《教材订购管理系统》

数据库设计报告

组长：林 铭-2018210312057

组员：邱龙风-2018212212001

组员：章佳璐-2017211401036

组员：周 双-2017211401082

# 需求分析

## 1.1系统功能：

本系统主要为学生教材订购，书商对书库里的书籍进行管理，学院管理员对自己学院里的各班级订书情况进行管理提供服务。

## 1.2 具体模块功能分析：

### 1.2.1学生端：

1. 查询书籍的具体信息。
2. 选择并订购书籍。
3. 查看自己的订单详情。
4. 取消自己已订购的书籍。

### 1.2.2学院管理员端：

1.查看自己学院里每个班级的具体订书信息。

2.与书商进行教材领取确认。

### 1.2.3书商端：

1. 查看书库中所有书籍的信息包括库存数量。
2. 对现有书籍进行信息修改和删除。
3. 添加新的书籍信息。
4. 查询每本书的具体订购情况。
5. 对库存不足的书进行购买。

6.对各学院的管理员发送领书单通知来领书。

# 概念结构设计

## 2.1 系统ER图设计

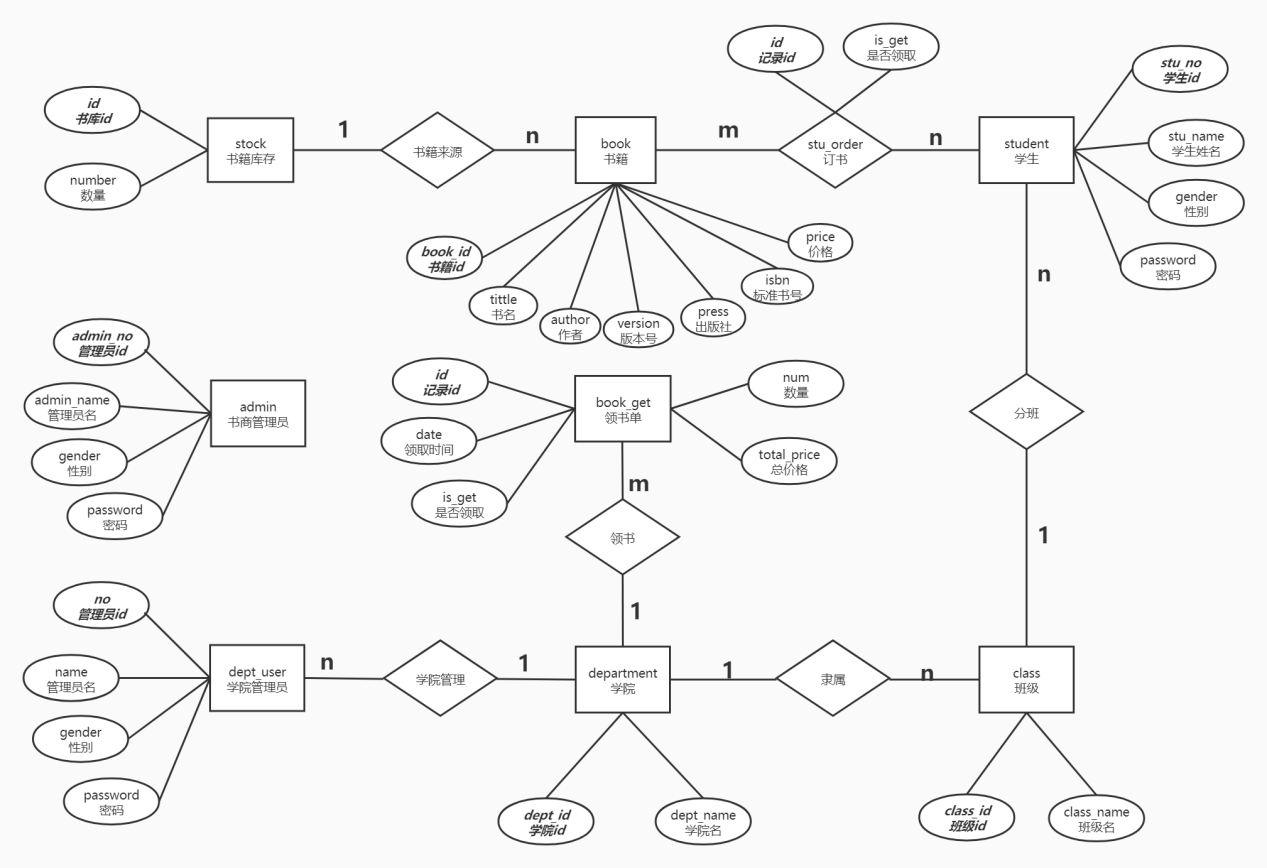


图 1 系统ER图

## 2.2 各个实体

### 2.2.1 stock

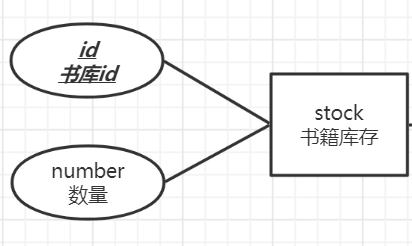


图 2 库存

### 2.2.2 book

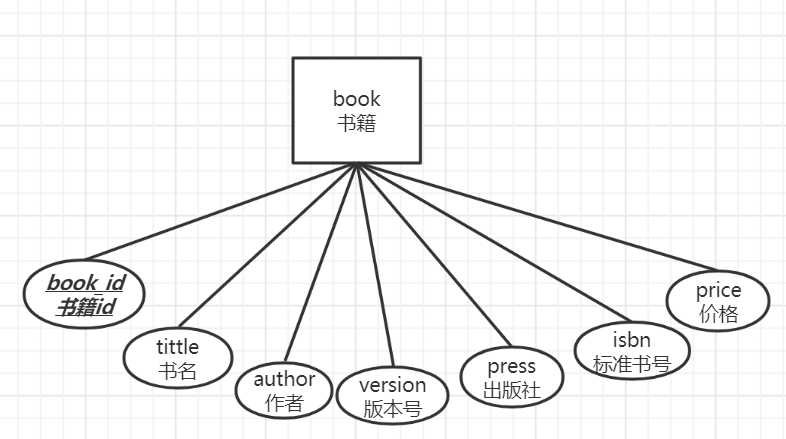


图 3 书籍

### 2.2.3 student

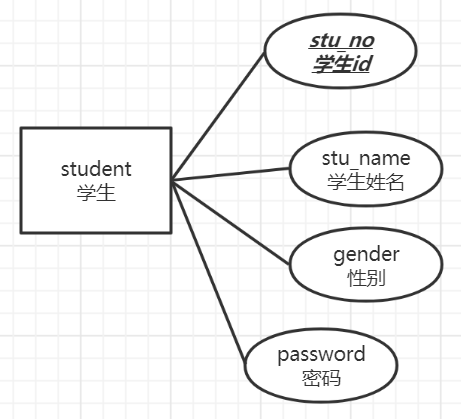


图 4 学生

### 2.2.4 class

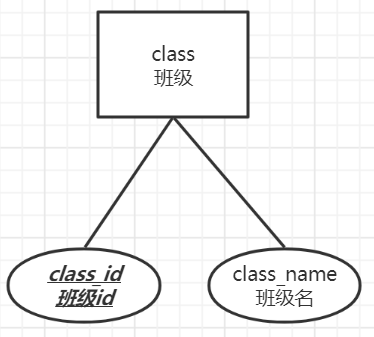


图 5 班级

### 2.2.5 department

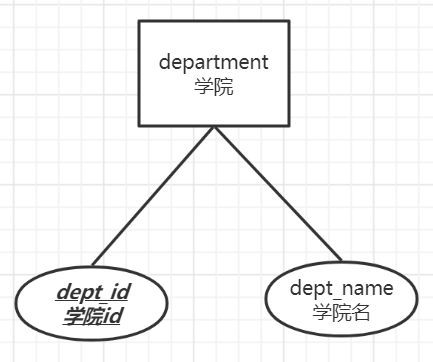


图 6 学院

### 2.2.6 depr\_user

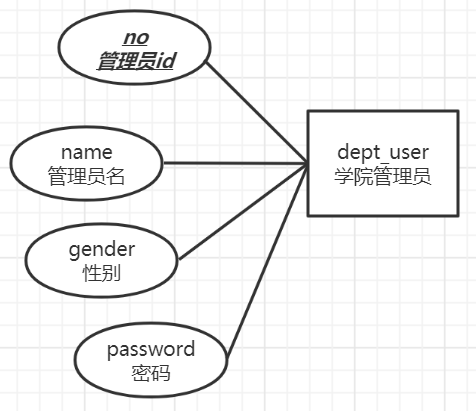


图 7 学院管理员

### 2.2.7 admin

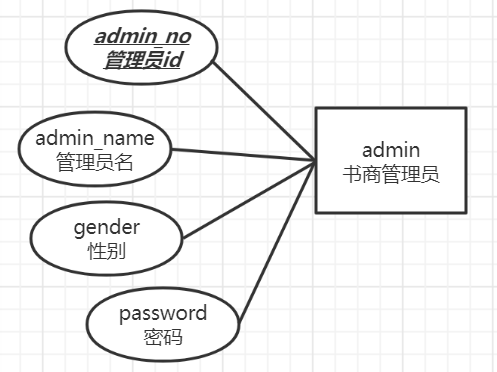


图 8 书商管理员

### 2.2.8 book\_get

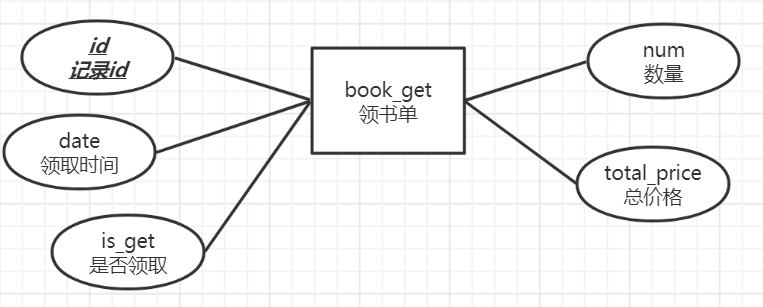


图 9 领书单

# 逻辑结构设计

## 3.1 实体关系ER图

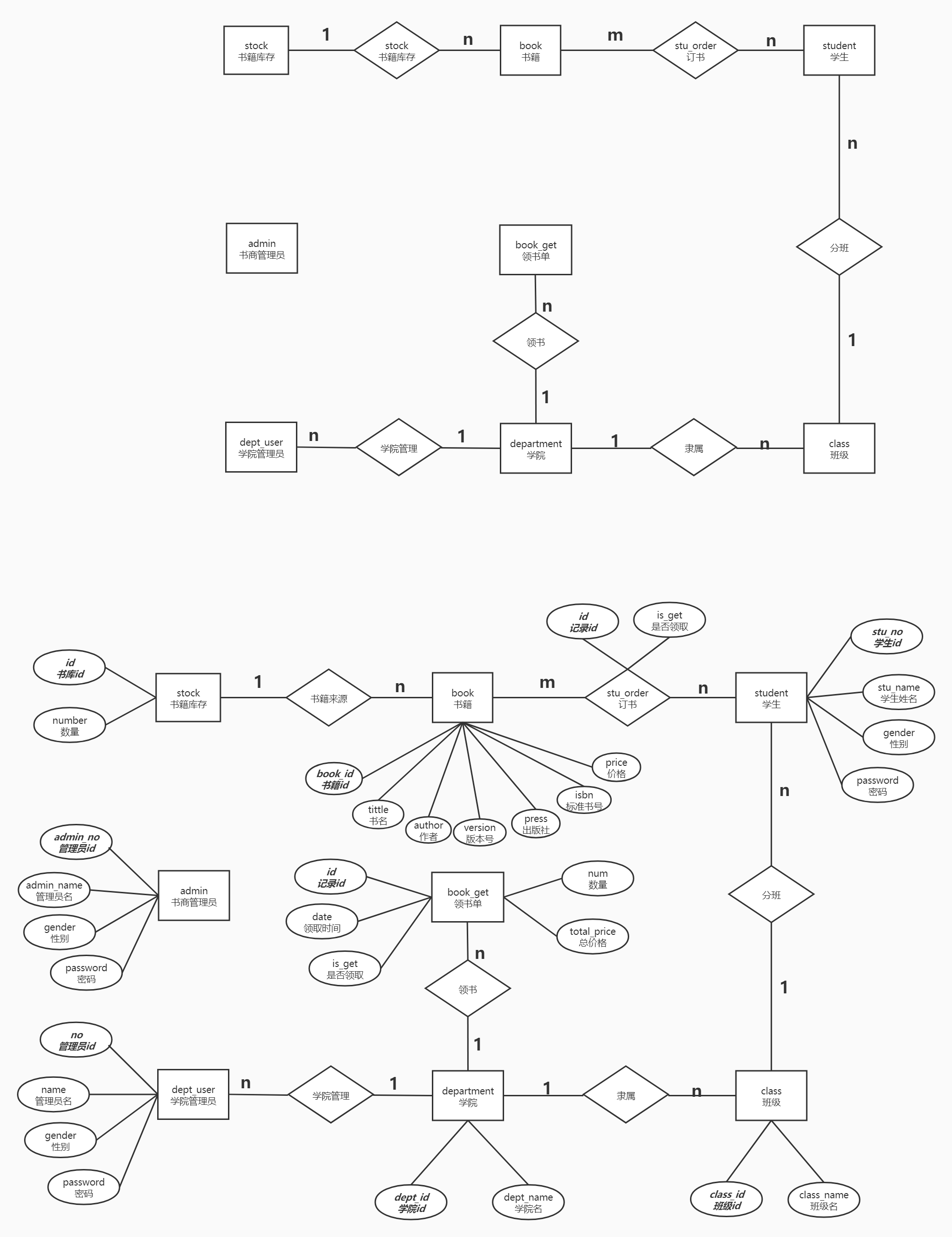
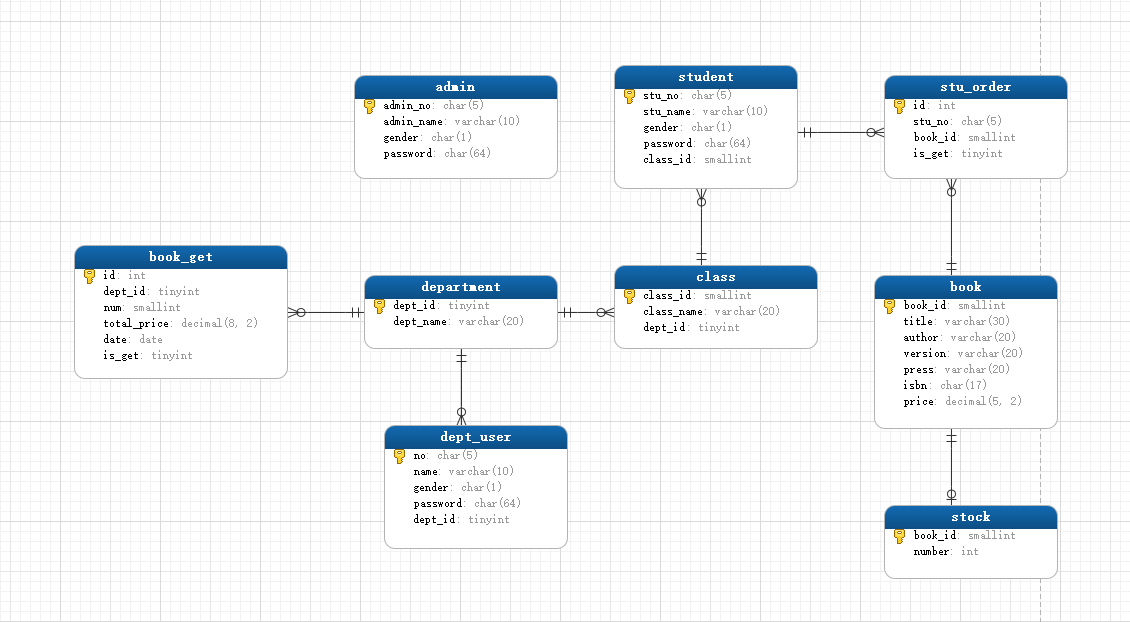


图 10 实体关系ER图

## 3.2 实体关系转换



### 1对N

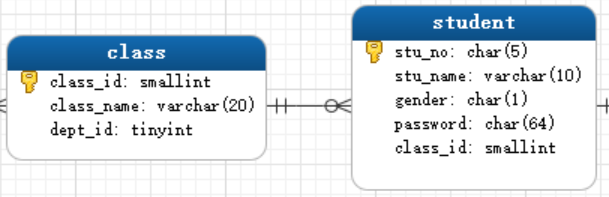


图 11 班级-学生 ER图

每个班级都有多名学生，每个学生都只属于一个班级。所以班级与学生之间是一对多关系，因此将class\_id作为student的外键。

class(class\_id, class\_name, dept\_id)

student(stu\_no, stu\_name, gender, password, class\_id)

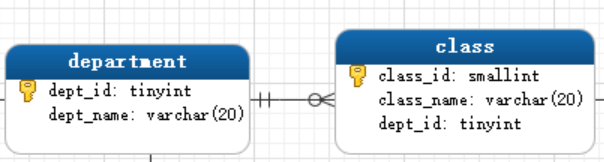


图 12 学院-班级 ER图

每个学院都有多个班级，每个班级都只属于一个学院，所以学院与班级之间是一对多的关系，因此将dept\_id作为class的外键。

department(dept\_id, dept\_name)

class(class\_id, class\_name, dept\_id)

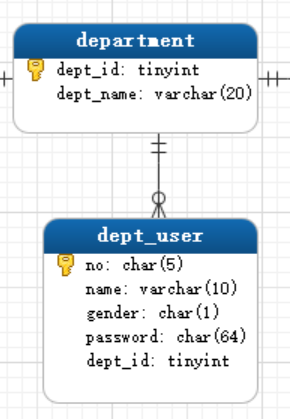


图 13 学院-学院管理员 ER图

每个学院都有多名学院管理员，每个学院管理员都只属于一个学院，所以学院与学员管理之间是一对多的关系，因此将dept\_id作为dept\_user的外键。

department(dept\_id, dept\_name)

dept\_user(no, name, gender, password, dept\_id)

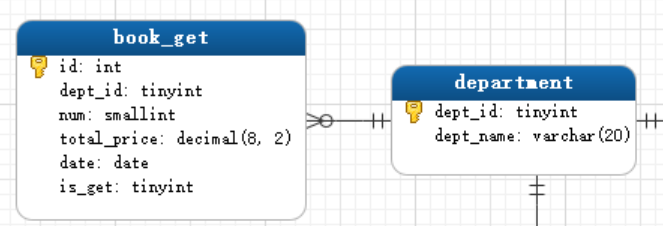


图 14 领书单-学院 ER图

领书单上有多条领书通知，每条领书通知对应一个学院，所以领书单与学院是一对多的关系，因此将dept\_id作为book\_get的外键。

department(dept\_id, dept\_name)

book\_get(id, dept\_id, num, total\_price, date, is\_get)

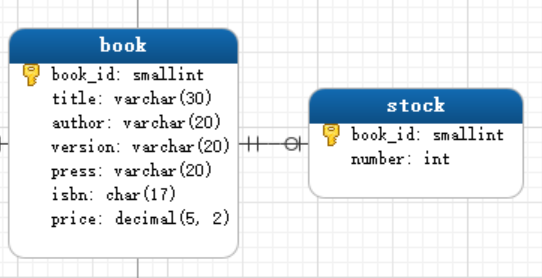


图 15 库存-书籍 ER图

book(book\_id, tittle, author, version, press, isbn, price)

stock(book\_id, num)

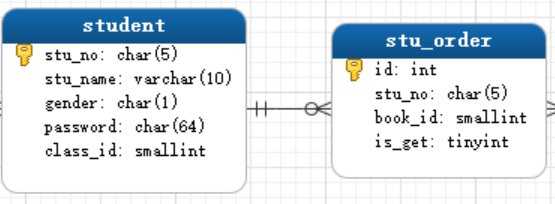


图 16 学生订单-学生 ER图

学生订购单中有多条订单记录，每条订单记录对应一个学生，所以学生订单与学生是一对多的关系，因此将stu\_no作为stu\_order的外键。

student(stu\_no, stu\_name, gender, password, class\_id)

stu\_order(id, stu\_no, book\_id, is\_get)

### M对N

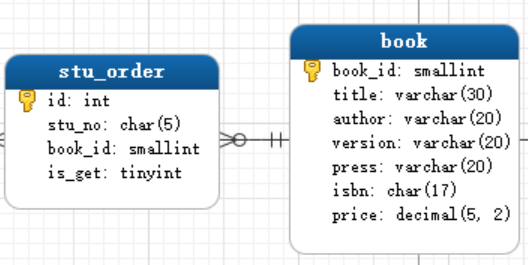


图 17 学生订单-学生 ER图

学生订购单中，每个学生有多条订单记录，对应多本书，一本书可以被同一个学生订购多次，所以学生订单与书籍是多对多的关系，因此将book\_id作为stu\_order的外键。stu\_order的候选键只有id。

book(book\_id, tittle, author, version, press, isbn, price)

stu\_order(id, stu\_no, book\_id, is\_get)

## 3.3 优化、分析关系模型

为了便于程序的设计和业务功能的实现，本项目对部分E-R图的关系模型进行了优化，最终得到了如下的数据表结构。

表 1 student表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型及长度 | 字段含义 | 约束 | **说明** |
| stu\_no | char(5) | 记录id | PK | auto\_increment，固定长度5 |
| stu\_name | varchar(10) | 学生姓名 | 非空 |  |
| gender | char(1) | 性别 | 非空 | 男/女 |
| password | char(64) | 密码 | 非空 | SHA256加密，固定长度64 |
| class\_id | smallint | 班级id | FK\_class(class\_id) |  |

函数依赖：stu\_no—>stu\_name, gender, password, class\_id

候选码：stu\_no

主属性：stu\_no

非主属性：stu\_name, gender, password, class\_id

student表中中每一个属性都是不可再分的原子值，满足1NF；候选键为stu\_no，主键为stu\_no，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 2 department表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| dept\_id | tinyint | 部门id | PK | auto\_increment |
| dept\_name | varchar(20) | 部门名 | 非空 |  |

函数依赖：dept\_id—>dept\_name

候选码：dept\_id

主属性：dept\_id

非主属性：dept\_name

department表中中每一个属性都是不可再分的原子值，满足1NF；候选键为dept\_id，主键为dept\_id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 3 class表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| class\_id | smallint | 班级id | PK | auto\_increment |
| class\_name | varchar(20) | 班级名 | 非空 |  |
| dept\_id | tinyint | 学院id | FK\_department(dept\_id) |  |

函数依赖：class\_id—>class\_name, dept\_id

候选码：class\_id

主属性：class\_id

非主属性：class\_name, dept\_id

class表中中每一个属性都是不可再分的原子值，满足1NF；候选键为class\_id，主键为class\_id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 4 dept\_user 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| no | int | 学院管理员号 | PK | auto\_increment |
| name | varchar(10) | 姓名 | 非空 |  |
| gender | char(1) | 性别 | 非空 | 男/女 |
| password | char(64) | 密码 | 非空 | SHA256加密，固定长度64 |
| dept\_id | tinyint | 部门id | FK\_department  (dept\_id) |  |

函数依赖：no—>name, gender, password, dept\_id

候选码：no

主属性：no

非主属性：name, gender, password, dept\_id

dept\_user表中中每一个属性都是不可再分的原子值，满足1NF；候选键为no，主键为no，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 5 admin 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| admin\_no | char(5) | 书商管理员号 | PK | auto\_increment  固定长度为5 |
| admin\_name | varchar(10) | 管理员姓名 | 非空 |  |
| gender | char(1) | 性别 | 非空 | 男/女 |
| password | char(64) | 密码 | 非空 | SHA256加密，固定长度64 |

函数依赖：admin\_no—>name, gender, password, dept\_id

候选码：admin\_no

主属性：admin\_no

非主属性：name, gender, password, dept\_id

admin表中中每一个属性都是不可再分的原子值，满足1NF；候选键为admin\_no，主键为admin\_no，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 6 stock 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| book\_id | smallint | 记录id | PK,  FK\_book(book\_id) | auto\_increment，级联更新删除 |
| number | int | 库存  数量 | 非空 | 触发器控制，订单发出时库存相应减少 |

函数依赖：book\_id—>number

候选码：book\_id

主属性：book\_id

非主属性：number

stock表中中每一个属性都是不可再分的原子值，满足1NF；候选键为book\_id，主键为book\_id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 7 book 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| book\_id | smallint | 记录id | PK | auto\_increment |
| tittle | varchar(30) | 书籍名 | 非空 |  |
| author | varchar(20) | 作者 | 非空 |  |
| version | varchar(20) | 版本号 |  |  |
| press | varchar(20) | 出版社 |  |  |
| isbn | char(17) | 标准书号 |  | isbn算上-后长度固定为17位 |
| price | decimal | 价格 | 非空 |  |

函数依赖：book\_id—> title, author, version, press, isbn, price

isbn—>book\_id, title, author, version, press, isbn, price

候选码：( book\_id, isbn )

主属性：book\_id

非主属性：title, author, version, press, isbn, price

book表中中每一个属性都是不可再分的原子值，满足1NF；候选键为 ( book\_id, isbn )，主键为book\_id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 8 stu\_order 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| id | int | 记录id | PK | auto\_increment |
| stu\_no | char(5) | 学生学号 | FK\_student(stu\_no) |  |
| book\_id | smallint | 书籍id | FK\_book(book\_id) | 级联更新删除 |
| is\_get | tinyint | 是否领取 | 非空 | 1-领取 0-未领取 默认为0 |

函数依赖：id—> stu\_no, book\_id, is\_get

候选码：id

主属性：id

非主属性： stu\_no, book\_id, is\_get

**分析：每个学生同一本可以定多次，所以stu\_no和book\_id不能做主键，故候选码只有订单id。**

stu\_order表中中每一个属性都是不可再分的原子值，满足1NF；候选键为id,，主键为id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

表 9 book\_get 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型及长度** | **字段含义** | **约束** | **说明** |
| id | int | 记录id | PK | auto\_increment |
| dept\_id | tinyint | 部门id | FK\_department(dept\_id) |  |
| num | smallint | 订单书籍总量 | 非空 |  |
| total\_price | decimal | 订单总价格 | 非空 |  |
| date | date | 领取时间 | 非空 |  |
| is\_get | tinyint | 是否领取 | 非空 | 默认0 学院管理员点击确认后为1 |

函数依赖：id—> dept\_id, num, total\_price, date, is\_get

候选码：id

主属性：id

非主属性： dept\_id, num, total\_price, date, is\_get

book\_get表中中每一个属性都是不可再分的原子值，满足1NF；候选键为id,，主键为id，所有非主属性都完全函数依赖于候选键，满足2NF；非主属性不存在对候选键的传递依赖，满足3NF；每一个决定属性集（因素）都包含（候选）码, 所有非主属性都完全函数依赖于码, 所有主属性都完全函数依赖于每一个不包含它的码，满足BCNF。

## 3.4 触发器

触发器控制，订单发出时库存相应减少



图 18 触发器

## 3.5 索引

书籍名称是常被用于搜索的，但书名存在重复，所以这里为tittle字段设置了一个普通索引，加快检索的速度。

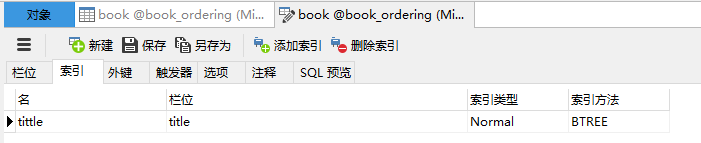


图 19 索引

# 系统使用手册

## 4.1登录界面

学生、学院管理员、商家可以从登陆界面输入自己的账号密码然后选择相应的用户登录系统。后台会对账号密码进行校验，校验成功后，用户成功进入系统。否则提示账号或密码错误。

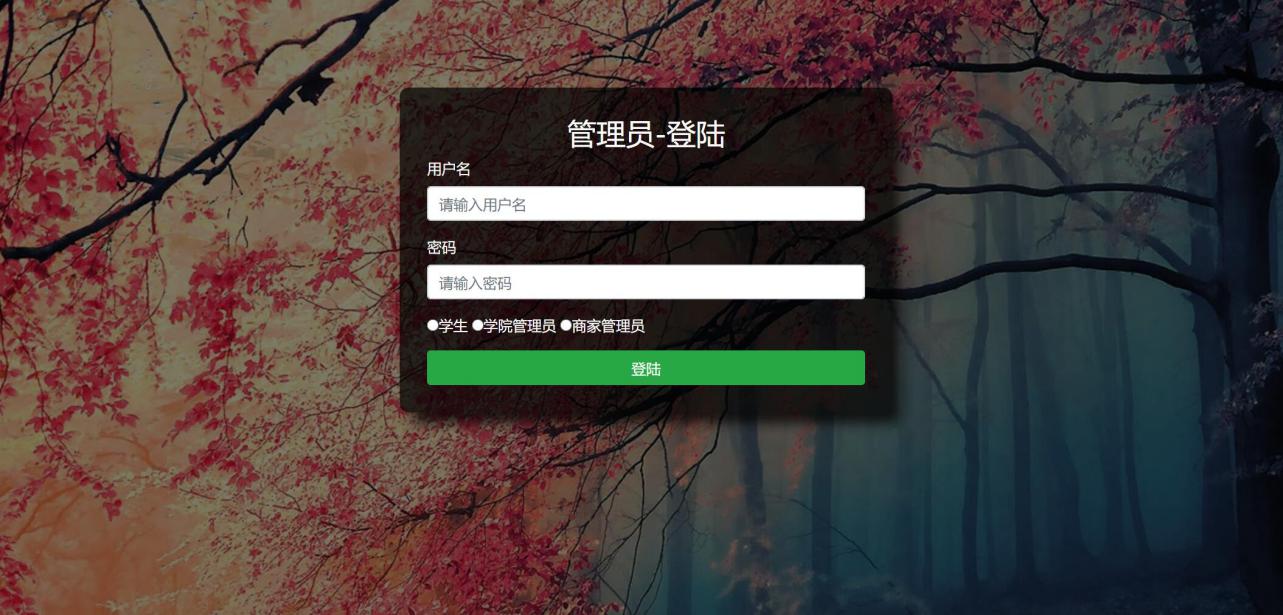


图 20 登录界面

## 4.2进入系统后

### 4.2.1学生端

点击书籍查询订购模块后进入到如下界面，学生可以通过书籍的某个属性来检索到他想要买的书籍，然后单击订购按钮后即可购买此书。

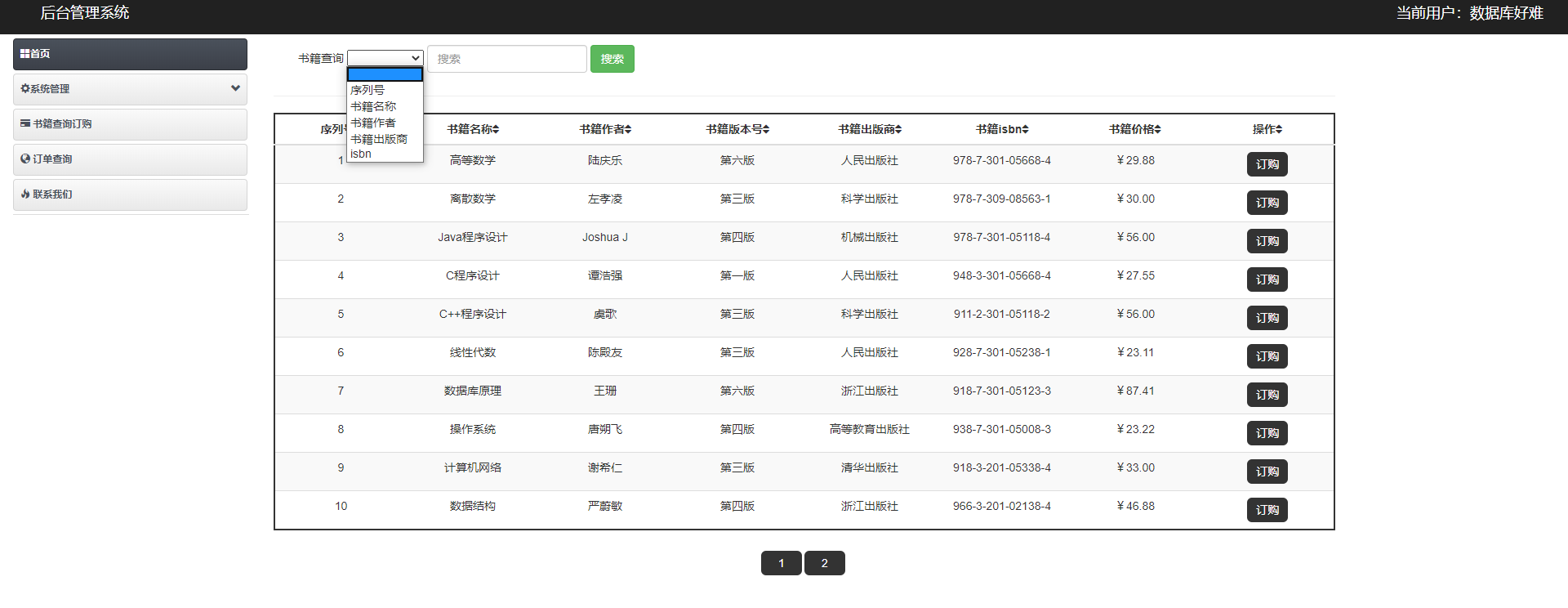


图 21 书籍查询订购界面

点击订单查询模块后进入到如下界面，学生可以查看到自己已订购的书籍有哪些，并且单击取消按钮后，可以取消订购该书

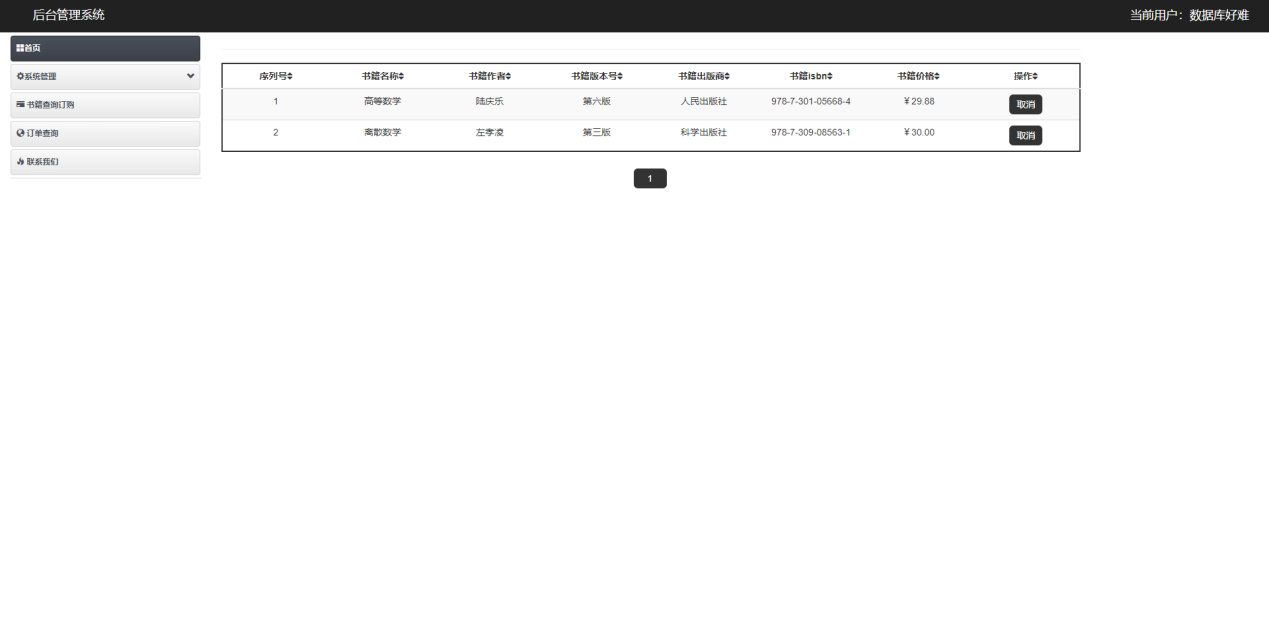


图 22 学生订单查询取消界面

### 4.2.2商家端

点击书籍管理模块后进入到如下界面可以看到书库所有书的信息



图 23 书籍管理界面

点击右上角的添加书籍链接后会进入到添加书籍界面如下。填写完信息后提交即可。

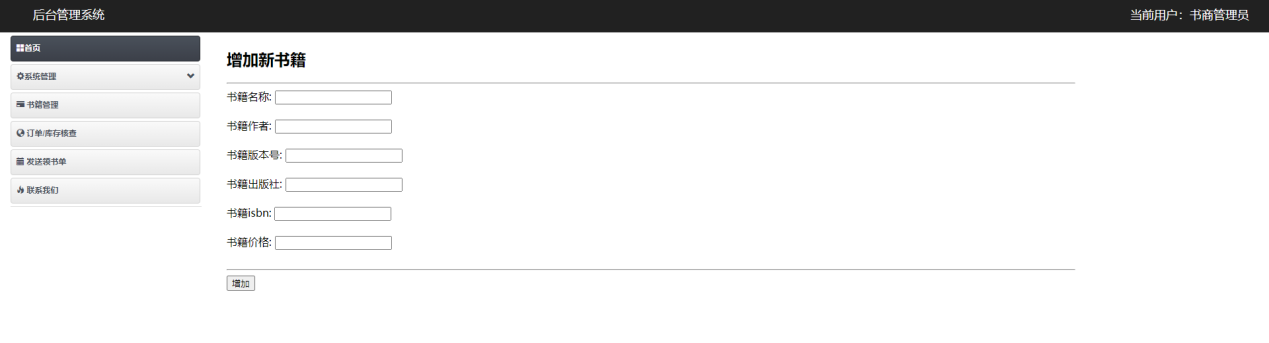


图 24 添加书籍界面

点击修改后会进入修改书籍界面如下，可对书籍信息进行修改

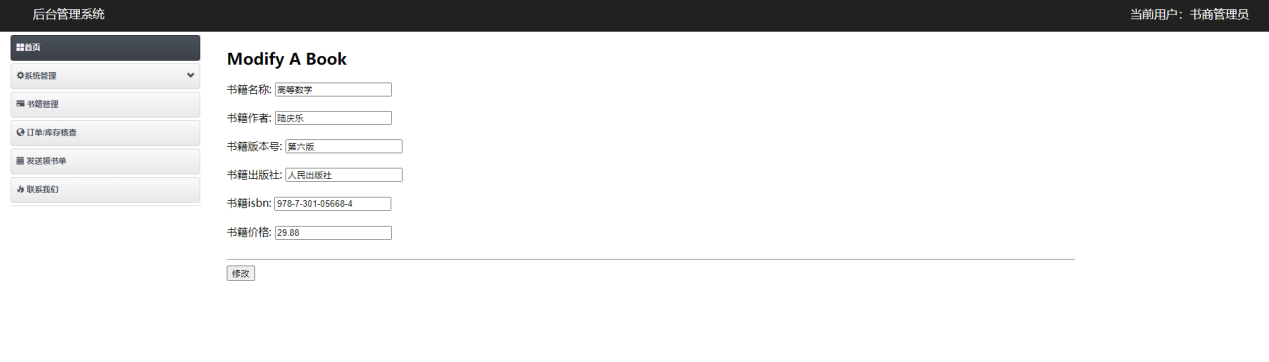


图 25 添加书籍界面

在订单/库存管理界面可以看到每一本书的的库存数量及被订购数量，若库存数量少于被订购数量则会出现输入框和增加按钮，在输入框中输入想要增加的数量提交即可，库存内该书的数量便会更新。



图 26 订单/库存管理界面

在发送领书单界面能看到各学院教材订购数量，如果书籍不足系统会提示书籍缺少，字体变为红色。若学院订购的各书籍库存充足则会出现发送领书单按钮，选择相应的领取时间后点击按钮后即可通知到学院管理员前来领书。



图 27 发送领书单界面

### 4.2.3学院管理员端

进入订单查询界面，即能查询到到学员管理员所属学院里各班级的订书数量和总价格。倘若学院所在班级过多，可在输入框搜索班级关键词，点击搜索即可在最上方显示搜索结果。值得一提的是，该功能支持模糊查询。

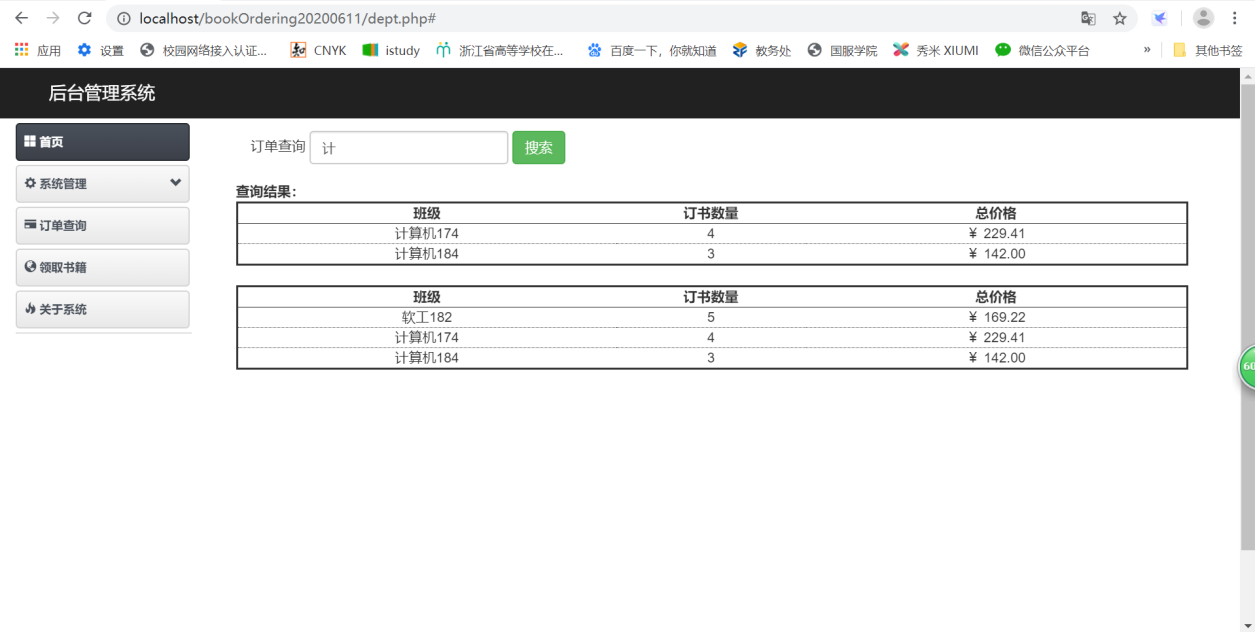


图 28 学院管理订单查询界面

在领取书籍界面，可以看到学院总体领取书籍的情况，如果核对无误的话点击右侧的确认领取按钮即可与商家确认已经领取完毕。点击后，按钮即会自动变换为“已领取”。

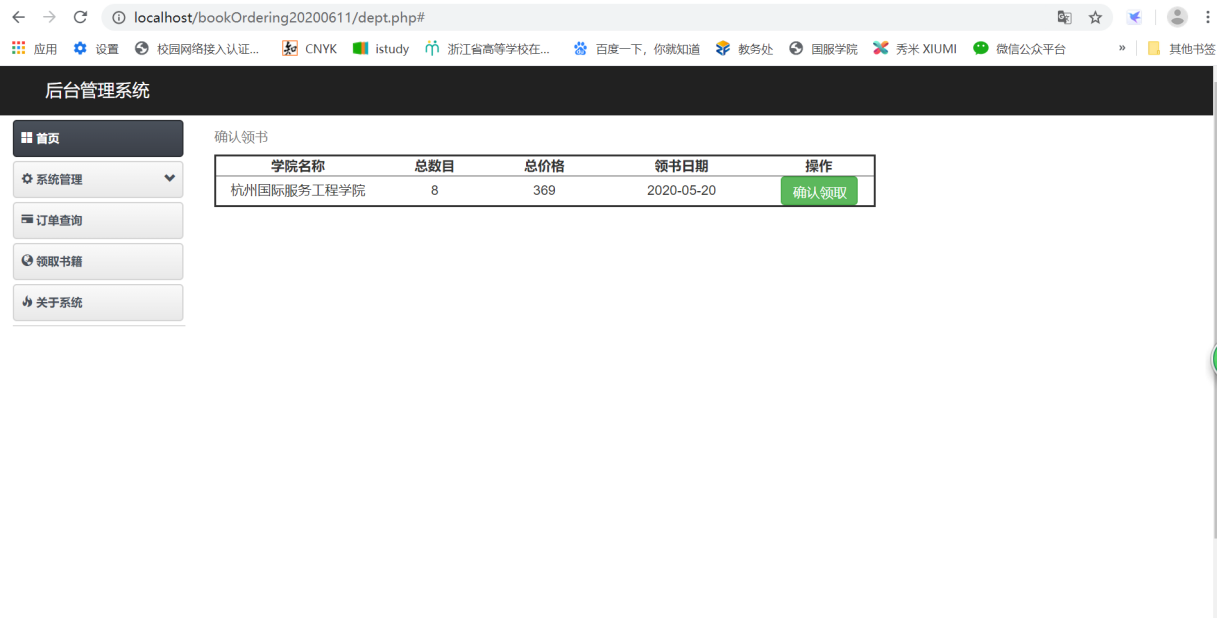


图 29 领取书籍界面

### 4.2.4 修改密码

单击系统管理下的修改密码，即可进入修改密码界面。登录状态下，若输入原密码正确，且两次新密码输入一致，即可改密成功，否则会提示相应的错误，表明改密失败。



图 30 修改密码界面

## 4.3 退出系统

单击系统管理下的退出登录，即可退出系统。



图 30 退出登录界面

# 总结

## 组员分工

林铭：书商管理员发送领书单模块，数据字典，整合代码，录制视频

周双：学生端全部模块，界面设计，登录模块，设计文档

章佳璐：学院管理员端全部模块，修改密码，ER图，ppt

邱龙风：书商管理员书籍管理模块，库存查核模块，ppt，设计文档

需求分析，数据库设计：小组讨论

## 心得体会

通过这次数据库大作业，我进一步认识到数据库设计对实践项目的重要性——可以说，数据库设计是整个项目的精髓，其起到了举足轻重的作用。虽然，一个项目成果，带给用户最直接的体验是UI和交互，仿佛前端美观即为至上，但在此次开发中，我渐渐坚定了“美观”绝不是项目最终评价标准的想法。  
一个项目的好坏，与其数据库设计、项目延展性、鲁棒性等密切相关，而这些往往要求项目背后有一个考虑全面的数据库支撑，否则在开发过程中一旦想要重新设计数据库的某部分或修改结构将会事倍功半，debug永无止境……从侧面看，这也反映了另一个不容小觑的事实——项目前期的需求分析非常重要，一定要留充足的时间（哪怕这段时间占了整个项目开发过程的一半）讨论需求，尽可能考虑周全，确定实体、设计实体间的各种关系，再进行逻辑结构设计、物理结构设计，最后再进行实施、并运行维护数据库。  
学无止境，奥力给~  
——章佳璐

从这次实践，我发现展现在的客户端的动态数据，都存储在数据库中，因此良好的数据库设计十分重要，可以提升性能，使代码变得简单、清晰。

我们在进行数据库设计时，要从需求分析到概念结构设计，再到逻辑结构设计和物理结构设计，最后再进行数据库实施及数据库运营与维护，结合数据库设计的原则一步步仔细推敲，多方面考虑问题。

经过这次实践，我亲身体验了数据库设计的全过程，在实践中了解了数据库系统设计的步骤、流程以及思路，对数据库的增删改查更加熟悉，通过对范式的分析，对相关概念的理解也更加清晰。我深刻认识到以前所学的基础课程的重要性。还对老师上课拓展的知识，触发器、索引等内容有了更深的了解。通过团队合作，从其他组员身上也学习到了很多东西，对于一些理解不深刻的概念，通过小组探讨，慢慢理解，加深了认识。

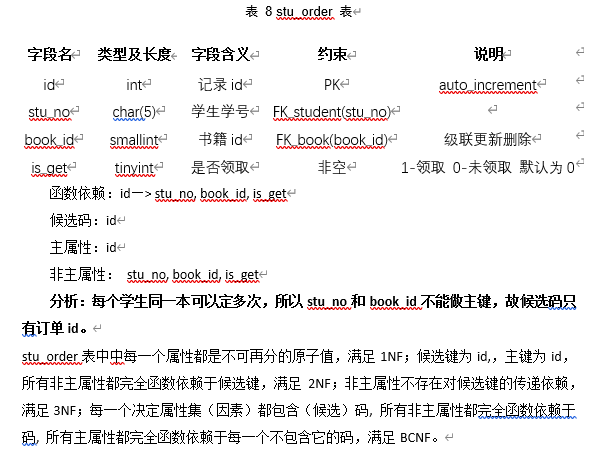
—— 周双

通过这次的数据库大作业实践，深刻的感受到了对于一个项目来说数据库是有多么的重要。前期我们根据需求分析，花了很多的时间在数据库的设计上。数据库的设计决定了这个项目之后的数据是否容易维护，以及需求是否容易扩展。若数据库没有设计好，一个稍微的小改动可能就要设计到多张表的改动，所以前期在设计环节花了很多时间是值得的。前期最早没有计划写数据字典，但后面发现常常会忘记自己设置的字段的含义。写好数据字典后可以常常翻阅，就如一本手册一样很是便利。在实践的过程中，对sql的增删改查更加的熟练了，也将所学的触发器，索引等知识用在了实际开发中。通过这次的经验，我想今后应该在写这些项目的过程中会在数据库的设计环节上投入更多的时间。  
——林铭

这次数据库的大作业是自己第一个参与完成的小型数据管理系统，感触良多。从一开始的需求分析到最后的设计总结，也算是让我体会到了要去系统的完成一个项目具体要经过哪些流程。其中我觉得最重要的便是数据库的设计阶段，因为后面的功能开发都要围绕数据库里表的结构和表之间的联系来展开。我自己在功能开发的时候也深刻的意识到了这一点。如果一开始拥有一个结构设计优良的数据库，那么后面开发的工作量会大大减少。因为在维护数据的时候，逻辑会非常清晰，代码编写难度自然会大大减少。反之如果数据库里各表之间的联系杂乱无章那无疑会给后面的开发工作造成很多不必要的麻烦，甚至影响功能的实现。在完成这个项目的过程中，也加深了我对很多基础概念的理解，比如ER图的设计要考虑哪些，如何合理的设计表的字段属性，加强了对数据库中几大范式的理解... 总的来说，这次数据库大作业还是让自己受益匪浅。

——邱龙风

## 答辩后修改内容



若一个学生同一本书只能订购一次，则id，stu\_no+book\_id均可作为候选码。

若一个学生同一本书可以订购多次，则只有id可以作为候选码。

这里我们的系统设置的是一个学生同一本书可以订购多次，所以候选码只有id。