中国人民大学
RENMIN UNIVERSITY OF CHINA

# 网上 B2C 购物系统

项目名称：Bikini Town 网上食品购买系统

组长：郑语悉

组员：高一画　何暾　林哲伊　谢文杰

目录

# 一 实验环境

操作系统：Windows 10
语言：SQL Server，C#
环境：Visual Studio 2019 .NET Framework, SQL Server Management Studio 2021

# 二 需求分析

## 2.1 系统任务

比奇堡是一座坐落于太平洋底的城市，比奇堡拥有着超过 1500 家汉堡店，因而一直承担着比基尼海滩超过 80%的汉堡供应，但随着时代的发展，比基尼海滩的客户已经不只满足于线下的服务，进而向比奇堡的线上服务提出诉求。

为充分满足客户需求，比奇堡市长向本小组成员委托完成比奇堡线上购物系统的开发。系统将用于进行汉堡的线上售卖，此系统需要面向三类群体：购买者、商家、管理员，为他们提供使用起来简单便捷、安全性高的数据库系统，通过查询、更新、删除、插入等数据库操作，来实现用户查询、购买、评价商品，管理购物车，商家管理商品信息、订单，管理员维护各类信息及管理评论等功能。（见图表 1）

**1. 会员管理**

（1） 申请注册成为会员。用户初次登录系统时会被要求注册成为会员，会员属性包括编号、昵称、性别、出生日期、手机号码、密码、会员省份；

（2） 查询商品信息（商品编号、名称、单价、商品介绍、库存），查看某一商品售价最便宜的五个商家，点击商品查看其他购买者给出的评价以及平均打分；

（3） 购物车：每个会员都有一个购物车，存储需要购买的商品，可以更改需要购买的数量以及收货地址、删除购物车中的商品、一键购买购物车中所有商品；

（4） 会员个人中心：会员可以查看自己的历史订单、自己购买各种商品的频繁程度排序和自己给出的历史评价，给购买过的商品做出评价并打分，删除历史评价。

**2. 商家管理**

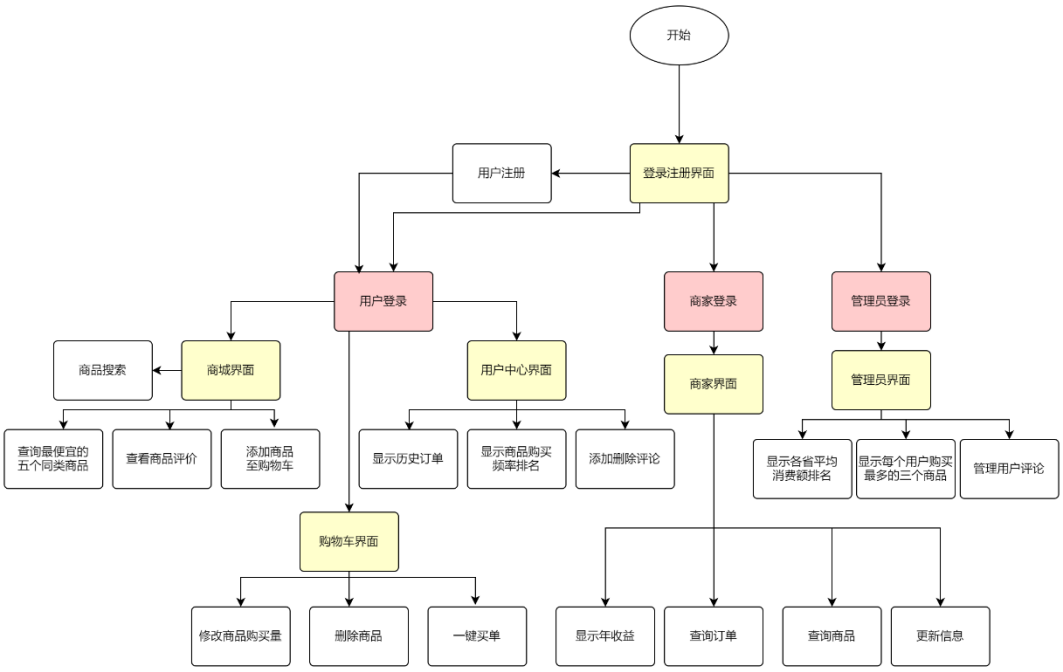
（1） 查看店铺的历史订单、本店销量前三的商品和年销售额；

（2） 商品管理：查询本店商品，更改商品库存。

**3. 管理员管理**

（1） 查看各省会员购买额的均值、最大值、最小值，查看每个会员购买次数最多的商品；

（2） 评论管理：查看全部评论，删除违规评论。



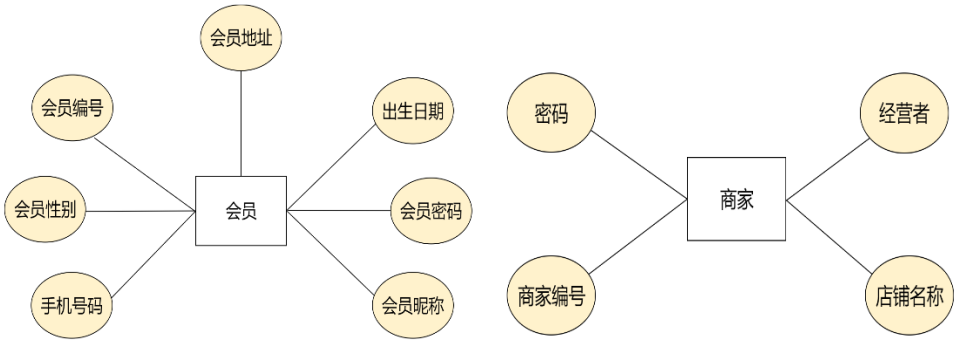

图表 1 系统功能图

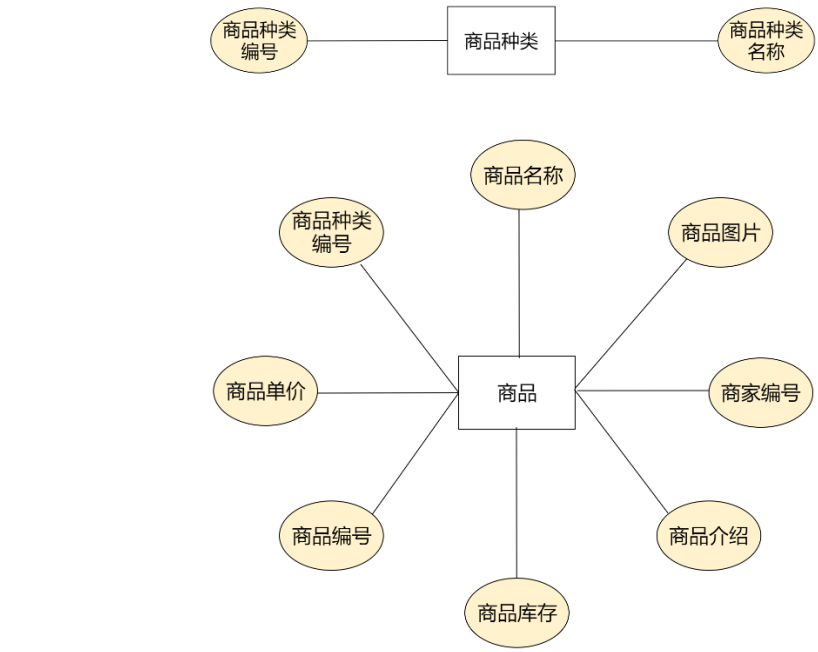
# 三 系统设计

## 3.1 数据库概念结构设计

### 3.1.1 E-R 模型设计

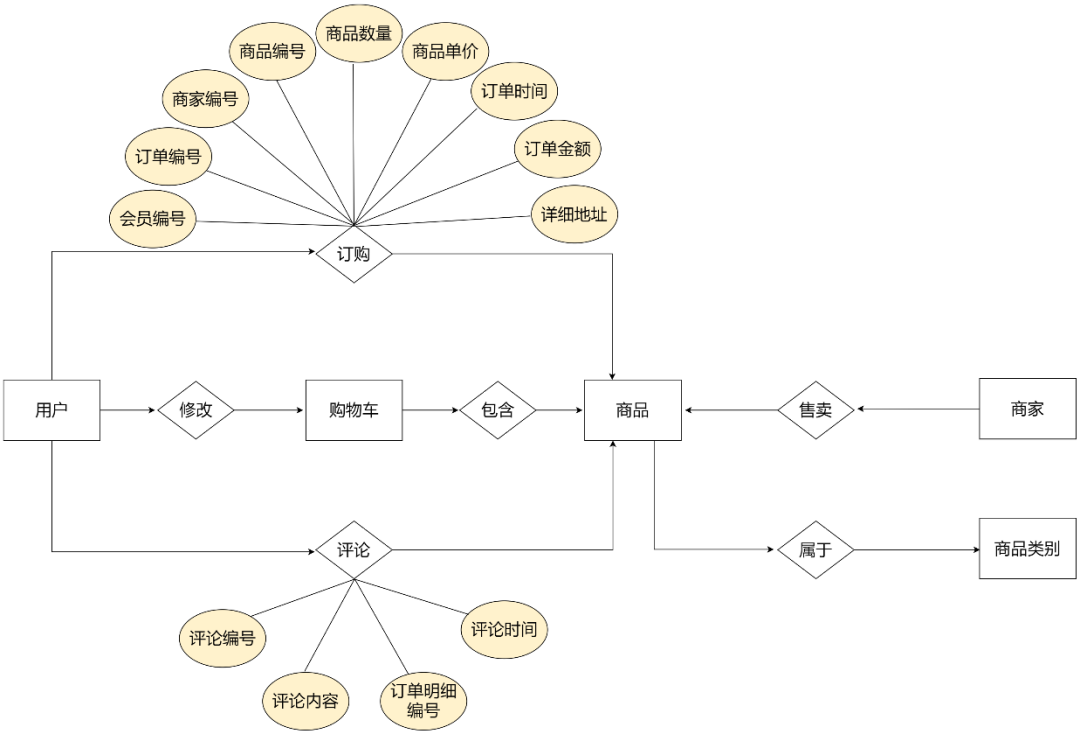
首先将以上需求抽象得到系统局部 E-R 图。在 E-R 结构的设计过程中，采用由下到上的设计方式，即先完成各个实体和关系的 E-R 结构模型，找出分别的属性，再根据它们之间的关系对各实体进行组合，从而得到总体的 E-R 模型。模型一共包含六个实体，分别是会员、商家、商品、商品种类、订单明细、购物车，其中会员和商品之间又存在着订购关系和评论关系。






图表 2 数据库局部 E-R 图

在构建好各子系统 E-R 图后，分析他们之间是否存在冲突。发现不存在属性冲突、命名冲突、结构冲突，可以将它们合并为数据库整体的 E-R 模型。




图表 3 数据库整体 E-R 模型

## 3.1.2 数据字典

会员表

| Users | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| UserID | 会员编号 | CHAR(8) | 是 | 否 | |
| UserName | 会员昵称 | VARCHAR(16) | 否 | 否 | |
| gender | 会员性别 | CHAR(1) | 否 | 否 | |
| birthday | 出生日期 | DATE | 否 | 否 | |
| tel | 手机号码 | CHAR(11) | 否 | 否 | |
| UserPwd | 会员密码 | VARCHAR(18) | 否 | 否 | |
| province | 会员省份 | VARCHAR(15) | 否 | 否 | |

商家表

| Businesses | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| BusinessID | 商家编号 | CHAR(8) | 是 | 否 | |
| ShopName | 店铺名称 | VARCHAR(50) | 否 | 否 | |
| OprName | 经营者 | VARCHAR(16) | 否 | 否 | |
| BusinessPwd | 商家密码 | VARCHAR(18) | 否 | 否 | |

商品种类表

| Producttype | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| TypeID | 商品种类编号 | CHAR(12) | 是 | 否 | |
| type | 商品种类名称 | VARCHAR(18) | 否 | 否 | |

商品表

| Products | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| ProductID | 商品编号 | CHAR (12) | 是 | 否 | |
| BusinessID | 商家编号 | CHAR(8) | 否 | 是 | |
| TypeID | 商品种类编号 | VARCHAR(12) | 否 | 是 | |
| name | 商品名称 | VARCHAR(18) | 否 | 否 | |
| unitprice | 商品单价 | REAL | 否 | 否 | |
| introduction | 商品介绍 | TEXT | 否 | 否 | |
| inventory | 商品库存 | SMALLINT | 否 | 否 | |
| image | 商品图片 | VARCHAR（200） | 否 | 否 | |

订单表

| Orders | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| OrderID | 订单编号 | CHAR(12) | 是 | 否 | |
| UserID | 会员编号 | CHAR(8) | 否 | 是 | |
| BusinessID | 商家编号 | CHAR(8) | 否 | 是 | |
| Address | 详细地址 | CHAR(100) | 否 | 否 | |
| amount | 订单金额 | REAL | 否 | 否 | |
| OrderDate | 订单时间 | DATETIME | 否 | 否 | |

订单明细表

| Orderdetail | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| DetailID | 订单明细编号 | CHAR(12) | 是 | 否 | |
| OrderID | 订单编号 | CHAR(12) | 否 | 是 | |
| ProductID | 商品编号 | CHAR (12) | 否 | 是 | |
| unitprice | 商品单价 | REAL | 否 | 否 | |
| quantity | 商品数量 | INTEGER | 否 | 否 | |

评论表

| Comments | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| CommentID | 评论编号 | INT | 是 | 否 | |

| | | | | | |
|---|---|---|---|---|---|
| DetailID | 订单明细编号 | CHAR(12) | 否 | 是 | |
| comment | 评论内容 | TEXT | 否 | 否 | |
| date | 评论时间 | DATETIME | 否 | 否 | |

购物车表

| Cart | | | | | |
|---|---|---|---|---|---|
| 列名 | 含义 | 类型 | 主键 | 外键 | 备注 |
| CartID | 购物车编号 | CHAR(12) | 是 | 否 | |
| UserID | 会员编号 | CHAR(8) | 否 | 是 | |
| ProductID | 商品编号 | CHAR(12) | 否 | 是 | |
| ProductQuantity | 商品数量 | INTEGER | 否 | 否 | |

## 3.2 数据库逻辑结构设计

### 3.2.1 基本表和视图的关系模式

将以上 E-R 图转化为**基本表的关系模式**，得到：

会员（会员编号，会员昵称，会员性别，出生日期，手机号码，会员密码，会员省份）

商家（商家编号，店铺名称，经营者，商家密码）

商品（商品编号，商品名称，商品种类编号，商品单价，商品介绍，商家编号，商品库存，商品图片）

商品种类（商品种类编号，商品种类名称）

订单明细表（订单明细编号，订单编号，商品编号，商品单价，商品数量）

订单（订单编号，会员编号，商家编号，详细地址，订单金额，订单时间）

评论（评论编号，订单明细编号，评论内容，评论时间）

购物车（购物车编号，会员编号，商品编号，商品数量）

为了达到一个订单中包含多个商品的功能，同时考虑到如果只有订单表可能会造成订单编号、会员编号、商家编号、订单金额等数据的大量冗余，而在订单表之外添加订单明细表。

由于评价部分涉及三个表的连接，且需要进行多次的查询。因此考虑查询效率，引入视图将评价表、订单明细表和订单表按它们的 ID 连接在一起，得到**视图的关系模式：**

全部评论（评论编号，订单明细编号，订单编号，会员编号，商品编号）

### 3.2.2 完整性约束

实体完整性要求以上关系模式中的主码不能为空值。以上关系模式已根据参照完整性设置主码和外码，确定参照与被参照关系。根据用户定义的完整性，要求商品库存为大于零的整数，并根据这一要求设置触发器。

### 3.2.3 范式

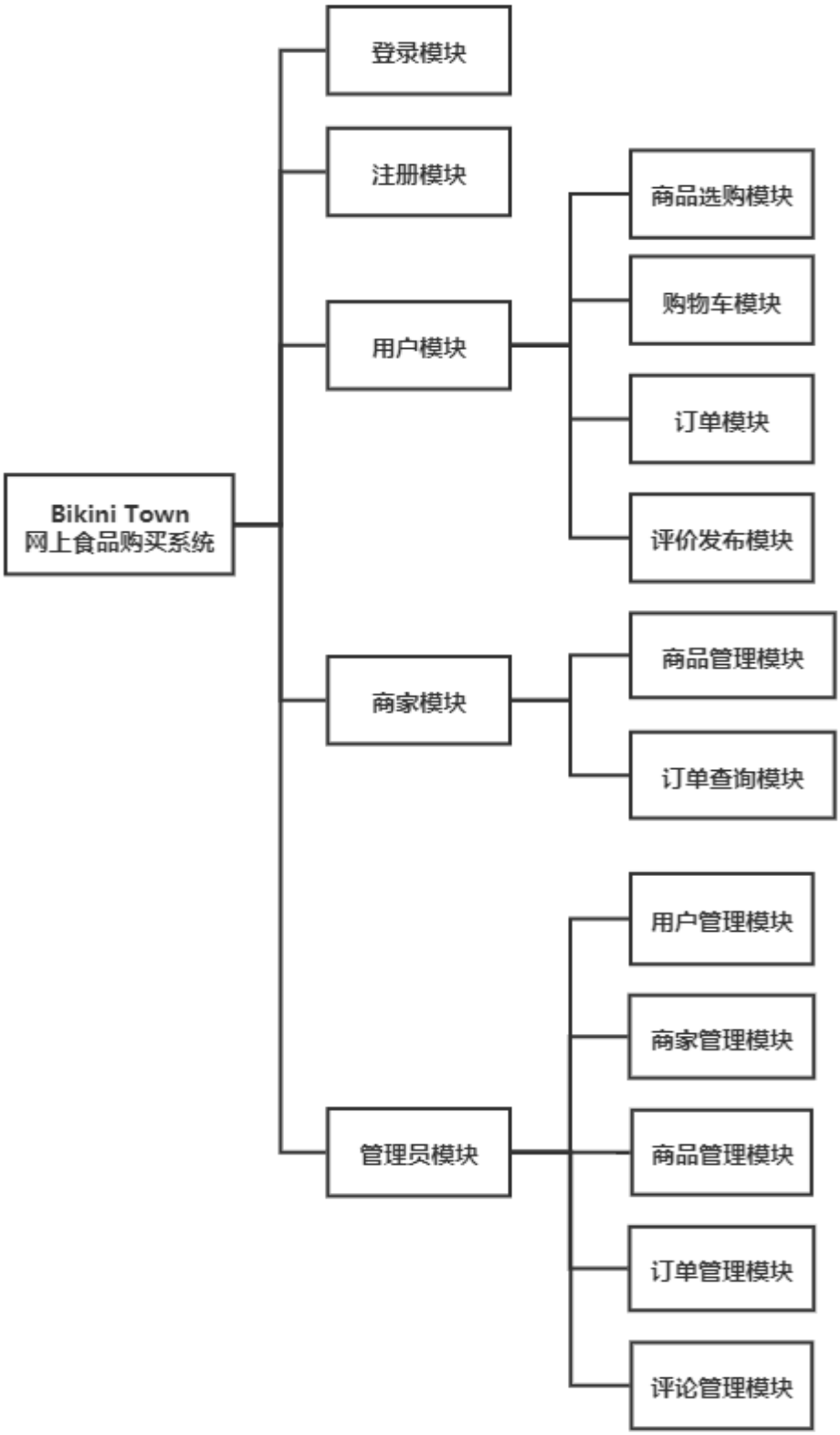
除了因为查询便捷需要添加了商品编号导致商品表只能达到 2NF 外，其余各表均达到 3NF。除商品外，其他均能够满足在一张表中，在属性（或属性组）X 的值确定的情况下，必定能确定属性 Y 的值，即消除了函数依赖。也不存在着部分函数依赖、传递函数依赖、多值依赖等问题，满足 3NF。而商品关系模式中，不满足每一个非主属性都完全函数依赖于商品编号，故不满足 3NF。

由于我们的关系模式基本达到要求，所以没有进一步进行模型上的优化。

## 3.3 数据库物理结构设计

由于评价部分涉及三个表的连接，且需要进行多次的查询。因此考虑查询效率，引入视图将评价表、订单明细表和订单表按它们的 ID 连接在一起，并在视图上建立了一个关于订单 ID 的聚集索引和关于其他两个表 ID 的两个非聚集索引，从而提高查询效率。

## 3.4 系统功能设计




图表 4 数据库功能模块

# 四 数据库实施

## 4.1 建表语句

根据以上总结出的关系模式，创建 B2C 数据库以及会员表、商家表、商品种类表、订单表、订单明细表、评论表、购物车表。代码如下：

```
CREATE DATABASE B2C --创建数据库
USE B2C
--创建会员基本表
CREATE TABLE Users
(UserID  CHAR(8) PRIMARY KEY,
UserName VARCHAR(16),
gender CHAR(1),
birthday DATE,
tel CHAR(11),
UserPwd VARCHAR(18),
province VARCHAR(15))

--创建商家基本表
CREATE TABLE Businesses
(BusinessID  CHAR(8) PRIMARY KEY,
ShopName VARCHAR(50),
OprName VARCHAR(16),
BusinessPwd VARCHAR(18))

--创建商品种类基本表
CREATE TABLE Producttype
(TypeID CHAR(12) PRIMARY KEY,
type VARCHAR(18))

--创建商品基本表
CREATE TABLE Products
(ProductID CHAR (12) PRIMARY KEY,
BusinessID CHAR(8),
TypeID CHAR(12),
name VARCHAR(18),
unitprice REAL,
introduction TEXT,
image VARCHAR(200),
inventory SMALLINT,
FOREIGN KEY(BusinessID)REFERENCES Businesses(BusinessID),
```

```
FOREIGN KEY(TypeID)REFERENCES Producttype(TypeID))

--创建订单基本表
CREATE TABLE Orders
(OrderID CHAR(12) PRIMARY KEY,
UserID CHAR(8),
BusinessID CHAR(8),
Address CHAR(100),
amount REAL,
OrderDate DATETIME,
FOREIGN KEY(UserID)REFERENCES Users(UserID),
FOREIGN KEY(BusinessID)REFERENCES Businesses(BusinessID))

--创建订单明细基本表
CREATE TABLE Orderdetail
(DetailID CHAR(12) PRIMARY KEY,
OrderID CHAR(12),
ProductID CHAR (12),
unitprice REAL,
quantity INTEGER,
FOREIGN KEY(OrderID)REFERENCES Orders(OrderID),
FOREIGN KEY(ProductID)REFERENCES Products(ProductID))

--创建评论基本表
CREATE TABLE Comments
(CommentID INT IDENTITY(1,1) PRIMARY KEY,
DetailID CHAR(12),
comment TEXT,
date DATETIME,
FOREIGN KEY(DetailID)REFERENCES Orderdetail(DetailID))

--创建购物车基本表
CREATE TABLE Cart
(CartID CHAR(12) PRIMARY KEY,
UserID CHAR(8),
ProductID CHAR(12),
ProductQuantity INTEGER,
FOREIGN KEY(UserID)REFERENCES Users(UserID),
FOREIGN KEY(ProductID)REFERENCES Products(ProductID))

GO
```

## 4.2 视图语句

为方便查询评论，建立视图，实现代码如下：

```
use B2C;
-- 为commentid，userid，productid创建索引，建立comment视图，显示评价信息--
if exists(select * from sys.views where name='AllID_Comment')
--删除单个视图
drop view AllID_Comment;
go
create view AllID_Comment WITH SCHEMABINDING
as
select c.CommentID, c.DetailID, o.OrderID, o.UserID, od.ProductID
from dbo.Comments c,dbo.Orderdetail od, dbo.Orders o
where od.OrderID=o.OrderID and c.DetailID=od.DetailID;
go
create unique clustered index ic_commentid on AllID_Comment(CommentID);
go
create index ic_userid on AllID_Comment(UserId);
go
create index ic_productid on AllID_Comment(ProductId);
go
```

## 4.3 从操作出发设计用户子模式

### 4.3.1 会员部分

（1） 在用户首次登陆时，邀请注册为会员

```
declare @UserID char(8),@UserName varchar(16),@gender char(1),@birthday varchar(8),@tel
char(11),@UserPwd varchar(18),@province varchar(15)
insert into Users (UserID,UserName,gender,birthday,tel,Userpwd,province)
values(@UserID,@UserName,@gender,@birthday,@tel,@Userpwd,@province)
```

（2） 查询符合条件的商品

```
declare @keyword varchar(8), @shopname varchar(8)
select p.productid, p.name,p.unitprice,p.introduction,p.inventory,p.image,b.shopname
from products p,Businesses b where p.BusinessID = b.BusinessID and name like '%'+@keyword+
'%' and b.ShopName like '%' + @shopname + '%'
```

（3） 对符合条件的商品排名，选取最便宜的前五个

```
select t.*
```

```
from
(select p.productid,
p.name,p.unitprice,p.introduction,p.inventory,p.image,b.shopname,rank() over(order by
p.unitprice) as rank_num
from products p,Businesses b
where p.BusinessID = b.BusinessID and name like '%" + keyword + "%' and b.ShopName like '%"
+ shopname + "%')
t where rank_num < 6
```

（4） 选取所有商品

```
select p.productid, p.name,p.unitprice,p.introduction,p.inventory,p.image,b.shopname
from products p,Businesses b
where p.BusinessID = b.BusinessID
```

## 4.3.2 订单部分

（1） 显示用户的所有订单细节

```
select o.OrderDate,o.OrderID,od.DetailID,p.name,o.Address
FROM Orders o,Orderdetail od, Products p
where p.ProductID = od.ProductID and od.OrderID = o.OrderID and o.UserID=@userid
```

（2） 按商品名称，显示特定用户购买的次数的排名

```
select name,BusinessID,count(name) as frequency,rank() over(order by count(name) desc) as
rank from (SELECT p.name, o.BusinessID
FROM Orders o, Orderdetail od, Products p where p.ProductID = od.ProductID and od.OrderID
= o.OrderID and o.UserID = @userid) as a
group by name,BusinessID
```

（3） 购物车买单

```
exec check_out_final '00000004','street'
select UserID,ProductID,ProductQuantity from Cart where UserID='00000004'
```

## 4.3.3 评价部分

（1） 为 commentid，userid，productid 创建索引，建立 comment 视图，显示评价信息

```
if exists(select * from sys.views where name='AllID_Comment')
--删除单个视图
drop view AllID_Comment;
go
create view AllID_Comment WITH SCHEMABINDING
as
```

```
select c.CommentID, c.DetailID, o.OrderID, o.UserID, od.ProductID
from dbo.Comments c,dbo.Orderdetail od, dbo.Orders o
where od.OrderID=o.OrderID and c.DetailID=od.DetailID;
go
create unique clustered index ic_commentid on AllID_Comment(CommentID);
go
create index ic_userid on AllID_Comment(UserId);
go
create index ic_productid on AllID_Comment(ProductId);
go
```

（2） 给定用户，显示所有评价

```
declare @userid char(8)
set @userid = '00000004'
select c.date,ac.UserID,c.comment from AllID_Comment ac,dbo.Comments c
where ac.CommentID=c.CommentID and ac.UserID=@userid;
```

（3） 建立 productid 的索引

```
create index ip_businessid on Products(Businessid);
go
```

（4） 给定商品，显示所有评价

```
declare @productid char(12)
set @productid = '000000000001'
select c.CommentID, ac.DetailID, c.comment, c.date from AllID_Comment ac,dbo.Comments c
where ac.CommentID=c.CommentID and ac.ProductID=@productid;
```

## 4.3.4 商家部分

（1） 显示给定商家的最热卖的三个商品

```
if exists(select * from sys.procedures where name='Business_top3')
drop procedure dbo.Business_top3;
go
create procedure Business_top3
@businessid char(8)
as
select * from
(select BusinessID, ProductID, sum(quantity) sum_quantity,
    rank() over (partition by BusinessID order by sum(quantity) desc) as ranks
    from Orders, Orderdetail
    where Orders.OrderID = Orderdetail.OrderID
    group by BusinessID, ProductID
    ) as t1
where ranks <= 3 and BusinessID=@businessid;
```

```
go
```

（2） 显示商家总销售额

```
if exists(select * from sys.procedures where name='Business_amout_year')
drop procedure dbo.Business_amout_year;
go
create procedure Business_amout_year
@year int,@businessid char(8)
as
select sum(amount) Sum
from Orders
where OrderDate between CONCAT(@year, '-01-01') and CONCAT(@year, '-12-31') and
BusinessID=@businessid;
go
```

（3） 查询语句：订单，评价

```
declare @businessid char(8)
select * from orders where businessid = @businessid;
go
declare @businessid char(8)
select * from products where businessid = @businessid;
go
```

## 4.3.5 管理部分

（1） 显示每个用户购买最多的三个商品

```
if exists(select * from sys.procedures where name='User_purchasemost')
drop procedure dbo.User_purchasemost;
go
create procedure User_purchasemost
as
select * from
    (select UserID, ProductID, sum(quantity) Maxq,
    rank() over (partition by UserID order by sum(quantity) desc) as ranks
    from Orders, Orderdetail
    where Orders.OrderID = Orderdetail.OrderID
    group by UserID, ProductID
    ) as t1
where ranks <= 1;
go
```

（2） 显示每个省的会员购买额信息

```
if exists(select * from sys.procedures where name='Province_detail')
drop procedure dbo.Province_detail;
```

```
go
create procedure Province_detail
as
select province, avg(UserSum) Avg_sales_amount, max(UserSum) Max_sales_amount, min(UserSum)
Min_sales_amount
from (
select province, Users.UserID, sum(amount) UserSum
from dbo.Users, dbo.Orders
where Users.UserID = Orders.UserID
group by province, Users.UserID
) as user_sum_province
group by province
order by Avg_sales_amount desc;
go
```

## 4.4 触发器

1. 在客户购买商品创建订单后自动更改库存：当库存数量小于订单数量时提示“库存不足”同时回滚；当库存数量大于订单数量时，库存数量自动更改。

```
GO
CREATE TRIGGER tgr_orders --创建订单联系库存的触发器
ON Orderdetail
FOR INSERT
AS
DECLARE @DetailID CHAR(12),@OrderID CHAR(12),@ProductID CHAR(12),@unitprice
REAL,@quantity INTEGER;
SELECT
@DetailID=DetailID,@OrderID=OrderID,@ProductID=ProductID,@unitprice=unitprice,@quantity
=quantity
FROM INSERTED;
  IF((SELECT inventory FROM Products WHERE @ProductID=ProductID)>=@quantity)
  BEGIN
    UPDATE Products
    SET inventory=inventory-@quantity
    where ProductID=@ProductID;
  END
  ELSE
  BEGIN
    PRINT('库存不足！')
    ROLLBACK
  END
GO
```

2. 在商家更改库存时，若输入小于零的数，提示修改失败，同时回滚。

```
GO
CREATE TRIGGER update_inventory --创建更改库存的触发器
ON Products
FOR update
AS
BEGIN
 declare @newinv int
 select @newinv = inventory from inserted;
 if (@newinv <0)
  print('修改失败！')
  rollback
END
GO
```

## 4.5 购买存储过程——并发过程及订单和订单明细存储过程

购买存储过程分为两部分，一是判断商品库存是否满足购买要求，若不满足则终止购买。若满足则进入下一步，修改商品库存并生成订单和相应的订单明细。

**并发过程：**对于多个买家同时进行购买操作的情况，我们设立一个事务，先读后写。在读取库存的时候，对商品表的对应列加行锁，然后检测库存是否符合要求，若符合则修改库存进行购买，之后结束事务，设置延迟时间，若有不同买家同时购买同一个商品，其中一个买家需要等待另一个买家购买完成后一段时间继续购买。

**订单及订单明细存储过程：**通过在存储过程中，定义临时购物车表，在结账时，购物车中同一个商家的商品，获得相同的订单号，其中不同的商品对应不同的订单细节号，从而实现订单分流和捆绑。

```
--create random string----
CREATE VIEW v_rand
AS
SELECT RAND() AS val;
go
CREATE OR ALTER FUNCTION rand_num(@n INT)
RETURNS INT
AS BEGIN
    SELECT @n=@n * val FROM v_rand
    RETURN @n
END;
go
CREATE OR ALTER FUNCTION random_string(
  @num INT,
  @chars VARCHAR(1024) =
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
) RETURNS VARCHAR(1024)
```

```
AS
BEGIN
  DECLARE @res_str VARCHAR(1024) = ''
  DECLARE @i INT=0

  WHILE(@i<@num)BEGIN
        SET @res_str = @res_str + SUBSTRING(@chars, FLOOR(dbo.rand_num(len(@chars))) + 1,
1)
        SET @i = @i + 1
  END

  RETURN @res_str
END;
go
--create type table----
if exists(select 1 from sys.types where name='Cart_MultiID')
    drop type dbo.Cart_MultiID
go
create type dbo.Cart_MultiID as table
(
CartID char(12) null,
UserID char(8) not null,
ProductID char(12) not null,
ProductQuantity int not null,
new_inventory int null default -1,  --new inventory
amount real default null,
BusinessID char(8) default null,
OrderID char(12) default null,
unitprice real default null
);
go

----Procedure Check inventory----
if exists(select * from sys.procedures where name='Check_inventory')
drop procedure dbo.Check_inventory;
go
create procedure Check_inventory
(@Carts_temp as Cart_MultiID readonly)
as
begin
    select ProductID
    from @Carts_temp
    where new_inventory < 0;
end
```

```
go

select * from Cart

----Procedure Buy products----
if exists(select * from sys.procedures where name='Buy')
drop procedure dbo.Buy;
go
create procedure Buy
(@Carts_temp as Cart_MultiID readonly, @userid as char(8), @addr as char(100))
as
begin
declare @orderdate_temp Date=Convert(varchar(20), getdate(),101)
--generate Orders
begin
    insert into Orders(OrderID, BusinessID, amount, UserID, OrderDate, Address)
    select distinct OrderID, BusinessID, amount, UserID,  @orderdate_temp, @addr
    from @Carts_temp
end

--generate Orderdetails
begin
    insert into Orderdetail
    select dbo.random_string(12, default), OrderID, ProductID, unitprice, ProductQuantity
    from @Carts_temp
end

--update Products.inventory
begin
    update Products
    set inventory = ctemp.new_inventory
    from Products, @Carts_temp ctemp
    where Products.ProductID = ctemp.ProductID;
end
end

go
--final procedure to check out
if exists(select * from sys.procedures where name='check_out_final')
drop procedure dbo.check_out_final;
go
create procedure check_out_final
(@UserID char(8), @Address char(100))
as
```

```
begin
declare @Carts_temp as dbo.Cart_MultiID
insert into @Carts_temp (UserID,ProductID,ProductQuantity) (select
UserID,ProductID,ProductQuantity from Cart where UserID=@UserID)
select * from @Carts_temp

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
begin tran;
update @Carts_temp
set new_inventory = Products.inventory-ProductQuantity, unitprice=Products.unitprice,
BusinessID=Products.BusinessID
from @Carts_temp ctemp, Products with(updlock, holdlock)
where ctemp.ProductID = Products.ProductID;

with temp
as (select BusinessID, sum(ctemp.ProductQuantity * ctemp.unitprice) as amount
from @Carts_temp ctemp
group by ctemp.BusinessID)

update @Carts_temp
set amount = temp.amount
from temp, @Carts_temp ctemp
where ctemp.BusinessID = temp.BusinessID;

declare @idtemp table(BusinessID char(8), OrderID char(12))
insert into @idtemp(BusinessID) select distinct BusinessID from @Carts_temp
update @idtemp
set OrderID = dbo.random_string(12, default)

update @Carts_temp
set OrderID = idtemp.OrderID
from @idtemp idtemp, @Carts_temp ctemp
where ctemp.BusinessID = idtemp.BusinessID;

begin
if not exists(select ProductID from @Carts_temp where new_inventory < 0)
    begin
        exec Buy @Carts_temp,@UserID,@Address;
    end
select ProductID from @Carts_temp where new_inventory < 0;
waitfor delay '00:00:30';
commit tran;
end
```

```
end

-- example
exec check_out_final '00000004','street'
select UserID,ProductID,ProductQuantity from Cart where UserID='00000004';

go
```

# 五 数据准备

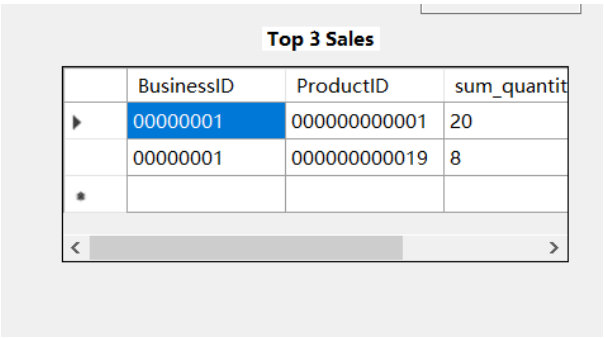
系统测试数据中商品、地址、介绍、评价大部分为手动生成，用户编号、商家编号等编号则为利用 EXCEL 自动填充的编号，既让数据生成具有一定效率，又保证了数据的真实性和多样性。

# 六 分析性查询

以下图例均为在购物系统图形化界面查询得到的结果。

**1. 显示每个商家最热卖的三个商品**

```
if exists(select * from sys.procedures where name='Business_top3')
drop procedure dbo.Business_top3;
go
create procedure Business_top3
as
select * from
(select BusinessID, ProductID, sum(quantity) sum_quantity,
rank() over (partition by BusinessID order by sum(quantity) desc) as ranks
from Orders, Orderdetail
where Orders.OrderID = Orderdetail.OrderID
group by BusinessID, ProductID
) as t1
where ranks <= 3;
go
```



**2. 给定一个商品，显示售卖此商品价格最低的 5 个商家。**

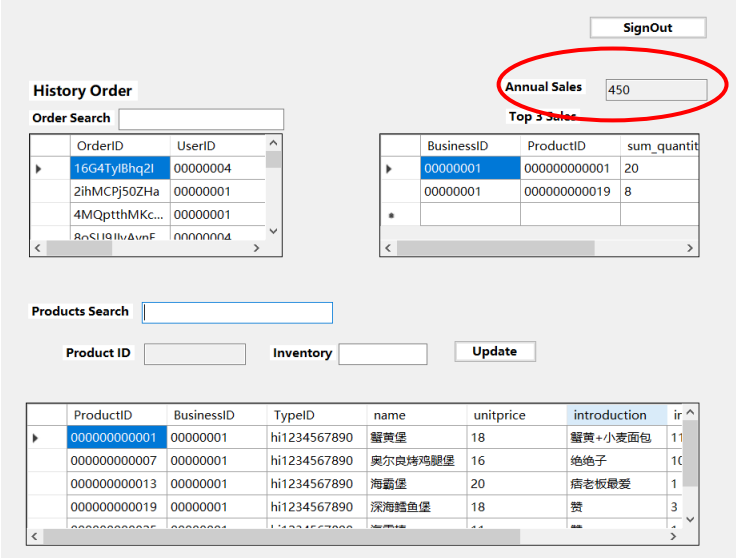
```
if exists(select * from sys.procedures where name='Product_price_low5')
drop procedure dbo.Product_price_low5;
go
create procedure Product_price_low5
@name varchar(18)
as
select top 5 BusinessID
from Products
where @name=name
order by unitprice
go
```


| price | introduction | inventory | image | shopname | rank_num |
|---|---|---|---|---|---|
| | 好喝 | 6 | "<a href=""ht... | 水川石 ... | 1 |
| | 好喝 | 9 | "<a href=""ht... | 当代 ... | 1 |
| | 好喝 | 9 | "<a href=""ht... | 双安 ... | 1 |
| | 好喝 | 0 | "<a href=""ht... | 华宇 ... | 1 |
| | 好喝 | 5 | "<a href=""ht... | 集天 ... | 5 |

**3. 显示每个商家的年销售总额。**

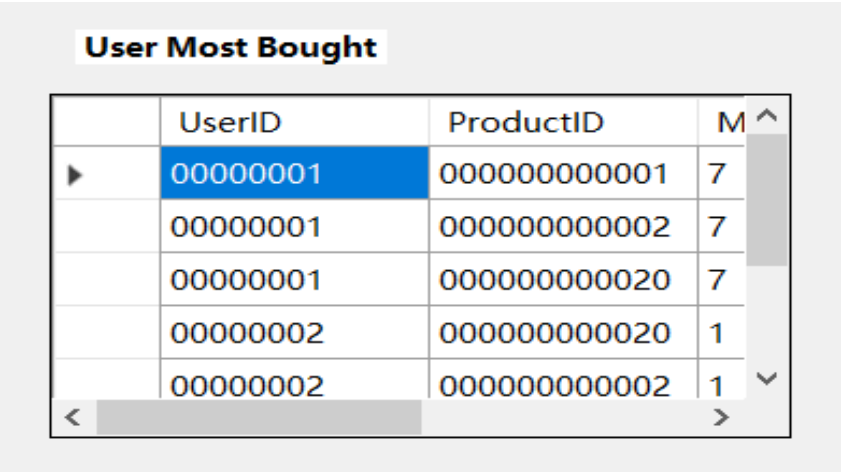
```
if exists(select * from sys.procedures where name='Business_amout_year')
drop procedure dbo.Business_amout_year;
go
create procedure Business_amout_year
@year int
as
select BusinessID, sum(amount) Sum
from Orders
where OrderDate between CONCAT(@year, '-01-01') and CONCAT(@year, '-12-31')
group by BusinessID;
```

go



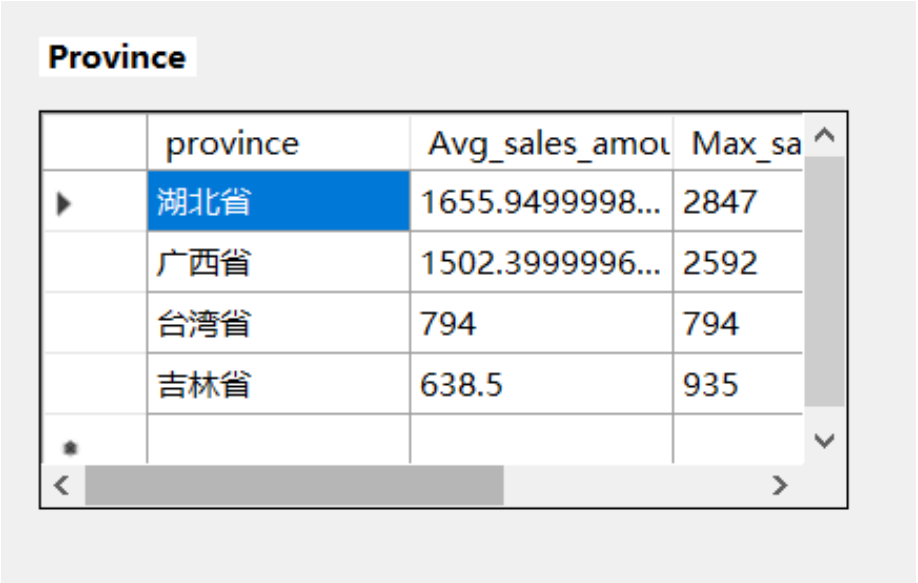

4. 显示每个会员购买次数最多的三个商品

```
if exists(select * from sys.procedures where name='User_purchasemost')
drop procedure dbo.User_purchasemost;
go
create procedure User_purchasemost
as
select * from
(select UserID, ProductID, sum(quantity) Maxq,
rank() over (partition by UserID order by sum(quantity) desc) as ranks
from Orders, Orderdetail
where Orders.OrderID = Orderdetail.OrderID
  group by UserID, ProductID
) as t1
where ranks <= 3;
go
```




**5. 显示每个省份会员的平均消费额、最大消费额和最小消费额，并按平均消费额降序排列。**

```
if exists(select * from sys.procedures where name='Provinc_detail')
drop procedure dbo.Province_detail;
go
create procedure Province_detail
as
select province, avg(UserSum) Avga, max(UserSum) Maxa, min(UserSum) Mina
from (
select province, Users.UserID, sum(amount) UserSum
from dbo.Users, dbo.Orders
where Users.UserID = Orders.UserID
group by province, Users.UserID
) as user_sum_province
group by province
order by Avga desc;
go
```



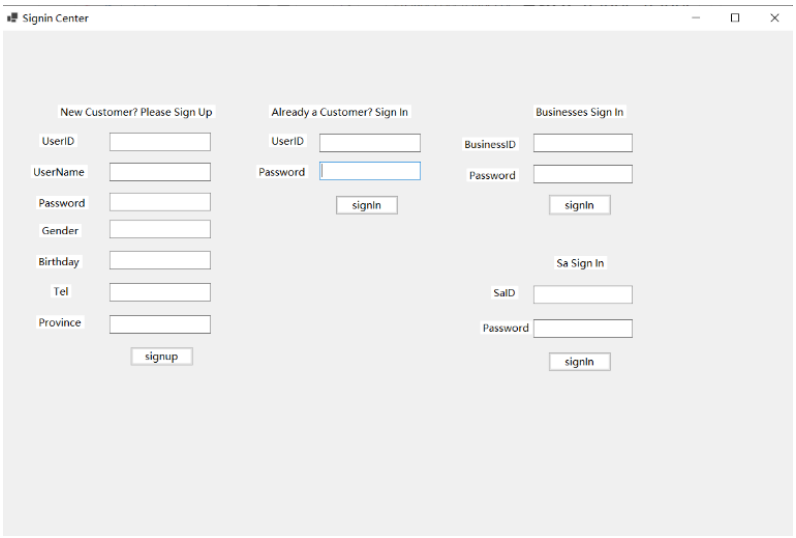

# 七 图形化界面

本购物系统使用 C#语言在 visual studio 的环境下连接 SQL 数据库设计用户界面。

**1. 登陆注册界面**

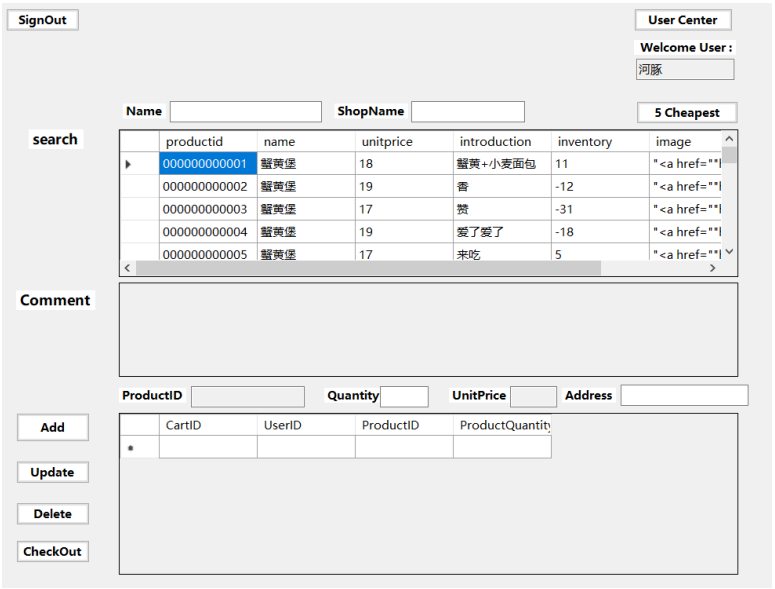
在登陆时，首次使用的用户可以选择注册成为会员，有账号的用户可以根据自己是会员、商家、管理员身份输入账号和密码进行登录操作。

注册需要填写账号、姓名、密码、性别、生日、电话、省份信息，与数据库中的 Users 表对应。注册成功后会员信息会插入到 Users 表中。

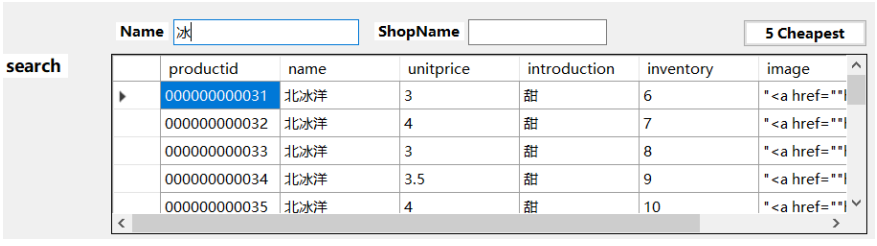



## 2. 商城界面

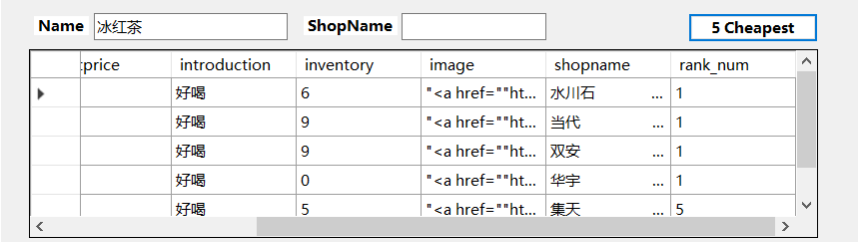
以会员身份登陆系统，能够登陆后显示商城界面如下。




（1）商品搜索：输入商品名称，商店名称，获得对应商品。

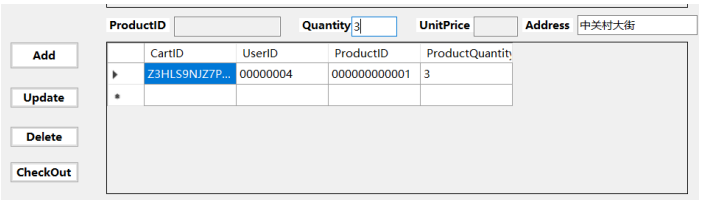


（2）商品排名：点击“5 Cheapest”按钮，得到符合搜索条件的最便宜的 5 件商品。

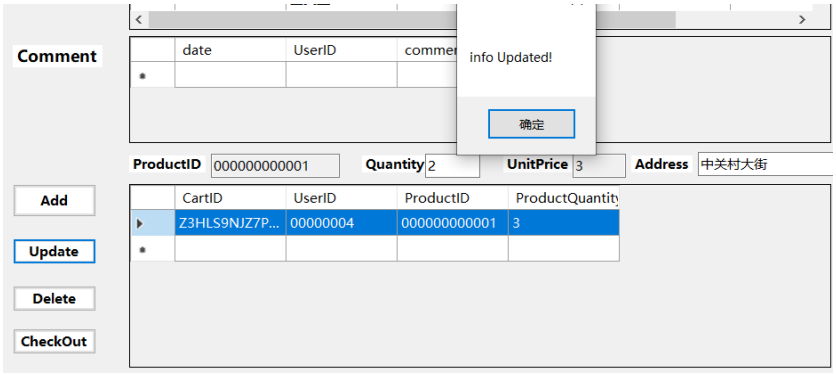



（3）评价显示：点击对应的商品，阅览商品评价。

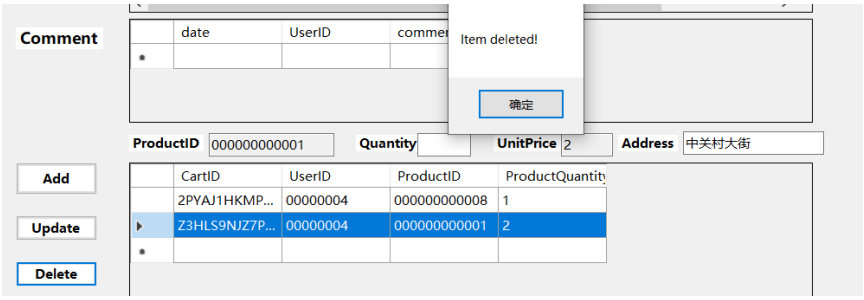

（4）将商品添加至购物车：点击对应商品，给定购买量、地址，点击“Add”按钮添加进购物车。



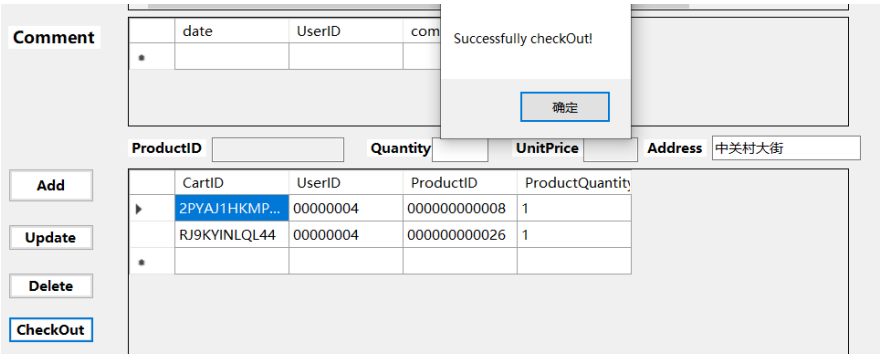

（5）修改购物车：选中购物车需要修改的商品，填写修改后的购买量或修改后的地址，点击“Update”完成修改。



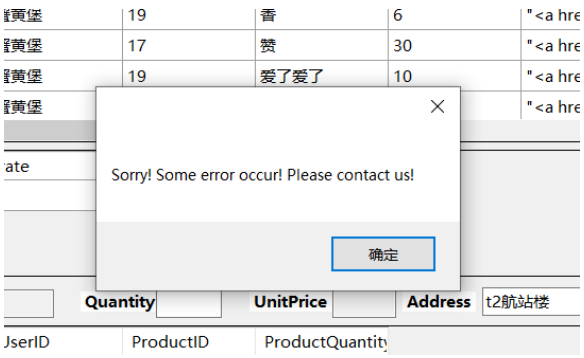

（5）删除商品：选中购物车中不需要的商品，点击“Delete”删除。



（6）购物车买单：点击“CheckOut”按钮，订单自动按照不同的商家分配，单个商品对应一个订单细节号。购买成功会有提示提示界面跳出。

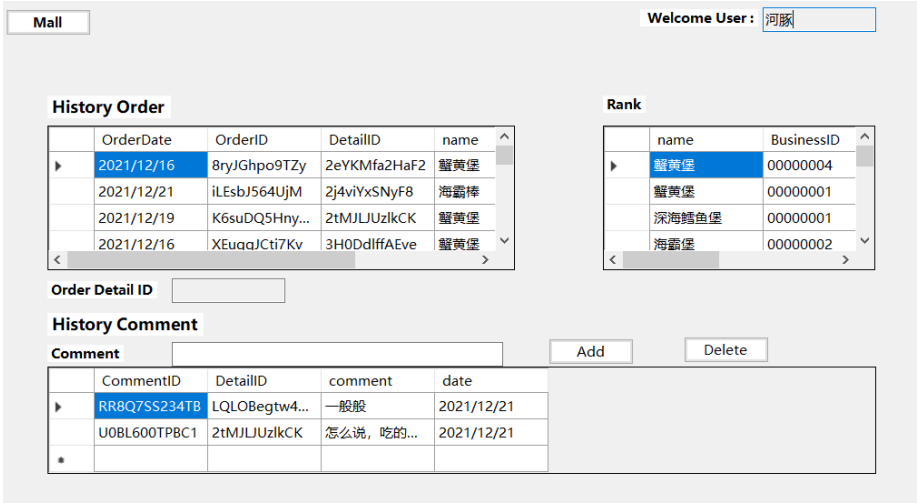



若库存不足，则会无法购买，提示出错。

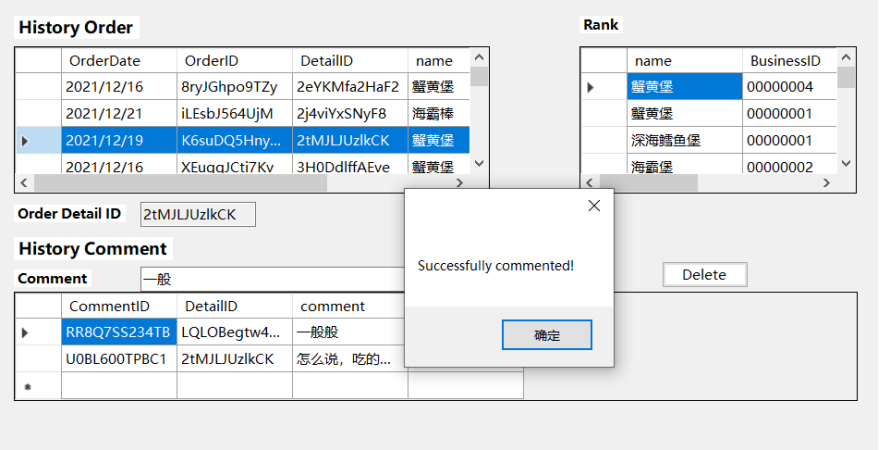



### 3. 用户管理中心界面

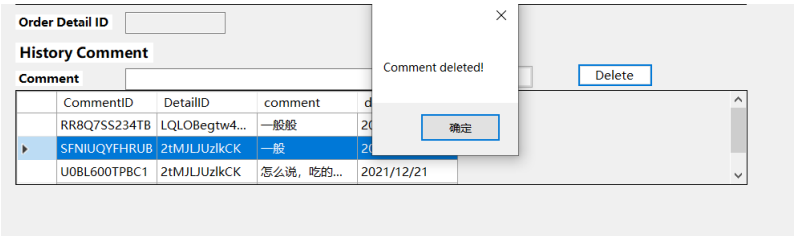
顾客点击商城界面右上角的“User Center”按钮可进入用户中心界面。在用户中心会包含该会员购买的历史订单记录，商品购买频率排名，可以商品评价进行相关操作。点击左上角“Mall”按钮可返回商城界面。

（1）填写评价：点击历史订单，输入评价，点击“Add”添加。

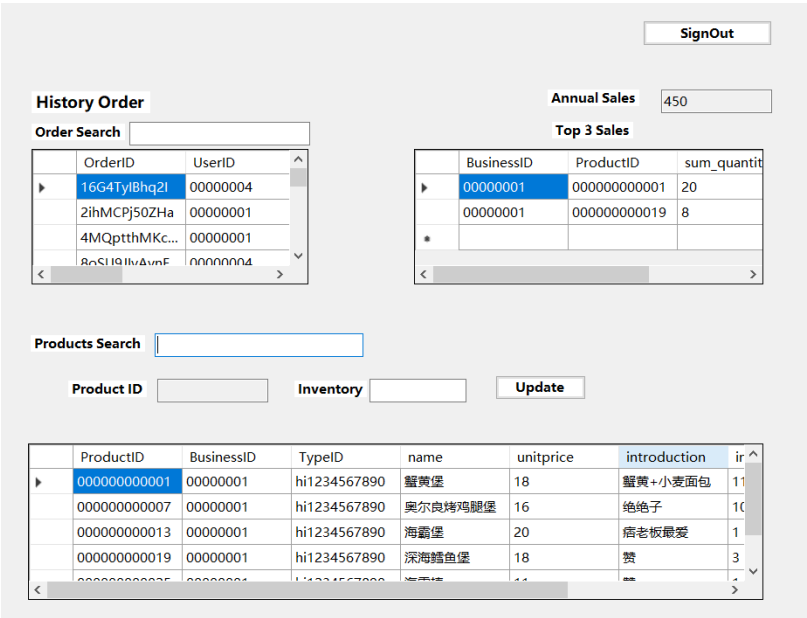



（2）删除评价：选择历史评价中的某一条，点击“Delete”删除。

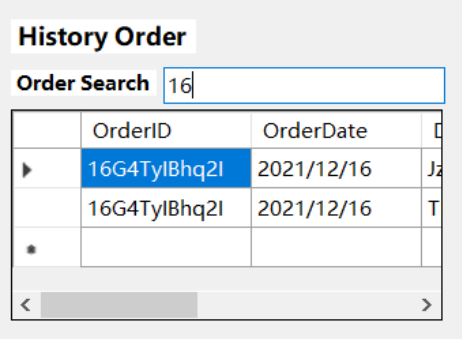



## 4. 商家界面

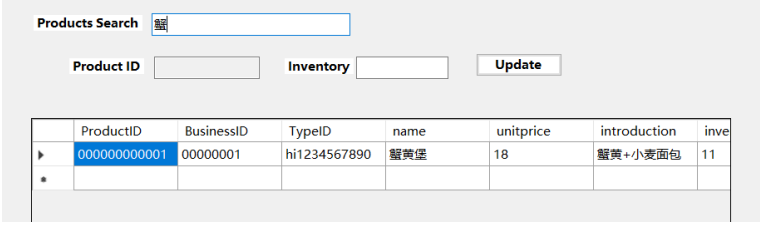
以商家身份登陆后，显示商家界面。界面包括历史订单查询，显示热卖商品，显示年收益，查询商品，更改库存功能。



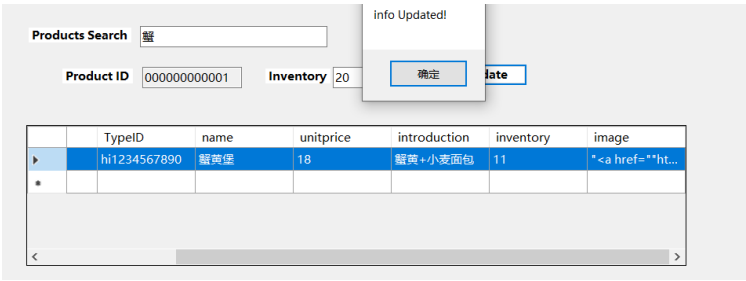
（1）查询本店订单：填入订单信息，查询订单。



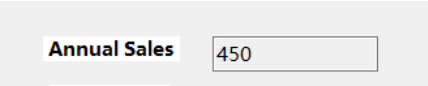

（2）查询商品：填入信息，查询本店对应商品。




（3）更新库存：选中需要修改的商品，填入库存，点击“Update”更新。

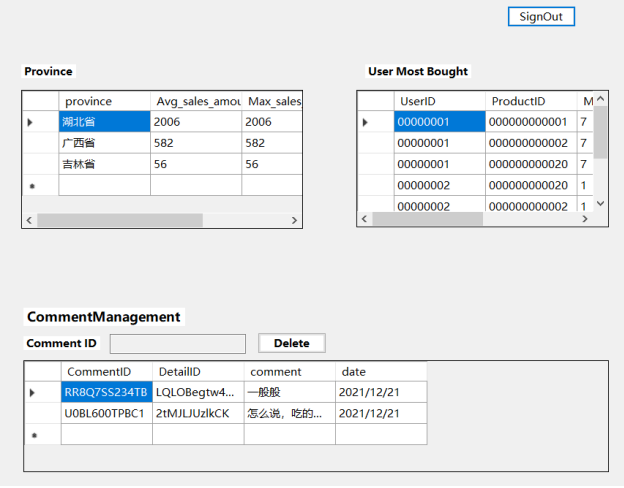



（4）年收益显示

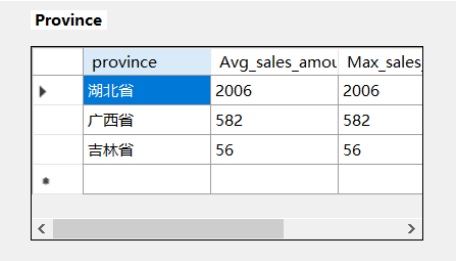



## 5. 管理员界面

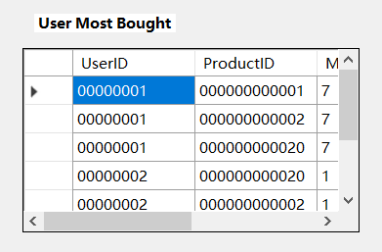
以管理员身份登陆后显示管理员界面。界面包括各省消费情况查看、各用户购买最多的三个商品查看和评价管理。



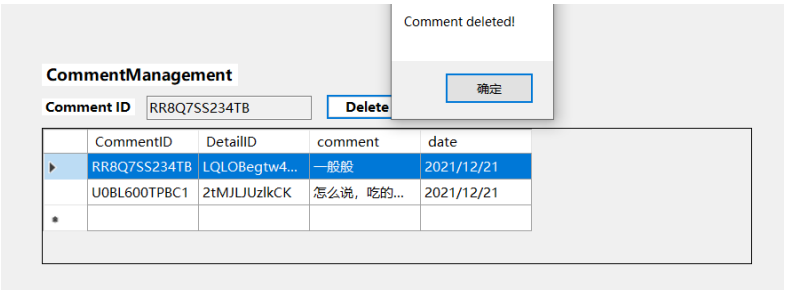
（1）显示每个省的会员的平均消费额，最高消费额，最低消费额，按照平均消费额排名。




（2）显示每个用户购买最多的三个商品。




（3）管理用户评论：显示所有用户评论，点击“Delete”按钮，删除用户不合规定的评论。



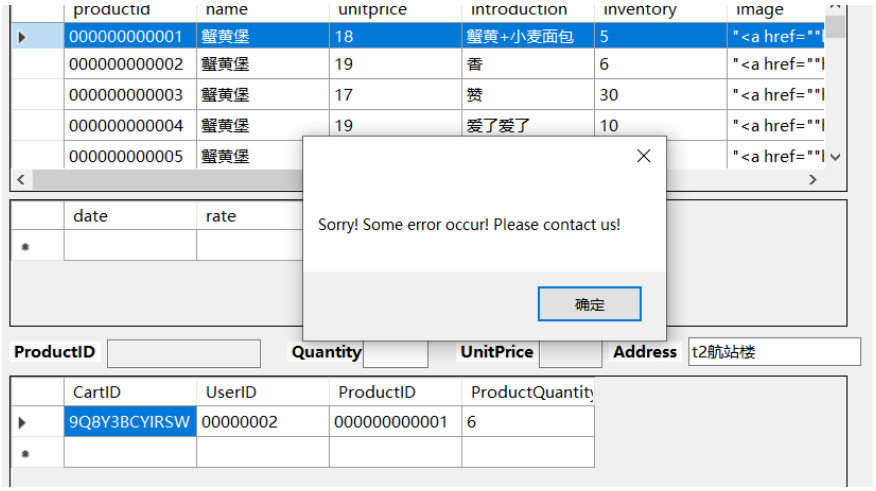

# 八 系统调试和测试

## 8.1 正常流程调试

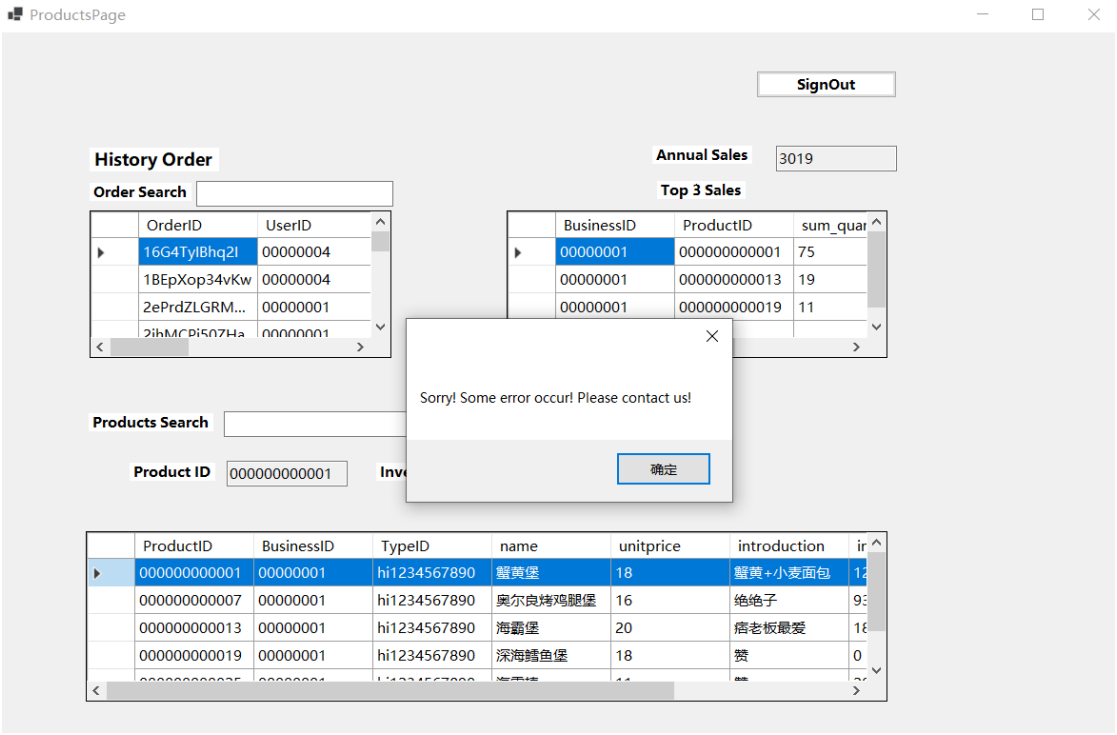
正常流程调试如上面展示的各个图片结果所示，系统正常功能均可正确运行。

## 8.2 异常测试模式

1. 会员购买商品时，输入大于库存的商品数量，系统提示出错，无法进行购买。

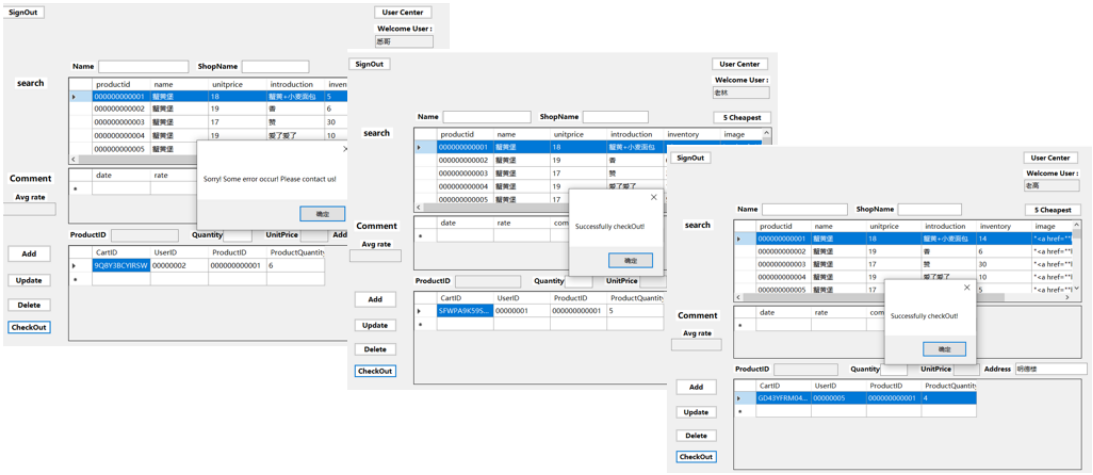



2. 商家更改库存时，无法输入除了自然数意外的其他数字（如负数）。




## 8.3 并发性测试

在测试时设置等待时间，打开多个窗口进行不同用户的同一个商品购买，可以看到系统允许多个用户在商城中同时进行购买操作，购买时间偏后的用户需要等待相应时间，在等待时间结束后会由于库存不足无法购买商品。等待时间只在测试时设置，最终版本没有等待时间。




# 九 系统评价

## 9.1 系统特色

**在图形化界面中加入分析性查询**。将分析性查询问题根据不同用户角色的需求融入图形化界面，无需进入数据库系统运行 SQL 语句进行查询，使购买平台在使用上更加方便、人性化，贴近现实使用需求。

## 9.2 不足与改进

1. 目前系统界面较为简单。各种信息通过表格展示，不够直观生动。未来可以加入商品图片、用户头像等的展示。

2. 在功能上仍有欠缺。在下单时只能对购物车中的所有商品进行一键全部下单，不能选择某几样商品下单，未来可加入选择框。

# 十 设计总结

## 10.1 困难及解决方案

**查询模块：**

1. 不清楚如何用 SQL 语句处理 topK 问题。

解决：使用 SQL Server 提供的开窗函数解决 topK 问题。

参考：https://blog.csdn.net/TomAndersen/article/details/106458042

2. 实际操作中发现有的查询任务需要根据输入特定值来输出查询结果。创建视图存放查询语句难以实现这一任务。
解决：改用 SQL Server 存储过程封装查询语句，通过传入参数来实现特定参数的查询。
参考：
https://docs.microsoft.com/zh-cn/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver15

3. 查询时由于查询对象同时存在：聚类函数、作为分类依据的列以及其他不包含聚类函数同时不作为分类依据的列，报错。
解决：使用 rank() over(partition by) 开窗函数解决。
参考：
① https://blog.csdn.net/u012071918/article/details/77246535
② https://www.cnblogs.com/lihaoyang/p/6756956.html

**评价部分的索引视图**
4. 在视图上创建索引（索引视图）时创建失败。
解决：
① 视图中的 SELECT 语句选择列表不能使用 * 或 table_name.* 语法指定列，要将列名全部显式给出。
② 视图不能包含 text、ntext 或 image 列。因此选择只取三个表的 ID 列进行连接。
③ 要先创建唯一的聚集索引后才能创建其他的非聚集索引。
参考：索引视图：
https://docs.microsoft.com/zh-cn/sql/relational-databases/views/create-indexed-views?view=sql-server-ver15

**Visual Studio 中向 sql server 传入参数**
5. 向 sql server 中传入参数报错。
解决：
① 传入搜索字符时，直接将搜索字符和 sql 语句合并，和 SqlConnection 一并，定义 SqlDataAdapter。
② 使用 SqlCommand，用 Parameters.AddWithValue( )方法传入参数。
参考：SqlClient 文档：
https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-6.0

**在 C#中创建全局变量**
6. C#中没有全局变量的概念，而购物系统需要全局变量以存储基本信息和函数。
解决：定义一个 Globals 类，通过静态变量来存放所有需要的全局变量，调用的时候通过 Globals 来调用即可。
参考：https://www.cnblogs.com/China3S/p/5179808.html

## 10.2 分工

1. 概念设计、逻辑设计——何暾；

2. 分析性查询——高一画；
3. 数据、实验报告——林哲伊；
4. 用户界面——郑语悉、谢文杰；
5. 存储过程——高一画；
6. 触发器——何暾、郑语悉。

## 10.3 收获与感悟

通过小组成员的共同努力，目前该 B2C 系统已具备基本的线上售卖商品及其他相关功能，但仍在数据库框架合理性、查询效率、测试数据完备性、高级并发控制等方面存在许多不足和问题，未来也仍然需要努力进行改良。

不得不承认的，完成本次大作业的过程让大家收获良多，一方面要用课堂上学习过的数据库系统相关的理论知识熟练地进行实践，另一方面又要对课堂上未涉及或涉及较少的页面设计方面的知识进行学习和运用，不但巩固了大家知识体系中已学的部分，也增加了新的部分，并锻炼了大家的学习能力和合作分工的能力。

# 参考文献

[1]王珊 萨师煊. (2015). 数据库系统概论(第 5 版). 高等教育出版社.