**说明文档**

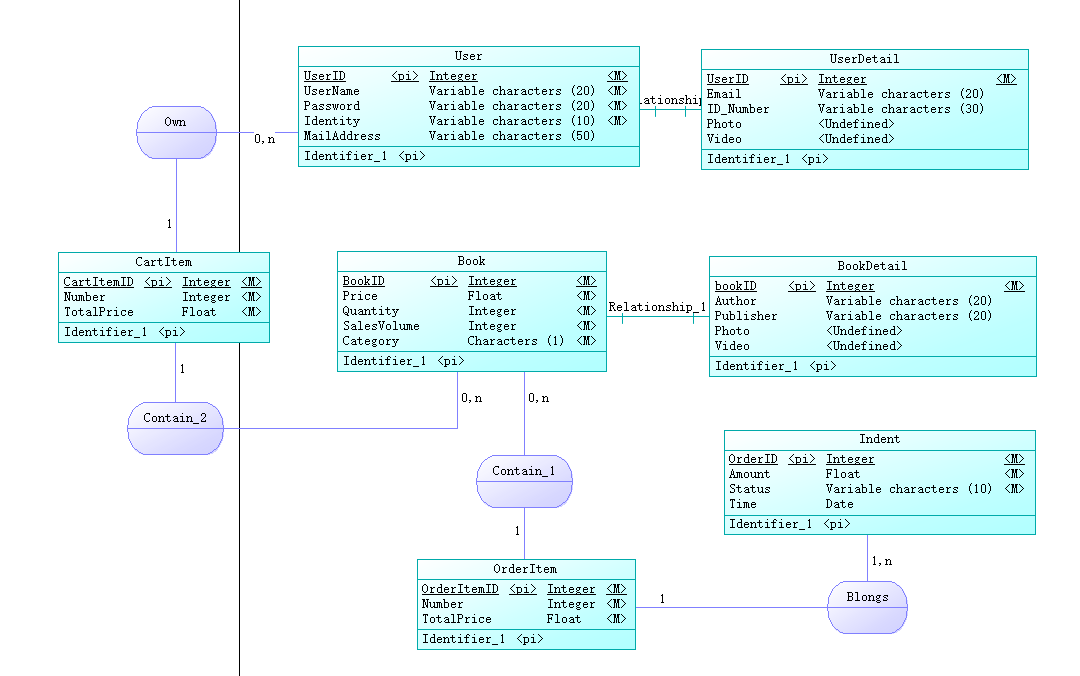
---- 王贇超

---- 5140379042

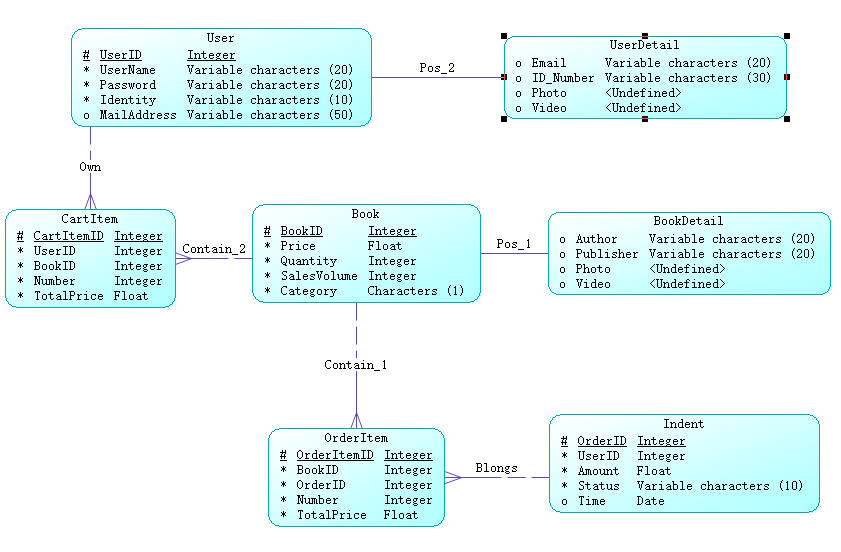
1. 配置
2. 将data文件夹中的BookStore.sql和data.sql导入数据库
3. 配置bookstore\_It3\src\applicationContext.xml中的dataScurce Bean
4. 访问bookstore\_It3打开书店
5. 实现说明
6. 在index页面，有登录、用户注册两个选项，没有管理员注册，只允许通过数据库管理员进行书店管理员的增加。
7. 可以用默认的用户名密码进行登陆，也可查看数据文件后用其他用户名进行登陆。
8. 用户在“全部图书”界面，可以进行书籍的查询，查看详情，加入购物车操作。

查看详情功能通过Ajax实现，使用JSON格式进行数据传输。

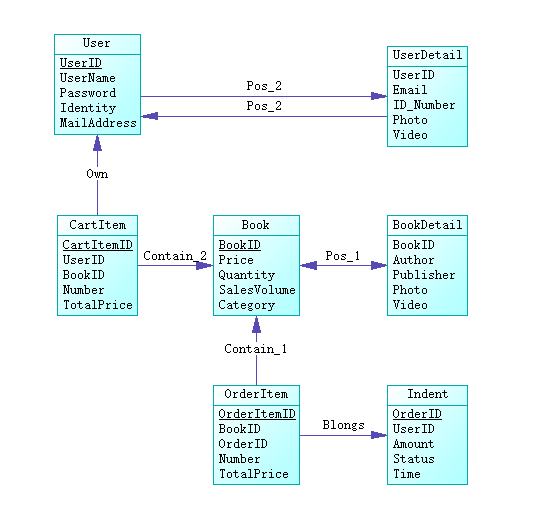
1. 用户在购物车页面，可以进行商品的增减、删除、查看详情、以及选择一些条目生成订单。生成订单后，进入订单详情界面，可进行支付。若库存充足，则支付成功，支付后，书籍的销量会增加，书籍的库存会减少。若库存不足，则支付失败。
2. 用户在“我的订单”界面，可以查看订单、支付订单、确认收货、查看书籍详情。
3. 用户在“个人信息”界面可以查看个人信息，点击修改，可以修改个人信息。
4. 点击退出，可以退出登录并返回起始页面。
5. 管理员在“管理图书”界面可以进行书籍的查询、新增、删除或修改信息。
6. 管理员在“管理用户”界面可以进行用户的删除，考虑到隐私问题，管理员无法修改和查看用户具体信息。
7. 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。
8. 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。
9. 点击退出，可以退出登录并返回起始页面。
10. 数据库设计
11. CDM



1. LDM



1. PDM



1. 范式分析

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)

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。

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

BookID -> Price, BookID -> Quantity, BookID -> SalesVolume, BookID -> Category

此表是符合BCFN的。

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

BookID -> Author, BookID -> Publisher, BookID -> Photo, BookID -> Video

此表是符合BCFN的。

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

OrderID -> UserID, OrderID -> Amount, OrderID -> Status, OrderID -> Time

此表是符合BCFN的。

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 )

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。

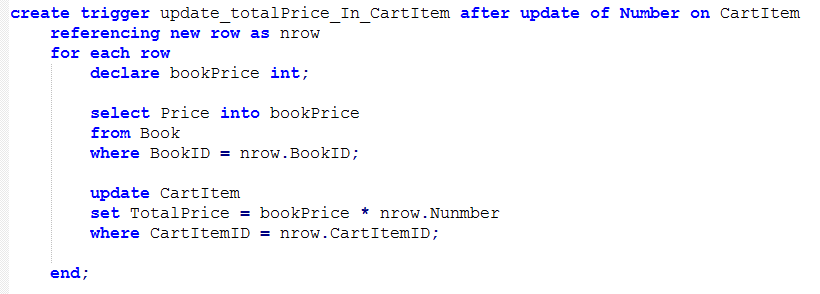
但由于 BookID+Number -> TotalPrice 不符合第三范式。

因此，此表仅满足第二范式。

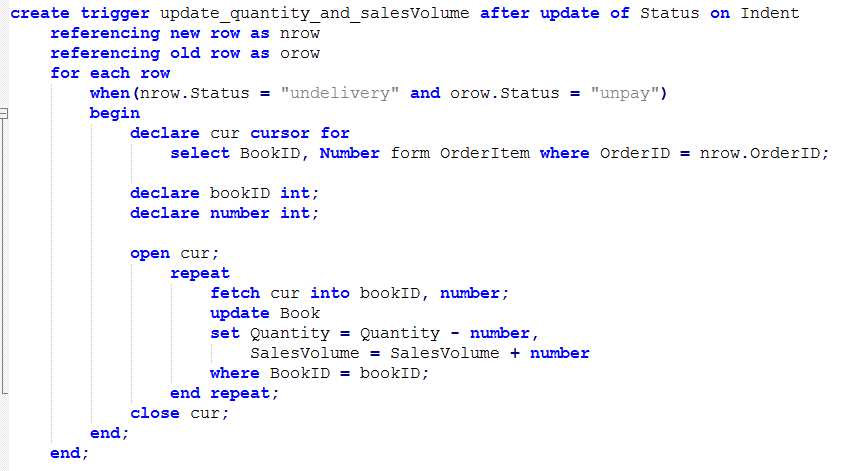
( 说明：OrderItemID存在的理由是多属性主码在使用时极其不便。

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 )

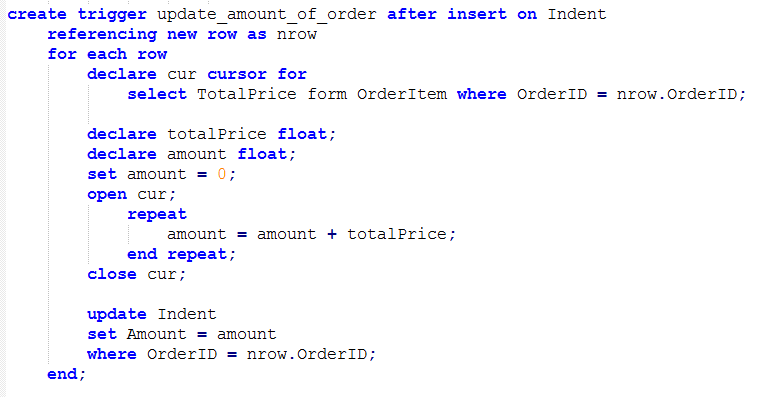
1. Trigger设计
2. Trigger\_1: 当CartItem中Book数发生变化时，更新TotalPrice。



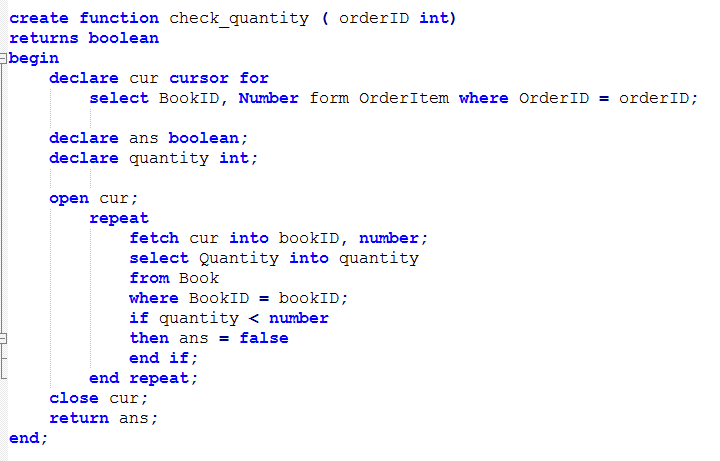
1. Trigger\_2: 当订单状态由unpay变为underivery时，更新书的库存和销量



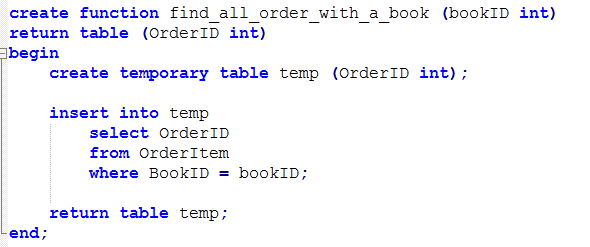
1. Trigger\_3: 当新建订单时，计算Amount。



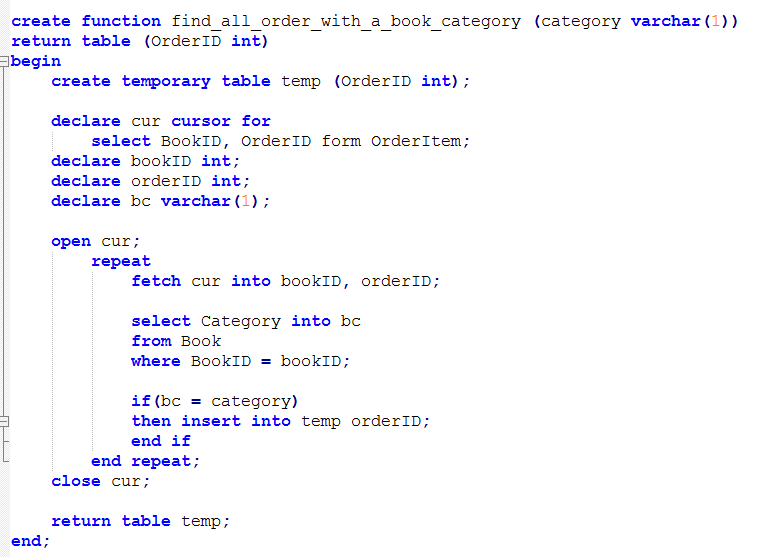
1. 函数设计
2. Function\_1: 检查一个订单中的所有书是否都具有足够的库存



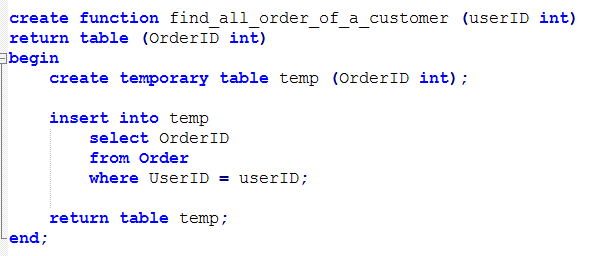
1. Function\_2: 查找包含特定书的订单



1. Function\_3: 查找包含特定一类书的订单



1. Function\_4: 查找一个特定用户的订单



1. 计算一组Order的总金额

