****

**数据库系统课程设计**

**实验报告**

**-------图书馆信息管理系统**

**计算机科学与技术学院 计算机科学与技术专业 22级 X班**

**姓名：ZYT 学号：2022XXXXXXXX**

**姓名：WQN 学号：2022XXXXXXXX**

**姓名：QJY 学号：2022XXXXXXXX**

**任课教师：LWQ 实验教师：LWQ**

目 录

[前言 3](#_Toc154753215)

[一、系统开发平台 4](#_Toc154753216)

[1.1 开发平台简介 4](#_Toc154753217)

[1.2 开发语言：Java 4](#_Toc154753218)

[1.3 开发工具：Eclipse Vx.x JDK 1.6 Navicat for MySQL 4](#_Toc154753219)

[1.4 数据库: MySQL 5.7 4](#_Toc154753220)

[1.5 操作系统:Microsoft Windows 11 5](#_Toc154753221)

[二、数据库规划 5](#_Toc154753222)

[2.1 任务陈述 5](#_Toc154753223)

[2.2 任务目标 5](#_Toc154753224)

[三、系统定义 6](#_Toc154753225)

[3.1 系统边界 6](#_Toc154753226)

[3.2 用户视图 7](#_Toc154753227)

[四、需求分析 9](#_Toc154753228)

[4.1 用户需求说明 9](#_Toc154753229)

[4.2 系统需求说明 11](#_Toc154753230)

[五、数据库逻辑设计 12](#_Toc154753231)

[5.1 ER图](#_Toc154753232)[【3】](#_Toc154753232)[： 12](#_Toc154753232)

[5.2 数据字典 13](#_Toc154753233)

[六、数据库物理设计 15](#_Toc154753234)

[6.1 索引 15](#_Toc154753235)

[6.2 视图 16](#_Toc154753236)

[6.3 安全机制 16](#_Toc154753237)

[6.4 其他 16](#_Toc154753238)

[七、应用程序设计 17](#_Toc154753239)

[7.1 功能模块 17](#_Toc154753240)

[7.2 界面设计 18](#_Toc154753241)

[7.3 事务设计 40](#_Toc154753242)

[八、测试和运行 81](#_Toc154753243)

[九、总结 102](#_Toc154753244)

[9.1 系统优点 102](#_Toc154753245)

[9.2 系统不足 103](#_Toc154753246)

[9.3 系统改进 103](#_Toc154753247)

[9.4 经验与收获 103](#_Toc154753248)

[附. 参考文献 104](#_Toc154753249)

前言

在当今世界，科技的迅猛发展带来了巨大的变革和机遇。随着社会经济的不断发展，图书馆作为知识传播的重要场所，也随之得到了极大的提升。现代图书馆不仅是传统阅读场所，更是现代文化中心、学术交流中心和信息服务中心。图书馆的服务功能和管理水平得到了空前的提高。

然而，在现代信息化时代，图书馆面临着新的挑战和机遇。如何利用现代计算机信息技术，提高图书馆的管理效率和服务质量，成为了亟待解决的问题。传统的图书馆管理方式已经无法满足现代信息化的需求。比如，图书资源管理、借阅服务、读者档案管理等方面存在一些问题，例如手动操作、数据不全面、查询困难等。这些问题使得图书馆管理变得复杂和低效。

为此，我们团队结合现代计算机信息技术，设计了一套图书馆信息管理系统。该系统具有操作简便、功能齐全、数据安全性高等特点，可以有效地提高图书馆的管理效率和服务质量。系统采用现代化的信息技术手段，能够有效地实现图书管理、读者管理、借阅服务等方面的自动化和智能化，提高了图书馆的管理水平和服务质量。

在本次论文报告中，我们将详细介绍系统设计的整体思路、功能模块以及技术实现方案等内容。同时，我们也将对系统的优缺点进行详细分析，并提出一些有效的改进方案，以期为图书馆信息化建设提供有益的参考和借鉴。

# 一、系统开发平台

## 1.1 开发平台简介

本图书馆信息管理系统采用C-S架构，使用Java的技术路线，遵守MVC Model模型。

Model模型层使用数据库作为持久性储存结构，选用的后台数据库是MySQL5.7。这是业界领先的开源数据库，在开源产品中具有仅次于Apache服务器的市场占有率。本数据库开放源代码，具有免费使用，比较稳定的特点，适合于小型系统的持久性存储。

View视图层使用JAVA GUI技术。这是Java 本身的图形界面技术。管理系统中严格控制界面代码中的业务逻辑部分，实现了很好的业务和表现的分离。

Controller控制层实现了对数据库中数据的增删查改的具体方法。界面获取的数据通过调用相应方法进行数据的封装和与数据库的交互，从而实现业务逻辑和表现的完全分离。

## 1.2 开发语言：Java

用Java做设计流程清晰、结构合理，有良好的可扩充性和耦合性。

## 1.3 开发工具：Eclipse Vx.x JDK 1.6 Navicat for MySQL

## 1.4 数据库: MySQL 5.7

MySQL是业界领先的开源数据库，在开源产品中具有仅次于Apache服务器的市场占有率。本数据库开放源代码，具有免费使用，比较稳定的特点，适合于小型系统的持久性存储。

## 1.5 操作系统:Microsoft Windows 11

# 二、数据库规划

## 2.1 任务陈述

图书馆在正常运营中面对大量书籍、读者信息以及两者间相互联系产生的借书信息、还书信息。现有的人工记录方法既效率低又错误多，大大影响了图书馆的正常管理工作。因此我们需要建立一个自动化的系统来管理图书馆的资源和信息，通过及时了解这些环节中的信息变更，提高图书馆的管理效率，减少人为错误的发生，并且更快地进行数据查询和更新。通过有效的资源和信息管理，图书馆将能够更好地满足读者的需求，提供更高效的服务。

本项目设计并实现了一个简单的图书馆信息管理系统，该管理系统对书籍资源、读者资源、借书信息、还书信息进行管理和整合归纳，掌握各个环节中信息的变更情况，具有用户注册和登录、图书查询、办理借书证、图书借阅、图书归还、管理图书等基本功能。同时，本图书馆信息管理系统提供了三个用户方，一是对管理员，二是对用户，三是对访客。管理员可以对图书进行添加、查询、修改以及删除操作；可设置借书还书的期限，一键归还全部书籍与注销借书证；也可以对用户进行管理。用户能进行登录和注册，在主页面查询和借还图书、办理和查看借书证。

## 2.2 任务目标

|  |  |  |
| --- | --- | --- |
| 角色 | 功能 | 功能介绍 |
| 用户 | 登录或注册 | 进入“用户”模式，输入用户名和密码进行账号的注册或登录，对应显示“注册成功！”或“登录成功！”，否则显示“用户名或密码错误！” |
| 查询图书 | 通过书号、书名、作者、类型、出版社、库存量等关键字查询图书 |
| 借阅图书 | 在查询图书界面选中待借阅的图书，输入借书证卡号和账号密码完成借阅，显示“完成借阅！” |
| 归还图书 | 输入借书证卡号和账号密码进入借书证查看界面，选中已借阅的图书信息完成归还，显示“成功归还！” |
| 办理借书证 | 通过输入用户名和密码办理借书证，办理成功后生成借书证卡号，显示“办理成功！卡号：xxxxxx” |
| 查看借书证 | 通过借书证查询借书信息 |
| 管理员 | 登录 | 进入“管理员”模式，输入用户名和密码进行账号登录（对应显示“登录成功！”，否则显示“用户名或密码错误！”） |
| 查询图书 | 通过书号、书名、作者、类型、出版社、库存量等关键字查询图书 |
| 管理图书 | 进入管理图书模块，通过书号、书名对图书信息进行修改、删除，对应显示“修改成功！”或“删除成功！” |
| 管理借书证 | 进入管理借书证模块，可一键归还全部书籍与注销借书证，对应显示“归还成功！”或“注销成功！” |
| 管理用户 | 进入管理用户模块，可以注销用户信息 |
| 访客 | 查询图书 | 进入“访客”模式，通过书号、书名、作者、类型、出版社、库存量等关键字查询图书 |

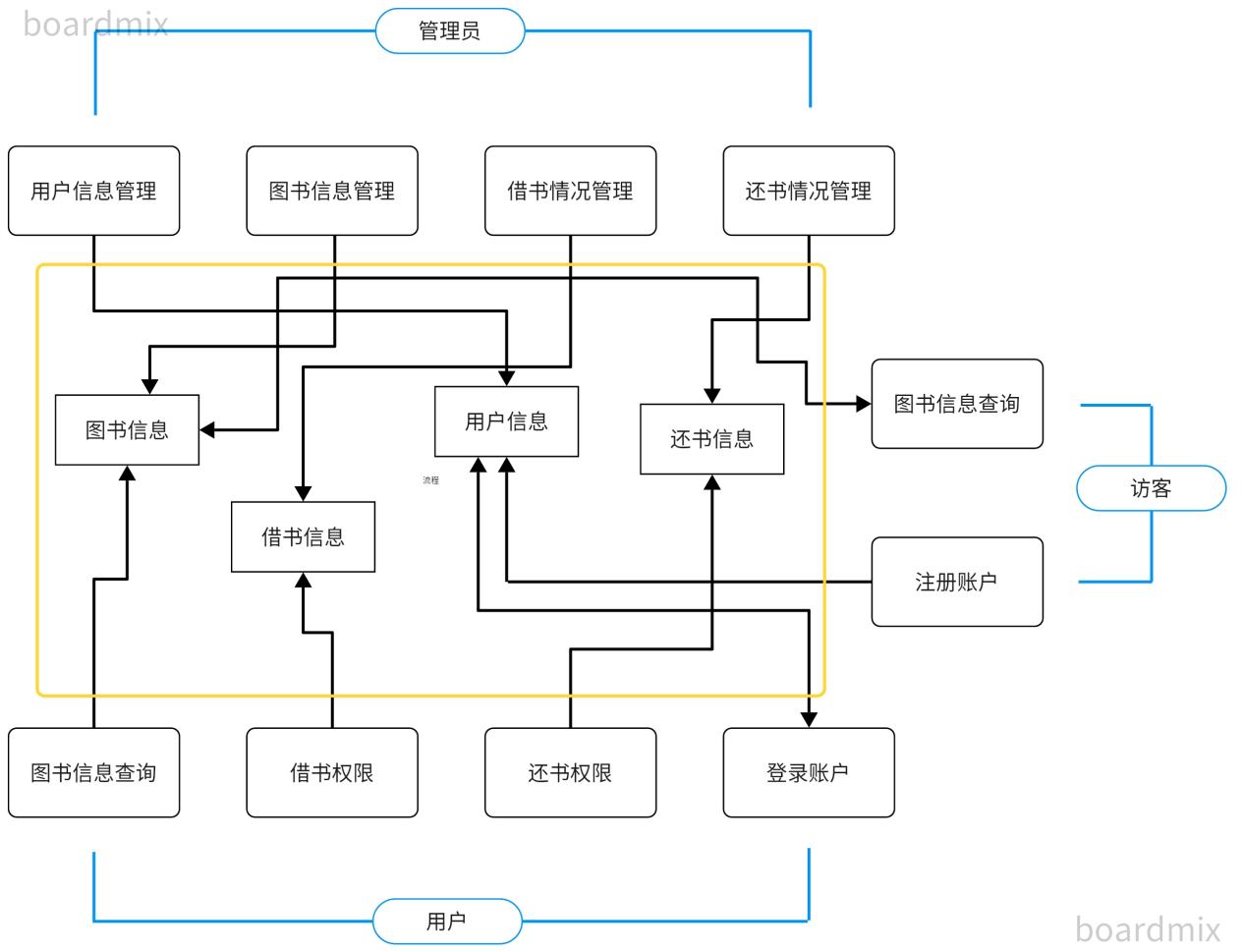
# 三、系统定义

## 3.1 系统边界

系统边界描述数据库系统和企业信息系统的其他部分的接口，是信息系统内部构成元素与外部有联系实体之间的信息关系的描述与分割。它并不需要在它们之间划一条物理边界，而只需要弄清它们之间信息输入与输出的分割。

本数据库系统共包括个模块，分别为用户信息管理（包括新建用户信息及查看修改用户信息）、图书信息管理（包括新建、修改图书编号、类别等信息）、借书情况管理（包括馆内图书的借阅状态、借阅用户、归还日期等信息）、归还情况管理（包括在借图书的借阅用户、归还日期等信息）、查询图书信息（用户可通过单条件或多条件查询图书信息）；普通访客可登录信息系统进行登录注册或仅查询图书信息，注册分为用户注册及管理员注册，访客注册后可通过登录界面进行登录，用户具有查询图书信息、借阅图书、归还图书等权限，管理员具有用户信息管理（包括查看用户信息及办理借书证等）、图书信息管理、借书情况管理、还书情况管理等权限。

数据库系统的内部构成元素与外部其他部分之间的信息关系如下图所示：



## 3.2 用户视图

**管理员（Administrator）视图**

1. 用户信息管理：

管理员可以制定、输入、修改、查询用户信息，包括用户种类、性别、结束数量、借书期限、备注等。

1. 图书信息管理：

管理员可以制定、输入、修改、查询图书信息，包括书籍编号、一级类别、关键词、备注等。

1. 借书情况管理：

管理员可以制定、输入、修改、查询借书信息，包括书籍编号、用户编号、借书日期、借书期限、备注等。

1. 还书情况管理：

管理员可以制定、输入、修改、查询还书信息，包括书籍编号、用户编号、还书日期、还书期限、备注等。

**用户（User）视图**

1. 图书信息查询：

查询书籍基本信息，包括书籍编号、类别、关键词、备注等。

1. 借书权限：

查询图书信息后进行书籍借阅，最终显示包括书籍编号、读者编号、借书日期、借书期限、备注等信息。

1. 还书权限：

在借阅图书之后进行图书归还（需要判断是否超期），如果未超期完成图书归还，显示书籍编号、读者编号、还书日期、备注等信息。

1. 多条件查询信息：

通过书籍编号、类别、关键词、借阅状态等多种信息多条件查询馆内图书信息情况

**访客（Visitor）视图**

（1） 注册账户：

在登录界面可选择跳转至注册界面，在注册界面中选择注册身份（分别为用户、管理员），输入设定的账号（如果账号已存在，会注册失败）和密码，并再次确认密码，由系统自动分配用户编号；

（2） 登录账户：

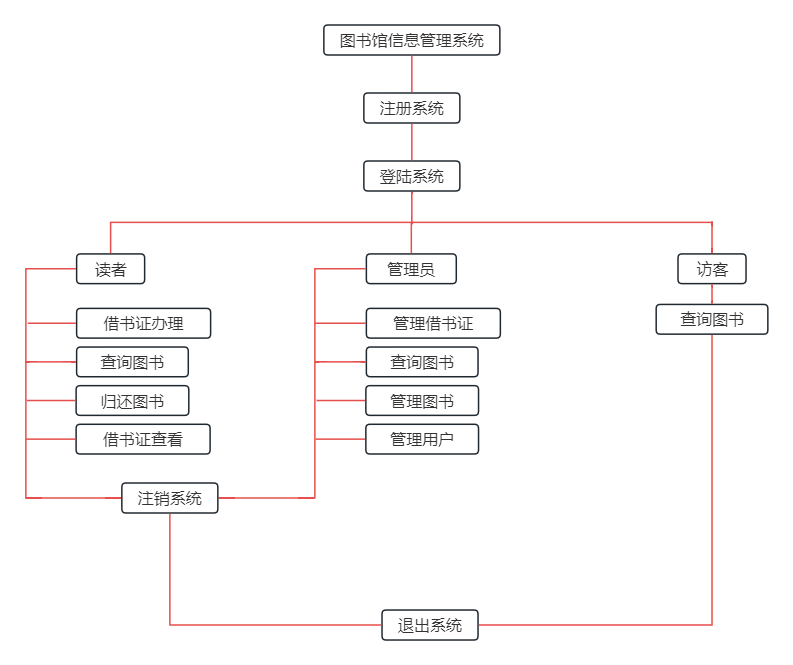
注册成功后或已有账号的用户或管理员可在此页面登录图书馆信息管理系统。

（3）图书信息查询：

查询书籍基本信息，包括书籍编号、类别、关键词、备注等。

# 四、需求分析

图书管理系统主要功能：



## 4.1 用户需求说明

**4.1.1 数据需求**

1. 图书信息：

书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍备注、书籍库存量

1. 读者信息：

读者姓名、读者借阅证id、读者借阅密码

1. 管理者信息：

管理员id,管理员密码

**4.1.2 事务需求**

**1、数据录入：**

**读者视图下：**

1. 注册：读者用户名、密码
2. 登录：读者用户名、密码
3. 借书证办理：读者用户名、密码

**管理者视图下：**

1. 管理员模式：查询图书、管理图书、管理借书证、管理用户
2. 登录：用户名、密码

**访客视图下：**

（1）查询图书：书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量、书籍可借时长

1. **数据更新/删除：**

**读者视角下：**

1. 图书借阅：书籍名称、书籍编号、借书证号、借书证密码、借书时间、书籍可借时间、书籍归还期限
2. 图书归还：借书证卡号、书籍编号

**管理者视角下：**

1. 图书修改、删除：书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量、书籍可借时长
2. 管理图书：书籍名称（必填）、书籍编号（必填）、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量调整
3. 借书证管理：用户姓名、借书证号、借书数量、书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、借书时间、书籍归还期限
4. 用户管理：读者用户名、读者借书数量
5. **数据查看：**

**读者视角下：**

1. 查询图书：书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量、书籍可借时长
2. 借书证查看：借书证号、书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、借书数量、借阅时间、书籍归还期限

**管理者视角下：**

1. 图书查询：书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量、书籍可借时长
2. 借书证管理：用户姓名、借书证号、借书数量、书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、借书时间、书籍归还期限
3. 用户管理：读者用户名、读者借书数量

## 4.2 系统需求说明

该图书馆信息管理系统需要较强的数据处理功能，理论上应该能够容纳较多的图书以及借阅者用户的数据资料，并且在搜索方面理应具有较快的响应速度，能够处理多方面的数据请求。系统能够有效的处理各种异常，具有较好的健壮性。

**4.2.1 初始数据库大小**

（1）书籍类目有书籍名称、书籍编号、书籍作者、书籍类别、书籍关键词、书籍出版社、书籍库存量、书籍可借时长

（2）管理员账号一个

**4.2.2 网络和共享需求**

（1）必须能够支持至少50名用户同时访问，需要考虑这么大数量并发访问的许可需求。

**4.2.3 性能**

高峰期：每天的上午、下午

（1） 单个记录查询时间少于1秒，高峰期少于5秒

（2） 多个记录查询时间少于5秒，高峰期少于10秒

（3） 更新/保存记录时间少于1秒，高峰期少于5秒

**4.2.4 安全性**

（1） 数据库必须有口令保护

（2） 每个用户分配特定的用户视图所应有的访问权限

（3） 用户只能在分配到权限的窗口中看到需要的数据

**4.2.5 备份和恢复**

每天24点备份

**4.2.6 用户界面**

菜单驱动，联机帮助

**4.2.7 法律问题**

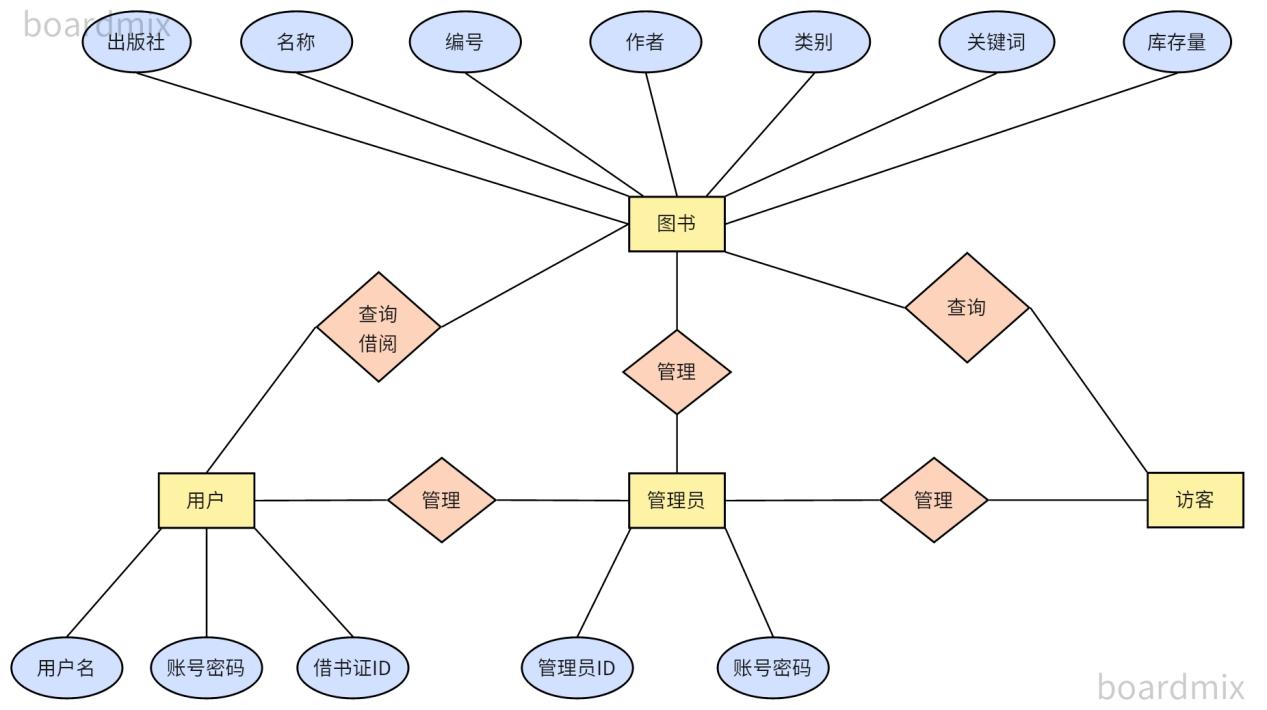
对用户信息管理，遵守法律

# 五、数据库逻辑设计

## 5.1 ER图【3】：

本系统的ER图如下：

该ER图包括管理员、用户、访客、图书四个实体；以及管理、借阅、查询三种联系。



## 5.2 数据字典

**5.2.1 从数据字典中抽取出来的系统实体描述：**

总共设计了5张表：user, administrators, book, card, record。具体内容如下：

#### （1）用户实体表：user

| **字段名** | **类型** | **性质/限制** | **说明** |
| --- | --- | --- | --- |
| id | int | 主码，自增 | 用户的id，用于辨别身份 |
| password | Varchar(45) | 长度不超过12字符 | 登录密码 |
| username | Varchar(45) | 唯一，长度不超过20字符 | 用户名 |

#### （2）管理员实体表：administrators

| **字段名** | **类型** | **性质/限制** | **说明** |
| --- | --- | --- | --- |
| id | int | 主码，自增 | 管理员的id，用于辨别身份 |
| password | Varchar(45) | 长度不超过12字符 | 登录密码 |
| username | Varchar(45) | 唯一，长度不超过20字符 | 管理员用户名 |

#### （3）书籍实体表：book

| **字段名** | **类型** | **性质/限制** | **说明** |
| --- | --- | --- | --- |
| no | VARCHAR(45) | 主码，唯一 | 书号 |
| name | VARCHAR(45) | 非空 | 书名 |
| author | VARCHAR(45) | 默认“佚名” | 作者 |
| type | VARCHAR(45) | 默认“未知” | 图书类型 |
| publisher | VARCHAR(45) | 默认“未知” | 出版社 |
| storage | VARCHAR(45) | 大于等于0 | 库存量 |

#### （4）借书证实体表：card

| **字段名** | **类型** | **性质/限制** | **说明** |
| --- | --- | --- | --- |
| cardNo | int | 主码，自增 | 卡号 |
| amount | int | 0到5 | 当前借书量 |
| username | Varchar(45) | 外码，参照user | 用户名 |

#### （5）借书记录实体表：record

| **字段名** | **类型** | **性质/限制** | **说明** |
| --- | --- | --- | --- |
| rid | int | 主码，自增 | 借书记录的编号 |
| cardNp | int | 外码参照card | 卡号 |
| no | Varchar(45) | 外码参照book | 书号 |

**5.2.2 从数据字典中抽取出来的联系的描述：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实体 | 多样性 | 联系 | 多样性 | 实体 |
| 用户 | m | 注册 | n | 借书证 |
| 用户 | m | 注销 | n | 借书证 |
| 用户 | m | 查看 | n | 借书证 |
| 用户 | m | 查询 | 1 | 书籍 |
| 用户 | m | 归还 | l | 书籍 |
| 管理员 | q | 管理 | n | 借书证 |
| 管理员 | q | 查询 | l | 书籍 |
| 管理员 | q | 管理 | l | 书籍 |
| 管理员 | q | 管理 | m | 用户 |

# 六、数据库物理设计

## 6.1 索引

数据库中的索引：

|  |  |  |
| --- | --- | --- |
| 表名 | 主键 | 外键 |
| t\_book | t\_book\_id | t\_booktype(t\_booktype\_id) |
| t\_user | t\_user\_id |  |
| t\_reader | t\_reader\_id |  |
| t\_borrow | t\_borrow\_id | t\_booktype(t\_booktype\_id) |
| t\_booktype | t\_booktype\_id |  |

## 6.2 视图

此环节设计在数据库应用生命周期的需求分析和收集阶段标识的用户视图。通常，视图使用SQLyog创建。

## 6.3 安全机制

**6.3.1 系统安全**

1、本图书馆信息管理系统提供了充足的异常处理机制，能够捕获由各种错误引发的异常(如：输入数据类型和格式与数据库要求类型和格式不一致、添加图书过程中出现的错误、操作不规范等)。

2、管理员与读者登录时输入信息如果与数据库对应信息不相等，则提示管理者或读者重新输入。

3.图书类别的添加、修改和删除以及图书的添加、修改和删除在操作出现错误时都会显示操作失败提示。

**6.3.2 数据安全**

1、在登录界面输入分不同身份进行选择，并输入账号与密码才能登录。只有管理员才能对图书类别以及图书进行信息修改、添加与删除，以确保数据库不被随意更改，保证数据安全。

2、本系统相关操作都是通过事务实现的，很好的保护了关系型数据库的强一致性，体现了数据安全。

## 6.4 其他

规范化是一种可以创造一个结构上一致且最小冗余的逻辑数据库设计的方法。但是，尽管规范化的设计很好，有时候它并不能提供最大的处理效率。因此，我们可能会接受规范化设计方面的一些损失，以实现更好的性能。

# 七、应用程序设计

## 7.1 功能模块

该图书馆信息管理系统的功能模块有管理员图书类别管理（查询、添加、修改和删除图书类别）、图书管理（查询、添加、修改和删除图书），读者查询图书、借书（查询要借阅的书）和还书，管理员和读者注册账号、登录账号，访客模式查询图书，功能具体实现如下：

**7.1.1 管理员视图下的功能模块**

（1）图书类别管理：

* 图书类别添加（添加新的图书类别信息，包括名称和描述）；
* 图书类别维护（根据图书类别名称查询，对图书类别进行修改或删除）；

（2）图书管理：

* 图书添加（添加新的图书信息，包括名称、作者、作者性别、价格、类别、描述）；
* 图书维护（根据图书名称、作者或类别查询，对图书进行修改或删除）；

（3）安全退出：

* 安全退出系统；
  + 1. **读者视图下的功能模块**

（1） 图书查询：

* 根据图书名称、图书作者或图书类别进行图书查询；

（2） 借阅图书：

* 根据图书名称、图书作者或图书类别进行图书查询；
* 选择想要借阅的图书进行借阅；

（3） 归还图书：

* 选择已借阅的图书进行归还；

（4） 安全退出：

* 安全退出系统；
  + 1. **所有用户视图下的功能模块**

（1）注册账号：

在登录界面可选择跳转至注册界面，在注册界面中选择注册身份（分别为管理员、读者），输入想要注册的账号和密码，并再次确认密码。另有重置功能可以清空错误输入。若第一次输入的密码与再次确认的密码不一致，或者密码格式不符合要求、密码为空，都会有错误提示弹窗；

（2）登录账号：

登录界面，选择登录的角色（管理员或读者），输入账号和密码，点击登录。另有重置功能可以清空错误输入。若密码与对应数据库账号表格中的数据不一致，会有错误提示弹窗。

* + 1. **访客视图下的功能模块**

（1） 查询图书：

* 根据图书名称、图书作者或图书类别进行图书查询；

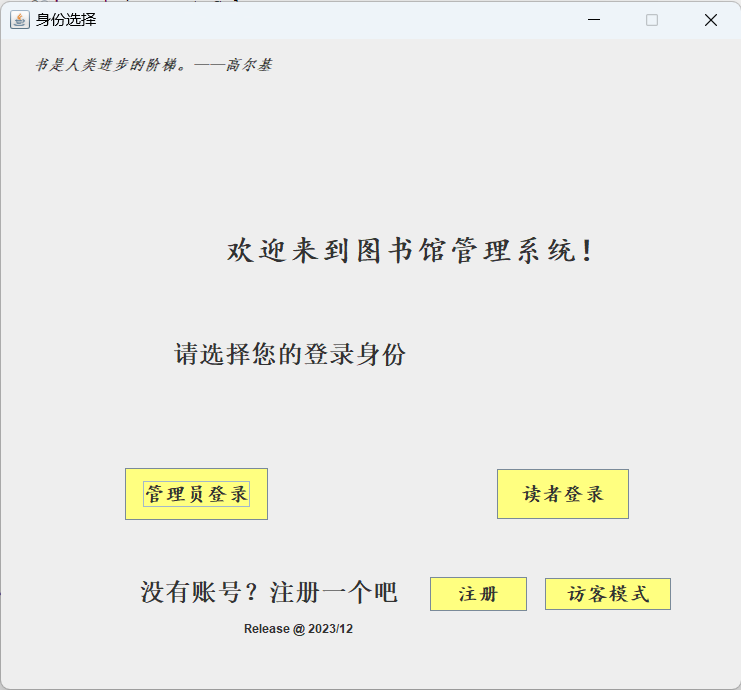
（2）安全退出：

* 安全退出系统；

## 7.2 界面设计

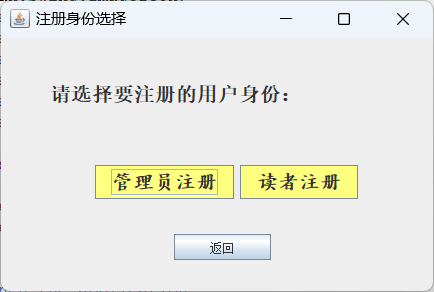
**1 主界面**

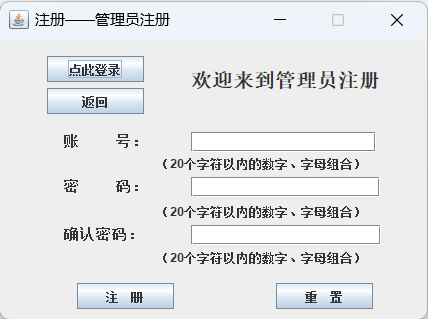
在主界面可以选择注册（注册能够注册管理员与读者）后，进行登录（访客模式无需注册和登录）：不同身份的账号密码各自独立；选择不同身份登录后的界面、权限等均不同；（为空或者输入错误均会提示）

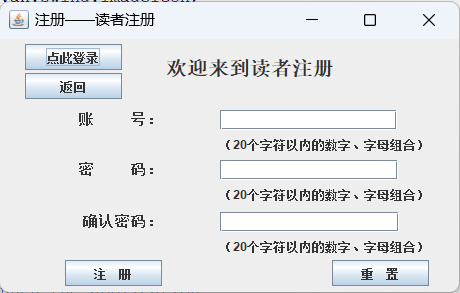


**2 注册界面**

在登录界面可选择注册按钮，进行注册：选择注册身份（管理员/读者），账户名，账号密码和确认密码；

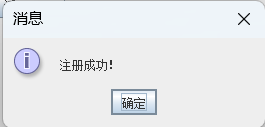






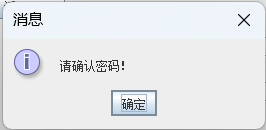
·正确输入情况：

会提示

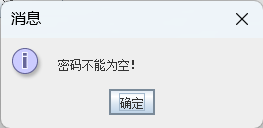


·几种不同错误输入情况：

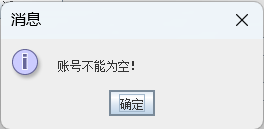
①如果确认密码处为空：



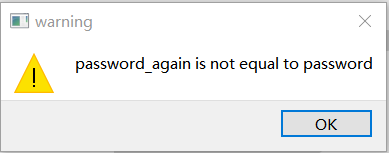
②如果未输入密码：



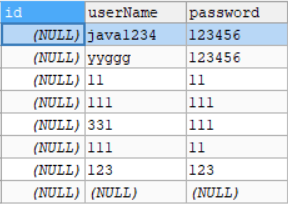
③如果未输入账号：



④如果确认密码和第一次输入密码不同，会提示：

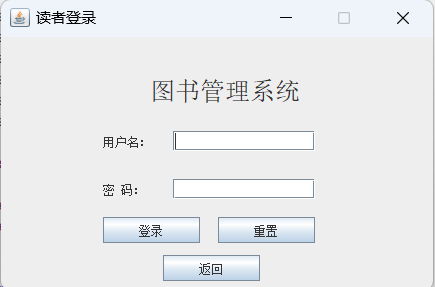


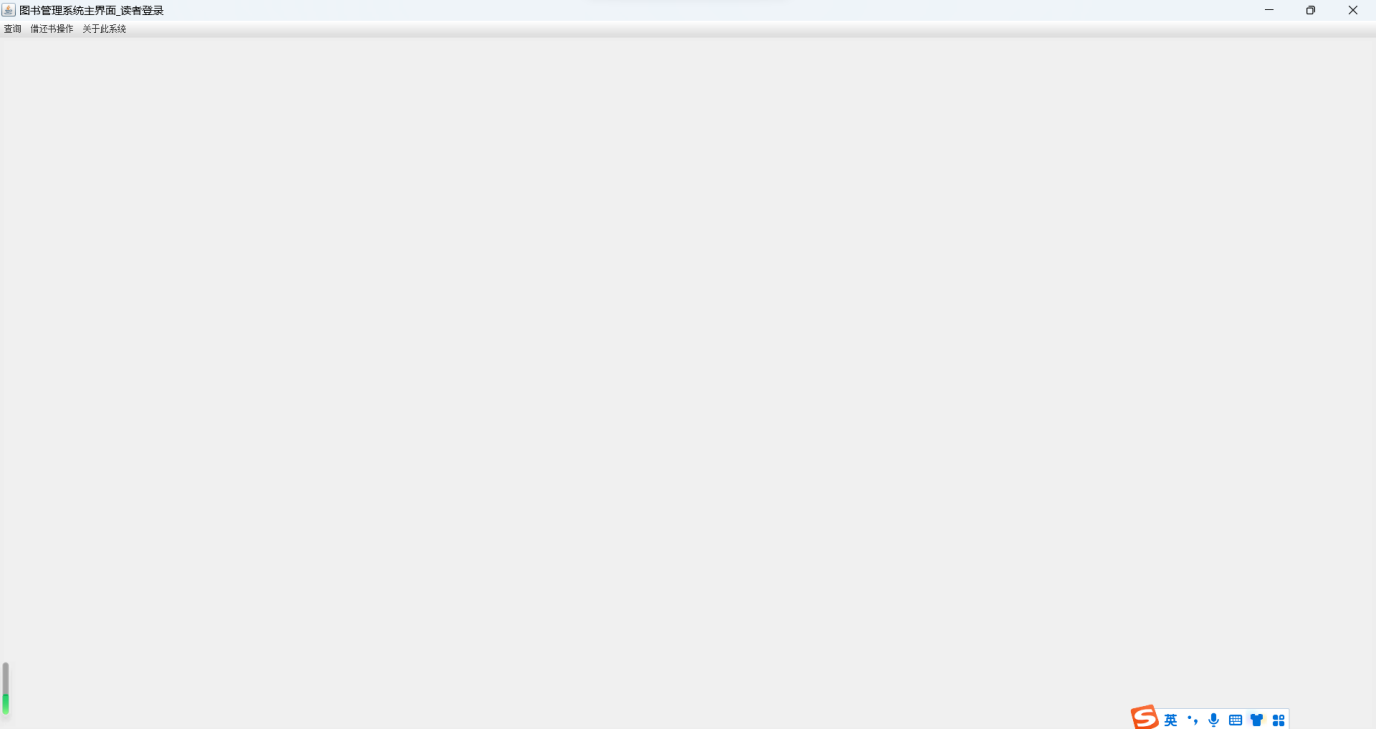
成功注册则表单信息：



**3 读者主界面**

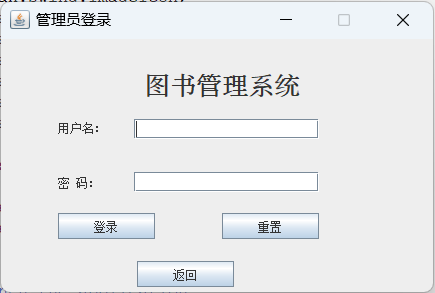
选择读者登录后，输入账号密码正确，就进入读者主界面：（右上方有退出按钮 可以点击退出系统）





**4 管理员主界面**

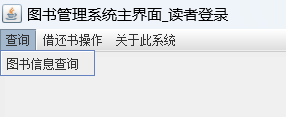
选择买家身份后，输入账号密码正确，就进入买家菜单界面：（右下方有退出按钮 可以点击退出系统）



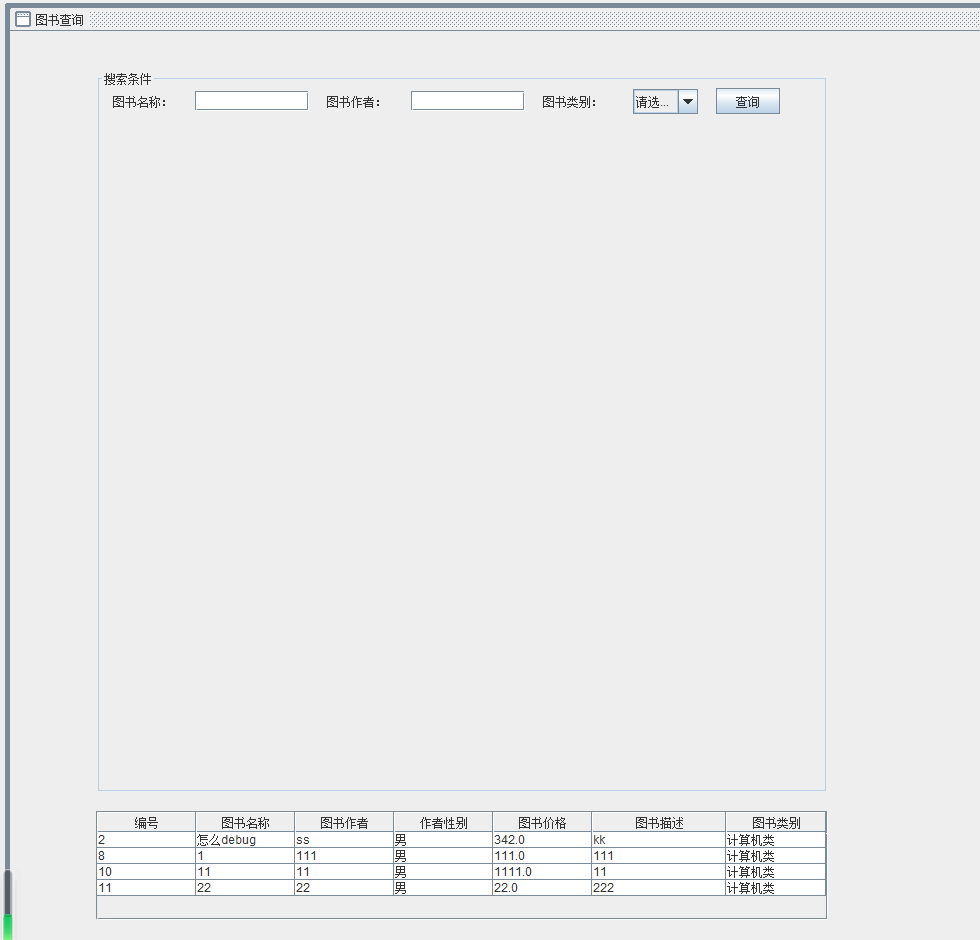
****

**5 读者功能实现（读者）**

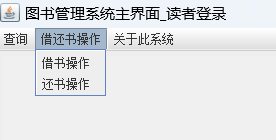
1. 选择读者身份后，输入账号密码正确，就进入读者界面，包括查询、借还书操作、关于此系统三个板块:
2. 其中查询——图书信息查询：



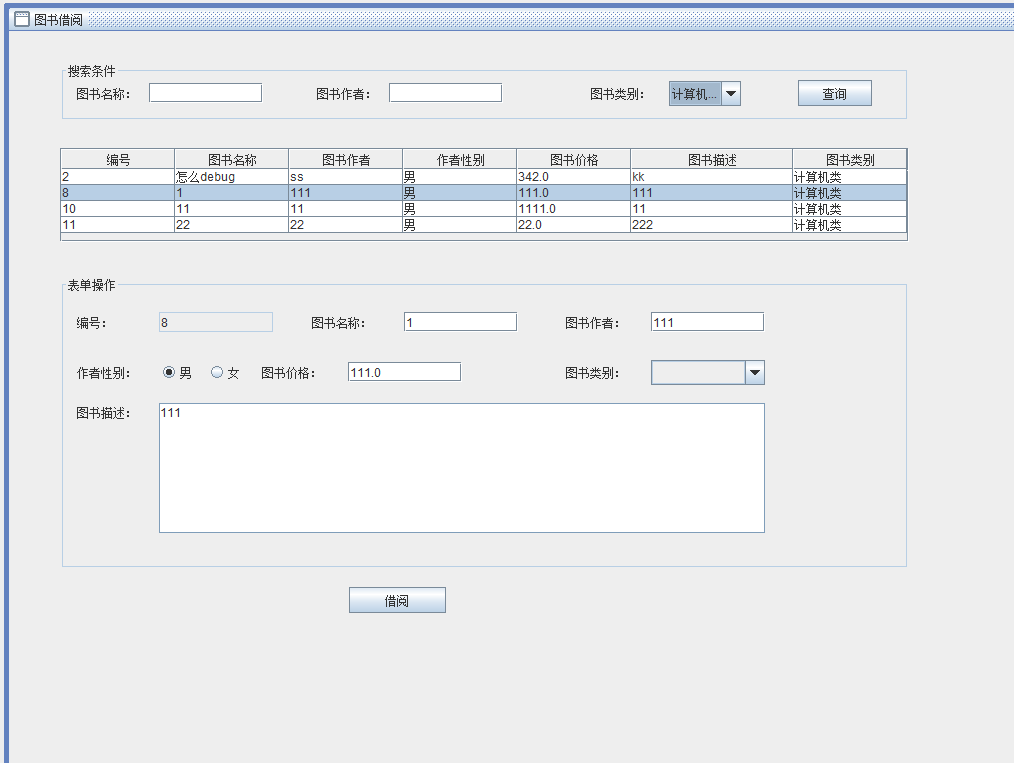
图书查询：



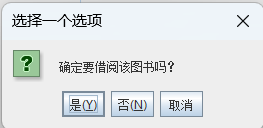
1. 借还书操作包括借书和还书：



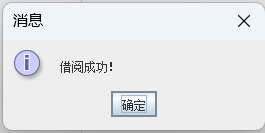
借书，通过查询找出要借阅的图书，选择图书：



点击借阅，显示：



借阅成功，显示：



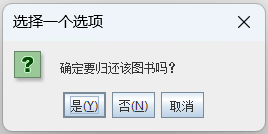
同时数据库中表单更新：



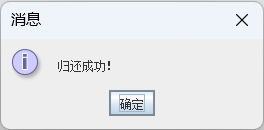
图书归还，上方界面显示所借图书：



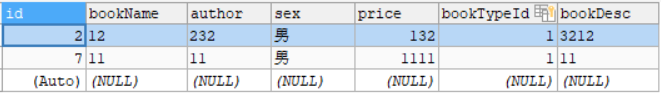
选择图书，下方表单操作处显示该书相关信息，点击归还，显示：



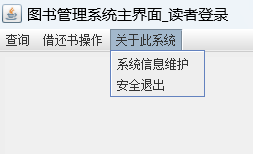
归还成功，则显示：



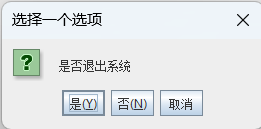
同时表单借书信息删除：



1. 关于此系统包括系统信息维护和安全退出：

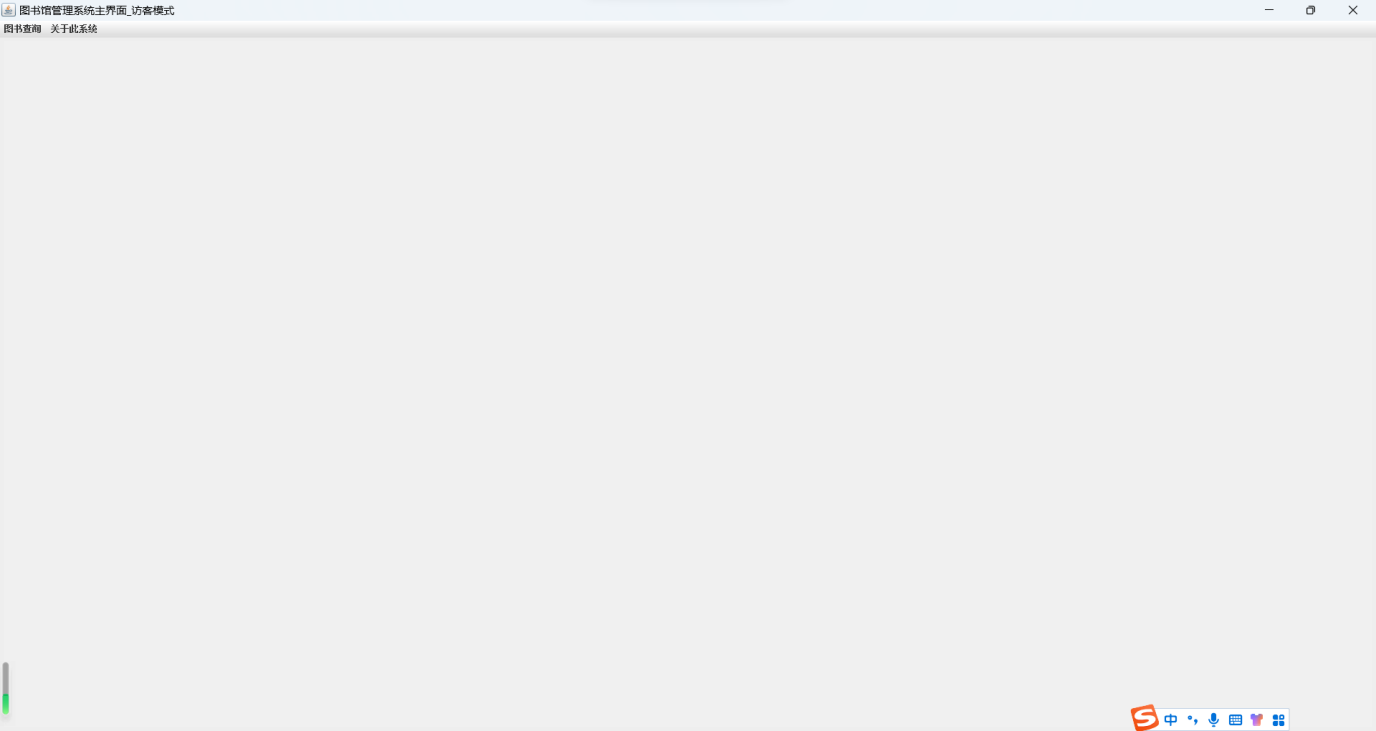


点击安全退出显示：

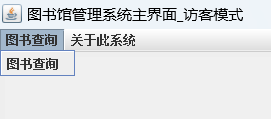


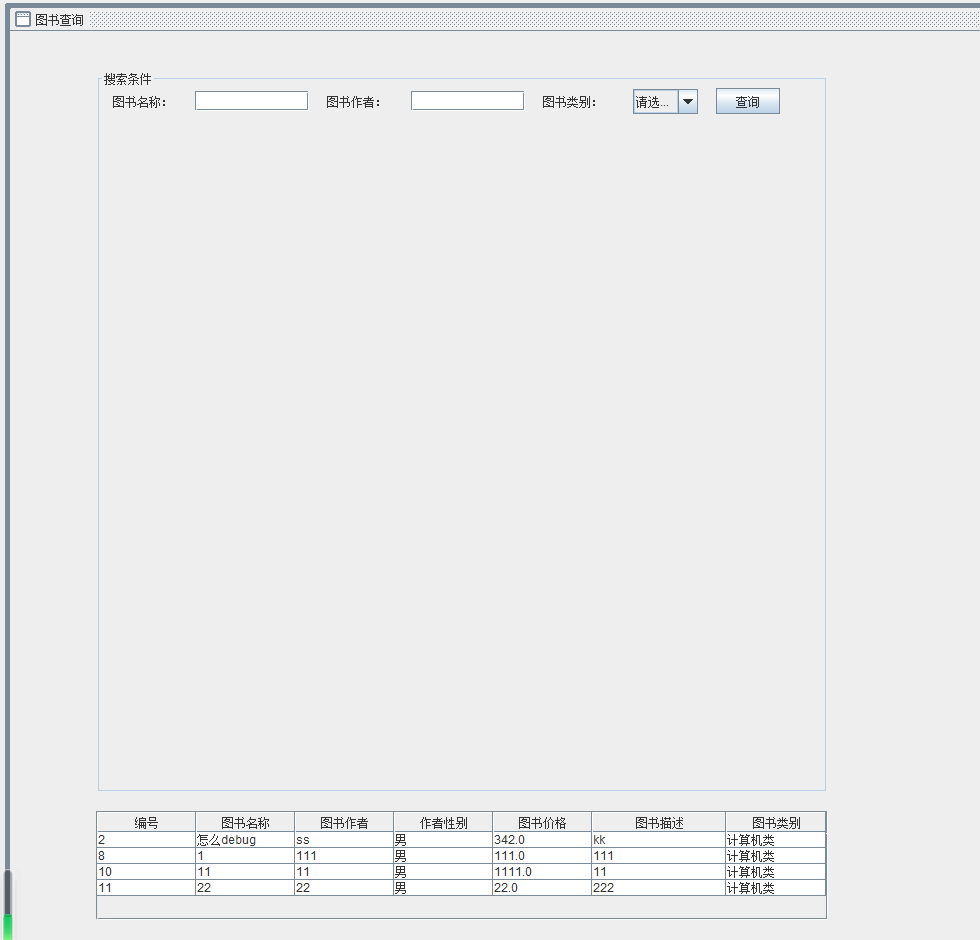
**6 访客模式**

1. 访客主界面如下图所示：

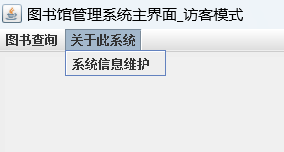


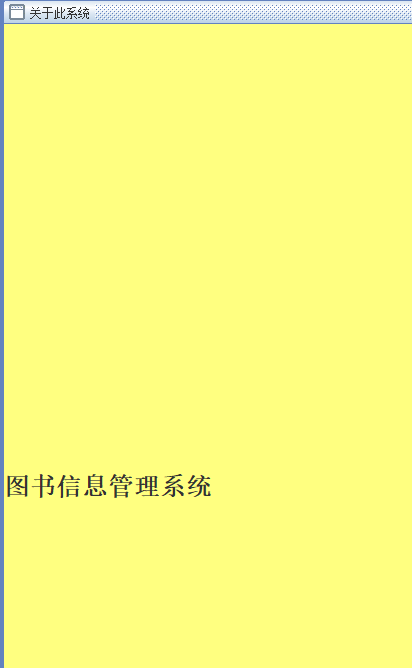
1. 图书查询：





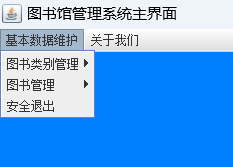
1. 关于此系统：



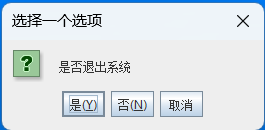


**7 基本数据维护（管理员）**

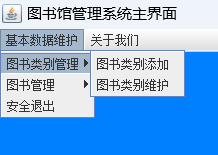
1. 基本数据维护包括图书类别管理、图书管理以及安全退出三个功能板块：



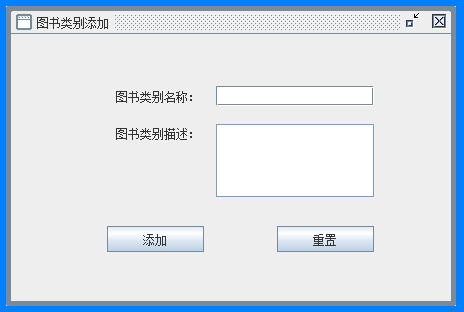
点击安全退出：



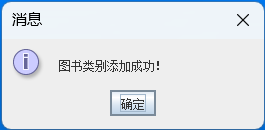
1. 图书类别管理分为图书类别添加和图书类别维护两个功能实现:



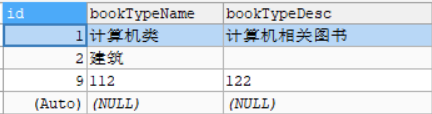
图书类别添加：



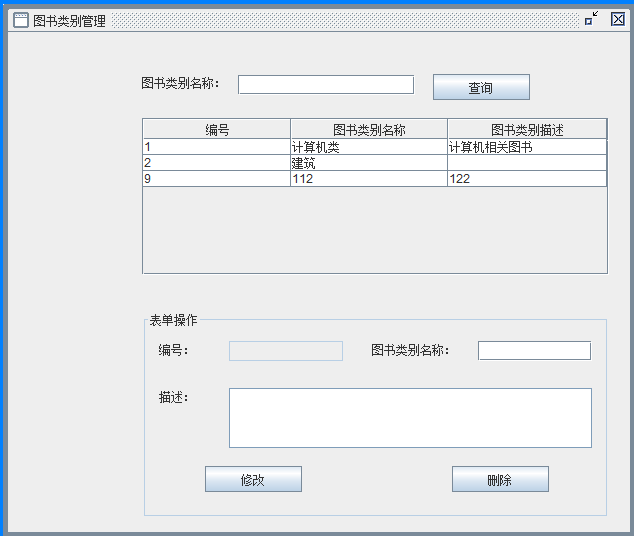
添加成功弹窗显示：



在数据库中显示：

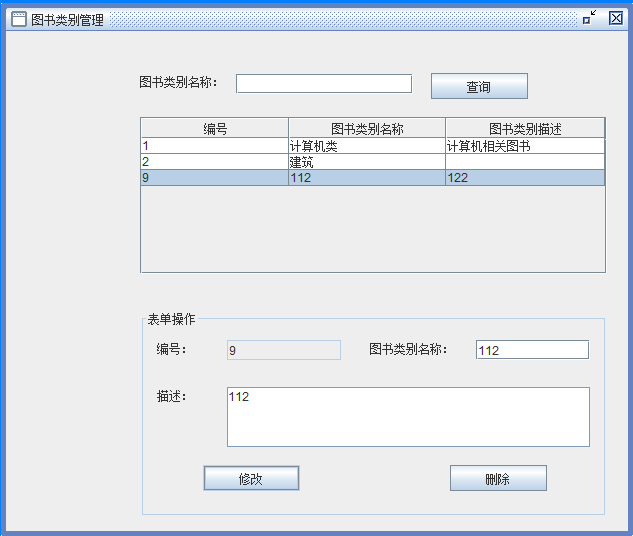


图书类别维护：

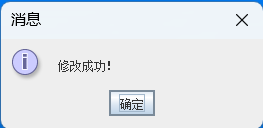


可以在表单操作处进行图书类别的修改与删除：

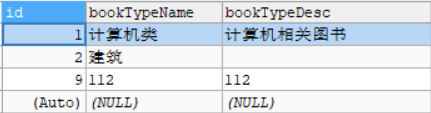
输入修改信息：



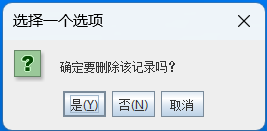
成功修改则显示：



数据库数据随之改变：



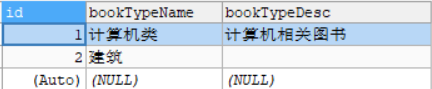
点击删除：



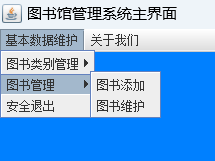
删除成功则显示：



数据库数据随之改变：

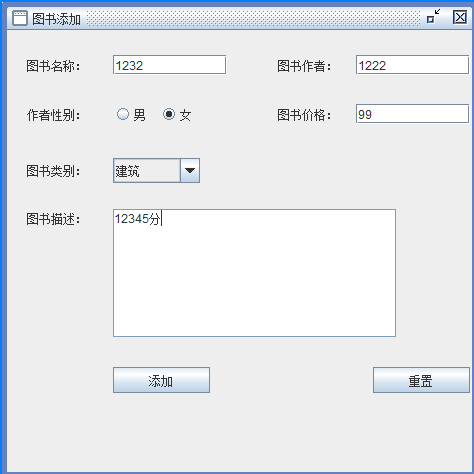


1. 图书管理分为图书添加和图书维护两个功能实现：

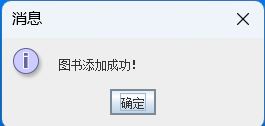


图书添加：





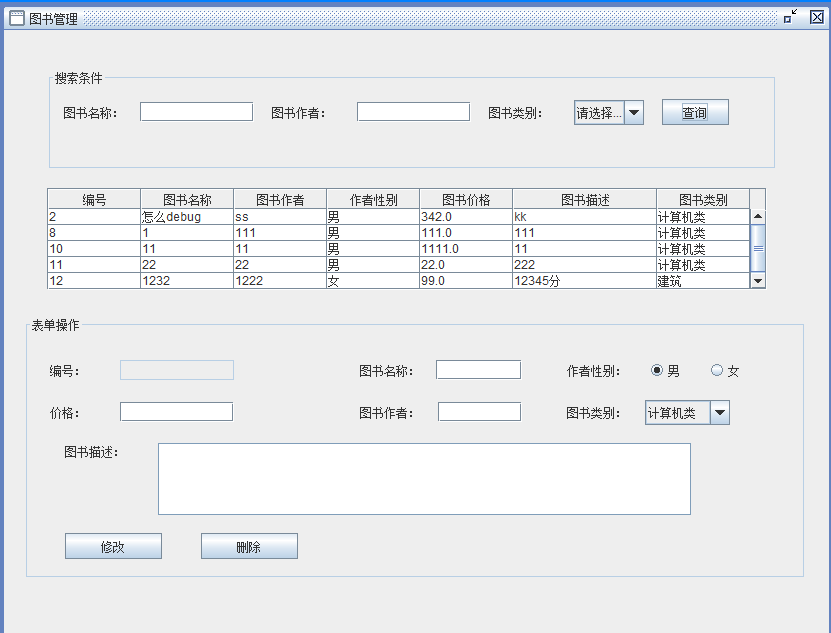
点击添加，添加成功显示：



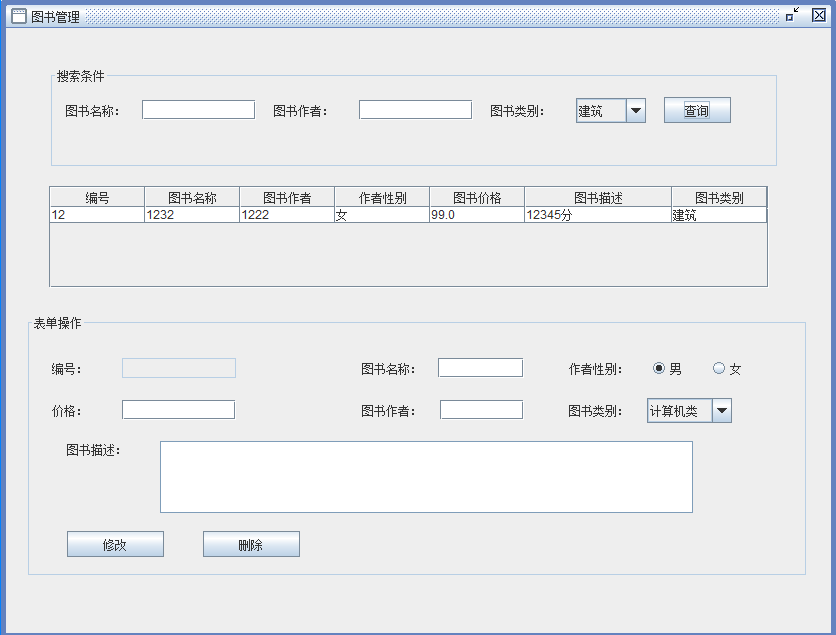
数据库信息随之改变：



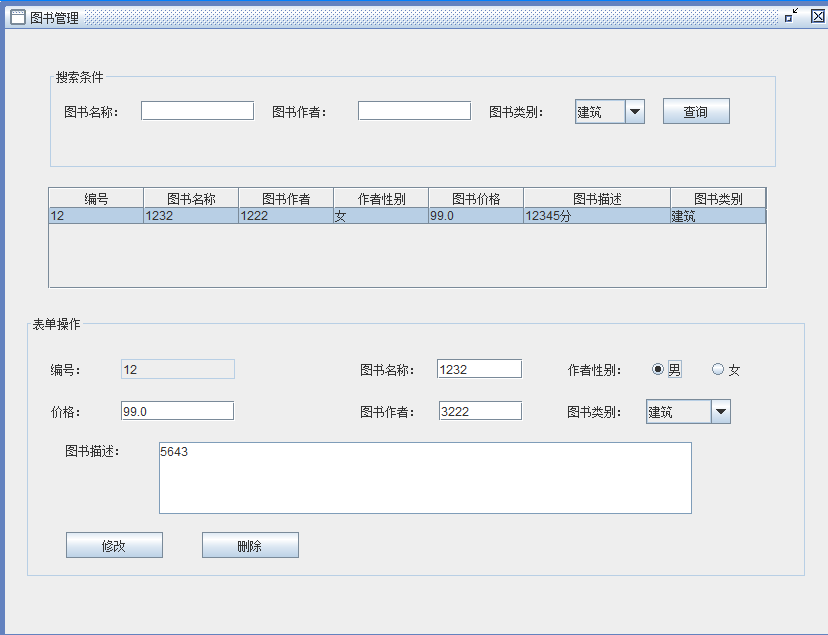
图书管理：



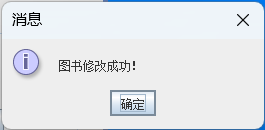
点击查询：



可在表单操作处进行图书的修改与删除：



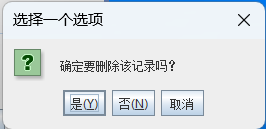
输入修改信息并点击，成功修改则显示：



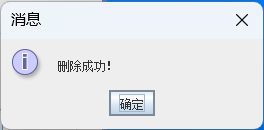
数据库信息随之改变：



选中某本图书点击删除：



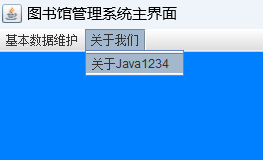
删除成功则显示：



数据库信息随之改变：



1. 关于我们：



## 7.3 事务设计

该系统中，所有关于查询的是无基本上都可以由select语句实现，而更新事务则用update语句，删除的事务则用delete语句实现。注意：注意：各种信息的插入和删除前都设计了条件查询，查询是否已经存在，修改前都设计了修改信息是否正确。

**7.3.1 图书类别事务**

有关图书类别的事务包括添加、查询、删除、修改图书类别信息等。具体实现代码如下：

//图书类别添加

JButton btnNewButton = new JButton("添加");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookTypeAddActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetValueActionPerformed(e);

}

});

/\*\*

\* 图书类别添加事件处理

\* @param e

\*/

private void bookTypeAddActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String bookTypeName=this.bookTypeNameTxt.getText();

String bookTypeDesc=this.bookTypeDescTxt.getText();

if(StringUtil.isEmpty(bookTypeName)) {

JOptionPane.showMessageDialog(null, "图书类别不能为空！");

return;

}

BookType bookType=new BookType(bookTypeName,bookTypeDesc);

Connection con=null;

try {

con=dbUtil.getCon();

int n=bookTypeDao.add(con, bookType);

if(n==1){

JOptionPane.showMessageDialog(null, "图书类别添加成功！");

resetValue();

}else {

JOptionPane.showMessageDialog(null, "图书类别添加失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "图书类别添加失败！");

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 重置事件处理

\* @param evt

\*/

private void resetValueActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

this.resetValue();

}

private void resetValue() {

this.bookTypeNameTxt.setText("");

this.bookTypeDescTxt.setText("");

}

}

//图书类别管理（包括修改图书类别信息、删除等）

JButton btnNewButton = new JButton("查询");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookTypeSearchActionPerformed(e);

}

});

bookTypeTable = new JTable();

bookTypeTable.addMouseListener(new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

bookTypeTableMousePressed(e);

}

});

JButton btnNewButton\_1 = new JButton("修改");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookTypeUpdateActionEvent(e);

}

});

JButton btnNewButton\_2 = new JButton("删除");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookTypeDeleteActionEvent(e);

}

});

/\*\*

\* 图书类别删除事件处理

\* @param e

\*/

private void bookTypeDeleteActionEvent(ActionEvent evt) {

String id=idTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要删除的记录！");

return;

}

int n=JOptionPane.showConfirmDialog(null, "确定要删除该记录吗？");

if(n==0) {

Connection con=null;

try {

con=dbUtil.getCon();

int deleteNum=bookTypeDao.delete(con, id);

if(deleteNum==1) {

JOptionPane.showMessageDialog(null, "删除成功！");

this.resetValue();

this.fillTable(new BookType());

}else {

JOptionPane.showMessageDialog(null, "删除失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "删除失败！");

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

}

/\*\*

\* 图书类别修改事件处理

\* @param evt

\*/

private void bookTypeUpdateActionEvent(ActionEvent evt) {

String id=idTxt.getText();

String bookTypeName=bookTypeNameTxt.getText();

String bookTypeDesc=bookTypeDescTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要修改的记录！");

return;

}

if(StringUtil.isEmpty(bookTypeName)) {

JOptionPane.showMessageDialog(null, "图书类别名称不能为空！");

return;

}

BookType bookType=new BookType(Integer.parseInt(id),bookTypeName,bookTypeDesc);

Connection con=null;

try {

con=dbUtil.getCon();

int modifyNum=bookTypeDao.update(con, bookType);

if(modifyNum==1) {

JOptionPane.showMessageDialog(null, "修改成功！");

this.resetValue();

this.fillTable(new BookType());

}else {

JOptionPane.showMessageDialog(null, "修改失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "修改失败！");

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

/\*\*

\* 表格行点击事件处理

\* @param e

\*/

private void bookTypeTableMousePressed(MouseEvent evt) {

int row=bookTypeTable.getSelectedRow();

idTxt.setText((String)bookTypeTable.getValueAt(row, 0));

bookTypeNameTxt.setText((String)bookTypeTable.getValueAt(row, 1));

bookTypeDescTxt.setText((String)bookTypeTable.getValueAt(row, 2));

}

/\*\*

\* 图书类别搜索事件处理

\* @param evt

\*/

private void bookTypeSearchActionPerformed(ActionEvent evt) {

String s\_bookTypeName=this.s\_bookTypeNameTxt.getText();

BookType bookType=new BookType();

bookType.setBookTypeName("s\_bookTypeName");

this.fillTable(bookType);

}

/\*\*

\* 初始化表格

\*/

private void fillTable(BookType bookType) {

DefaultTableModel dtm=(DefaultTableModel) bookTypeTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, bookType);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookTypeName"));

v.add(rs.getString("bookTypeDesc"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 重置表单

\*/

private void resetValue() {

this.idTxt.setText("");

this.bookTypeNameTxt.setText("");

this.bookTypeDescTxt.setText("");

}

}

7.3.2 图书事务

有关图书的事务包括查询、删除、修改图书信息等。具体实现代码如下：

有关查询的事务的具体实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

VisitorSearch frame = new VisitorSearch();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

JButton btnNewButton = new JButton("查询");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

bookSearchActionPerfomed(evt);

}

});

/\*\*

\* 图书查询事件处理

\* @param e

\*/

private void bookSearchActionPerfomed(ActionEvent evt) {

// TODO Auto-generated method stub

String bookName=this.s\_bookNameTxt.getText();

String author=this.s\_authorTxt.getText();

BookType bookType=(BookType) this.s\_bookTypeJcb.getSelectedItem();

int bookTypeId=bookType.getId();

}

/\*\*

\* 初始化下拉框

\*/

private void fillBookType(String type) {

Connection con=null;

BookType bookType=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, new BookType());

if("search".equals(type)) {

bookType=new BookType();

bookType.setBookTypeName("请选择...");

bookType.setId(-1);

this.s\_bookTypeJcb.addItem(bookType);

}

while(rs.next()) {

bookType=new BookType();

bookType.setBookTypeName(rs.getString("bookTypeName"));

bookType.setId(rs.getInt("id"));

this.s\_bookTypeJcb.addItem(bookType);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 初始化表格数据

\* @param book

\*/

private void fillTable(Book book) {

DefaultTableModel dtm=(DefaultTableModel) bookTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookDao.list(con, book);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookName"));

v.add(rs.getString("author"));

v.add(rs.getString("sex"));

v.add(rs.getFloat("price"));

v.add(rs.getString("bookDesc"));

v.add(rs.getString("bookTypeName"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

有关图书添加实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

BookAddInterFrme frame = new BookAddInterFrme();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

JButton btnNewButton = new JButton("添加");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookAddActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetValueActionPerformed(e);

}

});

manJrb = new JRadioButton("男");

buttonGroup.add(manJrb);

manJrb.setSelected(true);

femaleJrb = new JRadioButton("女");

/\*\*

\* 重置事件处理

\* @param e

\*/

private void resetValueActionPerformed(ActionEvent e) {

this.resetValue();

}

/\*\*

\* 图书添加事件处理

\* @param e

\*/

protected void bookAddActionPerformed(ActionEvent evt) {

String bookName=this.bookNameTxt.getText();

String author=this.authorTxt.getText();

String price=this.priceTxt.getText();

String bookDesc=this.bookDescTxt.getText();

if(StringUtil.isEmpty(bookName)) {

JOptionPane.showMessageDialog(null, "图书名称不能为空！");

return;

}

if(StringUtil.isEmpty(author)) {

JOptionPane.showMessageDialog(null, "图书作者不能为空！");

return;

}

if(StringUtil.isEmpty(price)) {

JOptionPane.showMessageDialog(null, "图书价格不能为空！");

return;

}

String sex="";

if(manJrb.isSelected()) {

sex="男";

}else if(femaleJrb.isSelected()){

sex="女";

}

BookType bookType=(BookType) bookTypeJcb.getSelectedItem();

int bookTypeId=bookType.getId();

Book book=new Book(bookName,author,sex,Float.parseFloat(price) ,bookTypeId,bookDesc);

Connection con=null;

try {

con=dbUtil.getCon();

int addNum=bookDao.add(con, book);

if(addNum==1){

JOptionPane.showMessageDialog(null, "图书添加成功！");

resetValue();

}else {

JOptionPane.showMessageDialog(null, "图书添加失败！");

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

/\*\*

\* 重置表单

\*/

private void resetValue() {

this.bookNameTxt.setText("");

this.authorTxt.setText("");

this.priceTxt.setText("");

this.manJrb.setSelected(true);

this.bookDescTxt.setText("");

if(this.bookTypeJcb.getItemCount()>0) {

this.bookTypeJcb.setSelectedIndex(0);

}

}

/\*\*

\* 初始化图书类别

\*/

private void fillBookType() {

Connection con=null;

BookType bookType=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, new BookType());

while(rs.next()) {

bookType=new BookType();

bookType.setId(rs.getInt("id"));

bookType.setBookTypeName(rs.getString("bookTypeName"));

this.bookTypeJcb.addItem(bookType);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

}

}

}

有关图书信息管理（包括删除、信息修改维护等）实现代码如下：

public class BookManageInterFrm extends JInternalFrame {

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

BookManageInterFrm frame = new BookManageInterFrm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public BookManageInterFrm() {

JButton btnNewButton\_1 = new JButton("修改");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

bookUpdateActionperformed(evt);

}

});

JButton btnNewButton\_2 = new JButton("删除");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

bookDeleteActionPerformed(evt);

}

});

JButton btnNewButton = new JButton("查询");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookSearchActionPerfomed(e);

}

});

this.fillBookType("search");

this.fillBookType("modify");

this.fillTable(new Book());

}

/\*\*

\* 图书删除事件处理

\* @param evt

\*/

private void bookDeleteActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String id=idTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要删除的记录！");

return;

}

int n=JOptionPane.showConfirmDialog(null, "确定要删除该记录吗？");

if(n==0) {

Connection con=null;

try {

con=dbUtil.getCon();

int deleteNum=bookDao.delete(con, id);

if(deleteNum==1) {

JOptionPane.showMessageDialog(null, "删除成功！");

this.resetValue();

this.fillTable(new Book());

}else {

JOptionPane.showMessageDialog(null, "删除失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "删除失败！");

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

}

/\*\*

\* 图书修改事件处理

\* @param evt

\*/

private void bookUpdateActionperformed(ActionEvent evt) {

// TODO Auto-generated method stub

String id=this.idTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要修改的记录！");

return;

}

String bookName=this.bookNameTxt.getText();

String author=this.authorTxt.getText();

String price=this.priceTxt.getText();

String bookDesc=this.bookDescTxt.getText();

if(StringUtil.isEmpty(bookName)) {

JOptionPane.showMessageDialog(null, "图书名称不能为空！");

return;

}

if(StringUtil.isEmpty(author)) {

JOptionPane.showMessageDialog(null, "图书作者不能为空！");

return;

}

if(StringUtil.isEmpty(price)) {

JOptionPane.showMessageDialog(null, "图书价格不能为空！");

return;

}

String sex="";

if(manJrb.isSelected()) {

sex="男";

}else if(femaleJrb.isSelected()){

sex="女";

}

BookType bookType=(BookType) bookTypeJcb.getSelectedItem();

int bookTypeId=bookType.getId();

Book book=new Book(Integer.parseInt(id),bookName,author,sex,Float.parseFloat(price),bookTypeId,bookDesc);

Connection con=null;

try {

con=dbUtil.getCon();

int addNum=bookDao.update(con, book);

if(addNum==1){

JOptionPane.showMessageDialog(null, "图书修改成功！");

resetValue();

this.fillTable(new Book());

}else {

JOptionPane.showMessageDialog(null, "图书修改失败！");

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

/\*\*

\* 重置表单

\*/

private void resetValue() {

this.idTxt.setText("");

this.bookNameTxt.setText("");

this.authorTxt.setText("");

this.priceTxt.setText("");

this.manJrb.setSelected(true);

this.bookDescTxt.setText("");

if(this.bookTypeJcb.getItemCount()>0) {

this.bookTypeJcb.setSelectedIndex(0);

}

}

/\*\*

\* 表格行点击事件处理

\* @param met

\*/

private void bookTableMousePressed(MouseEvent met) {

// TODO Auto-generated method stub

int row=this.bookTable.getSelectedRow();

this.idTxt.setText((String)bookTable.getValueAt(row, 0));

this.bookNameTxt.setText((String)bookTable.getValueAt(row, 1));

this.authorTxt.setText((String)bookTable.getValueAt(row, 2));

String sex=(String)bookTable.getValueAt(row, 3);

if("男".equals(sex)) {

this.manJrb.setSelected(true);

}else if("女".equals(sex)) {

this.femaleJrb.setSelected(true);

}

this.priceTxt.setText(bookTable.getValueAt(row, 4)+"");

this.bookDescTxt.setText((String)bookTable.getValueAt(row, 5));

String bookTypeName=(String)this.bookTable.getValueAt(row, 6);

int n=this.bookTypeJcb.getItemCount();

for(int i=0;i<n;i++)

{

BookType item=(BookType)this.bookTypeJcb.getItemAt(i);

if(item.getBookTypeName().equals(bookTypeName)) {

this.bookTypeJcb.setSelectedIndex(i);

}

}

}

/\*\*

\* 图书查询事件处理

\* @param e

\*/

private void bookSearchActionPerfomed(ActionEvent evt) {

// TODO Auto-generated method stub

String bookName=this.s\_bookNameTxt.getText();

String author=this.s\_authorTxt.getText();

BookType bookType=(BookType) this.s\_bookTypeJcb.getSelectedItem();

int bookTypeId=bookType.getId();

Book book=new Book(bookName,author,bookTypeId);

this.fillTable(book);

}

/\*\*

\* 初始化下拉框

\*/

private void fillBookType(String type) {

Connection con=null;

BookType bookType=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, new BookType());

if("search".equals(type)) {

bookType=new BookType();

bookType.setBookTypeName("请选择...");

bookType.setId(-1);

this.s\_bookTypeJcb.addItem(bookType);

}

while(rs.next()) {

bookType=new BookType();

bookType.setBookTypeName(rs.getString("bookTypeName"));

bookType.setId(rs.getInt("id"));

if("search".equals(type)) {

this.s\_bookTypeJcb.addItem(bookType);

}else if("modify".equals(type)) {

this.bookTypeJcb.addItem(bookType);

}

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 初始化表格数据

\* @param book

\*/

private void fillTable(Book book) {

DefaultTableModel dtm=(DefaultTableModel) bookTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookDao.list(con, book);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookName"));

v.add(rs.getString("author"));

v.add(rs.getString("sex"));

v.add(rs.getFloat("price"));

v.add(rs.getString("bookDesc"));

v.add(rs.getString("bookTypeName"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

7.3.3 管理员事务

有关管理员事务包括管理员注册、管理员登录。具体实现代码如下：

管理员注册代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

Register frame = new Register();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public Register() {

JButton btnNewButton = new JButton("注 册");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

RegisterActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重 置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ResetValueActionPerformed(e);

}

});

JButton btnNewButton\_2 = new JButton("点此登录");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

LogInActionPerformed(e);

}

});

JButton btnNewButton\_3 = new JButton("返回");

btnNewButton\_3.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

BackActionPerformed(e);

}

});

private void BackActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new RegisterChoose().setVisible(true);

}

private void LogInActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new LogOnFrm().setVisible(true);

}

private void ResetValueActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

this.resetValue();

}

private void resetValue() {

// TODO Auto-generated method stub

this.AdmNametxt.setText("");

this.AdmPasswordtxt.setText("");

this.AdmPasswordtxtRE.setText("");

}

private void RegisterActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String userName=this.AdmNametxt.getText();

String userPassword=new String(this.AdmPasswordtxt.getPassword());

String userPasswordRE=new String(this.AdmPasswordtxtRE.getPassword());

if(StringUtil.isEmpty(userName)) {

JOptionPane.showMessageDialog(null, "账号不能为空！");

return;

}

if(userName.length()>20) {

JOptionPane.showMessageDialog(null, "账号长度超过限制！");

}

if(StringUtil.isEmpty(userPassword)) {

JOptionPane.showMessageDialog(null, "密码不能为空！");

return;

}

if(userPassword.length()>20) {

JOptionPane.showMessageDialog(null, "密码长度超过限制！");

}

if(StringUtil.isEmpty(userPasswordRE)) {

JOptionPane.showMessageDialog(null, "请确认密码！");

return;

}

if(!userPassword.equals(userPasswordRE)) {

JOptionPane.showMessageDialog(null, "两次输入的密码不一致，请重新输入！");

return;

}

User user=new User(userName,userPassword);

Connection con=null;

try {

con=dbUtil.getCon();

int n=userDao.add(con, user);

if(n==1) {

JOptionPane.showMessageDialog(null, "注册成功！");

resetValue();

}else {

JOptionPane.showMessageDialog(null, "注册失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "注册失败！");

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

管理员登录具体实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

LogOnFrm frame = new LogOnFrm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public LogOnFrm() {

JButton btnNewButton = new JButton("登录");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

loginActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetValueActionPerformed(e);

}

});

JButton btnNewButton\_2 = new JButton("返回");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

BackActionPerformed(e);

}

});

}

private void BackActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new StartFrm().setVisible(true);

}

private void loginActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

String userName=this.userNametxt.getText();

String password=new String(this.passwordtxt.getPassword());

if(StringUtil.isEmpty(userName)) {

JOptionPane.showMessageDialog(null, "用户名不能为空！");

return;

}

if(StringUtil.isEmpty(password)) {

JOptionPane.showMessageDialog(null, "密码不能为空！");

return;

}

User user=new User(userName,password);

Connection con=null;

try {

con=dbUtil.getCon();

User currentUser=userDao.login(con, user);

if(currentUser!=null) {

dispose();

new MainFrm().setVisible(true);

}else {

JOptionPane.showMessageDialog(null, "用户名或密码错误！");

}

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

}

/\*\*

\*

\* @param evt

\*/

private void resetValueActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

this.userNametxt.setText("");

this.passwordtxt.setText("");

}

}

7.3.4 读者事务

有关读者事务包括读者注册、读者登录、借还书等操作。具体实现代码如下：

读者注册代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

ReaderRegister frame = new ReaderRegister();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ReaderRegister() {

JButton btnNewButton = new JButton("注 册");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

readerRegisterActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重 置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ResetValueActionPerformed(e);

}

});

JLabel lblNewLabel\_4 = new JLabel("（20个字符以内的数字、字母组合）");

JLabel lblNewLabel\_4\_1 = new JLabel("（20个字符以内的数字、字母组合）");

JLabel lblNewLabel\_4\_1\_1 = new JLabel("（20个字符以内的数字、字母组合）");

JButton btnNewButton\_2 = new JButton("点此登录");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ReaderLogInActionPerformed(e);

}

});

JButton btnNewButton\_3 = new JButton("返回");

btnNewButton\_3.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ReaderBackActionPerformed(e);

}

});

}

private void ReaderBackActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new RegisterChoose().setVisible(true);

}

private void ReaderLogInActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new ReaderLogOnFrm().setVisible(true);

}

private void readerRegisterActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String readerName=this.readerNametxt.getText();

String readerPassword=new String(this.readerPasswordtxt.getPassword());

String readerPasswordRE=new String(this.readerPasswordtxtRE.getPassword());

if(StringUtil.isEmpty(readerName)) {

JOptionPane.showMessageDialog(null, "账号不能为空！");

return;

}

if(readerName.length()>20) {

JOptionPane.showMessageDialog(null, "账号长度超过限制！");

return;

}

if(StringUtil.isEmpty(readerPassword)) {

JOptionPane.showMessageDialog(null, "密码不能为空！");

return;

}

if(readerPassword.length()>20) {

JOptionPane.showMessageDialog(null, "密码长度超过限制！");

return;

}

if(StringUtil.isEmpty(readerPasswordRE)) {

JOptionPane.showMessageDialog(null, "请确认密码！");

return;

}

if(!readerPassword.equals(readerPasswordRE)) {

JOptionPane.showMessageDialog(null, "两次输入的密码不一致，请重新输入！");

return;

}

Reader reader=new Reader(readerName,readerPassword);

Connection con=null;

try {

con=dbUtil.getCon();

int n=readerDao.add(con, reader);

if(n==1) {

JOptionPane.showMessageDialog(null, "注册成功！");

resetValue();

}else {

JOptionPane.showMessageDialog(null, "注册失败！");

}

}catch(Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "注册失败！");

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

private void ResetValueActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

this.resetValue();

}

private void resetValue() {

this.readerNametxt.setText("");

this.readerPasswordtxt.setText("");

this.readerPasswordtxtRE.setText("");

}

}

读者登录实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

ReaderLogOnFrm frame = new ReaderLogOnFrm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ReaderLogOnFrm() {

JButton btnNewButton = new JButton("登录");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

loginActionPerformed(e);

}

});

JButton btnNewButton\_1 = new JButton("重置");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetValueActionPerformed(e);

}

});

JButton btnNewButton\_2 = new JButton("返回");

btnNewButton\_2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

BackActionPerformed(e);

}

});

}

private void BackActionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

dispose();

new StartFrm().setVisible(true);

}

/\*\*

\* 登录事件处理

\* @param e

\*/

private void loginActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String reader\_name=this.ReaderNameTxt.getText();

String reader\_password=new String(this.ReaderpasswordTxt.getPassword());

if(StringUtil.isEmpty(reader\_name)) {

JOptionPane.showMessageDialog(null, "用户名不能为空！");

return;

}

if(StringUtil.isEmpty(reader\_password)) {

JOptionPane.showMessageDialog(null, "密码不能为空！");

return;

}

Reader reader=new Reader(reader\_name,reader\_password);

Connection con=null;

try {

con=dbUtil.getCon();

Reader currentReader=readerDao.login(con, reader);

if(currentReader!=null) {

dispose();

new ReaderMainFrm().setVisible(true);

}

else {

JOptionPane.showMessageDialog(null, "用户名或密码错误！");

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

\* 重置事件处理

\* @param e

\*/

private void resetValueActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

this.ReaderNameTxt.setText("");

this.ReaderpasswordTxt.setText("");

}

}

读者借书实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

ReaderBorrowFrm frame = new ReaderBorrowFrm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ReaderBorrowFrm() {

JButton btnNewButton\_1 = new JButton("借阅");

btnNewButton\_1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ReaderBorrowActionPerformed(e);

}

});

JButton btnNewButton = new JButton("查询");

btnNewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

bookSearchActionPerfomed(evt);

}

});

}

/\*\*

\* 表格行点击事件处理

\* @param met

\*/

private void ReaderBorrowMousePressed(MouseEvent met) {

// TODO Auto-generated method stub

int row=this.bookTable.getSelectedRow();

this.idTxt.setText((String)bookTable.getValueAt(row, 0));

this.bookNameTxt.setText((String)bookTable.getValueAt(row, 1));

this.authorTxt.setText((String)bookTable.getValueAt(row, 2));

String sex=(String)bookTable.getValueAt(row, 3);

if("男".equals(sex)) {

this.manJrb.setSelected(true);

}else if("女".equals(sex)) {

this.femaleJrb.setSelected(true);

}

this.priceTxt.setText(bookTable.getValueAt(row, 4)+"");

this.bookDescTxt.setText((String)bookTable.getValueAt(row, 5));

String bookTypeName=(String)this.bookTable.getValueAt(row, 6);

int n=this.bookTypeJcb.getItemCount();

for(int i=0;i<n;i++)

{

BookType item=(BookType)this.bookTypeJcb.getItemAt(i);

if(item.getBookTypeName().equals(bookTypeName)) {

this.bookTypeJcb.setSelectedIndex(i);

}

}

}

/\*\*

\* 图书借阅事件处理

\* @param e

\*/

private void ReaderBorrowActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String id=idTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要借阅的图书！");

return;

}

int n=JOptionPane.showConfirmDialog(null, "确定要借阅该图书吗？");

if(n==0) {

Connection con=null;

try {

con=dbUtil.getCon();

JOptionPane.showMessageDialog(null, "借阅成功！");

this.resetValue();

this.fillTable(new Book());

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

}

/\*\*

\* 图书查询事件处理

\* @param e

\*/

private void bookSearchActionPerfomed(ActionEvent evt) {

// TODO Auto-generated method stub

String bookName=this.s\_bookNameTxt.getText();

String author=this.s\_authorTxt.getText();

BookType bookType=(BookType) this.s\_bookTypeJcb\_1.getSelectedItem();

int bookTypeId=bookType.getId();

Book book=new Book(bookName,author,bookTypeId);

this.fillTable(book);

}

/\*\*

\* 重置表单

\*/

private void resetValue() {

this.idTxt.setText("");

this.bookNameTxt.setText("");

this.authorTxt.setText("");

this.priceTxt.setText("");

this.manJrb.setSelected(true);

this.bookDescTxt.setText("");

if(this.bookTypeJcb.getItemCount()>0) {

this.bookTypeJcb.setSelectedIndex(0);

}

}

/\*\*

\* 初始化下拉框

\*/

private void fillBookType(String type) {

Connection con=null;

BookType bookType=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, new BookType());

if("search".equals(type)) {

bookType=new BookType();

bookType.setBookTypeName("请选择...");

bookType.setId(-1);

this.s\_bookTypeJcb\_1.addItem(bookType);

}

while(rs.next()) {

bookType=new BookType();

bookType.setBookTypeName(rs.getString("bookTypeName"));

bookType.setId(rs.getInt("id"));

this.s\_bookTypeJcb\_1.addItem(bookType);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 初始化表格数据

\* @param book

\*/

private void fillTable(Book book) {

DefaultTableModel dtm=(DefaultTableModel) bookTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookDao.list(con, book);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookName"));

v.add(rs.getString("author"));

v.add(rs.getString("sex"));

v.add(rs.getFloat("price"));

v.add(rs.getString("bookDesc"));

v.add(rs.getString("bookTypeName"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

读者还书实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

ReaderReturnFrm frame = new ReaderReturnFrm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ReaderReturnFrm() {

/\*\*

\* 图书归还事件处理

\* @param e

\*/

private void ReaderReturnActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

String id=idTxt.getText();

if(StringUtil.isEmpty(id)) {

JOptionPane.showMessageDialog(null, "请选择要归还的图书！");

return;

}

int n=JOptionPane.showConfirmDialog(null, "确定要归还该图书吗？");

if(n==0) {

Connection con=null;

try {

con=dbUtil.getCon();

int deleteNum=bookDao.delete2(con, id);

if(deleteNum==1) {

JOptionPane.showMessageDialog(null, "归还成功！");

this.resetValue();

this.fillTable(new Book());

}else {

JOptionPane.showMessageDialog(null, "归还失败！");

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

e.printStackTrace();

}

}

}

}

/\*\*

\* 重置表单

\*/

private void resetValue() {

this.idTxt.setText("");

this.bookNameTxt.setText("");

this.authorTxt.setText("");

this.priceTxt.setText("");

this.manJrb.setSelected(true);

this.bookDescTxt\_1.setText("");

if(this.bookTypeJcb.getItemCount()>0) {

this.bookTypeJcb.setSelectedIndex(0);

}

}

/\*\*

\* 表格行点击事件处理

\* @param met

\*/

private void ReaderReturnMousePressed(MouseEvent met) {

// TODO Auto-generated method stub

int row=this.bookTable.getSelectedRow();

this.idTxt.setText((String)bookTable.getValueAt(row, 0));

this.bookNameTxt.setText((String)bookTable.getValueAt(row, 1));

this.authorTxt.setText((String)bookTable.getValueAt(row, 2));

String sex=(String)bookTable.getValueAt(row, 3);

if("男".equals(sex)) {

this.manJrb.setSelected(true);

}else if("女".equals(sex)) {

this.femaleJrb.setSelected(true);

}

this.priceTxt.setText(bookTable.getValueAt(row, 4)+"");

this.bookDescTxt\_1.setText((String)bookTable.getValueAt(row, 5));

String bookTypeName=(String)this.bookTable.getValueAt(row, 6);

int n=this.bookTypeJcb.getItemCount();

for(int i=0;i<n;i++)

{

BookType item=(BookType)this.bookTypeJcb.getItemAt(i);

if(item.getBookTypeName().equals(bookTypeName)) {

this.bookTypeJcb.setSelectedIndex(i);

}

}

}

/\*\*

\* 初始化表格数据

\* @param book

\*/

private void fillTable(Book book) {

DefaultTableModel dtm=(DefaultTableModel) bookTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookDao.list2(con, book);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookName"));

v.add(rs.getString("author"));

v.add(rs.getString("sex"));

v.add(rs.getFloat("price"));

v.add(rs.getString("bookDesc"));

v.add(rs.getString("bookTypeName"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

7.3.5 访客事务

有关访客的事务包括访客界面的图书信息查询。具体实现代码如下：

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

VisitorSearch frame = new VisitorSearch();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public VisitorSearch() {

/\*\*

\* 图书查询事件处理

\* @param e

\*/

private void bookSearchActionPerfomed(ActionEvent evt) {

// TODO Auto-generated method stub

String bookName=this.s\_bookNameTxt.getText();

String author=this.s\_authorTxt.getText();

BookType bookType=(BookType) this.s\_bookTypeJcb.getSelectedItem();

int bookTypeId=bookType.getId();

}

/\*\*

\* 初始化下拉框

\*/

private void fillBookType(String type) {

Connection con=null;

BookType bookType=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookTypeDao.list(con, new BookType());

if("search".equals(type)) {

bookType=new BookType();

bookType.setBookTypeName("请选择...");

bookType.setId(-1);

this.s\_bookTypeJcb.addItem(bookType);

}

while(rs.next()) {

bookType=new BookType();

bookType.setBookTypeName(rs.getString("bookTypeName"));

bookType.setId(rs.getInt("id"));

this.s\_bookTypeJcb.addItem(bookType);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

}catch(Exception e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 初始化表格数据

\* @param book

\*/

private void fillTable(Book book) {

DefaultTableModel dtm=(DefaultTableModel) bookTable.getModel();

dtm.setRowCount(0);//设置成0行

Connection con=null;

try {

con=dbUtil.getCon();

ResultSet rs=bookDao.list(con, book);

while(rs.next()) {

Vector v=new Vector();

v.add(rs.getString("id"));

v.add(rs.getString("bookName"));

v.add(rs.getString("author"));

v.add(rs.getString("sex"));

v.add(rs.getFloat("price"));

v.add(rs.getString("bookDesc"));

v.add(rs.getString("bookTypeName"));

dtm.addRow(v);

}

}catch(Exception e) {

e.printStackTrace();

}finally {

try {

dbUtil.closeCon(con);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

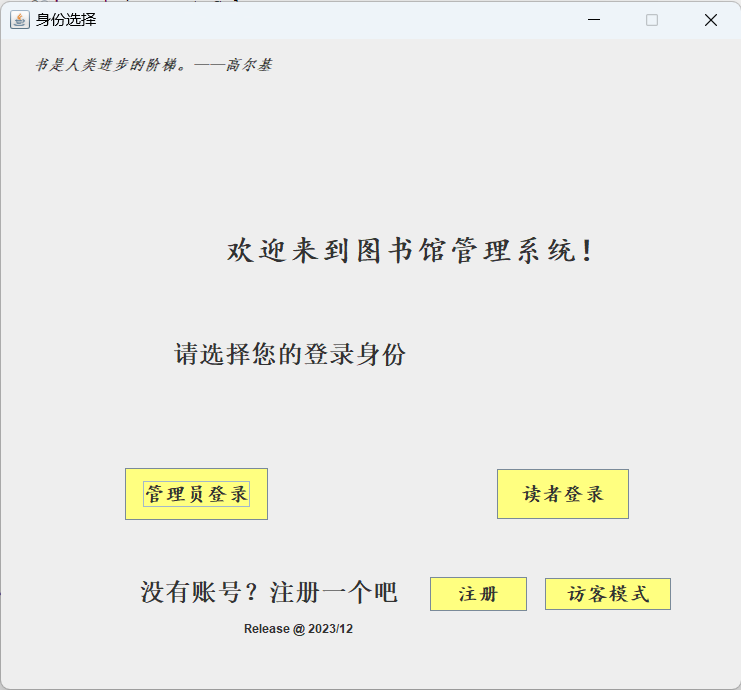
}

}

# 八、测试和运行

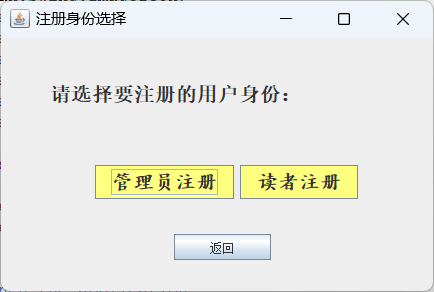
**1 主界面**

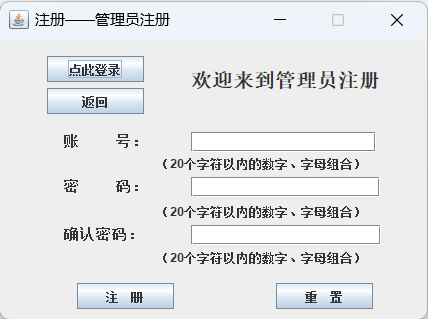
在主界面可以选择注册（注册能够注册管理员与读者）后，进行登录（访客模式无需注册和登录）：不同身份的账号密码各自独立；选择不同身份登录后的界面、权限等均不同；（为空或者输入错误均会提示）

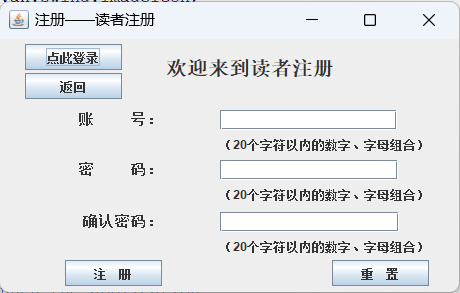


**2 注册界面**

在登录界面可选择注册按钮，进行注册：选择注册身份（管理员/读者），账户名，账号密码和确认密码；

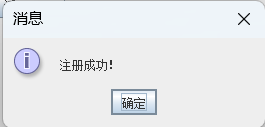






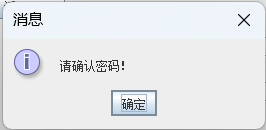
·正确输入情况：

会提示

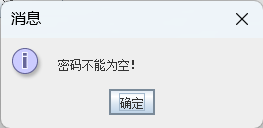


·几种不同错误输入情况：

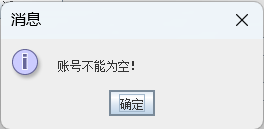
①如果确认密码处为空：



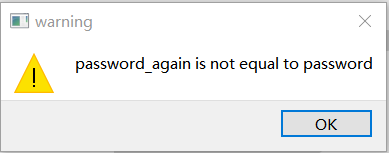
②如果未输入密码：



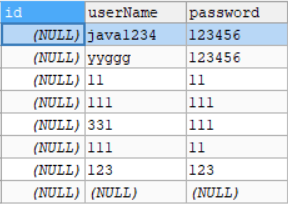
③如果未输入账号：



④如果确认密码和第一次输入密码不同，会提示：

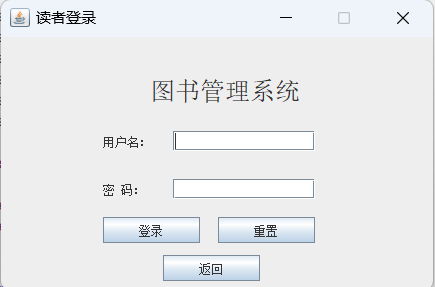


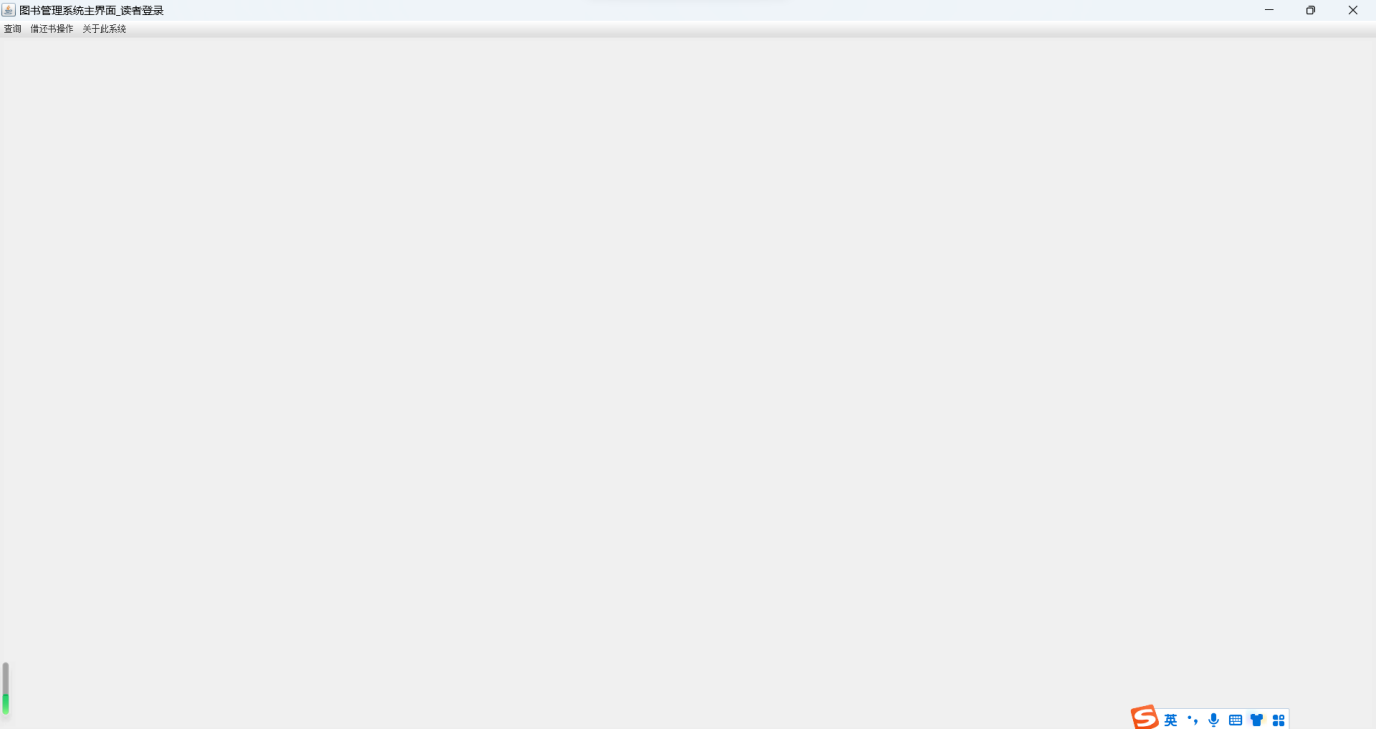
成功注册则表单信息：



**3 读者主界面**

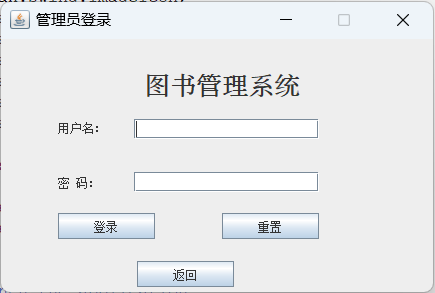
选择读者登录后，输入账号密码正确，就进入读者主界面：（右上方有退出按钮 可以点击退出系统）





**4 管理员主界面**

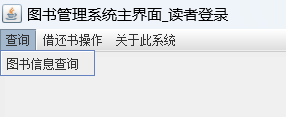
选择买家身份后，输入账号密码正确，就进入买家菜单界面：（右下方有退出按钮 可以点击退出系统）



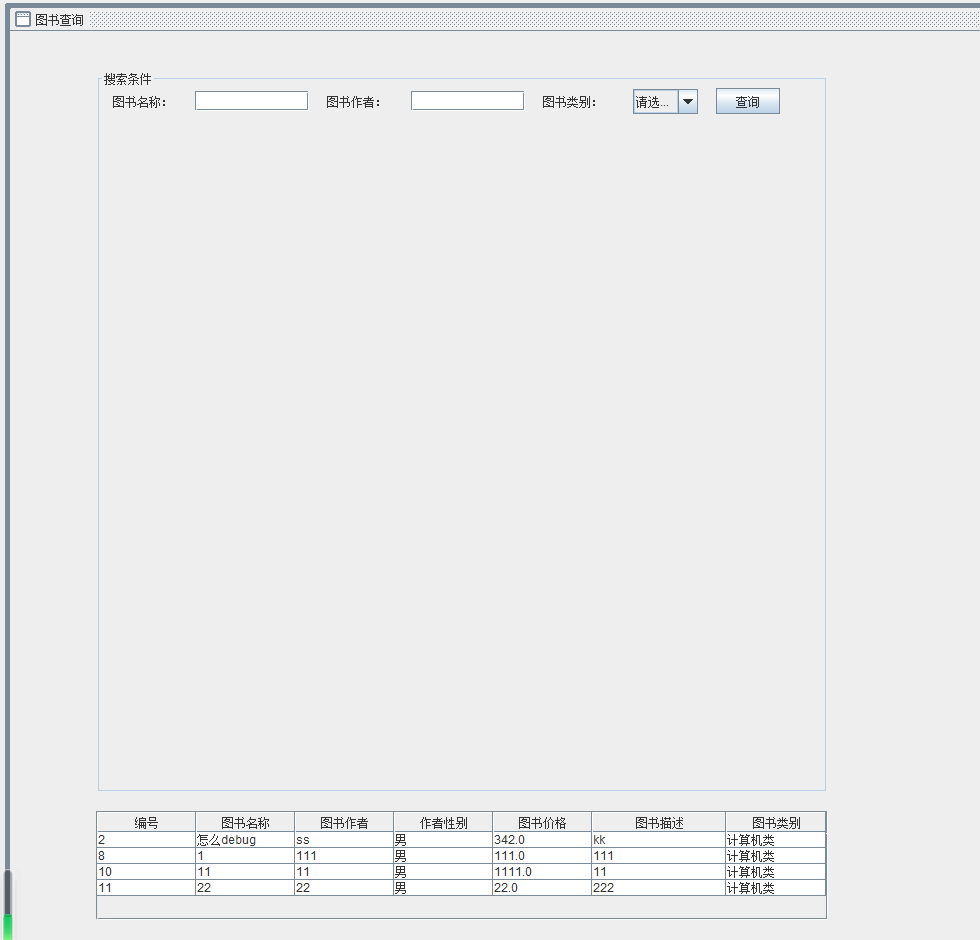
****

**5 读者功能实现（读者）**

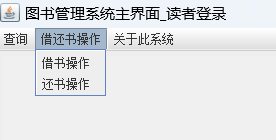
1. 选择读者身份后，输入账号密码正确，就进入读者界面，包括查询、借还书操作、关于此系统三个板块:
2. 其中查询——图书信息查询：



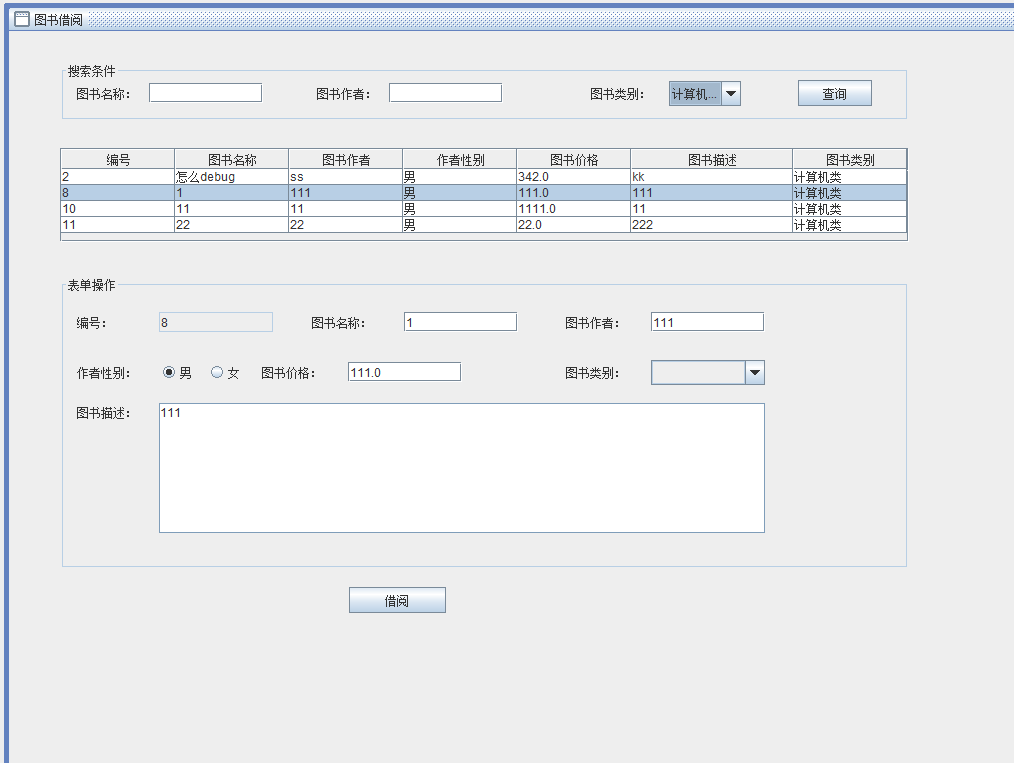
图书查询：



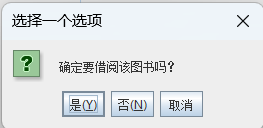
1. 借还书操作包括借书和还书：



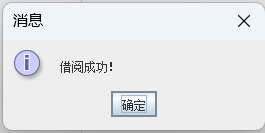
借书，通过查询找出要借阅的图书，选择图书：



点击借阅，显示：



借阅成功，显示：



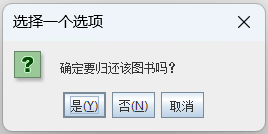
同时数据库中表单更新：



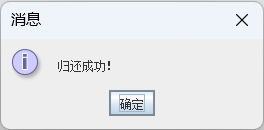
图书归还，上方界面显示所借图书：



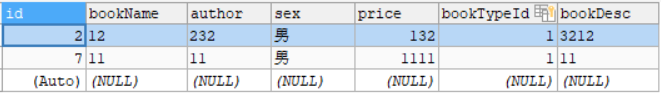
选择图书，下方表单操作处显示该书相关信息，点击归还，显示：



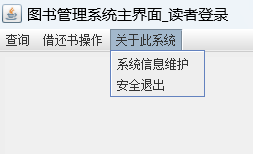
归还成功，则显示：



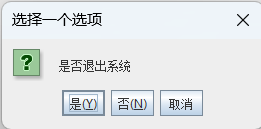
同时表单借书信息删除：



1. 关于此系统包括系统信息维护和安全退出：

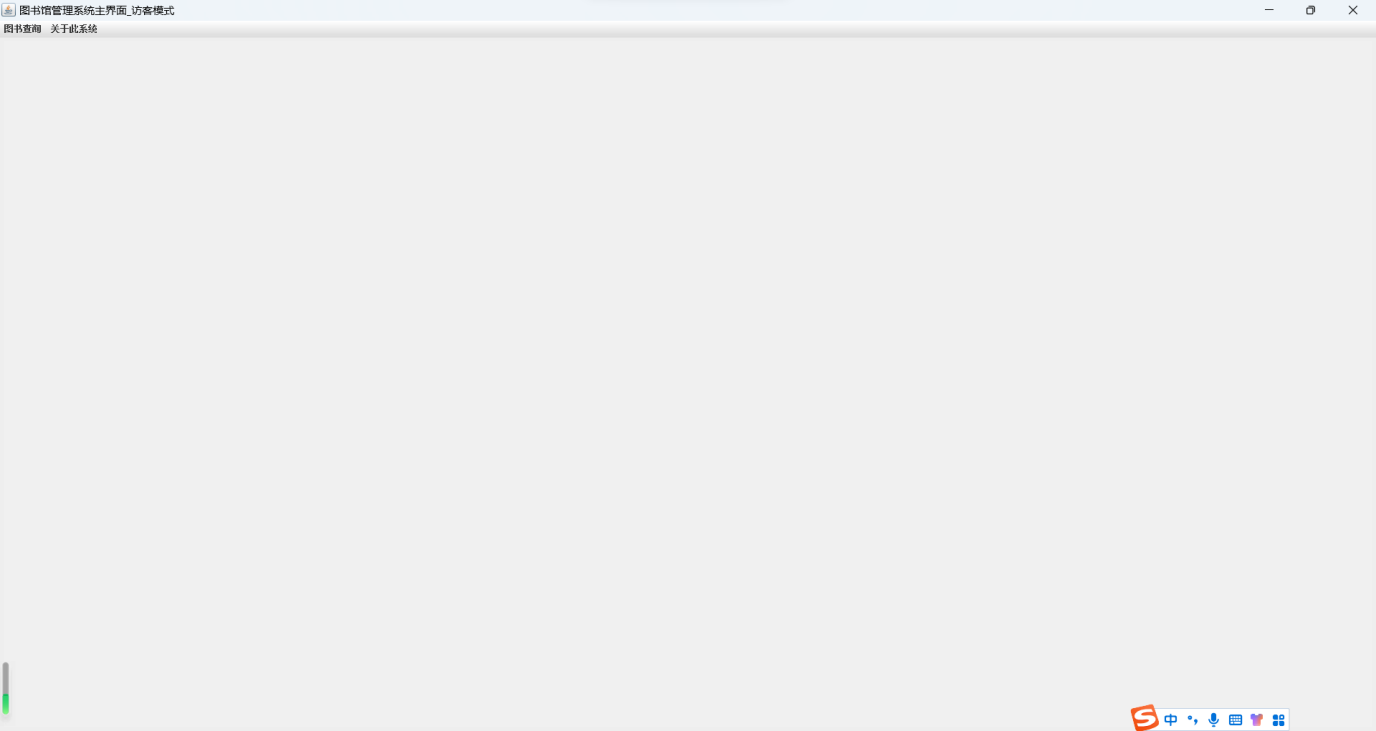


点击安全退出显示：

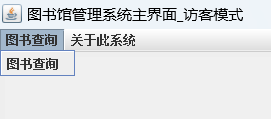


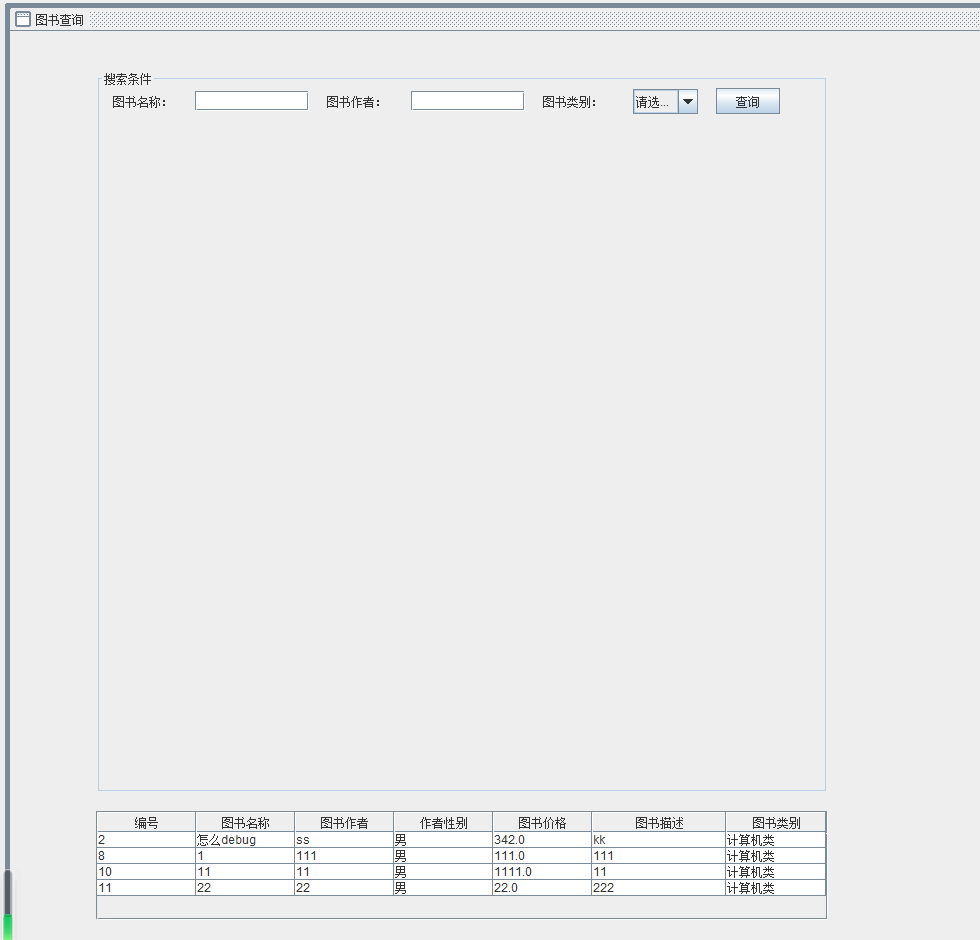
**6 访客模式**

1. 访客主界面如下图所示：

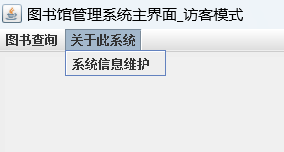


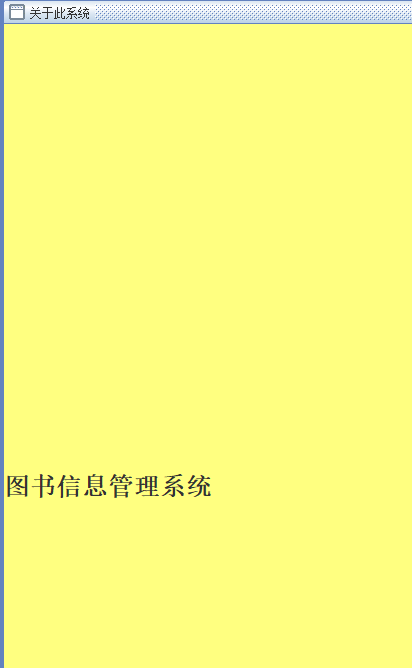
1. 图书查询：





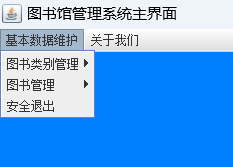
1. 关于此系统：



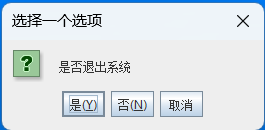


**7 基本数据维护（管理员）**

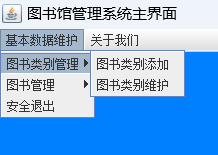
1. 基本数据维护包括图书类别管理、图书管理以及安全退出三个功能板块：



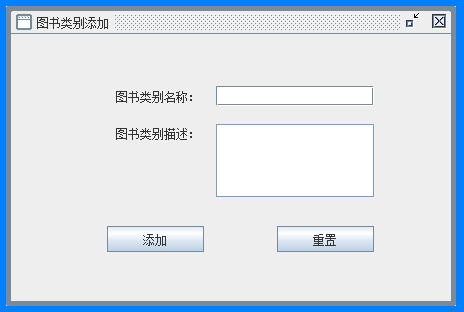
点击安全退出：



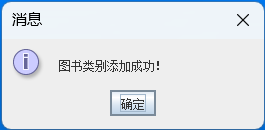
1. 图书类别管理分为图书类别添加和图书类别维护两个功能实现:



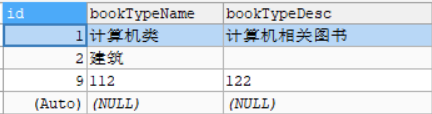
图书类别添加：



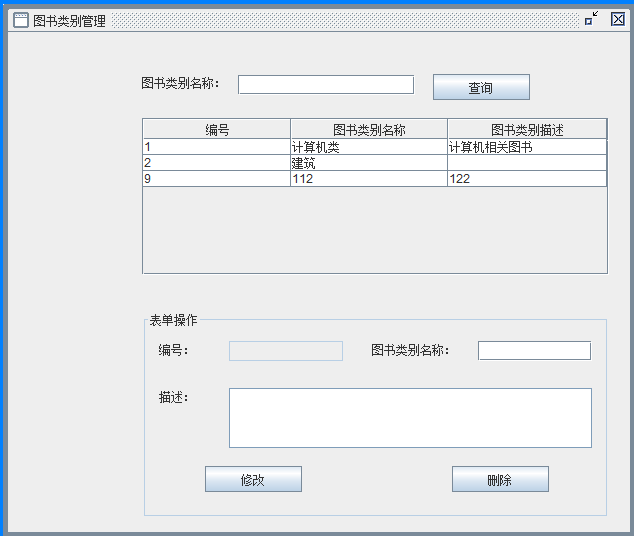
添加成功弹窗显示：



在数据库中显示：

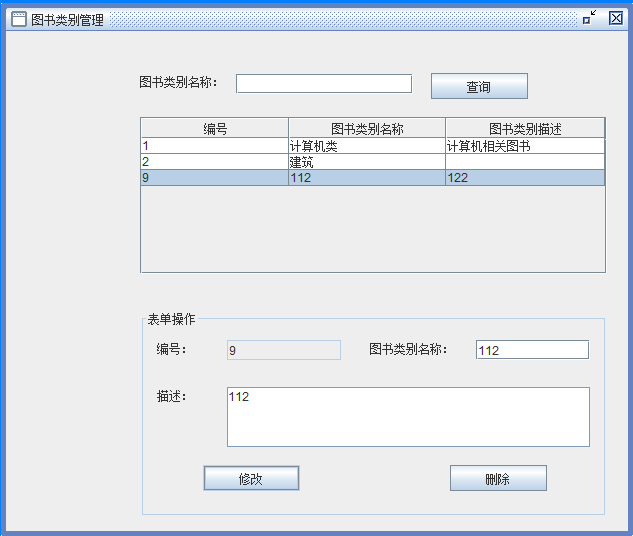


图书类别维护：

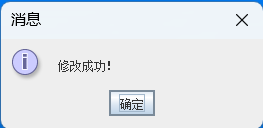


可以在表单操作处进行图书类别的修改与删除：

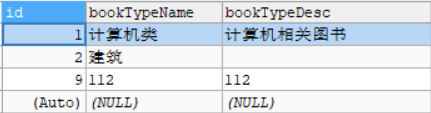
输入修改信息：



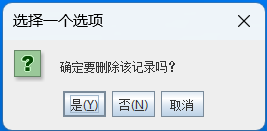
成功修改则显示：



数据库数据随之改变：



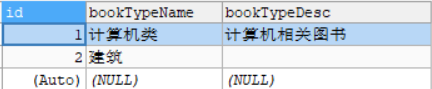
点击删除：



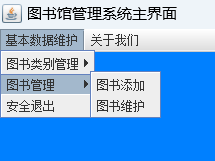
删除成功则显示：



数据库数据随之改变：

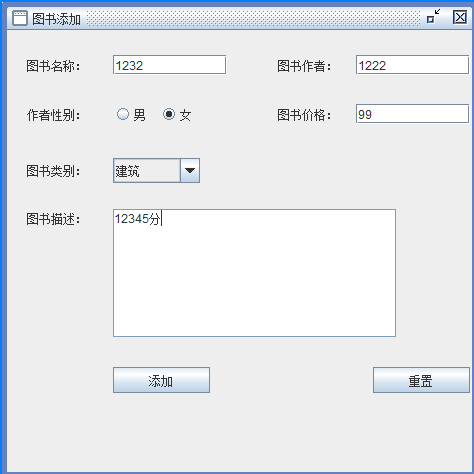


1. 图书管理分为图书添加和图书维护两个功能实现：

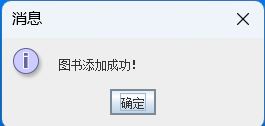


图书添加：





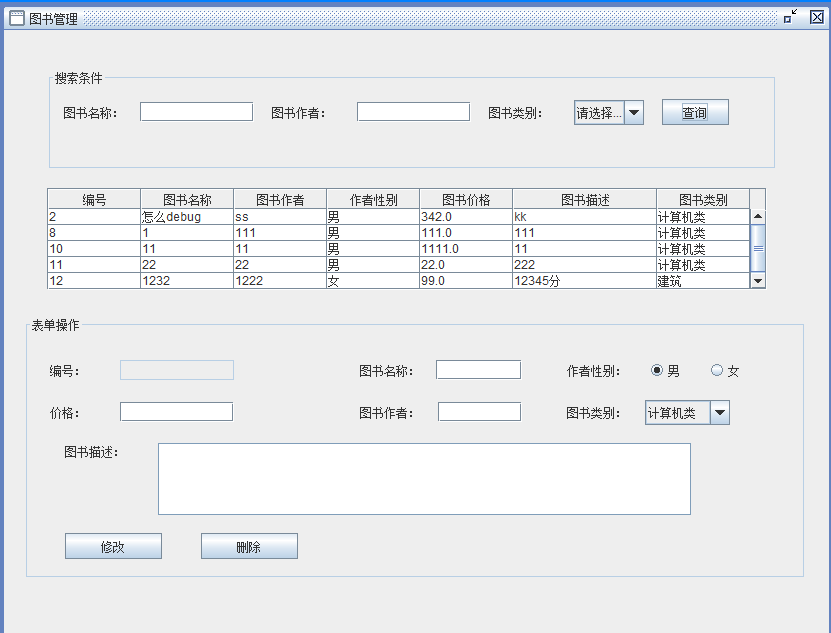
点击添加，添加成功显示：



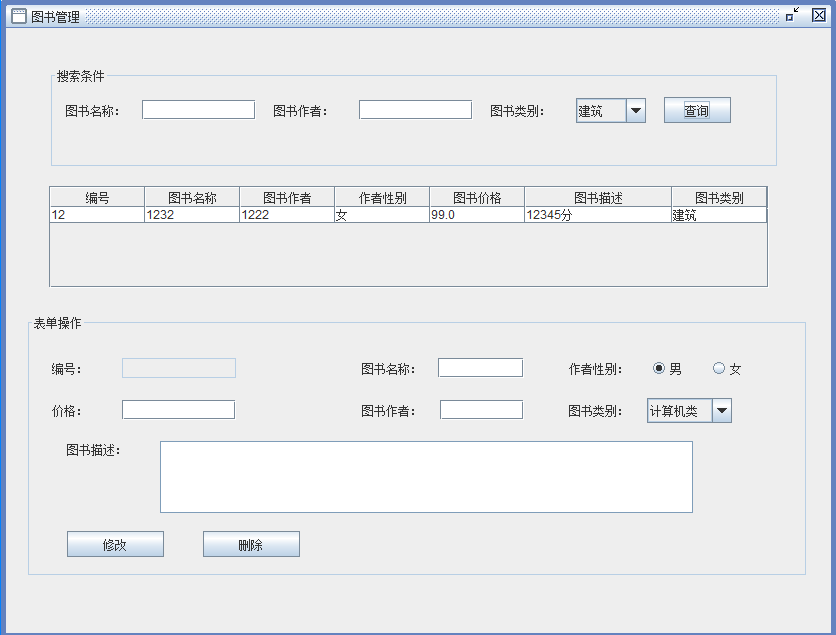
数据库信息随之改变：



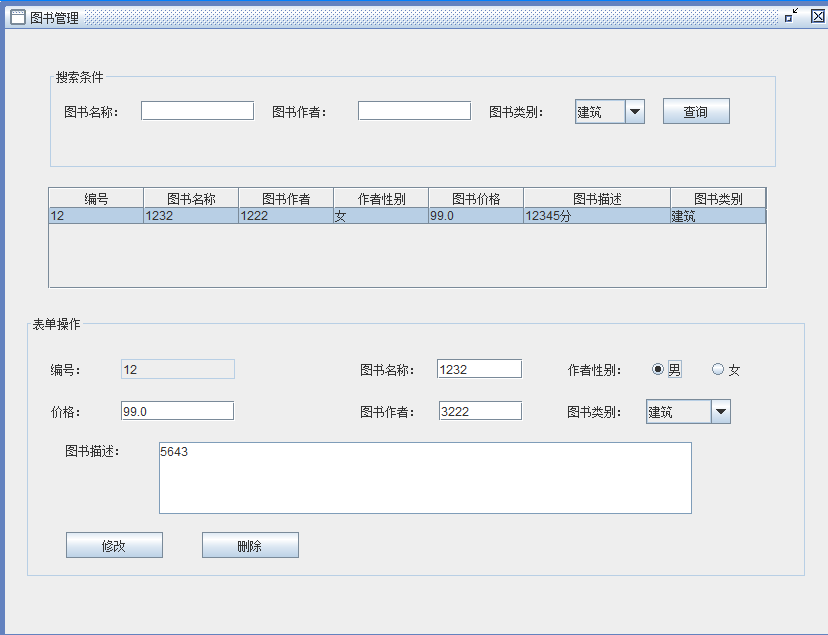
图书管理：



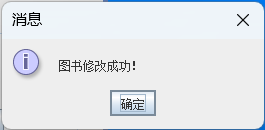
点击查询：



可在表单操作处进行图书的修改与删除：



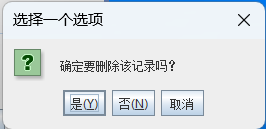
输入修改信息并点击，成功修改则显示：



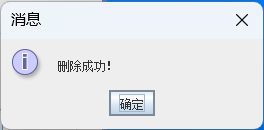
数据库信息随之改变：



选中某本图书点击删除：



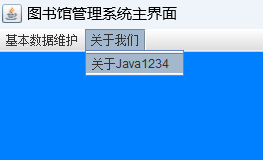
删除成功则显示：



数据库信息随之改变：



1. 关于我们：



**九、总结**

## 9.1 系统优点

1、系统设计全面、功能齐全，在有限时间内将需求的功能完全实现。系统包括读者、管理员、访客三个视角；所属读者的功能有注册与登录、查询图书、借阅图书、归还图书四个功能模块；所属的管理员的功能有注册与登录、查询图书、管理图书、查看借还书记录四大功能模块；所属访客的功能有查询图书该功能模块。

2、界面美观，软件易用性高，操作简单，能够很好的满足需求

3、数据库设计内容具体详细，条理清晰，关系明确，能够遵循数据库设计的准则来描述信息关系，可以稳定地为系统提供服务。

4、信息提示系统细致完善，对于用户可能发生的错误操作，给予错误信息提示。

5、对异常进行了处理，在用户操作不规范时给出错误信息

## 9.2 系统不足

1、数据库的设计中存在一些问题，如管理员图书类别管理中 的查询功能未能实现；读者借书后借书表的数据不会更新等

2、还缺少部分功能，比如图书的库存量、管理员对于读者账号的管理等；

## 9.3 系统改进

数据库的设计中存在一些问题，如管管理员图书类别管理中 的查询功能未能实现；读者借书后借书表的数据不会更新；还缺少部分功能，比如图书的库存量、管理员对于读者账号的管理等；后期进行功能完善可进行改进。

## 9.4 经验与收获

在此次课程设计中，体会到一个数据库系统从想法到实现的整个过程。该过程包括功能的设想、表的设计、功能的确定、表设计的确定、sqlite建表、界面的设计、代码对界面功能的实现、功能的改进、功能的测试和完善、界面美化。将理论用于实践，在实践中更深地理解理论知识。

本次使用的开发软件是Eclipse Vx.x JDK 1.6，用到了其中的功能设计和界面设计；数据库用的是mysql；用Eclipse实现数据库可视化设计；学会这些软件在数据库系统设计中的配合使用方法。

掌握了整个开发流程、开发软件以及开发细节，在未来的数据库系统设计中可以进行完善更高性能的数据库系统开发。本次开发用了两个星期，期中自行学习软件用法、设计方法、学习相关代码函数，掌握了自学方法，在未来的学习中有了接触学习更多知识的能力。

**附. 参考文献**

[1] Abraham Silberschatz,Henry F.Korth，《数据库系统概念》，机械工业出版社，2008。

[2] 刘勇，周学军，[《SQL Server 2000基础教程》](http://www.mendui.com/h_book/more206552.shtml" \t "_blank)，人民邮电出版社，2007。