数据库大作业

图书管理系统数据库设计

张庭源　　2015301500148

2017.12

分工信息

蹇奇芮 2015301500150	视图设计 触发器设计 过程设计
张庭源 2015301500148	数据库设计 工程实现 代码测试

目录

1 系统概述 1
1.1 简介 1
1.2 功能需求 1
1.2.1 面向读者 1
1.2.2 面向管理员 1
2 系统功能设计 2
2.1 功能模块图 2
2.2 功能模块描述 2
2.2.1 读者端 2
2.2.2 管理员端 2
3 数据库设计方案 3
3.1 系统 E-R 模型 3
3.2 表项设计 3
3.2.1 book 表 3
3.2.2 Reader 表 4
3.2.3 manager 表 5
3.2.4 borrowInfo 表 6
3.2.5 managerInfo 表 7
3.3 设计索引(由队友完成) 7
3.3.1 reader（在读者名上建立索引） 7
3.3.2 borrowInfo 8
3.3.3 book（在书名上建立索引） 8
3.3.4 managerInfo 8
3.4 设计视图(由队友完成) 9
3.4.1 书籍借阅数量（月榜） 9
3.4.2 用户借阅数量（月榜） 9
3.4.3 逾期用户记录 9
3.4.4 30 天内新入库书籍 9
3.5 设计触发器(由队友完成) 9
3.5.1 borrowInfo 表的触发器 9
3.5.2 book 表的触发器 9
3.5.3 manageInfo 表的触发器 9
3.6 数据库过程设计（由队友完成） 9
3.6.1 查询从某时间起书籍被借阅的次数 9
3.6.2 查询从某时间读者借阅书籍的数量 9
4 应用程序程序设计与编程实现 10
4.1 应用程序设计综述 10
4.2 系统功能实现（ManageSystem） 10
4.2.1 用户登陆（ReaderLogin） 10
4.2.2 用户注册（ReaderRegist） 11
4.2.3 管理员登陆（ManagerLogin） 11
4.3 公共功能实现（Public） 12

4.3.1 图书查询（SearchBook） 12
4.3.2 热门书籍查询（BorrowRank） 13
4.3.3 借阅排行查询（UserRank） 14
4.3.4 新书上架 15
4.4 读者功能实现（Reader） 15
4.4.1 更改账户信息（ChangeAccount） 15
4.4.2 查看本人的借阅信息（BorrowRecords） 16
4.4.3 借书（Borrow） 17
4.4.4 还书（Return） 17
4.5 管理员功能实现（Manager） 18
4.5.1 新书籍入库（AddNewBook） 18
4.5.2 销毁一本书（DeleteBook） 19
4.5.3 禁用/解禁一个用户（BanReader） 20
4.5.4 列表所有违约账户（InvalidAcount） 20
5 实习体会 22

1 系统概述

1.1 简介

该图书管理系统用于实现简单的书籍和借阅的管理，分为两部分，面向读者的部分和面向管理员的部分。面向读者的部分主要实现图书信息的查询和展示，借书和还书申请，以及当前借书状态的查看。面向图书管理员的部分主要实现对读者，图书借阅情况，和图书的管理，以及系统维护等功能。

1.2 功能需求

1.2.1 面向读者

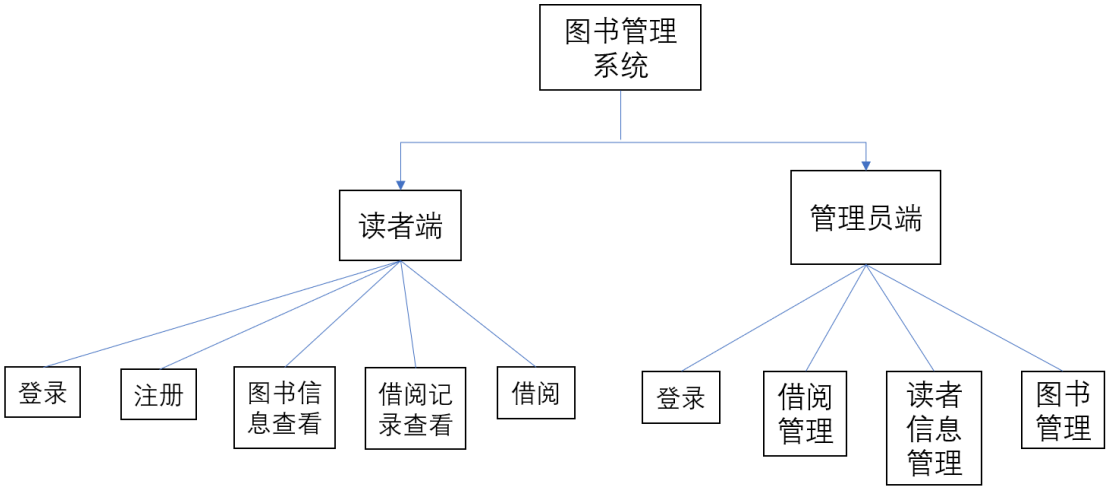
读者需要能够注册账户，登录读者系统，在系统中能够查看所有图书信息，按照图书的部分信息（编号，书名，类型等）检索对应图书，查看图书的借阅排行榜，申请借书和还书，查看自己的借/还书记录，查看自己账户信息以及更改自己的登录密码。

1.2.2 面向管理员

管理员首先需要能够登录管理员系统，管理员系统管理功能分为三个部分，第一，**借阅管理**，管理员需要能够查询任意图书的借阅次数，能够查看借阅排行榜，能够查询/检索/修改/新增任意读者的借阅记录，能够查询所有逾期记录，并能够能查看对应用户信息，以及修改读者账户状态。第二，**账户管理**，管理员能够查看和检索所有读者账户信息，能够删除和修改所有账户信息，能够新增账户。第三，**图书管理**，管理员能够查看和检索所有图书的信息，并能删除，修改以及新增图书信息。管理员能够查看/检索/修改所有出入库记录，并能新增/删除出入库记录。

2 系统功能设计

2.1 功能模块图



2.2 功能模块描述

2.2.1 读者端

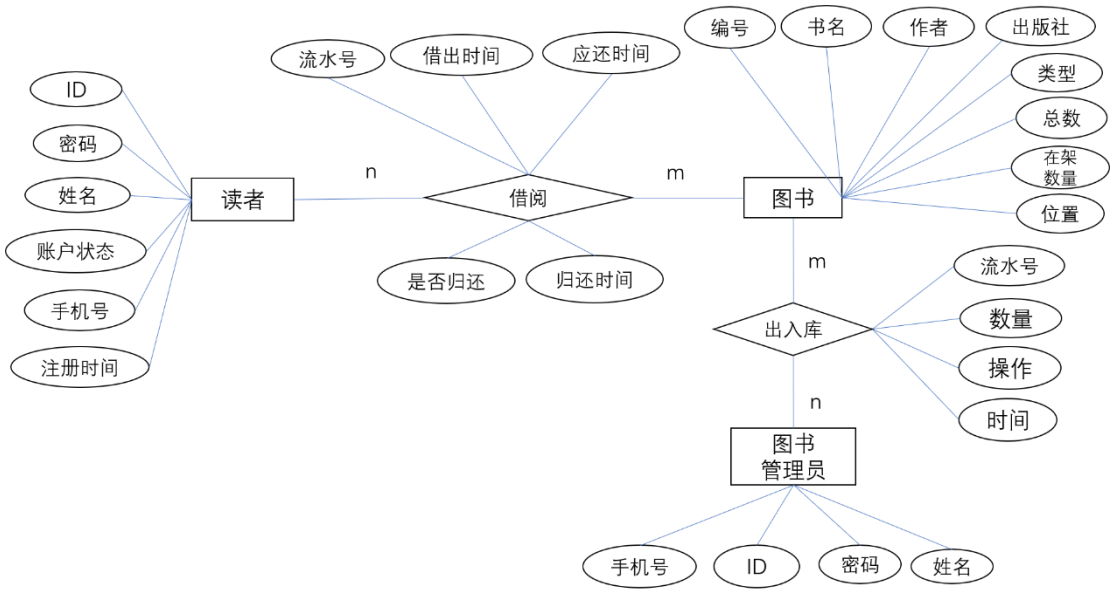
a) **登录：**该模块用于读者登录读者系统时进行用户名和密码检查
b) **注册：**该模块用于录入读者提供的必要信息，创建一个读者系统的账户
c) **图书信息查看：**该模块用于显示图书信息，可以根据需求（书名，编号，类型，作者等）搜索图书，以及显示图书借阅榜
d) **借阅记录查看：**该模块用于显示本用户的借阅记录，可以根据需求（时间，书名，借阅状态）搜索对应记录
e) **借阅**：该模块完成借书和还书操作，根据书籍信息信息提交借书申请并返回借书的结果或者根据书籍信息提交还书申请并返回还书的结果

2.2.2 管理员端

a) **登录：**该模块用于管理员登录管理员系统时进行用户名和密码检查
b) **借阅管理：**该模块可以查看各图书的借阅情况，以及显示所有的借阅记录，并能根据需求（账户编号，姓名，时间，书名，借阅状态…）搜索对应的借阅记录，可以查看和检索读者的账户信息，可以更改其用户状态
c) **读者信息管理：**该模块可以查看和检索以及更改所有读者的账户信息，可以新增/删除用户
d) **图书管理：**该模块可以查看/检索/更改所有图书信息，可以新增/删除图书信息，可以查看/检索/修改/新增/删除所有图书出入库记录

3 数据库设计方案

3.1 系统 E-R 模型



3.2 表项设计

3.2.1 book 表

字段名	类型	默认值	键类型	描述
bookID	int(11)		主键	
bookName	varchar(45)			
author	varchar(45)			
publisher	varchar(45)			
total	int(11)			书籍总数
onShelf	int(11)			在架数
type	varchar(45)			
location	varchar(45)			

数据定义 DDL:

```
CREATE TABLE `book` (
  `bookID` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `bookName` varchar(45) NOT NULL,
  `author` varchar(50) DEFAULT NULL,
```

```
  `publisher` varchar(50) DEFAULT NULL,
  `total` int(11) NOT NULL DEFAULT '0',
  `onShelf` int(11) NOT NULL DEFAULT '0',
  `type` varchar(45) DEFAULT NULL,
  `location` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`bookID`)
) ENGINE=InnoDB AUTO_INCREMENT=836 DEFAULT CHARSET=utf8
```

#	bookID	bookName	author	publisher	total	onShelf	type	location
1	826	计算机网络	黄传河	科学出版社	10	10	计算机...	C3
2	827	计算机网络	Marry	宇宙出版社	15	15	计算机...	B2
3	828	密码学引论	张焕国	武汉大学...	5	4	计算机...	B2
4	829	深入理解计算机系统	David	机械工业...	5	4	计算机...	D2
5	830	密码学C++实现	迈克尔	机械工业...	5	2	计算机...	A2
6	831	他改变了中国	罗伯特	世纪文景	20	20	人物传记	A5
7	832	神们自己	阿西...	凤凰文艺...	10	10	科幻小说	E5
8	833	计算机组成与设计	约翰	机械工业...	3	3	计算机...	E5
9	834	Python核心编程	史密斯	人民邮电...	3	3	计算机...	C5
10	835	C语言程序设计基础	谭成予	武汉大学...	10	10	计算机...	B1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.2.2 Reader 表

字段名	类型	默认值	键类型	描述
readerID	int(11)		主键	
password	varchar(45)			
readerName	varchar(45)			
telephone	varchar(11)			
banned	tinyint(1)			用户禁用标志
CreateTime	timestamp	current_time		

数据定义 DDL:

```
CREATE TABLE `reader` (
  `readerID` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `password` varchar(45) NOT NULL DEFAULT '123456',
  `readerName` varchar(45) NOT NULL,
  `telephone` varchar(11) NOT NULL,
  `banned` tinyint(1) DEFAULT '0',
  `CreateTime` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
```

```
    PRIMARY KEY (`readerID`) USING BTREE,
    UNIQUE KEY `userID_UNIQUE` (`readerID`),
    UNIQUE KEY `telephone` (`telephone`)
)       ENGINE=InnoDB       AUTO_INCREMENT=36       DEFAULT
CHARSET=utf8
```

#	readerID	password	readerName	telephone	banned	CreateTime
1	33	123456	Alice	18022011302	0	2017-10-13 03:17:12
2	34	123456	Bob	18022011302	1	2017-10-13 03:18:03
3	35	123456	Tony	18022011584	0	2017-10-13 03:18:24
*	NULL	NULL	NULL	NULL	NULL	NULL

3.2.3 manager 表

字段名	类型	默认值	键类型	描述
managerID	int(11)		主键	
password	varchar(45)			
name	varchar(45)			
telephone	varchar(45)			

数据定义 DDL:

```
CREATE TABLE `manager` (
    `managerID` int(11) unsigned NOT NULL AUTO_INCREMENT,
    `password` varchar(45) NOT NULL DEFAULT '123456',
    `name` varchar(45) DEFAULT NULL,
    `telephone` char(11) DEFAULT NULL,
    PRIMARY KEY (`managerID`),
    UNIQUE KEY `telephone` (`telephone`)
)       ENGINE=InnoDB       AUTO_INCREMENT=6       DEFAULT
CHARSET=utf8
```

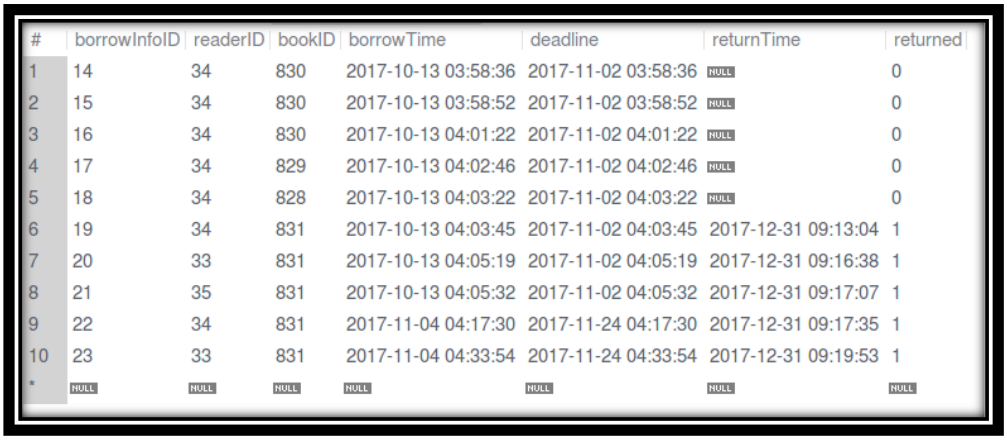
#	managerID	password	name
1	10000	123456	Alice
2	10001	123456	Bob
3	10002	123456	Ciri
*	NULL	NULL	NULL

3.2.4 borrowInfo 表

字段名	类型	默认值	键类型	描述
borrowInfoID	int(11)		主键	
readerID	int(11)		外键	
bookID	int(11)		外键	
borrowTime	timestamp	current_time		借书时间
deadline	timestamp			到期时间
returnTime	timestamp			还书时间
returned	tintint(1)			归还否

数据定义 DDL:

```
CREATE TABLE `borrowInfo` (
  `borrowInfoID` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `readerID` int(11) unsigned NOT NULL,
  `bookID` int(11) unsigned NOT NULL,
  `borrowTime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `deadline` timestamp NULL DEFAULT NULL,
  `returnTime` timestamp NULL DEFAULT NULL,
  `returned` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`borrowInfoID`),
  KEY `fk_borrowInfo_1_idx` (`bookID`),
  KEY `fk_borrowInfo_2_idx` (`readerID`),
  CONSTRAINT `bookID` FOREIGN KEY (`bookID`) REFERENCES `book` (`bookID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `readerID` FOREIGN KEY (`readerID`) REFERENCES `reader` (`readerID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8
```



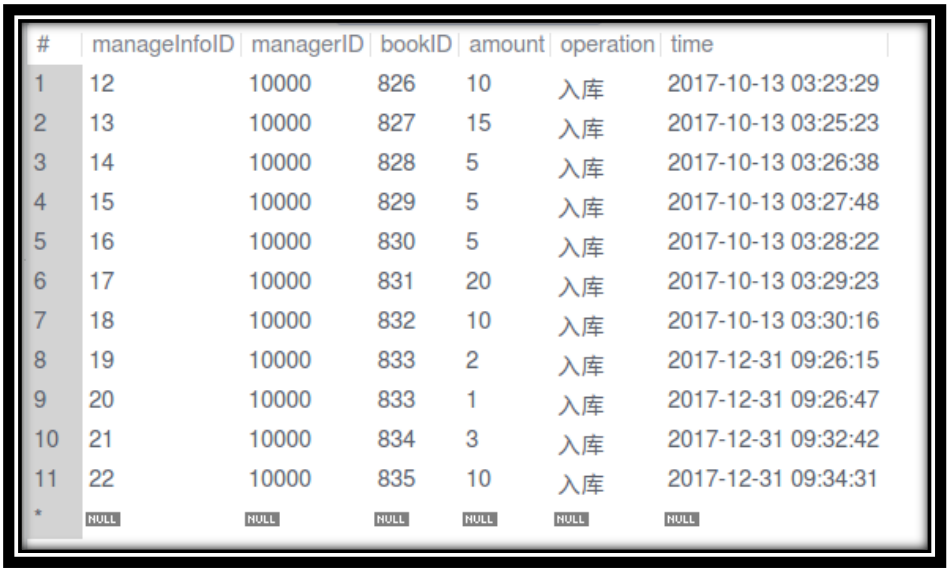
#	borrowInfoID	readerID	bookID	borrowTime	deadline	returnTime	returned
1	14	34	830	2017-10-13 03:58:36	2017-11-02 03:58:36	NULL	0
2	15	34	830	2017-10-13 03:58:52	2017-11-02 03:58:52	NULL	0
3	16	34	830	2017-10-13 04:01:22	2017-11-02 04:01:22	NULL	0
4	17	34	829	2017-10-13 04:02:46	2017-11-02 04:02:46	NULL	0
5	18	34	828	2017-10-13 04:03:22	2017-11-02 04:03:22	NULL	0
6	19	34	831	2017-10-13 04:03:45	2017-11-02 04:03:45	2017-12-31 09:13:04	1
7	20	33	831	2017-10-13 04:05:19	2017-11-02 04:05:19	2017-12-31 09:16:38	1
8	21	35	831	2017-10-13 04:05:32	2017-11-02 04:05:32	2017-12-31 09:17:07	1
9	22	34	831	2017-11-04 04:17:30	2017-11-24 04:17:30	2017-12-31 09:17:35	1
10	23	33	831	2017-11-04 04:33:54	2017-11-24 04:33:54	2017-12-31 09:19:53	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.2.5 managerInfo 表

字段名	类型	默认值	键类型	描述
manageInfoID	int(11)		主键	
managerID	int(11)		外键	
bookID	int(11)		外键	
amount	int(11)			数量
operation	ENUM('入库', '出库')			操作内容
time	timestamp	current_time		操作时间

数据定义 DDL:

```
CREATE TABLE `manager` (
  `managerID` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `password` varchar(45) NOT NULL DEFAULT '123456',
  `name` varchar(45) DEFAULT NULL,
  `telephone` char(11) DEFAULT NULL,
  PRIMARY KEY (`managerID`),
  UNIQUE KEY `telephone` (`telephone`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8
```



#	manageInfoID	managerID	bookID	amount	operation	time
1	12	10000	826	10	入库	2017-10-13 03:23:29
2	13	10000	827	15	入库	2017-10-13 03:25:23
3	14	10000	828	5	入库	2017-10-13 03:26:38
4	15	10000	829	5	入库	2017-10-13 03:27:48
5	16	10000	830	5	入库	2017-10-13 03:28:22
6	17	10000	831	20	入库	2017-10-13 03:29:23
7	18	10000	832	10	入库	2017-10-13 03:30:16
8	19	10000	833	2	入库	2017-12-31 09:26:15
9	20	10000	833	1	入库	2017-12-31 09:26:47
10	21	10000	834	3	入库	2017-12-31 09:32:42
11	22	10000	835	10	入库	2017-12-31 09:34:31
*	NULL	NULL	NULL	NULL	NULL	NULL

3.3 设计索引(由队友完成)

3.3.1 reader（在读者名上建立索引）

CREATE INDEX readerNameDex ON reader(name)

3.3.2 borrowInfo

CREATE INDEX userBookIdDex ON borrowInfo(readerID,bookID)

3.3.3 book（在书名上建立索引）

CREATE INDEX bookNameDex ON book(name)

3.3.4 managerInfo

CREATE INDEX managerBookIdDex ON manager(managerId,bookId)

3.4 设计视图(由队友完成)

3.4.1 书籍借阅数量（月榜）

3.4.2 用户借阅数量（月榜）

3.4.3 逾期用户记录

3.4.4 30 天内新入库书籍

3.5 设计触发器(由队友完成)

3.5.1 borrowInfo 表的触发器

3.5.2 book 表的触发器

3.5.3 manageInfo 表的触发器

3.6 数据库过程设计（由队友完成）

3.6.1 查询从某时间起书籍被借阅的次数

3.6.2 查询从某时间读者借阅书籍的数量

4 应用程序程序设计与编程实现

4.1 应用程序设计综述

作为管理系统的后端程序，此程序主要目的是作为接口与前端系统进行通信，提供合适的接口，使前端可以从数据库中查询到所需要数据表单、或通过一系列的处理将前端提供的数据合法写入数据库。

后端程序由 python 语言开发而成。python 语言具有方便测试、结构简单、面向对象的特点，适合作为本程序的开发语言。同时，也因为该语言方便测试的特点，我们对后端提供的所有接口进行了逐个的测试，确保了后端接口可以正常使用。后端程序使用了 MySQLdb 包作为连接数据库的工具。该包提供了多种操作数据库的方式。

后端程序的基本设计思路是将数据库中每个表单抽象为类，将所需要的接口作为相关类的方法，并在此基础上进行程序设计。

系统的主要接口功能由以下四个类实现：

a) ManageSystem，提供系统的基本功能的实现
b) Public，提供公共功能的实现
c) Reader，提供读者相关功能的实现
d) Manager，提供管理员相关功能的实现

4.2 系统功能实现（ManageSystem）

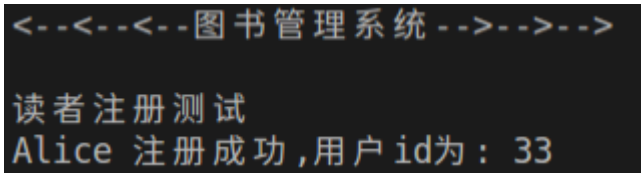
4.2.1 用户登陆（ReaderLogin）

传入参数	a) readerID b) passwd
处理过程	a) 在数据库 reader 表中查找此用户，并返回用户信息 b) 存在此用户时，使用该用户参数实例化一个 reader 对象并作为自身属性
返回值	是否登录成功
功能	判断用户是否成功登录

```
def readerLogin(self,readerID,passwd):
    '''读者登录'''
    sql = '''select * from reader
            where readerID = %s and password = %s'''
    param = (readerID,passwd)
    result = self.dbControler.Search(sql,param)
    if not result:
        '''返回值是一个False,说明没有查到,登录失败'''
        return False
```

```
        result = result[0]
        self.currentReader = Reader(result,self.dbControler)
        return True
```

4.2.2 用户注册（ReaderRegist）



传入参数	a） passwd b） readerName c） telephone
处理过程	a） 将新的用户信息插入 reader 表中，其 readerID 将由数据库自动生成 b） 查询刚刚自增生成的 readerID
返回值	readerID
功能	注册一个新用户

```
def ReaderRegist(self,passwd, readerName, telephone):
        '''读者注册'''
        # 生成一个 userid,构造 Reader 实例
        # 现在将由触发器自动递增 readerID
        sql = '''insert into reader (password,readerName,telephone)
                value (%s,%s,%s); '''
        param = (passwd, readerName, telephone)
        if not self.dbControler.Execute(sql, param):
            return False
        # 注册成功后自动登录
        sql = '''select @@IDENTITY''' #返回刚刚注册的那个 id
        param = ()
        readerID = self.dbControler.Search(sql,param)
        readerID = readerID[0][0]
        return readerID
```

4.2.3 管理员登陆（ManagerLogin）

传入参数	a） managerID b） passwd
处理过程	a） 在 manager 表中查询该管理员及密码是否存在，并返

	回管理员详细信息 b）若存在，实例化一个 manager 对象作为 system 属性
返回值	是否登录成功
功能	判断管理员是否成功登录

```
def ManagerLogin(self, managerID, passwd):
    '''管理员登录'''
    sql = '''select * from manager
            where managerID = %s and password = %s'''
    param = (managerID, passwd)
    result = self.dbControler.Search(sql,param)
    if not result:
        '''返回值是一个False,说明没有查到,登录失败'''
        return False
    result = result[0]
    self.currentManager = Manager(result,self.dbControler)
    return True
```

4.3 公共功能实现（Public）

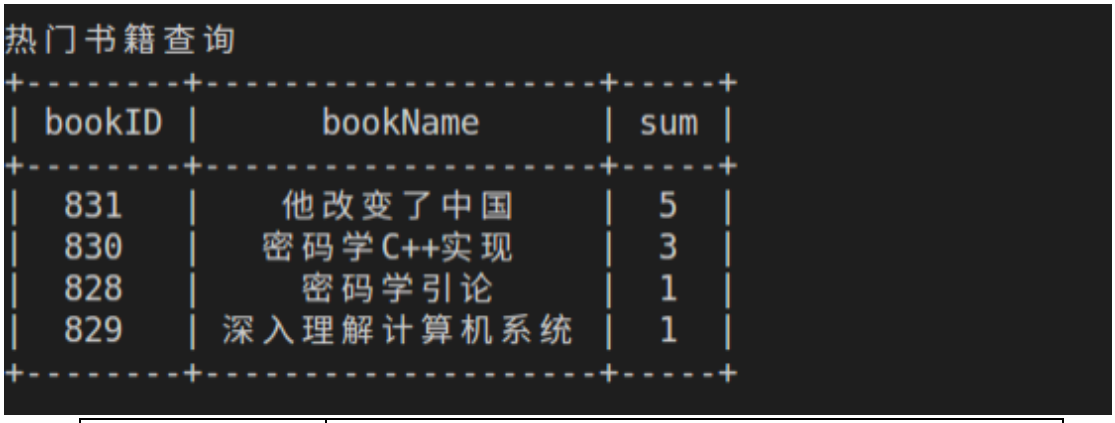
4.3.1 图书查询（SearchBook）

传入参数	searchMessage
处理过程	a）在 book 表中查询所有包含'searchMassage'的 bookName、author、publisher、type 字段，并将其返回 b）创建一个列表 book_list，包含所有根据数据库信息实例化的 book 对象
返回值	book_list
功能	查找有关键字的字段

```
def SearchBook(self, searchMessage):
    '''查找图书'''
    searchMessage = '%'+searchMessage+'%'
    sql = '''select * from book
            where bookName like %s or author like %s or publisher
             like %s or type like %s;'''
    param = (searchMessage, searchMessage, searchMessage,
               searchMessage,)
    books = self.dbControler.Search(sql,param)
    if not books:
```

```
            return False
        book_list = []
        for book in books:
            book_list.append(Book(book))
        return book_list
```

4.3.2 热门书籍查询（BorrowRank）



传入参数	a） timelimit
处理过程	a） 判断是否有 timelimit 传入 b） 若无 timelimit，查询视图 bookRankingList，将结果列表返回 c） 若传入了 timelimit，则调用数据库 bookRankingList 过程查询相关数据
返回值	bookRankingList
功能	将不同书籍借阅次数从上到下排序输出

```
def BorrowRank(self,timeLimit = ''):
        '''热门书籍榜'''
        if not timeLimit:
            sql = '''select * from bookRankingList'''
            param = ()
            result = self.dbControler.Search(sql,param)
            if not result:
                return False
            return result
        else:
            if self.dbControler.cursor.callproc('bookRankingList',
(timeLimit,)):
                return self.dbControler.cursor.fetchall()
        pass
```

4.3.3 借阅排行查询（UserRank）

借书榜单查询

readerID	readerName	sum
34	Bob	7
33	Alice	2
35	Tony	1

传入参数	a） timelimit
处理过程	a） 判断是否有 timelimit 传入 b） 若无 timelimit，查询视图 readerRankingList，将结果列表返回 c） 若传入了 timelimit，则调用数据库 readerkRankingList 过程查询相关数据
返回值	rederRankingList
功能	返回借阅次数最多的用户

```
    def UserRank(self,timeLimit = ''):
        '''借阅排行榜'''
        if not timeLimit:
            sql = '''select * from readerRankingList'''
            param = ()
            result = self.dbControler.Search(sql,param)
            if not result:
                return False
            return result
        else:
            if
self.dbControler.cursor.callproc('readerRankingList',(timeLimit,)):
                return self.dbControler.cursor.fetchall()
```

4.3.4 新书上架

```
<--<--<--公用功能测试-->-->-->

新书上架
+--------+--------------------+------------+-----+
| bookID |      bookName      |    type    | sum |
+--------+--------------------+------------+-----+
|  833   |  计算机组成与设计  | 计算机科学 |  3  |
|  834   |   Python核心编程   | 计算机科学 |  3  |
|  835   | C语言程序设计基础  | 计算机科学 |  10 |
+--------+--------------------+------------+-----+
```

传入参数	无
处理过程	a） 查询视图 newBook
返回值	书籍列表
功能	返回最近一个月内的书籍列表

```
    def newBook(self):
        '''新书上架'''
        sql = '''select * from newBook'''
        param = ()
        result = self.dbControler.Search(sql,param)
        if not result:
            return False
        return result
        pass
```

4.4 读者功能实现（Reader）

4.4.1 更改账户信息（ChangeAccount）

传入参数	a） passwd b） readerName c） telephone
处理过程	a） 传入更改信息 b） update 数据库中 reader 表 c） 更新当前 reader 对象
返回值	是否更新成功
功能	更新当前用户的个人信息，不能更改 id、banned

```
    def ChangeAccount(self, passwd, readerName, telephone):
        '''更改账户信息'''
        sql = '''update reader
                set password = %s,readerName = %s, telephone = %s
                where readerID = %s;'''
        param = (passwd,readerName,telephone,self.readerID)
        if not self.dbControler.Execute(sql,param):
            return False
        '''成功后更改账户当前信息'''
        self.passwd = passwd
        self.readerName = readerName
        self.telephone = telephone
        return True
```

4.4.2 查看本人的借阅信息（BorrowRecords）

```
我的借阅信息查询
+--------+---------+--------+---------------------+---------------------+------------+----------+
| infoID | rederID | bookID |      borrowTime     |       deadLine      | returnTime | returned |
+--------+---------+--------+---------------------+---------------------+------------+----------+
|   20   |    33   |  831   | 2017-10-13 04:05:19 | 2017-11-02 04:05:19 |    None    |    0     |
|   23   |    33   |  831   | 2017-11-04 04:33:54 | 2017-11-24 04:33:54 |    None    |    0     |
+--------+---------+--------+---------------------+---------------------+------------+----------+
```

传入参数	无
处理过程	a） 在 borrowInfo 表中查询当前用户 id 产生的条目 b） 将每条信息实例化为 borrowInfo 对象，产生一个列表
返回值	borrowInfo_list
功能	查询当前用户产生的借阅信息

```
    def BorrowRecords(self):
        '''查看的读者的借阅信息'''
        borrowInfo_list = []
        sql = '''select * from borrowInfo
                where readerID = %s;'''
        param = (str(self.readerID),)
        result = self.dbControler.Search(sql,param)
        if not result:
            return False
        for item in result:
            borrowInfo_list.append(BorrowInfo(item))
        # 查询成功后返回一个 borrowInfo 的 list
        return borrowInfo_list
```

4.4.3 借书（Borrow）

传入参数	bookID
处理过程	a）在 borrowInfo 中插入一条借阅 b）查询刚刚借阅信息的流水 id c）用该流水 id 查询完整 borrowInfo，并实例化一个 borrowInfo 对象
返回值	borrowInfo 对象
功能	借书

```
def Borrow(self, bookID):
    '''借书'''
    sql = '''insert into borrowInfo (readerID,bookID)
            values (%s,%s)'''
    param = (str(self.readerID),str(bookID))
    if not self.dbControler.Execute(sql,param):
        return False
    sql = '''select @@IDENTITY''' #返回刚刚注册的那个 id
    param = ()
    borrowInfoID = self.dbControler.Search(sql,param)
    borrowInfoID = borrowInfoID[0][0]
    sql = '''select * from borrowInfo
            where borrowInfoID = %s'''
    param = (borrowInfoID,)
    result = self.dbControler.Search(sql,param)
    if not result:
        return False
    result =result[0]
    borrowInfo = BorrowInfo(result)
    return borrowInfo
```

4.4.4 还书（Return）

```
还书
True

我的借阅信息查询
+--------+---------+--------+---------------------+---------------------+---------------------+----------+
| infoID | rederID | bookID |      borrowTime     |       deadLine      |      returnTime     | returned |
+--------+---------+--------+---------------------+---------------------+---------------------+----------+
|   20   |    33   |  831   | 2017-10-13 04:05:19 | 2017-11-02 04:05:19 | 2017-12-31 09:16:38 |    1     |
|   23   |    33   |  831   | 2017-11-04 04:33:54 | 2017-11-24 04:33:54 |         None        |    0     |
+--------+---------+--------+---------------------+---------------------+---------------------+----------+
```

传入参数	bookID

处理过程	a) 在 borrowInfo 表中查询该 bookID 且未归还的信息 b) 选择其中 deadline 最早的一条，将其更新，触发器会自动增加 book 表的数量
返回值	是否成功
功能	还一本已借出的书，若借了多本该书，归还最早应还的

```
    def Return(self, bookID):
        '''还书'''
        # 同一本书能不能借两次?否则不知道还哪本?一次只能还一本?逻辑
        sql = '''select borrowInfoID from borrowInfo
                where bookID = %s and returned = 0'''
        borrowInfoID = self.dbControler.Search(sql,param)
        print borrowInfoID
        borrowInfoID = borrowInfoID[0][0]
        sql = '''update borrowInfo
                set returned = 1
                where borrowInfoID = %s;'''
        param = (str(borrowInfoID),)
        success = self.dbControler.Execute(sql,param)
        return success
```

4.5 管理员功能实现（Manager）

4.5.1 新书籍入库（AddNewBook）

传入参数	a) bookName b) author c) publisher d) total e) onShelf f) type g) location
处理过程	a) 将获取到的新书插入数据库 book 表 b) 获取刚刚插入信息的自增生成的 bookID c) 插入 manageInfo 表，记载插入信息
返回值	bookID
功能	插入一条新书，并记录此操作

```
    def AddNewBook(self, bookName,author,publisher, total, onShelf, type,
location):
        '''新书籍入库'''
```

```
        if total != onShelf:
            return False
        sql = '''insert into book
(bookName,author,publisher,type,location)
                values (%s,%s,%s,%s,%s);'''
        param = (bookName,author,publisher,type,location,)
        if not self.dbControler.Execute(sql,param):
            return False
        sql = '''select @@IDENTITY'''
        param = ()
        book = self.dbControler.Search(sql,param)
        bookID = book[0][0]
        sql = '''insert into manageInfo
(managerID,bookID,amount,operation)
                values (%s,%s,%s,%s)'''
        param = (str(self.managerID),str(bookID),str(total),'入库',)
        self.dbControler.Execute(sql,param)
        return bookID
```

4.5.2 销毁一本书（DeleteBook）

传入参数	a） bookID b） decrement
处理过程	a） 向 manageInfo 表中插入一条减书记录 b） 触发器自动更新 book 表
返回值	是否删书成功
功能	删除一本已有的书

```
    def DeleteBook(self, bookID, decrement):
        '''销毁一本书'''
        # 销毁被借书?销毁在架书?
        sql = '''insert into manageInfo
(managerID,bookID,amount,operation)
                values (%s,%s,%s,%s);'''
        param = (self.managerID,bookID,decrement,'出库',)
        if not self.dbControler.Execute(sql,param):
            return False
        return True
```

4.5.3 禁用/解禁一个用户（BanReader）

传入参数	a） readerID b） banned
处理过程	在 reader 表中查找并更新 readerID 的记录，将该账户记录为 Banned
返回值	操作是否成功
功能	禁用/解禁一个用户

```
    def BanReader(self, readerID, banned = False):
        '''禁用/解禁一个用户'''
        banned = int(banned)
        sql = '''update reader
                set banned = %s
                where readerID = %s'''
        param = (banned, readerID,)
        if not self.dbControler.Execute(sql,param):
            return False
        return True
```

4.5.4 列表所有违约账户（InvalidAcount）

```
查看逾期账户
+-------------+----------+------------+-------------+--------------------+-------------+--------+
| borrowInfoID | readerID | readerName |  telephone  |      bookName      | overdueTime | banned |
+-------------+----------+------------+-------------+--------------------+-------------+--------+
|      14     |    34    |    Bob     | 18022011302 |   密码学C++实现    |      59     |   1    |
|      15     |    34    |    Bob     | 18022011302 |   密码学C++实现    |      59     |   1    |
|      16     |    34    |    Bob     | 18022011302 |   密码学C++实现    |      59     |   1    |
|      17     |    34    |    Bob     | 18022011302 | 深入理解计算机系统 |      59     |   1    |
|      18     |    34    |    Bob     | 18022011302 |    密码学引论      |      59     |   1    |
+-------------+----------+------------+-------------+--------------------+-------------+--------+
```

传入参数	无
处理过程	在 overdue 视图中查找违约账户信息
返回值	违约账户列表
功能	列表所有违约账户

```
    def InvalidAcount(self):
        '''列表所有违约账户'''
        readers_list = []
        sql = '''select * from overdue;'''
        param = ()
        result = self.dbControler.Search(sql,param)
        if not result:
```

```
            return False
        readers_list = result
        return readers_list
```

5 实习体会

本次数据库实验中，我主要负责了数据库设计、工程代码实现和测试的内容。

本次实验本组选择了图书管理系统的实现，这个系统主要有五个表组成 :读者、管理员、书籍、借阅信息、管理信息，其中比较重要的表是读者、管理员、书籍、借阅信息这四个表，他们集中了大部分的查询、插入、更新操作，而管理信息表的设计初衷是增加表的关联性，同时给管理人员的操作留下记录。

在设计之初，本组同学低估了数据库功能的强大，打算把着重点放在后端实现上，即通过加大后端的复杂度保证系统的健壮性，但之后在本组另一位同学的设计下，数据库可以完成更多的工作，如触发器、过程等，这为后端的设计增加了很多便利性，如在实现读者借阅功能时，后端实现只需要插入 borrowInfo 一个表即可，对 book 表的更新则将由前表的触发器代为完成。这一点说明了数据库管理的重要性。

后端设计的主要目的是为前端开发提供接口，但本组并未进行前端的实现，因为我们认为前端的实现已经脱离了数据库设计的范畴。但同时，利用 python 方便测试的特点，我们对后端提供的接口进行了反复的测试，测试了如书籍在 onShelf 为零时借阅、用户被 ban 时的借阅、同一读者借阅相同书多次并同时归还等特殊情况，也修改了一些存在的 bug。通过测试的方法，我们保证了系统的可用性。

数据库的使用为软件数据的管理提供了方便，但也对数据库的安全性提出了要求。我们的系统中，用户登陆的密码在数据库中以明文的形式存放，但在工程实现中，密码在传输、存储过程中都可以用 md5 或其他加密方法进行加密处理，以提供数据库的安全性。另一方面，数据库的数据安全也很重要，比如在本次实验中，我们曾不慎损坏一次数据库数据，虽然只是实验数据，但也拖慢了实验进度。因此在工程实现中，数据库的使用也要保证无价数据的安全，以免造成更大的损失。

数据库的真正工程实现比实验更多变、面临着更加严肃的挑战，应合理运用本次实验的经验，为之后更复杂的情况做好准备。