**数据库课程设计**

题 目 学校图书借阅管理系统

专业班级

姓 名

学 号

# 需求分析

## 数据结构

**book（书籍）数据结构：**

Book=书籍id（书号/ISBN）+书名+出版商+作者+分类+总量+余量+价格

* 数据项：书籍id（书号/ISBN）

含义说明：唯一标识每本书

类型：字符型

长度：255

取值范围：255位字符

* 数据项：书名

含义说明：书籍名称

类型：字符型

长度：255

取值范围：255位字符

* 数据项：出版商

含义说明：书籍出版商

类型：字符型

长度：255

* 数据项：作者

含义说明：书籍作者

类型：字符型

长度：255

* 数据项：分类

含义说明：书籍分类

类型：字符型

长度：255

* 数据项：总量

含义说明：书籍在册总数

类型：整型

长度：4

* 数据项：余量

含义说明：书籍在册数

类型：整型

长度：4

* 数据项：价格

含义说明：书籍价格

类型：double型

长度：10，小数点2位

**user（用户）数据结构：**

User=用户编号（身份证）+姓名+电话+地址

* 数据项：用户编号（身份证）

含义说明：用户的身份证，唯一对应

类型：字符型

长度：255

取值范围：255位字符

* 数据项：姓名

含义说明：用户姓名

类型：字符型

长度：255

取值范围：255位字符

* 数据项：电话

含义说明：用户的联系方式

类型：字符型

长度：11

* 数据项：地址

含义说明：用户的家庭住址

类型：字符型

长度：255

**card（借阅卡）数据结构：**

Card=借阅卡id（借阅卡登陆账户）+密码+对应用户+借阅总量+借阅余量+借阅卡头像

* 数据项：借阅卡id（借阅卡登陆账户）

含义说明：借阅卡，一个用户可以对应多张借阅卡

类型：字符型

长度：255

取值范围：255位字符

* 数据项：密码

含义说明：借阅卡登陆密码

类型：字符型

长度：255

取值范围：255位字符

* 数据项：对应用户

含义说明：外键，关联至用户id

类型：字符型

长度：255

* 数据项：借阅总量

含义说明：该借阅卡可借阅的总量

类型：整数型

长度：4

* 数据项：借阅余量

含义说明：该借阅卡可借阅的余量

类型：整数型

长度：4

* 数据项：借阅卡头像

含义说明：借阅卡的头像

类型：字符型

长度：255

**payment（缴费）数据结构：**

Payment=缴费id +缴费价格+是否支付+缴费时间+对应书籍+对应借阅记录+对应借阅卡

* 数据项：缴费id

含义说明：缴费条目id

类型：整数型，自增

长度：15

* 数据项：缴费价格

含义说明：需缴纳罚款值

类型：double型

长度：10，2位小数

* 数据项：是否支付

含义说明：标识本条目是否支付过

类型：tinyint型

长度：1位整数，标志位

* 数据项：缴费时间

含义说明：记录本条目缴费时间

类型：datetime型

* 数据项：对应书籍

含义说明：外键，对应书籍id

类型：字符型

长度：255

* 数据项：对应借阅记录

含义说明：标识本条目是否支付过

类型：tinyint型

长度：1位整数，标志位

**borrow（借阅）数据结构：**

Borrow=借阅id+对应书籍+对应借阅卡+借阅开始时间+借阅结束时间+实际归还时间+是否超期+是否续期+是否申请归还+是否批准归还+是否批准借阅

* 数据项：借阅id

含义说明：借阅条目id

类型：整数型，自增

长度：15

* 数据项：对应书籍

含义说明：对应书籍的id

类型：字符型

长度：255

* 数据项：对应借阅卡

含义说明：对应借阅卡的id

类型：字符型

长度：255

* 数据项：借阅开始时间

含义说明：该条借阅开始的时间

类型：datetime型

* 数据项：借阅结束时间

含义说明：该条借阅结束的时间

类型：datetime型

* 数据项：实际归还时间

含义说明：该条借阅实际归还的时间

类型：datetime型

* 数据项：是否超期

含义说明：该条借阅是否超期

类型：tinyint型

* 数据项：是否续期

含义说明：该条借阅是否续期

类型：tinyint型

* 数据项：是否申请归还

含义说明：该条借阅是否申请归还

类型：tinyint型

* 数据项：是否批准归还

含义说明：该条借阅是否批准归还

类型：tinyint型

* 数据项：是否批准借阅

含义说明：该条借阅是否批准借阅

类型：tinyint型

**admin（管理员）数据结构：**

Admin=管理员账号+密码

* 数据项：管理员账号

含义说明：管理员登陆账号

类型：字符型

长度：255

* 数据项：密码

含义说明：管理员账号密码

类型：字符型

长度：255

## 数据流

表1.数据流表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据流名 | 数据流来源 | 数据流去向 | 组成 |
| 图书借阅申请 | 借阅者 | 借阅表 | 借阅条目 |
| 批准/拒绝借阅 | 管理员 | 借阅表 | 是否批准借阅 |
| 图书续期 | 借阅者 | 借阅表 | 续期时间 |
| 图书归还申请 | 借阅者 | 借阅表 | 是否申请归还 |
| 批准归还 | 管理员 | 借阅表 | 是否批准归还 |
| 超期罚款 | 系统 | 借阅表，缴费表 | 是否超期，缴费条目 |
| 缴费 | 借阅者 | 缴费表 | 是否支付+支付时间 |
| 创建/修改用户 | 管理员/用户 | 用户表 | 用户条目 |
| 创建/修改借阅卡 | 管理员/用户 | 借阅卡表 | 借阅卡条目 |
| 创建/修改书籍 | 管理员 | 书籍表 | 书籍条目 |

## 数据存储

表2.数据存储表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据存储名 | 输入的数据流 | 输出的数据流 | 组成 |
| 借阅表 | 申请借阅信息 | 借阅信息 | 书号+借阅卡号 |
| 借阅表 | 归还/续期/批准 | 借阅信息 | 是否归还/续期时间/是否批准 |
| 缴费表 | 缴费信息 | 缴费信息 | 缴费条目 |
| 书籍表 | 书籍信息 | 书籍信息 | 书籍条目 |
| 借阅卡表 | 借阅卡信息 | 借阅卡信息 | 借阅卡条目 |
| 用户表 | 用户信息 | 用户信息 | 用户条目 |

## 处理过程

## 1、处理过程“图书借阅”可如下描述：

处理过程：图书借阅

说明：用户借阅，管理员批准或拒绝

输入：图书，借阅卡号

输出：是否批准借阅

处理：用户登陆后通过搜索书籍确定要借阅的书籍，有余量的书籍可以点击借阅，点击借阅后将添加一条记录到借阅表，同时用户和书籍余量均减一，管理员登陆系统之后即可看到借书申请，可以选择批准或拒绝借阅，批准后将批准时间与默认归还时间添加到对应记录，书籍状态改为正在借阅；拒绝后用户和书籍余量均恢复。

## 2、处理过程“图书归还”可如下描述：

处理过程：图书归还

说明：用户归还，管理员批准或拒绝

输入：图书，借阅卡号

输出：是否批准归还

处理：类似于上述图书借阅处理过程，区别是如果归还书籍时归还时间超过应还时间，则新建一条罚款信息至缴费表

## 3、处理过程“创建/修改用户”可如下描述：

处理过程：创建/修改用户

说明：创建/修改用户

输入：用户信息

输出：是否新建成功

处理：管理员在系统中提交新用户的信息，检查系统中是否存在该用户，没有该用户的话则创建新用户。

## 4、处理过程“创建/修改书籍”可如下描述：

处理过程：创建/修改书籍

说明：创建/修改书籍

输入：书籍信息

输出：是否新建成功

处理：类似于上述过程

## 5、处理过程“创建/修改借阅卡”可如下描述：

处理过程：创建/修改借阅卡

说明：创建/修改借阅卡

输入：借阅卡信息

输出：是否创建/修改成功

处理：类似于上述过程

## 6、处理过程“缴费”可如下描述：

处理过程：缴费

说明：缴纳对应条目的罚款费用

输入：缴费信息

输出：是否缴费成功

处理：用户登陆系统之后点击对应的缴费条目进行缴费，确定缴费后将缴费时间存到对应的条目中，缴费完成。

# 概念结构设计

## E-R图

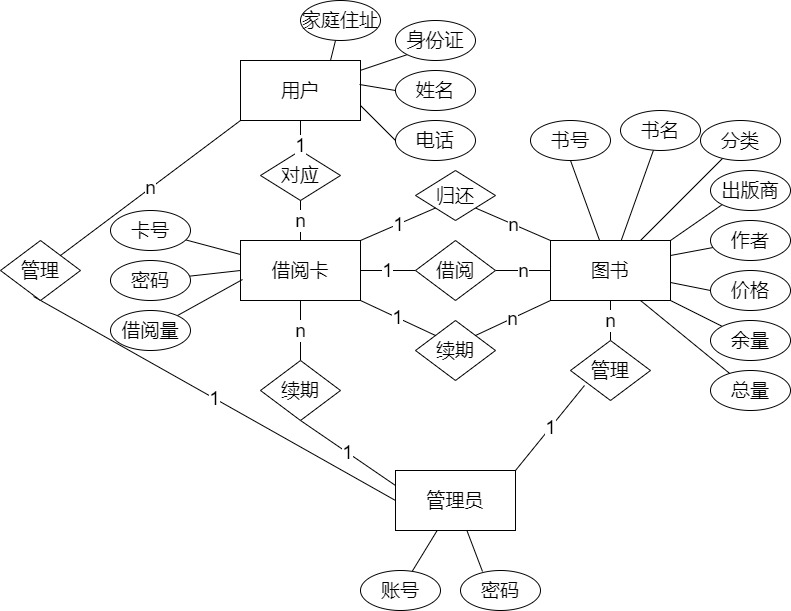


图1.系统的基本E-R图

## 系统说明书

## 1、系统要求

实现图书信息、类别、出版社等信息的管理；实现读者信息、借阅证信息的管理；实现图书的借阅、续借、归还管理；实现超期罚款管理、收款管理；创建触发器，分别实现借书和还书时自动更新图书信息的在册数量；创建视图查询各种图书的书号、书名、总数和在册数；创建存储过程查询指定读者借阅图书的情况；建立数据库相关表之间的参照完整性约束。

## 2、数据流图

**(1)用户模块数据流图**

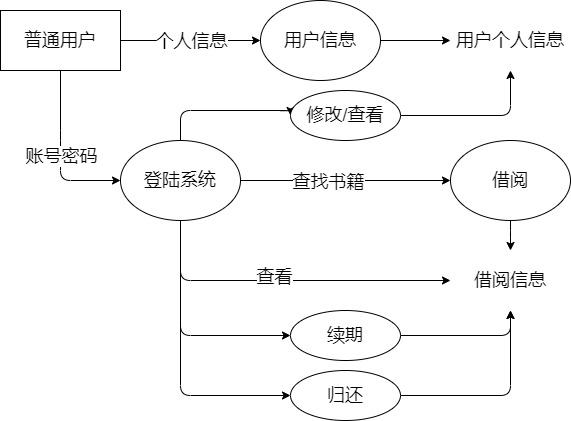


图2.用户模块数据流图

**(2)管理员模块数据流图**

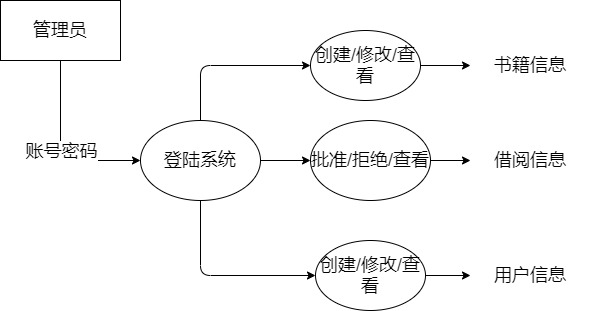


图3.管理员模块数据流图

# 逻辑结构设计

## 关系模式

根据E-R图向关系模型的转换原则，学校图书管理系统的E-R图可以转换为下列关系模式：

1、书籍（书籍id，书名，出版商，作者，分类，总量，余量，价格）

2、借阅（借阅id，对应书籍，对应借阅卡，借阅开始时间，借阅结束时间，实际归还时间，是否超期，是否续期，是否申请归还，是否批准归还，是否批准借阅）

3、用户（用户编号，姓名，电话，地址）

4、借阅卡（借阅卡id，密码，对应用户，借阅总量，借阅余量，借阅卡头像）

5、缴费（缴费id ，缴费价格，是否支付，缴费时间，对应书籍，对应借阅记录，对应借阅卡）

## 数据表的详细结构信息

表3. 管理员表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | AdminAccount | VARCHAR(255) | 是 | 是 |  | 管理员账号 |
| 2 | AdminPwd | VARCHAR(255) | 否 | 是 |  | 管理员密码 |

表4. 书籍表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | bookId | VARCHAR(255) | 是 | 是 |  | 书籍编号 |
| 2 | bookName | VARCHAR(255) | 否 | 是 |  | 书籍名称 |
| 3 | bookPress | VARCHAR(255) | 否 | 否 |  | 书籍出版社 |
| 4 | bookAuthor | VARCHAR(255) | 否 | 否 |  | 书籍作者 |
| 5 | bookClass | VARCHAR(255) | 否 | 否 |  | 书籍分类 |
| 6 | bookTotal | INT(10) | 否 | 是 |  | 书籍总量 |
| 7 | bookAllowance | INT(10) | 否 | 是 |  | 书籍余量 |
| 8 | bookMoney | DOUBLE(10,2) | 否 | 是 |  | 书籍价格 |

表5. 借阅表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | borId | INT UNSIGNED | 是 | 是 |  | 借阅号 |
| 2 | borBook | VARCHAR(255) | 否 | 是 |  | 对应书籍 |
| 3 | borCard | VARCHAR(255) | 否 | 是 |  | 对应借阅卡 |
| 4 | borStart | DATETIME | 否 | 否 |  | 开始时间 |
| 5 | borEnd | DATETIME | 否 | 否 |  | 结束时间 |
| 6 | borRealtime | DATETIME | 否 | 否 |  | 归还时间 |
| 7 | borIsBeyond | BIT | 否 | 是 | 0 | 是否超期 |
| 8 | borIsRenew | BIT | 否 | 是 | 0 | 是否续期 |
| 9 | borIsReturn | BIT | 否 | 是 | 0 | 是否申请归还 |
| 10 | borIsBorrowApprove | BIT | 否 | 是 | 0 | 是否批准借阅 |
| 11 | borIsReturnApprove | BIT | 否 | 是 | 0 | 是否批准归还 |

表6. 缴费表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | payId | INT UNSIGNED | 是 | 是 |  | 缴费号 |
| 2 | payCard | VARCHAR(255) | 否 | 是 |  | 对应卡 |
| 3 | payMoney | DOUBLE(10,2) | 否 | 是 |  | 缴费价格 |
| 4 | payBook | VARCHAR(255) | 否 | 是 |  | 对应书籍 |
| 5 | payIsPay | BIT | 否 | 是 | 0 | 是否支付 |
| 6 | payTime | DATETIME | 否 | 否 |  | 支付时间 |
| 7 | payBor | INT UNSIGNED | 否 | 是 |  | 对应借阅记录 |

表7. 借阅卡表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | cardId | VARCHAR(255) | 是 | 是 |  | 卡号 |
| 2 | cardPwd | VARCHAR(255) | 否 | 是 |  | 借阅卡密码 |
| 3 | cardUser | VARCHAR(255) | 否 | 是 |  | 对应用户 |
| 4 | cardTotal | INT(10) | 否 | 是 | 5 | 借阅总量 |
| 5 | cardAllowance | INT(10) | 否 | 是 | 5 | 借阅余量 |
| 6 | cardAvatar | VARCHAR(255) | 否 | 否 |  | 借阅卡头像 |

表8. 用户表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名** | **数据类型** | **主键** | **非空** | **默认值** | **描述** |
| 1 | userId | VARCHAR(255) | 是 | 是 |  | 用户身份证 |
| 2 | userName | VARCHAR(255) | 否 | 是 |  | 用户姓名 |
| 3 | userTel | INT(10) | 否 | 否 |  | 用户电话 |
| 4 | userAddress | VARCHAR(255) | 否 | 否 |  | 用户地址 |
| 5 | userRole | VARCHAR(10) | 否 | 否 |  | 用户角色 |

## 系统结构图（模块结构）

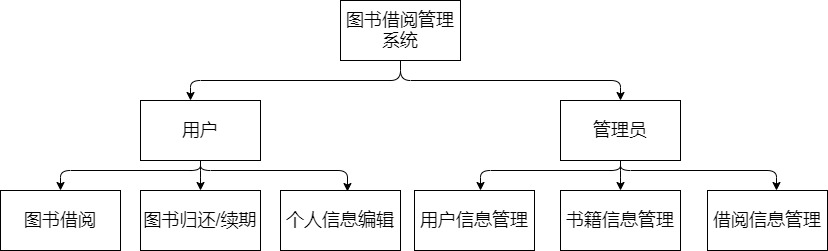


图4.系统结构图

# 物理设计

## 存储安排

确定数据库物理结构的内容要确定数据的存放位置和存储结构（关系、索引、聚簇、日志、备份）和确定系统配置。影响数据存放位置和存储结构的因素既包括硬件环境也包括应用需求（存取时间、存储空间利用率、维护代价）。由于此系统规模较小，所以可以将此系统放于同一磁盘上进行使用。在系统配置上，同时打开此系统的用户数至少有2个，即普通用户与管理员，系统负担较小，配置较低的电脑也可以正常运行。

## 1、存取方法选择

DBMS常用存取方法有，索引方法，目前主要是B+树索引方法、聚簇（Cluster）方法和HASH方法。本系统对数据库物理设计的要求不高，采用MySQL默认分配的物理空间即可满足实践要求，本系统对物理设计并未采取特别设计。

## 2、存取路径建立

数据库数据备份、日志文件备份等由于只在故障恢复时才使用，而且数据量很大，可以考虑与数据库对象（表、索引等）放在不同的磁盘以改进系统的性能。

# 数据库实施

## 创建关系模式

1. **表的创建SQL**

/\*管理员表\*/

CREATE TABLE `admin` (

`AdminAccount` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`AdminPwd` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

PRIMARY KEY (`AdminAccount`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*书籍表\*/

CREATE TABLE `book` (

`bookId` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL COMMENT '书籍编号',

`bookName` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`bookPress` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL,

`bookAuthor` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL,

`bookClass` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL,

`bookTotal` int NOT NULL,

`bookAllowance` int NOT NULL COMMENT '余量',

`bookMoney` double(10, 2) NOT NULL,

PRIMARY KEY (`bookId`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*借阅表\*/

CREATE TABLE `borrow` (

`borId` int(15) UNSIGNED ZEROFILL NOT NULL AUTO\_INCREMENT,

`borBook` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`borCard` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`borStart` datetime NULL DEFAULT NULL,

`borEnd` datetime NULL DEFAULT NULL,

`borRealtime` datetime NULL DEFAULT NULL,

`borIsBeyond` tinyint(1) NOT NULL DEFAULT 0,

`borIsRenew` tinyint(1) NOT NULL DEFAULT 0,

`borIsReturn` tinyint(1) NOT NULL DEFAULT 0,

`borIsBorrowApprove` tinyint(1) NOT NULL DEFAULT 0,

`borIsReturnApprove` tinyint(1) NOT NULL DEFAULT 0,

PRIMARY KEY (`borId`) USING BTREE,

INDEX `book12312`(`borBook`) USING BTREE,

INDEX `user123`(`borCard`) USING BTREE

) ENGINE = InnoDB AUTO\_INCREMENT = 11 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*借阅卡表\*/

CREATE TABLE `card` (

`cardId` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`cardPwd` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`cardUser` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`cardTotal` int NOT NULL DEFAULT 5,

`cardAllowance` int NOT NULL DEFAULT 5,

`cardAvatar` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL,

PRIMARY KEY (`cardId`) USING BTREE,

INDEX `cuser`(`cardUser`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*缴费表\*/

CREATE TABLE `payment` (

`payId` int(15) UNSIGNED ZEROFILL NOT NULL AUTO\_INCREMENT,

`payCard` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`payMoney` double(10, 2) NOT NULL,

`payBook` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`payIsPay` tinyint(1) NOT NULL DEFAULT 0,

`payTime` datetime NULL DEFAULT NULL,

`payBor` int UNSIGNED NOT NULL,

PRIMARY KEY (`payId`) USING BTREE,

INDEX `book1123`(`payBook`) USING BTREE,

INDEX `user1`(`payCard`) USING BTREE

) ENGINE = InnoDB AUTO\_INCREMENT = 3 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*用户表\*/

CREATE TABLE `user` (

`userId` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL COMMENT '用户身份证',

`userName` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`userTel` varchar(11) NULL DEFAULT NULL,

`userAddress` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL,

PRIMARY KEY (`userId`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

/\*外键关联\*/

ALTER TABLE `borrow` ADD CONSTRAINT `book12312` FOREIGN KEY (`borBook`) REFERENCES `book` (`bookId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `borrow` ADD CONSTRAINT `user123` FOREIGN KEY (`borCard`) REFERENCES `card` (`cardId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `card` ADD CONSTRAINT `cuser` FOREIGN KEY (`cardUser`) REFERENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `payment` ADD CONSTRAINT `book1123` FOREIGN KEY (`payBook`) REFERENCES `book` (`bookId`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `payment` ADD CONSTRAINT `user1` FOREIGN KEY (`payCard`) REFERENCES `card` (`cardId`) ON DELETE CASCADE ON UPDATE CASCADE;

1. **视图创建SQL**

CREATE ALGORITHM = UNDEFINED DEFINER = `root`@`localhost` SQL SECURITY DEFINER VIEW `booktoborrow` AS select `borrow`.`borId` AS `borId`,`book`.`bookId` AS `bookId`,`card`.`cardId` AS `cardId`,`book`.`bookName` AS `bookName`,`book`.`bookPress` AS `bookPress`,`book`.`bookAuthor` AS `bookAuthor` from ((`borrow` join `book` on((`borrow`.`borBook` = `book`.`bookId`))) join `card`) where ((`borrow`.`borCard` = `card`.`cardId`) and (`borrow`.`borIsBorrowApprove` = 0));

CREATE ALGORITHM = UNDEFINED DEFINER = `root`@`localhost` SQL SECURITY DEFINER VIEW `booktoreturn` AS select `borrow`.`borId` AS `borId`,`book`.`bookId` AS `bookId`,`card`.`cardId` AS `cardId`,`book`.`bookName` AS `bookName`,`book`.`bookPress` AS `bookPress`,`book`.`bookAuthor` AS `bookAuthor` from ((`borrow` join `book` on((`borrow`.`borBook` = `book`.`bookId`))) join `card`) where ((`borrow`.`borCard` = `card`.`cardId`) and (`borrow`.`borIsReturn` = 1) and (`borrow`.`borIsReturnApprove` = 0));

CREATE ALGORITHM = UNDEFINED DEFINER = `root`@`localhost` SQL SECURITY DEFINER VIEW `class` AS select `book`.`bookClass` AS `className`,sum(`book`.`bookTotal`) AS `classTotal`,sum(`book`.`bookAllowance`) AS `classAllowance` from `book` group by `book`.`bookClass`;

## 程序代码

**（1） 用户登录代码** （用户通过表单提交登录信息，系统通过查询数据库来确定是否可以登录，如果登陆成功则返回session，session中包含用户信息，用户角色，后续访问中均需要带上session）

router.post('/login', async (req, res) => {

let rr = await db.models.card\_model.findOne({where: {[Op.and]: [{cardId: req.body.username ? req.body.username : 'null'}, {cardPwd: req.body.password ? req.body.password : 'null'}]},raw: true})

if (rr) {

let user = await db.models.user\_model.findOne({where: {userId: rr.cardUser},raw: true})

req.session.userName = user.userName

req.session.userId = user.userId

req.session.cardId = rr.cardId

return res.status(200).send({status: 200, message: '登陆成功', userName: user.userName, userId: user.userId, cardId: rr.cardId, role: 'user'})}

else {rr = await db.models.admin\_model.findOne({where: { [Op.and]: [{adminAccount: req.body.username ? req.body.username : 'null'}, {adminPwd: req.body.password ? req.body.password : 'null'}]}})

if (rr) {

return res.send({status: 200, cardId: req.body.userName, role: 'admin'})

} else {

return res.send({status: 500})

}}})

**（2） 分页查询借阅信息** （管理员批量管理借阅信息，为防止前端数据过多导致表格卡顿，前后端协调将数据分页）

router.get('/searchborrow/:radio/:value/:cur?', async (req, res) => {

const obj = {},{cur = 1, count = 15} = req.params,offset = (cur - 1) \* count,radio = req.params.radio === 'cardId' ? 'card' : (req.params.radio === 'bookId' || req.params.radio === 'bookName' ? 'book' : 'borrow'),rr = await db.query(`SELECT\tcard.cardId, \tbook.bookName, \tborrow.borStart, \tborrow.borEnd, \tborrow.borId,\tborrow.borIsReturn, \tborrow.borIsBorrowApprove, \tborrow.borIsReturnApprove, \tbook.bookId\tFROM\tcard\tINNER JOIN\tborrow\tON \t\tcard.cardId = borrow.borCard\tINNER JOIN\tbook\tON \t\tborrow.borBook = book.bookId\tWHERE\tborrow.borIsReturnApprove = 0 AND\t\`${radio}\`.${req.params.radio} LIKE '%${req.params.value}%' and\tborrow.borIsBorrowApprove <> 2\tLIMIT ${count}\tOFFSET ${offset}`)

let tmp = []

for (let i = 0; i < rr[0].length; i++) {

let tt = rr[0][i]

tt.borStart = tt.borStart ? dayjs(tt.borStart).format('YYYY-MM-DD HH:mm:ss') : ''

tt.borEnd = tt.borEnd ? dayjs(tt.borEnd).format('YYYY-MM-DD HH:mm:ss') : ''

tmp.push(tt)

}

let borrowTotal = 0

if (req.params.radio === 'borCard') borrowTotal = await db.models.borrow\_model.count({

where: {borIsReturnApprove: 0, borCard: {[Op.like]: `%${req.params.value}%`}}

})

else {

obj[req.params.radio] = req.params.value

let rrrr = await db.models.book\_model.findOne({where: obj, raw: true})

borrowTotal = await db.models.borrow\_model.count({borBook: {[Op.like]: `%${rrrr.bookId}%`}})

}

return res.send({status: 200, data: {data: tmp, total: borrowTotal}})})

## 测试

**（1）初始界面**

**图形用户界面

描述已自动生成**

图6.初始界面测试图

**（2）用户信息**

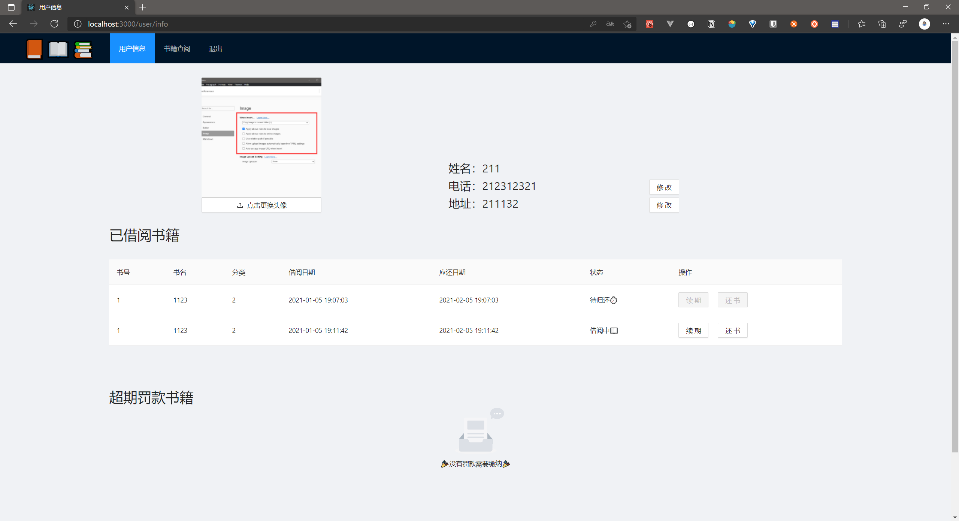
****

图7.用户信息测试图

**（3）书籍查阅**

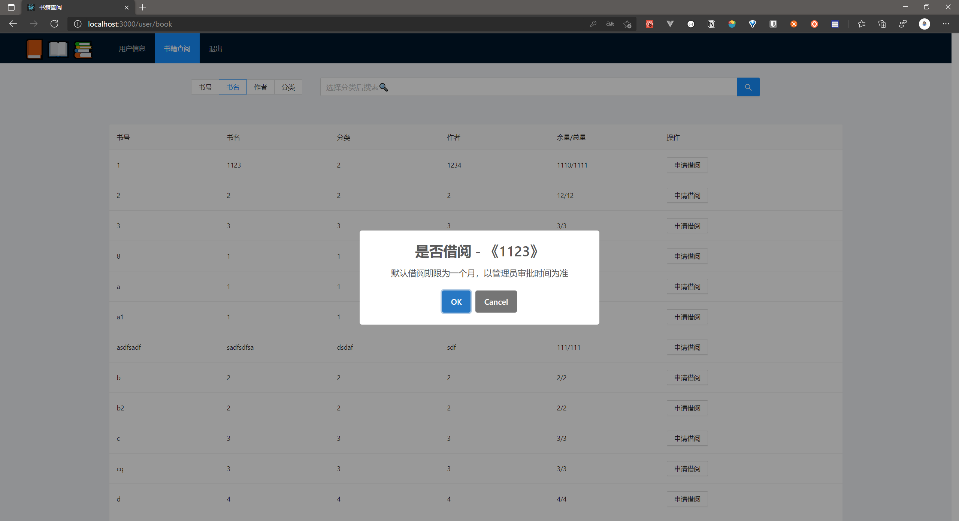
****

图8.用户书籍查阅测试图

**（4）借阅管理**

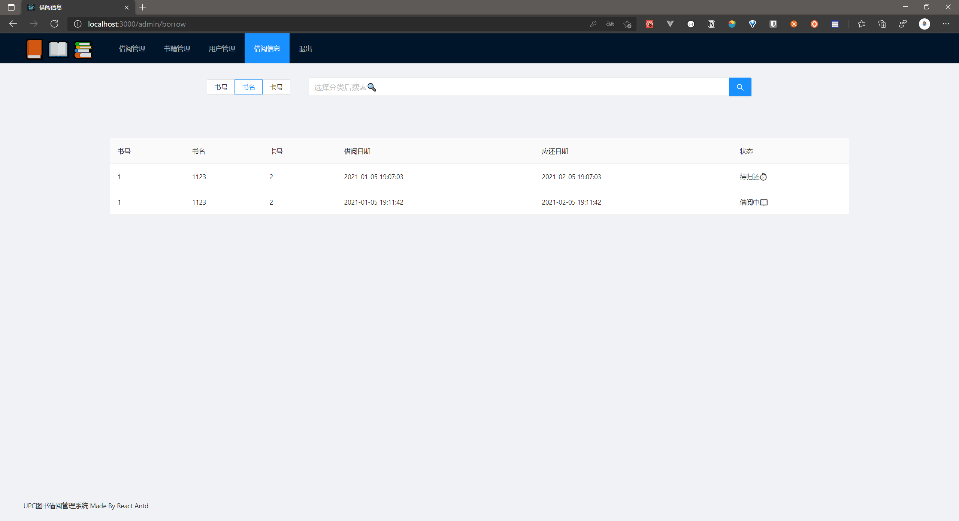
****

图9.借阅管理测试图

**（5）图书管理**

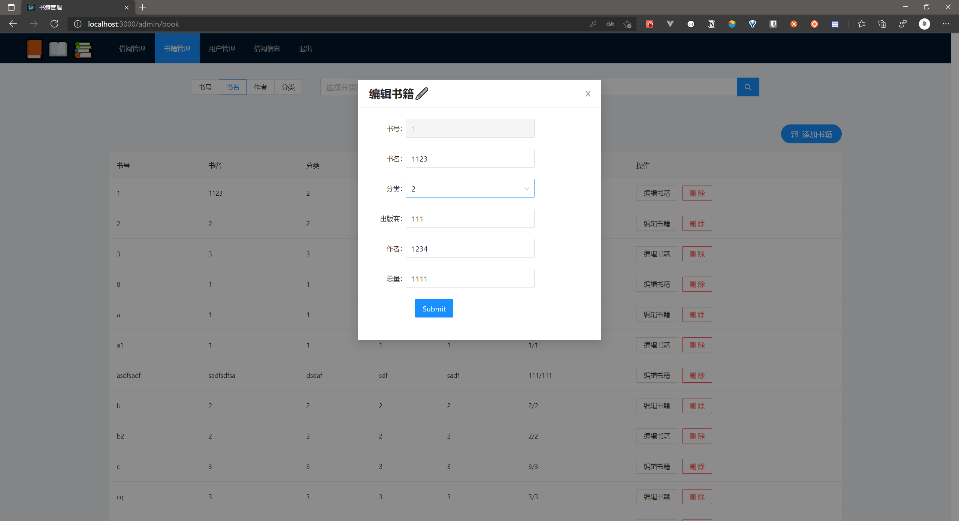
****

图10.图书管理测试图

# 系统设计相关代码

## 存储过程的代码

1. **创建存储过程查询指定读者借阅图书的情况**

CREATE DEFINER=`root`@`localhost` PROCEDURE `borrowed\_book`(IN cardId VARCHAR(100))

BEGIN

SELECT card.cardId,book.bookId,book.bookName,borrow.borStart,CASE borrow.borIsReturnApprove WHEN 0 THEN '否' ELSE '是' END AS '已还' FROM book,card,borrowWHERE card.cardId = cardId AND borrow.borCard = cardIdAND book.bookId = borrow.borBook;

END

## 触发器的代码

**（1）借阅与归还时自动更新卡和书的余量（当提出借阅申请之后，卡和书的余量均减一，归还后自动恢复）**

CREATE DEFINER = `root`@`localhost` TRIGGER `借阅` BEFORE INSERT ON `borrow` FOR EACH ROW BEGIN

/\*当插入数据时将借阅量和余量-1\*/

UPDATE Card SET cardAllowance = cardAllowance - 1 WHERE cardId = new.borCard;

UPDATE Book set bookAllowance = bookAllowance -1 WHERE bookId = new.borBook;

END;

CREATE DEFINER = `root`@`localhost` TRIGGER `批准` AFTER UPDATE ON `borrow` FOR EACH ROW BEGIN

IF new.borIsBorrowApprove <>old.borIsBorrowApprove and new.borIsBorrowApprove = 2

THEN

UPDATE Card set cardAllowance = cardAllowance+1 WHERE cardId = new.borCard;

UPDATE Book set bookAllowance = bookAllowance+1 WHERE bookId = new.borBook;

ELSEIF new.borIsReturn <>old.borIsReturn then

UPDATE Book set bookAllowance = bookAllowance +1 WHERE bookId = new.borBook;

UPDATE Card SET cardAllowance = cardAllowance + 1 WHERE cardId = new.borCard;

end if;

END;

1. **当修改卡或者书的总量时自动更新余量**

CREATE TRIGGER `自动增加余量` BEFORE UPDATE ON `book` FOR EACH ROW BEGIN

IF new.bookTotal <> old.bookTotal

THEN

SET new.bookAllowance = new.bookAllowance + new.bookTotal - old.bookTotal;

END IF;

END;

CREATE TRIGGER `自动增加余量` BEFORE UPDATE ON `card` FOR EACH ROW BEGIN

IF new.cardTotal <> old.cardTotal

THEN

SET new.cardAllowance = new.cardAllowance + new.cardTotal - old.cardTotal;

END IF;

END;

# 感想及心得体会

这次数据库实习时间相对于以前的实习来说，时间较短。但是这次的实习内容非常多，要设计一个拥有完整前后端的系统，挺复杂的。这次对于工作量的预估有所失误，以为工作量不大，所以前端方面选择了React做框架，边学边写，到后期导致时间有点不够，加了点班才能完成。

虽然整个系统开发期间比较累，但是确实学到了很多东西，特别是对前端框架React有了一个全面的认识，可以通过它开发一个完整的前端系统；其次对数据库的整体设计也有了一个更深的认识，以前也做过这种项目，像web数据库一类的课程。但是以前做的项目更偏向于层次较高的前端，主要做展示用，这次偏向于底层数据库的设计，弥补了我以前只关注顶层实现的缺点，收获很大。