# 图书管理系统

## 目的

通过完成从用户需求分析、概念结构设计、逻辑结构设计等一系列的数据库设计到编程、调试和应用等全过程，进一步理解和掌握教材中的相关内容。

## 要求

1. **步骤**
   * 1. 先对选定的实验做简单的需求分析。

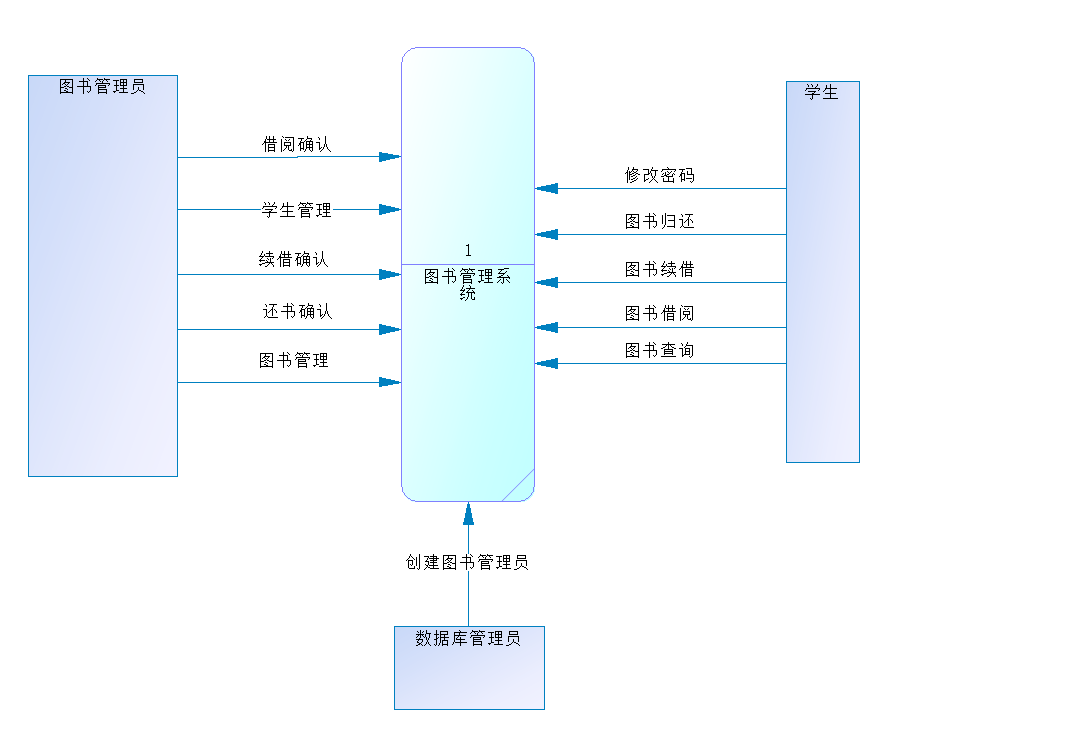
以前，人们使用传统的人工方式管理图书馆的日常业务，其操作流程比较繁琐，在借书时，读者首先将要借的书和自己的学生卡交给图书管理员才能借书，图书管理员需要在相应的借阅表中写上这次借阅记录。还书时又要再次在还书表中登记，并且还要对借书表进行修改。

从上述描述可以看到，传统的图书管理系统有很多的缺点。第一，结束和还书业务的效率低下；第二，处理能力有限，图书管理员能服务的学生人数是有限的。所以，图书管理系统的设计就是为了解决上述问题，为学生和图书馆提供快捷的图书查阅服务和高效的借书还书业务。

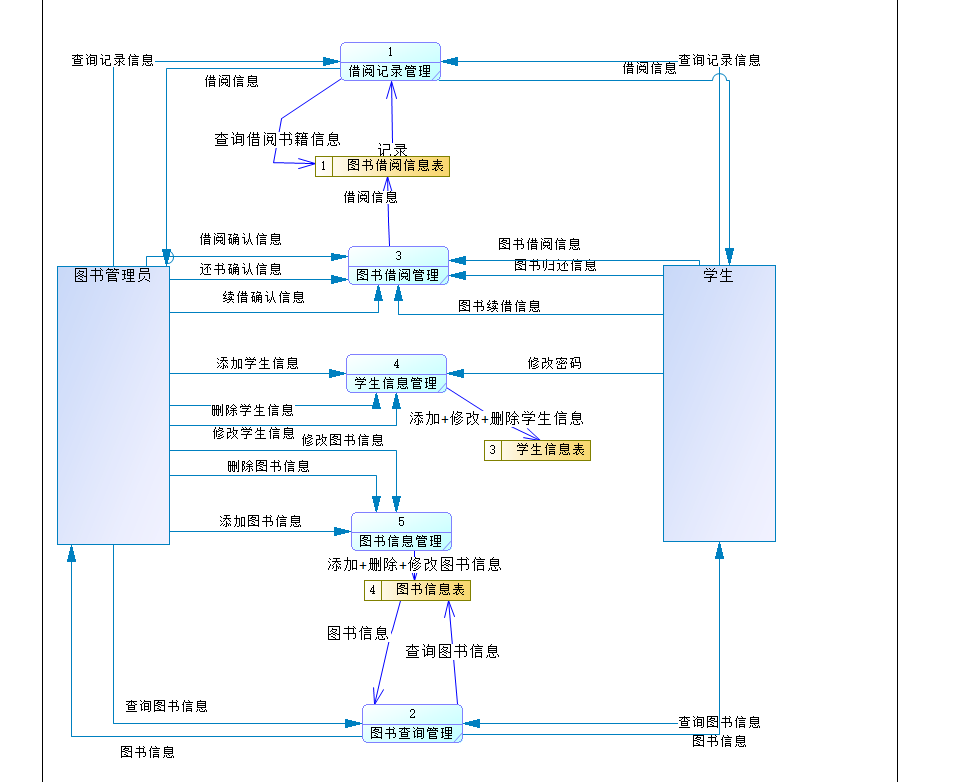
* + 1. 做出数据流图和数据字典。

数据流图：

0层数据流图



1层数据流图



数据字典：

* 数据元素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 长度 | 描述 |
| 001 | Book\_id | char | 20 | 主键 |
| 002 | Book\_name | char | 20 |  |
| 003 | Book\_auth | char | 20 |  |
| 004 | Student\_id | char | 20 | 主键 |
| 005 | Student\_name | char | 20 |  |
| 006 | Student\_pass | char | 20 |  |
| 007 | Student\_mail | char | 20 |  |
| 008 | Manager\_id | char | 20 | 主键 |
| 009 | Manager\_name | char | 20 |  |
| 010 | Time\_begin | Time | 20 | 主键 |
| 011 | Time\_end | Time | 20 |  |
| 012 | Status | int |  | 0/1/2 |
| 013 | Book\_status | int |  | 0/1/2 |
| 014 | Time\_return | Time | 20 |  |
| 015 | Time | time | 20 | 主键 |
| 016 | Type | char | 20 | 修改/删除/添加 |

* + - 数据结构

|  |  |  |
| --- | --- | --- |
| 数据结构名 | 含义说明 | 组成 |
| book | 定义了图书的各项信息 | 书籍编号，书籍作者，书籍名，书籍状态 |
| student | 定义了学生的各项信息 | 学生编号，学生名字，学生密码，学生邮箱 |
| manager | 定义了图书管理员的各项信息 | 管理员编号，管理员名字，管理员密码 |
| borrow | 定义了借书的记录 | 学生编号，书籍编号，借书开始实践，借书结束时间，管理员编号，本次借书状态 |
| return | 定义了还书的记录 | 学生编号，书籍编号，还书时间，管理员编号 |
| MS | 定义了管理员对学生的管理 | 学生编号，管理员编号，时间，类型 |
| MB | 定义了管理员对书籍的管理 | 书籍编号，管理员编号，时间，类型 |

* + - 数据流

|  |  |  |
| --- | --- | --- |
| 数据流名 | 来源 | 去向 |
| 学生信息 | 学生，图书管理员 | Student表 |
| 图书信息 | 图书管理员 | Book表 |
| 借阅信息 | 借阅图书 | Borrow表 |
| 还书信息 | 归还图书 | Return表 |
| 图书管理员信息 | DBA | Manager表 |

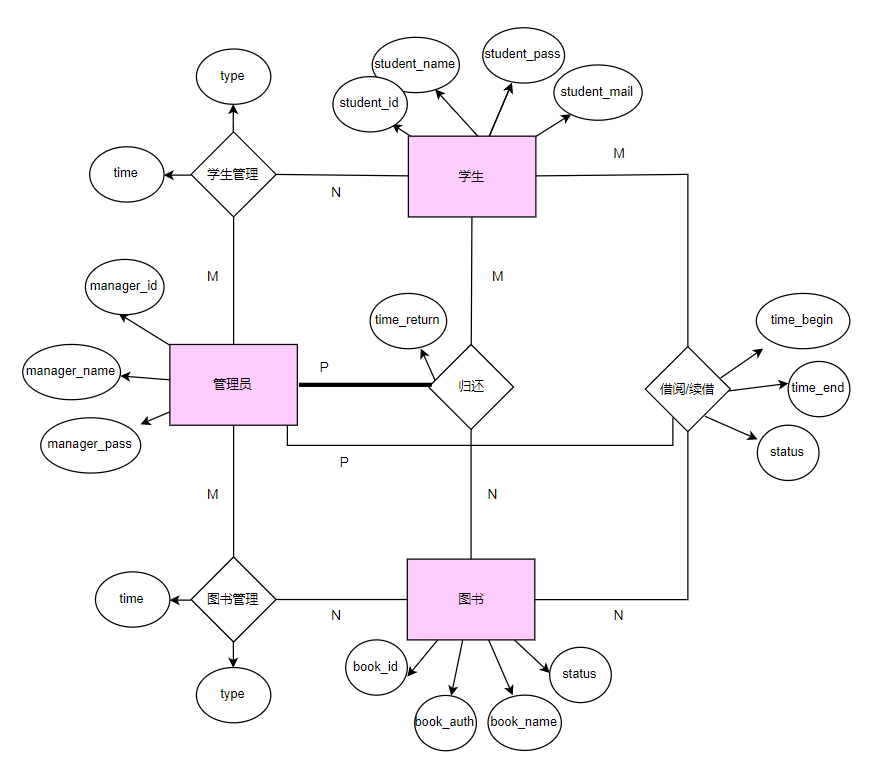
* + - 数据存储

|  |  |  |
| --- | --- | --- |
| 数据存储名 | 含义说明 | 组成 |
| 书籍信息 | 定义了图书的各项信息 | 书籍编号，书籍作者，书籍名，书籍状态 |
| 学生信息 | 定义了学生的各项信息 | 学生编号，学生名字，学生密码，学生邮箱 |
| 管理员信息 | 定义了图书管理员的各项信息 | 管理员编号，管理员名字，管理员密码 |
| 借书信息 | 定义了借书的记录 | 学生编号，书籍编号，借书开始实践，借书结束时间，管理员编号，本次借书状态 |
| 学生管理信息 | 定义了管理员对学生的管理 | 学生编号，管理员编号，时间，类型 |
| 图书管理信息 | 定义了管理员对书籍的管理 | 书籍编号，管理员编号，时间，类型 |

* + - 处理过程：

|  |  |  |  |
| --- | --- | --- | --- |
| 处理过程名 | 前提条件 | 来源 | 结果 |
| 管理学生 | 学生编号唯一 | 图书管理员 | 学生信息 |
| 管理图书 | 图书编号唯一 | 图书管理员 | 图书信息 |
| 借阅图书 | 图书未借出 | 学生 | 借阅信息 |
| 续借图书 | 本人借阅记录中有此书借阅记录且没有超过时限 | 学生 | 借阅信息 |
| 归还图书 | 本人借阅记录中有此条记录 | 学生 | 还书信息 |
| 添加管理员 | 管理员编号唯一 | DBA | 管理员信息 |

* + 1. 在数据流图和字典的基础上做出E-R图(概念结构设计)。



* + 1. 在E-R图基础上进行关系模式设计（至少满足3NF）（逻辑结构设计）。

(加粗的为主键)

Book(**book\_id**, book\_auth, book\_name, book\_status)

Student(**student\_id**, student\_name, student\_pass, student\_mail)

Manager(**manager\_id**, manager\_name, manager\_pass)

Borrow(**student\_id**, **book\_id**, **manager\_id**, **time\_begin**, time\_end, status)

Return(**student\_id**, **book\_id**, **time\_return**, **manager\_id**)

MB(**manager\_id**, **book\_id, time**,type)

MS(**manager\_id**, **student\_id, time**,type)

* + 1. 根据需求分析中的各种数据请求得出各种视图以及各种约束、规则、触发器脚本描述。

约束：

* + - book\_id,student\_id,manager\_id唯一且不为NULL
    - (book\_id,student\_id,manager\_id,time\_begin)唯一
    - (book\_id,student\_id,manager\_id,time\_return)唯一
    - Status和Book\_status只能为0或者1，0代表借出/已还，1代表未接触/未还,2代表续借。
    - Type只能是‘add’，‘remove’，‘change’这几个值

规则：

* + - 一位同学在同一时间只能借一本同名的书。
    - 只有存在于student的学生才能借书
    - 只有存在于manager的管理员才能拥有管理权
    - 只有存在于book的书才能被借
    - 还书的前提是此书的借阅记录显示为未还
    - 借书的前提是此书的状态为未借出
    - 续借的前提是借阅记录有此书，且未到还书日期。

触发的脚本：

* + - 添加学生：在student中添加此条记录，在MS中添加此次操作
    - 修改学生：在student中修改此条记录，在MS中添加此次操作
    - 删除学生：在student中删除此条记录，在MS中添加此次操作
    - 添加图书：在book中添加此条记录，在MB中添加此次操作
    - 修改图书：在book中修改此条记录，在MB中添加此次操作
    - 删除图书：在book中删除此条记录，在MB中添加此次操作
    - 查询图书：在book中查询，并且显示此书的借阅状态，可借几本，馆藏几本
    - 查询借阅记录：从borrow和return中查询相关的记录，borrow中的是已借已还和已借未还，return中的是还书记录。
    - 借阅：由student发起借阅请求，经过图书管理员确认后，将此信息插入到borrow中，同时将已借图书的book\_status置为0，并且将status的值改为1.
    - 续借：首先检查这个同学在已借未还的书籍中是否借了这本书，并且这本书的时间在借阅期限之内的才可以续借，如果都满足，那么把对应已借未还的记录的status改为2，代表续借；再添加一条记录，status为1，代表未还。
    - 还书：首先检查是否在已借未还中有这条记录，如果有，方可进行还书。将已借未还的记录中的status改为0，同时将book\_status改为1，在return中加入这一还书记录。
    1. 根据第(5)的内容在MySQL, SQLServer等中设计中数据库。

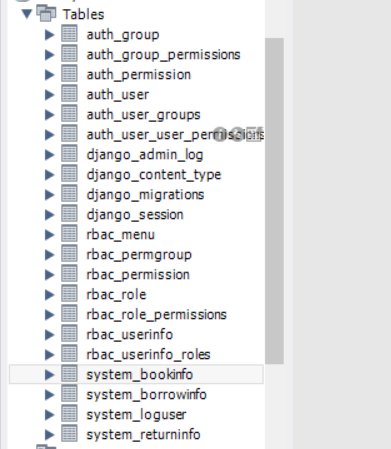
在做这一项目时，我使用了django的框架，考虑到便捷性，我使用框架中的Django框架的ROM进行连接本机的MySQL数据库。

同时，我是用了RBAC进行权限管理，最终用户数据保存在RBAC中。

由于框架的一些限制，为较好地实现上述数据库和功能，我做了一些适应框架地调整。

最终数据库如下：

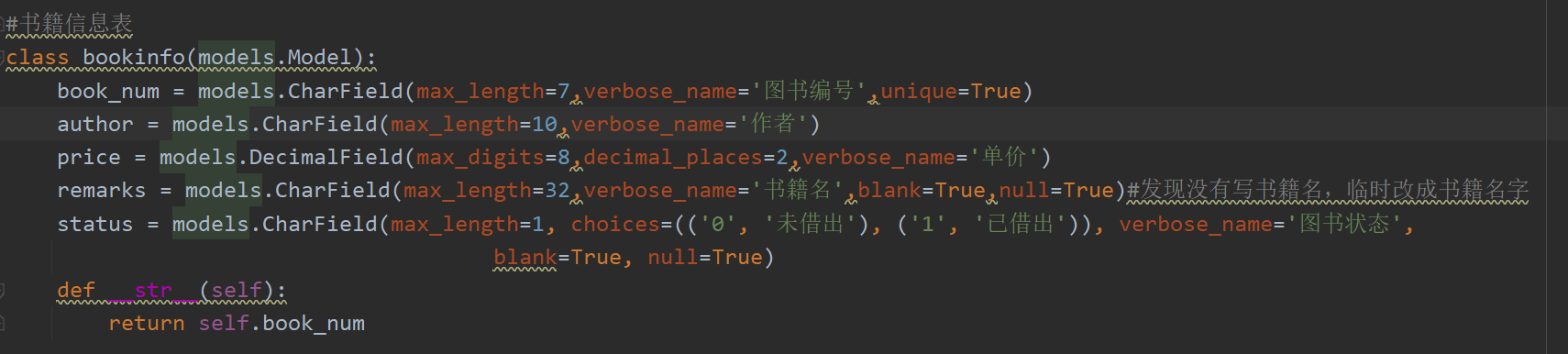
Models.py在框架中存放表地信息，一个class代表一个表，所以在models.py中进行设计，在MySQL中可以看到表格。

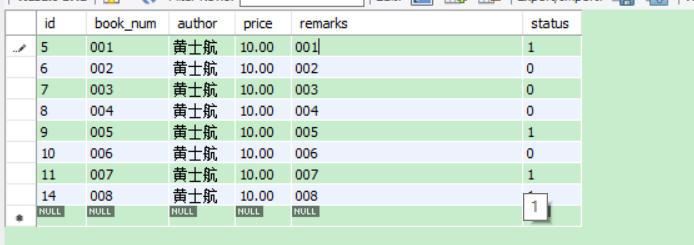


这是我的表，以”system\_”开头地是借阅信息相关，当然，也有一些使用者地信息，但是大多数的信息仍然放在RABC的UserInfo表中，以”rabc\_”开头的是RABC权限管理所需建立的数据库。其余的都是框架里面自建的表，是关于后台管理的表，将会简要介绍。

先介绍借阅信息相关的三个表：

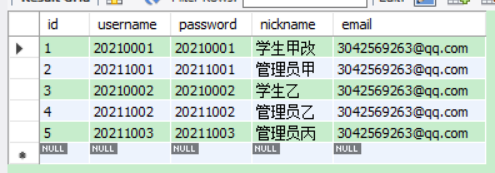
（1）书籍信息表

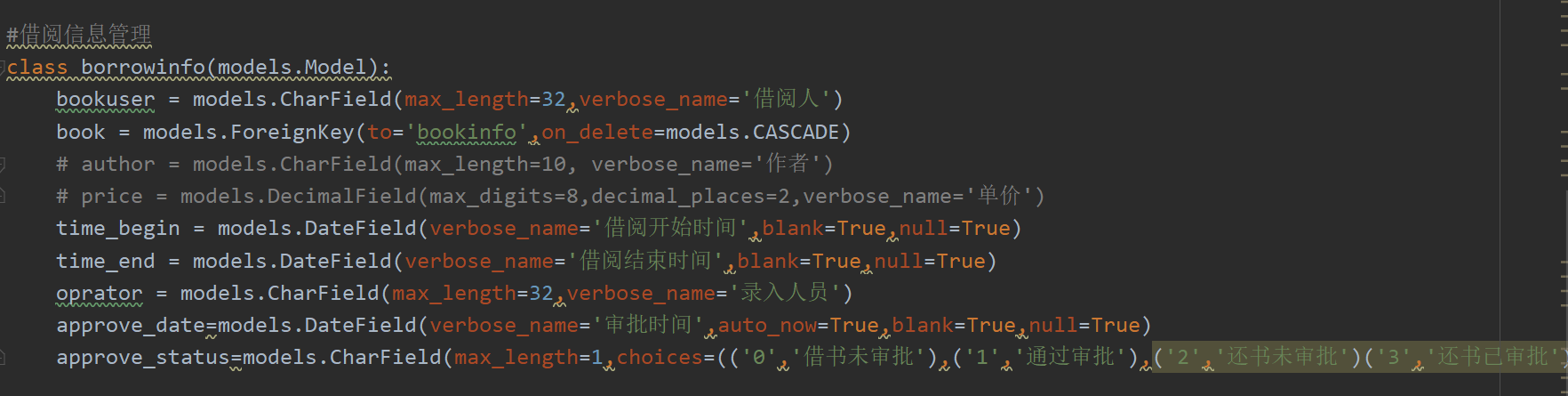


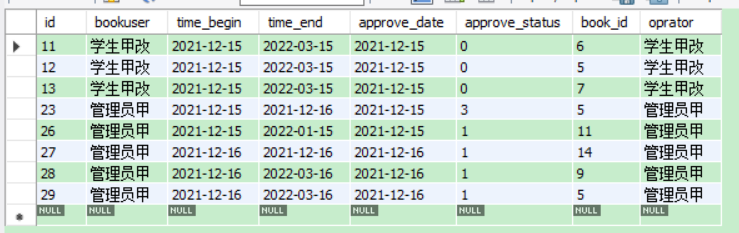


（2）用户数据表（包含学生和管理员，最终可以通过RBAC统一分配权限）





（3）借阅信息表



接着，介绍权限管理的表：

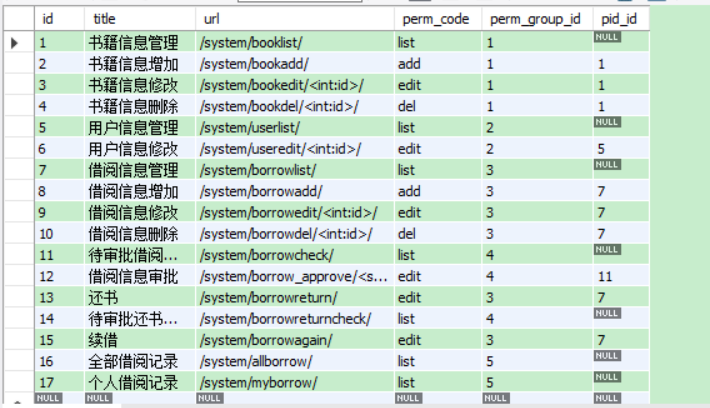
（1）rabc\_menu表，里面包含我所设置的全限一级菜单。



（2）rbac\_permgroup表，里面有权限分组，一般一个表一个权限组



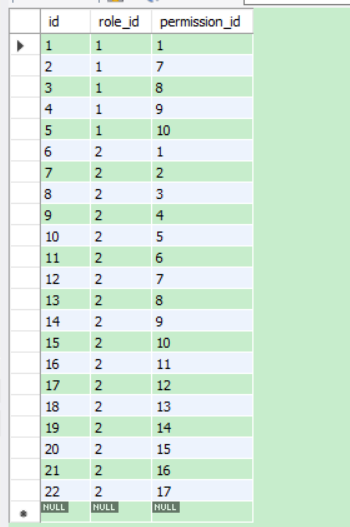
（3）rbac\_permission表，里面包含对所有表的增删改查权限。



（4）rbac\_role表，用来分配权限，可以让用户属于一个角色，便可以得到角色的权限。



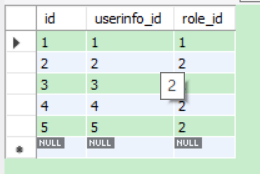
（5）rbac\_role\_permission表，里面有角色和权限的对应关系



（6）rbac\_userinfo表，里面有相关的用户信息。



（7）rbac\_userinfo\_roles表，用户和角色的对应表

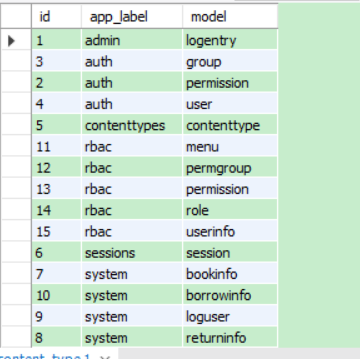


最后，我们看看使用框架搭建的后台程序相关的数据库表

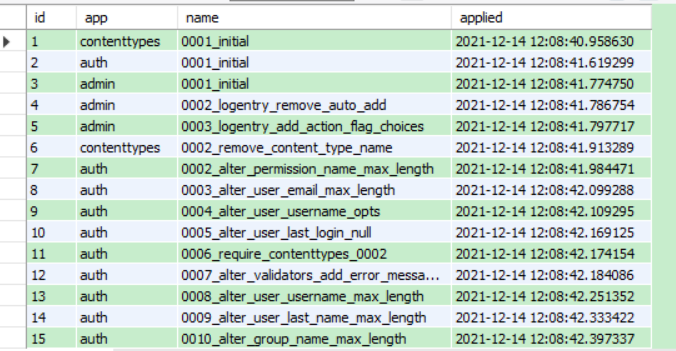
1. django\_admin\_log中有相关的用户修改记录信息



1. django\_content\_type中是此程序所有的文件



1. django\_migration表，包含使用信息



* + 1. 用自己熟悉的一门语言进行设计的功能的实现，必须包括对数据的查询、增加、修改和删除。

这里面设计一些增删改查操作主要使用django中的ROM进行相关操作，下面对ROM进行简要的介绍：

ORM 概念

对象关系映射（Object Relational Mapping，简称ORM）模式是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。 简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中。 ORM在业务逻辑层和数据库层之间充当了桥梁的作用。

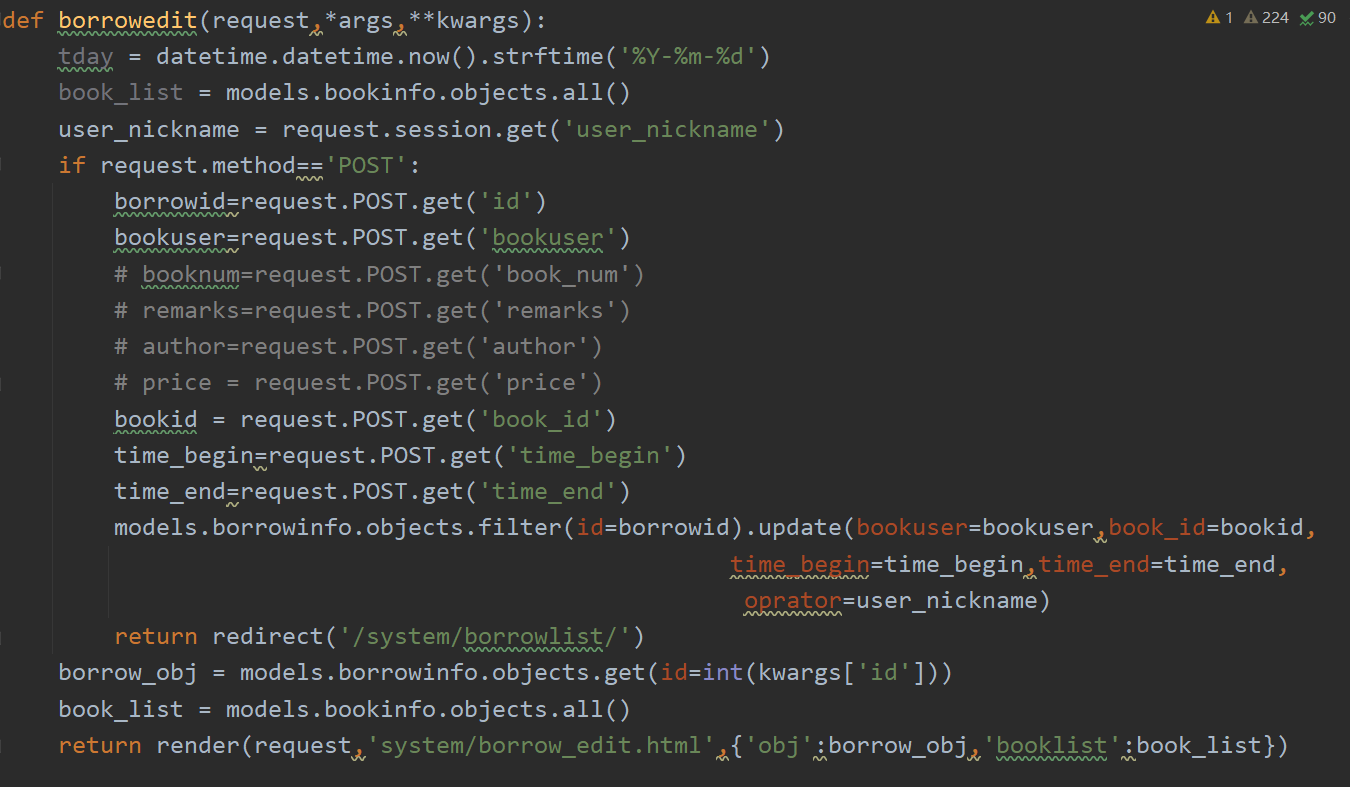
ORM 与 DB 的对应关系

ORM 面向对象和关系型数据库的一种映射,通过操作对象的方式操作数据库数据,不支持对库的操作,只能操作表 对应关系: 类 --> 表 对象 --> 数据行 属性 --> 字段

Model 模块 在Django中model是你数据的单一、明确的信息来源。它包含了你存储的数据的重要字段和行为。通常， 一个模型（model）映射到一个数据库表。

在程序中的views.py中，每个视图函数都要使用增删改查，以满足相关操作的要求，下面举一个例子：

这个是借阅信息修改的视图函数，倒数第二行和倒数第三行就是对数据库使用ORM进行操作，之后将数据传入HTML文件中。



* + 1. 整理前面几步的文档和程序源码，可执行程序。
    2. 实验报告的最后写出本次实验完成中遇到的问题及解决方法。

**问题1：从SQL语言向ORM的转变**

对SQL语言已经比较熟悉了，ORM是对SQL的一层封装，对底层的查询等操作进行了一些优化，可以提高整体性能。但是ORM与SQL的差别很大，他的增删改查更加接近于python语言。同时，ORM里面的跨表操作比较麻烦，涉及正向查找和反向查找，里面的细节比较多，刚开始我并不熟练，所以出现了一系列的错误，造成了时间的浪费。

**问题2：权限分配造成很大的困扰，框架中的语言并不能很方便地设置不同对象地权限。**

为进行权限地分配，学习了RBAC方法，具体说就是三个要素，权限，角色和用户。将权限分配给角色，不同的用户属于不同地角色，这样不同类别地用户就可以拥有所需的权限。十分便捷，十分好用。

**问题3：管理员如何创建的问题，设计的方案并不能很好的完成管理员的创建。**

我通过框架的后台，在实现相关的权限管理时，建立了一个superuser，让superuser进行人员的增删，将学生和管理员都作为用户，只是两者对数据表的可操作权限不一样。

**问题4：Bootstrap中的模态框出现了一些问题。**

为了界面的扁平化，我配置了bootstrap的模态框，但是中途出现了一些尴尬的问题。由于使用不熟练，导致在模态框中选中的信息无法很好的传入到调用模态框的页面。最终，传入相关数据表项的id才解决了问题。（注意：这里的id是django为每个表都建一个，是主键，唯一标识一项。）

**问题4：jquary的ajax请求没有注意网址中数字和列表的区别**

在管理员对借阅信息确认的界面里，使用了一次确认多个借阅记录的方式进行管理，但是传回来的是列表，我没又有在意，最后发现是没有将int改为str，修改后才能正常运行。

**开发环境：python 3.8 + Django 3.2.9 + pillow 8.4.0 + Bootstrap + jQuary**

1. **提交成果物：**
   * 1. 完整的实验报告（报告中包括上述要求中的1）-5)和9），注意不需要将客户端程序源代码贴入实验报告中，但可以将触发器、存储过程等数据库端的代码加以说明写入实验报告中，另外请将开发环境（包括语言、平台、所使用数据库的名称以及版本号）写入实验报告中。
     2. 备份好的数据库备份文件，以及相关的SQL语句。
     3. 源代码以及可执行程序。
     4. 配置运行说明文档：说明程序该如何配置使之能正常运行。
     5. 其他：安装包、帮助文件、演示视频等（可选）。