北京邮电大学

数据库系统原理课程设计实验

（详细设计） 报 告

2019211304 班 2019211285 窦天杰

2019211304 班 2019211284 林源

2019211304 班 2019211278 赵文龙

2022 年 5 月 16 日

目录

一、 数据库表的设计

账户（account）

- 其中包含属性：用户 id（username），密码（password），昵称（nickname），类别（category），状态（state），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.account
          (
              username varchar(256),
              password varchar(256),
              nickname varchar(256),
              category varchar(256),
              state varchar(256),
              flag int2,
              primary key(username)
          );
```

商家（business）

- 其中包含属性：商家 id（username），店铺简介（introduction），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.business
              (
                  username varchar(256),
                  introduction varchar(256),
                  flag int2,
                  primary key(username)
              );
```

会员（vip）

- 其中包含属性：买家 id（account_username），商家 id（business_username），累计消费（spend），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.vip
          (
              account_username varchar(256),
              business_username varchar(256),
              spend varchar(256),
              flag int2,
              primary key(account_username, business_username)
          );
```

会员体系（vip_system）

- 其中包含属性：商家 id（business_username），折扣阶梯（discount_ladder），金额阶梯（cost_ladder），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.vip_system
              (
                  business_username varchar(256),
                  discount_ladder varchar(128),
                  cost_ladder varchar(128),
                  flag int2,
                  primary key(business_username)
              );
```

订单（order）

- 其中包含属性：订单 id（id），买家 id（account_username），商家 id

（business_username），商品 id（commodity_id），商品名称（name），单价（price），数量（number），折扣（discount），提交时间（commit_time），标记（flag）

- 使用 PostgreSQL 建立表的代码为：

```
create table test.order
           (
               id varchar(512),
               account_username varchar(256),
               business_username varchar(256),
               commodity_id varchar(512),
               name varchar(256),
               price varchar(256),
               number int8,
               discount varchar(256),
               commit_time varchar(128),
               flag int2,
               primary key(id)
           );
```

商品（commodity）

- 其中包含属性：商品 id（id），商家 id（business_username），商品名称（name），单价（price），数量（number），商品简介（introduction），商品类别（category），销量（sale），状态（state），上架时间（online_time），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.commodity
               (
                   id varchar(512),
                   business_username varchar(256),
                   name varchar(256),
                   price varchar(256),
```

```
                number int8,
                introduction varchar(256),
                category varchar(256),
                sale int8,
                state varchar(128),
                online_time varchar(128),
                flag int2,
                primary key(id)
            );
```

购物车（cart）

- 其中包含属性：买家 id（account_username），商家 id（business_username），标记（flag）
- 使用 PostgreSQL 建立表的代码为：

```
create table test.cart
       (
           account_username varchar(256),
           commodity_id varchar(512),
           flag int2,
           primary key(account_username, commodity_id)
       );
```

一些数据不能直接 drop/delete 掉，我们采用标记删除方法，每个表项中都有 flag（标记），值为 0 时为隐藏（已删除）状态，值为 1 时为正常状态。

二、 客户端设计

以下每个页面都应具备“登出”按钮。

商家

商家主页

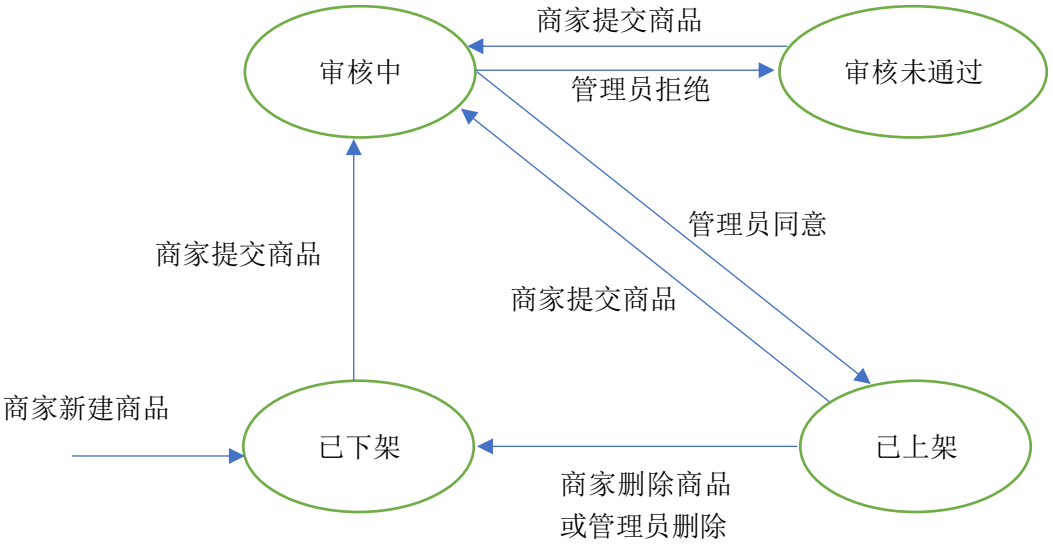
顶部需要具备商品种类选择的下拉菜单，金额区间文本框，关键字文本框，搜索按钮；下方适当位置展示店铺状态与店铺简介信息，其下需要给出显示商品信息的空间，包括商品种类、单价、状态、数量、id、简介，每一件商品应该包含提交和删除按钮。

商品下方应该具备“上新”按钮。

侧边栏需要具有“订单查询”，“会员折扣”，“修改信息”按钮，同时表明该用户的 id 和昵称。

应该具备翻页按钮。

这里给出商品状态转换示意图：



其中若商品状态为已下架、审核中、审核未通过时而被商家删除，该商品信息将不会显示在商家主页中。

修改信息页面

需要给出用户昵称文本框、店铺简介文本框、密码文本框（两次），供用户修改信息，同时具备提交按钮。

侧边栏需要具有“订单查询”，“会员折扣”，“修改信息”按钮，同时表明该

用户的 id。

折扣设置页面

需要给出折扣——金额对应的文本框，供商家设置折扣阶梯和金额阶梯，同时具备提交按钮。

侧边栏需要具有“订单查询”,“会员折扣”,“修改信息”按钮，同时表明该用户的 id 和昵称。

查看订单页面

顶部应该具备关键字输入框以及查询按钮。

网页内应该显示订单详细信息，其应该包括：订单 id，订单提交时间，商品名称，商品 id，商品价格，商品数量，顾客昵称，顾客 id，商品折扣，订单总价。

侧边栏需要具有“订单查询”,“会员折扣”,“修改信息”按钮，同时表明该用户的 id 和昵称。

应该具备翻页按钮。

买家

买家主页

需要能够切换功能为“查询商品”或“查看店铺”。

功能为“查询商品”时，顶部需要商品种类选择的下拉菜单，金额区间文本框，关键字文本框，搜索按钮；下方展示商品信息，其包括：商品名称，商品价格，店铺名称，商品 id，商品简介，每一个商品还要包含“添加购物车”按钮。

功能为“查询店铺”时，顶部需要关键字文本框，搜索按钮；下方展示店铺信息：店铺名称，店铺 id，店铺简介，每一个店铺还应该包含跳转按钮用来进入店铺。

侧边栏需要具有“购物车”,“订单查询”,“修改信息”按钮，同时表明该用

户的 id 和昵称。

应该具备翻页按钮。

查看店铺页面

顶部需要商品种类选择的下拉菜单，金额区间文本框，关键字文本框，搜索按钮；下方展示店铺名称，店铺 id，店铺简介；其下应该是商品列表，展示方式和主页的展示方式相同。

侧边栏需要具有“购物车”,“订单查询”,“修改信息”按钮，同时表明该用户的 id 和昵称。

应该具备翻页按钮。

购物车页面

购物车页面需要展示每一个商品的详细信息：商品名称，商品价格，商品折扣，店铺名称，商品 id，商品剩余，用户购买数量文本框，以及“移出购物车”按钮。

商品底部应该具备“提交”按钮用来提交订单。

侧边栏需要具有“购物车”,“订单查询”,“修改信息”按钮，同时表明该用户的 id 和昵称。

修改信息页面

需要给出用户昵称文本框、密码文本框（两次），供用户修改信息，同时具备提交按钮。

侧边栏需要具有“购物车”,“订单查询”,“修改信息”按钮，同时表明该用户的 id。

查看订单页面

顶部应该具备关键字输入框以及查询按钮。

网页内应该显示订单详细信息，其应该包括：订单 id，订单提交时间，商品名称，商品 id，商品价格，商品数量，商品折扣，订单总价。

侧边栏需要具有“购物车”,“订单查询”,“修改信息”按钮，同时表明该用户的 id。

管理员

管理员主页

顶部应具备选择功能的按钮。

选择功能为“审核商品”时，应具备筛选商品种类以及关键字文本框，同时具有查询按钮，下面应该展示商品信息：商品名称，商品价格，店铺名称，店铺 id，商品简介，其中每一个商品应该具备“同意”“拒绝”按钮。

选择功能为“管理店铺”时，应具备关键字文本框，同时具有查询按钮，下面展示店铺：店铺名称，店铺状态，店铺 id，店铺简介以及“切换”按钮。

选择功能为“管理商品”时，应具备筛选商品种类以及关键字文本框，同时具有查询按钮，下面应该展示商品信息：商品名称，商品价格，店铺名称，店铺 id，商品简介，每一个商品还应该具备“下架”按钮。

侧边栏需要具有 “订单查询”,“修改信息”按钮，同时表明该用户的 id 和昵称。

应该具备翻页按钮。

查看订单页面

应具备关键字文本框，同时具有查询按钮，网页内应该展示订单相关信息：订单 id，订单提交时间，商品名称，商品 id，商品价格，商品数量，店铺名称，店铺 id，顾客名称，顾客 id，订单折扣，订单总价。

侧边栏需要具有 “订单查询”,“修改信息”按钮，同时表明该用户的 id 和昵称。

应该具备翻页按钮。

修改信息页面

需要给出用户昵称文本框、密码文本框（两次），供用户修改信息，同时具备提交按钮。

侧边栏需要具有 “订单查询”,“修改信息”按钮，同时表明该用户的 id。

公共页面

登录页面

包含 id 和密码输入框，登录按钮以及切换注册页面按钮。

注册页面

包含 id 和密码输入框（两次），商家注册、买家注册、管理员注册三个按钮

JS 简要说明

翻页

在前端一些页面中很多地方采用了翻页功能，我才用的具体方法是：后端返回需要展示的全部数据，js 控制每一页需要展示的商品个数（5 个），以下为翻页的具体实现方式：

```
/* 点击 下一页 */
    $('#nextButton').on('click', function () {
        goToTop()
        if (maxNo < totalNo-1) {
            minNo = minNo + 5
            maxNo = maxNo + 5
            showPage()
        }
    })
    /* 点击 上一页 */
    $('#lastButton').on('click', function () {
```

```
        goToTop()
        if (minNo != 0) {
            minNo = minNo - 5
            maxNo = maxNo - 5
            showPage()
        }
    })
```

其中 minNo 和 maxNo 表示需要展示的数据的索引，这样每翻一次页就展示 minNo 和 maxNo 之间的内容，实现了翻页的功能。

数据发送

js 在向后端发送数据时，一般是直接从网页中提取，之后打包成为 js 对象，进而使用 json 格式字符串发送，例如：

```
var name = $(this).closest('#item_1').find('.itemName').val()
var price = parseFloat($(this).closest('#item_1').find('.itemPrice').val())
var number = parseInt($(this).closest('#item_1').find('.itemNum').val())
var id = $(this).closest('#item_1').find('.itemID').attr('placeholder')
var introduction = $(this).closest('#item_1').find('#introduction').val()
var send_data = {
    category: category,
    name: name,
    price: price,
    number: number,
    introduction: introduction,
}
```

这样 send_data 就成为了需要发送的内容。

数据填充

在与后端交互方面，除了向后端请求网页，还有就是需要请求数据，至于数据处理大多是结合 html 代码拼接到网页中，具体需要找到所要填充数据的位置，例如：

```
function showPage(){
    $('#itemList').empty()
    for (i = minNo; i < totalNo && i <= maxNo; i++){
        totalPrice = parseFloat(data[i].number) * parseFloat(data[i].discount) * parseFloat(data[i].price)
        $("#itemList").append('<li id="item_1">\
        <div class="indentInfo1 clearfix">\
            <p> # \
                <span class="indentID">'+data[i].id+'</span><!--在此填充订单ID-->\
            </p>\
            <p> date \
                <span>'+data[i].commit_time+'</span><!--在此填充订单时间-->\
            </p>\
        </div>\
        <div class="indentInfo2 clearfix">\
            <p> \
                <span>'+data[i].name+'</span><!--在此填充商品名称-->\
            </p>\
```

这是商家填充订单页的代码，除了需要计算出 totalPrice，还需要在相应标签下填充 html，而这一段 html 里面的数据也需要从获取的数据 data 中取得。

三、 前后端接口设计

商家

url	请求方式	功能
merchant	GET	返回店铺网页（merchant/shop.html）
getBusinessInfo	POST	返回店铺状态和店铺简介
getBusinessCommodity	POST	返回当前店铺所有商品
offTheShelf	POST	给定商品的 id，改动其状态为已下架；遍历 cart，将相应的商品 id 的购物车 flag 置为 0
deleteCommodity	POST	给定商品的 id，改动其 flag=0
newCommodityForReview	POST	给定新商品相关属性，将其插入到 commodity 表中
oldCommodityForReview	POST	给定特定商品相关属性，修改其在数据库中的相关属性，并将其状态设为审核中
orderForBusiness	GET	返回商品网页（merchant/indent.html）
vipDiscount	GET	返回商铺折扣页面 merchant/discount.html
modifyBusinessInfo	GET、POST	返回商铺信息页面 merchant/information.html 根据当前商家用户名查找查店铺名称与简介

orderForBusiness	POST	根据当前商家用户名查所有订单信息与商家的昵称
vipDiscount	POST	根据当前商家用户名查 vip_system 全部信息
changeVipDiscount	POST	根据当前商家用户名、post_ladder、discount_ladde 在 vip 系统中进行更新
changeBusinessInfo	POST	根据当前商家用户名、nickname、introduction、password 在账户、商铺表中修改相关信息

买家

url	请求方式	功能
customer	GET	根据前端发送的 type 返回商品信息和店铺名称，或店铺信息和店铺简介
specifiedBusiness	GET、POST	返回 customer/shop.html 或返回商家的昵称、id、店铺简介
addToCart	POST	将前端发送的商品添加至 cart 表中
checkCart	GET、POST	返回 customer/wish.html 或返回这个用户的购物车全部信息 （需要补全商品相关信息）
orderForCustomer	GET、POST	返回 customer/indent.html 或返回这个用户的全部商品订单信息+店铺昵称
modifyCustomerInfo	GET、POST	返回 customer/information.html
specifiedBusinessCommodity	POST	根据前端发送的商家 id 返回这家店的全部商品信息（已上架）
deleteFromCart	POST	将前端发送的商品从购物车删除
createOrder	POST	提交订单信息（需要修改 order、commodity、vip、cart 四张表）

管理员

url	请求方式	功能
adminWishData	POST	具备三种可能： 1. 功能为管理商品：查询全部已上架的商品信息+店铺名称并返回 2. 功能为管理店铺：全部店铺信息+店铺状态+店铺昵称 3. 功能为审核商品：全部审核中的商品信息+店铺名称
offTheShelf	POST	修改商品状态为已下架
switchState	POST	切换商家状态
reviewResult	POST	修改商品状态
orderForAdmin	GET、POST	返回 administrator/indent.html 或全部订单信息
modifyAdminInfo	GET、POST	返回 administrator/information.html 或本人的基本信息
changeAdminInfo	POST	更新本人基本信息

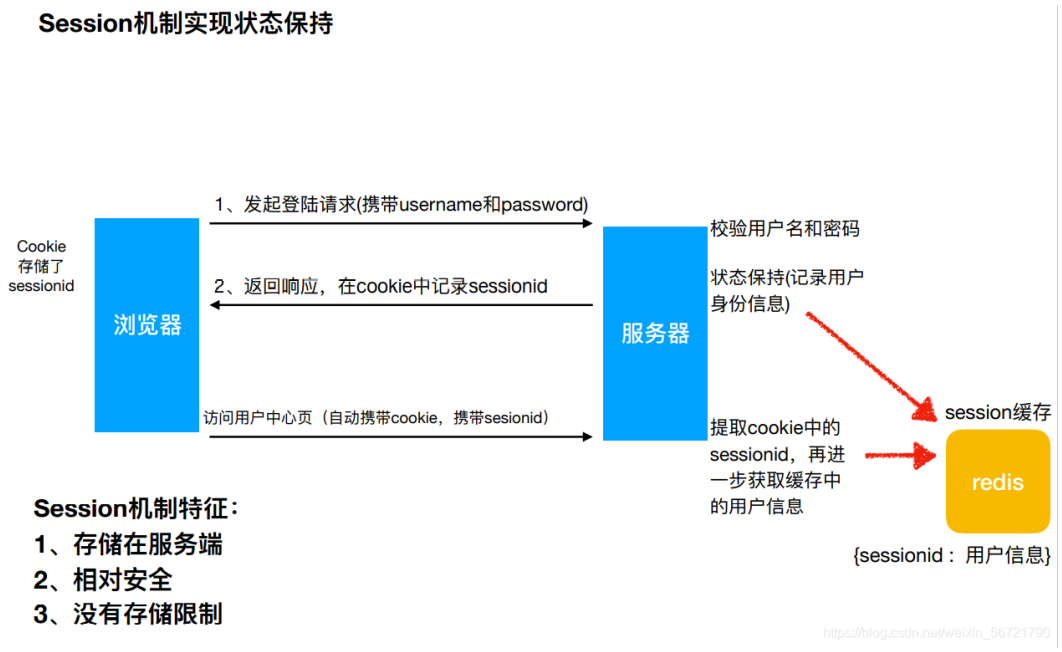
公共

url	请求方式	功能
register	POST	判断注册信息是否重复，若不重复添加至数据库中
login	POST	查找登录信息是否存在于数据库中，返回成功或失败信息
customer	GET	返回买家主页
merchant	GET	返回商家主页
admin	GET	返回管理员主页

getNewUser	GET	返回注册页面
exit	POST	从 session 中删除当前登录用户信息
publicInfo	POST	获取当前登录用户的昵称、id

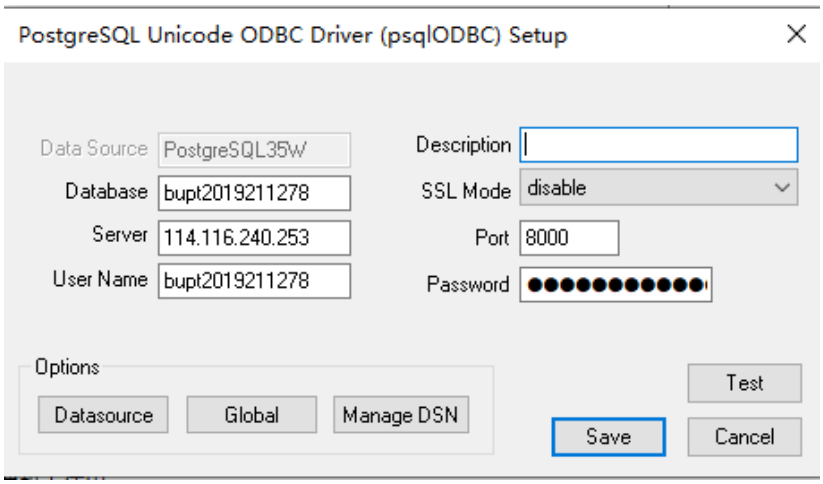
四、 数据库接口设计

后端采用 session 用来记录账户的登陆状态。



前后端对接模块通过 requests 库，以 json 格式字符串的形式放在请求头中传递数据给服务器端。

数据库依托华为云 DAS（PostgresSQL），使用 Python 及 ODBC（pyodbc 模块）进行连接：下载相关驱动程序并安装，在 windows 自带 odbc 数据源管理器中进行配置：



在 pyodbc 中使用如下语句连接到数据库：

```
conn = pyodbc.connect(
    '''DRIVER={PostgreSQL Unicode};
    SERVER=114.116.240.253;
    PORT=8000;
    DATABASE=bupt2019211278;
    UID=bupt2019211278;
    PWD=bupt2019211278@''')
```

其中包括驱动、数据库 ip 地址、端口号、数据库名以及登录账号和密码。

根据对应表项及其相关类型撰写对应创建表语句，例如在注册过程中需要将 id 和 password 插入数据库中，我们可以写如下语句：

```
cursor = conn.cursor()
cursor.execute("insert into test.account values('{}','{}','{}','{}','{}',{})".form
at(get_data["username"],get_data["password"],'', get_data["category"], '正常', 1))
```

有些功能需要从前后端中继模块得到相关数据（json 格式传递）已进行数据库查询、插入等操作，需要 requests 模块中 get_json 函数获取相关数据；同样，中继模块需要数据库相关数据时，数据库查询到的数据与相关属性名构成 json 格式后再传回中继模块。

以筛选当前商家商品部分为例，相关函数截图如下：

```python
@app.route('/specifiedBusinessCommodity', methods=['POST'])
def specifiedBusinessCommodity():
    get_data = request.get_json()
    results=[]
    cursor.execute("select * from test.commodity where business_username ='{}' and flag = 1 and number > 0".format(get_data["username"]))
    columns = [column[0] for column in cursor.description]
    for row in cursor:
        results.append(dict(zip(columns, row)))
    return json.dumps(results)
```