实验报告

刘妍　17307130328

一、实验题目

图书销售管理系统的设计与实现。

二、开发环境

操作系统：Windows10 家庭普通版

数据库管理软件：Microsoft SQL Server 2008 • SQL Server Management Studio

编程语言：javascript

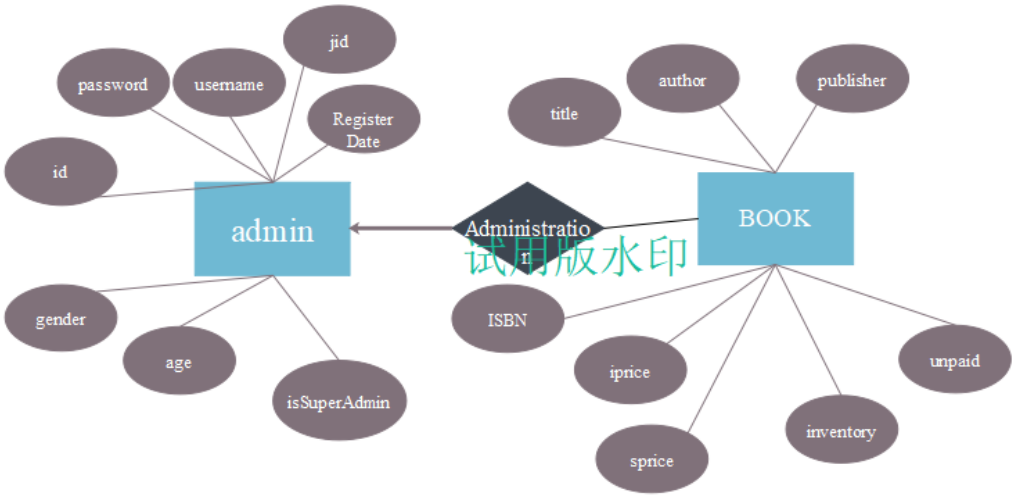
Web 开发环境：node.js+express

三、数据库设计

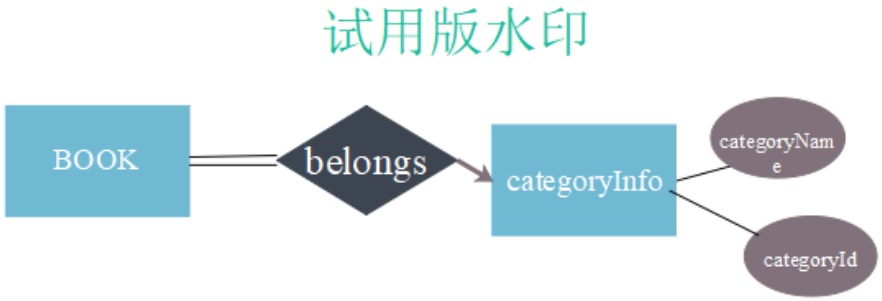
3.1 概念设计

根据图书销售管理系统分析和设计，确定数据库的实体有管理员、书、书籍种类信息、顾客、购物车、想要的书、订单、销售表、进货表，各实体之间的联系如下图所示。（完整 E-R 图请见附录 1）

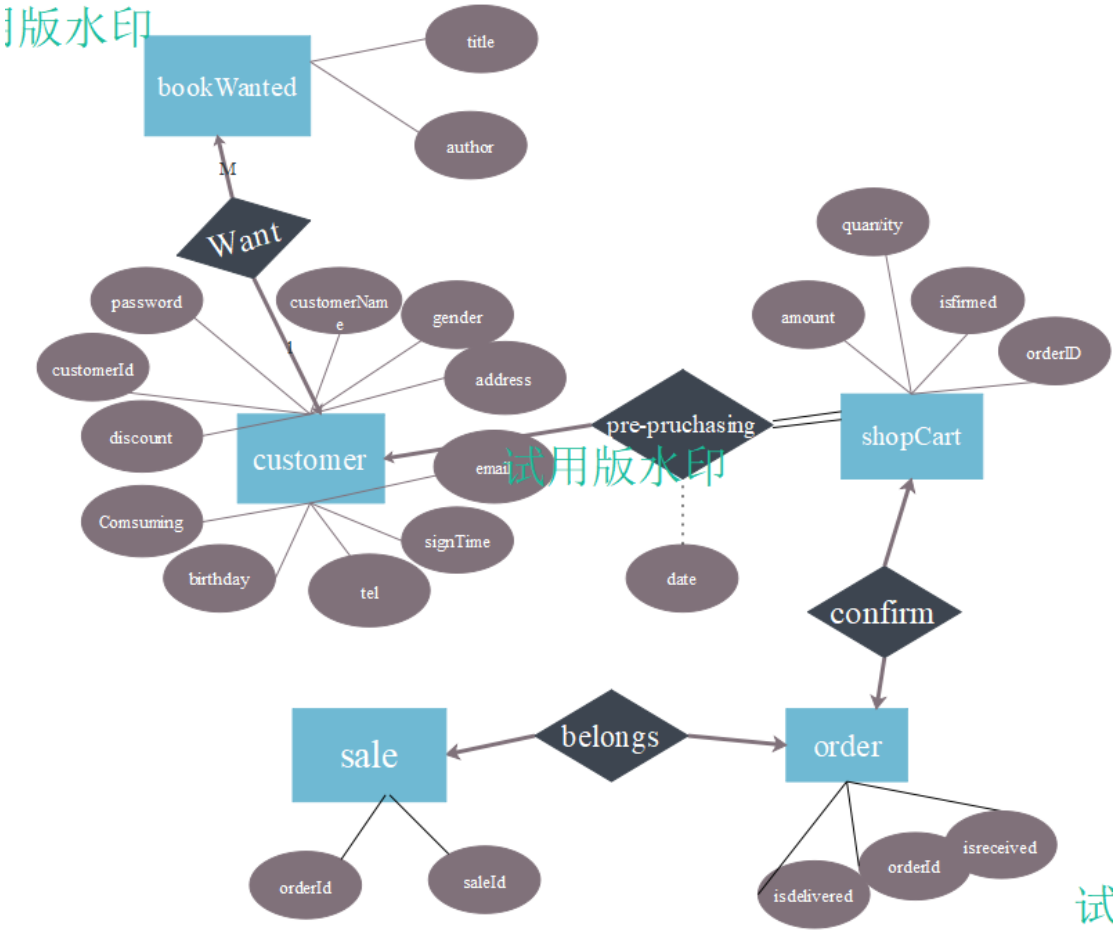
3.1.1 管理员与管理书籍



3.1.2.书籍与书籍类型信息

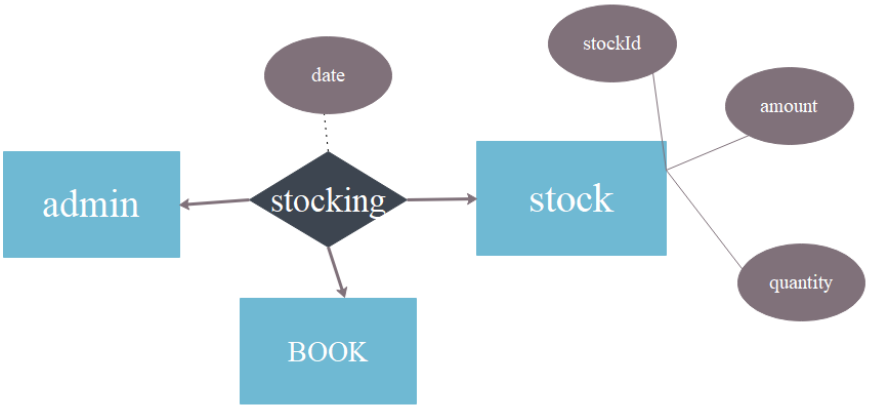


3.1.3 顾客、想看的书、购物车、订单和销售表



3.1.4 管理员与订单

3.1.5 管理员、书籍与进货表



3.2 逻辑设计

经过 E-R 向关系模型转化并模式合并之后，得到如下的数据库表设计，如表 1-8 所示。

表 1 管理员

字段名称	数据类型	长度	备注
职工号	Varchar	20	主键
密码	varchar	50	not null
真实姓名	varchar	20	not null
管理员 ID	varchar	20	not null
性别	varchar	20	not null
年龄	Integer	默认	not null
是否为超级管理员	varchar	20	Default ‘no’
注册日	Date	默认	GETDATE()

表 2 书

字段名称	数据类型	长度	备注
书号	varchar	20	主键
作者	varchar	20	not null
出版社	varchar	20	not null
书籍类型	varchar	20	外键
书名	varchar	20	not null
进价	Numeric	(6,2)	Not null
定价	numeric	(6,2)	
库存	Integer	默认	Not null
待支付	Integer	默认	Default 0

表 3 顾客

字段名	数据类型	长度	备注
Id	Varchar	20	主键
密码	varchar	50	not null
性别	varchar	20	not null
电话	varchar	20	not null
地址	varchar	20	not null
邮件	varchar	20	Not null

生日	Date	默认	
注册日期	Date	默认	GETDATE()
累积消费	Numeric	（6,2）	Default 0
折扣	Numeric	（3,2）	Default 0.9

表 4 书籍类型

字段名	数据类型	长度	备注
类型 id	varchar	20	主键
类型名称	varchar	20	not null

表 5 想要的书

字段名	数据类型	长度	备注
书名	varchar	20	主键
作者	varchar	20	主键

表 6 购物车

字段名	数据类型	长度	备注
订单号	Integer	默认	主键
客户 ID	varchar	20	外键
已确认？	varchar	20	Default ‘no’
时间	Date	默认	GETDATE（）
书号	varchar	20	外键
数量	Integer	默认	not null
总额	Numeric	（6,2）	not null

表 7 订单

字段名	数据类型	长度	备注
订单号	Integer	默认	主键、外键
已发货？	varchar	20	Default ‘no’
已接收？	Varchar	20	Default ‘no’
职工号	Varchar	20	外键

表 8　销售记录

字段名	数据类型	长度	备注
销售编号	Integer	默认	主键
订单编号	Integer	默认	外键
时间	Date	默认	GETDATE（）

表 9　进货记录

字段名	类型	长度	备注
进货编号	Integer	默认	主键
职工号	Varchar	20	外键
书号	varchar	20	外键
数目	Integer	默认	not null
总额	Numeric	（6,2）	not null
时间	Date	默认	GETDATE（）

3.3 视图设计

1）视图名称：alljoin

视图功能：因为设计时防止数据冗余，很多信息只存在一个表中，但在很多查询时都需要用到，所以设计一个把订单、购物车、顾客、书籍全都 natural join 后得到的视图，便于其他查询。

```
create view alljoin
as select
O.orderId,C.customerId,B.title,S.quantity,B.sprice,C.discount,S.amount,
B.isbn,C.address,C.tel,
O.isdelivered,S.isfirmed,O.isreceived,C.accCon
from _order O , shopCart S,customer C,book B
where O.orderId=S.orderId and S.customerId=C.customerId and
S.isbn=B.isbn;
```

2）视图名称：queryShopCart

视图功能：顾客查看购物车

```
create view queryShopCart
as select
S.orderId,C.customerId,B.title,S.quantity,B.sprice,C.discount,S.amount
from shopCart S,book B,customer C
where S.isbn=B.isbn and C.customerId = S.customerId and
S.isfirmed='no';
```

3）视图名称： queryOrder

视图功能：顾客查看订单

```
create view queryOrder
as select orderId,customerId,title,quantity,sprice,discount*sprice as
newprice,amount,isreceived,isdelivered
from alljoin;
```

4）视图名称：queryNotDelivered

视图功能：管理员查看待发货的订单

```
create view queryNotDelivered
as select orderId,isbn,title,quantity,customerId,[address],tel
from alljoin
where isdelivered='no';
```

5）视图名称：queryUnpaid

视图功能：管理员查看还未付款的书

```
create view queryUnpaid
as select isbn,title,author,iprice,inventory,unpaid
from book
where unpaid>0
```

3.4 触发器设计（因代码较长，不附于实验报告中，请见于 sql 文件）

1）名称：afterOrder

功能：在顾客下单之后在_order 中插入一条数据

2）名称：afterReceive

功能：在顾客确认确认收货之后对 sale 表插入一条记录并更新顾客的累积消费

3）名称：changeDiscount

功能：当顾客累积消费增加后更改折扣

4）名称：afterSale

功能：当 sale 表插入一条记录后，自动更新库存数

5）名称：afterStock

功能：当 stock 表插入一条数据之后，自动更新库存数和未付款数

3.5 约束规则

3.5.1 外键约束：

1）建立书表和种类表之间的参照关系：

```
foreign key(categoryId) references categoryInfo(categoryId) on delete cascade
```

2）建立购物车表和书表、顾客表的参照关系

```
FOREIGN KEY(customerId) REFERENCES customer(customerId),
```

3）建立订单表和管理员的参照关系

```
FOREIGN KEY(jid)REFERENCES _admin(jid),
```

4）建立进货表和管理员的参照关系

```
FOREIGN KEY(jid)REFERENCES _admin(jid)
```

3.5.2**default 约束：**如 3.2 中表设计所示

3.5.3**check 约束**

1）设置性别为女或者男 `CHECK(gender='male'or gender='female')`

2）限制顾客表折扣范围 `check(discount in(0.90,0.80,0.75))`

3）　设置每条订单书籍数目≥1 `check(quantity>=1)`

四、系统设计

4.1 用户管理

4.1.1 用户登录：

前端：通过 POST 方法进行表单提交，向服务器发送管理员填写的 id 和 password。表单提交代码如下：

```
<form action="admin" method="POST">
      <h1>管理员登录</h1>
      <div class="form-group">
          <label for="">账号</label>
          <input type="text" id="id" name="id" class="form-control"  >
      </div>
      <div class="form-group">
          <label for="">密码</label>
          <input type="text" id="pwd" name="pwd" class="form-
control"onfocus="this.type='password'"  >
      </div>
      <input class="button-control btn-block" id="login" type="submit"
value="登陆">
  </form>
```

- <form> 标记中提交方法指定为 POST
- action 的值被指定为用于接收表单数据的 URL
- <input> 中的 name 属性，是服务器能够识别的字段。服务器通过 name 索引获取值

后端：

1）通过 mssql 模块连接数据库，用 select 语句查询接收到的 id 对应的用户信息

2）对接收到的密码进行 md5 加密（md5 函数来自引入地‘md5’模块），并和从数据库获取的密码对比

3）若验证通过，则根据此 id 的‘isSuperAdmin’字段判断是否为超级管理员，并跳转到相应页面；若验证不通过，则当前页面刷新。

具体代码如下：

```
var sql=require('mssql');//要操作数据库必须先引入 mssql 模块

//封装连接数据库所需参数
var db =
{
    user:'ly',
    password:'111',
    server:'localhost\\SQLEXPRESS',
```

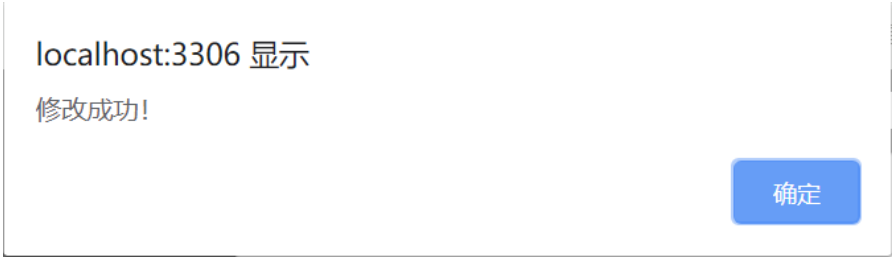
```
    database:'bookSystem',
    port:'1433'
};

router.route('/admin').post(function(request, response) {
  sql.connect(db, function (err) {//连接数据库的函数
    var re = new sql.Request();//申请一个 sql 语句
    re.input('id', sql.NVarChar(20), request.body.id);//向语句中注入参数
    //调用 sql 语句
    re.query("select * from _admin where id=@id", function (err, result)
    {
      sql.close();//断开连接，此时从数据库获取的数据已经储存在了 result 中
      var p=md5(request.body.pwd);//对用户输入的密码进行 md5 加密，以便对照
    if(result.recordset[0]['password']===p){//验证通过
        if(!(result.recordset[0]['isSuperAdmin'])) //如果是超级管理员
            response.redirect('/adminPage');//redirect 实现页面跳转
        else
          response.redirect('/superPage');
      }
      response.end(); //结束当前页面的执行
    });
});
});
```

（超级管理员界面：比普通管理员界面多了“其他用户”功能）

4.1.2 编辑资料：

前端：管理员点击修改之后，同样通过提交表单将数据发送到后端

后端：接收数据，连接数据库，通过 update 语句更新用户资料，更新成功后会提示：

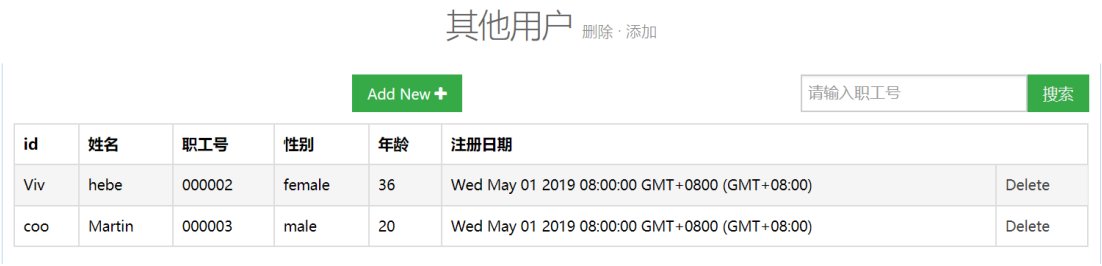


4.1.3：超级管理员查看其它用户资料、创建新用户

当点击左侧“功能栏”的“其他用户”后，路由会变为已设置的值，如此处为‘/otherAdmin’，相当于发起 get 请求，后端定义该路由 get 时的方法：

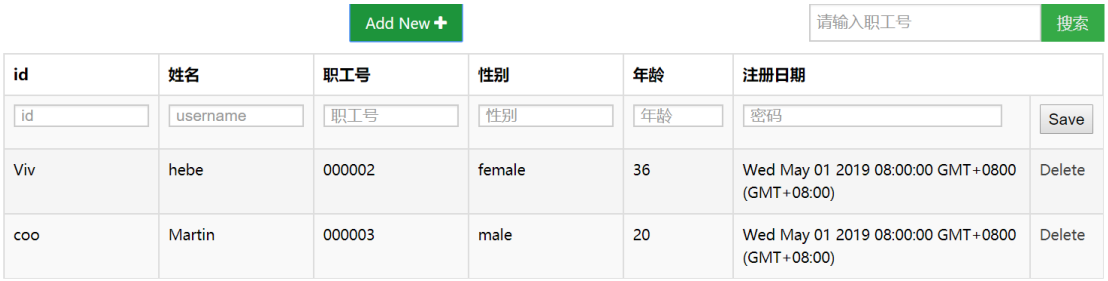
```
router.route('/otherAdmin').get(function(request, response) {
   sql.connect(db, function (err) {//连接数据库
        var re = new sql.Request();
        re.query("select * from _admin where isSuperAdmin='False'
",(err,result) =>{
       sql.close();
// response.render('前端模板文件’,{从数据库获取的数据}) 实现页面渲染
       response.render('otherAdmin',{data:result.recordset,id:adminId});
      });
   });
 });
```

所有渲染前端原理都相同：发送 get/post 请求+后端访问数据库+render 方法渲染



1） 搜索用户：超级管理员可在右上角的输入框中输入职工号具体搜索。实现原理：ajax 发送请求+后端 select 查询语句获取数据渲染到该页面

2） 添加用户：超级管理员可点击“Add New+”按钮添加用户，在表格第一行插入输入框，可输入新用户信息，再点击右侧“Save”保存。实现原理：监听“Add New+”的 click 事件，通过 insertRow()和 insertCell()插入单元格；监听“Save”的 click 事件，ajax 发送请求+后端 insert 语句插入_admin 表



3）删除用户：点击“Delete”可实现删除用户。实现原理：ajax 发送请求+后端 delete 语句。

4.2 书籍查询、图书信息修改

4.2.1 查询书籍：在右上角输入框中输入 isbn/书名/作者/出版社/种类的任意字段，可实现近似查询。实现原理：ajax 发送请求，后端通过 sql 中的 like 通配符实现模糊查询：`"select * from book B,categoryInfo C where B.categoryId =C.categoryId and( title like '%'+@key+'%' or author like '%'+@key+'%' or publisher like '%'+@key+'%'or C.categoryName like '%'+@key+'%') "`

4.2.2：添加书籍：和“添加用户”类似，但逻辑上添加书籍时没有库存且售价未定（当书籍到货之后才能定售价），所以库存和售价都无法修改。

4.2.3：编辑书籍：可修改书名、作者等信息，无法修改 isbn 和库存（因为已经设置了触发器会自动更新库存）。在此处修改未付款的值，实现书籍进货；当书籍到货之后，可以在此处添加售价。实现原理：ajax 发送请求+后端 update 语句更新 book 表

4.3 图书进货

页面显示读者在“想看的书”表中登记的书名和作者，管理员可根据此表，在右上角的输入框中输入书名，点击“图书进货”，在书库中搜索该书籍。若已存在该书籍但库存为 0，则可直接 edit 修改未付款数实现进货；若不存在，则需要先点击“Add New+”添加书籍。

实现原理：通过提交表单，点击“图书进货”时，向“维护书库”的页面的路由 POST 书名，之后与查询书籍类似，不赘述。

4.4 进货付款/图书退款

4.4.1 付款：页面初始会列出所有未付款的书籍，点击付款按钮后提示“付款成功”，点击“退货”提示退货成功

实现原理：

1） 页面初始化：在该路由的 get 方法中，连接数据库，`select * from queryUnpaid` 获取视图 queryUnpaid 中所有数据，该视图定义为从书籍中 select 所有 unpaid>0 的书籍；获取数据后通过 render 渲染到前端。

2） 付款：监听“付款”的 click 事件，点击时获取该行的 isbn、未付款数、进价，通过 ajax 发送到后端；后端接收数据，向进货记录表中插入一条数据：

`insert into stock(jid,isbn,quantity,amount) values (@jid,@isbn,@quantity,@amount)`

定义了 afterStock 触发器，当 stock 表插入一条数据时，会自动更新书的库存数和未付款数

```
create trigger afterStock
on Stock
after insert AS
BEGIN
update book
set inventory=inventory+(select quantity from inserted),unpaid=0
where isbn=(select isbn from inserted)
end
```

（afterStock 触发器）

3）退货：监听“退货”的 click 事件，点击时获取该行的 isbn，通过 ajax 发送到后端；后端接收数据，“update book set unpaid=0” where isbn=@isbn”更新未付款数

4.5 图书销售

点击“图书发货”，页面显示待发货的订单，点击“发货”提示“已发货”。

实现原理：

1） 页面初始：select * from queryNotDelivered 从 queryNotDelivered 视图中获取数据渲染到前端，该视图定义为 select 表_order 表 isdelivered 字段为‘no’的数据

2） 发货：监听“发货”的 click 事件，点击时获取该行的单号，通过 ajax 发送到后端；后端接收数据，“update _order set isdelivered='yes',jid=@jid where orderId=@orderId”更新 isdelivered 字段为‘yes’，并且补充处理该订单的职工号。

4.6 财务管理：

4.6.1 购买书籍：如上所述，付款时后端向进货记录表中插入一条数据

4.6.2 销售书籍：在顾客页面，当顾客点击“确认收货”之后，后端会 update _order set isreceived='yes' where orderId=@orderId（更新_order 表的 isreceived 字段为'yes'），此时 afterReceive 触发器会对销售记录表插入一条数据。

```
create trigger afterReceive
on _order
after update
as
if update(isreceived)
  begin
    declare @OI varchar(20);
    select @OI=orderId from inserted;
    insert into sale(orderId) values(@OI);
```

4.7 查看账单

点击“收支状况”，选择起始时间和结束时间，选择收入或支出，点击查询，下方会显示搜索结果。

单号	总额	职工号	时间
100073	￥ 31.5	000001	Sat Apr 27 2019 08:00:00 GMT+0800 (GMT+08:00)

实现原理：

1） 时间选择器：jquery UI 有一个时间选择器的部件，引入 jquery-ui.js,实例化一个 datepicker（）即可

2） 查询：表单提交到后端，后端获取时间范围和选中的单选框的值，`select * from sale/stock` 获取所有收入/支出记录，渲染到前端

五、特色和创新点

5.1 增添顾客模块

5.1.1 顾客登陆：

选择我是顾客后，页面跳转到顾客登陆，若验证成功（验证原理和管理员登陆相同）则跳转到顾客主页

（顾客主页）

5.1.2 搜索图书

5.1.2.1 类别导航栏:点击左侧的类别条目，可搜索书库中该类别的所有图书

实现原理：后端 `select * from book B,categoryInfo C where C.categoryName='xx' and C.categoryId=B.categoryId` 获取数据+render 渲染

5.1.2.2 手动搜索：在上方的输入框中可输入书名或作者实现模糊查询，原理类似于管理员模块的模糊查询——点击“搜索”，提交表单，后端访问数据库，利用 like 通配符实现模糊查询，然后 render 渲染前端

例如，输入祈祷，得到书库中书名为《祈祷落幕时》的书

5.1.3 加入购物车

顾客可点击每个条目左侧的框框选择图书，点击“+”增加数量，点击“-”减少数量，下方的合计实时显示总金额，点击“加入购物车”。

实现原理：通过.toggle()函数改变选择框的选中状态；监听“+”和“-”的 click 事件，对 id=“数量”的输入框的值作出相应改变；定义 tm_total()函数，对每一条目判断选中状态，若选中，则通过.text()获取单价框和数量框的值，进行运算；监听“加入购物车”的 click 事件，点击时，对每一条目判断选中状态，若选中，获取 isbn、单价、数目， ajax 将数据 POST 到后端，后端接收数据，向购物车表中插入：`insert into shopCart (isbn,customerId,amount,quantity) values (@isbn,@customerId,@amount,@quantity)`

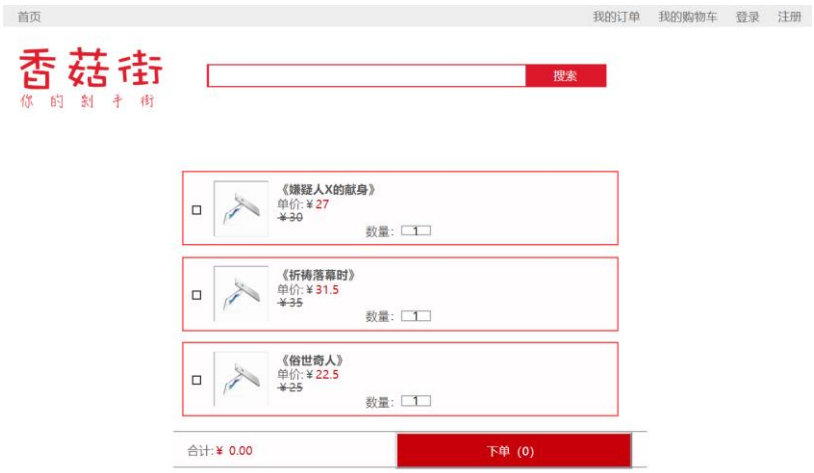
5.1.4 登记想看的书

在右上角的输入框中输入书名和作者，点击加入书单。

实现原理：表单提交+后端 `insert into bookWanted`

5.1.5 查看购物车、下单

点击上方导航栏的“我的购物车”，选择条目，点击下单。



实现原理：

1）页面初始：`select * from queryShopCart where customerId=@id` 从 queryShopCart 视图中获取当前顾客的购物车数据渲染到前端

2）下单：前端类似“加入购物车”功能，后端 `update shopCart set isfirmed='yes' where orderId=@orderId` 将 shopCart 中对应的行的 isfirmed 字段更新为‘yes’代表确认下单，同时 afterOrder 触发器会向 _order 表插入一条数据

```
create trigger afterOrder
on shopCart
after update
as
if update(isfirmed)
begin
    declare @OI varchar(20);
       select @OI=orderId from inserted;
       insert into _order(orderId) values(@OI);
    end
```

5.1.6 查看订单、确认收货

订单有不同的状态：“卖家已发货”、“卖家未发货”、“已确认收货”，显示“卖家已发货”的订单，可选择并确认收货

实现原理：

1）页面初始：`select * from queryOrder where customerId=@id`从 queryOrder 视图中获取当前顾客的订单数据渲染到前端

2）确认收货：前端类似“加入购物车”功能，但是只有处于“卖家已发货”状态的条目可以选择；后端`update _order set isreceived='yes' where orderId=@orderId`将_Order 中对应的行的 isreceived 字段更新为‘yes’代表确认收货；当 isreceived 更新后，afterReceive 触发器会向 sale 表插入一条数据并更新该顾客的累积消费；当向 sale 表插入一条数据后，afterSale 触发器会自动更新书籍库存数；当累积消费更新后，changeDiscount 触发器又会更新该顾客的折扣。

5.2 增添顾客累计消费与折扣：在“我的购物车”和“我的订单”中都有原价和折后价，更加贴合现实情况；当顾客确认收货后，触发器会自动更新顾客的累积消费，接着再根据累积消费的变化自动更新顾客享有折扣。

5.3 增加缺书登记表：当顾客搜索书籍未果时，可登记自己想看的书，管理员在进货时便有了根据，更贴合实际。

5.4 模糊查询：管理员和顾客在搜索书籍时无需输入完整名字，通过 sql 的 like 通配符支持近似查询。

5.5 分类查询：顾客除了手动输入内容外，还可直接在首页点击左侧类别导航栏，查看书库中该类别的所有书籍

5.6 特色：本项目的特色其实就是顾客模块和管理员模块的交互：

1）顾客登记想看的书，管理员据此进货

3） 顾客下单，管理员根据订单发货，顾客查看已发货的订单确认收货

六、提交文件说明

6.1 文件结构

1）bin：存放项目启动的执行文件

2）node_modules：通过 npm 安装的依赖（npm 安装完成时，该文件夹生成）

3）package.json：需要安装的依赖

4）public：静态资源文件夹,包括图片、css 文件、js 文件

5）routes：路由文件，存放处理路由 get、post 的函数

6）views：视图文件，存放 ejs 前端渲染模板

7）sql：ddl.sql 是建表、视图和触发器的代码；dml.sql 是插入初始数据的代码

8）app.js: 应用的核心配置文件，加载初始化依赖模块，应用的入口。

zdiy.js：默认的名称为 app.js，应用的核心配置文件，加载初始化依赖模块，应用的入口。

6.2 使用说明：

1）下载 nodejs

2）在根目录下，在终端输入 npm install 下载所有依赖

3） 在根目录下，在终端输入 nodemon，然后在浏览器打开 localhost：3306

七、实验总结

7.1 问题

1）一开始做的时候是模仿别人用 electron 构建一个 app，但发现别人的需求和自己的不一致，只好弃之，转战 nodejs+express 构建网页

2）最初对数据库并没有什么设计，仅仅按照给出的要求建了几张表，实体之间也不存在什么联系。然后就开始动手前端，做完之后觉得自己的数据库设计太粗糙，于是重新设计——增添顾客模块、重新规划实体和联系、画 E-R 图、模式合并，因此前端也需要大改。

3）最初后端连接数据库使用‘mysql’模块，却老是连接超时，弃之，改用‘mssql’

7.2 学习

1)一定要先设计好数据库，再做前端

2）数据库的设计遵循：功能分析->概念设计（实体和联系、E-R 图）->逻辑设计(转换为关系模式，进行模式合并)->根据需要建立视图和触发器

3）关于前端数据提交到后端的几种方式：

有 request 且需要跳转页面：form 表单提交

有 post 且不需要跳转页面：ajax

没有 request 且需要跳转页面：a