说明文档

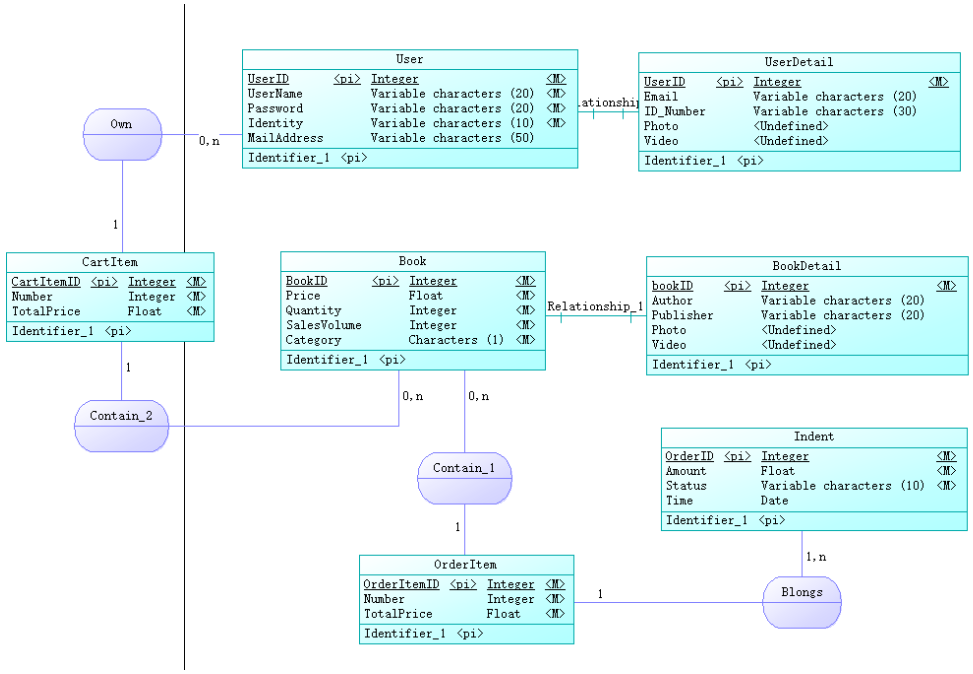
---- 王赟超
---- 5140379042

1. 配置
   （1） 将 data 文件夹中的 BookStore.sql 和 data.sql 导入数据库
   （2） 配置 bookstore_It3\src\applicationContext.xml 中的 dataScurce Bean
   （3） 访问 bookstore_It3 打开书店

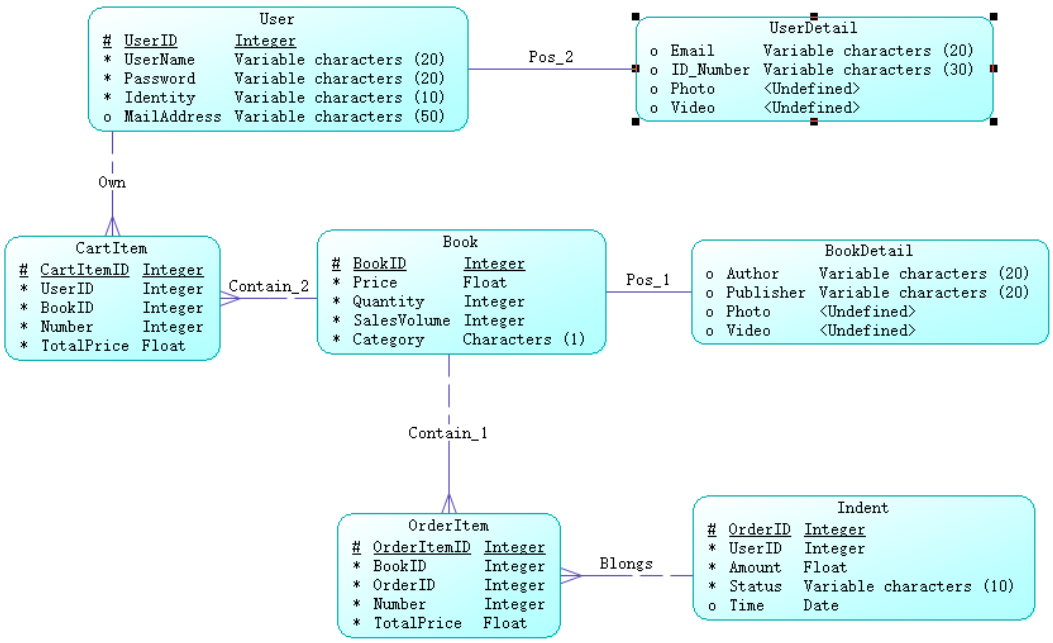
2. 实现说明
   （1） 在 index 页面，有登录、用户注册两个选项，没有管理员注册，只允许通过数据库管理员进行书店管理员的增加。
   （2） 可以用默认的用户名密码进行登陆，也可查看数据文件后用其他用户名进行登陆。
   （3） 用户在“全部图书”界面，可以进行书籍的查询，查看详情，加入购物车操作。查看详情功能通过 Ajax 实现，使用 JSON 格式进行数据传输。
   （4） 用户在购物车页面，可以进行商品的增减、删除、查看详情、以及选择一些条目生成订单。生成订单后，进入订单详情界面，可进行支付。若库存充足，则支付成功，支付后，书籍的销量会增加，书籍的库存会减少。若库存不足，则支付失败。
   （5） 用户在“我的订单”界面，可以查看订单、支付订单、确认收货、查看书籍详情。
   （6） 用户在“个人信息”界面可以查看个人信息，点击修改，可以修改个人信息。
   （7） 点击退出，可以退出登录并返回起始页面。
   （8） 管理员在“管理图书”界面可以进行书籍的查询、新增、删除或修改信息。
   （9） 管理员在“管理用户”界面可以进行用户的删除，考虑到隐私问题，管理员无法修改和查看用户具体信息。
   （10） 管理员在“订单管理”界面可以进行订单价格的修改（对于未支付的订单），可以进行发货，查看书籍详情。
   （11） 管理员在“个人信息”界面可以查看个人信息，点击修改，可以修改个人信息。
   （12） 点击退出，可以退出登录并返回起始页面。

3. 数据库设计

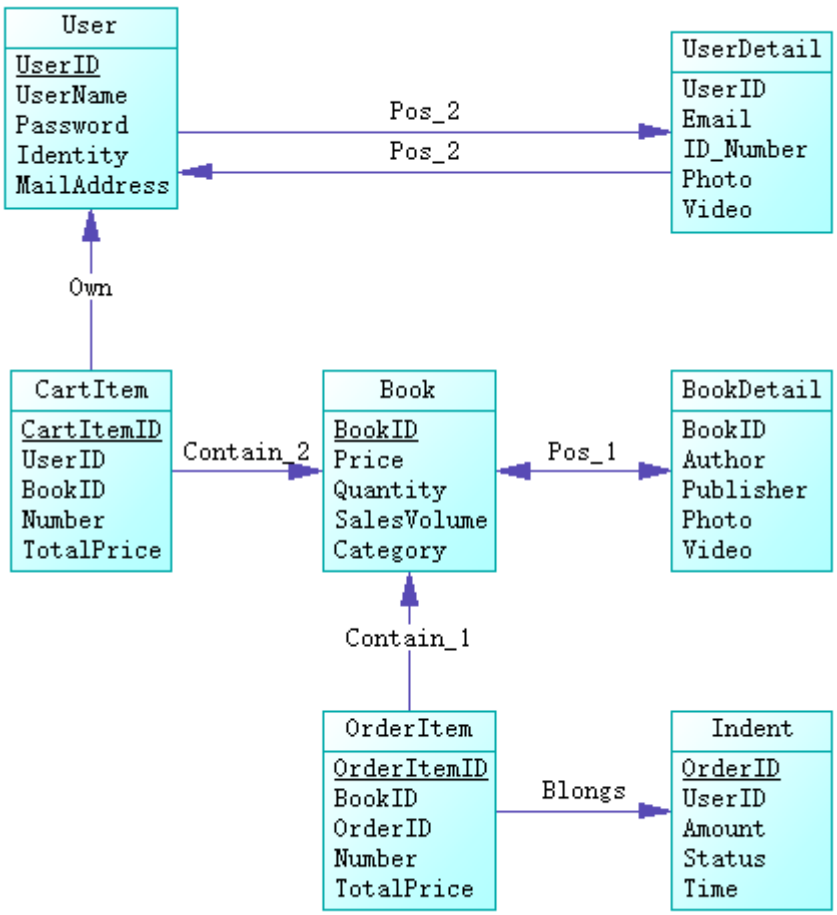
（1） CDM



（2） LDM



（3） PDM



（4） 范式分析

① User (UserID， UserName, Password, Identity, MailAddress)

UserID -> UserName, UserID -> Password, UserID -> Identity, UserID -> MailAddress, UserName -> UserID, UserName -> Password, UserName -> Identity, UserName -> MailAddress

结论 UserID 是候补码，UserName 也是候补码。

因此，此表符合 BCFN。

(说明，UserID 的存在是为了减小 User 表被 ref 时的消耗，UserName 的存在是为了方便用户记忆)

② UserDetail (UserID, Email, ID_Number, Photo, Video)

UserID -> Email, UserID -> ID_Number, UserID -> Photo, UserID -> Video

(前置说明：我们允许一个人注册多个账户，因此，ID_Number 和 Email 都不是候补码)

因此，此表符合 BCFN。

③ Book (BookID, Price, Quantity, SalesVolume, Category)

BookID -> Price, BookID -> Quantity, BookID -> SalesVolume, BookID -> Category
此表是符合 BCFN 的。

④ BookDetail (BookID, Author, Publisher, Photo, Video)

BookID -> Author, BookID -> Publisher, BookID -> Photo, BookID -> Video
此表是符合 BCFN 的。

⑤ Indent (OrderID, UserID, Amount, Status, Time)

OrderID -> UserID, OrderID -> Amount, OrderID -> Status, OrderID -> Time
此表是符合 BCFN 的。

⑥ OrderItem (OrderItemID, BookID, OrderID, Number, TotalPrice)

OrderItemID -> BookID, OrderItemID -> OrderID, OrderItemID -> Number, OrderItemID -> TotalPrice, BookID+OrderID -> OrderItemID, BookID+OrderID -> Number, BookID+OrderID -> TotalPrice, BookID+Number -> TotalPrice
结论，OrderItemID 和 OrderID+BookID 都是候补码。
但由于 BookID+Number -> TotalPrice 不符合第三范式。
因此，此表仅满足第二范式。
(说明：OrderItemID 存在的理由是多属性主码在使用时极其不便。
TotalPrice 存在的理由是，这个属性会经常被访问，如果每次都通过 BookID 查找到 Price 再与 Number 相乘，效率很差，所以这里做了一点冗余)

⑦ CartItem (CartItemID, UserID, BookID, Number, TotalPrice)

CartItemID -> UserID, CartItemID -> BookID, CartItemID -> Number, CartItemID -> TotalPrice, UserID+BookID -> CartItemID, UserID+BookID -> Number, UserID+BookID -> Number, UserID+BookID -> TotalPrice, BookID+Number -> TotalPrice
结论，CartItemID 和 UserID+BookID 都是候补码。
但由于 BookID+Number -> TotalPrice 不符合第三范式。
因此，此表仅满足第二范式。
(说明：OrderItemID 存在的理由是多属性主码在使用时极其不便。
TotalPrice 存在的理由是，这个属性会经常被访问，如果每次都通过 BookID 查找到 Price 再与 Number 相乘，效率很差，所以这里做了一点冗余)

（5） Trigger 设计

A. Trigger_1: 当 CartItem 中 Book 数发生变化时，更新 TotalPrice。

```
create trigger update_totalPrice_In_CartItem after update of Number on CartItem
    referencing new row as nrow
    for each row
        declare bookPrice int;

        select Price into bookPrice
        from Book
        where BookID = nrow.BookID;

        update CartItem
        set TotalPrice = bookPrice * nrow.Nunmber
        where CartItemID = nrow.CartItemID;

    end;
```

B. Trigger_2: 当订单状态由 unpay 变为 underivery 时，更新书的库存和销量

```
create trigger update_quantity_and_salesVolume after update of Status on Indent
    referencing new row as nrow
    referencing old row as orow
    for each row
        when(nrow.Status = "undelivery" and orow.Status = "unpay")
        begin
            declare cur cursor for
                select BookID, Number form OrderItem where OrderID = nrow.OrderID;

            declare bookID int;
            declare number int;

            open cur;
                repeat
                    fetch cur into bookID, number;
                    update Book
                    set Quantity = Quantity - number,
                        SalesVolume = SalesVolume + number
                    where BookID = bookID;
                end repeat;
            close cur;
        end;
    end;
```

C. Trigger_3: 当新建订单时，计算 Amount。

```
create trigger update_amount_of_order after insert on Indent
    referencing new row as nrow
    for each row
        declare cur cursor for
            select TotalPrice form OrderItem where OrderID = nrow.OrderID;

        declare totalPrice float;
        declare amount float;
        set amount = 0;
        open cur;
            repeat
                amount = amount + totalPrice;
            end repeat;
        close cur;

        update Indent
        set Amount = amount
        where OrderID = nrow.OrderID;
    end;
```

（6） 函数设计

A. Function_1: 检查一个订单中的所有书是否都具有足够的库存

```
create function check_quantity ( orderID int)
returns boolean
begin
    declare cur cursor for
        select BookID, Number form OrderItem where OrderID = orderID;

    declare ans boolean;
    declare quantity int;

    open cur;
        repeat
            fetch cur into bookID, number;
            select Quantity into quantity
            from Book
            where BookID = bookID;
            if quantity < number
            then ans = false
            end if;
        end repeat;
    close cur;
    return ans;
end;
```

B. Function_2: 查找包含特定书的订单

```
create function find_all_order_with_a_book (bookID int)
return table (OrderID int)
begin
    create temporary table temp (OrderID int);

    insert into temp
        select OrderID
        from OrderItem
        where BookID = bookID;

    return table temp;
end;
```

C. Function_3: 查找包含特定一类书的订单

```
create function find_all_order_with_a_book_category (category varchar(1))
return table (OrderID int)
begin
    create temporary table temp (OrderID int);

    declare cur cursor for
        select BookID, OrderID form OrderItem;
    declare bookID int;
    declare orderID int;
    declare bc varchar(1);

    open cur;
        repeat
            fetch cur into bookID, orderID;

            select Category into bc
            from Book
            where BookID = bookID;

            if(bc = category)
            then insert into temp orderID;
            end if
        end repeat;
    close cur;

    return table temp;
end;
```

D. Function_4: 查找一个特定用户的订单

```
create function find_all_order_of_a_customer (userID int)
return table (OrderID int)
begin
    create temporary table temp (OrderID int);

    insert into temp
        select OrderID
        from Order
        where UserID = userID;

    return table temp;
end;
```

E. 计算一组 Order 的总金额

```
create function total_amount (table orders ( OrderID int))
return sum float
begin
    declare cur cursor for orders;
    declare orderID int;
    declare amount float;
    declare sum float;

    set sum = 0;
    open cur;
        repeat
            fetch cur into orderID;
            select Amount into amount
            from Indent
            where OrderID = orderID;
            sum = sum + amount;
        end repeat;
    close cur;
    return sum;
end;
```