《数据库原理》实验报告 4

实验内容：图书馆借还书系统数据库设计

学号：21009200991

姓名：盖乐

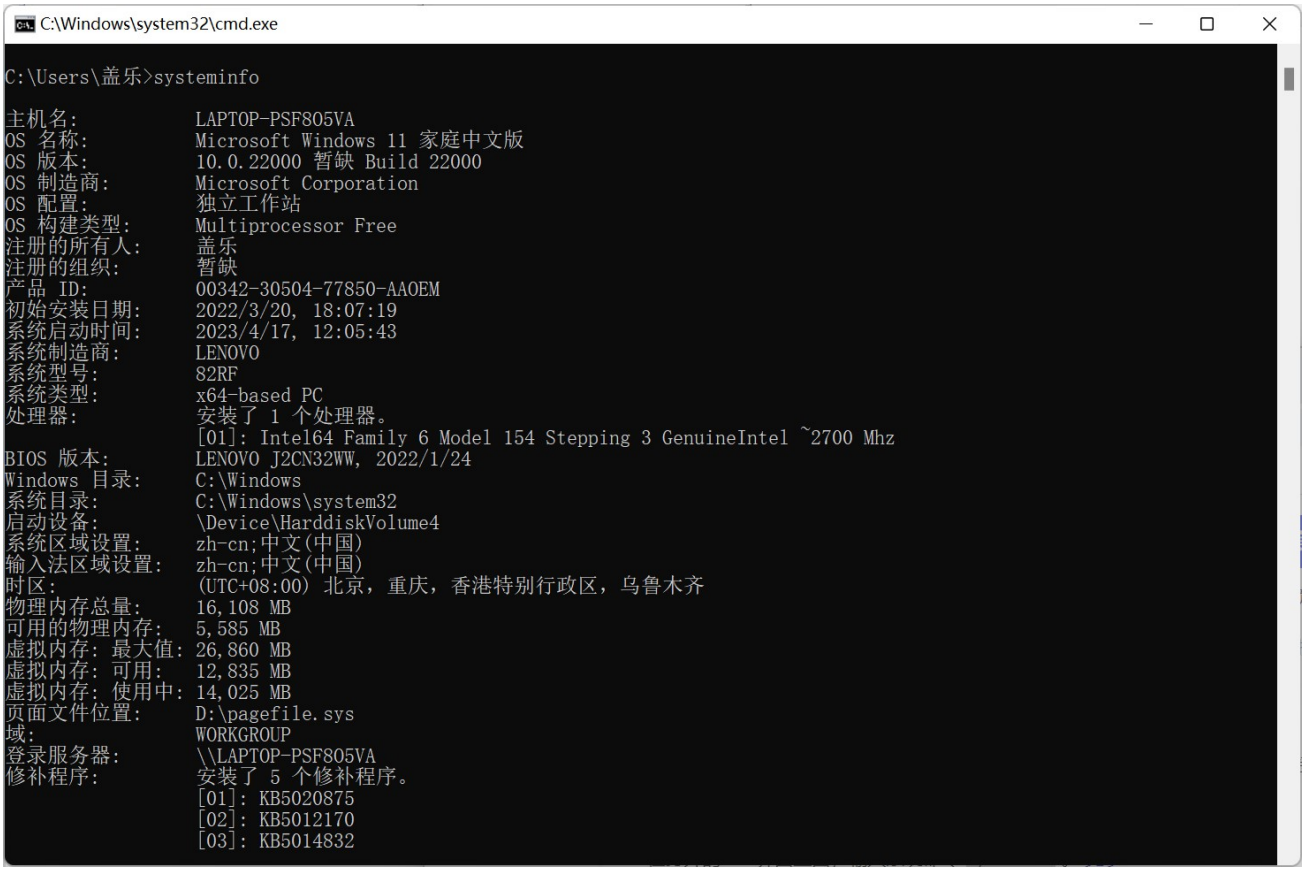
班级：2118021

1 实验内容简介

图书馆借还书系统：图书信息表支持购入同一个 ISBN 的书籍多本，支持按 ISBN、图书名、出版社、作者、图书分类、出版年份进行书籍登记，可查询和删除，使用 python 的前端对数据库进行对应操作。

2 实验环境描述

2.1 硬件配置



2.2 软件配置

2.2.1 操作系统

```
主机名:           LAPTOP-PSF8O5VA
OS 名称:          Microsoft Windows 11 家庭中文版
OS 版本:          10.0.22000 暂缺 Build 22000
```

2.2.2 数据库软件和版本

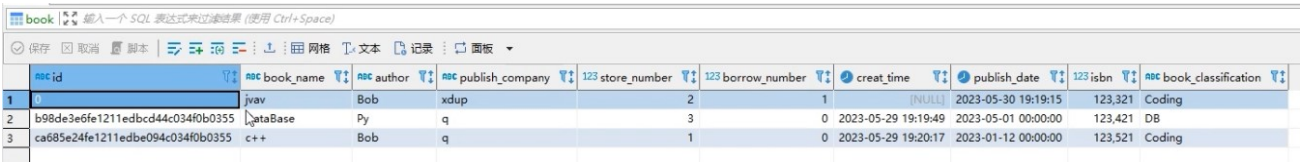

```
PS E:\kingbase\KESRealPro\V008R006C007B0012\Server\bin> .\kingbase.exe -V
KINGBASE (KingbaseES) V008R006C007B0012
```

3 实验步骤和抓图

3.1 数据库设计

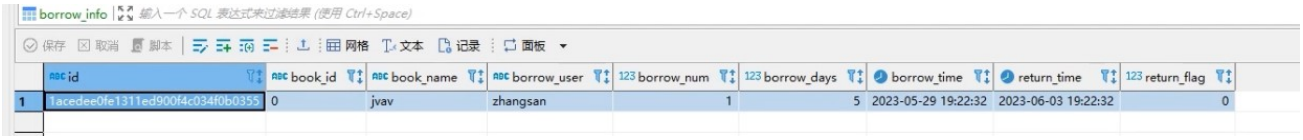
设计一个图书馆借还书系统涉及到的表有图书表、借书记录表以及用户表。由此我分别设计了三个不同的表进行功能实现。

在图书表 book 中定义了图书的 id、图书名、作者、出版社、数量、借出数量、登记时间、出版年份、ISBN、图书分类等属性。



id	book_name	author	publish_company	store_number	borrow_number	creat_time	publish_date	isbn	book_classification
1	jvav	Bob	xdup	2	1	[NULL]	2023-05-30 19:19:15	123,321	Coding
2	b98de3e6fe1211edbcd44c034f0b0355	ataBase	Py	q	3	0	2023-05-29 19:19:49	2023-05-01 00:00:00	123,421	DB
3	ca685e24fe1211edbe094c034f0b0355	c++	Bob	q	1	0	2023-05-29 19:20:17	2023-01-12 00:00:00	123,521	Coding

在借书记录表 borrow_info 中定义了 id、借出图书 id、借出图书名、借阅人、借阅数量、借阅日期、借出时间、应还时间、是否归还标记等基本信息。



id	book_id	book_name	borrow_user	borrow_num	borrow_days	borrow_time	return_time	return_flag
1acedee0fe1311ed900f4c034f0b0355	0	jvav	zhangsan	1	5	2023-05-29 19:22:32	2023-06-03 19:22:32	0

在用户表 user 中定义了 id、用户名、密码、分组、创建时间、删除时间、最近登陆时间等基本操作。其中分组分为了普通用户 1 和管理员 0，普通用户只能进行查找和借还书等基本操作，而管理员可以进行图书编辑、添加书籍和提醒还书的操作。

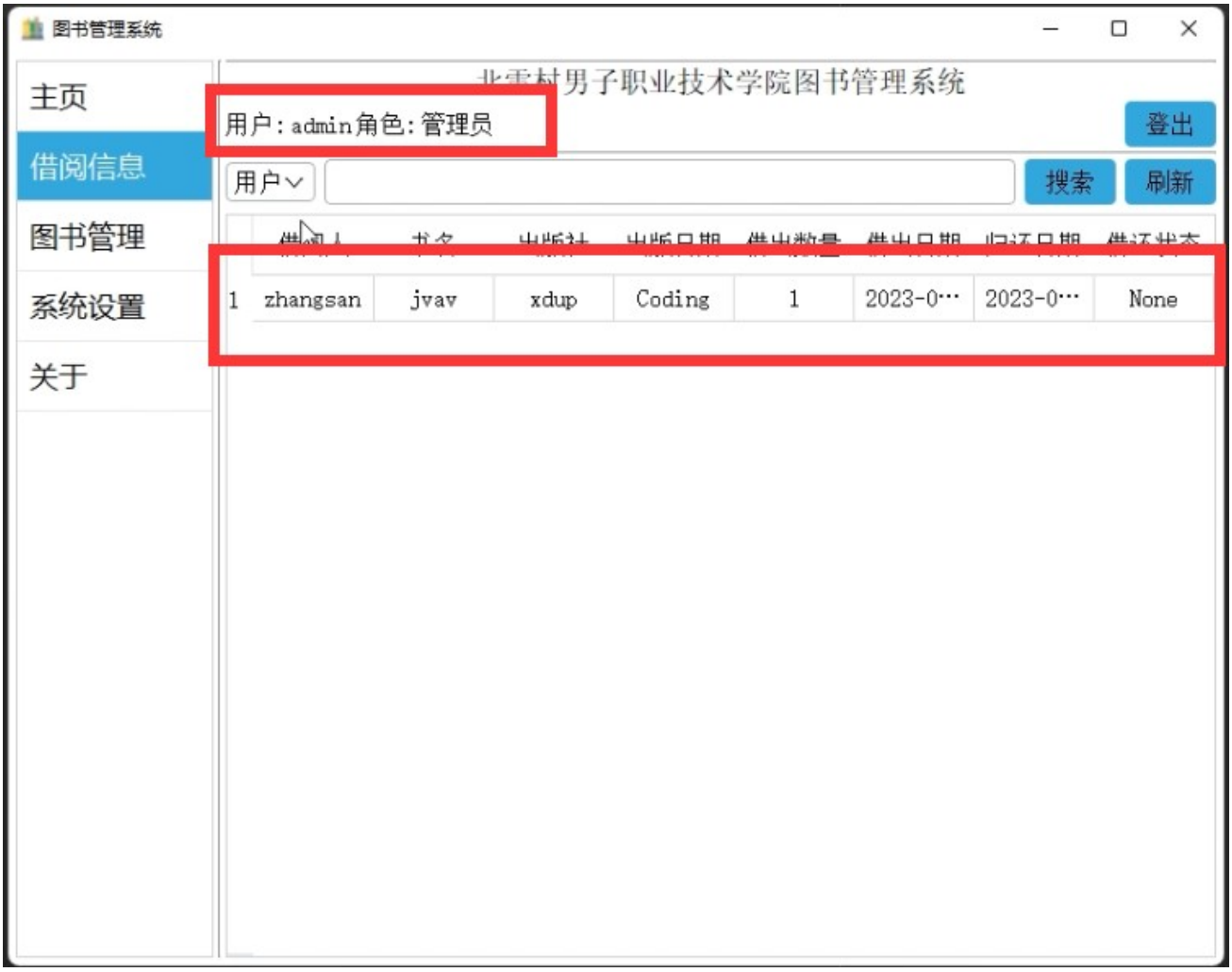
id	username	password	role	create_time	delete_flag	current_login_time
1	12644064935811ea9063d8c497639e37	admin	21232f297a57a5a743894a0e4a801fc3	0	2020-05-11 15:23:12	0	2020-05-11 15:24:23
2	99477a9e935811ea8171d8c497639e37	zhangsan	e10adc3949ba59abbe56e057f20f883e	1	2020-05-11 15:23:12	0	2020-05-11 15:24:23

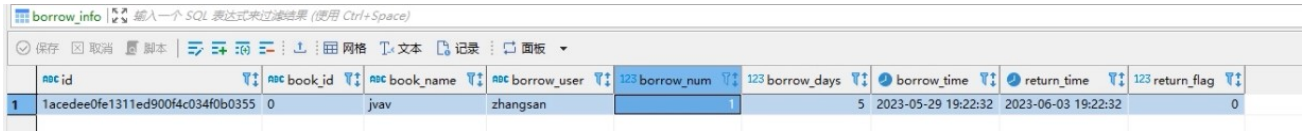
在上述的表中，所定义的 id 和密码均进行了 MD5 加密操作，以保证数据安全。

3.2 结合代码解释 ui 界面按钮的实现


```
def login(self, username, password):
    """
    登陆子线程用户验证
    :param username: 需要验证的用户名
    :param password: 匹配的密码
    :return: 验证出错返回,并发射相应TAG的信号
    """
    db = DBHelp()
    count, res = db.query_super(table_name='user', column_name='username', condition=username)
    if count == 0:
        self.login_done_signal.emit(1)
        return
    if get_md5(password) != res[0][2]:
        self.login_done_signal.emit(11)
        return
    self.role = res[0][3]
    self.login_done_signal.emit(111)
```

在登陆界面可以进行登录和新用户的注册；根据不同的输入情况进行反馈，例如用户不存在，密码错误以及成功登录。






```
def query_all(self, table_name):
    sql = 'select * from {}'.format(table_name)
    count = self._cur.execute(sql)
    res = self._cur.fetchall()
    return count, res
```

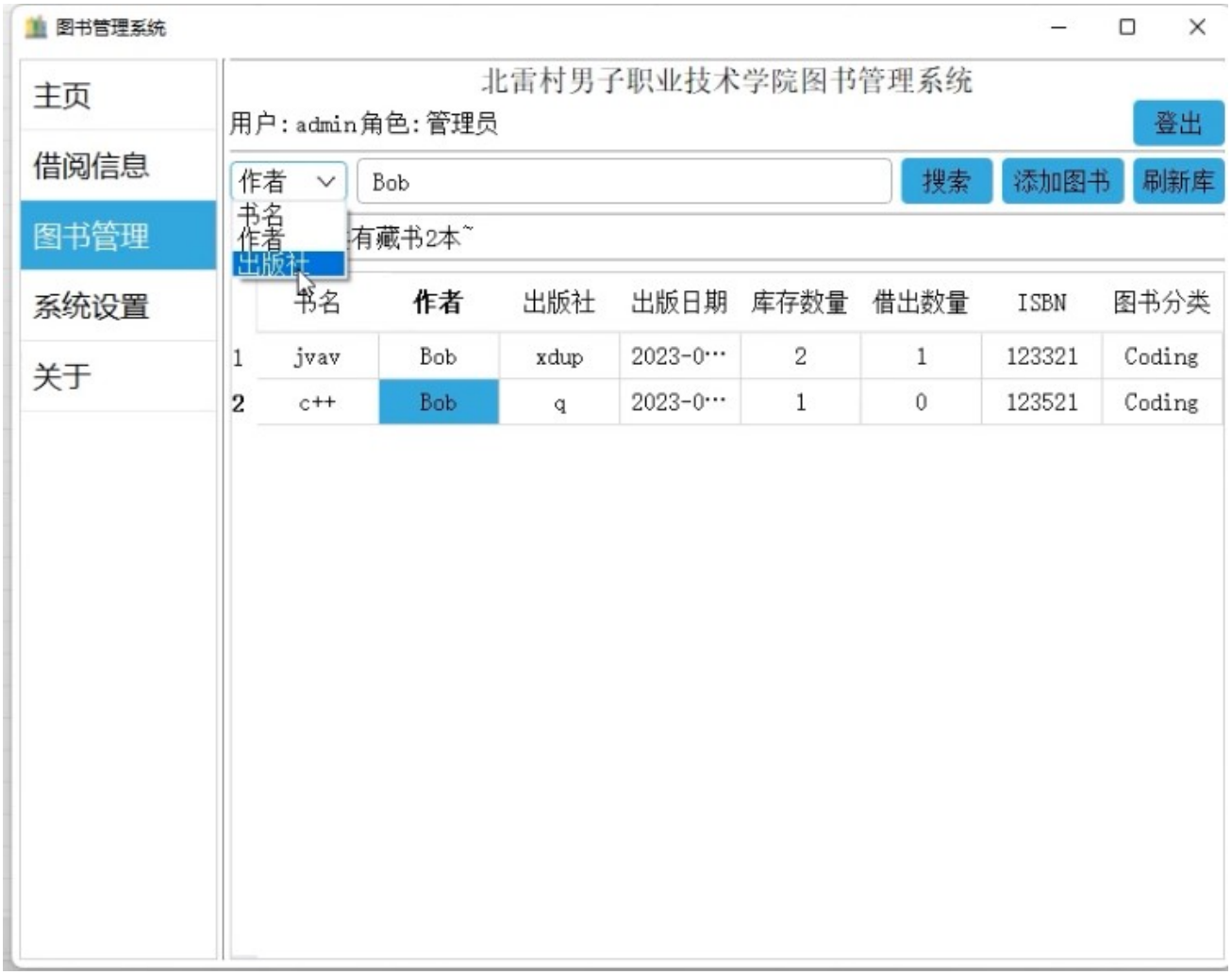
在管理员页面中可以看到所有的借阅记录。


```
def add_book(self, data):
    sql = "insert into book (id, book_name,author,publish_company,store_number,ISBN_Code,Book_Classification,borrow_number,create_time," \
          "publish_date) values (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
    self._cur.execute(sql, data)
```

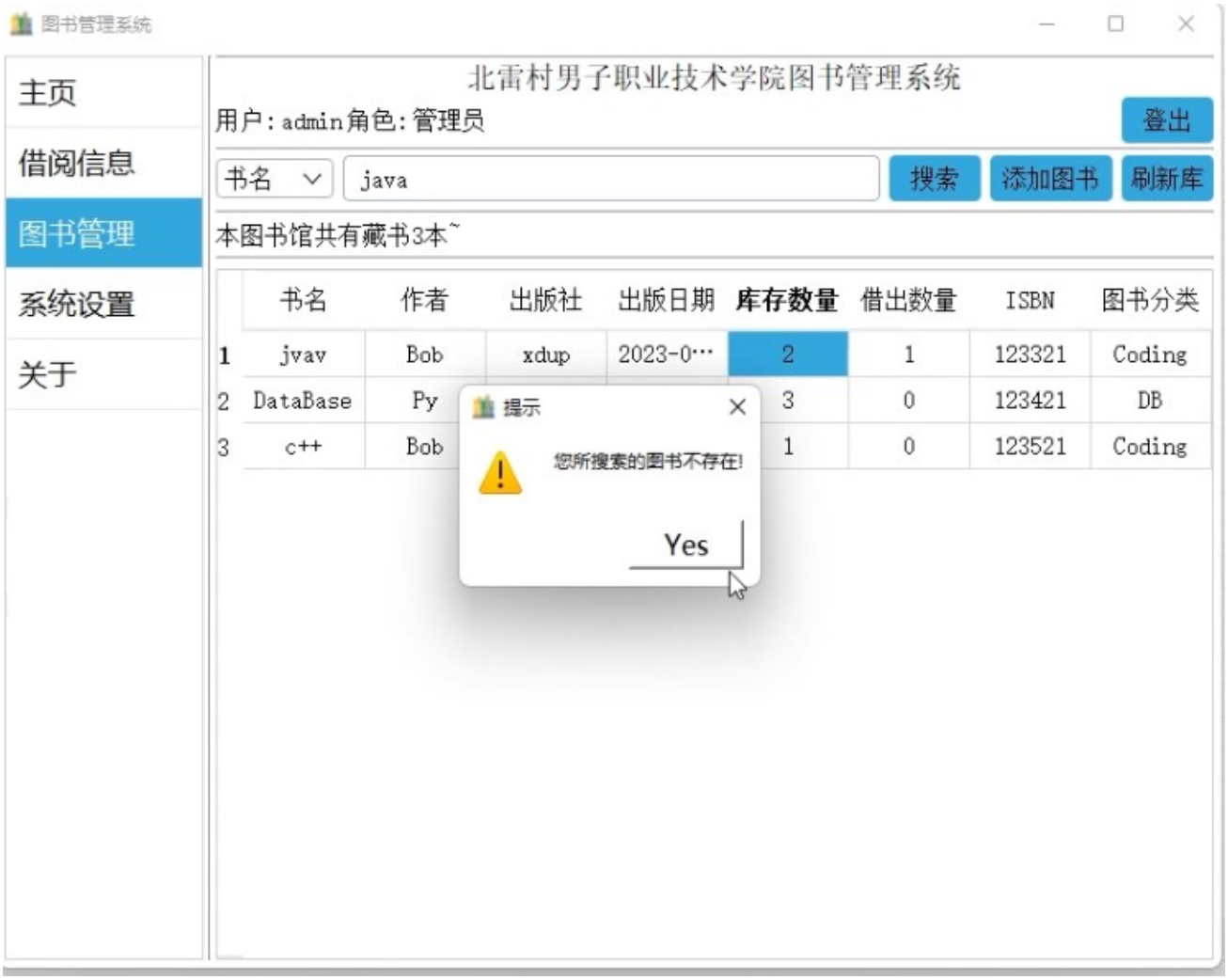
向图书表添加一条新记录，用于添加新图书操作，使用到了 insert 语句实现。


```
def delete(self, table_name, column_name, condition):
    sql = "delete from {} where {}='{}'".format(table_name, column_name, condition)
    self._cur.execute(sql)
```

管理员可以进行图书的编辑和删除操作，其中删除是使用的 delete 语句实现。

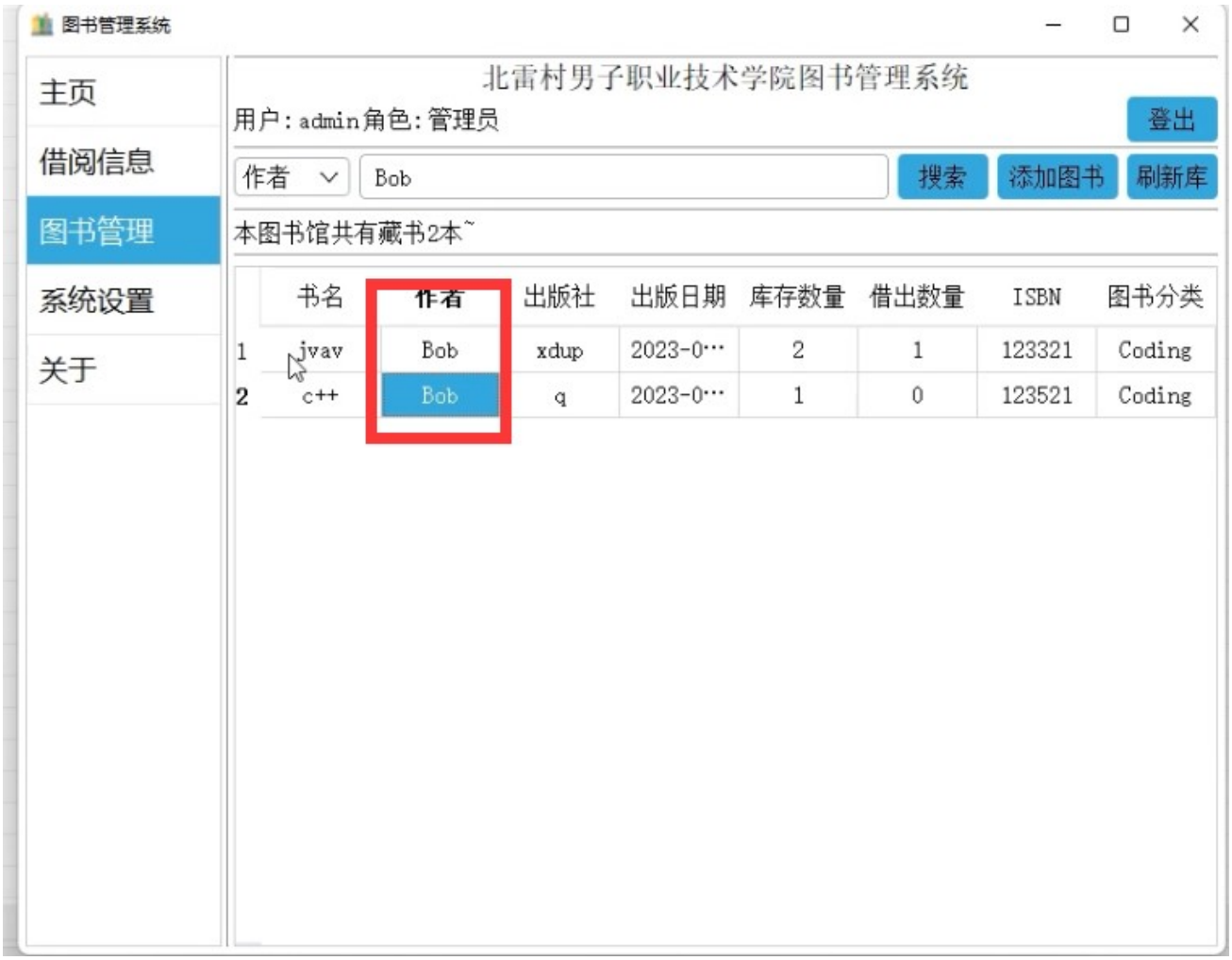


在图书查询功能页面可以通过书名、作者以及出版社进行查找。

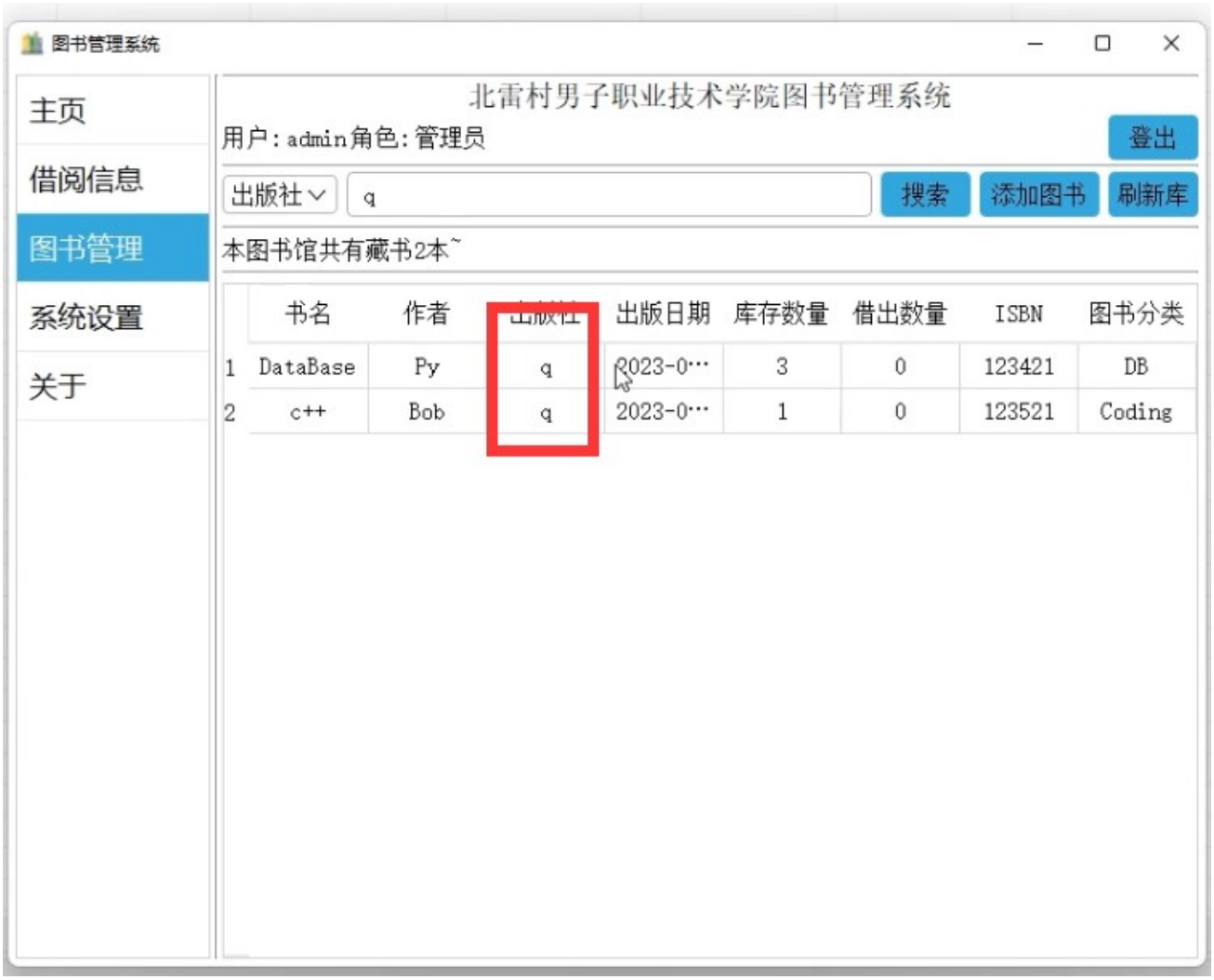


若数据库中并无相关记录，则给出相应提示。

根据书名进行图书查找。


图书管理系统
主页
借阅信息
图书管理
系统设置
关于
北雷村男子职业技术学院图书管理系统
用户：admin角色：管理员
登出
作者
Bob
搜索
添加图书
刷新库
本图书馆共有藏书2本~
书名
作者
出版社
出版日期
库存数量
借出数量
ISBN
图书分类
1
jvav
Bob
xdup
2023-0…
2
1
123321
Coding
2
c++
Bob
q
2023-0…
1
0
123521
Coding

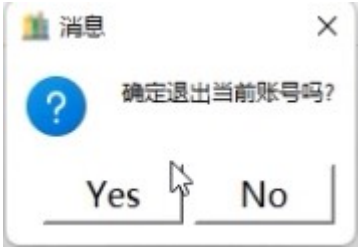
根据作者进行图书查找。



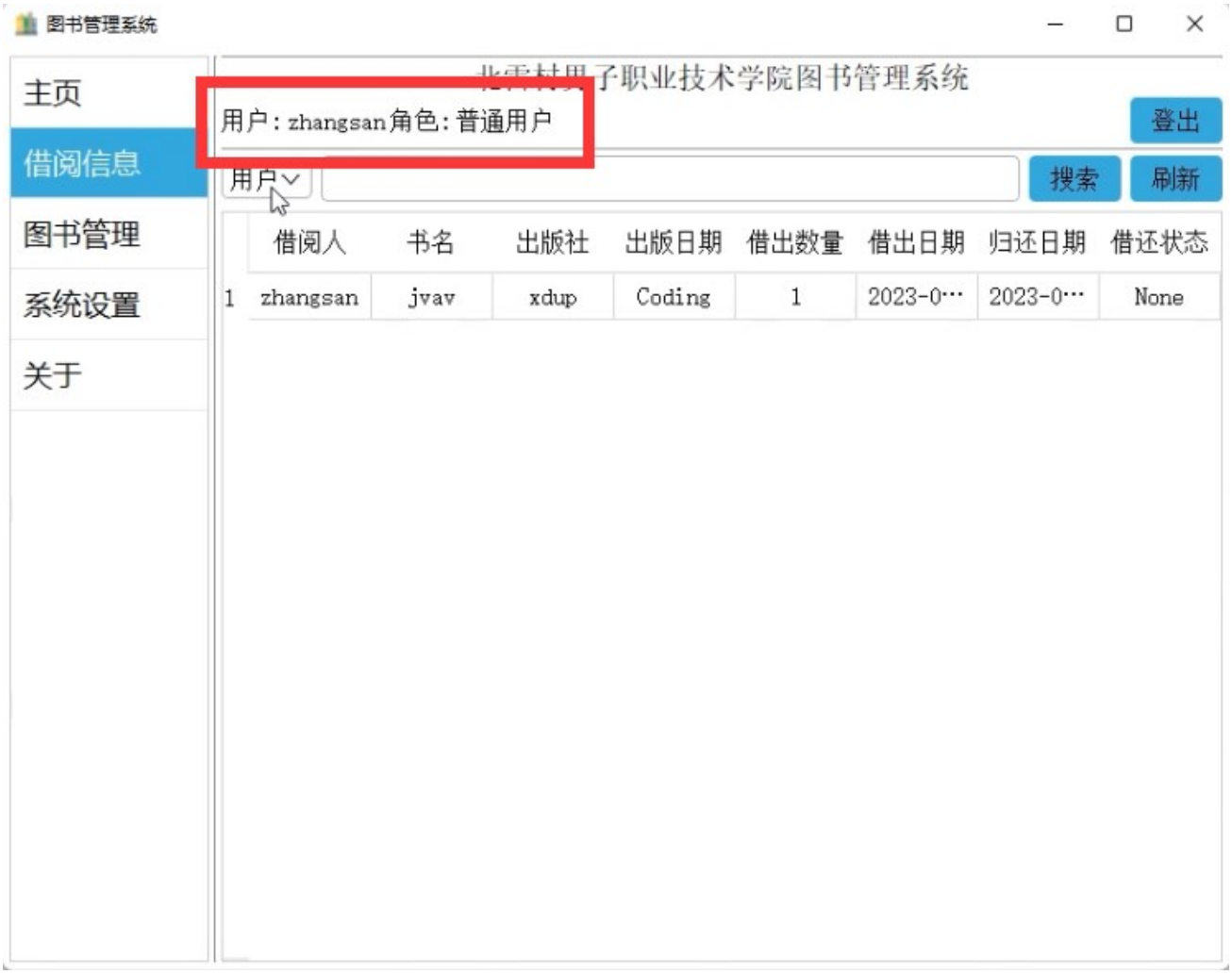
根据出版社进行图书查找。

```
def query_super(self, table_name, column_name, condition):
    sql = "select * from {} where {}='{}'".format(table_name, column_name, condition)
    count = self._cur.execute(sql)
    res = self._cur.fetchall()
    return count, res
```

如存在相应的记录，则将相关记录进行展示。使用 select * from where 语句实现。



执行退出操作也会进行给出相应的提示。



可以看到更换用户后会显示不同的组别。


```
def update_borrow_return(self, book_id):
    sql = "update book set store_number=store_number+1 where id='{}'".format(book_id)
    self._cur.execute(sql)
    sql = "update book set borrow_number=borrow_number-1 where id='{}'".format(book_id)
    self._cur.execute(sql)
```

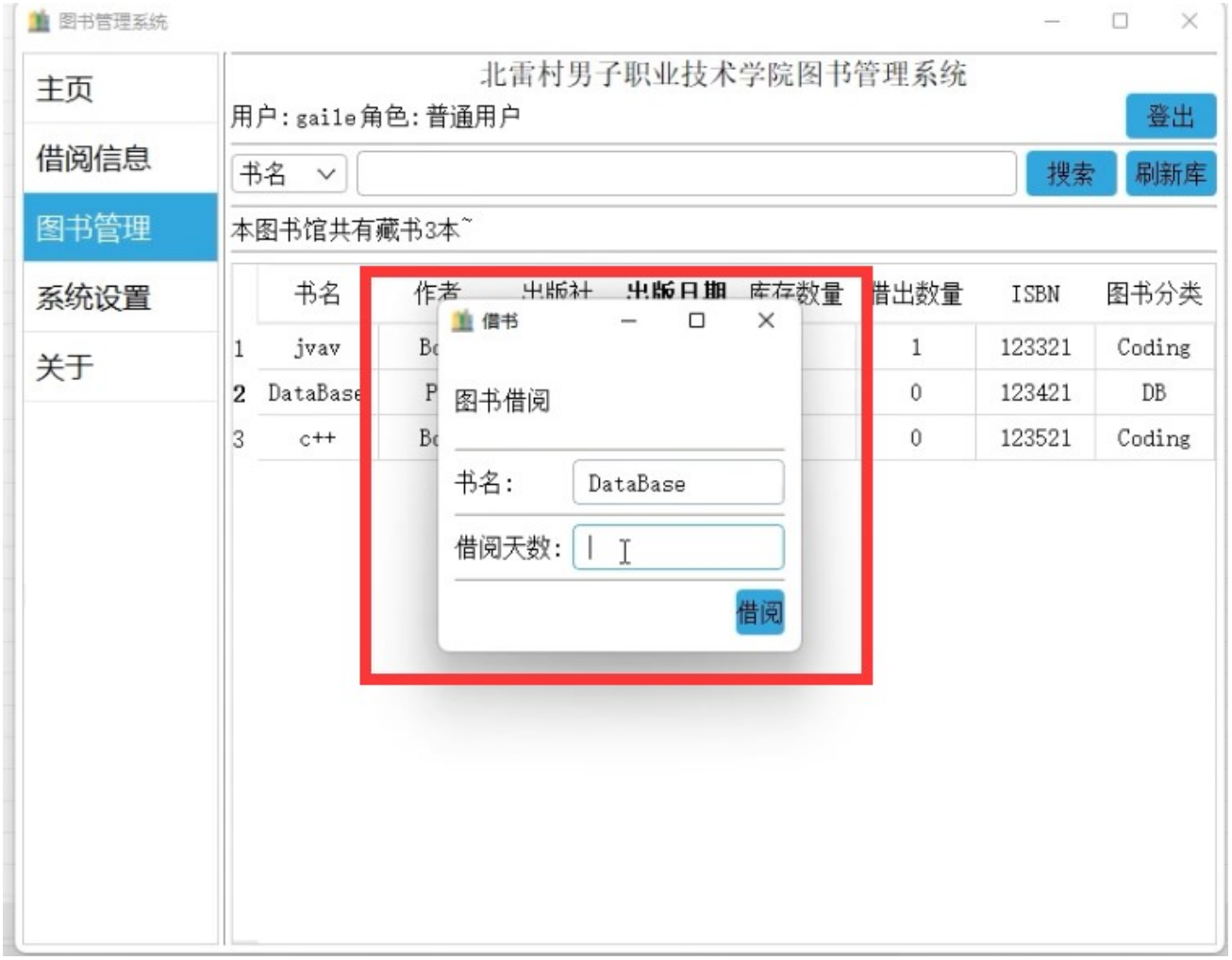
借阅人进行续借或者还书操作，通过上述 SQL 语句实现。

```
def update_borrow_statue(self, book_id):
    sql = "update borrow_info set return_flag=1 where id='{}'".format(book_id)
    self._cur.execute(sql)
```

之后将指定 ID 的借阅标记设置为已归还状态。


```
def add_user(self, data):
    sql = "insert into user (id, username, password, role, create_time, delete_flag, current_login_time) " \
          "values (%s, %s, %s, %s, %s, %s, %s)"
    self._cur.execute(sql, data)
```

向用户表添加一条新记录，创建一个新用户。




```
def update_renew(self, data):
    sql = "update borrow_info set borrow_days=borrow_days+{}, return_time='{}' where id='{}'".format(data[0],
                                                                                                     data[1],
                                                                                                     data[2])
    self._cur.execute(sql)
```

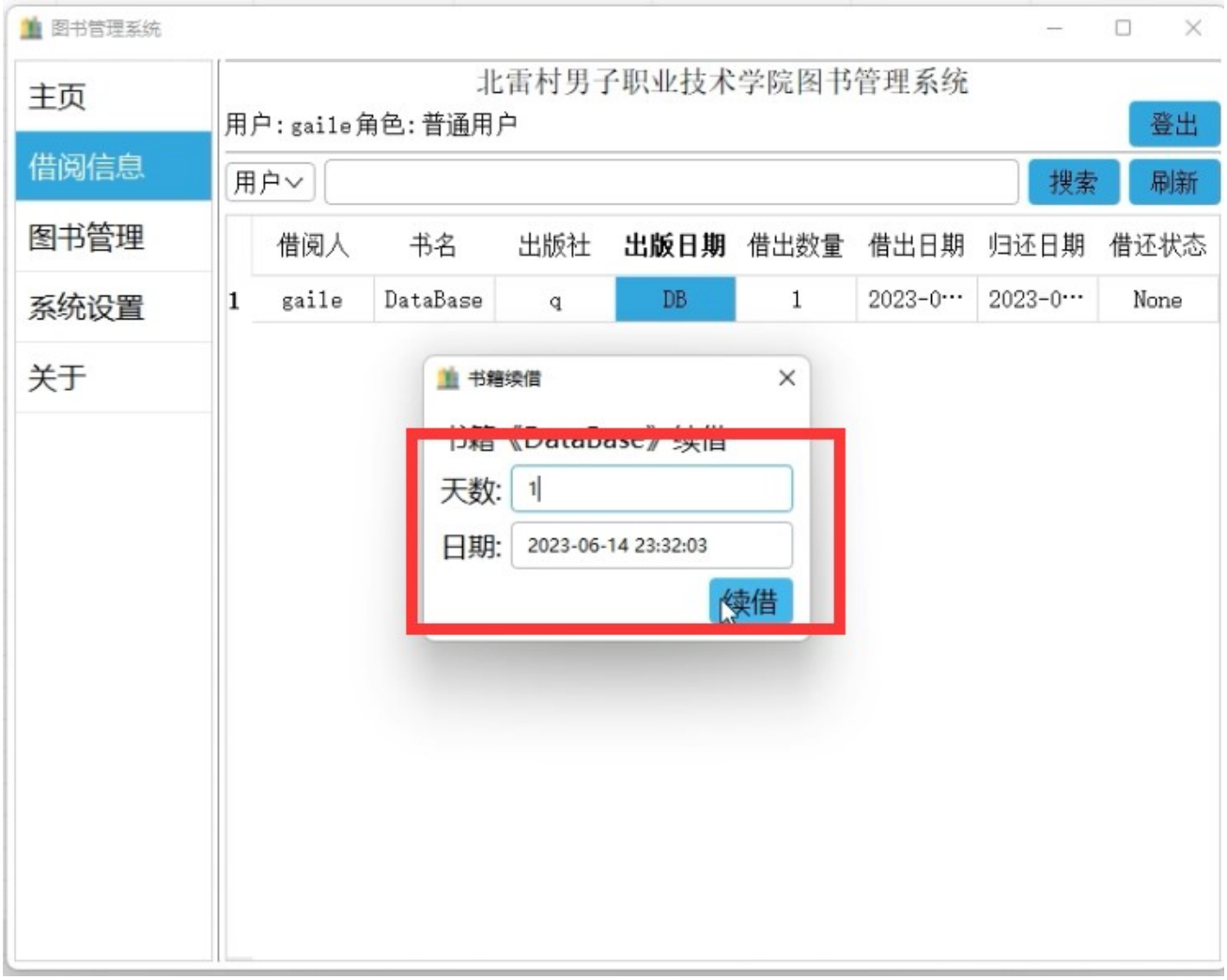
上述 SQL 语句实现更新借阅信息表中指定 ID 的记录的借阅天数和归还时间（用于续借操作）。

```
def update_borrow(self, book_id):
    sql = "update book set store_number=store_number-1 where id='{}'".format(book_id)
    self._cur.execute(sql)
    sql = "update book set borrow_number=borrow_number+1 where id='{}'".format(book_id)
    self._cur.execute(sql)
```

上述 SQL 语句实现更新图书表中指定 ID 的记录的借出数量和库存数量。

```
def insert_borrow_info(self, data):
    sql = "insert into borrow_info (id, book_id, book_name, borrow_user, borrow_num, borrow_days, borrow_time," \
          "return_time, return_flag) values (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
    self._cur.execute(sql, data)
```

之后向借阅信息表添加一条新记录。




```
def update_super(self, table_name, column_name, condition, data):
    sql = "update {} set book_name='{}', author='{}', publish_company='{}', publish_date='{}', store_number={}, ISBN_Code={}, Book_Classification={}, " \
          " where {}='{}'".format(table_name, data[0], data[1], data[2], data[3], data[4], data[5], data[6], column_name, condition)
    self._cur.execute(sql)
```

根据指定条件更新指定表中的记录，将归还时间进行更新。

```
def __init__(self, host='127.0.0.1', port=3306, user='root', pwd='admin', db='book', charset='utf8'):
    self._conn = pymysql.connect(host=host, port=port, user=user, passwd=pwd, db=db, charset=charset)
    self._cur = self._conn.cursor()
```

连接数据库的相关操作。

```
def db_commit(self):
    self._conn.commit()

def db_rollback(self):
    self._conn.rollback()
```

最后提交并回滚对数据库的更改。

3.3 功能举例

编辑图书信息功能展示：

```
def setupUi(self, Form):
    Form.setObjectName("Form")
    Form.resize(725, 452)
    self.gridLayout = QtWidgets.QGridLayout(Form)
    self.gridLayout.setContentsMargins(0, 0, 0, 0)
    self.gridLayout.setObjectName("gridLayout")
    self.verticalLayout = QtWidgets.QVBoxLayout()
    self.verticalLayout.setObjectName("verticalLayout")
    self.horizontalLayout = QtWidgets.QHBoxLayout()
    self.horizontalLayout.setObjectName("horizontalLayout")
    self.comboBox = QtWidgets.QComboBox(Form)
    self.comboBox.setStyleSheet("font: 12pt \"宋体\";")
    self.comboBox.setObjectName("comboBox")
    self.comboBox.addItem("")
    self.comboBox.addItem("")
    self.horizontalLayout.addWidget(self.comboBox)
    self.borrow_user_search_lineEdit = QtWidgets.QLineEdit(Form)
    self.borrow_user_search_lineEdit.setStyleSheet("font: 12pt \"宋体\";")
    self.borrow_user_search_lineEdit.setObjectName("borrow_user_search_lineEdit")
    self.horizontalLayout.addWidget(self.borrow_user_search_lineEdit)
    self.search_borrow_user_pushButton = QtWidgets.QPushButton(Form)
    self.search_borrow_user_pushButton.setStyleSheet("font: 12pt \"宋体\";")
    self.search_borrow_user_pushButton.setObjectName("search_borrow_user_pushButton")
    self.horizontalLayout.addWidget(self.search_borrow_user_pushButton)
    self.refresh_pushButton = QtWidgets.QPushButton(Form)
    self.refresh_pushButton.setStyleSheet("font: 12pt \"宋体\";")
    self.refresh_pushButton.setObjectName("refresh_pushButton")
    self.horizontalLayout.addWidget(self.refresh_pushButton)
```

上述代码主要是进行界面 UI 的定义，设置了字的大小，使用的字体和对应按钮等。

```python
def retranslateUi(self, Form):
    _translate = QtCore.QCoreApplication.translate
    Form.setWindowTitle(_translate("Form", "Form"))
    self.search_comboBox.setItemText(0, _translate("Form", "书名"))
    self.search_comboBox.setItemText(1, _translate("Form", "作者"))
    self.search_comboBox.setItemText(2, _translate("Form", "出版社"))
    self.search_book_pushButton.setText(_translate("Form", "搜索"))
    self.add_book_pushButton.setText(_translate("Form", "添加图书"))
    self.refresh_pushButton.setText(_translate("Form", "刷新库"))
    self.book_total_label.setText(_translate("Form", "TextLabel"))
    item = self.tableWidget.horizontalHeaderItem(0)
    item.setText(_translate("Form", "书名"))
    item = self.tableWidget.horizontalHeaderItem(1)
    item.setText(_translate("Form", "作者"))
    item = self.tableWidget.horizontalHeaderItem(2)
    item.setText(_translate("Form", "出版社"))
    item = self.tableWidget.horizontalHeaderItem(3)
    item.setText(_translate("Form", "出版日期"))
    item = self.tableWidget.horizontalHeaderItem(4)
    item.setText(_translate("Form", "库存数量"))
    item = self.tableWidget.horizontalHeaderItem(5)
    item.setText(_translate("Form", "借出数量"))
    item = self.tableWidget.horizontalHeaderItem(6)
    item.setText(_translate("Form", "ISBN"))
    item = self.tableWidget.horizontalHeaderItem(7)
    item.setText(_translate("Form", "图书分类"))
```

进行相应标签的设定和按钮的设置。

```
def init_ui(self):
    self.setWindowTitle('编辑图书')
    self.setWindowModality(Qt.ApplicationModal)
    self.add_book_pushButton.setText('保存信息')
    self.setWindowIcon(QIcon(APP_ICON))
    self.setWindowFlags(Qt.WindowCloseButtonHint)
    self.add_book_pushButton.setProperty('class', 'Aqua')
    self.setStyleSheet(SYS_STYLE)
    self.add_book_pushButton.setMinimumWidth(60)
```

init_ui 方法初始化 UI 小部件，如 tableWidget、pushButton，并设置它们的属性。如果用户角色是“管理员”，它还会生成用于编辑或删除书籍的上下文菜单。

```
def init_data(self):
    self.book_name_lineEdit.setText(self.current_book_info[1])
    self.author_lineEdit.setText(self.current_book_info[2])
    self.publish_company_lineEdit.setText(self.current_book_info[3])
    self.publish_date_lineEdit.setText(str(self.current_book_info[-1]))
    self.store_num_lineEdit.setText(str(self.current_book_info[4]))
```

在 init_data 方法中，根据从数据库查询到的当前图书信息，将其填充到窗口控件中。

```
def get_book_info(self):
    db = DBHelp()
    count, res = db.query_super(table_name='book', column_name='booK_name', condition=self.book_info)
    self.current_book_info = list(res[0])
    self.init_book_info_done_signal.emit()
    db.instance = None
    del db
```

过 DBHelp 类实例化一个数据库对象，查询指定 book_info 的图书信息，并将其转换成列表类型 self.current_book_info，发送自定义的信号 init_book_info_done_signal 表示初始化图书信息完成。

```python
def update_book_info(self):
    book_name = self.book_name_lineEdit.text()
    author = self.author_lineEdit.text()
    publish_company = self.publish_company_lineEdit.text()
    publish_time = self.publish_date_lineEdit.text()
    ISBN_Code = self.book_name_lineEdit.text()
    Book_Classification = self.book_name_lineEdit.text()
    store_num = int(self.store_num_lineEdit.text())
    new_book_info = [book_name, author, publish_company, publish_time, store_num, ISBN_Code, Book_Classification]
    is_update = False
    if '' in new_book_info:
        msg_box(self, '错误', '图书的关键信息不能为空!')
        return
    for new_info in new_book_info:
        if new_info not in self.current_book_info:
            db = DBHelp()
            db.update_super(table_name='book', column_name='id', condition=self.current_book_info[0],
                            data=new_book_info)
            db.db_commit()
            db.instance = None
            del db
            self.close()
            is_update = True
    if is_update:
            msg_box(self, '提示', '图书信息更新成功!')
    self.close()
```

此方法首先获取输入的新图书信息，判断是否有关键信息为空，如果有则提示错误信息并返回。否则，遍历新信息列表 new_book_info，如果与当前图书信息不同，则更新数据库中的数据，并提示更新成功信息。最后关闭当前窗口。

上述代码实现了编辑已有图书信息的功能。可以修改图书的基本信息，并保存到数据库中。

其余功能已于视频中进行展示，本系统已实现图书馆借还书系统数据库中所使用的基本功能。

4 作业诚信承诺

本报告是我个人独立完成的，并非从网上或从其他同学及从其他来源获取得到的，报告中引用他人的成果、数据、观点均已一一注明出处。

承诺人电子签名：