实验报告

1. 实验题目

图书销售管理系统的设计与实现。

1. 开发环境

操作系统：Windows 10

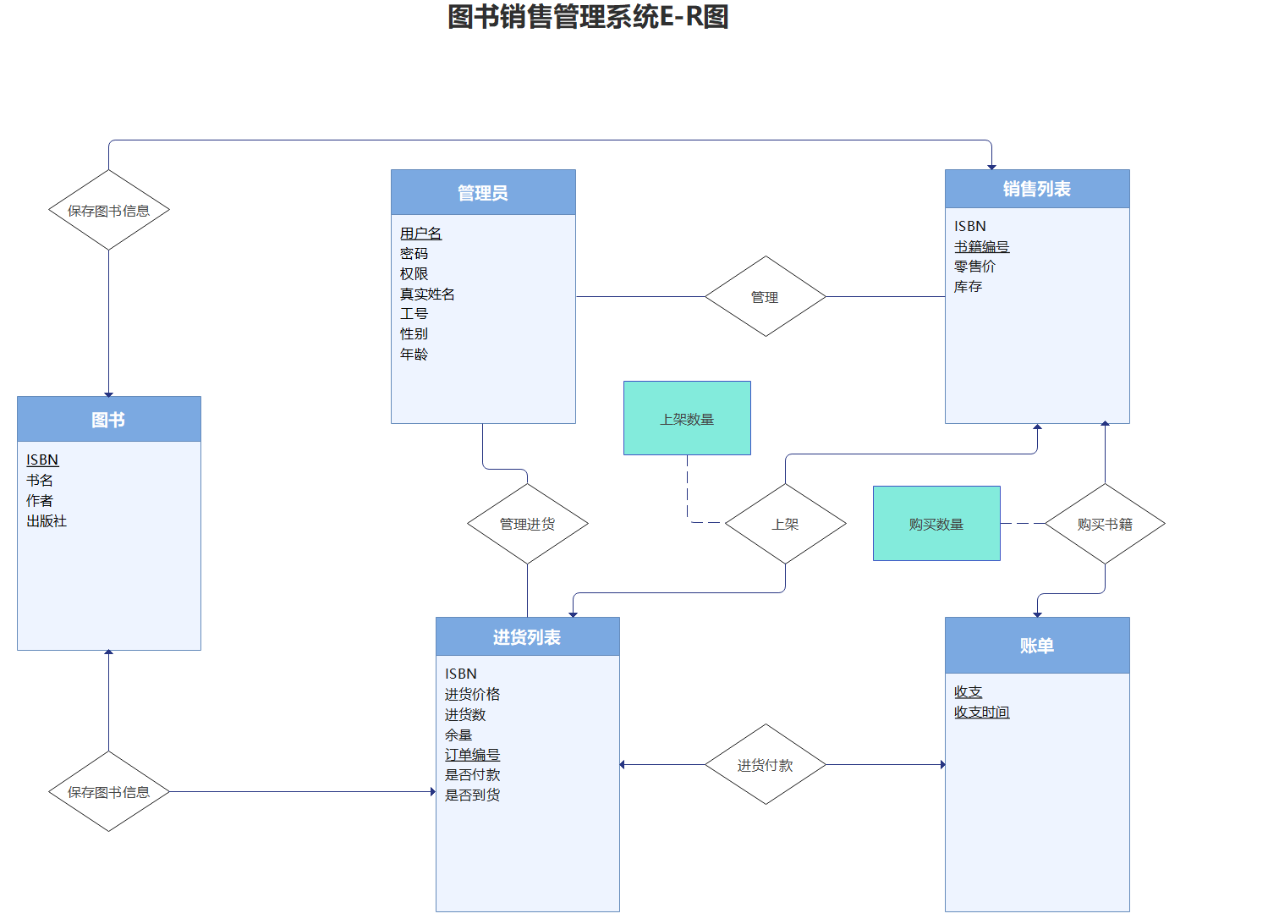
数据库管理软件：pgAdmin4

编程语言：C/C++

客户端开发环境：MFC

1. 数据库设计

本数据库的E-R图如下：



（一）实体表

先说明4个自定义的数据类型：

gen：enum(‘男’，’女’) 表明性别

perm：enum(‘超级管理员’，’普通管理员’) 表明权限

stat：enum(‘未付款’，’已付款’，’已退货’) 表明付款/退货情况

isarrived：enum(‘未到货’，’已到货’) 表明是否到货

共有5张表：

①管理员表 administrator：用于存放用户(管理员)的信息。

属性：

用户名(username) varchar(20) primary key：用户的用户名，是管理员表的主码，唯一标识一位用户。

密码(password) nchar(32) not null：用户的密码，以md5形式加密为32位形式保存在数据库中。

权限(permission) perm default ‘普通管理员’：用户的权限，有2类：超级管理员和普通管理员。普通管理员可以查看和修改自己的信息，修改书籍信息、管理图书进货及付款退货和图书上架；超级管理员在此基础上，还可以查看和修改所有用户的信息，并能创建新用户。

真实姓名(realname) varchar(20)：用户的真实姓名。

工号(jobnumber) int unique：用户的工号。每个工号唯一对应一位用户。

性别(gender) gen：用户的性别。

年龄(age) int check(age>0)：用户的年龄。

②书籍表 book：存放、管理书籍相关信息的表。

属性：

ISBN nchar(13)：13位字符串，相当于书籍的“身份证”。

书名(title) varchar(50) not null：书籍的标题。

作者(author) varchar(30)：书籍的作者名。

出版社(press) varchar(30)：书籍的出版社。

③销售列表 sale：存放上架书籍相关信息。

属性：

ISBN nchar(13)：13位字符串，相当于书籍的“身份证”。这里，ISBN作为外码参照book的主码。

书籍编号(bookID) serial primary key：书籍的编号，是书籍表的主键，唯一标识一本书。书籍编号类型为serial，即递增类型，表明其是自动分配的，不需要用户指定。

零售价(price) decimal check(price>0)：向顾客出售时，书籍的价格。

库存(inventory) int check(inventory>=0)：书籍的剩余量。

④进货列表 restocklist：用于保存进货的订单。

属性：

ISBN nchar(13)：13位字符串，相当于书籍的“身份证”。这里，ISBN作为外码参照book的主码。

进货价(restockprice) decimal check(restockprice>0)：在进货时，书籍的购买价。

进货数(restocknum) int check(restocknum>0)：进货时书籍的购买数。

订单编号(orderID) serial primary key：restocklist的主键，唯一标识一条订单。类型为serial，不需要用户指定。

付款状态(status) stat default’未付款’：标识付款状态，有3类：未付款、已付款、已退货。

是否到货(arrival) isarrived default ‘未到货’：标识是否到货，到货了才能将仓库的书籍上架。有2类：未到货、已到货。

⑤账单表 account：用来记录每条收支记录。

属性：

收支(diff) decimal check(diff<>0)：记录书籍出售时的收入和进货付款时的支出。

收支时间(difftime) timestamp default current\_timestamp：记录收支的时间戳。

account表的主键为(diff,difftime)，两者合起来唯一标识一条收支。

（二）联系

①管理：标识用户管理书籍信息的联系，为多对多的联系。由于每本书籍都可以被所有管理员管理，每个管理员都能管理所有书籍，所以实际上并不需要建立对应的联系表。

②管理进货：标识用户管理书籍进货的联系，为多对多的联系。实际上不需要建立对应联系表，理由同①。

③上架：标识将进货列表的订单上架到书籍列表的联系，为一对一的联系：一条订单唯一对应到一条书籍记录，一条书籍记录也唯一对应到一条订单。同时还有联系集属性上架数，标识要将本条订单对应的书籍上架的数量。

④购买：标识顾客完成一次书籍购买后在账单表中生产一条收入的联系，为一对一的联系：一条收支唯一对应到一条书籍记录，一条书籍记录也唯一对应到一条收支。同时还有联系集属性购买数，标识购买的数量。

⑤进货付款：标识管理员完成一次书籍进货后在账单表中生产一条支出的联系，为一对一的联系：一条收支唯一对应到一条进货记录，一条进货记录也唯一对应到一条收支。

⑥保存书籍信息：销售列表和进货列表都可以通过ISBN获取对应的书名、作者、出版社，为一对一的联系：每个ISBN对应于唯一的书籍信息。

1. 系统设计

**1、用户管理**

**1）系统用户分为超级管理员用户和普通管理员用户。普通管理员用**

**户只能对图书进货、销售等信息进行管理，只能查询和修改自己的**

**用户信息；而超级管理员除了可以对图书进货、销售等信息进行管**

**理，还能创建新的用户和查看所有用户的资料。**

**2）超级管理员用户在系统完成时便已经存在（即其用户名和密码已**

**经存在于数据库中）。而普通管理员用户的用户名和密码需要由超**

**级管理员用户来创建。**

**3）用户的密码不能以明文形式保存于数据库中，而必须先加密，一**

**般采用MD5算法进行加密。**

**4）每位用户除了用户名和密码信息外，还有真实姓名，工号，性别，**

**年龄等基本信息。**

**5）系统所有功能只有用户登录了才能进行操作。**

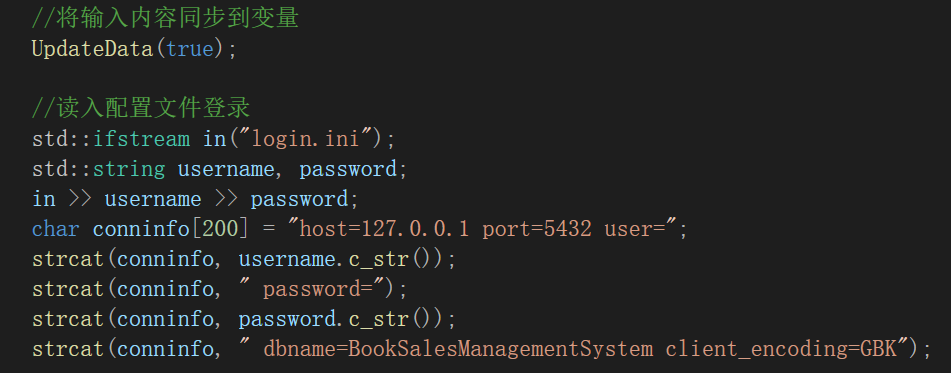
启动程序时首先进入登录界面，要求输入用户名和密码。

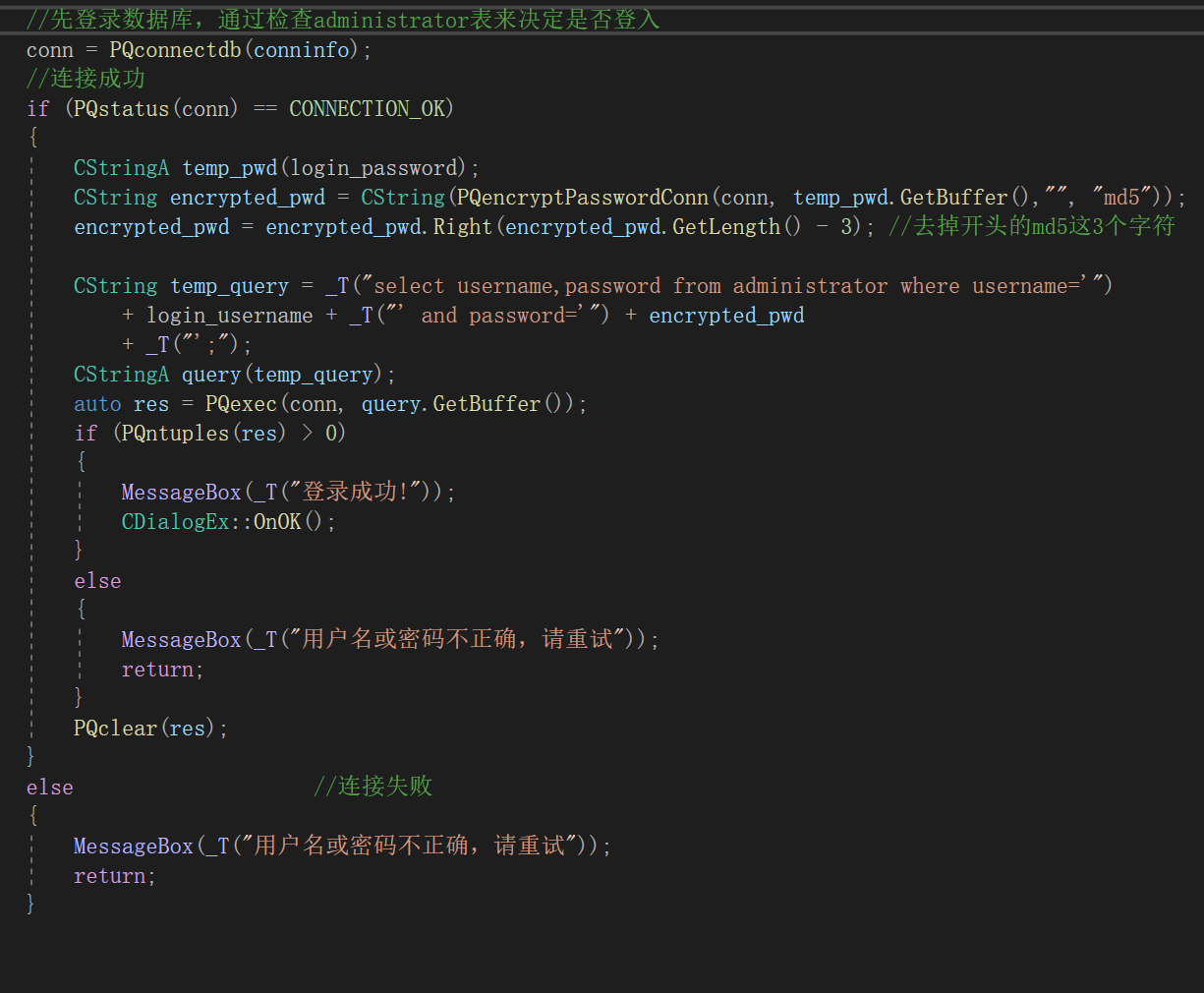


若用户名或密码输入错误则弹出提示，输入正确无误才能进入销售系统界面。

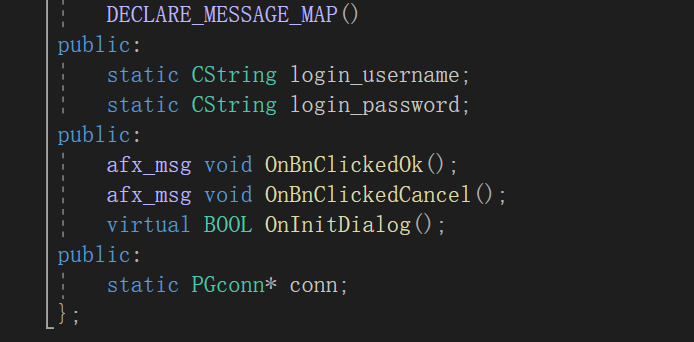


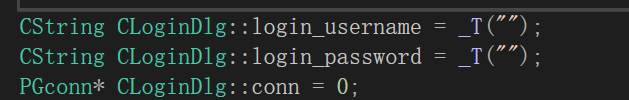
实现：通过先设定好的登录信息先接入数据库，再到数据库的administrator表中寻找对应登录信息的元组，如果存在这样的元组则登录成功，否则登陆失败。



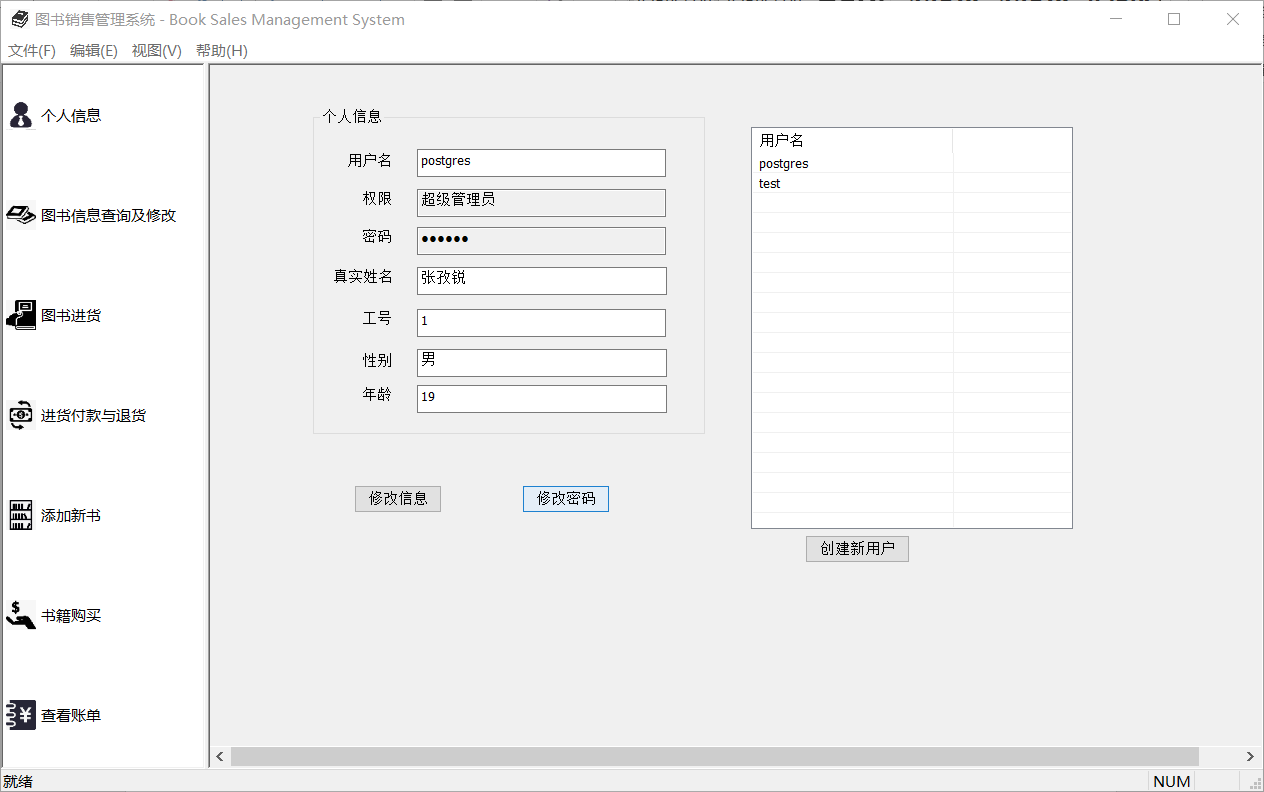


为了能在后续继续使用数据库端口和登录使用的用户名和密码，考虑将数据库指针conn、登录用的用户名和密码设置为public static变量，并在类外部初始化。



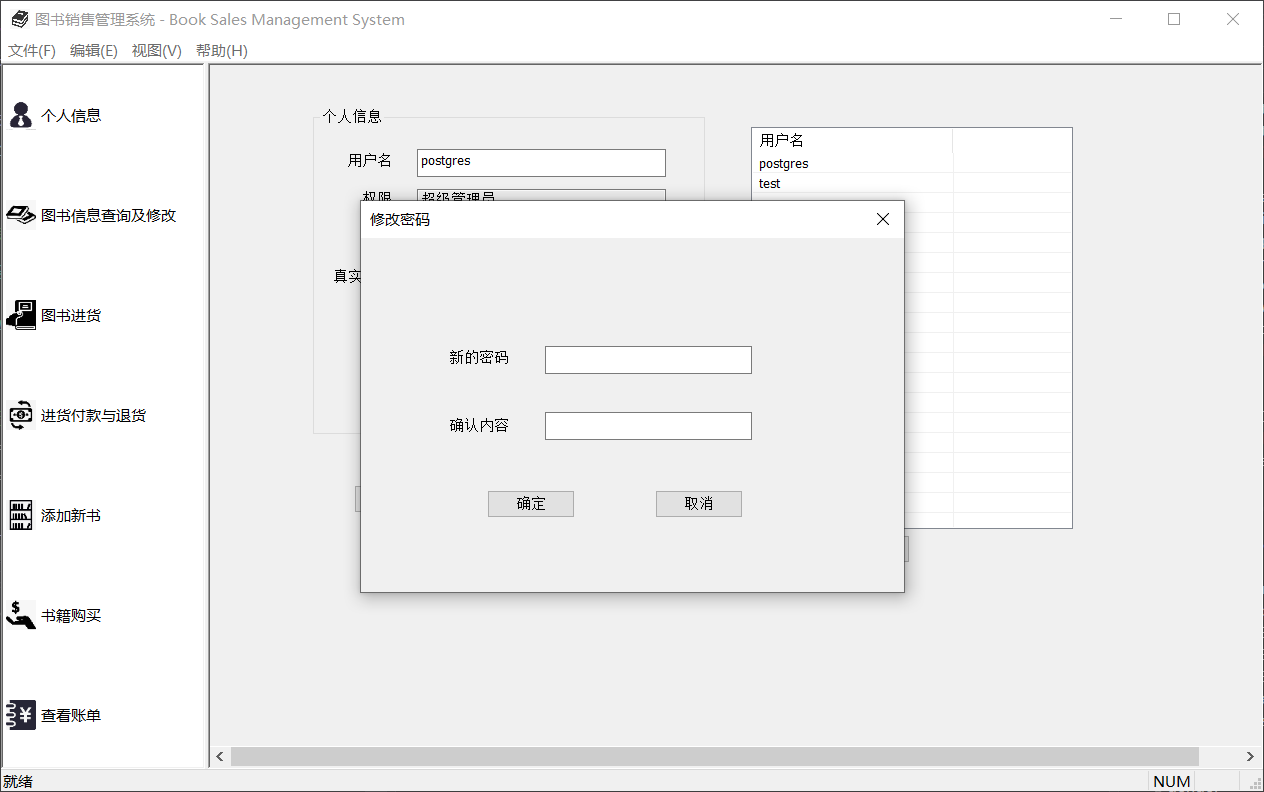


进入界面后，可以查看用户的信息，超级管理员可以在右侧列表中通过左键选取相应用户，可以在下方选择创建新用户，可以修改相应用户的信息和密码。



查看信息：点击右边列表的用户名后，获取该用户名，并用select语句在administrator表中查找该用户名的信息，并把内容设置到各属性的编辑框上。这里需要注意的是，由于密码在数据库中以md5形式存储，所以无法获取其他人的实际密码，所以我设计成了查看他人密码时，他人密码显示为空的形式。

修改信息：获取对应编辑框的内容，用update语句修改对应用户的信息。注意只能修改自己的信息（包括超级管理员）。

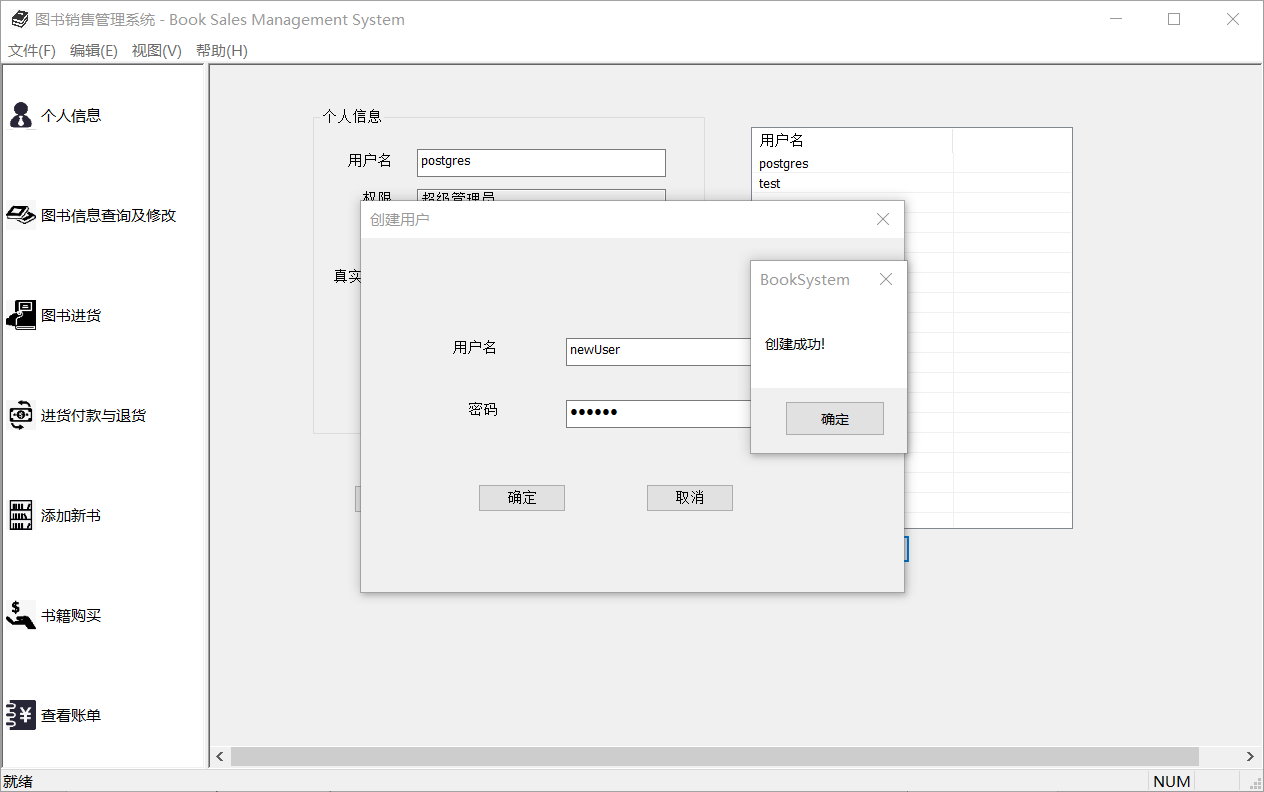


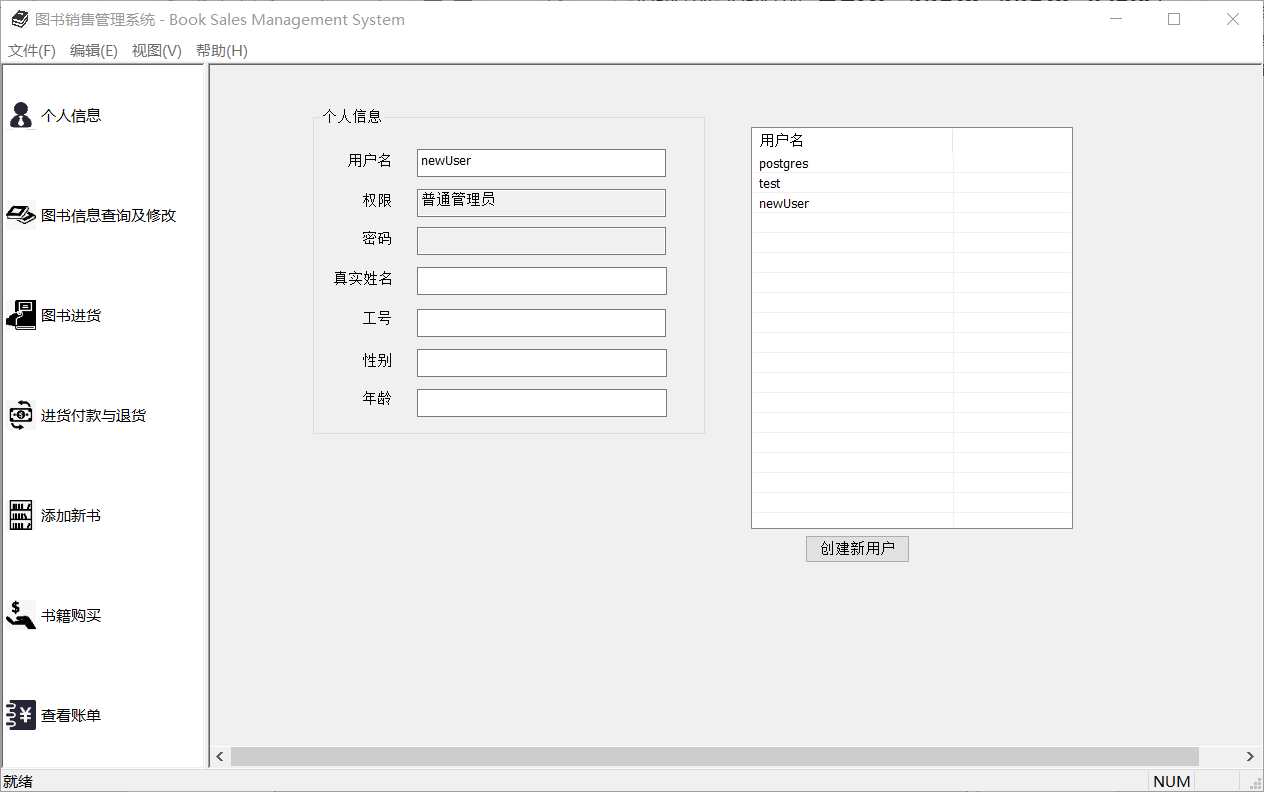
修改密码：点击后弹出修改密码对话框，需要2次输入相同密码修改。没有将密码同其他信息一起修改，主要是考虑到密码修改的重要性。获取用户输入内容后先考察密码长度、两次密码是否相同，再用update语句将新密码更新到对应用户信息。密码的md5加密使用libpq库函数PqencryptPasswordConn。由于我发现这个函数会把用户名也放进去一起加密，从而导致修改用户名后登录会出问题，所以使用这个函数时用户名参数一律传入空字符串。



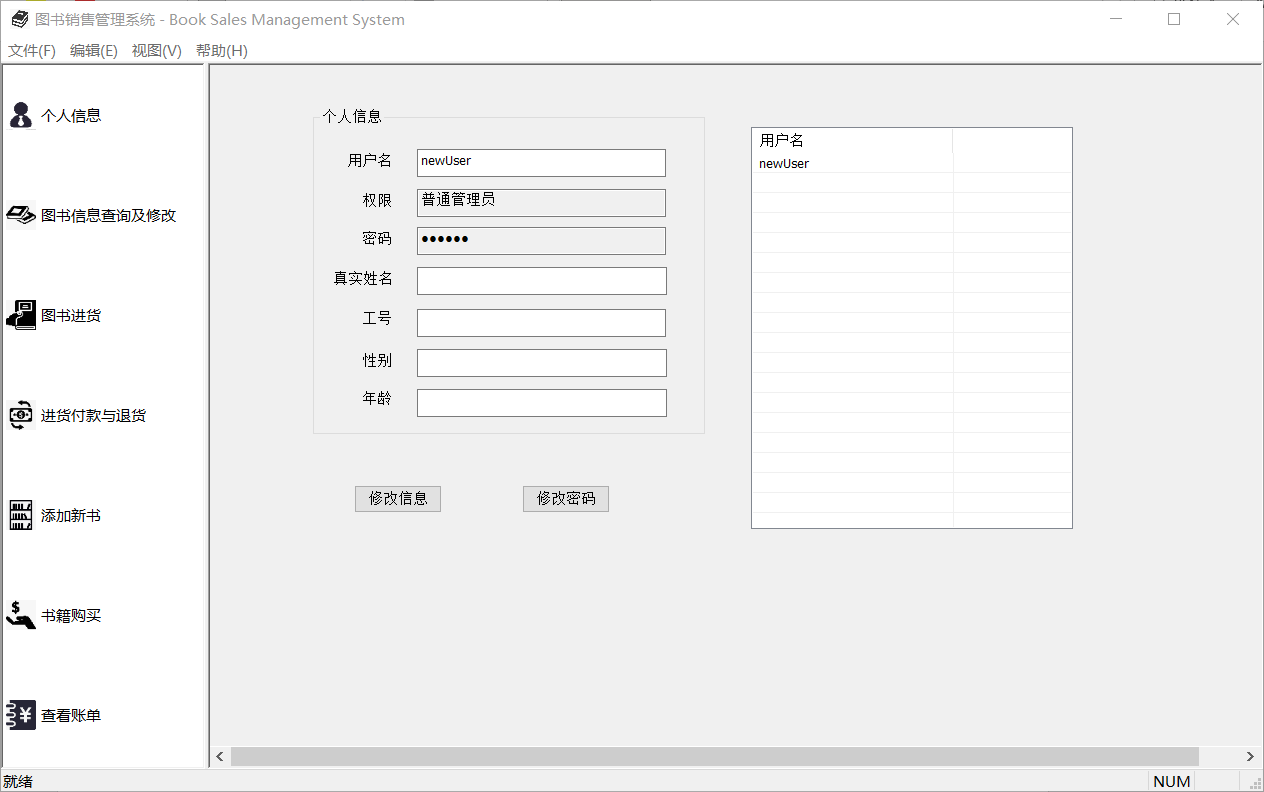
创建新用户：点击后弹出创建用户对话框，需要设定用户名和密码，然后基于输入的内容用insert语句在administrator表中插入一条记录，权限默认为普通管理员，其他信息为空，可后续修改。

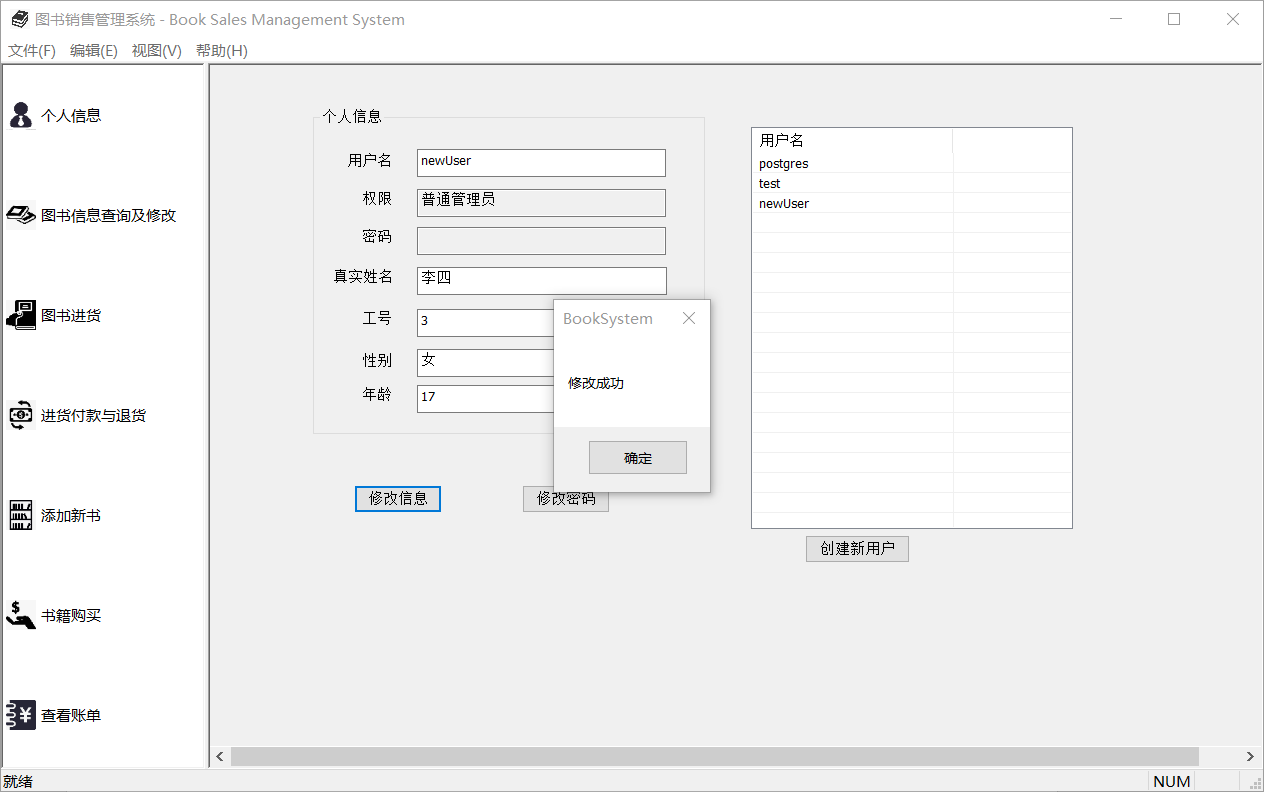
示例：创建一个新用户并修改信息，再查看信息。



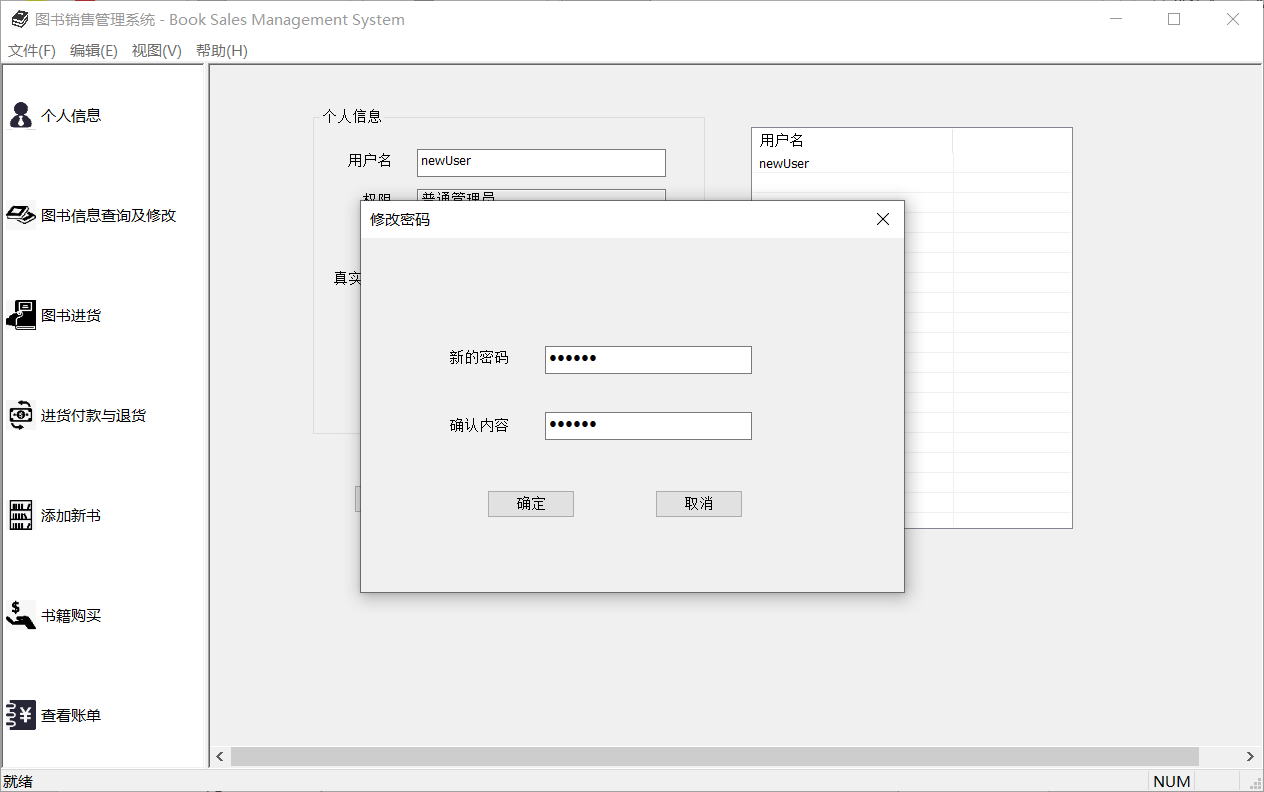


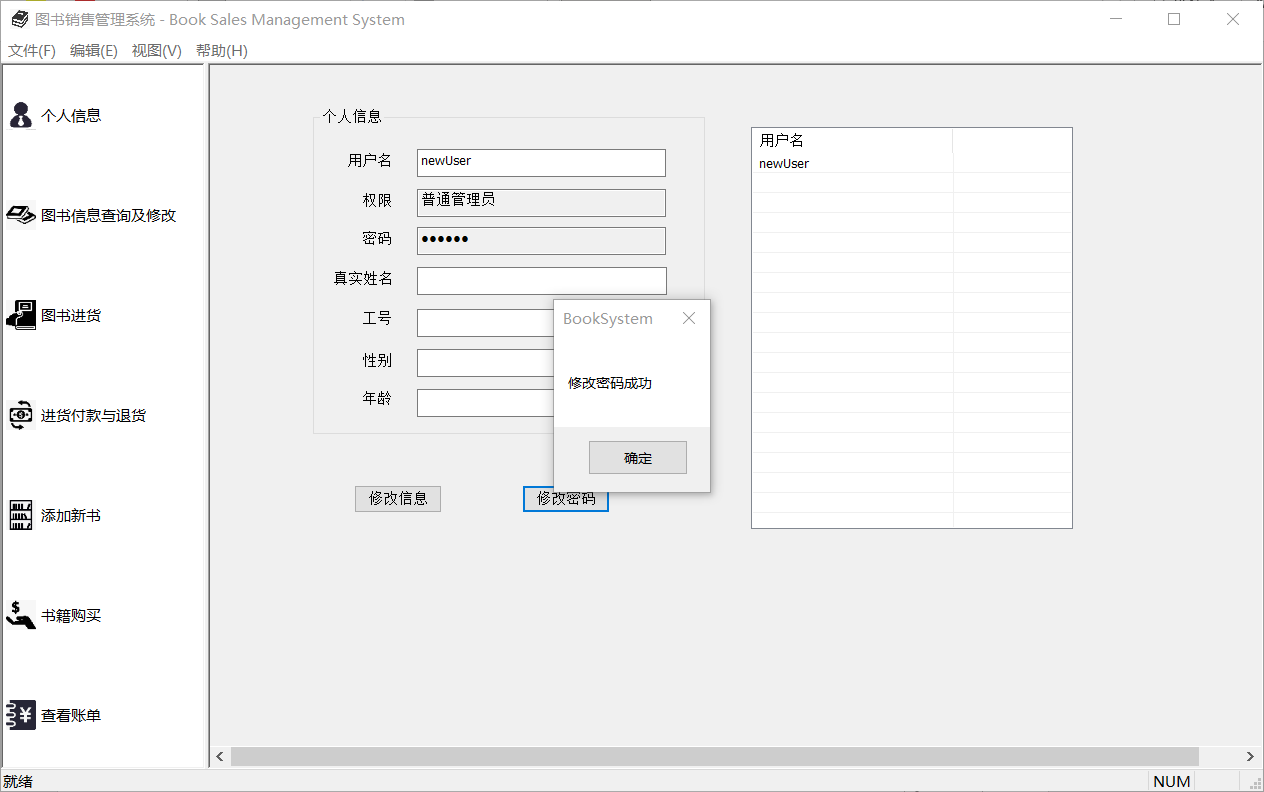
登录新用户修改信息和密码：



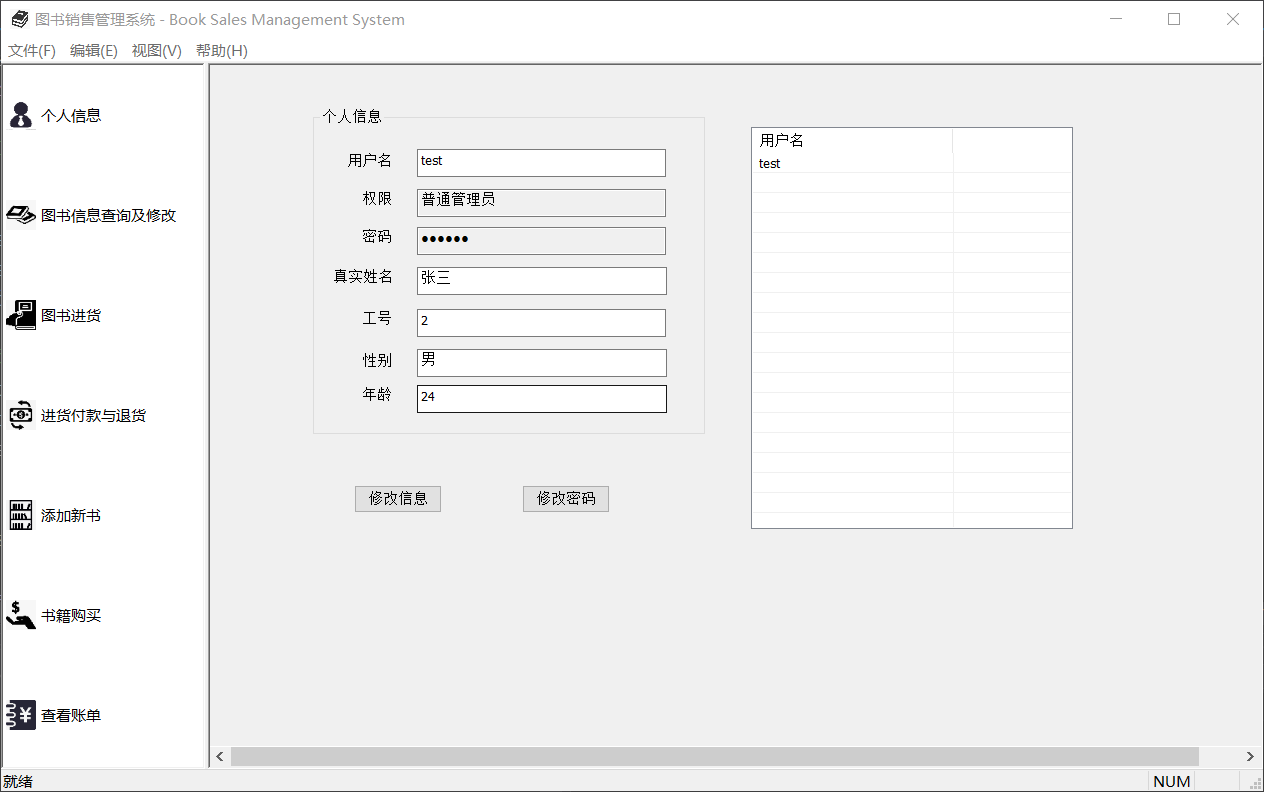


修改密码演示：

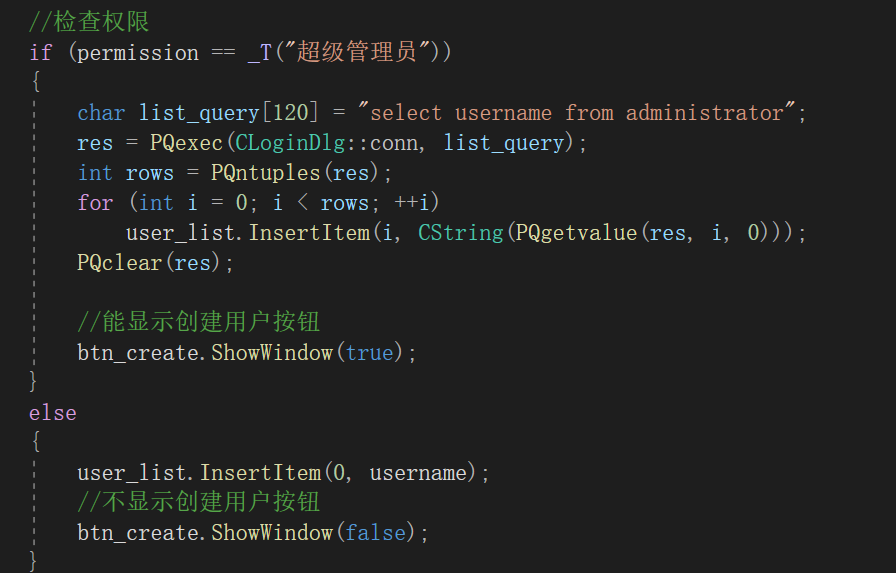




这里区分超级管理员和普通管理员的方法是在显示上做区分：超级管理员显示所有内容，但普通管理员的界面没有创建新用户按钮，右侧的用户列表也只有自己。如下为普通管理员登录后的个人信息界面。



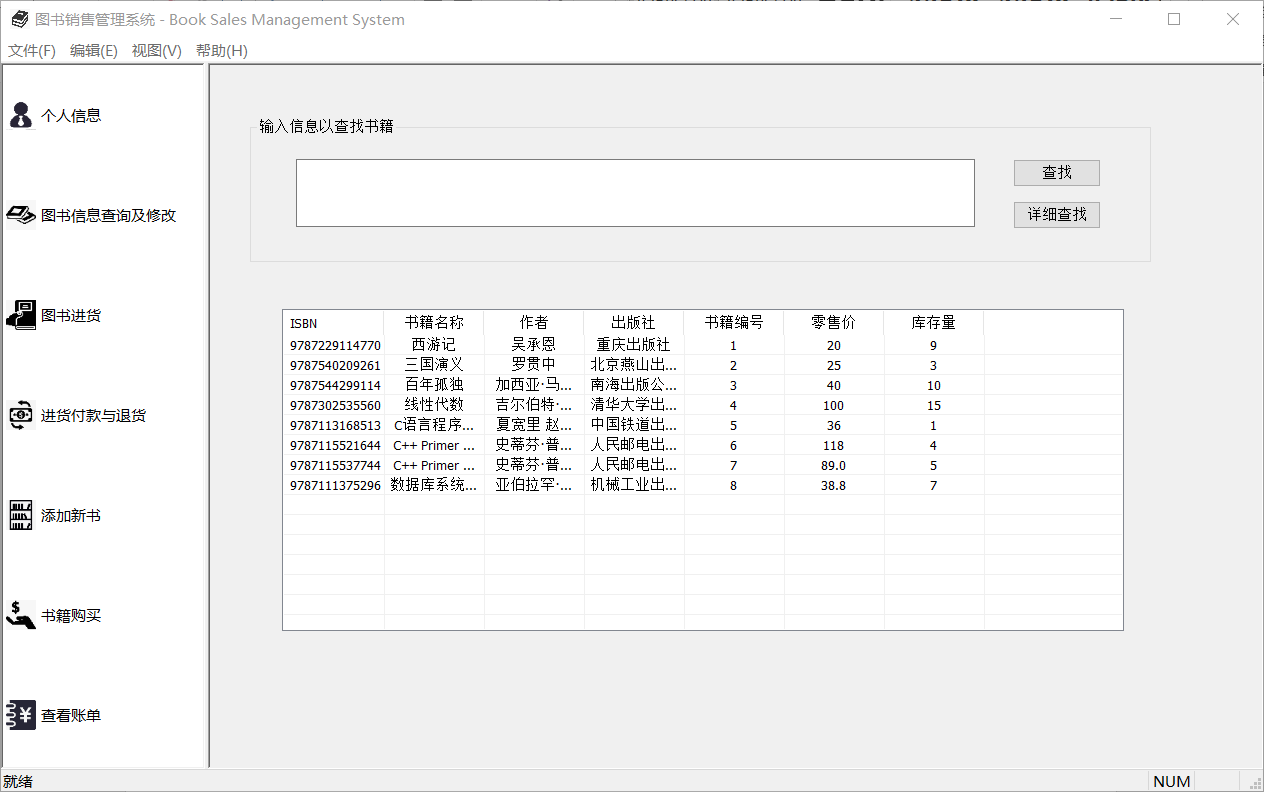
实现方法为：通过检查登录获得的用户名，用select语句在数据库中获取相应的信息（包括权限），检查权限来确定哪些内容需要显示。



**2. 库存书籍管理**

**系统中需要维护整个书城目前库存的所有书籍信息，包括书籍ISBN号，书籍名称，出版社，作者，零售价格，当前库存数量等。**

点击左侧列表的“图书信息查询及修改”选项，可以查看目前数据库中所有书籍的相关信息。通过select语句查询book ⋈ sale，将查询的内容插入到列表上。

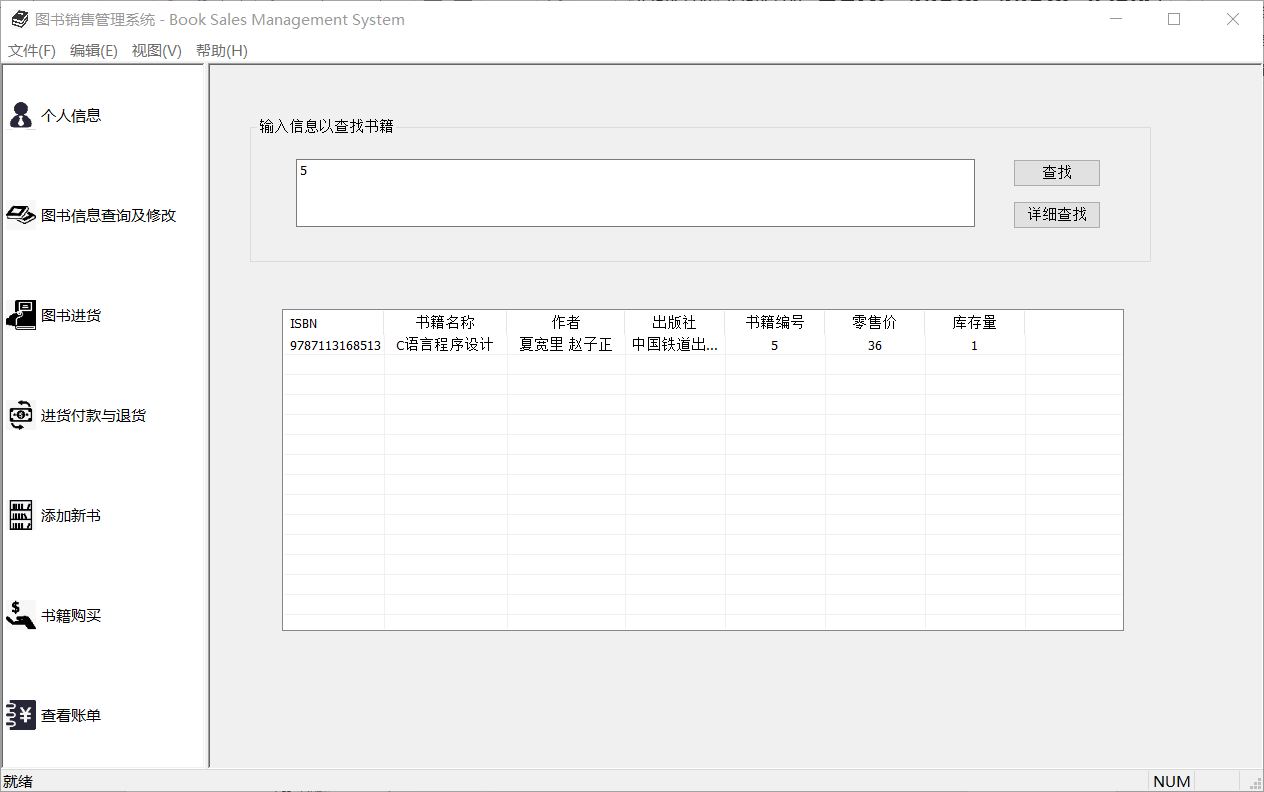


**3. 书籍查询**

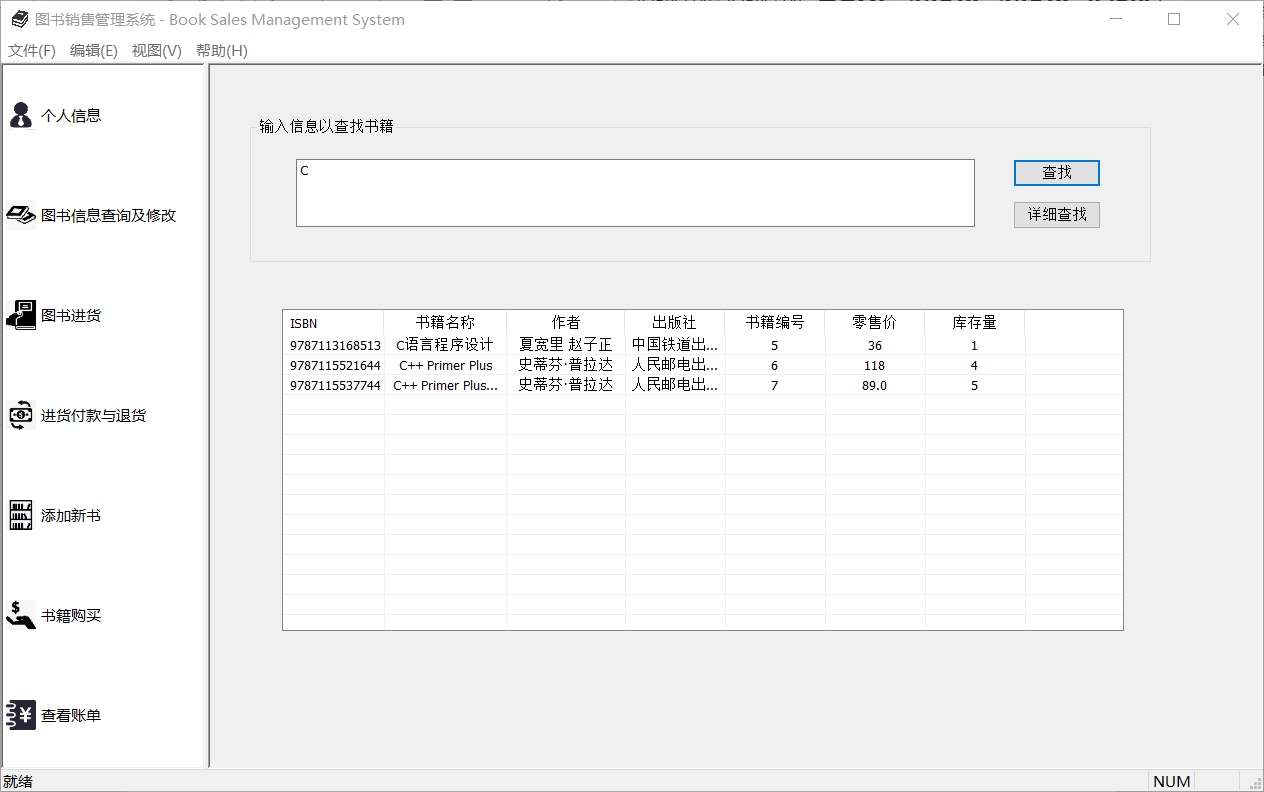
**可以使用书籍编号、书籍ISBN号、书名、作者、出版社等方式查询库存的相关书籍。**

在上方的编辑框中输入信息，支持的查找方式：输入ISBN、书籍编号（必须全部匹配），输入书名、作者、出版社的相关信息（可输入部分）。查找后会显示符合的书籍信息，并提示查找到的相关数据条数。

例1：查找书籍编号



例2：输入相关信息匹配

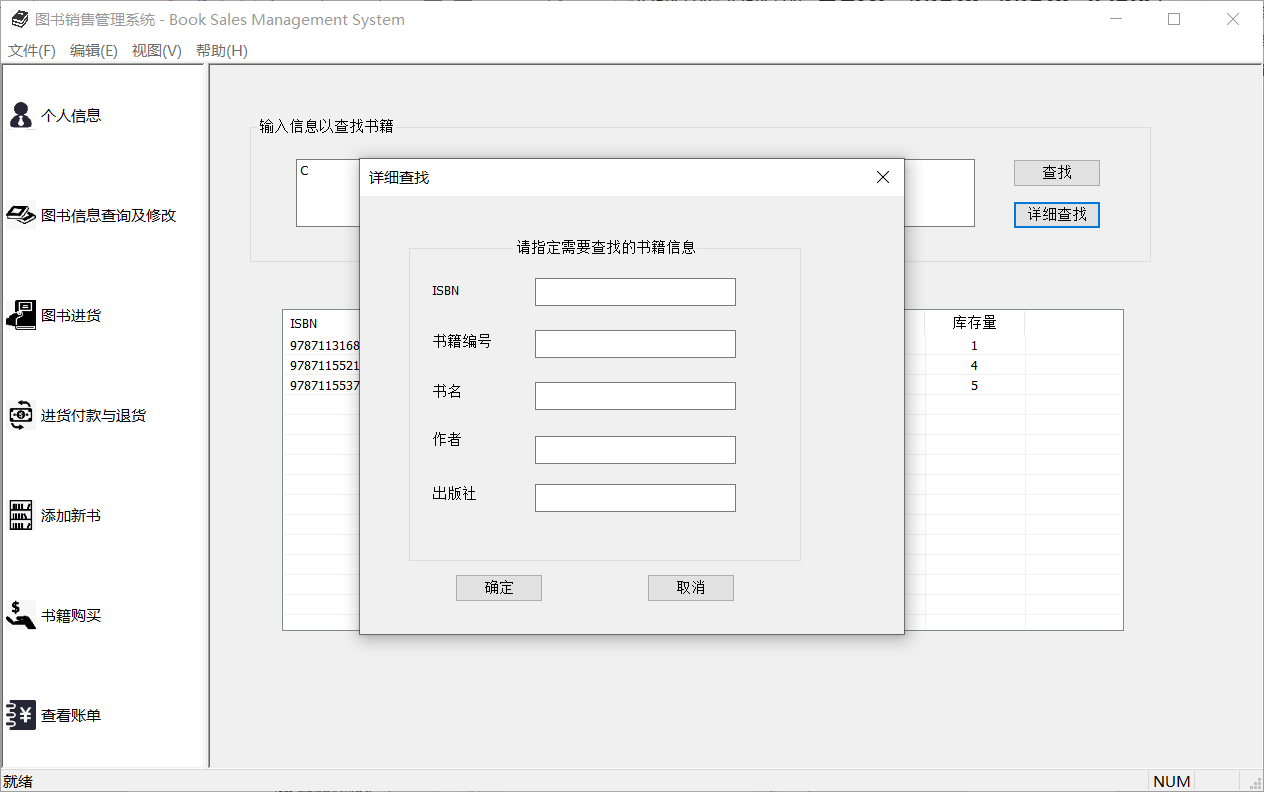


实现：每次点击按钮时先清空列表，用select语句配合like %’ ‘%进行内容匹配。这里需要注意的是，由于书籍编号设定为serial(int)类型，如果与字符串一同匹配则整个query语句都会执行失败，所以需要分开执行，检测到没有匹配到相关数据后再用一次select语句匹配对应的书籍ID。最后将查找到的信息插入到列表更新显示。

（在后面的几个功能实现中也都用到了查找功能，实现原理与这里是一致的，后面就不展开介绍查找功能的实现流程了）

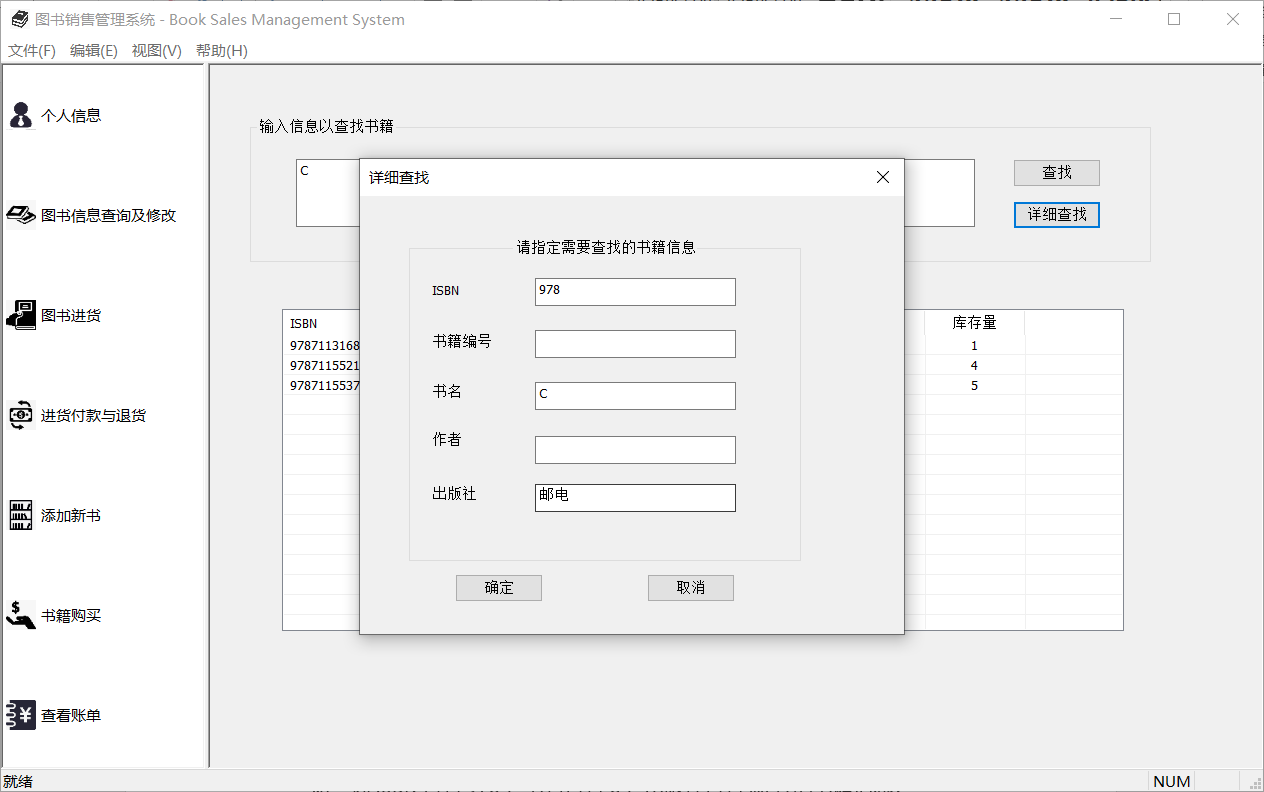
例3：详细信息查找

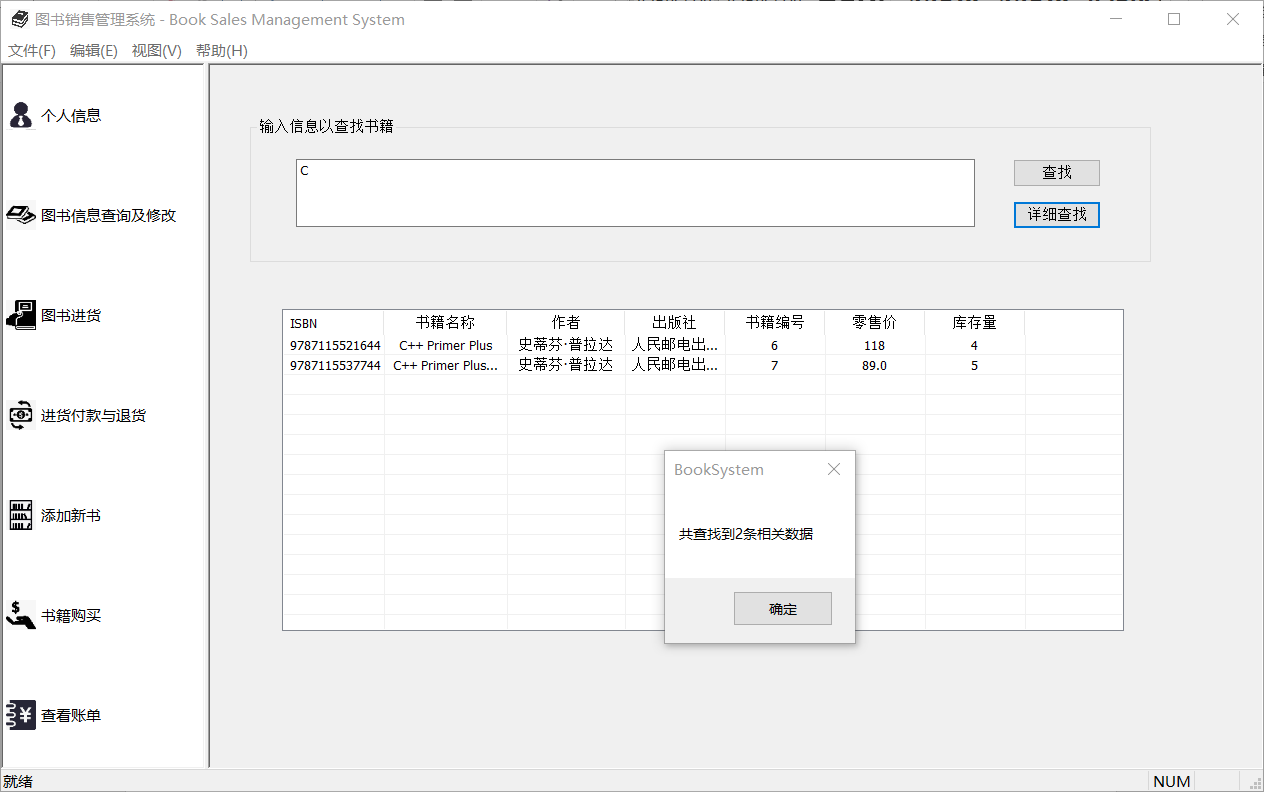
点击“详细查找”按钮，跳转到详细查找界面。



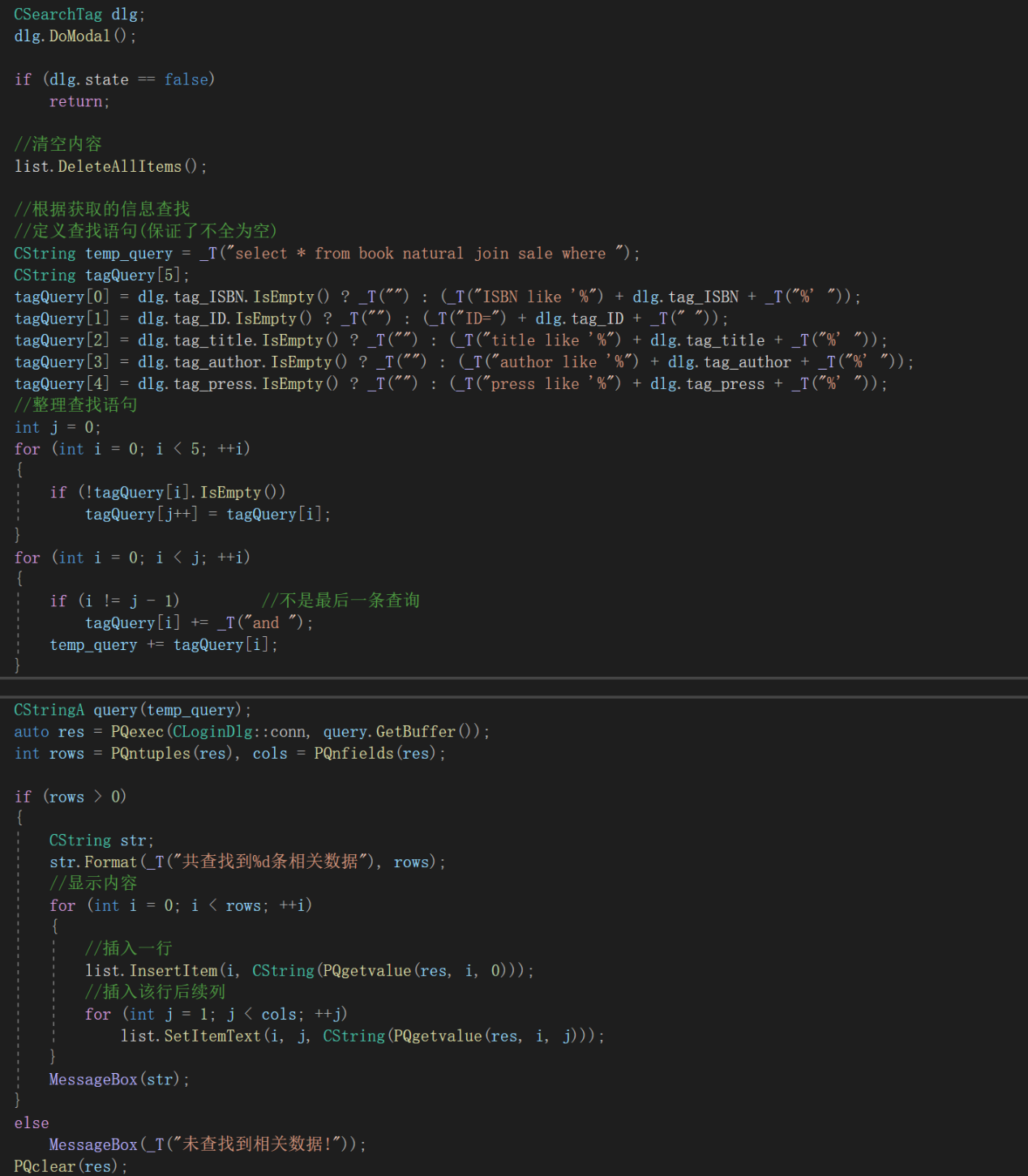
输入相应的信息进行具体的查找，结果显示满足所有要求的书籍信息。

如查找ISBN含有’978’，书名含有’C’，出版社含有’邮电’的书籍信息：





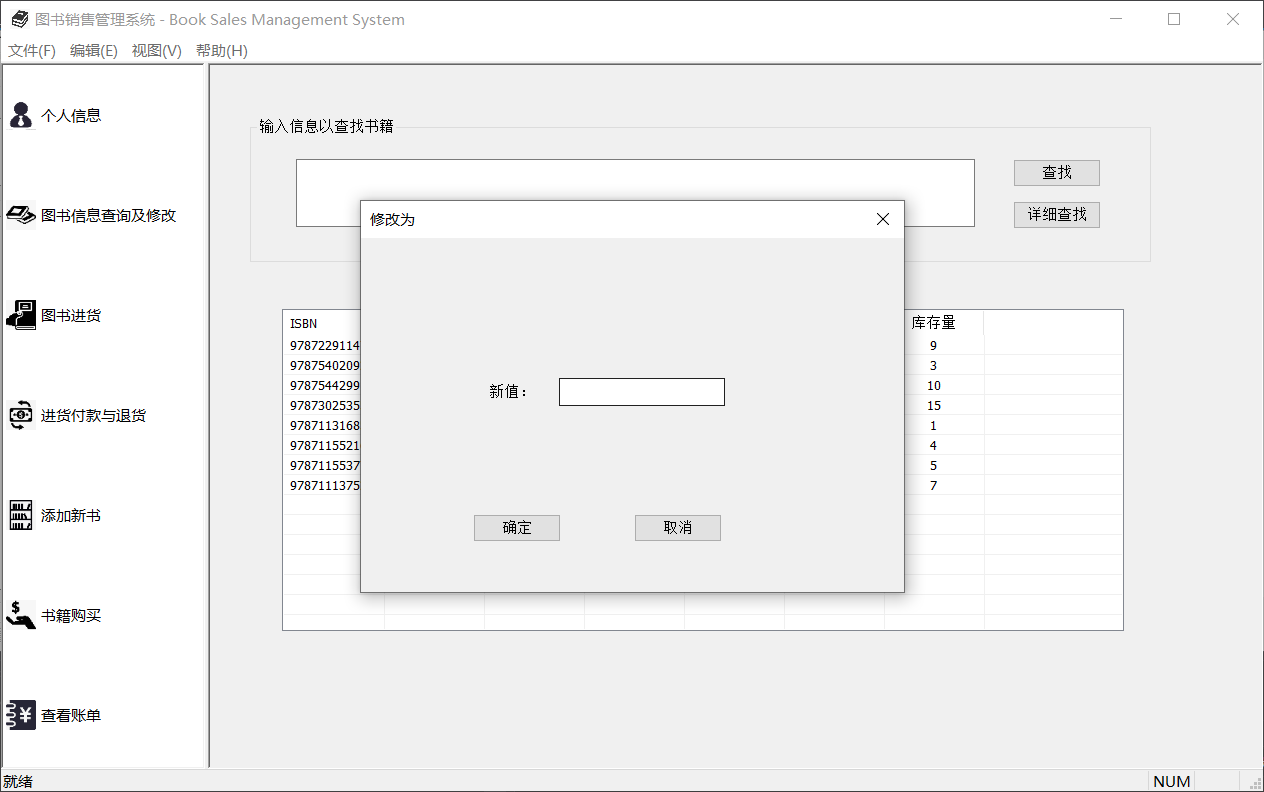
实现：获取详细查找界面的各项输入，根据输入是否为空来定义查询语句，并在所有非空语句中，给除了最后一条语句的末尾都加上and。



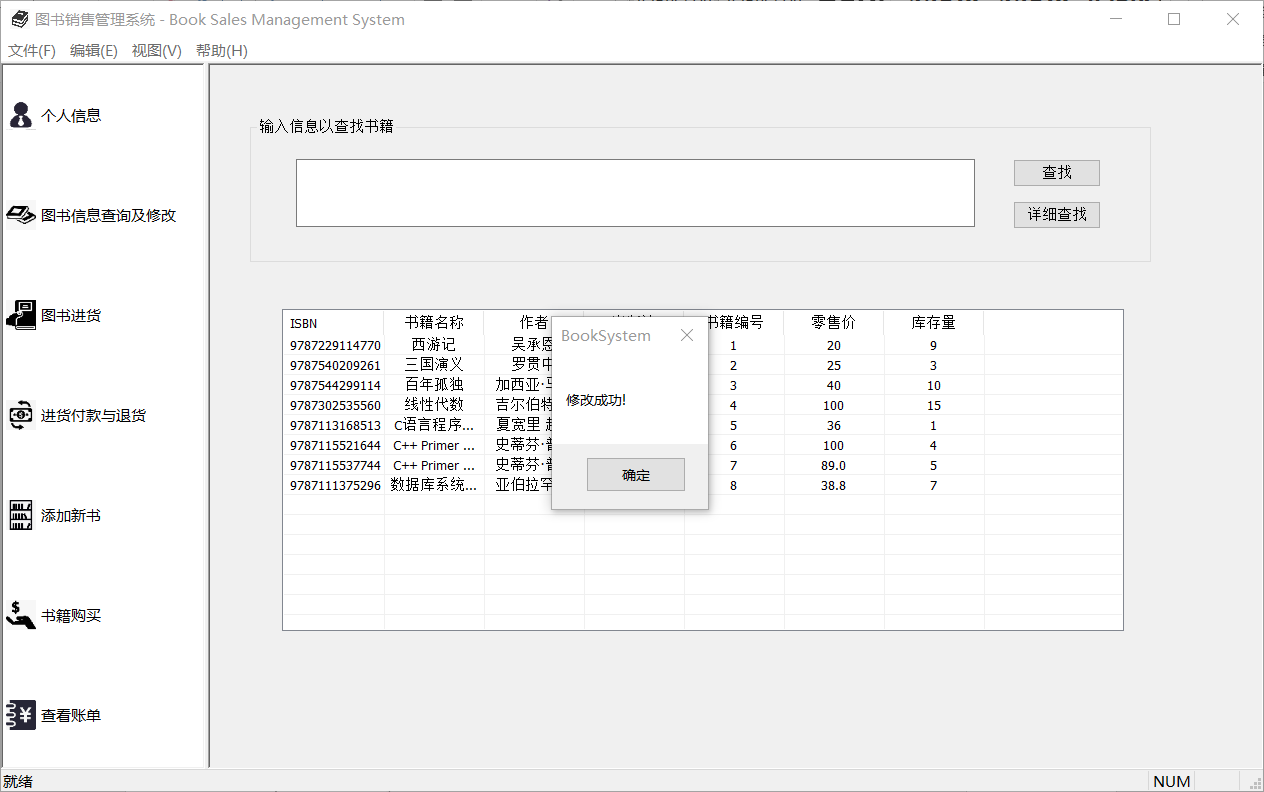
**4. 图书信息修改**

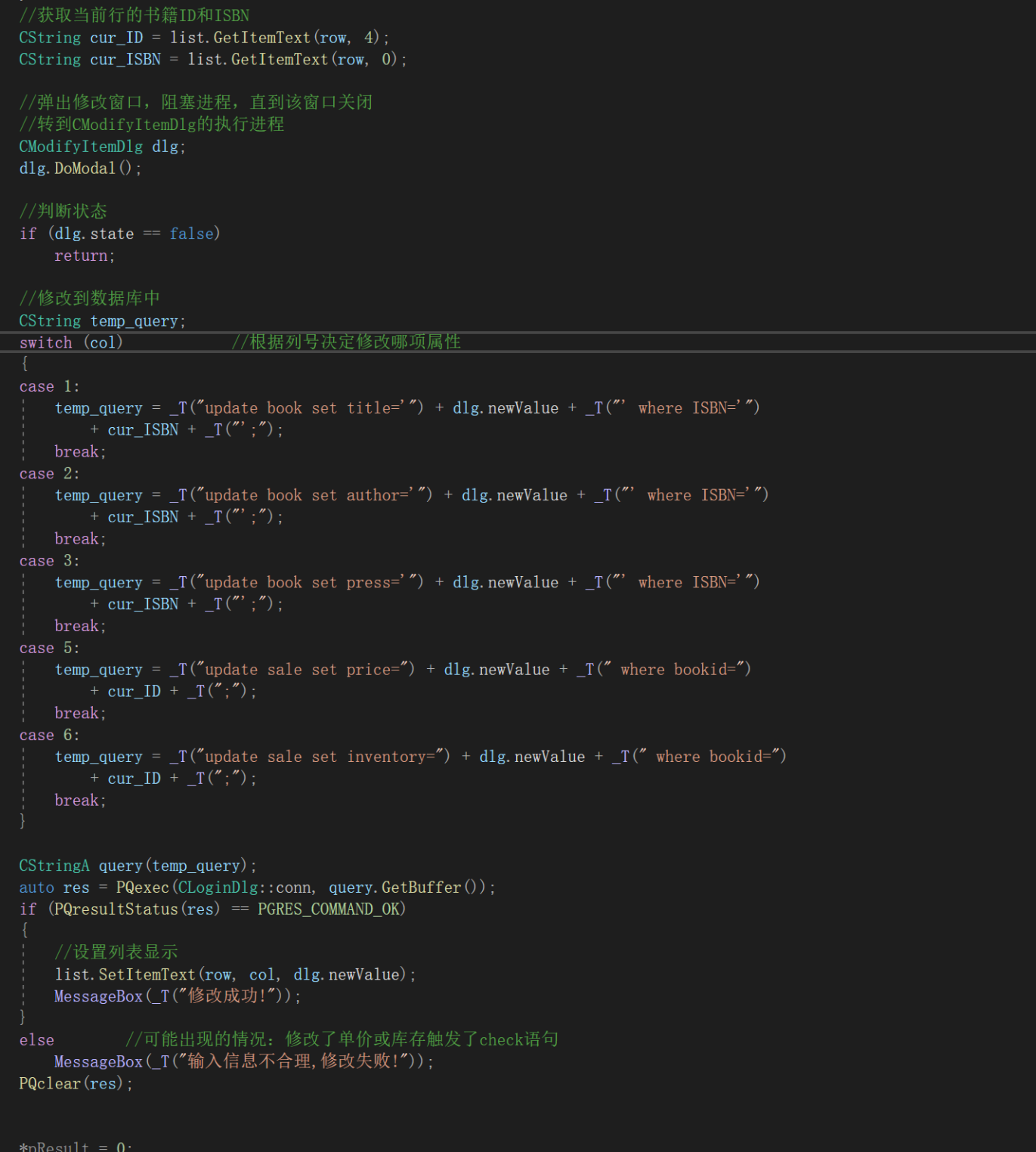
**可以修改书籍名称、作者、出版社、零售价格等信息。**

仍然是在图书信息查询和修改界面，对查找界面上的书籍信息选中后进行双击就能进行直接修改。比如这里点击第6行信息的零售价列，进行修改，会弹出修改新值的窗口：



接收用户输入的信息后，根据双击的位置有相应的MFC函数确定对应的行列，先通过行获取对应的书籍ISBN或ID，再通过列确定要修改的属性，并用update语句修改。修改成功或失败都有相应提示。这里修改零售价为100。





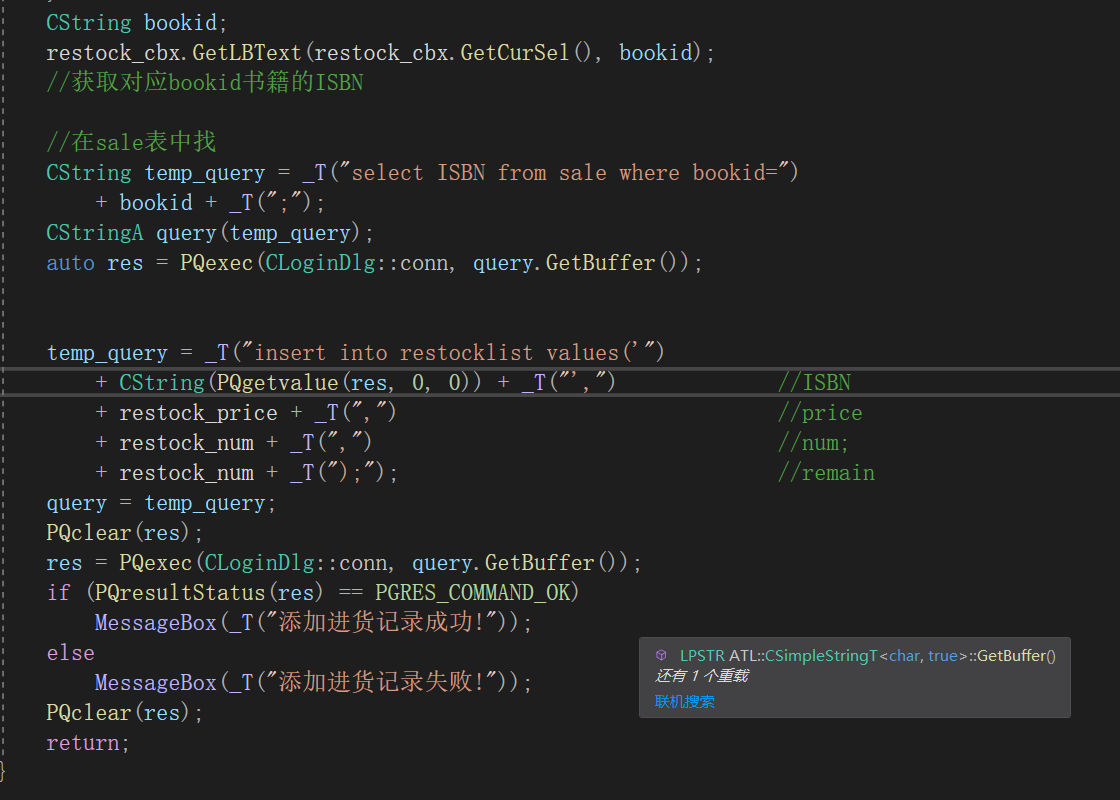
5. 图书进货：

**对于需要进货的书籍，如果库存中曾经有这本书的信息的话，则直接将这本书的ID列入进货清单，否则需要输入进货书籍的相关信息，包括ISBN号，书名，作者，出版社等。此外，每种书都要指定其进货价格和购买数量。对于刚列入进货清单的书籍给予未付款状态。**

点击左侧的“图书进货”选项，进入进货页面。



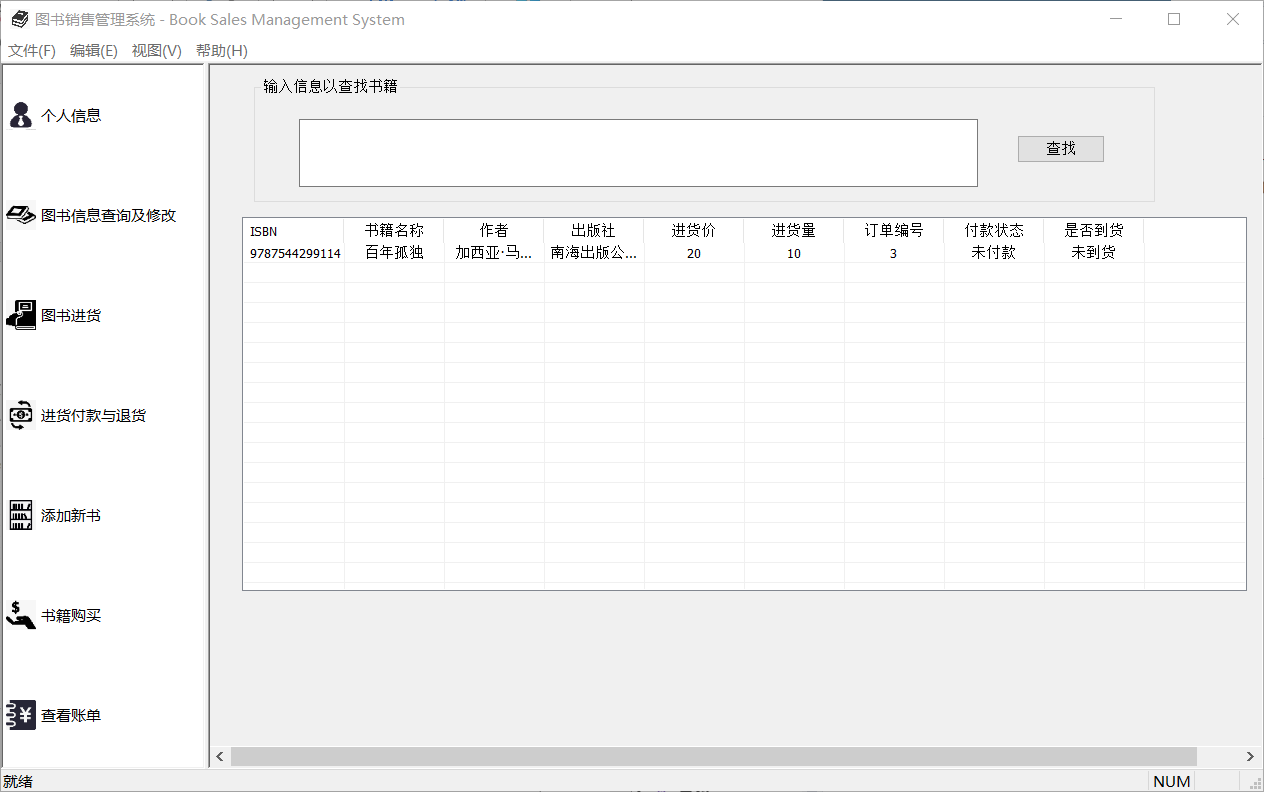
从上方的下拉框可以选择曾经有的书籍进货：在初始化窗口时，通过select语句从sale表获取所有书籍的bookID（曾经有这本书→这本书上架过，现在库存可能是0），并加入到下拉框。按下进货按钮时首先检查是否输入了进货价和进货数量，都输入后获取当前下拉框内容的bookID，用select语句在book⋈sale表中获取该bookID对应的书籍信息，用insert语句向restocklist插入一条相应的进货订单信息。



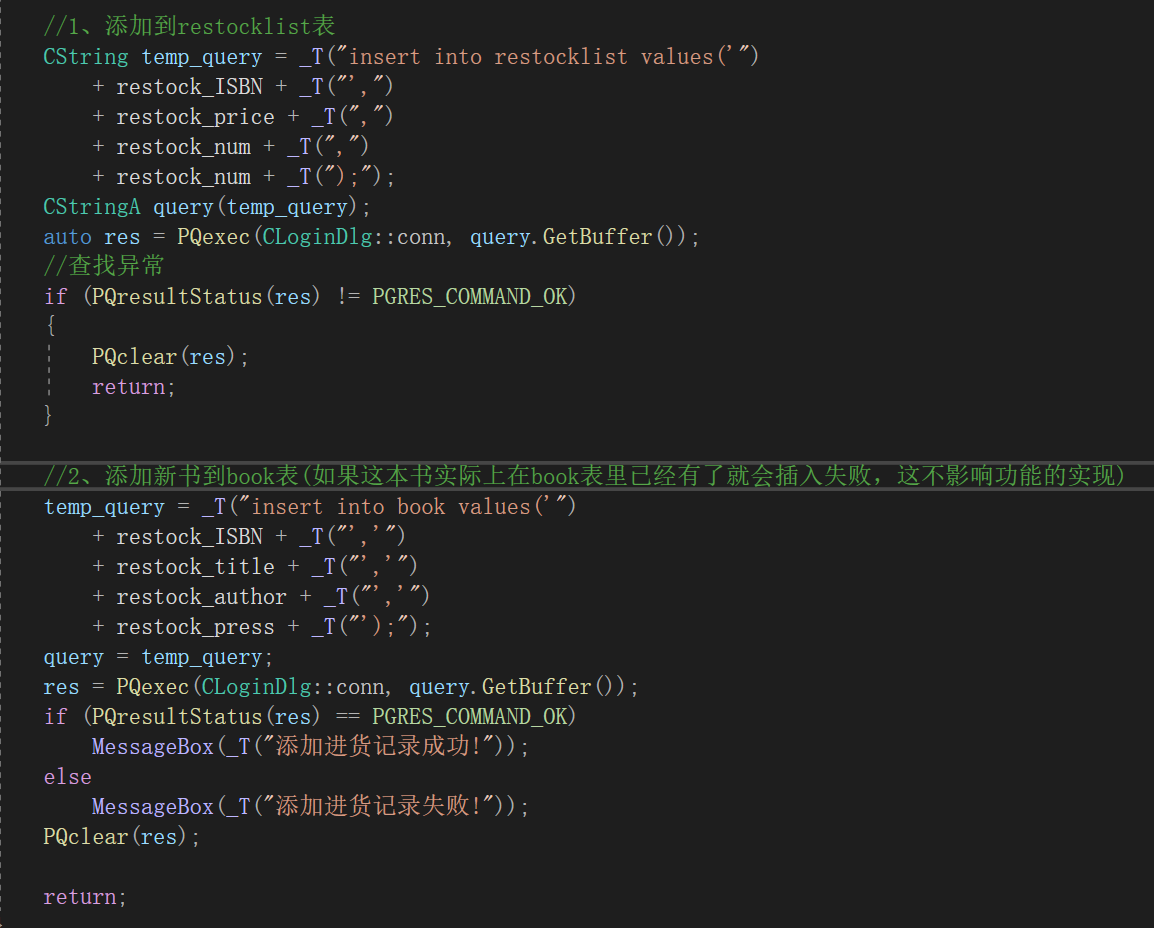
进货曾经有的书示例：比如进货bookID为3的书，进货价20，进货数10



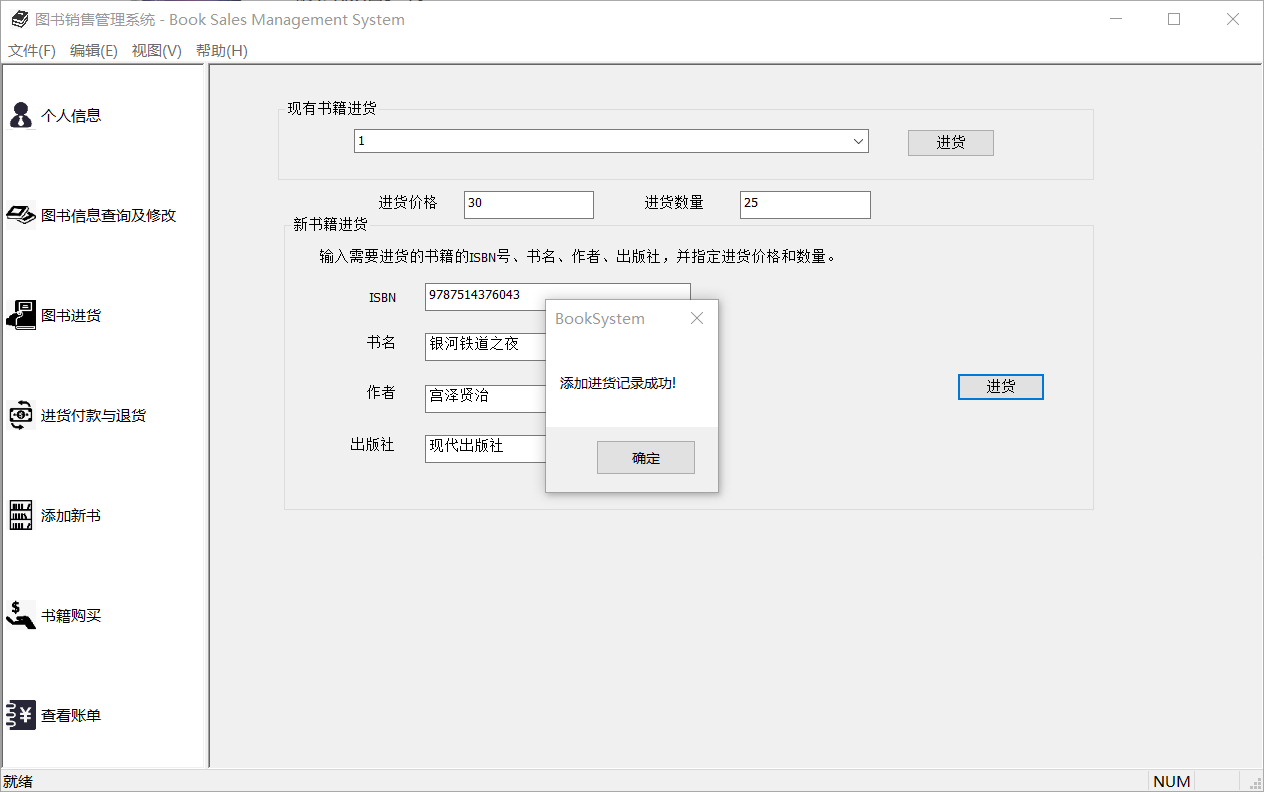
点击左侧“进货付款与退款”项，可以看到已经多了一条对应的订单。



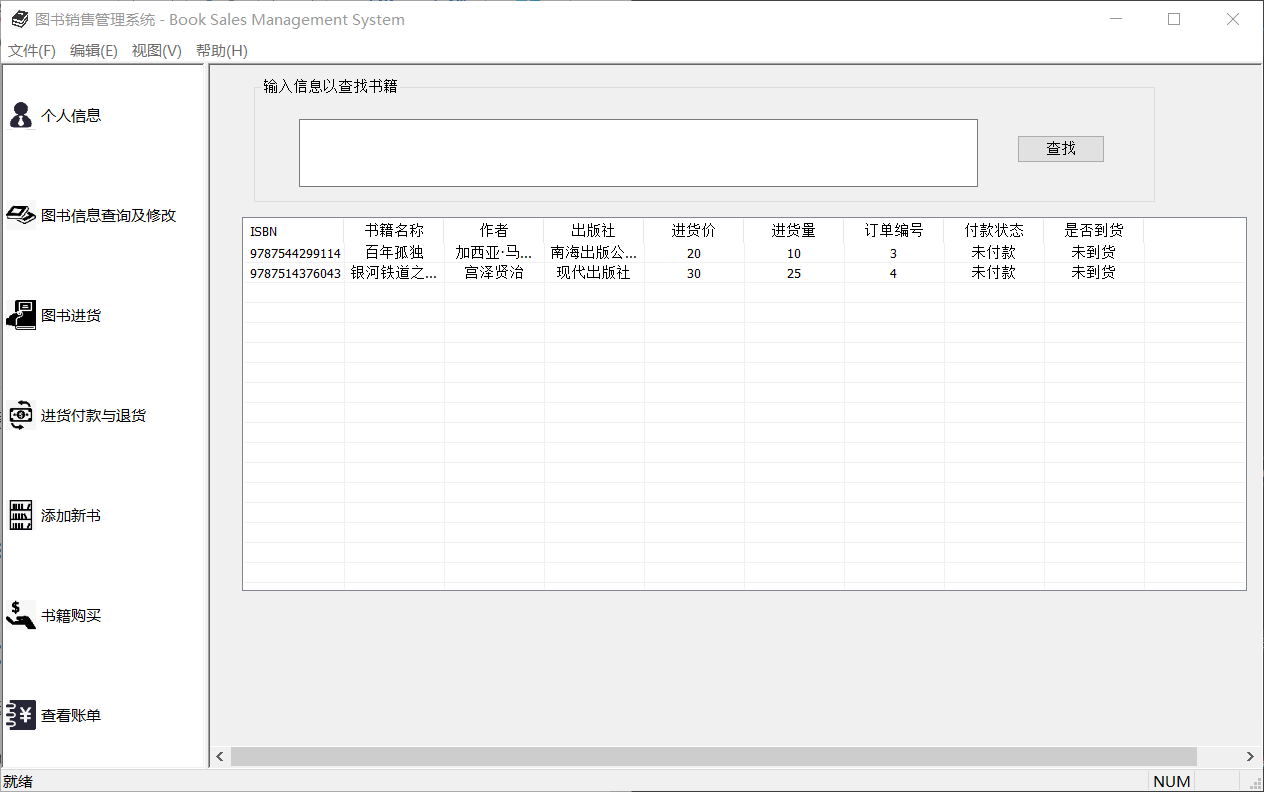
如果需要进货新书，按下下面按钮时检查这些编辑框是否都填入相应信息，都填入后将这些信息直接插入到restocklist表。且由于是新书，还需要在book表中插入相关信息。（如果输入的不是新书，则会产生主码冲突从而插入失败）



示例：如图进货一本新书。



查看进货列表，多了一条对应的订单。



**6. 进货付款：**

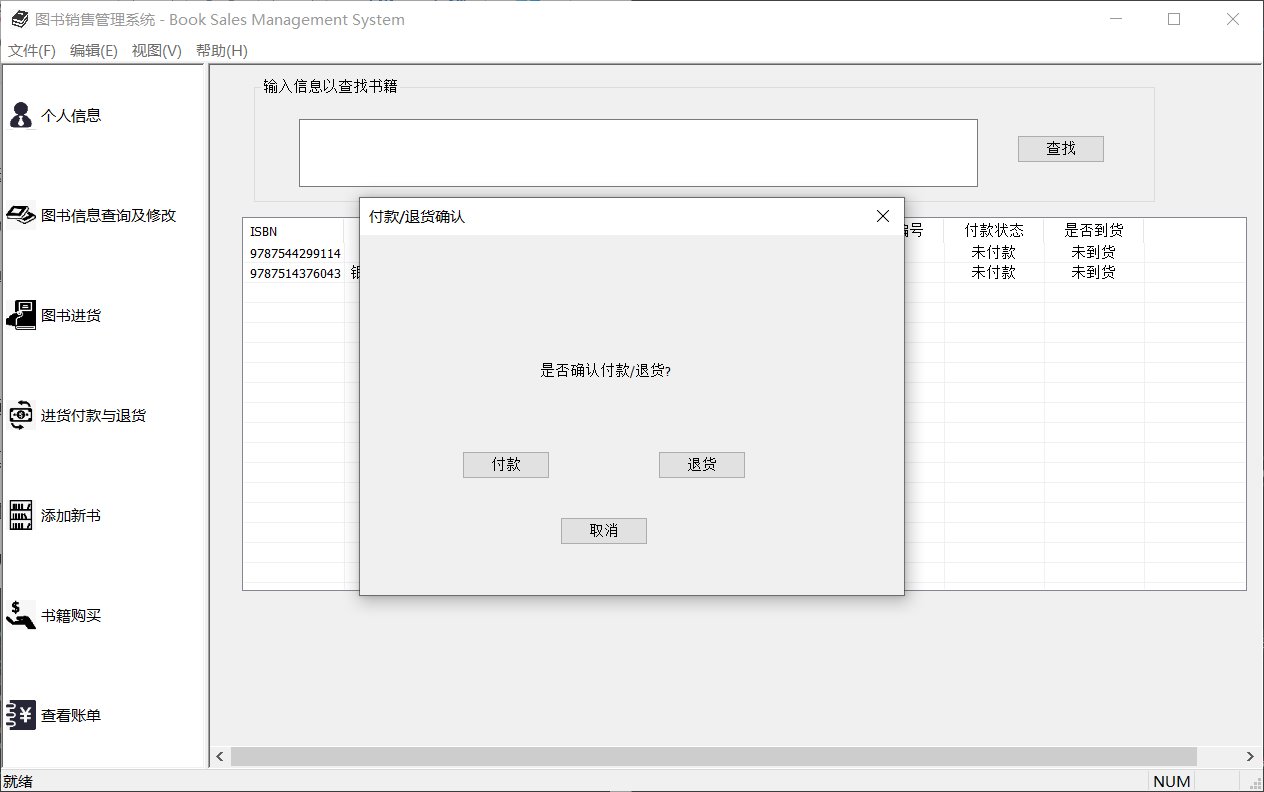
**查询正在进货的书籍，并给予付款，付款后书籍状态为已付款。**

**7. 图书退货：**

**对未付款的书籍可以进行退货，即将书籍状态改为已退货。**

在左侧选择“进货付款与进货界面”，可以查看所有尚未到货的订单。用select语句在book⋈restocklist中选择符合条件的数据，插入到列表显示即可。

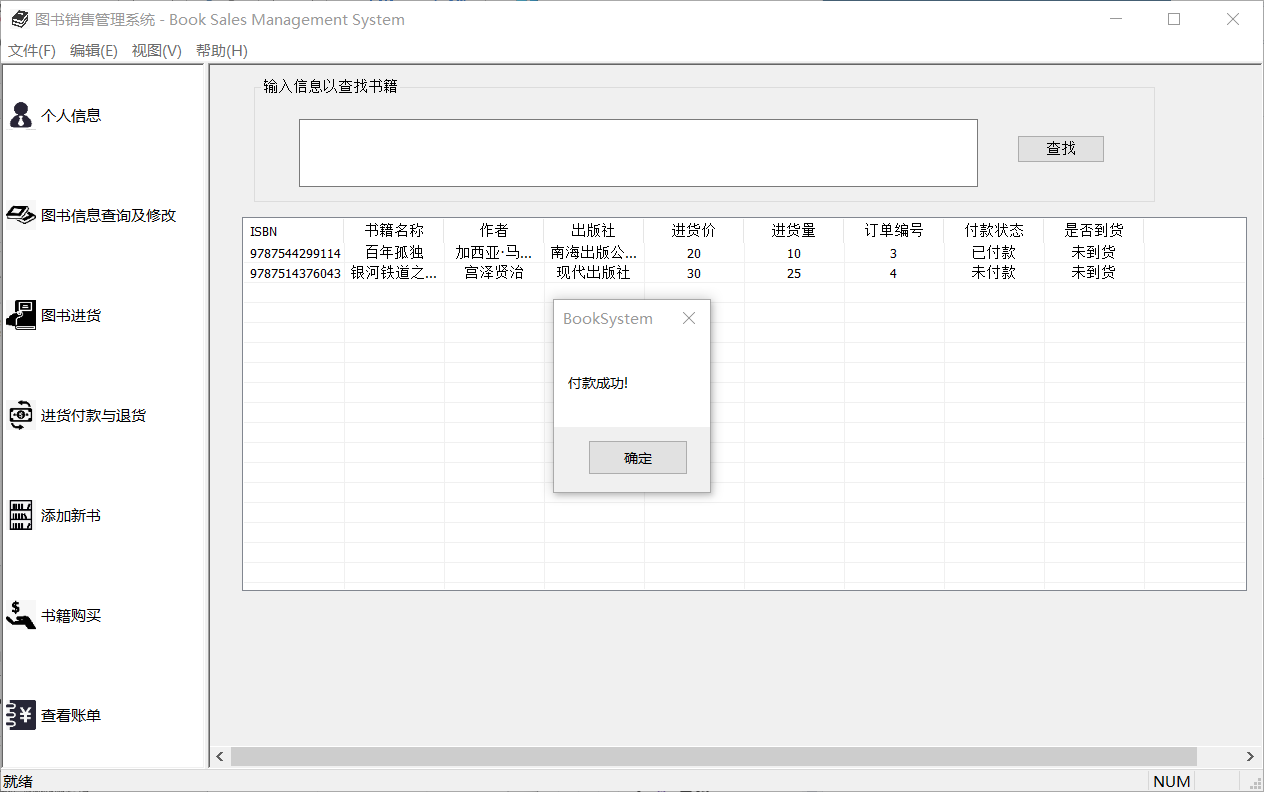
选中一条订单后双击，先判断该订单的付款状态，如果已付款或是已退货则弹出提示窗提醒，否则弹出付款/退货确认窗口。



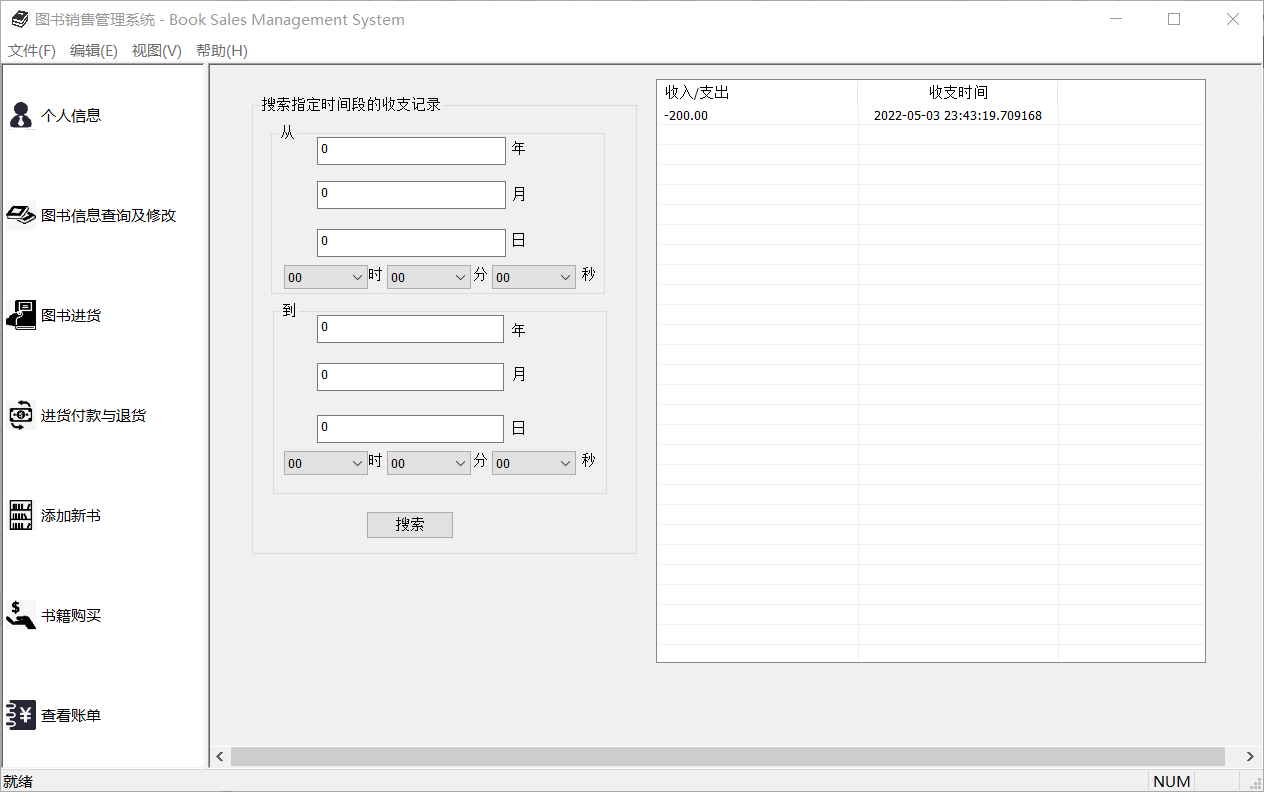
根据选择的项，执行相应操作：选择付款时，该订单状态变为已付款（先用MFC相关函数获取订单编号，用update语句修改restocklist，并向account表中用insert语句插入一条支出记录，其中支出的计算用进货价×进货量获得；选择退货时，订单状态变为已退货（方法同上）；选择取消时，不产生改变。



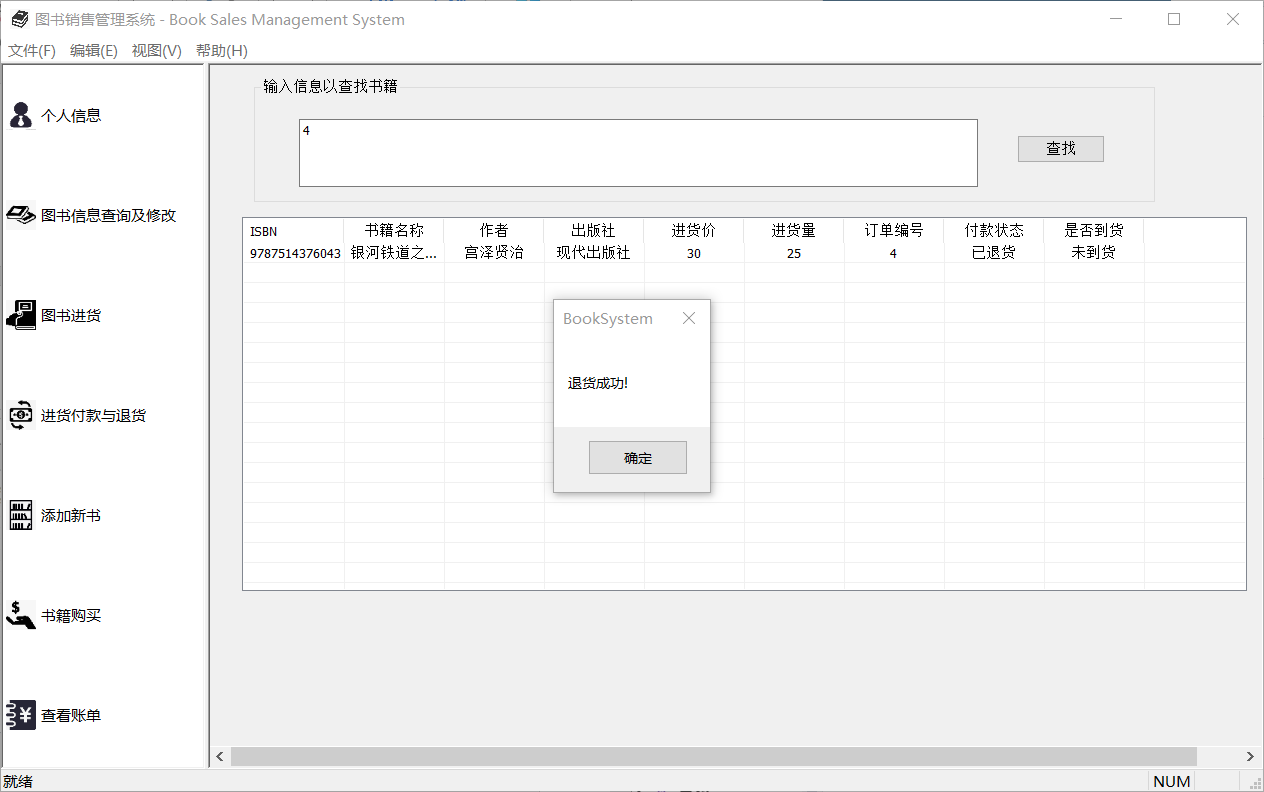
示例：为《百年孤独》的订单付款。



从左侧选择“查看账单”，可以看到多了一条支出。



搜索4（订单号），将《银河铁道之夜》退货，该订单状态变为“已退货”。



**8. 添加新书：**

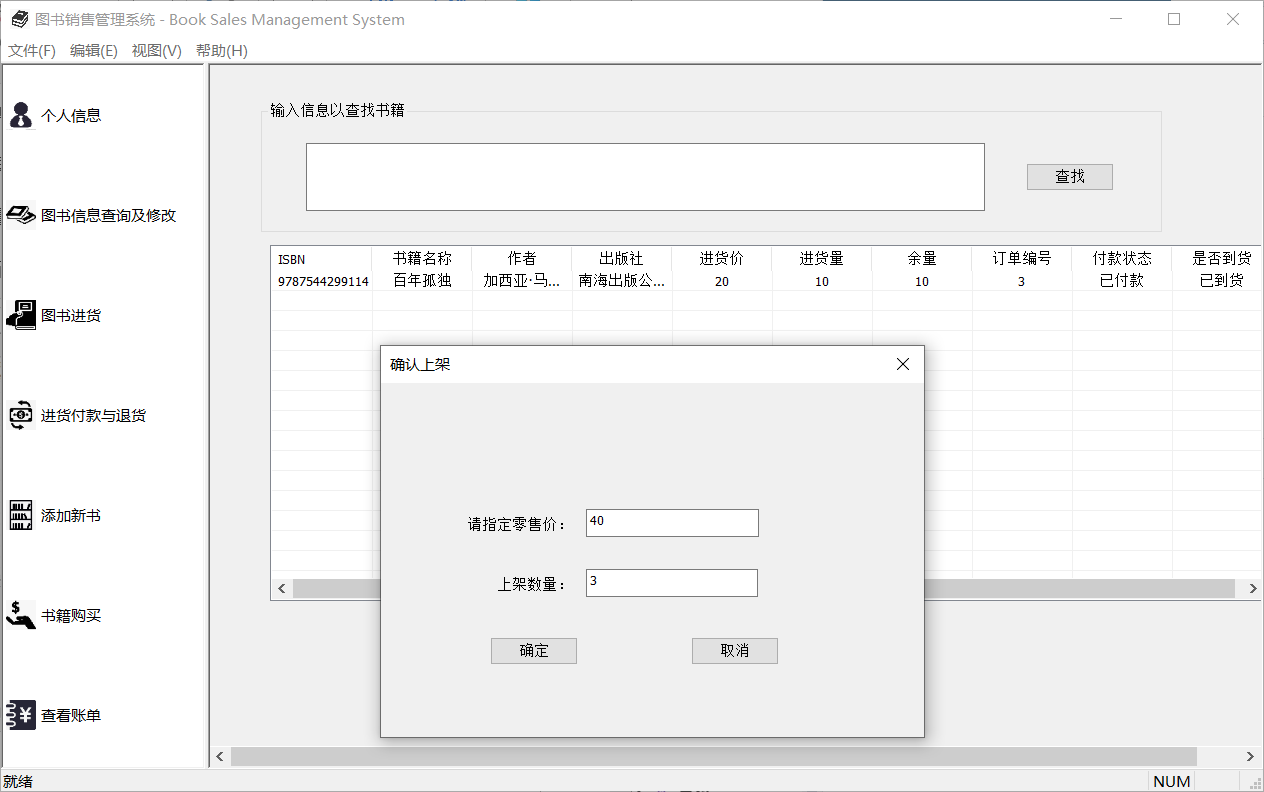
**对于已付款的书籍，当书籍到货后，可以将其添加到库存中，此时需要添加上书籍的零售价格。**

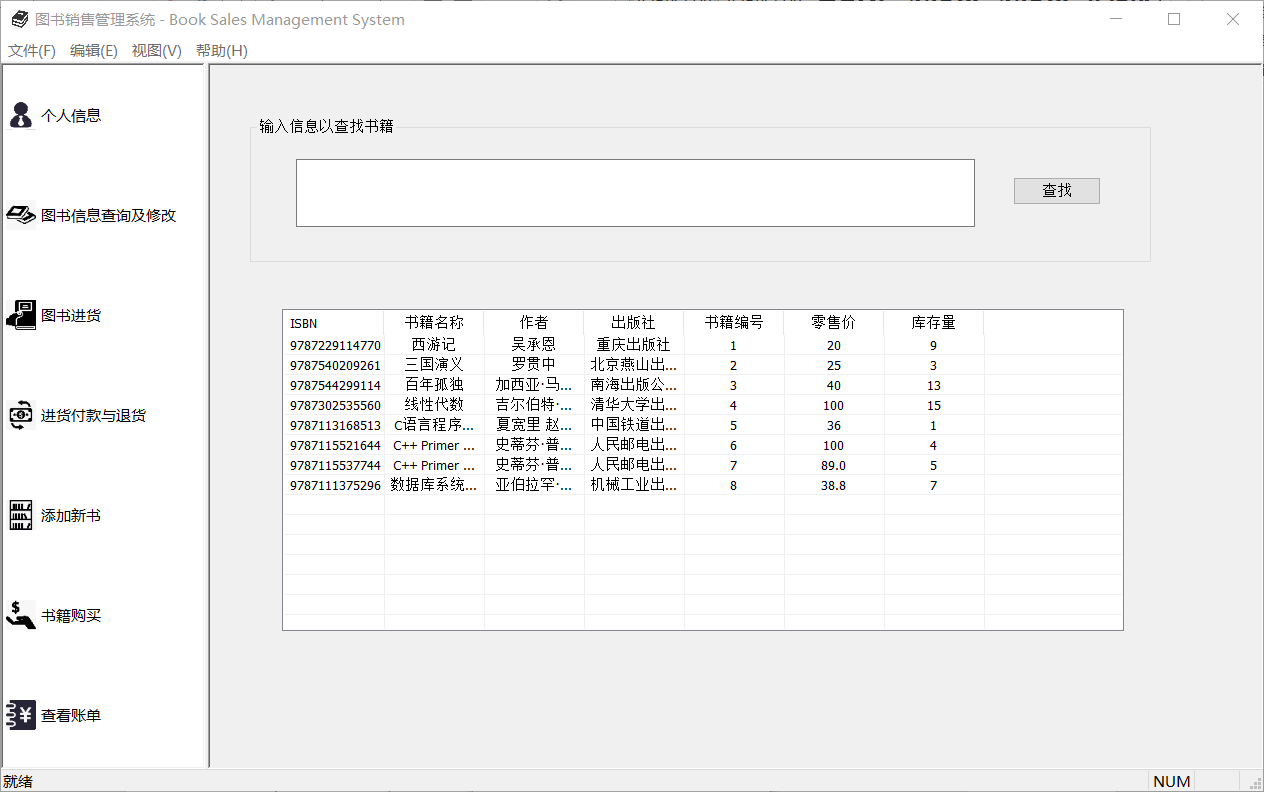
选择左侧“添加新书”项，进入添加新书页面，会显示所有已到货的订单：这通过select语句在book ⋈ restocklist中选择获取，并插入显示列表。



双击某行订单，可以将该订单的书籍上架。首先弹出上架窗口，指定上架数量和零售价。如果指定书籍在sale表中已存在，且指定零售价与原本书籍一样上，则将上架数量加到这本书的库存上；否则作为一条新信息插入sale表。

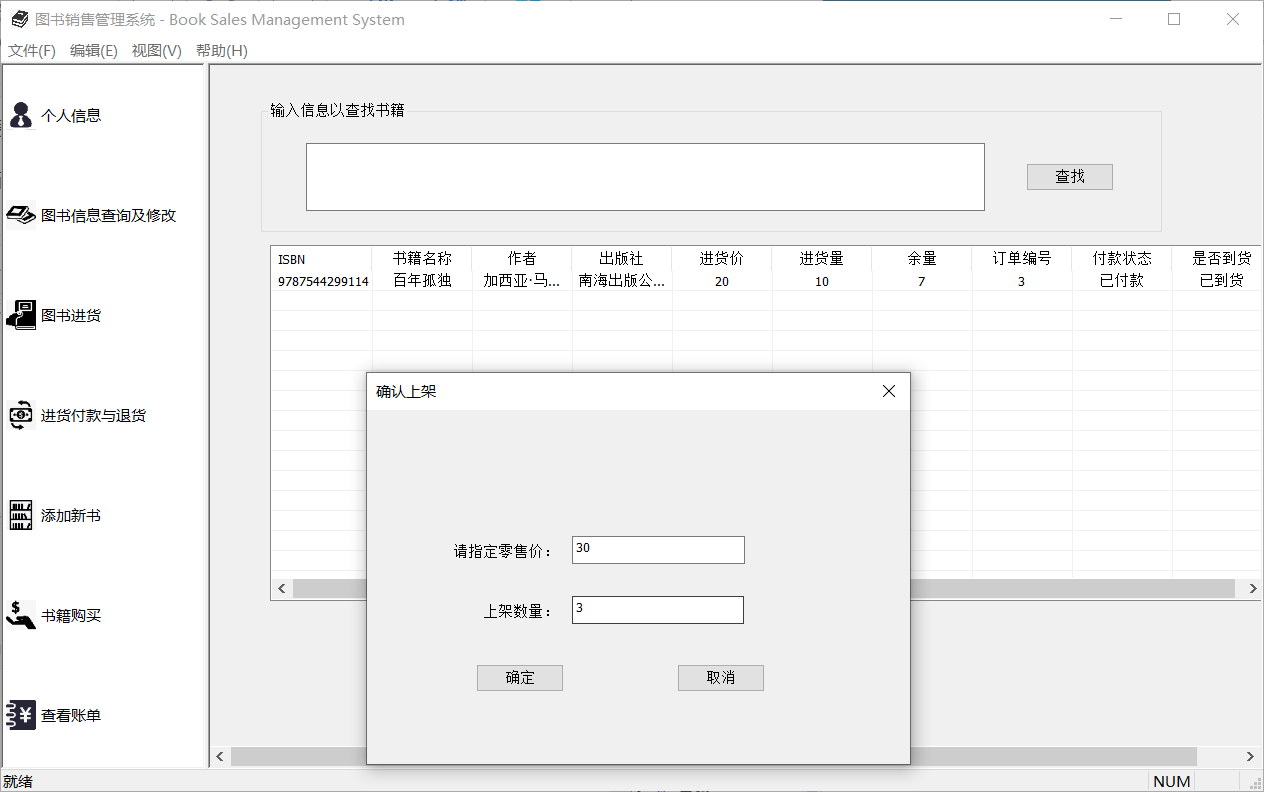
演示：将《百年孤独》上架3本，售价与原售价(40)一致

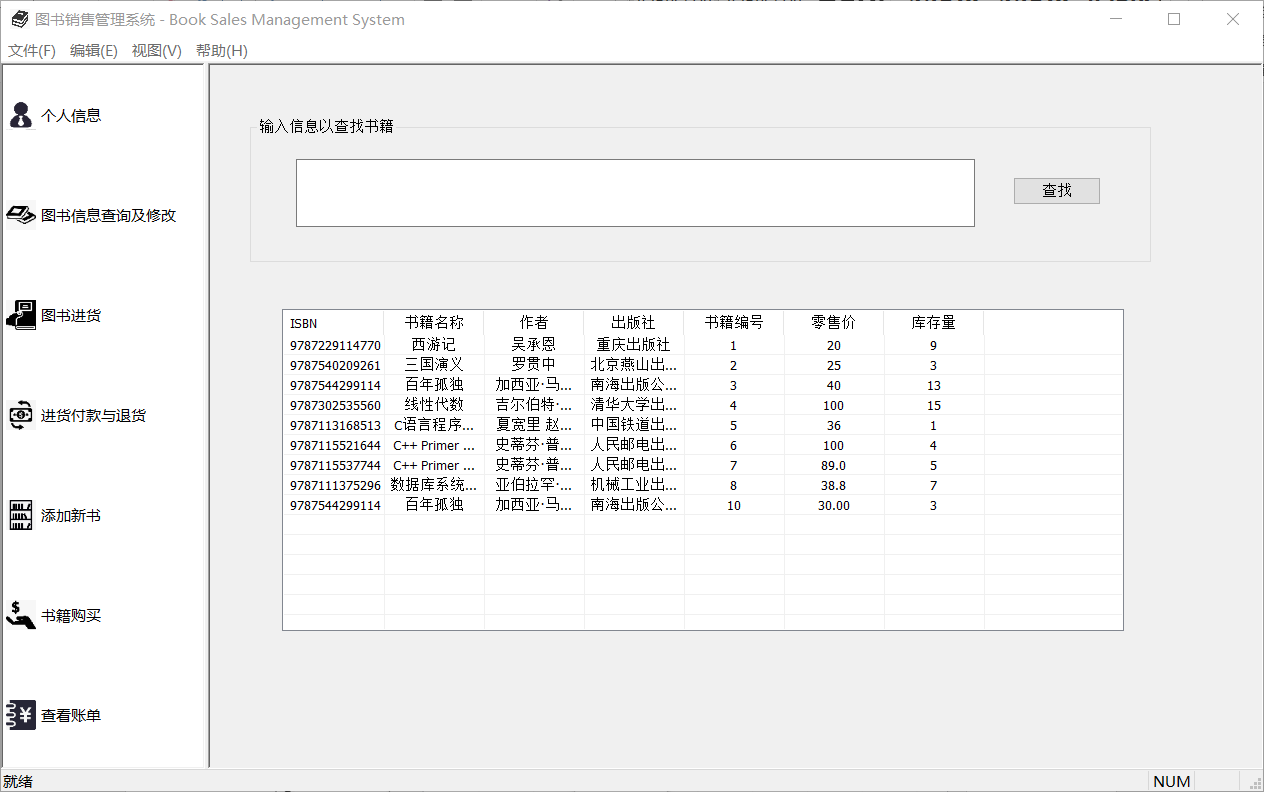




查看书籍列表，可以看到《百年孤独》的库存量从10变成了13。

再次上架《百年孤独》3本，售价设为30。





可以看到书籍列表中多出一条新记录，它的书籍信息与编号1一致，但售价和库存量不一样。

再次上架5本，提示库存不足。



实现：双击时获取行号，从而通过select订单编号获取该行信息。用select语句检查book表中是否有书籍的ISBN与该订单的ISBN一致，如果一致且指定售价也与该书籍售价一致，则用update语句更新该书籍库存，否则用insert语句插入一条新记录，最后更新订单（仓库）余量。此外如果检测到指定上架量大于余量（或者余量已经为0）则提示余量不足。



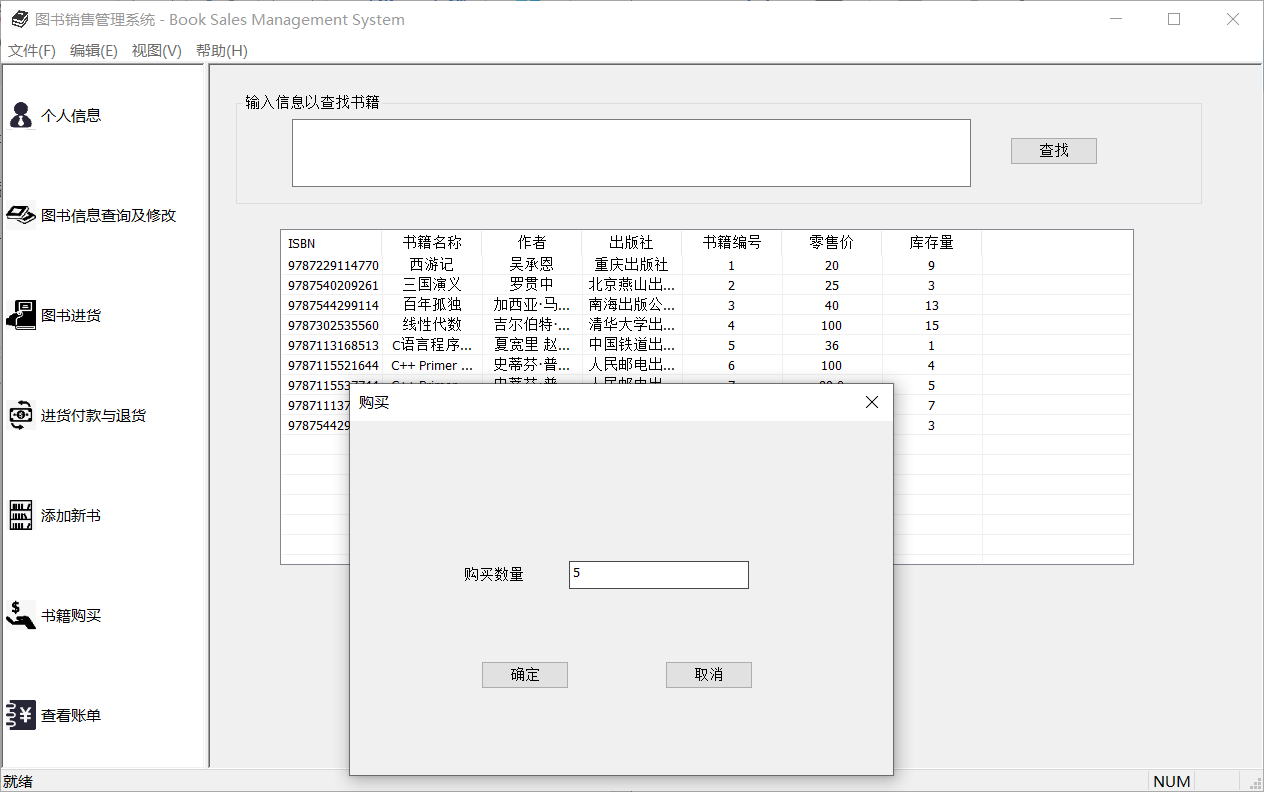
**9. 书籍购买：**

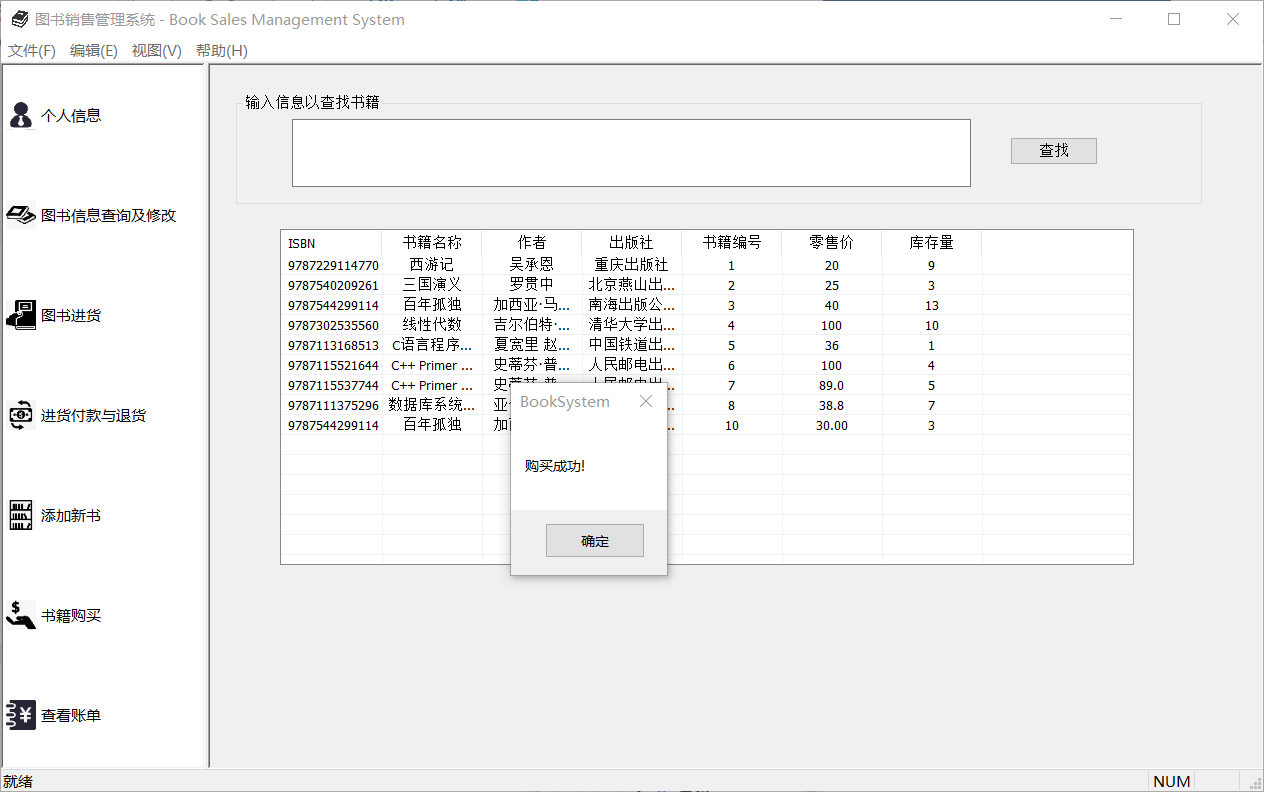
**使用标售零售价格购买书籍，这时书籍的库存数量需要相应地减少。**

选择左侧的“购买书籍”项，进入购买书籍界面：通过select语句查看book ⋈ sale表。

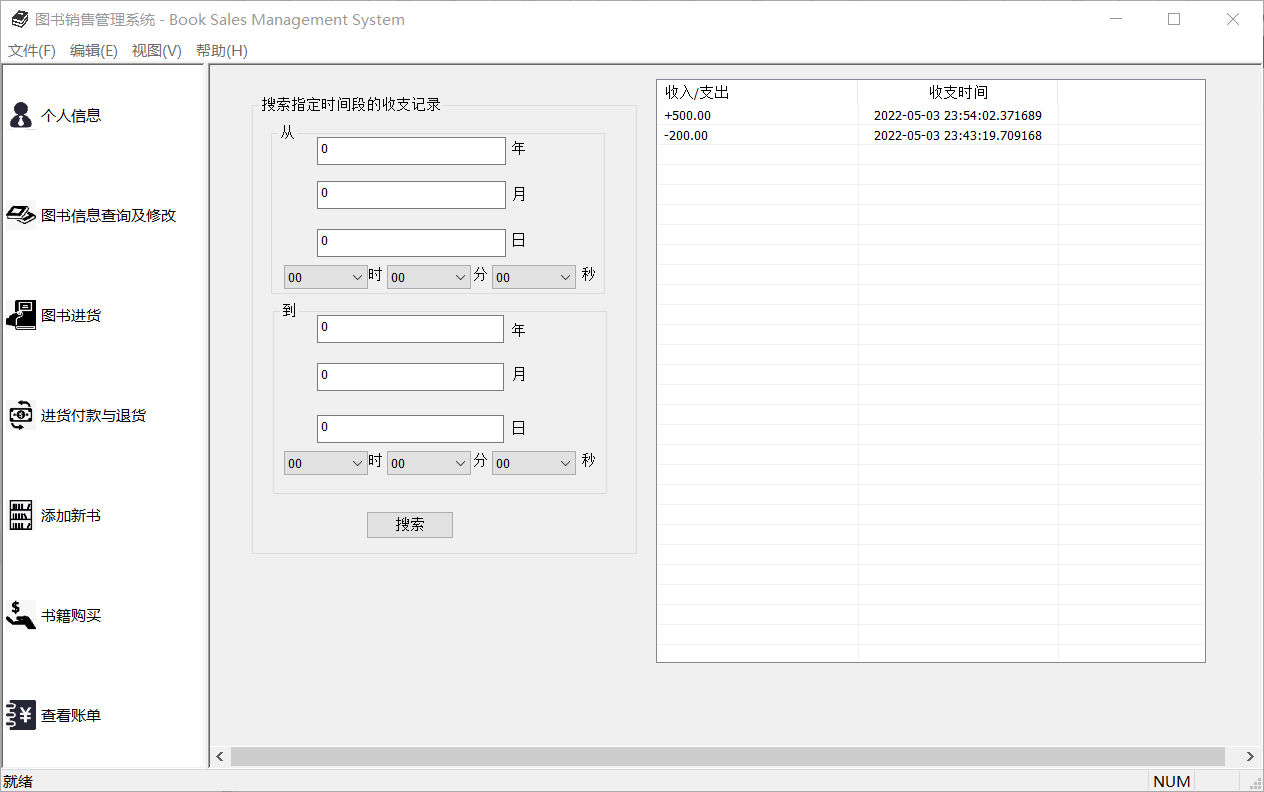
双击时，选择某本书籍购买，弹出购买窗口，并指定购买数量。先用update语句更新书籍库存，再用insert语句向account表插入一条收入记录，其中收入由购买数×售价计算得到。

示例：购买5本《线性代数》。





可以看到书籍库存有相应减少。



账单中增加的收入记录。

另外如果检测到购买数量大于库存时，会提示购买失败，并且不会添加收入记录。



**10. 财务管理：**

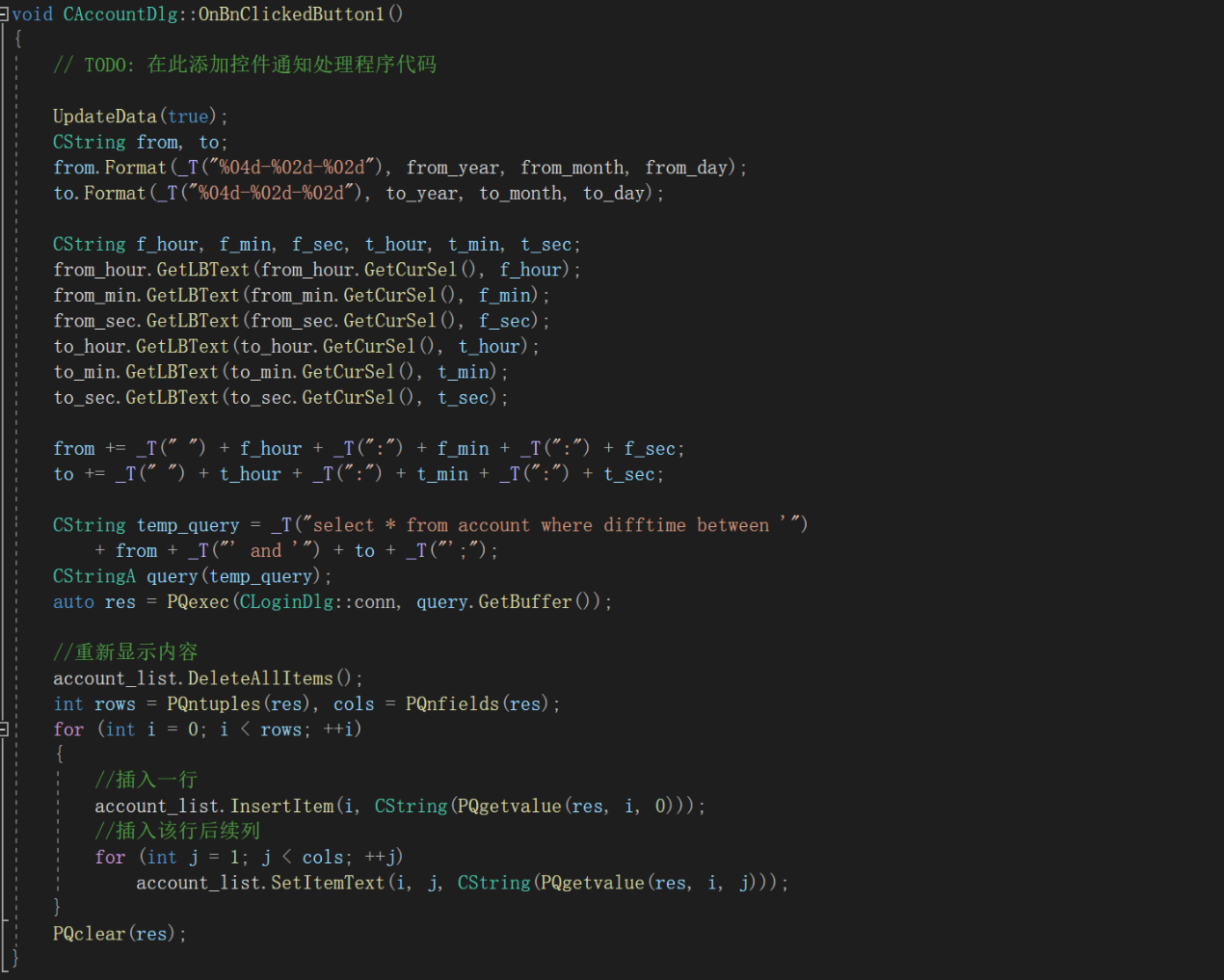
**当对书籍进货进行付款，或购买书籍时，系统的财务账户都要添加一条账单记录，记录下财务账户的支出或收入。**

**11. 查看账单**

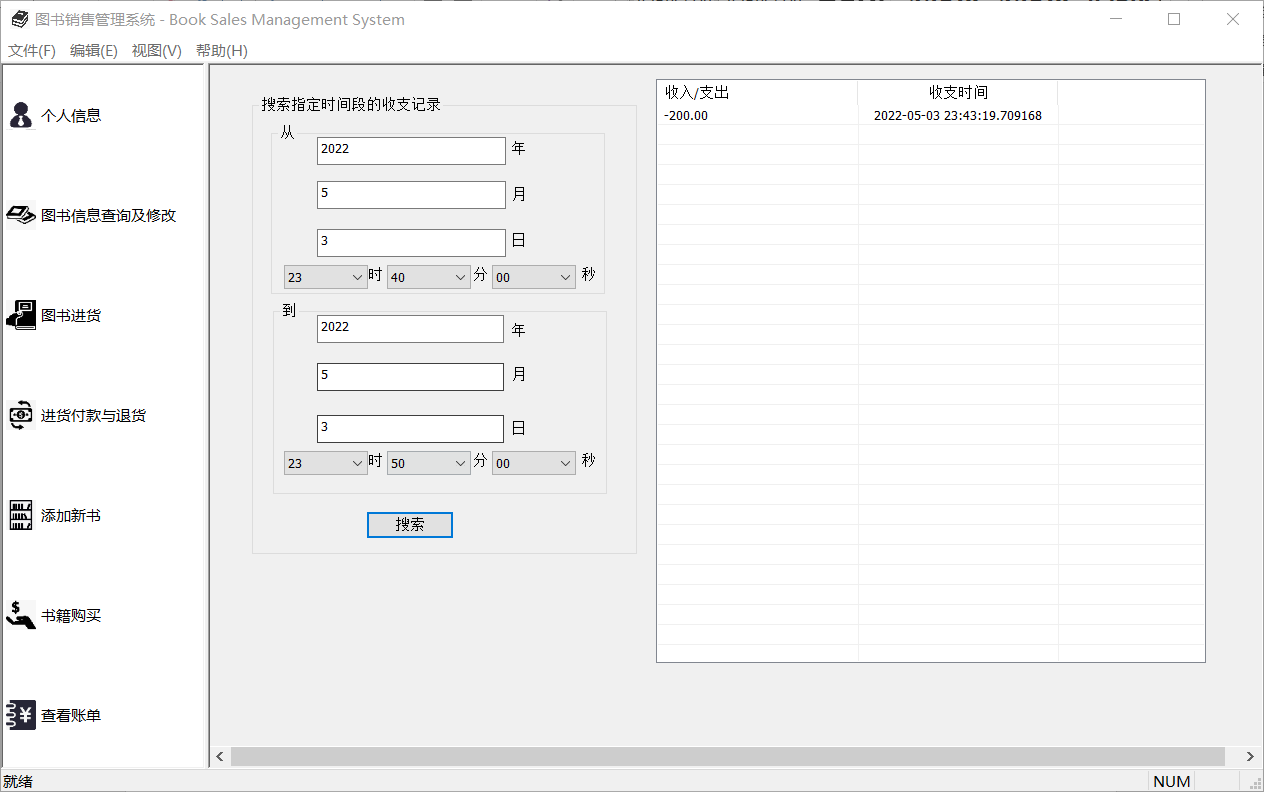
**查看某段时间内财务账户的收入或支出记录。**

功能10在前面已进行演示。

在左侧选择“查看账单”项进入账单页面，默认显示所有账单。通过填入指定时间段就能筛选需要的账单数据。实现：获取填入的时间段后，用select语句加上between…and条件来选择需要的收支。



示例：查找2022/5/3 23:40到2022/5/3 23:50的收支记录。点击搜索后显示相应的记录。



1. 特色和创新点

1、数据库设计形式较好。每个表都属于BCNF、3NF，数据的冗余少，如销售列表和进货列表的书籍使用同一个书籍表来记录ISBN与书名、作者、出版社之间的函数依赖关系(ISBN→title,author,press)。这样做相比在销售列表和进货列表分别记录一次书籍相关信息的好处是：

①修改书籍信息时，查看进货列表、购买列表的书籍信息也能相应改变，且：当查看信息（或购买）列表中有相同ISBN但售价库存不同的两本书时，只修改其中一本书的信息另一本书的信息也能得到同步。

②新书上架时，不需要特意添加新书信息到数据库（在进货阶段完成）。

③减少了信息的冗余。

2、在销售列表和进货列表中都没有使用ISBN作为主码而是分别使用了书籍编号和订单编号。这样做很大程度上是为了处理重复书籍的订单：在本图书销售系统中，销售列表或进货列表中都允许存在ISBN完全相同的书籍，这更真实地反映了现实中的销售系统的情况，如一本书可能有不同的售价或进货价，其相应的库存和进货量也可能不同。

3、允许用户在上架书籍时指定数量，从而更好地模拟了现实中书籍到货到仓库后，从仓库上架的行为。

4、在查询时，除了一般的模糊查询外，也为用户提供了详细信息查询，这样用户在对想要查找的书籍较了解时，能快速找到这本书的相关信息。

5、将查询与修改信息、进货、上架、购买相结合，管理员/用户在使用时可以先输入相关信息查询需要的书籍，更符合现实中对大量书籍处理的情况。

1. 实验分工

无

1. 提交文件说明

（一）数据库脚本

CreateDB：创建BookSalesManagementSystem数据库。

CreateTableAccount：创建Account表的脚本。

CreateTableAdministrator：创建Administrator表的脚本。

CreateTableBook：创建book表的脚本。

CreateTableSale：创建Sale表的脚本。

CreateTableRestocklist：创建Restocklist表的脚本。

AdminInsertion：插入元组，初始化Administrator表。

BookInsertion：插入元组，初始化Book表。

SaleInsertion：插入元组，初始化Sale表。

（二）源代码

BookSystem(CbookSystemApp类)（包括.cpp和.h文件，下同理）：MFC自动生成的类，定义应用程序的类行为。

BookSystemDoc(CBookSystemDoc类)：MFC自动生成的类，负责文本相关内容（如标题）。

BookSystemView(CBookSystemView类)：MFC自动生成的类，定义窗口外框。

MainFrame(CMainFrame类)：MFC自动生成的类，定义窗口内部行为。

CLoginDlg：定义登录窗口。

CSelectView：定义系统左侧的选择窗口。

CDisplayView：定义系统右侧用于显示不同功能对应的子窗口。

CUserDlg：定义用户界面。

CModifyPwdDlg：定义修改密码对话框。

CCreateUserDlg：定义创建用户对话框。

CSearchDlg：定义搜索、修改界面。

CSearchTag：定义详细修改对话框。

CModifyItemDlg：定义修改的输入新值对话框。

CRestockDlg：定义进货界面。

CPayDlg：定义对进货付款/退货界面。

CPayConfirmDlg：定义弹出的付款/退货确认对话框。

COnsaleDlg：定义上架界面。

COnsaleConfirm：定义上架数量及确认对话框。

CBuyDlg：定义购买界面。

CBuyConfirmDlg：定义购买数量及确认对话框。

CAccountDlg：定义查看账单界面。

（三）其他资源

res文件中的一些icon是我在网上找到素材后转换成ico文件得到的，主要是作为窗口图标和左侧选择列表各项内容的图标。

1. 实验总结

本次实验需要从无到有实现一个图书销售管理系统。首先需要实现的就是系统的图形界面：一方面，使用图形界面使得整个系统显得简洁明了，方便操作和展示；另一方面，设计成图形界面也利于整个系统的设计，使设计更简单。

我使用的客户端开发环境是MFC，因为它是内置在Visual Studio中的，能和C/C++方便地结合，且也容易上手使用。为此，我首先去简单自学了一下MFC的使用，了解了如何使用MFC制作图形界面。这次实验也是我第一次制作GUI，在这方面有非常大的收获。

另一方面，本次实验也是第一次使用高级语言来进行数据库管理与开发。这让我学会了如何正确使用相应的PostgreSQL手册，主要参考了[第 34 章 libpq - C 库 (postgres.cn)](http://www.postgres.cn/docs/14/libpq.html)。

在设计过程中也遇到了相当多的问题。比如说，我最开始设计的数据库并没有把书籍信息单独作为一张表book，这样做的缺陷在上面就已经分析过了，诸如更新不能同步、数据冗余等。在课程进行到第八章规范化设计时，我才意识到我之前的设计是根本不满足BCNF的，需要重新修改。又比如，关于登录方法，我最开始的想法是先用设定好的语句连接到数据库，再通过select语句来查询是否有与所输入的用户名和密码所匹配的元组，因为我最初并不太清楚如何用SQL指令来修改用户名和密码。但这样设计有很大的弊端：程序的实现依赖于事先设定的连接语句，可移植性差，当其他数据库管理者使用时还需要在cpp文件中进行修改，这无疑是不好的。在查询了[SQL 命令 (postgres.cn)](http://www.postgres.cn/docs/14/sql-commands.html)后，我重新设计了登录方法，改为了用输入的用户名和密码来连接数据库，这样使得登录数据库不依赖于设定的固定登录语句，但这样做又有两个问题：一是创建/修改用户信息时，必须同时在数据库的administrator表和数据库实际用户中进行修改，显得冗余；二是由于PostgreSQL不允许用户修改自己的用户名（只能修改别人的），这是最大的问题，所以我最后也放弃了这个设计。在查阅了[34.14. 环境变量 (postgres.cn)](http://www.postgres.cn/docs/14/libpq-envars.html) 、[34.15. 口令文件 (postgres.cn)](http://www.postgres.cn/docs/14/libpq-pgpass.html)后，我选择了一种比较简单的设计，即提供给用户一个配置文件(login.ini)，其中保存了数据库的用户名和密码，登陆时读取文件内容，从而提供修改的能力。

一些仍未解决的问题也是存在的。一个是我在设计时，本想将两条关联性较强的语句设计为一个事务（如购买书籍时，更改书籍存量和插入一条账单），但使用begin语句后总是出错，查询后才知道事务被执行后不允许有后续的破坏（“被发生破坏时仍然是持久的”），我不太清楚如何处理所以最后还是回到了最开始的设计，即所有语句分开执行。另一个问题是关于“到货“相关：在要求描述中提到了到货后才能上架，所以我添加了是否到货这一属性标识这一状态，但在图书销售系统中是没有渠道能修改是否到货的状态的，或许是与其他程序相关联在一起的。

总的来讲，这次实验我确实学到了很多东西，也花了很多时间反复修改自己的设计。最终的成品虽然还是有一些瑕疵，但我还是比较满意了。希望在之后的期末实验中也能有同样的收获。