《图书管理系统实验报告》

课程设计报告

题目	图书销售管理系统		
小组成员信息			
姓名	**学号**	**班级**	**分工**
陈土丽	13349011	计科 1 班	实验报告，数据库设计
劳嘉辉	13349051	计科 1 班	程序设计

提交时间：2016 年 1 月 17 日

目录

一、开发环境与开发工具

开发语言：MFC

开发工具：vs2015

数据库系统：Mysql Server 5.5

开发操作系统：Windows 7 & Windows 10

本软件适合系统：所有平台

二、系统需求分析

随着现代图书销售网络越来越发达，不仅可以到实体店购买图书，网上购买的方式更受年轻人欢迎，图书销售遍布全球，图书销售管理的数据也就不能像以前一样全靠人工进行管理，图书销售管理系统也就应运而生了。

要通过图书销售管理系统对图书销售进行全面的管理，图书销售管理系统就应具备以下功能：

1） 进货：系统应能显示图书的库存量、销售情况及供应商对该图书的报价等，以便于职工或管理员根据实际情况进行进货。

2） 退货：现实中难免会有顾客退货的情况发生，系统应能根据退货信息修改图书的销售情况及库存量。

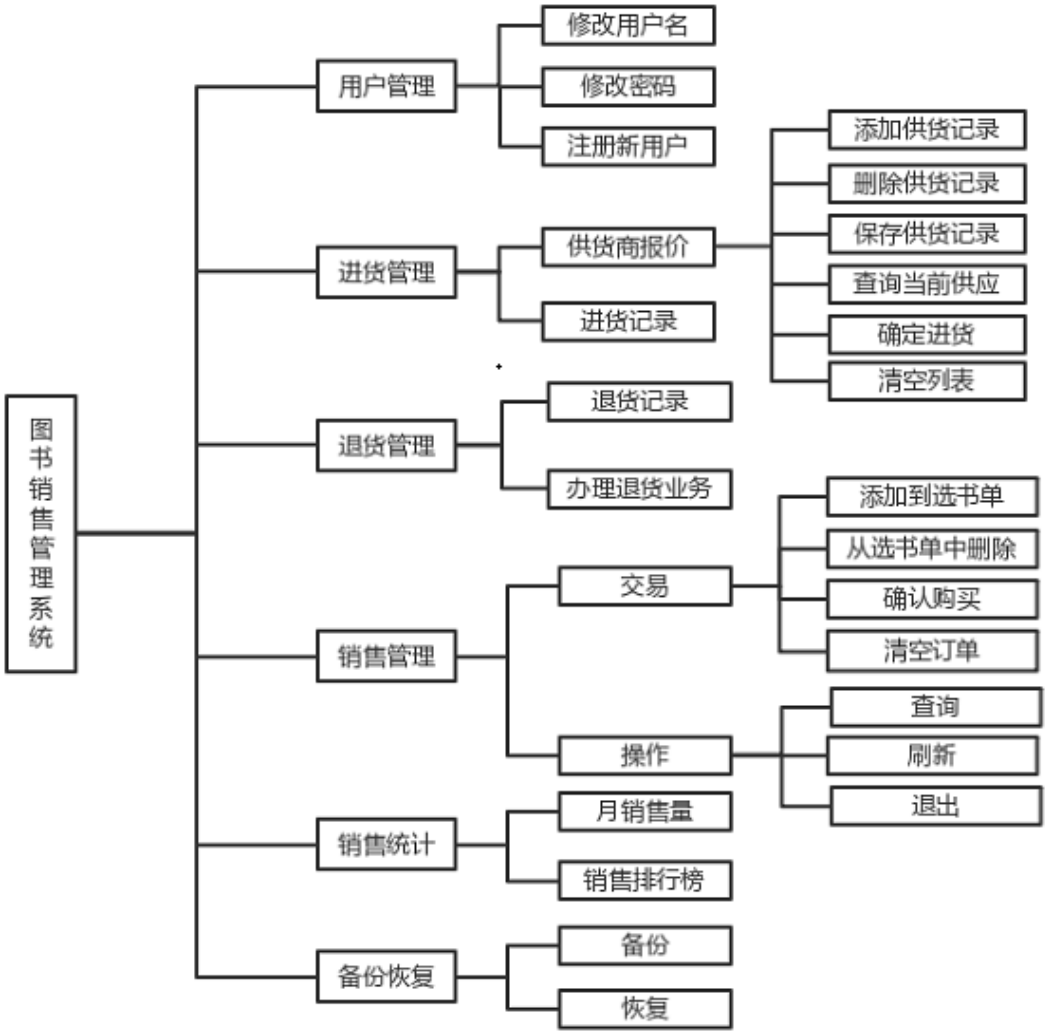
3） 统计：系统应具备对图书销售量、销售总额进行统计排行的功能。

4） 销售：图书销售的过程中，系统可以提供图书查询、会自动修改图书销售情况及库存量。

5） 用户管理：对于登录图书销售管理系统的不同用户，应当有不同的权限，管理员有全部权限，职工有销售方所应具有的权限，如：查看图书库存量、进货、销售、退货管理等，供货商有添加供货记录等权限，而不能有销售管理的权限。

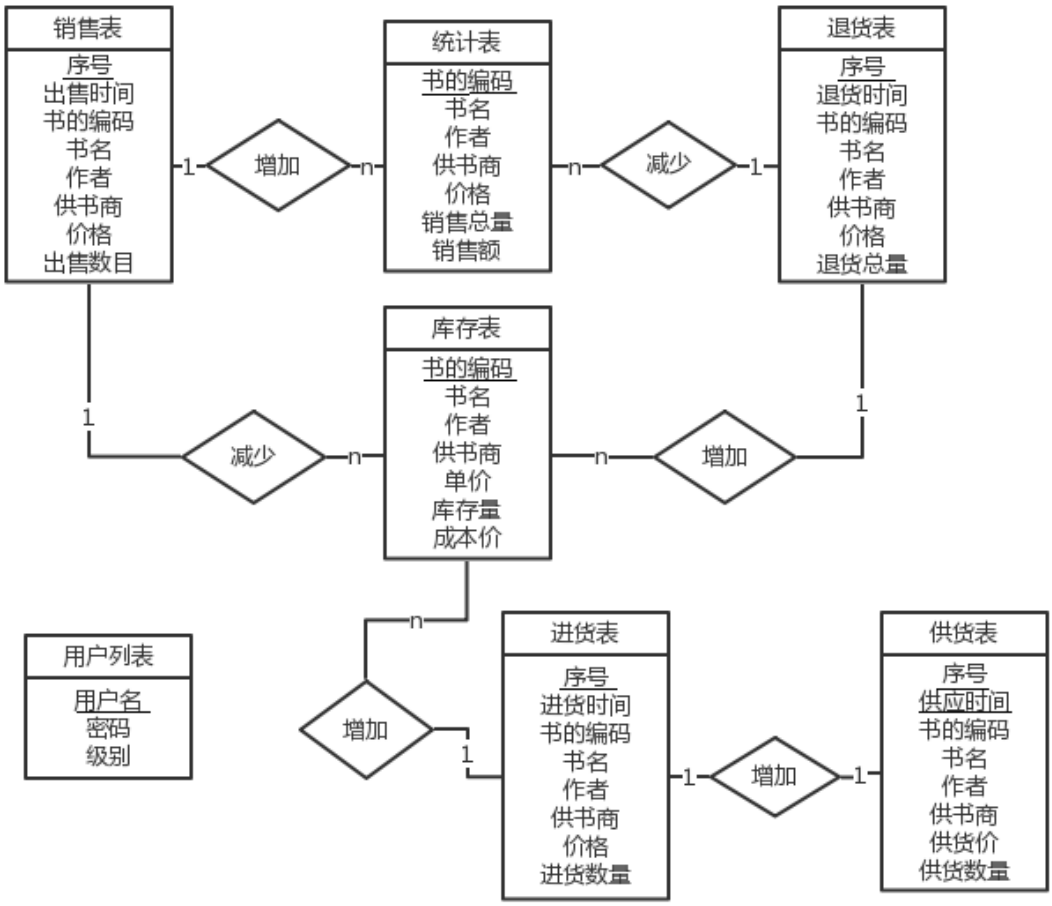
6） 备份恢复：对于图书销售管理系统的数据，为避免丢失，系统应具备备份和恢复的功能。

三、功能需求分析



系统功能模块图

1、数据概念结构设计（包括数据流程图、系统 ER 图）



系统 ER 图

2、数据库关系模式设计

用户列表（用户名，密码，级别）

库存表（ISBN，书名，作者，供书商，单价，库存量，成本价）

供货表（序号，供应时间，ISBN，书名，作者，供书商，供货价，供货数量）

进货表（序号，进货时间，ISBN，书名，作者，供书商，价格，进货数量）

统计表（ISBN，书名，作者，供书商，价格，销售总量，销售额）

销售表（序号，出售时间，ISBN，书名，作者，供货商，价格，出售数目）

退货表（序号，退货时间，ISBN，书名，作者，供货商，价格，退货总量）

可以看到这里有很多冗余的地方，所以我们后来想到可以改进一下这个关系模式如下：

用户列表（用户名，密码，级别）

库存表（ISBN，书名，作者，单价，库存量）

供货表（序号，供应时间，ISBN，书名，作者，供书商，供货价，供货数量）

进货表（序号，进货时间，ISBN，进货数量）

统计表（ISBN，销售总量，销售额）

销售表（序号，出售时间，ISBN，出售数目）

退货表（序号，退货时间，ISBN，退货总量）

在销售表、退货表、统计表与库存表之间可以有一个 ISBN 的外键约束，通过创建视图将这几个表联系起来，在界面输出显示的时候就可以显示出跟前面一样的信息。同样的，进货表和供货表之间也可以有一个 ISBN 的外键约束。

（但最后由于时间关系也并没有把改进后的关系模式应用到数据库系统中。）

3、数据库物理结构设计

本系统数据库表的物理设计通过创建表的 sql 命令来呈现，下面只列出 sql 创建命令，针对其他数据库系统的创建命令就不一一列出来了。

创建数据库表的 sql 命令如下，使用 UTF8 字符集：

```
Create database booksalesmanagement character set utf-8;
Use booksalesmanagement;
//创建用户列表
create table UserTable(
username varchar(100) NOT NULL,
password varchar(100) ,
rank ENUM('职工','供货商','管理员'),
Primary key(username)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
//创建图书库存表
create table BookStorage(
```

```
ISBN char(13) NOT NULL,
bookname varchar(300) NOT NULL,
author varchar(200),
bookconcern varchar(200),
price float(8) NOT NULL,
storecount int(4) NOT NULL,
cost float(8) NOT NULL,
Primary Key(ISBN)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
//图书供应列表
create table supplier(
sno int(8) NOT NULL AUTO_INCREMENT,
supplytime timestamp NOT NULL,
ISBN char(13) NOT NULL,
bookname varchar(300) NOT NULL,
author varchar(200) ,
supplier varchar(300) NOT NULL,
supplycost float(8) NOT NULL,
supplycount int(4) NOT NULL,
Primary Key(sno,supplytime)
)Engine=InnoDB DEFAULT CHARSET=utf8;
//进货列表
create table BookStock(
sno      int(8)     NOT NULL AUTO_INCREMENT,
stocktime   timestamp   NOT NULL,
ISBN char(13) NOT NULL,
bookname  varchar(300),
author varchar(200),
bookconcern     varchar(200),
```

```
cost      float(8)  NOT NULL,
stockcount  int(4)     NOT NULL,
Primary Key(sno)
)Engine=InnoDB DEFAULT CHARSET=utf8;
//销售统计表
create table BooKReport(
ISBN char(13) NOT NULL,
bookname  varchar(300) NOT NULL,
author varchar(200),
bookconcern     varchar(200),
price    float(8)  NOT NULL,
soldcount     int(4)   NOT NULL,
soldsum     float(8)  NOT NULL,
Primary Key(ISBN)
)Engine=InnoDB DEFAULT CHARSET=utf8;
//图书销售列表
create table booksold(
sno      int(8)     NOT NULL AUTO_INCREMENT,
soldtime     timestamp    NOT NULL,
ISBN char(13) NOT NULL,
bookname  varchar(300) NOT NULL,
author varchar(200),
bookconcern     varchar(200),
price    float(8)  NOT NULL,
soldcount     int(4)   NOT NULL,
Primary Key(sno)
)Engine=InnoDB DEFAULT CHARSET=utf8;
//退货表
create table bookrefund(
```

```
sno      int(8)     NOT NULL AUTO_INCREMENT,
refundtime timestamp    NOT NULL,
ISBN     char(13) NOT NULL,
bookname  varchar(300) NOT NULL,
author varchar(200),
bookconcern     varchar(200),
price    float(8)  NOT NULL,
refundcount      int(4)     NOT NULL,
Primary Key(sno)
)Engine=InnoDB DEFAULT CHARSET=utf8;
//下面是一些实现完整性约束的触发器和存储过程
delimiter //
create trigger update_after_refund after insert on bookrefund
for each row
begin
      update bookstorage set storecount=storecount+new.refundcount where ISBN=new.ISBN;
      update bookreport set soldcount=soldcount-new.refundcount where ISBN=new.ISBN;
      update bookreport set soldsum=soldcount*price;
end//

delimiter //
create procedure insert_into_refund(refundtime timestamp,ISBN char(13),bookname varchar(50),author varchar(20), bookconcern varchar(20),price float(8),refundcount int(4))
Begin
          if(refundcount <= (select bookreport.soldcount from bookreport where bookreport.ISBN=ISBN)) then
```

```
            insert into bookrefund (refundtime, ISBN, bookname, author,
bookconcern, price, refundcount)
values(refundtime,ISBN,bookname,author,bookconcern,price,refundcount);
        end if;
End;//
delimiter //
create trigger insert_after_stock after insert on bookstock
for each row
begin
    if((select count(*) from bookstorage where bookstorage.ISBN = new.ISBN)
<> 0) then
        update bookstorage set storecount = storecount + new.stockcount,
cost=new.cost where ISBN=new.ISBN;
    else
        insert into bookstorage values(new.ISBN, new.bookname,
new.author, new.bookconcern, new.cost*1.12,new.stockcount,new.cost);
    end if;
end//
```

四、系统功能的实现

登录界面

实现方法：（Login.cpp）

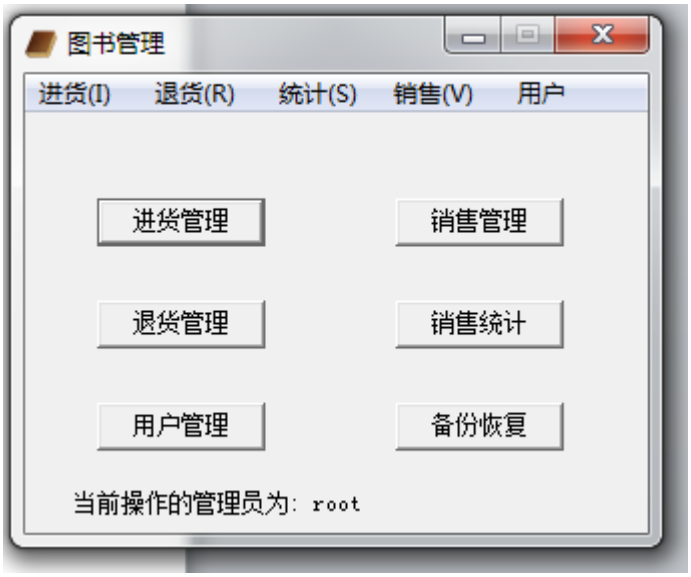
新建一个登陆窗口，设置各种按键和文字编辑框后，读入用户名和密码，并用 MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)函数登陆 MYSQL 数据库。如果出错的话会利用以下函数提示用户：（所有 sql 语句执行后都会用 mysql_errno 函数判断是否出错）

```
if (mysql_errno(&mysql_login))
{
    CString err;
    err.Format("Error: mysql_error(%s)\n", mysql_error(&mysql_login));
    MessageBox(err, "提示！");
    return;
}
```

【注意：首次登录前先以根用户的身份登录数据库】

在数据库中的 usertable 中，修改“管理员”这一行的用户名和密码为自己的根用户名和密码（或者是用所有权限的用户），就可以以管理员的身份登录了。

登录之后可以看到该图书管理系统的功能模块如下：



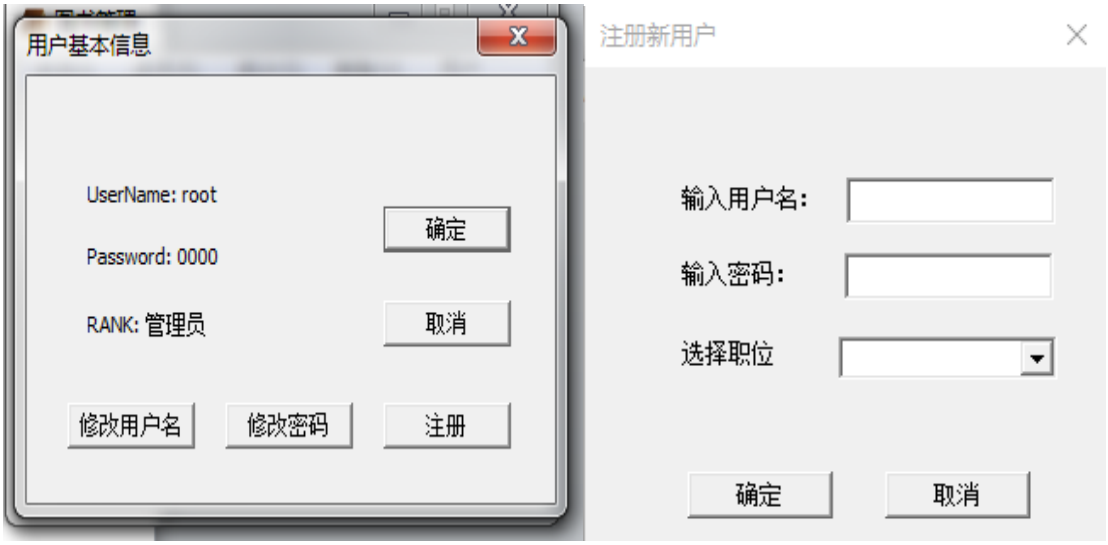
用户管理模块

实现方法：(UserInfo.cpp)

创建了用户基本信息的对话框，先使用 sql 语句查询到当前的用户名，密码以及职位，并用 SetWindowText 设置文字显示。

修改用户名和修改密码分别从文字编辑框读入新用户名和新密码，并使用 sql 语句进行修改。

注册模块读入新用户名和密码，并分别职工和供货商写入不同的授权(with grant option) sql 语句。做到了不同用户的授权处理。



点击用户管理，可以修改用户名和用户密码，并且可以注册新用户。用户分为 3 个级别：管理员、职工和供货商，管理员拥有图书管理系统的全部权限，可

以注册新的职工和供货商，供货商只能注册供货商，职工只能注册职工。

PS:在删除所建数据库重新建一个同名的数据库后，重新注册同名用户发现会出错，原因是删除数据库的时候之前所注册的用户并没有被删除，所以当要删除整个数据库的时候，需要同时删除相应的用户，sql 语句：dorp user username; 也可以不删除，直接沿用之前所注册的用户，但这时在命令行查看数据库用户表的用户里是没有之前所注册的用户的，可以直接在命令行 insert 之前所注册用户的信息进去。但若之前没有注册过的用户，直接在数据库用户列表用 Insert 命令添加信息进去，是不能注册用户成功的，也就是说，注册新用户只能在上图所示的运行界面注册。

进货管理模块

实现方法：StockManager.cpp StockDlg1.cpp StockDlg2.cpp

这里为了简洁美观，所以的话使用了选项卡对话框来显示进货模块的管理。StockManager 该类生成了两个对话框 CStockDlg1 和 CStockDlg2，并将其连接在选项卡中，初始化一些配件和文字显示，并且建立了一些选项卡转换的一些函数。

StockDlg1.cpp 包含了进货记录的实现，即图中第一个选项卡的对话框的实现。这里用到的是 list control 来显示进货记录。每次点击这个选项卡的时候，都会使用查询 sql 语句来更新当前列表的内容，保证看到的进货记录是最新的。

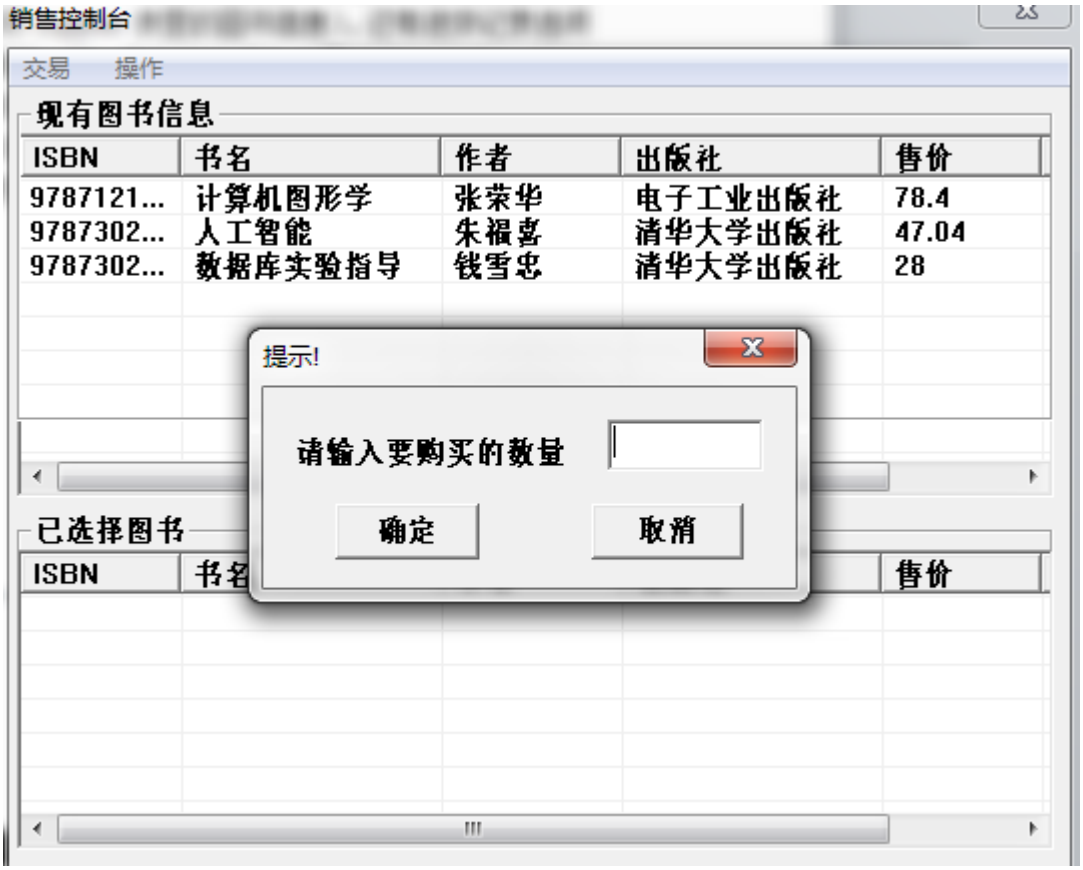
StockDlg2.cpp 包含了进货操作的各种实现。各种按钮的用处可以看下面的解释。每次更新列表时，先调用 list control 中的 DeleteAllItems()先将原有的记录删除，再调用查询 sql 语句得到最新的记录，并调用 InsertItem() 和 SetItemText() 这个函数将新的记录显示到列表中。

进货管理模块中有进货记录和供货商报价两个选项卡，上面还有一个操作，点开可以选择刷新。供货商在这个图书管理系统中的所有权限就在这里了，其权限为：**添加供货记录**（在上面填好图书的 ISBN 等信息，点击添加供货记录即可把上面的信息添加到下面的记录栏）、**清空列表**（就是清空上面所填的 ISBN 等图书信息，以写下一个信息）、**删除供货记录**（就是若是发现下面的记录写错了，在还没保存到数据库前可以删去）、**保存供货信息**（就是把供货记录栏的记录保存 insert 到数据库里（对，是记录栏里的所有记录都保存，即使是之前保存过的也以新的序号保存在数据库的 supplier 表里，相当于又添加了一批）此时查询数据库的 supplier 表就可以看到相应的信息）、**查询当前供应**（就是当前可以提供进货的图书信息，即 supplier 表里的图书信息），还有进货记录选项可以**查看进货记录**（查看的时候记录刷新一下，否则可能看到的是旧的记录哦），此外供货商没有其他权限，这也体现了数据库的安全性。

职工的权限是除了保存供货信息外的其他权限都有，通过查看当前供应，选中当前供应中的某种供应的图书，点击确定进货便完成了一种图书的进货。

销售管理模块

实现方法和进货管理相似。(VendDlg.cpp VendDlg1.cpp VendDlg2.cpp VendManager.cpp)



这里有交易和操作两个选项卡：在交易选项卡中有添加到书单、从选书单中删除、清空订单、确定购买几个功能。

从现有图书信息栏选中所要选择的图书，点击交易，选择添加到书单就会弹出上面的提示框，输入要购买的数量，点击确定便把所选中的图书添加到下面的已选择图书栏中，点击确定购买，便把已选择图书栏里的图书信息写进了数据库的销售表里。

选中已选择图书栏里的图书，点击从书单中删除，便可删除所选择的图书，或者也可以清空订单。

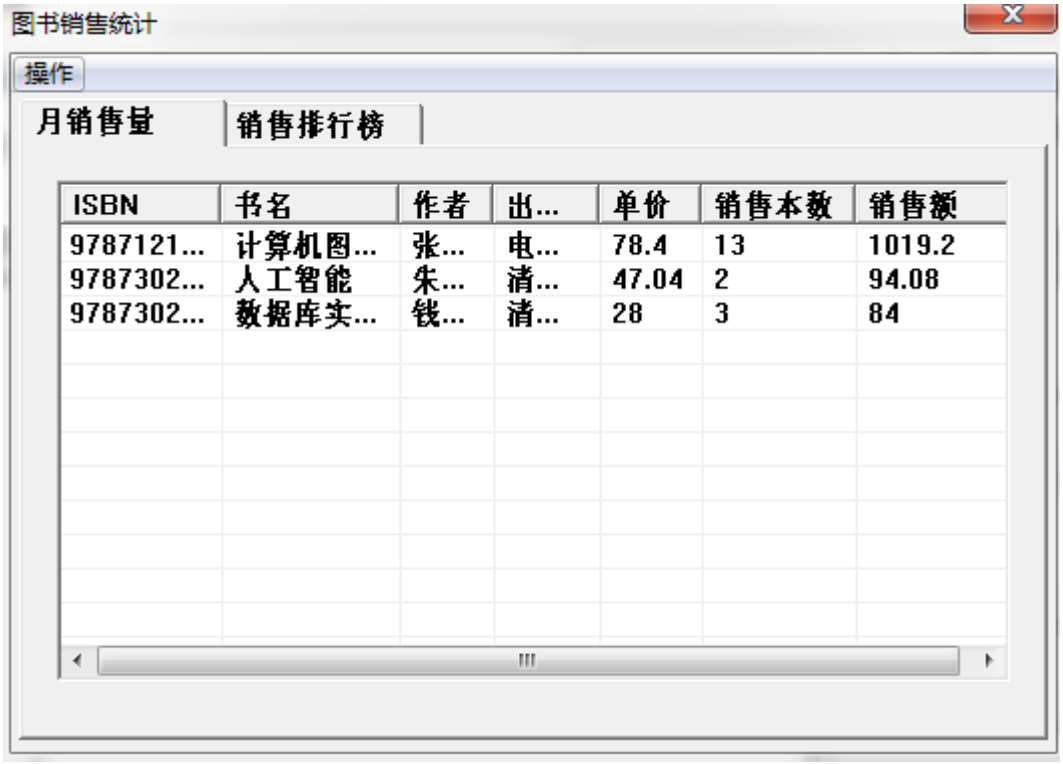
在操作选项卡中有查询、刷新、退出几个功能。

查询为模糊查询，输入部分信息便可查询出符合所写信息要求的书，通过刷新可以返回到所有可选择的图书的界面。

退货管理模块

这里包括退货和查看退货记录两个功能。输入相应的退货信息，提交清单便可完成退货，退货后相应的销售统计、图书库存数据等也会做出相应的更改。

销售统计模块



这里有月销售量和销售排行榜两个统计，排行按销售额排序。

备份恢复模块

实现方法：(SaveData.cpp RestoreData.cpp)

这里调用了 Visual C++(VC)中的 BROWSEINFO 结构弹出文件夹对话框，用于方便用户选择文件路径。

【只有管理员才能备份恢复】

点击备份->点击选择路径（选好备份路径）->开始备份，便发现所选择的路径文件夹中多了备份文件（使用的是表备份，生成的 bak 文件）（若是文件夹中本来就存在备份文件，须先删除）。备份好后，点击恢复就可以进行恢复操作了。若要进行恢复操作，则必须保证该目录下必须包含本程序生成的各种 bak 文件，一般情况下如果不改动的话跟备份的目录是一样的。

（这里的实行表恢复时需要先使用 delete 语句清空原表，否则由于恢复使用的是 replace into 语句，这样多出的元组不会被清空）

五、课程设计心得体会

设计系统的过程中总会有很多想得不够周全的地方，比如一些冗余问题，还有一些关系模式到底是用实体还是用联系来表示的问题，总是令人很纠结。在测试的过程中，也总发现这样那样的问题，有时候这一个功能的实现正常了，另一个功能却又出现另外一些问题，总是令人很揪心。通过不断改善，最后的结果虽然不尽完美，但终于是让自己觉得可以接受，也是一件令人欣慰的事情。

虽然说我们设计的这个图书管理系统并不是很高级，而且实用性可能也没有一些现有的软件强，界面没有非常美观，不过我们小组可以这样说，我们的实验作业几乎涵盖了本学期所学的数据库知识。从初级，中级到高级数据库知识，我们列出了这学期学过的数据库知识，并以这个列表一个个实现这个图书管理系统的功能，从无到有，从低级到高级，不断解决各种出现的问题，最后做成了这个

可以在多平台上使用的简易管理软件。

看到我们小组的软件能够实现这么多功能，我们感觉这学期的数据库没白学！

六、课程设计过程中所用到的数据库知识：

1、 利用初级 SQL 创建数据库和数据库中的表。

2、 使用了 UTF8 字符集创建数据库和数据库中的表。保证了可以在多个平台上正确的显示汉字，完美解决了在多平台上可能出现的汉字乱码问题。

3、 使用了 select、insert、delete、update、replace 和 alter 这些基本的 sql 语句。

4、 在创建表的过程中用到了 NOT NULL、ENUM、AUTO_INCREMENT、PRIMARY KEY 等完整性约束。

5、 用户管理的过程中用到了用户管理的知识(create user、rename、set password、grant)，对不同的用户授予不同的权利，体现了数据库管理的安全性。

6、 在插入数据的过程中用到了存储过程(insert_into_refund)来限制不合法的插入。

7、 在图书查询的使用了模糊查询，该功能的实现用到了字符串的匹配的数据库知识。

8、 在所有的字符串输入后都进行了字符串的预处理，防止了 sql 注入，字符串输入同时也支持'\''和'\\'这两个字符。

9、 在插入数据和更新数据的过程中用到了触发器(update_after_refund、insert_after_vend、insert_after_stock)来限制不合法的更改。

10、 我们这里的操作都是事务级别，保证了多个用户同时使用软件时正确的并行操作，并使用了"serializable"最高的隔离级别。一旦出错，会使用 rollback 语句回滚事务。如果没有出错则会使用 commit 语句提交事务。

11、 在实现数据库备份和恢复的过程中用到了数据库备份和恢复的知识。

12、 使用 mysqldump 备份数据，这里使用了 mysqldump –u root –p**** -R booksalesmanagement > data.sql 该语句连同存储过程和数据库备份到 data.sql 文件中，用户可以通过恢复数据库的方式快速地建立数据库。