浙江大学

数据库系统实验报告

作业名称：	图书管理系统
姓　　名：	李谞远
学　　号：	3220102970
电子邮箱：	497410282@qq.com
联系电话：	15268453576
指导老师：	苗晓晔

2024 年 3 月 30 日

图书管理系统

一、 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。

二、 系统需求

基本数据对象：

所有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，其它信息请参考类内的注释。

书：类名为 Book，属性有书号，类别，书名，出版社，年份，作者，价格，剩余库存。

借书证：类名为 Card，属性有卡号，姓名，单位，身份(教师或学生)。

借书记录：类名为 Borrow，属性有卡号，书号，借书日期，还书日期。

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，以下是 LibraryManagementSystem 中声明的模块，有关各模块的详细注释和说明已经在接口 LibraryManagementSystem 中给出，请自行阅读：

● 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。请注意，book_id 作为自增列，应该插入时由数据库生成。插入完成后，需要根据数据库生成的 book_id 值去更新 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 modifyCardInfo(Card card)：借书证修改模块。修改已存在的借书证的基本信息，该接口不能修改借书证的卡号。

● ApiResult showCards()：借书证查询模块。列出所有的借书证。

数据库（表）设计：

以下是该图书管理系统的数据表定义：

```
create table `book` (
    `book_id` int not null auto_increment,
    `category` varchar(63) not null,
    `title` varchar(63) not null,
    `press` varchar(63) not null,
    `publish_year` int not null,
    `author` varchar(63) not null,
    `price` decimal(7, 2) not null default 0.00,
    `stock` int not null default 0,
    primary key (`book_id`),
    unique (`category`, `press`, `author`, `title`, `publish_year`)
);
create table `card` (
    `card_id` int not null auto_increment,
    `name` varchar(63) not null,
    `department` varchar(63) not null,
    `type` char(1) not null,
    primary key (`card_id`),
    unique (`department`, `type`, `name`),
    check ( `type` in ('T', 'S') )
);
create table `borrow` (
    `card_id` int not null,
    `book_id` int not null,
    `borrow_time` bigint not null,
    `return_time` bigint not null default 0,
    primary key (`card_id`, `book_id`, `borrow_time`),
    foreign key (`card_id`) references `card`(`card_id`) on delete
cascade on update cascade,
    foreign key (`book_id`) references `book`(`book_id`) on delete
cascade on update cascade
);
```

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随机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
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)，价格范围差>中随机选取 N 个条件，并随机选取一个排序列和顺序

三、 实验环境

操作系统：Windows 11

开发环境：VSCode + Java + Maven + Vue.js

数据库系统：MySQL Server 8.3.0

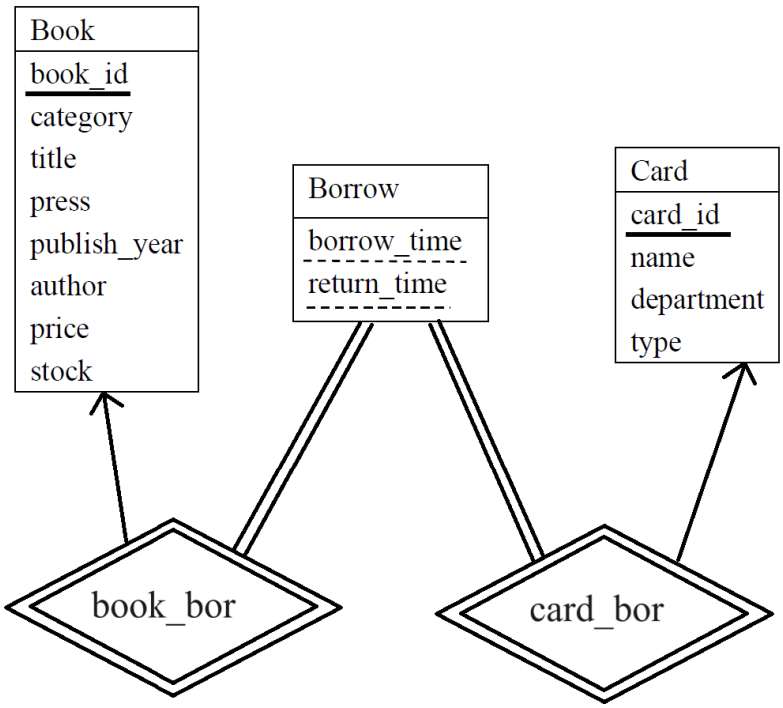
Java 和 Maven 版本截图如下：

```
C:\Users\ycyaw>java -version
java version "21.0.1" 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)

C:\Users\ycyaw>mvn -v
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)
Maven home: D:\Maven\apache-maven-3.9.6
Java version: 21.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: zh_CN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

四、 系统设计及实现

1. 绘制该图书管理系统的 E-R 图（思考题 1）



2. 系统各函数的设计思路和实现

● ApiResult storeBook(Book book)：图书入库模块。

设计思路：先用 SELECT 语句在入库前查询是否有相同图书，如果没有则用 INSERT 语句往 book 表中插入该图书，插入后获取自增列 book_id。

代码实现：

```
1.  public ApiResult storeBook(Book book) {
2.      Connection conn = connector.getConn();
3.      PreparedStatement pStmt = null;
4.      ResultSet rSet = null;
5.      try {
6.          String category = book.getCategory();
7.          String title = book.getTitle();
8.          String press = book.getPress();
9.          int publishYear = book.getPublishYear();
10.         String author = book.getAuthor();
11.         double price = book.getPrice();
12.         int stock = book.getStock();
13.
14.         /* check if there are same books */
15.         String sameBookCheck = "SELECT * FROM book WHERE category = ? AND ti
tle = ? AND press = ? AND publish_year = ? AND author = ?";
16.
17.         pStmt = conn.prepareStatement(sameBookCheck);
```

```
18.         pStmt.setString(1, category);
19.         pStmt.setString(2, title);
20.         pStmt.setString(3, press);
21.         pStmt.setInt(4, publishYear);
22.         pStmt.setString(5, author);
23.         rSet = pStmt.executeQuery();
24.         if (rSet.next()) {
25.             return new ApiResult(false, "图书添加失败：存在相同图书");
26.         }
27.
28.         String storeBookQuery = "INSERT INTO book (category, title, press, p
ublish_year, author, price, stock) VALUES (?, ?, ?, ?, ?, ?, ?)";
29.
30.         pStmt = conn.prepareStatement(storeBookQuery, PreparedStatement.RETU
RN_GENERATED_KEYS);
31.         pStmt.setString(1, category);
32.         pStmt.setString(2, title);
33.         pStmt.setString(3, press);
34.         pStmt.setInt(4, publishYear);
35.         pStmt.setString(5, author);
36.         pStmt.setDouble(6, price);
37.         pStmt.setInt(7, stock);
38.         pStmt.executeUpdate();
39.
40.         rSet = pStmt.getGeneratedKeys();
41.         if (rSet.next()) {
42.             int bookId = rSet.getInt(1);
43.             book.setBookId(bookId);
44.         }
45.
46.         commit(conn);
47.     } catch (Exception e) {
48.         rollback(conn);
49.         return new ApiResult(false, e.getMessage());
50.     } finally {
51.         try {
52.             if (rSet != null) {
53.                 rSet.close();
54.             }
55.             if (pStmt != null) {
56.                 pStmt.close();
57.             }
58.         } catch (SQLException e) {
59.             e.printStackTrace();
```

```
        }
    }
    return new ApiResult(true, "图书添加成功");
}
```

● ApiResult incBookStock(int bookId, int deltaStock)：图书增加库存模块。

设计思路：先用 SELECT 语句查询图书是否存在，然后获取图书库存，看修改后的库存是否会为负，如果条件都满足，则用 UPDATE 语句更新图书库存。

代码实现：

```
public ApiResult incBookStock(int bookId, int deltaStock) {
    Connection conn = connector.getConn();
    PreparedStatement pStmt = null;
    ResultSet rSet = null;
    try {
        String selectStockQuery = "SELECT stock FROM book WHERE book_id = ?"
;
        pStmt = conn.prepareStatement(selectStockQuery);
        pStmt.setInt(1, bookId);
        rSet = pStmt.executeQuery();

        int currentStock = 0;
        if (rSet.next()) {
            currentStock = rSet.getInt("stock");
        }
        else {
            return new ApiResult(false, "库存修改失败：图书不存在");
        }

        if (currentStock + deltaStock < 0) {
            return new ApiResult(false, "库存修改失败：库存为负");
        }

        String incBookStockQuery = "UPDATE book SET stock = stock + ? WHERE
book_id = ?";

        pStmt = conn.prepareStatement(incBookStockQuery);
        pStmt.setInt(1, deltaStock);
        pStmt.setInt(2, bookId);
        pStmt.executeUpdate();

        commit(conn);
    } catch (Exception e) {
        rollback(conn);
        return new ApiResult(false, e.getMessage());
```

```
    } finally {
        try {
            if (rSet != null) {
                rSet.close();
            }
            if (pStmt != null) {
                pStmt.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return new ApiResult(true, "库存修改成功");
}
```

- ApiResult storeBook(List<Book> books)：
- ApiResult removeBook(int bookId)：
- ApiResult modifyBookInfo(Book book)：
- ApiResult queryBook(BookQueryConditions conditions)：
- ApiResult borrowBook(Borrow borrow)：
- ApiResult returnBook(Borrow borrow)：
- ApiResult showBorrowHistory(int cardId)：
- ApiResult registerCard(Card card)：
- ApiResult removeCard(int cardId)：
- ApiResult modifyCardInfo(Card card)：
- ApiResult showCards()：

3. 程序运行结果场景以及截图说明（即实验指导文档中的系统功能验证）
   a. 图书入库：

   b. 增加库存：
   c. 修改图书信息：
   d. 添加借书证：
   e. 查询借书证：
   f. 借书：
   g. 还书：
   h. 借书记录查询
   i. 图书查询：

五、 遇到的问题及解决方法

1. 描述 SQL 注入攻击的原理，并简要举例。在图书管理系统中，哪些模块可能会遭受 SQL 注入攻击？如何解决？（思考题 2）
2. 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？（思考题 3）

六、 总结