

**《数据库设计与开发》**

**课程设计**

**题 目 企业人员管理系统**

姓 名

学 号

院 系

专 业

**二Ｏ一九 年 月 日**

目录

[一 概述 1](#_Toc13576983)

[1.1目的与要求 1](#_Toc13576984)

[1.2设计环境 2](#_Toc13576985)

[二 需求分析 2](#_Toc13576986)

[2.1 系统功能要求设计 2](#_Toc13576987)

[2.2 系统模块设计 2](#_Toc13576988)

[三 总体设计 3](#_Toc13576989)

[3.1 数据字典 3](#_Toc13576990)

[3.2系统E—R图转化而得到的关系模式如下： 4](#_Toc13576991)

[3.3 确定关系模型的存取方法 4](#_Toc13576992)

[3.4 确定数据库的存储结构 5](#_Toc13576993)

[五 系统实现 7](#_Toc13577002)

[5.1 数据库的实施： 7](#_Toc13577003)

[六 功能测试 11](#_Toc13577004)

[6.1部分操作截图： 11](#_Toc13577005)

[6.3数据库的维护 14](#_Toc13577007)

[七 安全性测试 14](#_Toc13577008)

[八 总结 14](#_Toc13577009)

# 一 概述

1.1目的与要求

随着科技的发展，基本上所有的具有一定数量数据的机构都开始使用计算机数据库来做管理。几乎所有企业也都已经在使用计算机管理数据的机制，大大减少了企业人员管理的工作量。该课程设计要求设计一个企业人员的数据库管理系统，数据库中要求包含企业员工的的基本信息，登陆密码等一系列内容。要方便员工进行信息查询，通过该课程设计，应该达到把数据库理论知识更加的巩固加深，加强动手能力与实践能力，学以致用，与现实生活中的应用充分的结合起来。

1.2设计环境

① MySQL WorkBench

② eclipse

# 二 需求分析

2.1 系统功能要求设计

此系统实现如下系统功能：

**（**1）使得员工的管理工作更加清晰、条理化、自动化。

（2）登陆功能：获取页面上 账号和密码，判断用户输入的是不是空的。如果不是空的，就发送ajax请求，验证用户名和密码是否正确；正确就跳转页面；不正确就提示用户，账号或者密码错误。

(3) 管理登陆成功之后判断 账号级别（S,A）。如果是S级，超级管理员，就在右上角增加2个功能按钮（添加管理，角色管理），如果是A级，普通管理员，就没有这个2个功能。

（4）个人信息：点击页面上个人信息按钮，可以显示管理员个人的信息（admininfo）管理员可以对个人信息进行修改，并保存更新到empinfo表中。

（5）密码修改：输入旧密码，输入2次新密码，对比2次新密码是否一致。旧密码先拿后台书数据库对比，是否正确。输入错误，就提示用户旧密码输入错误；输入正确，就可以将旧密码改成新密码。

（6）创建新用户：点击页面左边的创建用户，填写相关信息，ID是序列生成，姓名，年龄，性别，身份证号，手机号，地址。判断用户是否是新用户，使用身份证号去用户表(userInfo)查询判断。如果查询到信息，就说明不是新用户；如果没有查询到信息，就可以注册为新用户。

* 1. 系统模块设计

人员管理系统大体可以分成人员信息以及人员职务，还涉及人员信息变动，职务变动，优先级的更改等操作。根据分析，可以得到系统流程图：

用户

提示错误

个人信息查询

信息修改

管理员设置

登录失败

登陆成功

创建新用户

级别管理



# 三 总体设计

3.1 数据字典

数据项是数据库的关系中不可再分的数据单位，下表分别列出了数据的名称、数据类型、长度、取值能否为空。利用MySQL建立testa数据库，其基本表清单及表结构描述如下：

数据库中用到的表：

|  |  |
| --- | --- |
| 数据库表名 | 关系模式名称 |
| dx\_admin | 管理员 |
| empinfo | 员工 |
| userinfo | 用户 |

dx\_admin基本情况数据表，结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | Not Null | 说明 |
| id | int | Primary key | id |
| account | Varchar(20) | Not Null | 登录 |
| password | Varchar(20) |  | 密码 |
| Level | Varchar(20) |  | 级别 |
| phone | Varchar(13) |  | 电话 |

empinfo数据表，结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| id | Int | 主键（primary key） | id |
| name | Varchar(20) | not null | 姓名 |
| idcard | Varchar(20) | not null | 身份证 |
| age | Varchar(20) | not null | 年龄 |
| account | Varchar(20) |  | 登录 |
| phone | Varchar(13) |  | 电话 |
| Address | Varchar(50) |  | 地址 |

userinfo情况数据表，结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| id | int | 主键 | ID |
| name | Varchar(20) |  | 姓名 |
| age | Varchar(20) |  | 年龄 |
| sex | Varchar(20) | ‘男’或’女’ | 性别 |
| idcard | Varchar(20) |  | 身份证 |
| phone | Varchar(13) |  | 电话 |
| Address | Varchar(50) |  | 地址 |

3.2系统E—R图转化而得到的关系模式如下：

（1）dx\_admin(id, account, password , level, phone),主键时id

（2）empinfo(员工信息表)(id, name, idcard, age, account, phone, address),主键为id

（3）userinfo(用户表)(id, name, age, sex, idcard, phone, address),主键为id

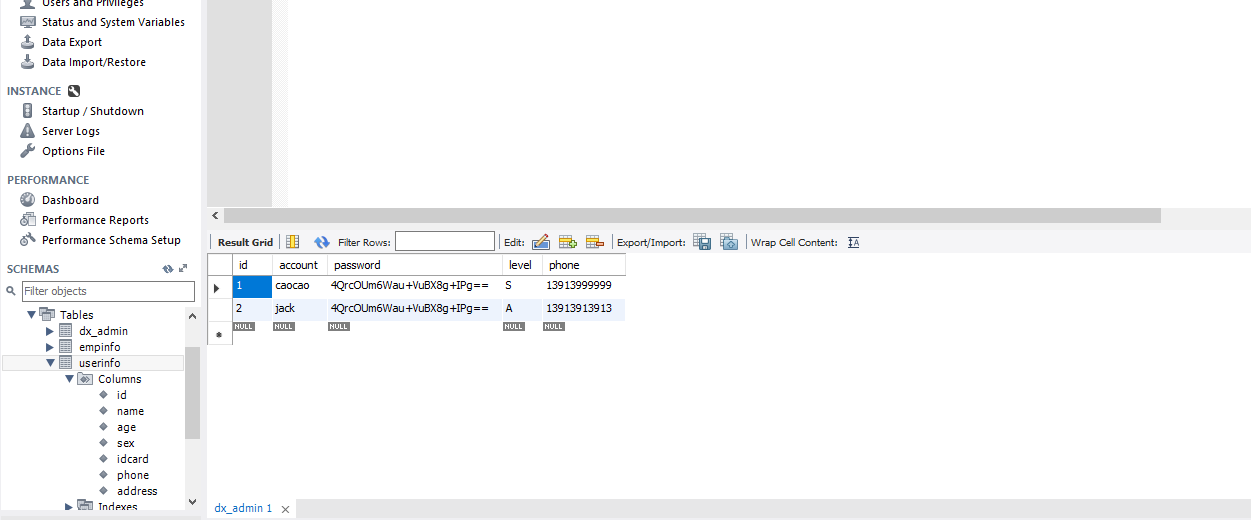
3.3 确定关系模型的存取方法

在将概念模型转换成物理模型之后，我们可以对物理模型进行设计，双击物理模型的关系，可以对该关系的名称、注释等信息进行查询。可对该关系的属性列进行设计，可分别设置其名称、码、数据类型以及主码、是否为空等。在实际设计中最常用的存取方法是索引发，使用索引可以大大减少数据的查询时间，在建立索引时应遵循：在经常需要搜索的列上建立索引；在主关键字上建立索引；在经常用于连接的列上建立索引，即在外键上建立索引；在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的等规则。才能充分利用索引的作用避免因索引引起的负面作用。

3.4 确定数据库的存储结构

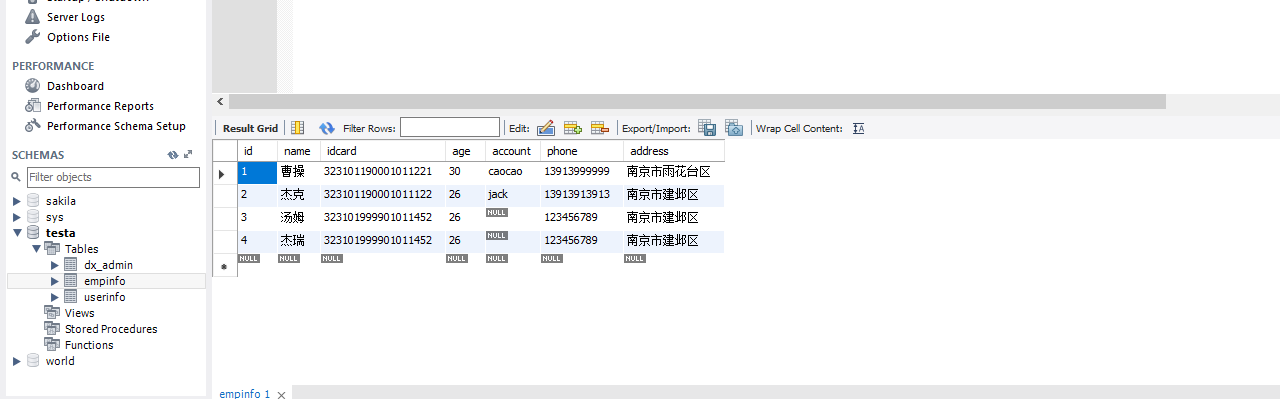
确定数据库的存储结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、日志、备份等的存储安排及存储结构，以及确定系统存储参数的配置。因为该人员管理系统的数据量小，，所以我们只用把数据存储在使用的电脑硬盘上，不用作更多的安排。

1. 创建dx\_admin：



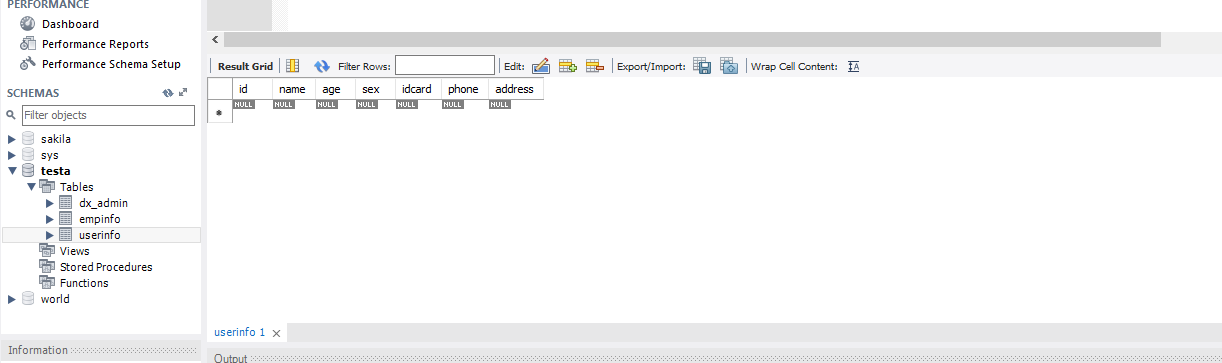
（图4-1）

(2)创建empinfo(员工信息表):



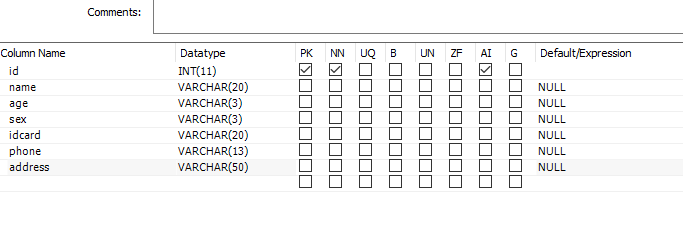
（图4-2）

（3）创建userinfo(用户表)



（图4-3）

下图显示了关系的主键：



（图4-4）

# 四 数据库结构示意图

由需求分析的结果可知，本系统设计的实体包括：

（1）管理员：id，account，password，phone，level。

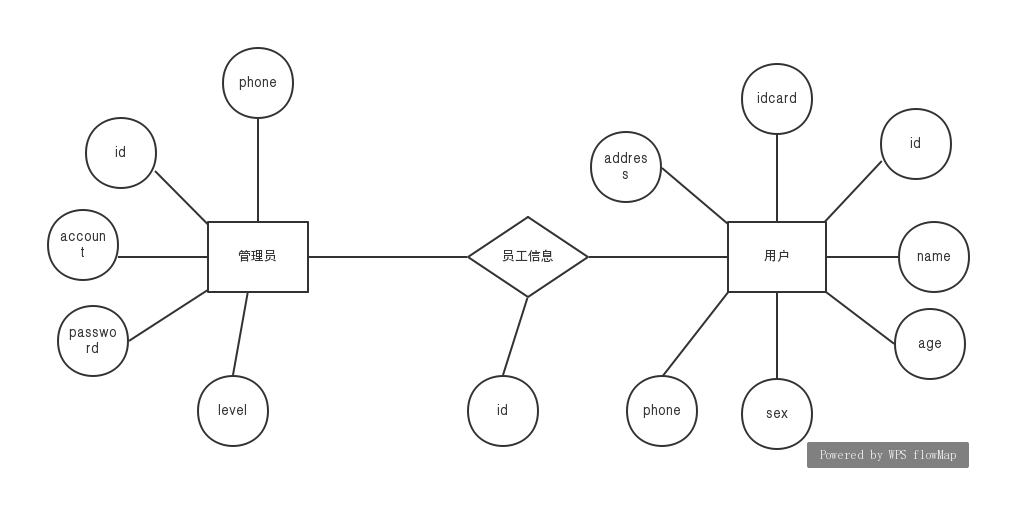
（2）员工个人信息： id，name，idcard,age,account,phone,address。

(3)用户：id,name,age,idcard,address,phone,sex.

这些实体间的联系包括：

（1）每位用户只有一个id,可以有多个管理员。

由上述分析可得到系统的E—R图：



（图4-1）

# 五 系统实现

5.1 数据库的实施：

此阶段主要任务包括创建数据库，加载初始数据，数据库试运行，数据库的安全性和完整性控制数据库的备份与恢复，数据库性能的监督分析和改仅，数据库的重组和重构等。首先在数据库中建立一个系统数据库，然后新建一个数据源。

主要代码如下：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"

"http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">

<mapper namespace=*"org.jscucc.cn.dao.AdminDao"*>

<cache/>

<select id=*"checkLogin"*

parameterType=*"string"*

resultType=*"org.jscucc.cn.entity.Admin"*>

select \* from dx\_admin

where account=#{0}

and password=#{1}

</select>

<select id=*"findByIdAdmin"*

parameterType=*"int"*

resultType=*"org.jscucc.cn.entity.Admin"*>

select \* from dx\_admin

where id=#{id}

</select>

<select id=*"findAdminAll"*

resultType=*"org.jscucc.cn.entity.Admin"*>

select \* from dx\_admin

</select>

<select id=*"findEmpInfo"*

parameterType=*"string"*

resultType=*"org.jscucc.cn.entity.EmpInfo"*>

select \* from empinfo

where account=#{account}

</select>

<update id=*"updatePwd"* parameterType=*"string"*>

update dx\_admin set password=#{0} where account=#{1}

</update>

<update id=*"updateEmpInfo"* parameterType=*"org.jscucc.cn.entity.EmpInfo"*>

update empinfo set name=#{name},idcard=#{idcard},age=#{age},phone=#{phone},address=#{address}

where account=#{account}

</update>

<select id=*"findByPhone"* parameterType=*"string"* resultType=*"org.jscucc.cn.entity.EmpInfo"*>

select \* from empinfo where phone=#{phone}

</select >

<select id=*"findByAccount"* parameterType=*"string"* resultType=*"org.jscucc.cn.entity.Admin"*>

select \* from dx\_admin where account=#{account}

</select>

<insert id=*"addAdmin"* parameterType=*"org.jscucc.cn.entity.Admin"*>

insert into dx\_admin(account,password,leve,phone)

values(#{account},#{password},#{leve},#{phone})

</insert>

<update id=*"updateAccount"* parameterType=*"org.jscucc.cn.entity.Admin"*>

update empinfo set account=#{account} where phone=#{phone}

</update>

<update id=*"updateAdmin"* parameterType=*"org.jscucc.cn.entity.Admin"*>

update dx\_admin set account=#{account},leve=#{leve},phone=#{phone}

where id=#{id}

</update>

<update id=*"updateEmpInfoAccount"*

parameterType=*"map"*>

update empinfo set account=#{newaccount}

where account=#{oldaccount}

</update>

<delete id=*"deleteAdmin"* parameterType=*"String"*>

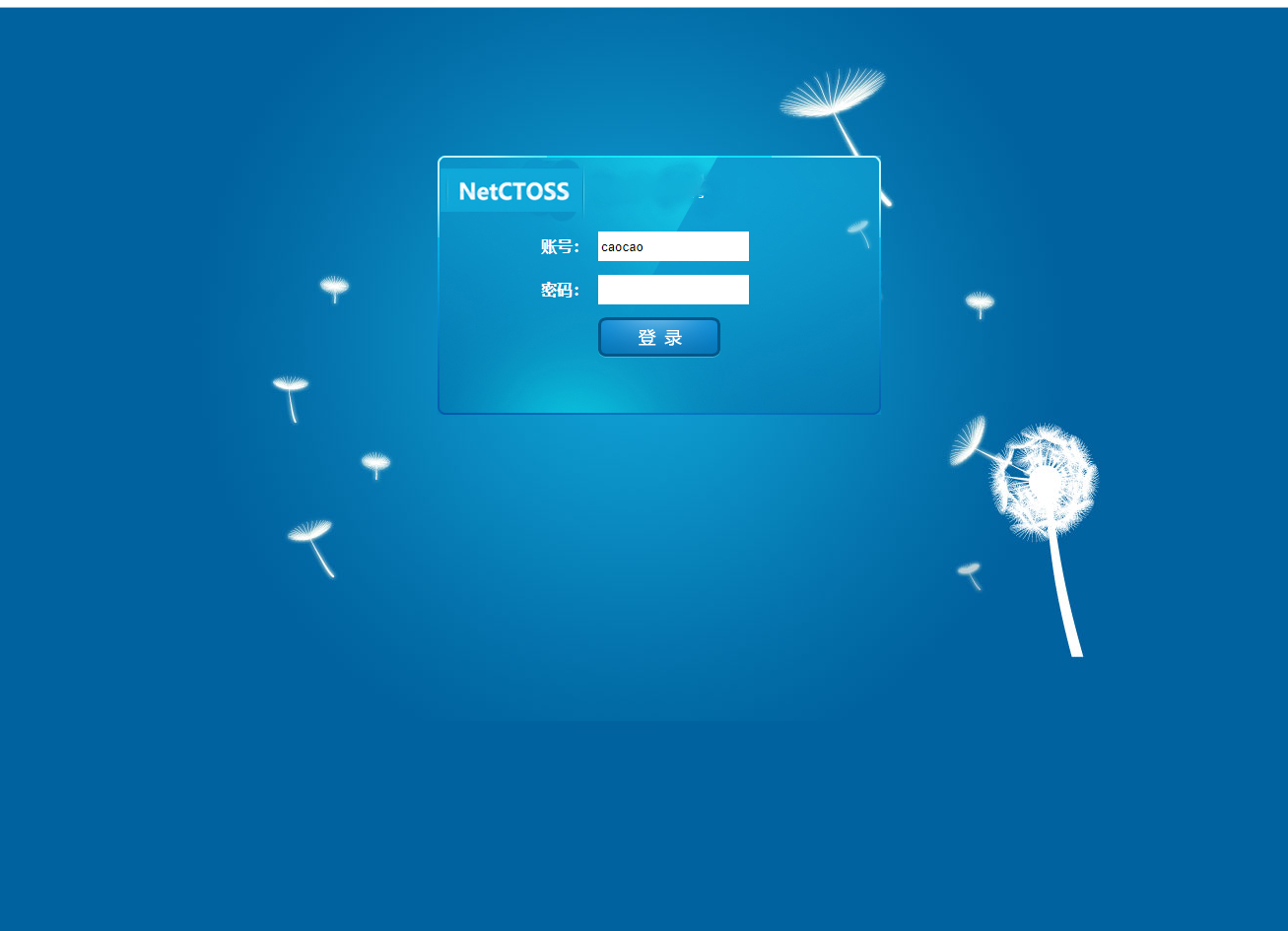
delete from dx\_admin where account =#{account}

</delete>

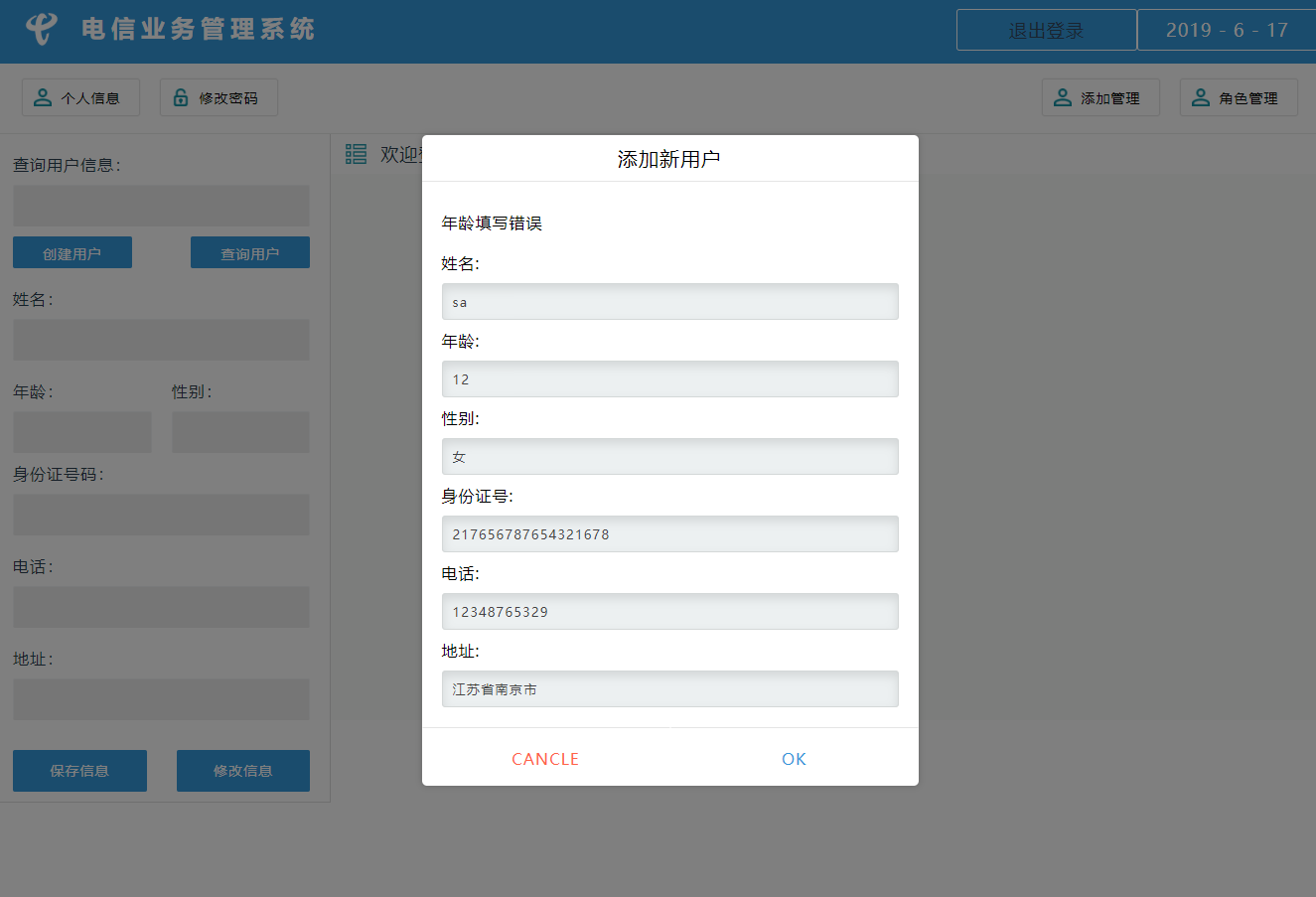
</mapper>

# 六 功能测试

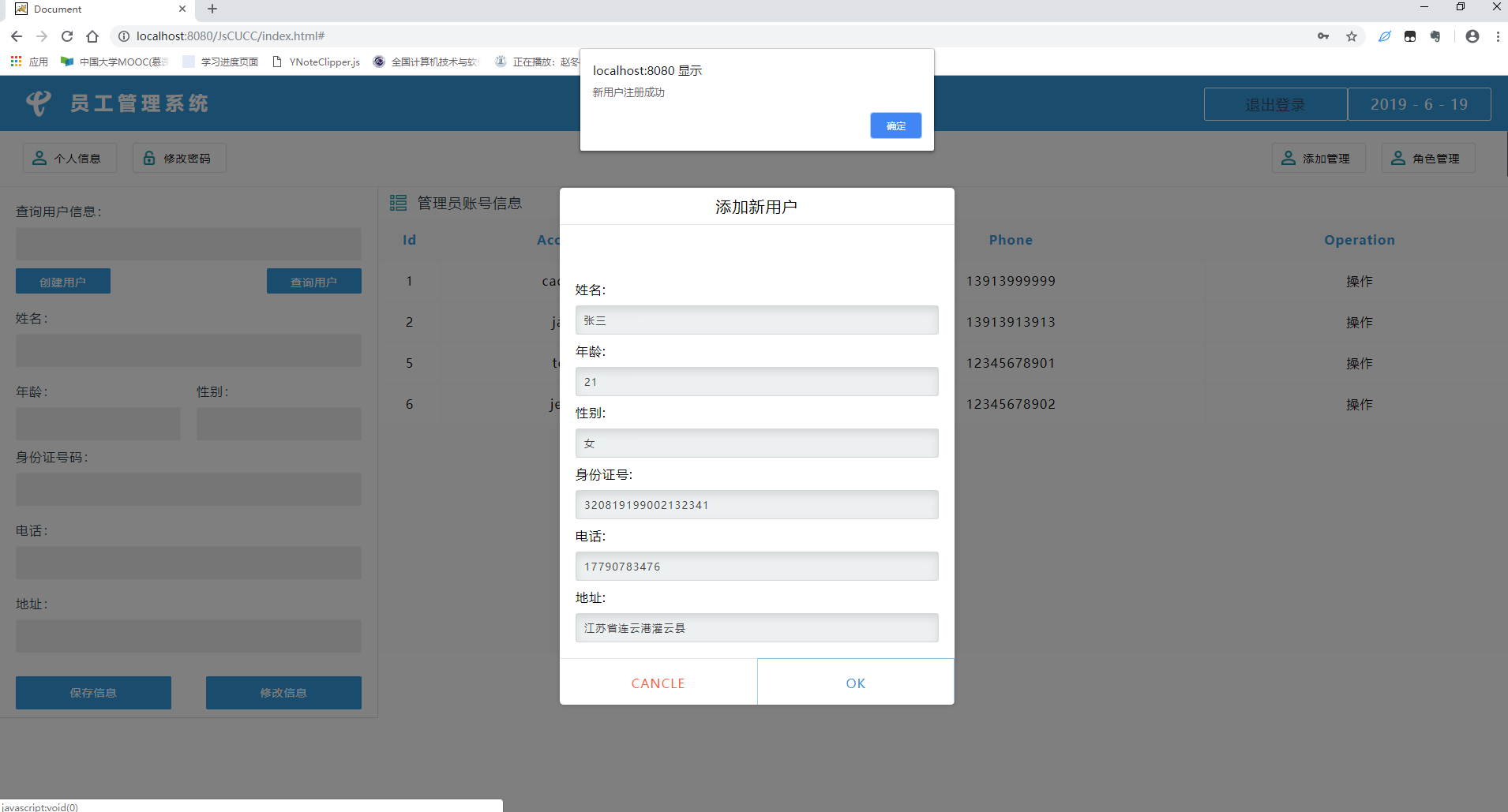
6.1部分操作截图：



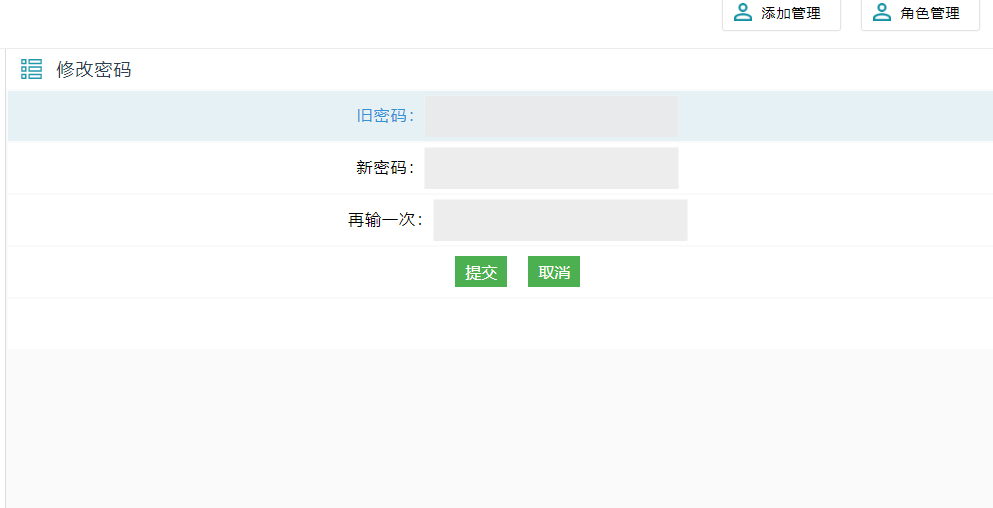
（图6-1 用户登陆界面）



（图6-2 添加新用户）

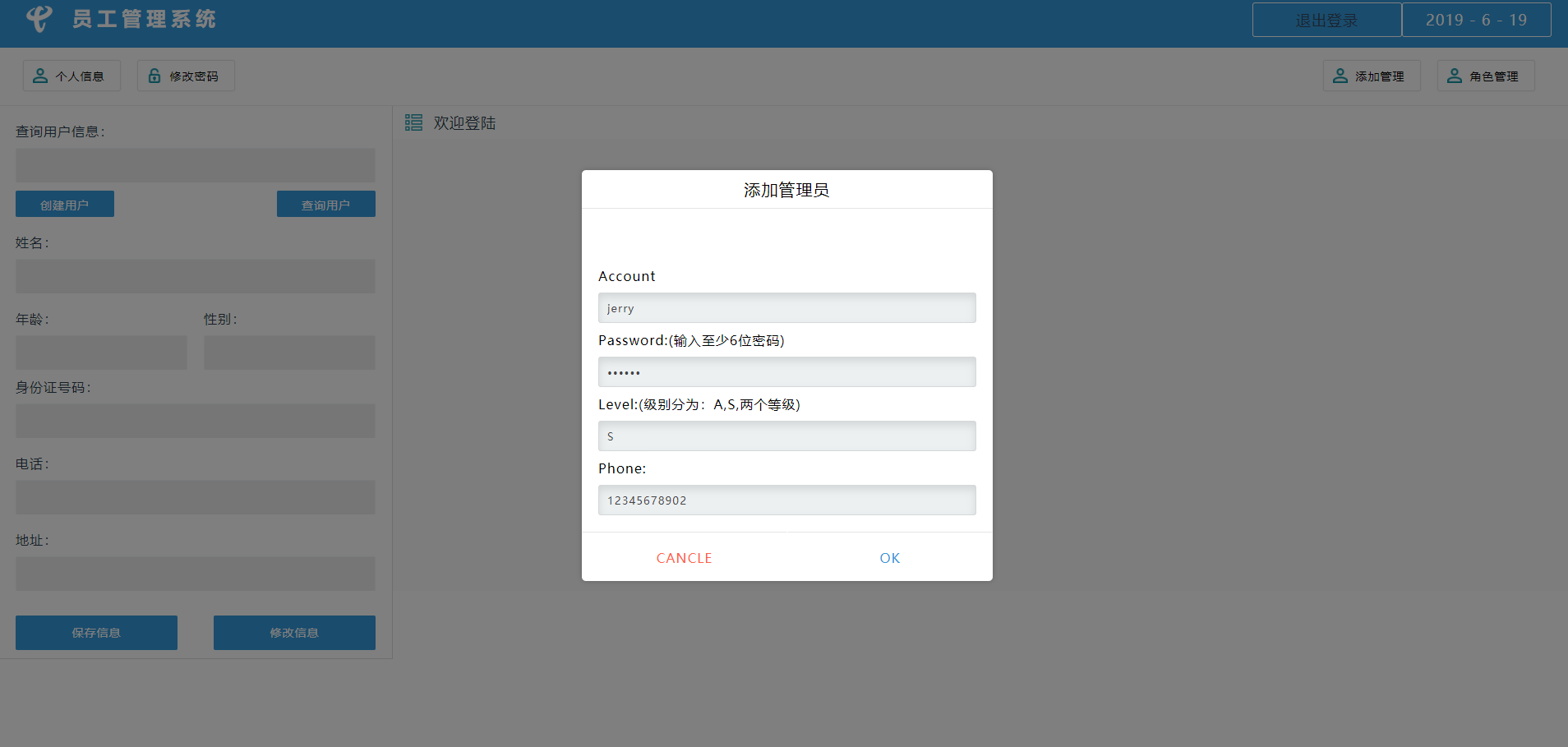


（图6-3 添加用户成功）

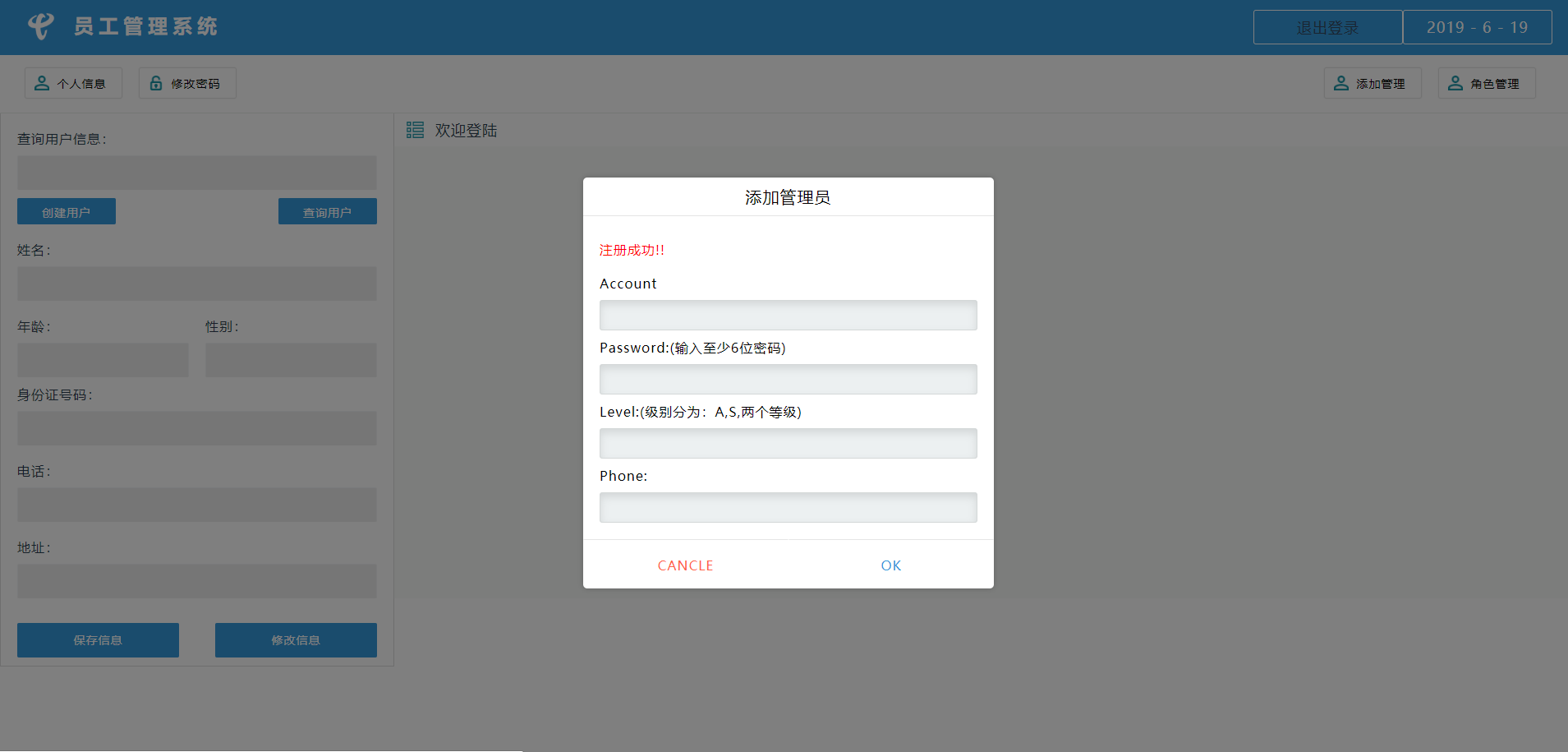


（图6-4 修改密码界面）

（图6-5 查询用户信息）



（图6-6 添加管理员）



（图6-7 添加管理成功）

6.3数据库的维护

当试运行数据库合格后，数据库开发设计的工作就基本完成了，接下来就是正式运行中的调试，应为该系统比较简单，数据量小，数据库中几乎不会发生什么大的变化，但是还是需要做好数据的备份，在MySQL中我们可以利用备份数据库的功能对已经设计好的数据做备份，如果数据库受到破坏或系统故障，我们便可轻松的利用备份文件恢复数据库的数据。当然也可以利用其他各种方法进行数据维护。

# 七 安全性测试

JAVA JDBC经过十几年的发展，针对sql注入这种攻击其实提供了各种各样的防护机制。列如使用占位符来代替直接使用Java字符串连接sql语句。被占位符传入的sql语句参数都会经过函数处理，过滤掉任何与sql语句有关的字符。凡是与sql语句有关的字符都会被加反斜杠做转义处理。使用这些安全函数可以大大减小应用收到sql注入攻击的可能性。

然而sql注入现如今依旧大行其道的原因，多数是因为应用开发者没有经过系统的开发培训。没有最基本的安全意识以及安全能力。所以直到现在sql注入漏洞依旧层出不穷。

# 八 总结

进一步深入理解数据库设计的整个过程，该学生选课系统的需求分析、数据库概念结构设计有关内容等。掌握使用eclipse和MySQL结合设计C /S结构的管理信息系统。综合练习了MySQL的使用和数据库得管理技术。综合熟悉并练习了SQL语言在编程中的使用。 综合训练编程能力。认识系统，取得系统需求分析，更重要得是在编程方面提高了实现目标代码得能力及调试代码得能力。从课设开始的第一天，我们在老师的指导和阅读指导书的前提下自主完成了一个简单的数据库系统的编写。从系统的分析，到整体的设计，再到数据库的建立，以及java前端页面的编写。数据库驱动的加载。可以说整个的项目开发都是由一个人完成。这是在本学期学习了java课程和数据库课程之后的第一次也是在校的一次很宝贵的实战经验总结。之前对两门课程的学习只是简单的基础学习，没有设计到整个系统的开发。但是在这次的实践中对于整个系统开发有了更深的认识。从专业知识学习的角度说，首先我学会了作为一个系统的整个数据库的建立和完善。学会将SQL语句嵌入java中，对数据库进行操作。也学会了如何利用java的MySQL驱动连接数据库，这为之后的java的其他开发奠定了坚实的基础。其次数据库可课程设计更是提升了java的实战能力。在本次课设中，通过主函数中对其他各个累的调用和 其他设计，更深层次的了解或者说是掌握了java的面向对象这一概念。此次开发总共涉及了自己编写的大大小小的十几个java文件，但正是由于面向对象的思想，对于java的操作却没有想象的那么复杂。系统的维护和改动也只需要在相应的子文件里面做改动。本次实战中具有突出的是在java的主界面的编写中，为了对于各个子文件（子页面）的操作更加的简便，也为了主菜单格式的统一，我使用了树目录结构的方式，使得主界面的操作灵活方便。也使得各个操作在主界面的显示样式更加的统一方便好看。总体来说本次的数据库课程设计从整体上独立完成了一个系统的开发。为之后多项专业课的学习打下了坚实的基础，也是一个检验自己java与数据库知识掌握程度的机会，在此次课程设计中收获颇丰，再次有类似的学习事件机会。