《数据库系统》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**
方维远	18340039	计算机科学与技术	销售模块、图形化界面模块
侯少森	18340055	计算机科学与技术	进货模块、退货模块
胡霆熙	18340057	计算机科学与技术	库存查询模块、销售数据模块

提交时间： 2021 年 1 月 9 日

一. 开发环境与开发工具

开发环境：

操作系统：Windows 10 Enterprise Version 20H2

数据库管理系统：MySQL 5.7

编程语言：Python 3.8

开发工具：

图形界面库：Tkinter

MySQL 客户端库：PyMySQL

二. 系统需求分析

书店可以通过图书销售管理系统实现对图书的销售管理, 本系统主要处理的数据有: 书籍信息、进货单、销售数据、退货单、供应商信息等。

系统数据字典:

数据结构:

书表:

描述: 书店库存的书籍的信息

数据项:

书籍编号:

描述: 唯一标识库中书籍的数字编号

定义: 整型数

书名:

描述: 该书的书名

定义: 字符型名称

数量:

描述: 该书的库存量

定义: 整型数

成本价:

描述: 该书的进货价格

定义: 浮点型数(保留两位小数)

出售价:
描述: 该书的卖出价格
定义: 浮点型数(保留两位小数)

供应商表:
描述: 进货书籍的供应商信息
数据项:
编号:
描述: 标识供应商的数字编号
定义: 整型数
名字:
描述: 供应商的名字
定义: 字符型汉字名称
书号:
描述: 供应商提供的书的编号
定义: 整型数
价格:
描述: 供应商卖出该书的价格
定义: 浮点型数(保留两位小数)

进货表:
描述: 进货书籍的信息
数据项:
书号:
描述: 进货书的编号
定义: 整型数
数量:
描述: 进货书的数量
定义: 非负整型数
供应商编号:
描述: 进货该书的供应商编号
定义: 整型数

卖出、退货表:
描述: 卖出、退货书籍的信息
数据项:
书号:
描述: 卖出/退货时书的编号
定义: 整型数
月:
描述: 卖出该书的月份
定义: 非零整型数
日:
描述: 卖出该书的日数
定义: 非零整型数
数量:
描述: 卖出该书的数量(含退货)

定义: 整型数

顾客号:

描述: 买此书的顾客编号

定义: 整型数

数据流:

书表数据流:

说明: “书表”数据结构在系统内的流向

数据流来源: 进货、退货、销售事务

数据流去向: 书本库存查询事务、书本销售数据查询事务

进货表数据流:

说明: “进货表”数据结构在系统内的流向

数据流来源: 进货事务

数据流去向: 书表信息事务

卖出、退货表数据流:

说明: “卖出、退货表”数据结构在系统内的流向

数据流来源: 退货、销售事务

数据流去向: 书表信息事务

处理过程:

实时书籍库存计算:

说明: 随着进货、销售、退货事务的不断进行, 能实时计算出当前各书本的库存

输入: 进货数据流、卖出、退货数据流

输出: 计算出各书本的当前库存

销售数据计算:

说明: 根据日期来计算日/月销售数据

输入: 进货数据流、卖出、退货数据流

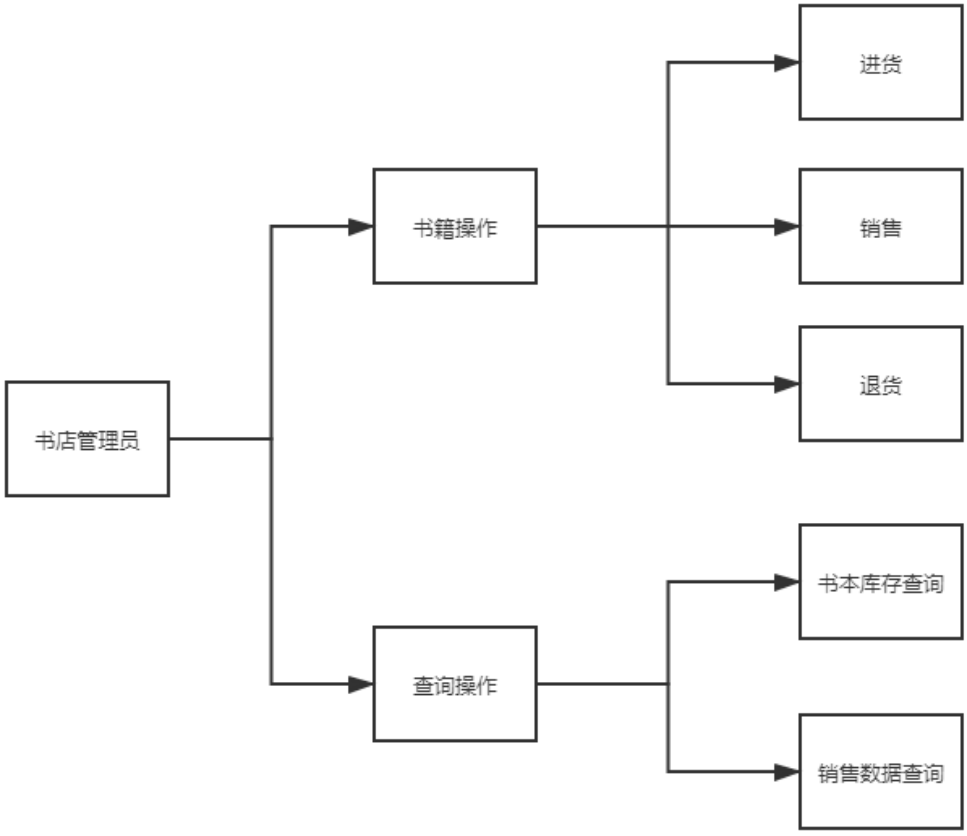
输出: 计算出日/月销售数据

三．功能需求分析

书店可以通过图书销售管理系统实现对图书的销售管理，系统主要包括进货、退货、统计、销售功能，具体如下：

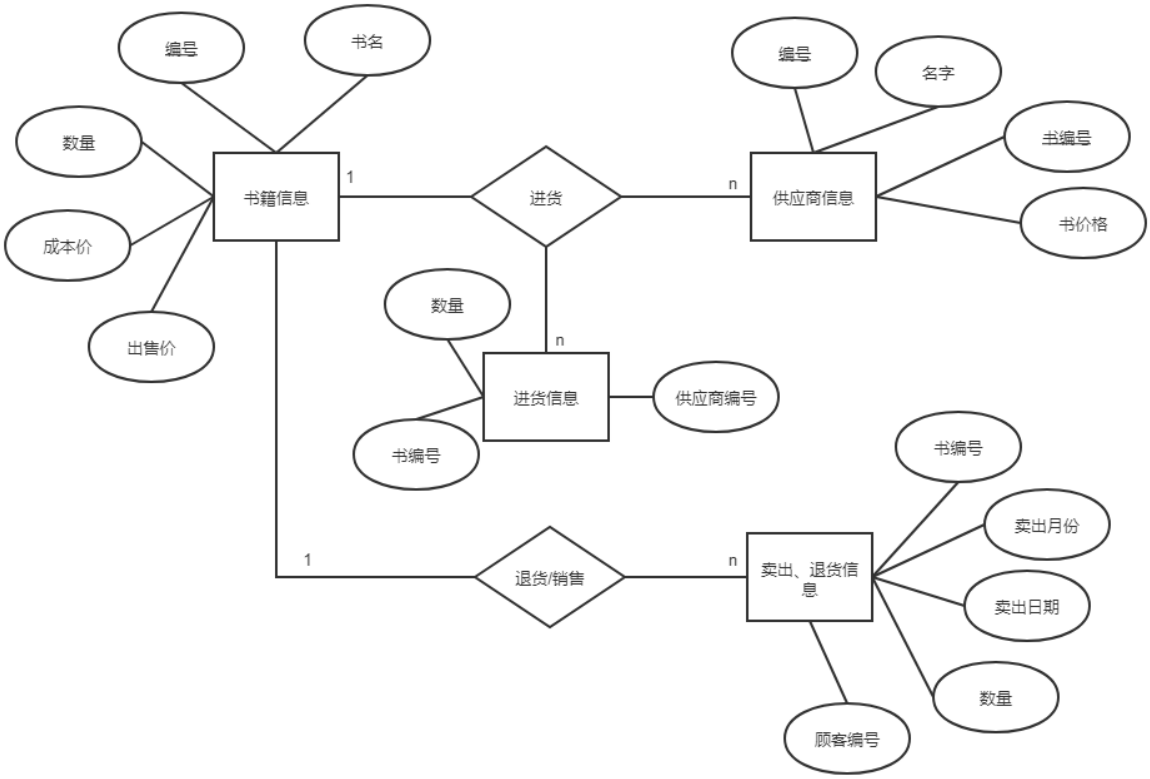
1. 进货：根据某种书籍的库存量及销售情况确定进货数量，根据供应商报价选择供应商。输出一份进货单并自动修改库存量，把本次进货的信息添加到进货库中。
2. 退货：顾客把已买的书籍退还给书店。输出一份退货单并自动修改库存量，把本次退货的信息添加到退货库中。
3. 统计：根据销售情况输出统计的报表。一般内容为每月的销售总额、销售总量及排行榜。
4. 销售：输入顾客要买书籍的信息，自动显示此书的库存量，如果可以销售。打印销售单并修改库存，同时把此次销售的有关信息添加到日销售库中。

系统功能模块图:



四．系统设计

1. 数据概念结构设计（系统 ER 图）



2. 数据库关系模式设计

   按照 ER 图到逻辑关系模式的转换规则，可得到系统关系如下:

   (1) 书籍信息(编号，书名，数量，成本价，出售价)

   (2) 供应商信息(编号，名字，书编号，书价格)

   (3) 进货信息(书编号，数量，供应商编号)

   (4) 卖出、退货信息(书编号，卖出月份，卖出日期，数量，顾客编号)

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 MySQL 命令来呈现。

   创建数据库表的 MySQL 命令如下:

```
create database booksell default character set utf8;
use booksell;
create table book(
        id int,                  # 编号（主码）
    title varchar(20),           # 标题
    count int unsigned,          # 剩余量
    price_in numeric(4,2),       # 成本价
    price_out numeric(4,2),      # 出售价
    primary key(id)
);
create table supplier(
    id int,                      # 供应商 id
    name varchar(20),            # 供应商名字
    book_id int,                 # 书的编号
    price numeric(4,2),          # 该书的价格
    foreign key(book_id) references book(id),
    primary key(id, book_id)
);
create table purchase(
        id int,                  # 书的编号
    num int unsigned,            # 数量
    supplier_id int,             # 供应商 id
    foreign key (id) references book(id),
    foreign key (supplier_id) references supplier(id)
);
create table sold(
        id int,                  # 书的编号
    month int unsigned,          # 卖出的月份（用于销售统计）
    day int unsigned,            # 卖出的日期
    num int,                     # 数量
    customer_id int,             # 顾客 id
    foreign key (id) references book(id)
```

```
);
```

五．系统功能的实现

1. 登录

通过输入的用户名和密码来连接到数据库，从而可以进行后续操作。

2. 进货

根据书籍的编号、供货商编号以及进货数量来进购图书。

```
def purchase():#进货
    global win
    win = tk.Tk()
    win.title('图书销售管理')
    win.geometry('900x600')
    win.wm_attributes('-topmost', 1)
    tk.Label(win, text='请填写进购图书的信息:', font=('微软雅黑',
20)).place(x=200, y=30)

    global tree#建立树形图
    yscrollbar = ttk.Scrollbar(win, orient='vertical')#右边的滑动按钮
    tree = ttk.Treeview(win, columns=('1', '2', '3', '4', '5', '6'),
show="headings",yscrollcommand=yscrollbar.set)
    tree.column('1', width=100, anchor='center')
    tree.column('2', width=100, anchor='center')
    tree.column('3', width=100, anchor='center')
    tree.column('4', width=100, anchor='center')
    tree.column('5', width=100, anchor='center')
    tree.column('6', width=100, anchor='center')
    tree.heading('1', text='图书 id')
    tree.heading('2', text='书名')
    tree.heading('3', text='供应商 id')
    tree.heading('4', text='供应商')
    tree.heading('5', text='进价')
    tree.heading('6', text='库存')
    tree.place(x=120, y=100)
    yscrollbar.place(x=800,y=150)
    db = pymysql.connect(host="localhost", user=user_name,
password=pass_word, database="booksell")
    cursor = db.cursor()
    try:
        cursor.execute("select book_id,title,supplier.id,name,price,count
from supplier,book where book_id = book.id group by supplier.id,book_id
order by book_id,price")
    except:
        msg._show(title='错误！',message='数据库查询出错！')
        return
    results=cursor.fetchall()
    if results:
        l= len(results)
        for i in range(0,l):#查询到的结果依次插入到表格中
            tree.insert('',i,values=(results[i]))
    else :
```

```
        tree.insert('', 0,values=('查询不到结果','查询不到结果','查询不到结果','查询不到结果','查询不到结果','查询不到结果'))
    db.close()

    global b_id
    tk.Label(win, text='图书 id：', font=('宋体', 12)).place(x=100, y=400)
    b_id = tk.Entry(win, font=('宋体', 12), width=5)
    b_id.place(x=170, y=400)

    global p_id
    tk.Label(win, text='供货商 id：', font=('宋体', 12)).place(x=250, y=400)
    p_id = tk.Entry(win, font=('宋体', 12), width=5)
    p_id.place(x=350, y=400)

    global amount
    tk.Label(win, text='数量：', font=('宋体', 12)).place(x=430, y=400)
    amount = tk.Entry(win, font=('宋体', 12), width=5)
    amount.place(x=500, y=400)

    tk.Button(win, text='确认', font=('宋体', 12), width=10, command=lambda:add(b_id,p_id,amount)).place(x=600, y=400)

def add(b_id,p_id,num):#添加图书信息到数据库中
    db = pymysql.connect(host="localhost", user=user_name, password=pass_word, database="booksell")
    cursor = db.cursor()

    try:
        cursor.execute('select count,price from book,supplier where book_id = book.id and book.id = ' + b_id.get() + ' and supplier.id = ' + p_id.get())
        res = cursor.fetchone()
        rest_num = res[0]
        price_in = res[1]
    except:
        msg._show(title='错误！',message='数据库查询出错！')
        cursor.execute("rollback")                    # 出错时需要回滚
        db.close()
        return

    try:
        cursor.execute("start transaction")
```

```
        cursor.execute("update book set count = " +
str(rest_num+int(num.get())) + ', price_in = ' +
                       str(price_in) + ', price_out=' + str(format(1.2 *
float(price_in), '.2f')) + 'where id = ' + b_id.get())
        # 添加进货记录
        cursor.execute('select num from purchase where id =' + b_id.get()
+ ' and supplier_id=' + p_id.get())
        record_num = cursor.fetchone()
        # 如果没有当天的进货记录则创建新的进货记录
        if record_num == None:
            cursor.execute("insert into purchase values(" + b_id.get() +
',' + num.get() + ',' + p_id.get() + ")")

        # 否则直接修改当天的进货记录
        else:
            if record_num[0] + int(num.get()) == 0:
                cursor.execute("delete from purchase where id=" +
b_id.get() + " and supplier_id=" + p_id.get())

            else:
                cursor.execute(
                    "update purchase set num=" + str(record_num[0] +
int(num.get())) + " where id=" + b_id.get() + " and supplier_id=" +
p_id.get())

    except:
        msg._show(title='错误！',message='数据库查询出错！')
        cursor.execute("rollback")                # 出错时需要回滚
        db.close()
        return

    db.commit()#这句不可或缺，当我们修改数据完成后必须要确认才能真正作用到数据
库里

    db.close()
    msg.showinfo(title='成功！', message='新书已入库！')
```

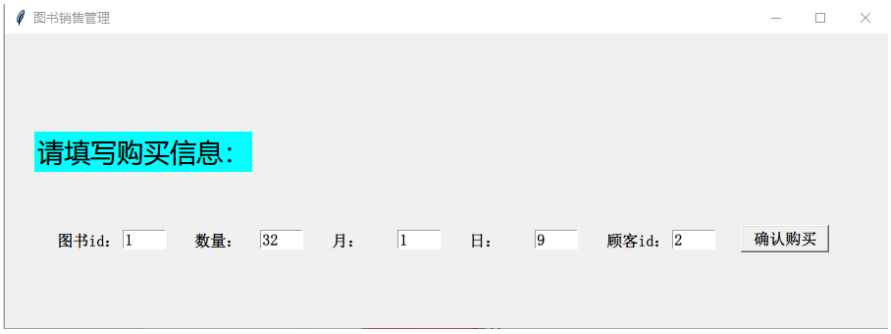
3. 销售

根据书籍编号、购买数量、月份日期、顾客编号来销售图书。

```
def sell():#卖书
    global win
    win = tk.Tk()
    win.title('图书销售管理')
    win.geometry('900x300')
    win.wm_attributes('-topmost', 1)
    tk.Label(win, text='请填写购买信息: ', bg='cyan',font=('微软雅黑',
20)).place(x=30, y=100)

    global b_id2
    tk.Label(win, text='图书 id: ', font=('宋体', 12)).place(x=50, y=200)
    b_id2 = tk.Entry(win, font=('宋体', 12), width=5)
    b_id2.place(x=120, y=200)

    global amount2
    tk.Label(win, text='数量: ', font=('宋体', 12)).place(x=190, y=200)
    amount2 = tk.Entry(win, font=('宋体', 12), width=5)
    amount2.place(x=260, y=200)

    global month2
    tk.Label(win, text='月: ', font=('宋体', 12)).place(x=330, y=200)
    month2 = tk.Entry(win, font=('宋体', 12), width=5)
    month2.place(x=400, y=200)
```

```
    global day2
    tk.Label(win, text='日：', font=('宋体', 12)).place(x=470, y=200)
    day2 = tk.Entry(win, font=('宋体', 12), width=5)
    day2.place(x=540, y=200)

    global customer2
    tk.Label(win, text='顾客 id：', font=('宋体', 12)).place(x=610, y=200)
    customer2 = tk.Entry(win, font=('宋体', 12), width=5)
    customer2.place(x=680, y=200)

    tk.Button(win, text='确认购买', font=('宋体', 12), width=10,
command=lambda:confirm_sell(b_id2,amount2,month2,day2,customer2)).place(
x=750, y=195)

def confirm_sell(b_id,num,month,day,c_id):
    db = pymysql.connect(host="localhost", user=user_name,
password=pass_word, database="booksell")
    cursor = db.cursor()
    if (not validNum(int(num.get()))) or (not
validDate(int(month.get()),int(day.get()))) or (not
validCustomer(int(c_id.get()))):
        return
    # 输入合法则从数据库获取数据
    try:

        cursor.execute("start transaction")
        # 如果使用书名查询，则先获取 id
        cursor.execute("select id from book where id = " + b_id.get())
        book = cursor.fetchone()
        if book == ():
            msg._show(title='错误！',message='查询错误：查询不到该书数据')
            db.commit()
            return
        # 获取该书的库存数量
        cursor.execute('select count from book where id = ' + b_id.get())
        rest_num = cursor.fetchone()
        if rest_num == ():
            msg._show(title='错误！',message='查询错误：查询不到该书数据')
            db.commit()
            return
        rest_num = rest_num[0]
        # 库存不足则出错
        if int(num.get()) > rest_num:
```

```
            msg._show(title='错误！',message='库存不足：该书库存只剩余' +
str(rest_num) + '本')
            db.commit()
            return
        # 否则可以出售
        else:
            # 修改库存
            cursor.execute("update book set count = " + str(rest_num -
int(num.get())) + ' where id = ' + b_id.get())
            cursor.execute('select num from sold where id = ' +
b_id.get() + ' and month=' + month.get() + ' and day=' +
                           day.get() + ' and customer_id=' + c_id.get())
            record_num = cursor.fetchone()
            # 如果没有当天的销售记录则创建新的销售记录
            if record_num == None:
                cursor.execute("insert into sold values(" + b_id.get() +
',' + month.get() + ',' +  day.get() + ',' +
                                    num.get() + ',' + c_id.get() + ")")
            # 否则直接修改当天的销售记录
            else:
                if record_num[0]+int(num.get()) == 0:
                    cursor.execute("delete from sold where id=" +
b_id.get() +
                                   " and month=" + month.get() +" and
day=" + day.get() + " and customer_id=" + c_id.get())
                else:
                    cursor.execute("update sold set num=" +
str(record_num[0]+int(num.get())) + " where id=" + b_id.get() +
                                   " and month=" + month.get() +" and
day=" + day.get() + " and customer_id=" + c_id.get())
    except:
        msg._show(title='错误！',message='数据库查询或修改出错')
        cursor.execute("rollback")                  # 出错时需要回滚
    db.commit()
    db.close()
    msg.showinfo(title='购买成功', message='购买成功！')
    win.destroy()
```



4. 退货

根据书籍编号，数量，售出月份日期以及顾客编号来进行退货。

```
def refund():#退货
    global win
    win = tk.Tk()
    win.title('图书销售管理')
    win.geometry('900x300')
    win.wm_attributes('-topmost', 1)
    tk.Label(win, text='请填写退货信息： ', bg='cyan',font=('微软雅黑',
20)).place(x=30, y=100)

    global b_id3
    tk.Label(win, text='图书 id： ', font=('宋体', 12)).place(x=50, y=200)
    b_id3 = tk.Entry(win, font=('宋体', 12), width=5)
    b_id3.place(x=120, y=200)

    global amount3
    tk.Label(win, text='数量： ', font=('宋体', 12)).place(x=190, y=200)
    amount3 = tk.Entry(win, font=('宋体', 12), width=5)
    amount3.place(x=260, y=200)

    global month4
    tk.Label(win, text='售出月： ', font=('宋体', 12)).place(x=330, y=200)
    month4 = tk.Entry(win, font=('宋体', 12), width=5)
    month4.place(x=400, y=200)

    global day4
    tk.Label(win, text='售出日： ', font=('宋体', 12)).place(x=470, y=200)
    day4 = tk.Entry(win, font=('宋体', 12), width=5)
    day4.place(x=540, y=200)

    global customer3
    tk.Label(win, text='顾客 id： ', font=('宋体', 12)).place(x=610, y=200)
    customer3 = tk.Entry(win, font=('宋体', 12), width=5)
    customer3.place(x=680, y=200)
```

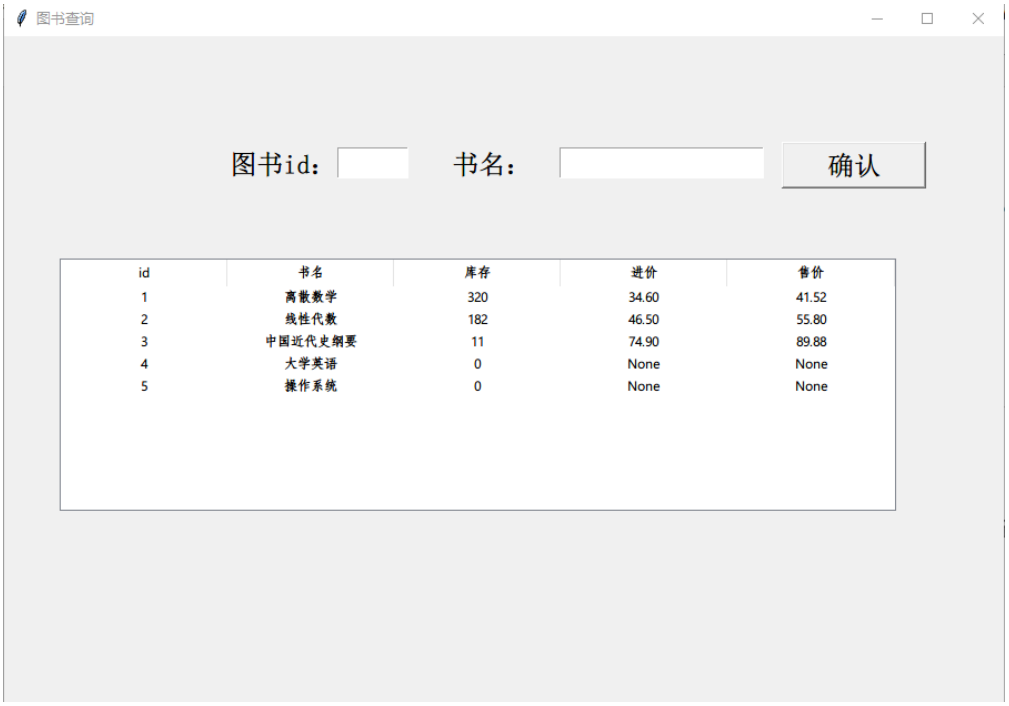
```
    tk.Button(win, text='确认退货', font=('宋体', 12), width=10,
command=lambda:confirm_refund(b_id3,amount3,month4,day4,customer3)).plac
e(x=750, y=195)

def confirm_refund(b_id,num,sell_month,sell_day,c_id):
    db = pymysql.connect(host="localhost", user=user_name,
password=pass_word, database="booksell")
    cursor = db.cursor()
    if (not validNum(int(num.get()))) or (not
validCustomer(int(c_id.get()))) or\
            (not validDate(int(sell_month.get()), int(sell_day.get()))):
        return
        # 输入合法则从数据库获取数据
    try:
        cursor.execute("start transaction")
        # 如果使用书名查询，则先获取 id
        cursor.execute("select id from book where id = " + b_id.get())
        book = cursor.fetchone()
        if book == ():
            msg._show(title='错误！',message='查询错误：查询不到该书数据')
            db.commit()
            return
        # 获取该书的销售记录
        cursor.execute('select * from sold where id = ' + b_id.get() + '
and month=' + sell_month.get() + ' and day=' + sell_day.get()
                       + ' and customer_id=' + c_id.get())
        sell_log = cursor.fetchone()

        if sell_log == ():
            msg._show(title='错误！',message='查询错误：查询不到该日期下该顾
客对该书的购买信息')
            db.commit()
            return
        sell_log = sell_log[3]
        # 退货数超出购买数则报错
        if int(num.get()) > sell_log:
            msg._show(title='错误！',message='数目错误：退货数大于购买数')
            db.commit()
            return
        # 否则可以退货
        else:
            # 修改库存
            #print(0)
```

```
            cursor.execute("select count from book where id = " +
b_id.get())
            rest_num = cursor.fetchone()[0]
            cursor.execute("update book set count = " + str(rest_num +
int(num.get())) + ' where id = ' + b_id.get())
            cursor.execute('select num from sold where id = ' +
b_id.get() + ' and customer_id=' + c_id.get())
            record_num = cursor.fetchone()
            # 如果没有当天的退货记录则创建新的退货记录
            if record_num == None:
                cursor.execute("insert into sold values(" + b_id.get() +
',' +
                    str(-int(num.get())) + ',' + c_id.get() + ")")
            # 否则直接修改退货记录
            else:
                if record_num[0]-int(num.get()) == 0:
                    cursor.execute("delete from sold where id=" +
b_id.get() + " and customer_id=" + c_id.get())
                else:
                    cursor.execute("update sold set num=" +
str(record_num[0]-int(num.get())) + " where id=" + b_id.get() + " and
customer_id=" + c_id.get())
    except:
        msg._show(title='错误！',message='Error:数据库查询或修改出错')
        cursor.execute("rollback")  # 出错时需要回滚
    db.commit()
    msg.showinfo(title='退书成功', message='退书成功！')
    win.destroy()
```

5. 书本库存查询
   直接点击”确定”按钮即可看到所有书籍的库存信息，也可以使用搜索栏来进行高级搜索。

```
def book_search(book_id,title):

    tree.delete(*tree.get_children())
```

```
    db = pymysql.connect(host="localhost", user=user_name,
password=pass_word, database="booksell")
    cursor = db.cursor()
    #print(book_id.get(),title.get())
    try:
        # 默认情况：查看所有图书
        if book_id.get() == '' and title.get() == '':
            cursor.execute('select * from book')
        # 按书的id进行索引
        elif book_id.get() != '' and title.get() == '':
            cursor.execute('select * from book where id = ' +
book_id.get())
        elif book_id.get() == '' and title.get() != '':
            cursor.execute('select * from book where title ="' +
title.get()+'"')
        # 按书名进行索引
        else:
            cursor.execute("select * from book where id
="+book_id.get()+'title = "' + title.get()+'"')
    except:
        msg._show(title='错误！',message='数据库查询出错！')
        return
    results=cursor.fetchall()
    if results:
        l= len(results)
        for i in range(0,l):#查询到的结果依次插入到表格中
            tree.insert('',i,values=(results[i]))
    else :
        tree.insert('', 0,values=('查询不到结果','查询不到结果','查询不到结
果','查询不到结果','查询不到结果'))
    db.close()
```



6. 销售数据查询

直接点击”确定”按钮即可看到所有的销售数据，也可以通过搜索栏来进行高级搜索。

```
def sell_search(book_id,month):

    tree2.delete(*tree2.get_children())
    db = pymysql.connect(host="localhost", user=user_name,
password=pass_word, database="booksell")
    cursor = db.cursor()
    #print(book_id.get(),title.get())
    try:
        # 默认情况：查看所有图书
        if book_id.get() == '' and month.get() == '':
            cursor.execute('select
month,title,sum(num),price_in,price_out,sum(num)*(price_out-price_in)
from book,sold where book.id=sold.id group by month,book.id order by
month,book.id')
        # 按书的id进行索引
        elif book_id.get() != '' and month.get() == '':
            cursor.execute('select
month,title,sum(num),price_in,price_out,sum(num)*(price_out-price_in)
from book,sold where book.id=sold.id and book.id = '+book_id.get()+ '
group by month,book.id order by month,book.id' )
        # 按月份进行索引
        elif book_id.get() == '' and month.get() != '':
            cursor.execute('select
month,title,sum(num),price_in,price_out,num*(price_out-price_in) from
```

```
book,sold where book.id=sold.id and month = '+month.get()+' group by
month,book.id order by month,book.id' )
        else:
            cursor.execute("select
month,title,sum(num),price_in,price_out,num*(price_out-price_in) from
book,sold where book.id=sold.id and book.id= "+book_id.get()+ " and
month= "+month.get()+" group by month,book.id order by month,book.id")
    except:
        msg._show(title='错误！',message='数据库查询出错！')
        return
    results=cursor.fetchall()
    if results:
        l= len(results)
        for i in range(0,l):#查询到的结果依次插入到表格中
            tree2.insert('',i,values=(results[i]))
    else :
        tree2.insert('', 0,values=('查询不到结果','查询不到结果','查询不到结
果','查询不到结果','查询不到结果','查询不到结果'))
    db.close()
```

六. 总结

1. 关系模型: 关系模型的查询、插入、删除、修改操作。
2. 关系的完整性条件: 实体完整性、参照完整性和用户自定义完整性。
3. 事务: 原子性、一致性以及持久性。
4. 触发器: 作为避免非法更新、插入和删除的额外约束。
5. 实体-联系模型: 实体-联系数据模型基于对现实世界的这样一种认识：现实世由一组称作实体的基本对象以及这些对象间的联系构成。实体是现实世界中可区别于其他对的一件“事情”或一个“物体”。