403. }
404.
405.
406. public void pro_insertProcess() {
407.
408.     String single_proNo = scoer.temp_proNo;
409.     String single_proName = scoer.temp_proName;
410.     String single_proCK = scoer.temp_proCK;
411.     String single_teaNo = scoer.temp_teaNo;
```

```
412.
413.     try{
414.         if(Login.identify.equals("孙淼")) {
415.
416.             String sql = "insert into Project (proNo, proName, pro
                CK, teaNo) values ('";
417.             sql = sql + single_proNo + "','";
418.             sql = sql + single_proName + "','";
419.             sql = sql + "0" + "','";
420.             sql = sql + single_teaNo + "');";
421.
422.             String user = "孙淼";
423.             String password = "123456";
424.
425.             Connection con = DriverManager.getConnection(connectDB, us
                er, password);// 连接数据库对象
426.
427.             Statement stmt = con.createStatement();
428.
429.             int rs = stmt.executeUpdate(sql);
430.             //System.out.println("");
431.
432.             //         while (rs > 0) {
433.             //             list.add(rs.getString("proNo"));
434.             //             list.add(rs.getString("proName"));
435.             //             list.add(rs.getString("proCK"));
436.             //             list.add(rs.getString("teaNo"));
437.             //             rs--;
438.             //         }
439.             System.out.println("项目验收完毕");
440.
441.             // 将查询获得的记录数据，转换成适合生成 JTable 的数据形式
442.         }
443.
444.     }catch(SQLException sqle){
445.         System.out.println("sqle = " + sqle);
446.         JOptionPane.showMessageDialog(null,
447.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
448.     }catch(Exception e){
449.         System.out.println("e = " + e);
450.         JOptionPane.showMessageDialog(null,
451.             "数据操作错误", "错误", JOptionPane.ERROR_MESSAGE);
452.     }
453. }
```

```
454.  
455. }
```

Login. java

```
1. package Miao;  
2.  
3. //科研管理系统登录页面  
4.  
5. import javax.imageio.ImageIO;  
6. import javax.swing.*;  
7.  
8. import Miao.Manager;  
9. import java.awt.*;  
10. import java.awt.event.ActionEvent;  
11. import java.awt.event.ActionListener;  
12. import java.awt.image.BufferedImage;  
13. import java.io.File;  
14. import java.sql.Connection;  
15. import java.sql.DriverManager;  
16. import java.sql.SQLException;  
17. import java.sql.Statement;  
18.  
19. class Login extends JFrame implements ActionListener {  
20.  
21.     // 定义组件  
22.     //String stu_num="adminView";  
23.     //String stu_pwd="123456";  
24.  
25.     static String identify = "登陆者身份";  
26.  
27.     JLabel jluser = null;//用户  
28.     JLabel jlpassword = null;//密码  
29.     JLabel jllogging=null;  
30.  
31.     JTextField jtfuser = null;//姓名  
32.     JTextField jtfpassword = null;//性别  
33.  
34.     JButton logings = null;//登录  
35.     JButton register = null;//登录  
36.     public void clear()  
37.     {  
38.         jtfuser.setText("");  
39.         jtfpassword.setText("");  
40.     }  
41.
```

```
42. // 构造函数
43. public Login() {
44.     // 创建组件
45.     jLlogin = new JLabel("高校科研管理系统");
46.     jLlogin.setForeground(Color.blue);
47.     jLlogin.setFont(new java.awt.Font("Dialog", 0, 25));
48.
49.     jLuser = new JLabel("用户:");
50.     jLpassword = new JLabel("密码:");
51.     jLuser.setFont(new java.awt.Font("Dialog", 0, 19));
52.     jLpassword.setFont(new java.awt.Font("Dialog", 0, 19));
53.
54.     jTFuser = new JTextField(10); // 用户名 TextField
55.     jTFpassword = new JTextField(10); // 密码 TextField
56.     jTFuser.setFont(new java.awt.Font("Dialog", 0, 19));
57.     jTFpassword.setFont(new java.awt.Font("Dialog", 0, 19));
58.     ;
59.     logins = new JButton("登录");
60.     logins.setFont(new java.awt.Font("Dialog", 1, 16));
61.     // register = new JButton("注册");
62.     // register.setFont(new java.awt.Font("Dialog", 1, 16));
63.
64.     JPanel jP1, jP2, jP3, jP4 = null;
65.     JPanel jPtop, jPbottom = null;
66.
67.     // 设置监听
68.     // logins.addActionListener();
69.     logins.addActionListener(this);
70.
71.     jP1 = new JPanel();
72.     jP2 = new JPanel();
73.     jP3 = new JPanel();
74.     jP4 = new JPanel();
75.
76.     jPtop = new JPanel();
77.     jPbottom = new JPanel();
78.
79.     jP1.add(jLlogin);
80.
81.     jP2.add(jLuser);
82.     jP2.add(jTFuser);
```

```
83.         jP2.setLayout(new FlowLayout(FlowLayout.CENTER));
84.         jP2.setPreferredSize(new Dimension(20,20));
85.
86.         jP3.add(jLpassword);
87.         jP3.add(jTFpassword);
88.         jP3.setLayout(new FlowLayout(FlowLayout.CENTER));
89.         jP3.setPreferredSize(new Dimension(20,20));
90.
91.         jP4.add(logings);
92.         //jP4.add(register);
93.
94.         jP4.setLayout(new FlowLayout(FlowLayout.CENTER));
95.
96.
97.         jPTop.setLayout(new GridLayout(4, 1));
98.         jPTop.add(jP1);
99.         jPTop.add(jP2);
100.        jPTop.add(jP3);
101.        jPTop.add(jP4);
102.
103.        this.add("North", jP1);
104.        this.add("Center", jP2);
105.        this.add("Center", jP3);
106.        this.add("Center", jP4);
107.
108.        /*
109.        BufferedImage img=null;
110.        try {
111.            img=ImageIO.read(new File("./img/bg.gif"));
112.        }catch (IOException e) {
113.            e.printStackTrace();
114.        }
115.        JLabel labl=new JLabel(new ImageIcon(img));
116.        getContentPane().add(labl);
117.        labl.setBounds(0, 0, img.getWidth(), img.getHeight());
118.        */
119.        this.setLayout(new GridLayout(4, 1));
120.        this.setTitle("高校科研管理系统登录");
121.        this.setSize(370,270);
122.        this.setLocation(555, 225);
123.        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
124.        this.setVisible(true);
125.        this.setResizable(false);
126.
```

```

127.         //dbProcess = new DbProcess();
128.     }
129.     @Override
130.     public void actionPerformed(ActionEvent arg0) {
131.         // TODO Auto-generated method stub
132.         //if(stu_num.equals(jTFuser.getText())&&stu_pwd.equals(jTF
        password.getText()))
133.
134.         String user = jTFuser.getText();
135.         String password = jTFpassword.getText();
136.
137.         String JDriver = "com.microsoft.sqlserver.jdbc.SQLServerDr
        iver";// SQL 数据库引擎
138.         String connectDB = "jdbc:sqlserver://127.0.0.1:1433;Databa
        seName=test";// 数据源 DatabaseName 是已经创建的数据库的名字 不是表的名
        字
139.
140.         //if((jTFuser.getText()).equals(admin_num) && (jTFpassword
        .getText()).equals(admin_pwd))
141.             //{
142.         //             if(user.equals("adminView") && password.equals
        ("123456")) {
143.         //                 Connection con = DriverManager.getConn
        ection(connectDB, user, password);
144.         //                 System.out.println("admin 连接数据库成功
        ");
145.         //
146.         //
147.         //                 Statement stmt = con.createStatement()
        ;// 创建 SQL 命令对象
148.         //
149.         //                 // 创建表
150.         //                 System.out.println("开始创建表");
151.         //                 String query = "create table TABLE1(ID N
        CHAR(2),NAME NCHAR(10))";// 创建表 SQL 语句
152.         //                 stmt.executeUpdate(query);// 执行 SQL 命令
        对象
153.         //                 System.out.println("表创建成功");
154.         //                 //创建新界面
155.         //                 //new Manager();
156.         //                 //this.setVisible(false);
157.         //
158.         //             }
159.         //         try {

```

```
160.                //确定登录的是 admin 还是教师
161.                if(jTFuser.getText().equals("adminView") && jT
                Fpassword.getText().equals("123456")) {
162.                    identify = "adminView";
163.                    //Connection con = DriverManager.getConnec
                tion(connectDB, user, password);// 连接数据库对象
164.                    System.out.println("admin 连接数据库成功
                ");
165.                    new Manager();
166.                    this.setVisible(false);
167.                    System.out.println("成功进入");
168.                    //Statement stmt = con.createStatement();/
                / 创建 SQL 命令对象
169.
170.                }
171.                else if(jTFuser.getText().equals("孙淼
                ") && jTFpassword.getText().equals("123456")) {
172.                    identify = "孙淼";
173.                    //Connection con = DriverManager.getConnec
                tion(connectDB, user, password);// 连接数据库对象
174.                    System.out.println("孙淼连接数据库成功");
175.                    new Manager();
176.                    this.setVisible(false);
177.                    System.out.println("成功进入");
178.                    //Statement stmt = con.createStatement();/
                / 创建 SQL 命令对象
179.
180.                }
181.                else if(jTFuser.getText().isEmpty()&&jTFpasswo
                rd.getText().isEmpty())
182.                {
183.                    JOptionPane.showMessageDialog(null,"请输入
                用户名和密码! ", "提示消息",JOptionPane.WARNING_MESSAGE);
184.                }else if(jTFuser.getText().isEmpty())
185.                {
186.                    JOptionPane.showMessageDialog(null,"请输入
                用户名! ", "提示消息",JOptionPane.WARNING_MESSAGE);
187.                }else if(jTFuser.getText().isEmpty())
188.                {
189.                    JOptionPane.showMessageDialog(null,"请输入
                密码! ", "提示消息",JOptionPane.WARNING_MESSAGE);
190.                }else
191.                {
```



```

192.                JOptionPane.showMessageDialog(null, "用户名
           或者密码错误! \n 请重新输入", "提示消息", JOptionPane.ERROR_MESSAGE);
193.                //清空输入框
194.                clear();
195.            }
196.
197.        //        }
198.        //        catch (SQLException e) {
199.        //            e.printStackTrace();
200.        //            System.out.println("数据库连接错误");
201.        //            System.exit(0);
202.        //        }
203.    }
204. }

```

Main. java

```

1. package Miao;
2.
3. import javax.imageio.ImageIO;
4. import javax.swing.*;
5. import java.awt.*;
6. import java.awt.event.ActionEvent;
7. import java.awt.event.ActionListener;
8. import java.awt.image.BufferedImage;
9. import java.io.File;
10. import java.sql.*;
11.
12. public class Main {
13.
14.     public ResultSet rs;//test
15.
16.     public static void main(String[] args) {
17.
18.         String JDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
           ;// SQL 数据库引擎
19.         String connectDB = "jdbc:sqlserver://127.0.0.1:1433;DatabaseNam
           e=test";// 数据源 DatabaseName 是已经创建的数据库的名字 不是表的名字
20.
21.         try {
22.             Class.forName(JDriver);// 加载数据库引擎,返回给定字符串名的类
23.         } catch (ClassNotFoundException e) {
24.             // e.printStackTrace();
25.             System.out.println("加载数据库引擎失败");
26.             System.exit(0);
27.         }

```

```

28.
29.     System.out.println("数据库驱动成功");
30.
31.     Login login = new Login();
32.
33. }
34. }

```

Manager. java

```

1. package Miao;
2.
3. import java.awt.*;
4. import javax.swing.*;
5. import javax.swing.event.TreeSelectionEvent;
6. import javax.swing.event.TreeSelectionListener;
7. import javax.swing.tree.*;
8. import java.awt.event.*;
9. import javax.imageio.ImageIO;
10.
11. public class Manager extends JFrame implements TreeSelectionListener
12. {
13.     JPanel jp;
14.     JSplitPane js;
15.     JScrollPane jsp;
16.     JTree tree;
17.     DefaultMutableTreeNode root, t1, t2, t1_1, t1_2, t1_3, t1_4, t1_5
18.     , t2_1,
19.     t2_2;
20.
21.     CardEmploy ae;//
22.
23.     public static void main(String[] args) {
24.         Manager manager = new Manager();
25.     }
26.
27.     public Manager() {
28.
29.         // 给树的各个结点赋值
30.         root = new DefaultMutableTreeNode("高校科研管理系统");
31.
32.         t1 = new DefaultMutableTreeNode("科研处");
33.         //t1_1 = new DefaultMutableTreeNode("科研项目管理");
34.         t1_2 = new DefaultMutableTreeNode("项目信息管理");
35.         //t1_3 = new DefaultMutableTreeNode("学生选课结果表");
36.         //t1_4 = new DefaultMutableTreeNode("删除员工资料");

```

```

35.         //t1_5 = new DefaultMutableTreeNode("查询全体员工");
36.
37.         t2 = new DefaultMutableTreeNode("人事处");
38.         //t2_1 = new DefaultMutableTreeNode("教师信息管理");
39.         t2_2 = new DefaultMutableTreeNode("教师信息管理");
40.
41.         //t1.add(t1_1);
42.         t1.add(t1_2);
43.         //t1.add(t1_3);
44.         //t1.add(t1_4);
45.         //t1.add(t1_5);
46.
47.         //t2.add(t2_1);
48.         t2.add(t2_2);
49.
50.         root.add(t1);
51.         root.add(t2);
52.
53.         tree = new JTree(root);
54.         // 对树进行监听
55.
56.         tree.addTreeSelectionListener(this);
57.
58.         // 实例化 CardEmploy 面板 并加到 jsplitpane 的边
59.         ae = new CardEmploy();
60.
61.         js = new JSplitPane();
62.         js.setLeftComponent(tree);
63.         js.setRightComponent(ae);
64.
65.         this.getContentPane().add(js);
66.         this.setTitle("高校科研管理系统");
67.         this.setVisible(true);
68.         this.setSize(900, 700);
69.         this.setResizable(false);
70.         this.setLocation(150, 150);
71.         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
72.     }
73.
74.     public void valueChanged(TreeSelectionEvent e) {
75.
76.         // 获取点击结点名称
77.         DefaultMutableTreeNode dpath = (DefaultMutableTreeNode) tree

```

```

78.         .getLastSelectedPathComponent();
79.
80.         // 通过点击不同结点切换不同界面
81.         if (dpath.equals(t1_1)) {
82.             ae.c.show(ae, "1");
83.         } else if (dpath.equals(t1_2)) {
84.             ae.c.show(ae, "2");
85.         } else if (dpath.equals(t1_3)) {
86.             ae.c.show(ae, "3");
87.         } else if (dpath.equals(t2_1)) {
88.             ae.c.show(ae, "4");
89.         } else if (dpath.equals(t2_2)) {
90.             ae.c.show(ae, "5");
91.         }
92.     }
93. }

```

scoer. java

```

1. package Miao;
2.
3. import javax.swing.*;
4. import javax.swing.table.DefaultTableModel;
5.
6. import java.awt.*;
7. import java.awt.event.ActionEvent;
8. import java.awt.event.ActionListener;
9. import java.awt.event.ItemEvent;
10. import java.awt.event.ItemListener;
11. import java.awt.event.MouseAdapter;
12. import java.awt.event.MouseEvent;
13. import java.sql.Connection;
14. import java.sql.DriverManager;
15. import java.sql.PreparedStatement;
16. import java.sql.ResultSet;
17. import java.sql.SQLException;
18. import java.sql.Statement;
19. import java.util.Vector;
20.
21. public class scoer extends Panel implements ActionListener {
22.     // 定义组件
23.     JLabel jlStudentInfoTable = null; // 学生信息表
24.     JLabel jlSelectQueryField = null; // 选择查询字段
25.     JLabel jlEqual = null; // =
26.     JLabel jlSNo = null; // 学号
27.     JLabel jlSName = null; // 姓名

```

```
28.    JLabel jLcourse = null;//课程号
29.    JLabel jLscoer = null;//成绩
30.
31.    JTextField jTFQueryField = null;//查询字段
32.    JTextField jTFSNo = null;//学号
33.    JTextField jTFSName = null;//姓名
34.    JTextField jTFcourse = null;//课程号
35.    JTextField jTFscoer = null;//成绩
36.
37.    //定义界面上的 button
38.    JButton jBQuery = null;//查询
39.    JButton jBQueryAll = null;//查询所有记录
40.    JButton jBInsert = null;//插入
41.    JButton jBUpdate = null;//更新
42.    JButton jBDeleteCurrentRecord = null;//删除当前记录
43.    JButton jBDeleteAllRecords = null;//删除所有记录
44.
45.    //JComboBox jCBSelectQueryField = null;
46.    //下拉框
47.    JComboBox<String> jCBSelectQueryField = null;//查询字段
48.    JPanel jP1, jP2, jP3, jP4, jP5 = null;
49.    JPanel jPTop, jPBottom = null;
50.    DefaultTableModel studentTableModel = null;
51.    JTable studentJTable = null;
52.    JScrollPane studentJScrollPane = null;
53.    Vector studentVector = null;
54.    Vector titleVector = null;
55.
56.    private static DbProcess dbProcess;
57.
58.    String SelectQueryFieldStr = "项目编号";
59.
60.    // 构造函数
61.    public scoer() {
62.        // 创建组件
63.        jLStudentInfoTable = new JLabel("科研项目信息");
64.        jLSelectQueryField = new JLabel("请通过项目编号查找");
65.        jLEqual = new JLabel(" = ");
66.        jLSNo = new JLabel("项目编号");
67.        jLSName = new JLabel("项目名称");
68.        jLcourse = new JLabel("验收情况");//课程号
69.        jLscoer = new JLabel("教师号");//成绩
70.
71.        jTFQueryField = new JTextField(10);//查询字段
```

```

72.         jTFSNo = new JTextField(18);//学号
73.         jTFSName = new JTextField(18);//姓名
74.         jTFcourse = new JTextField(18);//课程号
75.         jTFScoer = new JTextField(18);//成绩
76.
77.         jBQuery = new JButton("查询");
78.         jBQueryAll = new JButton("查询所有记录");
79.         jBInsert = new JButton("插入");
80.         jBUpdate = new JButton("更新");
81.         jBDeleteCurrentRecord = new JButton("删除当前记录");
82.         jBDeleteAllRecords = new JButton("删除所有记录");
83.         // 设置监听
84.         jBQuery.addActionListener(this);
85.         jBQueryAll.addActionListener(this);
86.         jBInsert.addActionListener(this);
87.         jBUpdate.addActionListener(this);
88.         jBDeleteCurrentRecord.addActionListener(this);
89.         jBDeleteAllRecords.addActionListener(this);
90.
91.         jCBSelectQueryField = new JComboBox<String>();//查询字段
92.         jCBSelectQueryField.addItem("项目号");
93.         jCBSelectQueryField.addItem("项目名");
94.         jCBSelectQueryField.addItem("验收情况");
95.         jCBSelectQueryField.addItem("教师号");
96.
97. //         jCBSelectQueryField.addItemListener(new ItemListener() { //下拉
           框事件监听
98. //             public void itemStateChanged(ItemEvent event) {
99. //                 switch (event.getStateChange()) {
100. //                     case ItemEvent.SELECTED:
101. //                         SelectQueryFieldStr = (String) event.getItem()
102. //                         ;
103. //                         System.out.println("选中:
104. //                         " + SelectQueryFieldStr);
105. //                         break;
106. //                     case ItemEvent.DESELECTED:
107. //                         System.out.println("取消选中:
108. //                         " + event.getItem());
109. //                         break;
110. //                     }
111. //                 }
112. //             });
113.
114.         studentVector = new Vector();

```

```
112.         titleVector = new Vector();
113.
114.         // 定义表头
115.         titleVector.add("项目号");
116.         titleVector.add("项目名");
117.         titleVector.add("验收情况");
118.         titleVector.add("教师号");
119.         //studentTableModel = new DefaultTableModel(tableTitle, 15
120.         );
121.         studentJTable = new JTable(studentVector, titleVector);
122.         studentJTable.setPreferredScrollableViewportSize(new Dimen
123.         sion(600,260));
124.         studentJScrollPane = new JScrollPane(studentJTable);
125.         //分别设置水平和垂直滚动条自动出现
126.         studentJScrollPane.setHorizontalScrollBarPolicy(
127.         JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
128.         studentJScrollPane.setVerticalScrollBarPolicy(
129.         JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
130.         //为表格添加监听器
131.         studentJTable.addMouseListener(new MouseAdapter()
132.         {
133.             public void mouseClicked(MouseEvent e)
134.             {
135.                 int row = ((JTable) e.getSource()).rowAtPoint(e.ge
136.                 tPoint()); // 获得行位置
137.                 System.out.println("mouseClicked(). row = " + row)
138.                 ;
139.                 Vector v = new Vector();
140.                 v = (Vector) studentVector.get(row);
141.                 jTFSNo.setText((String) v.get(0)); // 学号
142.                 jTFSName.setText((String) v.get(1)); // 姓名
143.                 jTFcourse.setText((String) v.get(2)); // 课程号
144.                 jTFscoer.setText((String) v.get(4)); // 成绩号
145.             }
146.         });
147.
148.         jP1 = new JPanel();
149.         jP2 = new JPanel();
150.         jP5 = new JPanel();
```

```
150.         jP3 = new JPanel();
151.         jP4 = new JPanel();
152.         jPTop = new JPanel();
153.         jPBottom = new JPanel();
154.
155.         jP1.add(jLStudentInfoTable, BorderLayout.SOUTH);
156.         jP2.add(studentJScrollPane);
157.
158.
159.         jP3.add(jLSelectQueryField);    //选择查询字段
160.         //jP3.add(jCBSelectQueryField);    //查询字段
161.         //jP3.add(jLEqual);    //=
162.         jP3.add(jTFQueryField);
163.         jP3.add(jBQuery);
164.         jP3.add(jBQueryAll);
165.         jP3.setLayout(new FlowLayout(FlowLayout.CENTER));
166.         jP3.setPreferredSize(new Dimension(20,20));
167.
168.         jP4.add(jLSNo);
169.         jP4.add(jTFSNo);
170.         jP4.add(jLSName);
171.         jP4.add(jTFName);
172.         jP4.add(jLcourse);
173.         jP4.add(jTFcourse);
174.         jP4.add(jLscoer);
175.         jP4.add(jTFscoer);
176.
177.         jP4.setLayout(new FlowLayout(FlowLayout.CENTER));
178.         jP4.setPreferredSize(new Dimension(30,30));
179.
180.
181.         jP5.add(jBInsert);
182.         jP5.add(jBUpdate);
183.         jP5.add(jBDeleteCurrentRecord);
184.         jP5.add(jBDeleteAllRecords);
185.         jP5.setLayout(new FlowLayout(FlowLayout.CENTER));
186.         jP5.setPreferredSize(new Dimension(20,20));
187.
188.         jPTop.add(jP1);
189.         jPTop.add(jP2);
190.
191.         jPBottom.setLayout(new GridLayout(3, 1));
192.         jPBottom.add(jP3);
193.         jPBottom.add(jP4);
```



```
194.         jPBottom.add(jP5);
195.         this.add("North", jPTop);
196.         this.add("South", jPBottom);
197.
198.         this.setLayout(new GridLayout(2, 1));
199.         //this.setTitle("成绩信息操作");
200.         this.setSize(530, 500);
201.         this.setLocation(150, 150);
202.         //this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
203.         this.setVisible(true);
204.         //this.setResizable(false);
205.
206.         dbProcess = new DbProcess();
207.     }
208.
209.     @Override
210.     public void actionPerformed(ActionEvent e) {
211.         if(e.getActionCommand().equals("查询")
212.             && !jTFQueryField.getText().isEmpty()){
213.             System.out.println("actionPerformed(). 查询");
214.             String sQueryField = jTFQueryField.getText().trim(
215.                 );
216.             queryProcess(sQueryField);
217.             jTFQueryField.setText("");
218.         }
219.         else if(e.getActionCommand().equals("查询所有记录")) {
220.             System.out.println("actionPerformed(). 查询所有记录
221. ");
222.             queryAllProcess();
223.         }
224.         else if(e.getActionCommand().equals("插入")
225.             && !jTFSNo.getText().isEmpty()
226.             && !jTFSName.getText().isEmpty()
227.             && !jTFscoer.getText().isEmpty()){
228.             System.out.println("actionPerformed(). 插入");
229.             insertProcess();
230.         }
231.     }
232.
233.     else if(e.getActionCommand().equals("更新")
234.         && !jTFSNo.getText().isEmpty()
235.         && !jTFcourse.getText().isEmpty()
```

```
236.         ){
237.             System.out.println("actionPerformed(). 更新");
238.             updateProcess();//验收
239.         }
240.
241.         //     else if(e.getActionCommand().equals("删除当前记录")){
242.         //         System.out.println("actionPerformed(). 删除当前记录
243.         //         ");
244.         //         deleteCurrentRecordProcess();
245.         //     }
246.
247.         /*
248.
249.         public static void main(String[] args) {
250.             scoer getcon = new scoer();
251.         }
252.
253.         */
254.
255.         public static String temp_proNo;
256.
257.         public void queryProcess(String sQueryField)
258.         {
259.             temp_proNo = sQueryField;
260.
261.             dbProcess.pro_queryProcess();
262.
263.             studentVector.clear();
264.
265.             while (!dbProcess.list.isEmpty()) {
266.
267.                 //System.out.print(dbProcess.list.poll());
268.                 Vector v = new Vector();
269.                 v.add(dbProcess.list.poll());
270.                 v.add(dbProcess.list.poll());
271.                 v.add(dbProcess.list.poll());
272.                 v.add(dbProcess.list.poll());
273.                 studentVector.add(v);
274.             }
275.
276.             studentJTable.updateUI();
277.
278.             dbProcess.disconnect();
```

```
279.     }
280.
281.     public static String sum = "";
282.     public static String checked = "";
283.
284.     public void queryAllProcess()
285.     {
286.         dbProcess.pro_queryAllProcess();
287.
288.         studentVector.clear();
289.
290.         sum = dbProcess.list.size() / 4 + "";
291.         checked = dbProcess.list_sum.size() + "";
292.         dbProcess.list_sum.clear();
293.
294.         jTfcourse.setText(checked + "/" + sum);
295.
296.         while (!dbProcess.list.isEmpty()) {
297.
298.             //System.out.print(dbProcess.list.poll());
299.             Vector v = new Vector();
300.             v.add(dbProcess.list.poll());
301.             v.add(dbProcess.list.poll());
302.             v.add(dbProcess.list.poll());
303.             v.add(dbProcess.list.poll());
304.             studentVector.add(v);
305.         }
306.         //
307.         studentJTable.updateUI();
308.
309.         System.out.println("读取完毕");
310.
311.         dbProcess.disconnect();
312.     }
313.
314.     //public static String temp_proNo;
315.     public static String temp_proCK;
316.
317.     public void updateProcess()
318.     {
319.         temp_proNo = jTFSNo.getText().trim();
320.         temp_proCK = jTfcourse.getText().trim();
321.
322.         dbProcess.pro_updataProcess();
```

```

323.
324.         studentVector.clear();
325.
326.         dbProcess.disconnect();
327.     }
328.
329.     public static String temp_proName;
330.     public static String temp_teaNo;
331.     public void insertProcess()
332.     {
333.         System.out.println("准备申报");
334.
335.         temp_proNo = jTFSNo.getText().trim();
336.         temp_proName = jTFSName.getText().trim();
337.         temp_proCK = jTFcourse.getText().trim();
338.         temp_teaNo = jTFscoer.getText().trim();
339.
340.         dbProcess.pro_insertProcess();
341.
342.         System.out.println("申报成功");
343.
344.         studentVector.clear();
345.
346.         dbProcess.disconnect();
347.     }
348.
349. }

```

teacher_couse_aet. java.

```

1. package Miao;
2.
3. import javax.swing.*;
4. import javax.swing.table.DefaultTableModel;
5.
6. import java.awt.*;
7. import java.awt.event.ActionEvent;
8. import java.awt.event.ActionListener;
9. import java.awt.event.ItemEvent;
10. import java.awt.event.ItemListener;
11. import java.awt.event.MouseAdapter;
12. import java.awt.event.MouseEvent;
13. import java.sql.Connection;
14. import java.sql.DriverManager;
15. import java.sql.PreparedStatement;
16. import java.sql.ResultSet;

```

```

17. import java.sql.SQLException;
18. import java.util.Vector;
19.
20. public class teacher_couse_aet extends Panel implements ActionListener
    {
21.     // 定义组件
22.     JLabel jlStudentInfoTable = null;//学生信息表
23.     JLabel jlSelectQueryField = null;//选择查询字段
24.     //JLabel jlEqual = null;//=
25.     JLabel jlteacher = null;//学号
26.     JLabel jlScourse = null;//姓名
27.     JLabel jlJidian = null;//班级
28.     JLabel jlclassroom = null;//班级
29.     JLabel jlWe_day = null;//班级
30.     JLabel jlclasnumer = null;//班级
31.
32.     JTextField jtteacher = null;//查询字段
33.     JTextField jtScourse = null;//学号
34.     JTextField jtJidian = null;//姓名
35.     JTextField jtclassroom = null;//班级
36.     JTextField jtWe_day = null;//班级
37.     JTextField jtclasnumer = null;//班级
38.     JTextField jtFQueryField=null;
39.
40.     //定义界面上的 button
41.     JButton jbQuery = null;//查询
42.     JButton jbQueryAll = null;//查询所有记录
43.     JButton jbInsert = null;//插入
44.     JButton jbUpdate = null;//更新
45.     JButton jbDeleteCurrentRecord = null;//删除当前记录
46.     //JButton jbDeleteAllRecords = null;//删除所有记录
47.
48.     //JComboBox jCBSelectQueryField = null;
49.     //下拉框
50.     JComboBox<String> jCBSelectQueryField = null;//查询字段
51.     JPanel jP1, jP2, jP3, jP4, jP5, jP6 = null;
52.     JPanel jPTop, jPBottom = null;
53.     DefaultTableModel studentTableModel = null;
54.     JTable studentJTable = null;
55.     JScrollPane studentJScrollPane = null;
56.     Vector studentVector = null;
57.     Vector titleVector = null;
58.
59.     private static DbProcess dbProcess;

```

```
60.     String SelectQueryFieldStr = "学号";
61.
62.     // 构造函数
63.     public teacher_couse_aet() {
64.         // 创建组件
65.         jlStudentInfoTable = new JLabel("教师信息");
66.         jlSelectQueryField = new JLabel("请通过教师号查询");
67.         //jlEqual = new JLabel(" = ");
68.         jlteacher = new JLabel("教师号");
69.         jlScourse = new JLabel("教师名");
70.         jlJidian = new JLabel("部门");
71.         jlclassroom = new JLabel("职务");
72.         jlwe_day = new JLabel("职称");
73.         jlclasnumber = new JLabel("性别");
74.
75.
76.         jTteacher = new JTextField(10);//查询字段
77.         jTScourse = new JTextField(10);//学号
78.         jTjidian = new JTextField(10);//姓名
79.         jTclassroom = new JTextField(10);//性别
80.         jTwe_day = new JTextField(10);//性别
81.         jTclasnumber = new JTextField(10);//性别
82.         jTFQueryField=new JTextField(10);
83.
84.         JBQuery = new JButton("查询");
85.         JBQueryAll = new JButton("查询所有记录");
86.         JBInsert = new JButton("插入");
87.         JBUpdate = new JButton("更新");
88.         JBDeleteCurrentRecord = new JButton("删除当前记录");
89.         //JBDeleteAllRecords = new JButton("删除所有记录");
90.         // 设置监听
91.         JBQuery.addActionListener(this);
92.         JBQueryAll.addActionListener(this);
93.         JBInsert.addActionListener(this);
94.         JBUpdate.addActionListener(this);
95.         JBDeleteCurrentRecord.addActionListener(this);
96.         //JBDeleteAllRecords.addActionListener(this);
97.
98.         jCBSelectQueryField = new JComboBox<String>();//查询字段
99.         jCBSelectQueryField.addItem("教师号");
100.        jCBSelectQueryField.addItem("教师名");
101.        jCBSelectQueryField.addItem("部门");
102.        jCBSelectQueryField.addItem("职务");
103.        jCBSelectQueryField.addItem("职称");
```

```
104.         jCBSelectQueryField.addItem("性别");
105. //         jCBSelectQueryField.addItemListener(new ItemListener() {
106. //             public void itemStateChanged(ItemEvent event) {
107. //                 switch (event.getStateChange()) {
108. //                     case ItemEvent.SELECTED:
109. //                         SelectQueryFieldStr = (String) event.getItem()
110. //                         ;
111. //                         System.out.println("选中:
112. //                         " + SelectQueryFieldStr);
113. //                         break;
114. //                     case ItemEvent.DESELECTED:
115. //                         System.out.println("取消选中:
116. //                         " + event.getItem());
117. //                         break;
118. //                     }
119. //                 }
120. //             });
121.
122.         studentVector = new Vector();
123.         titleVector = new Vector();
124.
125.         // 定义表头
126.         titleVector.add("教师号");
127.         titleVector.add("教师名");
128.         titleVector.add("部门");
129.         titleVector.add("职务");
130.         titleVector.add("职称");
131.         titleVector.add("性别");
132.         //studentTableModel = new DefaultTableModel(tableTitle, 15
133.         );
134.         studentJTable = new JTable(studentVector, titleVector);
135.         studentJTable.setPreferredScrollableViewportSize(new Dimen
136.             sion(600,200));
137.         studentJScrollPane = new JScrollPane(studentJTable);
138.         //分别设置水平和垂直滚动条自动出现
139.         studentJScrollPane.setHorizontalScrollBarPolicy(
140.
141.             JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
142.         studentJScrollPane.setVerticalScrollBarPolicy(
143.
144.             JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
145.
146.         //为表格添加监听器
```

```
140.         studentJTable.addMouseListener(new MouseAdapter()  
141.         {  
142.             public void mouseClicked(MouseEvent e)  
143.             {  
144.                 int row = ((JTable) e.getSource()).rowAtPoint(e.ge  
tPoint()); // 获得行位置  
145.                 System.out.println("mouseClicked(). row = " + row)  
146.                 ;  
147.                 Vector v = new Vector();  
148.                 v = (Vector) studentVector.get(row);  
149.                 jTteacher.setText((String) v.get(0)); // 学号  
150.                 jTscourse.setText((String) v.get(1));  
151.                 jTjidian.setText((String) v.get(2)); // 姓名  
152.                 jTclassroom.setText((String) v.get(3)); // 班级  
153.                 jTwe_day.setText((String) v.get(4));  
154.                 jTclasnumber.setText((String) v.get(5));  
155.  
156.             }  
157.         });  
158.  
159.  
160.         jP1 = new JPanel();  
161.         jP2 = new JPanel();  
162.         jP5 = new JPanel();  
163.         jP3 = new JPanel();  
164.         jP4 = new JPanel();  
165.         jP5 = new JPanel();  
166.         jP6 = new JPanel();  
167.  
168.         jPTop = new JPanel();  
169.         jPBottom = new JPanel();  
170.  
171.         jP1.add(jLStudentInfoTable, BorderLayout.SOUTH);  
172.         jP2.add(studentJScrollPane);  
173.  
174.  
175.         jP3.add(jLSelectQueryField); //选择查询字段  
176.         jP3.add(jCSelectQueryField); //查询字段  
177.         //jP3.add(jLEqual); //=  
178.         jP3.add(jTFQueryField);  
179.         jP3.add(jBQuery);  
180.         jP3.add(jBQueryAll);  
181.         jP3.setLayout(new FlowLayout(FlowLayout.CENTER));
```



```
182.         jP3.setPreferredSize(new Dimension(20,20));
183.
184.         jP4.add(jLteacher);
185.         jP4.add(jTteacher);
186.         jP4.add(jLScourse);
187.         jP4.add(jTScourse);
188.         jP4.add(jLjidian);
189.         jP4.add(jTjidian);
190.         jP4.setLayout(new FlowLayout(FlowLayout.CENTER));
191.         jP4.setPreferredSize(new Dimension(30,30));
192.         jP5.add(jLclassroom);
193.         jP5.add(jTclassroom);
194.         jP5.add(jLwe_day);
195.         jP5.add(jTwe_day);
196.         jP5.add(jLclasnumber);
197.         jP5.add(jTclasnumber);
198.         jP5.setLayout(new FlowLayout(FlowLayout.CENTER));
199.         jP5.setPreferredSize(new Dimension(20,20));
200.
201.
202.         jP6.add(jBInsert);
203.         jP6.add(jBUpdate);
204.         jP6.add(jBDeleteCurrentRecord);
205.         //jP6.add(jBDeleteAllRecords);
206.         jP6.setLayout(new FlowLayout(FlowLayout.CENTER));
207.         jP6.setPreferredSize(new Dimension(20,20));
208.
209.         jPTop.add(jP1);
210.         jPTop.add(jP2);
211.
212.         jPBottom.setLayout(new GridLayout(4, 1));
213.         jPBottom.add(jP3);
214.         jPBottom.add(jP4);
215.         jPBottom.add(jP5);
216.         jPBottom.add(jP6);
217.         this.add("North", jPTop);
218.         this.add("South", jPBottom);
219.
220.
221.
222.         this.setLayout(new GridLayout(2, 1));
223.         //this.setTitle("教室课程设置表");
224.         this.setSize(580, 500);
225.         this.setLocation(150, 150);
```

```

226.         //this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
227.         this.setVisible(true);
228.         //this.setResizable(false);
229.
230.
231.         dbProcess = new DbProcess();
232.     }
233.
234.     @Override
235.     public void actionPerformed(ActionEvent e) {
236.         if(e.getActionCommand().equals("查询")
237.             && !jTFQueryField.getText().isEmpty()){
238.             System.out.println("actionPerformed().查询");
239.             String sQueryField = jTFQueryField.getText().trim(
240.             );
241.             queryProcess(sQueryField);
242.             jTFQueryField.setText("");
243.             }else if(e.getActionCommand().equals("查询所有记录
244.             ")) {
245.                 System.out.println("actionPerformed(). 查询所有记录
246.                 ");
247.                 queryAllProcess();
248.             }
249.             // else if(e.getActionCommand().equals("插入")
250.             //         && !jTteacher.getText().isEmpty()
251.             //         && !jTScourse.getText().isEmpty()
252.             //         && !jTjidian.getText().isEmpty()
253.             //         && !jTclassroom.getText().isEmpty()
254.             //         && !jTwe_day.getText().isEmpty()
255.             //         && !jTclasnumer.getText().isEmpty()){
256.             //         System.out.println("actionPerformed(). 插入");
257.             //         insertProcess();
258.             //     }
259.         }
260.
261.
262.         /*
263.         public static void main(String[] args) {
264.             teacher_couse_aet getcon = new teacher_couse_aet();
265.         }
266.         */
267.         public static String temp_teaNo;
268.
269.         public void queryProcess(String jTFQueryField)

```

```
267.     {
268.
269.         temp_teaNo = jTFQueryField;
270.
271.         dbProcess.tea_queryProcess();
272.
273.         studentVector.clear();
274.
275.         while (!dbProcess.list.isEmpty()) {
276.
277.             //System.out.print(dbProcess.list.poll());
278.             Vector v = new Vector();
279.             v.add(dbProcess.list.poll());
280.             v.add(dbProcess.list.poll());
281.             v.add(dbProcess.list.poll());
282.             v.add(dbProcess.list.poll());
283.             v.add(dbProcess.list.poll());
284.             v.add(dbProcess.list.poll());
285.             studentVector.add(v);
286.         }
287.
288.         studentJTable.updateUI();
289.         dbProcess.disconnect();
290.     }
291.
292.     public void queryAllProcess()
293.     {
294.         dbProcess.tea_queryAllProcess();
295.
296.         studentVector.clear();
297.
298.         while (!dbProcess.list.isEmpty()) {
299.
300.             Vector v = new Vector();
301.             v.add(dbProcess.list.poll());
302.             v.add(dbProcess.list.poll());
303.             v.add(dbProcess.list.poll());
304.             v.add(dbProcess.list.poll());
305.             v.add(dbProcess.list.poll());
306.             v.add(dbProcess.list.poll());
307.             studentVector.add(v);
308.         }
309.
310.         studentJTable.updateUI();
```

```
311.
312.         System.out.println("读取完毕");
313.
314.         dbProcess.disconnect();
315.     }
316.
317.     public String jCBSelectQueryFieldTransfer(String InputStr)
318.     {
319.         String outputStr = "";
320.         System.out.println("jCBSelectQueryFieldTransfer(). InputStr = " + InputStr);
321.
322.         if(InputStr.equals("教师")){
323.             outputStr = "教师";
324.         }else if(InputStr.equals("课程名")){
325.             outputStr = "课程名";
326.         }else if(InputStr.equals("学分绩点")){
327.             outputStr = "学分绩点";
328.         }
329.         else if(InputStr.equals("上课教室")){
330.             outputStr = "上课教室";
331.         }
332.         else if(InputStr.equals("上课周次")){
333.             outputStr = "上课周次";
334.         }
335.         else if(InputStr.equals("节次")){
336.             outputStr = "节次";
337.         }
338.         System.out.println("jCBSelectQueryFieldTransfer(). outputStr = " + outputStr);
339.         return outputStr;
340.     }
341. }
```