

**本科数据库实训课程设计与开发报告**

|  |  |
| --- | --- |
| **学 号：** | **181340638** |
| **学生姓名：** | **刘硕** |
| **所在学院：** | **信息管理学院** |
| **专 业：** | 计算机科学与技术 |
| **班 级：** | **6班** |
| **得 分：** |  |

2021年 5 月 20日

图书馆借书数据库系统的设计与实现

# 1 系统分析

## 1.1 需求分析

### 1.1.1需求分析描述

随着计算机的普及和信息技术的发展，人们的生活发生了日新月异的变化，各类计算机软件逐渐渗透到了社会的每个角落，大大地改善了人们的生活质量，提高了人们的工作效率。在高校中，图书借阅是学生获取知识的一个很重要的途径，如何既能方便学生借书，又能减轻图书馆管理人员的工作负担，高效地完成图书借阅管理工作，是一件非常重要的事情。

X高校拥有一个小型图书馆，为全校师生提供学习、阅读的空间。近几年来，随着生源的不断扩大，图书馆的规模也随之扩大，图书数量也相应地打量增加，有关图书借阅的各种信息成倍增加。面对如此巨大的信息量，图书馆管理人员很难支撑，因此，学校领导决定建立一套合理实用的图书借阅管理系统软件，以对校内的图书借阅信息进行统一、集中的管理。

图书借阅管理系统，能够实现计算机化的图书借阅管理，能够提供方便快速的图书信息检索功能和便捷的图书借阅和归还功能，并且能够对图书信息和读者信息进行管理，方便管理员和读者的借阅处理。要求系统具备以下特点：要求系统具备以下特点：

（1）操作简单便捷，设计合理，利于人的理解。

（2）数据安全性较高，可靠性较高

（3）要求系统的功能较为完善，有利于人机交互

### 1.1.2系统范围

（1） 图书信息管理：加入，删除、修改或查找图书信息

（2） 图书管理员信息管理：加入，删除、修改或查找图书管理员信息

（3） 借书信息管理：加入，删除、修改或查找借书信息

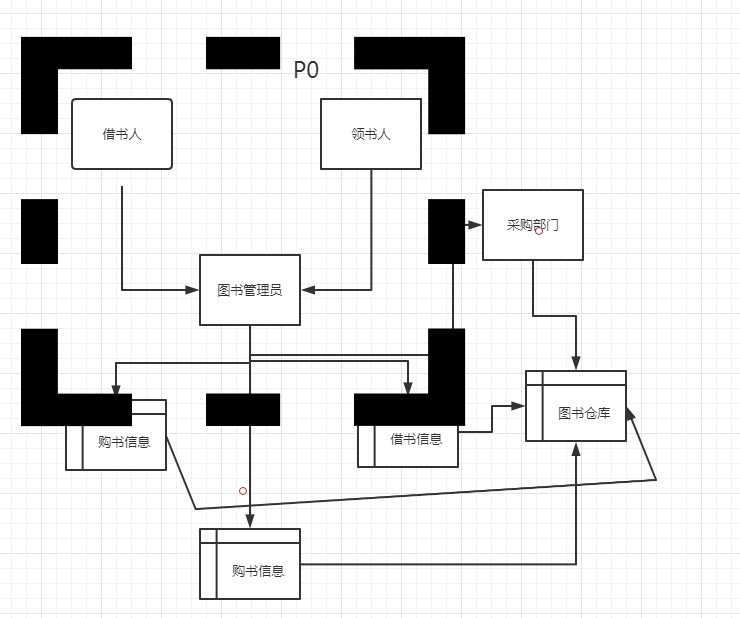
（4） 学生信息管理：加入，删除、修改或查找学生信息

### 1.1.3业务流程图

### 

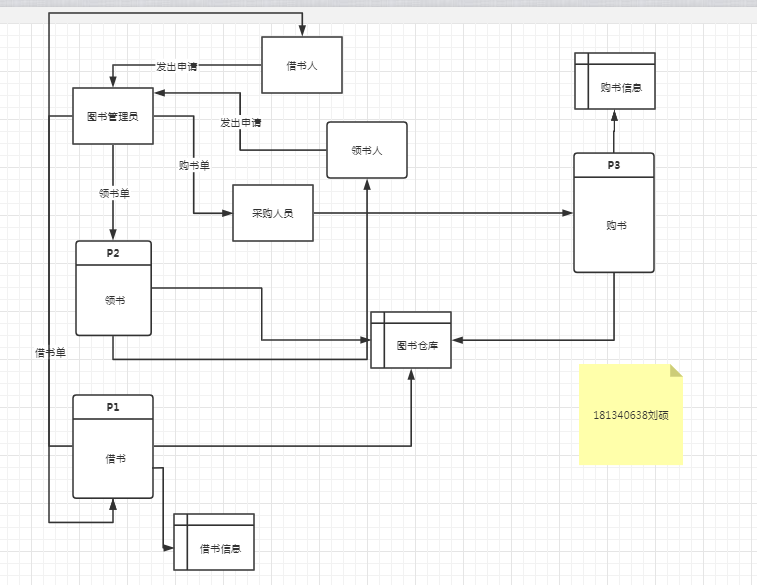
图1.1.3

### 1.1.4数据流图DFD

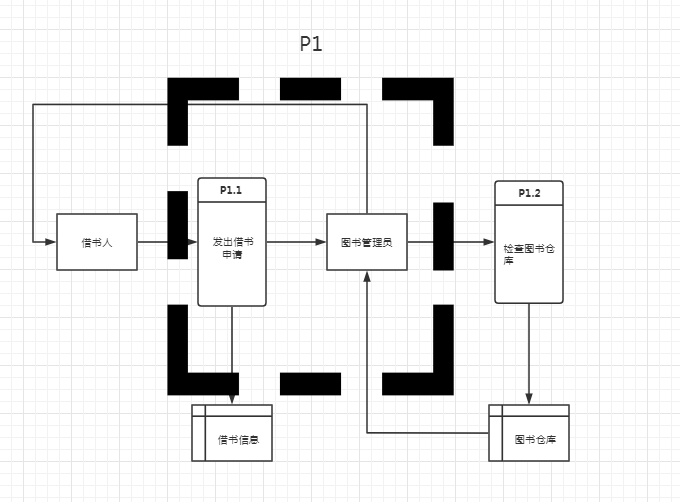


**图1.1.4数据流图（顶层）**

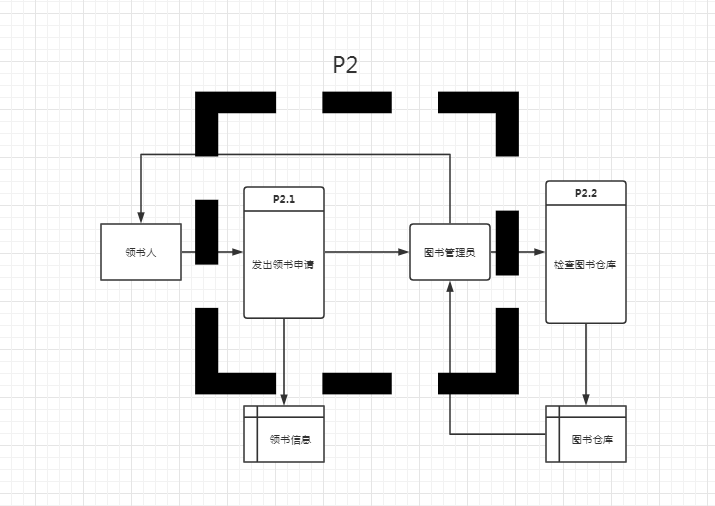
**第一层**



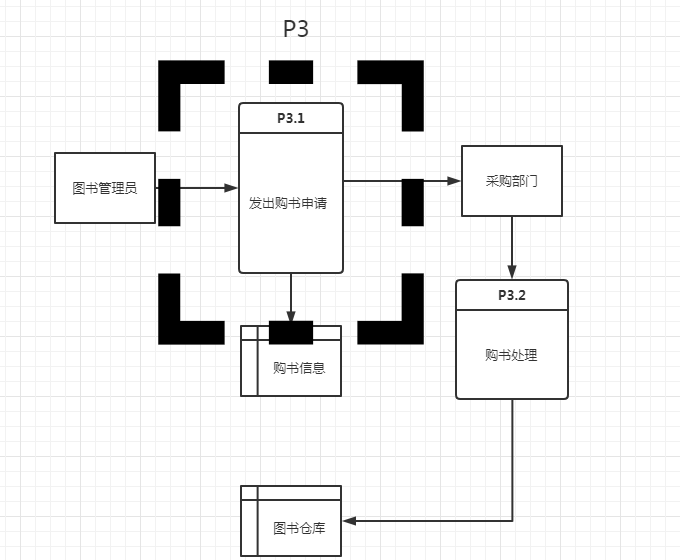
**图1.1.4数据流图（第一层）**



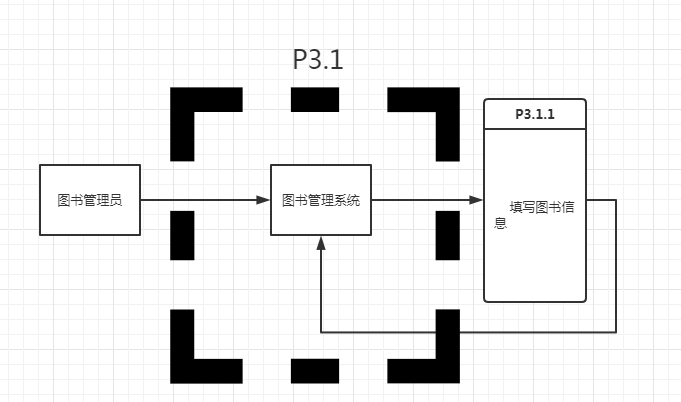
**图1.1.4数据流图（第二层）**



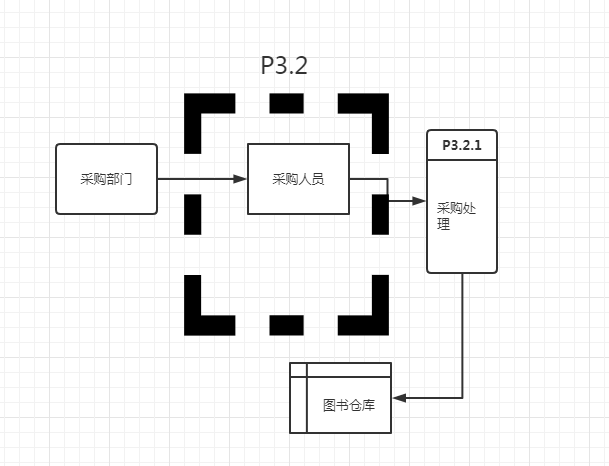
**图1.1.4数据流图(第二层)**



**图1.1.4数据流图(第二层)**



**图1.1.4数据流图(第三层)**



**图1.1.4数据流图(第三层)**

### 1.1.5数据字典DD

**数据项：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据项名称 | 含义说明 | 别号 | 类型 | 长度 | 取值范围 | 取值含义 |
| 用户编号 | 唯一标识每个用户 | 用户号 | 整型 | 6 | 000000-999999 | 给用户按顺序编号 |
| 员工编号 | 唯一标识每个员工 | 员工号 | 整型 | 6 | 000000-  999999 | 给员工按顺序编号 |
| 图书编号 | 唯一标识每个图书 | 图书号 | 整型 | 6 | 000000-999999 | 给图书按顺序编号 |
| 用户名称 | 用户名字 | 用户名 | 字符型 | 12 |  | 记录用户名称 |
| 员工名称 | 员工名字 | 员工名 | 字符型 | 12 |  | 记录员工名称 |
| 图书名称 | 图书名字 | 图书名 | 字符型 | 12 |  | 记录图书名称 |
| 性别 | 人的性别 | 性别 | 字符型 | 1 | 男，女，无 | 记录人的性别 |
| 身份证号 | 人的身份证号 | 身份证 | 字符型 | 18 | -2^18-2^18-1 | 记录人的身份证号 |

**数据结构：**

|  |  |  |
| --- | --- | --- |
| 数据结构名 | 含义 | 组成 |
| 用户 | 定义了一个用户的相关信息 | 用户编号+用户名字+性别+电话+身份证号 |
| 员工 | 定义了一个员工的相关信息 | 员工编号+员工名字+性别+电话+身份证号 |
| 图书 | 定义了一个图书的相关信息 | 图书编号+图书名称+出版社名称 |

**数据流：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据流名称 | 来源 | 去向 | 组成 | 平均流量 | 高峰期流量 |
| 借书 | 用户编号+用户名称+员工编号 | 图书编号+图书名称 | 员工编号+用户编号+图书编号+图书名称 | 6份/小时 | 12份/小时 |
| 领书 | 用户编号+用户名称+员工编号 | 图书编号+图书名称 | 员工编号+用户编号+图书编号+图书名称 | 8份/小时 | 15份/小时 |
| 购书 | 员工编号+员工名称 | 图书编号+图书名称 | 员工编号+图书编号+图书名称 | 15份/小时 | 30份/小时 |

**数据存储：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据存储名称 | 说明 | 流入数据流 | 流出数据流 | 组成 | 数据量 | 存储方式 |
| 用户登记表 | 记录用户的基本情况 |  |  | 用户编号+用户名称+性别+电话+身份证号 | 4000 | 随机存取 |
| 员工登记表 | 记录员工的基本情况 |  |  | 员工编号+员工名称+性别+电话+身份证号 | 50 | 随机存取 |
| 图书登记表 | 记录图书的基本情况 |  |  | 图书编号+图书名称 | 40000 | 随机存取 |

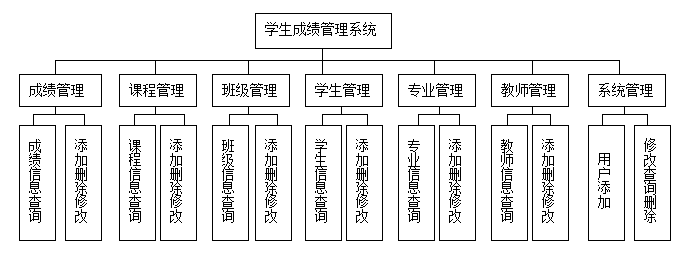
**处理过程：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 处理过程名称 | 说明 | 输入 | 输出 | 处理 |
| 借书 | 用户进行借书 | 用户+图书 | 借书处理 | 从图书库中调取所借书给用户 |
| 领书 | 用户进行领书 | 用户+图书 | 领书处理 | 从图书库中调取所领书给用户 |
| 购书 | 员工进行购书 | 员工+图书 | 购书处理 | 把从外面购买的书登记在图书库里，并及时更新信息 |

## 1.2 功能结构

*{系统功能模块描述。}*

系统功能模块图如图1.1所示。



成绩信息查询

图1.1 系统功能模块图

### 1.2.1 成绩管理模块

本模块实现有关成绩基本信息的录入、修改、查询与删除。同时能够实现按照课程代码或者学生学号进行查询并且当登陆身份为学生时，只能查询自己的成绩，当身份为教师时，可以录入学生成绩和查询成绩的功能，但是不可以修改和删除学生成绩。

1.2.2 课程管理模块

对各专业课程信息进行添加、删除、修改同时能够实现按照课程编号、课程名称进行查询。并且当登陆用户身份为学生或者任课教师时，只能进行查询课程信息操作。

### 1.2.3 班级管理模块

对班级信息进行添加、删除和修改，并且当登陆用户身份是学生或这任课教师时，只能进行查询班级信息的操作。

### 1.2.4 学生管理模块

有关学生基本信息的录入、修改查询和删除。同时能够实现按照学生的学号、姓名等条件进行查询，并且登陆用户身份是学生或者任课教师时只能进行学生信息查询。

### 1.2.5 专业管理模块

有关专业基本信息的录入、修改查询和删除。同时能够实现按照专业代码或专业名称进行查询，并且登陆身份是学生或者任课教师时，只能进行查询操作。

### 1.2.6 教师管理模块

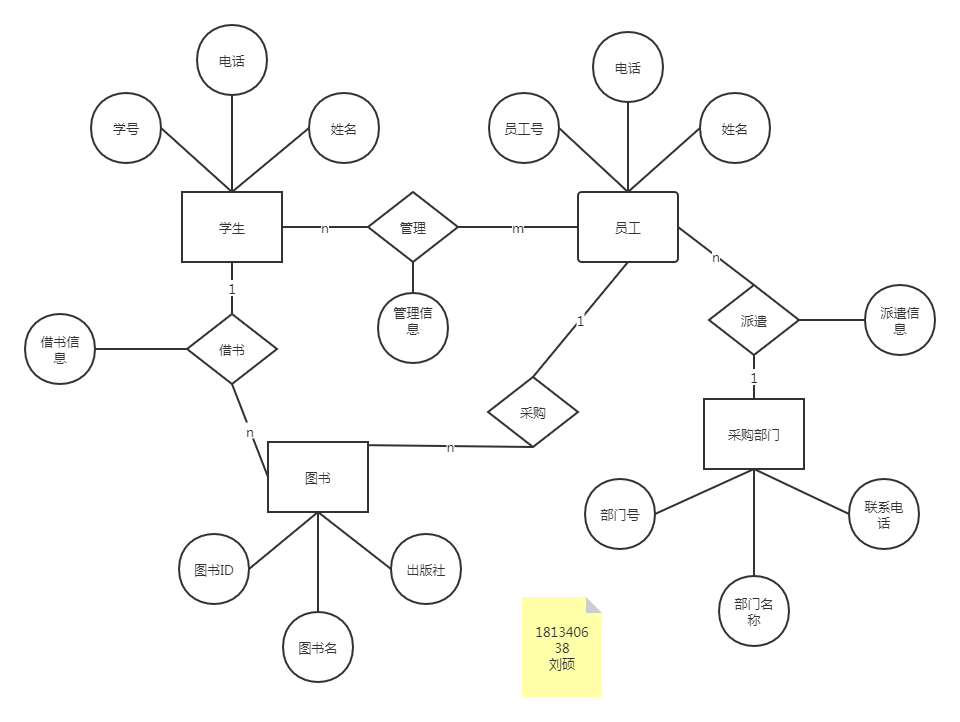
有关教师基本信息的录入、修改查询和删除。同时能够实现按照教师号、姓名等信息进行查询，并且登陆用户为学生时不能对此模块进行任何操作。

### 1.2.7 系统管理模块

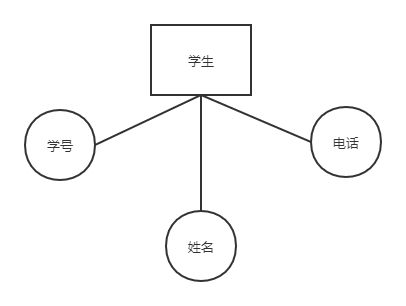
有系统管理员身份的用户可以登陆，负责用户管理。对用户进行添加修改、查询和删除功能。

# 2 数据库设计

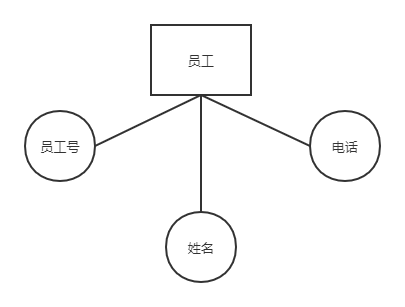
## 2.1概念设计



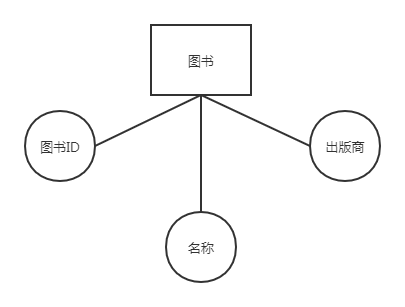
**图1-1 总ER图**

****

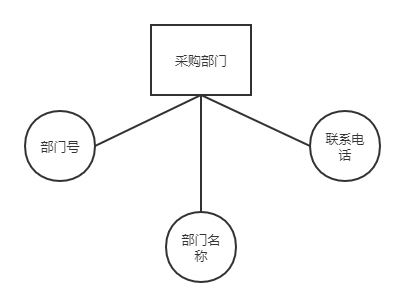
**图1-2 学生基本信息实体图**

****

**图1-3 员工基本信息实体图**

****

**图1-4 图书基本信息实体图**

****

**图1-5 采购部门基本信息实体图**

## 2.2 逻辑设计

### 学生关系模式

Student(Sno, Sname, Stel, Cno)

1. Sno: 学生学号（主码）
2. Sname：学生姓名
3. Stel：学生电话
4. Cno: 员工号（外码）

### 图书关系模式

Book(Bid, Bname, Bpub, Sno)

1. Bid: 图书ID（主码）
2. Bname: 图书名字
3. Bpub: 图书出版社
4. Sno: 学生学号（外码）

### 员工关系模式

Clark(Cno, Cname, Ctel, Dno)

1. Cno: 员工号（主码）
2. Cname: 员工姓名
3. Ctel: 员工电话
4. Dno: 部门号（外码）

### 采购部门关系模式

Department(Dno, Dname, Dtel)

1. Dno: 部门号(主码)
2. Dname: 部门名字
3. Dtel: 部门联系电话

## 2.3 物理设计



**图2.1学生信息表**



**图2.2 图书信息表**



**图2.3 员工信息表**



**图2.4 采购部门信息**

| **字段名** | **数据类型** | **长度** | **约束** | **名称** |
| --- | --- | --- | --- | --- |
| ID | int | 8 | 主键 | id |
| Sno | int | 8 | 外键 | 学生学号 |
| Cno | int | 8 | 外键 | 员工号 |

**图2.5 管理学生表**

# 3 数据库实现

## 3.1 DBMS的选择

选择了HeidiSQL

## 

由于使用的数据库较小，使用Navicat有点大材小用，选择了适合较小数据库的软件HeidiSQL

## 3.2 数据库的创建

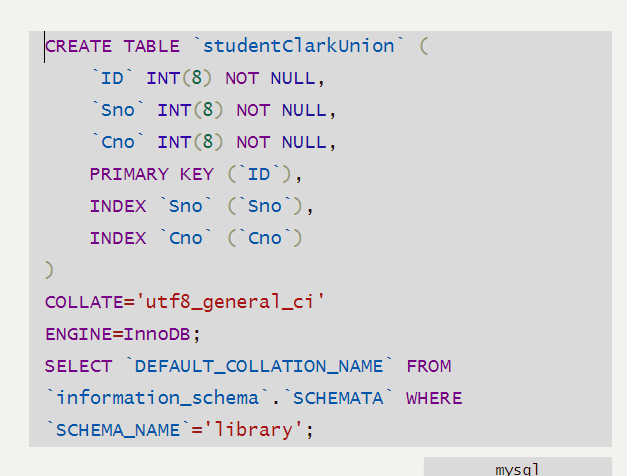
## 创建library数据库

CREATE DATABASE `Library` /\*!40100 COLLATE 'utf8\_general\_ci' \*/

MySQL自动在其所在路径的data文件夹下生成mysql.txt作为日志

## 3.3基本表的创建

创建studentClarkUnion表

cc

CREATE TABLE `studentClarkUnion` (

`ID` INT(8) NOT NULL,

`Sno` INT(8) NOT NULL,

`Cno` INT(8) NOT NULL,

PRIMARY KEY (`ID`),

INDEX `Sno` (`Sno`),

INDEX `Cno` (`Cno`)

)

COLLATE='utf8\_general\_ci'

ENGINE=InnoDB;

SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

创建Student表

CREATE TABLE `Student` (  
`Sno` INT(8) NOT NULL,  
`Sname` VARCHAR(20) NOT NULL,  
`Stel` VARCHAR(11) NOT NULL,  
`Cno` INT(8) NOT NULL,  
PRIMARY KEY (`Sno`),  
INDEX `Sname` (`Sname`),  
INDEX `Stel` (`Stel`),  
INDEX `Cno` (`Cno`)  
)  
COLLATE='utf8\_general\_ci'  
ENGINE=InnoDB;  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

创建Book表

CREATE TABLE `Book` (  
`Bid` INT(8) NOT NULL,  
`Bname` VARCHAR(20) NOT NULL,  
`Bpub` VARCHAR(20) NOT NULL,  
`Sno` INT(8) NOT NULL,  
PRIMARY KEY (`Bid`),  
INDEX `Bpub` (`Bpub`),  
INDEX `Sno` (`Sno`),  
INDEX `Bname` (`Bname`)  
)  
COLLATE='utf8\_general\_ci'  
ENGINE=InnoDB;  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

创建Clark表

CREATE TABLE `Clark` (  
`Cno` INT(8) NOT NULL,  
`Cname` VARCHAR(20) NOT NULL,  
`Ctel` VARCHAR(11) NOT NULL,  
`Dno` INT(8) NOT NULL,  
PRIMARY KEY (`Cno`),  
INDEX `Cname` (`Cname`),  
INDEX `Ctel` (`Ctel`),  
INDEX `Dno` (`Dno`)  
)  
COLLATE='utf8\_general\_ci'  
ENGINE=InnoDB;  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

创建Department表

CREATE TABLE `Department` (  
`Dno` INT(8) NOT NULL,  
`Dname` VARCHAR(20) NOT NULL,  
`Dtel` VARCHAR(11) NOT NULL,  
PRIMARY KEY (`Dno`),  
INDEX `Dname` (`Dname`),  
INDEX `Dtel` (`Dtel`)  
)  
COLLATE='utf8\_general\_ci'  
ENGINE=InnoDB;  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

## 3.4索引的创建

### 3.4.1 索引用来快速地寻找那些具有特定值的记录，所有MySQL索引都以B-树的形式保存。如果没有索引，执行查询时MySQL必须从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。表里面的记录数量越多，这个操作的代价就越高。

### 创建Sno关于Bpub, Bname以及它们各自的索引

ALTER TABLE `book`  
ADD INDEX `Sno` (`Sno`),  
ADD INDEX `Bpub` (`Bpub`, `Sno`),  
ADD INDEX `Bname` (`Bname`, `Sno`);  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

## 创建Dno关于Cname,Ctel以及它们各自的索引

ALTER TABLE `book`  
ADD INDEX `Sno` (`Sno`),  
ADD INDEX `Bpub` (`Bpub`, `Sno`),  
ADD INDEX `Bname` (`Bname`, `Sno`);  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

## 创建Cno关于Stel,Sname以及它们各自的索引

ALTER TABLE `student`  
ADD INDEX `Sname` (`Sname`, `Cno`),  
ADD INDEX `Stel` (`Stel`, `Cno`),  
ADD INDEX `Cno` (`Cno`);  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

## 创建Dtel关于Dname以及它们各自的索引

ALTER TABLE `department`  
ADD INDEX `Dname` (`Dname`, `Dtel`),  
ADD INDEX `Dtel` (`Dtel`);  
SELECT `DEFAULT\_COLLATION\_NAME` FROM `information\_schema`.`SCHEMATA` WHERE `SCHEMA\_NAME`='library';

## 3.5视图

### 3.5.1

一、可以简化数据查询语句

比如这些数据来源于多个基本表结构，或者数据不仅来自基本表，还有一部分数据来源于其他视图，并且检索条件又比较复杂时，需要编写的查询语句就会比较繁琐，此时定义视图可以使数据的查询语句变得简单可行。

定义视图可以将表与表之间的复杂的连接和搜索条件对用户不可见，因而可以简化操作

二、可以让用户能从多角度看待同一数据

视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户共享同一个数据库时，这种灵活性是非常必要的。

## 搜索名为三峡的书

SHOW CREATE VIEW `library`.`book\_sanxia`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/book\_sanxia.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='book\_sanxia' AND TABLE\_SCHEMA='library';

## 搜索ID小于181340638的学生

SHOW CREATE VIEW `library`.`student\_181340638`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/student\_181340638.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='student\_181340638' AND TABLE\_SCHEMA='library';

## 搜索ID小于20000000的员工

SHOW CREATE VIEW `library`.`clark\_20000000`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/clark\_20000000.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='clark\_20000000' AND TABLE\_SCHEMA='library';

## 搜索部门名为采购部门二的部门

SHOW CREATE VIEW `library`.`department\_2`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/department\_2.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='department\_2' AND TABLE\_SCHEMA='library';

## 搜索名为三峡的书

SHOW CREATE VIEW `library`.`book\_sanxia`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/book\_sanxia.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='book\_sanxia' AND TABLE\_SCHEMA='library';

## 搜索ID小于181340638的学生

SHOW CREATE VIEW `library`.`student\_181340638`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/student\_181340638.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='student\_181340638' AND TABLE\_SCHEMA='library';

## 搜索ID小于20000000的员工

SHOW CREATE VIEW `library`.`clark\_20000000`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/clark\_20000000.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='clark\_20000000' AND TABLE\_SCHEMA='library';

## 搜索部门名为采购部门二的部门

SHOW CREATE VIEW `library`.`department\_2`;  
SELECT LOAD\_FILE(CONCAT(IFNULL(@@GLOBAL.datadir, CONCAT(@@GLOBAL.basedir, 'data/')), 'library/department\_2.frm'));  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE   TABLE\_NAME='department\_2' AND TABLE\_SCHEMA='library';

# 4 数据库的初始化

## 4.1 数据初始化的方法

INSERT INTO `library`.`book` (`Bid`, `Bname`, `Bpub`, `Sno`) VALUES (00000001, '老人与海', '三峡出版社', 181340638);

INSERT INTO `library`.`student` (`Sno`, `Sname`, `Stel`, `Cno`) VALUES (181340638, 'ls', '18117032558', 00000001);

INSERT INTO `library`.`clark` (`Cno`, `Cname`, `Ctel`, `Dno`) VALUES (00000001, 'mxy', '18645441122', 00000001);

INSERT INTO `library`.`department` (`Dno`, `Dname`, `Dtel`) VALUES (00000001, 'department1', '00000000001');

## 4.2 Insert/LOAD/IMPORT语句

INSERT INTO `library`.`book` (`Bid`, `Bname`, `Bpub`, `Sno`) VALUES (00000001, '老人与海', '三峡出版社', 181340638);

INSERT INTO `library`.`student` (`Sno`, `Sname`, `Stel`, `Cno`) VALUES (181340638, 'ls', '18117032558', 00000001);

INSERT INTO `library`.`clark` (`Cno`, `Cname`, `Ctel`, `Dno`) VALUES (00000001, 'mxy', '18645441122', 00000001);

INSERT INTO `library`.`department` (`Dno`, `Dname`, `Dtel`) VALUES (00000001, 'department1', '00000000001');

# 5 增删改查

*{主要是查询。要求提供丰富的查询功能，不要只简单进行单表查询。必须有JOIN汇总分类统计分析的复杂查询，查询必须有现实意义。}*

## 5.1 增加数据

INSERT INTO `library`.`book` (`Bid`, `Bname`, `Bpub`, `Sno`) VALUES (00000001, '老人与海', '三峡出版社', 181340638);

INSERT INTO `library`.`student` (`Sno`, `Sname`, `Stel`, `Cno`) VALUES (181340638, 'ls', '18117032558', 00000001);

INSERT INTO `library`.`clark` (`Cno`, `Cname`, `Ctel`, `Dno`) VALUES (00000001, 'mxy', '18645441122', 00000001);

INSERT INTO `library`.`department` (`Dno`, `Dname`, `Dtel`) VALUES (00000001, 'department1', '00000000001');

## 5.2修改数据

UPDATE `library`.`book` SET `Bid`=2 WHERE `Bid`=1;

UPDATE `library`.`clark` SET `Cno`=2 WHERE `Cno`=1;

UPDATE `library`.`department` SET `Dno`=2 WHERE `Dno`=1;

UPDATE `library`.`student` SET `Sno`=181340637 WHERE `Sno`=181340638;

## 5.3删除数据

DELETE FROM `library`.`book` WHERE `Bid`=2;

DELETE FROM `library`.`clark` WHERE `Cno`=2;

DELETE FROM `library`.`department` WHERE `Dno`=2;

DELETE FROM `library`.`student` WHERE `Sno`=181340637;

## 5.4 简单查询

SELECT DISTINCT Cno FROM student group by Cno

select count(\*) from student where Cno='1' having count(\*) > 0

select Sno from student where Cno='1' group by Sno

## 5.5 复杂查询





1. select Sname,Cname from Student,Clark
2. select Sno,Cno from student where Cno<='3' and Cno>='1' group by Sno order by Cno asc
3. SELECT Sname,Cname FROM student INNER JOIN clark ON student.Cno=clark.Cno;
4. SELECT Sname,Cname FROM student left JOIN clark ON student.Cno=clark.Cno;
5. SELECT Sname,Cname FROM student cross join clark order by Cname
6. select \* from Student where Cno = (select Cno from clark where Cname = 'scy')
7. select \* from(

student inner join clark on student.Cno=clark.Cno

)

inner join department;

# 6 存储过程

## 6.1 存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。在数据量特别庞大的情况下利用存储过程能达到倍速的效率提升。

我创建了查询名为ls的学生的id， 更新ls为ls1, 输出所有学生信息

1.查询名为ls的学生的id

CREATE DEFINER=`root`@`localhost` PROCEDURE `getStudentLSId`()

BEGIN

SELECT Sno from student where Sname='ls';

END

EXEC getStudentLSId

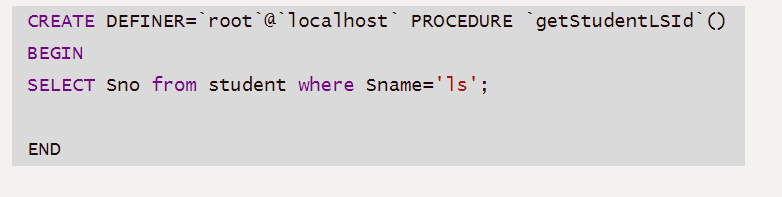
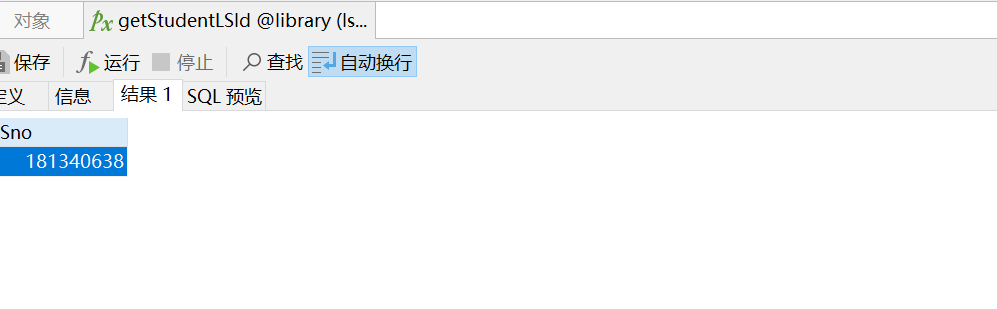
图6.1

图6.2

2. 更新ls为ls1

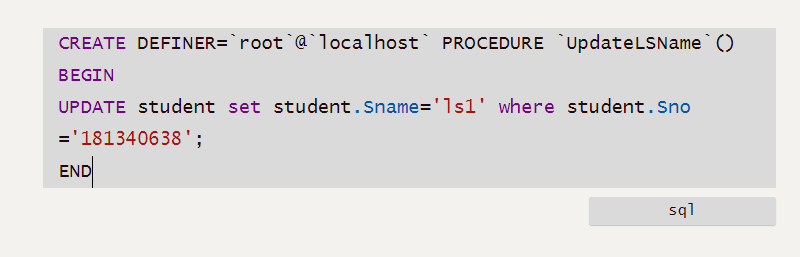
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateLSName`()

BEGIN

UPDATE student set student.Sname='ls1' where student.Sno ='181340638';

END

EXEC UpdateLSName

图6.3

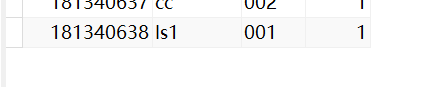


图6.4

3.输出所有学生的信息

CREATE DEFINER=`root`@`localhost` PROCEDURE `PrintStudentInformation`()

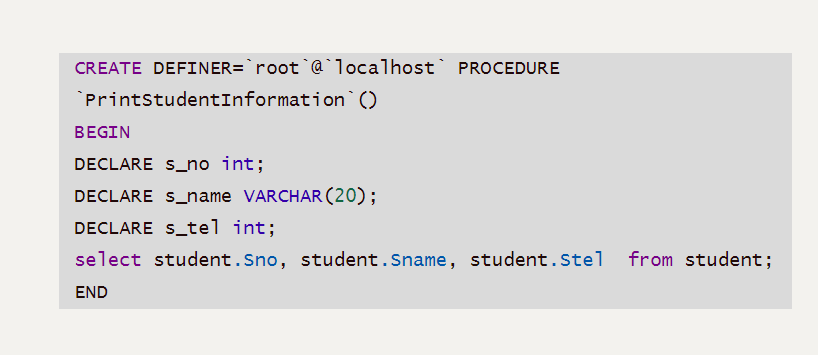
BEGIN

DECLARE s\_no int;

DECLARE s\_name VARCHAR(20);

DECLARE s\_tel int;

select student.Sno, student.Sname, student.Stel from student;

END

**6 函数**

1.通过学号获得学生姓名

CREATE DEFINER=`root`@`localhost` FUNCTION `GetSname`(`s\_no` int) RETURNS varchar(50) CHARSET utf8

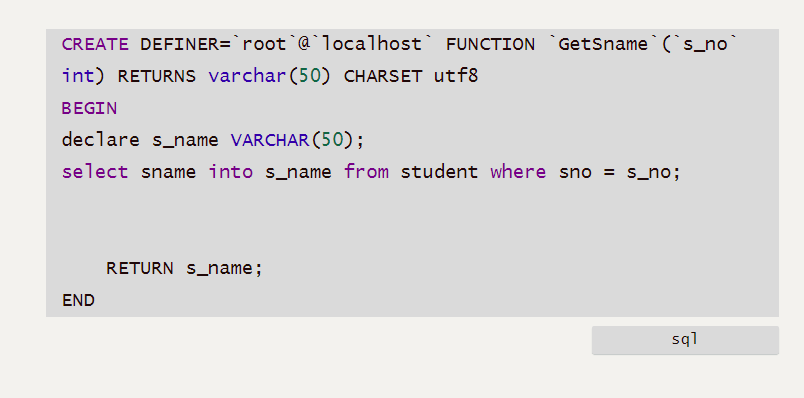
BEGIN

declare s\_name VARCHAR(50);

select sname into s\_name from student where sno = s\_no;

RETURN s\_name;

END



输入学号181340638

# 7 触发器

## 7.1触发器是一种专用类型的存储过程，它被捆绑到SQL Server 的表格或者视图上。

## 触发器是一个特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

实现了 计算插入图书的id总和， 不允许删除图书信息， 输出更新时的ID

1.计算插入图书的id总和

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `BidSum` **BEFORE** **INSERT** **ON** `book` **FOR** **EACH** **ROW** **BEGIN**

**set** @**sum** = @**sum** + **New**.Bid;

**END**

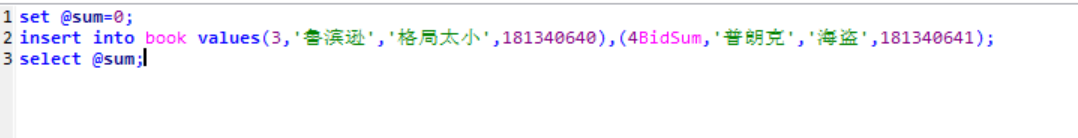


图7.1

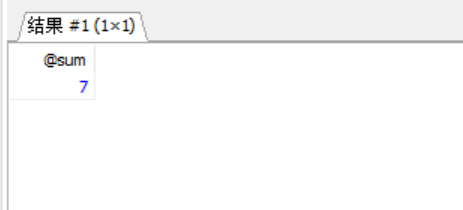


图7.2

2.

不允许删除图书信息

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `BanDelete` **BEFORE** **DELETE** **ON** `book` **FOR** **EACH** **ROW** **BEGIN**

**declare** msg **varchar**(255);

**set** msg="不允许修改图书信息";

**SIGNAL** **SQLSTATE** 'HY000' **SET** MESSAGE\_TEXT = msg;

**END**

在选择删除一个图书信息之后



图7.3

3.

输出更新时的ID

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `student\_after\_update` **AFTER** **UPDATE** **ON** `clarkstudentunion` **FOR** **EACH** **ROW** **BEGIN**

**set** @ID=@ID+**New**.ID;

**END**

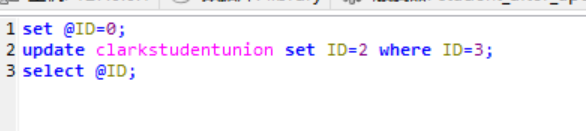


图7.4

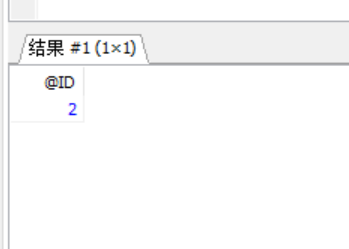


图7.5

1. 添加书籍的时候检查表中是否有重复数据



图7.6



图7.7

1. 添加书籍的时候检查书号是否合理

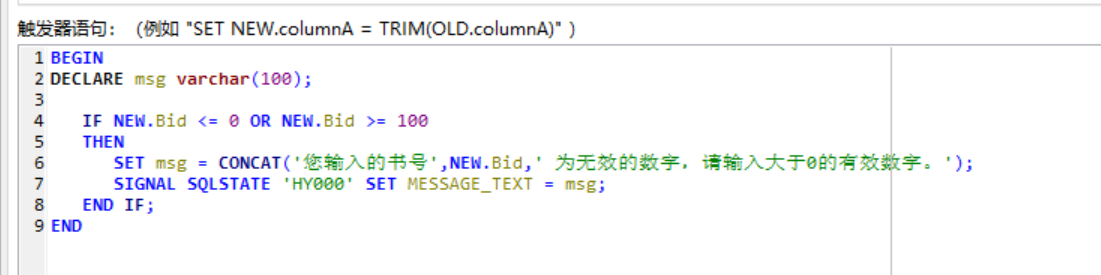


图7.8

当添加为书号为-1时

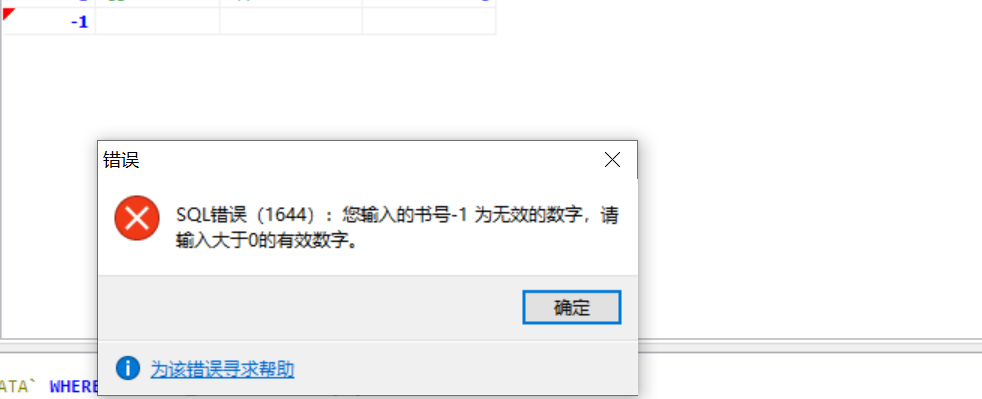


图7.9

1. 当学生表更改学号时，以学生为外键的表全部同时更改

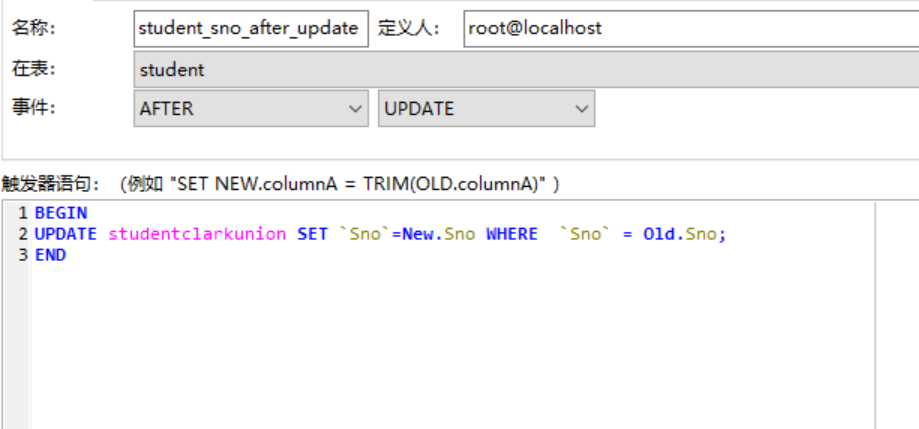


图7.10

当前信息

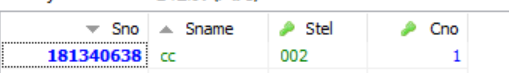


图7.11

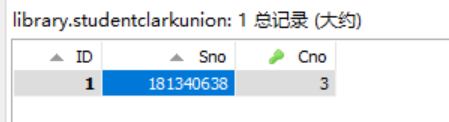


图7.12

修改图7.11里的学号

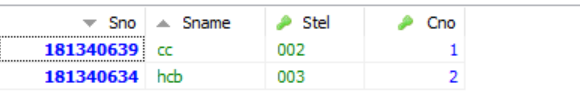


图7.13

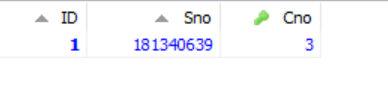


图7.14 同时修改成功

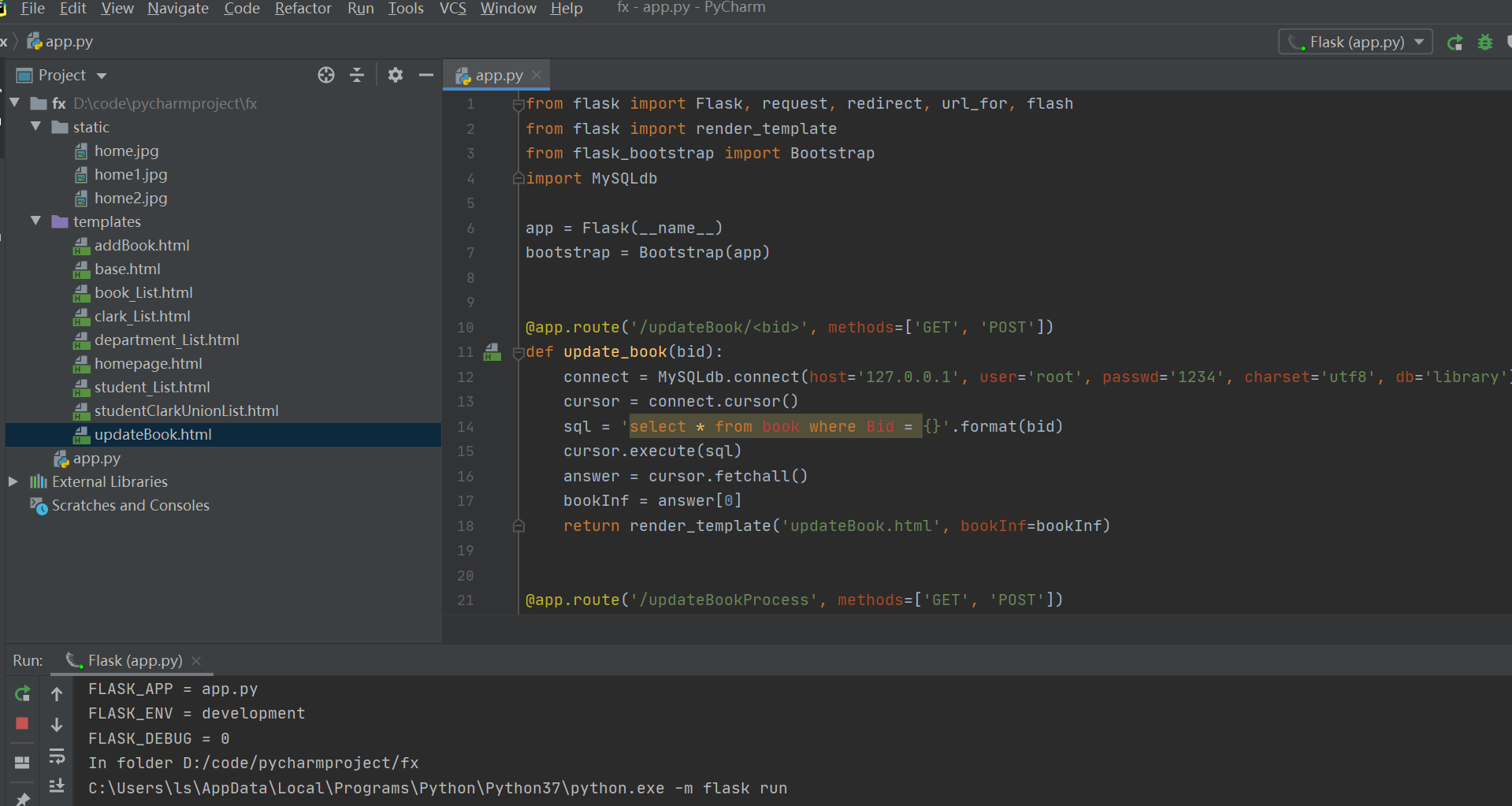
# 8 应用端的开发

不限开发工具和技术结构，要求实现应用端的运行界面。基本要求是能连后台数据库（DBMS不限，请说明联结方法）、前面的所有增删改查的SQL语句都需嵌入到程序或网页端被调用执行。不要求界面设计的美化和功能的丰富完善。

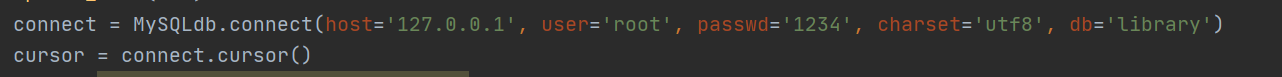
## 8.1 mysql + flask + bootstrap + python

## 8.2

利用python带的mysqldb库 和flask框架进行搭建

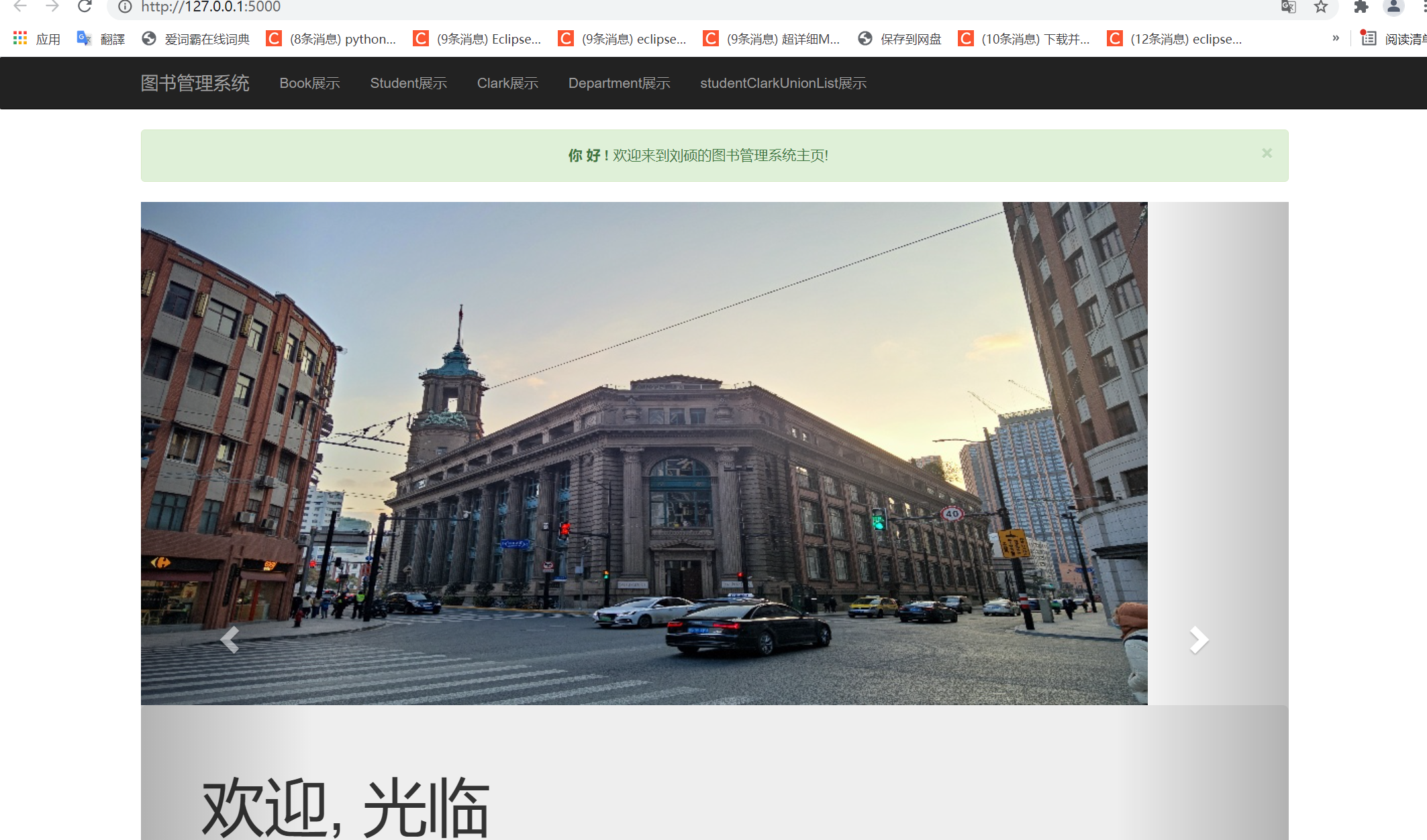


连接数据库方式

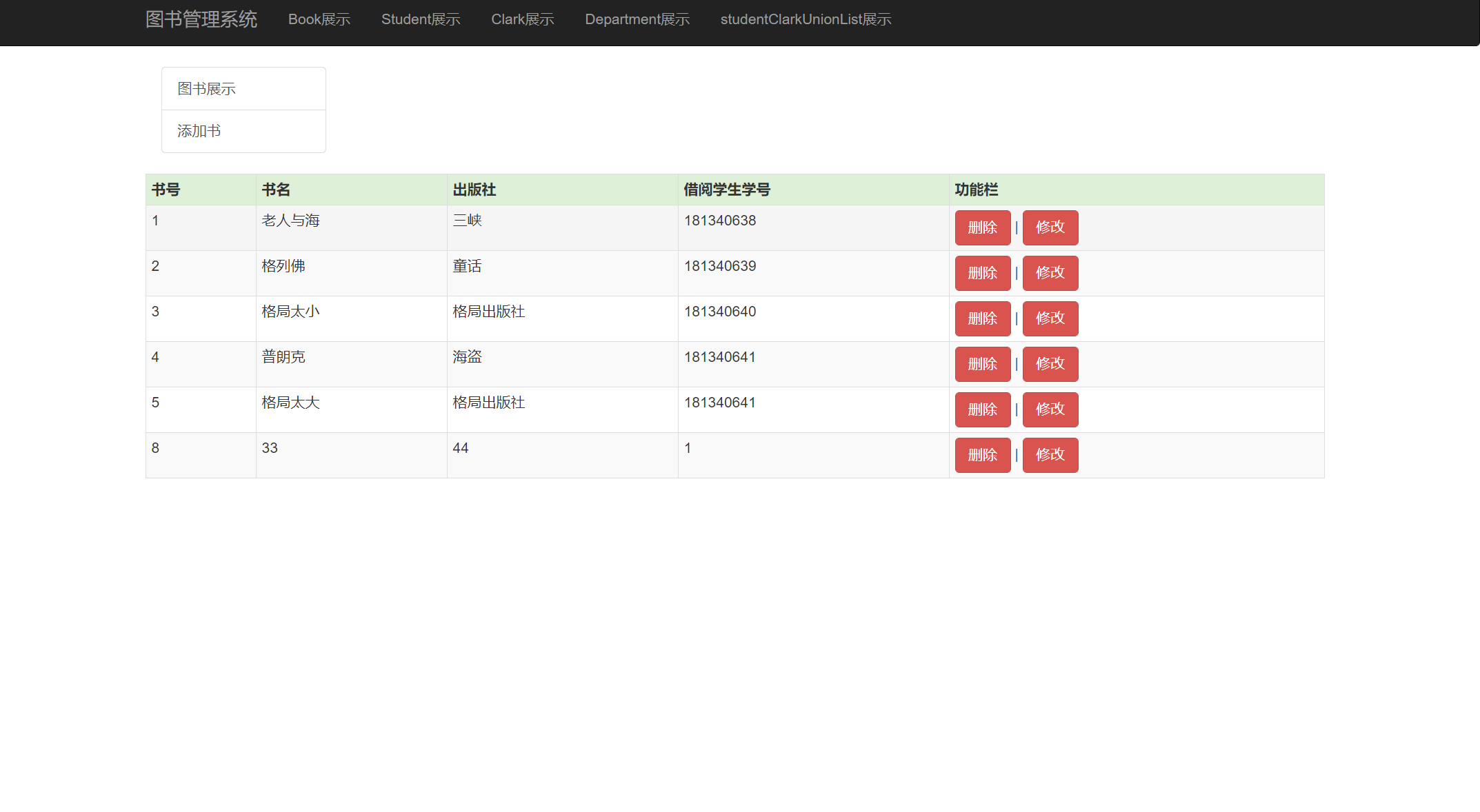


## 8.3

主页



图书管理



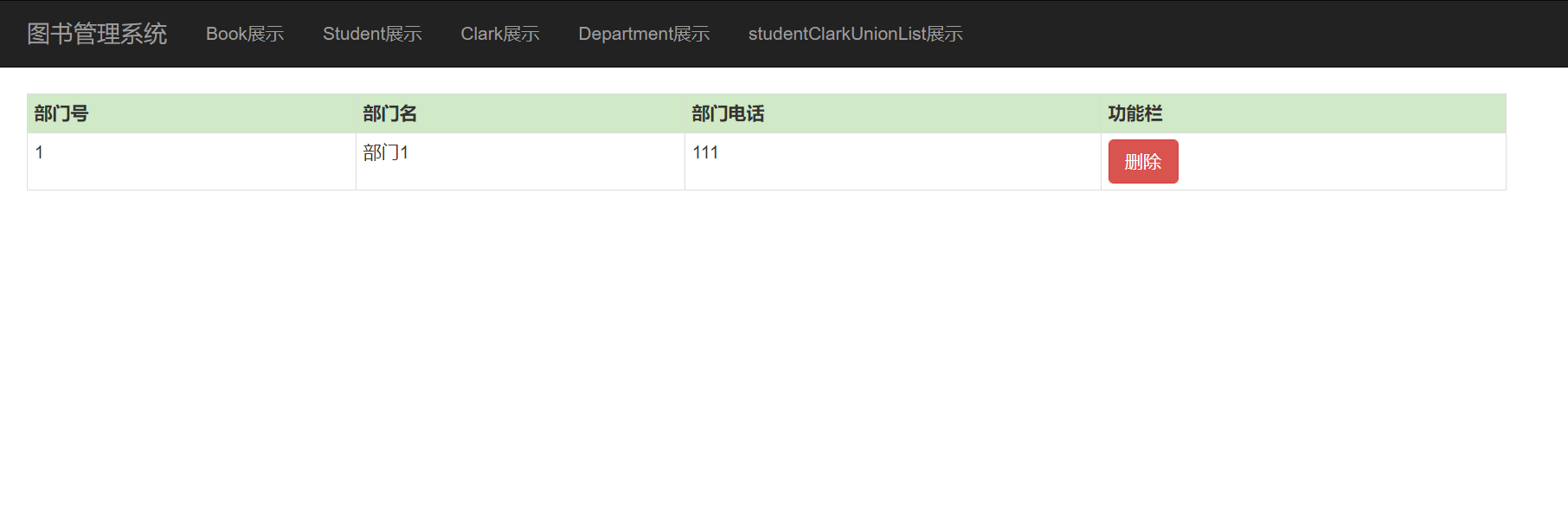
学生管理



员工管理



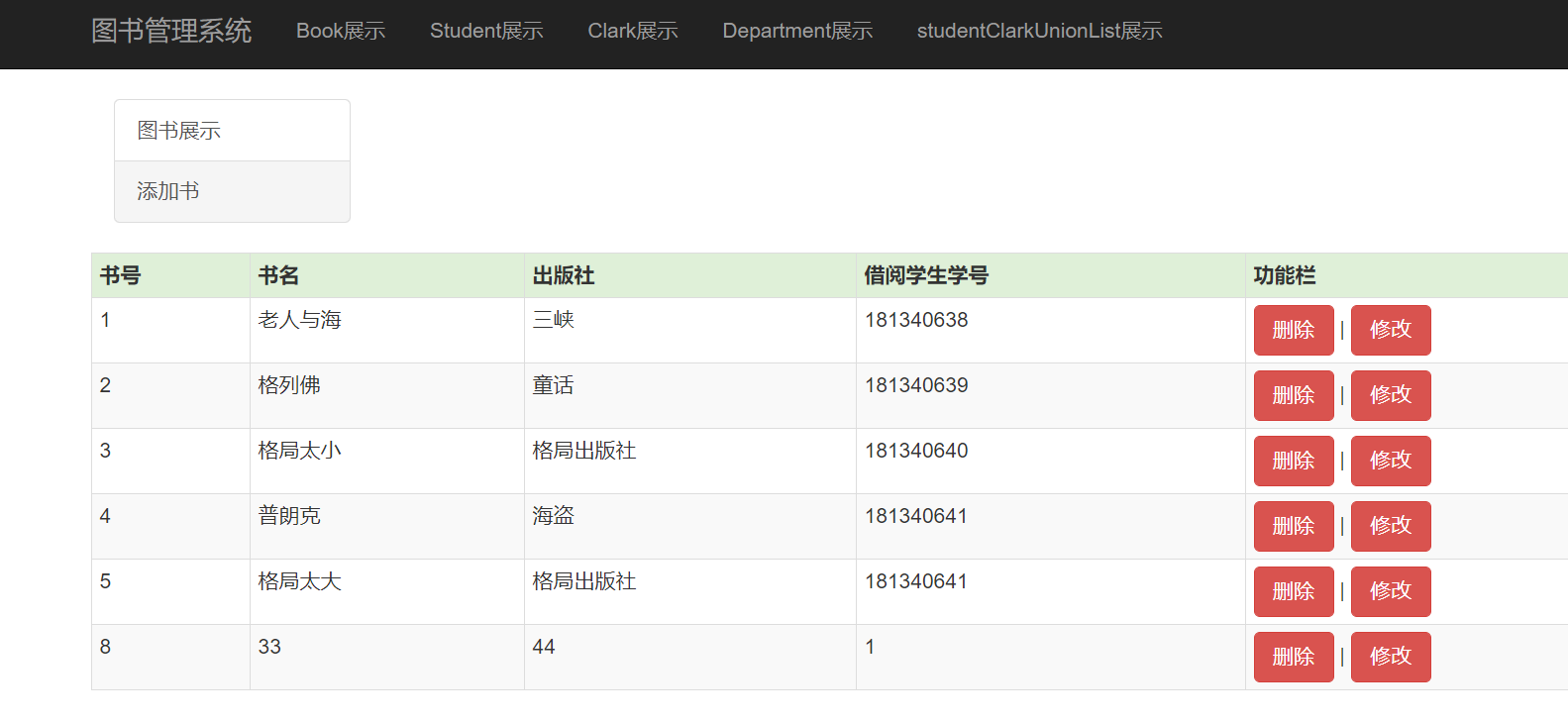
部门管理

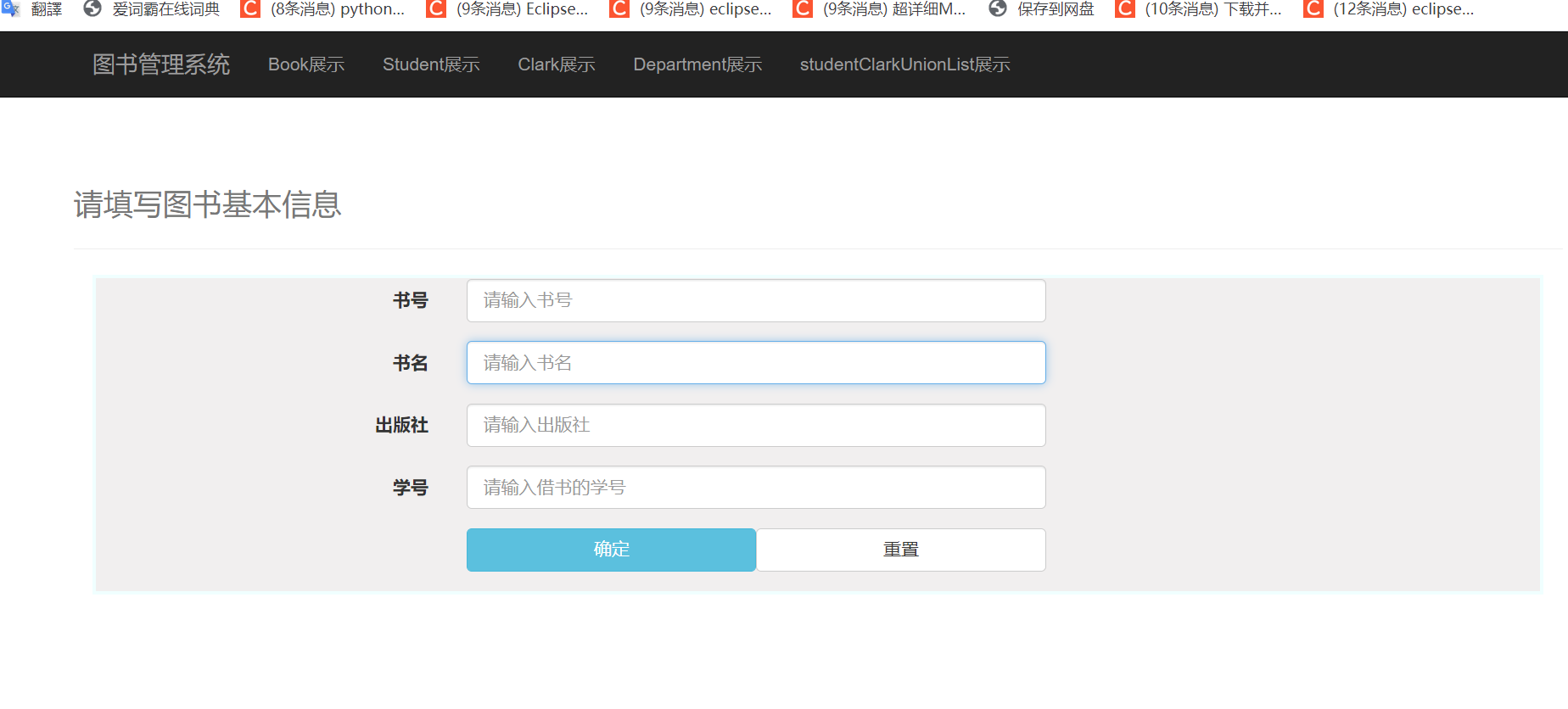


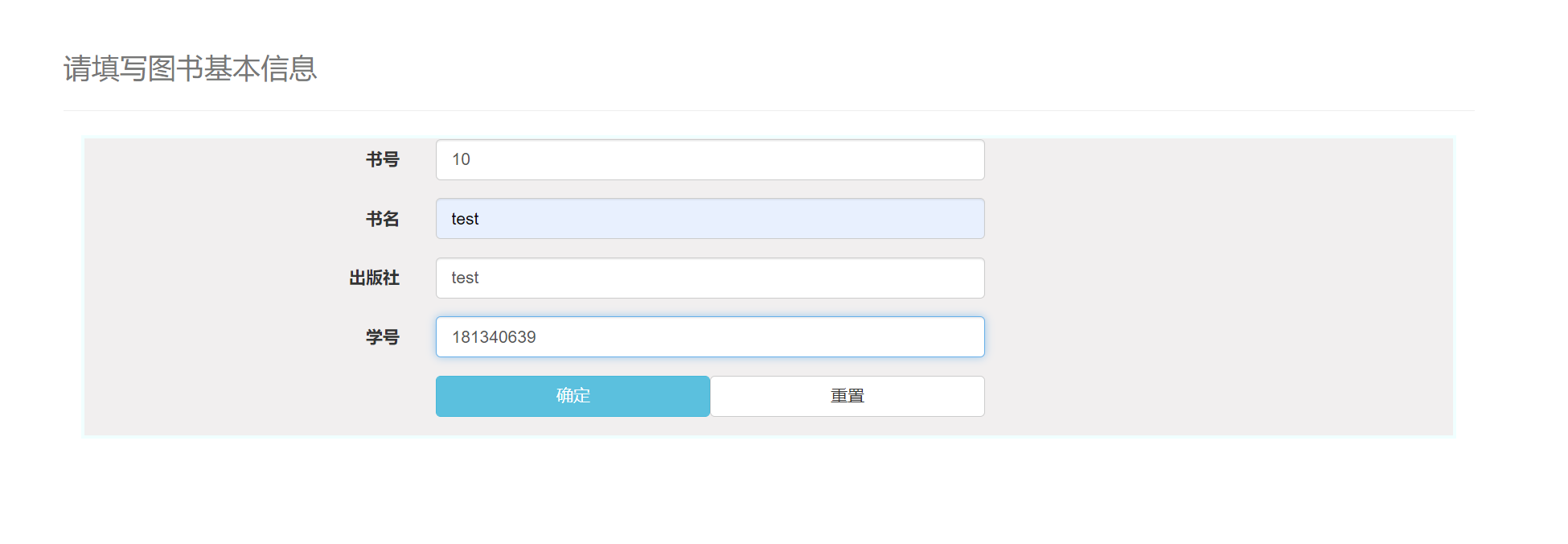
管理学生

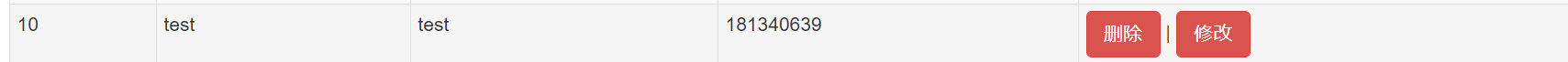


添加图书



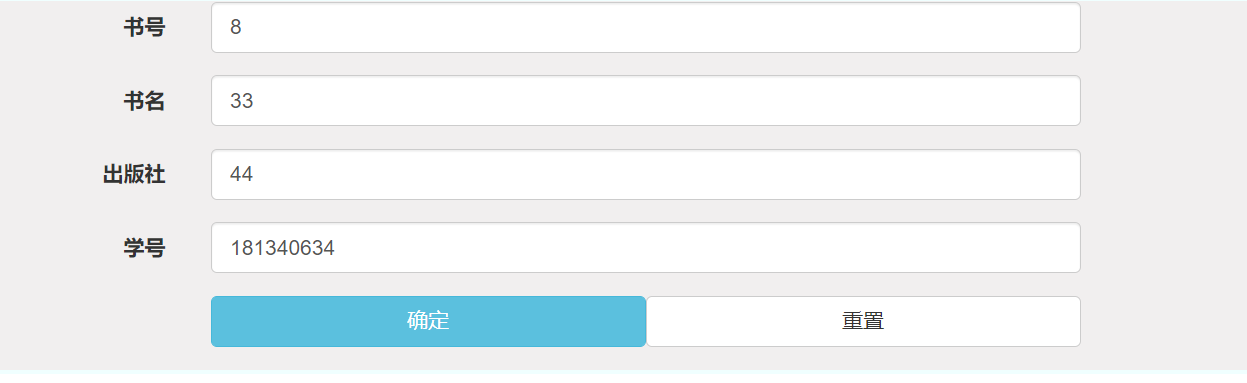


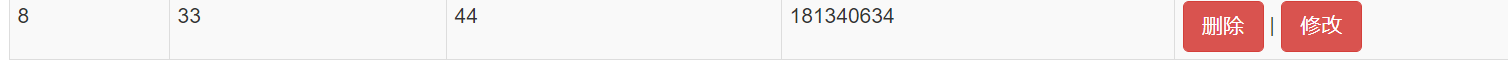




修改图书:







# 9 课程作品设计小结

## 9.1 对于数据库的语言运用不够熟练，对于触发器的重要性认知不够。

## 9.2原来从没有试过用python写网页并连接数据库，学会使用python的flask框架的基本使用以及Mysqldb连接数据库的方式，并且创建了多个触发器，简化了很多数据库中的冗余操作，十分方便，并对一些错误的操作有了限制。学会画各种数据库的图，为之后写毕设做铺垫。

# 10 附录

## 10.1 github https://github.com/huahuo999/SQL-design-web-fx-/tree/main

-- --------------------------------------------------------

-- 主机: 127.0.0.1

-- 服务器版本: 5.7.3-m13 - MySQL Community Server (GPL)

-- 服务器操作系统: Win64

-- HeidiSQL 版本: 8.3.0.4811

-- --------------------------------------------------------

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

-- 正在导出表 library.book 的数据：~7 rows (大约)

DELETE FROM `book`;

/\*!40000 ALTER TABLE `book` DISABLE KEYS \*/;

INSERT INTO `book` (`Bid`, `Bname`, `Bpub`, `Sno`) VALUES

(1, '老人与海', '三峡', 181340638),

(2, '格列佛', '童话', 181340639),

(3, '格局太小', '格局出版社', 181340640),

(4, '普朗克', '海盗', 181340641),

(5, '格局太大', '格局出版社', 181340641),

(8, '33', '44', 181340634),

(10, 'test', 'test', 181340639);

/\*!40000 ALTER TABLE `book` ENABLE KEYS \*/;

-- 正在导出表 library.clark 的数据：~1 rows (大约)

DELETE FROM `clark`;

/\*!40000 ALTER TABLE `clark` DISABLE KEYS \*/;

INSERT INTO `clark` (`Cno`, `Cname`, `Ctel`, `Dno`) VALUES

(3, 'mxy', '111', 2);

/\*!40000 ALTER TABLE `clark` ENABLE KEYS \*/;

-- 正在导出表 library.clarkstudentunion 的数据：~1 rows (大约)

DELETE FROM `clarkstudentunion`;

/\*!40000 ALTER TABLE `clarkstudentunion` DISABLE KEYS \*/;

INSERT INTO `clarkstudentunion` (`ID`, `Sno`, `Cno`) VALUES

(2, 181340638, 2);

/\*!40000 ALTER TABLE `clarkstudentunion` ENABLE KEYS \*/;

-- 正在导出表 library.department 的数据：~1 rows (大约)

DELETE FROM `department`;

/\*!40000 ALTER TABLE `department` DISABLE KEYS \*/;

INSERT INTO `department` (`Dno`, `Dname`, `Dtel`) VALUES

(1, '部门1', '111');

/\*!40000 ALTER TABLE `department` ENABLE KEYS \*/;

-- 正在导出表 library.student 的数据：~2 rows (大约)

DELETE FROM `student`;

/\*!40000 ALTER TABLE `student` DISABLE KEYS \*/;

INSERT INTO `student` (`Sno`, `Sname`, `Stel`, `Cno`) VALUES

(181340634, 'hcb', '003', 2),

(181340639, 'cc', '002', 1);

/\*!40000 ALTER TABLE `student` ENABLE KEYS \*/;

-- 正在导出表 library.studentclarkunion 的数据：~1 rows (大约)

DELETE FROM `studentclarkunion`;

/\*!40000 ALTER TABLE `studentclarkunion` DISABLE KEYS \*/;

INSERT INTO `studentclarkunion` (`ID`, `Sno`, `Cno`) VALUES

(1, 181340639, 3);

/\*!40000 ALTER TABLE `studentclarkunion` ENABLE KEYS \*/;

/\*!40101 SET SQL\_MODE=IFNULL(@OLD\_SQL\_MODE, '') \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=IF(@OLD\_FOREIGN\_KEY\_CHECKS IS NULL, 1, @OLD\_FOREIGN\_KEY\_CHECKS) \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

## 10.2

## Github: <https://github.com/huahuo999/SQL-design-web-fx->

仅列出了py文件和一个html文件

App.py

from flask import Flask, request, redirect, url\_for, flash  
from flask import render\_template  
from flask\_bootstrap import Bootstrap  
import MySQLdb  
  
app = Flask(\_\_name\_\_)  
bootstrap = Bootstrap(app)  
  
  
@app.route('/updateBook/<bid>', methods=['GET', 'POST'])  
def update\_book(bid):  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = 'select \* from book where Bid = {}'.format(bid)  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 bookInf = answer[0]  
 return render\_template('updateBook.html', bookInf=bookInf)  
  
  
@app.route('/updateBookProcess', methods=['GET', 'POST'])  
def update\_Book\_Process():  
 data = []  
 bid = request.form.get('bid')  
 bname = request.form.get('bname')  
 bpub = request.form.get('bpub')  
 sno = request.form.get('sno')  
 data.append(bname)  
 data.append(bpub)  
 data.append(sno)  
 data.append(bid)  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = 'UPDATE book SET Bname={},Bpub={},Sno={} where Bid={}'.format(bname, bpub, int(sno), int(bid))  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("更新失败")  
 return redirect(url\_for("bookList"))  
 return redirect(url\_for("bookList"))  
  
  
@app.route('/addBook')  
def add\_book():  
 return render\_template('addBook.html')  
  
  
@app.route('/addBookProcess', methods=['GET', 'POST'])  
def add\_book\_Process():  
 data = []  
 bid = request.form.get('bid')  
 bname = request.form.get('bname')  
 bpub = request.form.get('bpub')  
 sno = request.form.get('sno')  
 data.append(bid)  
 data.append(bname)  
 data.append(bpub)  
 data.append(sno)  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "insert into Book values (%s, %s, %s, %s)"  
 cursor.execute(sql, data)  
 connect.commit()  
 except Exception as e:  
 flash("添加失败")  
 return redirect(url\_for("bookList"))  
 return redirect(url\_for("bookList"))  
  
  
@app.route('/deleteBook/<bid>')  
def delete\_book(bid):  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "delete from book where Bid={}".format(bid)  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("删除失败")  
 return redirect(url\_for("bookList"))  
 return redirect(url\_for("bookList"))  
  
  
@app.route('/deleteClark/<Cno>')  
def delete\_Clark(Cno):  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "delete from clark where Cno={}".format(Cno)  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("删除失败")  
 return redirect(url\_for("clarkList"))  
 return redirect(url\_for("clarkList"))  
  
  
@app.route('/deleteDepartment/<Dno>')  
def delete\_Department(Dno):  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "delete from department where Dno={}".format(Dno)  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("删除失败")  
 return redirect(url\_for("departmentList"))  
 return redirect(url\_for("departmentList"))  
  
  
@app.route('/deleteStudent/<Sno>')  
def delete\_Student(Sno):  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "delete from student where Sno={}".format(Sno)  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("删除失败")  
 return redirect(url\_for("studentList"))  
 return redirect(url\_for("studentList"))  
  
  
@app.route('/deleteStudentClarkUnion/<ID>')  
def delete\_StudentClarkUnion(ID):  
 try:  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "delete from studentclarkunion where ID={}".format(ID)  
 cursor.execute(sql)  
 connect.commit()  
 except Exception as e:  
 flash("删除失败")  
 return redirect(url\_for("studentClarkUnionList"))  
 return redirect(url\_for("studentClarkUnionList"))  
  
  
@app.route('/studentClarkUnionList')  
def studentClarkUnionList():  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "select \* from studentclarkunion"  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 connect.close()  
 studentClarkUnions = []  
 for item in answer:  
 studentClarkUnions.append(item)  
 return render\_template("studentClarkUnionList.html", studentClarkUnions=studentClarkUnions)  
  
  
@app.route('/studentList')  
def studentList():  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "select \* from student"  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 connect.close()  
 students = []  
 for item in answer:  
 students.append(item)  
 return render\_template("student\_List.html", students=students)  
  
  
@app.route('/departmentList')  
def departmentList():  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "select \* from department"  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 connect.close()  
 departments = []  
 for item in answer:  
 departments.append(item)  
 return render\_template("department\_List.html", departments=departments)  
  
  
@app.route('/clarkList')  
def clarkList():  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "select \* from clark"  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 connect.close()  
 clarks = []  
 for item in answer:  
 clarks.append(item)  
 return render\_template("clark\_List.html", clarks=clarks)  
  
  
@app.route('/bookList')  
def bookList():  
 connect = MySQLdb.connect(host='127.0.0.1', user='root', passwd='1234', charset='utf8', db='library')  
 cursor = connect.cursor()  
 sql = "select \* from book"  
 cursor.execute(sql)  
 answer = cursor.fetchall()  
 connect.close()  
 books = []  
 for item in answer:  
 books.append(item)  
 return render\_template("book\_List.html", books=books)  
  
  
@app.route('/')  
def homepage():  
 return render\_template("homepage.html")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()  
 bootstrap.run()

base.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
  
{% extends "bootstrap/base.html" %}  
  
{% block title %}图书管理系统展示{% endblock %}  
  
{% block navbar %}  
<div class="navbar navbar-inverse" role="navigation">  
 <div class="container">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="/">图书管理系统</a>  
 </div>  
 <div class="navbar-collapse collapse">  
 <ul class="nav navbar-nav">  
 <li><a href="/bookList">Book展示</a></li>  
 <li><a href="/studentList">Student展示</a></li>  
 <li><a href="/clarkList">Clark展示</a></li>  
 <li><a href="/departmentList">Department展示</a></li>  
 <li><a href="/studentClarkUnionList">studentClarkUnionList展示</a> </li>  
 </ul>  
 </div>  
 </div>  
</div>  
{% endblock %}  
  
{% block content %}  
<div class="container">  
 {% block page\_content %}  
 {% endblock %}  
</div>  
{% endblock %}  
  
</body>  
</html>