

基于 MySQL 与 Django 框架的图书馆管理系统

李彦昊 2012286 特色班

一、项目整体介绍

该项目基于 Mysql 与 Django 框架设计开发了一个简单的图书馆数据库管理系统，包括图书馆内书籍的信息、出版社、作者、管理员、读者、租借信息、归还信息，此系统用户面向图书管理员和借阅读者，图书管理员可以完成书籍、读者、作者、出版社、读者、管理员、租借、归还等基本信息的增加、删除、修改、查阅。

本项目在考虑多种技术后，前端采用基于 Django Web 应用框架，同时网页页面采用 Bootstrap 框架，后端基于 MySQL 数据库，整体利用 python 语言连接。

二、技术特点

MySQL:

1) 功能强大

MySQL 中提供了多种数据库存储引擎，用户可以选择最合适的引擎以得到最高性能，可以处理每天访问量超过数亿的高强度的搜索 Web 站点，由于大型图书馆中书目数量多，因此对数据库的检索效率提出了一定的要求。同时，MySQL 支持事务、视图、存储过程、触发器等操作，为项目设计开发提供较好的基础。

2) 支持跨平台

MySQL 支持至少 20 种以上的开发平台，包括 Linux、Windows、Mac OS 等。这使得在任何平台下编写的程序都可以进行移植，而不需要对程序做任何修改。考虑到图书馆内机器设备的不同，MySQL 的跨平台性为图书馆管理系统提供较好基础。

3) 运行速度快

高速是 MySQL 的显著特性。在 MySQL 中，使用了极快的 B 树磁盘表（MyISAM）和索引压缩；通过使用优化的单扫描多连接，能够极快地实现连接；SQL 函数使用高度优化的类库实现，运行速度极快。图书馆书籍量大、读者人数

多，这就要求了查询时的高效运行速度，MySQL 的运行速度快契合这个图书馆管理系统要求。

4) 安全性高、成本低

灵活和安全的权限与密码系统，允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全，这对图书馆管理系统的数据做出了良好的保护。同时，MySQL 数据库是一种完全免费的产品，大大降低了图书馆管理系统的费用开销。

5) 支持各种开发语言

MySQL 为各种流行的程序设计语言提供支持，为它们提供了很多的 API 函数，包括 PHP、Java、Python、C、C++、Perl 语言等。本项目基于 Python 语言，在 Django 框架内简单调用 MySQL 包的函数，就可以实现开发所要求。

6) 数据库存储容量大

MySQL 数据库的最大有效表尺寸通常是由操作系统对文件大小的限制决定的，而不是由 MySQL 内部限制决定的。InnoDB 存储引擎将 InnoDB 表保存在一个表空间内，该表空间可由数个文件创建，表空间的最大容量为 64TB，可以轻松处理拥有上千万条记录的大型数据库。一般大型图书馆书籍数量大约 125 万册，总注册读者数大约在 30 万，因此对数据库存储容量有较高要求。MySQL 表空间最大容量 64TB 可以满足图书馆管理系统使用需求。

Django 框架：

1) 齐全的功能。

Django 框架自带大量常用工具，可轻松、迅速开发出一个功能齐全的 Web 应用。这对项目的前期准备与后续维护具有较大好处。

2) 强大的数据库访问组件。

Django 自带一个面向对象的、反映数据模型(以 Python 类的形式定义)与关系型数据库间的映射关系的映射器(ORM)，该项目可以通过设置 Django 中的 Setting 文件来连接本机的数据库，利用 Django 自带关系内置映射器访问所需的表，从而搜索相应内容。

3) 灵活的 URL 映射。

Django 提供一个基于正则表达式的 URL 分发器，开发者可清晰地编写 URL，这对设置图书馆管理系统界面的清晰跳转打下良好基础。

4) 完整的错误信息提示。

Django 提供了非常完整的错误信息提示和定位功能，可在开发调试过程中快速定位错误或异常。在图书馆管理系统设计时，可以通过对完整的错误信息的分析找出问题所作，并快速改正。

Bootstrap:

1) 预处理脚本

Bootstrap 的源码是基于最流行的 CSS 预处理脚本 Less 和 Sass 开发的。可以采用预编译的 CSS 文件快速开发，有很多同样的效果时，只需要写一个效果类。然后在用到的地方进行继承。在图书馆管理系统页面中，在 base 文件中，我实现了导航栏等常用模组，剩余具体的操作界面都继承了 base.html。

2) 一个框架，多种设备

图书馆管理系统界面能在 Bootstrap 的帮助下通过同一个代码，快速、有效适配手机，平板、PC 设备，同时由于 Bootstrap 的格栅化设计，常见分辨率下图书馆管理系统界面都不会跑偏。

3) 特效齐全

Bootstrap 中包含了丰富的 Web 组件，根据这些组件，可以快速的搭建一个漂亮、功能完备的图书馆管理系统。比如：下拉菜单、按钮组、按钮下拉菜单、导航、导航条、路径导航、分页、排版、缩略图、警告对话框、进度条、媒体对象等。同时自带了多个 jQuery 插件，这些插件为 Bootstrap 中的组件赋予了“生命”。其中包括：模式对话框、标签页、滚动条、弹出框等。

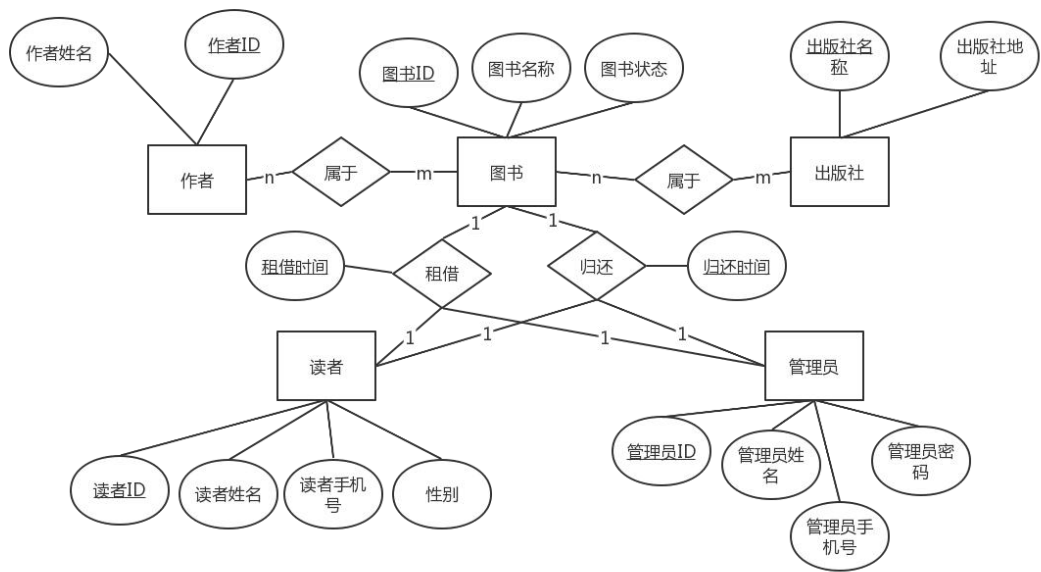
4) 开发高效:

由于 Bootstrap 的规范名称定义，规范化的项目开发流程，使得图书馆管理系统在开发时比较高效。同时，bootstrap 下的 css 代码更清晰简单，直接引用 css 后最终效果更好，同时省去大量编写 css 文件的时间，显著提高开发效率。

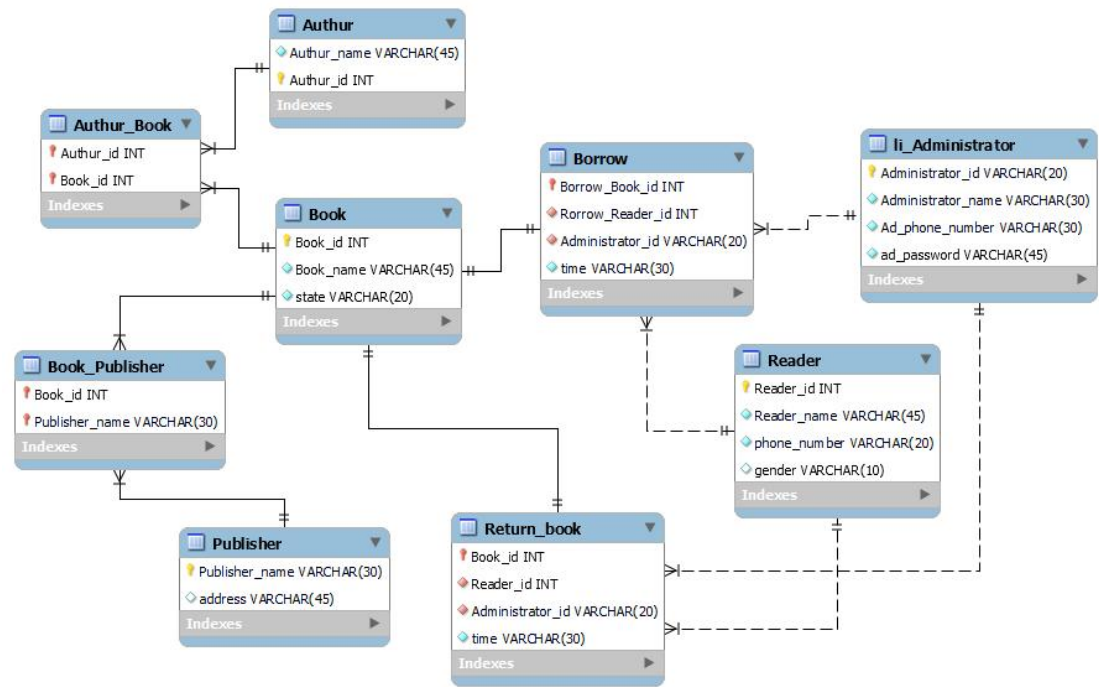
三、基本业务需求描述

1. 在图书管理系统中，管理员为每个读者建立一个账户，账户内存储读者个人的姓名，读者 ID 号，手机号，性别信息，并依据读者类别的不同给每个读者发放借书卡。读者 ID 号用于唯一地标识读者身份，每个读者对应一个读者 ID 号，同样每个读者 ID 号也不能对应多个读者。
2. 图书馆同名的图书会有多本，因此建立一个图书表，存放图书名称与图书 ID，图书 ID 号用于唯一地标识图书，每本图书对应多个图书 ID 号，但一个图书 ID 号不对应多本图书。
3. 图书有出版社信息，因此建立出版社表，用于存放出版社名称、出版社地址、出版社 ID。管理员可以对出版社信息进行增删改查，同时统计出版社的相关数据。
4. 图书有作者信息，因此建立作者表，用于存放作者姓名、作者 ID。管理员可以对作者信息进行增删改查，同时统计作者的相关数据。
5. 读者凭借读者借书卡的注册信息，在图书馆进行租借图书操作。但具体操作需要由管理员录入图书 ID 号，读者 ID 号，管理员 ID 号，操作时间信息，完成借书操作同时修改相应图书信息的状态。
6. 归还图书操作。但具体操作需要由管理员录入数目 ID 号，读者 ID 号，管理员 ID 号，操作时间信息，完成归还操作同时修改相应图书信息的状态。
7. 由于图书馆会不定期购进新书，也会下架损毁严重的读书，因此图书管理员不定期对图书信息进行添加，修改和删除、查询等操作
8. 由于读者会出现借书卡丢失或者手机号发生变动的情况，管理员也需要对读者的信息进行添加，删除，修改、查询等操作。
9. 从读者的角度，可以查阅图书馆相应的图书资料（也可以按要求查找，如输入一些关键字）以及每本书的在馆册数。
10. 由于图书馆会存在招聘新人与员工辞职的情况，因此需要图书馆管理员可以直接添加、修改、删除图书馆管理员信息。

四、通用的 E-R 模型(E-R 图)



五、基本"关系数据库 E-R 模型"



六、根据关系数据库范式理论进行分解优化

1. Book 表

列名	数据类型	是否为空	说明
Book_id	INT	否	图书 ID，主键
Book_name	VARCHAR(45)	否	图书名称
state	VARCHAR(20)	否	图书状态

2. Author 表

列名	数据类型	是否为空	说明
Author_id	INT	否	作者 ID，主键
Author_name	VARCHAR(45)	否	作者姓名

3. Author_Book 表

列名	数据类型	是否为空	说明
Author_id	INT	否	作者 ID，主键
Book_id	INT	否	图书 ID，主键

4. Publisher 表

列名	数据类型	是否为空	说明
Publisher_name	VARCHAR(30)	否	出版社名称，主键
Address	VARCHAR(45)	是	出版社地址

5. Book_Publisher 表

列名	数据类型	是否为空	说明
Book_id	INT	否	图书 ID，主键
Publisher_name	VARCHAR(30)	否	出版社名称，主键

6. Reader 表

列名	数据类型	是否为空	说明
Reader_id	INT	否	读者 ID，主键
Reader_name	VARCHAR(45)	否	读者姓名
Phone_number	VARCHAR(20)	否	读者手机号
Gender	VARCHAR(10)	是	读者性别

7. Li_Administrator 表

列名	数据类型	是否为空	说明
Administrator_id	VARCHAR(20)	否	管理员 ID, 主键
Administrator_name	VARCHAR(30)	否	管理员姓名
Ad_phone_number	VARCHAR(30)	否	管理员手机号
Ad_password	VARCHAR(45)	否	管理员密码

8. Borrow 表

列名	数据类型	是否为空	说明
Borrow_Book_id	INT	否	图书 ID, 主键
Borrow_Reader_id	INT	否	读者 ID, 主键
Administrator_id	VARCHAR(20)	否	管理员 ID, 主键
Time	VARCHAR(30)	否	操作时间

9. Return_Book 表

列名	数据类型	是否为空	说明
Book_id	INT	否	图书 ID, 主键
Reader_id	INT	否	读者 ID, 主键
Administrator_id	VARCHAR(20)	否	管理员 ID, 主键
Time	VARCHAR(30)	否	操作时间

1. 对于表 Book 而言, $Book_id \rightarrow Book_name$, $Book_id \rightarrow state$, $Book_id$ 是超码, 满足 BCNF 范式。
2. 对于表 Author 而言, $Author_id \rightarrow Author_name$, $Author_id$ 是超码, 满足 BCNF 范式。
3. 对于表 Author_Book 而言, $Book_id$ 与 $Author_id$ 均为超码, 满足 BCNF 范式。
4. 对于表 Publisher 而言, $Publisher_name \rightarrow Address$, $Publisher_name$ 是超码, 满足 BCNF 范式。
5. 对于表 Book_Publisher 而言, $Book_id$ 与 $Publisher_name$ 均为超码, 满足 BCNF 范式。
6. 对于 Reader 表而言, $Reader_id \rightarrow Reader_name$, $Reader_id \rightarrow Phone_number$,

Reader_id→Gender, Reader_id 是超码, 满足 BCNF 范式。

7. 对于表 Li_Administrator 而言, Administrator_id→Administrator_name, Administrator_id→Ad_phone_number, Administrator_id→Ad_password, Administrator_id 是超码, 满足 BCNF 范式。

8. 对于表 Borrow 而言, (Borrow_Book_id, Borrow_Reader_id, Administrator_id) → Time, Borrow_Book_id, Borrow_Reader_id, Administrator_id 做联合主键, 满足 BCNF 范式。

9. 对于表 Return_Book 表而言, (Book_id, Reader_id, Administrator_id) → Time, Book_id, Reader_id, Administrator_id 做联合主键, 满足 BCNF 范式。

七、全部 SQL DDL 语句

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- Schema mydb
-- -----
-- Schema mydb
-- -----
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- Table `mydb`.`Book`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`Book` (
  `Book_id` INT NOT NULL,
  `Book_name` VARCHAR(45) NOT NULL,
  `state` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`Book_id`))
ENGINE = InnoDB;

-- Table `mydb`.`Author`
-- -----
```



```

CREATE TABLE IF NOT EXISTS `mydb`.`Authur` (
  `Authur_name` VARCHAR(45) NOT NULL,
  `Authur_id` INT NOT NULL,
  PRIMARY KEY (`Authur_id`))
ENGINE = InnoDB;

-----
-- Table `mydb`.`Reader`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Reader` (
  `Reader_id` INT NOT NULL,
  `Reader_name` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(20) NOT NULL,
  `gender` VARCHAR(10) NULL,
  PRIMARY KEY (`Reader_id`))
ENGINE = InnoDB;

-----
-- Table `mydb`.`li_Administrator`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`li_Administrator` (
  `Administrator_id` VARCHAR(20) NOT NULL,
  `Administrator_name` VARCHAR(30) NOT NULL,
  `Ad_phone_number` VARCHAR(30) NOT NULL,
  `ad_password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Administrator_id`))
ENGINE = InnoDB;

-----
-- Table `mydb`.`Borrow`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Borrow` (
  `Borrow_Book_id` INT NOT NULL,
  `Rorrow_Reader_id` INT NOT NULL,
  `Administrator_id` VARCHAR(20) NOT NULL,
  `time` VARCHAR(30) NOT NULL,
  INDEX `fk_Borrow_Book_idx` (`Borrow_Book_id` ASC) VISIBLE,
  PRIMARY KEY (`Borrow_Book_id`),
  INDEX `fk_Borrow_Reader1_idx` (`Rorrow_Reader_id` ASC) VISIBLE,
  INDEX `fk_Borrow_Administrator1_idx` (`Administrator_id` ASC) VISIBLE,
  CONSTRAINT `fk_Borrow_Book`
    FOREIGN KEY (`Borrow_Book_id`)
    REFERENCES `mydb`.`Book` (`Book_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Borrow_Reader1`
    FOREIGN KEY (`Rorrow_Reader_id`)

```

```

REFERENCES `mydb`.`Reader` (`Reader_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Borrow_Administrator1`
FOREIGN KEY (`Administrator_id`)
REFERENCES `mydb`.`li_Administrator` (`Administrator_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb`.`Authur_Book`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Authur_Book` (
  `Authur_id` INT NOT NULL,
  `Book_id` INT NOT NULL,
  PRIMARY KEY (`Authur_id`, `Book_id`),
  INDEX `fk_Authur_has_Book_Book1_idx` (`Book_id` ASC) VISIBLE,
  INDEX `fk_Authur_has_Book_Authur1_idx` (`Authur_id` ASC) VISIBLE,
  CONSTRAINT `fk_Authur_has_Book_Authur1`
    FOREIGN KEY (`Authur_id`)
    REFERENCES `mydb`.`Authur` (`Authur_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Authur_has_Book_Book1`
    FOREIGN KEY (`Book_id`)
    REFERENCES `mydb`.`Book` (`Book_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb`.`Publisher`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Publisher` (
  `Publisher_name` VARCHAR(30) NOT NULL,
  `address` VARCHAR(45) NULL,
  PRIMARY KEY (`Publisher_name`))
ENGINE = InnoDB;

-----
-- Table `mydb`.`Book_Publisher`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Book_Publisher` (
  `Book_id` INT NOT NULL,
  `Publisher_name` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`Book_id`, `Publisher_name`),

```

```

        INDEX `fk_Book_has_Publisher_Publisher1_idx` (`Publisher_name` ASC)
VISIBLE,
        INDEX `fk_Book_has_Publisher_Book1_idx` (`Book_id` ASC) VISIBLE,
        CONSTRAINT `fk_Book_has_Publisher_Book1`
            FOREIGN KEY (`Book_id`)
            REFERENCES `mydb`.`Book` (`Book_id`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION,
        CONSTRAINT `fk_Book_has_Publisher_Publisher1`
            FOREIGN KEY (`Publisher_name`)
            REFERENCES `mydb`.`Publisher` (`Publisher_name`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb`.`Return_book`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Return_book` (
    `Book_id` INT NOT NULL,
    `Reader_id` INT NOT NULL,
    `Administrator_id` VARCHAR(20) NOT NULL,
    `time` VARCHAR(30) NOT NULL,
    INDEX `fk_Return_Book1_idx` (`Book_id` ASC) VISIBLE,
    PRIMARY KEY (`Book_id`),
    INDEX `fk_Return_Reader1_idx` (`Reader_id` ASC) VISIBLE,
    CONSTRAINT `fk_Return_Book1`
        FOREIGN KEY (`Book_id`)
        REFERENCES `mydb`.`Book` (`Book_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_Return_Administrator1`
        FOREIGN KEY (`Administrator_id`)
        REFERENCES `mydb`.`li_Administrator` (`Administrator_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_Return_Reader1`
        FOREIGN KEY (`Reader_id`)
        REFERENCES `mydb`.`Reader` (`Reader_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

八、各表数据展示

1. Book 表

Book_id	Book_name	state
1	深入理解计算机系统	在租
2	深入理解计算机系统	已还
3	深入理解计算机系统	在租
4	深入理解计算机系统	在租
5	深入理解计算机系统	已还
6	计算机网络	在租
7	计算机网络	在租
8	计算机网络	在租
9	计算机网络	已还
10	计算机网络	已还
11	计算机网络	已还

12	计算机网络	已还
13	数据库系统概念	在租
14	数据库系统概念	已还
15	数据库系统概念	在租
16	数据库系统概念	已还
17	数据库系统概念	在租
18	GitHub 从入门到精通	已还
19	GitHub 从入门到精通	在租
20	GitHub 从入门到精通	已还
21	GitHub 从入门到精通	在租
22	GitHub 从入门到精通	已还
23	GitHub 从入门到精通	已还

2. Author 表

Authur_name	Authur_id
Bryant,R.E.	10001

谢希仁	10002
Abraham Silberschatz	10003
高见龙	10004

3. Author_Book 表

Authur_id	Book_id
10001	1
10001	2
10001	3
10001	4
10001	5
10002	6
10002	7
10002	8
10002	9
10002	10
10002	11

10002	12
10003	13
10003	14
10003	15
10003	16
10003	17
10004	18
10004	19
10004	20
10004	21
10004	22
10004	23

4. Publisher 表

Publisher_name	address
北京大学出版社	北京市海淀区西大街 36 号昊海楼 213 号
机械工业出版社	北京市西城区百万庄大街 22 号
电子工业出版社	北京市海淀区万寿路 288 号华信大厦

5. Book_Publisher 表

Book_id	Publisher_name
1	机械工业出版社
2	机械工业出版社
3	机械工业出版社
4	机械工业出版社
5	机械工业出版社
6	电子工业出版社
7	电子工业出版社
8	电子工业出版社
9	电子工业出版社
10	电子工业出版社
11	电子工业出版社

12	电子工业出版社
13	机械工业出版社
14	机械工业出版社
15	机械工业出版社
16	机械工业出版社
17	机械工业出版社
18	北京大学出版社
19	北京大学出版社
20	北京大学出版社
21	北京大学出版社
22	北京大学出版社
23	北京大学出版社

6. Li_Administrator 表

Administrator_id	Administrator_name	Ad_phone_number	ad_password
14240220020226	李彦昊	15522081221	123456
14240220020227	邵天时	17720036688	123456
14240220020228	李朋瀚	13132074602	123456

7. Reader 表

Reader_id	Reader_name	phone_number	gender
2012283	曹烨	15222521023	女
2012284	陈嘉林	18630950066	男
2012285	王清铎	15222650049	男
2012286	张睿	18602602895	男
2012287	张子晔	18526513828	女

8. Return_Book 表

Book_id	Reader_id	Administrator_id	time
2	2012287	14240220020226	2022-6-4
5	2012287	14240220020227	2022-6-5
9	2012287	14240220020228	2022-6-6
10	2012286	14240220020226	2022-6-7
11	2012286	14240220020227	2022-6-8
12	2012286	14240220020228	2022-6-9
14	2012285	14240220020226	2022-6-10
16	2012285	14240220020227	2022-6-11
18	2012284	14240220020228	2022-6-12
20	2012284	14240220020226	2022-6-14
22	2012283	14240220020227	2022-6-13
23	2012283	14240220020228	2022-6-12

9. Borrow 表

Borrow_Book_id	Rorrow_Reader_id	Administrator_id	time
1	2012283	14240220020226	2022-6-4
3	2012283	14240220020227	2022-6-5
4	2012283	14240220020228	2022-6-6
6	2012284	14240220020226	2022-6-3
7	2012285	14240220020227	2022-6-7
8	2012285	14240220020228	2022-6-9
13	2012285	14240220020226	2022-6-19
15	2012286	14240220020227	2022-6-22
17	2012286	14240220020228	2022-6-11
19	2012287	14240220020226	2022-6-21
21	2012287	14240220020227	2022-6-9

九、使用视图的案例

在实际操作过程中，由于图书、出版社、作者的信息分别放置在三个表中，因此建立 book_and_publisher 视图与 book_and_author 视图，方便直接读取图书的整体信息。

```
create view book_and_publisher as
select
book.Book_id,book.Book_name,book.state,publisher.Publisher_name,publisher.address
from book,book_publisher,publisher
where book.Book_id = book_publisher.Book_id and
book_publisher.Publisher_name = publisher.Publisher_name
order by book.Book_id asc;
select *from book_and_publisher;
```

	Book_id	Book_name	state	Publisher_name	address
▶	1	深入理解计算机系统	在租	机械工业出版社	北京市西城区百万庄大街22号
	2	深入理解计算机系统	已还	机械工业出版社	北京市西城区百万庄大街22号
	3	深入理解计算机系统	在租	机械工业出版社	北京市西城区百万庄大街22号
	4	深入理解计算机系统	在租	机械工业出版社	北京市西城区百万庄大街22号
	5	深入理解计算机系统	已还	机械工业出版社	北京市西城区百万庄大街22号
	6	计算机网络	在租	电子工业出版社	北京市海淀区万寿路288号华信大厦
	7	计算机网络	在租	电子工业出版社	北京市海淀区万寿路288号华信大厦
	8	计算机网络	在租	电子工业出版社	北京市海淀区万寿路288号华信大厦
	9	计算机网络	已还	电子工业出版社	北京市海淀区万寿路288号华信大厦
	10	计算机网络	已还	电子工业出版社	北京市海淀区万寿路288号华信大厦
	11	计算机网络	已还	电子工业出版社	北京市海淀区万寿路288号华信大厦
	12	计算机网络	已还	电子工业出版社	北京市海淀区万寿路288号华信大厦
	13	数据库系统概念	在租	机械工业出版社	北京市西城区百万庄大街22号
	14	数据库系统概念	已还	机械工业出版社	北京市西城区百万庄大街22号
	15	数据库系统概念	在租	机械工业出版社	北京市西城区百万庄大街22号
	16	数据库系统概念	已还	机械工业出版社	北京市西城区百万庄大街22号
	17	数据库系统概念	在租	机械工业出版社	北京市西城区百万庄大街22号
	18	GitHub从入门到精通	已还	北京大学出版社	北京市海淀区西大街36号昊海楼213号
	19	GitHub从入门到精通	在租	北京大学出版社	北京市海淀区西大街36号昊海楼213号
	20	GitHub从入门到精通	已还	北京大学出版社	北京市海淀区西大街36号昊海楼213号
	21	GitHub从入门到精通	在租	北京大学出版社	北京市海淀区西大街36号昊海楼213号

(book_and_publisher 视图)

```
create view book_and_author as
select
book.Book_id,book.Book_name,book.state,author.Author_id,author.Author_name
from book,author_book,author
where book.Book_id = author_book.Book_id and author_book.Author_id =
author.Author_id
order by book.Book_id asc;
select *from book_and_author;
```


	Book_id	Book_name	state	Authur_id	Authur_name
▶	1	深入理解计算机系统	在租	10001	Bryant,R.E.
	2	深入理解计算机系统	已还	10001	Bryant,R.E.
	3	深入理解计算机系统	在租	10001	Bryant,R.E.
	4	深入理解计算机系统	在租	10001	Bryant,R.E.
	5	深入理解计算机系统	已还	10001	Bryant,R.E.
	6	计算机网络	在租	10002	谢希仁
	7	计算机网络	在租	10002	谢希仁
	8	计算机网络	在租	10002	谢希仁
	9	计算机网络	已还	10002	谢希仁
	10	计算机网络	已还	10002	谢希仁
	11	计算机网络	已还	10002	谢希仁
	12	计算机网络	已还	10002	谢希仁
	13	数据库系统概念	在租	10003	Abraham Silb...

(book_and_author 视图)

十、使用触发器的案例

在实际操作过程中,当发生租借时,可以通过创建触发器 book_check_borrow,直接将 book 表中的 state 修改为“在租”状态。

```
select *from book;
select *from borrow;
drop trigger if exists book_check_borrow;
DELIMITER $$
create trigger book_check_borrow after insert on borrow
  for each row
  begin
    update book,borrow set book.state = '在租' where book.Book_id =
borrow.Borrow_Book_id;
  end $$;
DELIMITER ;
insert into borrow values ('2','2012283','14240220020226','2002-6-20');
select *from book;
select *from borrow;
```

	Book_id	Book_name	state
▶	1	深入理解计算机系统	在租
	2	深入理解计算机系统	已还
	3	深入理解计算机系统	在租
	4	深入理解计算机系统	在租
	5	深入理解计算机系统	已还

(修改 borrow 前的 book 表)

	Book_id	Book_name	state
▶	1	深入理解计算机系统	在租
	2	深入理解计算机系统	在租
	3	深入理解计算机系统	在租
	4	深入理解计算机系统	在租
	5	深入理解计算机系统	已还

(修改 borrow 后的 book 表)

	Borrow_Book_id	Rorrow_Reader_id	Administrator_id	time
▶	1	2012283	14240220020226	2022-6-4
	2	2012283	14240220020226	2002-6-20
	3	2012283	14240220020227	2022-6-5
	4	2012283	14240220020228	2022-6-6

(修改前的 borrow 表)

	Borrow_Book_id	Rorrow_Reader_id	Administrator_id	time
▶	1	2012283	14240220020226	2022-6-4
	3	2012283	14240220020227	2022-6-5
	4	2012283	14240220020228	2022-6-6

(修改后的 borrow 表)

十一、使用存储函数、存储过程的案例

存储函数：

在实际使用过程中，由于读者对书本的喜爱程度不同，因此需要分析图书馆图书的数量后，采购常被借阅的书籍。这就需要有一个存储函数，用来判断是否是常被借阅书籍。

```
drop FUNCTION if exists BookLevel;
DELIMITER $$
CREATE FUNCTION BookLevel(book_sum INT,state varchar(30)) RETURNS
VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE lv1 varchar(20);
IF book_sum >= 4 and state = '在租' THEN
SET lv1 = '多人借阅';
ELSEIF book_sum < 4 and state = '在租' THEN
SET lv1 = '较长被借阅';
else
SET lv1 = '未被借阅';
END IF;
RETURN (lv1);
END $$
DELIMITER ;
drop view if exists countbook;
create view countbook as select *, count(Book_name) as num from book group
by Book_name,state;
select countbook.Book_name,countbook.state,num,BookLevel(num,state) from
countbook group by Book_name,state;
```

	Book_name	state	num	BookLevel(num,state)
▶	深入理解计算机系统	在租	4	多人借阅
	深入理解计算机系统	已还	1	未被借阅
	计算机网络	在租	3	较长被借阅
	计算机网络	已还	4	未被借阅
	数据库系统概念	在租	3	较长被借阅
	数据库系统概念	已还	2	未被借阅
	GitHub从入门到精通	已还	4	未被借阅
	GitHub从入门到精通	在租	2	较长被借阅

(调用存储函数 countbook)

由此可分析出《深入理解计算机系统》经常被借阅，需要进行补货。

存储过程：

与上述大致同样需求，但此时需求为输入一本书，来输出该书是否需要补货。

```
drop PROCEDURE if exists bookcheck;
DELIMITER $$
CREATE PROCEDURE bookcheck (in bookname varchar(30),out book_level
varchar(40))
    DETERMINISTIC
begin
    declare num int ;
    select count(book_name) as num from book where book.Book_name = bookname
and book.state = '在租' group by book.book_name,book.state into num;
    if num >= 4 then
        set book_level='需要补货';
    elseif num < 4 then
        set book_level='无需补货';
    else
        set book_level='无需补货';
    end if;
END $$;
DELIMITER ;
call bookcheck('深入理解计算机系统',@book_check);
select @book_check;
```




输入是《深入理解计算机系统》，输出需要补货。

```

83 • call bookcheck('数据库系统概念',@book_check);
84 • select @book_check;

```



The screenshot shows a SQL query result in a table with one column named '@book_check' and one row containing the value '无需补货' (No need to restock).

输入是《数据库系统概念》，输出无需补货。

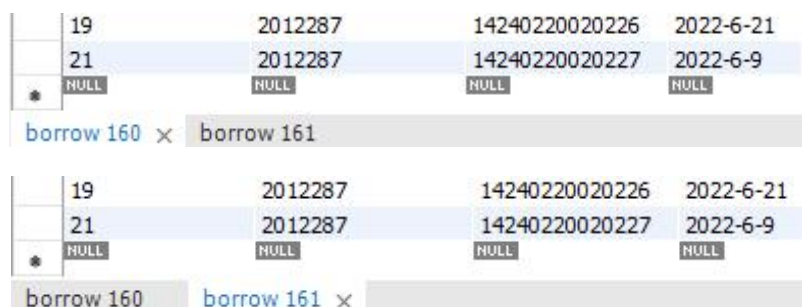
十二、需要事务处理的案例

图书馆如果突遇断电等可能会造成数据库事务发生错误的情况，此时进行回滚，保证数据没有损失。

```

select * from borrow;
start transaction;
insert into borrow values ('0023','2012287','14240220020227','2022-6-9');
rollback;
select * from borrow;

```



The first screenshot shows the state of the borrow table after the rollback operation. The table has two rows: one with values (19, 2012287, 14240220020226, 2022-6-21) and another with values (21, 2012287, 14240220020227, 2022-6-9). The second screenshot shows the state of the borrow table after the rollback operation, which is identical to the first screenshot.

id	book_id	reader_id	borrow_date
19	2012287	14240220020226	2022-6-21
21	2012287	14240220020227	2022-6-9

通过比对两张图，确认回滚成功。

十三、需要统计分析的案例

在实际操作过程中，图书馆管理员会分析图书数量，读者借阅次数，并绘制出柱状图，方便后续分析。

```

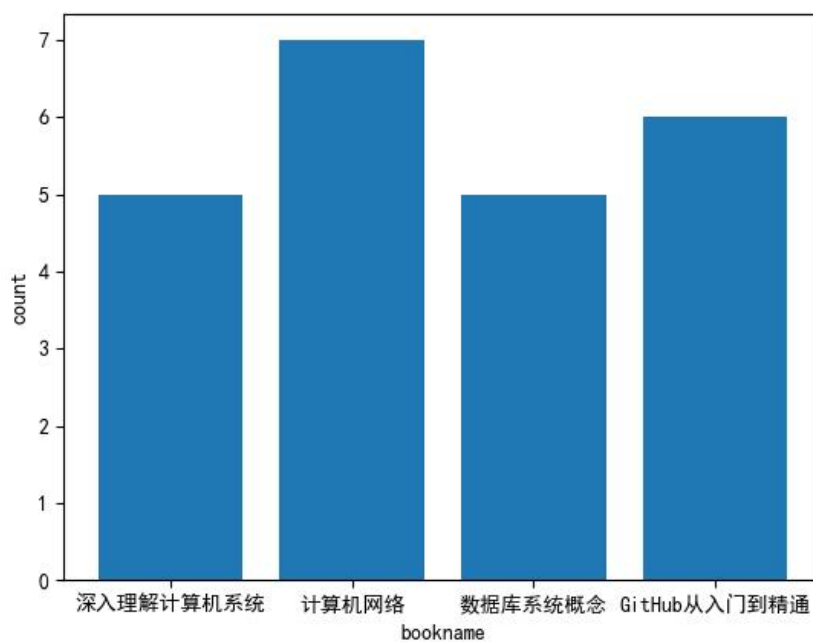
import pymysql
conn = pymysql.connect(host='localhost', user='root', password='root',
database='mydb', charset='utf8')
# 获取 cursor 对象
cursor = conn.cursor()

```

```

sql = "select *,count(book.Book_name) from book group by book.Book_name;
"
cursor.execute(sql)
retdata = cursor.fetchall()
index = []
book_name = []
for row in retdata:
    book_name.append(row[1])
    index.append(row[3])
conn.close()
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.xlabel("bookname")
plt.ylabel("count")
plt.bar(book_name,index)
plt.show()

```



(图书数量柱状图)

```

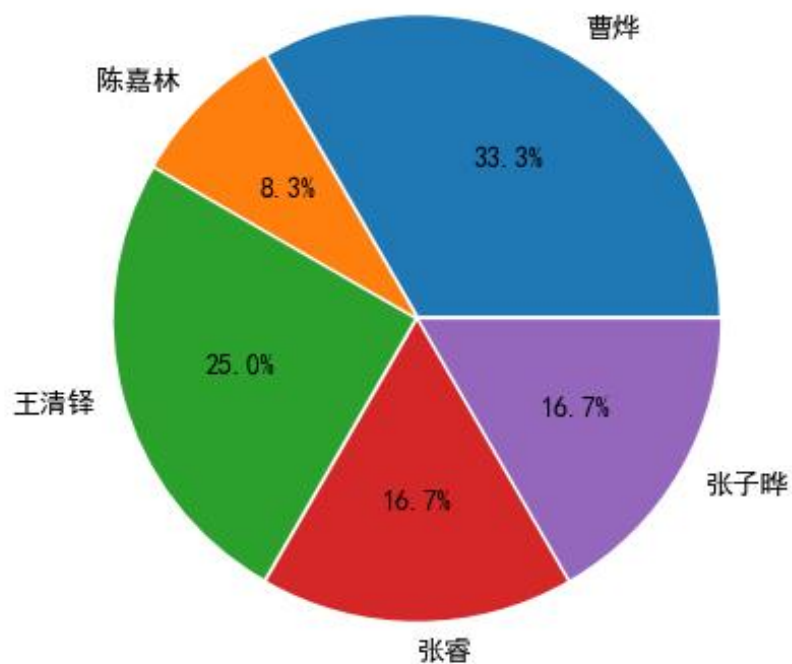
import pymysql
conn = pymysql.connect(host='localhost', user='root', password='root',
database='mydb', charset='utf8')
# 获取 cursor 对象
cursor = conn.cursor()

```

```

sql = "select reader.Reader_name,count(borrow.Rorrow_Reader_id) from
borrow,reader where borrow.Rorrow_Reader_id = reader.Reader_id group by
borrow.Rorrow_Reader_id; "
cursor.execute(sql)
retdat = cursor.fetchall()
index = []
Reader_name = []
for row in retdat:
    Reader_name.append(row[0])
    index.append(row[1])
conn.close()
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
explode=[0.01,0.01,0.01,0.01,0.01]#设定各项距离圆心 n 个半径
plt.pie(index,explode=explode,labels=Reader_name,autopct='%1.1f%%')
plt.show()

```



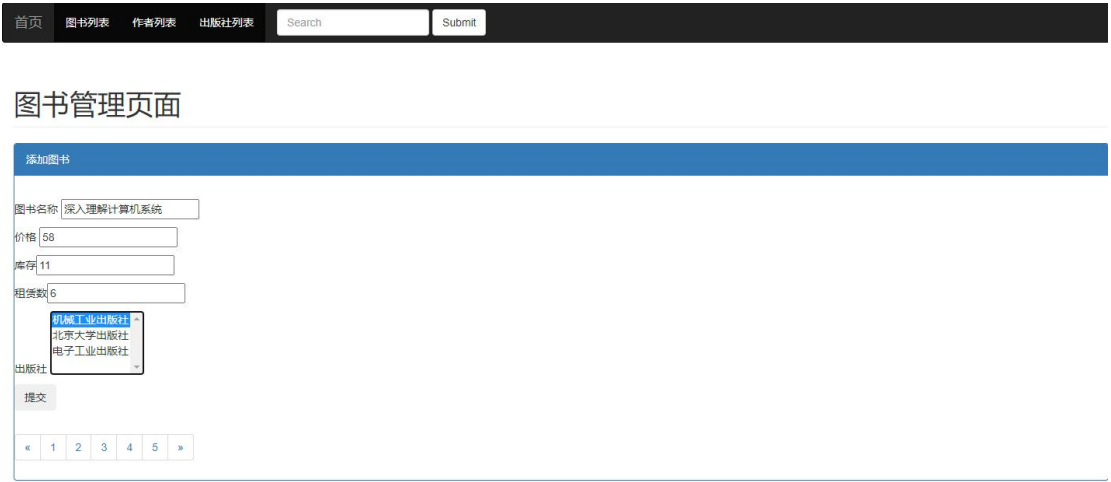
(读者借阅数量饼状图)

十四、完整应用系统介绍

1. 图书管理主页面



点击新增书籍，会跳转到新增书籍界面。



添加后如果需要修改，点击图书旁边的修改按键，即可修改图书信息。



点击删除按键，会删除该图书。例如删除《计算机网络》，效果如下。

图书管理页面

图书列表							
<input type="text" value="Search for..."/>				<input type="button" value="搜索"/>		<input type="button" value="新增图书"/>	
序号	图书id	图书名	价格	库存	卖出数	出版社	操作
1	1	深入理解计算机系统	65.00	11	5	机械工业出版社	修改 删除
2	2	数据库系统概念	85.00	5	4	机械工业出版社	修改 删除
3	3	GitHub从入门到精通	50.00	6	5	北京大学出版社	修改 删除
<div>« 1 2 3 4 5 »</div>							

2. 作者管理主界面

作者管理页面

作者列表				
<input type="text" value="Search for..."/>			<input type="button" value="搜索"/>	
			<input type="button" value="新增作者"/>	
序号	作者id	作者	图书	操作
1	1	Bryant,R.E.	深入理解计算机系统	修改 删除
2	2	谢希仁	计算机网络	修改 删除
3	3	Abraham Silberschatz	数据库系统概念	修改 删除
4	4	高见龙	GitHub从入门到精通	修改 删除
<div>« 1 2 3 4 5 »</div>				

如果需要新增作者，可以点击新增作者按钮，会跳转到增加界面。

作者管理页面

添加作者

作者

深入理解计算机系统

数据库系统概念

GitHub从入门到精通

计算机网络

▼

书名

提交

« 1 2 3 4 5 »

如果需要修改作者信息，直接点击修改，会跳转到修改界面。

作者管理页面

编辑作者

作者

书名

深入理解计算机系统

▼

提交

« 1 2 3 4 5 »

如果需要删除作者信息，直接点击删除。例如删除《GitHub 从入门到精通》

的作者，效果如下：

作者管理页面

作者列表				
<div>Search for...</div> <div>搜索</div>			<div>新增作者</div>	
序号	作者id	作者	图书	操作
1	1	Bryant, R. E.	深入理解计算机系统	修改 删除
2	2	谢希仁	计算机网络	修改 删除
3	3	Abraham Silberschatz	数据库系统概念	修改 删除
<div>« 1 2 3 4 5 »</div>				

3. 出版社管理主界面

出版社管理页面

出版社列表				
<div>Search for...</div> <div>搜索</div>			<div>新增出版社</div>	
序号	出版社id	名称	地址	操作
1	1	机械工业出版社	北京市西城区百万庄大街22号	修改 删除
2	2	北京大学出版社	北京市海淀区西大街36号昊海楼213号	修改 删除
3	3	电子工业出版社	北京市海淀区万寿路288号华信大厦	修改 删除
<div>« 1 2 3 4 5 »</div>				

如果需要新增出版社，可以点击新增出版社按钮，会跳转到增加界面。

添加出版社

出版社名称

北京大学出版社

出版社地址

北京市海淀区西大街36号昊海

提交

« 1 2 3 4 5 »

如果需要修改出版社信息，直接点击修改，会跳转到修改界面。

修改出版社

出版社名称:

机械工业出版社

出版社地址:

北京市西城区百万庄大街22

提交

« 1 2 3 4 5 »

如果需要删除出版社信息，直接点击删除。例如删除电子工业出版社，效果如下：

出版社列表				
Search for...			搜索	新增出版社
序号	出版社id	名称	地址	操作
1	1	机械工业出版社	北京市西城区百万庄大街22号	修改 删除
2	2	北京大学出版社	北京市海淀区西大街36号吴海楼213号	修改 删除
« 1 2 3 4 5 »				

4. 具体实现方法介绍

4 - 1 settings.py

通过设置 Django 框架下自带的 settings.py 文件，通过更改 setting 中的设置，连接本机中的 MySQL 数据库。代码如下：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': "library",
        "USER": "root",
        "PASSWORD": "root",
        "HOST": "127.0.0.1",
        "PORT": "3306",
    }
}
```

4 - 2 models.py

在 Django 框架下建立数据库表方式与原生 MySQL 建表方式有所不同，此时通过更改 models.py，声明 class 类，对应 MySQL 原生的 table 表。其中的元素就是 MySQL 原生的 Columns。

代码如下：

```
from django.db import models
# Create your models here.
class Publisher(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=64, null=False)
    address = models.CharField(max_length=64, null=False)
class Book(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=32)
    # 总共有 5 位, 小数站 2 位, 设置 default 因为已经存在的数据没有此字段
    price = models.DecimalField(max_digits=5, decimal_places=2,
default=10.01)
```

```

        inventory = models.IntegerField(verbose_name="库存数")
        sale_num = models.IntegerField(verbose_name="出租数")
        publisher = models.ForeignKey(to='Publisher',
on_delete=models.CASCADE)
class Author(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=32)
    book = models.ManyToManyField(to='Book')

```

4 - 3 urls.py

所有界面之间的链接跳转都是通过 Django 框架下的 urls.py 文件, 利用 Django 自带 urls 跳转关系实现界面切换。具体 urls.py 如下:

App01\urls.py:

```

from django.urls import path
from app01 import views
urlpatterns = [
    path("add_publisher/", views.add_publisher),
    path("publisher_list/", views.publisher_list),
    path("edit_publisher/", views.edit_publisher),
    path("delete_publisher/", views.delete_publisher),
    path("book_list/", views.book_list),
    path("add_book/", views.add_book),
    path("edit_book/", views.edit_book),
    path("delete_book/", views.delete_book),
    path("author_list/", views.author_list),
    path("add_author/", views.add_author),
    path("edit_author/", views.edit_author),
    path("delete_author/", views.delete_author),
]

```

BookShop\urls:

```

from django.contrib import admin
from django.urls import path, include
from app01.views import book_list
urlpatterns = [
    path('admin/', admin.site.urls),
    path('app01/', include("app01.urls")),
]

```

```
    path('', book_list),  
]
```

4 - 4 views.py

各个界面的功能都放置于这个文件中，由于该文件代码较多，在此只做部分展示，剩余代码请查看附件 2 的具体代码。

以图书管理界面为例：

```
def book_list(request):  
    # 获取图书信息  
    book_obj_list = models.Book.objects.all()  
    # 2.将数据放到页面上  
    return render(request, 'book_list.html', {"book_obj_list":  
book_obj_list})  
  
def add_book(request):  
    if request.method == 'POST':  
        # 1 获取表单提价过来的内容  
        name = request.POST.get('name')  
        price = request.POST.get("price")  
        inventory = request.POST.get("inventory")  
        sale_num = request.POST.get("sale_num")  
        publisher_id = request.POST.get('publisher_id')  
        # 2.保存到数据库 app01_book  
        models.Book.objects.create(name=name, price=price,  
inventory=inventory, sale_num=sale_num,  
                                publisher_id=publisher_id)  
        # 3.重定向到图书列表页面  
        return redirect('/app01/book_list/')  
    else:  
        # 1 获取所有出版社  
        publisher_obj_list = models.Publisher.objects.all()  
        return  
render(request, "add_book.html", {"publisher_obj_list": publisher_obj_list  
)  
  
def edit_book(request):  
    if request.method == 'GET':  
        # 1.获取 id  
        id = request.GET.get('id')  
        # 2.取数据库中查找相应数据  
        book_obj = models.Book.objects.filter(id=id).first()  
        # 3.查找所有的出版社
```

```

        publisher_list = models.Publisher.objects.all()
        # 4 返回页面
        return render(request, 'edit_book.html',
                        {'book_obj': book_obj, 'publisher_list':
publisher_list})
    else:
        # 1. 获取表单提交过来的内容
        id = request.POST.get('id')
        name = request.POST.get('name')
        inventory = request.POST.get("inventory")
        price = request.POST.get("price")
        sale_num = request.POST.get("sale_num")
        publisher_id = request.POST.get('publisher_id')
        # 2. 查询数据库进行更新
        models.Book.objects.filter(id=id).update(name=name,
inventory=inventory, price=price, sale_num=sale_num,
                                                publisher_id=publisher_id
)

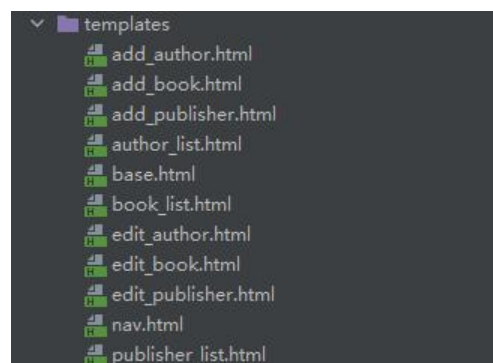
        # 3. 重定向到 boo_list
        return redirect("/app01/book_list")

def delete_book(request):
    # 1. 获取 id
    id = request.GET.get("id")
    # 2. 删除图书
    models.Book.objects.filter(id=id).delete()
    # 重定向到图书列表
    return redirect("/app01/book_list")

```

4 - 5 templates 文件夹

所有的前端文件都放入了 templates 文件夹中，其中的 base.html 与 nav.html 文件是后续各个界面的基础，后续界面都继承于该界面。



(templates 文件夹展示)

十五、性能与安全思考

1. MySQL 安全性

灵活和安全的权限与密码系统，允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全，这对图书馆管理系统的数据做出了良好的保护。

2. MySQL 效率

高速是 MySQL 的显著特性。在 MySQL 中，使用了极快的 B 树磁盘表（MyISAM）和索引压缩；通过使用优化的单扫描多连接，能够极快地实现连接；SQL 函数使用高度优化的类库实现，运行速度极快。图书馆书籍量大、读者人数多，这就要求了查询时的高效运行速度，MySQL 的运行速度快契合这个图书馆管理系统要求。

3. Django 效率

Django 自带一个面向对象的、反映数据模型（以 Python 类的形式定义）与关系型数据库间的映射关系的映射器（ORM），该项目可以通过设置 Django 中的 Setting 文件来连接本机的数据库，利用 Django 自带关系内置映射器访问所需的表，从而搜索相应内容，这为项目的前期开发大大节省时间。