

中国科学技术大学计算机学院

《数据库系统实验报告》



实验题目：图书馆信息管理

学生姓名：李宁

学生学号：PB21111715

完成时间：2024年5月22日

1 需求分析

1.1 背景

随着学校图书馆藏书量和访问量的增大，采用人工管理方式已经难以高效管控图书信息和学生借阅情况。因此，需要开发一个图书馆信息管理系统，实现图书信息、读者信息、借还信息等的电子化管理，提高图书馆的运营效率。

1.2 功能

目标用户包括管理员和学生两类：

- 图书馆管理员需要能够方便地录入、查询、修改图书信息，管理学生的借阅信息和预定信息，跟踪学生的借阅违期情况。
- 学生需要能够查询图书馆的藏书情况、预定图书、借阅图书、查询自己的借阅信息等。

具体功能可以分为以下几个方面：

- 图书信息管理
 - 录入、查询、修改图书基本信息(书名、作者、出版社、图书封面等)
 - 录入、查询图书库存状态(在馆、借出、预定等)
- 学生信息管理
 - 录入、查询、修改学生基本信息(学号、姓名、院系、联系方式等)
 - 查询学生的借阅历史、当前借阅状态
- 用户管理
 - 录入、查询、修改用户信息(姓名、联系方式等)
 - 必要时可以冻结用户，禁止预定或借阅操作
- 借还书管理
 - 学生可以查询、预定、借阅图书
 - 管理员可以录入学生的借还书记录，跟踪借书违期情况
- 统计报表
 - 生成图书借阅统计报表，包括总借阅量、热门图书等
 - 生成学生借阅数量统计报表

2 总体设计

2.1 系统模块结构

采用 C/S 架构。前端使用 `PyQt6`。通过 `QtDesigner` 设计界面，生成 `.ui` 设计文件，然后用 `PyUIC` 转换为 `.py` 代码文件。为按钮等组件绑定事件函数，与后端链接。

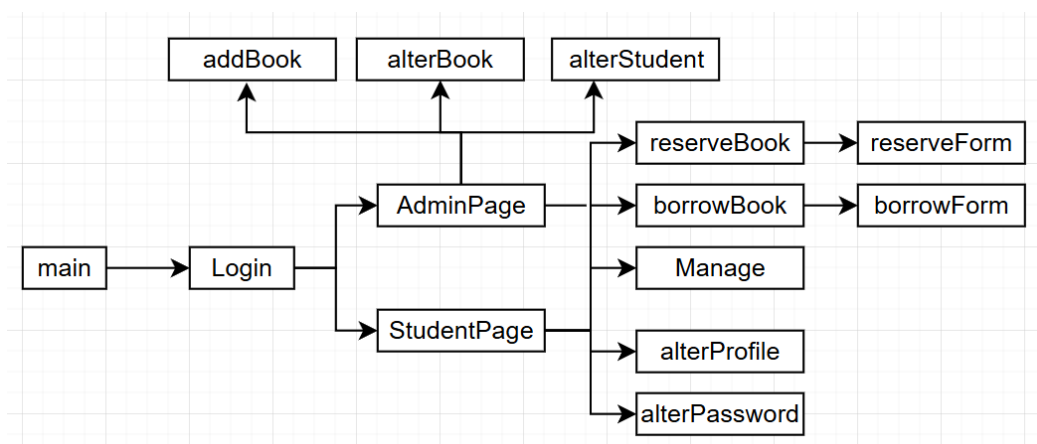
后端使用 `pymysql`。通过 `cursor.execute()` 函数执行 `MySQL` 语句，进行数据库的增删查改等操作。事件函数根据前端获得的信息对数据库执行相关操作。

系统可以细分为十个模块，分别实现十个不同的功能：

1. 登录模块：对应 `Login.py`，实现登录验证
2. 添加图书模块：对应 `addBook.py`，实现新增图书
3. 图书管理模块：对应 `alterBook.py`，实现图书信息的维护，包括修改、删除
4. 学生管理模块：对应 `alterStudent.py`，实现学生信息的维护，包括新增、修改、删除
5. 预约模块：对应 `reserveBook.py`，实现图书搜索、预约功能
6. 借阅模块：对应 `borrowBook.py`，实现图书搜索、借阅功能
7. 违期管理模块：对应 `Manage.py`，对于管理员，可以管理所有的预约和借阅记录，包括取消借阅、处理违期等功能；对于学生，只能管理自己的借阅记录，包括续借、还书等功能
8. 资料修改界面：对应 `alterProfile.py`，实现用户个人资料的修改
9. 密码更换界面：对应 `alterPassword.py`，实现用户个人登录密码的修改
10. 主界面：对应 `AdminPage.py` 和 `StudentPage.py`，分别表示管理员和学生的主界面，提供各个子功能的入口，并统计显示本月的热门书籍。

2.2 系统工作流程

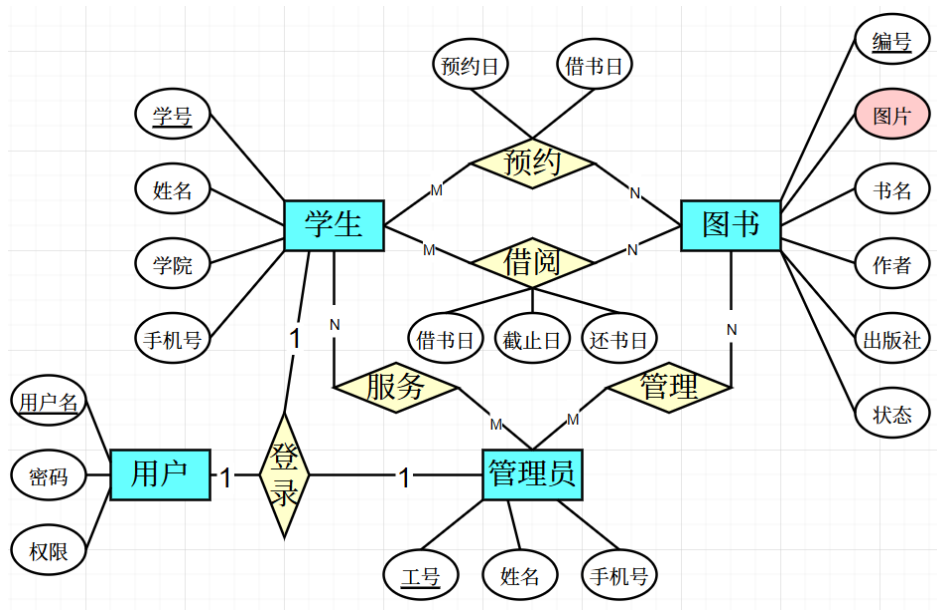
下图是各模块之间的调用逻辑：



运行程序进入登录页面，根据用户名查询用户身份，如果是管理员则进入管理首页，否则进入学生首页。首页提供各个功能的入口按钮，包含预约界面、借阅界面、记录管理界面，管理首页额外提供图书和学生信息管理的入口。首页菜单栏可以进行账号设置，包括修改资料、修改密码。如果选择某本书借阅(预约)，则会进入表单界面，输入学号和日期信息即可确定借阅(预约)。

2.3 数据库设计

2.3.1 ER图



2.3.2 模式分解

关系模式分解如下，主码用下划线标出：

- 学生信息表：Student(sid, sname, major, tel)
- 图书信息表：Book(bid, cover, bname, author, publisher, status)
- 管理员信息表：Manager(eid, ename, tel)
- 预约表：Reserve(sid, bid, reserve_date, take_date)
- 借阅表：Borrow(sid, bid, borrow_date, due_date, return_date)
- 用户表：User(username, password, uid, role, status)

预约记录过了期限或者借阅成功会删除，所以主码为 (sid, bid) 即可。而借阅记录会一直保存。

没有表中表：满足 1NF

每个非主属性都完全函数依赖于主码：满足 2NF

每一个非主属性都不传递依赖于主码：满足 3NF

创建数据库相关代码在 `create.sql` 中，以 `book` 表的创建为例，代码如下：

```

1 CREATE TABLE IF NOT EXISTS book (
2     bid INT PRIMARY KEY,
3     cover LONGBLOB,
4     bname VARCHAR(50) NOT NULL,
5     author VARCHAR(50) NOT NULL,
6     publisher VARCHAR(50) NOT NULL,
7     status INT DEFAULT 0
8 );

```

2.3.3 过程化 SQL 设计

应实验文档要求，存储过程、触发器、函数、事务都要有，相关代码在 `Procedure.sql` 中，下面分类举例讲解。事务包含在存储过程中。

2.3.3.1 存储过程

- `deleteBook(IN bookId INT)`

首先是删除书籍的存储过程，为了保持参照完整性，需要把各个表中有关该书籍的元组都删除。代码如下：

```

1 CREATE PROCEDURE IF NOT EXISTS deleteBook(IN bookId INT)
2 BEGIN
3     START TRANSACTION;
4     SET FOREIGN_KEY_CHECKS = 0;
5     DELETE FROM book WHERE bid = bookId;
6     DELETE FROM reserve WHERE bid = bookId;
7     DELETE FROM borrow WHERE bid = bookId;
8     SET FOREIGN_KEY_CHECKS = 1;
9     COMMIT;
10 END;

```

先暂时关闭外键检测，然后正常删除，最后重新开启，提交事务。

- `updateStudentId(IN oldId INT, IN newId INT)`

更新学生学号的存储过程，除了更新借阅和预约表外，还需要更新用户表。代码类似，此处不在赘述

- `deleteStudent(IN studentId INT)`

删除学生的存储过程，类似更新过程

- `reserveBook(IN studentId INT, IN bookId INT, IN takeDate DATE)`

预约图书的存储过程，根据提供的信息在预约表中添加相应记录，预约日期设置为当前日期。然后需要更新对应图书的状态为已预约。另外，需要先验证用户的账号状态，如果被冻结，则禁止操作。代码如下：

```

1 CREATE PROCEDURE IF NOT EXISTS reserveBook(IN studentId INT, IN bookId INT, IN
  takeDate DATE)
2 label: BEGIN
3     START TRANSACTION;
4     IF (SELECT status FROM student, user WHERE student.sid = studentId AND
  student.sid = user.uid AND user.role = 0) = 1 THEN
5         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '您的账号已被冻结，无法操
  作!';
6         ROLLBACK;
7         LEAVE label;
8     END IF;
9     INSERT INTO reserve(sid, bid, reserve_date, take_date) VALUES (studentId,
  bookId, CURDATE(), takeDate);
10    UPDATE book SET status = 1 WHERE bid = bookId;
11    COMMIT;
12 END;

```

- borrowBook(IN studentId INT, IN bookId INT, IN dueDate DATE)

借阅图书的存储过程，过程和上面类似

- killUser(IN studentId INT)

冻结用户的存储过程，首先删除该用户相关的预约记录，然后设置用户状态为 1，表示冻结。代码如下：

```

1 CREATE PROCEDURE IF NOT EXISTS killUser(IN studentId INT)
2 label: BEGIN
3     START TRANSACTION;
4     DELETE FROM reserve WHERE sid = studentId;
5     UPDATE user SET status = 1 WHERE uid = studentId and role = 0;
6     COMMIT;
7 END;

```

2.3.3.2 触发器

- addUser

新增学生的触发器，自动添加对应的用户，默认用户名为 0+sid，默认密码为 123456。代码如下：

```

1 CREATE TRIGGER IF NOT EXISTS addUser
2 AFTER INSERT ON student
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO user(uid, username, password, role) VALUES (NEW.sid,
  CONCAT('0', NEW.sid), '123456', 0);
6 END;

```

- deleteUser

删除学生的触发器，自动删除对应用户，类似。

- `cancelReserve`

取消预约的触发器，当预约表中某个记录被删除后，自动设置对应书籍状态为在馆。代码如下：

```

1 CREATE TRIGGER IF NOT EXISTS cancelReserve
2 AFTER DELETE ON reserve
3 FOR EACH ROW
4 BEGIN
5     UPDATE book SET status = 0 WHERE bid = OLD.bid;
6 END;
```

- `returnBook`

还书的触发器，类似

2.3.3.3 函数

- `hotBook()`

统计本月最热门书籍的函数，查询借阅记录，返回本月被借阅次数最多的书籍。代码如下：

```

1 CREATE FUNCTION IF NOT EXISTS hotBook()
2 RETURNS INT
3 READS SQL DATA
4 BEGIN
5     DECLARE hotBookId INT;
6     SELECT bid INTO hotBookId FROM borrow WHERE MONTH(borrow_date) =
    MONTH(CURDATE()) GROUP BY bid ORDER BY COUNT(*) DESC LIMIT 1;
7     RETURN hotBookId;
8 END;
```

3 核心代码解析

鉴于篇幅有限，而且很多模块逻辑和代码高度相似，所以这里只以部分模块为例做解析

3.1 仓库地址

<https://github.com/liano3/LibraryManageSys.git>

3.2 目录

```

1 E: .
2 |   addBook.py      ----- 添加书籍
3 |   AdminPage.py    ----- 管理员首页
4 |   alterBook.py     ----- 修改书籍信息
5 |   alterPassword.py ----- 修改密码
```

6		alterProfile.py	-----	修改个人信息
7		alterStudent.py	-----	修改学生信息
8		borrowBook.py	-----	借书
9		borrowForm.py	-----	借书表单
10		Login.py	-----	登录
11		main.py	-----	程序入口
12		Manage.py	-----	管理页面
13		reserveBook.py	-----	预约书籍
14		reserveForm.py	-----	预约表单
15		StudentPage.py	-----	学生首页
16		tree.txt	-----	目录树
17				
18		.idea	-----	项目控制信息，省略
19		.venv	-----	虚拟环境，省略
20		database		
21		create.sql	-----	创建数据库
22		insert_testcase.sql	-----	插入测试数据
23		Procedure.sql	-----	存储过程、触发器、函数、事务
24				
25		view	-----	ui设计文件，省略
26				
27		__pycache__	-----	缓存文件，省略

3.3 登录

对应源文件 [Login.py](#)

使用 [QtDesigner](#) 设计 ui 如下：

包括 [label](#)、[line edit](#)、[radio button](#)、[push button](#) 组件。

首先为生成的 [Ui_Form](#) 类添加 `__init__` 函数，传递数据库的游标，初始化界面。代码如下：


```

1 def __init__(self, cursor):
2     super(Ui_Form, self).__init__()
3     self.cursor = cursor
4     self.setupUi(self)

```

其中 `cursor` 是游标参数，需要在 `main.py` 中连接数据库，然后获得游标，传递给登录模块，这样可以一次连接，一直使用。代码如下：

```

1 app = QtWidgets.QApplication(sys.argv)
2 db = pymysql.connect(
3     host="localhost",
4     user="root",
5     password="10086",
6     database="library",
7     charset="utf8"
8 )
9 cursor = db.cursor()
10 ui = Login.Ui_Form(cursor)
11 ui.show()
12 sys.exit(app.exec())

```

然后为登录按钮绑定事件函数 `login()`，当用户点击登录按钮就会执行。代码如下：

```

1 self.LoginBtn.clicked.connect(self.login)

```

其中 `LoginBtn` 是对登录按钮的命名。

在 `login()` 函数中，根据用户输入的信息查询数据库，如果是管理员就进入管理首页，否则进入学生首页。代码如下：

```

1 def login(self):
2     username = self.usernameInput.text()
3     password = self.passwordInput.text()
4     role = 0 if self.isStudent.isChecked() else 1
5     if username == "" or password == "":
6         QtWidgets.QMessageBox.warning(None, "错误", "用户名或密码不能为空")
7     elif role == 1 and not self.isManager.isChecked():
8         QtWidgets.QMessageBox.warning(None, "错误", "请选择身份")
9     else:
10         sql = "select * from user where username = %s and password = %s and
role = %s"
11         cursor = self.cursor
12         cursor.execute(sql, (username, password, role))
13         result = cursor.fetchone()
14         if result:
15             if role == 1:
16                 from AdminPage import Ui_MainWindow
17                 self.adminPage = Ui_MainWindow(cursor, username)
18                 self.adminPage.show()

```

```

19         self.close()
20     else:
21         from StudentPage import Ui_MainWindow
22         self.studentPage = Ui_MainWindow(cursor, username)
23         self.studentPage.show()
24         self.close()
25     else:
26         QtWidgets.QMessageBox.warning(None, "错误", "登录失败")

```

3.4 图书信息维护

对应源文件 `alterBook.py`

ui 如下:

筛选

编 号:

书 名:

作 者:

状 态: 全部

搜索

编号	书名	作者	出版社	状态
1001	西游记	吴承恩	人民文学出版社	在馆
1002	红楼梦	曹雪芹	人民文学出版社	在馆
1003	三国演义	罗贯中	人民文学出版社	在馆
1004	水浒传	施耐庵	人民文学出版社	在馆
1005	资治通鉴	司马光	中华书局	在馆

表单

编 号: 1001


书 名: 西游记

作 者: 吴承恩

出版社: 人民文学出版社

状 态: 在馆

更换封面



删除

修改

包括 `table widget`、`graphics view` 等组件。

主要有四个接口，筛选图书、预览图书、修改图书、删除图书。

- 筛选图书

首先根据筛选条件构造相应的 sql 语句，然后通过 `execute()` 函数执行，然后通过 `fetchall()` 函数获得数据库返回的表，将结果呈现在表格中。这里对表格进行了一些设置，隐藏行号、设置不可编辑、然后均匀拉伸列宽。代码如下：

```

1 self.BookTable.verticalHeader().setVisible(False)
2 self.BookTable.setEditTriggers(QtWidgets.QAbstractItemView.EditTrigger.NoEditTriggers)
3 self.BookTable.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.ResizeMode.Stretch)

```

为搜索按钮绑定函数 `selectBook`，函数代码如下：

```

1 def selectBook(self):
2     bid = self.selectBid.text()
3     bname = self.selectBname.text()
4     author = self.selectAuthor.text()
5     status = self.selectBstatus.currentIndex() - 1
6     if status == -1:
7         status = ""
8     sql = "select bid, bname, author, publisher, status from book where
bid like '{}{}%' and bname like '{}{}%' and author like '{}{}%' and status like '{}{}%'".format(bid, bname, author, status)
9     print(sql)
10    self.cursor.execute(sql)
11    self.BookTable.setRowCount(0)
12    for row in self.cursor.fetchall():
13        rowPosition = self.BookTable.rowCount()
14        self.BookTable.insertRow(rowPosition)
15        for i in range(4):
16            self.BookTable.setItem(rowPosition, i,
QtWidgets.QTableWidgetItem(str(row[i])))
17            if row[4] == 0:
18                self.BookTable.setItem(rowPosition, 4,
QtWidgets.QTableWidgetItem("在馆"))
19            elif row[4] == 1:
20                self.BookTable.setItem(rowPosition, 4,
QtWidgets.QTableWidgetItem("已预约"))
21            elif row[4] == 2:
22                self.BookTable.setItem(rowPosition, 4,
QtWidgets.QTableWidgetItem("已借出"))

```

利用 sql 中的 `like` 语法可以实现部分搜索。

- 预览图书

为表格点击动作绑定函数 `clickTable`，当点击表格时，在下方表单处可以预览图书信息，方便修改。首先获得当前表格条目的内容，然后显示在表单上即可。封面需要额外查询数据库获得二进制流，然后转为 `QImage` 类，最后在 `graphics view` 组件上显示。代码如下：

```

1 def clickTable(self):
2     row = self.BookTable.currentRow()
3     self.alterBid.setText(self.BookTable.item(row, 0).text())
4     self.alterBname.setText(self.BookTable.item(row, 1).text())

```

```

5     self.alterAuthor.setText(self.BookTable.item(row, 2).text())
6     self.alterPublisher.setText(self.BookTable.item(row, 3).text())
7     status = self.BookTable.item(row, 4).text()
8     if status == "在馆":
9         self.alterBstatus.setCurrentIndex(0)
10    elif status == "已预约":
11        self.alterBstatus.setCurrentIndex(1)
12    elif status == "已借出":
13        self.alterBstatus.setCurrentIndex(2)
14    self.cursor.execute("select cover from book where bid = " +
self.alterBid.text())
15    cover = self.cursor.fetchone()[0]
16    img = QtGui.QImage.fromData(cover)
17    img = QtGui.QPixmap.fromImage(img).scaled(self.alterCover.width() - 10,
self.alterCover.height() - 10)
18    scene = QtWidgets.QGraphicsScene()
19    scene.addPixmap(img)
20    self.alterCover.setScene(scene)

```

- 修改图书

这里只介绍如何实现更换封面，其他属性的更改很简单。

为更换封面按钮绑定了函数 `uploadImg`，首先使用库函数 `QFileDialog.getOpenFileName` 打开一个选择文件的页面，得到文件的路径，然后用库函数 `QPixmap` 根据路径获得图片，再显示在预览表单上。最后以二进制的形式暂时保存在 `self.img` 中，当用户点击修改按钮时，会调用 `alter` 接口上传到数据库。代码如下：

```

1 def uploadImg(self):
2     imgName, imgType = QtWidgets.QFileDialog.getOpenFileName(self, "选择图片",
    "", "Image Files (*.jpg *.png)")
3     if not imgName:
4         return
5     img = QtGui.QPixmap(imgName).scaled(self.alterCover.width() - 10,
self.alterCover.height() - 10)
6     scene = QtWidgets.QGraphicsScene()
7     scene.addPixmap(img)
8     self.alterCover.setScene(scene)
9     with open(imgName, "rb") as f:
10        self.img = f.read()

```

3.5 图书预约

对应源文件 `reserveBook.py`

ui 如下：

筛选


编号:
书名:
作者:

搜索

编号	书名	作者	出版社
1001	西游记	吴承恩	人民文学出版社
1002	红楼梦	曹雪芹	人民文学出版社
1003	三国演义	罗贯中	人民文学出版社
1004	水浒传	施耐庵	人民文学出版社
1005	资治通鉴	司马光	中华书局

预览

编号:
书名:
作者:
出版社:



预约

此页面和图书信息维护页面类似，区别在于下面的预览表单不可编辑。设置不可编辑，同时改变背景色，代码如下：

```

1 self.viewBid.setReadOnly(True)
2 self.viewBid.setStyleSheet("background-color: #eee;")

```

搜索接口和上面一样，预约接口指向预约表单界面。

学号:

书号:

取书日期:

确认预约

预约表单的书号为当前所选书号(从上个页面传参)，学号和取书日期需要自己填写。

设置取书日期默认为当天，且不可在当天之前。代码如下：

```

1 self.takeDate.setMinimumDate(QCoreApplication.currentDate())
2 self.takeDate.setDate(QCoreApplication.currentDate())

```

确认预约的按钮绑定 `reserveBook` 函数，函数内部调用存储过程 `reserveBook` 完成图书预约。代码如下：

```

1 def reserveBook(self):
2     sid = self.reserveSid.text()

```

```

3     take_date = self.takeDate.text()
4     bid = self.reserveBid.text()
5     if sid == "" or take_date == "":
6         QtWidgets.QMessageBox.warning(self, "警告", "请填写完整信息")
7         return
8     try:
9         self.cursor.callproc("reserveBook", (sid, bid, take_date))
10        QtWidgets.QMessageBox.information(self, "提示", "预约成功")
11        self.close()
12        self.parent.searchBook()
13    except Exception as e:
14        QtWidgets.QMessageBox.warning(self, "警告", e.args[1])
15        return

```

3.6 借阅管理

对应源文件 [Manage.py](#)

此界面对管理员和学生均开放，但呈现的内容不同。对于管理员，可以看到预约表和借阅表的所有条目，而且可以一键处理违期记录。对于学生，只能看到自己的预约和借阅记录。

ui 如下：(管理员登录)

预约管理

学号:

书号:

状态: 全部

搜索

学号	书号	预约日期	取书日期
1	1001	2024-05-22	2024-05-24
2	1002	2024-05-22	2024-05-25

取消预约

借阅管理

学号:

书号:

状态: 全部

搜索

学号	书号	借书日期	截止日期	还书日期
3	1003	2024-05-22	2024-05-29	None
4	1004	2024-05-22	2024-05-24	None

续借

还书

取消借阅

批处理违期

搜索接口依然和上面一样。这里主要介绍取消预约、续借、还书、取消借阅和批处理违期的接口。

- 取消预约

在预约表中删除所选中的条目即可，由于创建了触发器，删除后触发器会自动把对应书籍的状态改在馆。最后刷新预约管理界面内容，代码如下：

```

1 def cancelReserve(self):
2     row = self.RTable.currentRow()
3     sid = self.RTable.item(row, 0).text()
4     bid = self.RTable.item(row, 1).text()
5     sql = "delete from reserve where sid = '{}' and bid = '{}'".format(sid,
bid)
6     try:
7         self.cursor.execute(sql)
8         self.cursor.connection.commit()
9     except Exception as e:
10        print(e)
11        self.Rselect()

```

- 取消借阅、还书、续借

类似，取消借阅就直接删除借阅表对应条目，还书就更改 `return_date` 属性。也有触发器更改对应书籍的状态。续借默认 `due_date` 加七天。

- 处理违期

对于预约表，因为借阅成功机会删除，所以如果 `take_date` 比当前日期小，说明被鸽了；对于借阅表，如果 `due_date` 比当前日期小，且 `return_date` 为空，即还未还书，则说明违期。对违期的用户调用存储过程 `killUser` 进行冻结。代码如下：

```

1 def Kill(self):
2     sql = "select sid from reserve where take_date < CURDATE() \
3         union select sid from borrow where due_date < CURDATE() and
return_date is NULL"
4     self.cursor.execute(sql)
5     result = self.cursor.fetchall()
6     for i in range(len(result)):
7         sid = result[i][0]
8         sql = "call killUser('{}')".format(sid)
9         try:
10            self.cursor.execute(sql)
11            self.cursor.connection.commit()
12        except Exception as e:
13            print(e)
14        self.Bselect()
15        self.Rselect()

```

3.7 账号设置



首先在首页绑定菜单栏的触发函数，代码如下：

```

1 self.AccountManage.triggered.connect(self.manageAccount)
2 def manageAccount(self, m):
3     if m.text() == "修改资料":
4         from alterProfile import Ui_Form
5         self.alterProfile = Ui_Form(self.cursor, self.username)
6         self.alterProfile.show()
7     elif m.text() == "修改密码":
8         from alterPassword import Ui_Form
9         self.alterPassword = Ui_Form(self.cursor, self.username)
10        self.alterPassword.show()
11    elif m.text() == "退出登录":
12        # 回到登录页面
13        from Login import Ui_Form
14        self.login = Ui_Form(self.cursor)
15        self.login.show()
16        self.close()

```

修改资料或密码的逻辑很简单，先生成表单页面，然后根据表单数据更改数据库即可。此处不再赘述。

4 实验与测试

4.1 依赖

用命令 `pip freeze > requirements.txt` 生成的依赖文件如下：

```
1 cffi==1.16.0
2 click==8.1.7
3 colorama==0.4.6
4 cryptography==42.0.7
5 numpy==1.26.4
6 pycparser==2.22
7 PyMySQL==1.1.0
8 PyQt6==6.4.2
9 pyqt6-plugins==6.4.2.2.3
10 PyQt6-Qt6==6.4.3
11 PyQt6-sip==13.6.0
12 pyqt6-tools==6.4.2.3.3
13 pyqtgraph==0.13.7
14 python-dotenv==1.0.1
15 qt6-applications==6.4.3.2.3
16 qt6-tools==6.4.3.1.3
```

运行环境为 `PyCharm 2024.1.1` 专业版的虚拟环境。

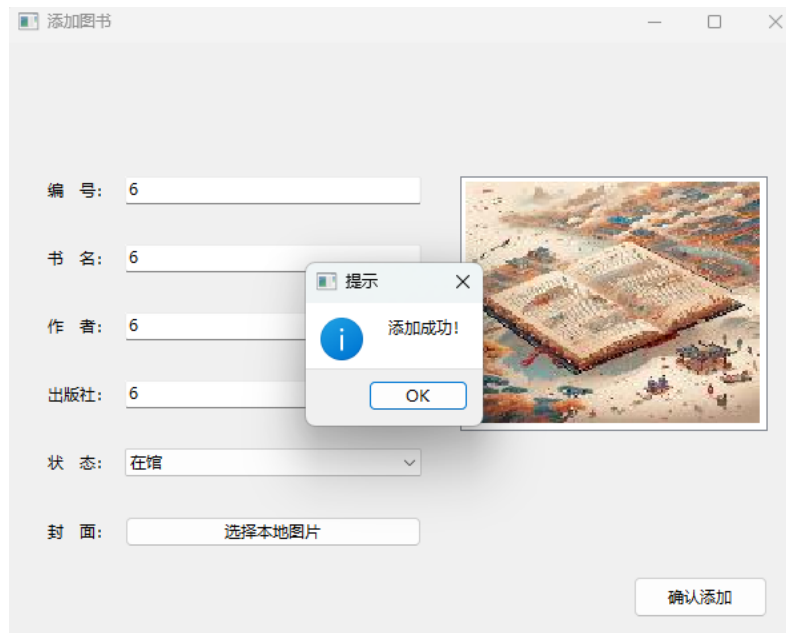
4.2 部署

使用命令 `python main.py` 即可启动程序。

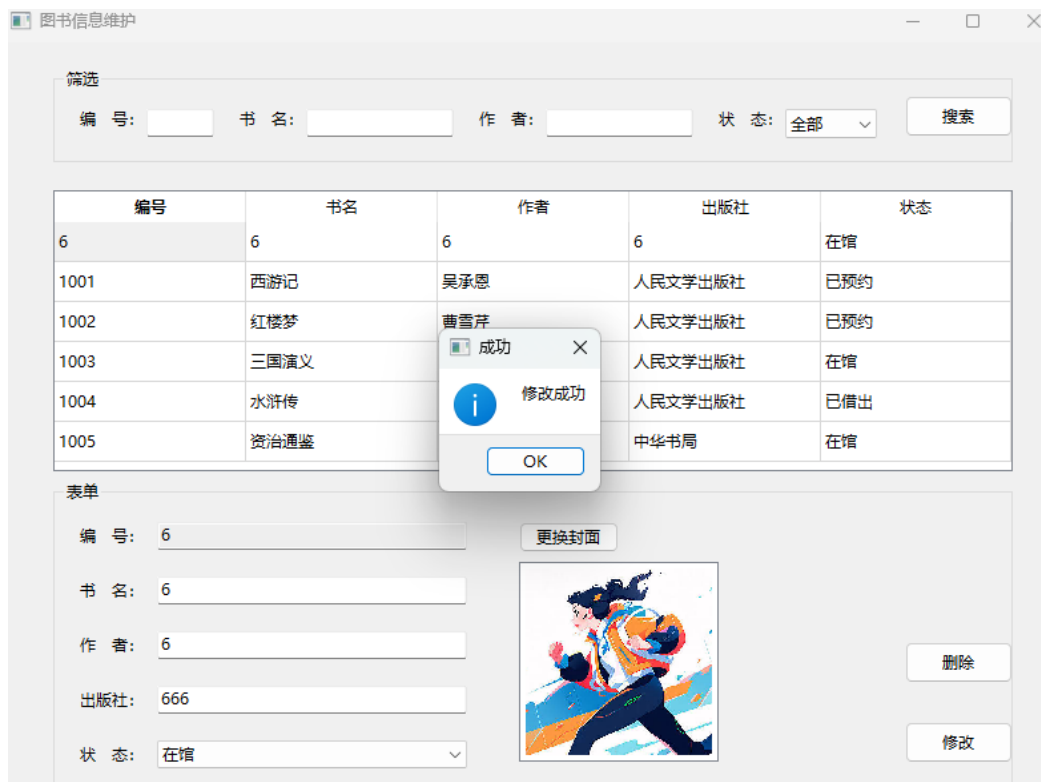
4.3 实验结果

这里展示添加图书->修改图书信息->添加学生->学生登录->学生修改密码->学生预约->学生取消预约->学生借阅->学生还书的全过程。

- 添加图书



- 修改图书信息



- 添加学生

学生信息维护

筛选

学 号: 姓 名: 专 业:

学号	姓名	专业	电话号码	状态
1	张三	计算机科学与技术	13812345678	正常
2	李四	软件工程	13898765432	正常

提示

添加成功

OK

表单

学 号:

姓 名:

专 业:

电 话:

状 态:

- 学生登录

登录

图书馆信息管理系统

用户名:

密 码:

身 份: ☒ 学生 ☐ 管理员

成功

登录成功

OK

- 学生修改密码

更换密码

原密码:

新密码:

确认密码:

提示

修改密码成功!

OK

确认修改

- 学生预约

The '预约图书' (Book Reservation) interface includes a search bar with '编号' (ID) and '书名' (Book Name) fields. Below it is a table of books:

编号	书名
6	6
1003	三国演义
1005	资治通鉴

Below the table is a '预览' (Preview) section with fields for '编号' (6), '书名' (6), '作者' (6), and '出版社' (666). To the right is a '预约表单' (Reservation Form) with fields for '学号' (0111), '书号' (6), and '取书日期' (2024/5/23). A '确认预约' (Confirm Reservation) button is at the bottom right. A '提示' (Prompt) dialog box in the center shows '预约成功' (Reservation Successful) with an 'OK' button.

- 学生取消预约

The '预约管理' (Reservation Management) interface includes a search bar with '学号' (111), '书号' (6), and '状态' (全部) dropdown. Below it is a table of reservations:

学号	书号	预约日期	取书日期
111	6	2024-05-22	2024-05-23

A '取消预约' (Cancel Reservation) button is at the bottom right.

- 学生借阅

The '借阅表单' (Borrowing Form) interface includes fields for '学号' (0111), '书号' (6), and '还书日期' (2024/5/23). A '确认借阅' (Confirm Borrowing) button is at the bottom right. A '提示' (Prompt) dialog box in the center shows '借阅成功' (Borrowing Successful) with an 'OK' button.

- 学生还书

借阅管理

学 号: 111 书 号: 状 态: 全部 搜索

学号	书号	借书日期	截止日期	还书日期
111	6	2024-05-22	2024-05-25	None

续借 还书 取消借阅

5 功能完善

发现冻结用户时删除预约记录的动作设计不合理，本意是想尽快释放图书资源供其他用户借阅，但这会导致用户查不到自己为什么被冻结。所以这里将预约表和借阅表一样设计。改为：

```
Reserve(sid, bid, reserve_date, take_date, pass_date)
```

已借阅或者被冻结不用删除预约记录，而是设置 `pass_date`，表示实际取书的日期或者被冻结的日期，当其不为 `null` 时表示预约已经失效。这样既能及时释放资源，也能保存预约记录。

当然，代码各接口也要略微修改，这里就不赘述了。

结果如下：

学生 1 被冻结，能看到自己的违期记录。

预约管理

学 号: 1 书 号: 状 态: 违期 搜索

学号	书号	预约日期	取书日期	过期日期
1	1001	2024-05-22	2024-05-23	2024-05-24

书籍 1001 状态更新为在馆。

筛选

编 号:

书 名:

作 者:

状 态: 全部 ▼

搜索

编号	书名	作者	出版社	状态
6	6	6	666	在馆
1001	西游记	吴承恩	人民文学出版社	在馆
1002	红楼梦	曹雪芹	人民文学出版社	已预约
1003	三国演义	罗贯中	人民文学出版社	在馆
1004	水浒传	施耐庵	人民文学出版社	已借出
1005	资治通鉴	司马光	中华书局	在馆

表单

编 号:


书 名:

作 者:

出版社:

状 态: 在馆 ▼

更换封面



删除

修改

6 参考

- 绘图工具: draw.io
- PyQt6 官网: [PyQt6 · PyPI](https://www.pyqt6.org/)
- 环境配置: [CSDN博客](#)
- pip 换源: [CSDN博客](#)
- git 设置代理: [腾讯云 \(tencent.com\)](https://cloud.tencent.com/developer/article/1234567)
- B 站教程: [哔哩哔哩_bilibili](https://www.bilibili.com/video/BV1234567890)