

编号(09)

复旦大学高等教育自学考试

(本科)

毕业论文

题目 小水果信息管理系统

专 业 计 算 机 软 件

作 者 霍 强

准考证号 374313130646

指导教师 于 玉

完成日期 2016 年 11 月 25 日

小水果信息管理系统

摘要

主题 [互联网、营销、B2C、电商、水果商城]

1946 年 2 月 14 日，世界上第一台电脑 ENIAC 在美国宾夕法尼亚大学诞生。经过 70 年的发展，计算机先后经历了电子管计算机、晶体管计算机、集成电路计算机、大规模集成电路计算机、人工智能计算机等。随着计算机及计算机技术的发展，这个伟大的发明正在悄然的改变着我们的世界。互联网，移动互联网、互联网+ 时代，我们不断的享受到它给我们带来的便利。在家办公、在家购物等以前我们无法想象的现在已经变为现实，甚至我们可以拿着手机随时随地的观看和购买自己喜欢的商品。

在计算机技术飞速发展、互联网思维不断深入人心，我们已经习惯了计算机给我们带来的便利。那么小水果就是在这个时代的一个为人提供便利的产品，它可以让你在家可以买到自己最爱的水果，不仅有丰富多彩的国产水果，还有多式多样的进口水果，价格最优惠，物美价廉。

业务层面，本系统对外消费者提供商品的浏览、搜索、查看、加入购物车、结算等功能。对内，本系统具有自己的后台管理系统，分基础信息管理和业务信息管理两部分。基础信息管理主要包括菜单管理、权限管理、员工管理、安全信息管理等。业务信息管理主要包括商品上下架管理、销售接单管理、仓库管理、采购管理、入库管理、出库管理等。

技术层面，本系统主要采用了目前互联网最常用的技术。采用 Java 语言来编写服务端，数据库管理系统采用 mysql。后端框架用 Spring + SpringMVC + MyBatis，做到了数据持久层与业务逻辑层及事物管理层的分离，让系统的整体编码层次结构清晰，可读性高。前端采用 Freemarker 视图解析，减少前端代码开发工作量。数据库连接采用阿里的数据库连接池 (com.alibaba.druid.pool.DruidDataSource) 工具，减少操作数据库时建立数据库连接所消耗的时间。数据库采用 master+slave 主备模式，对实时性要求不是很高的查询可走备库，主库与备库之间进行数据的准实时同步。同时，当主库出现故障，可进行备库的切换，此时备库升级为主库。

目录

小水果信息管理系统.....	1
摘要.....	1
第一章 序言.....	1
1.1 组织机构概况.....	1
1.2 项目开发背景及信息系统目标.....	1
1.3 信息系统范围.....	1
1.4 项目开发方法和计划.....	1
1.5 作者的主要贡献.....	1
第二章 系统需求分析.....	2
2.1 现行业务系统描述.....	2
2.1.1 组织结构图介绍.....	2
2.1.2 业务流程分析.....	3
2.2 现行系统主要问题分析.....	4
2.3 解决方案.....	4
2.3.1 方案 1.....	4
2.3.2 方案 2.....	4
2.3.3 方案 3.....	4
2.3.4 结论.....	4
2.4 可行性分析.....	5
2.4.1 技术可行性分析.....	5
2.4.2 经济可行性分析.....	5
2.4.3 营运可行性分析.....	5
第三章 新系统逻辑方案.....	6
3.1 用例(用况)建模.....	6
3.1.1 小水果网信息管理系统用例示例.....	6
3.1.2 CRM 信息管理子系统.....	7
3.1.3 商品信息管理系统.....	7
3.1.4 订单信息管理子系统.....	7
3.1.4.1 订单信息管理子系统用例图.....	7
3.1.4.2 订单信息管理子系统用例文档.....	7
3.1.5 支付信息管理子系统.....	9
3.1.6 储运信息管理子系统.....	9
3.1.6.1 储运信息管理子系统用例图.....	9
3.1.6.2 储运信息管理子系统用例文档.....	10
3.2 包图(系统包图).....	11
3.3 静态建模.....	13
3.3.1 类图.....	13
3.3.1.1 关联图.....	13
3.3.1.2 类的定义.....	13
3.3.2 对象图.....	20
3.3.2.1 销售对象图.....	20

3.3.2.2 采购对象图	21
3.3.2.3 入库对象图	22
3.2.3.4 出库单	23
3.4 动态建模	23
3.4.1 状态图	23
3.4.1.1 商品状态图	23
3.4.1.2 订单状态图	24
3.4.1.3 用户状态图	24
3.4.2 时序图	25
3.4.2.1 添加购物车时序图	25
3.4.2.2 用户下单时序图	26
3.4.2.3 出库时序图	27
3.4.3 协作图	28
3.4.3.1 添加购物车协作图	28
3.4.3.2 下单协作图	28
3.4.3.3 出库协作图	29
3.4.4 活动图	30
3.4.4.1 添加购物车活动图	30
3.4.4.2 用户下单活动图	30
3.4.4.3 出库活动图	31
第四章 系统总体设计	32
4.1 软件模块结构设计	32
4.1.1 HIPO 分层图	32
4.1.2 IPO 图	33
4.1.2.1 树根节点 IPO 图	33
4.1.2.2 内部节点 IPO 图	33
4.1.2.3 树叶节点 IPO 图	34
4.2 数据库设计	34
4.2.1 需求分析	34
4.2.1.1 消费者需求	34
4.2.1.2 工作人员需求	35
4.2.2 实体描述	35
4.2.3 联系描述	35
4.2.3.1 一对一的二元关系有 :	35
4.2.3.2 一对多的二元关系有 :	36
4.2.3.3 多对多的二元关系有 :	36
4.2.4 ER 图	37
4.2.5 转换规则	37
4.2.5.1 实体类型的转换	37
4.2.5.2 联系类型的转换	37
4.2.6 关系模式	42
4.2.7 数据库表	43
4.2.7.1 员工表	43
4.2.7.2 消费者表	43

4.2.7.3 商品表.....	44
4.2.7.4 收货地址表.....	44
4.2.7.5 购物车表.....	45
4.2.7.6 购物车明细表.....	45
4.2.7.7 销售订单表.....	45
4.2.7.8 销售订单明细表.....	45
4.2.7.9 供应商表.....	46
4.2.7.10 仓库表.....	46
4.2.7.11 库存表.....	46
4.2.7.12 采购单表.....	47
4.2.7.13 采购单明细表.....	47
4.2.7.14 出库单表.....	47
4.2.7.15 出库单明细表.....	48
4.2.7.16 入库单表.....	48
4.2.7.17 入库单明细表.....	48
4.2.7.16 补货单表.....	49
4.2.7.17 补货单明细表.....	49
4.3 计算机系统配置方案.....	49
4.3.1 计算机系统硬件配置.....	49
4.3.2 计算机系统软件配置.....	49
4.4 系统安全性、可靠性方案.....	50
4.4.1 系统安全性.....	50
4.4.2 系统可靠性.....	50
4.4.2.1 环境可靠性.....	50
4.4.2.2 架构及代码可靠性.....	50
第五章 系统详细设计.....	50
5.1 代码设计.....	50
5.1.1 销售单代码设计.....	50
5.1.2 采购单代码设计.....	51
5.1.3 商品代码设计.....	51
5.2 用户界面设计.....	51
5.2.1 登录注册界面设计.....	51
5.2.2 首页界面设计.....	53
5.2.3 购物车列表界面设计.....	54
5.2.4 采购管理界面设计.....	54
5.2.5 出库管理界面设计.....	55
5.2.6 入库管理界面设计.....	56
5.3 模块处理设计.....	56
5.3.1 登录模块处理设计.....	56
5.3.2 首页模块处理设计.....	60
5.3.2.1 首页脚本.....	60
5.3.2.2 页面 inner_goods_list.ftl 脚本.....	62
5.3.2.3 搜索后台处理.....	63
5.3.3 购物车列表模块处理设计.....	64

5.3.3.1 前台视图解析处理.....	64
5.3.3.2 后台逻辑处理.....	67
5.3.4 采购管理模块处理设计.....	67
5.3.4.1 Controller 层处理.....	67
5.3.4.2 Service 层处理.....	69
5.3.5 出库管理模块处理设计.....	69
5.3.6 入库管理模块处理设计.....	70
5.4 数据库配置.....	71
5.4.1 数据库名称、软件环境.....	71
5.4.2 建立 ODBC 或 JDBC 等.....	72
5.4.3 数据库连接.....	72
5.4.4 数据库恢复.....	72
第六章 实施概况.....	72
6.1 实施环境与工具.....	72
6.1.1 计算机系统平台.....	72
6.1.2 编程环境与工具.....	72
6.1.3 数据准备.....	73
6.2 系统测试.....	77
6.2.1 测试规程.....	77
6.2.2 测试计划及测试记录.....	77
6.2.2.1 测试计划.....	77
6.2.2.2 测试记录.....	77
6.3 系统转换方案.....	79
6.3.1 老系统迁移新系统方案.....	79
6.3.2 新系统回退至老系统方案.....	80
6.3.3 废弃老系统.....	80
6.4 系统运行与维护概况.....	80
第七章 总结与展望.....	80
7.1 总结.....	80
7.2 展望.....	81
7.3 回顾.....	81
参考文献.....	81
致谢.....	82
附录.....	82

第一章 序言

1.1 组织机构概况

小水果是“小水果(上海)电子商务有限公司”(以下简称“小水果”)打造的水果 C2C, O2O 电商平台。小水果下辖总经办、水果事业部、技术平台中心、无线中心、财务部、网站运营中心、人力资源部、IT 部、品牌公关部、法务部、采购部、风险控制部、客服中心、营销中心等职能部门。

1.2 项目开发背景及信息系统目标

随着公司业务的发展及加盟店家的增多, 公司原有的系统已不能满足日益增长的业务需求。互联网、移动互联网已在悄悄的改变人们的生活方式, 为顺应时代的变化, 小水果担负起了提高人们生活便捷度的使命, 公司在多年的发展中得到了广大用户的信赖, 为了用户得到更好的体验, 公司决定升级原有的“网上交易”平台, 提供更人性化的服务。新一代的小水果信息管理系统(小水果网)应运而生。

1.3 信息系统范围

本系统主要涉及了客户关系管理、商品管理、订单管理、支付管理、物流管理及点评管理。

涵盖了消费者的登录注册及网上浏览搜索商品, 挑选商品, 网上下单及线上支付(对未完成的订单, 支付的全暂由小水果托管, 订单完成支付给水果店)。对已购买的商品在线上查看物流的实时状况, 在线下收到货后可在网上确认收货。

针对水果店涵盖了申请入驻、为小果店注册线上店铺及账号、小果店网上上下架商品、对已支付的订单且购买的是自家商品的订单进行发货, 对水果店的库存及采购供应链的管理等。

1.4 项目开发方法和计划

项目采用 Scrum 敏捷开发模式。每个子系统进行独立的小 team 开发, 每个小 team 按迭代周期进行每个迭代任务的启动、开发、验收、总结等。

每个迭代结束、新开发的成果交付测试, 产品经理验收。最后上线试运营。当前团队共 11 人, 分成 3 个小 team, 在两个半月的时间并行开发 CRM 管理子系统, 商品管理子系统、订单管理子系统。待此三个子系统开发完毕后, 依次开发后续子系统。

1.5 作者的主要贡献

我在本团队中, 主要参与小水果信息管理的整体架构设计, 用户信息管理子系统、商品信息管理子系统、订单信息管理子系统的设计及研发工作。

第二章 系统需求分析

2.1 现行业务系统描述

2.1.1 组织结构图介绍

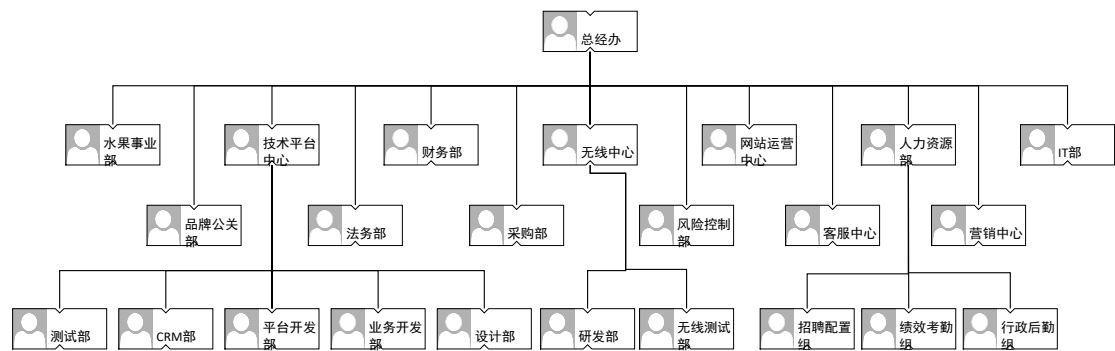


图 2-1 组织结构图

主要分为水果事业部、技术平台中心、无线中心、财务部、网站运营中心、人力资源部、IT 部、品牌公关部、法务部、采购部、风险控制部、客服中心、营销中心等二级职能部门。

仓管员查看到发货清单后，根据发货清单出库，出库后修改库存，将订单状态改为已出库。若商品的库存量小于最小库存量，则需要生成采购单，由采购员进行商品的采购。配送员根据快递单，取到出为别的商品后，核验货物清单，然后打包进行配送，同时将订单的状态改为配送中。每经过一个物流点，需要将物流信息更新到物流表中，可供消费者看到自己的货物已到地点，准备收货。

采购员根据仓管员出的采购清单，将联系供应商，然后将采购单提交给供应商。供应商根据采购单进行供货，入库。至此整个流程结束。

2.2 现行系统主要问题分析

小水果成立初期，只有自营业务，自营的水果店面也只有十来家，随着互联网移动互联网的兴起，水果电商也得到的快速的发展。在家就能买到自己喜爱的水果，日渐得到了消费者的青睐。在这个好时机中，公司不断在壮大业务，也有更多的水果店家想申请加入到小水果这个在线交易平台。现行的系统，因公司初创期技术、经济、互联网经验等的局限性，原系统在今天已遇到各种瓶颈问题。

现行系统不支持第三方商家的加盟业务，同时也不支持第三方的库存管理。造成了业务发展的瓶颈。

原业务面比较窄，随着业务量及成效量的倍增，现行的非分布式系统压力较大，经常出现宕机，系统维护成本增加。

2.3 解决方案

2.3.1 方案 1

采购一套进销存管理系统，用于管理小水果公司的商品，仓库及进出货。将小水果网上交易平台外包给第三方技术公司去做。后期上线后技术支持仍然由第三方公司维护。小水果公司只做业务运营。仓储由购买的进销存信息管理系统进行管理。

2.3.2 方案 2

小水果公司外包给第三方技术公司开发，由小水果公司的相关业务提需求，第三方技术公司根据小水果公司的业务需求开发出对应的系统，待上线试运营验收通过后，交由小水果公司管理。

2.3.3 方案 3

小水果公司自主研发。包括小水果系统的线上交易平台，后台管理平台及仓储管理平台的研发及维护。后期因业营运方向或务需求的调整，公司技术团队对系统做对应的改造，以实现新的功能需求。

后期用户增长达到一定量后，公司技术团队专门对用户行为数据进行分析，创建用户数据中心、用户画像等基础数据平台，用于后期做数字化营销及用户精准推荐等。

2.3.4 结论

根据公司的现实业务增长情况，资金支持情况，运营团队建设情况综合分析。

方案 1，公司前期比较省事，为公司省下了研发及项目管理成本。但后期公司变更业务方向或是运营策略，还得依赖第三方技术公司。总成本并未降低，且系统对公司未来发展的

支持前景堪忧。

方案 2，同方案 1，前期省事，为公司省下了研发及项目管理成本。对第三方技术公司过于依赖。后期运营及研发成本过高。公司现有的技术团队人才会造成浪费。

方案 3，小水果公司现有自己的技术平台中心，技术团队中本科及以上学历占据 80%，他们中有从事过大型互联网电商行业的研发工作。对 java、javascript、数据库等技术有资深的带头人。对于小水果信息管理系统的研发能提供技术保障。

此方案前期会投入成本较高，同时会增加项目管理成本，但目前公司已有技术研发团队，不用也是一种人才浪费，后期系统维护成本及业务变更等能很好的支持，因公司有熟悉本系统的技术团队，对后期维护可省下大量成本，综述不论从总成本还是后期系统的发展前景上看，方案三都是最优的。

故优选方案 3。

2.4 可行性分析

2.4.1 技术可行性分析

本人所在的技术团队目前有 11 人。其中架构师 1 人，Team Leader 1 人，资深开发工程师 3 人，中级 java 开发工程师 4 人，实习生 2 人。他们中有在大型互联网公司担任过核心模块的架构及研发工作，有做过系统安全方面的宝贵经验，其中有 2 人在互联网电商行业做过用户画像、精准营销的相关技术工作。百分之八十人员熟悉各种主流的 java 框架(Spring/MyBatis/SpringMVC)，熟悉各主流关系型数据库(Oracle, MySQL,DB2)、NoSQL 数据库(Hbase, MongoDB)等。

小水果网计划使 Java 主流框架(Spring/MyBatis/SpringMVC)+MySQL 数据库来开发。后期用户精准营销计划用大数据计算框架 Hadoop、Spark 做离线计算和实时计算操作。用 Hbase 和 Redis 做用户画像数据存储。

从本团队的技术经验及所擅长的技术综合来看，都与小水果网的技术方案匹配。在这些团队人员的互相协作下，小水果网上线的一天指日可待。

2.4.2 经济可行性分析

现因老系统无法满足日益增长的业务，需要重新开发能够适应业务增长的新系统。在一年前公司领导层决定了开发新的系统来支持，取名为“小水果网信息管理系统”，后简称为“小水果网”。

公司经过多年的发展，积累了大量用户，虽然目前投入较大，尚未盈利，但公司的前景客观，得到了大多投资机构的青睐。目前正在进行的 C 轮融资已得到了包括红杉资本在内的等知名投资机构的投资，在技术研发经费上公司投入有足够的保障。公司前期已投入 2 千万的研发经费。小水果网的建设，从经济方面得到了保障。

2.4.3 营运可行性分析

目前项目已提上日程，前期开发投入已到位，项目管理工具用禅道，采用 scrum 敏捷开发模式。禅道项目管理工具基本与 scrum 开发模式匹配，对开发的每个过程都能很好的控制，严格按照时间节点进行，确保了项目进度。项目代码采用 git 管理，这种版本控制软件目前应用比较广阔，特别适合大型项目多人协作开发。营运整个项目的正常进行是可行的。

第三章 新系统逻辑方案

3.1 用例(用况)建模

3.1.1 小水果网信息管理系统用例示例

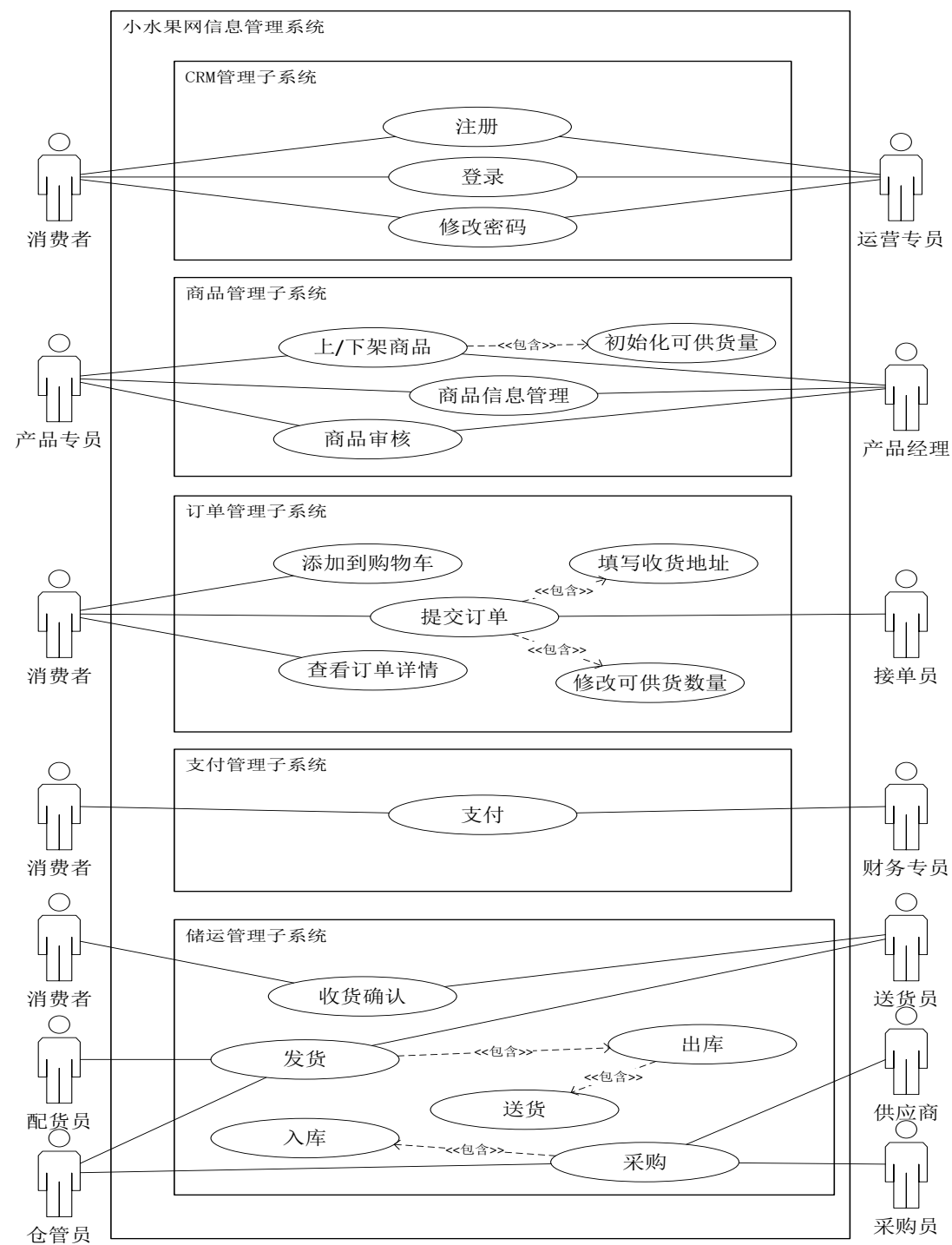


图 3-1 小水果用例示例

3.1.2 CRM 信息管理子系统

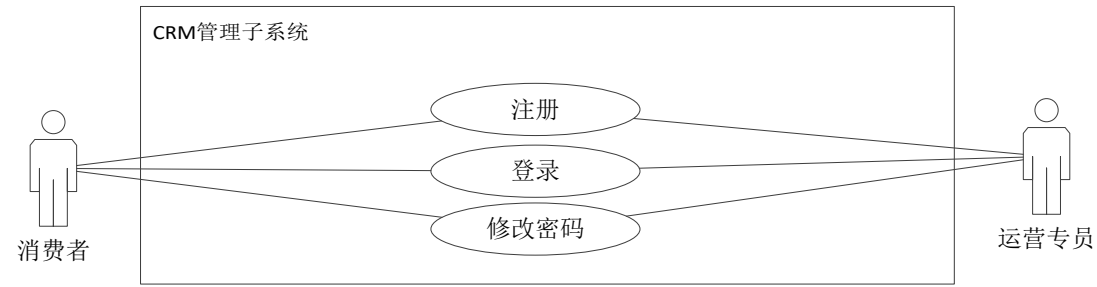


图 3-2 CRM 信息管理子系统用例

3.1.3 商品信息管理子系统

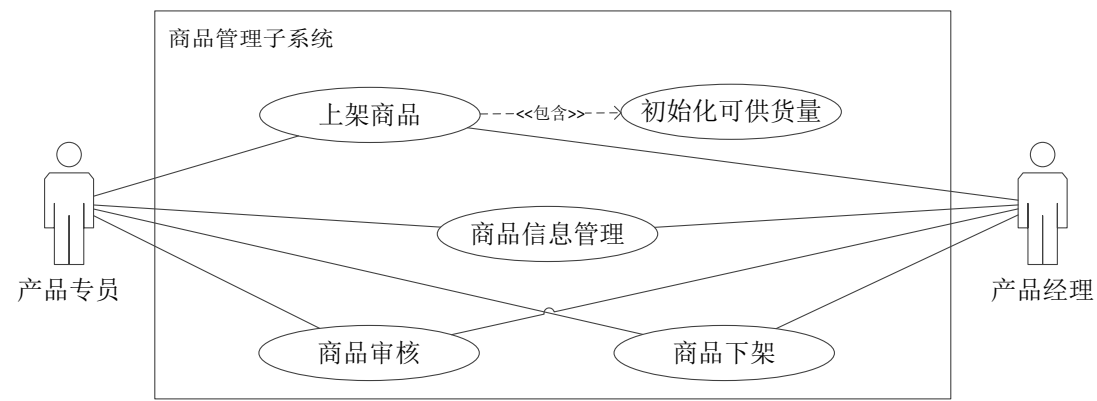


图 3-3 商品信息管理子系统用例

3.1.4 订单信息管理子系统

3.1.4.1 订单信息管理子系统用例图

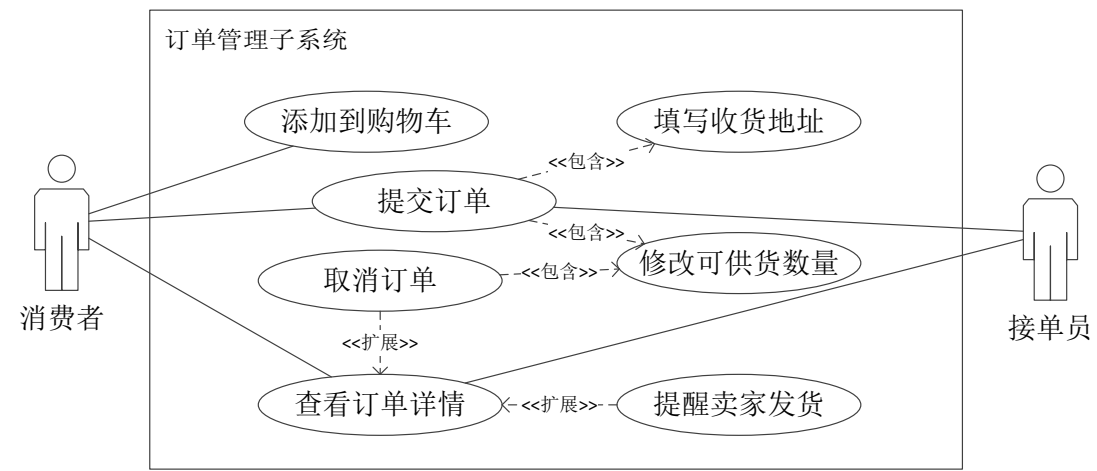


图 3-4 订单信息管理子系统用例

3.1.4.2 订单信息管理子系统用例文档

用例名称	添加到购物车
参与者	消费者
前置条件	一个消费者进入到商品列表页面

基本路径	1、消费者看到自己的意向商品，点击进入到商品详情页面。 2、修改商品数量(意向购买的数量)。 3、点击添加到购物车。 4、在弹出的登录页面输入用户名密码并提交登录按钮，系统进行登录。 a)、登录成功，转至步骤 5； b)、登录失败，提示登录失败。 5、查询该用户的购物车表中是否已有该商品。 6、修改该购物车中此商品的数量 7、向此购物车列表中新插入一条商品及数量 8、系统显示添加购物车结果 9、点击购物车列表 可选路径 5、9 为可选路径，6、7 为二选一路径，若 5 有该商品，则至路径 6，否则至路径 7。步骤 7，9 可结束用例。
后置条件	购物车表中新增一条或多条商品记录。

表 3-1 添加到购物车用例

用例名称	提交订单
参与者	消费者、接单员
前置条件	一个合法登录的消费者进入到购物车列表页
基本路径	1、消费者在购物车列表页勾选当次准备购买的商品 2、修改购买数量。 3、点击去结算按钮。 4、若无收货地址，可点击新增收货地址。 5、填写收货地址并保存。 6、系统展示收货地址在页面，供消费者选择其中一个作为本次购买的收货地址。 7、选择一条收货地址作为当前订单的收货地址。 8、点击提交订单按钮。 9、向订单表中添加一条记录。 10、向订单明细表中添加多条记录。 11、系统显示订单表和订单详情表插入结果。 a)、若成功，显示订单号； b)、失败、给出失败提示信息。 可选路径： 2、4、5、8 为可选路径。11 可结束用例。
后置条件	订单表新增一条记录、订单详情表新增一条或多条记录。

表 3-2 提交订单用例文档

3.1.5 支付信息管理子系统

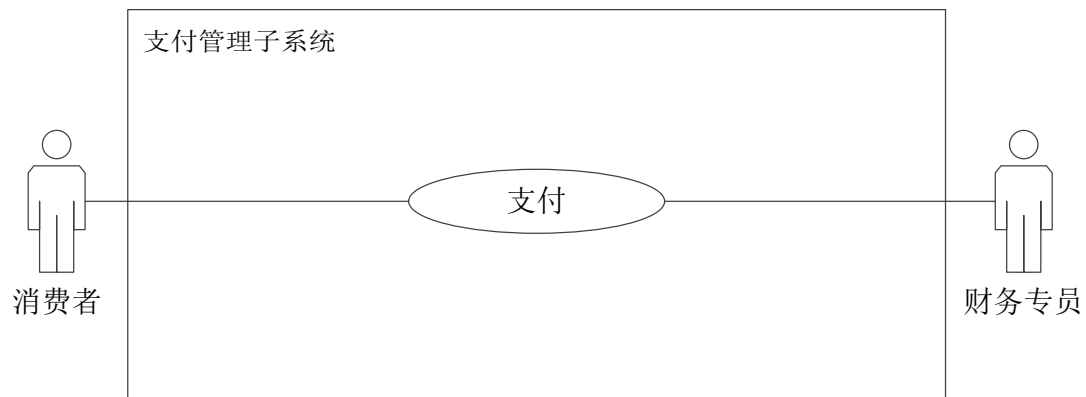


图 3-5 支付管理子系统

3.1.6 储运信息管理子系统

3.1.6.1 储运信息管理子系统用例图

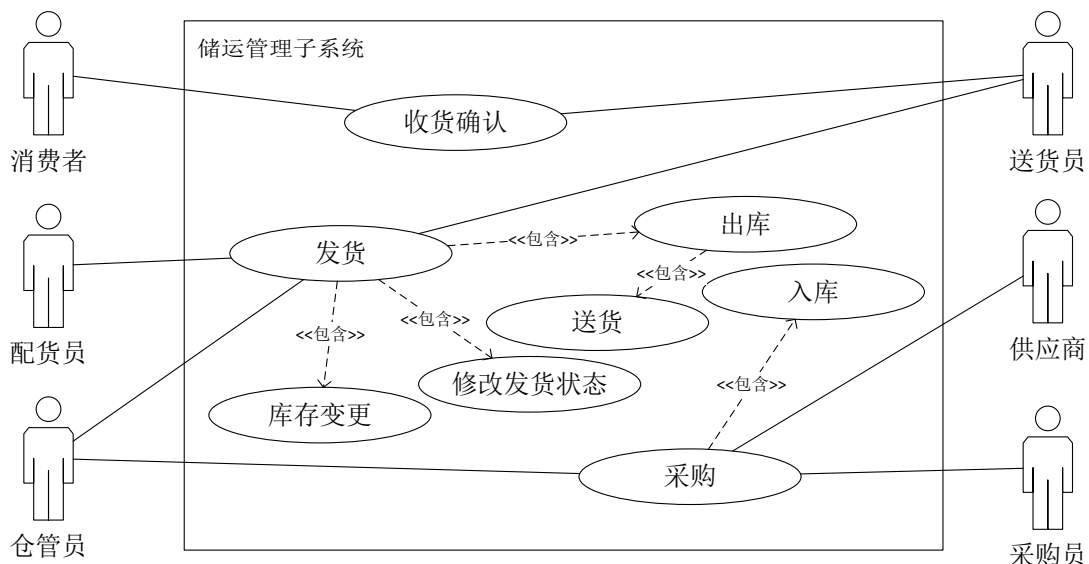


图 3-6 储运信息管理子系统

储运管理子系统主要有消费者、送货员、配货员、仓管员，供应商、采购员六种角色参与。

发货用例说明：

仓管员登录小水果后台管理系统后，点击发货管理菜单，进入到发货管理页面。页面列出未发货的订单，发货单下展示有对应发货单商品明细情况，仓管员可选择点击每条商品进行出库，出库会减少库存。出库后会修改对应订单的状态为已出库。至此发货完成。

采购用例说明：

采购员登录小水果后台管理系统后，可进行采购单的制作，填写采购名称，选择供应商后，可循环选择对应采购的商品及数量，一条条添加采购明细，最后提交采购单，同时生成采购明细记录。

3.1.6.2 储运信息管理子系统用例文档

用例名称	发货(出库, 库存变更, 修改发货状态)
参与者	仓管员、配货员、送货员
前置条件	仓管员进入到小水果后台发货管理界面
基本路径	1、系统显示待出库的订单列表, 列表下包含待出库的商品及数量 2、选择对应出库的商品 3、点击出库按钮 4、系统查询该条商品对应的库存 5、现有库存量减去出库的商品数量后为新的库存数量(出库后数量大于 0, 下单时有判断库存, 没有库存不可下单成功) 6、根据明细 ID 将该条明细修改为已出库 7、转至步骤 2(该订单下仍然有未出库商品时) 8、修改订单状态为出库完成 可选路径 7 为可选路径。8 可结束用例。
后置条件	订单状态被修改为出库完成。该条商品库存量被修改。

表 3-3 发货用例文档

用例名称	采购
参与者	供应商、采购员
前置条件	采购员进入到采购管理界面
基本路径	1、录入采购单名称 2、选择供应商 3、选择采购商品, 录入数量 4、点击添加明细按钮 5、转至路径 3 6、点击提交采购单按钮 7、向采购单表插入一条记录, 并返回采购单号 8、将采购单号添加第一条明细中去 9、向采购明细表中插入该条明细 10、内存中移出经一条采购明细 11、转到路径 8 12、返回采购单创建结果 路径 5、11 为可选择路径, 可循环。路径 12 可结束用例。
	对应订单状态被修改为已收获, 对应商品可能新增一条点评记录。

表 3-4 收货用例文档

用例名称	入库
参与者	供应商、仓管员
前置条件	仓管员进入到小水果后台管理系统的入库管理界面
基本路径	1、仓库管理员进入系统入库主界面 2、仓库管理员输入供应商提供供货单号, 根据供货单号查询采购单号

	<p>出的供货单</p> <p>3、如果没有查询到采购单及对应的供货单，提示输入有误，请核对供货单号。</p> <p>4、查询到采购单及对应的供货单信息，显示供货单信息及供货单明细。</p> <p>5、选择一条明细，点击入库。</p> <p>6、修改该条明细的状态为已入库。</p> <p>7、修改对应商品的库存记录。</p> <p>8、向库存表中插入一条记录。</p> <p>9、转至步骤 5。</p> <p>10、转至步骤 1。</p> <p>11、用例结束。</p> <p>可选路径</p> <p>路径 7、8、9 为可选路径。10、11 可结束用例。</p>
后置条件	供货单涉及的供货明细状态被修改、供货明细中商品的库存记录被修改。

表 3-5 入库用例文档

3.2 包图(系统包图)



图 3-7 系统功能包图

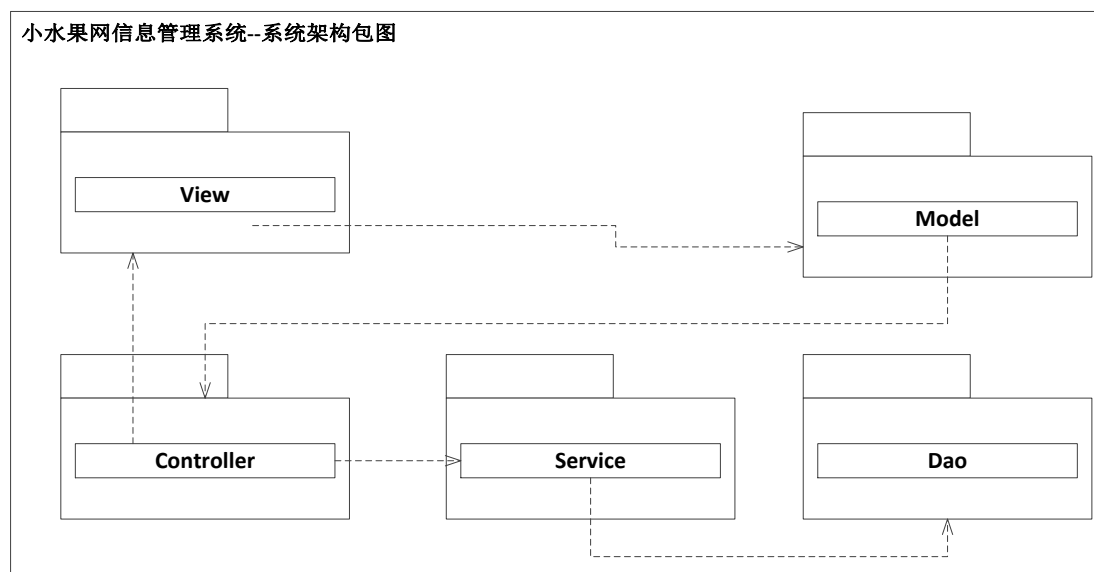


图 3-8 系统架构包图

本系统主要采用 Java 的 MVC 模式来开发。视图(View)层以 Spring MVC 来实现，模型(Model)层以 java bean 来充当，控制层(Controller)以 Spring MVC 来实现，View, Model, Controller 三层之间相互进行着数据的通信。业务逻辑(Service)层主要 Java 代码来实现，数据库持久(Dao)层主要是 Mybatis 框架来实现。Controller 调用 Service 接口，Service 接口调用 Dao 接口，返回数据则从 Dao 到 Service 再到 Controller 层，最终将数据返回至视图层(View)。

3.3 静态建模

3.3.1 类图

3.3.1.1 关联图

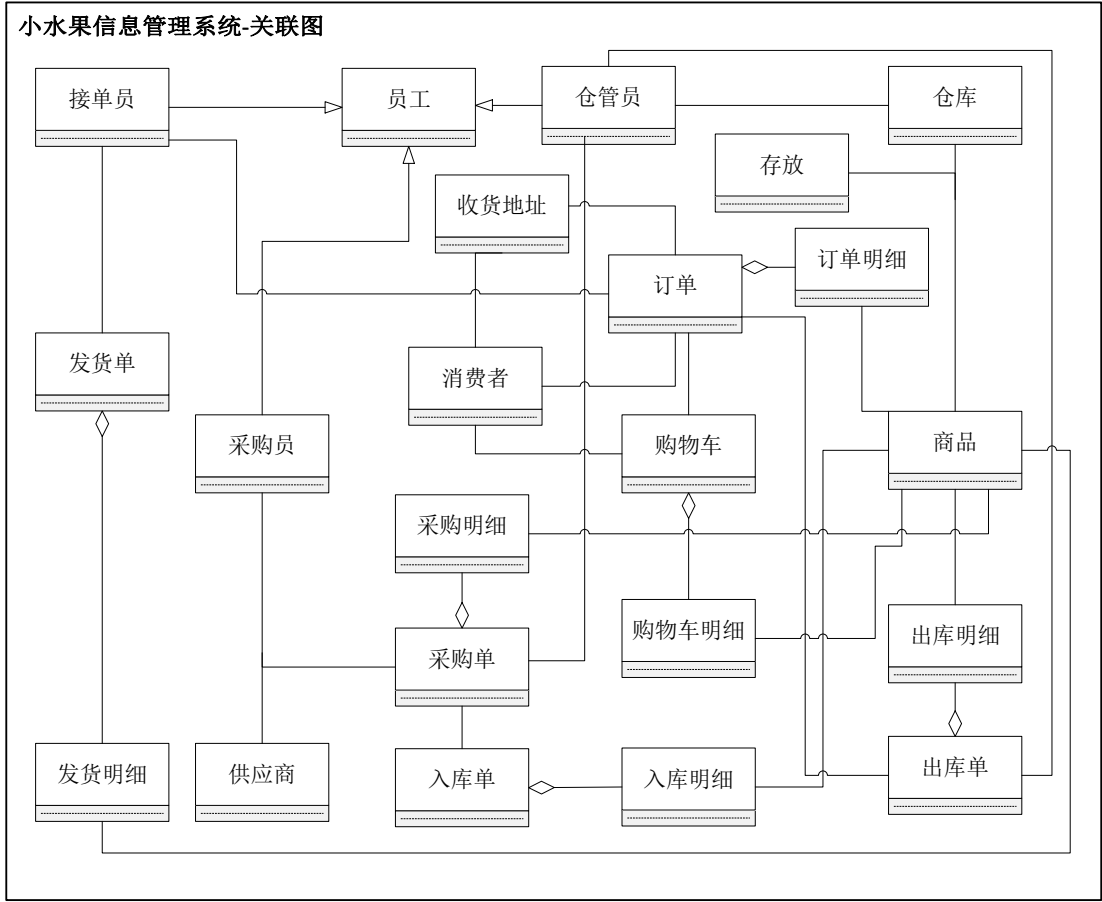


图 3-9 关联类图

接单员、仓管员、采购员泛化为员工。

下单涉及消费者、接单员和商品，产生订单，订单由多条订单明细组成，订单明细涉及商品。

接单员生成发货单，发货单位由多条发货明细组成，发货明细涉及商品。

出库涉及仓管员、仓库和商品。出库单由多条出库明细组成，出库明细射击商品。

采购单涉及采购员和供应商，采购单由多条采购明细组成，明细涉及商品。因采购入库，既一个采购单可以关联多个入库单，入库单由入库明细组成，入库明细涉及商品，商品涉及仓库。

3.3.1.2 类的定义

3.3.1.2.1 员工类

类名	员工
属性	1、员工编号 Long PK 2、员工姓名 String

	3、性别 char 4、生日 date 5、部门 String 6、职位 String 7、角色 String 8、员工状态 short (0：未入职，1：实习生，2：试用期，3：已转正，4：离职，5：被解雇) 9、用户名 String 10、密码 String 11、备注 String
方法	1、创建员工信息 Import:员工基本信息， export:结果标识 说明：将录入的员工信息插入到员工表。插入成功结果标识为 SUCCESS，否则为 FAILED。 2、获取员工信息 Import:客户编号， export:员工基本信息 说明：根据员工编号，从员工表中查询员工信息。查询到该员工，返回该员工基本信息，否则返回 null。 3、修改员工信息 Import:员工编号， export:更新结果标识 说明：将录入的员工信息根据员工编号更新到员工表中去。更新成功结果标识为 SUCCESS，否则为 FAILED。 5、删除员工 Import:员工编号， export:删除结果标识 说明：根据录入的员工编号，删除该员工。删除成功结果标识为 SUCCESS，否则为 FAILED。 6、设定员工状态 Import:员工编号， export:结果标识 说明：根据员工编号去更新员工的状态。更新成功结果标识为 SUCCESS，否则为 FAILED。 7、员工登录后台 Import:用户名、密码， export:登录结果标识 说明：根据输入的用户名查询该用户，如果查询到该用户，用查询出的密码与用户输入的密码进行 MD5 加密后的密码进行比较，如果相匹配，则登录成功。登录成功结果标识为 SUCCESS，否则为 FAILED。

表 3-6

3.3.1.2.2 消费者类

类名	消费者
属性	1、用户编号 Long PK 2、用户名 String 3、密码 String 4、性别 char

	5、生日 date 6、手机号 String 7、邮箱 String 8、vip 等级 int 9、注册时间 date 10、用户状态 byte 11、备注 String
方法	1、注册 Import:用户基本信息, export:结果标识 说明:将录入的用户信息插入到用户(customer)表。插入成功结果标识为 SUCCESS, 否则为 FAILED。 2、获取用户信息 Import:用户编号, export:用户基本信息 说明:根据用户编号, 从用户表中查询用户信息。查询到该用户, 返回该用户基本信息, 否则返回 null。 3、修改用户信息 Import:用户编号, export:更新结果标识 说明:将录入的用户信息根据用户编号更新到用户表中去。更新成功结果标识为 SUCCESS, 否则为 FAILED。 4、冻结 Import:用户编号, export:冻结结果标识 说明:根据录入的员工编号, 将员工状态改为已冻结。冻结成功结果标识为 SUCCESS, 否则为 FAILED。 5、登录 Import:员工编号, export:结果标识 说明:根据输入的用户名查询该用户(customer)表, 如果查询到该用户, 用查询出的密码与用户输入的密码进行 MD5 加密后的密码进行比较, 如果相匹配, 则登录成功。登录成功结果标识为 SUCCESS, 否则为 FAILED。

表 3-7

3.3.1.2.3 商品类

类名	商品
属性	1、商品编号 Long PK 2、商品名称 String 3、商品类别 short 0:进口类, 1:国产类 4、市场价 Long 单位分 5、促销价 Long 单位分 6、可供货数量 int 7、状态 byte 0:未上架(商品已录入到系统中) 1:正常在售, 2:已下架 8、备注 String
方法	1、新增商品 Import:商品基本信息, export:结果标识

	<p>说明：将录入的商品信息插入到商品(goods)表。插入成功结果标识为 SUCCESS，否则为 FAILED。</p> <p>2、获取商品信息 Import:商品编号， export:商品基本信息 说明：根据商品编号，从商品表中查询商品信息。返回该商品的基本信息。</p> <p>3、修改商品信息 Import:商品编号及其商品的基本信息， export:更新结果标识 说明：将录入的商品信息根据商品编号更新到商品表中去。更新成功结果标识为 SUCCESS，否则为 FAILED。</p> <p>4、上架 Import:商品编号， export:上架结果标识 说明：根据录入的商品编号，将商品状态改为 1。修改成功结果标识为 SUCCESS，否则为 FAILED。</p> <p>5、下架 Import:商品编号， export:结果标识 说明：根据录入的商品编号，将商品状态改为 2。修改成功结果标识为 SUCCESS，否则为 FAILED。</p>

表 3-8

3.3.1.2.4 购物车类

类名	购物车
属性	<p>1、编号 Long PK</p> <p>2、员工编号 Long FK</p> <p>3、商品编号 Long FK</p> <p>4、意向购买数量 int</p> <p>5、添加时间 date</p> <p>6、备注 String</p>
方法	<p>1、加入购物车 Import:商品编号、员工编号、购买数量， export:结果标识 说明：将录入的员工编号、商品编号、意向购买数量等插入到购物车表中。插入成功结果标识为 SUCCESS，否则为 FAILED。</p> <p>2、获取购物车商品列表 Import:用户编号， export:购物车商中商品列表信息 说明：根据用户编号，从购物车表中查询加入购物车中的商品编号，然后根据查询到的商品编号从调用商品类的获取商品信息查询商品信息，将查询的结果封装成购物车列表信息对象返回。返回该用户的购物车中商品列表。</p> <p>3、修改数量 Import:商品编号、用户编号， export:更新结果标识 说明：根据输入的商品编号及用户编号及意向购买数量，修改对应的意向购买数量为输入的数量。修改成功结果标识为 SUCCESS，否则为 FAILED。</p> <p>4、删除购物车中商品 Import:商品编号、用户编号， export:上架结果标识</p>

	说明 :根据输入的用户编号及商品编号, 将对应的购物车表中的记录删除掉。修改成功结果标识为 SUCCESS, 否则为 FAILED。

表 3-9

3.3.1.2.5 订单类

类名	订单
属性	1、订单编号 Long PK 2、员工编号 Long FK 4、订单金额 Long 单位分 5、购买时间时间 date 6、订单状态 short 0：待接单, 1：待发货, 2：待收获, 3：已完成 7、支付状态 short 0：待支付, 1：已支付 8、收货地址编号 Long FK 6、备注 String
方法	1、生成订单 Import:订单信息, export:结果标识 说明：将录入订单信息等插入到订单表中。插入成功结果标识为 SUCCESS, 否则为 FAILED。 2、修改收货地址 Import:订单编号, 收货地址编号, export:修改收货地址结果 说明：根据输入的订单编号作为条件, 将数据库表中对应的记录的收货地址改为输入的收货地址编号。返回修改结果, 修改成功返回标识为 SUCCESS, 否则返回 FAILED。 3、获取订单信息 Import:订单编号, export:订单信息 说明：根据输入的订单编号, 查询订单表中对应的订单的信息, 如果查询到订单, 将订单信息返回, 否则返回 null。 4、修改支付状态(支付后, 系统自动调用) Import:订单编号、支付状态, export:修改订单状态结果信息 说明 :根据输入的订单编号, 然后将对应的订单记录的支付状态改为已支付, 此方法在用户支付完成订单后, 系统调用。 5、修改订单状态 Import:订单编号、订单状态, export:订单信息 说明：根据输入的订单编号, 将对应订单表中的此条记录的订单状态改为对应输入的状态。后台工作人员可将订单状态改为待发货和待收货和已完成, 而消费者在收货后, 点击确认收货, 会将此状态改为已完成。

表 3-10

3.3.1.2.6 订单详情类

类名	订单详情
属性	1、订单详情编号 Long PK 2、订单编号 Long FK

	4、商品编号 Long FK 5、购买数量 int 6、应付金额 Long 单位分 7、实付金额 Long 单位分 8、创建时间 date 9、物流状态 short：未发货，1：待收获，2：已收货 10、是否点评 char Y:是, N:否 11、备注 String
方法	1、生成订单详情 Import:订单详情信息, export:结果标识 说明：将录入订单详情信息等插入到订单详情表中。插入成功结果标识为 SUCCESS, 否则为 FAILED。 2、获取订单详情 Import:订单编号, export:订单详情信息 说明：根据输入的订单编号作为条件, 从订单详情表中查询出该订单的订单详情列表并返回。 3、修改实付金额（用户支付时系统调用） Import:订单详情编号、支付金额, export:修改结果 说明：根据输入的订单编号, 将订单详情表中对应的商品的实付金额修改为输入的金額。

表 3-11

3.3.1.2.7 收货地址类

类名	收货地址
属性	1、收货地址编号 Long PK 2、用户标号 String FK 4、收货地址名称 String 为收货地址取的名称 5、收货人 String 该地址的签收人姓名 6、详细地址 String 收货的详细地址 7、手机号 String 8、邮政编码 String 11、备注 String
方法	1、保存收货地址 Import:收货地址信息, export:结果标识 说明：将录入收货地址信息等插入到收货地址表中。插入成功结果标识为 SUCCESS, 否则为 FAILED。 2、删除收货地址 Import:收货地址编号, export:删除结果标识 说明：根据输入的收货地址编号编号作为条件, 将对应的收货地址表中的记录删除掉。删除成功, 结果标识返回 SUCCESS, 否则为 FAILED。 3、修改收货地址（用户支付时系统调用） Import:收货地址编号、收货地址信息 export:修改结果 说明：根据输入收货地址编号, 将对应的记录修改为输入的收货地址信息。修改成功则返回结果标识为 SUCCESS, 否则返回为 FAILED。

	<p>4、查询收货地址</p> <p>Import:用户编号 export:收货地址列表</p> <p>说明：根据输入用户编号，查询出该用户下的所有的收货地址列表返回。在下单的时候会调用，将该用户下的收货地址列出来，供用户选择当前订单邮寄到哪个收货地址下。</p>
--	--

表 3-12

3.3.1.2.8 采购单类

类名	采购单类
属性	<p>1、采购单编号 Long PK</p> <p>2、采购名称 String</p> <p>3、采购员编号 Long FK</p> <p>4、供应商编号 Long FK</p> <p>5、采购日期 date</p> <p>6、供应时间 date</p> <p>7、备注 String</p>
方法	<p>1、生成采购单</p> <p>Import:订单采购信息, export:结果标识</p> <p>说明：将录入采购单信息等插入到采购单表中。插入成功结果标识为 SUCCESS, 否则为 FAILED。</p> <p>2、获取采购单信息</p> <p>Import:采购单编号, export:采购单信息</p> <p>说明：根据输入的采购编号，查询采购表中对应的采购的信息，如果查询到采购单，将采购信息返回，否则返回 null。</p>

表 3-13

3.3.1.2.9 采购单明细类

类名	采购单明细类
属性	<p>1、采购明细编号 Long PK</p> <p>2、采购单编号 Long FK</p> <p>4、商品编号 long FK</p> <p>5、采购数量 smallint(5)</p> <p>6、采购单价 Long 单位分</p> <p>8、创建时间 date</p> <p>9、供货状态 short 0：未入库，1：已入库</p> <p>11、供货时间 date</p> <p>10、备注 String</p>
方法	<p>1、生成采购单明细</p> <p>Import:采购单明细信息, export:结果标识</p> <p>说明：将录入采购单明细等插入到采购明细表中。插入成功结果标识为 SUCCESS, 否则为 FAILED。</p> <p>2、根据主键获取采购明细</p>

	<p>Import:订单编号, export:采购明细编号</p> <p>说明:根据输入的采购明细编号作为条件,从采购明细表中查询出该采购明细记录并返回。</p> <p>3、根据采购单查询所有采购明细</p> <p>Import:采购单编号, export:该采购单对应的采购明细 list</p> <p>说明:根据输入的采购单编号,从采购明细表中查询该条采购单的所有采购明细,返回采购明细 list</p> <p>4、修改供货状态</p> <p>Import:采购单编号, 状态 export:受影响行数</p> <p>说明:根据采购单编号,将该采购单的状态修改为指定的状态。</p>
--	---

表 3-14

3.3.2 对象图

3.3.2.1 销售对象图

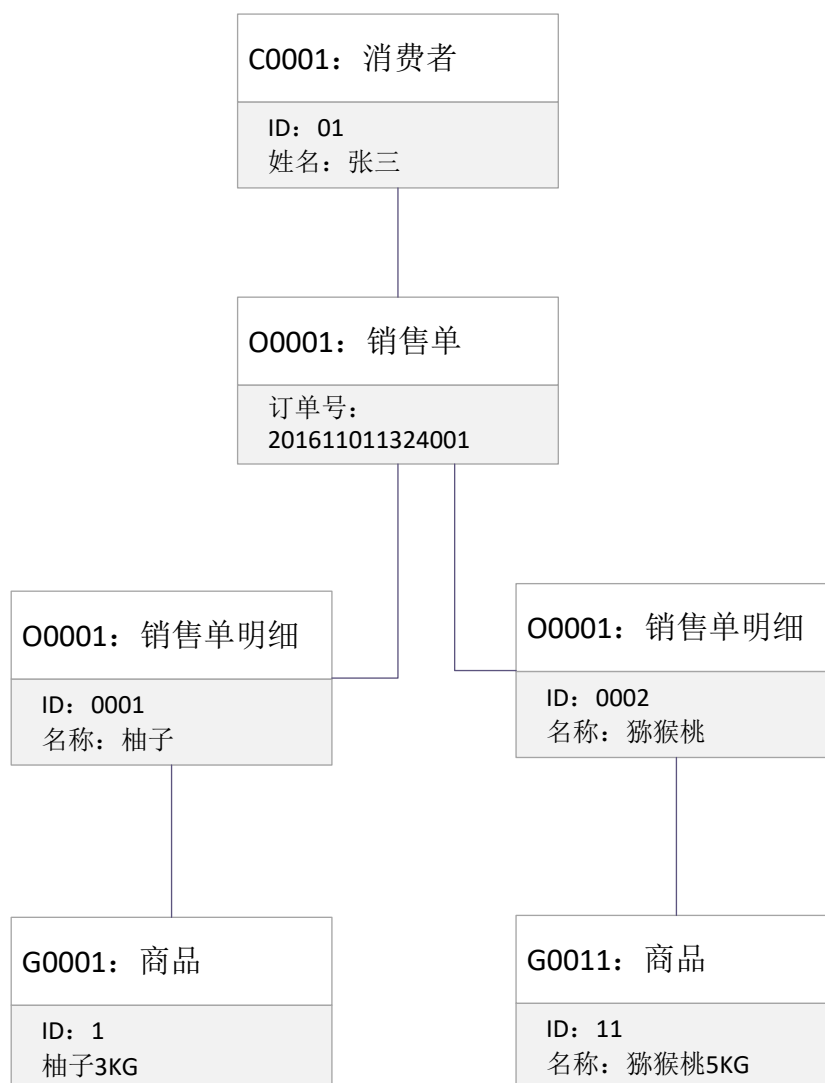


图 3-10

3.3.2.2 采购对象图

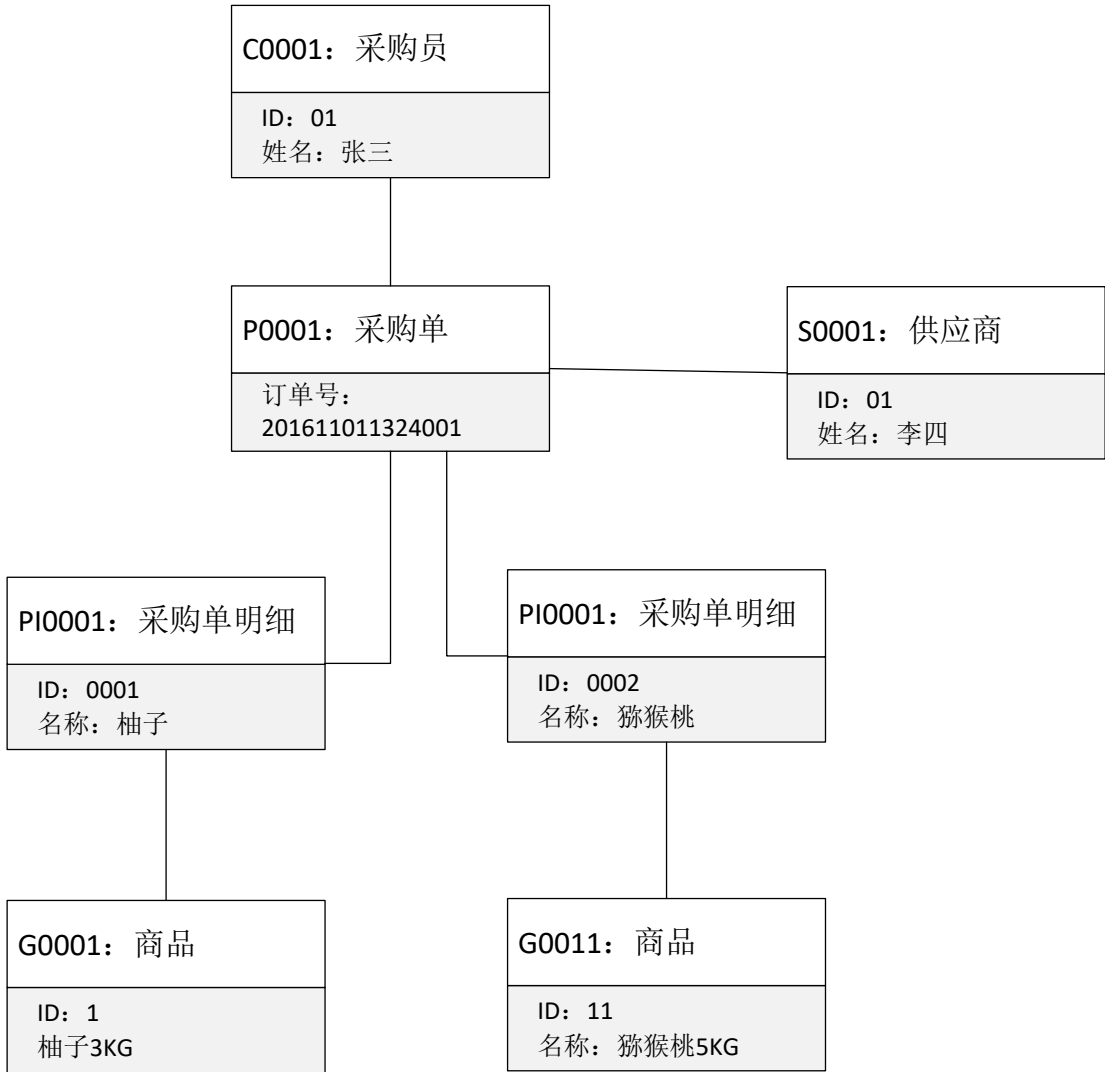


图 3-11

3.3.2.3 入库对象图

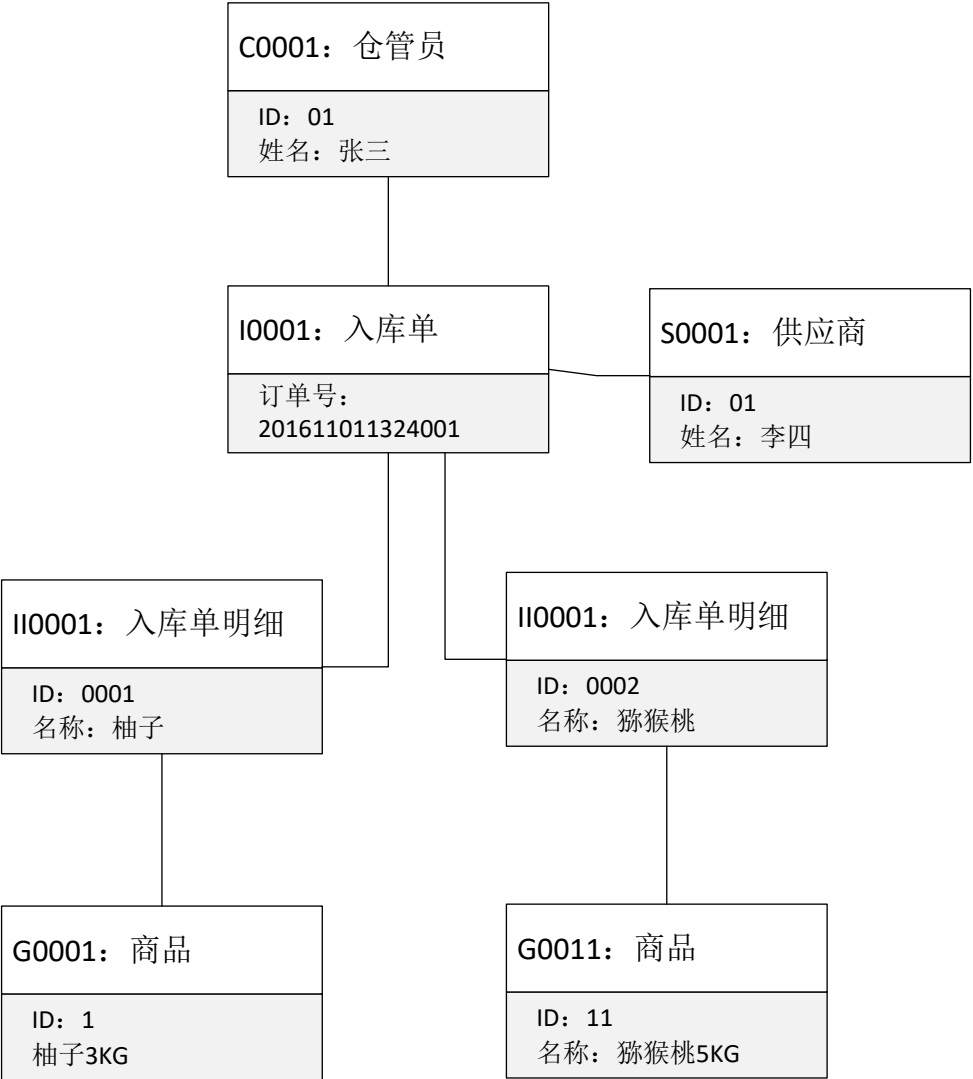


图 3-12

3.2.3.4 出库单

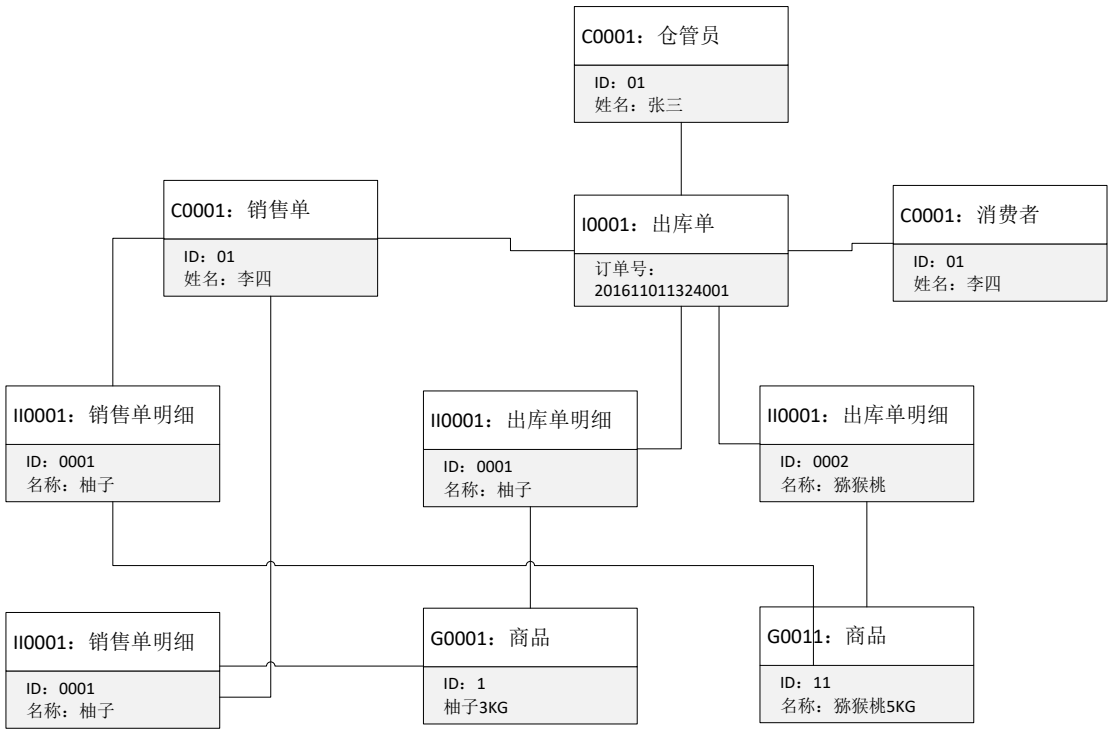


图 3-13

3.4 动态建模

3.4.1 状态图

3.4.1.1 商品状态图

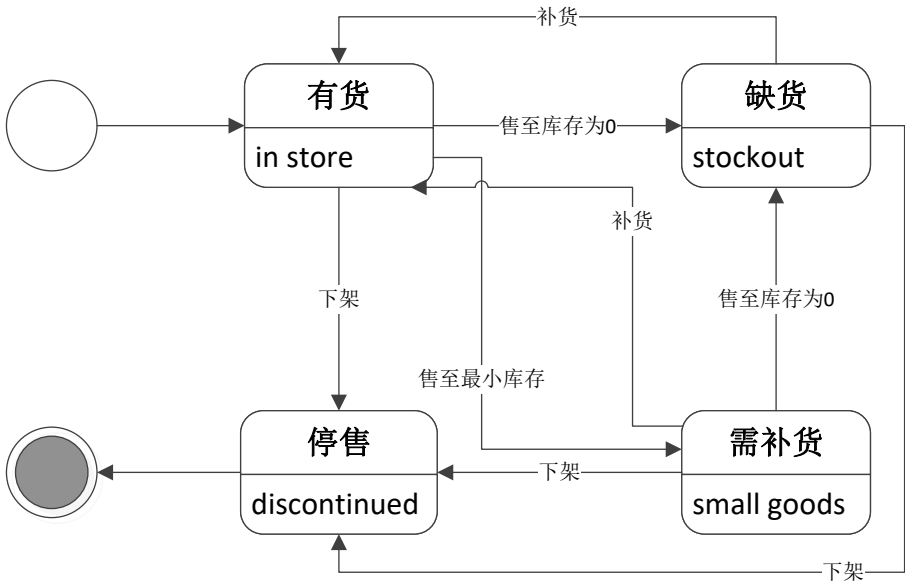


图 3-14 商品状态图

3.4.1.2 订单状态图

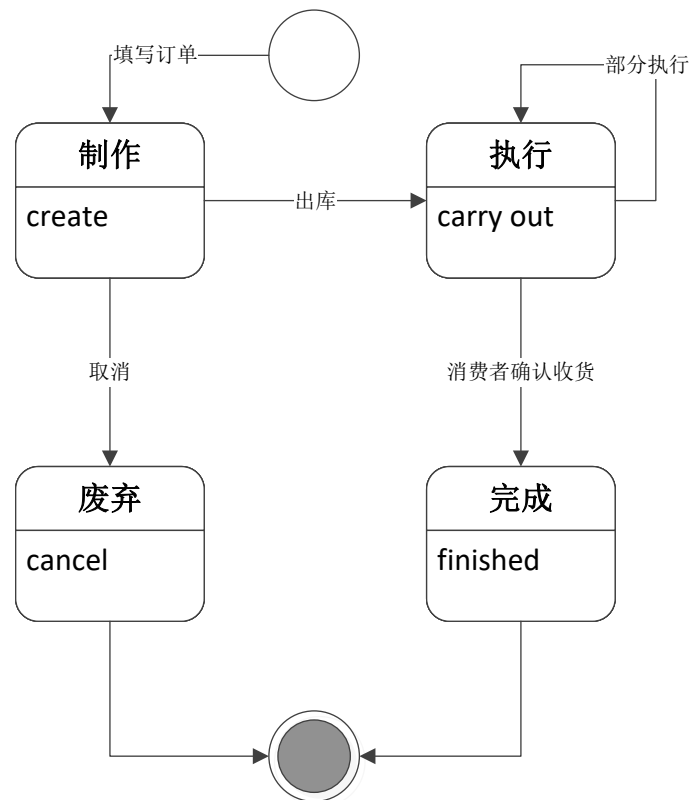


图 3-15 订单状态图

3.4.1.3 用户状态图

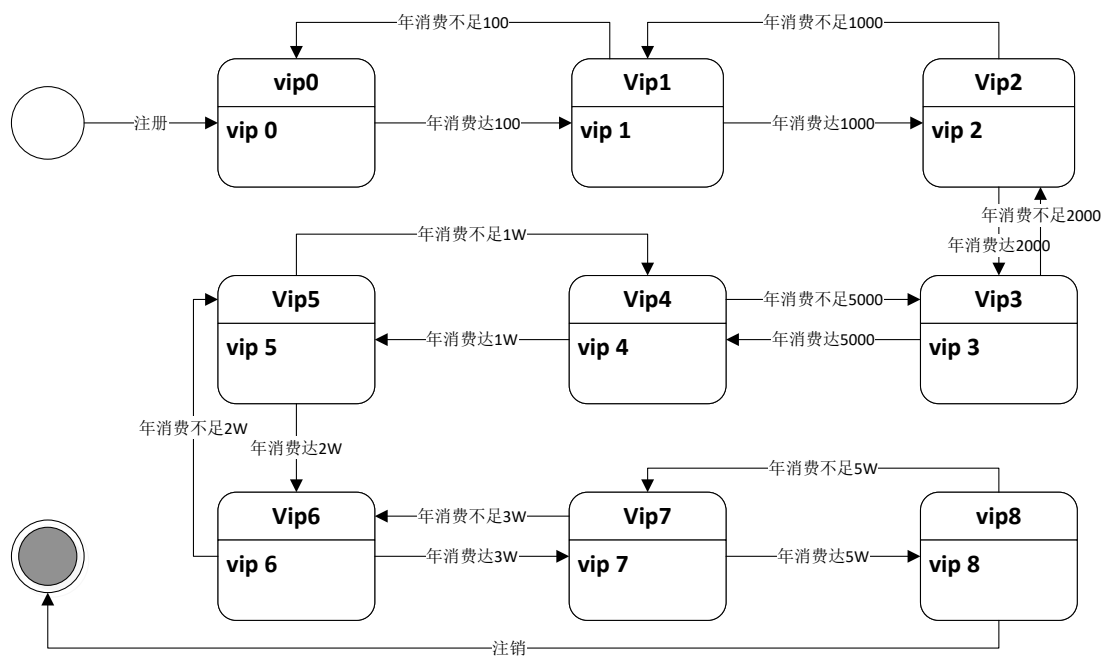


图 3-16 用户状态图

3.4.2 时序图

3.4.2.1 添加购物车时序图

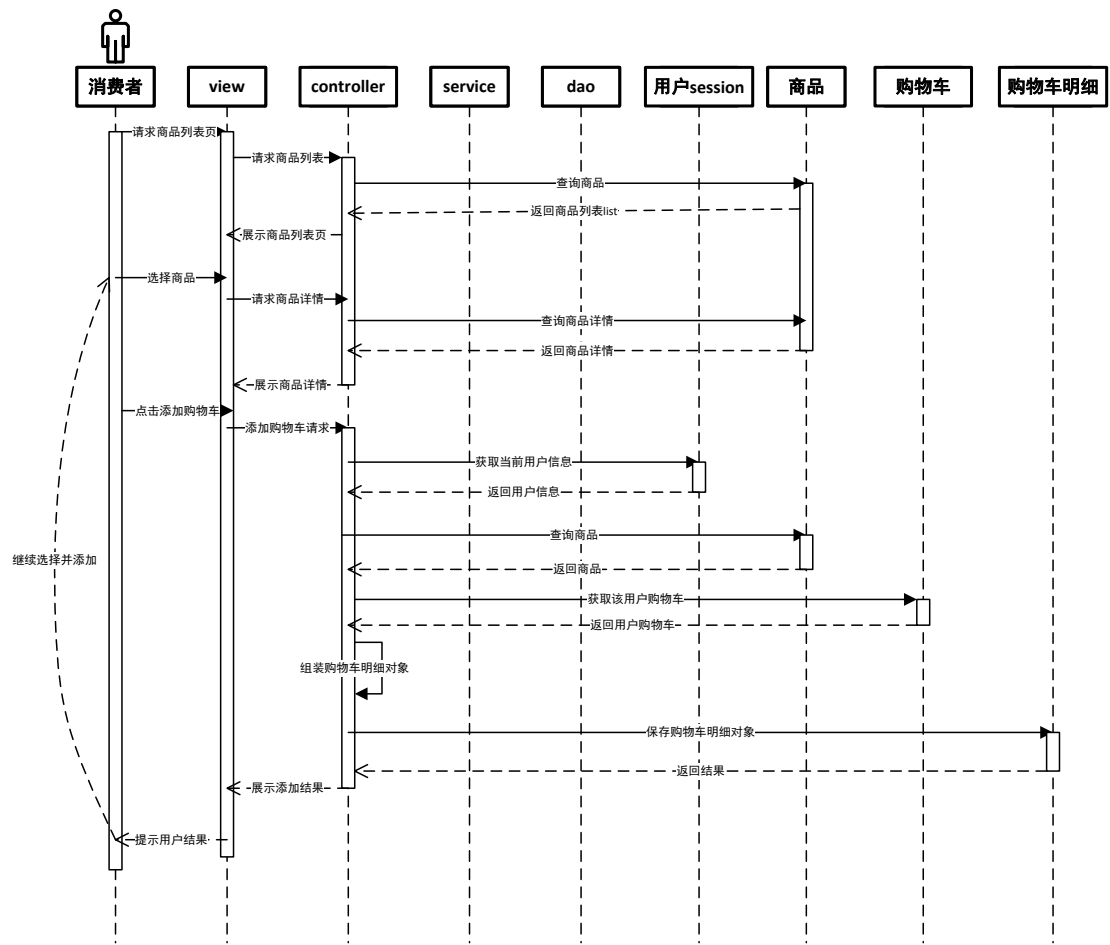


图 3-17 添加购物车时序图

3.4.2.2 用户下单时序图

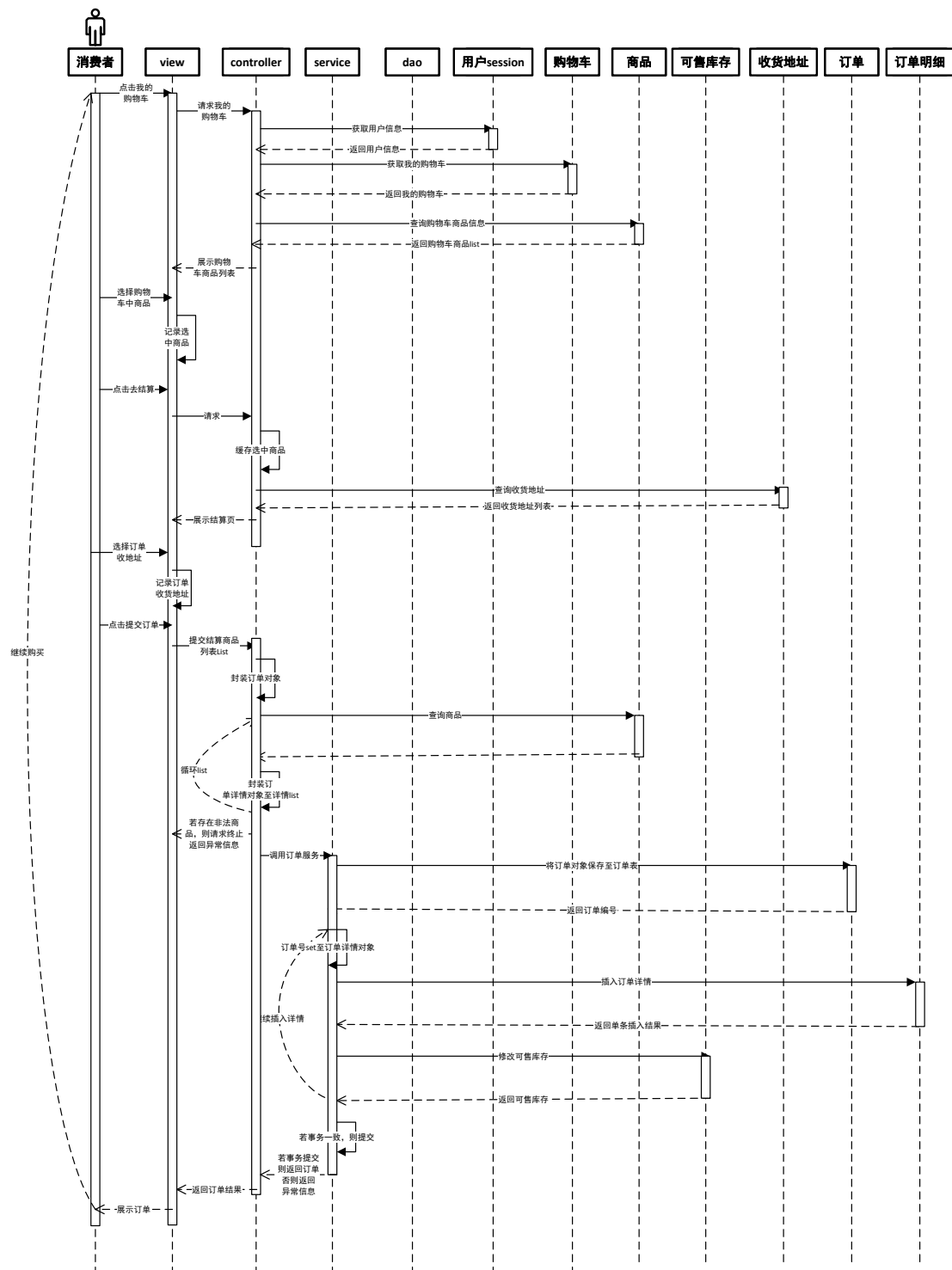


图 3-18 下单时序图

非法商品：指在小水果系统中无法查到的商品(因页面篡改而提交，故需后台校验商品)

事务一致：指一次请求，需要对数据库进行多次写操作，如订单和订单详情需要都写成功，则事务方可提交，否则需要回滚，表示订单失败。

封装对象：将页面提交的多条信息，在 controller 层封装成与数据库表对应的对象，方便持久化至数据库

3.4.2.3 出库时序图

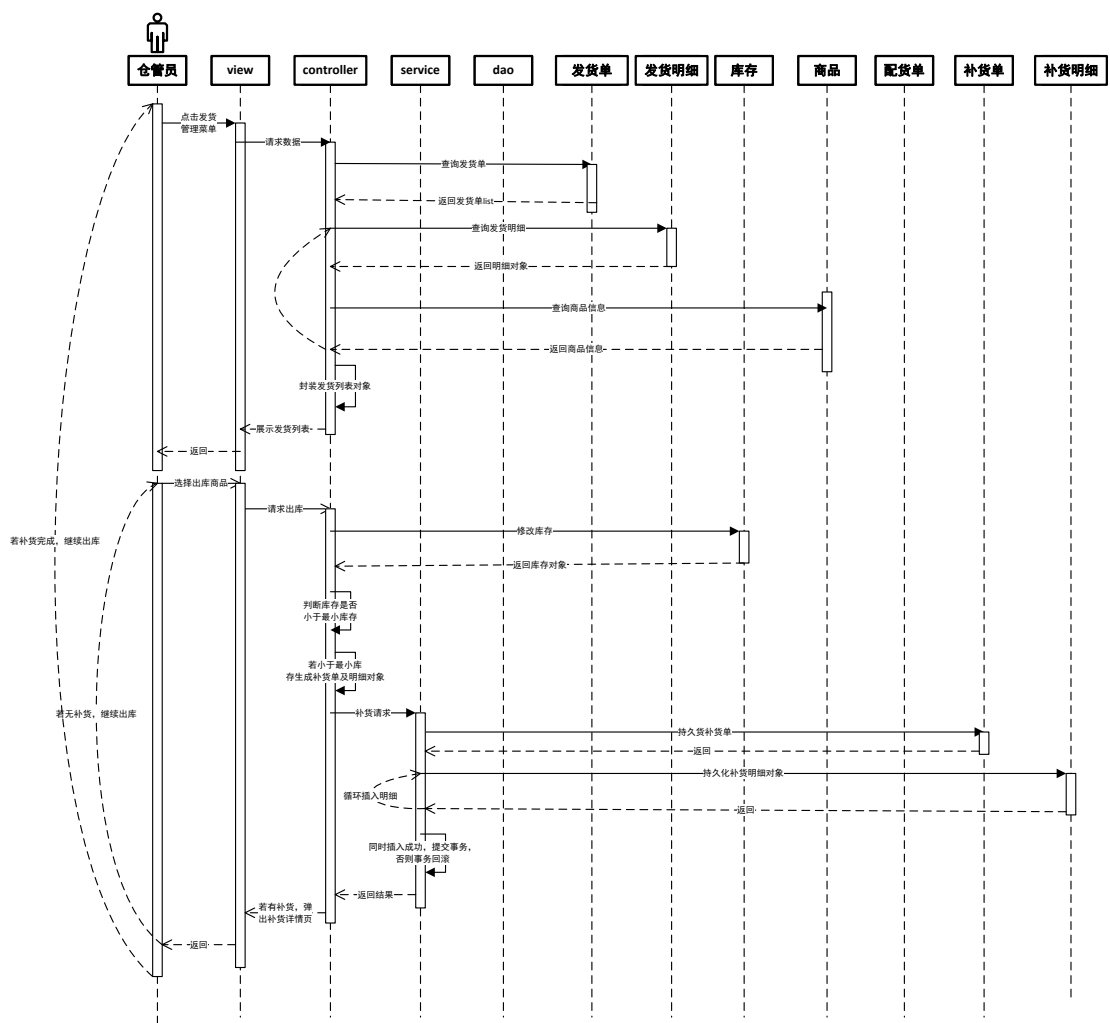


图 3-19 出库时序图

3.4.3 协作图

3.4.3.1 添加购物车协作图

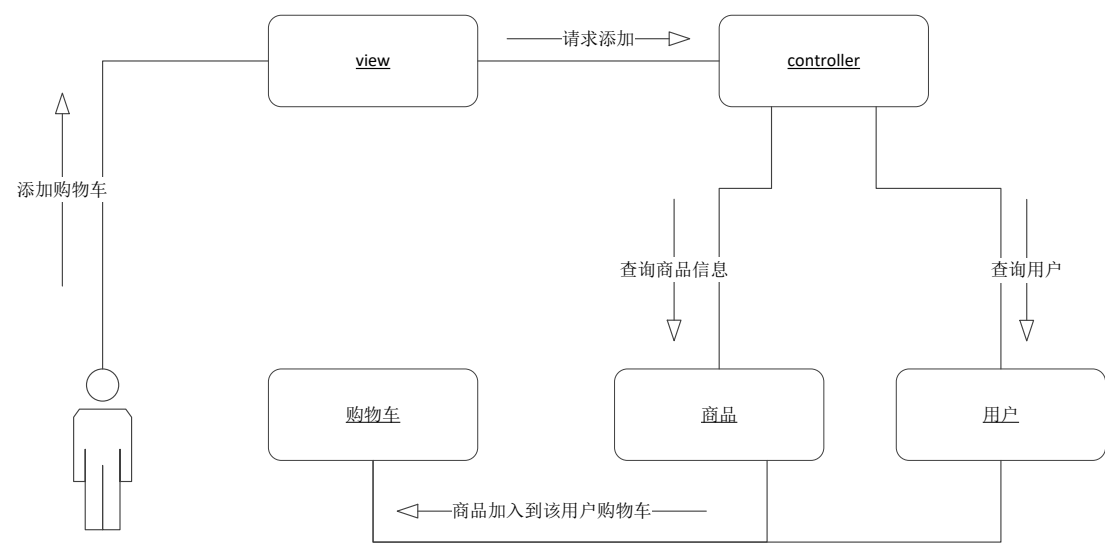


图 3-20 添加购物车协作图

3.4.3.2 下单协作图

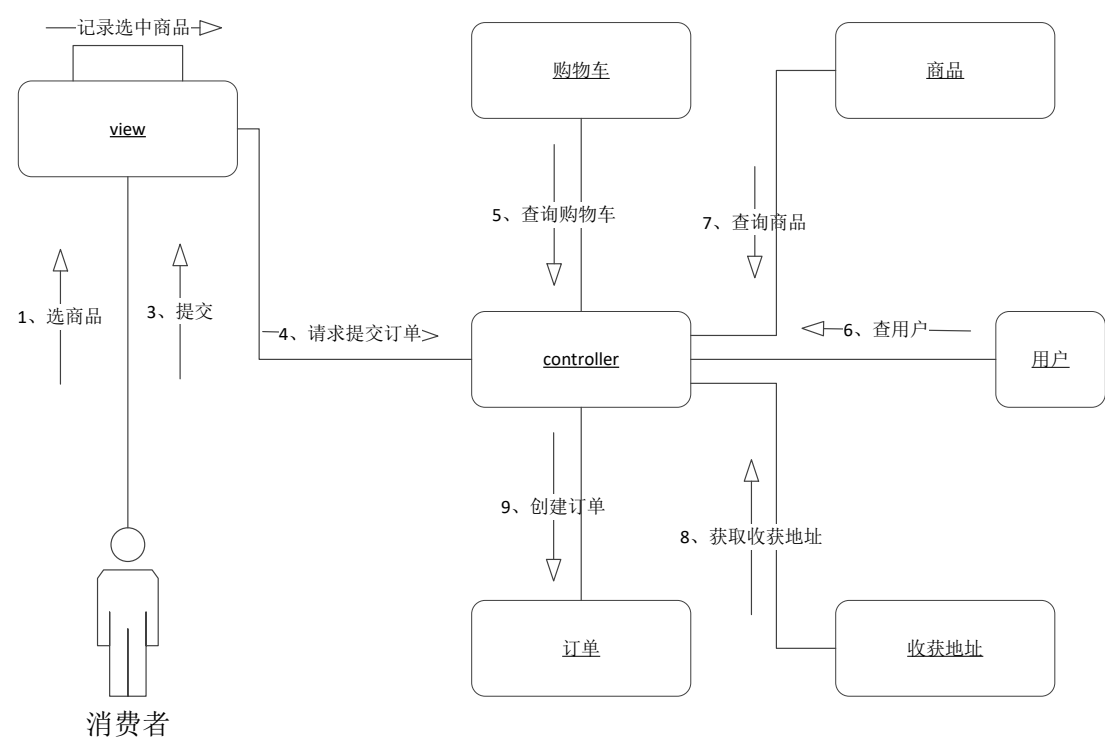


图 3-21 下单协作图

3.4.3.3 出库协作图

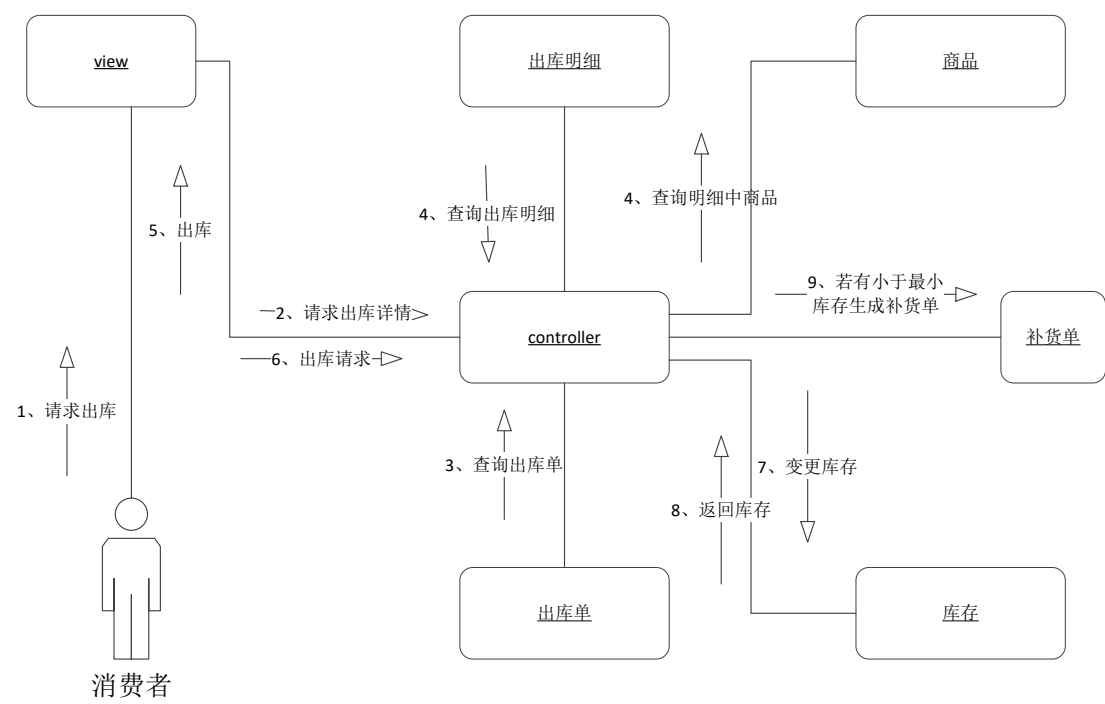


图 3-22 出库协作图

3.4.4 活动图

3.4.4.1 添加购物车活动图

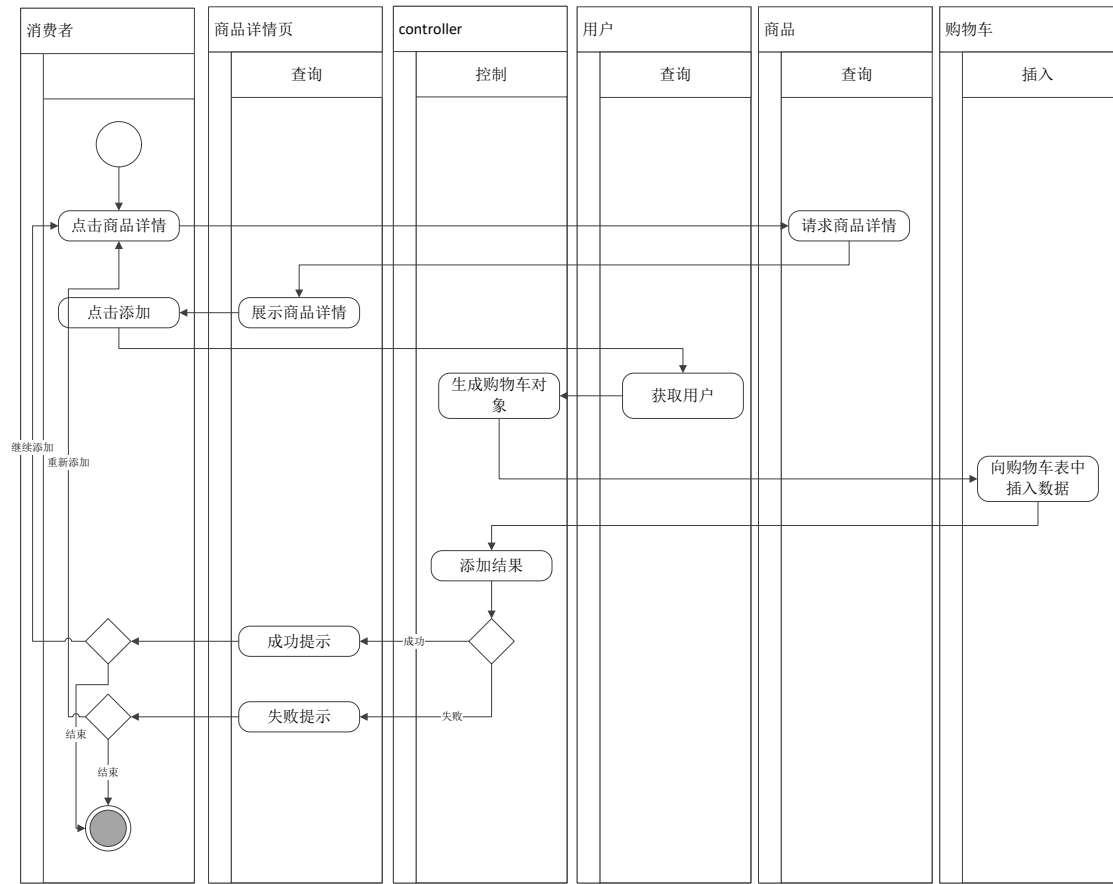


图 3-23 添加购物车活动图

3.4.4.2 用户下单活动图

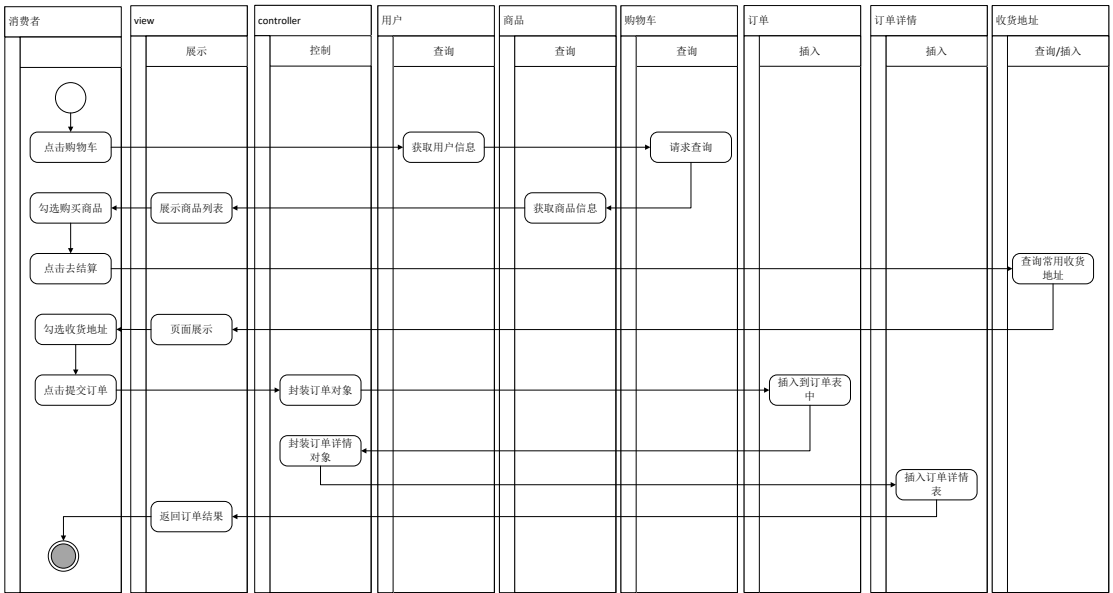


图 3-24 用户下单活动图

3.4.4.3 出库活动图

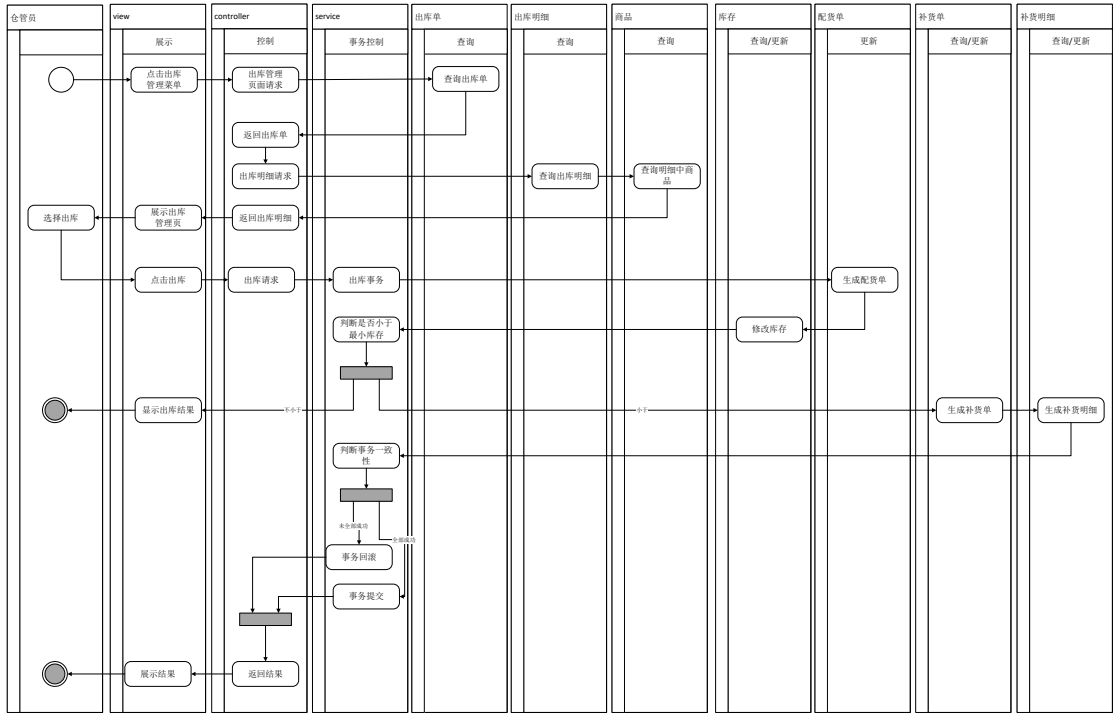


图 3-25 出库活动图

第四章 系统总体设计

4.1 软件模块结构设计

4.1.1 HIPO 分层图

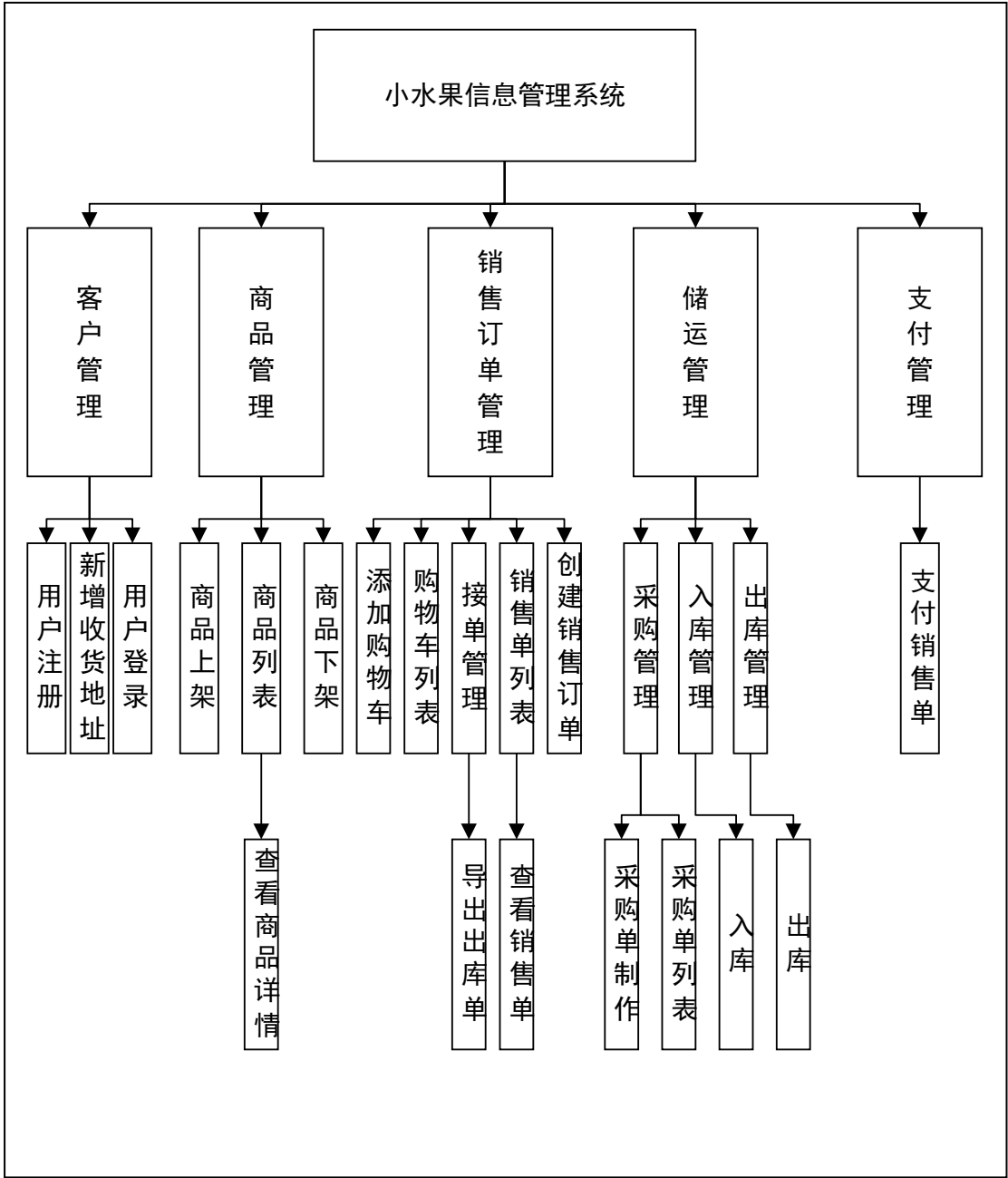


图 4-1 HIPO 分层图

4.1.2 IPO 图

4.1.2.1 树根节点 IPO 图

系统名称：小水果信息管理系统 模块名称：小水果信息管理系统	设计者：霍强 日期：2016/10/15
上层调用模块： 无	可调用下层模块： 1、 客户管理 2、 商品管理 3、 订单管理 4、 储运管理 5、 支付管理
输入： 1、 用户名 2、 密码 3、 验证码	输出： 1、 商品信息 2、 订单信息 3、 物流信息
处理： 1、 根据用户名查询用户信息 2、 查询的结果与密码进行匹配 3、 查询商品品类 4、 查询商品信息	

图 4-2 根节点 IPO 图

4.1.2.2 内部节点 IPO 图

系统名称：小水果信息管理系统 模块名称：销售订单管理	设计者：霍强 日期：2016/10/15
上层调用模块： 小水果信息管理系统	可调用下层模块： 1、 添加购物车 2、 购物车列表 3、 创建销售订单 4、 销售订单列表 5、 接单管理
输入： 1、 员工编号	输出： 1、 销售订单管理界面 2、 显示该登录员工的待处理销售订单
处理： 1、 请求销售订单管理后台界面 2、 查询销售订单数据 3、 返回销售订单数据	

图 4-3 内部节点 IPO 图

4.1.2.3 树叶节点 IPO 图

系统名称：小水果信息管理系统 模块名称：创建销售单	设计者：霍强 日期：2016/10/15
上层调用模块： 小水果信息管理系统	可调用下层模块： 无
输入： 2、商品编号或商品名称 3、订单编号 4、消费者编号	输出： 3、商品详情 4、订单详情
处理： 1、根据商品编号或商品名称查询商品详情 2、从 session 中获取用户登录信息，获取用户 ID 3、根据商品信息及用户信息创建订单 4、根据订单编号查询商品详情	

图 4-4 创建销售单 IPO 图

系统名称：小水果信息管理系统 模块名称：出库	设计者：霍强 日期：2016/10/15
上层调用模块： 出库管理	可调用下层模块： 无
输入： 1、出库单	输出： 1、已出库的销售单 2、已发货的订单 3、变更后的库存
处理： 1、根据出库单中的商品查询库存 2、减少对应的库存数量 3、修改对应订单状态为已出库	

图 4-5 出库 IPO 图

4.2 数据库设计

4.2.1 需求分析

4.2.1.1 消费者需求

对于消费者，可以在小水果网上查看浏览及搜索各处商品，对于意向的商品可以进行添加到购物车操作(在添加购物车前需要登录该网站，否则弹出登录框)，可对购物车中的商品进行购买(下订单)操作，下单后可立即支付，支付的订单会让接单员去做接单处理，然后由仓管员进行发货。线上收货后可以在网上进行确认收货及对购买的商品进行线上点评操

作。

4.2.1.2 工作人员需求

对于运营人员的需求：运营人员可以能过后台上架和下架商品。管理商品的信息及商品图片的上传等操作。

对于采购人员的需求：采购人员可通过后台创建采购单及采购明细，然后将采购单发给对应的供应商。

对于仓管员的需求：仓管员可能过后台进行仓库的管理，包括出库，采购入库及对应库存的管理等。

对于接单员的需求：接单员可以查看到未接收的订单，可以对未接收的订单作接收处理，接收处理的订单会让仓管员看到该订单，并能对此订单进行发货操作等。

4.2.2 实体描述

- 1、仓库管理员[名称、性别、生日、备注]。
- 2、采购员[名称、性别、生日、备注]。
- 3、运营专员[名称、性别、生日、备注]。
- 4、消费者[消费者姓名、性别、出生年月、手机号、邮箱、用户名、密码、备注]。
- 5、商品[商品名称、商品产地、商品售价、备注]。
- 6、收货地址[收货地址名称、收货人手机号、区号、收货人电话、收货人姓名、省、市、区/县、地址、邮政编码、备注]。
- 7、供应商[供应商名称、地址、联系人、联系电话]。
- 8、仓库[仓库名称、仓库地址、仓库启用时间、仓库面积]。
- 10、购物车[备注]。
- 11、购物车商品明细[备注]。
- 12、销售订单[备注]。
- 13、销售订单明细[商品、销售价格、数量、备注]。
- 14、采购单[备注]。
- 15、采购单明细[商品、采购单价、数量、备注]。
- 16、入库单[备注]。
- 17、入库单明细[备注]。
- 18、出库单[备注]。
- 19、出库单明细[备注]。
- 20、补货单[备注]。
- 21、补货单明细[备注]。

4.2.3 联系描述

4.2.3.1 一对一的二元关系有：

接单员与销售订单、消费者与销售订单、消费者与出库单、仓管员与出库单、仓管员与入库单、供应商与入库单、供应商与采购单、采购员与采购单、销售单明细与商品、采购单与明细与商品、出库单明细与商品、入库单明细与商品、消费者与购物车、仓管员与仓库。

涉及 1：每一个销售订单会与对应的一个接单员进行关联，销售订单涉及接单员。

涉及 2：每一个销售订单都是一个消费者创建的，销售订单涉及消费者。

涉及 3：每一个出库单涉及一个销售订单，出库单是因订单而导出。

涉及 4：每一个出库单都会与一个仓管员对应，出库单涉及一名仓管员。

涉及 5：每一个入库单同样会与一个仓管员对应，入库单涉及一名仓管员。

涉及 6：每一个入库单会与一名供应商对应，入库单涉及一名供应商。

涉及 7：每一个采购单都会与供应商进行对应，采购单涉及一名供应商。

涉及 8：每一个采购单同样会与一名采购员进行关联，采购单涉及采购员。

涉及 9：每一个购物车商品明细，记录的是消费者购买的一种商品，购物车商品明细涉及商品。

涉及 10：每一个销售单明细，都记录的是消费都购买的一种商品，销售单明细涉及商品。

涉及 11：每一个出库明细，同样记录的是一种待出库的商品，出库明细涉及商品。

涉及 12：每一个入库明细记录着的是一种待入库的商品，入库明细涉及商品。

涉及 13：每一个采购明细记录着的是一种准备采购的商品，采购明细涉及商品。

涉及 14：每个在销售的商品，都会与小水果的运营人员进行对应，小水果的商品是由运营人员专门来负责运营的，由运营人员进行商品的上下架管理，每个商品涉及一名运营人员。

涉及 15：每一个补货单设计一名仓管员，因此补单和仓管员是 1 对 1 的二元关系。

涉及 16：每一个补货明细跟一种商品进行对应，补货明细涉及商品。

使用：一名消费是可以使用自己的购物车进行网上购物的，一个购物车只能被一名消费者来使用，则消费者与购物车之间的使用关系是一种一对一的二元关系。

4.2.3.2 一对多的二元关系有：

销售订单与销售订单明细、采购单与采购明细、出库单与出库明细、入库单与入库明细、购物车与购物车商品明细、消费者与收货地址、运营专员与商品、商品与货架。

包含 1：购物车是存放的是商品，一个消费者的购物车可以存放多件商品，购物车包含购物车商品明细。

包含 2：销售订单是由多条销售订单明细组成的，销售订单包含多条销售订单明细。

包含 3：出库单是由多条出库明细组成的，出库单包含多条出库明细。

包含 4：入库单是由多条入库明细组成的，入库单包含多条入库明细。

包含 5：采购单是由多条采购明细组成的，采购单包含多条采购明细。

包含 6：补货单是由多条补货明细组成的，补货单包含多条补货明细。

属于：每一条收货地址，都是属于一个消费者的。

运营：每一种商品都由一名运营专员负责来运营的。

4.2.3.3 多对多的二元关系有：

存放：小水果的每一种商品都会存储在小水果自己的仓库中的，一种商品是存放在同一地区或是不同地区的多个仓库中的，同样一个仓库也是存放着小水果的多种商品的。则小水果的商品与仓库是一种多对多的二元关系。

由于小水果是一种网上销售的电商企业，故不像超市一样存在货架，所以只有商品存放仓库这一种存放关系。

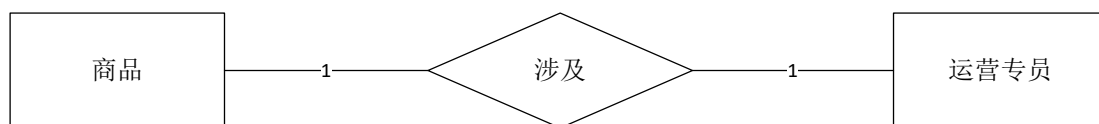


图 4-7



图 4-8

商品[商品编号、商品名称、商品产地、商品售价、商品描述、备注]。

由于某个商品涉及一个运营专员运营(商品为专人运营的), 运营专员运营着多个商品, 运营专员和商品之间是 1 对多的二元关系, 则需要在商品中添加运营专员编号作为商品的外码。运营专员可以上架属于自己的商品。得到新的模式有:

商品[商品编号、商品名称、商品产地、商品售价、商品描述、运营专员编号#、备注]。

运营专员运营一件商品, 运营包括上架商品、销售商品和下架商品, 为了区别上架与下架, 可将商品添加一个状态, 用于区分商品, 上架为可售状态, 下架为停售状态, 得到新的关系模式为:

商品[商品编号、商品名称、商品产地、商品售价、商品描述、运营专员编号#、商品状态(可售/停售)、备注]。

4.2.5.2.2 消费者与收货地址联系的转换



图 4-9

收货地址[收货地址编号、收货地址名称、收货人手机号、区号、收货人电话、收货人姓名、省、市、区/县、地址、邮政编码、备注]。

由于收货地址是属于某个用户的, 用户(消费者)与收货地址是 1 对多的关系, 故收货地址中引入消费者编号作为收货地址的外码, 得到新关系模式如下:

收货地址[收货地址编号、收货地址名称、消费者编号#、收货人手机号、区号、收货人电话、收货人姓名、省、市、区/县、地址、邮政编码、备注]。

4.2.5.2.3 消费者与商品联系的转换

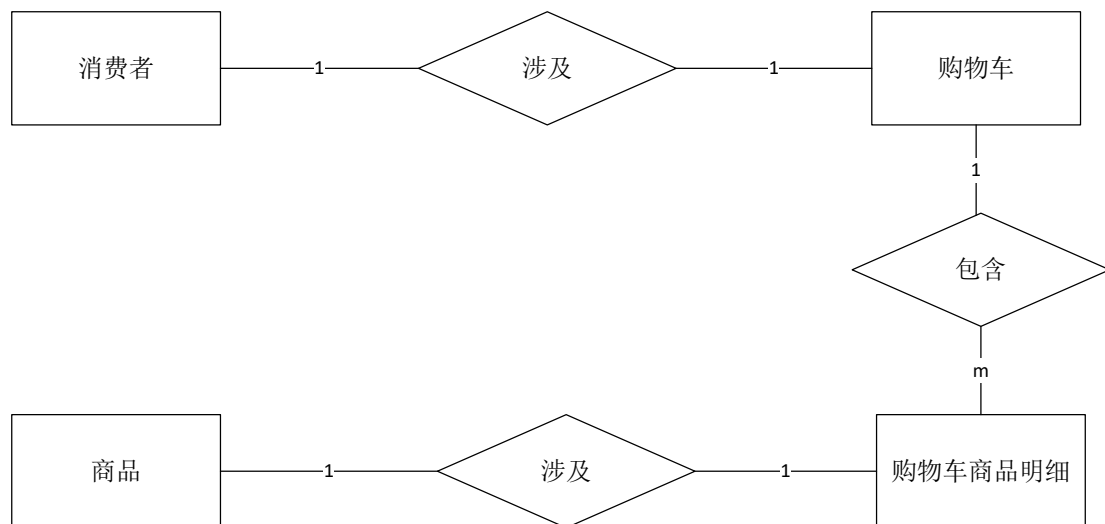


图 4-10

由于添加购物车是一个动作，关联了消费者与商品，现引入虚实体购物车，购物车包含购物车商品明细，存储了消费者与商品的一种意向购买关系，所以消费者编号作为购物车的外码，而购物车商品明细记录的则是购物车中意向购买的商品的情况，则购物车商品明细中引入购物车编号作为外码。

购物车[购物车编号、消费者编号#、备注]。

购物车明细[购物车编号#、商品编号#、数量、添加日期、备注]

4.2.5.2.4 消费者购买联系的转换

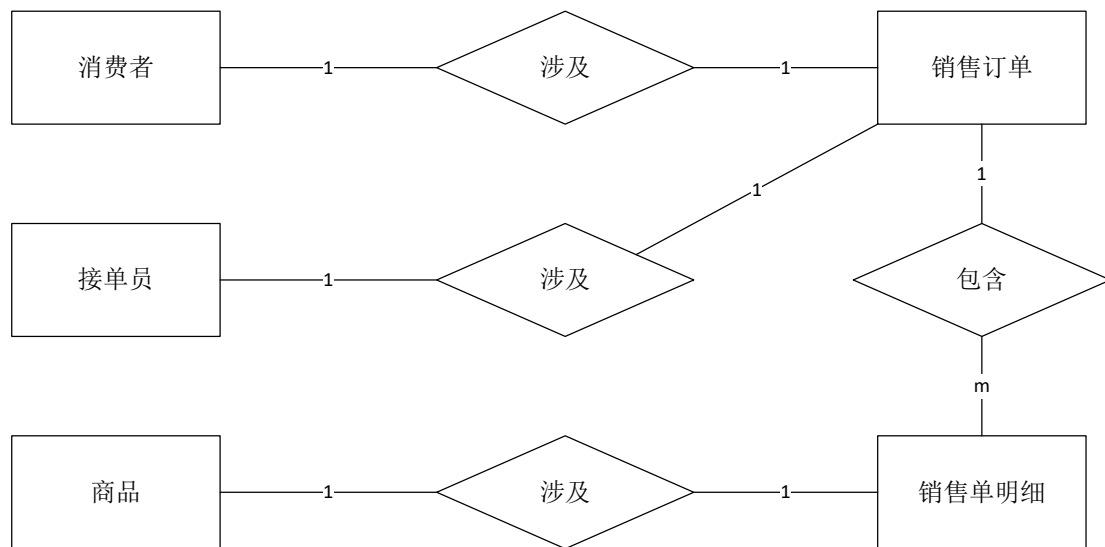


图 4-11

消费者[消费者编号、消费者姓名、用户名、密码、性别、出生年月、手机号、邮箱、VIP 会员等级、备注]。

购买产生虚实体“订单”，订单中引用消费者编号作为外码，由于订单发货需要填写收货地址，故将收货地址编号作为订单的外码。

销售订单[订单编号、消费者编号#、创建时间、收货地址编号#]。

销售订单创建后，有接单员来处理，故订单中还应包含接单员编号，作为订单的外码。

新的模式如下：

销售订单[订单编号、消费者编号#、创建时间、收货地址编号#、接单员编号#、备注]。

消费者一笔订单可以购买多个商品，故引入虚实体订单详情，订单与订单明细为 1 对多的关系，故在订单详情中引用订单编号作为订单详情的外码。

销售订单明细[订单编号#、商品、销售单价、数量、生成时间、备注]。

由于商品存储了商品价格等相关信息，则将销售订单明细中引入商品编号作为外码进行关联，删除销售价格等冗余字段，得到销售订单明细如下：

销售订单明细[订单编号#、商品编号#、数量、销售单价、生成时间、备注]。

4.2.5.2.4 商品和仓库的转换关系

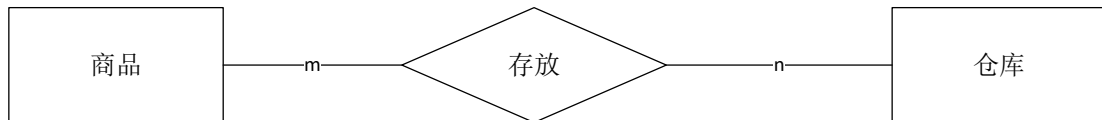


图 4-12

由于仓库中存放了多种商品，则商品和仓库有存放(库存)关系，仓库和货架存在着 1 对多的二元关系，关系模式如下

仓库[仓库编号、仓库名称、仓库地址、仓库面积、备注]。

存放[商品编号#、仓库编号#、存放数量、备注]。

由于存放体现的是一种商品在某仓库中的库存关系，则将名称改为库存，将存放数量改为库存数量，得到新的关系模式如下：

库存 [商品编号#、仓库编号#、库存数量、备注]。

4.2.5.2.5 采购和入库联系的转换

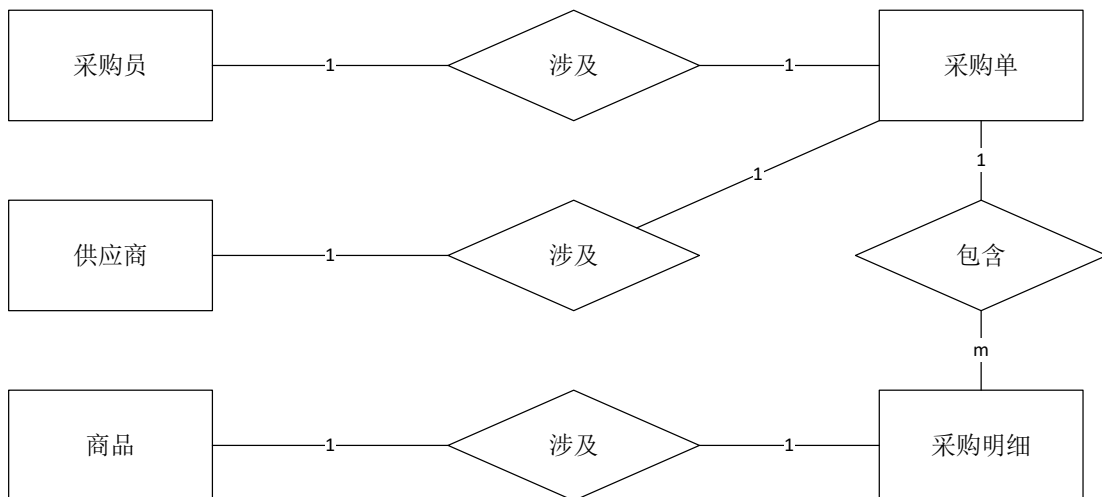


图 4-13

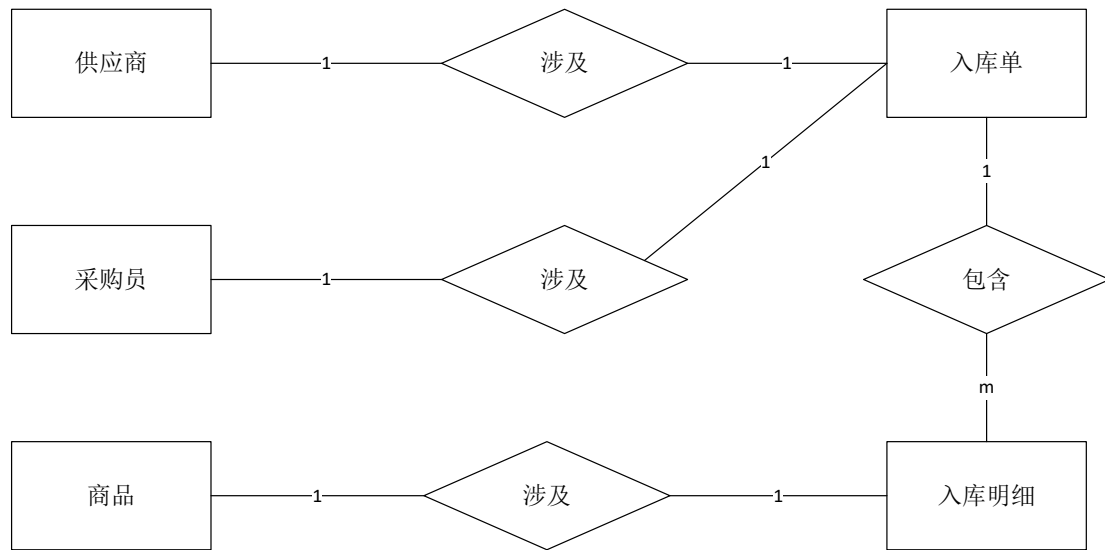


图 4-14

供应商[供应商编号、名称、地址、联系人、联系电话]。

采购员向供应商采购商品，引入采购单虚实体，由于一次可采购多种商品，故采购单引入采购明细模式，采购明细中引用采购单编号作为外码进行关联，采购明细中的商品与仓库进行关联，需要引入仓库编号作为采购明细的外码。

采购单[采购单编号、采购名称、采购员编号#、供应商编号#、采购时间、供货时间、数量、备注]。

采购明细[采购单编号#、商品编号#、仓库编号#、采购单价、采购数量、创建时间、备注]。

入库单[入库单编号、采购单编号#、采购员编号#、入库时间、数量、备注]。

入库明细[入库单编号#、商品编号#、仓库编号#、入库数量、入库时间、备注]。

由于入库是因采购而入库，既供应商按照采购员提供的采购单及明细向指定的仓库供应商品的。故采购入库可与采购明细进行合并，添加状态标识(待入库/已入库)、创建采购明细的时候默认为待入库，供应商将商品送达指定仓库后状态改为已入库，同时记录当前时间作为入库时间，得到新的关系模式为：

采购明细[采购单编号#、商品编号#、仓库编号#、采购单价、采购数量、创建时间、供货状态(待入库/已入库)、入库时间、备注]。

因采购入库和出库而产生仓库中商品数量的动态变化，引入虚实现库存，库存中引用商品编号作为外码，得到关系模式如下：

库存[商品编号#、仓库编号#、商品批次号、库存数量、可售库存量、最小库存量、创建时间、备注]。

由于库存包含了商品编号及仓库编号，已经表达了商品与库存的一种存放关系，则与前面的存放关系模式进行合并，得到新的关系模式为：

库存[商品编号#、仓库编号#、库存数量、可售库存量、最小库存量、创建时间、备注]。

4.2.5.2.5 出库联系的转换

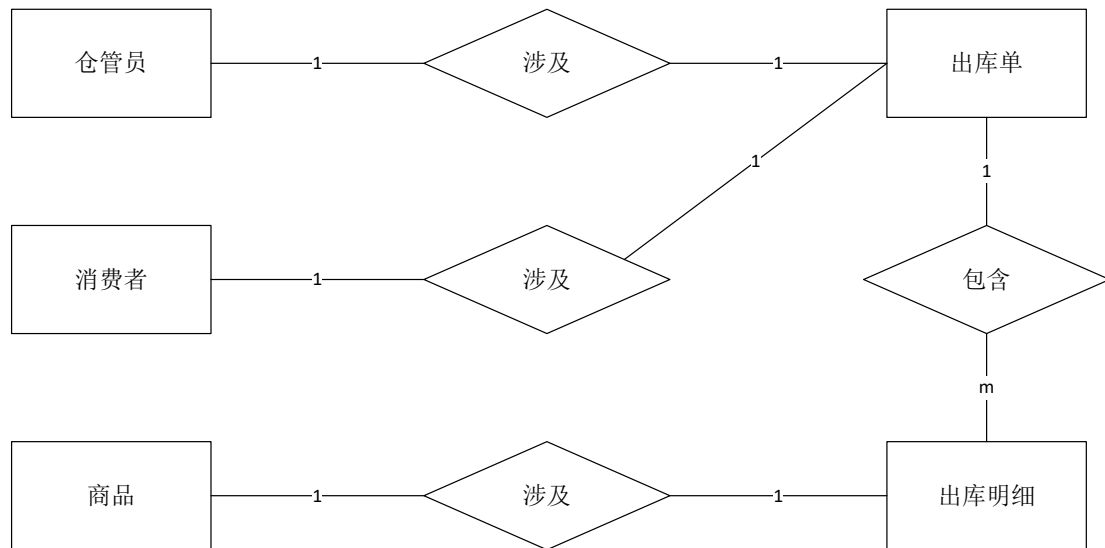


图 4-15

由于出库是因订单发货而出库, 故出库应和订单关联, 一个订单对应 1 或多个出库单(如果订单中的所有商品在同一个仓库, 则对应一个出库单, 否则对应多个出库单), 引入出库单虚实体出库单, 出库单引用订单编号作为外码, 而订单中可能有多件商品, 则还需引入出库明细, 出库明细与商品一一对应, 出库明细中引入商品编号作为外码, 得到关系模式如下:

出库单[出库单编号、订单编号#、出库时间、出库人#、备注]。

出库明细[出库单编号#、商品编号#、仓库编号#、出库时间、出库状态、备注]。

4.2.5.2.6 仓管员与仓库联系的转换



图 4-16

由于仓库配置有仓库管理员, 则仓库中引入仓管员编号作为外码, 得到新的关系模式为:
仓库[仓库编号、仓库名称、仓库地址、仓库面积、仓管员编号#、备注]。

4.2.6 关系模式

员工[员工编号、名称、性别、生日、角色、备注]

消费者[消费者编号、消费者姓名、用户名、密码、性别、出生年月、手机号、邮箱、VIP 会员等级、备注]。

商品[商品编号、商品名称、商品产地、商品售价、商品描述、运营专员编号#、商品状态(可售/停售)、备注]。

收货地址[收货地址编号、收货地址名称、消费者编号#、收货人手机号、区号、收货人电话、收货人姓名、省、市、区/县、地址、邮政编码、创建时间、修改时间]。

购物车[购物车编号、消费者编号#、备注]。

购物车明细[购物车编号#、商品编号#、数量、添加日期、备注]

销售订单[订单编号、消费者编号#、创建时间、收货地址编号#、接单员编号#、备注]。

销售订单明细[订单编号#、商品编号#、数量、销售单价、生成时间、备注]。

供应商[供应商编号、名称、地址、联系人、联系电话]。

仓库[仓库编号、仓库名称、仓库地址、仓库面积、仓管员编号#、备注]。

库存[商品编号#、仓库编号#、库存数量、可售库存量、最小库存量、备注]。

采购单[采购单编号、采购名称、采购员编号#、供应商编号#、采购时间、供货时间、数量、备注]。

采购明细[采购单编号#、商品编号#、仓库编号#、采购单价、采购数量、创建时间、供货状态(待入库/已入库)、入库时间、备注]。

出库单[出库单编号、订单编号#、出库时间、出库人#、备注]。

出库明细[出库单编号#、商品编号#、仓库编号#、出库时间、出库状态、备注]。

入库单[入库单编号、采购单编号#、采购员编号#、入库时间、数量、备注]。

入库明细[入库单编号#、商品编号#、仓库编号#、入库数量、入库时间、备注]。

补货单[补货单编号、仓管员编号#、数量、备注]。

补货明细[补货单编号#、商品编号#、仓库编号#、补货数量、备注]。

4.2.7 数据库表

4.2.7.1 员工表

表名	employee				
外键引用表 [字段名]					
说明	员工表，包括运营专员、接单员、仓管员等				
字段	名称	数据类型	长度	空值	备注
Employee_id	员工编号	Bigint			PK
name	名称	varchar	50		
gender	性别	char	1	null	
birthday	生日	Date		Null	
role	角色	tinyint			
remark	备注	varchar	200	null	

表 4-1 员工表

4.2.7.2 消费者表

表名	Customer				
外键引用表 [字段名]					
说明	消费者表，用来存放小水果网站的用户，他们是小水果的主要消费者群体				
字段	名称	数据类型	长度	空值	备注
Customer_id	消费者编号	Bigint			PK
Name	消费者姓名	varchar	50		
User_name	用户名	Varchar	50		
password	密码	Varchar	255		
Gender	性别	char	1	null	
Birtyday	出生年月	date		Null	
mobile	手机号	Varchar	15	Null	
email	邮箱	varchar	50	Null	

Vip_level	Vip 会员等级	tinyint		Null	
remark	备注	varchar	200	null	

表 4-2 消费者表

4.2.7.3 商品表

表名	Goods				
外键引用表 [字段名]					
说明	用来存放小水果用于网上售卖的商品				
字段	名称	数据类型	长度	空值	备注
Goods_id	商品编号	bigint			FK
Goods_name	商品名称	Varchar	50		
producing_area	商品产地	varchar	200	null	
Sell_price	商品售价	bigint			
description	商品描述	varchar	200	null	
Employee_id	运营专员编号	bigint			
status	商品状态	tinyint			
remark	备注	varchar	20	null	

表 4-3 商品表

4.2.7.4 收货地址表

表名	Address				
外键引用表 [字段名]	Customer[customer_id]				
说明	用来存放用户的常用收货地址。				
字段	名称	数据类型	长度	空值	备注
Address_id	收货地址编号	bigint			FK
Address_name	收货地址别名	Varchar	50		
Customer_id	消费者编号	bigint			
Receiver_mobile	收货人手机号	Varchar	15		
Area_code	区号	Varchar	4	Null	
Receiver_phone	收货人电话	Varchar	12	Null	
Receiver_name	收货人姓名	Varchar	50		
province	省	Varchar	20		
city	市	Varchar	20		
county	区/县	Varchar	20		
address	地址	Varchar	50		
Post_code	邮政编码	Char	6	Null	
Create_time	添加时间	Datetime			
Update_time	修改时间	datetime			

表 4-4 收货地址表

4.2.7.5 购物车表

表名	Shopping_cart				
外键引用表 [字段名]	Customer[customer_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Shp_cart_id	购物车编号				
Customer_id	消费者编号				
remark	备注				

表 4-5 购物车表

4.2.7.6 购物车明细表

表名	Shopping_cart_detail				
外键引用表 [字段名]	Shopping_cart[shp_cart_id]				
说明	用来存放消费者在购物车中的商品				
字段	名称	数据类型	长度	空值	备注
Shp_cart_id	购物车编号	Bigint			
Goods_id	商品编号	Bigint			
Quantity	数量	Smallint			
Add_time	添加时间	Datetime		Null	
remark	备注	Varchar	200	Null	

表 4-6 购物车明细表

4.2.7.7 销售订单表

表名	Order				
外键引用表 [字段名]	Customer[customer_id]				
说明	用来存放消费者在网上进行提交的订单				
字段	名称	数据类型	长度	空值	备注
Order_id	订单号	Bigint			PK
Customer_id	消费者编号	Bigint			
Create_time	创建时间	Datetime			
Receiver_id	收货地址编号	Bigint			
Employee_id	接单员编号	Bigint			
remark	备注	Varchar	200		

表 4-7 销售订单表

4.2.7.8 销售订单明细表

表名	Order_detail				
外键引用表 [字段名]	Order[order_id]				
说明					

字段	名称	数据类型	长度	空值	备注
Order_id	订单编号	Bigint			
Goods_id	商品编号	Bigint			
Quantity	数量	Smallint			
Sell_price	销售单价	Bigint			
Create_time	创建时间	Datetime			
remark	备注	Varchar	200	null	

表 4-8 销售订单明细表

4.2.7.9 供应商表

表名	supplier				
外键引用表 [字段名]					
说明					
字段	名称	数据类型	长度	空值	备注
Supplier_id	供应商编号	Bigint			
Name	名称	Varchar	50		
Address	地址	Varchar	50		
contacts_name	联系人姓名	Varchar	50		
contacts_mobile	联系人电话	Varchar	15		

表 4-9 供应商表

4.2.7.10 仓库表

表名	repostory				
外键引用表 [字段名]	Employee[employee_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Repostory_id	仓库编号	Bigint			
Repository_name	仓库名称	Varchar	50		
Repository_address	仓库地址	Varchar	100		
area	仓库面积	Smallint		Null	
Employee_id	仓管员编号	Bigint			
remark	备注	Varchar	200	Null	

表 4-10 仓库表

4.2.7.11 库存表

表名	Repository_inventory				
外键引用表 [字段名]	Repository[repository_id] Goods[goods_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Goods_id	商品编号	Bigint			
Repository_id	仓库编号	Bigint			

quantity	库存数量	Bigint			
selling_quantity	可售库存量	Bigint			
Min_quantity	最小库存量	Bigint		Null	
remark	备注	Varchar	200	Null	

表 4-11 库存表

4.2.7.12 采购单表

表名	Purchase_order				
外键引用表 [字段名]	Employee[employee_id] Supplier[supplier_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Purchase_id	采购单编号	Bigint			
Purchase_name	采购名称	Varchar	50		
Employee_id	采购员编号	Bigint			
Supplier_id	供应商编号	Bigint			
Purchase_date	采购时间	Datetime			
Supplier_date	供货时间	Datetime		Null	
quantity	数量	Bigint			
remark	备注	Varchar	200	null	

表 4-12 采购单表

4.2.7.13 采购单明细表

表名	Puschase_orde_detail				
外键引用表 [字段名]	Reporstory[repository_id] Purchar_order[employee_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Purchase_id	采购单编号	bigint			
Goods_id	商品编号	Bigint			
Repository_id	仓库编号	Bigint			
Purchase_price	采购单价	Bigint			
quantity	采购数量	Bigint			
Create_time	创建时间	Datetime			
status	供货状态	Tinyint		null	
In_store_time	入库时间	Datetime		Null	
remark	备注	Varchar	200	Null	

表 4-13 采购单明细表

4.2.7.14 出库单表

表名	Outbound				
外键引用表 [字段名]	Order[order_id]				
说明					

字段	名称	数据类型	长度	空值	备注
Outbound_id	出库单编号	Bigint			
Order_id	订单编号	Bigint			
Outbound_time	出库时间	Datetime			
Employee_id	出库人	Bigint			
remark	备注	Varchar	200	Null	

表 4-14 出库单表

4.2.7.15 出库单明细表

表名	Outbound_detail				
外键引用表 [字段名]	Outbound[outbound_id] Repository[repository_id]				
说明					
字段	名称	数据类型	长度	空值	备注
Outbound_id	出库单编号	Bigint			
Goods_id	商品编号	Bigint			
Repository_id	仓库编号	Bigint			
Outbound_time	出库时间	Datetime		null	
status	出库状态	Tinyint			
remark	备注	Varchar	200	Null	

表 4-15 出库单明细表

4.2.7.16 入库单表

表名	instore				
外键引用表 [字段名]	Purchase_order[purchase_order_id] Employee[employee_id]				
说明					
字段	名称	数据类型	长度	空值	备注
In_store_id	入库单编号	Bigint			
Purchase_id	采购单编号	Bigint			
Employee_id	采购员编号	Bigint			
In_store_time	入库时间	Datetime			
quantity	数量	Bigint			
remark	备注	Varchar	200	Null	

表 4-16 入库单表

4.2.7.17 入库单明细表

表名	Instore_detail				
外键引用表 [字段名]	Instore[instore_id] Repository[repository_id]				
说明					
字段	名称	数据类型	长度	空值	备注
In_store_id	入库单编号	bigint			
Goods_id	商品编号	Bigint			

Repository_id	仓库编号	Bigint			
quantity	入库数量	Bigint			
In_store_time	入库时间	Datetime			
remark	备注	Varchar	200	null	

表 4-17 入库单明细表

表 4-15 出库单明细表

4.2.7.16 补货单表

表名	to_purchase				
外键引用表 [字段名]	Employee[employee_id]				
说明					
字段	名称	数据类型	长度	空值	备注
To_purchase_id	补货单编号	Bigint			
Employee_id	仓管员编号	Bigint			
quantity	数量	Bigint			
remark	备注	Varchar	200	Null	

表 4-18 补货单表

4.2.7.17 补货单明细表

表名	to_purchase_detail				
外键引用表 [字段名]	to_purchase [To_purchase_id] Repository[repository_id]				
说明					
字段	名称	数据类型	长度	空值	备注
In_store_id	补货单编号	bigint			
Goods_id	商品编号	Bigint			
repository_id	仓库编号	Bigint			
quantity	数量	Bigint			
remark	备注	Varchar	200	null	

表 4-19 补货单明细表

4.3 计算机系统配置方案

4.3.1 计算机系统硬件配置

最小内存 1G、主频 1.3GHz 以上、单核及以上处理器、显示器分辨率 1024*768 及以上、需要有网卡，可以至少可以 ping 通回环 IP 地址(127.0.0.1)。

4.3.2 计算机系统软件配置

Window2003 及以上 window 操作系统或 linux 操作系统(centOS、ubantu 等，客户机需安装界面) IE6 及以上 IE 版本浏览器、chrome 浏览器、firefox 浏览器、360 安全浏

览器、safari 浏览器等。(IE10、chrome、firefox 效果更佳) JDK1.4 及以上版本。My sql DBMS 4.0 及以上版本。

4.4 系统安全性、可靠性方案

4.4.1 系统安全性

本系统采用 SpringMVC+MyBatis+Spring 框架模式来开发。前端用 Freemarker 做视图解析层面。持久化数据库用 MyBatis, sql 语句手写灵活, 性能较优。用 Spring 来进行事务管理。同时对 url 进行了拦截器的配置, 对于特定的访问会通过拦截器进行拦截, 然后验证用户身份信息, 确保未登录的用户不可访问部分操作。

数据库和服务分开部署, 数据库采用 slave, master 主备模式, 确保数据库不被供给, 同时当数据库负载过高, 可以进行热切换, 让用户无感知。主库主要进行数据的写操作, 而一些非实时性较高的需求, 则从备库进行数据的读取, 减少对主库的压力。备库的数据通过实时向备库进行同步, 确保数据的完整性。

综上所述, 服务安全, 数据安全, 前端页面解析安全均是可靠的。

4.4.2 系统可靠性

4.4.2.1 环境可靠性

使系统的元器件工作在正常状态下, 没有过载超负荷等现象的发生, 并且要有一定的资源空闲度。也可以采用冗余贮备, 使系统即使有个别元器件或设备出现故障仍能正常工作, 譬如大型客机拥有四个发动机, 中型客机拥有两个发动机。也就是说有一个设备出现故障, 有另一个设备顶替它工作。数据库可以主备的方式配置, 同时支持读写分离操作。

4.4.2.2 架构及代码可靠性

本系统采用 java 语言 +mysql 数据库管理系统的完美结合, 采用 SpringMVC+MyBatis+Spring 框架, 系统架构分控制器、服务层和 DAO 层, 后台操作数据库采用数据库连接池, 提交效率, 在选取语言和数据库管理系统以及架构设计方面具有较高的可靠性。代码本身遵循强壮, 规范的理念书写, 条理清晰, 注视规范, 具有较高的可读性及可拓展性。总体而言, 本系统具有不错的可靠性。

第五章 系统详细设计

5.1 代码设计

5.1.1 销售单代码设计

销售单号: 销售订单号由一串数字组成, 由 mysql 数据库的自增长类型为一串不重复的数字。

销售单详情号: 同销售单号, 由数据库的自增长类型生成, 为一串不重复的数字。

收货地址编号: 收货地址由一串字符组成, 第一位为大区编号, 第 2 至 3 位为省份编号, 第 3 至 6 位为城市编号, 9 至 14 位为区县编号, 14 位及以上为地址顺序码。

5.1.2 采购单代码设计

采购单号：由一串 5 位以上数字组成，最小从 10000 开始，采购单根据创建时间的先后顺序，单号每次自加 1。

采购单明细号：由一串 8 位以上的数字组成，小从 10000000 开始，根据采购单明细的生成的时间先后顺序，单号每次自加 1。

5.1.3 商品代码设计

商品编号：商品编号为一串数字，前两位为商品的品类编号，中间三位为商品的种属编号，5 位到 10 及以上为商品的顺序号。商品编号最小为 10 位。

5.2 用户界面设计

5.2.1 登录注册界面设计

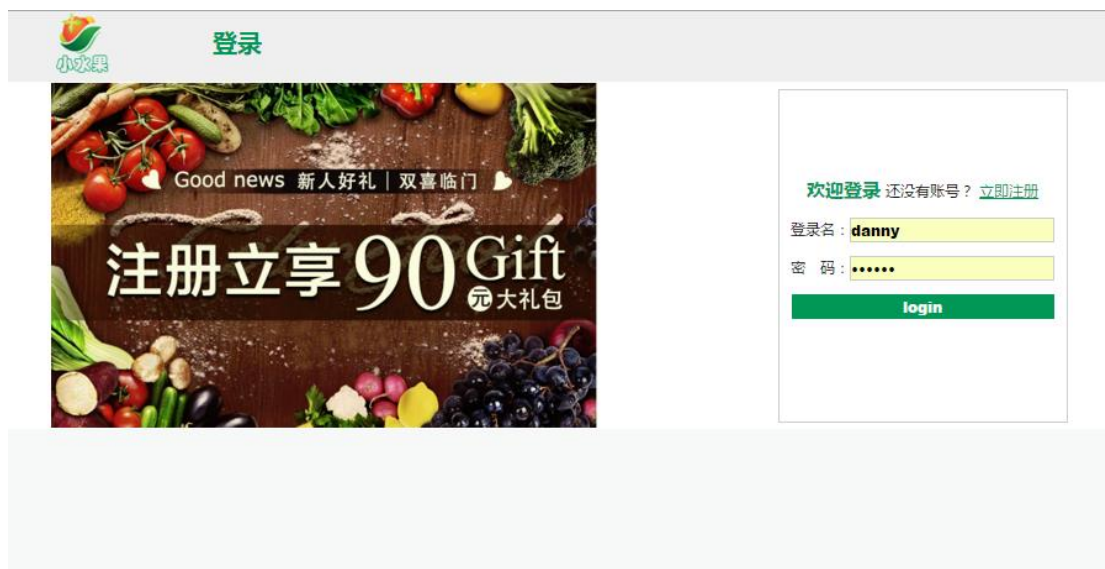



图 5-1 登录界面

注册

登录名：

密 码：

确认密码：

姓 名：

昵 称：

手机号：

邮 箱：

立即注册

图 5-2 注册界面

5.2.2 首页界面设计

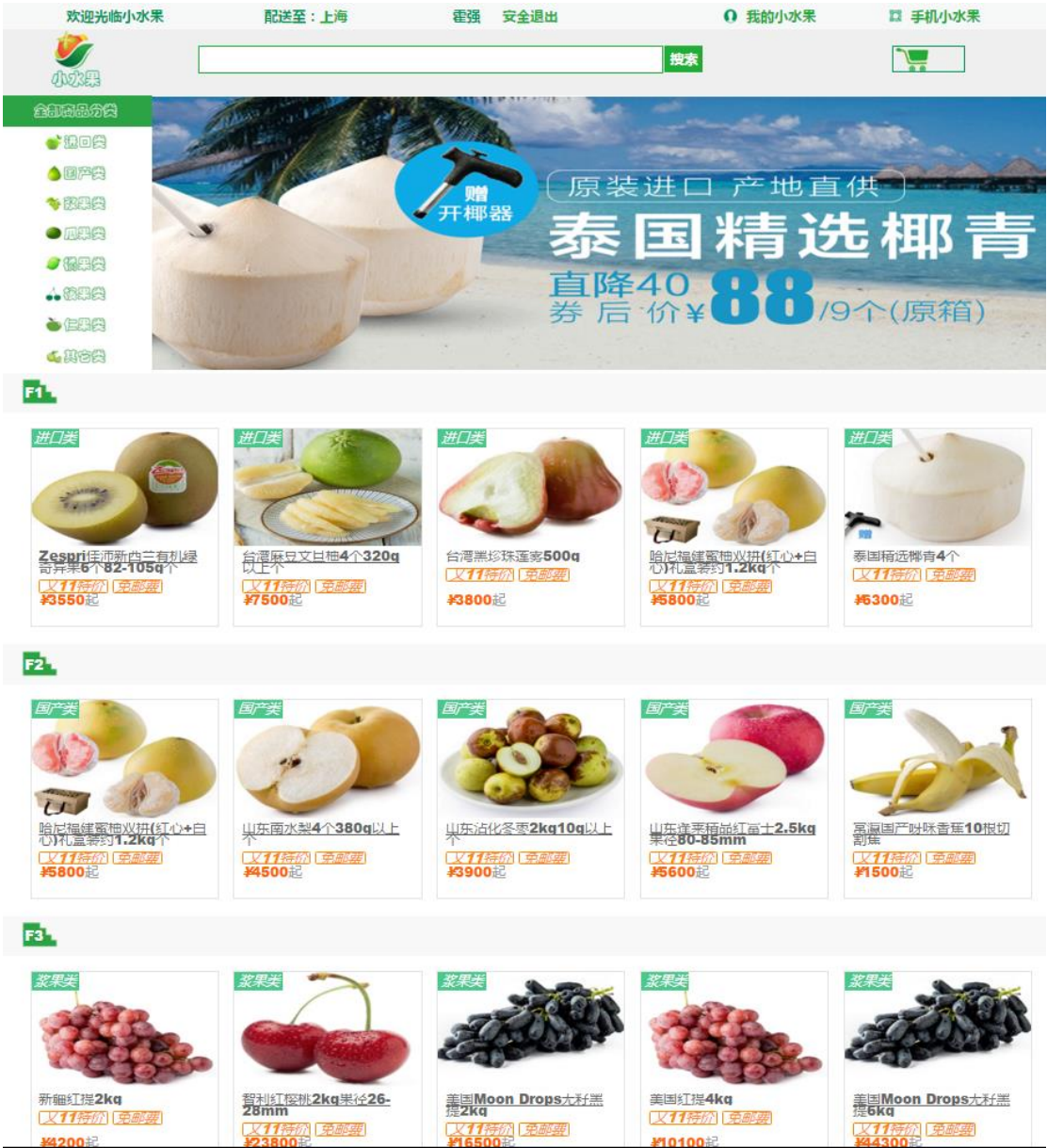



图 5-3 销售首页界面

图 5-15 采购管理界面

小水果后台管理系统

danny[管理员]安全退出



菜单管理

权限分配

出库管理

入库管理

仓库管理

供应商管理

员工管理

采购管理

商品管理

商品上架

采购名称：

供应商采购：

陕西红富士果园

商品明细列表：

1:Zespri佳沛新西兰有机绿奇异果6个82-105g个,100份

2:台湾麻豆文旦柚4个320g以上个,100份

商品：

台湾黑珍珠莲雾500g

采购数量：

添加明细

提交采购

图 5-6 制作采购单界面

5.2.5 出库管理界面设计

小水果后台管理系统

danny[管理员]安全退出



菜单管理

权限分配

出库管理

入库管理

仓库管理

供应商管理

员工管理

采购管理

商品管理

商品上架

序号	订单号	创建时间	订单状态	发货地址	操作
1	14	2016-11-25 15:37:50	待支付	崔先生 上海 普陀 中江路302弄68号	全部出库
	商品编号：45	数量：1		状态：待出库	出库
	商品编号：17	数量：1		状态：已出库	
	商品编号：26	数量：1		状态：待出库	出库
2	15	2016-11-25 15:38:00	待支付	崔先生 上海 普陀 中江路302弄68号	全部出库
	商品编号：20	数量：1		状态：待出库	出库
	商品编号：21	数量：2		状态：待出库	出库

图 5-7 出库管理界面设计


```

public CustomerVo login(String loginName, String password)
    throws UnsupportedEncodingException, NoSuchAlgorithmException{
    Map<String, Object> param = new HashMap<String, Object>();
    param.put("loginName", loginName);
    CustomerVo customerVo = new CustomerVo();
    customerVo.setLoginSuccess(false);
    List<CustomerVo> customers = customerMapper.querySelective(param);
    if(null != customers && customers.size() > 0){
        customerVo = customers.get(0);
        String inputPwd = PwdUtil.password(password, customerVo.getSafetyFactor());
        if(customerVo.getPassword().equals(inputPwd)){
            customerVo.setLoginSuccess(true);
            return customerVo;
        }
    }
    customerVo.setLoginMessage("用户名或密码有误!");
    return customerVo;
}

```

```

<select id="querySelective" resultMap="BaseResultMap"
parameterType="java.util.HashMap" >
    select
    <include refid="Base_Column_List" />
    from customer
    <where>
        1 = 1
        <if test="customerId != null">
            AND customer_id = #{customerId,jdbcType=BIGINT}
        </if>
        <if test="customerNo != null" >
            AND customer_no = #{customerNo,jdbcType=CHAR}
        </if>
        <if test="loginName != null" >
            AND login_name = #{loginName,jdbcType=VARCHAR}
        </if>
        <if test="password != null" >
            AND password = #{password,jdbcType=VARCHAR}
        </if>
        <if test="safetyFactor != null" >
            AND safety_factor = #{safetyFactor,jdbcType=VARCHAR}
        </if>
        <if test="realName != null" >
            AND real_name = #{realName,jdbcType=VARCHAR}
        </if>
        <if test="nickName != null" >
            AND nick_name = #{nickName,jdbcType=VARCHAR}
        </if>
        <if test="mobileNumber != null" >
            AND mobile_number = #{mobileNumber,jdbcType=VARCHAR}
        </if>
        <if test="email != null" >
            AND email = #{email,jdbcType=VARCHAR}

```

```

</if>
<if test="gender != null" >
    AND gender = #{gender,jdbcType=CHAR}
</if>
<if test="qq != null" >
    AND qq = #{qq,jdbcType=VARCHAR}
</if>
<if test="wechat != null" >
    AND wechat = #{wechat,jdbcType=VARCHAR}
</if>
<if test="birthday != null" >
    birthday = #{birthday,jdbcType=TIMESTAMP}
</if>
<if test="imageUrl != null" >
    AND image_url = #{imageUrl,jdbcType=VARCHAR}
</if>
<if test="vipLevel != null" >
    AND vip_level = #{vipLevel,jdbcType=TINYINT}
</if>
<if test="createTime != null" >
    AND create_time = #{createTime,jdbcType=TIMESTAMP}
</if>
<if test="updateTime != null" >
    AND update_time = #{updateTime,jdbcType=TIMESTAMP}
</if>
<if test="lastLoginTime != null" >
    AND last_login_time =
#{lastLoginTime,jdbcType=TIMESTAMP}
</if>
<if test="isValid != null" >
    AND is_valid = #{isValid,jdbcType=CHAR}
</if>
<if test="costomerStatus != null" >
    AND costomer_status =
#{costomerStatus,jdbcType=TINYINT}
</if>
<if test="remark != null" >
    AND remark = #{remark,jdbcType=VARCHAR}
</if>
</where>
</select>

/**

```



```

    * @see HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response)
        throws ServletException, IOException {
        //1、验证,验证码及token的验证, 防止机器恶意攻击登录
        String validateCode = request.getParameter("validateCode");
        String token = request.getParameter("token");
        if (null == validateCode || validateCode.trim().equals("")) {
            response.sendRedirect(INDEX_PAGE);
            return;
        }
        if (null == token || token.trim().equals("")) {
            response.sendRedirect(INDEX_PAGE);
            return;
        }

        if (!validateCode.equals(request.getSession().getAttribute(token))
        ) {
            response.sendRedirect(INDEX_PAGE);
            return;
        }

        //2、登录信息验证
        String loginName = request.getParameter("loginName");
        String password = request.getParameter("password");
        try {
            CustomerVo customerVo = loginService.login(loginName,
password);
            if (customerVo.isLoginSuccess()) {
                request.getSession().setAttribute("customer", (Customer)
customerVo);
                response.sendRedirect(INDEX_PAGE);
            } else {
                log.info("login failed :" +
customerVo.getLoginMessage());
            }
        } catch (NoSuchAlgorithmException e) {
            log.info("登录发生异常。", e);
        }
    }
}

```

5.3.2.1 首页脚本

```
<!DOCTYPE htm
```

60

```

        <td><a></a></td>
    </tr>
    <tr>
        <td><a></a></td>
    </tr>
    <tr>
        <td><a></a></td>
    </tr>
    <tr>
        <td><a></a></td>
    </tr>
</table>
</td>
<td></td>
</tr>
</table>
</div>

```

<!--首页商品列表-->

```

<#if goodses?has_content>
    <#assign index = 0>
    <#list goodses as listGoodses>
        <#if listGoodses?has_content>
            <#assign index = index + 1>
            <!--楼层栏, height 45px-->
            <div class="floor_flag">
                <table width="100%" cellpadding="0" cellspacing="0">
                    <tr>
                        <td width="30px"></td>
                        <td class="floor_icon">F${index}</td>
                        <td></td>
                        <td></td>
                    </tr>
                </table>
            </div>

            <!--楼层对应商品展示栏, height 300px-->
            <div class="floor_goods_list">

```

```

        <#include
"/webpage/back/goods/inner_goods_list.ftl"/>
    </div>
</#if>
</#list>
</#if>

<#-- 首页搜索列表-->
<#if listGoodses?has_content>
    <#-- 楼层对应商品展示栏, height 400px-->
    <div class="floor_goods_list">
        <#include "/webpage/back/goods/inner_goods_list.ftl"/>
    </div>
</#if>
</body>
</html>

```

5.3.2.2 页面 inner_goods_list.ftl 脚本

```

<#if listGoodses?has_content>
    <ul class="fav-content js-fav-hover clearfix">
        <#list listGoodses as goods>
            <li>
                <input value="${goods.goodsId}" type="hidden">
                <a href="/fruit/goodsDetail.do?goodsId=${goods.goodsId}"
target="_blank" class="fav-content-img">
                    <i class="fav-content-icon
fav-line">${goods.goodsCategoryName}</i>
                    <i class="conmon_icon icon-checkbox"></i>
                    <span class="fav-dele commonFilter">
                        <i class="conmon_icon"></i>
                    </span>
                    
                    </a>
                    <div class="fav-cotent-box">
                        <p class="fav-cotent-txt">
                            <a
href="/fruit/goodsDetail.do?goodsId=${goods.goodsId}" target="_blank"
title="">${goods.goodsName}</a>
                        </p>
                        <p class="fav-tag-box">
                            <i class="fav-icon-bg tag" tip-content="">又11特
价</i>

```

```

        <i class="fav-icon-bg tag" tip-content=" ">免邮费
    </i>

    </p>
    <p class="fav-cotent-date ellipsis"></p>
    <div class="fav-price-box">
        <p
class="fav-price"><em>¥</em>${goods.sellPriceYuan}<span>起</span></p>
    </div>
</div>
</li>
</#list>
</ul>
</#if>

```

5.3.2.3 搜索后台处理

```

@RequestMapping(value="/goodsSearch")
    public ModelAndView goodsSearch(GoodsVo goodsVo) {
        ModelAndView mav = new ModelAndView("/webpage/front/index");

        if(null == goodsVo.getKeyWords() ||
"".equals(goodsVo.getKeyWords().trim())) {
            goodsVo = new GoodsVo();
        }
        mav.addObject("listGoodses",
goodsService.searchByKeyWords(goodsVo));
        mav.addObject("keyWords", goodsVo.getKeyWords());
        return mav;
    }

<select id="search" resultMap="BaseResultMap"
parameterType="com.fruit.model.vo.GoodsVo" >
    select
    <include refid="Base_Column_List" />
    from goods
    where 1 = 2
    <if test="goodsName != null" >
        <if test="isSearchOr">or</if>
        <if test="!isSearchOr">and</if>
        goods_name like concat('%',{goodsName,jdbcType=VARCHAR},'%')
    </if>
    <if test="producingArea != null" >
        <if test="isSearchOr">or</if>
        <if test="!isSearchOr">and</if>
        producing_area like concat('#',{producingArea,jdbcType=VARCHAR},'%')
    </if>

```

```

<if test="description != null" >
    <if test="isSearchOr">or</if>
    <if test="!isSearchOr">and</if>
    description like concat('%',{description,jdbcType=VARCHAR},'%')
</if>
<if test="tag != null" >
    <if test="isSearchOr">or</if>
    <if test="!isSearchOr">and</if>
    tag like concat('%',{tag,jdbcType=VARCHAR},'%')
</if>
<if test="goodsTypeId != null" >
    <if test="isSearchOr">or</if>
    <if test="!isSearchOr">and</if>
    goods_type_id = #{goodsTypeId,jdbcType=BIGINT}
</if>
<if test="goodsCategory != null" >
    <if test="isSearchOr">or</if>
    <if test="!isSearchOr">and</if>
    goods_category = #{goodsCategory,jdbcType=TINYINT}
</if>
<if test="goodsStatus != null" >
    <if test="isSearchOr">or</if>
    <if test="!isSearchOr">and</if>
    goods_status like concat("#{goodsStatus,jdbcType=TINYINT},'%')
</if>

<if test="startRow != null" >
    limit #{startRow,jdbcType=BIGINT}, #{pageSize,jdbcType=BIGINT}
</if>
</select>

```

5.3.3 购物车列表模块处理设计

5.3.3.1 前台视图解析处理

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>小水果-我的购物车</title>
        <#include "/webpage/front/comm/fruit_meta.ftl"/>
    </head>
    <body>
        <#include "/webpage/front/comm/fruit_header.ftl"/>

```

```

<!--购物车中商品列表-->
<div class="my_shp_cart_div">
    <#if myShpCards?has_content>
        <table cellpadding="0" cellspacing="0" border="0">
            <tr>
                <th>序号</th>
                <th colspan="2">商品信息</th>
                <th>单价</th>
                <th>购买数量</th>
                <th>合计</th>
                <th>操作</th>
            </tr>
            <#assign index = 0/>
            <#list myShpCards as shpCardGoods>
                <#assign index = index + 1/>
                <tr>
                    <td align="center">
                        <input name="shpCartGoodsCheck"

value="${shpCardGoods.shoppingCartId}:${shpCardGoods.goodsId}:${s
hpCardGoods.quantity}"

                                type="checkbox" />${index}
                    </td>
                    <td align="right">
                        <a target="_blank"
href="/fruit/goodsDetail.do?goodsId=${shpCardGoods.goodsId}">
                            
                        </a>
                    </td>
                    <td
align="left">${shpCardGoods.goodsName}</td>
                    <td
align="center">${shpCardGoods.sellPriceYuan}</td>
                    <td
align="center">${shpCardGoods.quantity}</td>
                    <td
align="center">${shpCardGoods.sellPriceYuan*shpCardGoods.quantity}</t
d>
                    <td align="center">
                        <a
href="/fruit/shoppingCart/removeCartGoodsById.do?shoppingCartId=${shp
CardGoods.shoppingCartId}">从购物车中移出</a>
                    </td>
                </tr>
            </#list>
        </table>
    </div>

```

```
        </tr>
    </#list>
</table>
<div class="my_buy_div">
    <div class="my_buy_btn" onclick="toBuy()">
        <table cellspacing="0" cellpadding="0"
border="0">
            <tr>
                <td align="center" valign="middle">
                    <span>去结算</span>
                </td>
            </tr>
        </table>
    </div>
</div>
</#if>
</div>

<script>
    function toBuy(){
        var boxs =
document.getElementsByName("shpCartGoodsCheck");
        var buyGoodses = [];
        for(var i = 0; i < boxs.length; i++){
            var box = boxs[i];
            if(box.checked){
                buyGoodses.push(box.value);
            }
        }
        var buyForm = document.createElement("FORM");
        var input = document.createElement("INPUT");
        buyForm.method = "POST";
        buyForm.style.display = "none"
        buyForm.action = "/fruit/order/saveOrder.do";
        input.type = "hidden";
        input.name = "shpGoods";
        input.value = buyGoodses.join(",");
        buyForm.appendChild(input);
        document.body.appendChild(buyForm);
        buyForm.submit();
    }
</script>
</body>
</html>
```


5.3.3.2 后台逻辑处理

```

/**
 *
 * @return
 */
@RequestMapping(value="myShoppingCart")
public ModelAndView listShpCartGoods(HttpServletRequest request){
    ModelAndView mav = new
ModelAndView("/webpage/front/shpCart/myShpCartView");
    ShoppingCart shoppingCart = new ShoppingCart();

    Customer customer = (Customer)
request.getSession().getAttribute("customer");
    shoppingCart.setCustomerId(customer.getCustomerId());
    shoppingCart.setStatus(SHOPPING_CART_STATUS.AADED.getCode());

    List<ShoppingCartVo> shpCardGoods =
shoppingCartService.queryShpCartByParam(shoppingCart);
    if(null != shpCardGoods){
        mav.addObject("myShpCards", shpCardGoods);
        short quantity = 0;
        float totalMoney = 0;
        //计算总数量和总钱数
        for (ShoppingCartVo shoppingCartVo : shpCardGoods) {
            quantity += shoppingCartVo.getQuantity();
            totalMoney += shoppingCartVo.getSellPriceYuan() *
shoppingCartVo.getQuantity();
        }
        mav.addObject("totalMoney", totalMoney);
        mav.addObject("totalCount", quantity);
    }
    return mav;
}

```

5.3.4 采购管理模块处理设计

5.3.4.1 Controller 层处理

```

/**
 * 提交采购单
 * @param request
 * @return
 */
@RequestMapping("/doPurchase")

```

```

    public ModelAndView doPurchase(PurchaseOrderVo purchaseOrderVo,
    HttpServletRequest request){
        ModelAndView mav = new
    ModelAndView("/webpage/back/purchase/purchaseManage");
        //1、构建采购单对象
        String detailJson =
    purchaseOrderVo.getPurchaseOrderDetailsJson();
        JSONArray detailJsonArray = JSONArray.fromObject(detailJson);
        List<PurchaseOrderDetailVo> purchaseOrderDetails =
    (List<PurchaseOrderDetailVo>) JSONArray.toCollection(detailJsonArray,
    PurchaseOrderDetailVo.class);
        purchaseOrderVo.setPurchaseOrderDetails(purchaseOrderDetails);

        //2、获取用户（采购员）信息
        EmployeeVo employeeVo =
    (EmployeeVo) request.getSession().getAttribute("employee");
        purchaseOrderVo.setEmployeeId(employeeVo.getEmployeeId());

        //3、保存采购单
        purchaseOrderService.savePurchaseOrder(purchaseOrderVo);

        //4、查询采购单位
        getPurchaseDetail(mav);

        return mav;
    }

    /**
     * 查询采购单及采购明细
     * @param mav
     */
    private void getPurchaseDetail(ModelAndView mav){
        Map<String, Object> params = new HashMap<>();
        params.put("startRow", 0);
        params.put("pageSize", 5);
        List<PurchaseOrderVo> purchases =
    purchaseOrderService.querySelective(params);
        for (PurchaseOrderVo purchaseOrderVo : purchases) {
            Map<String, Object> detailParams = new HashMap<>();
            detailParams.put("purchaseOrderId",
    purchaseOrderVo.getPurchaseOrderId());
            List<PurchaseOrderDetailVo> details =
    purchaseOrderDetailService.querySelective(detailParams);
            purchaseOrderVo.setPurchaseOrderDetails(details);
        }
    }

```

```

    }
    mav.addObject("purchaseOrders", purchases);
}

```

5.3.4.2 Service 层处理

- 1、保存采购单，并返回采购单号
- 2、循环将采购单号 set 到采购明细对象中
- 3、循环保存采购明细对象

```

/**
 * 创建采购单
 * @param purchaseOrderVo
 */
public void savePurchaseOrder(PurchaseOrderVo purchaseOrderVo) {
    if(null != purchaseOrderVo) {
        purchaseOrderMapper.insertSelective(purchaseOrderVo);
        if(null != purchaseOrderVo.getPurchaseOrderDetails() &&
            purchaseOrderVo.getPurchaseOrderDetails().size() >
0) {
            for (PurchaseOrderDetailVo purchaseOrderDetailVo :
purchaseOrderVo.getPurchaseOrderDetails()) {
                purchaseOrderDetailVo.setPurchaseOrderId(purchaseOrderVo.getPurch
aseOrderId());
                purchaseOrderDetailMapper.insertSelective(purchaseOrderDetailVo);
            }
        }
    }
}

```

5.3.5 出库管理模块处理设计

```

/**
 * 商品出库
 * @return
 */
@RequestMapping("/goodsOutStore")
public ModelAndView goodsOutStore(GoodsInventoryVo
goodsInventoryVo) {
    ModelAndView mav = new
ModelAndView("/webpage/back/storage/inStore");

    //出库操作
    Map<String, Object> params = new HashMap<>();
    params.put("goodsId", goodsInventoryVo.getGoodsId());
    List<GoodsInventoryVo> goodesInventories =
goodsInventoryService.querySelective(params);
    if(null != goodesInventories && goodesInventories.size() > 0) {

```

```

        GoodsInventoryVo vo = goodsInventories.get(0);
        short newQuantity = (short) (vo.getQuantity().shortValue() -
goodsInventoryVo.getQuantity().shortValue());
        if(newQuantity > 0){
            vo.setQuantity(newQuantity);
            goodsInventoryService.updateByPrimaryKeySelective(vo);
        }
    }

    //修改订单出库状态
    FruitOrderDetailVo fruitOrderDetailVo = new
FruitOrderDetailVo();

    fruitOrderDetailVo.setOrderDetailId(goodsInventoryVo.getOrderDetailId());

    fruitOrderDetailVo.setStatus(ORDER_OUT_STORE_STATUS.OUT_STORED.getCode());

    fruitOrderDetailService.updateByPrimaryKeySelective(fruitOrderDetailVo);

    //查询订单
    mav.addObject("purchaseOrders", mav.addObject("outStoreOrders",
getFruitOrder()));
    return mav;
}

```

5.3.6 入库管理模块处理设计

```

/**
 * 商品入库
 * @return
 */
@RequestMapping("/goodsInStore")
public ModelAndView goodsInStore(GoodsInventoryVo
goodsInventoryVo){
    ModelAndView mav = new
ModelAndView("/webpage/back/storage/inStore");
    goodsInventoryVo.setGoodsBatchNo(Long.parseLong(IDManager.getTime
Stamp(8))); //批次号生成, 时间戳yyyyMMdd
    Date curDate = new Date();
    curDate.setTime(System.currentTimeMillis() + 1000 * 60 * 60 * 24
* 7);
    goodsInventoryVo.setExpiredTime(curDate);
}

```

```

        goodsInventoryVo.setSellingQuantity(goodsInventoryVo.getQuantity(
    ));
    //1/入库操作
    Map<String, Object> params = new HashMap<>();
    params.put("repostoryId", goodsInventoryVo.getRepostoryId());
    params.put("goodsId", goodsInventoryVo.getGoodsId());
    List<GoodsInventoryVo> goodesInventories =
goodsInventoryService.querySelective(params);
    if(null != goodesInventories && goodesInventories.size() > 0){
        GoodsInventoryVo vo = goodesInventories.get(0);
        short newQuantity = (short)
(goodsInventoryVo.getQuantity().shortValue() +
vo.getQuantity().shortValue());
        short newCanSellQuantity = (short)
(goodsInventoryVo.getQuantity().shortValue() +
vo.getSellingQuantity().shortValue());
        vo.setQuantity(newQuantity);
        vo.setSellingQuantity(newCanSellQuantity);
        vo.setGoodsBatchNo(goodsInventoryVo.getGoodsBatchNo());
        vo.setExpiredTime(goodsInventoryVo.getExpiredTime());
        goodsInventoryService.updateByPrimaryKeySelective(vo);
    }else{
        goodsInventoryService.insertSelective(goodsInventoryVo);
    }
    //2/修改入库状态
    PurchaseOrderDetailVo purchaseOrderDetailVo = new
PurchaseOrderDetailVo();
    purchaseOrderDetailVo.setPurchaseOrderDetailId(goodsInventoryVo.g
etPurchaseOrderDetailId());
    purchaseOrderDetailVo.setStatus(PURCHASE_STATUS.SUPPLIED.getCode(
));
    purchaseOrderDetailService.updateByPrimaryKeySelective(purchaseOr
derDetailVo);
    //3、查询采购单
    mav.addObject("purchaseOrders", getPurchaseOrder());
    return mav;
}

```

5.4 数据库配置

5.4.1 数据库名称、软件环境

数据库名：fruit

软件环境：jdk1.7、mysql dbms5.6、tomcat8

5.4.2 建立 ODBC 或 JDBC 等

采用 mybatis 配置数据源来简历 JDBC 数据库连接池。数据库连接池配置如下

#小水果mysql库数据源配置

```
fruit.mysql.driverClassName=com.mysql.jdbc.Driver
fruit.mysql.maxActive=10
fruit.mysql.maxIdle=1
fruit.mysql.maxWait=10000
fruit.mysql.initSize=1
fruit.mysql.username=root
fruit.mysql.password=123456
fruit.mysql.url=jdbc:mysql://localhost:3306/fruit?useUnicode=true
&characterEncoding=UTF-8&allowMultiQueries=true
```

5.4.3 数据库连接

数据库连接采用连接池的模式，这样可以提高数据库连接的利用率，不用每次操作数据库都去创建一个数据库连接，降低了操作数据库的响应时间和数据库的压力。

数据库连接的 url 为：

```
jdbc:mysql://localhost:3306/fruit?useUnicode=true&characterEn
coding=UTF-8&allowMultiQueries=true
```

5.4.4 数据库恢复

MySql 数据库的线上配置为 Master slave 模式。Master 的数据实时同步至 slave 中。当 master 失效或故障导致数据丢失，可以将 slave 的数据恢复到 master 中去。写操作在 master，读操作除实时性要求特别高的连接外，从 slave 进行数据库的查询，减轻 master 的负载。

第六章 实施概况

6.1 实施环境与工具

6.1.1 计算机系统平台

Windows server 2000/2003/2008 或 windows xp 或 window7/8 或 linux(CentOS,ubuntu, redhat,mondrivar 等)。

6.1.2 编程环境与工具

硬件环境：PC 机/机柜上服务器

软件环境：操作系统：windows/linux 服务器组件：tomcat/jboss/weblogic 数据库软件：数据库管理系统，此系统以 mysql 作为数据库管理系统

IDE：Eclipse3.2 for java ee

DBMS：Mysql 5.6.18。

数据库查询分析器：Sql yog。

文本编辑器：Notepad++。

6.1.3 数据准备

ID	商品名称	产地	单价	描述			时间
6	Zespri 佳沛新西兰有机绿奇 异果 6 个 82-105g 个	新西兰	355000	进口新 鲜水果	1	1	25:05.0
7	台湾麻豆文旦柚 4 个 320g 以上个	台湾	750000	进口	1	1	26:07.0
8	台湾黑珍珠莲雾 500g	台湾	380000	新鲜	1	1	26:45.0
9	哈尼福建蜜柚双拼(红心+白 心)礼盒装约 1.2kg 个	福建	580000	新鲜	1	0	27:27.0
10	哈尼福建蜜柚双拼(红心+白 心)礼盒装约 1.2kg 个	福建	580000	新鲜	1	1	28:16.0
11	山东南水梨 4 个 380g 以上 个	山东	450000	新鲜	1	1	29:17.0
12	山东沾化冬枣 2kg10g 以上 个	山东	390000	新鲜	1	1	30:03.0
13	山东蓬莱精品红富士 2.5kg 果径 80-85mm	山东蓬 莱	560000	新鲜	1	1	30:49.0
14	山东金秋红蜜桃 4 个 180g 以上个	山东	330000	新鲜	1	1	31:48.0
15	常瀛国产呀咪香蕉 10 根切	常瀛	150000	新鲜	1	1	32:41.0

	割蕉						
16	广东麒麟瓜 2 个 2.5kg 以上 个	广东	680000	新鲜	1	1	33:31.0
17	广西小青芒 2.5kg 约 150g 个	广西	460000	新鲜	1	1	34:39.0
18	广西脆柿 1kg 约 120g 个	广西	210000	新鲜	1	1	35:15.0
19	广西高乐蜜甜芒果 1.5kg200g 个以上	广西	240000	新鲜	1	1	35:57.0
20	广西黄金百香果 4 个	广西	200000	新鲜	1	1	36:58.0
21	新疆库尔勒香梨 3kg100g 以 上个	新疆	520000	新鲜	1	1	37:48.0
22	新疆王林苹果 1kg 果径 70mm 以上	新疆	290000	新鲜	1	1	38:30.0
23	新疆红提 2kg	新疆	420000	新鲜	1	1	39:35.0
24	智利红樱桃 2kg 果径 26-28mm	智利	2380000	进口	1	1	40:53.0
25	智利青苹果 12 个约 135g 个	智利	620000	新鲜	1	1	41:35.0
26	河北皇冠梨 4 个约 200g-300g 个	河北	118000	新鲜	1	1	42:18.0
27	河南荥阳突尼斯软籽石榴 4 个 300g 个以上	河南	524000	新鲜	1	1	43:07.0
28	泰国精选椰青 4 个	泰国	630000	进口	1	1	43:51.0
29	海南树上熟木瓜 5kg (约	海南	880000	新鲜	1	1	44:32.0

	450g 个)						
30	海南西州蜜瓜 1 个 1.5kg 以上个	海南	260000	新鲜	1	1	45:11.0
31	海南金钻凤梨 2 个约 1kg 个	海南	660000	新鲜	1	1	46:14.0
32	甘肃黄蕉苹果 3kg170g 个 , +-10g	甘肃	680000	新鲜	1	1	46:50.0
33	福建平和黄心蜜柚 1 个约 1.1kg 个	福建	220000	新鲜	1	1	47:30.0
34	美国 Moon Drops 无籽黑提 2kg	美国	1650000	进口	1	1	48:12.0
35	美国红啤梨 12 个约 210g 个	美国	1260000	新鲜	1	1	48:51.0
36	美国红蛇果 8 个装(约 190g 个)	美国	330000	新鲜	1	1	49:22.0
37	美国青啤梨 12 个约 210g 个	美国	850000	进口	1	1	49:56.0
38	越南红心火龙果 1kg(大果) 约 450g 个	越南	390000	进口	1	1	50:37.0
39	陕西徐香猕猴桃 16 个 100-120g 个	陕西	230000	新鲜	1	0	51:13.0
40	哈尼福建蜜柚双拼(红心+白心)礼盒装约 5.2kg 个	福建	1820000	新鲜	1	1	53:06.0
41	台湾麻豆文旦柚 8 个 320g 以上个	台湾	970000	新鲜	1	1	54:09.0
42	福建平和黄心蜜柚 4 个约	福建	560000	新鲜	1	1	55:08.0

	1.1kg 个						
43	台湾麻豆文旦柚 6 个 320g 以上个	台湾	730000	新鲜	1	1	56:24.0
44	广东麒麟瓜 5 个 2.5kg 以上 个	广东	1260000	新鲜	1	1	57:20.0
45	海南树上熟木瓜 9kg (约 450g 个)	海南	1500000	新鲜	1	1	58:08.0
46	海南西州蜜瓜 1 个 2.5kg 以 上个	海南	835000	新鲜	1	1	59:05.0
47	广东麒麟瓜 2 个 3.5kg 以上 个	广东	965000	新鲜	1	1	00:05.0
48	美国红提 4kg	美国	1010000	进口	1	1	01:21.0
49	美国 Moon Drops 无籽黑提 4kg	美国	3520000	进口	1	1	01:57.0
50	美国 Moon Drops 无籽黑提 6kg	美国	4430000	新鲜	1	1	03:19.0
51	美国 Moon Drops 无籽黑提 6kg	美国	4430000	新鲜	1	1	29:10.0

编号	用户名	密码	性别	生日	时间
1	danny	fVssTooHg3DOWWMS2GLbMw==			24:43.0
2	huoqiang	6ZOYA5IZKsHk5VP497TV7g==			53:36.0
4	tomy	5zlZOexnty95EpKBfnRelw==			42:21.0
5	张小五	v21sgZ7JdbBDrsUCFnw9FQ==			12:47.0

6	霍强	7t3SinWDMikva/wSI4ZStA==			16:37.0
---	----	--------------------------	--	--	---------

以上为商品和用户，用户登录系统后，可查看该系统中售卖的商品，可对商品进行加入购物车和从购物车中去结算购买操作。

6.2 系统测试

6.2.1 测试规程

单元测试：单元测试是确保系统正常运行的基础，本系统主要采用的是 Junit 单元测试，编写 dao 层及 service 层的单元测试用例，依次去执行每个个 dao 及 service 层的单个方法，确保每个个基础函数都运行正常，返回正确结果。

集成测试：集成测试是在基础的单元测试通过后，开始将各种模块结合起来进行整体性的测试。本系统的集成测试是 CRM 模块、商品管理模块、储运管理模块等模块进行整体性的测试，确保每个模块的正确性。

功能测试：功能测试是测试系统的业务流程，确保每个业务流程的正确性。如本系统功能测试流程如下：

- 1、上架商品，系统录入 6.1.3 准备的数据至商品表。
- 2、制作做采购，制作采购单，每件商品采购 100 份，采购的仓库为上海 1 号仓库。
- 3、采购入库，将采购的商品进行入库操作，此时上海 1 号库中的每件商品的可售数量和库存均为 100。
- 4、制作销售订单。此时消费者角色的用户进行登录，然后选择商品将其加入至购物车中，然后点击购物车列表，选择购物车中的商品去结算，点击提交订单(默认为已支付，由于支付需第三方系统交互，所以此系统默认已经做了支付操作。)
- 5、查看库存，当前购买的产品的可售库存数量是否减少至原数量减去购买数量。
- 6、接单员选择订单进行接单处理
- 7、仓管员角色登录，然后进入出库管理界面，选择商品进行出库操作。
- 8、查看库存，当前出库的产品的数量是否减少了库存量。

6.2.2 测试计划及测试记录

6.2.2.1 测试计划

- 1、单元测试。
- 2、集成测试。
- 3、功能测试。

6.2.2.2 测试记录

- 1、将台湾黑珍珠莲雾 500g 加入 1 份到购物车

序号	商品信息	单价	购买数量	合计	操作
1	 台湾黑珍珠莲雾500g	38	1	38	从购物车中移出
去结算					

2、点击去结算并选择收货地址

序号	商品信息	应付款
1	 台湾黑珍珠莲雾500g	3800
公司 崔先生 上海上海普陀中江路302弄68号 新增收货地址		
提交订单		

3、点击提交订单

13查看订单详情

恭喜您预订成功，请您牢记订单号13

4、查看库存，可售库存量减少为 299

	goods_id	goods_name	quantity	selling_quantity	repostory_id
1	8	台湾黑珍珠莲雾500g	300	299	1

5、进入到出库管理界面并点击出库

序号	订单号	创建时间	订单状态	发货地址	操作
1	12时间		待支付	霍先生 上海 上海 普陀 中江路302弄68号	全部出库
商品编号：8		数量：1		状态：待出库	出库

6、出库看状态查看，变为已出库

序号	订单号	创建时间	订单状态	发货地址	操作
1	12时间		待支付	霍先生 上海 上海 普陀 中江路302弄68号	全部出库
商品编号：8		数量：1		状态：已出库	

7、出库后查看库存[和前面对比，库存 quntity 应变为 299]

	goods_id	goods_name	quantity	selling_quantity	repostory_id
1	8	台湾黑珍珠莲雾500g	299	299	1

8、切换至采购员登录，采购台湾黑珍珠莲雾 500g,10 份

采购名称：采购台湾黑珍珠莲雾500G 10份

供应商采购：陕西红富士果园

商品明细列表：

1:台湾黑珍珠莲雾500g,10份

商品：Zespri佳沛新西兰有机绿奇异果6个82-1

采购数量：10 [添加明细](#)

[提交采购](#)

9、提交采购后，查看采购单位

5	6 采购台湾黑珍珠莲雾500G 10份	2016-11-25 10:31:17	
商品：哈尼福建蜜柚双拼(红心+白心)礼盒装约1.2kg个	采购数量：10	供货状态：待供货	

10、切换至仓管员登录，进入到采购入库管理界面，并将刚采购的商品入库

5	6 采购台湾黑珍珠莲雾500G 10份	2016-11-25 10:31:17	全部入库
商品：哈尼福建蜜柚双拼(红心+白心)礼盒装约1.2kg个	采购数量：10	供货状态：待供货	入库

11、入库后的入库管理中的此采购商品的状态变为已供货

5	6 采购台湾黑珍珠莲雾500G 10份	2016-11-25 10:31:17	全部入库
商品：哈尼福建蜜柚双拼(红心+白心)礼盒装约1.2kg个	采购数量：10	供货状态：已供货	

12、查看库存[可售库存量和库存量均变为 309 正确]

	goods_id	goods_name	quantity	selling_quantity	repostory_id
1	8	台湾黑珍珠莲雾500g	309	309	1

6.3 系统转换方案

6.3.1 老系统迁移新系统方案

1、事先准备好数据迁移同步脚本，本系统采用 ogg 将老系统表中的数据同步至新系统。用 kettle 定时将新系统中的增量数据同步回老系统，防止新系统因不可挽回性故障能安

全平稳的切换回原系统中。

- 2、在新系统数据库服务器上，初始化数据库及相关表。

- 3、转换前将老系统中的数据同步至新系统中。

- 4、在凌晨系统访问量低谷时段开始系统转换工作。将应用程序服务停止，将数据库服务停止。

- 5、用 ogg 将最后一次同步至停机前的增量数据同步至新系统中

- 6、启动新系统中的应用服务，启动新系统的数据库服务

- 7、在新系统中线上验证各项功能是否正常

- 8、启动 kettle 定时同步数据 job,选择每天凌晨 2 点执行，同步昨天的数据。

以上老系统迁新系统工作完成。

6.3.2 新系统回退至老系统方案

若新系统故障，切换回老系统方案如下：

- 1、将新系统应用服务停止，数据库服务停止。

- 2、用 kettle 同步最后一次的增量数据至新系统

- 3、数据同步完成后，启动老系统服务

- 4、功能验证及数据完整性验证。

以上新系统回退至老系统工作完成。

6.3.3 废弃老系统

若老系统切换至新系统工作完成，且线上验证稳定，则待新系统运行与老系统并行运行一段时间后，方可将老系统彻底废弃。6.3.2 为老系统迁移新系统故障回退方案，迁移顺利，此方案不会用到。

至此整个系统转换结束。

6.4 系统运行与维护概况

本系统由 java 语言编写，采用的是 B/S 架构的。服务端可运行在 linux 服务器(推荐)上，也可运行在 window 操作系统中。访问则在装有浏览器，可与此服务器进行连网的机器上均可。

数据库采用 mysql，可与应用程序服务器装机在同一服务器上，也可装在能与应用程序服务器进行连网(最好是同一个内网中)的服务器上。

系统维护主要由小水果公司的系统运维人员来负责，主要包括新功能的上线发布，系统配置的增删改，线上服务器故障的处理，数据库服务的监测，数据库磁盘的监测，数据库拓展问题等。

第七章 总结与展望

7.1 总结

在今年下半年，我开始了论文的编写。此次的论文，采用 java+mysql 数据库来设计和

实现人，采用 java 的三层架构（Spring+SpringMvc+MyBatis 框架）。主要实现的是一个小的电商系统，网上售卖水果这样一个系统。包括了上下架商品、库存的管理、出库的管理，采购的管理、入库的管理及消费者网上选货添加购物车及购买这样一个过程。

从最初于老师的初次辅导，每隔一个周末周末的辛苦辅导加上平时的邮件交流，每一次老师面对面的辛勤辅导，用心回复的每一封邮件，都让我铭记在心，对于老师的指导，不管多忙，都根据自己的理解，再次做修改。

在老师的指导这个过程中，让我学会了很多。专业理论知识更进一步的得到了巩固，对于细节方面能更好的去把控。独立搭建了基于 java 语言和 mysql 数据库的 demo，实现了论文编写的功能，让以前团队做项目的每一个过程又得到了一次锻炼。

7.2 展望

回顾自己，已从事软件研发工作四年月余，从最开始的“小菜鸟”，成长成为现在的“大菜鸟”，先后从事了数字电视软件、互联网电商、大数据、用户数据中心等系统的研发工作。本次论文我采用的是自己熟悉的电商系统，网上水果店（小水果信息管理系统），一是作为论文来用心去做，二是打算自己独立完成一个从需求分析、数据库设计、系统架构设计到最终实现这样一个过程，三是想把这个系统在后续的工作中去使用，用于我自己对电商系统的研究及实践，为后续的电商产品的用户精准推荐，用户画像大数据，系统风险控制等方面的研究及实践。

本系统离商业性质的电商系统还缺少很多，还需要很多进一步的优化和完善，比如商品的评价功能、用户追踪所购买商品的发货后的物流信息，对用户进行产品的精准化推荐、系统的风险控制、优惠券的发放等功能。精准推荐和系统风险控制是后续的一个研究方向，跟我目前所从事的工作比较契合。

7.3 回顾

论文从最开始的初次辅导到后续的每一次邮件交流，再到每一次面对面的辅导工作，经历了这四个来月的学习、交流、修改、优化、实现等，每一次的过程都是一个自我学习、与人交流学习及自我锻炼的一个过程，真正让我感受到了最初的坚持是对的。

记得那是刚来上海工作，之前都听说过自考很难，考复旦更难，最开始想抱着试一试的心态，我开始报考了复旦大学的自考计算机软件专业。结果 13 年四月初次考试之前，我开始准备了两个多月的复习，天天下班看书，周末去图书馆。结果功夫不负有心人，报了 6 科，一次通过的有 5 科，让我有了更加坚定的信息，从最开始的试一试的心态，到不想放弃，到再坚持坚持，终于于今年 6 月通过了全部的课程的考试。

这个过程锻炼了我的学习能力，同时也培养了我的毅力，用心去对待自己所选择的每一件事情。

参考文献

- | | | | |
|---------------|------------------|---------------|---------|
| 软件工程 | 王立福著 | 9787111338123 | 机械工业出版社 |
| 数据库系统原理 | 丁宝康著 | 9787805861220 | 经济科学出版社 |
| UML 面向对象建模与设计 | (美) 巴拉赫, (美) 兰宝著 | | 人民邮电出版社 |

致谢

从自考的每一次考试、到论文的初次辅导到答辩前的每一个细小的过程中，最想感谢的是老师的辛勤指导和培养，感谢复旦大学提供的这样一个好的平台。感谢自己的每一次细小的坚持，感谢于老师的每一次用心的邮件回复、感谢指导老师的每一次面对面的辛勤指导、感谢一同参加自考及论文的小伙伴们，在这个过程中我们在共同学习，共同提高，在这个过程中让我又有了一次整体自我提升的过程。

附录

```

-----
--tab1 Table structure for `customer`
-----

DROP TABLE IF EXISTS `customer`;
CREATE TABLE `customer` (
  `customer_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `customer_no` char(32) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
  COMMENT '用户 32 位 16 进制 ID',
  `login_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '登录名',
  `password` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '密码',
  `safety_factor` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '安全因子，用于密码安全',
  `real_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '真实姓名',
  `nick_name` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '昵称',
  `mobile_number` varchar(15) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '手机号',
  `email` varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT 'EMAIL',
  `gender` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '性别',
  `qq` varchar(15) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT 'qq 号码',
  `wechat` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '微信号码',
  `birthday` datetime NULL DEFAULT NULL COMMENT '生日',
  `image_url` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '头像 URL',
  `vip_level` tinyint(2) NOT NULL DEFAULT 0 COMMENT 'vip 等级，刚注册为 0 级',
  `create_time` datetime NULL DEFAULT NOW() COMMENT '创建时间',

```



```

`update_time` datetime NULL DEFAULT NULL COMMENT '更新时间',
`last_login_time` datetime NULL DEFAULT NULL COMMENT '最近一次登录时间',
`is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'Y'
COMMENT '是否有效, Y/N',
`costomer_status` tinyint(2) NOT NULL DEFAULT 1 COMMENT '用户当前状态 1:正常
2: 已冻结 3:已注销',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`customer_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='消费者'
;
--创建索引
CREATE INDEX `IDX_CUSTOMER_NO` ON `customer`(`customer_no`) USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_CREATE_TIME` ON `customer`(`create_time`) USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_EMAIL` ON `customer`(`email`) USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_LOGIN_NAME` ON `customer`(`login_name`) USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_MOBILE_NUMBER` ON `customer`(`mobile_number`)
USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_QQ` ON `customer`(`qq`) USING BTREE ;
CREATE INDEX `IDX_CUSTOMER_WECHAT` ON `customer`(`wechat`) USING BTREE ;

-----
--tab2 Table structure for `delivery_address`
-----
DROP TABLE IF EXISTS `delivery_address`;
CREATE TABLE `delivery_address` (
  `address_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `address_alias` char(32) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '地址别名',
  `customer_id` bigint(11) NOT NULL COMMENT '所属消费者 ID',
  `mobile_number` varchar(15) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '手机号码',
  `areacode` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '区号',
  `phone_number` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '电话号码',
  `consignee` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '收货人',
  `province` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '省',
  `city` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL

```

```

COMMENT '市',
`county` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '区/县',
`address` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '地址',
`postcode` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '邮编',
`is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'Y'
COMMENT '是否有效,Y/N',
`create_time` datetime NULL DEFAULT now() COMMENT '创建时间',
`update_time` datetime NULL DEFAULT NULL COMMENT '最近修改时间',
PRIMARY KEY (`address_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='收货地址'
;

```

```

CREATE INDEX `IDX_ADDRESS_CUSTOMER_ID` ON `delivery_address`(`customer_id`) USING
BTREE ;
CREATE INDEX `IDX_ADDRESS_NO` ON `delivery_address`(`address_alias`) USING BTREE ;
CREATE INDEX `IDX_ADDRESS_CREATE_TIME` ON `delivery_address`(`create_time`) USING
BTREE ;
CREATE INDEX `IDX_ADDRESS_UPDATE_TIME` ON `delivery_address`(`update_time`) USING
BTREE ;

```

```

-----
--tab3 Table structure for `dept`
-----

```

```

DROP TABLE IF EXISTS `dept`;
CREATE TABLE `dept` (
`dept_id` smallint(3) NOT NULL AUTO_INCREMENT COMMENT 'PK,部门编号',
`dept_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '部门名称',
`establish_date` DATETIME NULL COMMENT '创立日期',
`is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'Y'
COMMENT '是否有效,Y/N',
`remark` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`dept_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='部门'

```

```

;

-----
--tab4 Table structure for `employee`
-----

DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `employee_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK,工号',
  `login_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '登录名',
  `password` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '密码',
  `real_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '真实姓名',
  `gender` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '性别',
  `birthday` date NULL DEFAULT NULL COMMENT '生日',
  `dept` smallint(3) NOT NULL COMMENT '部门',
  `position` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '职位',
  `rank` char(3) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '职级 t3,
  t4, t5, t6, t7, t8, p2, p3, p4, p5, p6,o1, o2, o3, o4',
  `monthly_pay` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '月薪',
  `hired_date` date NULL DEFAULT NULL COMMENT '雇佣日期',
  `hire_year` smallint(2) NULL DEFAULT NULL COMMENT '雇佣时限,年',
  `employee_status` tinyint(2) NOT NULL DEFAULT 0 COMMENT '员工状态 => 0 : 未入职, 1 :
  实习生, 2 : 试用期, 3 : 已转正, 4 : 离职, 5 : 被解雇',
  `role` tinyint(2) NULL COMMENT '角色 : 0 : 管理员, 1 : 运营专员, 2 : 接单员, 3 : 仓管员,
  4 : 采购员',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'Y'
  COMMENT '是否有效,Y/N',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注',
  PRIMARY KEY (`employee_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='员工'
;

--创建索引
CREATE INDEX `IDX_EMPLOYEE_LOGIN_NAME` ON `employee`(`login_name`) USING BTREE ;

-----

```

--tab5 Table structure for `menu_authority`

```
DROP TABLE IF EXISTS `menu_authority`;
CREATE TABLE `menu_authority` (
  `menu_authority_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK,菜单权限ID',
  `back_menu_id` bigint(11) NOT NULL COMMENT 'PK,菜单ID',
  `role` tinyint(2) NULL COMMENT '角色：0：管理员，1：运营专员，2：接单员，3：仓管员，4：采购员',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL default 'Y' COMMENT '是否有效，Y：是，N：否',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`menu_authority_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='菜单权限';
```

--创建索引

```
CREATE INDEX `IDX_MENU_AUTHORITY_BACK_MENU_ID` ON
`menu_authority`(`back_menu_id`) USING BTREE;
CREATE INDEX `IDX_MENU_AUTHORITY_ROLE` ON `menu_authority`(`role`) USING BTREE;
```

--tab6 Table structure for `back_menu`

```
DROP TABLE IF EXISTS `back_menu`;
CREATE TABLE `back_menu` (
  `back_menu_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK,菜单ID',
  `menu_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '菜单名',
  `menu_url` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '菜单地址',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `is_valid` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL default 'Y' COMMENT '是否有效，Y：是，N：否',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`back_menu_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
```

```

COMMENT='菜单权限'
;
--创建索引
CREATE INDEX `IDX_BACK_MENU_NAME` ON `back_menu`(`menu_name`) USING BTREE ;
CREATE INDEX `IDX_BACK_MENU_URL` ON `back_menu`(`menu_url`) USING BTREE ;

-----
--tab7 Table structure for `goods_type`
-----
DROP TABLE IF EXISTS `goods_type`;
CREATE TABLE `goods_type` (
  `goods_type_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK,商品 ID',
  `goods_type_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '商品类型名称, 苹果、橘子、柚子。。。',
  `goods_category` tinyint(2) NULL DEFAULT NULL COMMENT '商品分类, 水果分的大类别',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注',
  PRIMARY KEY (`goods_type_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='商品类型'
;
--创建索引
CREATE INDEX `IDX_GOODS_TYPE_TYPE_NAME` ON `goods_type`(`goods_type_name`)
USING BTREE ;
CREATE INDEX `IDX_GOODS_TYPE_GOODS_CATEGORY` ON `goods_type`(`goods_category`)
USING BTREE ;
CREATE INDEX `IDX_GOODS_TYPE_CREATE_TIME` ON `goods_type`(`create_time`) USING
BTREE ;

-----
--tab8 Table structure for `outbound_order_detail`
-----
DROP TABLE IF EXISTS `outbound_order_detail`;
CREATE TABLE `outbound_order_detail` (
  `outbound_order_detail_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '出库明
  细编号',
  `outbound_order_id` bigint(11) NOT NULL COMMENT 'FK、出库单编号',
  `order_detail_id` bigint(11) NOT NULL COMMENT 'FK、订单明细编号',
  `goods_id` bigint(11) NOT NULL COMMENT 'FK、商品编号',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `outbound_time` datetime null default null COMMENT '出库时间',

```

```

`status` tinyint(2) NOT NULL default 0 COMMENT '出库状态, 0 :待出库, 1 :已出库, 2 : ?
',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`outbound_order_detail_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='出库明细'
;
--创建索引
CREATE          INDEX          `IDX_OUTBOUND_ORDER_DETAIL_ID`          ON
`outbound_order_detail`(`outbound_order_id`) USING BTREE ;
CREATE          INDEX          `IDX_OUTBOUND_ORDER_DETAIL_ORDER_DETAIL_ID`          ON
`outbound_order_detail`(`order_detail_id`) USING BTREE ;
CREATE          INDEX          `IDX_OUTBOUND_ORDER_DETAIL_GOODS_ID`          ON
`outbound_order_detail`(`goods_id`) USING BTREE ;
CREATE          INDEX          `IDX_OUTBOUND_ORDER_DETAIL_OUTBOUND_TIME`          ON
`outbound_order_detail`(`outbound_time`) USING BTREE ;
CREATE          INDEX          `IDX_OUTBOUND_ORDER_DETAIL_CREATE_TIME`          ON
`outbound_order_detail`(`create_time`) USING BTREE ;

-----
--tab9 Table structure for `outbound_order`
-----
DROP TABLE IF EXISTS `outbound_order`;
CREATE TABLE `outbound_order` (
  `outbound_order_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '出库单号',
  `order_id` bigint(11) NOT NULL COMMENT 'FK、订单编号',
  `employee_id` bigint(11) NOT NULL COMMENT '出库人',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注',
  PRIMARY KEY (`outbound_order_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='出库单'
;
--创建索引
CREATE INDEX `IDX_OUTBOUND_ORDER_ID` ON `outbound_order`(`order_id`) USING
BTREE ;
CREATE          INDEX          `IDX_OUTBOUND_ORDER_EMPLOYEE_ID`          ON
`outbound_order`(`employee_id`) USING BTREE ;

```

```
CREATE INDEX `IDX_OUTBOUND_ORDER_CREATE_TIME` ON `outbound_order`(`create_time`)
USING BTREE ;
```

```
-----
--tab10 Table structure for `goods_inventory`
-----
```

```
DROP TABLE IF EXISTS `goods_inventory`;
CREATE TABLE `goods_inventory` (
  `goods_inventory_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '库存 ID',
  `goods_id` bigint(11) NOT NULL COMMENT 'FK、商品编号',
  `goods_batch_no` bigint(11) NOT NULL COMMENT 'FK、商品批次号',
  `repostory_id` bigint(11) NOT NULL COMMENT 'FK、仓库编号',
  `expired_time` datetime not null COMMENT '过期时间',
  `quantity` smallint(5) NOT NULL COMMENT '库存数量',
  `selling_quantity` smallint(5) NOT NULL COMMENT '可售库存数量',
  `sold_quantity` smallint(5) NOT NULL default 0 COMMENT '售出数量',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注',
  PRIMARY KEY (`goods_inventory_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='商品库存'
;
```

```
--创建索引
```

```
CREATE INDEX `IDX_GOODS_INVENTORY_GOODS_ID` ON `goods_inventory`(`goods_id`)
USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_GOODS_BATCH_NO` ON
`goods_inventory`(`goods_batch_no`) USING BTREE ;
CREATE INDEX `UNQ_IDX_GOODS_INVENTORY_REPOSTORY_ID` ON
`goods_inventory`(`repostory_id`) USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_EXPIRED_TIME` ON
`goods_inventory`(`expired_time`) USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_QUANTITY` ON `goods_inventory`(`quantity`)
USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_SELLING_QUANTITY` ON
`goods_inventory`(`selling_quantity`) USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_SOLD_QUANTITY` ON
`goods_inventory`(`sold_quantity`) USING BTREE ;
CREATE INDEX `IDX_GOODS_INVENTORY_CREATE_TIME` ON
`goods_inventory`(`create_time`) USING BTREE ;
CREATE UNIQUE INDEX `UNQ_IDX_GOODS_INVENTORY` ON
`goods_inventory`(`goods_id`,`goods_batch_no`,`repostory_id`) USING BTREE ;
```

```
--tab11 Table structure for `purchase_order_detail`
```

```

DROP TABLE IF EXISTS `purchase_order_detail`;
CREATE TABLE `purchase_order_detail` (
  `purchase_order_detail_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK, ID',
  `purchase_order_id` bigint(11) NOT NULL COMMENT 'FK、采购单号',
  `goods_id` bigint(11) NOT NULL COMMENT 'FK、商品编号',
  `quantity` smallint(5) NOT NULL COMMENT '采购数量',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `repostory_id` bigint(11) NOT NULL COMMENT 'FK、仓库编号, 商品供给到此仓库',
  `status` tinyint(2) NOT NULL default 0 COMMENT '供货状态, 0 :待供货, 1 :已供货, 2 : ?
  ',
  `supply_time` datetime null default null COMMENT '供货时间, 入库时间',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注',
  PRIMARY KEY (`purchase_order_detail_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='采购明细'
;

```

```
--创建索引
```

```

CREATE INDEX `IDX_PURCHASE_ORDER_DETAIL_PURCHASE_ORDER_ID` ON
`purchase_order_detail`(`purchase_order_id`) USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_DETAIL_GOODS_ID` ON
`purchase_order_detail`(`goods_id`) USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_DETAIL_REPOSTORY_ID` ON
`purchase_order_detail`(`repostory_id`) USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_DETAIL_CREATE_TIME` ON
`purchase_order_detail`(`create_time`) USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_DETAIL_SUPPLY_TIME` ON
`purchase_order_detail`(`supply_time`) USING BTREE ;

```

```
--tab12 Table structure for `purchase_order`
```

```

DROP TABLE IF EXISTS `purchase_order`;
CREATE TABLE `purchase_order` (
  `purchase_order_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '仓库 ID',
  `purchase_order_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '采购名称',
  `employee_id` bigint(11) NOT NULL COMMENT '采购员编号',

```



```

`supplier_id` bigint(11) NOT NULL COMMENT '供应商编号',
`create_time` datetime not null default NOW() COMMENT '采购时间',
`supply_time` datetime not null default NOW() COMMENT '最后供货时间',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`purchase_order_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='采购单'
;
--创建索引
CREATE INDEX `IDX_PURCHASE_ORDER_CREATE_TIME` ON `purchase_order`(`create_time`)
USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_SUPPLY_TIME` ON `purchase_order`(`supply_time`)
USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_EMPLOYEE_ID` ON `purchase_order`(`employee_id`)
USING BTREE ;
CREATE INDEX `IDX_PURCHASE_ORDER_SUPPLIER_ID` ON `purchase_order`(`supplier_id`)
USING BTREE ;

-----
--tab13 Table structure for `repostory`
-----
DROP TABLE IF EXISTS `repostory`;
CREATE TABLE `repostory` (
`repostory_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '仓库 ID',
`repostory_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '仓库名称',
`repostory_area` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '仓库所属区域',
`repostory_province` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL COMMENT '仓库所在省',
`repostory_city` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '仓库所在市',
`repostory_county` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL COMMENT '仓库所在区县',
`repostory_address` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL COMMENT '仓库地址',
`repostory_acreage` bigint(11) NOT NULL COMMENT '仓库面积',
`employee_id` bigint(11) NOT NULL COMMENT '仓管员编号',
`start_using_date` date null default null COMMENT '启用日期',
`create_time` datetime not null default NOW() COMMENT '创建时间',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL

```

```

COMMENT '备注',
PRIMARY KEY (`repostory_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='仓库'
;
--创建索引
CREATE INDEX `IDX_REPOSTORY_NAME` ON `repostory`(`repostory_name`) USING BTREE ;
CREATE INDEX `IDX_REPOSTORY_PROVINCE` ON `repostory`(`repostory_province`) USING
BTREE ;
CREATE INDEX `IDX_REPOSTORY_CITY` ON `repostory`(`repostory_city`) USING BTREE ;
CREATE INDEX `IDX_REPOSTORY_COUNTY` ON `repostory`(`repostory_county`) USING
BTREE ;
CREATE INDEX `IDX_REPOSTORY_ADDRESS` ON `repostory`(`repostory_address`) USING
BTREE ;
CREATE INDEX `IDX_REPOSTORY_EMPLOYEE_ID` ON `repostory`(`employee_id`) USING
BTREE ;

-----
--tab14 Table structure for `supplier`
-----
DROP TABLE IF EXISTS `supplier`;
CREATE TABLE `supplier` (
  `supplier_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '供应商 ID',
  `supplier_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
COMMENT '供应商名称',
  `address` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '供应商地址',
  `contacts_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '联络人姓名',
  `contacts_phone` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '联络人固定电话',
  `contacts_mobile` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '联络人手机',
  `supplier_category` tinyint(3) NULL DEFAULT NULL COMMENT '供应商品类别',
  `employee_id` bigint(11) NOT NULL COMMENT '采购员编号',
  `create_time` datetime not null default NOW() COMMENT '创建时间',
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`supplier_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci

```

```

COMMENT='供应商'
;
--创建索引
CREATE INDEX `IDX_SUPPLIER_EMPLOYEE_ID` ON `supplier`(`employee_id`) USING BTREE ;
CREATE INDEX `IDX_SUPPLIER_CATEGORY` ON `supplier`(`supplier_category`) USING BTREE ;
CREATE INDEX `IDX_SUPPLIER_CONTACTS_MOBILE` ON `supplier`(`contacts_mobile`) USING
BTREE ;
CREATE INDEX `IDX_SUPPLIER_CONTACTS_PHONE` ON `supplier`(`contacts_phone`) USING
BTREE ;

-----
--tab15 Table structure for `fruit_order_detail`
-----
DROP TABLE IF EXISTS `fruit_order_detail`;
CREATE TABLE `fruit_order_detail` (
  `order_detail_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '订单明细 ID' ,
  `order_id` bigint(11) NOT NULL COMMENT 'FK, 订单号' ,
  `goods_id` bigint(11) NOT NULL COMMENT 'FK, 商品编号' ,
  `quantity` smallint(5) NOT NULL COMMENT '购买数量' ,
  `total_pay` bigint(11) NOT NULL COMMENT '总金额' ,
  `total_ought_pay` bigint(11) NOT NULL COMMENT '应付总金额' ,
  `create_time` datetime not null default NOW() COMMENT '创建时间' ,
  `status` tinyint(2) NOT NULL default 0 COMMENT '状态, 0 : 待出库, 1 : 已出库' ,
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注' ,
  PRIMARY KEY (`order_detail_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='订单明细'
;
--创建索引
CREATE INDEX `IDX_ORDER_DETAIL_ORDER_ID` ON `fruit_order_detail`(`order_id`) USING
BTREE ;
CREATE INDEX `IDX_ORDER_DETAIL_GOODS_ID` ON `fruit_order_detail`(`goods_id`) USING
BTREE ;
CREATE INDEX `IDX_ORDER_DETAIL_CREATE_TIME` ON `fruit_order_detail`(`create_time`)
USING BTREE ;

-----
--tab16 Table structure for `fruit_order`
-----
DROP TABLE IF EXISTS `fruit_order`;
CREATE TABLE `fruit_order` (

```

```

`order_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT '订单号',
`customer_id` bigint(11) NOT NULL COMMENT 'FK, 所属消费者 ID',
`address_id` bigint(11) NOT NULL COMMENT 'FK, 收货地址编号',
`create_time` datetime not null default NOW() COMMENT '下单时间',
`order_pay` bigint(11) NULL COMMENT '订单总金额',
`order_ought_pay` bigint(11) NULL COMMENT '订单应付总金额',
`status` tinyint(2) NOT NULL default 0 COMMENT '状态, 0:待支付, 1:已支付, 2:未支付取消, 3:申请退款(消费者主动意向), 4:退款处理中, 5:退款完成',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`order_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='订单'
;
--创建索引
CREATE INDEX `IDX_ORDER_CUSTOMER_ID` ON `fruit_order`(`customer_id`) USING BTREE ;
CREATE INDEX `IDX_ORDER_CREATE_TIME` ON `fruit_order`(`create_time`) USING BTREE ;

-----
--tab17 Table structure for `shopping_cart`
-----
DROP TABLE IF EXISTS `shopping_cart`;
CREATE TABLE `shopping_cart` (
`shopping_cart_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK',
`customer_id` bigint(11) NOT NULL COMMENT 'FK, 所属消费者 ID',
`goods_id` bigint(11) NOT NULL COMMENT 'FK, 商品 ID',
`quantity` smallint(5) NOT NULL COMMENT '意向购买数量',
`create_time` datetime not null default NOW() COMMENT '添加时间',
`status` tinyint(2) NOT NULL default 0 COMMENT '状态, 0:添加, 1:已购买, 2:主动删除',
`remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
COMMENT '备注',
PRIMARY KEY (`shopping_cart_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='购物车'
;
--创建索引
CREATE INDEX `IDX_SHOPPING_CART_CUSTOMER_ID` ON `shopping_cart`(`customer_id`)
USING BTREE ;
CREATE INDEX `IDX_SHOPPING_CART_GOODS_ID` ON `shopping_cart`(`goods_id`) USING

```

```

BTREE ;
CREATE INDEX `IDX_SHOPPING_CART_CREATE_TIME` ON `shopping_cart`(`create_time`)
USING BTREE ;

-----
--tab18 Table structure for `goods`
-----
DROP TABLE IF EXISTS `goods`;
CREATE TABLE `goods` (
  `goods_id` bigint(11) NOT NULL AUTO_INCREMENT COMMENT 'PK,商品 ID' ,
  `goods_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '商品名称' ,
  `producing_area` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '商品产地' ,
  `market_price` bigint(11) NOT NULL COMMENT '市场价,单位分' ,
  `sell_price` bigint(11) NOT NULL COMMENT '售价, 单位分' ,
  `description` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  NULL COMMENT '商品描述' ,
  `goods_image_url` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT NULL COMMENT '商品图片 url' ,
  `tag` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '标签' ,
  `goods_type_id` bigint(11) NULL DEFAULT NULL COMMENT 'FK, 商品类型 ID(苹果、橘子、
  柚子、。。。)' ,
  `goods_category` tinyint(2) NULL DEFAULT NULL COMMENT '商品分类, 水果分的大类别' ,
  `employee_id` bigint(11) NOT NULL COMMENT '运营专员编号,哪个运营专员上架的商品' ,
  `goods_status` tinyint(2) NOT NULL COMMENT '0：待上架, 1：已上架, 2：已下架' ,
  `putaway_time` datetime null COMMENT '上架时间' ,
  `sold_out_time` datetime null default null COMMENT '下架时间' ,
  `create_time` datetime not null default NOW() COMMENT '创建时间' ,
  `remark` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL
  COMMENT '备注' ,
  PRIMARY KEY (`goods_id`)
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
COMMENT='商品'
;
--创建索引
CREATE INDEX `IDX_GOODS_TAG` ON `goods`(`tag`) USING BTREE ;
CREATE INDEX `IDX_GOODS_MARKET_PRICE` ON `goods`(`market_price`) USING BTREE ;
CREATE INDEX `IDX_GOODS_SELL_PRICE` ON `goods`(`sell_price`) USING BTREE ;
CREATE INDEX `IDX_GOODS_PRODUCING_AREA` ON `goods`(`producing_area`) USING
BTREE ;

```

```
CREATE INDEX `IDX_GOODS_EMPLOYEE_ID` ON `goods`(`employee_id`) USING BTREE ;  
CREATE INDEX `IDX_GOODS_PUTAWAY_TIME` ON `goods`(`putaway_time`) USING BTREE ;
```