

浙江大学



数据库系统实验五

题 目：	图书管理系统
授课教师：	周波
姓 名：	王伟杰
学 号：	3210106034
日 期：	2023-05-02

目录

- 1 实验目的
- 2 实验需求
 - 2.1 基本数据对象
 - 2.2 基本功能模块
 - 2.3 系统功能验证
- 3 实验环境
 - 3.1 操作系统
 - 3.2 DBMS
 - 3.3 IDE
- 4 系统设计
 - 4.1 数据库(表)设计
 - 4.2 功能函数接口设计
- 5 实验过程
 - 5.1 框架准备
 - 5.1.1 环境配置
 - 5.1.2 项目开发准备
 - 5.2 实现 `LibraryManagementImpl`
 - 5.3 图形界面
 - 5.3.1 连接和断开数据库
 - 5.3.2 侧边栏
 - 5.3.3 数据库重置
 - 5.3.4 图书管理部分
 - 5.3.4.1 界面表格
 - 5.3.4.2 添加书籍
 - 5.3.4.3 修改库存
 - 5.3.4.4 批量添加书籍
 - 5.3.4.5 删除书籍
 - 5.3.4.6 修改书籍
 - 5.3.4.7 查询书籍
 - 5.3.5 借书证管理
 - 5.3.5.1 界面表格
 - 5.3.5.2 注册借书卡
 - 5.3.5.3 注销借书卡
 - 5.3.6 借书还书
 - 5.3.6.1 界面表格
 - 5.3.6.2 借书
 - 5.3.6.3 还书
 - 5.3.6.4 查询借书历史

6 系统测试

6.1 正确性测试

6.2 功能性测试

7 遇到的问题及解决方法

7.1 环境配置

7.2 SQL语句表达

7.3 图形界面选择

7.4 `Main` 函数无法运行

8 思考题

8.1 绘制该图书管理系统的E-R图。

8.2 描述SQL注入攻击的原理。本系统哪些模块可能会遭受SQL注入攻击？如何解决？

8.3 在InnoDB的默认隔离级别(RR, Repeated Read)下，当出现并发访问时，如何保证借书结果的正确性？

1 实验目的

- 设计并实现一个精简的图书管理程序，要求具有图书入库、查询、借书、还书、借书证管理等功能。

2 实验需求

2.1 基本数据对象

所有的基本数据对象都被定义在 `*entities*` 包中，以下是这些对象的基本信息，其它信息参考类中的注释。

对象名称	类名	包含属性
书	Book	书号, 类别, 书名, 出版社, 年份, 作者, 价格, 剩余库存
借书证	Card	卡号, 姓名, 单位, 身份(教师或学生)
借书记录	Borrow	卡号, 书号, 借书日期, 还书日期

2.2 基本功能模块

图书管理系统中所有应具备的功能模块都在接口 `*LibraryManagementSystem*` 中被声明：

- 在实现交互界面之前，首先需要为该接口做一个具体的实现(在实现类中重写接口中的方法)，具体实现位于类 `*LibraryManagementSystemImpl*` 中。
- 在完成所有的接口后，可以在主类 `*Main*` 中通过 `*LibraryManagementSystem*` 的一个实例来实现与用户的交互。这样设计的好处在于它满足了(1) 封装性：对客户端(主类 `*Main*`)屏蔽了图书管理系统底层的具体实现；(2) 可扩展性：可以很方便地对图书管理系统做不同的实现(例如再实现一个不依赖于数据库的基于内存的图书管理系统)；(3) 可维护性：只需要在客户端中切换 `*LibraryManagementSystem*` 的实例即可使用不同实现方式的图书管理系统。

2.3 系统功能验证

系统功能验证测试分为功能性测试和正确性测试。

- 正确性测试通过测试用例进行评判，以验收时通过的测试用例数量占总测试用例数量的百分比来评定正确性测试部分的得分。
- 功能性测试通过验收时运行模拟场景的结果进行评判，以软件使用时的交互友好程度、效率、正确性等指标来评定功能性测试部分的得分。

功能性测试的参考模拟场景如下：

功能	描述
图书入库	输入<书号, 类别, 书名, 出版社, 年份, 作者, 价格, 初始库存>, 入库一本新书B
增加库存	将书B的库存增加到X, 然后减少到1
修改图书信息	随机抽取N个字段, 修改图书B的图书信息
批量入库	输入图书导入文件的路径U, 然后从文件U中批量导入图书
添加借书证	输入<姓名, 单位, 身份>, 添加一张新的借书证C
查询借书证	列出所有的借书证
借书	用借书证C借图书B, 再借一次B, 然后再借一本书K
还书	用借书证C还掉刚刚借到的书B
借书记录查询	查询C的借书记录
图书查询	从查询条件<类别点查(精确查询), 书名点查(模糊查询), 出版社点查(模糊查询), 年份范围查, 作者点查(模糊查询), 价格范围查>中随机选取N个条件, 并随机选取一个排序列和顺序

3 实验环境

3.1 操作系统

- Windows 10 专业版

3.2 DBMS

- MySQL

```

1 $ mysql -V
2 mysql Ver 8.0.30 for Win64 on x86_64 (MySQL Community Server - GPL)

```

3.3 IDE

由于我使用Java语言完成本次实验, 所以选择了[IntelliJ IDEA](#) 作为集成开发环境

- Java环境

```

PS E:\SOPHOMORE_SS\Database_System\lab\lab5\librarymanagementsystem> java --version
java 18.0.1.1 2022-04-22
Java(TM) SE Runtime Environment (build 18.0.1.1+2-6)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.1.1+2-6, mixed mode, sharing)
PS E:\SOPHOMORE_SS\Database_System\lab\lab5\librarymanagementsystem>

```

- mvn环境

```
D:\Java\bin\java.exe -Dmaven.multiModuleProjectDirectory=E:\SOPHOMORE_SS\Da
Apache Maven 3.8.1 (05c21c65bdfed0f71a2f2ada8b84da59348c4c5d)
Maven home: F:\IntelliJ IDEA 2023.1\plugins\maven\lib\maven3
Java version: 18.0.1.1, vendor: Oracle Corporation, runtime: D:\Java
Default locale: zh_CN, platform encoding: UTF-8
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

4 系统设计

4.1 数据库(表)设计

以下是该图书管理系统的数据表定义：

```
1  create table `book` (
2      `book_id` int not null auto_increment,
3      `category` varchar(63) not null,
4      `title` varchar(63) not null,
5      `press` varchar(63) not null,
6      `publish_year` int not null,
7      `author` varchar(63) not null,
8      `price` decimal(7, 2) not null default 0.00,
9      `stock` int not null default 0,
10     primary key (`book_id`),
11     unique (`category`, `press`, `author`, `title`, `publish_year`)
12 );
13
14 create table `card` (
15     `card_id` int not null auto_increment,
16     `name` varchar(63) not null,
17     `department` varchar(63) not null,
18     `type` char(1) not null,
19     primary key (`card_id`),
20     unique (`department`, `type`, `name`),
21     check ( `type` in ('T', 'S') )
22 );
23
24 create table `borrow` (
25     `card_id` int not null,
26     `book_id` int not null,
27     `borrow_time` bigint not null,
28     `return_time` bigint not null default 0,
29     primary key (`card_id`, `book_id`, `borrow_time`),
30     foreign key (`card_id`) references `card`(`card_id`) on delete cascade on update
    cascade,
31     foreign key (`book_id`) references `book`(`book_id`) on delete cascade on update
    cascade
32 );
```

4.2 功能函数接口设计

以下是 `*LibraryManagementSystem*` 中声明的模块：

- `*ApiResult storeBook(Book book)*`：图书入库模块。向图书库中注册(添加)一本新书，并返回新书的书号。如果该书已经存在于图书库中，那么入库操作将失败。当且仅当书的<类别, 书名, 出版社, 年份, 作者>均相同时，才认为两本书相同。插入完成后，需要根据数据库中自增列生成的book_id去更新book对象里的book_id。
- `*ApiResult incBookStock(int bookId, int deltaStock)*`：图书增加库存模块。为图书库中的某一本书增加库存。其中库存增量 `*deltaStock*` 可正可负，若为负数，则需要保证最终库存是一个非负数。
- `*ApiResult storeBook(List<Book> books)*`：图书批量入库模块。批量入库图书，如果有一本书入库失败，那么就需要回滚整个事务(即所有的书都不能被入库)。
- `*ApiResult removeBook(int bookId)*`：图书删除模块。从图书库中删除一本书。如果还有人尚未归还这本书，那么删除操作将失败。
- `*ApiResult modifyBookInfo(Book book)*`：图书修改模块。修改已入库图书的基本信息，该接口不能修改图书的书号和存量。
- `*ApiResult queryBook(BookQueryConditions conditions)*`：图书查询模块。根据提供的查询条件查询符合条件的图书，并按照指定排序方式排序。查询条件包括：类别点查(精确查询)，书名点查(模糊查询)，出版社点查(模糊查询)，年份范围查，作者点查(模糊查询)，价格范围差。如果两条记录排序条件的值相等，则按book_id升序排序。
- `*ApiResult borrowBook(Borrow borrow)*`：借书模块。根据给定的书号、卡号和借书时间添加一条借书记录，然后更新库存。若用户此前已经借过这本书但尚未归还，那么借书操作将失败。
- `*ApiResult returnBook(Borrow borrow)*`：还书模块。根据给定的书号、卡号和还书时间，查询对应的借书记录，并补充归还时间，然后更新库存。
- `*ApiResult showBorrowHistory(int cardId)*`：借书记录查询模块。查询某个用户的借书记录，按照借书时间递减、书号递增的方式排序。
- `*ApiResult registerCard(Card card)*`：借书证注册模块。注册一个借书证，若借书证已经存在，则该操作将失败。当且仅当<姓名, 单位, 身份>均相同时，才认为两张借书证相同。
- `*ApiResult removeCard(int cardId)*`：删除借书证模块。如果该借书证还有未归还的图书，那么删除操作将失败。
- `*ApiResult showBooks()*`：书籍查询模块。列出所有的书籍。
- `*ApiResult showCards()*`：借书证查询模块。列出所有的借书证。
- `*ApiResult showBorrows()*`：借书记录查询模块。列出所有的借书记录。

5 实验过程

5.1 框架准备

5.1.1 环境配置

下载IDEA，从ZJU GIT上下载项目源代码文件，用IDEA打开。

在项目的架构设置界面中，选择项目使用的SDK，本项目使用的是开发机已经配置好的SDK18，Maven版本为3.8.1。

5.1.2 项目开发准备

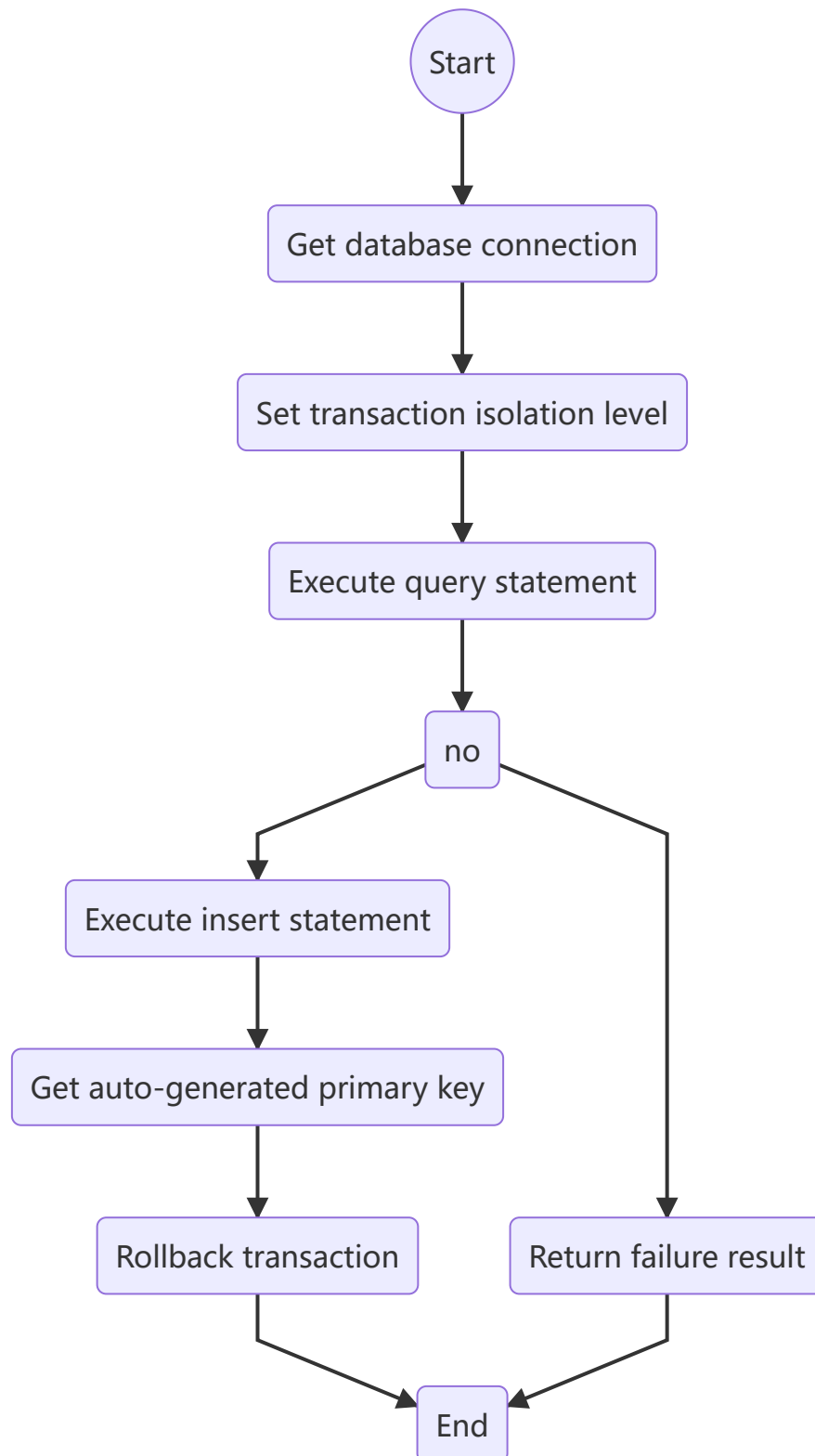
1. 在MySQL中创建 `library` 数据库
2. 将 `src/main/resources/application_template.yaml` 文件拷贝一份，并重命名为 `application.yaml`，然后按需修改该文件内的配置参数，各配置参数的含义如下：
 - `host`：主机名，默认值为 `"localhost"`

- `port`: 端口号, 默认值为 `"3306"`
- `user`: 用户名, 默认值为 `"root"`
- `password`: 密码, 默认值为 `" "`, 修改为本机密码
- `db`: 数据库名, 默认值为 `"library"`, 注意需要先数据库里创建
- `type`: 数据库类型, 可选值为 `["mysql", "postgresql", "sqlserver"]`, 默认值为 `"mysql"`

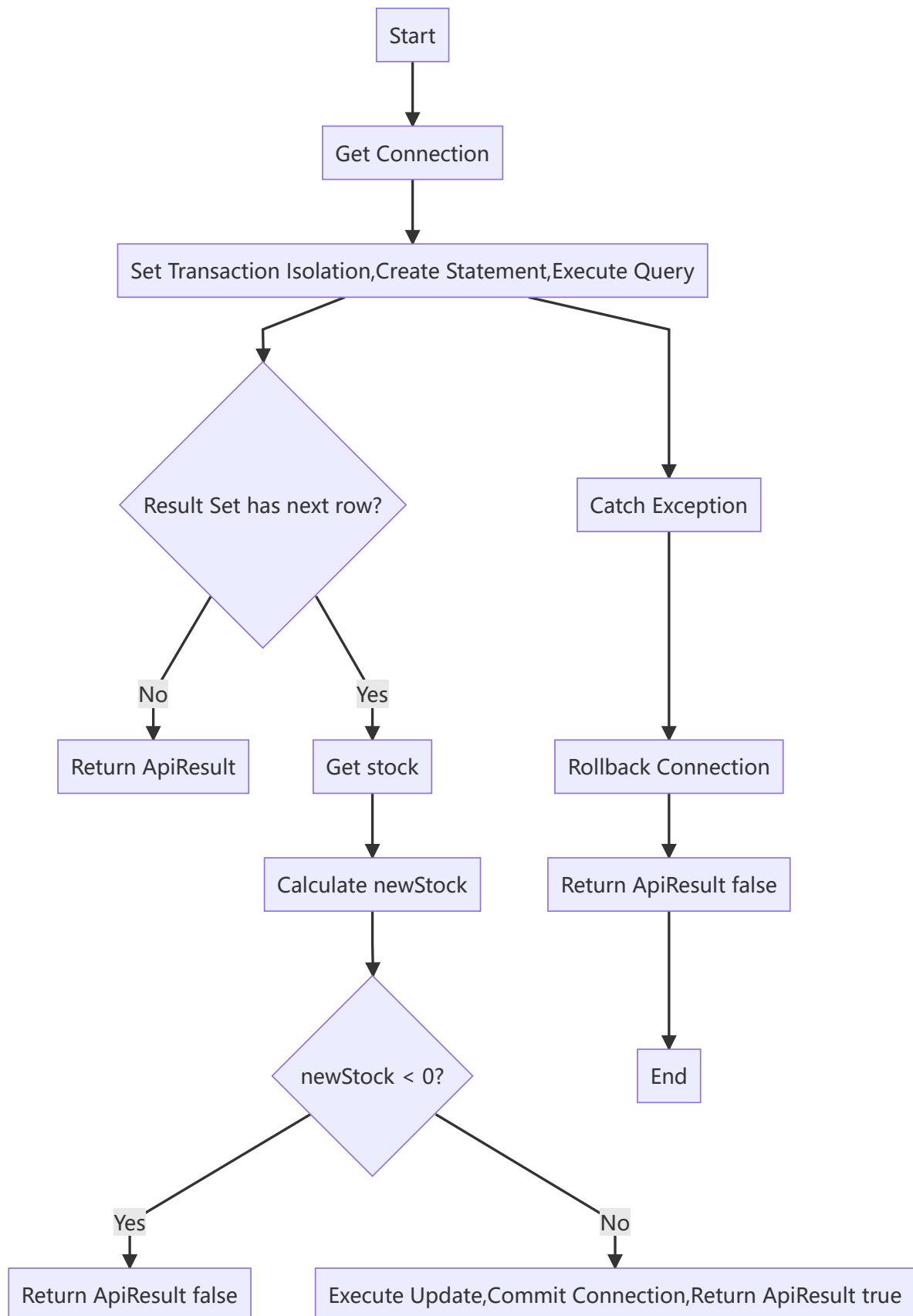
5.2 实现 `LibraryManagementImpl`

`LibraryManagementImpl` 文件中的各函数定义已经在 `LibraryManagement` 中给出, 需要按照相关提示进行开发

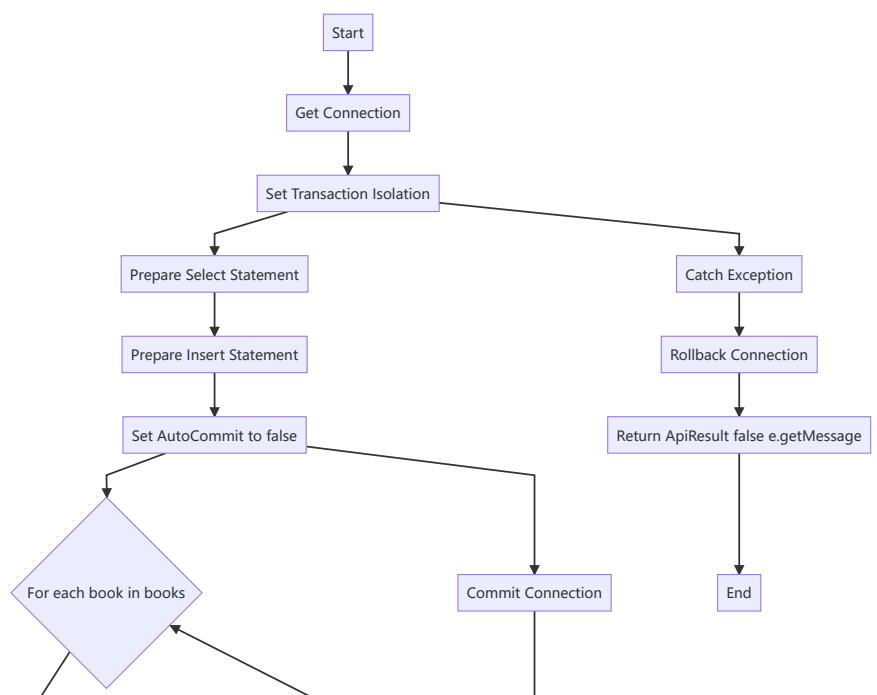
- `*ApiResult storeBook(Book book)*`: 图书入库模块。首先查询该图书是否已经存在，如果已经存在就抛出错误；如果不存在就按照给入的参数新增一本书。

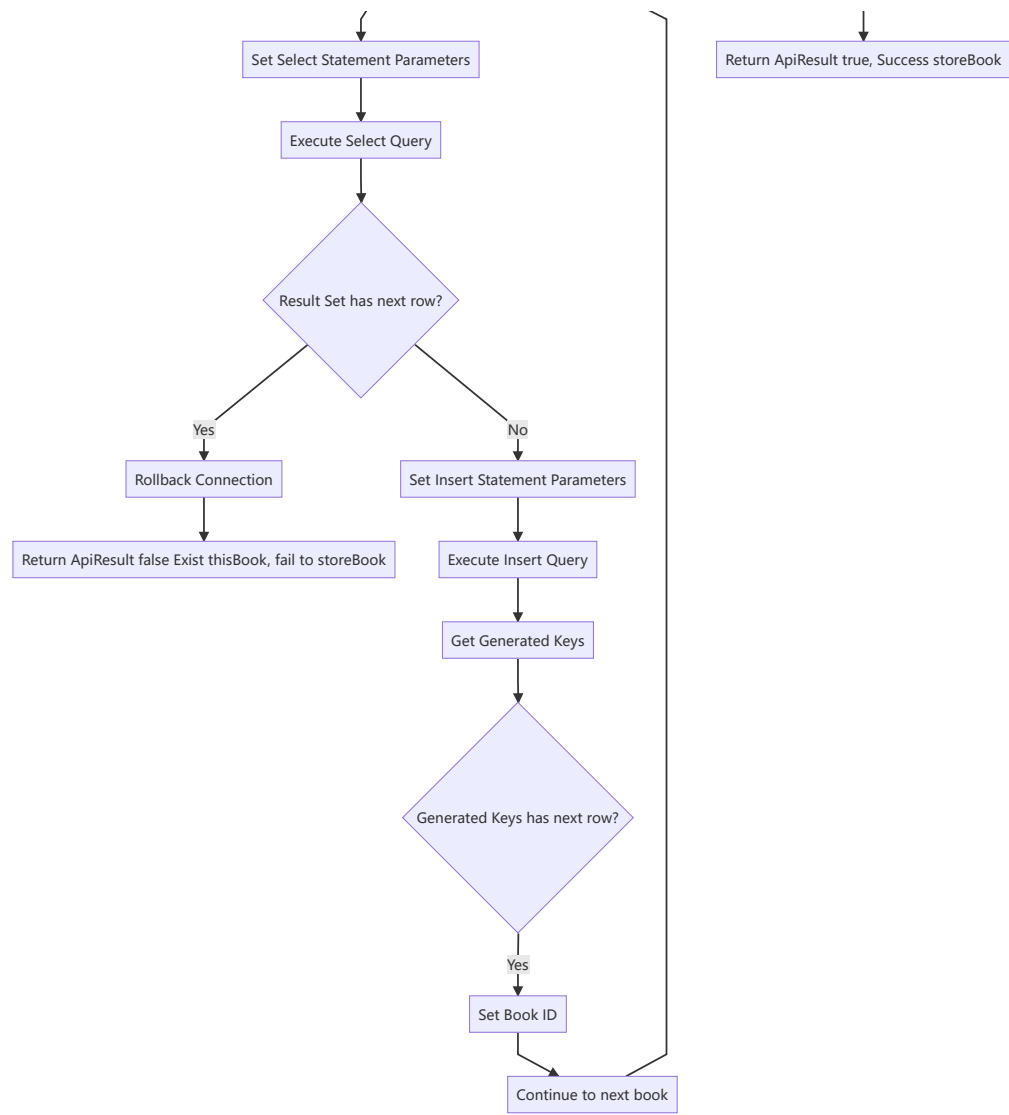


- `*ApiResult incBookStock(int bookId, int deltaStock)*`: 图书增加库存模块。首先检查书库里有没有这本书，如果没有就抛出错误，有的话就将 `delta` 的值增加到 `stock` 中，并判断是否为负数，如果是负数就抛出错误。

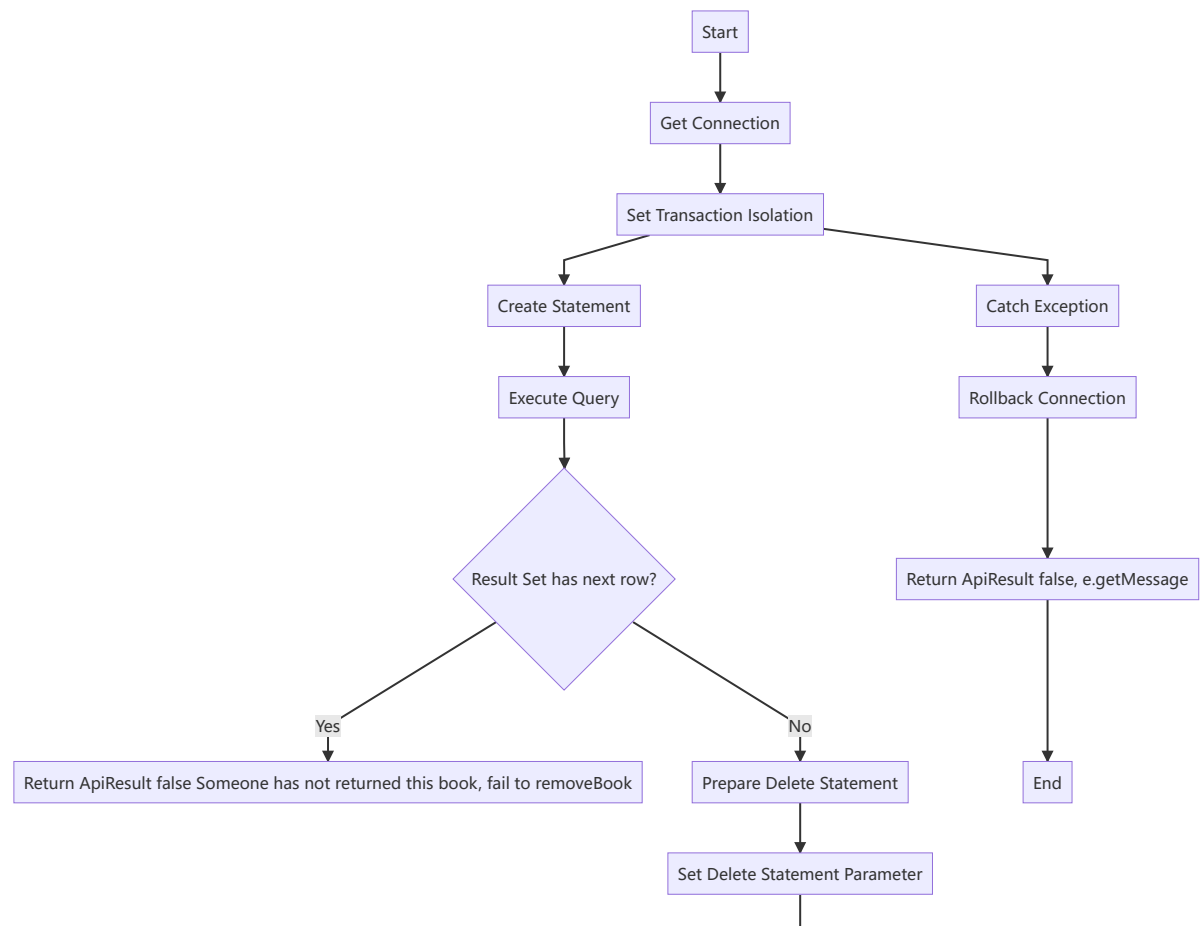


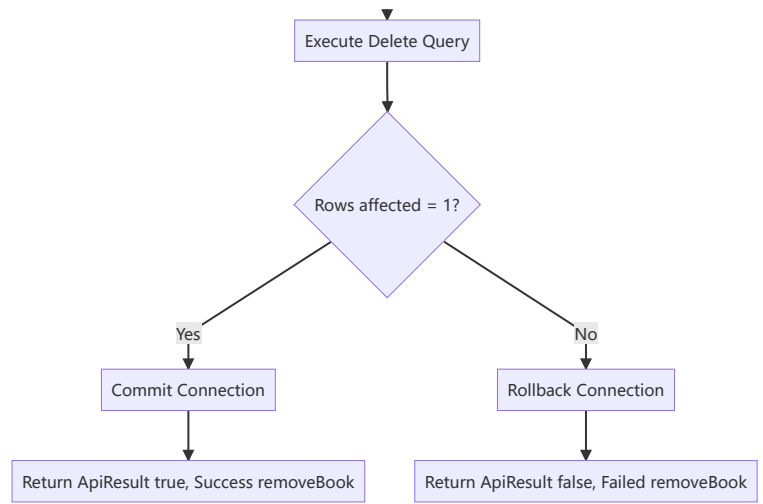
- `*ApiResponse storeBook(List<Book> books)*`: 图书批量入库模块。对于每一本要添加的书，查询是否已经在书库中，如果其中一本书已经存在，就不进行添加操作并进行回滚，否则就添加所有的书。



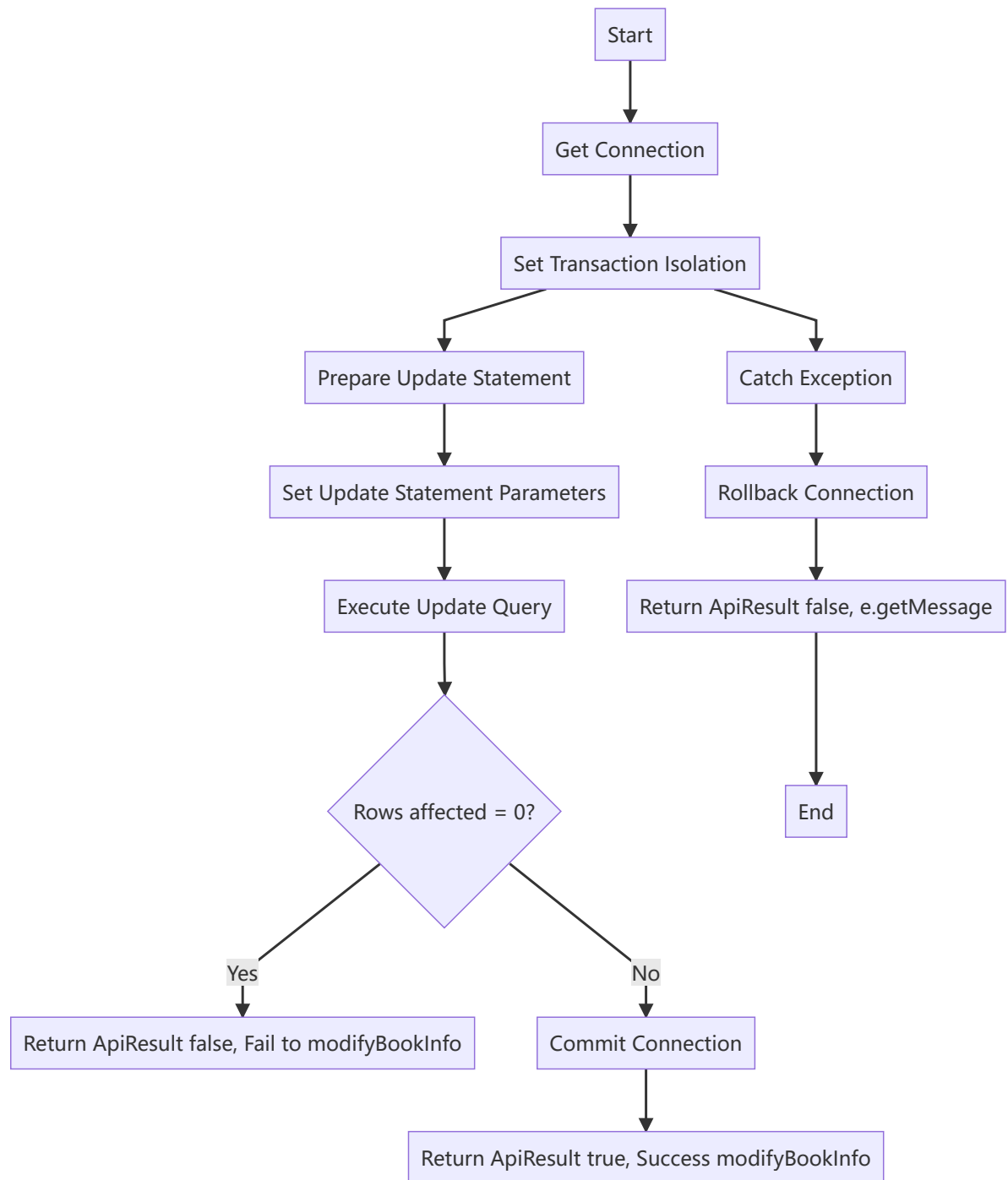


- `*ApiResult removeBook(int bookId)*`: 图书删除模块。对于每一本要添加的书，查询是否已经在书库中并且是否还有人没有归还，如果在书库中且没有人借出，则成功删除，否则抛出错误。

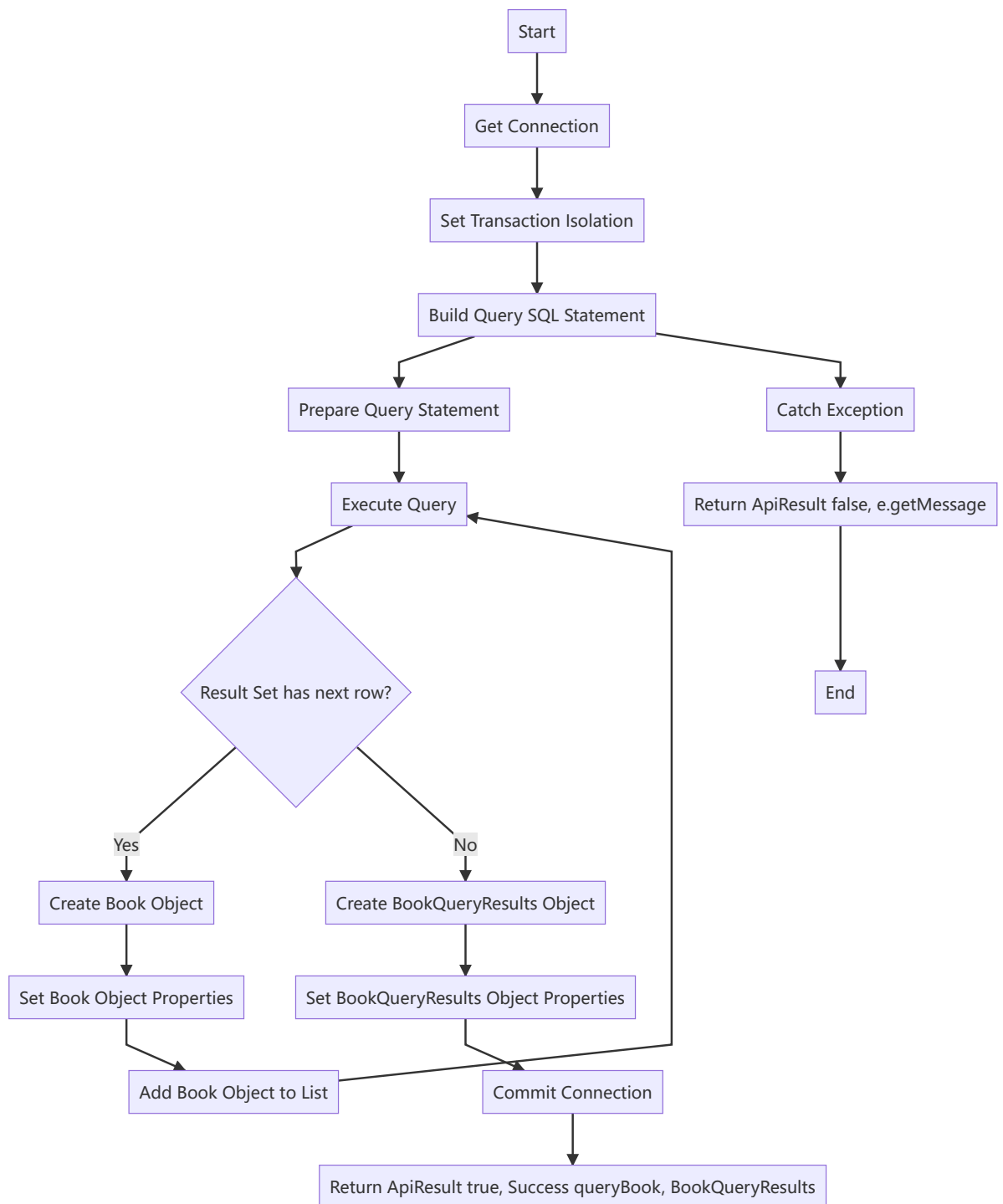




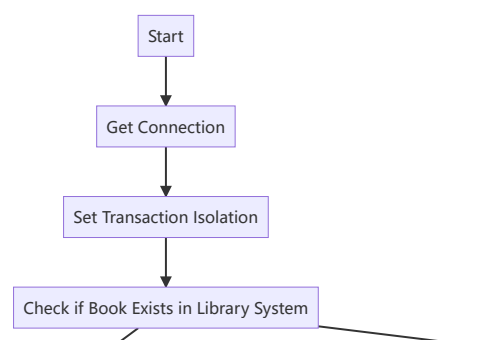
- `*ApiResult modifyBookInfo(Book book)*`: 图书修改模块。直接进行修改操作，如果不能修改则抛出错误(即找不到该图书)，否则成功修改

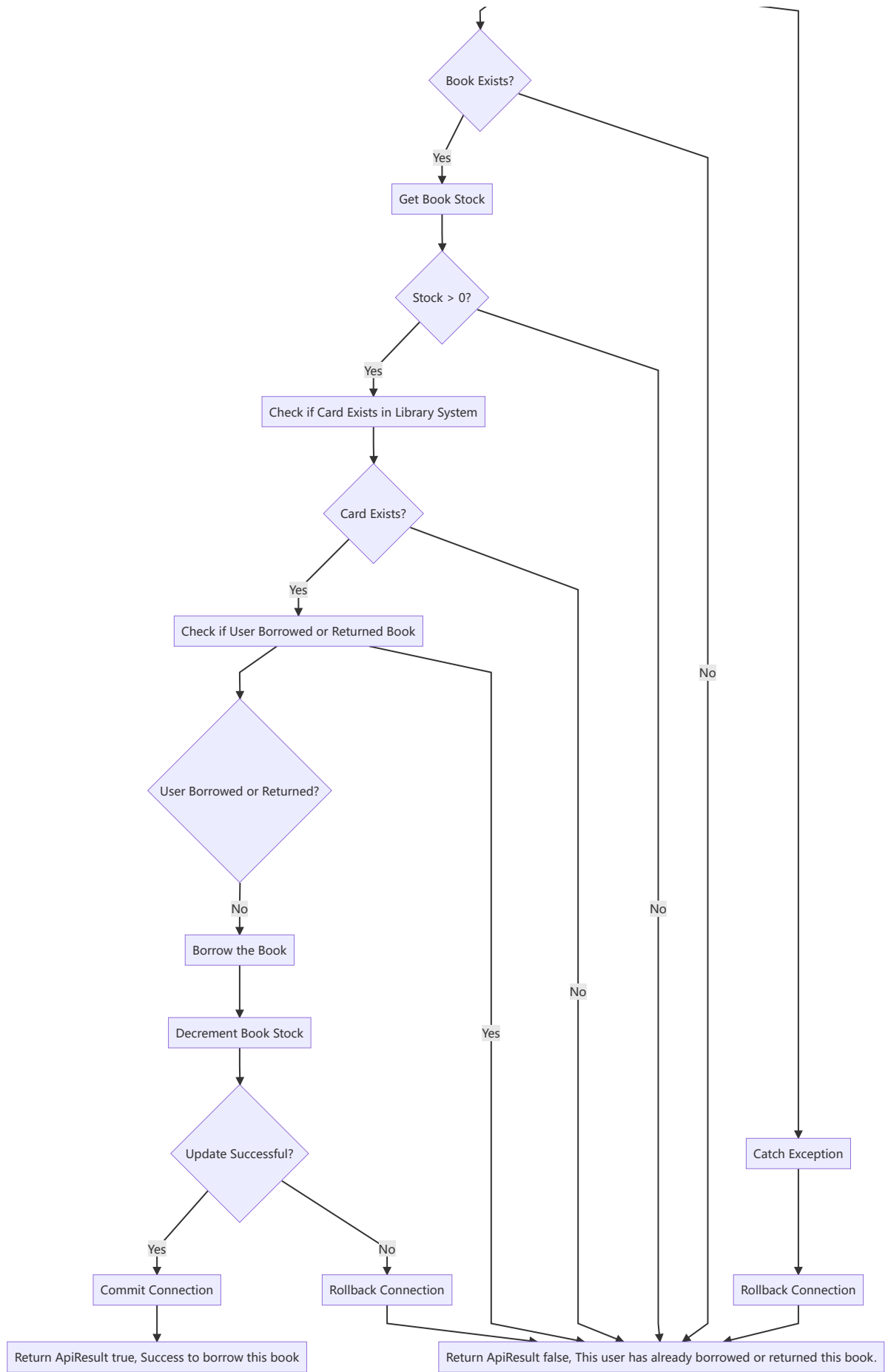


- `*ApiResult queryBook(BookQueryConditions conditions)*`: 图书查询模块。先根据提供的查询条件构造查询的sql语句，然后执行并返回结果。

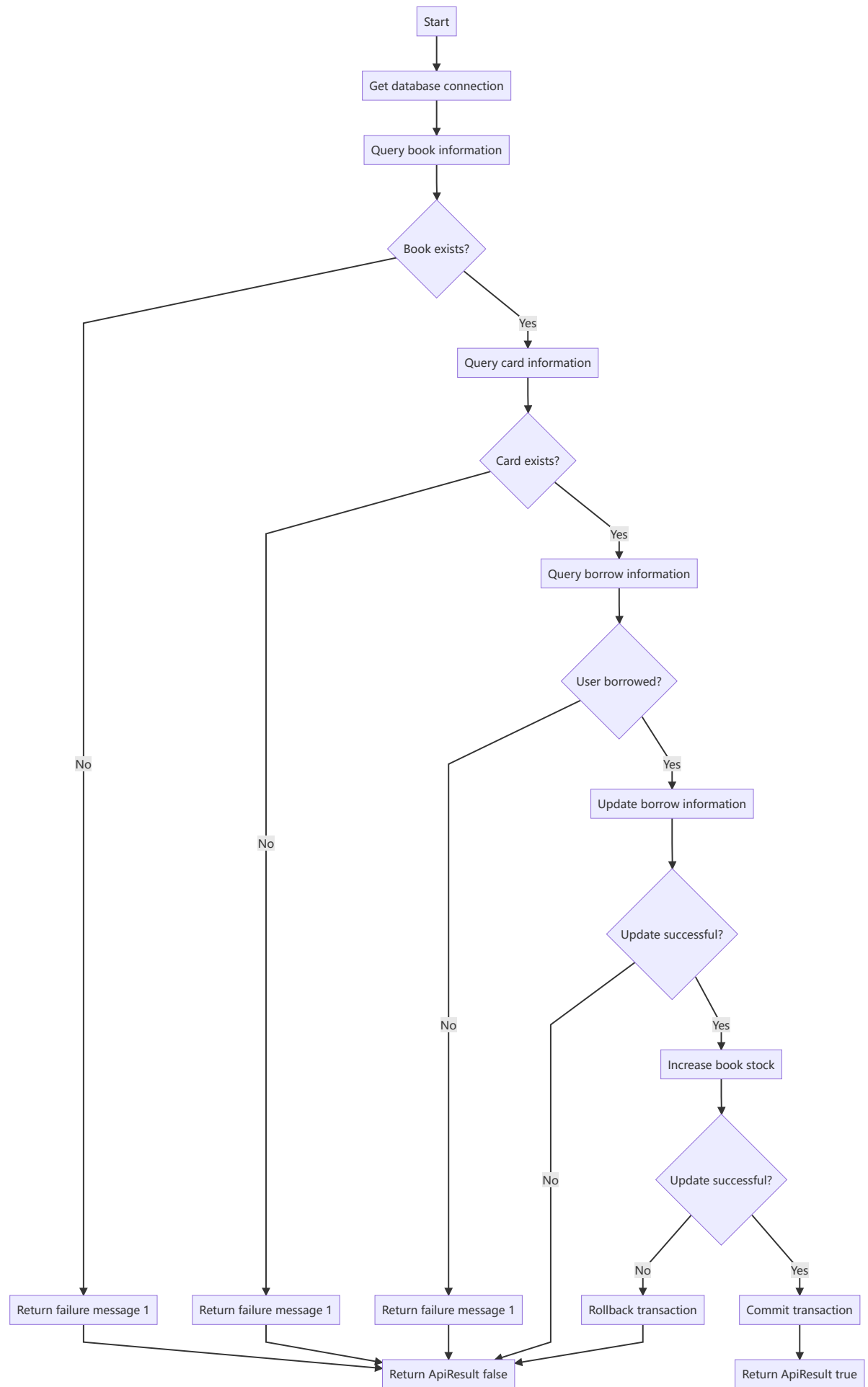


- `*ApiResult borrowBook(Borrow borrow)*`: 借书模块。根据给定的书号、卡号查询是否存在，若不存在就抛出错误，否则根据提供的借书时间添加一条记录。

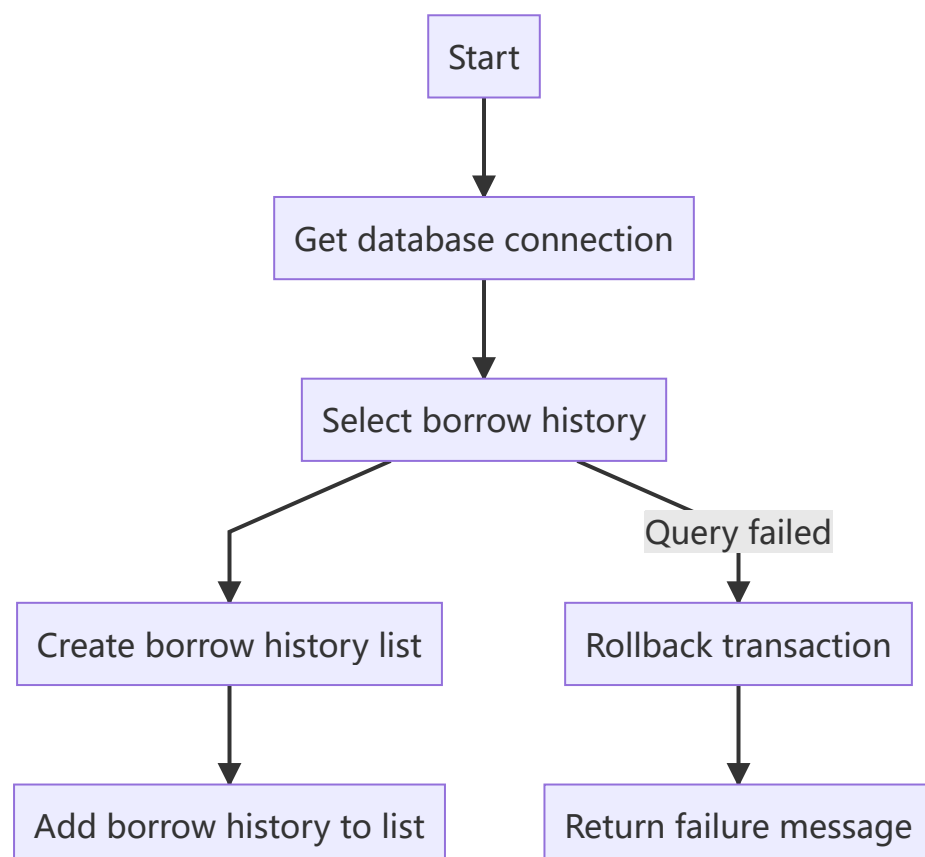


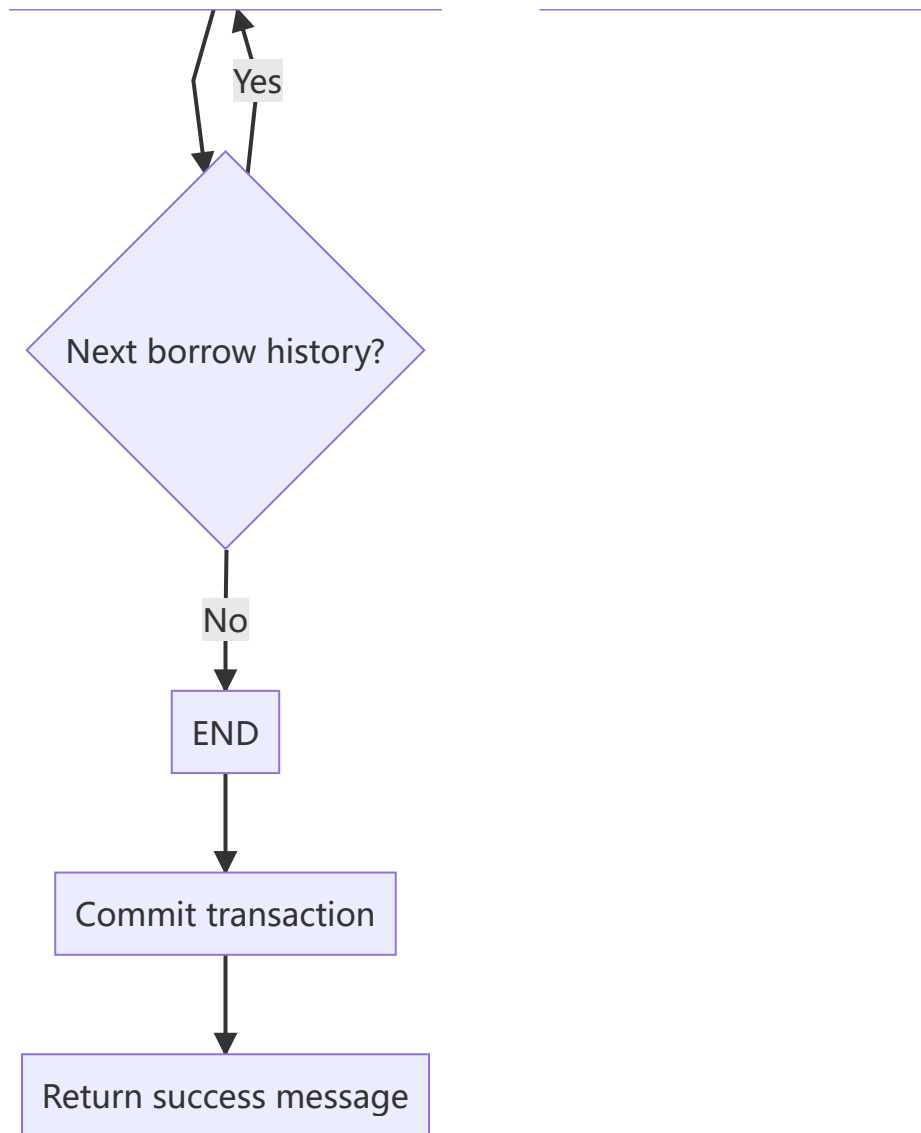


- `*ApiResponse returnBook(Borrow borrow)*`: 还书模块。根据给定的书号、卡号和还书时间，查询对应的借书记录，并补充归还时间，然后更新库存。

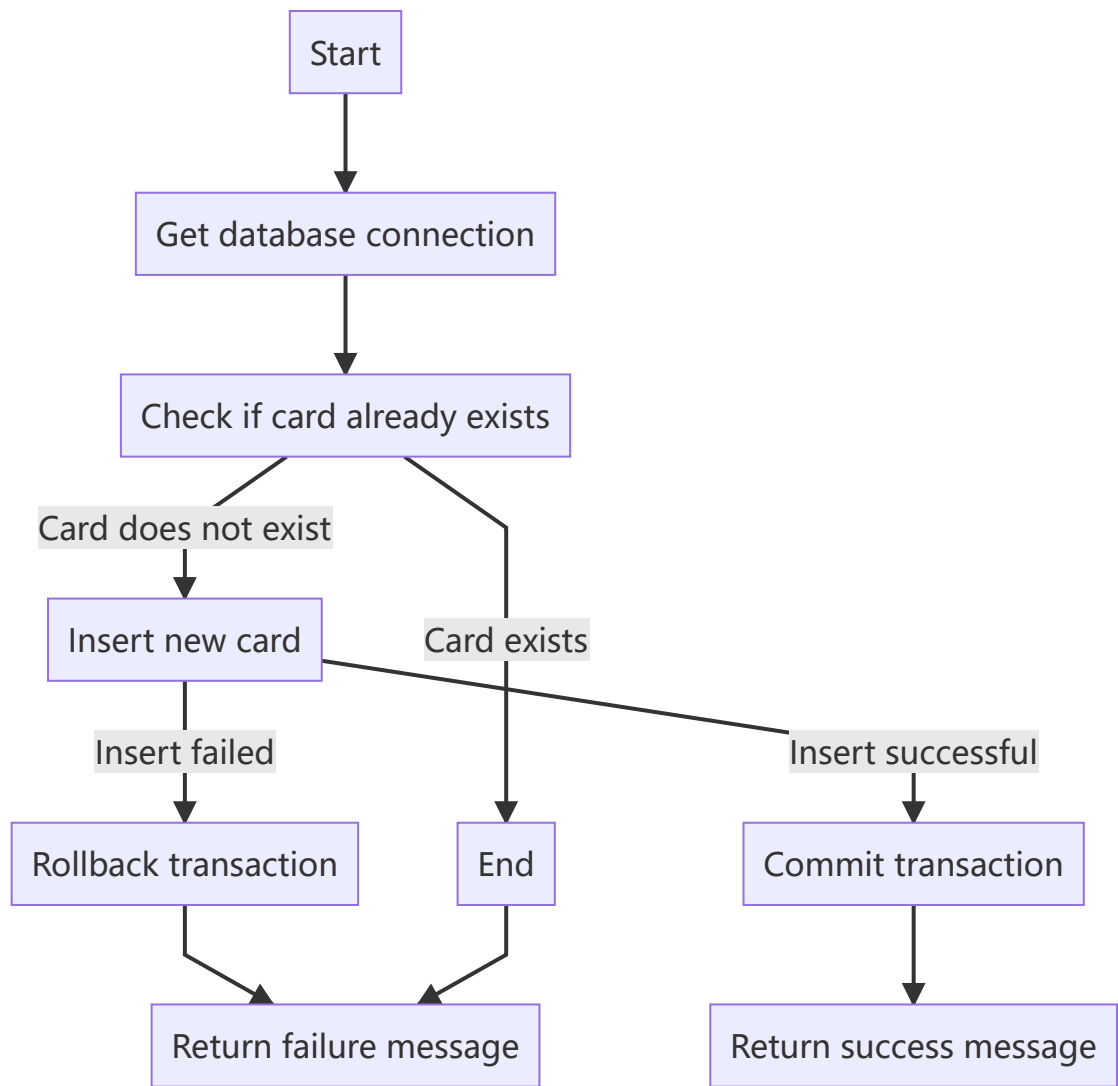


- `*ApiResponse showBorrowHistory(int cardId)*`: 借书记录查询模块。根据提供的 `cardID` 查询某个用户的借书记录，按照借书时间递减、书号递增的方式排序。

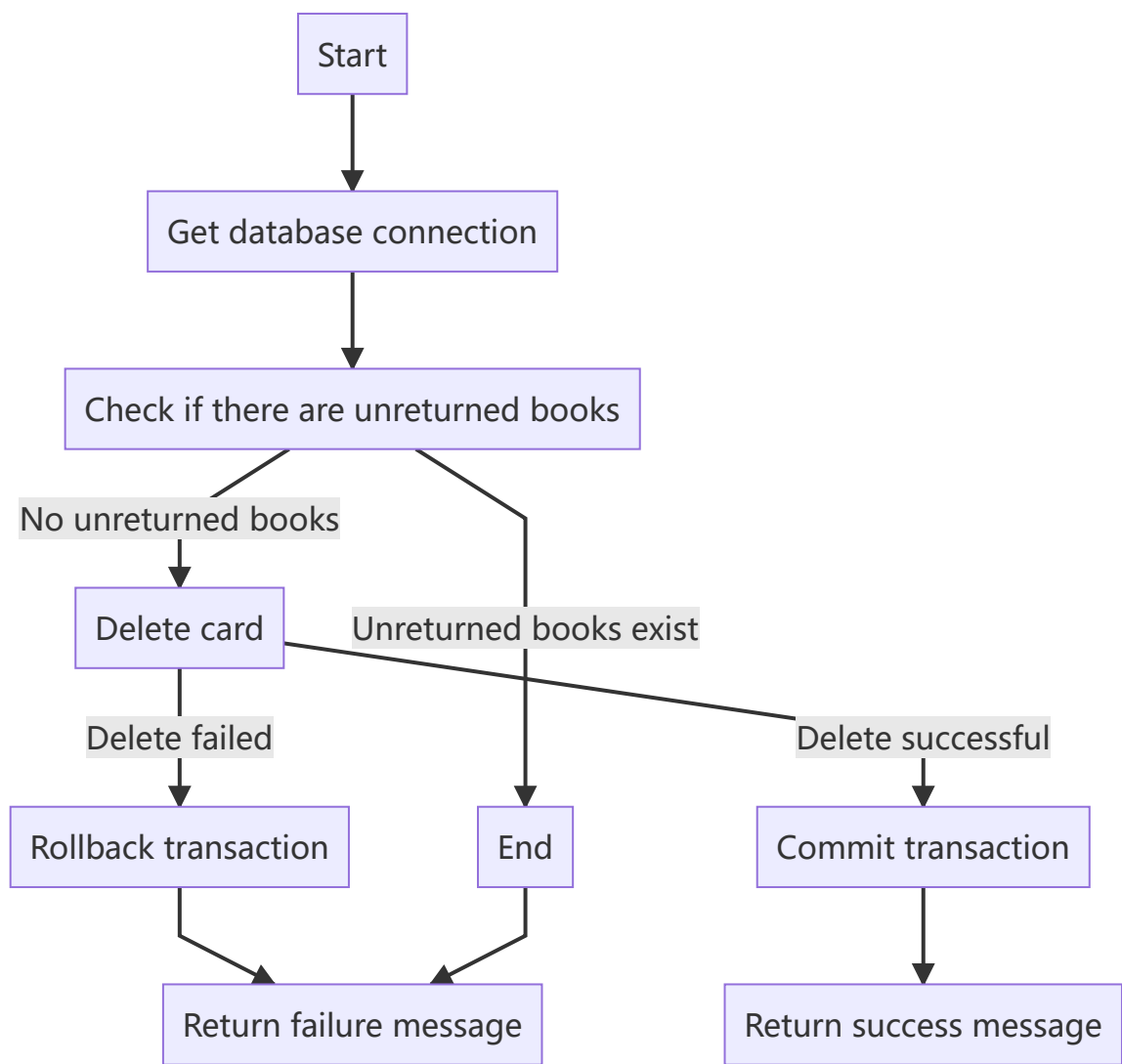




- `*ApiResult registerCard(Card card)*`: 借书证注册模块。注册一个借书证，若借书证已经存在，则该操作将失败。当且仅当<姓名, 单位, 身份>均相同时，才认为两张借书证相同。



- `*ApiResult removeCard(int cardId)*`: 删除借书证模块。如果该借书证还有未归还的图书，那么删除操作将失败。



- `*ApiResponse showBooks()*`: 书籍查询模块。查询所有的书籍，将返回结果存储在 `ApiResponse` 中。
- `*ApiResponse showCards()*`: 借书证查询模块。查询所有的借书证，将返回结果存储在 `ApiResponse` 中。
- `*ApiResponse showBorrows()*`: 借书记录查询模块。查询所有的借书记录，将返回结果存储在 `ApiResponse` 中。

5.3 图形界面

在 `Main.java` 中进行开发，编写图形界面并调用 `LibraryManagementImpl` 中实现过的所有函数接口，实现相关功能。

5.3.1 连接和断开数据库

在 `main` 函数的开始和结束部分连接和断开数据库。

5.3.2 侧边栏

图形界面将这些功能分为三个部分，分别是图书管理、借书证管理和借书还书。这三个模块通过 `sidebar` 进行切换，`sidebar`中包含三个按钮，分别对应三个部分。

5.3.3 数据库重置

数据库重置按钮位于整个界面右下方，点击后会弹出窗口确认是否要清空数据库。



5.3.4 图书管理部分



5.3.4.1 界面表格

图书管理的主界面中有一个表格，调用了 `ApiResponse showBooks()` 接口进行图书展示。

5.3.4.2 添加书籍

点击按钮，会弹出窗口询问要添加书籍的信息，如果能成功添加就会弹出success信号且在主界面同步展示新添加的书籍信息，否则弹出fail信号。



添加书籍



category:

title:

press:

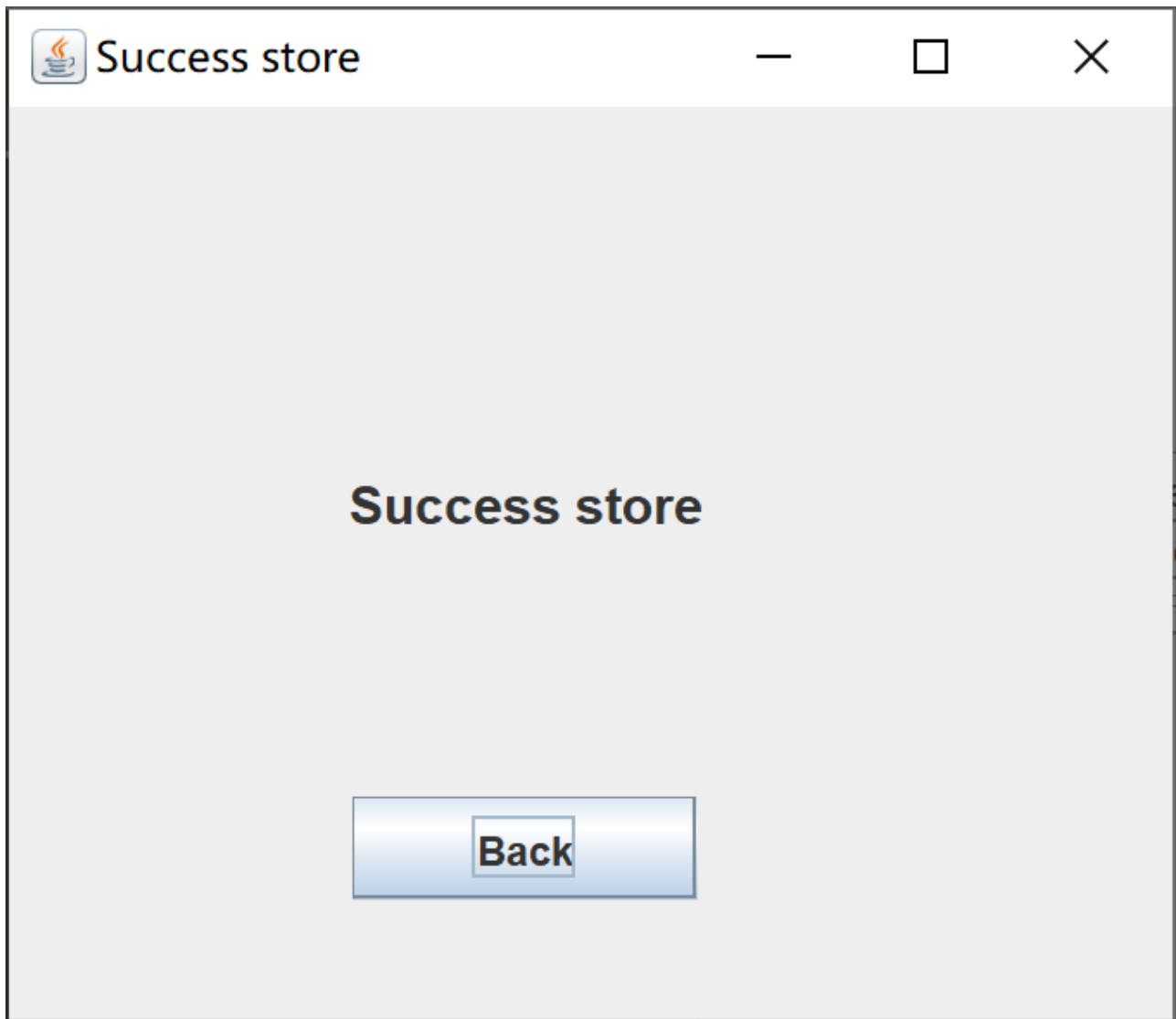
publishYear:

author:

price:

stock:

提交



5.3.4.3 修改库存

调用 `*incBookStock*` 接口进行库存修改，界面布局与上述类似。



修改库存



bookID:

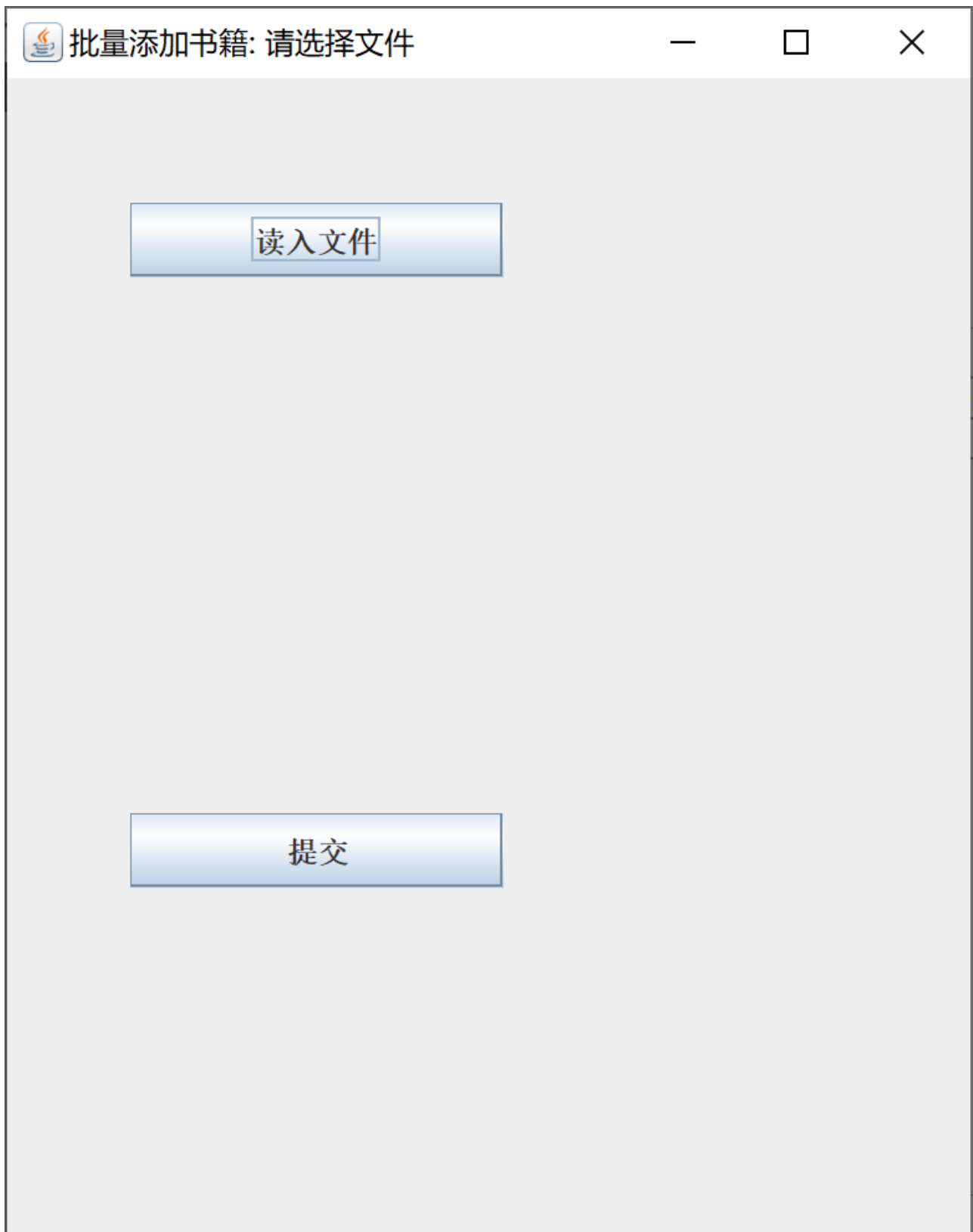
deltaStock:

提交

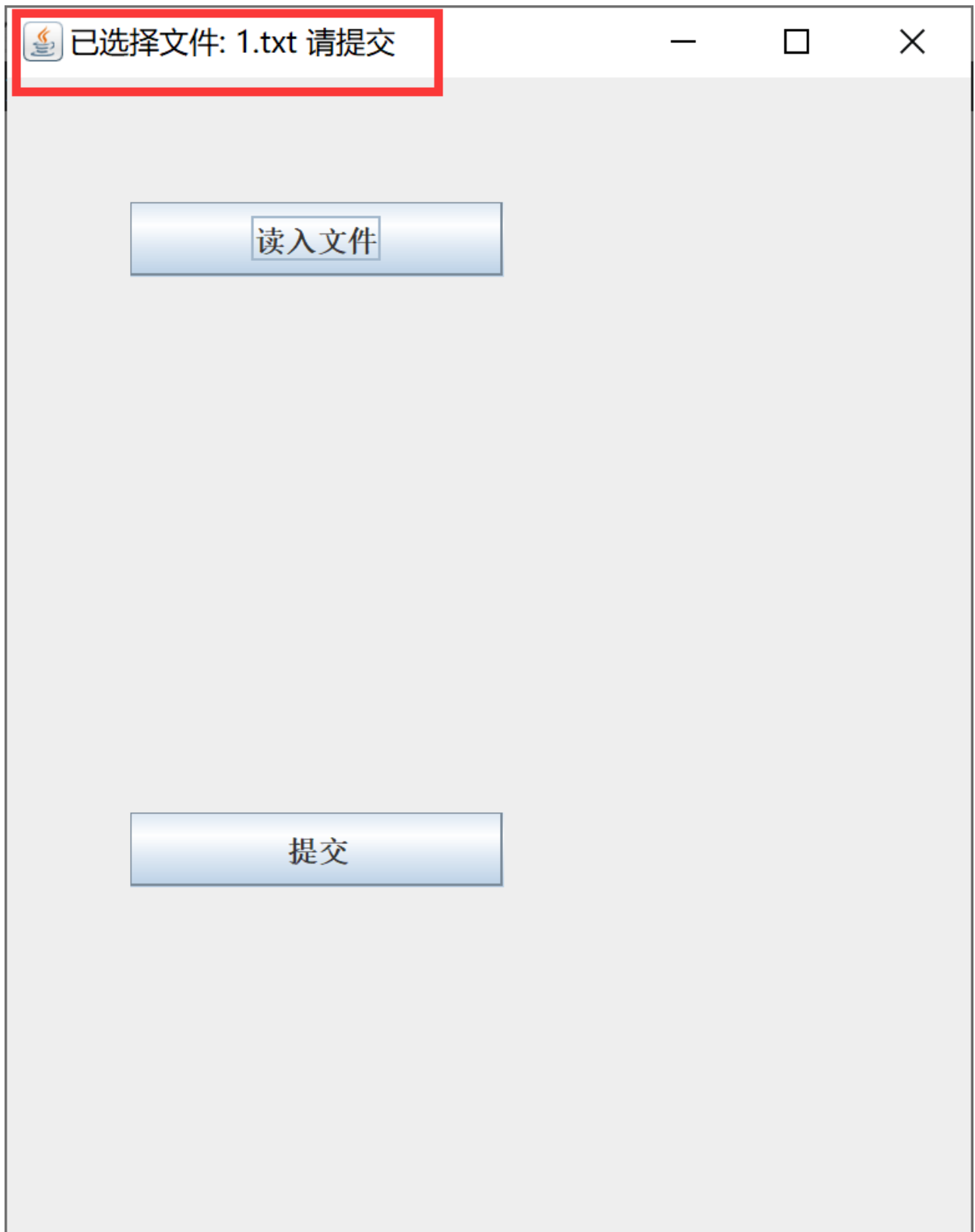
5.3.4.4 批量添加书籍

调用 `storeBook(List<Book> books)` 接口进行库存修改，界面布局与上述类似。

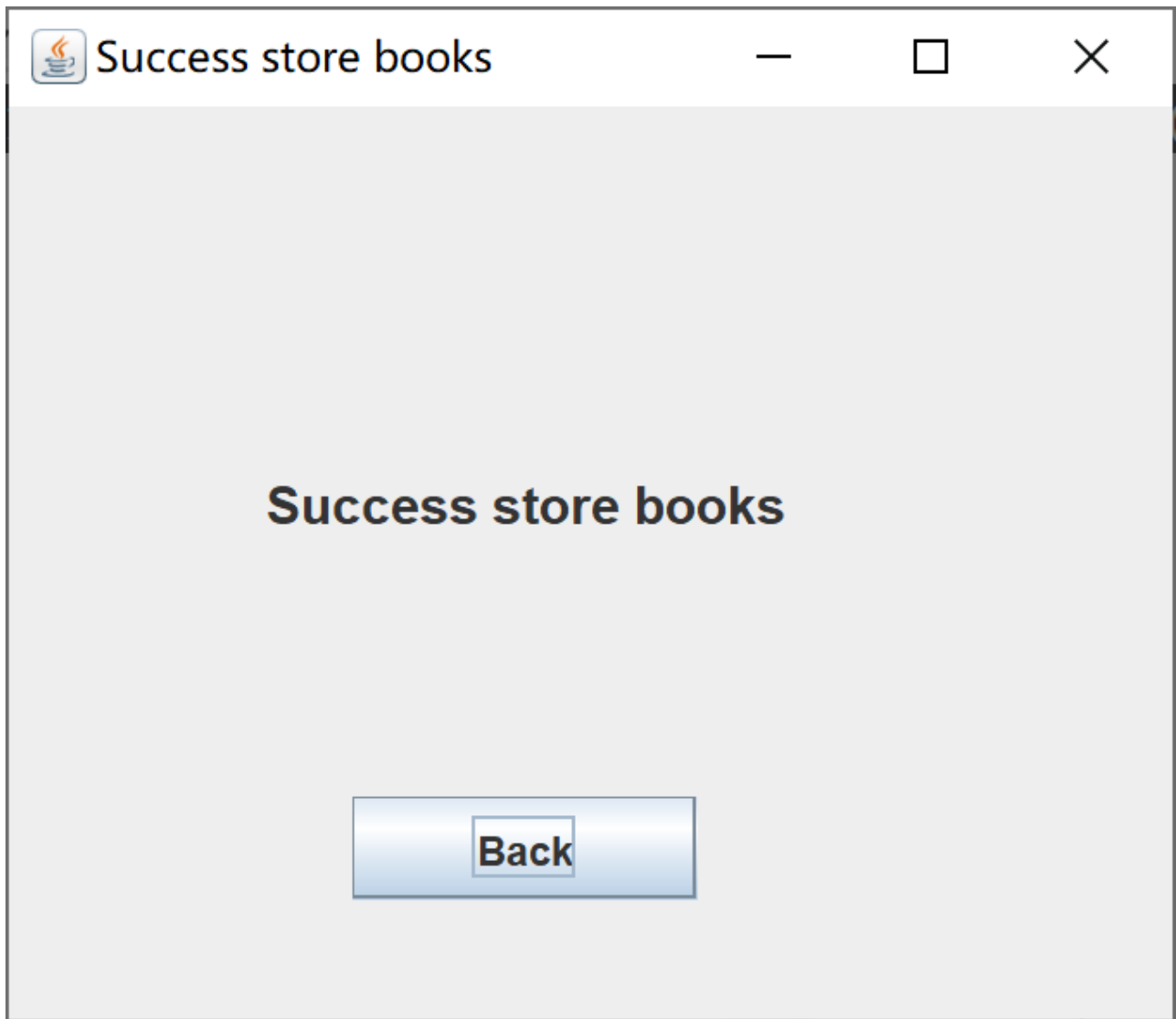
可以选择读入文件：



若成功读入会修改窗口标题：



提交成功:



5.3.4.5 删除书籍

调用 `removeBook` 接口进行库存修改，界面布局与上述类似。



删除图书



bookID:

提交

5.3.4.6 修改书籍

调用 `modifyBookInfo` 接口进行库存修改，界面布局与上述类似。



修改书籍



bookID:

category:

title:

press:

publishYear:

author:

price:

stock:

提交

5.3.4.7 查询书籍

调用 `queryBook` 接口进行库存查询，界面布局与上述类似。



查询书籍



category:

title:

press:

MinPublishY...

MaxPublish...

author:

MinPrice:

MaxPrice:

Sort By:



Sort Order:



提交

查询结束后弹出的窗口会展示查询结果：

查询成功

BookId	Category	Title	Press	Publish year	Author	Price	Stock
2	儿童图书	小胖历险记	pressB	2023	wwj	123.0	21
4	4	4	4	4	4	4.0	4
5	6	6	6	6	6	6.0	6
6	21	2133	2123	123	13	12.0	132
7	wer	31	132	123	123	13.0	123

5.3.5 借书证管理

5.3.5.1 界面表格

与上一部分类似，调用 `ApiResponse showCards()` 进行展示



5.3.5.2 注册借书卡

调用 `registerCard` 接口进行操作



注册借书卡



name:

王伟杰

department:

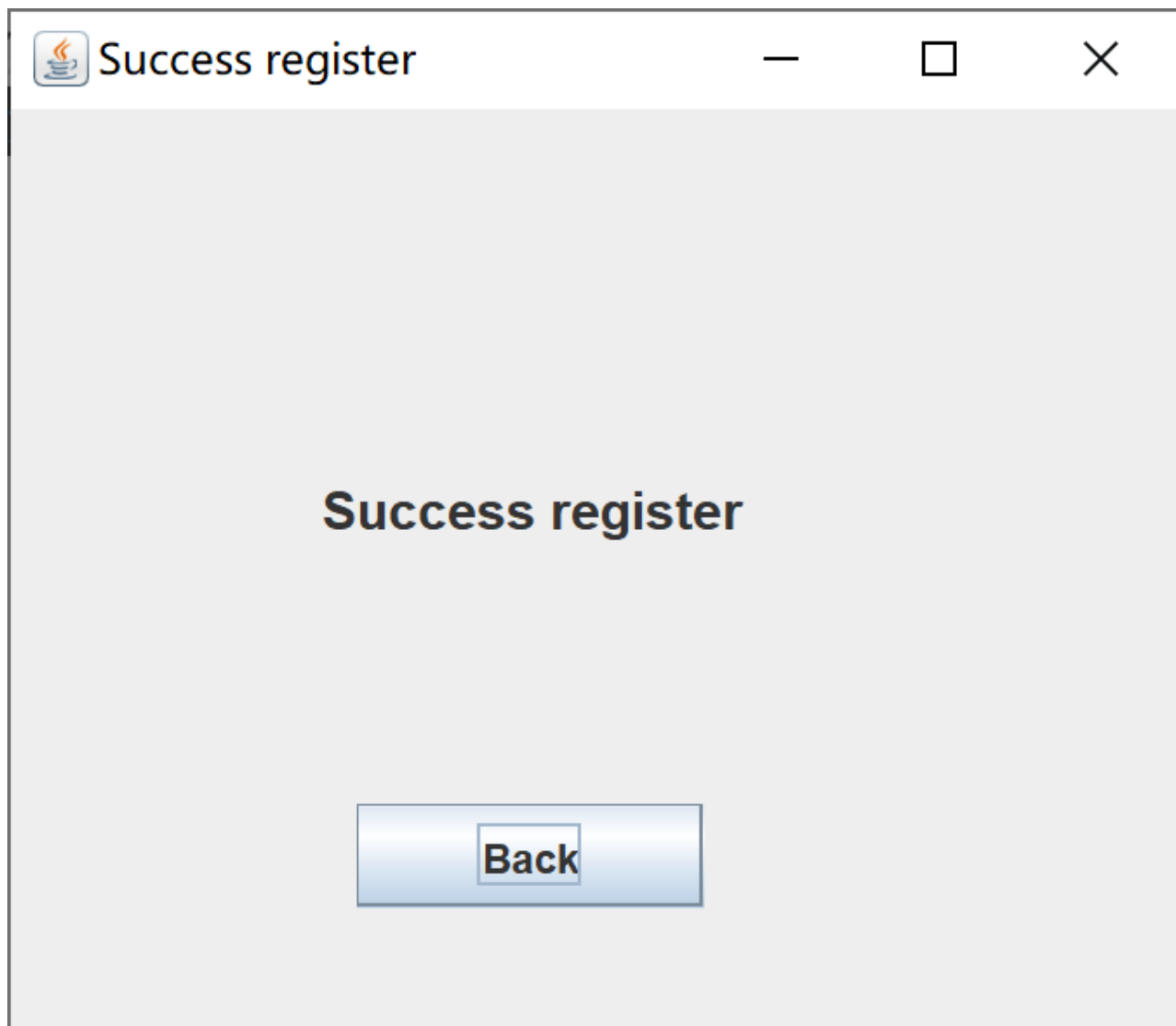
computer science

type:

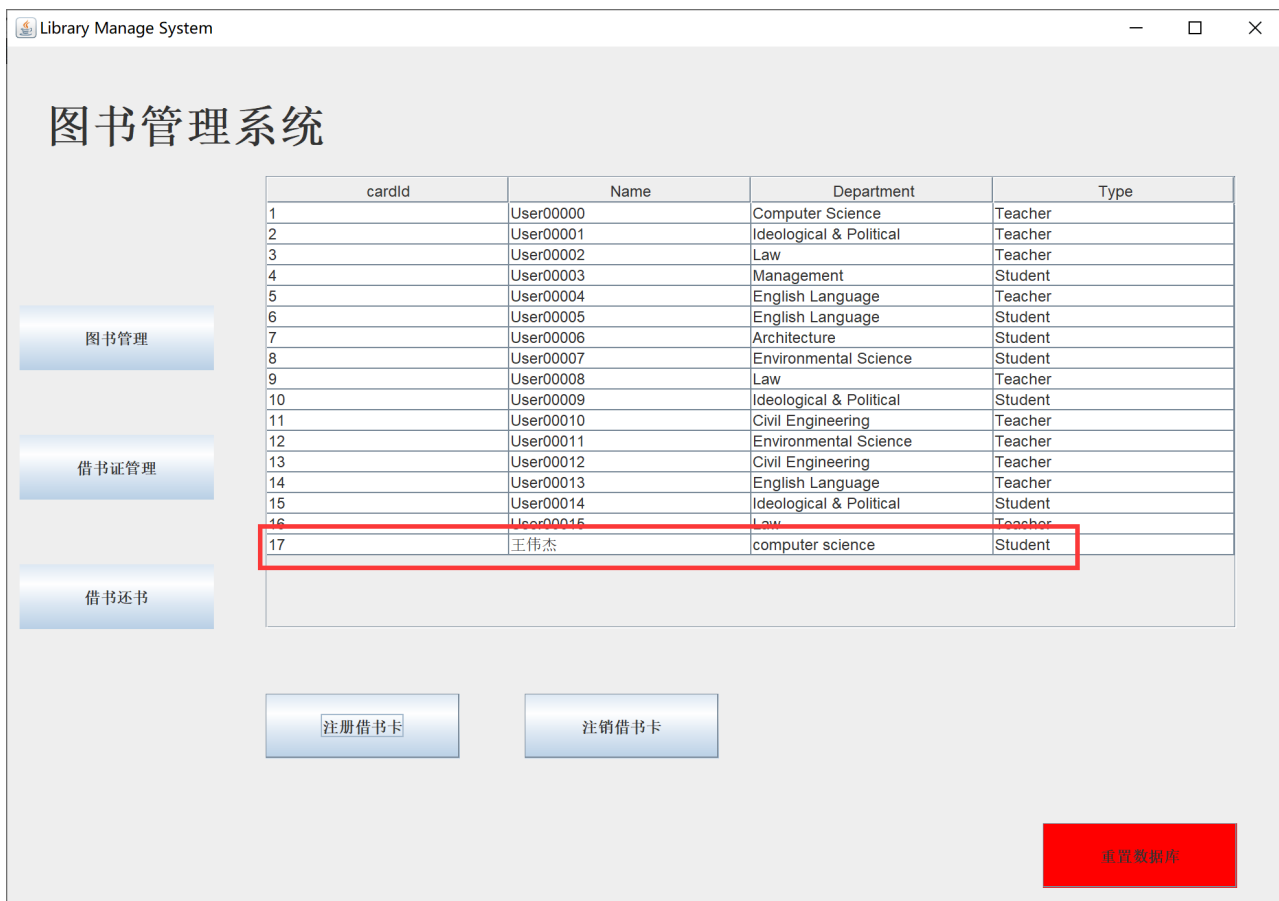
Student



提交



实时更新:



5.3.5.3 注销借书卡

调用 `removeCard` 接口进行操作



注销借书卡



cardID:

提交

5.3.6 借书还书

5.3.6.1 界面表格

与上一部分类似，调用 `ApiResponse showBorrows()` 进行展示



5.3.6.2 借书

调用 `borrowBook` 接口进行操作



借书



bookID:

cardID:

提交

成功更新：

cardId	BookId	Category	Title	Press	Publish year	Author	Price	borrowTime	returnTime
17	2	儿童图书	小胖历险记	pressB	2023	wwj	123.0	16825837...	0
2	1	Computer ...	Compiler ...	Press-A	2003	Authentic	202.07	16825723...	0

5.3.6.3 还书

调用 `returnBook` 接口进行操作



还书



bookID:

2

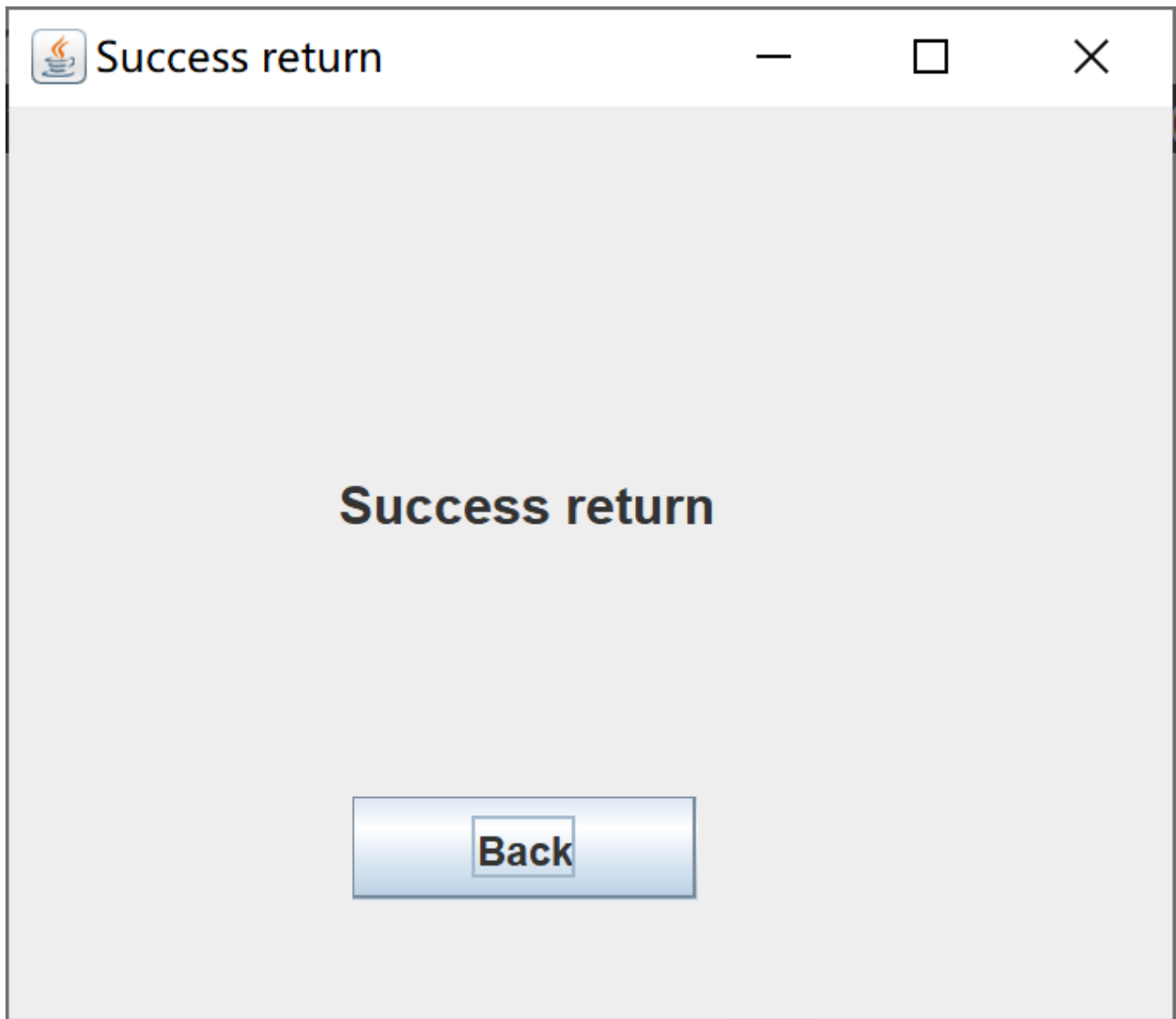
cardID:

17

Borrow Time:

1682583737000

提交



5.3.6.4 查询借书历史

调用 `showBorrowHistory` 接口进行操作，查询特定用户的借书记录



查询借书历史



cardID:

提交

结果在新窗口中展示：

查询成功


cardId	bookId	Category	Title	Press	Publish year	Author	Price	borrowTime	returnTime
17	2	儿童图书	小胖历险记	pressB	2023	wwj	123.0	1682583737000	1682583886504

6 系统测试

6.1 正确性测试

运行 `LibraryTest` 文件进行正确性测试：

运行所有的测试

 `mvn -Dtest=LibraryTest clean test`

测试全部通过：

运行 LibraryTest

测试 已通过: 9 共 9 个测试 - 11秒 204毫秒

LibraryTest 11秒 204毫秒

borrowAndReturnB 7秒 25毫秒

bulkRegisterBookTest 813毫秒

modifyBookTest 503毫秒

bookRegisterTest 161毫秒

incBookStockTest 1秒 378毫秒

queryBookTest 593毫秒

registerAndShowAndF 246毫秒

removeBookTest 168毫秒

parallelBorrowBookTe 317毫秒

D:\Java\bin\java.exe ...

Successfully init class BookTest.

Successfully connect to database.

Successfully reset database.

Successfully release database connection.

Successfully init class BookTest.

Successfully connect to database.

Successfully reset database.

Successfully release database connection.

Successfully init class BookTest.

Successfully connect to database.

6.2 功能性测试

运行 `Main` 文件代码，程序可以正常运行：



测试所有功能及界面展示、界面切换，均成功运行。



Library Manage System

—

□

×

图书管理系统

图书管理

借书证管理

借书还书

cardId	Name	Department	Type
1	User00000	English Language	Student
2	User00001	Ideological & Political	Student
3	User00002	Management	Student
4	User00003	English Language	Teacher
5	User00004	Computer Science	Student
6	User00005	Environmental Science	Teacher
7	User00006	General Education	Teacher
8	User00007	Computer Science	Teacher
9	User00008	Management	Teacher
10	User00009	English Language	Teacher
11	User00010	Law	Teacher
12	User00011	English Language	Teacher
13	User00012	Management	Teacher
14	User00013	Computer Science	Student
15	User00014	Civil Engineering	Teacher
16	User00015	Computer Science	Teacher

注册借书卡

注销借书卡

重置数据库

Library Manage System

—

□

×

图书管理系统

图书管理

借书证管理

借书还书

cardId	BookId	Category	Title	Press	Publish year	Author	Price	borrowTime	returnTime
6	1	Magazine	C++ Primer	Press-E	2002	ColaOtaku	68.37	16825844...	0

借书

还书

查询借书历史

重置数据库

7 遇到的问题及解决方法

7.1 环境配置

- 安装完Java，没有正确配置环境变量，导致无法运行，后面参照网上的教程正确配置。
- 在编写 `LibraryManagementImpl` 时发现程序无法运行，后发现是 `application.yaml` 没有正确配置，导致无法连接到数据库。

7.2 SQL语句表达

- 在编写代码时遇到很多SQL语句的构建较为复杂，需要对Java较为熟悉，为此我学习了很多Java相关接口的调用，成功解决了所有问题。

7.3 图形界面选择

- 最开始我想尝试将图形界面部署到网页上，为此学习了一些Java Spring的相关知识，但是后面发现框架迁移较麻烦，所以选择了Java Swing进行图形界面的编写。

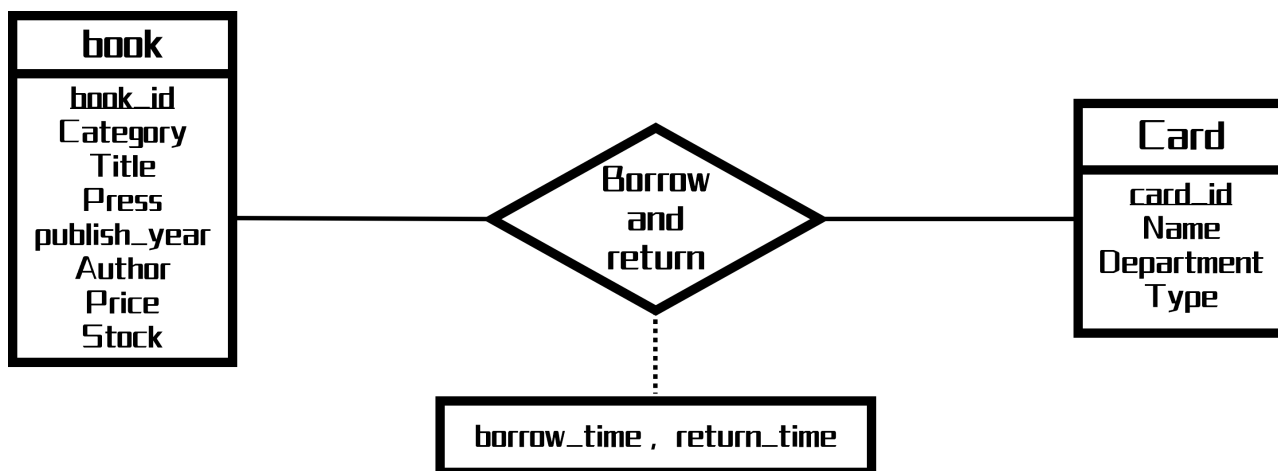
7.4 `Main` 函数无法运行

`pom.xml` 少了包，加入以下代码后执行一次 `mvn install clean`

```
1 <plugin>
2   <groupId>org.codehaus.mojo</groupId>
3   <artifactId>exec-maven-plugin</artifactId>
4   <version>3.1.0</version>
5   <configuration>
6     <mainClass>Main</mainClass>
7   </configuration>
8 </plugin>
```

8 思考题

8.1 绘制该图书管理系统的E-R图。



8.2 描述SQL注入攻击的原理。本系统哪些模块可能会遭受SQL注入攻击？如何解决？

SQL注入攻击是一种常见的网络攻击，攻击者通过在应用程序中注入恶意的SQL代码来执行未经授权的操作或获取敏感信息。攻击者通常会利用应用程序中的输入验证不足或错误来注入恶意代码。

例如，假设一个网站有一个搜索功能，用户可以在搜索框中输入关键字来搜索相关内容。如果网站没有对用户输入进行充分的验证和过滤，攻击者可以在搜索框中输入恶意的SQL代码，例如：

```
1 | ' OR 1=1; --
```

这个代码片段会将搜索语句修改为“SELECT * FROM book WHERE title = ' OR 1=1; --'”，后面的所有内容都被视为注释，因此攻击者可以绕过原始查询并获取所有书籍的信息。

在图书管理系统中，以下模块可能会遭受SQL注入攻击：

1. 添加/删除图书模块：攻击者可以注入恶意代码来修改或删除图书信息。
2. 图书查询模块：攻击者可以在查询时恶意删除或者修改图书信息。

为了防止SQL注入攻击，可以采取以下措施：

1. 对所有用户输入进行充分的验证和过滤，包括对特殊字符进行转义或删除。
2. 使用参数化查询或存储过程来执行SQL查询，而不是将用户输入直接拼接到查询语句中。
3. 限制数据库用户的权限，确保他们只能执行必要的操作。

8.3 在InnoDB的默认隔离级别(RR, Repeated Read)下，当出现并发访问时，如何保证借书结果的正确性？

1. 行级锁定：InnoDB使用行级锁定来避免并发事务之间的冲突。当一个事务正在借书时，它会锁定相关的行，防止其他事务修改或删除这些行。这样可以确保每个事务只能借阅可用的书籍，并且不会出现重复借阅或借阅不可用的书籍的情况。
2. 事务隔离：在RR隔离级别下，InnoDB使用多版本并发控制(MVCC)来实现事务隔离。每个事务都可以看到自己的“快照”，并且不会受到其他事务的影响。这样可以确保每个事务都能够正确地读取和修改数据，而不会受到其他事务的干扰。
3. 回滚段：InnoDB使用回滚段来存储事务的历史版本。如果一个事务需要回滚，它可以使用回滚段中的历史版本来还原数据。这样可以确保每个事务都能够正确地回滚，并且不会影响其他事务的执行。