

基于MySQL的精简图书管 理系统

叶容宇 3230101885

2025 年 4 月 14 日



目录	2
----	---

目录

1 实验目的	3
2 实验环境	3
3 项目文件结构	3
4 文件作用说明	4
4.1 config/CorsConfig.java	4
4.2 src/main/java/library/system/LibraryManagementSystem.java	5
4.3 src/main/java/library/system/LibraryManagementSystemImpl.java . . .	6
4.4 src/main/java/library/system/controller/BookController.java	9
5 实验结果	10
5.1 测试用例验证	10
5.2 前后端交互	11
5.2.1 图书查询出借注册	11
5.2.2 借书卡管理	11
5.2.3 借阅历史	12
6 思考题	13

1 实验目的

本实验旨在设计并实现一个图书管理系统的数据库，该系统能够有效地管理图书、借阅者卡片以及借阅记录，并通过一系列的测试用例来验证系统的正确性和性能。最终目的是提供一个基于MySQL的精简图书管理程序，该图书管理程序应具备较好的可扩展性、鲁棒性和安全性，并且在高并发场景下仍能正确运行。

2 实验环境

本实验基于MySQL数据库，使用Java语言进行编程。

前端使用Vue.js，后端使用Spring Boot框架。

在Windows环境下

启动后端

```
mvn spring-boot:run
```

启动前端（在librarymanagementsystem-frontend目录下）

```
npm run dev
```

运行所有的测试

```
mvn -Dtest=LibraryTest clean test
```

3 项目文件结构

本项目采用Maven项目结构，主要包含以下几个模块：

- `src/main/java`: 包含Java源代码。
- `src/main/resources`: 包含配置文件，如数据库连接信息。
- `src/test/java`: 包含测试代码。
- `librarymanagementsystem-frontend`: 前端代码。
- `report`: 实验报告及相关文档。

在 `src/main/java` 目录下，主要模块如下：

- `config`: 配置文件，如跨域配置。
- `entities`: 实体类，如书籍、借阅者卡片等。

- `queries`: 查询条件和结果类。
- `library.system`: 核心系统类和接口。
- `library.system.controller`: 控制器层，负责处理前端请求。
- `utils`: 工具类。

4 文件作用说明

4.1 `config/CorsConfig.java`

文件定义了跨域资源共享（CORS）的配置，以允许前端应用从不同的源访问API，确保前后端协同工作时不会遇到跨域问题。在实验中最初发现前后端不能完成正常通信。前端发送的请求会被后端404拒绝，原因是没有配置spring boot的跨域规则。

功能说明：

- 使用`@Configuration`注解标记为配置类。
- 通过`@Bean`注解定义了一个`CorsFilter` Bean，用于处理跨域请求。
- 配置允许所有来源（"*"）、所有请求头（"*"）和所有HTTP方法（"*"），并允许凭据（`setAllowCredentials(true)`）。

代码关键点：

- `CorsConfiguration`用于定义跨域规则。
- `UrlBasedCorsConfigurationSource`用于将跨域规则绑定到特定的URL路径（"*"表示所有路径）。
- `CorsFilter`用于实际拦截请求并应用跨域规则。

设计思路：

- 确保前后端分离时，前端可以无限制地访问后端API。
- 在开发和测试阶段，允许所有来源的访问以简化调试。

Listing 1: config/CorsConfig.java

```
1 package config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.web.cors.CorsConfiguration;
6 import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
7 import org.springframework.web.filter.CorsFilter;
8
9 @Configuration
10 public class CorsConfig {
11     private static final long MAX_AGE = 24 * 60 * 60; // 1 day
12
13     @Bean
14     public CorsFilter corsFilter() {
15         UrlBasedCorsConfigurationSource source = new
16             UrlBasedCorsConfigurationSource();
17         CorsConfiguration config = new CorsConfiguration();
18         config.addAllowedHeader("*");
19         config.addAllowedMethod("*");
20         config.addAllowedOrigin("http://localhost:5173");
21         config.setMaxAge(MAX_AGE);
22         source.registerCorsConfiguration("/**", config);
23         return new CorsFilter(source);
24     }
25 }
```

4.2 src/main/java/library/system/LibraryManagementSystem.java

文件定义了图书管理系统的接口，声明了系统需要实现的各种功能，如注册图书、借书、还书等。通过PreparedStatement预编译SQL语句，避免SQL注入攻击。使用stmt.executeUpdate()执行SQL语句，与数据库交互。功能说明：

- 定义了系统的核心功能接口，包括：
 - storeBook(Book book)：注册新图书。
 - borrowBook(Borrow borrow)：借书。
 - returnBook(Borrow borrow)：还书。

- `storeBook(List<Book> books)`: 批量注册图书。
- `queryBook(BookQueryConditions conditions)`: 查询图书。

代码关键点:

- 使用`ApiResponse`作为返回值, 统一处理成功和失败的响应。
- `BookQueryResults`用于封装查询结果, 包含状态、消息和图书列表。

设计思路:

- 通过接口定义系统功能, 便于后续实现和扩展。
- 使用统一的返回类型 (`ApiResponse`和`BookQueryResults`) 简化前端处理逻辑。

Listing 2: `library.system/LibraryManagementSystem.java`

```
1 public interface LibraryManagementSystem {  
2  
3     ApiResponse storeBook(Book book);  
4     ApiResponse incBookStock(int bookId, int deltaStock);  
5     ApiResponse storeBook(List<Book> books);  
6     ApiResponse removeBook(int bookId);  
7     ApiResponse queryBook(BookQueryConditions conditions);  
8     ApiResponse borrowBook(Borrow borrow);  
9     ApiResponse returnBook(Borrow borrow);  
10    ApiResponse showBorrowHistory(int cardId);  
11    ApiResponse registerCard(Card card);  
12    ApiResponse removeCard(int cardId);  
13    ApiResponse showCards();  
14    ApiResponse modifyCardInfo(Card card);  
15 }
```

4.3 `src/main/java/library/system/LibraryManagementSystemImpl.java`

文件实现了`LibraryManagementSystem`接口, 提供了具体的功能实现。在进行并行请求测试的时候使用`FOR UPDATE`锁定记录, 避免数据不一致的问题。

功能说明:

- 实现了`LibraryManagementSystem`接口的所有方法。

- 完成了图书注册、图书批量注册、并发借阅、借阅和还书等功能。

设计思路：

- 使用PreparedStatement预编译SQL语句，避免SQL注入攻击。
- 使用统一的返回类型

Listing 3: library.system/LibraryManagementSystemImpl.java

```
1 import org.springframework.stereotype.Component;
2 import org.springframework.stereotype.Service;
3
4 @Component
5 @Service
6 public class LibraryManagementSystemImpl implements LibraryManagementSystem
7 {
8     private final DatabaseConnector connector;
9
10    public LibraryManagementSystemImpl(DatabaseConnector connector) {
11        this.connector = connector;
12    }
13
14    @Override
15    public ApiResult storeBook(Book book) {
16        Connection conn = connector.getConn();
17        try {
18            PreparedStatement checkStmt = conn.prepareStatement(
19                "SELECT COUNT(*) FROM book WHERE category = ? AND title
20                = ? AND press = ? AND author = ? AND publish_year =
21                ?");
22
23            checkStmt.setString(1, book.getCategory());
24            checkStmt.setString(2, book.getTitle());
25            checkStmt.setString(3, book.getPress());
26            checkStmt.setString(4, book.getAuthor());
27            checkStmt.setInt(5, book.getPublishYear());
28
29            ResultSet rs = checkStmt.executeQuery();
30            rs.next();
31            int count = rs.getInt(1);
```

```
29
30     if (count > 0) {
31         return new ApiResult(false, "Book already exists in the
32             library system.");
33     }
34
35     PreparedStatement insertStmt = conn
36         .prepareStatement(
37         "INSERT INTO book (category, title, press,
38             publish_year, author, price, stock) VALUES
39             (?, ?, ?, ?, ?, ?, ?)",
40             Statement.RETURN_GENERATED_KEYS);
41     insertStmt.setString(1, book.getCategory());
42     insertStmt.setString(2, book.getTitle());
43     insertStmt.setString(3, book.getPress());
44     insertStmt.setInt(4, book.getPublishYear());
45     insertStmt.setString(5, book.getAuthor());
46     insertStmt.setDouble(6, book.getPrice());
47     insertStmt.setInt(7, book.getStock());
48
49     int affectedRows = insertStmt.executeUpdate();
50     if (affectedRows == 1) {
51         ResultSet generatedKeys = insertStmt.getGeneratedKeys();
52         if (generatedKeys.next()) {
53             book.setBookId(generatedKeys.getInt(1));
54         }
55         commit(conn);
56         return new ApiResult(true, "Book stored successfully");
57     } else {
58         rollback(conn);
59         return new ApiResult(false, "Failed to store book");
60     }
61
62 } catch (Exception e) {
63     rollback(conn);
64     return new ApiResult(false, e.getMessage());
65 }
```


4.4 src/main/java/library/system/controller/BookController.java

文件是控制器层的一部分，处理与图书相关的HTTP请求，如注册新图书、查询图书等。

功能说明：

- 使用@Controller注解标记为控制器。
- 使用@RequestMapping("/api")定义了API的基路径。

设计思路：

- 通过@RequestBody和@RequestParam注解解析请求参数。
- 将业务逻辑委托给LibraryManagementSystem接口实现类，保持控制器的简洁性。

Listing 4: library.system.controller/BookController.java

```
1 import org.springframework.beans.factory.annotation.Autowired;
2 import org.springframework.web.bind.annotation.*;
3
4 @RestController
5 @CrossOrigin(origins = "http://localhost:5173")
6 @RequestMapping("/api")
7 public class BookController {
8     @Resource
9     private LibraryManagementSystem library;
10    private static final Logger log = Logger.getLogger(BookController.class
11        .getName());
12
13    @PostConstruct
14    public void init() {
15        try {
16            ConnectConfig conf = new ConnectConfig();
17            DatabaseConnector connector = new DatabaseConnector(conf);
18            boolean connStatus = connector.connect();
19            if (!connStatus) {
20                throw new SQLException("Failed to connect to database.");
21            }
22            this.library = new LibraryManagementSystemImpl(connector);
23        } catch (Exception e) {
24            throw new RuntimeException("Failed to initialize the controller
25                .", e);
26        }
27    }
28 }
```

```
24     }
25     resetDatabase();
26
27 }
28
29 public BookController() {
30     init();
31 }
32
33 @GetMapping("/book")
34 public BookQueryResults queryBooks(@ModelAttribute BookQueryConditions
35     queryConditions) {
36     try {
37         ApiResult result = library.queryBook(queryConditions);
38         if (!result.ok) {
39             throw new RuntimeException("Failed" + result.message);
40         }
41         return (BookQueryResults) result.payload;
42     } catch (Exception e) {
43         throw new RuntimeException("Failed to query books.", e);
44     }
45 }
```

5 实验结果

5.1 测试用例验证

最终的测试用例验证了系统的正确性和性能，成功通过了LibraryTest中的所有测试函数

```
1 [INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
   18.576 s - in LibraryTest
2 [INFO]
3 [INFO] Results:
4 [INFO]
5 [INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
6 [INFO]
```

5.2 前后端交互

5.2.1 图书查询出借注册

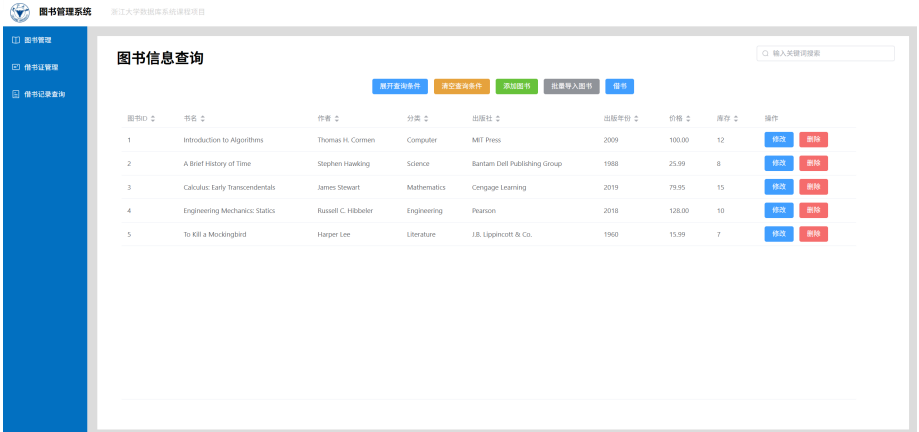


图 1: 图书查询界面

提供按条件查询按钮。提供导入单本或者批量导入的支持

5.2.2 借书卡管理

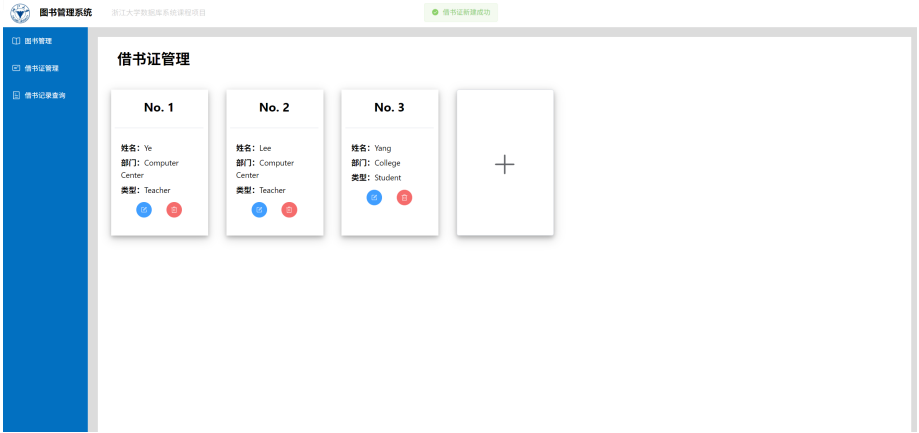


图 2: 借书卡管理界面

查询所有的借书卡，编辑、删除、新增借书卡。

5.2.3 借阅历史

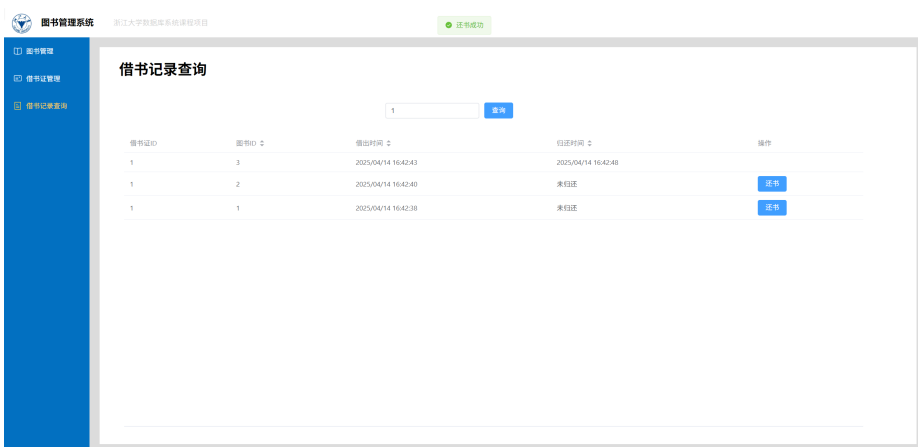
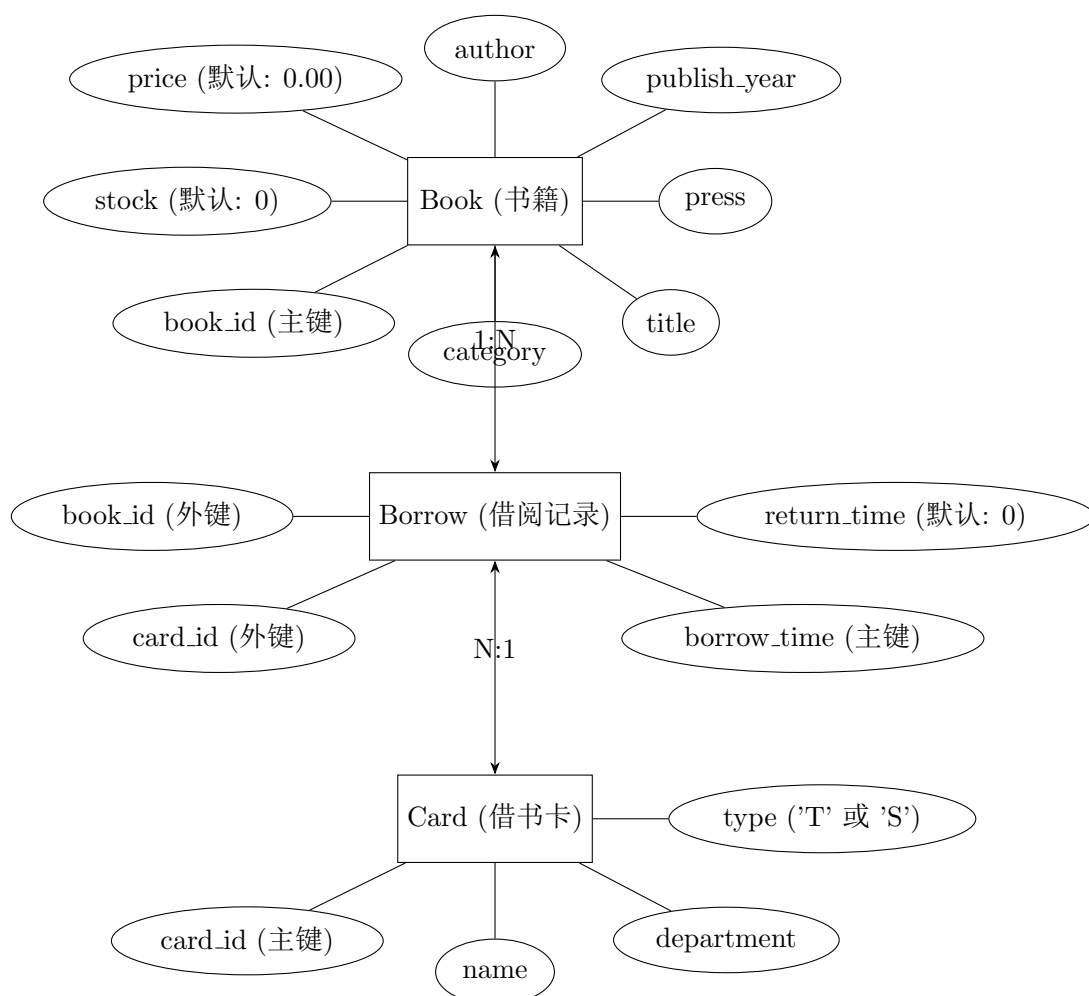


图 3: 借阅历史界面

可以根据借书卡号查找借阅历史，并且可以手动归还书籍。

6 思考题



图书管理系统中可能遭受SQL注入攻击的模块：

借阅模块：用户输入借阅信息时，如果输入验证不严格，可能会被注入恶意SQL代码。登录和注册模块：用户输入用户名和密码时，如果对输入的特殊字符处理不当，可能会被注入恶意SQL代码。搜索功能模块：用户输入搜索条件时，如果直接拼接到SQL语句中，可能会被注入恶意SQL代码。

解决方法：使用预编译语句（Prepared Statements）或参数化查询：通过预编译SQL语句，将用户输入的内容作为参数而不是直接拼接在SQL语句中，从而防止SQL注入。

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？

使用事务：对于借书操作，应将其放在事务中，确保借书、减少库存、记录借阅信息等操作要么全部成功，要么全部失败。锁机制：在事务中使用锁机制，比如SELECT ... FOR UPDATE，可以锁定正在被查询的记录，防止其他事务对这些记录进行修改，直到当前事务提交或回滚。