

**《数据库设计与开发》**

**课程设计**

**题 目 通讯录管理系统**

姓 名

学 号

院 系

专 业

**二Ｏ一九 年 七 月 七 日**

目 录

[第一章 概述 4](#_Toc13573991)

[1.1目的与要求 4](#_Toc13573992)

[1.2设计环境 4](#_Toc13573993)

[第二章 需求分析 5](#_Toc13573994)

[2.1系统功能要求设计 5](#_Toc13573995)

[2.2具体功能需求 5](#_Toc13573996)

[2.3数据需求 5](#_Toc13573997)

[2.4系统模块设计 6](#_Toc13573998)

[第三章 总体设计 7](#_Toc13573999)

[3.1数据字典 7](#_Toc13574000)

[3.2关系模式： 8](#_Toc13574001)

[3.3 确定关系模型的存取方法 8](#_Toc13574002)

[3.4 确定数据库的存储结构 8](#_Toc13574003)

[第四章 数据库结构示意图 10](#_Toc13574004)

[4.1结构设计 10](#_Toc13574005)

[第五章 系统实现 12](#_Toc13574006)

[5.1数据库的实施 12](#_Toc13574007)

[5.1.1连接数据库 12](#_Toc13574008)

[5.1.2管理员账号注册 13](#_Toc13574009)

[5.1.3管理员账号登陆 14](#_Toc13574010)

[5.1.4查询所有图书 15](#_Toc13574011)

[5.1.5增加图书信息 16](#_Toc13574012)

[5.1.6编辑图书信息 16](#_Toc13574013)

[5.1.7删除图书信息 17](#_Toc13574014)

[5.1.8存储操作记录 18](#_Toc13574015)

[5.1.9查询操作记录 18](#_Toc13574016)

[第六章 功能测试 19](#_Toc13574017)

[6.1页面操作 19](#_Toc13574018)

[6.1.1登陆、注册页面 19](#_Toc13574019)

[6.1.2图书查询页面 19](#_Toc13574020)

[6.1.3条件查询页面 20](#_Toc13574021)

[6.1.4图书添加页面 20](#_Toc13574022)

[6.1.5图书编辑页面 20](#_Toc13574023)

[6.1.6操作记录查询页面 21](#_Toc13574024)

[6.2数据库维护 21](#_Toc13574025)

[第七章 安全性测试 21](#_Toc13574026)

[第八章 总结 22](#_Toc13574027)

第一章 概述

1.1目的与要求

随着科技的发展，基本上所有的具有一定数量数据的机构都开始使用计算机数据库来做管理。几乎所有图书馆也都已经在使用计算机管理数据的机制，大大减少了图书管理的工作量。该课程设计旨在设计一个图书信息的数据库管理系统，数据库中要求包含管理员的基本信息，图书基本信息，以及管理员对图书的操作信息。要方便管理员进行图书查询等操作，通过该课程设计，应该达到把数据库理论知识更加的巩固加深，加强动手能力与实践能力，学以致用，与现实生活中的应用充分的结合起来。

1.2设计环境

①Mysql-5.7.26

②Eclipse

第二章 需求分析

2.1系统功能要求设计

此系统实现如下系统功能：

（1）使得图书的管理工作更加清晰、条理化、自动化。

（2）管理员注册账号（管理员账号注册需要相应的邀请码）、登陆账号、完成对图书的增加、删除、修改、查找操作和对操作记录的浏览。

(3) 设计人机友好界面，功能安排合理，操作使用方便，并且进一步考虑系统在安全性，完整性，并发控制，备份和恢复等方面的功能要求。

2.2具体功能需求

1、注册账号

主要完成管理员注册账号的功能，因管理员账号不能随便拥有，所以在管理员账号注册时有所限制，需要另一个管理员的邀请码。整个系统中，root管理员是一开始就具有的，无需注册。

2、登陆账号

主要完成管理员登录的功能，如果管理员想要对图书数据进行修改或调整，就必须登录。登录时必须正确添写用户名和密码。如果管理员没有注册。登录界面有到注册界面的连接。

3、图书增加

管理员可以对图书信息进行调整。根据输入的信息进行图书的增加操作。

4、图书删除

管理员可以对图书信息进行调整。根据查询到的图书信息，可以对其进行删除操作。

5、图书修改

管理员可以对图书信息进行调整。根据查询到的图书信息，可以对其进行修改内容操作。

6、图书查询

管理员可以对图书信息进行调整。通过输入一系列关键词进行筛选，查询出相应的图书信息。

7、操作记录查询

管理员可以对操作记录进行查看。可以得知是哪个管理员进行了哪些操作。

2.3数据需求

数据需求，即在系统中需要对哪些数据进行处理。根据对图书管理系统的分析，该系统主要的数据需求如图所示。

|  |  |  |
| --- | --- | --- |
| 数据输入 | 数据处理 | 数据反馈 |
| 管理员输入个人信息进行注册 | 将新数据存入数据库 | 如注册成功等 |
| 管理员进行登录 | 检索数据库进行信息匹配 | 跳转到登录成功后的页面 |
| 管理员输入密码 | 更新数据库内容 | 如密码修改成功等 |
| 管理员输入查询信息 | 检索数据库 | 符合查询内容的图书信息 |
| 管理员输入图书信息 | 将新数据存入数据库 | 添加成功等、图书信息显示 |
| 管理员对图书信息修改 | 检索数据库得到图书信息记录，增加管理员请求的修改记录，更新数据库 | 修改成功等、记录操作 |
| 管理员对图书信息删除 | 检索数据库得到图书信息记录，删除管理员请求的修改记录，更新数据库 | 删除成功等、记录操作 |

2.4系统模块设计

图书管理模块可以分成两个模块，管理员信息模块，图书信息模块。

管理员信息模块：主要包括管理员的基本信息，如账号、密码、邀请码（自动生成）。

图书信息模块：主要包括图书的基本信息，如编号（自动生成）、书名、作者、价格、类型、ISBN号、出版社、总册数、剩余册数。

操作记录模块：主要包括记录的基本信息，如记录编号、管理员编号、图书编号、具体操作。可以得到系统流程图：

图书管理

退出系统

管理员登陆

登陆失败

登陆成功

第三章 总体设计

3.1数据字典

数据项是数据库的关系中不可再分的数据单位，下表分别列出了数据的名称、数据类型、长度、取值能否为空。利用Mysql-5.7.26建立“图书管理系统”数据库，其基本表清单及表结构描述如下：

数据库中用到的表：

|  |  |  |
| --- | --- | --- |
| 数据库表名 | 关系模式名称 | 备注 |
| Administrator | 管理员 | 管理员账号信息表 |
| Book | 图书 | 图书基本信息表 |

Administrator数据表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| Admin\_id | Int | Primary key | 管理员编号 |
| Admin\_name | Char | Not null | 用户名 |
| Admin\_pwd | Char | Not null | 密码 |
| Admin\_code | Char | Not null | 邀请码 |

Book数据表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| Book\_id | Char | Primary key | 图书编号 |
| Book\_name | Char | Not null | 书名 |
| Book\_author | Char | Not null | 作者 |
| Book\_price | Numeric | Not null | 价格 |
| Book\_type | Char | Not null | 类别 |
| Book\_isbn | Char | Not null | ISBN号 |
| Book\_publisher | Char |  | 出版社 |
| Book\_allnum | Int | >=0 | 总册数 |
| Book\_surplus | Int | >=0 AND <=allnum | 剩余量 |

Record数据表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束控制 | 说明 |
| Record\_idi | Int | Primary key | 记录编号 |
| Admin\_id | Int | Foreign key | 管理员编号 |
| Book\_id | Char | Foreign key | 图书编号 |
| Record\_msg | varchar | Not null | 具体操作 |

3.2关系模式：

（1）管理员（编号，用户名，密码，邀请码），其主关键字为编号;

（2）图书（编号，书名，作者，价格，类别，ISBN，出版社，总数，剩余量），其中主关键字为编号；

（3）记录（记录编号，管理员编号，图书编号，操作记录）其中主关键字为记录编号，管理员编号（外部键），图书编号（外部键）。

3.3 确定关系模型的存取方法

在将概念模型转换成物理模型之后，我们可以对物理模型进行设计，双击物理模型的关系，可以对该关系的名称、注释等信息进行查询。可对该关系的属性列进行设计，可分别设置其名称、码、数据类型以及主码、是否为空等。在实际设计中最常用的存取方法是索引发，使用索引可以大大减少数据的查询时间，在建立索引时应遵循：在经常需要搜索的列上建立索引；在主关键字上建立索引；在经常用于连接的列上建立索引，即在外键上建立索引；在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的等规则。才能充分利用索引的作用避免因索引引起的负面作用。

3.4 确定数据库的存储结构

确定数据库的存储结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、日志、备份等的存储安排及存储结构，以及确定系统存储参数的配置。因为该图书管理系统的数据量小，所以我们只用把数据存储在使用的电脑硬盘上，不用作更多的安排。

（1）创建管理员基本信息表:

CREATE TABLE `administrator` (

`Admin\_id` int(11) NOT NULL AUTO\_INCREMENT,

`Admin\_name` varchar(40) NOT NULL,

`Admin\_pwd` varchar(40) NOT NULL,

`Admin\_code` varchar(6) NOT NULL,

PRIMARY KEY (`Admin\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

（2）创建图书基本信息表:

CREATE TABLE `book` (

`Book\_id` varchar(8) NOT NULL,

`Book\_name` varchar(20) NOT NULL,

`Book\_author` varchar(20) NOT NULL,

`Book\_price` decimal(10,2) NOT NULL,

`Book\_type` varchar(20) NOT NULL,

`Book\_isbn` varchar(20) NOT NULL,

`Book\_publisher` varchar(40) DEFAULT NULL,

`Book\_allnum` int(10) NOT NULL,

`Book\_surplus` int(10) NOT NULL,

PRIMARY KEY (`Book\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（3）创建操作记录信息表

CREATE TABLE `record` (

`Record\_id` int(8) NOT NULL AUTO\_INCREMENT,

`Admin\_id` int(8) NOT NULL,

`Book\_id` char(8) NOT NULL,

`Record\_msg` varchar(40) NOT NULL,

PRIMARY KEY (`Record\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（4）创建约束

CREATE TRIGGER `tri\_book` BEFORE INSERT ON `book` FOR EACH ROW BEGIN

IF new.Book\_price < 0 THEN

DELETE FROM book where book\_price = new.book\_price;

END IF;

IF new.book\_allnum < 0 THEN

DELETE FROM book WHERE book\_allnum = new.book\_allnum;

end IF;

IF NEW.book\_surplus<0 OR NEW.book\_surplus>NEW.book\_allnum THEN

DELETE FROM book WHERE book\_surplus = NEW.book\_surplus;

END IF;

END

第四章 数据库结构示意图

4.1结构设计

由需求分析的结果可知，本系统设计的实体包括：

（1）管理员基本信息：管理员编码、账号、密码、邀请码。

（2）图书基本信息：图书编号、书名、作者、价格、类别、ISBN号、出版社、总册数、剩余量。

这些实体间的联系包括：

（1）每位管理员可以管理多本图书，每本图书可供多位管理员管理。

由上述分析可得到系统的E—R图：

1）用长方形表示实体集，长方形内写明实体集名称

2）用椭圆形表示实体集的属性，并用线段将其与相应的实体集连接起来

3）用菱形表示实体集间的联系，菱形内写上联系名，用线段分别与有关实体集连接起来，在线段旁标出联系的类型。

管理员

编号

用户名

密码

邀请码

图3-1 管理员信息E-R图

图书

剩余量

编号

书名

作者

价格

类别

ISBN号

出版社

总册数

图3-2 图书信息E-R图

记录

记录编号

管理员编号

图书编号

操作记录

图3-3 记录信息E-R图

N

N

查询

管理员

编号

用户名

密码

邀请码

图书

剩余量

编号

书名

作者

价格

类别

ISBN

出版社

总数

记录

编号

管理员编号

操作记录

图书编号

记录编号

图3-4 整体E-R图

第五章 系统实现

5.1数据库的实施

此阶段主要任务包括创建数据库，加载初始数据，数据库试运行，数据库的安全性和完整性控制数据库的备份与恢复，数据库性能的监督分析和改进，数据库的重组和重构等。

5.1.1连接数据库

首先在数据库中建立一个图书管理系统数据库，然后新建一个数据源。在Eclipse中创建项目并对数据库进行连接。具体代码如下：

**public** **class** JDBCUtils {

//加载驱动，并建立数据库连接

**public** **static** Connection getConnection() **throws** ClassNotFoundException, SQLException {

Class.*forName*("com.mysql.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/BMS?useUnicode=true&characterEncoding=utf-8";

String username = "root";

String password = "root";

Connection conn = DriverManager.*getConnection*(url, username, password);

**return** conn;

}

//关闭数据库连接，释放资源

**public** **static** **void** release(Statement stmt, Connection conn) {

**if**(stmt != **null**) {

**try** {

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

stmt = **null**;

}

**if**(conn != **null**) {

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

conn = **null**;

}

}

**public** **static** **void** release(ResultSet rs, Statement stmt, Connection conn) {

**if**(rs != **null**) {

**try** {

rs.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

rs = **null**;

}

*release*(stmt, conn);

}

}

5.1.2管理员账号注册

管理员通过页面表单提交进行账号注册，在注册时需要另一个管理员的邀请码，并在注册后随机获得一组邀请码。Root用户一开始就存在与管理员表内，无需注册。具体代码如下：

**public** **boolean** insert(Administrator admin) {

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs =**null**;

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

//查询该用户所用邀请码是否在数据库中

String old\_code = admin.getAdmin\_old\_code();

String sql1 = "select Admin\_code from Administrator "

+ "where Admin\_code ='"+ old\_code+"'";

rs = stmt.executeQuery(sql1);

**if**(rs.next()) {//邀请码存在

//随机分配一个邀请码给新注册的账号

**int** code = GenerateCode.*code*();

String sql2 = "insert into Administrator(Admin\_id,Admin\_name,Admin\_pwd,Admin\_code)"

+ "values(" + **null**

+ ",'"

+ admin.getAdmin\_name()

+ "','"

+ admin.getAdmin\_pwd()

+ "','"

+ code

+ "')";

**int** num = stmt.executeUpdate(sql2);

**if**(num > 0) {

**return** **true**;//注册成功

}

**return** **false**;//注册失败

} **else** {//邀请码不存在

**return** **false**;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(rs, stmt, conn);

}

**return** **false**;

}

5.1.3管理员账号登陆

管理员通过页面表达提交进行登陆账号。将表单内容放入数据库进行查询，通过查询结果判断是否登陆。具体代码如下：

**public** **boolean** judegLogin(Administrator admin) {

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

String sql = "select Admin\_name,Admin\_pwd from Administrator where Admin\_name ='"

+ admin.getAdmin\_name()

+ "'and Admin\_pwd ='"

+ admin.getAdmin\_pwd()

+ "'";

rs = stmt.executeQuery(sql);

**if**(rs.next()) {

**return** **true**;

} **else** {

**return** **false**;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(rs, stmt, conn);

}

**return** **false**;

}

5.1.4查询所有图书

管理员可以在页面查询所有图书信息。具体代码如下：

**public** List<Book> getBook(){

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

List<Book> bookList = **new** ArrayList<Book>();

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

String sql = "select \* from Book";

rs = stmt.executeQuery(sql);

**while**(rs.next()) {

Book book = **new** Book();

book.setBook\_id(rs.getString("Book\_id"));

book.setBook\_name(rs.getString("Book\_name"));

book.setBook\_author(rs.getString("Book\_author"));

book.setBook\_price(rs.getDouble("Book\_price"));

book.setBook\_type(rs.getString("Book\_type"));

book.setBook\_isbn(rs.getString("Book\_isbn"));

book.setBook\_publisher(rs.getString("Book\_publisher"));

book.setBook\_allnum(rs.getInt("Book\_allnum"));

book.setBook\_surplus(rs.getInt("Book\_surplus"));

bookList.add(book);

}

**return** bookList;

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(rs, stmt, conn);

}

**return** **null**;

}

5.1.5增加图书信息

管理员可以在页面增加图书信息。具体代码如下：

**public** **boolean** addBook(Book book) {

String sql = "insert into book values(?,?,?,?,?,?,?,?,?)";

Connection conn = **null**;

Statement stmt = **null**;

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

QueryRunner runner = **new** QueryRunner();

**int** num = runner.update(conn, sql, book.getBook\_id(), book.getBook\_name(),

book.getBook\_author(), book.getBook\_price(), book.getBook\_type(),

book.getBook\_isbn(), book.getBook\_publisher(), book.getBook\_allnum(),

book.getBook\_surplus());

**if**(num > 0) {

**return** **true**;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(stmt, conn);

}

**return** **false**;

}

5.1.6编辑图书信息

管理员可以在页面编辑图书信息。具体代码如下：

**public** **void** editBook(Book book) {

Connection conn = **null**;

Statement stmt = **null**;

List<Object> obj = **new** ArrayList<Object>();

obj.add(book.getBook\_name());

obj.add(book.getBook\_author());

obj.add(book.getBook\_price());

obj.add(book.getBook\_type());

obj.add(book.getBook\_isbn());

obj.add(book.getBook\_publisher());

obj.add(book.getBook\_allnum());

obj.add(book.getBook\_surplus());

obj.add(book.getBook\_id());

String sql = "update book set" +

" Book\_name='" + book.getBook\_name() + "',"

+ " Book\_author='" + book.getBook\_author() + "',"

+ " Book\_price='"+ book.getBook\_price() + "',"

+ " Book\_type='" + book.getBook\_type() + "',"

+ " Book\_isbn='"+ book.getBook\_isbn() +"',"

+ " Book\_publisher='" + book.getBook\_publisher() + "',"

+ " Book\_allnum='" + book.getBook\_allnum() + "',"

+ " Book\_surplus='" + book.getBook\_surplus() + "'"

+ " where Book\_id='" + book.getBook\_id() + "'";

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

stmt.executeUpdate(sql);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(stmt, conn);

}

}

5.1.7删除图书信息

管理员可以在页面删除图书信息。具体代码如下：

**public** **void** deleteById(String Book\_id){

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

String sql = "delete from book where Book\_id='" + Book\_id + "'";

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

stmt.executeUpdate(sql);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(rs, stmt, conn);

}

}

5.1.8存储操作记录

在管理员进行一个动作（编辑，添加，删除）后，会自动记录该管理员的编号、所操作图书的编号和具体操作，方便管理员随时查看。具体代码如下：

**public** **void** setRecord(Record r) {

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs =**null**;

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

String sql = "insert into record(Record\_id,Admin\_id,Book\_id,Record\_msg)"

+ "values("

+ **null**

+ ",'"

+ r.getAdmin\_id()

+ "','"

+ r.getBook\_id()

+ "','"

+ r.getRecord\_msg()

+ "')";

stmt.execute(sql);

} **catch** (Exception e) {

e.printStackTrace();

}

}

5.1.9查询操作记录

在管理员可以随时查看之前所做的操作记录，方便更清晰的管理图书信息。具体代码如下：

**public** List<Record> getRecord(){

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

List<Record> reList = **new** ArrayList<Record>();

**try** {

conn = JDBCUtils.*getConnection*();

stmt = conn.createStatement();

String sql = "select \* from record";

rs = stmt.executeQuery(sql);

**while**(rs.next()) {

Record record = **new** Record();

record.setRecord\_id(rs.getString("Record\_id"));

record.setAdmin\_id(rs.getString("Admin\_id"));

record.setBook\_id(rs.getString("Book\_id"));

record.setRecord\_msg(rs.getString("Record\_msg"));

reList.add(record);

}

**return** reList;

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JDBCUtils.*release*(rs, stmt, conn);

}

**return** **null**;

}

第六章 功能测试

6.1页面操作

6.1.1登陆、注册页面

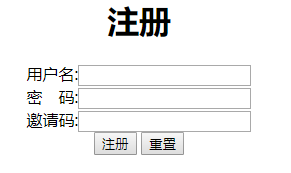
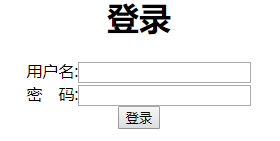


图5-1 登陆、注册页面

6.1.2图书查询页面



图5-2 图书查询页面

6.1.3条件查询页面



图5-3 条件查询页面

6.1.4图书添加页面

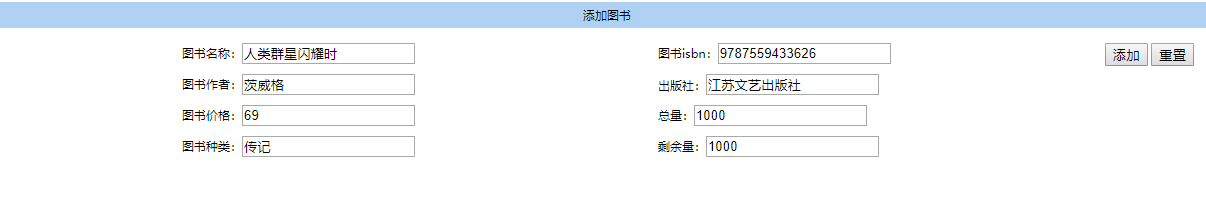
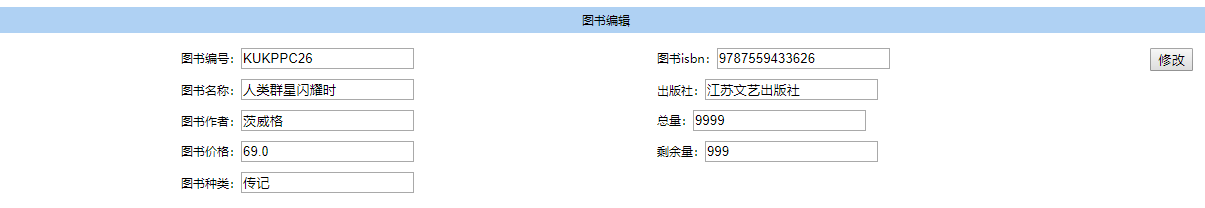




图5-4 图书添加页面

6.1.5图书编辑页面



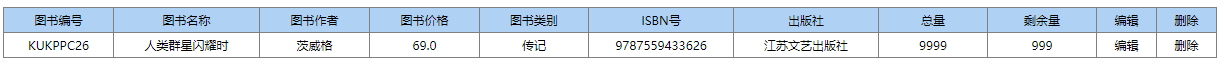


图5-5 图书编辑页面

6.1.6操作记录查询页面

图5-6 操作记录查询页面

6.2数据库维护

当试运行数据库合格后，数据库开发设计的工作就基本完成了，接下来就是正式运行中的调试，应为该系统比较简单，数据量小，数据库中几乎不会发生什么大的变化，但是还是需要做好数据的备份，在Mysql中我们可以利用备份数据库的功能对已经设计好的数据做备份，如果数据库受到破坏或系统故障，我们便可轻松的利用备份文件恢复数据库的数据。当然也可以利用其他各种方法进行数据维护。

第七章 安全性测试

常见的注入场景是系统的登录功能，当然现在很多大公司的项目在这一块是做的很好，也做的非常安全的，很少会留下sql注入这样的漏洞，而一些小的系统，小的项目或者自己写着玩的系统可能会存在这样的安全漏洞。通过在登录页面输入用户名和密码来完成登录，后台会拿到前台表单提交的数据去数据库做校验匹配，如果存在此用户就让登录成功，如果匹配不到就提示“错误的用户名或者密码”。

我们本来用户密码均为admin，接下来我们假设知道用户名的情况下没有密码利用sql注入一样可以进入系统。我们使用最简单的万能密码”’or1=1 --”点击登陆显示登陆成功，成功注入。



第八章 总结

我在本次的数据库课程设计中，一开始并没有认真进行需求分析以及数据库的设计，而是急于去编码实现界面等前台的东西。结果编到第二个界面的时候，发现数据库设计不合理，只好重新回去修改。我的解决方法是：详细进行需求分析，争取做到对整个系统了然于胸；详细进行系统功能划分，分到最小模块为止，然后再一个一个模块添加代码即可，当然，要考虑模块与模块之间的耦合程度以及模块与模块之间的信息共享。在本次数据库课程设计中遇到不少困难，在这时，我一般采取以下措施：上网查资料；在网上论坛上提问题；和同学讨论；在图书馆查找相关图书资料。总之，这些是我比较常用的方法。最重要的一点是，遇到困难时的心态要平和，冷静地去查找问题并解决问题。最近，其实不止是本次数据库课程设计带给我的收获，就是作为一个合格的程序员，要注意对知识的积累和知识体系的梳理，每隔一段时间就应该对自己近期的工作有个小结和反思，只有这样，才能不断进步。