网上书店的设计与实现

设计过程报告

## 需求分析

1. 设计背景：

图书书目庞大，种类繁杂，需要用数据库对图书数据进行有效科学的管理

1. 需求分析：

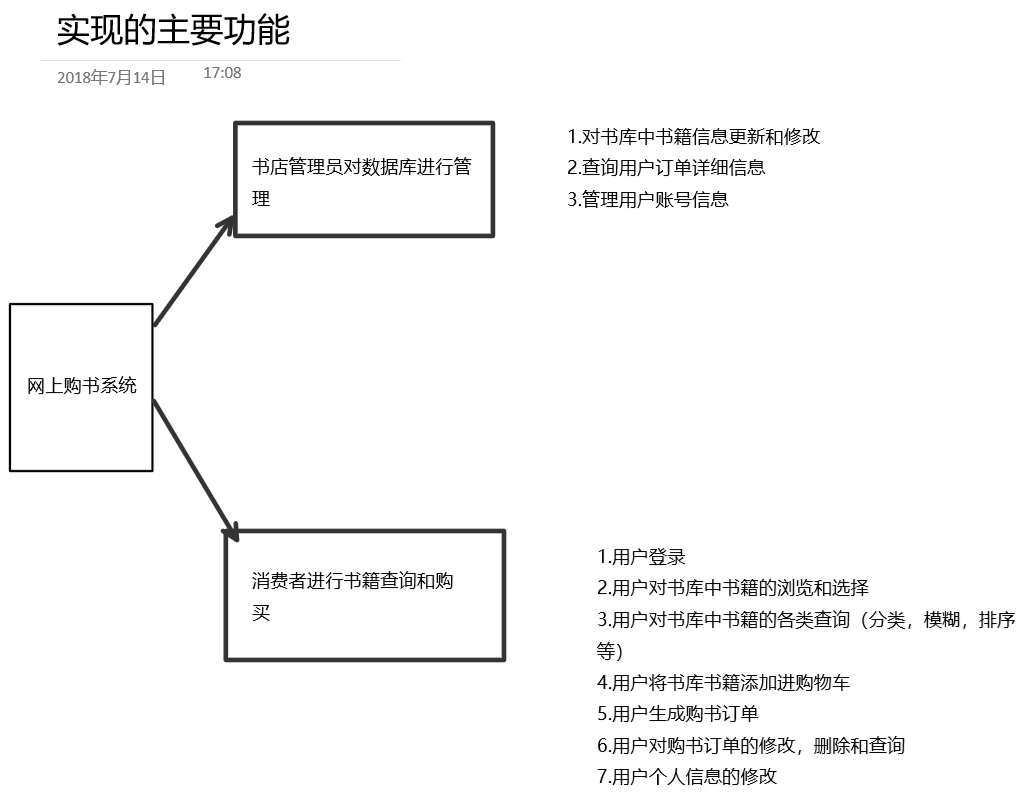
消费者需要通过网上书店实现浏览，查询，分类图书，进而完成添加购物车，支付，生成订单，修改查询用户信息的一系列功能；网上书店的管理员需要对书店内的书目信息进行更新或修改，以及对用户订单进行查询，进而及时给消费者发货。实现供需的关系。

1. 开发环境**：**

开发工具Microsoft SQLServer 2008 R

开发语言SQL

1. **系统主要功能：**



## 概念设计

**1.系统一共有4个实体：**消费者，管理员，书籍，购书订单，购物车

1. 消费者的属性：

自动编号（userId），用户名（userName），登录密码（passWord），备注信息（bak）

1. 书籍的属性：

自动编号（id），书名（bookName），书籍单价（bookPrice），分类（category），评价（mark），作者（writer），库存数量（storeMount）

1. 购书订单的属性：

自动编号（bookOrderId），消费者编号（customerId），书目编号（myBookId），消费者姓名（customerName），收货地址（customerAddress），联系电话（customerTelephone），备注（notic），订购日期（subDate），物流状态（logistics），支付状态（paymentStatus）

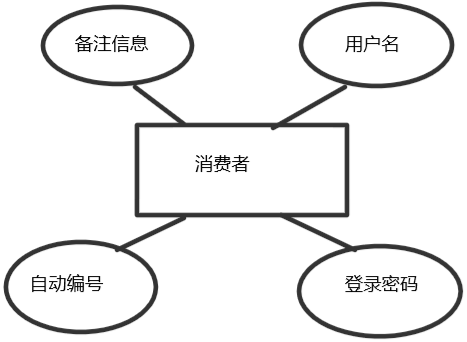
1. 购物车的属性：

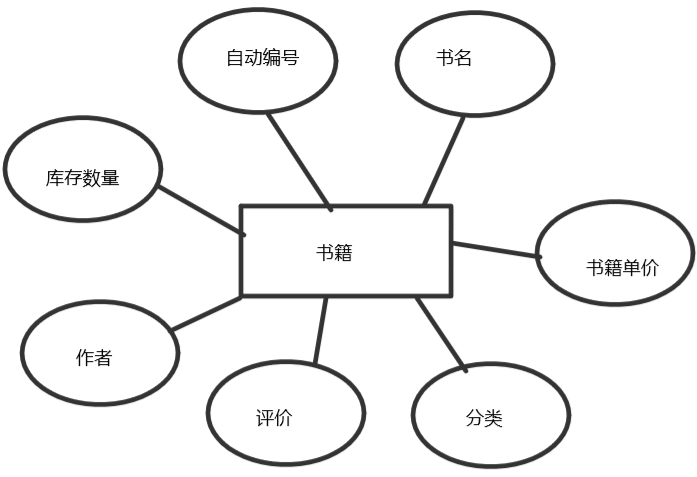
书目编号（bookId），数量（number），总价（totalPrice）

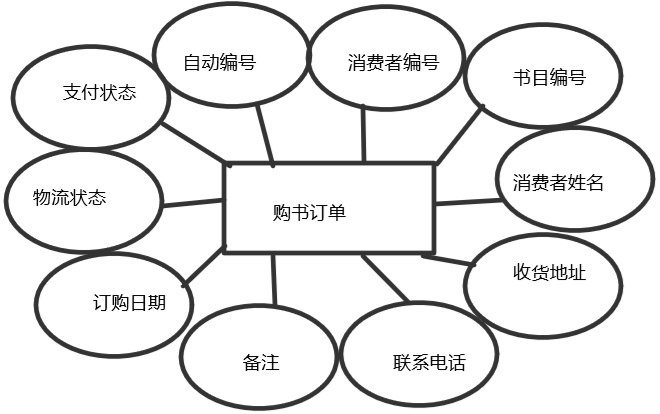
1. 管理员的属性

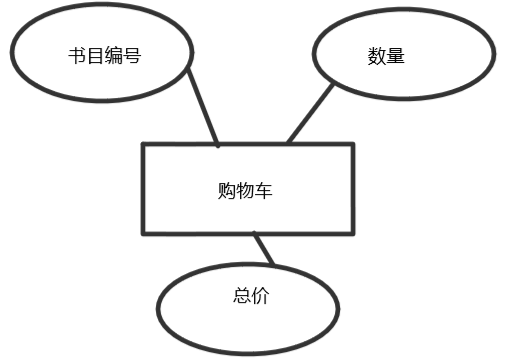
自动编号（adminId），管理员姓名（adminName），管理员密钥（passWord）

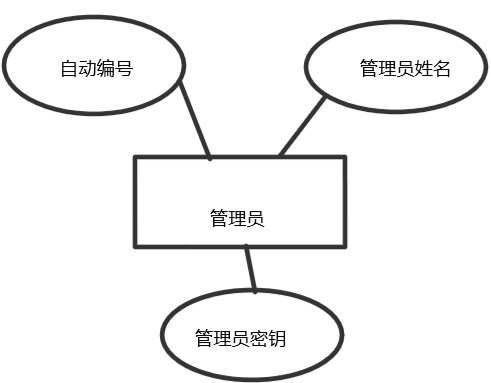
**2.E-R图**

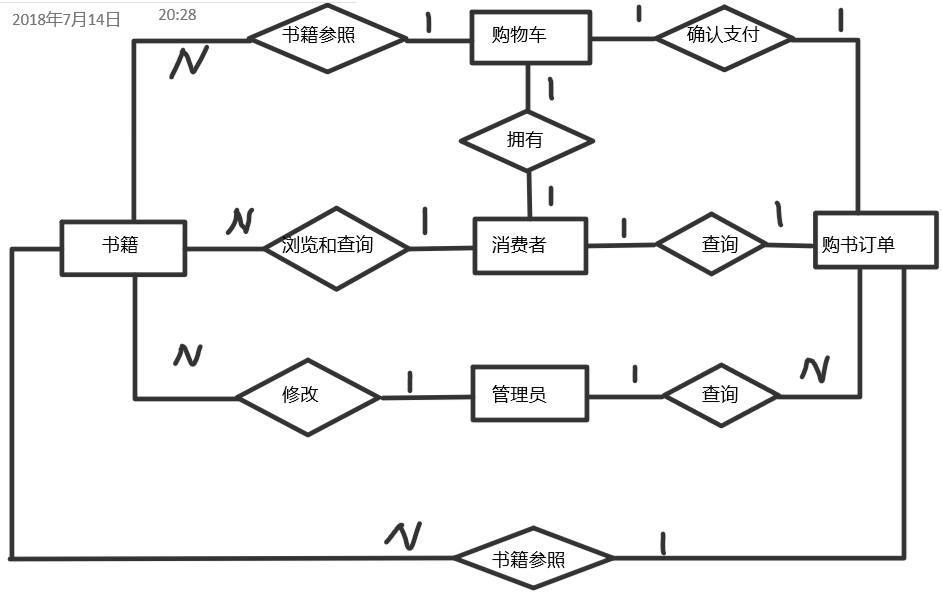












## 逻辑设计

1. **细化表结构表示约束条件**

管理员信息表（AdminInfo）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| adminId | 管理员编号 | Int | 主键（自增字段） |
| adminName | 管理员姓名 | Nvarchar(20) |  |
| passWord | 管理员密钥 | Int |  |

用户信息表（UserInfo）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| userId | 用户编号 | Int | 主键（自增字段） |
| userName | 用户昵称 | Nvarchar(50) |  |
| passWord | 登录密码 | Nvarchar(20) |  |
| bak | 备注 | Nvarchar(20) |  |

图书信息表（BookInfo）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| id | 图书编号 | Int | 主键（自增字段） |
| bookName | 书名 | Nvarchar(50) |  |
| bookPrice | 单价 | money |  |
| category | 分类 | Nvarchar(300) |  |
| mark | 评价 | Nvarchar(300) |  |
| writer | 作者 | Nvarchar(300) |  |
| storeMount | 剩余库存 | Int |  |

购物车（Cart）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| bookId | 图书编号 | Int | 主键（自增字段），引用BookInfo的外键约束  BookInfo.id和Cart.bookId |
| number | 数量 | Int |  |
| totalPrice | 总价 | Money |  |

购书订单（BookOrder）

说明：一个用户Id不对应一个收货人customerName，没有传递依赖，例如在淘宝买东西，一个账户下单时填的收货人可以是自己也可以是别人

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| bookOrderId | 订单编号 | Int | 主键（自增字段） |
| customerId | 用户编号 | Int | 引用UserInfo外键约束  Cart.userId和BookOrder.customerID |
| myBookId | 图书编号 | Int | 引用BookInfo外键约束BookInfo.id和BookOrder.myBookId |
| customerName | 收货人姓名 | Nvarchar(50) |  |
| customerAddress | 收货地址 | Nvarchar(30) |  |
| customerTelephone | 联系方式 | Nvarchar(20) |  |
| notic | 备注 | Nvarchar(200) |  |
| subDate | 订购日期 | date |  |
| logistics | 物流状态 | Nvarchar(20) |  |
| paymentStatus | 支付状态 | Nvarchar(20) |  |

## 物理设计 （以下所有相关的创建代码在压缩包的源代码文档中）

1. **创建数据库BookShopping**
2. **创建表**
3. **管理员信息表AdminInfo**
4. **图书信息表BookInfo**
5. **购书订单表BookOrder**
6. **购物车Cart**
7. **用户信息表UserInfo**
8. **创建约束条件（**四个表的主键约束和Cart，BookOrder中的外键约束**）**

**主键约束**

1. AdminInfo.adminId
2. BookInfo.id
3. Cart.bookId
4. BookOrder. bookOrderId

**外键约束**

（1）Cart.userId和BookOrder.customerID

（2）BookInfo.id和BookOrder.myBookId

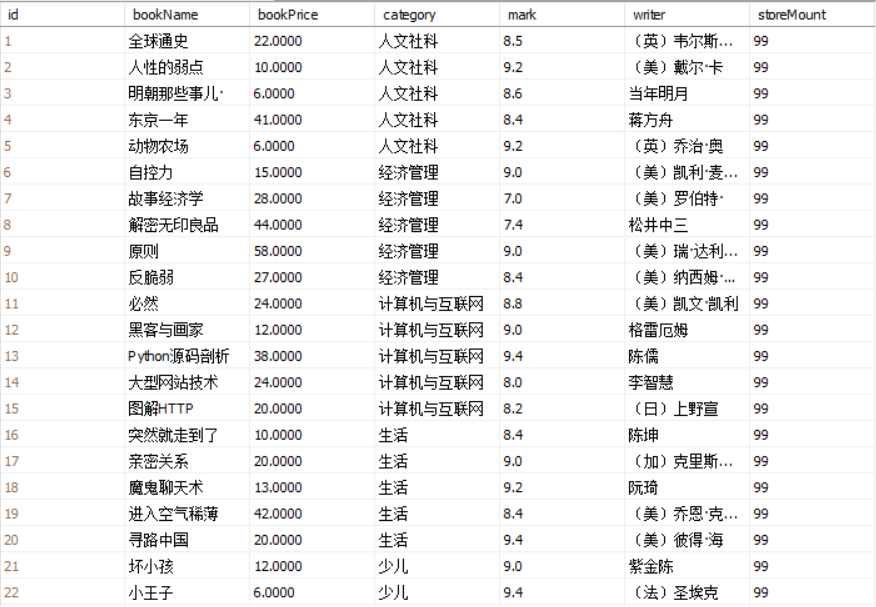
（3）BookInfo.id和Cart.bookId

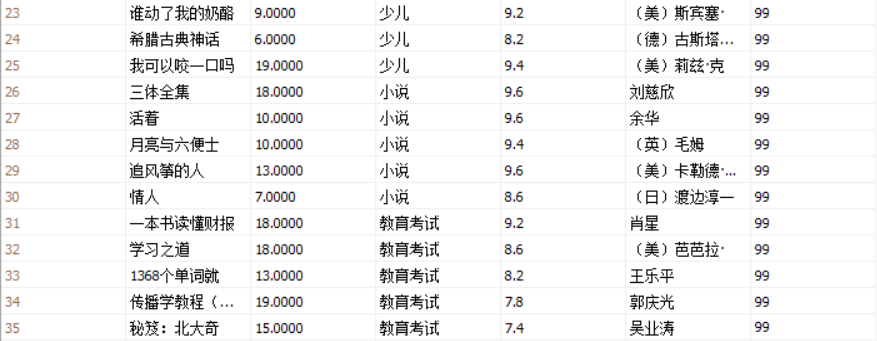
1. **添加数据**

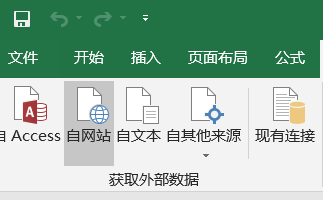
**管理员信息**



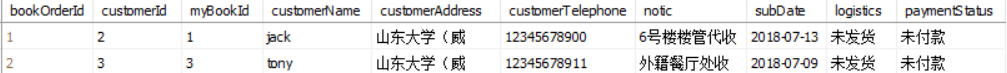
**图书信息**



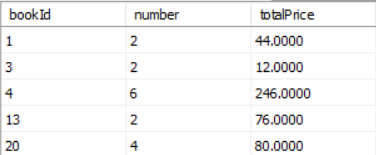


p.s.本来想用execl中的功能去网站上扒数据，但是从亚马逊，当当网，京东网上扒下来的数据排列太别扭了。本人水平有限就干脆直接手打输入了35个数据到表中。

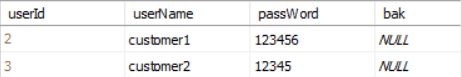
**购书订单信息**



**购物车信息**



**用户信息**



1. **创建触发器（**BookInfo表中的trig1和Cart表中的tri\_1**）**

（1）BookInfo表中的trig1用于检查修改库存数量时库存数量是否小于零，将提示“库存不足！”

（2）Cart表中的tri\_1用于输入图书编号和购买数量，自动计算图书总价，总价=单价\*数量

1. **创建存储过程**

（1）图书模糊查询（bookcheck）

（2）消费者信息查询（customercheck）,

（3）用户订单查询（ordercheck）

（4）模拟预先支付后订单状态和物流状态的改变（pay）

1. **创建用户,设置权限**

登录名账号：shopkeeper 密码：123456789

登录名账号：customer1 密码：123456

登录名账号：customer2 密码：12345

（1）管理员（shopkeeper）拥有所有权限

（2）客户(customer1,customer2)拥有的权限：浏览，选择Boolnfo，UserInfo；浏览，选择，增添，更新，删除BookOrder，Cart；执行存储过程。

## 成果展示

1. **图书分类查询**

具体实现：

BookInfo书籍的类别查询（一共有“人文社科”，“经济管理”，“计算机与互联网”，“生活”，“少儿”，“小说”，“教育考试”这7个大类）

查询分类“人文社科”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='人文社科';

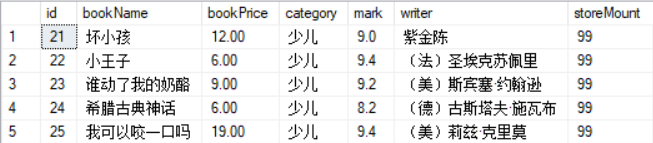


查询分类“少儿”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='少儿';



查询分类“小说”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='小说';



查询分类“生活”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='生活';



查询分类“教育考试”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='教育考试';



查询分类“经济管理”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

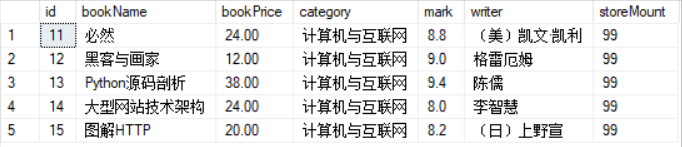
where category ='经济管理';



查询分类“计算机与互联网”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where category ='计算机与互联网'

1. **图书作者查询**

具体实现：

查询作者“当年明月”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where writer ='当年明月';



查询作者“〔日〕渡边淳一”代码及测试结果如下：

USE BookShopping;

Select \* from BookInfo

where writer ='〔日〕渡边淳一';



1. **图书评价的排序查询**

具体实现：

查询BookInfo书籍评价排序代码及测试结果如下：

（1）升序排序：

USE BookShopping;

Select \* from BookInfo

order by mark asc;



（2）降序排序：

USE BookShopping;

Select \* from BookInfo

order by mark desc;



1. **图书价格的排序查询**

具体实现：

查询BookInfo书籍的价格排序查询代码及测试结果如下：

（1）升序排序：

USE BookShopping;

Select \* from BookInfo

order by bookPrice asc;





（2）降序排序：

USE BookShopping;

Select \* from BookInfo

order by bookPrice desc;



1. **存储过程实现的图书模糊查询**

具体实现：

通过创建存储过程（bookcheck）实现图书模糊查询，输入书名关键字（例如'小'）即可查询相应的图书详细信息

执行查询订单代码：

exec bookcheck '小'



1. **存储过程实现的订单查询**

具体实现：

通过创建存储过程（ordercheck）实现订单查询，输入消费者姓名（例如'jack'）即可查询相应的订单详情

执行查询订单代码：

DECLARE

@a date,

@b nvarchar(50),

@c money,

@d money,

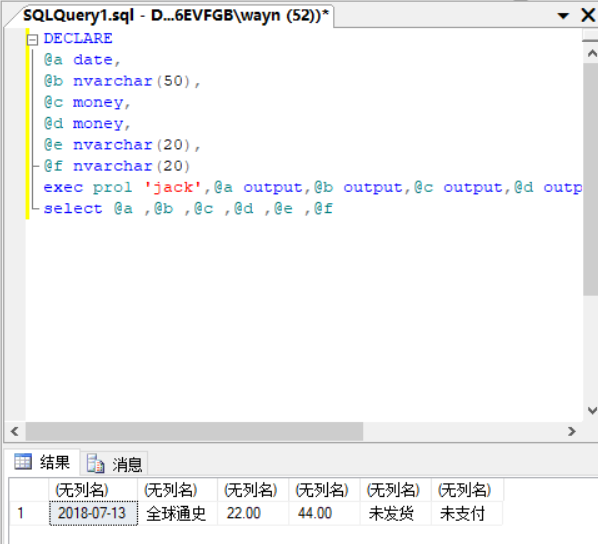
@e nvarchar(20),

@f nvarchar(20)

exec ordercheck 'jack',@a output,@b output,@c output,@d output,@e output,@f output

select @a ,@b ,@c ,@d ,@e ,@f

测试结果：



对应的列名分别是：

订购日期，书名，单价，总价，物流状态，支付状态

1. **存储过程实现的消费者查询**

具体实现：

通过创建存储过程（customercheck）实现消费者信息查询，输入消费者姓名（例如'tony'）即可查询消费者详细信息

执行该消费者查询代码：

DECLARE

@a nvarchar(50),

@b nvarchar(300),

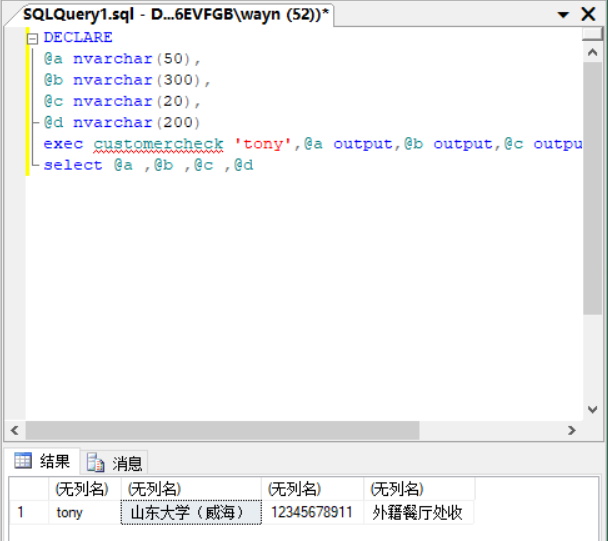
@c nvarchar(20),

@d nvarchar(200)

exec customercheck 'tony',@a output,@b output,@c output,@d output

select @a ,@b ,@c ,@d

测试结果：



对应列名分别为：

消费者姓名，收获地址，联系电话，消费者购物备注

1. **存储过程实现的预付款过程后的订单查询**

具体实现：

预付款的功能，通过创建存储过程pay实现预付款的功能，执行同时执行存储过程pay（输入支付金额和订单号）和ordercheck（输入消费者姓名）可以查询付款后订单支付状态和物流状态的改变

（1）执行存储过程pay前用存储过程ordercheck查询消费者“tony”和“jack”的订单状态详情：

DECLARE

@x date,

@y nvarchar(50),

@z money,

@d money,

@e nvarchar(20),

@f nvarchar(20)

exec ordercheck 'tony',@x output,@y output,@z output,@d output,@e output,@f output

select @x ,@y ,@z ,@d ,@e ,@f

DECLARE

@1 date,

@2 nvarchar(50),

@3 money,

@4 money,

@5 nvarchar(20),

@6 nvarchar(20)

exec ordercheck 'jack',@1 output,@2 output,@3 output,@4 output,@5 output,@6 output

select @1 ,@2 ,@3 ,@4 ,@5 ,@6

查询结果：



（2）当同时执行存储过程pay（输入预支付金额和订单号，例如金额=45单号=2，预支付金额45元大于等于订单号为2的总价12元）和ordercheck的查询代码查询消费者“tony”和“jack”的订单状态详情：

DECLARE

@a nvarchar(20),

@b nvarchar(20),

@c money

exec pay 45,2,@a output,@b output,@c output

DECLARE

@x date,

@y nvarchar(50),

@z money,

@d money,

@e nvarchar(20),

@f nvarchar(20)

exec ordercheck 'tony',@x output,@y output,@z output,@d output,@e output,@f output

select @x ,@y ,@z ,@d ,@e ,@f

DECLARE

@1 date,

@2 nvarchar(50),

@3 money,

@4 money,

@5 nvarchar(20),

@6 nvarchar(20)

exec ordercheck 'jack',@1 output,@2 output,@3 output,@4 output,@5 output,@6 output

select @1 ,@2 ,@3 ,@4 ,@5 ,@6

测试结果：

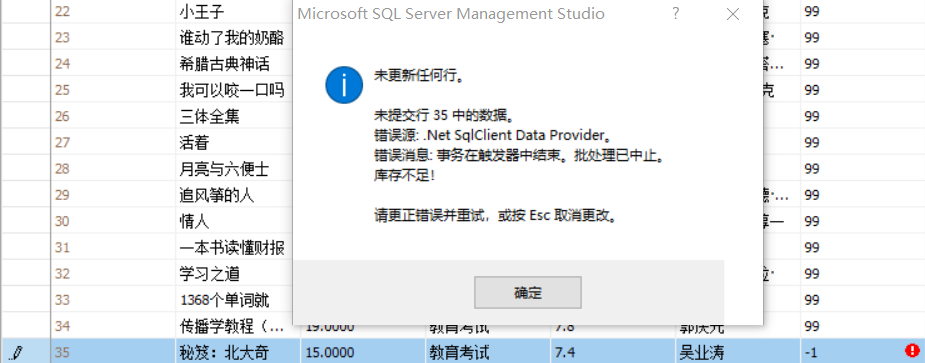


注：订单号2对应的是消费者“tony”的订单，订单号1对应的是消费者“jack”的订单

1. **触发器trig1实现BoookInfo表中当图书库存数量（storeMount）小于0时，触发提示信息“库存不足！”**

具体实现：

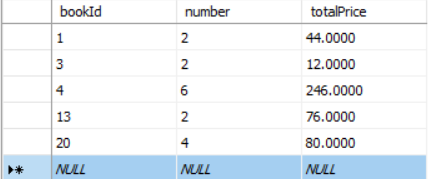
当管理员修改图书库存数量时，库存数量自动检测数量是否小于0，小于零时提示“库存不足！”



1. **触发器tri\_1实现Cart表中totalPrice列的自动求和，得出购书总价的功能**

具体实现：

当消费者在购物车中输入图书编号和购书数量时，购书的总价自动生成。例如“bookId=1的书是《全球通史》，单价为22，订购的数量是2本，自动生成总价等于44”



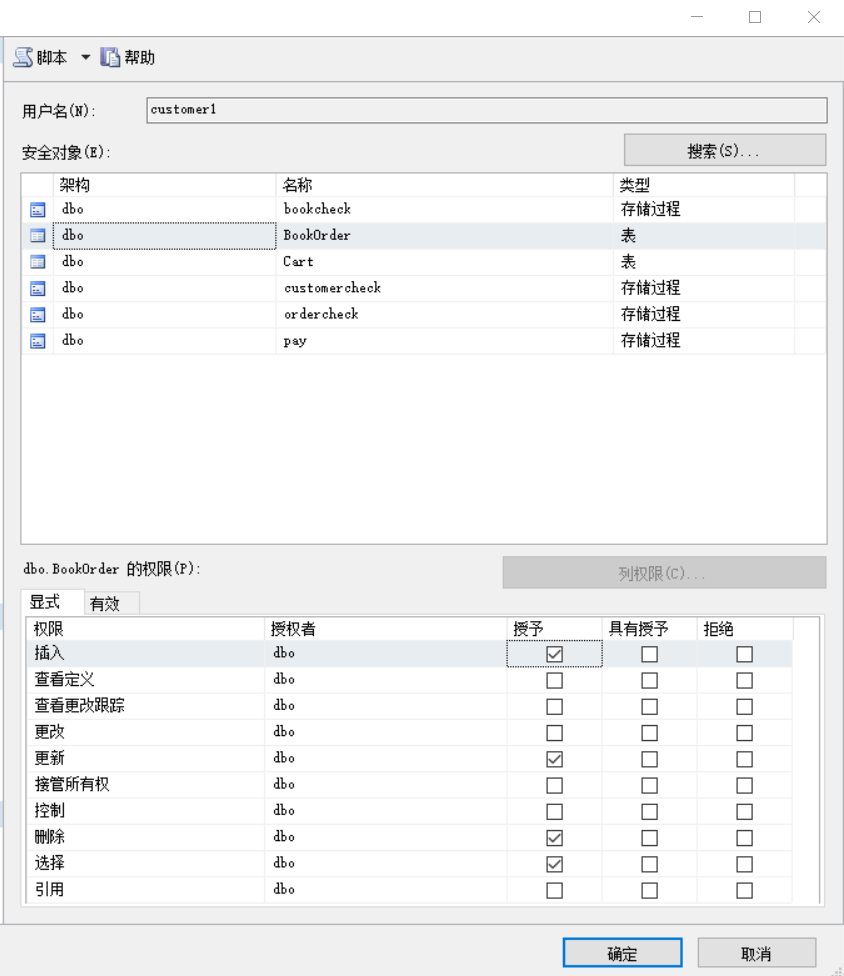
1. **客户 “customer1”和“customer2”有修改和删除BookOrder表中数据的权限，实现订单的修改和删除功能**

具体实现：

创建“customer1”用户，然后用grant语句授予customer1修改和删除BookOrder表中数据的权限

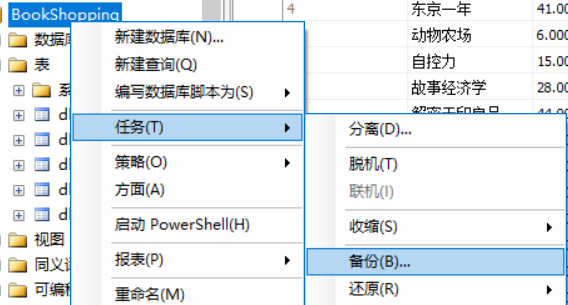
创建“customer2”用户，然后用grant语句授予customer2修改和删除BookOrder表中数据的权限

测试结果：

****

1. **数据库BookShopping的备份和还原**

具体实现：

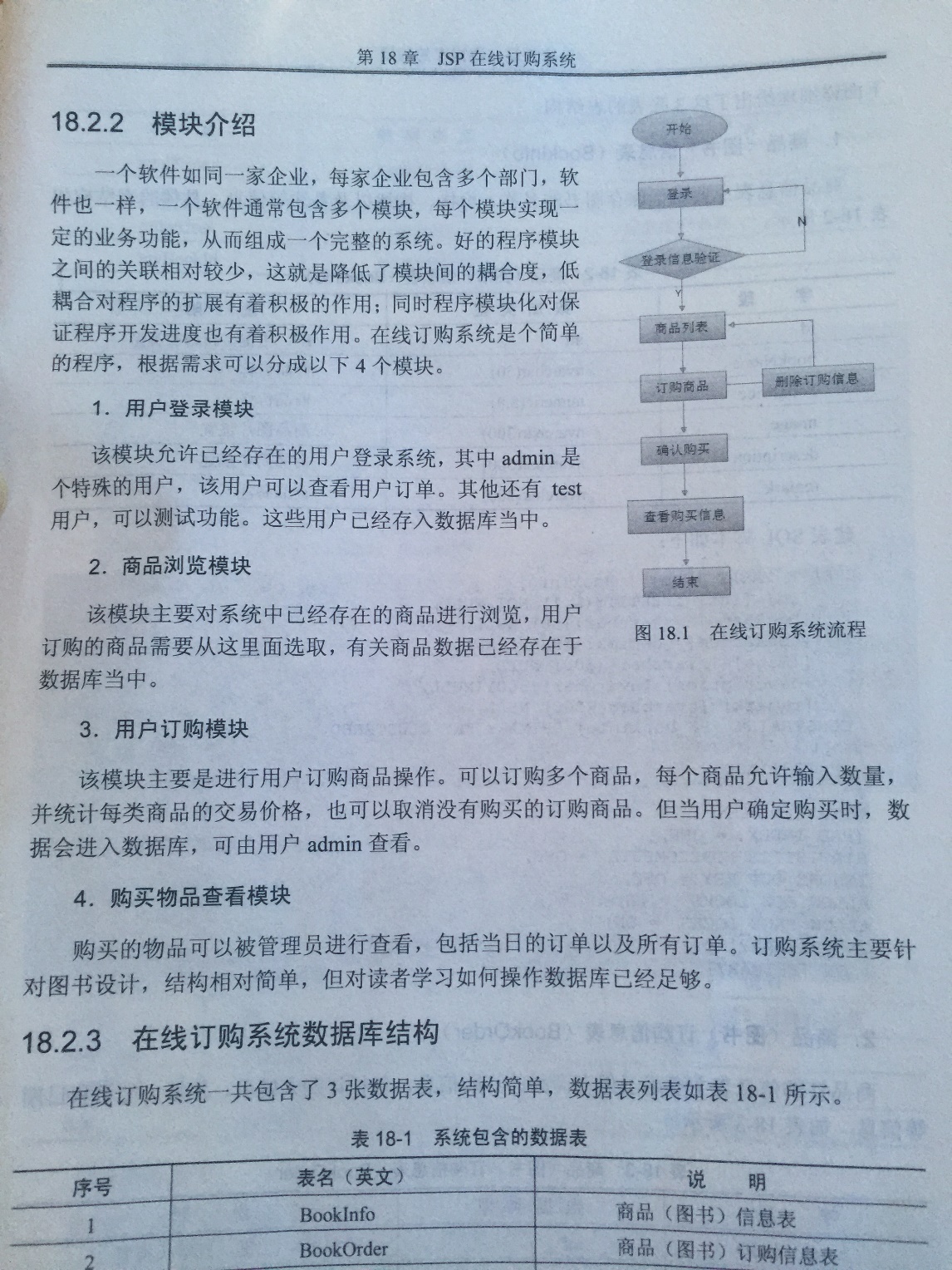


测试结果：

备份文件见压缩包中附件（网上书店备）

## 数据库外部连接部分

1. **JSP在线订购系统的模块介绍**



1. **利用JDBC Driver连接数据库**

package app.init;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class DBConnection {

private static String drivers="com.mircrosoft.sqlserver.jdbc.SQLServerDriver";

private static String url="jbdc:sqlserver://localhost:1433;DatabaseName=BookShopping";

private static String user="sa";

private static String password="1234";

public static Connection GetConnection(){

Connection conn=null;

try{

Class.forName(drivers).newInstance();

}catch(InstantiationException e){

e.printStackTrace();}

catch(IllegalAccessException e){

e.printStackTrace();}

catch(ClassNotFoundException e){

e.printStackTrace();}

try{conn=DriverManager.getConnection(url, user, password);

}

catch(SQLException e){e.printStackTrace();

}

return conn;

}

public static void close (Connection conn){

try{

if(conn != null&& !conn.isClosed())

conn.close();

}catch (SQLException e){

e.printStackTrace();

}

}

}

1. **订单查询BookOrderDao.java调用连接数据库代码**

public List<bookOrder> getBookOrderListAll() throws SQLException{

List<BookOrder> List =new ArrayList<BookOrder>();

Connection con =DBConnection.GetConnection();

PreparedStatement pstmt= null;

String sql= "SELECT \* FROM BOOKORDER WHERE ORDERUSER IS NOT NULL"+"ORDER BY SUBDATE";

BookOrder bookorder = null;

try{

pstmt=con.prepareStatement(sql);

ResultSet rs = pstmt.executeQuery();

while (rs.next()){

bookorder = new BoookOrder();

bookorder.setCustomerName(rs.getString(2));

bookorder.setCustomerAddress(rs.getString(3));

bookorder.setCustomerTelephone(rs.getString(4));

bookorder.setNotic(rs.getString(5));

bookorder.setTotalPrice(rs.getString(6));

bookorder.setSubdate(String.valueOf(rs.getDate(7)));

bookorder.setSessuser(rs.getString(8));

bookorder.setMyBookId(rs.getString(9));

list.add(bookorder);

}

rs.close();

}finally{

pstmt.close();

con.close();

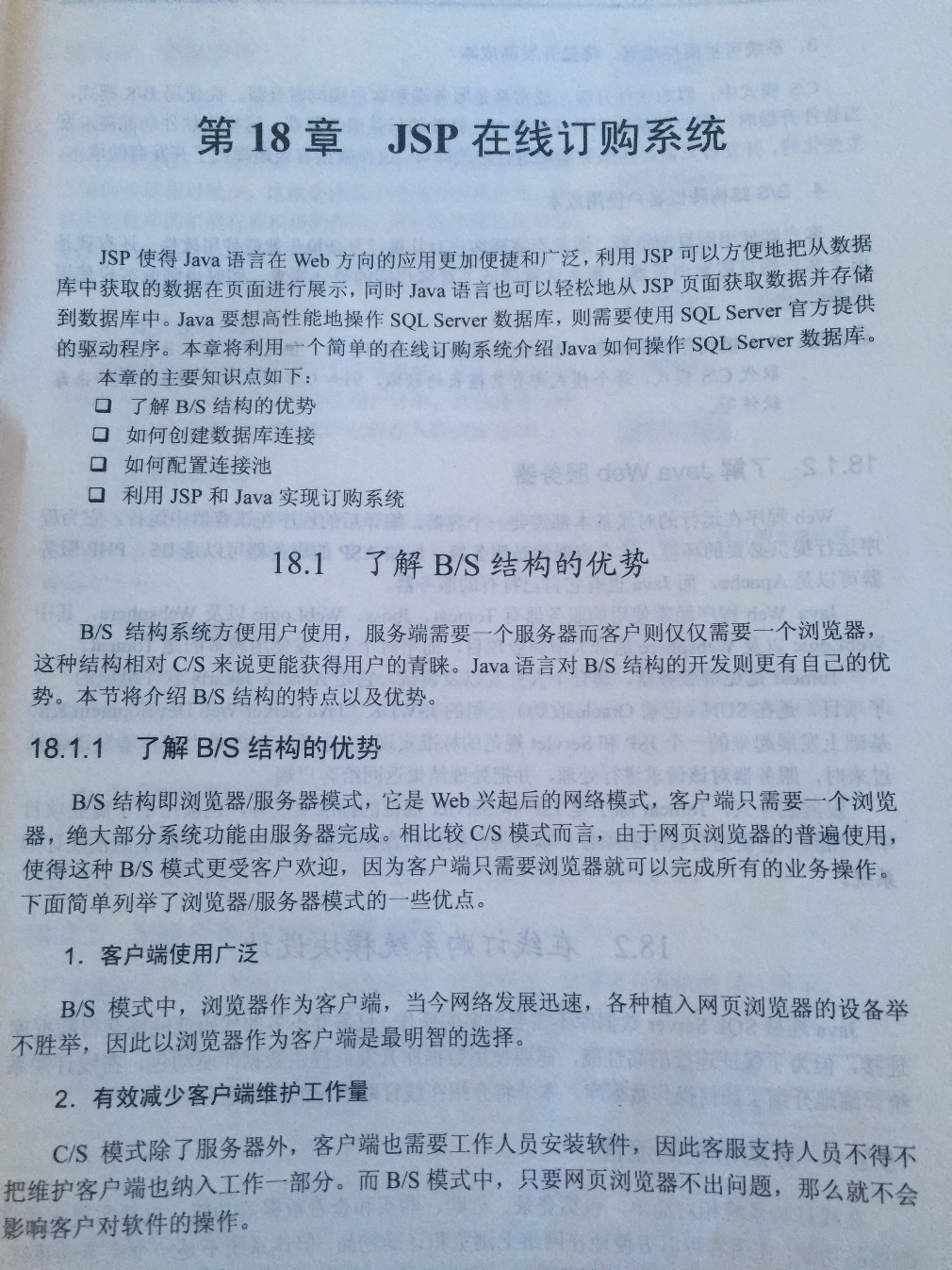
}

return list;

}

1. **参考书籍上的实例**

《SQL Server 入门很简单》第18章



参考图书馆书籍《SQL Server入门很简单》——秦婧编

## 实验小结和心得体会

这次实验不仅让我熟悉了SQL的语句，熟悉了SQL SERVER的使用，还让我加深了对三范式的理解和对数据库设计的流程更的体会。设计一个完整的数据库系统，不光是创建几个表，几个存储过程，几个触发器的问题，而关键在于数据之间建立的约束关系，相互之间的联系。这样个体才能以特定的关系联系在一起，才能实现所谓的功能，才能被称作是系统。这次实验带给了我很多启发，其中设计流程中需求分析、概念设计、逻辑设计、物理设计这几个步骤给了我解决问题的思路。

有一点小遗憾的是，本来我想参考借阅书籍上的实例，照葫芦画瓢用java语言做一个JSP在线订购系统，但是期间出现了各种问题，我捣鼓了2天JDK，eclipse，JDBC驱动，Tomcat等软件，发现我学的东西太少，实在弄不明白怎么操作。虽然书上也有详细的说明和源代码的解释，但是我理解不了，最后还是没能做出来。所以说有点遗憾吧，感觉自己还是水平太低。

小学期选这门课对我而言即使挑战也是机遇，之前我也没有修过数据库技术这门课，对数据库也没有什么了解，基本上是基础为0，从一开始头两天一直在借阅图书馆中有关数据库的教程书籍，到慢慢开始熟悉数据库SQL SERVER软件，做网上书店的数据库的设计，这几天确实不太好受。但是这对我而言，完成作业的压力能督促自己在小学期不要整天无所事事，真正去做一些有意义的事情，多学一些技术和知识，何尝不是一件好事。有机会我一定要补修数据库技术这门课。