数据系统课程设计报告——公司门户管理网站

15352441 钟丹彬 15352445 钟思根

2018 年 1 月 1 日

摘要

为大时代皮包责任有限公司开发个性化的公司门户管理管理系统,使公司事务的管理工作系统化、规范化、自动化，从而达到提高公司事务管理工作的效率和水平的目的。公司事务管理系统就是把分散的公司的各种信息实行统一、集中、规范的收集管理，建立分类编号管理、使用系统进行对事务的查询、修改等、提供个性化的管理系统，为公司解除后顾之忧。使系统功能能满足公司目前的管理需求，初步实现六大模块的功能，即项目管理（主要包括项目发布、项目查询、项目监控等），文档管理（主要包括文件查询、信函查询、报表查询），财产管理（主要包括财产登记、财产维修、财产投保和财产增减等），设备管理（主要实现对设备管理数据进行添加、修改、删除等操作），合同管理（主要包括合同登记、合同修改、合同备案等）、会议管理（主要包括会议发布、会议查询）等功能。公司事务管理系统为公司提供信息咨询、信息检索、信息存取等服务。本系统实现的公司事务管理系统基本上能够满足现代公司事务管理的需求。信息录入项目齐全、完整、系统。公司事务管理系统灵活使用表格对各种信息分门别类，组成公司事务管理系统，可以方便地查询、阅读、修改、交流和重复使用。

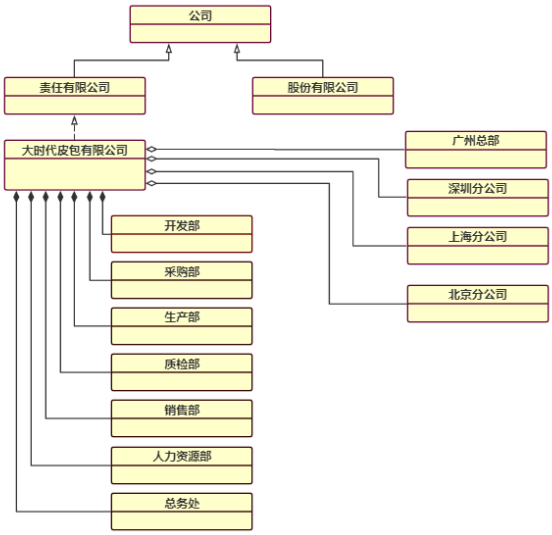
目录

1 需求建模

1.1 系统功能性需求

本网站的目标是为大时代皮包责任有限公司开发个性化的公司门户管理管理系统，侧重对于公司项目的各部门协作管理，使得公司能够高效的维护项目的运作。

图 1: 公司UML架构图



1.1.1 公司客户

作为客户，需要向系统登记信息，查询或者更改自己的信息，拥有唯一的身份标识，通过公司门户网站的客户端入口产生购买行为，并且能够实时跟进自己订单情况，无权对非该用户的订单进行操作。

1.1.2 公司员工

作为公司的员工（非经理），需要向系统登记信息，拥有唯一的工号，通过公司门户网站的端入口进入系统，只能够查询、修改自己的个人信息，可以与其他员工进行项目的协作，可以查询、更新自己被安排到的任务，和一起完成任务的其他员工，经理分派任务后，员工对任务进度进行更新。无权修改自己的薪资，也无权查看公司的核心订单，无权了解客户的情况。

1.1.3 公司部门经理

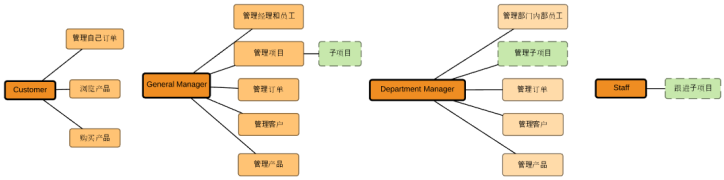
作为公司的经理，需要向系统登记信息，拥有唯一的工号，通过公司门户网站的端入口进入系统，能够查询、修改自己的个人信息，也可以查询、修改、辞退该经理所在部门的全部员工，主要负责安排员工跟进项目进度，可以查询、更新经理所在部门的项目，拥有更改员工薪资的权利，能够获取客户的信息，了解产品情况。

1.1.4 公司总经理

作为公司的经理，需要向系统登记信息，拥有唯一的工号，通过公司门户网站的端入口进入系统，能够查询、修改自己的个人信息，也可以查询、修改、辞退经所有部门的全部员工，主要负责掌握项目进度大局，可以查询、更新所有部门的项目，拥有更改员工薪资的权利，能够获取客户的信息，了解产品情况。

客户角色、员工角色、部门经理角色、总经理角色功能需求各不相同个，其功能需求架构图如图

图 2: 功能需求架构图



1.2 系统用例图

图 3: 客户用例图

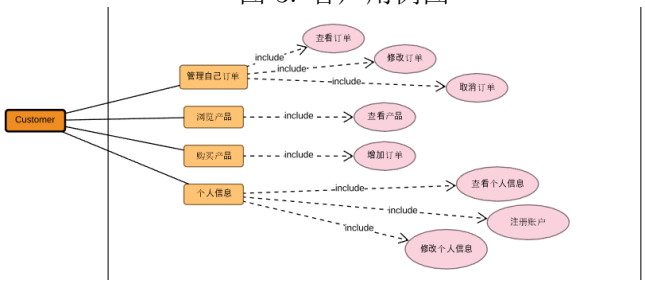


图 4: 员工用例图

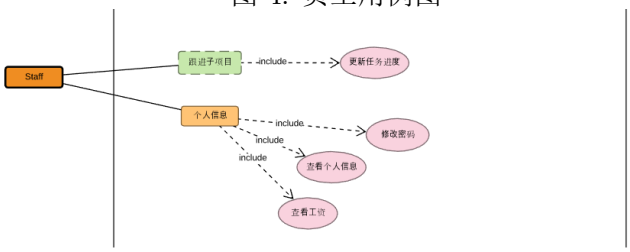


图 5: 部门经理用例图

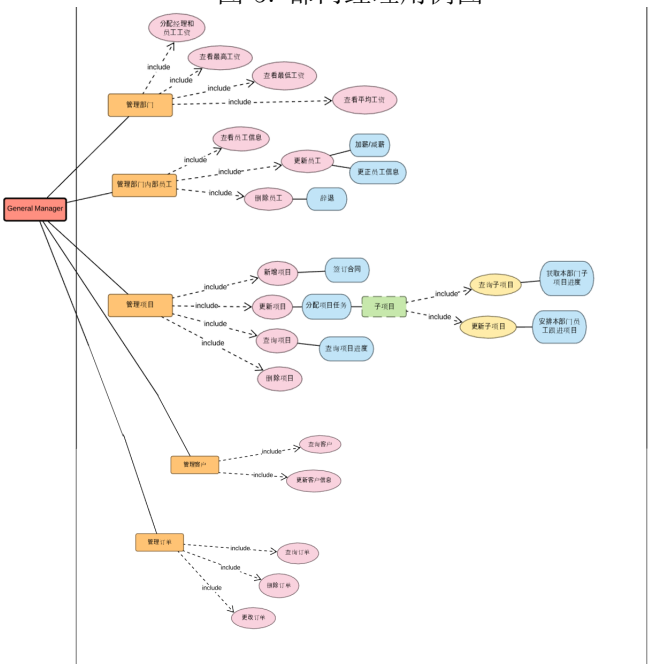
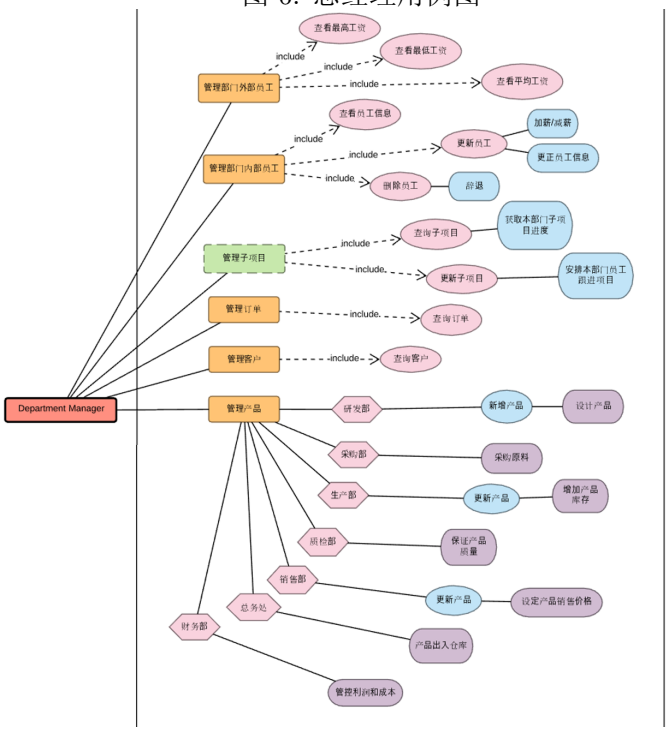


图 6: 总经理用例图



1.3 系统核心用例

本系统涉及到的用例很多，本节主要介绍以下三个核心用例，它们分别是：发布项目、分配项目、跟进项目。

1.3.1 发布项目

用例：发布项目

范围：公司内部

主要参与者：总经理、经理

关注点：

1. 任务：项目发布，新增任务。

2. 子任务：每个部门各司其职，任务分配成多个子任务，共同跟进项目。

3. 活动预算：财务做出相应支出作为成本等经费。

前置条件：客户产生订单，总经理签订合同。

成功保证：新增任务至数据库，并且同时新增各个部门不同的子任务至数据库，共同跟进项目。

主成功场景：

1. 总经理用帐号和密码进入公司门户网站入口。

2. 客户产生订单，总经理签订合同。

3. 客户收款、拨出预算。

4. 新增针对于客户订单的任务，任务状态默认为未完成，系统更新数据到数据库中。

5. 新增针对于客户订单的子任务，子任务状态默认为未完成，系统更新数据到数据库中。

扩展：

1. 总经理所输入的帐号或密码不正确

处理：系统提示用户帐号或密码错误，要求用户重新输入。

2. 财务赤字

处理：系统提示公司财务赤字，无法签订合同。

3. 客户产生订单时未登陆

处理：系统提示未登陆和注册，跳转置登陆页面。

4. 客户所购买的产品数量大于现有库存

处理：系统提示超出库存容量，必须购买小于等于库存的产品。

特殊需求：系统的设计保障系统的健壮性。

发生频率：频繁发生。

1.3.2 分配项目

用例：分配项目

范围：公司内部

主要参与者：总经理、经理

关注点：

1. 任务：项目由研发部、采购部、生产部、质检部、销售部、财务部共同跟进项目。

2. 子任务：每个部门的任务划分成多个子任务，经理安排员工共同跟进项目。

前置条件：新增项目。

成功保证：新增任务、子任务，状态默认未完成。并且各个部门收到任务安排，员工收到任务安排。

主成功场景：

1. 总经理用帐号和密码进入公司门户网站入口。

2. 客户产生订单，总经理签订合同。

3. 新增项目，为了项目高效的完成，总经理以部门为单位分派任务，新增子项目。

4. 部门经理用帐号和密码进入公司门户网站入口

5. 部门经理收到相应的任务安排。

6. 为了项目高效的完成，部门经理以员工为单位分派子任务，安排员工对项目进行跟进和具体落实。

7. 员工用帐号和密码进入公司门户网站入口

8. 员工收到相应的任务安排，在规定时间内，对项目进行跟进和具体落实。

扩展：

1. 总经理、部门经理、员工所输入的帐号或密码不正确

处理：系统提示用户帐号或密码错误，要求用户重新输入。

2. 财务部门未给各个部门分配经费

处理：各个部门的项目进行和具体落实无法进行。

3. 采购部门未给采购完成

处理：生产部门的项目进行和具体落实无法进行。

4. 客户所购买的产品数量大于现有库存

处理：系统提示超出库存容量，必须购买小于等于库存的产品。

特殊需求：系统的设计保障系统的健壮性。

发生频率：频繁发生。

1.3.3 跟进项目

用例：跟进项目

范围：公司内部

主要参与者：经理、员工

关注点：

1. 任务：部门对应子任务完成后，总任务进度更新。

2. 子任务：员工获得分配的子任务，进行子任务进度更新。

前置条件：总经理把任务划分成子任务分配给各个部门，部门经理把子任务划分给员工进行跟进。

成功保证：员工对子任务进行更新，部门经理可以查看本部门的任务进度的更新，总经理可以查看所有部门任务进度的更新情况。

主成功场景：

1. 员工用帐号和密码进入公司门户网站入口。

2. 查看需要完成的任务。

3. 落实任务，并更新任务状态为已完成。

4. 部门经理用帐号和密码进入公司门户网站入口。

5. 查看本部门任务进度，进一步安排子任务。

6. 总经理用帐号和密码进入公司门户网站入口。

7. 查看所有任务进度，进一步安排任务。

扩展：

1. 总经理、部门经理、员工所输入的帐号或密码不正确。

处理：系统提示用户帐号或密码错误，要求用户重新输入。

2. 员工想要查看除了自己的其他任务总进度

处理：系统提示无权访问

3. 部门经理想要安排员工到已完成的任务

处理：系统提示不能修改已完成任务

特殊需求：系统的设计保障系统的健壮性。

发生频率：频繁发生。

1.4 领域模型

本系统共有8个角色，分别是总经理、部门经理、员工、客户、订单、产品、任务、子任务他们的部分关系如下：

总经理可以管理0或多个经理和员工，因此总经理对于员工的关系数量是1：0..n，同样的，对于部门经理的关系数量也是是1：0..n

总经理可以管理0或多个项目，因此总经理对于项目的关系数量是1：0..n

一个部门经理可以管理0或多个项目，一个项目需要多个部门协同完成。因此部门经理对于项目的关系数量是1..n：0..n

一个员工可以完成多个任务，一个任务可以由员工一起协作完成，因此员工对于项目的关系数量是1..n：0..n

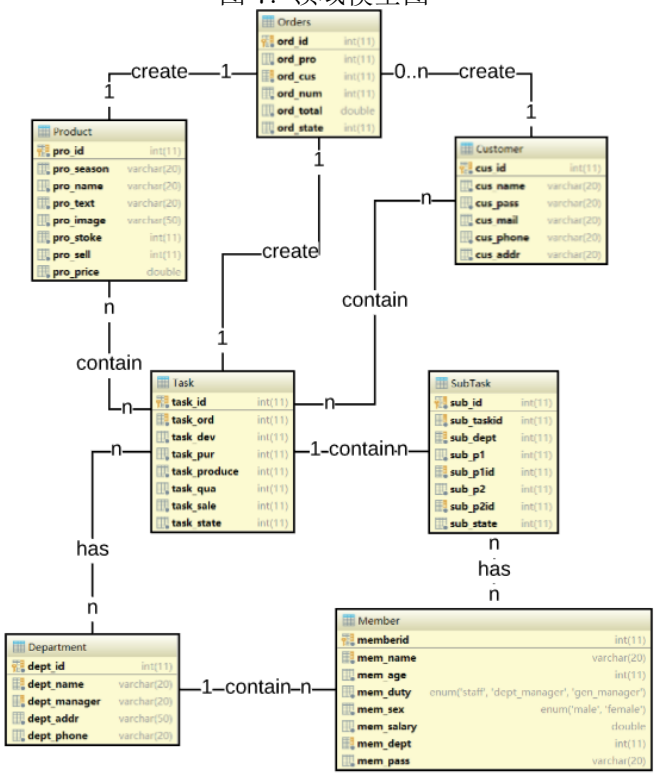
一个客户可以产生多个订单，每个订单只能由一个客户产生，因此客户对于订单是一对多的关系1：0..n。

一个订单包含一种产品，一种产品只能生成一个订单，因此订单和产品是一对一的关系。

一个员工只能在一个部门工作，一个部门可以包含多个员工，因此员工和部门是一对多的关系。

项目和产品是多对多的关系，部门和项目是多对多的关系，客户和产品是多对多的关系。

图 7: 领域模型图



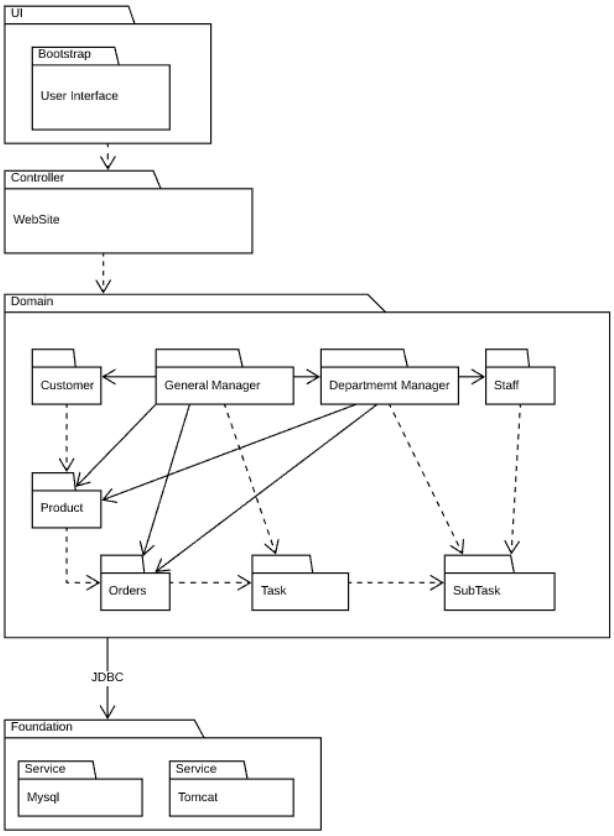
2 架构设计

为了更加清晰地显示其架构图，本系统基于Tomcat服务和Mysql服务，通过Java Service

Pages，通过JDBC的方式与储存数据库的服务主机进行连接。通过HTML/CSS/Javascript设计交互

其系统包图如下

图 8: 系统架构图



将model层再细分为两层：Domain层和Foundation层。

Domain层主要表示系统的数据模型。

Foundation层主要表示系统所使用的数据库类型支持和框架支持。

视图层：即UI层，在最顶层。其主要是显示与用户之间交互的界面，使用了HTML/CSS/Javascript框架为用户提供了一个可以跨平台的自适应的简约友好的界面，便于用户操作。

控制层，即Controller层，在第二层。其主要是连接UI层和Domain层，起到桥梁的作用。当用户通过UI层向系统发出请求时，控制层接收到请求并且调用Domain层中相应的模型进行处理，处理结束后再将结果保存在到模型中并返回给UI层。

领域层，即Domain层，在第三层。其主要是存放系统的各个模型数据。

基础层，即Foundation层，在最底层。其主要是为系统提供一些基础的服务。本系统基于Tomcat服务和Mysql服务

2.1 业务用例的实现

本系统涉及到的用例很多，本节主要介绍以下三个核心用例，它们分别是：发布项目、分配项目、跟进项目。

2.1.1 发布项目

1. 客户使用帐号和密码进入公司门户网站入口

2. 客户产生订单，选择产品发生购买行为，付款，订单记录被系统更新数据到数据库中。

3. 总经理用帐号和密码进入公司门户网站入口。

4. 查询订单事宜，总经理签订合同，拨出预算，收支记录被系统更新数据到数据库中。

5. 新增针对于客户订单的任务，任务状态默认为未完成，系统更新数据到数据库中

6. 新增针对于客户订单的子任务，子任务由经理从任务划分而来，每个部门各司其职，子任务状态默认为未完成，系统更新数据到数据库中。

2.1.2 分配项目

1. 部门经理使用帐号和密码进入公司门户网站入口

2. 部门经理收到来自总经理发布的任务和预算。

3. 部门经理根据本部门员工情况进行分配任务，安排不同的员工跟项目进行对接，安排记录被系统更新数据到数据库中。

4. 员工使用帐号和密码进入公司门户网站入口

5. 员工收到来自部门经理发布的任务和经费。

6. 员工使用经费、落实任务。

2.1.3 跟进项目

1. 员工使用帐号和密码进入公司门户网站入口

2. 员工使用经费、落实任务后，对任务状态标记为完成，更新记录被系统更新数据到数据库中。

1. 部门经理使用帐号和密码进入公司门户网站入口

2. 部门经理收到来自员工的任务进度更新。

1. 总经理使用帐号和密码进入公司门户网站入口

2. 总经理收到来自部门经理对的任务进度更新。

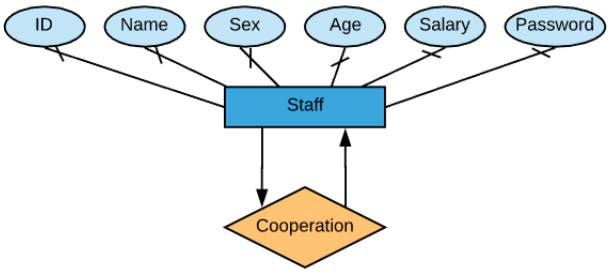
5. 员工收到来自部门经理发布的任务和经费。

6. 员工使用经费、落实任务。

2.2 数据库设计

2.2.1 员工与员工之间ER图

图 9: 员工与员工



2.2.2 员工与部门地点和时间的ER图

图 10: 员工与部门工作位置关系

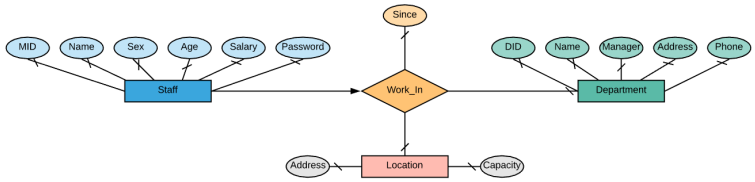
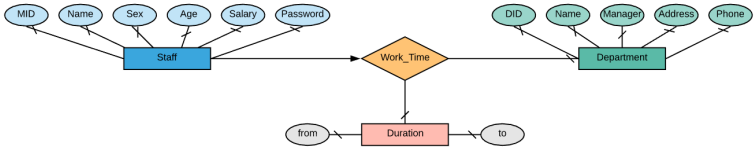
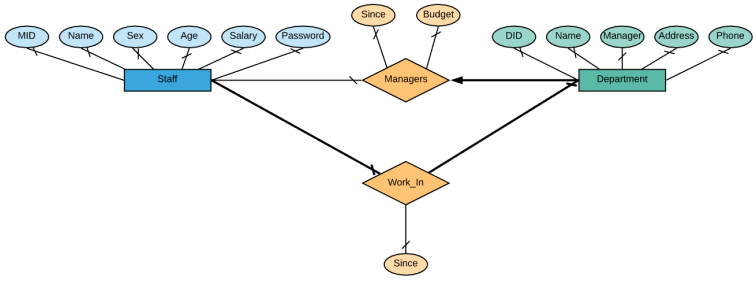


图 11: 员工与部门工作时长关系



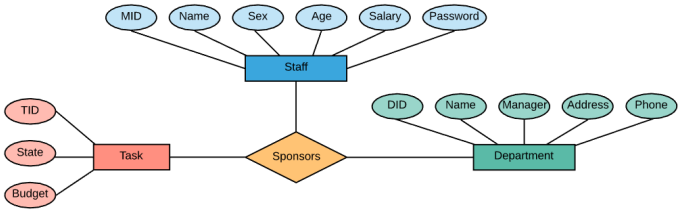
2.2.3 员工与部门经理工作关系的ER图

图 12: 员工与部门经理工作关系



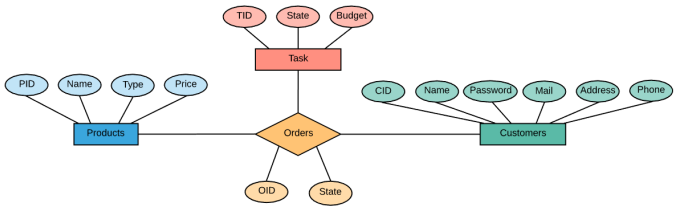
2.2.4 员工与部门经理对项目协作的ER图

图 13: 员工与部门经理对项目协作关系



2.2.5 顾客与产品订单的ER图

图 14: 顾客与产品订单关系图



2.3 数据库表设计

本系统共有7张数据库表，分别是：Customer、Department、Member、Orders、Product、Task、SubTask。

本系统数据库实现实体完整性和参照完整性，保障系统的安全性和高度可用性。

图 15: Customer客户表完整性约束

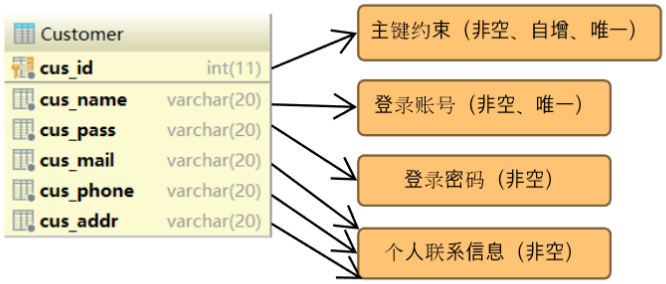


图 16: Department部门表完整性约束

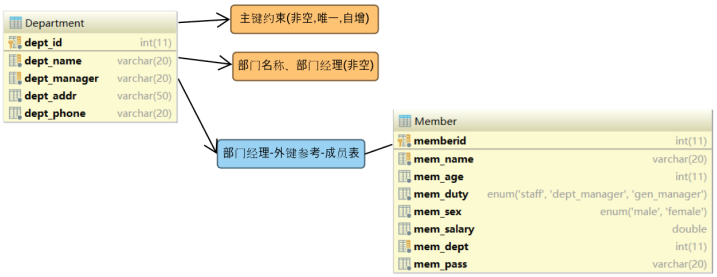


图 17: Member员工表完整性约束

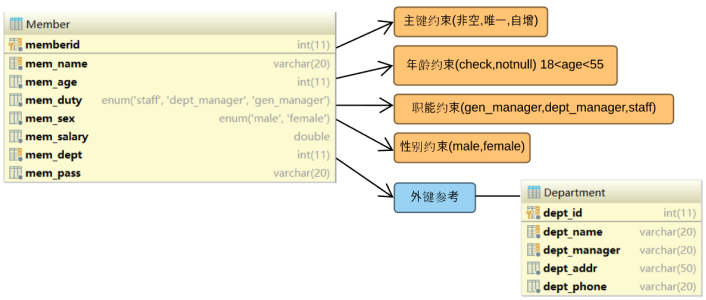


图 18: Orders订单表完整性约束

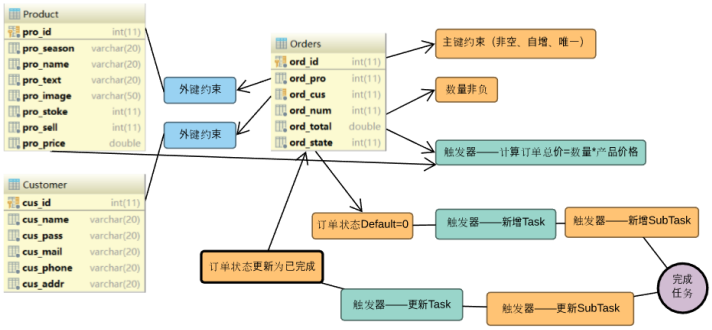


图 19: Product产品表完整性约束

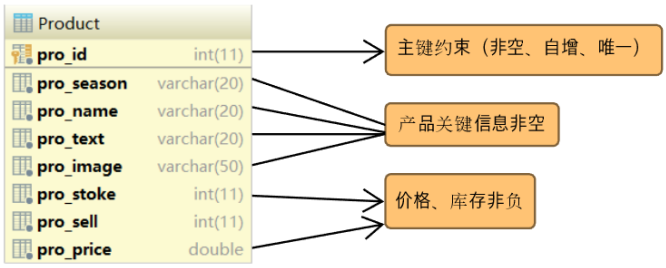


图 20: Task、SubTask任务表完整性约束

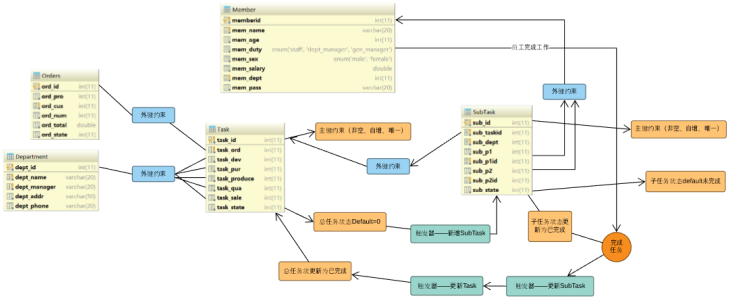


表 1: 数据库表汇总说明

表名	说明
Department	存放公司部门信息的部门表
Member	存放公司成员信息的成员表
Customer	存放顾客信息的顾客表
Orders	存放顾客下订单信息的订单表
Product	存放公司产品信息的产品表
Task	存放公司订单任务的总任务表
SubTask	存放总任务表下子任务信息的子任务表

表 2: Department 部门表

	说明			
字段	数据类型	主键、外键	允许空	说明
dept-id	INT	PRIMARY KEY	NOT NULL	部门编号
dept-name	VARCHAR(20)		NOT NULL	部门名称
dept-manager	VARCHAR(20)		NOT NULL	部门经理
dept-addr	VARCHAR(20)		NOT NULL	部门地址
dept-phone	VARCHAR(20)		NOT NULL	部门电话

表 3: Product 产品表

	说明			
字段	数据类型	主键、外键	允许空	说明
pro-id	INT	PRIMARY KEY	NOT NULL	商品编号
pro-season	VARCHAR(20)		NOT NULL	商品季节
pro-name	VARCHAR(20)		NOT NULL	商品名称
pro-text	VARCHAR(20)		NOT NULL	商品描述
pro-image	VARCHAR(20)		NOT NULL	商品图片
pro-stock	INT		NOT NULL	商品库存量
pro-sell	INT		NOT NULL	商品销售量
pro-price	DOUBLE		NOT NULL	商品单价

表 4: Task 总任务表

	说明			
字段	数 据 类型	主键、外键	允许空	说明
task-id	INT	PRIMARY KEY	NOT NULL	总任务编号
task-ord	INT	FOREIGN KEY，引 用Orders的ord-id	NOT NULL	总任务订单号
task-dev	INT		NOT NULL	开发部门完成任务情况
task-pur	INT		NOT NULL	采购部门完成任务情况
task-produce	INT		NOT NULL	生产部门完成任务情况
task-qua	INT		NOT NULL	质检部门完成任务情况
task-sale	INT		NOT NULL	销售部门完成任务情况
task-state	INT		NOT NULL	任务完成情况

表 5: SubTask 子任务表

	说明			
字段	数据类型	主键、外键	允许空	说明
sub-id	INT	PRIMARY KEY	NOT NULL	子任务编号
sub-taskid	INT	FOREIGNKEY引用Task的task-id	NOT NULL	子任务对应总任务号
sub-dept	INT	FOREIGNKEY引用Department的dept-id	NOT NULL	完成子任务的部门
sub-p1	INT		NOT NULL	完成任务1的员工编号
sub-p1id	INT	FOREIGN KEY，引用Member的memberid	NOT NULL	任务1完成情况
sub-p2	INT		NOT NULL	完成任务2的员工编号
sub-p2id	INT	FOREIGNKEY引用Member的memberid	NOT NULL	任务2完成情况
sub-state	INT		NOT NULL	子任务完成情况

表 6: Member 成员表

	说明			
字段	数据类型	主键、外键	允许空	说明
memberid	INT	PRIMARY KEY	NOT NULL	成员编号
mem-name	VARCHAR(20)		NOT NULL	成员姓名
mem-age	INT		NOT NULL	成员年龄
mem-duty	ENUM		NOT NULL	成员职责
mem-sex	ENUM		NOT NULL	成员性别
mem-salary	DOUBLE		NOT NULL	成员工资
mem-dept	INT	FOREIGNKEY，引用Department的dept-id	NOT NULL	成员所属部门
mem-pass	VARCHAR(20)		NOT NULL	成员登录密码

表 7: Customer 客户表

	说明			
字段	数据类型	主键、外键	允许空	说明
cus-id	INT	PRIMARY KEY	NOT NULL	客户编号
cus-name	VARCHAR(20)		NOT NULL	客户用户名
cus-pass	VARCHAR(20)		NOT NULL	客户密码
cus-mail	VARCHAR(20)		NOT NULL	客户邮箱
cus-phone	VARCHAR(20)		NOT NULL	客户联系电话
cus-addr	VARCHAR(20)		NOT NULL	客户地址

表 8: Orders 订单表

	说明			
字段	数据类型	主键、外键	允许空	说明
ord-id	INT	PRIMARY KEY	NOT NULL	订单编号
ord-pro	INT	FOREIGNKEY引用Product的pro-id	NOT NULL	订单商品号
ord-cus	INT	FOREIGNKEY引用Customer的cus-id	NOT NULL	订单客户号
ord-num	INT		NOT NULL	订单商品数量
ord-total	DOUBLE		NOT NULL	订单总价
ord-state	INT		NOT NULL	订单完成情况

3 模块设计

本数据系统有4大类角色：总经理、部门经理、普通员工、客户。因为本数据库系统表单与表单之间的连接较为复杂，不好单纯用不同角色视角划分。所以本系统按照对数据库中表单的操作分为多个模块：登录模块、对客户表单操作模块、对成员表单操作模块、对产品表单操作模块、对订单表单操作模块、对部门表单操作模块、对总任务表单操作模块、对子任务表单操作模块。多个模块下又根据对表单的实际操作分为多个子模块。任务完成模块设计到了客户和内部员工对商品表、订单表、总任务表、子任务表的操作，这里主要解释登录模块和任务完成模块，对基本表单的操作模块选择部分加以解释。

后端：数据库编程，完成存储过程、函数、触发器的设计。前端：使用JDBC连接服务器数据库，通过JSP编程，调用预先设计的存储过程、函数、触发器等实现对数据库的查询、修改、更新、删除操作。

```
// jdbc
```

```
String connectString = "jdbc:mysql://localhost:3306/company" +
    "?autoReconnect=true&useUnicode=true&characterEncoding=UTF-8";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(connectString, "root", "666");

Statement stmt=con.createStatement();

// end jdbc
```

3.1 登录模块

登录模块是通过网页进行操作者的身份角色确认，这一步是实现其他模块的基础。只有确认当前操作者的角色后，才能为当前操作者分配对于不同表单的不同权限。如流程图所示

图 21: 登录模块

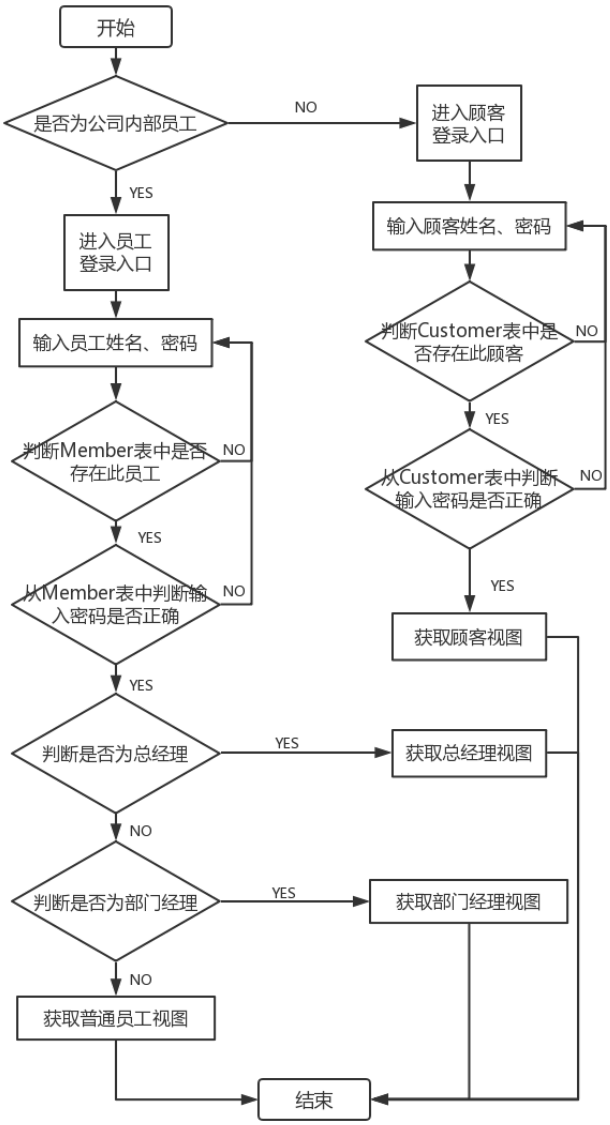


图 22: 内部员工登录入口JSP代码

```
ResultSet rs=stmt.executeQuery("select mem_pass,memberid from
Member where mem_name='"+mem_name+"'");
if(rs.next()){
    memberid = rs.getString("memberid");
    Upass = rs.getString("mem_pass");
}
```

图 23: 内部员工登录入口JSP代码

```
if(request.getParameter("login") != null && !mem_name.isEmpty()){
    if(Spass.equals(Upass)){
        alert = "登录成功";
        session.setAttribute("memberid",memberid);
        session.setAttribute("flag","1");
    }
    else{
        alert = "密码错误";
    }
}
```

图 24: 客户登录入口JSP代码

```
ResultSet rs=stmt.executeQuery("select cus_pass,cus_id from
Customer where cus_name='"+cus_name+"'");
if(rs.next()){
    Upass = rs.getString("cus_pass");
    cus_id = rs.getString("cus_id");
}
```

图 25: 客户登录入口JSP代码

```
if(request.getParameter("login") != null && !cus_name.isEmpty()){
    if(Spass.equals(Upass)){
        alert = "登录成功";
        session.setAttribute("cus_id", cus_id);
        session.setAttribute("flag","0");
    }
    else{
        alert = "密码错误";
    }
}
```

3.2 客户表单操作模块

不同角色对客户表操作不同。总经理和部门经理能够看到所有客户的信息，普通员工不能查看客户信息，客户只能查看自己的客户信息。

图 26: 各个角色对于客户表单操作逻辑

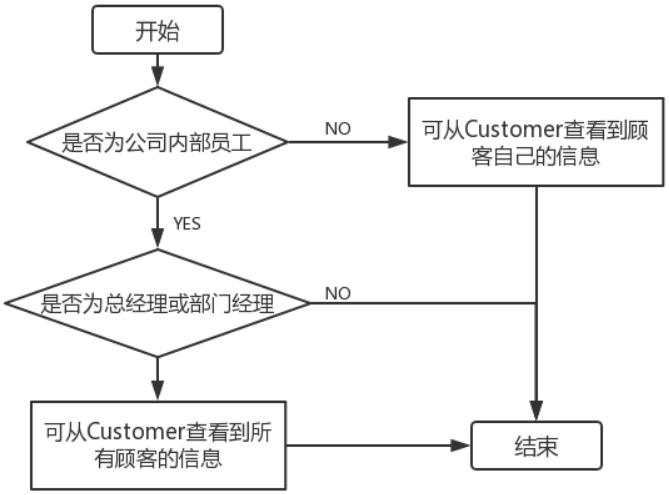


图 27: JSP中调用存储过程读取客户表单

```
ResultSet rs=stmt.executeQuery("call select_Customer("+id+","+flag
+")");
table.append("<table id=\"s_table\"><tr><th>客户id</th><th>
客户登陆名</th><th>客户登陆密码</th><th>客户邮箱</th><th>
客户联系方式</th><th>客户地址</th></tr>");
while(rs.next()){
    table.append(String.format("<tr><th>%s</th><th>%s</th><th>%s</
    th><th>%s</th><th>%s</th><th>%s</th></tr>",rs.getString("
    cus_id"),rs.getString("cus_name"),rs.getString("cus_pass"),rs.
    getString("cus_mail"),rs.getString("cus_phone"),rs.getString("
    cus_addr")));
}
```

存储过程——查看客户表单，总经理和部门经理能够看到所有客户的信息，普通员工不能查看客户信息，客户只能查看自己的客户信息。

图 28: 存储过程——查看客户表单

```
CREATE PROCEDURE select_Customer(IN id INT(10), IN flag INT(10))
  BEGIN
    IF flag=1
      THEN
      IF (SELECT mem_duty FROM Member WHERE memberid=id)='gen_manager'
          THEN
            SELECT * FROM Customer;
        ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)='dept_manager'
          THEN
            SELECT * FROM Customer;
        END IF;
    ELSEIF flag=0
      THEN
        SELECT * FROM Customer WHERE cus_id=id;
    END IF;
  END;
```

3.3 成员表单操作模块

图 29: 各个角色对于成员表单操作逻辑

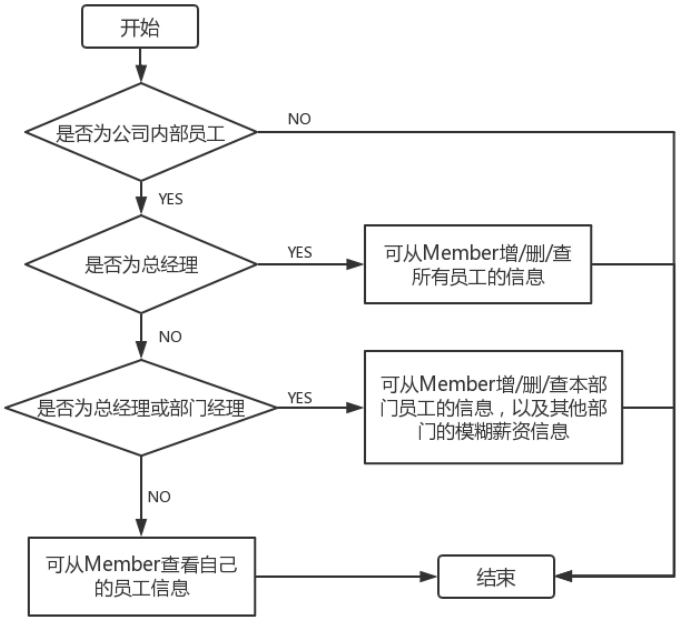


图 30: JSP种调用存储过程查询员工表单

```
ResultSet rs=stmt.executeQuery("call select_Member("+id+","+flag+")");
table.append("<table id=\"s_table\"><tr><th>员工号</th><th>姓名</th><th
>年龄</th><th>职责</th><th>性别</th><th>工资</th><th>所属部门</th><th>
密码</th><th>-</th></tr>");
while(rs.next()){
    table.append(String.format("<tr><th>%s</th><th>%s</th><th>%s</th><
    th>%s</th><th>%s</th><th>%s</th><th>%s</th><th>%s</th><th>%s %s %s<
    /th></tr>",rs.getString("memberid"),rs.getString("mem_name"),rs.
    getString("mem_age"),rs.getString("mem_duty"),rs.getString("mem_sex
    "),rs.getString("mem_salary"),rs.getString("dept_name"),rs.
    getString("mem_pass"),"<a value=\"test\" name=\"update\"
    href='MemberUpdate.jsp?pid="+rs.getString("memberid")+"'>修改</a>",
    "<a name='del'href='MemberList.jsp?pid="+rs.getString("memberid")+"
    '>删除</a>","<a name='add'href='MemberAdd.jsp?pid="+rs.getString("
    memberid")+"'>增加</a>"));
}
```

存储过程——查看员工表单，总经理能看到所有员工的信息；部门经理能看到本部门员工的信息；普通员工只能看到自己的信息；客户无权访问员工表单。

图 31: 存储过程——查看员工表单

```
CREATE PROCEDURE select_Member(IN id INT(10), IN flag INT(10))
BEGIN
  IF flag=1
    THEN
      IF (SELECT mem_duty FROM Member WHERE memberid=id)=
       'gen_manager'
        THEN
          SELECT Member.*,Department.dept_name FROM Member,Department
           WHERE Member.mem_dept = Department.dept_id;
      ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)=
       'dept_manager'
          THEN
            SELECT Member.*,Department.dept_name FROM Member,
            Department WHERE Member.mem_dept = Department.dept_id AND
             Member.mem_dept=(SELECT mem_dept FROM Member WHERE
            memberid=id);
      ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)='staff'
          THEN
            SELECT Member.*,Department.dept_name FROM Member,
            Department WHERE Member.mem_dept = Department.dept_id AND
             Member.memberid=id;
      END IF;
  END IF;
END;
```

图 32: 存储过程——查看各个部门的最高、最低、平均薪资

```
CREATE PROCEDURE part_salary()
BEGIN
  SELECT mem_dept,dept_name,max(Member.mem_salary),min(Member.
  mem_salary),avg(Member.mem_salary) FROM Member,Department WHERE
  Member.mem_dept=Department.dept_id GROUP BY Member.mem_dept;

END;
```

存储过程——增加员工，总经理能增加任何部门的员工；部门经理能增加本部门的员工；普通员工和客户无权增加员工。

图 33: 存储过程——增加员工

```
CREATE PROCEDURE insert_member(IN name   VARCHAR(20), IN age INT(10),
 IN duty ENUM (13), IN sex ENUM (7),
                              IN salary DOUBLE(22), IN dept INT(10),
                              IN pass VARCHAR(20), OUT mes VARCHAR(20)
BEGIN
  IF duty='dept_manager'
    THEN
      IF (SELECT count(*) FROM Member WHERE mem_duty=duty AND
      mem_dept=dept)<1
        THEN
          INSERT IGNORE INTO Member(`mem_name`,`mem_age`,`mem_duty`,`
          mem_sex`,`mem_salary`,`mem_dept`,`mem_pass`) VALUEs(name,
          age,duty,sex,salary,dept,pass);
          SET mes="successful";
        ELSE
          SET mes="failed";
      END IF;
  ELSEIF duty='gen_manager'
    THEN
      SET mes="failed";
  ELSE
      INSERT IGNORE INTO Member(`mem_name`,`mem_age`,`mem_duty`,`
      mem_sex`,`mem_salary`,`mem_dept`,`mem_pass`) VALUEs(name,age,
      duty,sex,salary,dept,pass);
      SET mes="successful";
  END IF;

  SELECT mes;
END;
```

图 34: JSP中调用存储过程增加员工过程

```
if(sub != null){
  String fmt = "call insert_member(\"%s\",\"%s\",\"%s\",\"%s\",\"%s\",\"
  %s\",\"%s\",@mes)";
  sql = String.format(
  fmt,mem_name,mem_age,mem_duty,mem_sex,mem_salary,mem_dept,mem_pass);
  ResultSet rs = stmt.executeQuery(sql);
  if(rs.next()){
    msg2 = rs.getString("mes");
  }
}
```

修改Member的员工信息。总经理能修改所有员工的任何信息；部门经理能修改本部门员工的信息和修改自己的密码；普通员工只能修改自己的密码。

图 35: 各个角色对于成员表单操作逻辑

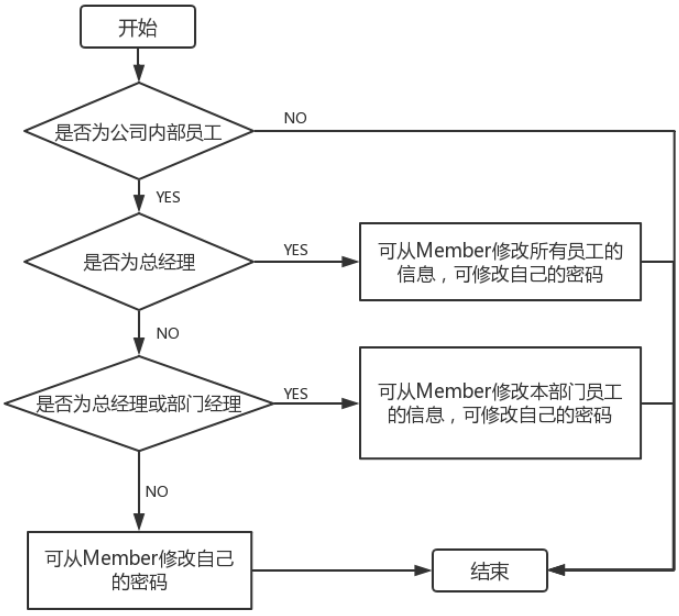


图 36: JSP中执行更新操作

```
if(request.getParameter("sub") != null){
  try{
    String mem_name2 = request.getParameter("mem_name");
    String mem_age2 = request.getParameter("mem_age");
    String mem_duty2 = request.getParameter("mem_duty");
    String mem_sex2 = request.getParameter("mem_sex");
    String mem_dept2 = request.getParameter("mem_dept");
    String mem_salary2 = request.getParameter("mem_salary");
    String mem_pass2 = request.getParameter("mem_pass");
    String fmt = "update Member set mem_name='%s',mem_age='%s',mem_duty='%s
    ',mem_sex='%s',mem_dept='%s',mem_salary='%s',mem_pass='%s' where
    memberid='%s'";
    sql = String.format(fmt,mem_name2,mem_age2,mem_duty2,mem_sex2,mem_dept2
    ,mem_salary2,mem_pass2,request.getParameter("Pid"));
    int cnt = stmt.executeUpdate(sql);
    if(cnt > 0) msg2="修改成功";
  }
```

图 37: Javascript实现表单限制

```
<script>
 var duty = '<%=duty%>';
 var pid = '<%=pid%>';
 var id = '<%=id%>';
 var mem_name = document.getElementById("mem_name");
 var mem_age = document.getElementById("mem_age");
 var mem_duty = document.getElementById("mem_duty");
 var mem_sex = document.getElementById("mem_sex");
 var mem_salary = document.getElementById("mem_salary");
 var mem_dept = document.getElementById("mem_dept");
 var sub = document.getElementById("sub");
 if(duty == "staff" || pid == id){
    mem_name.readOnly = "true";
    mem_age.readOnly = "true";
    mem_duty.disabled = "disabled";
    mem_sex.disabled = "disabled";
    mem_salary.readOnly = "true";
    mem_dept.disabled = "disabled";
 }
 sub.onclick=function(){
   mem_duty.removeAttribute("disabled");
   mem_sex.removeAttribute("disabled");
   mem_dept.removeAttribute("disabled");
 }
</script>
```

3.4 产品表单操作模块

所有角色都可对产品表单进行完全查询，但只有开发部经理能够增加新产品，只有生产部经理才能对产品库存量进行增加。

图 38: 各个角色对于产品的逻辑

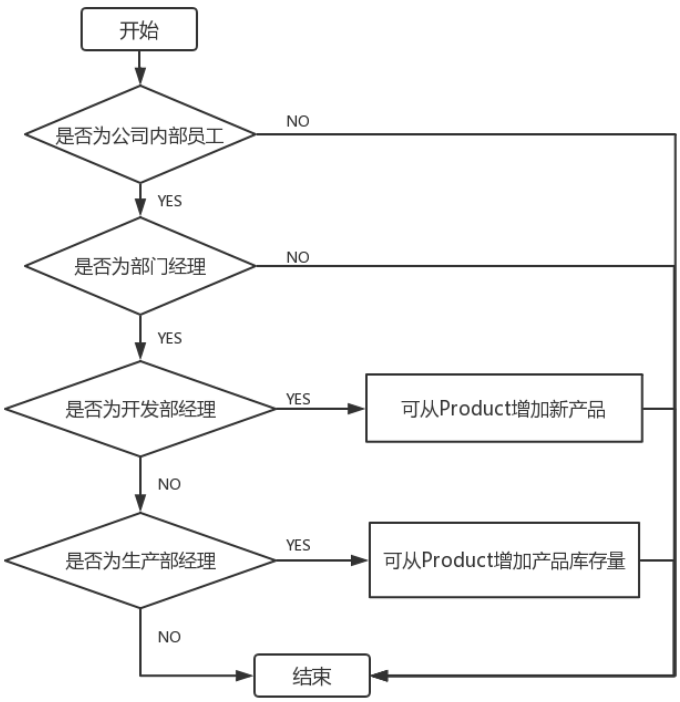


图 39: 存储过程——查询产品表单

```
CREATE PROCEDURE select_Product()
BEGIN
  SELECT * FROM Product;
END;
```

图 40: JSP中调用查询表单存储过程

```
ResultSet rs=stmt.executeQuery("select * from Product");
table.append("<table id=\"s_table\"><tr><th>商品号</th><th>季节</th><th
>名称</th><th>描述</th><th>图片</th><th>库存量</th><th>销售量</th><th>
单价</th><th>-</th></tr>");
while(rs.next()){
    table.append(String.format("<tr><th>%s</th><th>%s</th><th>%s</th><
    th>%s</th><th>%s</th><th>%s</th><th>%s</th><th>%s</th><th>%s %s %s<
    /th></tr>",rs.getString("pro_id"),rs.getString("pro_season"),rs.
    getString("pro_name"),rs.getString("pro_text"),rs.getString("
    pro_image"),rs.getString("pro_stoke"),rs.getString("pro_sell"),rs.
    getString("pro_price"),"<a name='update'
    href='ProductUpdate.jsp?pid="+rs.getString("pro_id")+"'>增加库存</
    a>","<a name='add' href='ProductAdd.jsp?pid="+rs.getString("pro_id"
    )+"'>增加新商品</a>","<button id='buy' name='buy'value='"+rs.
    getString("pro_id")+"'>购买</button>"));
}
```

存储过程——增加新产品，只有开发部经理才能增加新产品。

图 41: 存储过程——增加新产品

```
CREATE PROCEDURE insert_product_dev(IN season VARCHAR(20), IN name
VARCHAR(20), IN texts VARCHAR(20),
                                    IN images VARCHAR(20), IN price
                                    DOUBLE(22), OUT mes VARCHAR(20))
BEGIN
  INSERT IGNORE INTO Product(`pro_season`,`pro_name`,`pro_text`,`
  pro_image`,`pro_price`)
  VALUES(season,name,texts,images,price);
  SET mes="successful";
  SELECT mes;
END;
```

存储过程——对已有产品增加库存，只有生产部经理能执行此操作。

图 42: 存储过程——对已有产品增加库存

```
CREATE PROCEDURE update_product_stoke(IN proid INT(10), IN addstoke
INT(10))
BEGIN
    UPDATE Product SET Product.pro_stoke = Product.pro_stoke+addstoke
     WHERE Product.pro_id=proid;
END;
```

在JSP中调用存储过程

只有开发部经理能调用增加新产品；

只有生产部经理能调用增加库存存储过程；

只有客户能从商品列表中挑选商品下订单。

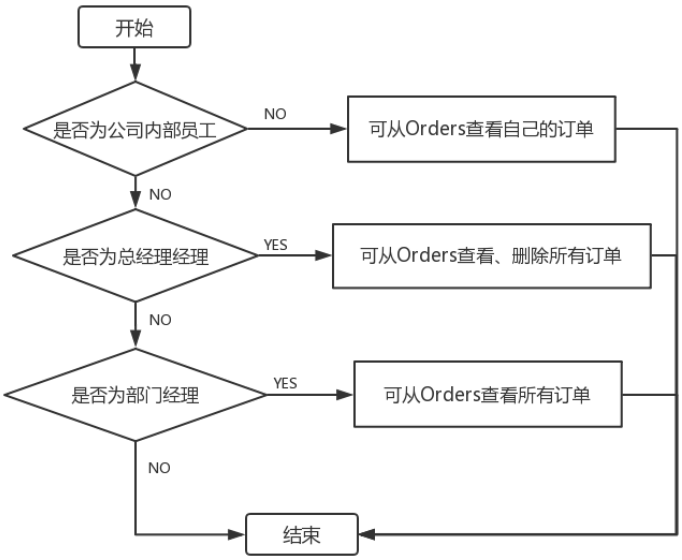
图 43: JavaScript限制表单操作

```
<script>
    var duty = '<%=duty%>';
    var dept = '<%=dept%>';
    var flag = '<%=flag%>'
    var update = document.getElementsByName("update");
    var add = document.getElementsByName("add");
    var buy = document.getElementsByName("buy");
    if(flag!="0"){
        for(var i=0;i<buy.length;++i){
            buy[i].style.display="none";
        }
    }
    if(duty!="dept_manager" || dept!="1" || flag!="1"){
        for(var i=0;i<add.length;++i){
            add[i].style.display = "none";
        }
    }
    if(duty!="dept_manager" || dept!="3" || flag!="1"){
        for(var i=0;i<update.length;++i){
            update[i].style.display = "none";
        }
    }
    for(var i=0;i<buy.length;++i){
        (function(i){
            buy[i].onclick=function(){
                var show = document.getElementById("show");
                var buy_pro = document.getElementById("buy_pro");
                var buy_num = document.getElementById("buy_num");
                show.style.display = "inline-block";
                buy_pro.value = buy[i].value;
            }
        })(i);
    }
    var sub = document.getElementById("sub");
    var buy_pro = document.getElementById("buy_pro");
    var buy_num = document.getElementById("buy_num");
</script>
```

3.5 订单表单操作模块

客户能够查询自己的订单，也可通过购买的方式增加自己的订单；总经理可以查看所有的订单以及删除订单；部门经理只能查看所有订单。

图 44: 各个角色对于产品的逻辑



存储过程——查看订单表。总经理、部门经理可以查看所有订单表；客户可以查看自己的订单。

图 45: 存储过程——查看订单表

```
CREATE PROCEDURE select_Orders(IN id INT(10), IN flag INT(10))
BEGIN
  IF flag=1
    THEN
      IF (SELECT mem_duty FROM Member WHERE memberid=id)='gen_manager'
      THEN
        SELECT * FROM Orders;
      ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)=
      'dept_manager'
        THEN
          SELECT * FROM Orders;
      END IF;
  ELSEIF flag=0
    THEN
      SELECT * FROM Orders WHERE Orders.ord_cus=id;
  END IF;
END;
```

JavaScript限制只有总经理能进行删除订单操作。其他表单的操作情况类似，这里不作赘述。

图 46: JavaScript限制表单删除操作

```
<script>
    var duty = '<%=duty%>';
    var del = document.getElementsByName("del");
    if(duty != "gen_manager"){
        for(var i=0;i<del.length;++i){
            del[i].style.visibility = "hidden";
        }
    }
</script>
```

存储过程——查看公司部门表。只有内部员工才能查看，客户不能查看部门表。

图 47: 存储过程——查看公司部门表

```
CREATE PROCEDURE select_Department(IN id INT(10), IN flag INT(10))
BEGIN
  IF flag=1
    THEN
      IF (SELECT mem_duty FROM Member WHERE memberid=id)='gen_manager'
        THEN
          SELECT * FROM Department;
      ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)=
      'dept_manager'
        THEN
          SELECT * FROM Department;
      ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)='staff'
        THEN
          SELECT * FROM Department;
      END IF;
  END IF;
END;
```

JavaScript中限制只有总经理才能对部门信息做出修改。

图 48: JavaScript限制表单更新操作

```
<script>
    var duty = '<%=duty%>';
    var update = document.getElementsByName("update");
    if(duty != "gen_manager"){
        for(var i=0;i<update.length;++i){
            update[i].style.visibility = "hidden";
        }
    }
</script>
```

存储过程——查看公司总任务表。只有总经理能够查看。

图 49: 存储过程——查看公司总任务表

```
CREATE PROCEDURE select_Task(IN id INT(10), IN flag INT(10))
BEGIN
  IF flag=1
    THEN
      IF (SELECT mem_duty FROM Member WHERE memberid=id)='gen_manager'
      THEN
        SELECT * FROM Task;
      END IF;
  END IF;
END;
```

存储过程——查看公司子任务表。总经理能够查看所有子任务；部门经理能查看本部门的子任务；普通员工只能查看自己需要完成的子任务；用户无权对此表做出操作。

图 50: 存储过程——查看公司子任务表

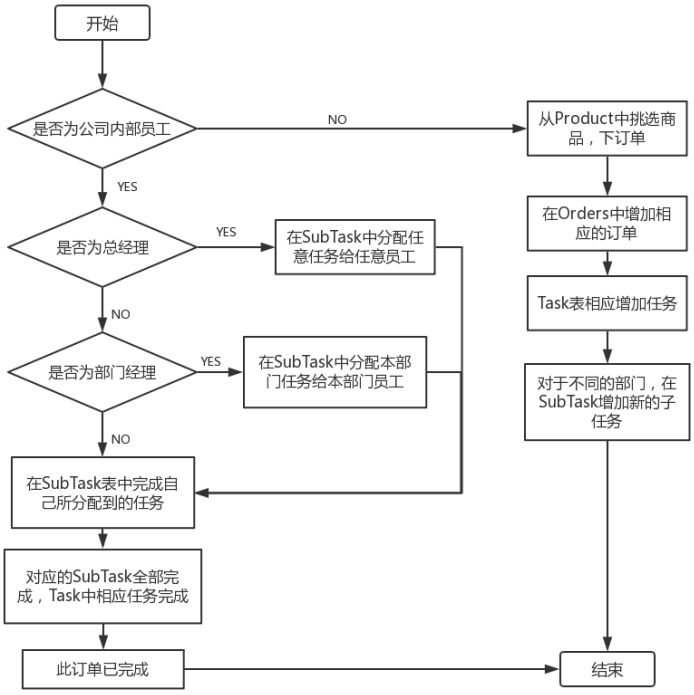
```
CREATE PROCEDURE select_SubTask(IN id INT(10), IN flag INT(10))
BEGIN
  IF flag=1
  THEN
    IF (SELECT mem_duty FROM Member WHERE memberid=id)='gen_manager'
    THEN
      SELECT * FROM SubTask;

      DROP TEMPORARY TABLE IF EXISTS Temp1;
      CREATE TEMPORARY TABLE Temp1(p1 INT,p1name VARCHAR(20));
      INSERT  INTO Temp1(`p1`,`p1name`)  SELECT SubTask.sub_p1id,
      Member.mem_name FROM Member,SubTask WHERE Member.memberid =
      SubTask.sub_p1id;

      DROP TEMPORARY TABLE IF EXISTS Temp2;
      CREATE TEMPORARY TABLE Temp2(p2 INT,p2name VARCHAR(20));
      INSERT  INTO Temp2(`p2`,`p2name`)  SELECT SubTask.sub_p2id,
      Member.mem_name FROM Member,SubTask WHERE Member.memberid =
      SubTask.sub_p2id;

      SELECT p1name FROM Temp1;
      SELECT p2name FROM Temp2;

    ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)=
    'dept_manager'
      THEN
        SELECT * FROM SubTask WHERE sub_dept=(SELECT mem_dept FROM
        Member WHERE memberid=id);

        DROP TEMPORARY TABLE IF EXISTS Temp1;
        CREATE TEMPORARY TABLE Temp1(p1 INT,p1name VARCHAR(20));
        INSERT  INTO Temp1(`p1`,`p1name`)  SELECT SubTask.sub_p1id,
        Member.mem_name FROM Member,SubTask WHERE SubTask.sub_dept=(
        SELECT mem_dept FROM Member WHERE memberid=id) AND Member.
        memberid=SubTask.sub_p1id;

        DROP TEMPORARY TABLE IF EXISTS Temp2;
        CREATE TEMPORARY TABLE Temp2(p2 INT,p2name VARCHAR(20));
        INSERT  INTO Temp2(`p2`,`p2name`)  SELECT SubTask.sub_p2id,
        Member.mem_name FROM Member,SubTask WHERE SubTask.sub_dept=(
        SELECT mem_dept FROM Member WHERE memberid=id) AND Member.
        memberid=SubTask.sub_p2id;

        SELECT Temp1.p1name FROM Temp1;
        SELECT Temp2.p2name FROM Temp2;

    ELSEIF (SELECT mem_duty FROM Member WHERE memberid=id)='staff'
    THEN
      SELECT * FROM SubTask WHERE sub_p1id=id OR sub_p2id=id;

      DROP TEMPORARY TABLE IF EXISTS Temp1;
      CREATE TEMPORARY TABLE Temp1(p1 INT,p2 INT);
      INSERT  INTO Temp1(`p1`,`p2`)  SELECT SubTask.sub_p1id,SubTask.
      sub_p2id FROM SubTask WHERE sub_p1id=id OR sub_p2id=id;

      SELECT Member.mem_name FROM Member,Temp1 WHERE Member.memberid=
      Temp1.p1;
      SELECT Member.mem_name FROM Member,Temp1 WHERE Member.memberid=
      Temp1.p2;

    END IF;
  END IF;
END;
```

3.6 订单完成模块

订单完成模块从客户下订单开始，客户从商品表中选择相应的商品和商品数量下订单。订单表中增加新订单，相应地，总任务表中新增加任务。总任务表记录了每个部门的任务完成情况，总任务表一增加，子任务表相应的增加。子任务表记录了某个部门的任务完成情况，

以及完成子任务的员工。员工完成子任务表中自己的任务，如果子任务表中两位员工都已完成任务，那么总任务表中相应的部门任务完成情况就会更新。如果所有部门都已完成总任务表中的部门任务，那么总任务表中的任务完成情况就被更新，相应的，订单完成情况也被更新即此订单完成。

图 51: 各个角色对于产品的逻辑



触发器——当客户从商品表确定下订单时，根据商品单价和购买熟练计算出此订单的价格。

图 52: 触发器——客户从商品表确定下订单

```
CREATE TRIGGER Trigger_Orders_2
BEFORE INSERT ON Orders
FOR EACH ROW
  BEGIN
    SET @total = (SELECT pro_price FROM Product WHERE pro_id = new.
    ord_pro);
    SET new.ord_total = new.ord_num * @total;
END;
```

触发器——当订单表中增加新订单时，在总任务表中增加相应的新任务。

图 53: 触发器——订单表中增加新订单

```
CREATE TRIGGER Trigger_Orders_1
AFTER INSERT ON Orders
FOR EACH ROW
  BEGIN
    SET @c = (SELECT pro_stoke FROM Product WHERE pro_id = new.ord_pro);
    SET @s = (SELECT pro_sell FROM Product WHERE pro_id = new.ord_pro);
    UPDATE Product
    SET pro_stoke = @c - new.ord_num
    WHERE pro_id = new.ord_pro;
    UPDATE Product
    SET pro_sell = @s + new.ord_num
    WHERE pro_id = new.ord_pro;
    INSERT IGNORE INTO Task(task_ord)
     VALUES(new.ord_id);
END;
```

触发器——当总任务表中增加新任务时，子任务表对于每个部门相应地增添新子任务。

图 54: 触发器——任务表中增加新任务

```
CREATE TRIGGER Trigger_Task_1
AFTER INSERT ON Task
FOR EACH ROW
  BEGIN
    INSERT IGNORE INTO SubTask(sub_taskid,sub_dept)
     VALUES(new.task_id,1);
    INSERT IGNORE INTO SubTask(sub_taskid,sub_dept)
     VALUES(new.task_id,2);
    INSERT IGNORE INTO SubTask(sub_taskid,sub_dept)
     VALUES(new.task_id,3);
    INSERT IGNORE INTO SubTask(sub_taskid,sub_dept)
     VALUES(new.task_id,4);
    INSERT IGNORE INTO SubTask(sub_taskid,sub_dept)
     VALUES(new.task_id,5);
END;
```

触发器——当子任务表中子任务两项任务都被完成后，此子任务的完成情况被更新。并且在总任务表中，对应的部门任务完成情况被更新。

图 55: 触发器——员工分配的任务完成

```
CREATE TRIGGER Trigger_SubTask_1
BEFORE UPDATE ON SubTask
FOR EACH ROW
  BEGIN
    SET @dept = NEW.sub_dept;
    IF new.sub_p1 = 1 AND new.sub_p2 = 1
        THEN
        SET new.sub_state = 1;

            IF @dept=1
            THEN
                UPDATE Task SET task_dev  = 1 WHERE task_id = NEW.
                sub_taskid;

            ELSEIF @dept=2
                THEN
                    UPDATE Task SET task_pur  = 1 WHERE task_id = NEW.
                    sub_taskid;

            ELSEIF @dept=3
            THEN
                UPDATE Task SET task_produce  = 1 WHERE task_id = NEW.
                sub_taskid;

            ELSEIF @dept=4
            THEN
                UPDATE Task SET task_qua  = 1 WHERE task_id = NEW.
                sub_taskid;

            ELSEIF @dept=5
            THEN
                UPDATE Task SET task_sale = 1 WHERE task_id = NEW.
                sub_taskid;
            END IF;
        END IF ;
END;
```

触发器——当总任务表中的各个部门任务都完成，此任务完成情况更新。相应的，此任务对应的订单完成情况也更新。

图 56: 触发器——部门分配的任务完成

```
CREATE TRIGGER Trigger_Task_2
BEFORE UPDATE ON Task
FOR EACH ROW
  BEGIN
        IF new.task_dev = 1 AND new.task_pur = 1 AND new.task_produce =
         1 AND new.task_qua=1 AND new.task_sale=1
            THEN
                SET new.task_state = 1;
                UPDATE Orders SET ord_state = 1 WHERE NEW.task_ord =
                ord_id;
        END IF;
END;
```

存储过程——经理安排员工工作，把部门任务划分为子任务分配

图 57: 存储过程——经理安排员工需要完成的任务

```
CREATE PROCEDURE arrang(IN subid INT(10), IN p1id INT(10), IN p2id INT(10))
BEGIN
  IF (SELECT sub_state FROM SubTask WHERE sub_id=subid)=0
    THEN
      UPDATE SubTask SET SubTask.sub_p1id = p1id WHERE SubTask.sub_id=subid;
      UPDATE SubTask SET SubTask.sub_p2id = p2id WHERE SubTask.sub_id=subid;
  END IF;
END;
```

存储过程——员工落实子任务并更新任务状态。

图 58: 存储过程——员工更新任务状态

```
CREATE PROCEDURE update_SubTaskp1(IN subid INT(10))
BEGIN
  UPDATE SubTask SET SubTask.sub_p1 = 1 WHERE SubTask.sub_id=subid;
END;

CREATE PROCEDURE update_SubTaskp2(IN subid INT(10))
BEGIN
  UPDATE SubTask SET SubTask.sub_p2 = 1 WHERE SubTask.sub_id=subid;
END;
```

JavaScript中中限制内部员工只能完成自己的子任务，不能对一起合作的其他员工任务状态进行更新。

图 59: JavaScript限制表单更新操作

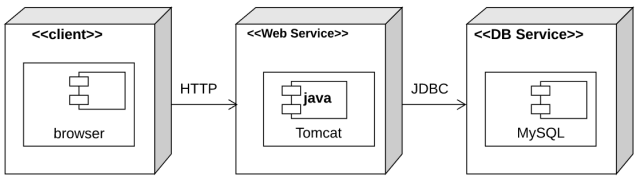
```
<script>
    var id = '<%=id%>';
    var duty = '<%=duty%>'
    var update = document.getElementsByName("update");
    var p1 = document.getElementsByName("p1");
    var p2 = document.getElementsByName("p2");
    for(var i=0;i<p1.length;++i){
        if(id != p1[i].getAttribute("value")){
            p1[i].style.visibility = "hidden";
        }
        if(id != p2[i].getAttribute("value")){
            p2[i].style.visibility = "hidden";
        }
    }
    if(duty == "staff"){
        for(var i=0;i<update.length;++i){
            update[i].style.visibility = "hidden";
        }
    }
</script>
```

4 部署与应用

4.1 部署图

本系统采用B/S模式，当服务器端完全部署应用后，客户端即可通过浏览器进行相应的访问。

图 60: 系统部署图



4.2 部署过程

腾讯云主机（centos 6.8）部署过程

1. 安装 java-jdk 2. 安装 mysql 3. 安装 tomcat

4.3 运行说明

公司内部登陆入口：

在网页中输入 http://111.231.202.192:8080/bigdb/MemLogin.jsp

总经理用户名：Carrie 密码：123

开发部经理用户名：Amy 密码：123456

开发部员工用户名：Debbie 密码：123456

客户登陆入口：http://111.231.202.192:8080/bigdb/CusLogin.jsp

测试登陆名：Test

测试密码： 123456

4.4 部分功能截图

图 61: 公司员工登陆入口

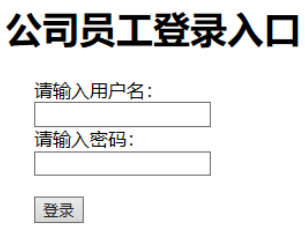


图 62: 公司员工登陆成功

图 63: 客户登陆入口

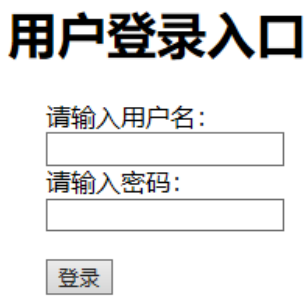


图 64: 客户登陆成功

图 65: 总经理、经理可以查看客户信息

图 66: 客户只能管理自己的信息——现登陆用户为Test

图 67: 总经理视图——可以管理所有部门的所有员工

成员表单

员工号	姓名	年龄	职责	性别	工资	所属部门	密码	-
1	Amy	20	dept_manager	female	5000	develop	1234567	修改 删除 增加
2	Debbie	20	staff	female	2000	develop	123456	修改 删除 增加
3	Mary	20	staff	female	2000	develop	123456	修改 删除 增加
25	Hani	55	staff	male	3000	develop	123456	修改 删除 增加
4	Baby	20	dept_manager	female	5000	purchase	123456	修改 删除 增加
5	Barbie	20	staff	female	2000	purchase	123456	修改 删除 增加
6	Cady	20	staff	female	2000	purchase	123456	修改 删除 增加
7	Daisy	20	dept_manager	female	5000	produce	1234567	修改 删除 增加
8	Damia	20	staff	male	2000	produce	123456	修改 删除 增加
9	Edwin	20	staff	male	2000	produce	123456	修改 删除 增加
27	Ok	23	staff	female	5000	produce	123456	修改 删除 增加
10	Gaiya	20	dept_manager	female	5000	quality	123456	修改 删除 增加
11	Hana	20	staff	female	2000	quality	123456	修改 删除 增加
12	Hans	20	staff	male	2000	quality	123456	修改 删除 增加
13	Jack	20	dept_manager	male	5000	sales	123456	修改 删除 增加
14	Jackson	20	staff	male	2000	sales	123456	修改 删除 增加
15	Kami	20	staff	male	2000	sales	123456	修改 删除 增加
17	Lily	20	dept_manager	female	5000	finance	123456	修改 删除 增加
18	Edision	20	staff	male	2000	finance	123456	修改 删除 增加
19	July	20	staff	female	2000	finance	123456	修改 删除 增加

部门编号	部门名称	部门最高工资	部门最低工资	部门平均工资
1	develop	5000	2000	3000
2	purchase	5000	2000	3000
3	produce	5000	2000	3500
4	quality	5000	2000	3000
5	sales	5000	2000	3000
6	finance	5000	2000	3000

客户表单
成员表单
产品表单
订单表单
部门表单
总任务表单
子任务表单

图 68: 开发部经理视图——可以管理开发部的员工

成员表单

员工号	姓名	年龄	职责	性别	工资	所属部门	密码	-
1	Amy	20	dept_manager	female	5000	develop	1234567	修改 删除 增加
2	Debbie	20	staff	female	2000	develop	123456	修改 删除 增加
3	Mary	20	staff	female	2000	develop	123456	修改 删除 增加
25	Hani	55	staff	male	3000	develop	123456	修改 删除 增加

部门编号	部门名称	部门最高工资	部门最低工资	部门平均工资
1	develop	5000	2000	3000
2	purchase	5000	2000	3000
3	produce	5000	2000	3500
4	quality	5000	2000	3000
5	sales	5000	2000	3000
6	finance	5000	2000	3000

图 69: 开发部员工视图——只能查看自己的员工信息

成员表单

员工号	姓名	年龄	职责	性别	工资	所属部门	密码	-
2	Debbie	20	staff	female	2000	develop	123456	修改

部门编号	部门名称	部门最高工资	部门最低工资	部门平均工资
1	develop	5000	2000	3000
2	purchase	5000	2000	3000
3	produce	5000	2000	3500
4	quality	5000	2000	3000
5	sales	5000	2000	3000
6	finance	5000	2000	3000

客户表单
成员表单
产品表单
订单表单
部门表单
总任务表单
子任务表单

图 70: 总经理视图——可以修改员工信息包括工资

图 71: 总经理视图——可以新增员工

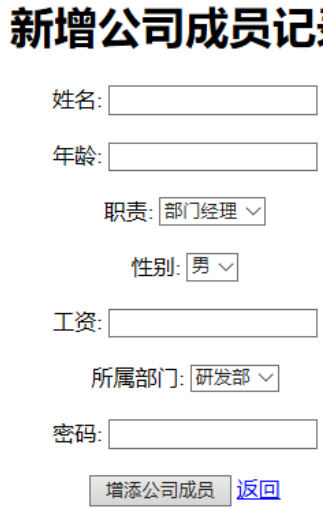


图 72: 公司产品概览

图 73: 客户视图——可以对产品产生购买行为

图 74: 开发部门经理视图——可以新增产品

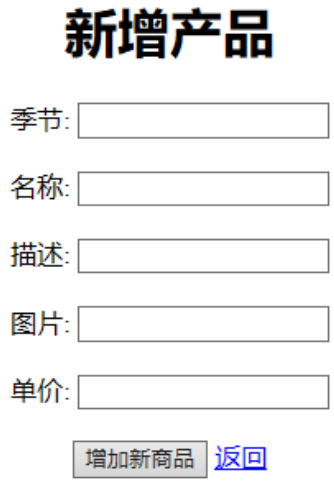


图 75: 生产部门经理视图——新增产品库存，但是不能修改产品信息

图 76: 总经理、部门经理视图——可管理公司订单

订单表单

订单号	商品号	客户号	数量	总价	完成情况	-
1	1	2	100	100000	0	删除
2	2	1	200	400000	0	删除
27	1	1	10	10000	0	删除
21	10	1	50	150000	0	删除
5	2	1	150	0	0	删除
6	2	1	150	0	0	删除
7	2	2	150	0	0	删除
8	2	1	150	0	0	删除
9	2	1	150	0	0	删除
10	2	1	150	0	1	删除
11	2	2	150	0	1	删除
26	10	1	50	150000	0	删除
16	2	1	20	40000	0	删除
28	8	1	50	100000	0	删除
15	2	1	20	40000	1	删除

图 77: 公司部门概况

部门表单

部门号	部门名	部门经理	部门地址	部门电话	-
1	develop	Amy	Guangzhou	1234567	修改
2	purchase	Baby	Shanghai	987654321	修改
3	produce	Daisy	Shanghai	917536482	修改
4	quality	Gaiya	Beijing	741852963	修改
5	sales	Jack	Beijing	123789456	修改
6	finance	Luna	Guangzhou	123456789	修改
7	HR	Sigama	hangzhou	753951462	修改

图 78: 总经理视图——可修改部门信息

修改公司部门信息

部门号: 1

部门名: develop

部门经理: Amy

部门地址: Guangzhou

部门电话: 1234567

修改公司部门信息 返回

图 79: 总经理视图——管理整个公司的项目和任务

总任务表单

任务编号	订单编号	开发部门完成情况	采购部门完成情况	生产部门完成情况	质检部门完成情况	销售部门完成情况	任务状态	-
8	18	0	0	0	0	0	0	删除
7	17	1	0	0	0	0	0	删除
3	12	1	1	0	0	0	0	删除
11	21	0	0	0	0	0	0	删除
12	22	1	0	0	0	0	0	删除
13	23	0	0	0	0	0	0	删除
16	26	0	0	0	0	0	0	删除
17	27	0	0	0	0	0	0	删除
18	28	0	0	0	0	0	0	删除
19	29	0	0	0	0	0	0	删除
20	30	0	0	0	0	0	0	删除
21	31	0	0	0	0	0	0	删除
22	32	0	0	0	0	0	0	删除

图 80: 总经理视图——管理整个公司的子任务

子任务表单

子任务编号	对应父任务编号	部门分工	员工1	员工1完成情况	员工2	员工2完成情况	子任务状态	-
1	2	1	2	1	25	1	1	修改
2	2	2	16	1 完成	16	1 完成	1	修改
3	2	3	16	1 完成	16	1 完成	1	修改
4	2	4	16	1 完成	16	1 完成	1	修改
5	2	5	16	1 完成	16	1 完成	1	修改
6	3	1	2	1	2	1	1	修改
7	3	2	16	1 完成	16	1 完成	1	修改
8	3	3	16	0 完成	16	0 完成	0	修改
9	3	4	16	0 完成	16	0 完成	0	修改
10	3	5	16	0 完成	16	0 完成	0	修改
11	4	1	16	0 完成	16	0 完成	0	修改
12	4	2	16	0 完成	16	0 完成	0	修改
13	4	3	16	0 完成	16	0 完成	0	修改

图 81: 总经理、部门经理视图——分配子任务给员工

分配子任务

子任务号: 1

任务号: 2

任务部门: develop

子任务1负责员工: 2 Debbie

子任务2负责员工: 25 Hani

分配子任务 返回

null

图 82: 开发部经理视图——管理开发部的项目，也可以为自己安排任务

子任务表单

子任务编号	对应父任务编号	部门分工	员工1	员工1完成情况	员工2	员工2完成情况	子任务状态	-
1	2	1	2	1	25	1	1	修改
6	3	1	2	1	2	1	1	修改
11	4	1	16	0	16	0	0	修改
16	5	1	16	0	16	0	0	修改
21	6	1	16	0	16	0	0	修改
26	7	1	2	1	16	1	1	修改
31	8	1	16	0	16	0	0	修改
36	9	1	16	0	16	0	0	修改
41	10	1	16	0	16	0	0	修改
46	11	1	16	1	16	0	0	修改
51	12	1	1	1 完成	3	1	1	修改
56	13	1	25	0	2	0	0	修改
61	14	1	2	0	1	1 完成	0	修改
66	15	1	3	1	3	1	1	修改
71	16	1	16	0	16	0	0	修改
76	17	1	16	0	16	0	0	修改
81	18	1	16	0	16	0	0	修改
86	19	1	16	0	16	0	0	修改
91	20	1	16	0	16	0	0	修改
96	21	1	16	0	16	0	0	修改
101	22	1	16	0	16	0	0	修改

图 83: 开发部员工视图——查询、更新自己被安排到的任务，不能更改一起协作的员工完成情况

子任务表单

子任务编号	对应父任务编号	部门分工	员工1	员工1完成情况	员工2	员工2完成情况	子任务状态
1	2	1	2	1 完成	25	1	1
6	3	1	2	1 完成	2	1 完成	1
26	7	1	2	1 完成	16	1	1
44	10	4	2	0 完成	3	1	0
45	10	5	2	0 完成	2	0 完成	0
56	13	1	25	0	2	0 完成	0
61	14	1	2	0 完成	1	1	0

客户表单
成员表单
产品表单
订单表单
部门表单
总任务表单
子任务表单

4.5 源代码

在附件中提交全部代码，包含：数据库(*.sql)代码、UI(*.jsp)代码