密级：_____

南昌大学

NANCHANG UNIVERSITY

学 士 学 位 论 文

THESIS OF BACHELOR

（2015—2019 年）

题　　目　布料批发管理系统的设计与实现

学　　院：软件学院　系　软件工程

专业班级：软件工程（卓越计划）152 班

学生姓名：周寒聿西　学号：8000115057

指导教师：欧阳皓　职称：副教授

南 昌 大 学

学士学位论文原创性申明

本人郑重申明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式表明。本人完全意识到本申明的法律后果由本人承担。

作者签名：　　日期：

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权南昌大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在　年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上相应方框内打“√”）

作者签名：　　日期：

导师签名：　　日期：

布料批发管理系统的设计与实现

专　　业：软件工程　　学　　号：8000115057
学生姓名：周寒聿西　　指导教师：　欧阳皓

摘要

Web 技术一直在朝着更复杂的业务逻辑，和更美观强大的显示交互效果的方向发展，虽然依然还存在网络传输速度限制、过于依赖服务器等缺点，但到现在 Web APP 已经越来越有取代胖客户端的趋势了。在应用软件方面，电子商务和企业资源管理方向的软件已经很成熟了，但是电商中更成熟的模式都是零售或是使用第三方平台的模式，前台线上销售与后台企业资源关系合为一体的、同时面向单个企业与其客户的结构还比较少见。

本选题是设计与实现一个布料批发管理 Web App，要求实现批发公司从采购到销售一系列业务的自动化，整合前台线上销售与后台企业资源管理的功能，实现企业信息的前后交流。

本选题使用 Java 语言和 SSM 框架进行开发，页面中使用网页局部刷新的 Ajax 技术，使用 Tomcat 服务器与 MySQL 数据库，实现一个在局域网内使用的布料批发管理系统。

关键词：Web App；电子商务；ERP 软件；布料批发；SSM；Ajax

The design and realization of cloth wholesale management system

Abstract

Web technology has been being developed to accomplish more complicate business logics, more beautiful and powerful pages as well as visible effects. Although there are still some disadvantages such as the limitation of internet speed or over depending on servers, Web apps are going to take the place of rich clients gradually. When it comes to utility software, those dealing with e-business and Enterprise Resource Plan are well-developed, but in e-business, the more mature patterns are those for retail and third-party platforms, by contrast, those combining online selling and ERP as well as facing a single company and its customers are fewer.

The design to be discussed in this article is a Web App about cloth wholesale management used in local area network. The requirements are realizing the automatable management of a series of wholesale operations from purchase to sales, transporting the information from selling in the front end to company management in the back end and combining the functions of sale and company management.

This design is going to be developed by Java, using SSM frame, Tomcat server and MySQL database, and Ajax technology is going to be used in the Web pages, developing a cloth wholesale management system used in local area network.

Keyword: Web App; e-business; ERP software; cloth wholesale; SSM; Ajax

目录

摘要 I
Abstract II
第一章 引言 1
1.1 Web 应用技术发展概况及主流技术比较 1
1.2 电子商务及商务管理软件概况 2
1.3 布料批发管理系统背景与前景 2
第二章 布料批发管理需求分析 4
2.1 主要特性、范围及局限性 4
2.2 技术要求 6
第三章 布料批发管理系统设计 7
3.1 功能模块 7
3.2 角色主要用例 8
3.2 具体用例 9
第四章 布料批发管理系统数据库设计 21
4.1 业务规则（Business Rule） 21
4.2 数据表设计 21
4.3 重点用例的数据流图 23
4.4 系统层次结构及数据类间关系 26
第五章 布料批发管理系统实现 29
5.1 系统目录层次及框架构建 29
5.2 系统实现 31
第六章 布料批发管理系统测试 44
6.1 测试范围及结果 44
6.2 重点功能测试截图 44
结论 49
参考文献 50
致谢 51
附录 52

第一章 引言

1.1 Web 应用技术发展概况及主流技术比较

互联网本质上是一种数字化的交流和交互的技术，自互联网出现之日起，人们的交流方式以及依赖于交流的各种日常生活和工作方式就发生了巨大的变化。

早期由于计算机硬件及传输设备的技术限制，仅仅能传送文字和图片这些数据量小的数据，那时的网页也仅仅能使用 HTML 展示数据和信息，交互效果也很弱，仅仅靠脚本没有办法完成复杂的业务交互，此时人们的交流局限性依旧还很大。

后来随着储存空间及网络传输设备和技术的发展，互联网可以传送音频视频等等数据量大的多媒体数据，网页的交互技术也进一步发展了，ASP 和 JavaEE 的出现使得网页可以使用三层架构来处理比较复杂的业务逻辑，Web 的用途开始变得多样起来。其中 JavaEE 规范因为依托于 Java 的可移植性，以及 servlet 技术更好的安全性和表示层与业务层更高的分离得到了广泛的认可，越来越多地被使用。后来各式各样的 Web 应用框架对使用 JavaEE 进行开发进行了进一步的封装，其中 spring 框架以其轻量级、容器式以及一站式的特点取得了优势，配合 struct 框架或者 springMVC 框架完成 MVC 架构，再加上数据层的 Hibernate 或者 MyBatis 框架，三层架构在 Web 应用上得到了最大化的使用。[1] 这使得 B/S 架构的 Web APP 逐渐可以替代 C/S 架构的胖客户端软件[2]，完成相当复杂的以前只适用于单机软件的业务逻辑，再加上 Web APP 不依赖于特定平台的特点，Web 应用技术已经越来越受到欢迎。

同时，网页显示技术现在也拥有了越来好的动态效果和设备兼容性，现在的网页局部刷新的 Ajax 技术、bootstrap 前端控件的应用，以及 HTML5 技术和响应式布局的出现，使得 Web APP 可以在多种多样的设备上表现出十分优越的可移植性和适应性，无需再像胖客户端软件一样，为不同的操作系统平台和不同分辨率的设备去编写不同版本的代码。

然而，由于 Web APP 归根结底依旧是基于互联网技术的，所以互联网技术的一些局限性依旧会给 Web APP 带来一些缺点。比如，虽然 Web 应用能处理的业务越来越复杂，但暂时还是无法替代像大型游戏这种规模的软件，因为这种软件会对网络传输速度和服务器的性能提出很高的要求。[3]随着网络传输技术的发展，在 5G 技术被成熟应用的时代，网络数据传输速度的问题会得到很好的解决，而分布式结构和云计算的发展，则可以让服务器性能的问题得到解决。

1.2 电子商务及商务管理软件概况

电子商务，正如字面所显示的意思，是依赖于电子设备和信息技术所进行的商业活动，除了与购买直接相关的活动的电子化、自动化，同时还涉及到物流配送、电子货币等等方面。我国的电子商务发展，从 B2C（business to customer）模式开始，后来逐渐变为以传统产业 B2B 模式（business to business）为主，现在惠及广大中小商家的 C2C 模式（customer to customer）也正迸发着强大的生命力。[4] 这三种形式的商业活动都十分活跃，其中最为突出的三个代表性平台就是 B2C 模式的京东，B2B 模式的阿里巴巴和 C2C 模式的淘宝。[5] 电子商务这种新型的交易方式，使得商业活动频繁跨地域进行更加方便，商业活动的各个参与者之间也能更加协调，将商业活动带入了一个数字化的新时代。本次选题选择的网上批发管理系统，就是非常典型的一个公司业务自动化与电子商务 B2C 模式的应用。

商务管理软件，又称企业资源计划（ERP ，Enterprise Resource Planning），是对公司业务流程及管理的电子化与自动化。[6] 早期，ERP 软件是对公司运营所产生的大量原始数据的记录查询和汇总，相当于一个电子数据库，后来，ERP 软件功能不断加强和完善，将公司的生产、物流、人力和财务管理功能全部包括起来。这形成了一个公司商务管理的封闭完善的系统，大大提高数据传递效果和业务周转效率，对业务的控制和记录也更为准确和方便，数据分析也更加省力和及时。

这两种系统独立来看，现在都已经发展得相当成熟，但对很多中小企业来说，他们想使用的两种软件的结合如今却还并不完善，线上销售系统与后台的 ERP 系统信息脱节，信息、财务、仓储物流不能做到数据自动共享统一，为此企业要进行重复冗余的工作去实现数据的传递。所以本次课题要进行设计的系统，就相当于是实现前端线上销售与后台 ERP 系统的对接整合。

1.3 布料批发管理系统背景与前景

随着互联网的普及和电子商务的兴起，生活中越来越多地商业活动都数字化了，足不出户，人们就可以手机充值、外卖点单，购买电影票、火车票、飞机票等等，网上购物业务范围越来越大，我国的零售业进入了一个新时代。

然而这些电子商务活动都是直接面对最终端的消费者个体的，与批发相关的商业活动虽然也有阿里巴巴这样的大型批发第三方平台，但对大中型批发商来说，本身企业内部管理系统无法嵌入到第三方平台中去。因此，对大中型企业来说，拥有一套自己的线上销售与内部管理相结合的系统就十分重要。本课题以布

料批发为切入点，要设计实现一个应用 B2C 模式的布料批发管理系统。

布料交易中货物种类繁多、管理困难，布料属性工艺复杂且品名色号编制不一，布料价格随着交易量的变化有梯度变化，仓库调度多且易出错，不使用在线支付平台进行交易结算的话一般需要一对一进行收账，将这些业务电子化、自动化后，销售业务与内部管理就可以进行有机结合，对产品、交易、仓储及财务统计的管理都会更加方便且准确。

第二章 布料批发管理需求分析

2.1 主要特性、范围及局限性

（1）主要特性（FEature）

FE-1：供应商信息增查改

FE-2：布料属性类别信息增查改（移除），布料产品信息增查改（下架）

FE-3：客户信息增查改

FE-4：仓库基础信息增查改（移除）

FE-5：管理员账户信息增查改

FE-6：采购订单、批发订单信息增删查改

FE-7：导出订单信息

FE-8：采购（下单入库）、批发（包括下单支付、取消、接单出库、退货换货、结单评价）业务服务流程的实现

FE-9：查询公司收付款流水

FE-10：导出公司收付款流水

FE-11：库存内容增删查改

FE-12：出入库流程的实现

FE-13：查询出入库流水

FE-14：导出库存内容、出入库流水

FE-15：客户进货单增删查改

FE-16：查询客户退付款流水

FE-17：导出客户退付款流水

FE-18：生成查看财务统计记录

FE-19：导出财务统计报表

（2）系统范围

表 2-1 系统范围

特性	版本 1	版本 2	版本 3
FE-1	完全实现		
FE-2	不可增删布料属性类别	完全实现	
FE-3	完全实现		
FE-4	完全实现		
FE-5	高级、采购、批发、库存管理员账户各仅有一个，不可增删改	超级、采购、批发、库存管理员账户各多于一个，不可修改信息	完全实现
FE-6	完全实现		
FE-7	不实现	不实现	完全实现
FE-8	不实现取消批发订单及退换货功能，且收付款均为虚拟操作，并不实际调用外部支付接口	实现取消批发订单及退换货功能，收付款均为虚拟操作	完全实现
FE-9	完全实现		
FE-10	不实现	不实现	完全实现
FE-11	完全实现		
FE-12	完全实现		
FE-13	完全实现		
FE-14	不实现	不实现	完全实现
FE-15	不实现	完全实现	
FE-16	完全实现		
FE-17	不实现	不实现	完全实现
FE-18	完全实现		
FE-19	不实现	不实现	完全实现

（3）局限性（LImitation）

LI-1：没有独立的支付系统，客户需通过外部支付接口提前付款，且付款之后产品才会出库发货，这对客户来说资金保障不够强；

LI-2：没有客服答疑模块，客户与员工之间的询问交流、退换货之前的协商只能

线下进行，无法留下对话记录；

LI-3：该系统对账务核对查验功能没有进行重点处理，仅仅记录订单及收付款流水信息并统计；

LI-4：该系统没有对产品的运输过程进行处理，仅仅记下了物流单号，计算了运输费用；

LI-5：该系统未实现自动备份功能，数据的安全性不够高；

LI-6：系统对并发的处理不够健壮。

2.2 技术要求

选择使用 Java 语言进行开发，使用 SpringMVC+Spring+Mybatis 框架，集成开发环境选择 eclipse，使用 mysql 数据库，Web 服务器使用 Tomcat。

使用 JSON 字符串将浏览器与服务器之间传送的数据进行对象化处理，使用 ajax 技术实现网页的部分更新和动态加载。[7]

安全性要求方面，所有用户的密码都要进行 MD5 加密操作，并且要对不同角色的系统使用者进行权限管理。

进行开发编码并测试，保证项目对三个主流浏览器（IE，Firefox，Chrome）基本兼容，可正常使用。

第三章 布料批发管理系统设计

3.1 功能模块

布料批发管理系统分为两大部分，分别为批发公司直接操作的部分（即类似于 ERP 软件或者“进销存”软件的功能），以及客户直接操作部分（即类似于电子商务网站的功能），如图 3-1 所示。

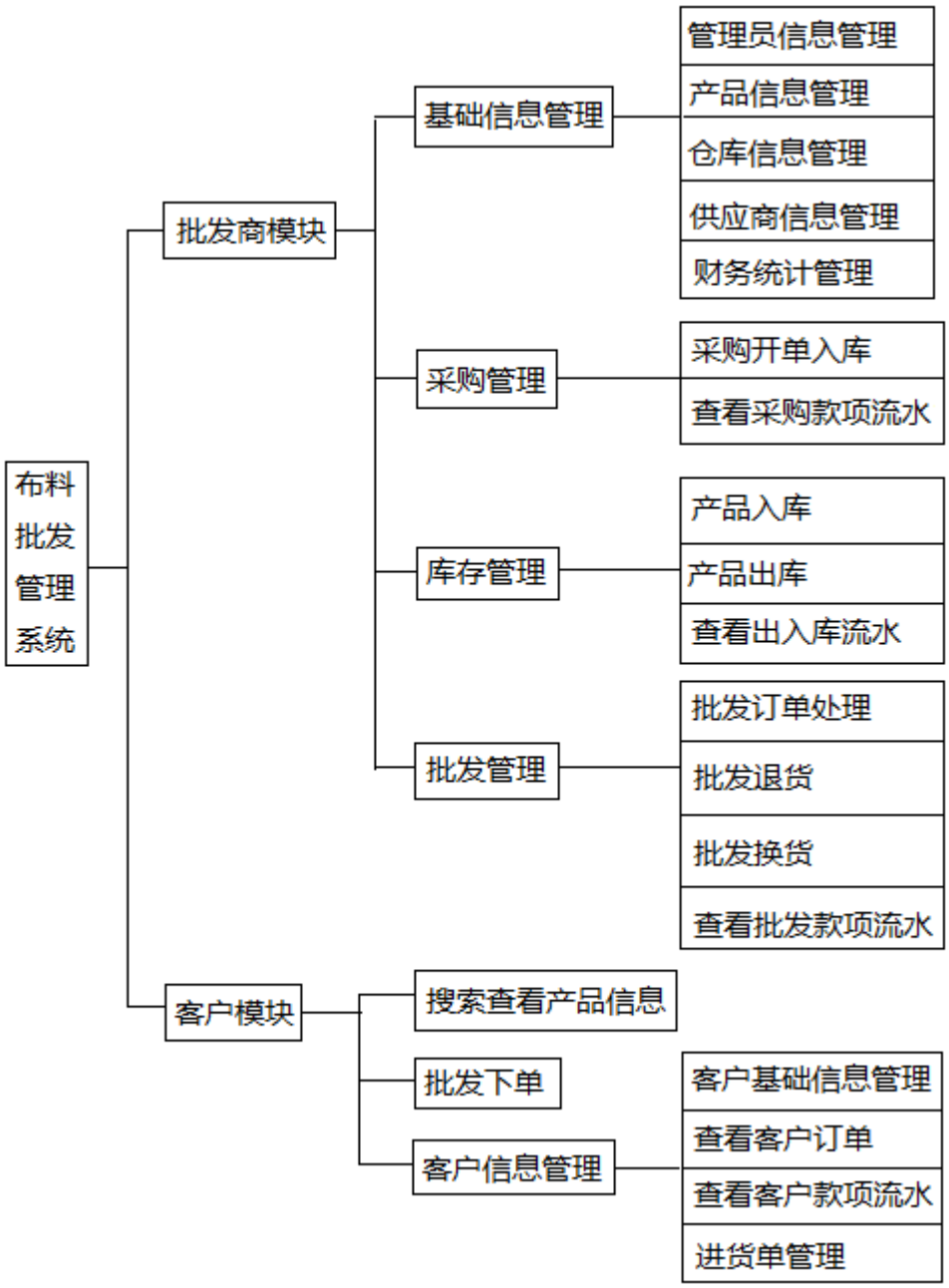

图 3-1 系统功能模块图

批发商部分分为四个模块：基础信息模块为对软件系统和批发公司运作所需的基础信息进行管理，包括管理员、供应商、产品、仓库和财务统计信息，其中

产品信息包括产品基本信息、产品属性类别以及产品的分梯度价格；采购模块为采购管理员对采购开单入库过程的管理以及对采购过程的款项流通进行查看；库存管理为仓库管理员对产品出入库过程的管理以及对仓库内容明细和出入库流水的查看；批发管理模块为批发管理员对批发过程（包含下单和退换货）以及批发订单的管理，对批发过程的款项流通进行查看。

客户部分分为三个模块：搜索查看产品信息功能是网站客户首页的展示、搜索功能和产品具体信息页的加入进货单、查看评论等等功能；批发下单是客户参与的批发全过程，包含付款、退换货、评价订单等功能；客户信息管理为客户对自身基本信息、订单和进货单的管理，对款项流水信息的查看。

3.2 角色主要用例

仓库管理员可以对仓库基础信息、库存内容和出入库流水及统计信息进行管理，完成出入库（含报损报溢）的工作；采购管理员可以进行对供应商信息、产品信息以及采购订单信息进行管理，完成采购的工作；批发管理员可以进行对批发订单信息的管理和对客户信息的查看，完成批发的工作；高级管理员除了上述普通管理员的权限外，还可以对管理员信息、产品费用信息和财务统计信息进行管理；而客户可以搜索查看产品信息，批发下单，以及对自己的信息以及进货单进行管理。

下图为系统的顶级用例图：

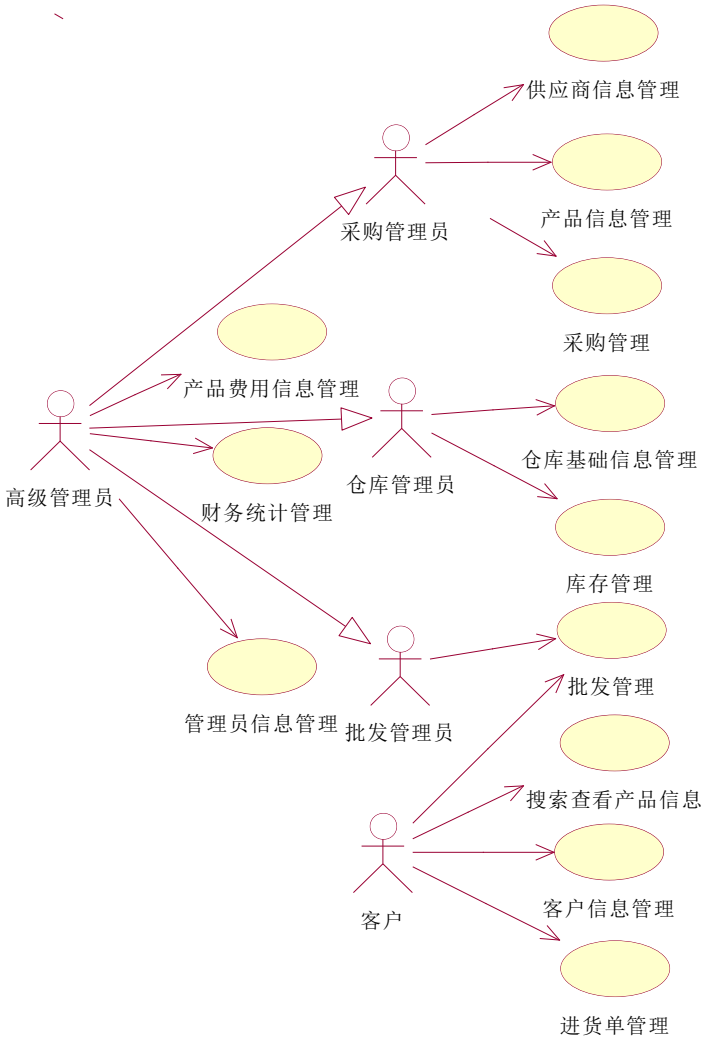

图 3-2　系统顶级用例图

3.2 具体用例

（1）系统具体用例

为系统特性范围中版本 2 的用例。

表 3-1 系统具体用例

主要参与者	用例（Use Case）
消费者（客户）	UC-1：搜索查看产品信息（展示、搜索）
	UC-2：客户基础信息管理
	UC-3：进货单管理（加入、移除货物，结算）
	UC-4：查看客户订单
	UC-5：查看客户款项流水信息
消费者（客户） 批发管理员	UC-6：批发下单
	UC-7：批发退货
	UC-8：批发换货
批发管理员	UC-9：管理批发订单
	UC-10：查看批发款项流水信息
采购管理员	UC-11：供应商信息管理
	UC-12：产品信息管理（属性类别、详细信息、价格）
	UC-13：采购开单
	UC-14：查看采购订单
	UC-15：查看采购款项流水信息
仓库管理员	UC-16：仓库基础信息管理
	UC-17：产品入库
	UC-18：产品出库
	UC-19：查看出入库流水信息
高级管理员	UC-20：管理员信息管理
	UC-21：查看财务统计信息（采购、批发）

（2）部分重点用例详细信息及活动图

批发下单：

用例 ID 号：UC-6

参与者：消费者、批发管理员

描述：消费者对所选产品进行付款、取消、收货结单、评价

前置条件：

1.消费者已成功登录系统

2.消费者选中产品，进入订单界面

后置条件：

1.批发订单更新

2.生成收付款流水

主干过程：

6.0 客户对所选产品付款

1.客户进入订单查看界面，客户点击确认下单

2.客户选择收货地址，确认信息后点击提交，系统生成新批发订单

3.客户付款，系统设置订单为已付款状态，生成付款流水

4.批发管理员接单、申请发货出库

5.客户收货评价，完成订单

分支过程：

6.1 客户点击取消订单（第 3 步分支出来）

1.询问客户是否确认取消

2a.客户确认取消，系统生成取消订单记录，修改订单信息，终止用例

3a.客户确认不取消，返回第 3 步

6.2 客户点击取消订单（第 5 步分支出来）

1.询问客户是否确认取消

2a.客户确认取消，系统申请退款，修改订单状态，终止用例

3a.客户确认不取消，返回第 5 步

异常：

6.0.E.1 新批发订单存入数据库失败（第 3 步）

1.系统提示订单提交失败，返回第 1 步

6.0.E.2 客户付款失败（第 4 步）

1.系统提示付款失败，返回第 4 步

包含: 无

优先级: 高

业务规则: BR-5

活动图:

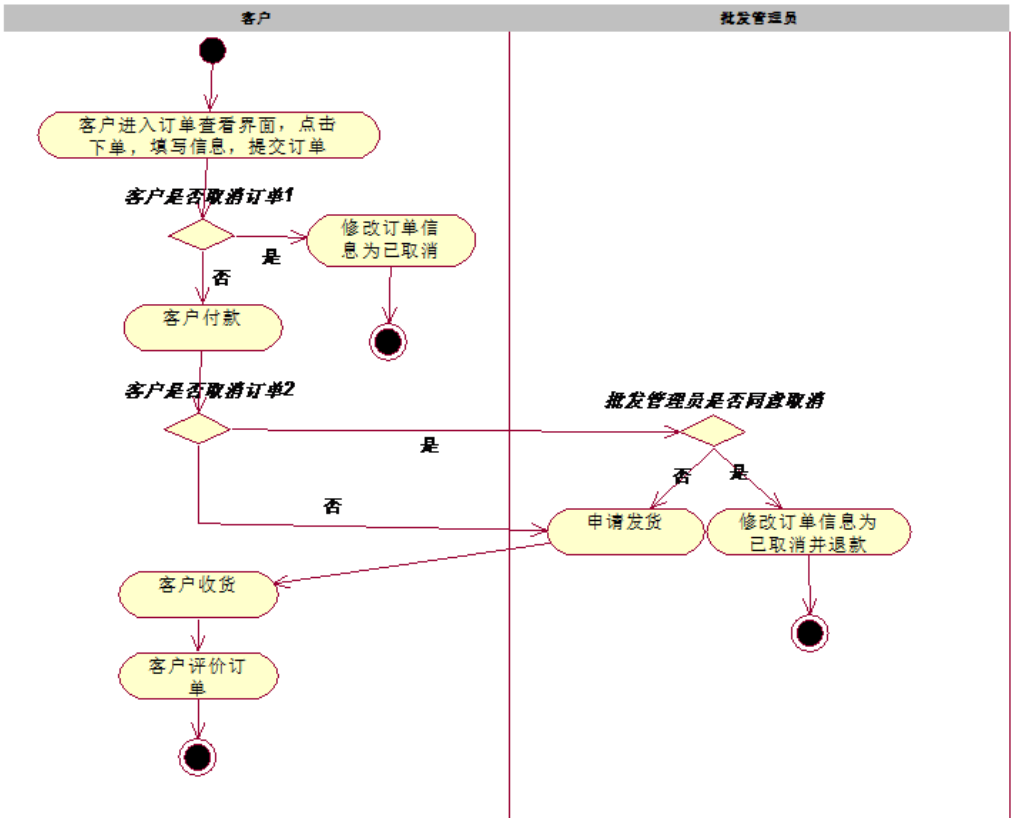

图 3-3 批发下单活动图

批发退货：

用例 ID 号：UC-7

参与者：客户、批发管理员、仓库管理员

描述：客户将不满意的产品退还，批发管理员将客户的钱款退还

前置条件：

1.客户已收到货物

2.客户进入订单查看界面，选中想退还的订单项

后置条件：

1.退还的产品入库

2.客户的钱款退还

主干过程：

7.0 客户退货

1.客户点击申请退货

2.管理员确认退货

3.客户填写退还货物的物流单号

4.管理员确认退款

5.仓库管理员将退还的产品入库

分支过程：

7.1 客户取消退货申请（从第 1 步分支出来）

1.询问客户是否确认取消

2a.客户确认取消，系统修改退货记录状态为已取消，终止用例

3a.客户确认不取消，返回第 1 步

7.2 管理员拒绝退货（从第 2 步分支出来）

1.修改退货记录状态为已拒绝，终止用例

异常：

7.0.E.1 记录信息修改失败

1.系统提示操作失败，返回当前界面

7.0.E.2 管理员退款失败（第 4 步）

1.系统提示退款失败，返回第 4 步

包含：产品入库

优先级：低

业务规则：BR-4

活动图：

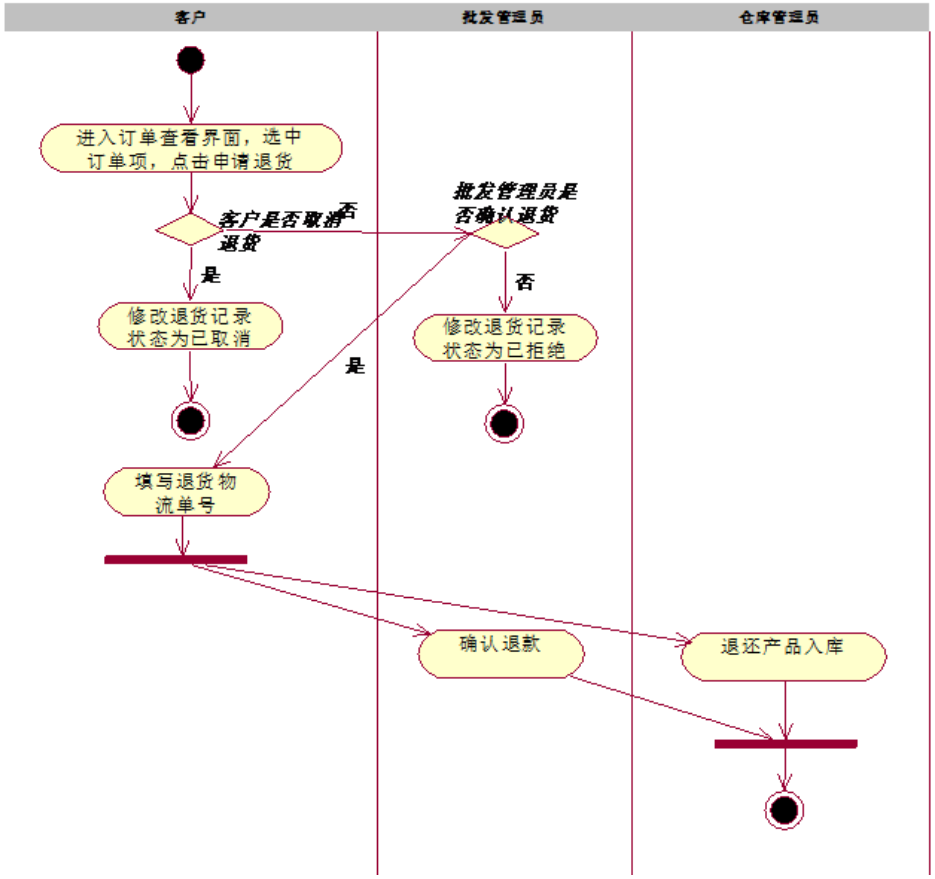

图 3-4 批发退货活动图

批发换货：

用例 ID 号：UC-8

参与者：客户、批发管理员、仓库管理员

描述：将客户不满意的货物退换

前置条件：

1.客户已收到货物

2.客户进入订单查看界面，选中想退换的订单项

后置条件：

1.退还的产品入库

2.重新发货

主干过程：

8.0 货物退还、重发

1.客户点击申请换货

2.批发管理员确认换货

3.客户退还货物，填写退还物流单号

4.仓库管理员确认货物返还

5.仓库管理员重新发货，填写重发物流单号

分支过程：

8.1 客户取消换货申请（从第 1 步分支出来）

1.询问客户是否确认取消

2a.客户确认取消

2b.系统修改换货记录状态为已取消，终止用例

3a.客户确认不取消

3b.返回第 1 步

8.2 管理员拒绝换货（从第 2 步分支出来）

1.修改退货记录状态为已拒绝，终止用例

异常：

8.0.E.1 记录信息修改失败

1.系统提示操作失败

2.返回当前界面

包含：产品入库、产品出库

优先级：低

业务规则：BR-4

活动图：

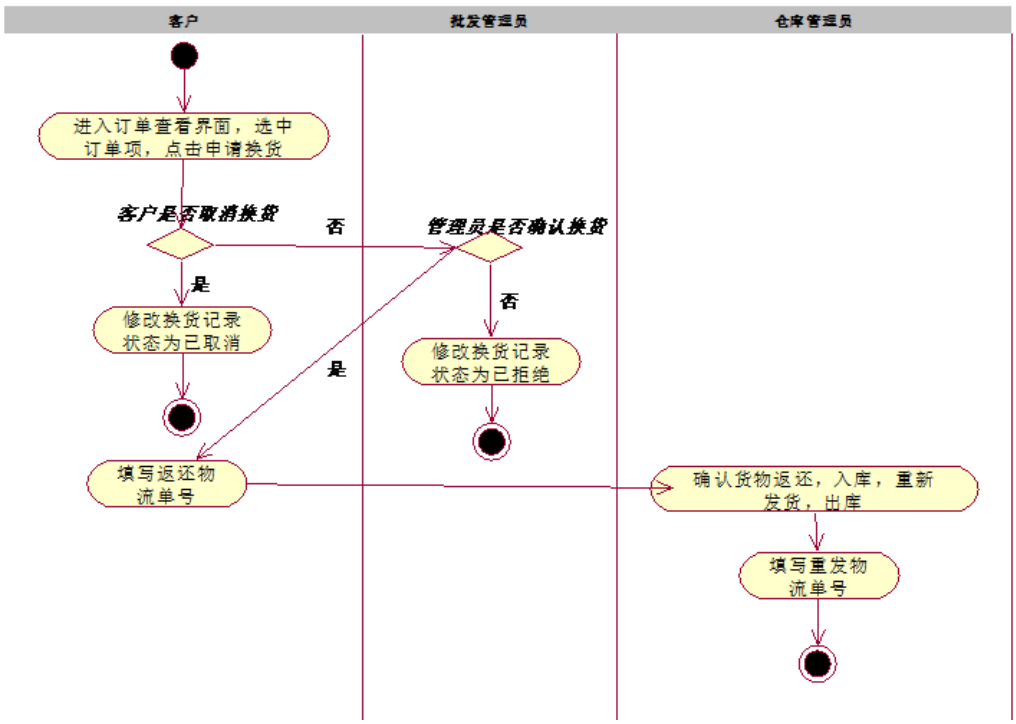

图 3-5 批发换货活动图

采购开单：

用例 ID 号：UC-13

参与者：采购管理员、仓库管理员

描述：为采购货物建立订单，记录付款入库情况

前置条件：

1.采购管理员已登录，进入采购开单界面

后置条件：

1.仓库管理员将质量不合格或被破坏的产品报损

主干过程：

13.0 采购管理员建立采购订单，记录付款信息，产品入库

1.采购管理员新建采购订单，填入产品信息，提交订单

2.采购管理员填写付款信息，生成采购款项流水

3.采购管理员申请产品入库，采购订单状态变为待入库

4.仓库管理员将产品入库

分支过程：

13.1 新产品在数据库中不存在（从第 1 步分支出来）

1.采购管理员新建产品

2.返回第 1 步

异常：

13.0.E.1 新采购订单存入数据库失败（第 1 步）

1.系统提示订单提交失败

2.返回第 1 步

13.0.E.2 采购管理员付款失败（第 2 步）

1.系统提示付款失败

2.返回第 2 步

包含：新建产品

优先级：高

业务规则：BR-6

活动图：

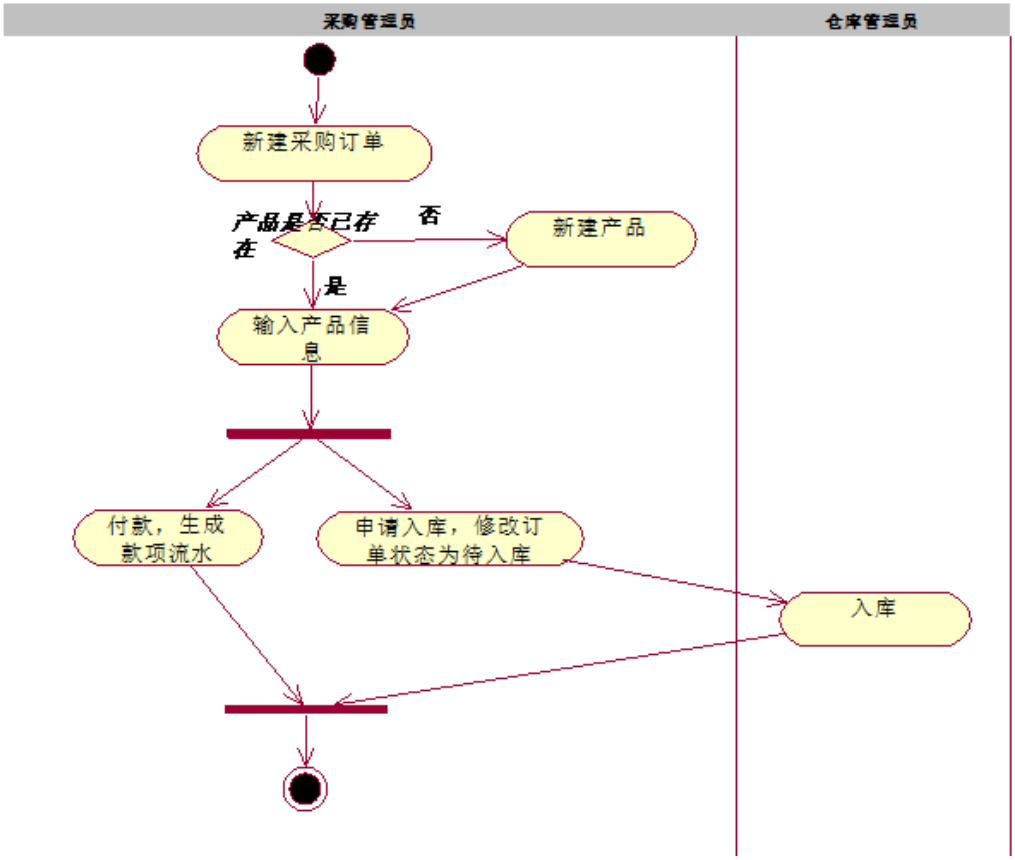

图 3-6 采购开单活动图

产品入库：

用例 ID 号：UC-17

参与者：仓库管理员

描述：仓库管理员将接收到的产品录入仓库

前置条件：

1.仓库管理员接收到产品

后置条件：

1.仓库管理员进行盘点，将不合格或被破坏的产品报损

主干过程：

17.0 将与订单相关的产品入库

1.仓库管理员选择订单相关入库操作

2.管理员选中待入库采购订单或批发退换货记录

3.管理员选择订单项，点击入库

4.系统新建入库记录，管理员输入信息，确认入库

5.管理员将同一订单的其他订单项全部录入完毕

分支过程：

17.1 将非订单相关产品入库，如报溢（从第 1 步分支出来）

1.仓库管理员点击新建入库记录

2.管理员填写产品信息，入库原因等信息，确认入库

17.2 管理员未全部录入完毕（从第 5 步分支出来）

1.管理员继续录入，返回第 3 步

异常：

17.0.E.1 新入库记录存入数据库失败（第 4 步）

1.系统提示记录提交失败

2.返回第 3 步

包含：无

优先级：高

业务规则：无

活动图：

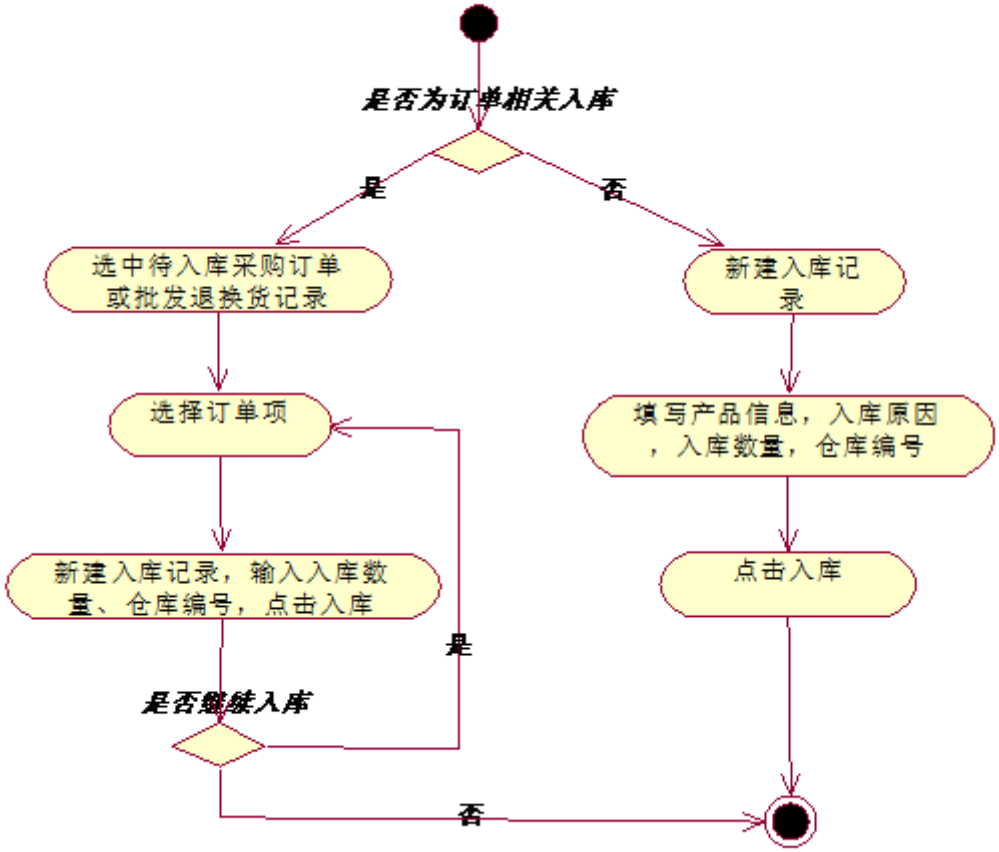

图 3-7　产品入库活动图

产品出库：

用例 ID 号：UC-18

参与者：仓库管理员

描述：仓库管理员将仓库中的产品送出仓库

前置条件：1.收到待发货订单或者换货记录

后置条件：无

主干过程：

18.0 将与订单相关的产品出库

1.仓库管理员选择订单相关出库操作

2.管理员选中待发货批发订单或批发换货记录

3.管理员选择订单项，点击出库

4.系统新建出库记录，自动填写信息，确认出库

5.管理员将同一订单的其他订单项全部出货完毕

分支过程：

18.1 将非订单相关产品入库，如报损（从第 1 步分支出来）

1.仓库管理员进入库存明细查看界面，选择仓库

2.管理员选择某件产品明细，点击出库

3.管理员填写产品数量、入库原因等信息，确认出库

4.系统生成入库记录

异常：

18.0.E.1 新入库记录存入数据库失败（第 4 步）

1.系统提示记录提交失败

2.返回第 3 步

包含：无

优先级：高

业务规则：无

活动图：

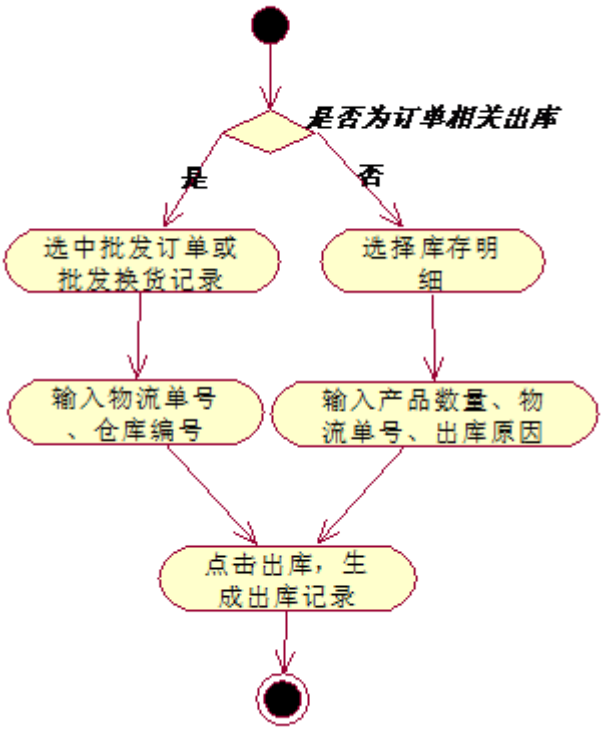
是否为订单相关出库
是
否
选中批发订单或批发换货记录
选择库存明细
输入物流单号、仓库编号
输入产品数量、物流单号、出库原因
点击出库，生成出库记录

图 3-8　产品出库活动图

第四章 布料批发管理系统数据库设计

4.1 业务规则（Business Rule）

BR-1：批发业务中，购买的产品数额的数量级不同，产品的单价也会有梯度，购买得越多单价越便宜；

BR-2：产品型号是同种产品的不同型号规格，通常是不同颜色，有时也会是不同幅宽等等；

BR-3：本系统中，布料计数使用的统一单位是米；

BR-4：运输费用默认由客户承担；

BR-5：客户需先付全款，批发管理员再接单申请发货；

BR-6：采购付款与入库为独立步骤，可并行进行不分先后，且一个采购订单可分多次进行付款。

4.2 数据表设计

本系统一共设计了 29 张数据表。

所有储存流水记录的数据表都是将出入记录存储在一起的，以出入标识作为区分。

采购订单和批发订单都分别由订单本身的属性和订单项构成，订单项就是一个订单中的不同产品项。所有的订单项数据表都是将订单编号和每个订单中的订单项序号组合在一起形成编号，作为主键。

主键为这种形式的还有产品价格表和产品型号表。因为购买数量不同的时候单价有不同，所以产品和价格在批发业务中是一对多关系，产品和产品型号之间的关系也是一对多。

产品型号和产品图片之间也是一对多关系，一种产品的型号序号从 1 开始实际计算，序号为 0 的型号代表空型号，不用于售卖，是用于储存为该种产品不同型号间的通用信息，如型号序号为 0 所对应的产品图片为该种产品的通用介绍图片，而不是某个型号的实物图片。

库存内容表中存储的数据是某个仓库存储某种产品型号的数量，单位是米。

批发订单的取消、换货和退货都有特定的数据表存储相关的记录，为批发订单取消记录表、换货记录表和退货记录表。因为这些操作中是涉及到的订单状态过多，换货退货还涉及到货物和款项的流通，如果不将其分解开来，数据表的结构和操作将很复杂，因此本系统中将其分解开。

产品属性及类别信息表是用于存储产品的属性类别及属性的。由于布料的属

性类别太多，如成分类型、原料及含量、纺织工艺、染整工艺等等[8]，每种属性类别又包含很多种属性，如成分类型就有棉织物、麻织物、毛织物、丝织物、化纤织物、混纺织物、交织物[9]，所以此处使用一张单独的数据表存储这些属性，和填写地址时使用下拉列表选择省份和城市类似，在新建产品填写信息的时候就可以直接使用下拉列表选择该产品的属性。本系统中产品属性及类别信息可以增删查改。

图 4-1　系统 PDM（Physical Data Model，物理数据模型）

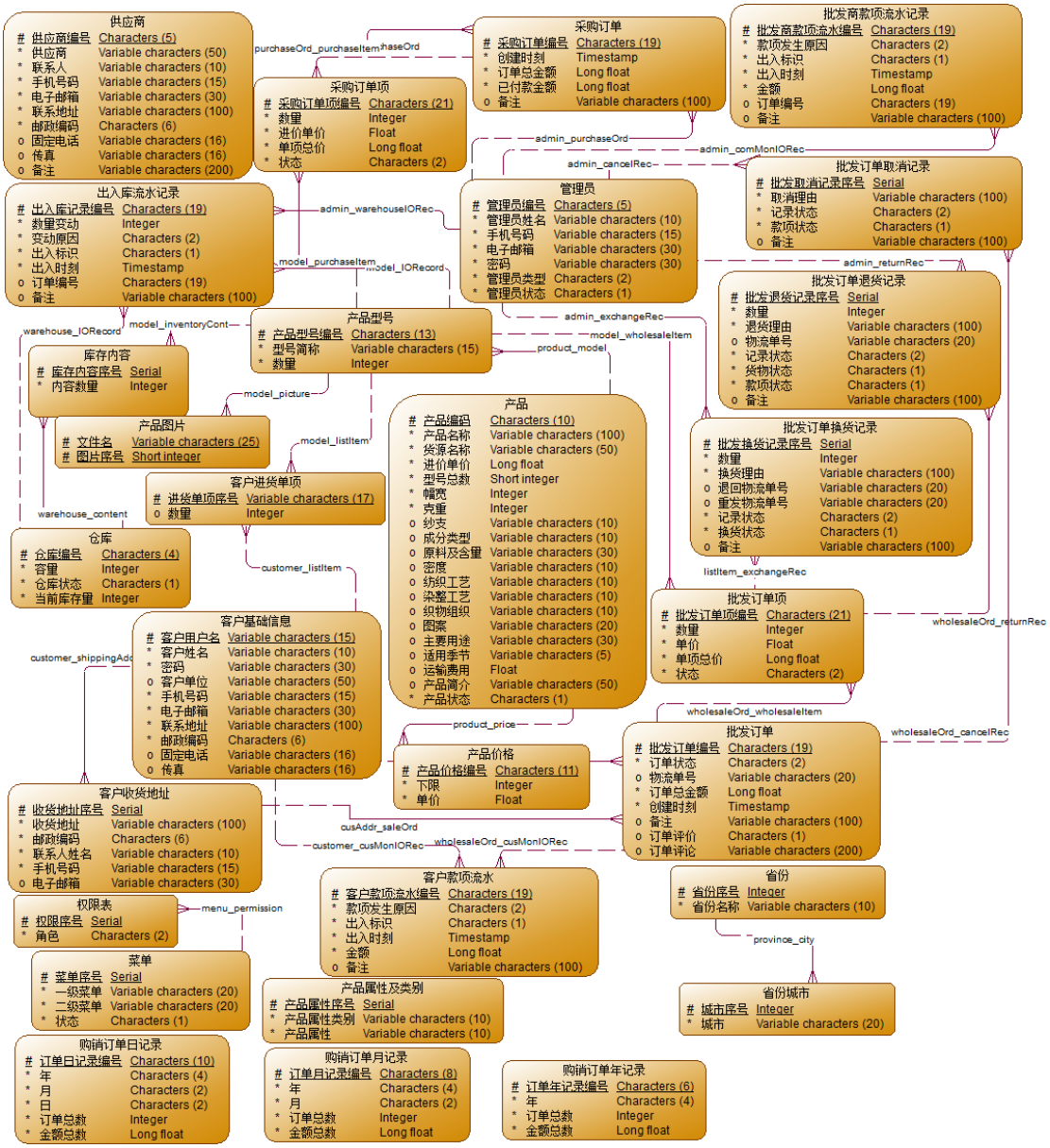

图 4-2　系统 CDM（Concept Data Model，概念数据模型）

数据字典

具体数据字典见附录

4.3 重点用例的数据流图

批发下单：

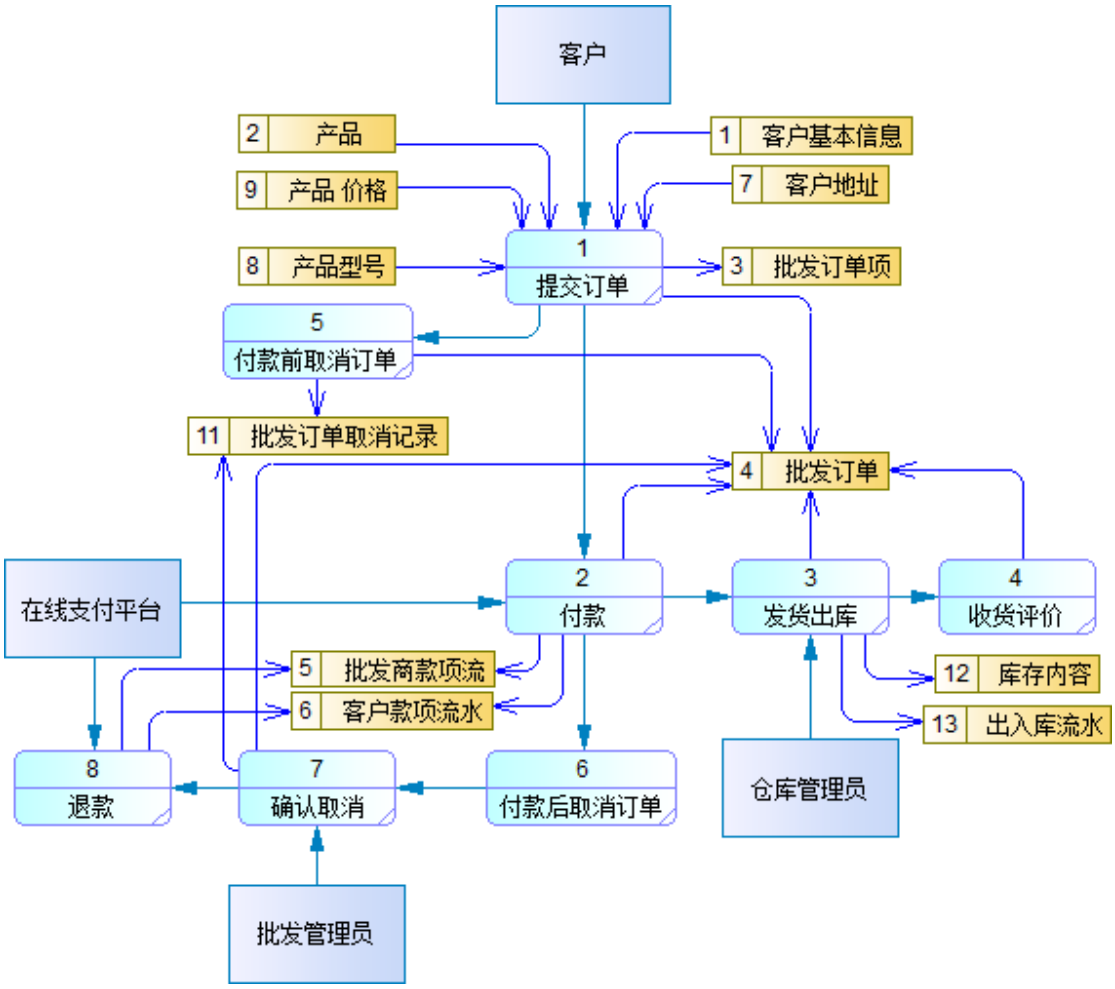

图 4-3 批发下单数据流图

批发换货：

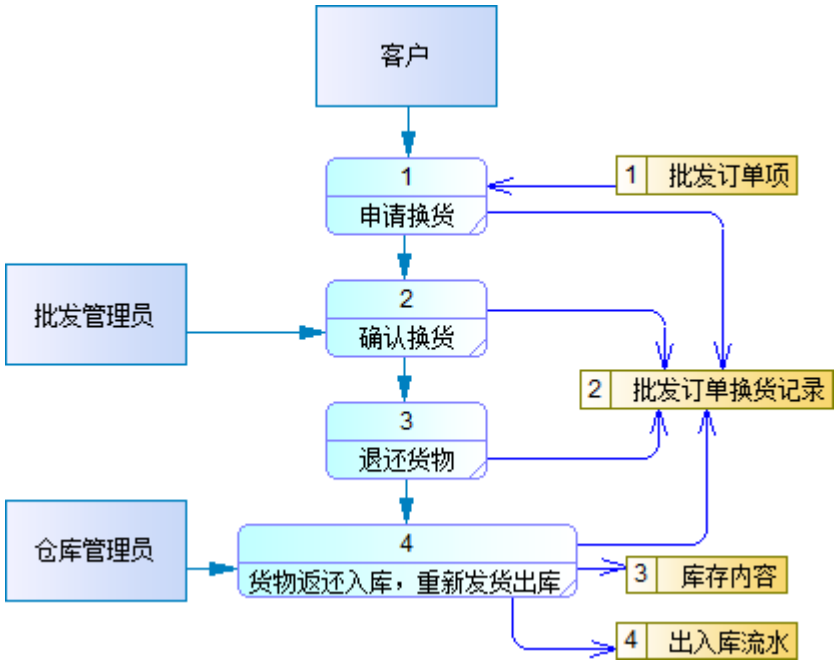

图 4-4 批发换货数据流图

批发退货：

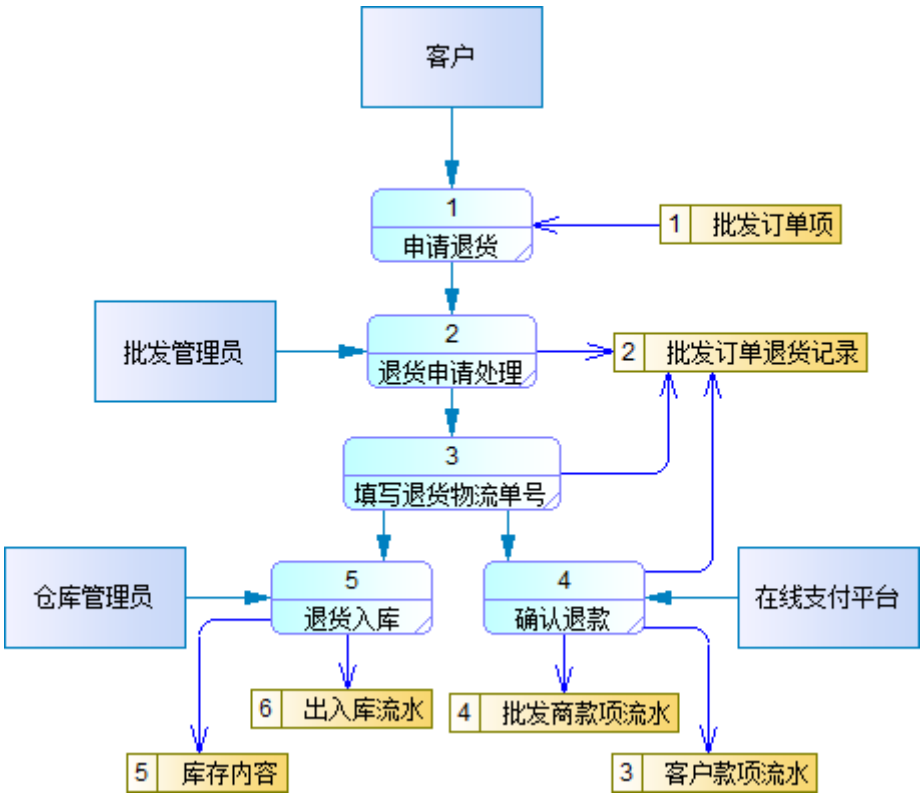

图 4-5 批发退货数据流图

采购下单：

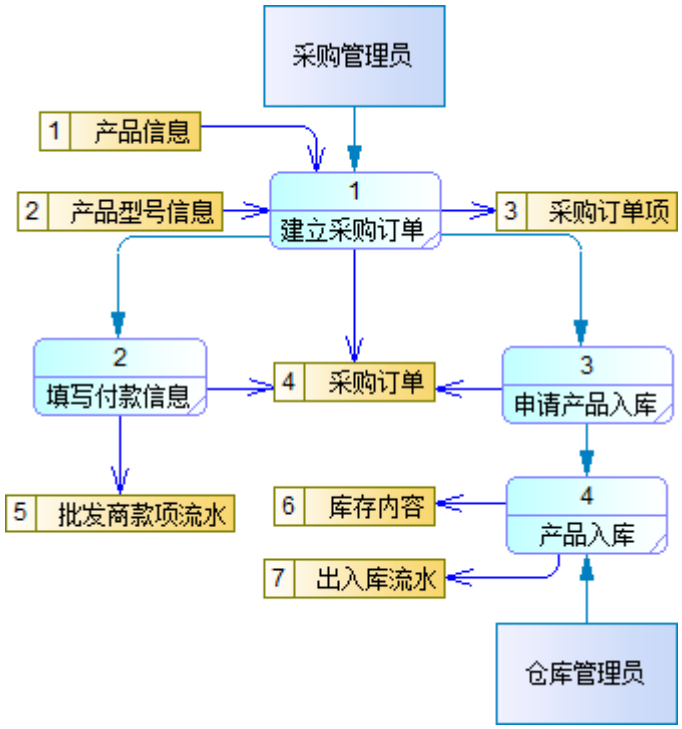

图 4-6 采购下单数据流图

产品入库：

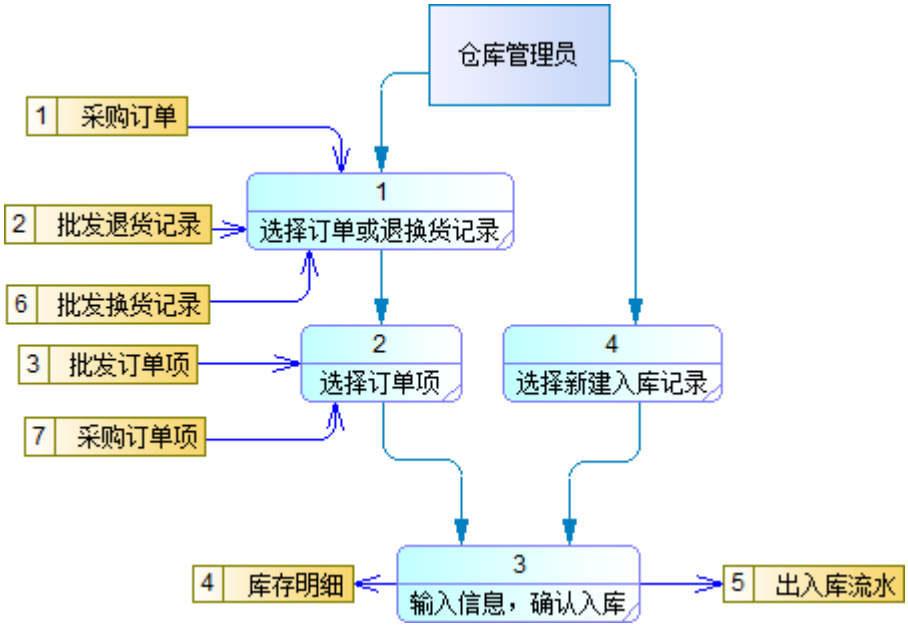

图 4-7 产品入库数据流图

产品出库：

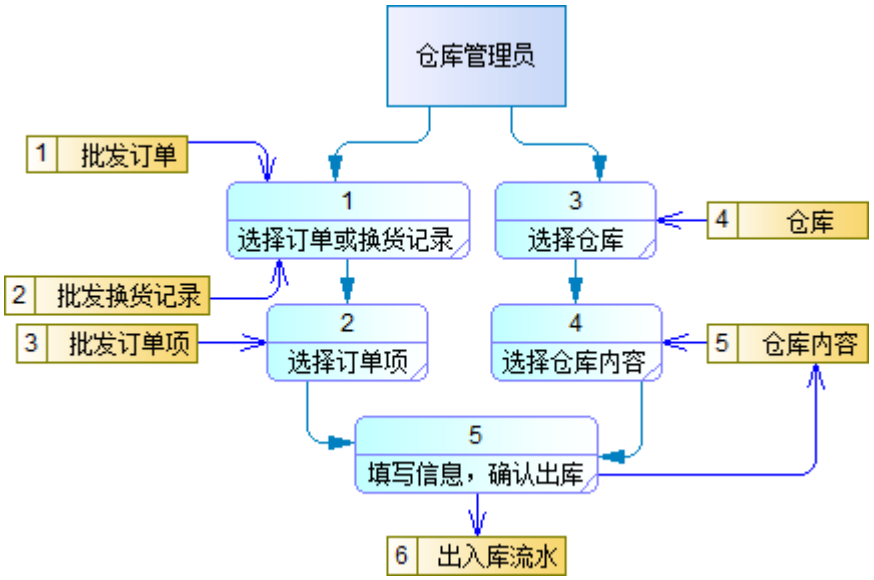

图 4-8 产品出库数据流图

4.4 系统层次结构及数据类间关系

（1）系统层次结构

本系统使用 MVC 模式为基础的四层结构，分别为数据层、业务层、服务层和视图层。数据层主要完成对系统中各种实体非实体数据的建模，建立各个 PO 类（Persistent Object，持久化对象），以及与 MySQL 数据库的交互，业务层主要完成对数据的增删查改，建立各个 DAO 类（Data Access Object，数据访问对象）

或 mapper 类（使用 Mybatis 框架时）；服务层调用业务层中增删查改的方法完成系统用例中的一个个功能，建立各个服务类（service）或者 BO 类（Business Object，业务对象）；在视图层，建立各个 DTO 类（Data Transfer Object，数据传输对象）和 VO 类（View Object，视图对象），完成后台与前端网页之间的数据传递和显示。

业务层进行对数据持久层的数据的增删查改，服务层调用业务层的方法完成系统功能的业务逻辑，视图层将服务层的操作得到的结果进行封装，传送给前端网页。

（2）数据持久层的类间关系

数据持久层的类间关系如下图所示

供应商类（Supplier）、管理员类（Admin）和客户类（Customer）分别为三个角色，其中客户地址类（CustomerAddress）组合进客户类中。

产品价格类（ProductPrice）和产品型号类（ProductModel）聚合进产品类（Product）中，产品类组合进采购订单项目类（PurchaseItem）、批发订单项目类（WholesaleItem）和进货单项目类（PurchaseListItem）中，其中客户的进货单项目类（PurchaseListItem）依赖于客户类，产品图片类（Picture）组合进产品型号类（ProductModel）中。每件产品中，型号序号为 0 的产品图片为产品通用介绍图片，而非具体某种型号的实物图片。

采购订单项目类和批发订单项目类分别聚合进采购订单类（PurchaseOrder）和批发订单类（WholesaleOrder）中，而采购订单取消记录（OrderCancelRec）、换货记录（OrderExchangeOrder）和退货记录（OrderReturnRec）都依赖于批发订单类。

仓库内容类（WarehouseContent）依赖于产品型号类和仓库类（Warehouse），出入库流水类（WarehouseIORec）依赖于仓库内容类。

批发商款项流水记录类（SellerMonRec）和客户款项流水记录类（CustomerMonRec）都继承自款项流水记录类（MoneyRecord）。

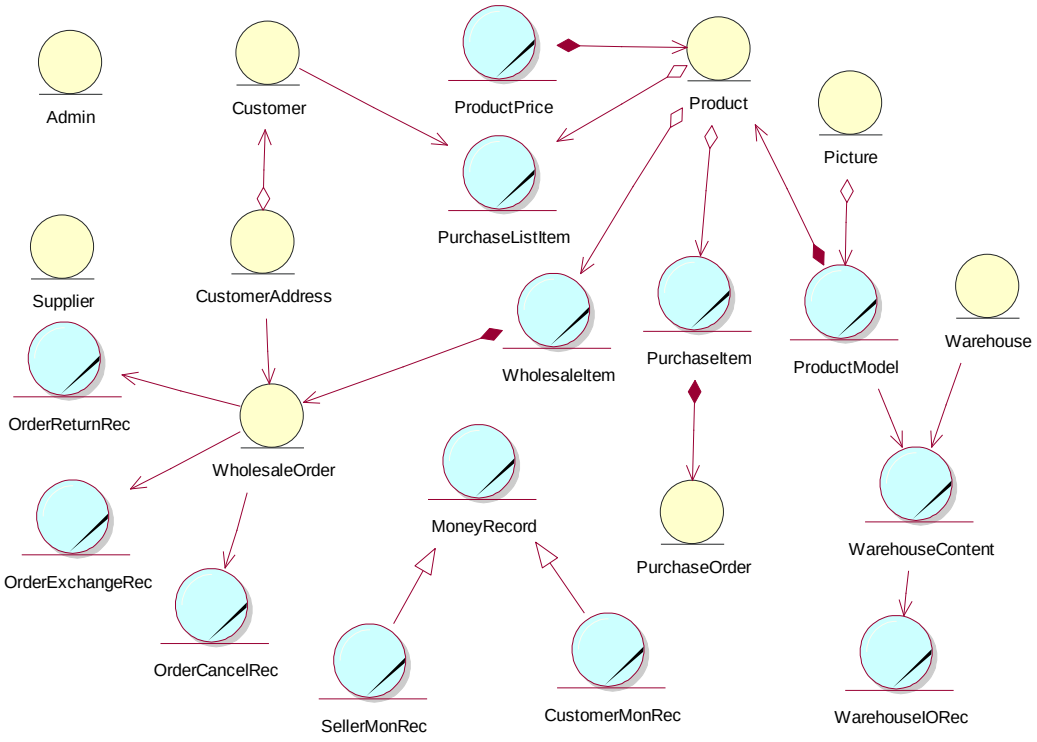

图 4-9 数据持久层类间关系

第五章 布料批发管理系统实现

5.1 系统目录层次及框架构建

（1）后台目录层次

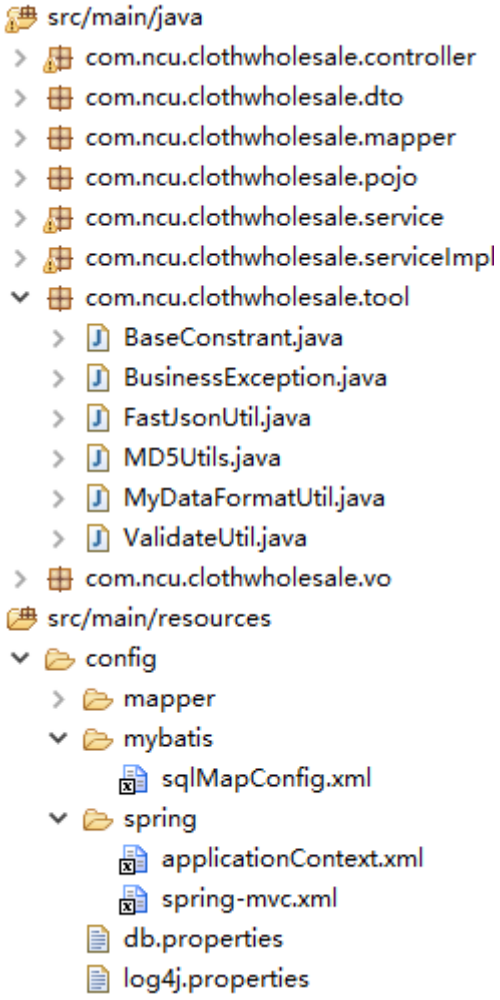

图 5-1 后台目录结构

如图 5-1 所示，系统的后台目录主要分为 java 代码和配置代码两部分。Java 代码主要有：

controller 包，实现视图层与业务层的连接与数据传输

dto 包，与前端交互、向前端传输数据的时候使用的数据对象

vo 包，封装前端页面显示、向后台提交的数据

mapper 包，业务层的实现，完成持久层数据对象的增删查改

pojo 包，持久层的数据对象

service 包，服务层的接口

serviceImpl 包，服务层接口的实现

tool 包，系统中使用的工具，包括 BaseConstrant.java（系统中使用的静态常量，如订单状态等），BusinessException.java（后台业务执行出错时抛出，包含错误信息，如登录时密码错误等），FastJsonUtil.java（统一 JSON 字符串格式的工具），MD5Util.java（MD5 加密工具），MyDataFormatUtil.java（日期格式与字符串之间的各种转换），ValidateUtil.java（手机号，电子邮箱，空字符串的验证）

配置代码主要有：

config/mapper/*.xml：数据库操作语句

config/mybatis/sqlMapConfig.xml：mybatis 批量扫描别名设置

config/spring/applicationContent.xml：spring 设置及 spring 与 MyBatis 整合

config/spring/spring-mvc.xml：spring-mvc 设置

config/db.properties：数据库数据源属性

config/log4j.properties：log4j 日志设置

（2）前端目录层次

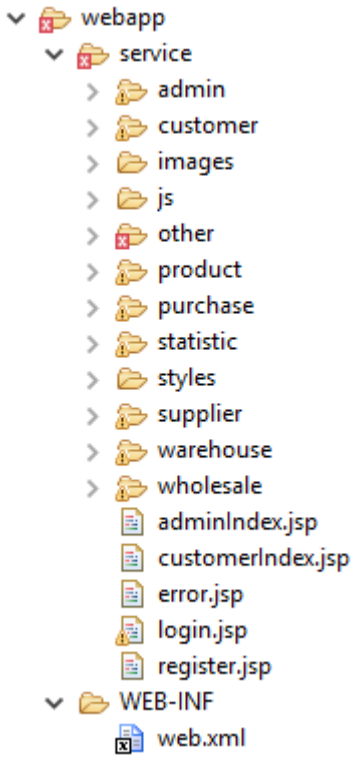

图 5-2　前端目录结构

webapp/service 目录下是各个模块的页面及使用的 JavaScript 代码文件（js 文件夹）、图片（images 文件夹）、样式表文件（styles 目录）、layui 插件（other 目录）

WEB-INF/web.xml 文件是 SSM 框架整合相关设置

5.2 系统实现

系统实现了系统特性范围中版本 1 的功能，以下是重点功能及部分关键技术的实现。

（1） 重点功能

采购开单：

每个订单中可能有多个产品及型号，购买的数目也各不相同，因此，为了实现订单项的灵活组合，将订单的存储与订单项的存储分解开来，先生成订单，再存入订单项，订单项保存成功后，更新订单总额。

```
/**
     * 插入订单项
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public boolean createPurchaseItems(List<PurchaseItem> itemList)
throws BusinessException {
        boolean ret = true;
        double sumPrice = 0;
        if(itemList.size()<=99){
            for(int i=0;i<itemList.size();i++){
                //设置信息
                PurchaseItem pItem = itemList.get(i);
                String purcItemCode =
pItem.getPurchaseOrdCode()+MyDataFormatUtil.unitFormatTen(i+1);
                pItem.setPurchaseItemCode(purcItemCode);

    pItem.setState(BaseConstrant.PURCHASE_ITEM_STATE_PAYING);

    pItem.setTotalPrice((double)pItem.getOrignalPrice()*pItem.getTota
lNum());
                sumPrice += pItem.getTotalPrice();

                if(purcItemMapper.insertSelective(pItem)!=1){
                    ret = false;
                    throw new BusinessException("保存失败，请重新保存！");
                }
            }
            //更新订单总价
            PurchaseOrder pOrder =
purcOrdMapper.selectByPrimaryKey(itemList.get(0).getPurchaseOrdCode()
);
            pOrder.setSumPrice(sumPrice);
            purcOrdMapper.updateByPrimaryKey(pOrder);
        }else{
            throw new BusinessException("每个订单最多99个订单项！");
        }
        return ret;
    }
```

批发下单：

批发订单也与采购订单一样有着订单项处理的问题，所以此处使用相同的方

法处理生成订单的问题。批发确认下单后，客户需要付清款项之后产品才会发货出库。

```
/**
     * 批发订单付款
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public boolean payWholesaleOrder(String wOrderCode, String username)
            throws BusinessException {
        WholesaleOrder wOrder =
saleOrderMapper.selectByPrimaryKey(wOrderCode);

        if(wOrder!=null){

    wOrder.setOrderRate(BaseConstrant.WHOLESALE_ORDER_STATE_PAID);
            saleOrderMapper.updateByPrimaryKey(wOrder);
            //插入客户款项流水
            CusMonIORec cusIORec = new CusMonIORec();
            Date now = Calendar.getInstance().getTime();

    cusIORec.setCusMonIORecCode("KO"+MyDataFormatUtil.DateToYMDHMSS(n
ow));
            cusIORec.setIOmark("O");
            cusIORec.setMoney(-wOrder.getSumPrice());

    cusIORec.setReason(BaseConstrant.CUSTOMER_IO_REASON_WHOLESALE);
            cusIORec.setTime(now);
            cusIORec.setUsername(username);
            cusIORec.setWholesaleOrdCode(wOrderCode);
            cusIORecMapper.insert(cusIORec);
            //插入批发公司流水
            ComMonIORecord comIoRec = new ComMonIORecord();
            comIoRec.setAdminCode("XT001");
            comIoRec.setAmount(wOrder.getSumPrice());

    comIoRec.setComIORecCode("PI"+MyDataFormatUtil.DateToYMDHMSS(now)
);
            comIoRec.setIOmark("I");
            comIoRec.setOrdCode(wOrderCode);

    comIoRec.setReason(BaseConstrant.COMPANY_IO_REASON_PURCHASE);
            comIoRec.setTime(now);
            comIORecMapper.insert(comIoRec);
            return true;
        }
        return false;
    }
```

产品出入库：

因为数据库中出入库记录存储在一张表中，用出入标识字段进行区分，所以生成出入库记录也使用一个方法完成，采购入库和批发出库分别各使用一个方法，但都调用同一个新建出入库记录函数。

```
/**
     * 新建出入库记录
```

```
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public boolean createIORecord(WarehouseIORecord
warehouseIORecord)throws BusinessException{
        Date now = Calendar.getInstance().getTime();
        warehouseIORecord.setTime(now);
        //设置记录编号
        String IORecCodeString = "";
        if(!ValidateUtil.JudgeEmpty(warehouseIORecord.getIOMark())){
            if(warehouseIORecord.getIOMark()=="I")
                IORecCodeString += "RK";
            else if(warehouseIORecord.getIOMark()=="O"){
                IORecCodeString += "CK";
            }

    warehouseIORecord.setWarehouseIORecCode(IORecCodeString+MyDataFor
matUtil.DateToYMDHMSS(now));
            //库存内容更新
            InventoryContentExample example = new
InventoryContentExample();
            InventoryContentExample.Criteria criteria =
example.createCriteria();

    criteria.andModelCodeEqualTo(warehouseIORecord.getModelCode());

    criteria.andWarehouseCodeEqualTo(warehouseIORecord.getWarehouseCo
de());
            List<InventoryContent> list =
invenContMapper.selectByExample(example);
            if(list.size()==1){//已有库存，更新库存数量
                InventoryContent invenCont = list.get(0);
                if(warehouseIORecord.getIOMark() == "I")

    invenCont.setContentSum(invenCont.getContentSum()+warehouseIOReco
rd.getNumberChange());
                else if(warehouseIORecord.getIOMark() == "O"){

    if(warehouseIORecord.getNumberChange()<=invenCont.getContentSum()
)

    invenCont.setContentSum(invenCont.getContentSum()-warehouseIOReco
rd.getNumberChange());
                    else{
                        throw new BusinessException("库存不足");
                    }
                }
                invenContMapper.updateByPrimaryKey(invenCont);
            }else{//新增库存
                InventoryContent invenCont = new InventoryContent();

    invenCont.setContentSum(warehouseIORecord.getNumberChange());

    invenCont.setModelCode(warehouseIORecord.getModelCode());

    invenCont.setWarehouseCode(warehouseIORecord.getWarehouseCode());
                invenContMapper.insert(invenCont);
            }
```

```
            //产品型号更新产品数量
            ProductModel pModel =
prodModelMapper.selectByPrimaryKey(warehouseIORecord.getModelCode());
            if(pModel!=null){
                if(warehouseIORecord.getIOMark() == "I")

    pModel.setTotalNum(pModel.getTotalNum()+warehouseIORecord.getNumb
erChange());
                else if(warehouseIORecord.getIOMark() == "O"){

    pModel.setTotalNum(pModel.getTotalNum()-warehouseIORecord.getNumb
erChange());
                }
                prodModelMapper.updateByPrimaryKey(pModel);
            }
            //插入出入库记录

    if(warehouseIORecMapper.insertSelective(warehouseIORecord) != 1){
                throw new BusinessException("插入出入库记录失败！");
            }else{
                return true;
            }
        }else{
            throw new BusinessException("出入标识为空");
        }
    }
```

采购入库：

```
/**
     * 采购订单项入库
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public boolean createPurcItemWarehouseIORec(WarehouseIORecord
warehouseIORecord)throws BusinessException{
        //订单相关的出入库记录
        if(warehouseIORecord.getOrdCode()!=null &&
warehouseIORecord.getOrdCode().startsWith("CG")){
            //查找订单项
            PurchaseItemExample example = new PurchaseItemExample();
            PurchaseItemExample.Criteria criteria =
example.createCriteria();

    criteria.andPurchaseOrdCodeEqualTo(warehouseIORecord.getOrdCode()
);

    criteria.andModelCodeEqualTo(warehouseIORecord.getModelCode());

            List<PurchaseItem> list =
purcItemMapper.selectByExample(example);
            if(list!= null && list.size()==1){

    if(list.get(0).getState().equals(BaseConstrant.PURCHASE_ITEM_STAT
E_STORING)){
                    if(createIORecord(warehouseIORecord)){//生成出入库记录

    list.get(0).setState(BaseConstrant.PURCHASE_ITEM_STATE_STORED);
```

```
    if(purcItemMapper.updateByPrimaryKey(list.get(0))==1)//更新订单项状
态
                            return true;
                        else{
                            throw new BusinessException("订单项状态更新失败!
");
                        }
                    }
                }else{
                    throw new BusinessException("该采购订单非待入库状态！");
                }
            }else{
                throw new BusinessException("未找到订单项！");
            }
        }else{
            throw new BusinessException("该订单非采购订单！");
        }
        return false;
    }
```

批发出库：

```
/**
     * 批发发货出库
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public boolean createWholesaleOrderIORec(WarehouseIORecord
warehouseIORecord)
            throws BusinessException {
        //订单相关的出入库记录
        if(warehouseIORecord.getOrdCode()!=null &&
warehouseIORecord.getOrdCode().startsWith("PF")){
            //查找批发订单
            WholesaleOrder wOrder =
saleOrdMapper.selectByPrimaryKey(warehouseIORecord.getOrdCode());
            if(wOrder!=null &&
wOrder.getOrderState().equals(BaseConstrant.WHOLESALE_ORDER_STATE_PAI
D)){
                //查找订单项
                WholesaleItemExample example = new
WholesaleItemExample();
                WholesaleItemExample.Criteria criteria =
example.createCriteria();

    criteria.andWholesaleOrdCodeEqualTo(warehouseIORecord.getOrdCode(
));

    criteria.andModelCodeEqualTo(warehouseIORecord.getModelCode());

                List<WholesaleItem> list =
saleItemMapper.selectByExample(example);
                if(list!= null && list.size()==1){
                    createIORecord(warehouseIORecord);//生成出入库记录

    list.get(0).setState(BaseConstrant.WHOLESALE_ITEM_STATE_SHIPPED);

    if(saleItemMapper.updateByPrimaryKey(list.get(0))==1)//更新采购订单
项状态
```

```
                    return true;
                else{
                    throw new BusinessException("订单项状态更新失败！");
                }
            }else{
                throw new BusinessException("未找到订单项！");
            }
        }else{
            throw new BusinessException("该批发订单非已付款状态！");
        }
    }else{
        throw new BusinessException("该订单非采购订单！");
    }
}
```

订单销售记录统计：

订单销售记录有日记录、月记录和年记录三种，统计时需要先生成日记录，再在日记录的基础上统计月记录，年记录同理。本系统中，因为批发订单和采购订单的记录存储在同一个表中，所以批发订单和采购订单的统计是一起处理的。

```
/**
     * 生成订单销售日记录
     */
    @Transactional(rollbackFor=Exception.class)
    @Override
    public OrderDayRec createOrderDayRecord(int year, int mon, int day,
String type)
            throws BusinessException {
        if("CG".equals(type)){
            //查找当日采购订单
            PurchaseOrderExample example = new PurchaseOrderExample();
            PurchaseOrderExample.Criteria criteria =
example.createCriteria();

    criteria.andPurchaseOrdCodeLike("%CG"+year+MyDataFormatUtil.unitF
ormatTen(mon)
                    +MyDataFormatUtil.unitFormatTen(day)+"%");
            List<PurchaseOrder> pList =
pOrderMapper.selectByExample(example);
            System.out.println("pList size:"+pList.size());
            //计算已付款总金额
            double sum = 0;
            for(int i=0;i<pList.size();i++){
                sum+=pList.get(i).getPaidPrice();
                System.out.println(pList.get(i).toString());
            }
            //判断当日记录是否已存在
            OrderDayRec dayRec =
ordDayRecMapper.selectByPrimaryKey("CD"+year+

    MyDataFormatUtil.unitFormatTen(mon)+MyDataFormatUtil.unitFormatTe
n(day));
            if(dayRec==null){
                System.out.println("当日记录不存在");
                dayRec = new OrderDayRec();
                //设置信息
```

```
                dayRec.setDay(MyDataFormatUtil.unitFormatTen(day));
                dayRec.setMonth(MyDataFormatUtil.unitFormatTen(mon));
                dayRec.setYear(Integer.toString(year));

    dayRec.setOrdDayRecCode("CD"+year+MyDataFormatUtil.unitFormatTen(
mon)+
                        MyDataFormatUtil.unitFormatTen(day));
                dayRec.setOrderNum(pList.size());
                dayRec.setMoneyNum(sum);
                //添加到数据库
                if(ordDayRecMapper.insert(dayRec)==1){
                    System.out.println(dayRec.toString());
                    return dayRec;
                }
            }else{
                dayRec.setOrderNum(pList.size());
                dayRec.setMoneyNum(sum);
                if(ordDayRecMapper.updateByPrimaryKey(dayRec)==1){
                    System.out.println(dayRec.toString());
                    return dayRec;
                }
            }
        }else if("PF".equals(type)){
            //查找当日已付款批发订单
            WholesaleOrderExample example1 = new WholesaleOrderExample();
            WholesaleOrderExample.Criteria criteria1 =
example1.createCriteria();

    criteria1.andWholesaleOrdCodeLike("%PF"+year+MyDataFormatUtil.uni
tFormatTen(mon)+
                    MyDataFormatUtil.unitFormatTen(day)+"%");

    criteria1.andOrderStateEqualTo(BaseConstrant.WHOLESALE_ORDER_STAT
E_PAID);
            List<WholesaleOrder> wList =
wOrderMapper.selectByExample(example1);
            double sum = 0;
            //计算总金额
            for(int i=0;i<wList.size();i++){
                sum+=wList.get(i).getSumPrice();
            }
            //判断当日记录是否已存在
            OrderDayRec dayRec =
ordDayRecMapper.selectByPrimaryKey("PD"+year
                    +MyDataFormatUtil.unitFormatTen(mon)+
                    MyDataFormatUtil.unitFormatTen(day));
            if(dayRec==null){
                dayRec = new OrderDayRec();
                //设置信息
                dayRec.setDay(MyDataFormatUtil.unitFormatTen(day));
                dayRec.setMonth(MyDataFormatUtil.unitFormatTen(mon));
                dayRec.setYear(Integer.toString(year));

    dayRec.setOrdDayRecCode("PD"+year+MyDataFormatUtil.unitFormatTen(
mon)+
                        MyDataFormatUtil.unitFormatTen(day));
                dayRec.setOrderNum(wList.size());
                dayRec.setMoneyNum(sum);
```

```
                //添加到数据库
                if(ordDayRecMapper.insert(dayRec)==1)
                    return dayRec;
            }else{
                dayRec.setOrderNum(wList.size());
                dayRec.setMoneyNum(sum);
                if(ordDayRecMapper.updateByPrimaryKey(dayRec)==1)
                    return dayRec;
            }
        }
        return null;
    }
```

使用日记录生成月记录以及使用月记录生成年记录的方法相似，不过由于采购和批发订单的记录是一样的结构，所以生成月记录和年记录的代码可重用性比处理采购和批发订单生成日记录要高一些。

```
/**
     * 生成订单月记录
     * @param year
     * @param mon
     * @param type 订单种类，采购订单为'C'，批发订单为'P'
     * @return
     */
    public OrderMonRec createOrderMonRecord(int year, int mon,char type){
        //查找当月订单日记录
        OrderDayRecExample example = new OrderDayRecExample();
        OrderDayRecExample.Criteria criteria =
example.createCriteria();

    criteria.andOrdDayRecCodeLike("%"+type+"D"+year+MyDataFormatUtil.
unitFormatTen(mon)+"%");
        List<OrderDayRec> dList =
ordDayRecMapper.selectByExample(example);
        //计算总金额
        double money = 0;
        int num = 0;
        for(int i=0;i<dList.size();i++){
            money+=dList.get(i).getMoneyNum();
            num+=dList.get(i).getOrderNum();
        }
        //判断当月采购月记录是否存在
        OrderMonRec monRec =
ordMonRecMapper.selectByPrimaryKey(""+type+"M"+
                year+MyDataFormatUtil.unitFormatTen(mon));
        if(monRec != null){
            monRec.setMoneyNum(money);
            monRec.setOrderNum(num);
            if(ordMonRecMapper.updateByPrimaryKey(monRec)==1)
                return monRec;
        }else{
            //设置信息
            monRec = new OrderMonRec();
            monRec.setMonth(MyDataFormatUtil.unitFormatTen(mon));
            monRec.setYear(Integer.toString(year));

    monRec.setOrdMonRecCode(""+type+"M"+year+MyDataFormatUtil.unitFor
```

```
matTen(mon));
            monRec.setMoneyNum(money);
            monRec.setOrderNum(num);
            //添加到数据库
            if(ordMonRecMapper.insert(monRec)==1)
                return monRec;
        }
        return null;
    }
```

（2） 部分关键技术

订单记录全部更新算法：

上一部分提到的订单销售记录的生成都是指定年月日的，如果要对所有记录全部进行更新，可以遍历所有的日期或者所有的订单，此处选择了遍历所有的订单。

先从数据库中读取所有采购订单，按日期排序，然后开始遍历。选中一个订单，将其同一天的同种类的订单查找出来，生成一条采购订单日记录，若该日期的采购订单日记录已存在则更新其数据。再选中下一个订单，若日期与上一订单相同，则跳过该订单，日期不同则生成新的采购订单日记录，直到遍历完所有采购订单，批发订单同理。这样得到一个前半部分是采购订单日记录，后半部分是批发订单日记录的列表，且列表中的项是按日期排序的。

```
//获取全部采购订单
        PurchaseOrderExample example = new PurchaseOrderExample();
        PurchaseOrderExample.Criteria criteria =
example.createCriteria();
        example.setOrderByClause("time");//按日期顺序排列
        List<PurchaseOrder> pList =
pOrderMapper.selectByExample(example);

        List<OrderDayRec> dList = new ArrayList<OrderDayRec>();
        int purchaseDayRecNum =0;
        //计算全部采购订单日记录
        if(pList!=null&&pList.size()>0){
            Date d = pList.get(0).getTime();
            OrderDayRec dayRec =
createOrderDayRecord(Integer.parseInt(MyDataFormatUtil.getYear(d)),
                    Integer.parseInt(MyDataFormatUtil.getMonth(d)),
                    Integer.parseInt(MyDataFormatUtil.getDay(d)),"CG");
            if(dayRec!=null)dList.add(dayRec);
            for(int i=0;i<pList.size();i++){
                Date dtemp = pList.get(i).getTime();
                //若日期相同则不重复计算

    if(!MyDataFormatUtil.DateToStrYMD(d).equals(MyDataFormatUtil.Date
ToStrYMD(dtemp))){
                    d = dtemp;
                    dayRec =
createOrderDayRecord(Integer.parseInt(MyDataFormatUtil.getYear(d)),
```

```
    Integer.parseInt(MyDataFormatUtil.getMonth(d)),

    Integer.parseInt(MyDataFormatUtil.getDay(d)),"CG");
                    if(dayRec!=null)dList.add(dayRec);
                }
            }
            purchaseDayRecNum = dList.size();
        }

        //获取全部已付款批发订单
        WholesaleOrderExample example2 = new WholesaleOrderExample();
        WholesaleOrderExample.Criteria criteria2 =
example2.createCriteria();
        example2.setOrderByClause("time");

    criteria2.andOrderStateEqualTo(BaseConstrant.WHOLESALE_ORDER_STAT
E_PAID);
        List<WholesaleOrder> wList =
wOrderMapper.selectByExample(example2);
        if(wList!=null &&wList.size()>0){
            //计算全部批发订单日记录
            Date d = wList.get(0).getTime();
            OrderDayRec dayRec =
createOrderDayRecord(Integer.parseInt(MyDataFormatUtil.getYear(d)),
                    Integer.parseInt(MyDataFormatUtil.getMonth(d)),
                    Integer.parseInt(MyDataFormatUtil.getDay(d)),"PF");
            if(dayRec!=null)dList.add(dayRec);
            for(int i=0;i<wList.size();i++){
                Date dtemp = wList.get(i).getTime();

    if(!MyDataFormatUtil.DateToStrYMD(d).equals(MyDataFormatUtil.Date
ToStrYMD(dtemp))){
                    d = dtemp;
                    dayRec =
createOrderDayRecord(Integer.parseInt(MyDataFormatUtil.getYear(d)),

    Integer.parseInt(MyDataFormatUtil.getMonth(d)),

    Integer.parseInt(MyDataFormatUtil.getDay(d)),"PF");
                    if(dayRec!=null)dList.add(dayRec);
                }
            }
        }
```

接下去使用类似的方法遍历日记录列表，生成一个月记录列表，再遍历月记录列表，生成年记录列表。

```
//计算全部采购订单月记录
        int year = 0;
        int month = 0;
        List<OrderMonRec> mList = new ArrayList<OrderMonRec>();
        int i=0;
        if(dList.size()>=purchaseDayRecNum){
            for(;i<purchaseDayRecNum;i++){
                int tYear = Integer.parseInt(dList.get(i).getYear());
                int tMonth = Integer.parseInt(dList.get(i).getMonth());
                if(year != tYear || month != tMonth){
```

```
                    year = tYear;
                    month = tMonth;
                    OrderMonRec monRec = createOrderMonRecord(year, month,
'C');
                    if(monRec!=null)mList.add(monRec);
                }
            }
        }
        int purchaseMonRecNum = mList.size();
        //计算全部批发订单月记录
        year = 0;
        month = 0;
        if(dList.size()>=purchaseDayRecNum){
            for(;i<dList.size();i++){
                int tYear = Integer.parseInt(dList.get(i).getYear());
                int tMonth = Integer.parseInt(dList.get(i).getMonth());
                if(year != tYear || month != tMonth){
                    year = tYear;
                    month = tMonth;
                    OrderMonRec monRec = createOrderMonRecord(year, month,
'P');
                    if(monRec!=null)mList.add(monRec);
                }
            }
        }
```

AJAX 技术：

系统中另一个关键技术是使用 AJAX 技术在前端与后台之间传递 JSON 字符串，以登录过程为例。

点击登录按钮，调用 login()函数（JavaScript 函数），发送数据到服务器，接受到服务器的响应数据后，进行局部刷新或者跳转操作。

```
function login() {
    //获取信息
    var name = $("#username").val();
    var pwd = $("#password").val();
    var role = document.getElementsByName("role");
    var roleval = "";
    for(var i=0; i<role.length; i ++){
        if(role[i].checked){
        roleval = role[i].value;
            break;
        }
    }
    //发送信息
    $.ajax({
        url : "./user/login",
        type : "POST",
        dataType:"json",
        contentType : "application/json;charset=UTF-8",
        //发送的数据应为JSON字符串格式
        data : JSON.stringify({
        username:name,
        password:pwd,
        role:roleval
        }),
```

```
        //回调函数
        success:function(result) {
            if(result.success){
             alert("登录成功");
             $("#loginCheck").text("查询成功" + result.message);
             if(result.message == "admin")
                 var href =
'${pageContext.request.contextPath}/service/adminIndex.jsp';
             else if(result.message == "customer")
                 var href =
'${pageContext.request.contextPath}/service/customer/customerIndex.js
p';
                 window.top.location.href = href;
             }else{
             alert("查询不到");
             $("#loginCheck").text(result.message);
             }
        },
        error:function(result){
         alert(result);
            $("#loginCheck").text("查询失败");
        }
    });
```

对应的 UserController 中的函数如下方代码所示，框架自动将接收到的 JSON 字符串转化为对象，操作执行完毕后的响应数据也可自动转换。

```
    @RequestMapping("/login")
    @ResponseBody
    public JSONObject login(@RequestBody User
user,HttpServletRequest request){
        JSONObject json = new JSONObject();
        Customer customer = null;
        Admin admin = null;
        try{
            if(user.getRole().equals("customer")){
                customer =
customerService.login(user.getUsername(),user.getPassword());
                if(customer!=null){
                    json.put("success", true);
                    json.put("message", "customer");
                    request.getSession().setAttribute("customer",
customer);
                }
            }else if(user.getRole().equals("admin")){
                admin =
adminService.login(user.getUsername(),user.getPassword());
                if(admin!=null){
                    json.put("success", true);
                    json.put("message", "admin");
                    System.out.println(json.toString());
```

```
                    request.getSession().setAttribute("admin",
admin);
                }
            }
        }catch(BusinessException be){
            be.printStackTrace();
            json.put("success", false);
            json.put("message", be.getMessage());
        }catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            json.put("success", false);
            json.put("message", "失败");
        }
        return json;
    }
```

第六章 布料批发管理系统测试

6.1 测试范围及结果

主要对版本 1 中的功能进行了测试。
重要级：Urgent、veryhigh、high、medium、low，其中 urgent 为最重要

表 6-1 测试范围及结果

一级模块	二级模块	三级模块	重要级	测试结果
客户模块	搜索查看产品信息	产品展示	Veryhigh	通过
		产品搜索	high	通过
	客户信息管理	客户基础信息管理（注册）	veryhigh	通过
		查看客户订单	veryhigh	通过
		查看客户款项流水信息	high	通过
批发公司模块	批发模块	批发下单	Urgent	通过
		查看批发订单	veryhigh	通过
		查看批发款项流水信息	high	通过
	采购模块	采购开单	Urgent	通过
		查看采购订单	veryhigh	通过
		查看采购款项流水信息	high	通过
	仓储模块	产品出入库	Urgent	通过
		查看出入库流水信息	high	通过
	基础信息管理	管理员信息管理	high	通过
		供应商信息管理	veryhigh	通过
		产品信息管理（详细信息、价格）	veryhigh	未实现产品属性类别的管理
		仓库基础信息管理	veryhigh	通过
		查看财务统计信息（采购、批发）	veryhigh	通过

6.2 重点功能测试截图

采购开单：

使用采购管理员身份登录系统，进行采购开单，选择供应商 GYS01 创建采购订单单，填写第一个订单项信息并提交，之后可在同一界面继续提交同一订单

的其他订单项。

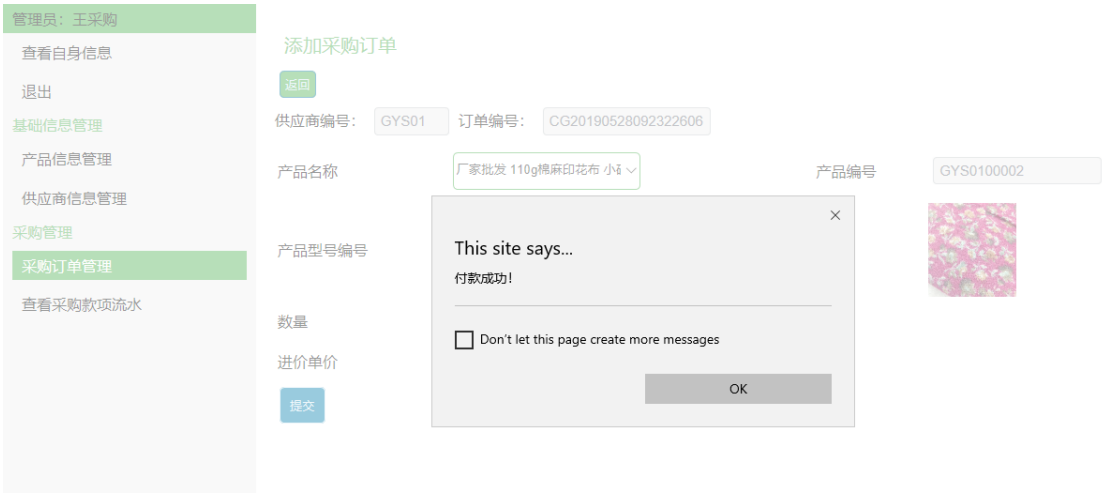

图 6-1　提交采购订单成功

采购入库：

以采购管理员身份登录系统，进行采购入库申请，查找到相应订单项后点击申请入库。

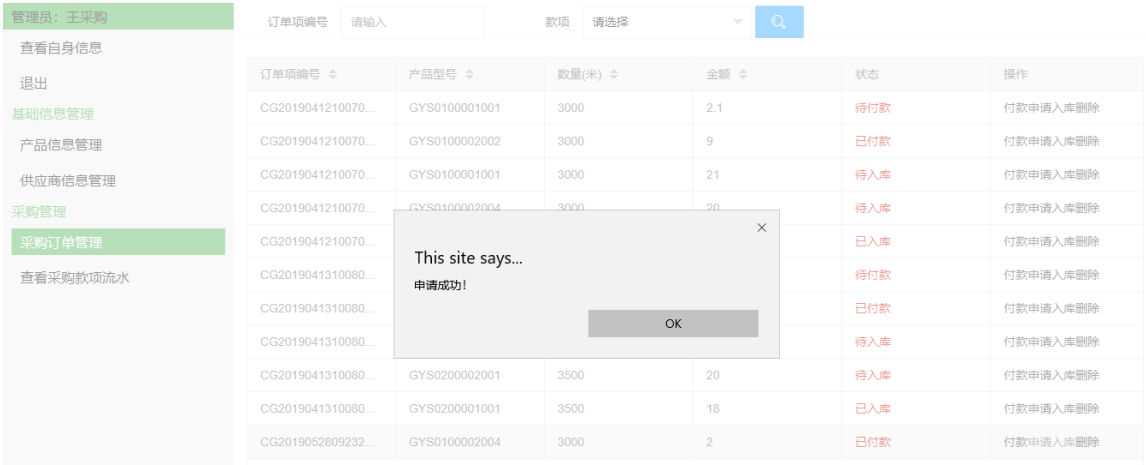

图 6-2　采购入库申请成功

以仓库管理员身份登录系统，进行采购入库，输入采购订单编号查找该订单中待入库的采购订单项，点击入库按钮，填入产品将进入的仓库的编号，点击确定。

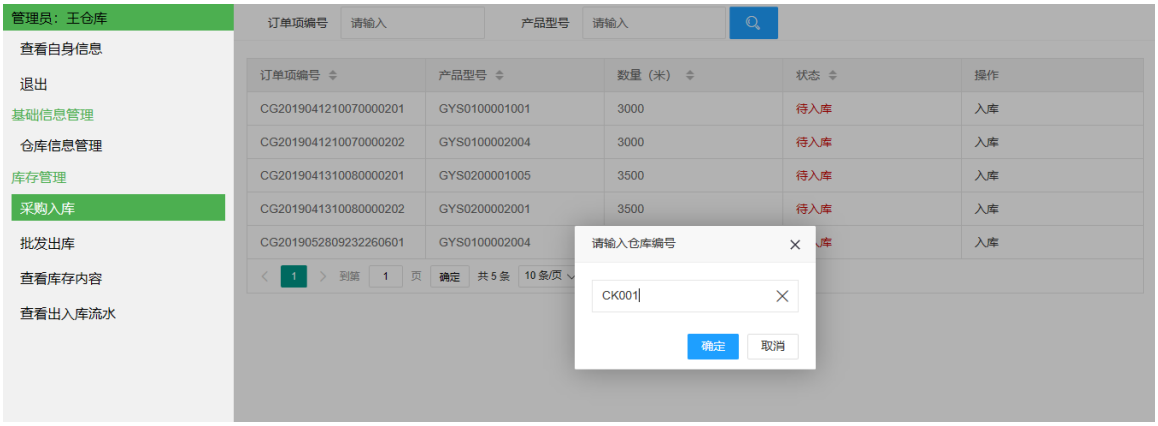

图 6-3　采购入库输入仓库编号

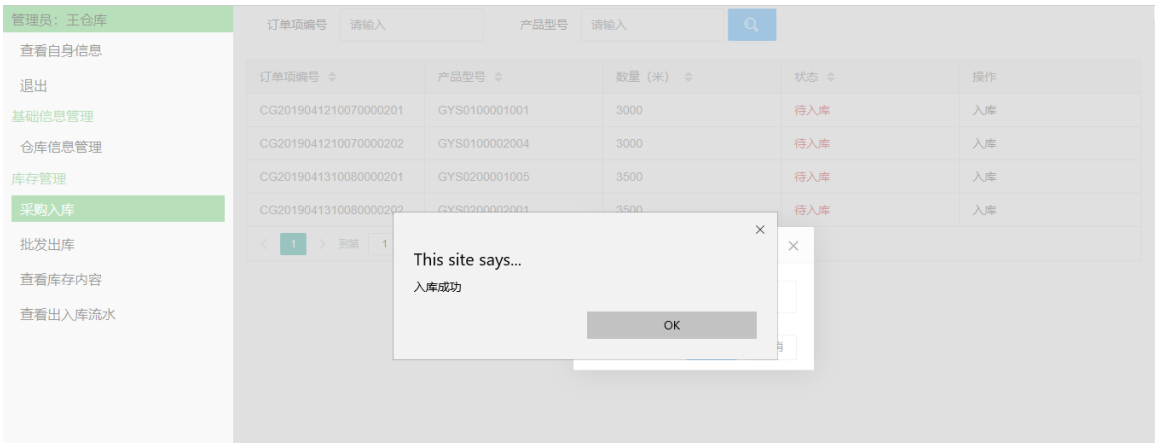

图 6-4　采购入库成功

批发下单：

以客户身份登录系统，查看产品，选择产品型号进行下单，选择收货地址，提交批发订单，进行付款。

图 6-5　选择产品型

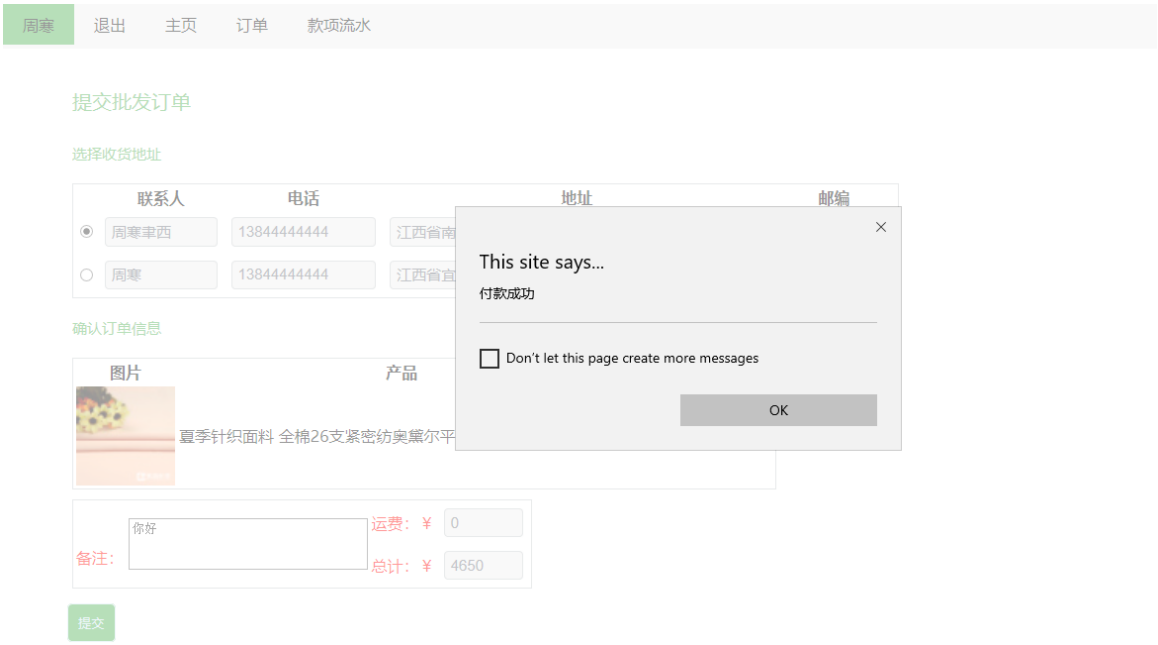

图 6-6 批发下单付款成功

发货出库：

以批发管理员身份登录系统，找到相应订单项，进行批发出库申请。

以仓库管理员身份登录系统，进行批发出库，输入批发订单编号，查找待出库批发订单项，输入储存该产品的仓库编号，点击确定。该批发订单的全部订单项出库完毕后，点击该批发订单出库完成，修改订单状态。

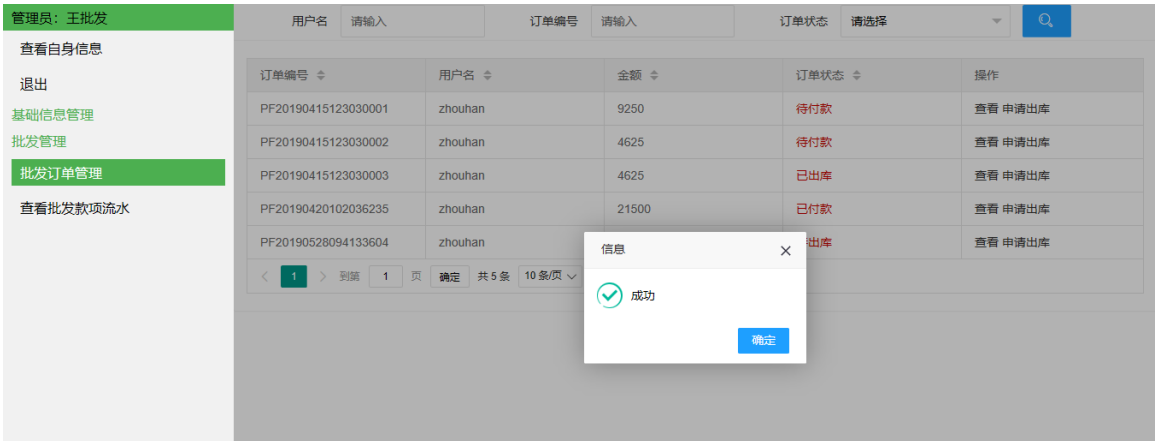

图 6-7 批发出库成功

财务统计：

以高级管理员身份登录系统，进行查看订单购销记录，点击更新全部数据记录，分别选择查看年、月、日记录，点击查找，得到相应记录。

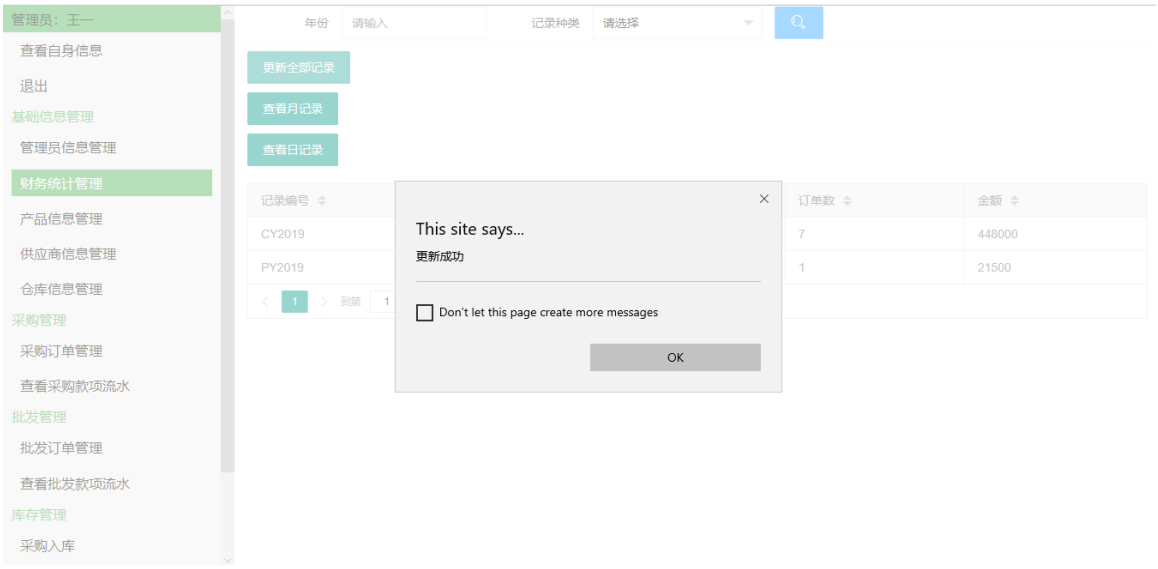

图 6-8 更新全部记录成功

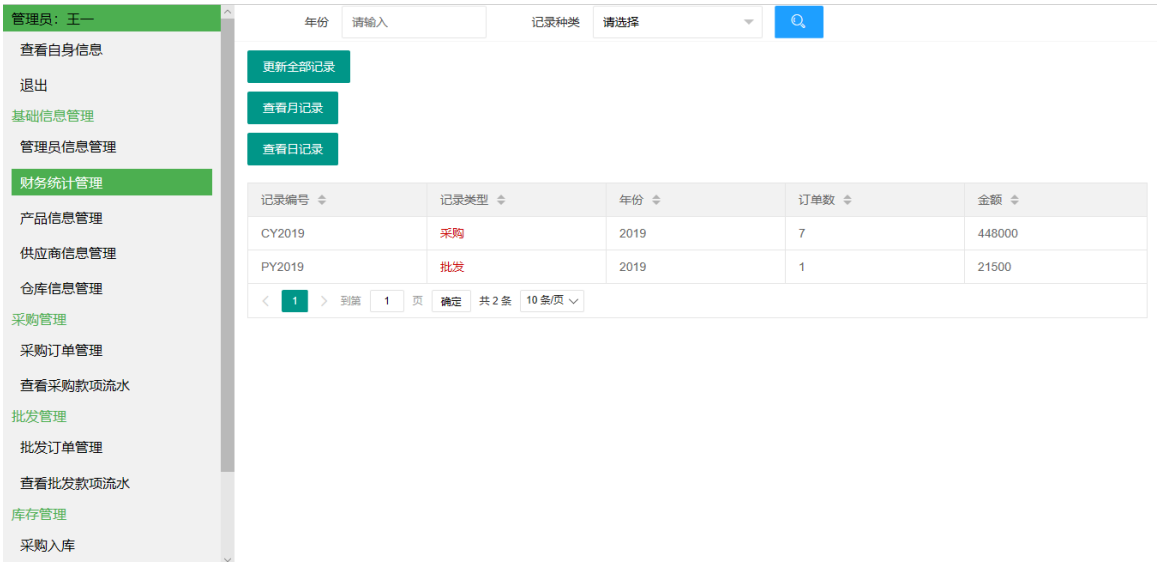

图 6-9 查看订单购销年记录

结论

本选题是设计并实现一个局域网内使用的布料批发管理系统，主要使用了SSM 框架和 AJAX 技术，能完成布料批发流程中的采购、入库、批发销售、出库和财务统计的功能，将布料线上批发销售与批发公司的内部业务管理进行有机的结合。

使用 SSM 框架能对 J2EE 平台上的开发进行有效地管理，注解和并入 spring 框架容器中的工具功能强大，使用方便，能大大提升开发效率和开发效果，但是构建 SSM 框架时，要求对 spring、spring-mvc 和 mybatis 框架分别都有一定的了解，各项配置比较繁杂，在熟悉程度上有要求。使用 AJAX 技术和 JSON 字符串可以对前后台数据之间的传递进行简化和格式的统一，用户使用网页时，流畅性和交互性较强，用户体验比较好，不过使用这种方式在开发时要配套使用数据传输对象（DTO）以及视图对象（VO）会比较合适。

本选题中完成的系统对布料批发采购功能的实现仅仅只达到了演示水平，如果要在实际工作生产中投入使用，还有很多改进之处。首先在客户模块还需要完成基本的客户进货单管理、热门产品统计推荐功能，在批发公司模块还需要完成导出、打印报表信息单据的功能，现在仅仅只能线上浏览以订单为基础的财务统计信息以及各种流水信息。其次需要构建客户与客服的线上对话交流答疑功能，以及调用真实的线上支付接口以完成支付功能。最后需要对系统的并发性进行维护，现在仅仅在数据库设计上留下进行一点并发操作的余地，如订单编号的设置允许每秒少于 1000 个订单生成。

综上所述，本选题使用 SSM 框架和 AJAX 技术实现了一个布料批发管理系统，完成了布料批发操作的基本功能，达到了可演示的程度，如果要投入实际使用，还需要在很多方面做出改进。

参考文献

[1] 李洋.SSM 框架在 Web 应用开发中的设计与实现[J].计算机技术与发展,2016,26(12):190-194.

[2] 李云云.浅析 B/S 和 C/S 体系结构[J].科学之友，2011(01)：6-8

[3] 龚华.基于 Java 的 Web 应用设计与开发[D].西南石油学院,2003.

[4] 岳昆，王晓玲，周傲英.Web 服务核心支撑技术：研究综述[J].软件学报，2004(03)：428-442.

[5] 电子商务[EB/OL] 使用电子工具从事商务活动. (2019-03-03) [2019-03-15]. https://baike.baidu.com/item/电子商务/98106

[6] 徐 雯, 高建华. 基于 Spring MVC 及 MyBatis 的 Web 应用框架研究[J]. 微型电脑应用, 2012, 28(7): 1~4

[7] 游丽贞，郭宇春，李纯喜.Ajax 引擎的原理和应用[J].微计算机信息,2006(06):205-207.

[8] 文欢欢, 刘振宇, 吴 霖. 基于 Mybatis 和 JDBC 的分页查询研究[J]. 电脑知识与技术, 2015, 11(25): 165~167

[9] Woychowsky E. AJAX: Creating Web Pages with Asynchronous JavaScript and XML Creating Web Pages with Asynchronous JavaScript and XML[M]. Prentice Hall Press, 2011

[10] 刘高军, 夏景隆. 基于 Spring MVC 和 iBATIS 框架的研究与应用[J]. 计算机安全, 2012(7): 25~30

致谢

在写作本文的过程中，许多师长、同学和朋友都给予了笔者不同方面的帮助。

在此，笔者感谢欧阳皓老师在课题的功能设计和需求分析方面的指导，以及在论文格式方面的帮助，最重要的还是在笔者写作进度方面的监督。同时，还要感谢吴沛旋同学和陈孝翰学长在编程技术和测试方面的帮助，赵书函同学在文档处理方面的协助和支持。最后，要感谢笔者的好友肖琰，在笔者进行毕业设计这段时间的心理支持、鼓励和开导。

感谢以上所有人在百忙之余对笔者的帮助，没有你们的支持，笔者无法顺利地完成毕业设计。最后，再次感谢你们对这篇论文的贡献和对笔者的帮助。

附录

术语表

名词	解释
进货单	客户想购买的产品项列表，类似于一般零售电商网站的购物车功能
产品属性类别	布料有多种属性类型，如成分类型、纺织工艺等，每种类型又有不同的属性，如成分类型有棉织物、麻织物、毛织物等等
分梯度价格	批发业务中，一般购买得越多单价越便宜，故价格有梯度
幅宽	布料的宽度
染整工艺	布料的染色、印花、整理工艺
克重	每平方米布料的重量，单位为：克每平方米（g/m^2）
纱支	描述纱线粗细的单位，指单位重量（1 磅）的纱线的长度，若 1 磅的棉纱长度为 10*840 码（0.9144 米/码），则该种纱线的纱支为 10 支，记做 10S

数据字典

P-主键，F-外键，M-非空

供应商表（supplier）

数据项名	代码	数据类型	P	F	M	说明
供应商编号	supplierCode	char(5)	√		√	前 3 位为供应商名称缩写，最后 2 位为缩写雷同时的区分数字数字，从 01 开始
供应商	supplierName	varchar(50)			√	
联系人	contactName	varchar(10)			√	
手机号码	mobile	varchar(15)			√	
电子邮箱	email	varchar(30)			√	
联系地址	contactAddr	varchar(100)			√	
邮政编码	postcode	char(6)			√	

固定电话	phone	varchar(16)				
传真	fax	varchar(16)				
备注	remark	varchar(200)				

管理员（admin）

数据项名	代码	数据类型	P	F	M	说明
管理员编号	adminCode	char(5)	√		√	前两位字符为类型，后 3 位为数字
管理员姓名	adminName	varchar(10)			√	
手机号码	mobile	varchar(15)			√	
电子邮箱	email	varchar(30)			√	
密码	password	varchar(30)			√	由大小写字母数字及标点组成
管理员类型	adminType	char(2)			√	GJ：高级管理员 CK：仓库管理员 CG：采购管理员 PF：批发管理员 XT：系统自动
管理员状态	state	char			√	T：启用 F：不启用

仓库（warehouse）

数据项名	代码	数据类型	P	F	M	说明
仓库编号	warehouseCode	char(4)	√		√	
容量	capacity	integer			√	单位为米（布料长度）
仓库状态	state	char			√	T：启用 F：不启用
当前库存量	inventory	integer			√	单位为米（布料长度）

库存内容（inventoryContent）

数据项名	代码	数据类型	P	F	M	说明

库存内容序号	inventoryContInd	bigint	√		√	自增序列
仓库编号	warehouseCode	char(4)		√	√	
产品型号编号	modelCode	char(13)		√	√	
内容数量	ContentSum	integer			√	单位为米（布料长度）

产品（product）

数据项名	代码	数据类型	P	F	M	说明
产品编码	productCode	char(10)	√		√	前 5 位为供应商编号，后五位为产品数字编号
产品名称	productName	varchar(100)			√	初始时与货源名称相同，上架之前一般需进行修改
货源名称	originalName	varchar(50)			√	供应商对产品的叫法，可与正式名称不同
进价单价	originalPrice	double			√	每米布料的价格
型号总数	typeTotal	smallint			√	该产品的型号数，一般为颜色种类数
幅宽	width	integer			√	单位为厘米
克重	weight	integer			√	单位为克每平方米
纱支	thickness	varchar(10)				
成分类型	composition	varchar(10)				
原料及含量	ingredient	varchar(30)				
密度	density	varchar(10)				
纺织工艺	textileProcess	varchar(10)				
染整工艺	dyeProcess	varchar(10)				
织物组织	fiberOrganization	varchar(10)				
图案	pattern	varchar(20)				
主要用途	mainUse	varchar(30)				
适用季节	season	varchar(5)				

运输费用	shippingFee	float				
产品简介	intro	varchar(50)				
产品状态	state	char			√	T：上架 F：下架

产品型号（productModel）

数据项名	代码	数据类型	P	F	M	说明
产品型号编号	modelCode	char(13)	√		√	产品编号-XXX（XXX 为序号，每个产品从 001 开始计数，000 用于表示通用信息）
产品编码	productCode	char(10)		√	√	
型号简称	modelName	varchar(15)			√	
数量	totalNum	integer			√	

产品图片（picture）

数据项名	代码	数据类型	P	F	M	说明
文件名	filename	varchar(25)	√		√	产品编码-产品型号序号-图片序号
图片序号	pictureIndex	smallint			√	每个产品型号从 1 开始
产品型号编号	modelCode	char(13)		√	√	

产品价格（sellPrice）

数据项名	代码	数据类型	P	F	M	说明
产品价格编号	sellPriceCode	char(11)	√		√	产品编码-X（X 为序号，每个产品从 1 开始）
产品编码	productCode	char(10)		√	√	
下限	lowerLimit	integer			√	
单价	price	float			√	

采购订单（purchaseItem）

数据项名	代码	数据类型	P	F	M	说明
采购订	purchaseOrdCod	char(19)	√		√	CG（采购）

单编号	e					+yyyyMMddhhmmss+SSS
供应商编号	supplierCode	char(5)		√	√	
管理员编号	adminCode	char(5)		√	√	
创建时刻	time	timestamp			√	
订单总金额	sumPrice	double			√	所有订单项的总金额
已付款金额	paidPrice	double			√	
备注	remark	varchar(100)				

采购订单项（purchaseItem）

数据项名	代码	数据类型	P	F	M	说明
采购订单项编号	purchaseItemCode	char(21)	√		√	采购订单编号+XX（XX 为序号，从 1 开始）
采购订单编号	purchaseOrdCode	char(19)		√	√	
产品型号编号	modelCode	char(13)		√	√	
数量	totalNum	integer			√	
进价单价	orignalPrice	float			√	
单项总价	totalPrice	double			√	该订单项的总金额
状态	state	char(2)			√	DF：待付款 YF：已付款 DR：待入库 YR：已入库

批发商款项流水（comMonIORecord）

数据项名	代码	数据类型	P	F	M	说明
批发商款项流水编号	comIORecCode	char(19)	√		√	C（采购）/P（批发）+I（收款）/O（付款）+yyyyMMddhhmmss+SSS
管理员编号	adminCode	char(5)		√	√	处理该项款项的管理员
款项发生原因	reason	char(2)			√	CG：采购 PF：批发 QX：批发订单取消 TH：退货 QT：其他（需在备注中注明）
出入标识	IOmark	char			√	I：收款 O：付款
出入时刻	time	timestamp			√	
金额	amount	double			√	收款为正，付款为负
订单编号	ordCode	char(18)				与该项流水相关的订单编号，若流水与订单无关可为空
备注	remark	varchar(100)				

出入库流水记录（warehouseIORecord）

数据项名	代码	数据类型	P	F	M	说明
出入库记录编号	warehouseIORecCode	char(19)	√		√	CK（出库）/RK（入库）yyyyMMddhhmmss+SSS
仓库编号	warehouseCode	char(4)		√	√	
产品型号编号	modelCode	char(13)		√	√	
管理员编号	adminCode	char(5)		√	√	
数量变动	numberChange	integer			√	
变动原因	reason	char(2)			√	CG：采购 PF：批发 TH：退货 HH：换货 QT：其他（在备注中注明）
出入标识	IOMark	char			√	I：入库 O：出库

出入时刻	time	timestamp			√	
订单编号	ordCode	char(19)				与该项流水相关的订单编号，若流水与订单无关可为空
备注	remark	varchar(100)				

客户基础信息（customer）

数据项名	代码	数据类型	P	F	M	说明
客户用户名	username	varchar(15)	√		√	
客户姓名	customerName	varchar(10)			√	
密码	password	varchar(30)			√	由大小写字母数字及标点组成
客户单位	customerCompany	varchar(50)				
手机号码	mobile	varchar(15)			√	
电子邮箱	email	varchar(30)			√	
联系地址	contactAddr	varchar(100)			√	
邮政编码	postcode	char(6)			√	
固定电话	phone	varchar(16)				
传真	fax	varchar(16)				

客户收货地址（shippingAddress）

数据项名	代码	数据类型	P	F	M	说明
收货地址序号	shippingAddrNum	bigint	√		√	自增
客户用户名	username	varchar(15)		√	√	
收货地址	shippingAddr	varchar(100)			√	
邮政编码	postcode	char(6)			√	
联系人姓名	contactName	varchar(10)			√	

手机号码	mobile	varchar(15)			√	
电子邮箱	email	varchar(30)				

客户进货单项（listItem）

数据项名	代码	数据类型	P	F	M	说明
进货单项序号	listItemCode	varchar(17)	√		√	客户用户名+XX（XX 为序号，每个进货单从 1 开始）
产品型号编号	modelCode	char(13)		√	√	
客户用户名	username	varchar(15)		√		
数量	totalNum	integer				

批发订单（wholesaleOrder）

数据项名	代码	数据类型	P	F	M	说明
批发订单编号	wholesaleOrdCode	char(19)	√		√	PF（批发）+yyyyMMddhhmmss+SSS
收货地址序号	shippingAddrNum	bigint		√	√	
客户用户名	username	varchar(15)		√	√	
订单状态	orderState	char(2)			√	DF：待付款 YF：已付款 QX：已取消 FH：已发货 TH：退货 HH：换货
物流单号	shippingCode	varchar(20)				
创建时刻	time	timestamp			√	

批发订单项（wholesaleItem）

数据项名	代码	数据类型	P	F	M	说明
批发订单项编号	wholesaleItemCode	char(21)	√		√	批发订单编号-XX（XX 为序号，每个订单从 1 开始）

批发订单编号	wholesaleOrdCode	char(19)		√	√	
产品型号编号	modelCode	char(13)		√	√	
数量	totalNum	integer			√	
单价	price	float			√	
单项总价	totalPrice	double			√	
状态	state	char(2)			√	ZC：正常 FH：已发货 TH：已退货 HH：已换货

批发订单取消记录（wholesaleCancelRec）

数据项名	代码	数据类型	P	F	M	说明
批发取消记录序号	cancelRecNum	bigint	√		√	
管理员编号	adminCode	char(5)		√		
批发订单编号	wholesaleOrdCode	char(18)		√	√	
取消理由	reason	varchar(100)			√	
记录状态	recordState	char(2)			√	DQ：待确认 YQ：已确认 JJ：已拒绝 QX：已取消
款项状态	moneyState	char			√	T：已退款 F：未退款
备注	remark	varchar(100)				

批发订单退货记录（wholesaleReturnRec）

数据项名	代码	数据类型	P	F	M	说明
批发退货记录序号	returnRecNum	bigint	√		√	
管理员编号	adminCode	char(5)		√	√	
批发订单项编号	wholesaleItemCode	char(21)		√	√	

数量	returnNum	integer			√	
退货理由	reason	varchar(100)			√	
物流单号	shippingCode	varchar(20)				
记录状态	recordState	char(2)			√	DQ：待确认 YQ：已确认 JJ：已拒绝 QX：已取消
货物状态	productState	char			√	T：已返还 F：未返还
款项状态	moneystate	char			√	T：已退款 F：未退款
备注	remark	varchar(100)				

批发订单换货记录（wholesaleExchangeRec）

数据项名	代码	数据类型	P	F	M	说明
批发换货记录序号	exchangeRecNum	bigint	√		√	
管理员编号	adminCode	char(5)		√	√	
批发订单项编号	wholesaleItemCode	char(21)		√	√	
数量	exchangeNum	integer			√	
换货理由	reason	varchar(100)			√	
退回物流单号	backShipCode	varchar(20)				
重发物流单号	resendShipCode	varchar(20)				
记录状态	recordState	char(2)			√	DQ：待确认 YQ：已确认 JJ：已拒绝 QX：已取消
换货状态	state	char			√	W：未退还 B：已退还 S：已重发
备注	remark	varchar(100)				

客户款项流水（cusMonIORec）

数据项名	代码	数据类型	P	F	M	说明
客户款项流水编号	custMonIORecCode	char(19)	√		√	K（客户）+I（收款）/O（付款）+yyyyMMddhhmmss+SSS
批发订单编号	wholesaleOrdCode	char(19)		√	√	
客户用户名	username	varchar(15)		√		
款项发生原因	reason	char(2)			√	PF：批发 QX：取消订单 TH：退货 QT：其他
出入标识	IOmark	char			√	I：收款 O：付款
出入时刻	time	timestamp			√	
金额	money	double			√	
备注	remark	varchar(100)				

产品属性及类别（productProperty）

数据项名	代码	数据类型	P	F	M	说明
产品属性序号	productPropNum	bigint	√		√	自增序列
产品属性类别	productPropType	varchar(10)			√	产品该属性所属的类别，如纺织工艺、染整工艺
产品属性	productProp	varchar(10)			√	产品的具体属性，如针织（纺织工艺的一种）

省份（province）

数据项名	代码	数据类型	P	F	M	说明
省份序号	provinceNum	integer	√		√	省份序号
省份名称	province	varchar(10)			√	省份名称

城市（city）

数据项名	代码	数据类型	P	F	M	说明
城市序号	cityNum	int	√		√	城市序号
省份序号	provinceNum	integer		√	√	省份序号
城市	city	varchar(20)			√	城市

菜单（menu）

数据项名	代码	数据类型	P	F	M	说明
菜单序号	menuNum	int	√		√	自增序列
一级菜单	Amenu	varchar(20)			√	
二级菜单	Bmenu	varchar(20)			√	
状态	state	char			√	T：启用 F：不启用

权限（permission）

数据项名	代码	数据类型	P	F	M	说明
权限序号	permissionNum	integer	√		√	自增序列
菜单序号	menuNum	int		√	√	
角色	role	char(2)			√	GJ：高级管理员 CK：仓库管理员 CG：采购管理员 PF：批发管理员

购销订单日记录（orderDayRec）

数据项名	代码	数据类型	P	F	M	说明
订单日记录编号	ordDayRecCode	char(10)	√		√	C（采购）/P（批发）+D（日记录）+yyyyMMdd
年	year	char(4)			√	
月	month	char(2)			√	
日	day	char(2)			√	
订单总数	orderNum	integer			√	
金额总数	moneyNum	double			√	

购销订单月记录（orderMonRec）

数据项名	代码	数据类型	P	F	M	说明
订单月记录编号	OrdMonRecCode	char(8)	√		√	C（采购）/P（批发）+M（月记录）+yyyyMM
年	year	char(4)			√	
月	month	char(2)			√	
订单总数	orderNum	integer			√	
金额总数	moneyNum	double			√	

购销订单年记录（orderYearRec）

数据项名	代码	数据类型	P	F	M	说明
订单年记录编号	ordYearRec	char(6)	√		√	C（采购）/P（批发）+Y（年记录）+yyyy
年	year	char(4)			√	
订单总数	orderNum	integer			√	
金额总数	moneyNum	double			√	