

**山东大学（威海）**

**物流综合管理系统结构化设计**

年级：19级 \_\_\_\_\_\_

专业：计算机科学与技术（中澳合作）

作者：赵旭东 徐晏辰 钱克洋\_\_\_\_\_\_\_\_

指导老师：王文玉 \_\_\_\_\_\_\_

编写日期：2021 年 4 月 24 日

目录

[一．引言 4](#_Toc70178460)

[二．体系结构设计 4](#_Toc70178461)

[2.1 改善后的数据流图 4](#_Toc70178462)

[2.2 系统模块划分 8](#_Toc70178463)

[2.2.1 寄售件管理子系统功能需求 8](#_Toc70178464)

[2.2.2 营业厅管理子系统功能需求 9](#_Toc70178465)

[2.2.3 中转中心子系统功能需求 10](#_Toc70178466)

[2.2.4 财务及人事管理子系统 10](#_Toc70178467)

[2.3 系统架构图 11](#_Toc70178468)

[2.4 系统结构图 11](#_Toc70178469)

[三． 结构设计 21](#_Toc70178470)

[3.1 人机交互接口设计 21](#_Toc70178471)

[3.2 模块间接口设计 38](#_Toc70178472)

[3.2.1 数据库连接初始化 38](#_Toc70178473)

[3.2.2 调用数据库增删改功能 41](#_Toc70178474)

[3.2.3 数据库查找功能 42](#_Toc70178475)

[3.3 服务器与客户端间通信 43](#_Toc70178476)

[四．数据设计 49](#_Toc70178477)

[4.1 调整后的实体-关系图 49](#_Toc70178478)

[4.2 关系模式的建立 50](#_Toc70178479)

[4.3 数据库中表的实现 52](#_Toc70178480)

[机构表 52](#_Toc70178481)

[账户表 52](#_Toc70178482)

[收件人表 52](#_Toc70178483)

[寄件人表 53](#_Toc70178484)

[寄件单表 53](#_Toc70178485)

[快递员表 54](#_Toc70178486)

[快递员收款单表 54](#_Toc70178487)

[营业厅业务员表 55](#_Toc70178488)

[派件单表 55](#_Toc70178489)

[营业厅装车单表 55](#_Toc70178490)

[车辆代号表 56](#_Toc70178491)

[营业厅到达单表 56](#_Toc70178492)

[司机信息表 56](#_Toc70178493)

[中转中心中转单表 57](#_Toc70178494)

[中转中心到达单表 57](#_Toc70178495)

[中转中心业务员表 58](#_Toc70178496)

[中转中心装车单表 58](#_Toc70178497)

[中转中心库存管理人员表 59](#_Toc70178498)

[付款单表 59](#_Toc70178499)

[仓库表 59](#_Toc70178500)

[入库单表 60](#_Toc70178501)

[出库单表 60](#_Toc70178502)

[总经理表 61](#_Toc70178503)

[快递表 61](#_Toc70178504)

[单据表 61](#_Toc70178505)

[管理员表 62](#_Toc70178506)

[价格距离表 62](#_Toc70178507)

[员工表 62](#_Toc70178508)

[财务记录表 63](#_Toc70178509)

[4.4 建表语言 63](#_Toc70178510)

[五．过程设计 72](#_Toc70178511)

[5.1 主要功能模块的程序流程图 72](#_Toc70178512)

[5.2 重要算法的伪代码表述 77](#_Toc70178513)

# 一．引言

结构化设计方法是在模块化、自顶向下逐步细化及结构化程序设计技术基础之上发展起来的。在管理上通常将软件设计划分为总体设计和详细设计两个阶段。结构化设计是面向数据流的设计以需求分析阶段产生的数据流图为基础，按一定的步骤映射成软件结构。结构化设计方法是目前使用最广泛的软件设计方法之一，他是建立良好程序结构的方法，提出了衡量模块质量的标准是“高内聚，低耦合”。

概要设计阶段的主要任务是通过详细分析软件需求规格说明，适当地对软件进行功能分解，从而把软件划分为模块，并且设计出完成预定功能的模块结构。

详细设计阶段详细地设计每个模块，确定完成每个模块功能所需要的算法和数据结构。

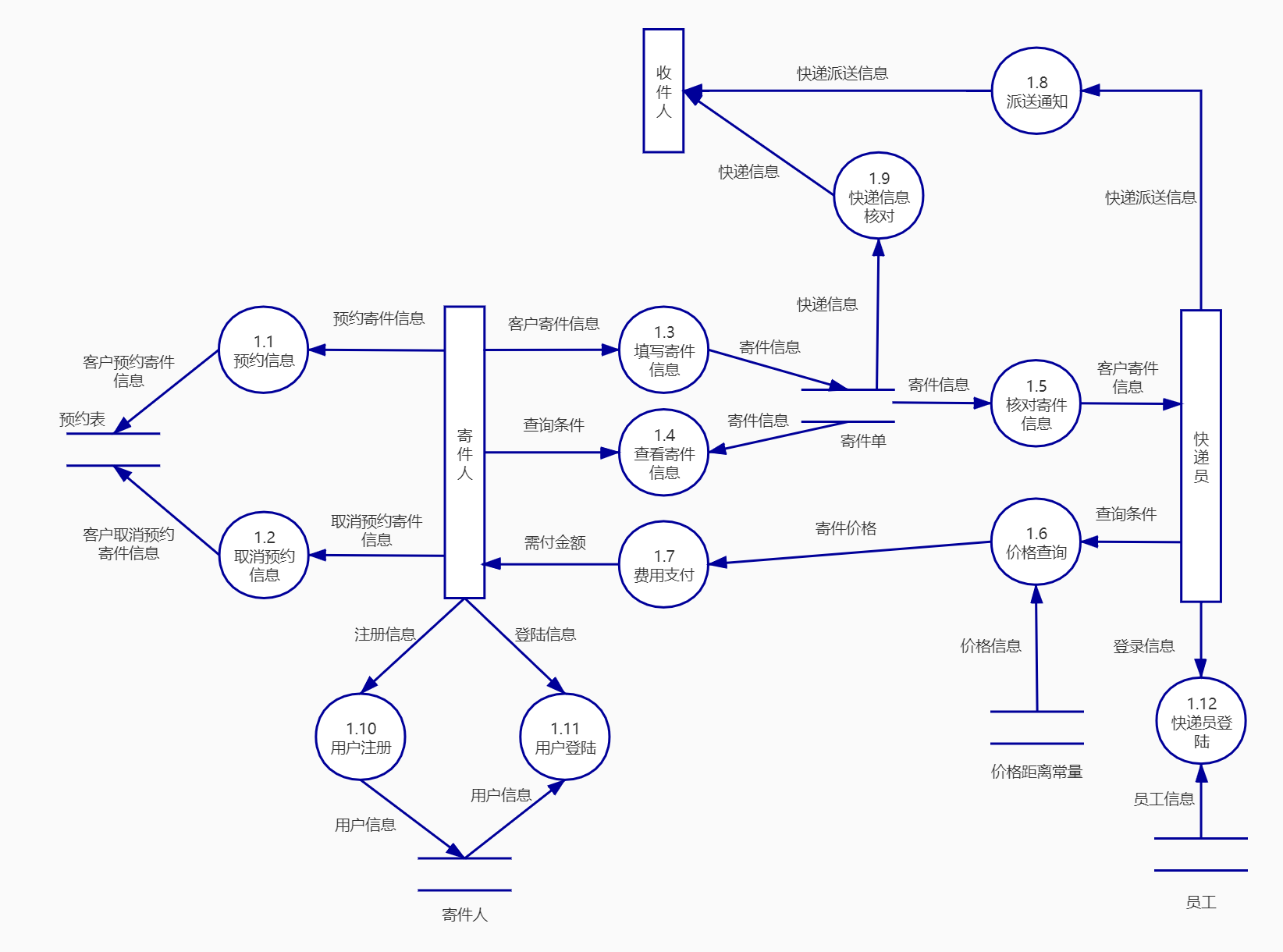
结构化设计既能与软件需求分析的结构化分析方法相衔接，可以方便的将数据流图转换为程序结构的设计描述，并且他还能与编码阶段的“程序化结构设计相适应”，起到了有需求分析到编码的承接作用。我们将继续修改数据流图使他更加完善合理，并且在数据流图的基础上分析得到程序结构的详细设计，做出接口设计，努力完成本次文档的编写。

# 二．体系结构设计

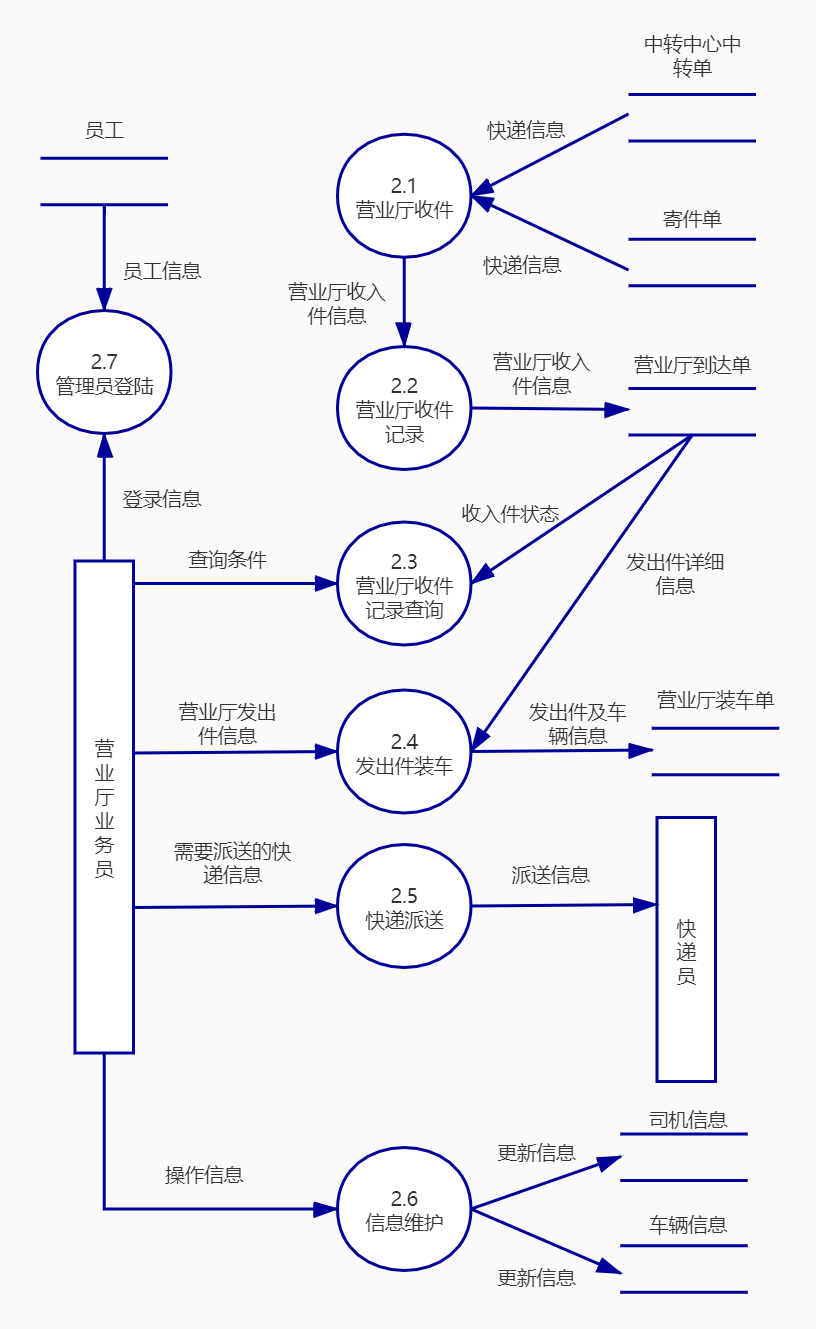
## 2.1 改善后的数据流图

**在需求分析阶段已完成顶层数据流图，一层数据流图，二层数据流图和三层数据流图，在此阶段对二层数据流图进行部分完善。**

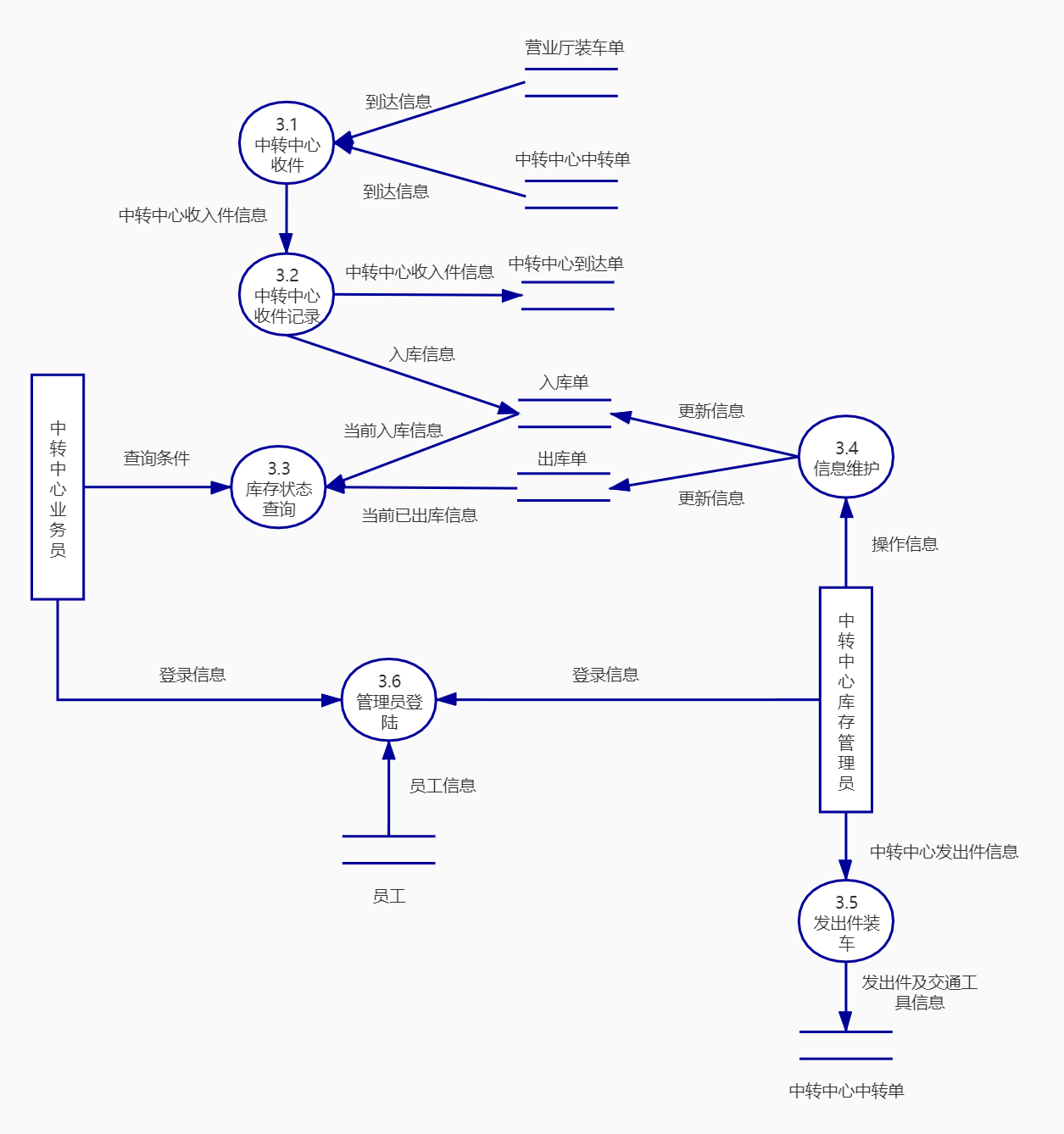
**（1）寄收件管理子系统数据流图**

****

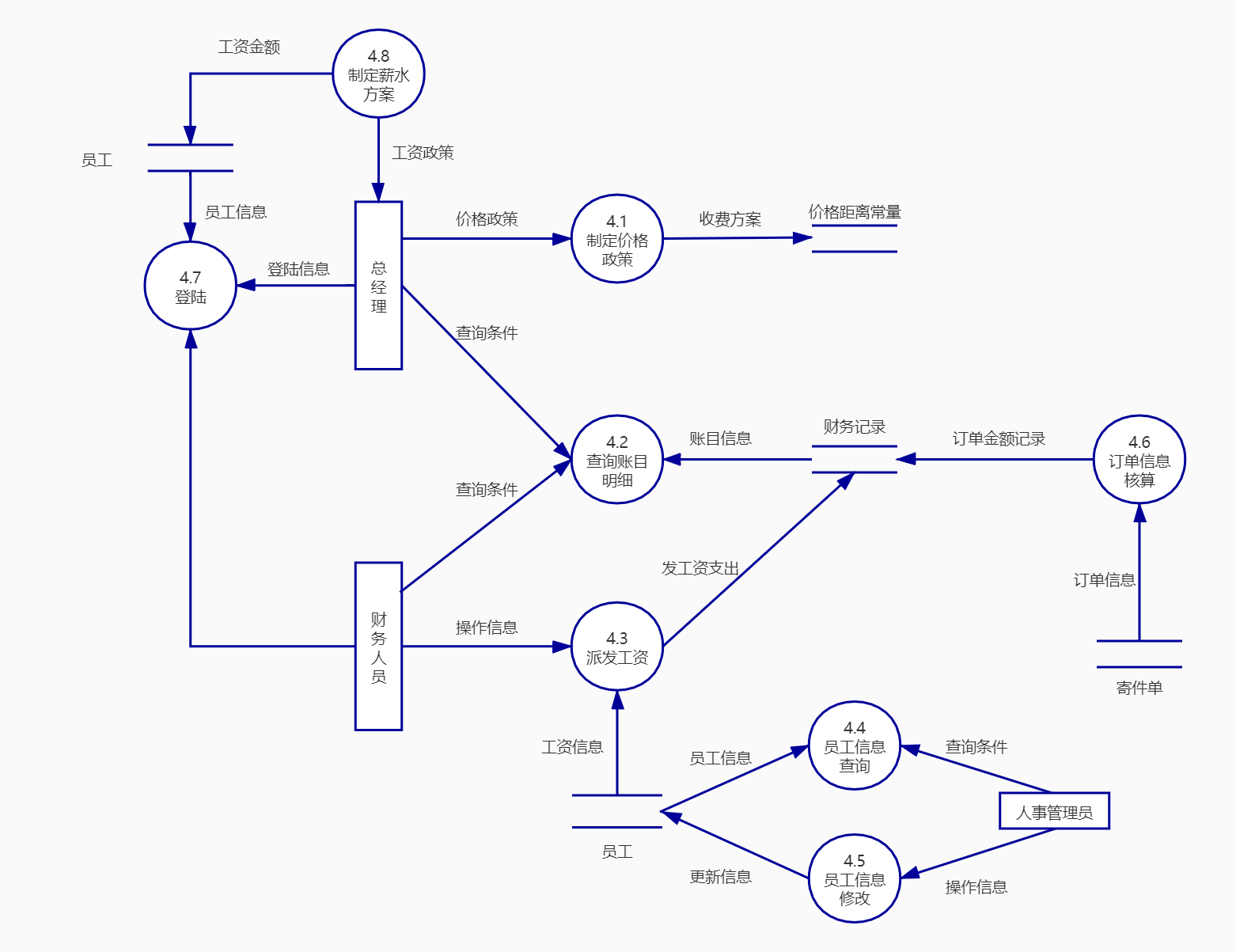
**（2）营业厅管理子系统数据流图**

****

**（3）中转中心管理子系统数据流图**

****

（4）财务及人事管理子系统

****

## 2.2 系统模块划分

我们最终将系统划分为4个子系统，每个子系统功能如下：

### 2.2.1 寄售件管理子系统功能需求

#### 客户预约寄件：

寄件人客户可以通过客户端进行预约快递员取件，时间和地点等信息可以在一定范围内自由选择，同时用户也可以取消之前预约。

#### 客户查看寄件信息：

寄件人客户和收件人客户均可以通过寄件信息查询到当前自己寄出或准备收到的快递的详细信息。

#### 客户寄件处理：

寄件人客户到达预约地点后，快递员辅助客户填写寄件表，并完成快递的封装和测量，依据公司目前的价格政策给客户报价并完成收费。

#### 快递员派件：

快递员接到派件任务将快递物品送到收件人手中，收件人核对后签收。

#### 收件人收件：

收件人在接收到快递员的派送信息后等待收件，在确定寄件信息无误后签收。

### 2.2.2 营业厅管理子系统功能需求

#### 营业厅收件：

营业厅从快递员手中或者从中转中心发来的车辆中收得快递，并录入到系统中。

#### 营业厅发件至中转中心

营业厅收件后并记录后，将其装车送至下一个中转中心。

#### 营业厅向快递员发送派件信息：

营业厅业务员会将需要派送的快递信息送至区域内的快递员，由快递员进行派送。

#### 营业厅维护司机和车辆信息

营业厅业务员可通过增加、删除、修改等操作对司机和车辆信息进行维护。

### 2.2.3 中转中心子系统功能需求

#### 中转中心收件：

中转中心会从其他中转中心或最近的一个营业厅收到发来的快递，中转中心在收到后会自动录入系统并录入仓库。

#### 中转中心收件信息监视与统计：

中转中心业务员负责对中转中心收到的快递信息出入库情况进行监视与统计。

#### 中转中心信息维护：

中转中心仓库管理员负责对入库和出库单进行维护，可以对出入库单进行增加，删除，修改等操作。

#### 中转中心发件：

中转中心仓库管理员控制中转中心的发件情况，在交通工具满足的情况下优先发出先到库的快递，发出的快递将依据目的地发往下一个中转中心或最近的营业厅。

### 2.2.4 财务及人事管理子系统

#### 公司高层指定价格政策：

公司高层可以依据当前物流经济形势指定公司的收费方案，价格距离常量将直接影响公司的业务收费。

#### 查询财务情况：

公司高层和财务人员均有权限查询公司的账目明细。并生成报表。

#### 派发工资：

公司财务人员在公司规定的时间范围内有派发工资的权限，在派发工资后计入公司的财务记录.

#### 订单结算：

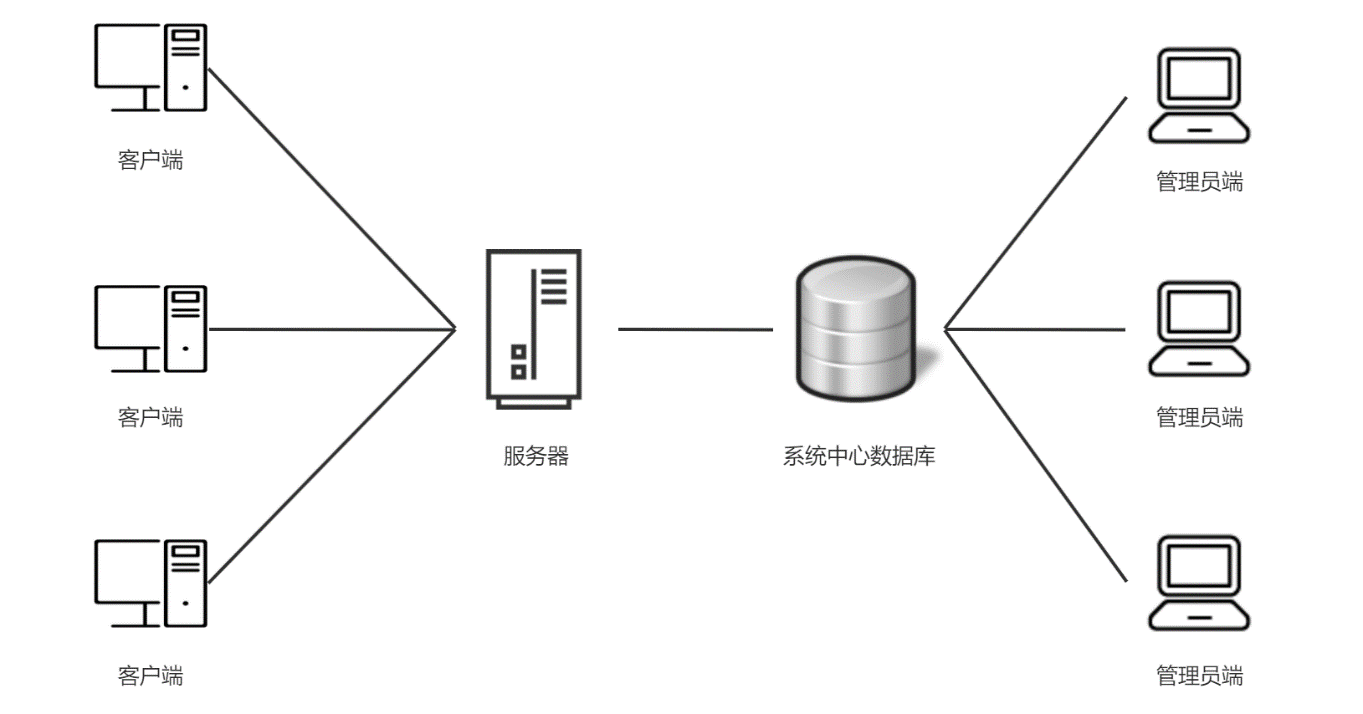
公司总部会将各地的寄件单汇总并计入公司的财务记录。

#### 员工信息维护：

公司总部的人事管理员可通过增加、删除、修改等操作对公司员工信息进行维护。

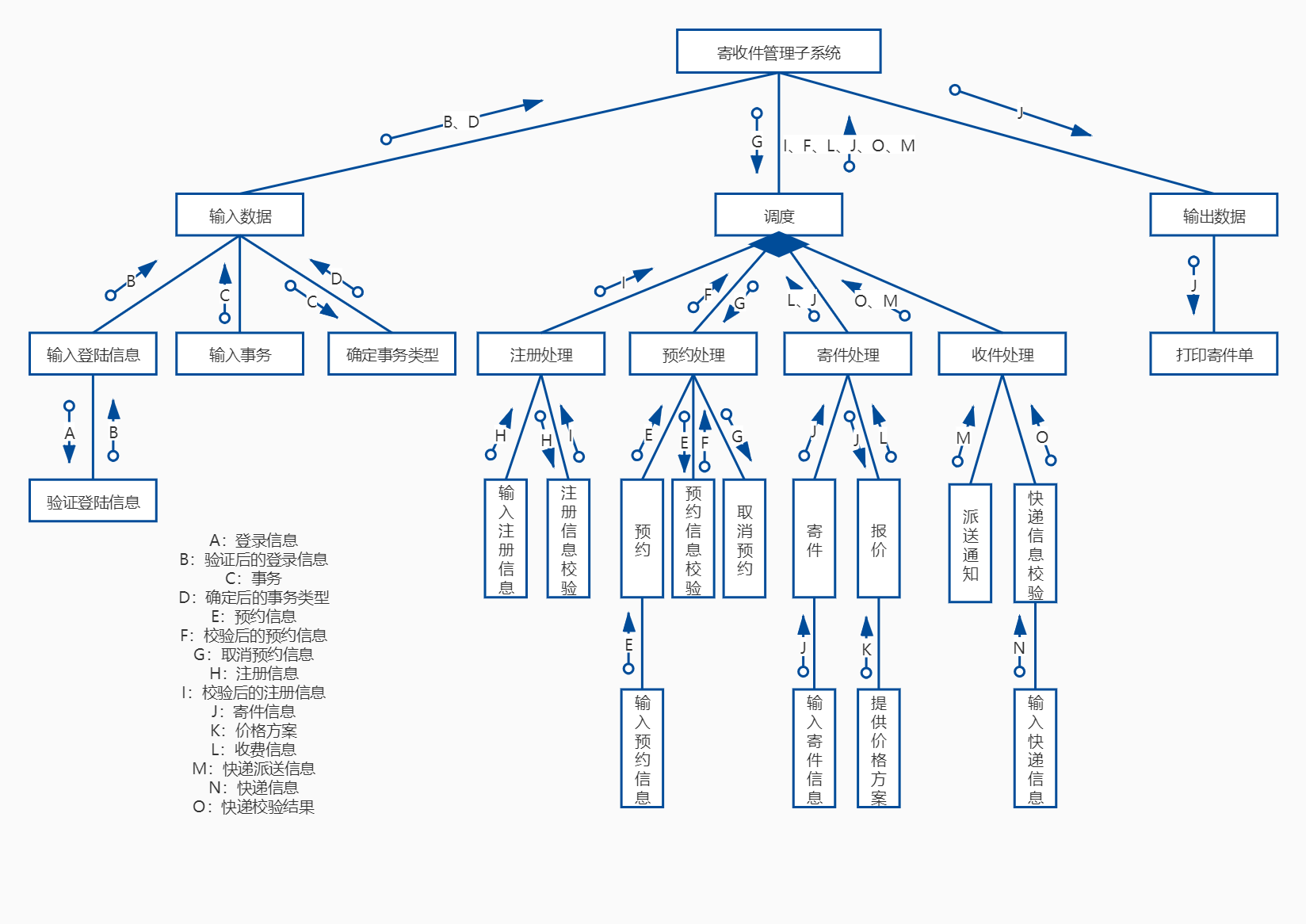
## 2.3 系统架构图

物流公司综合管理系统采用C/S架构，分为管理员端和客户端。在管理员端，不同的管理员对应不同权限功能。在客户端，客户可以实现预约寄件等功能。

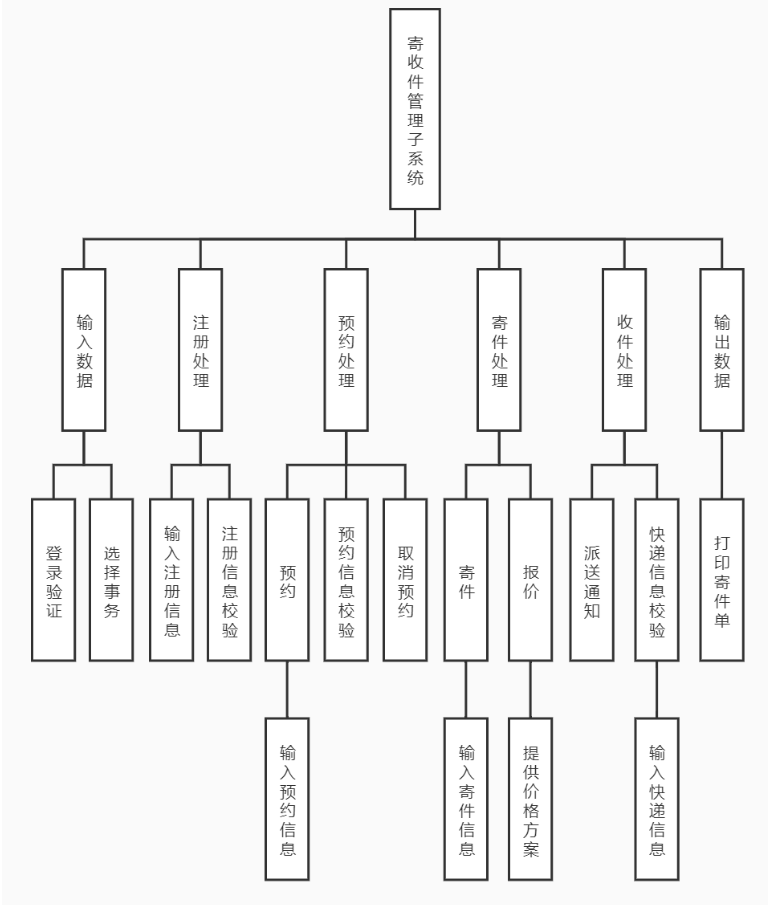


## 2.4 系统结构图

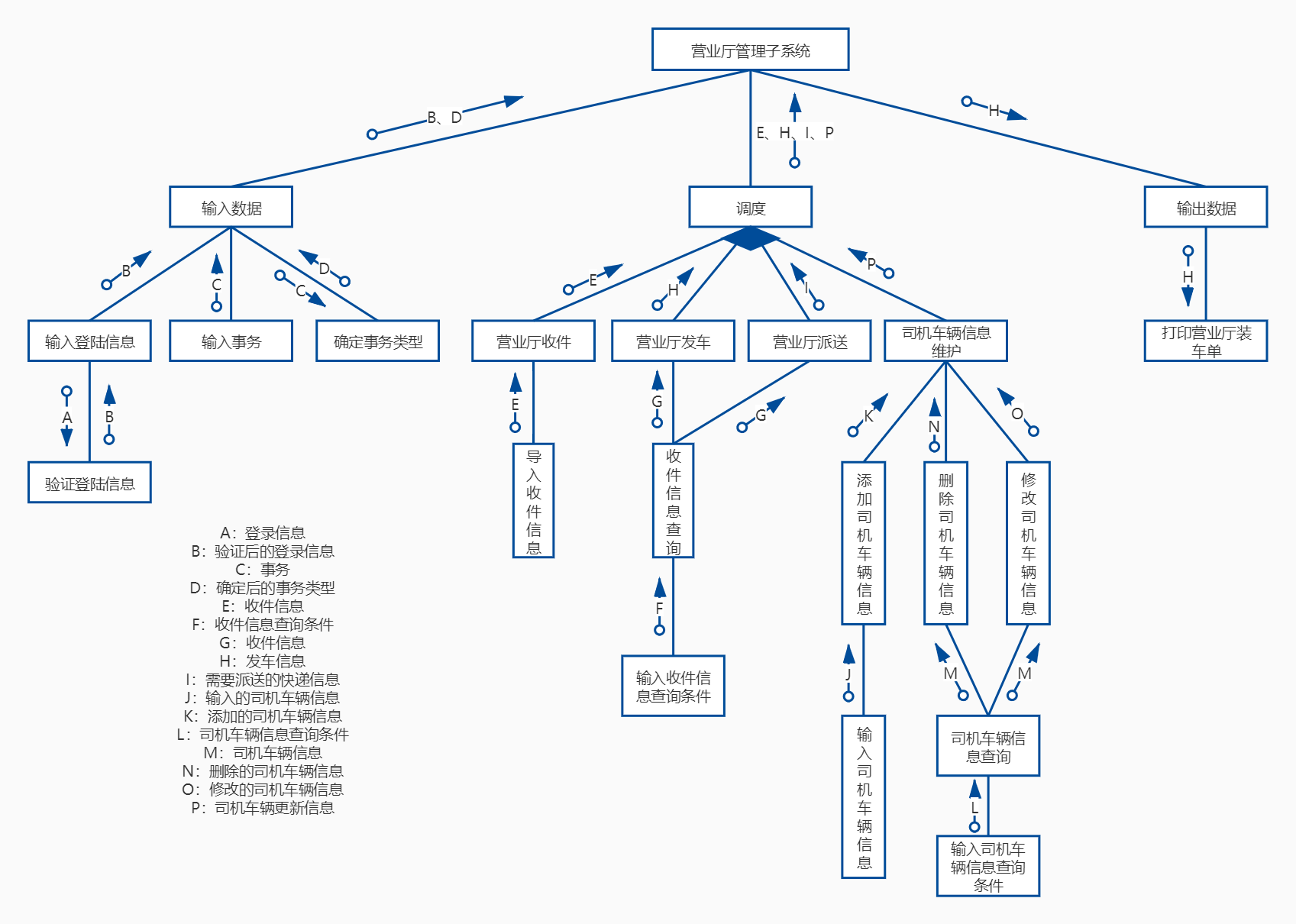
**（1）寄收件管理子系统的系统结构初始图**

****

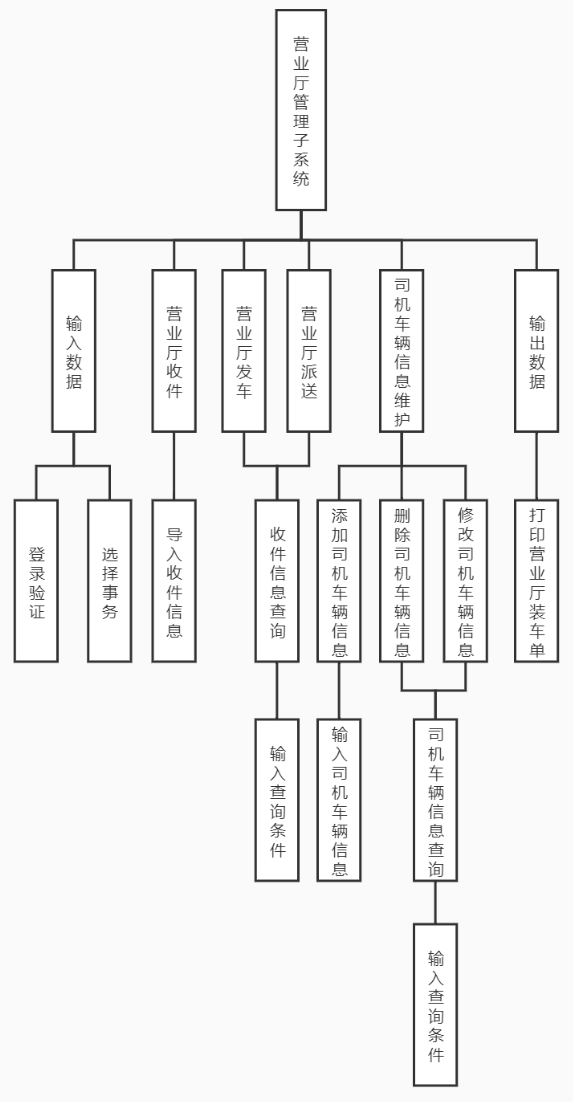
**优化后的系统结构图**

****

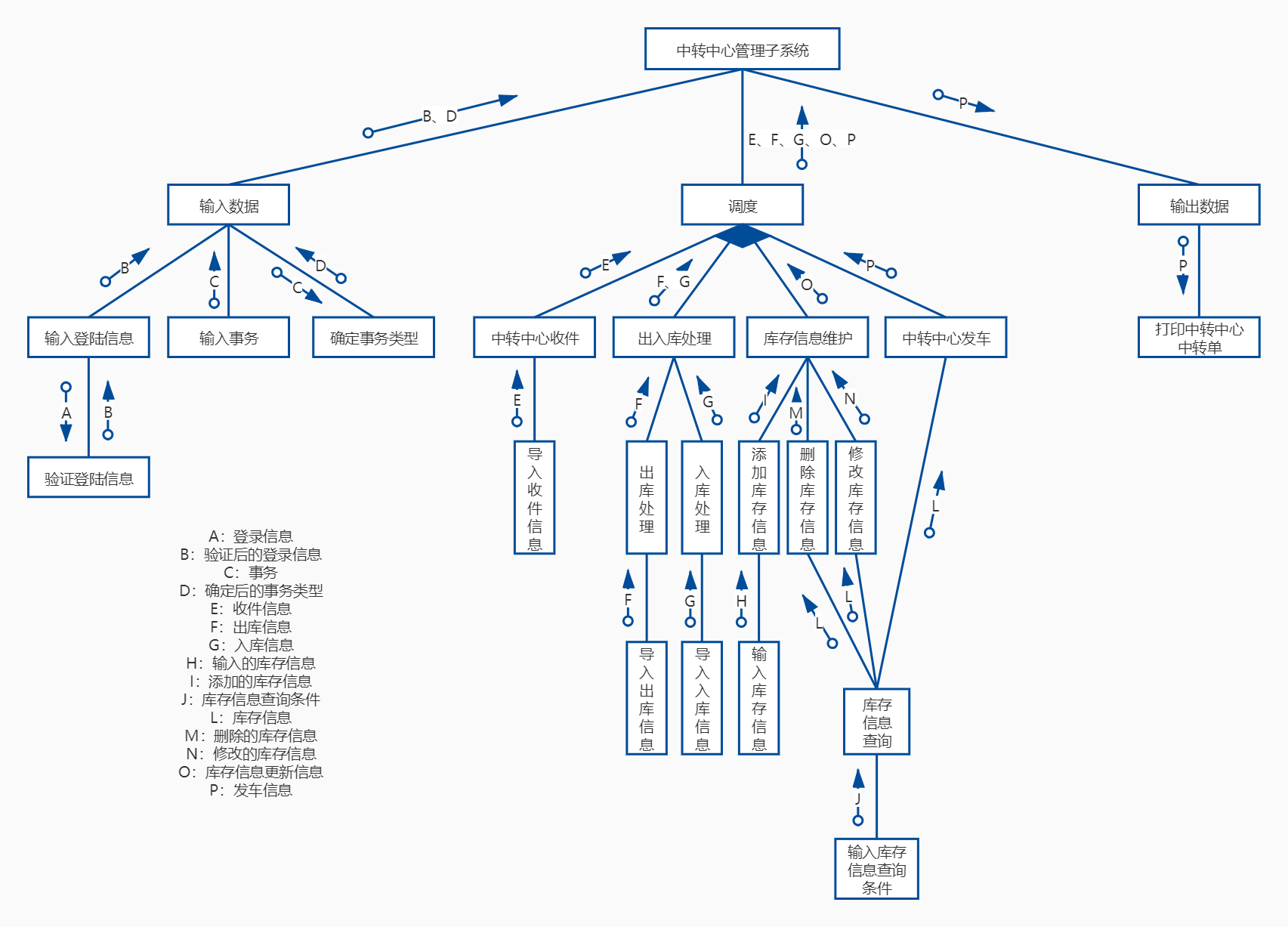
**（2）营业厅管理子系统的系统结构初始图**

****

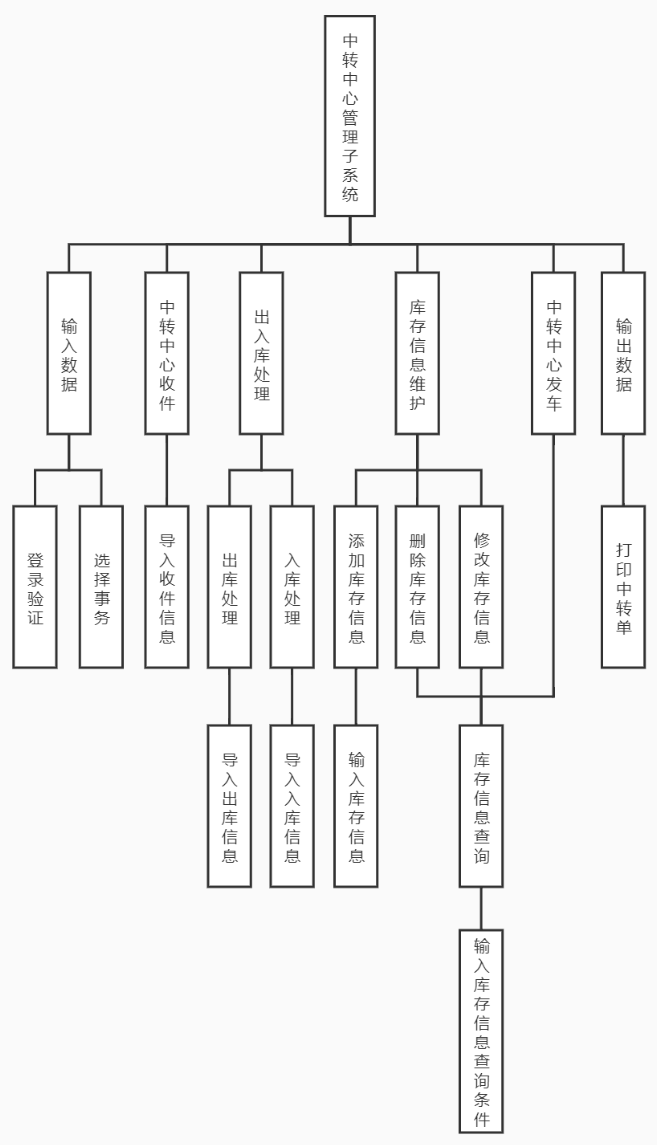
**优化后的系统结构图**

****

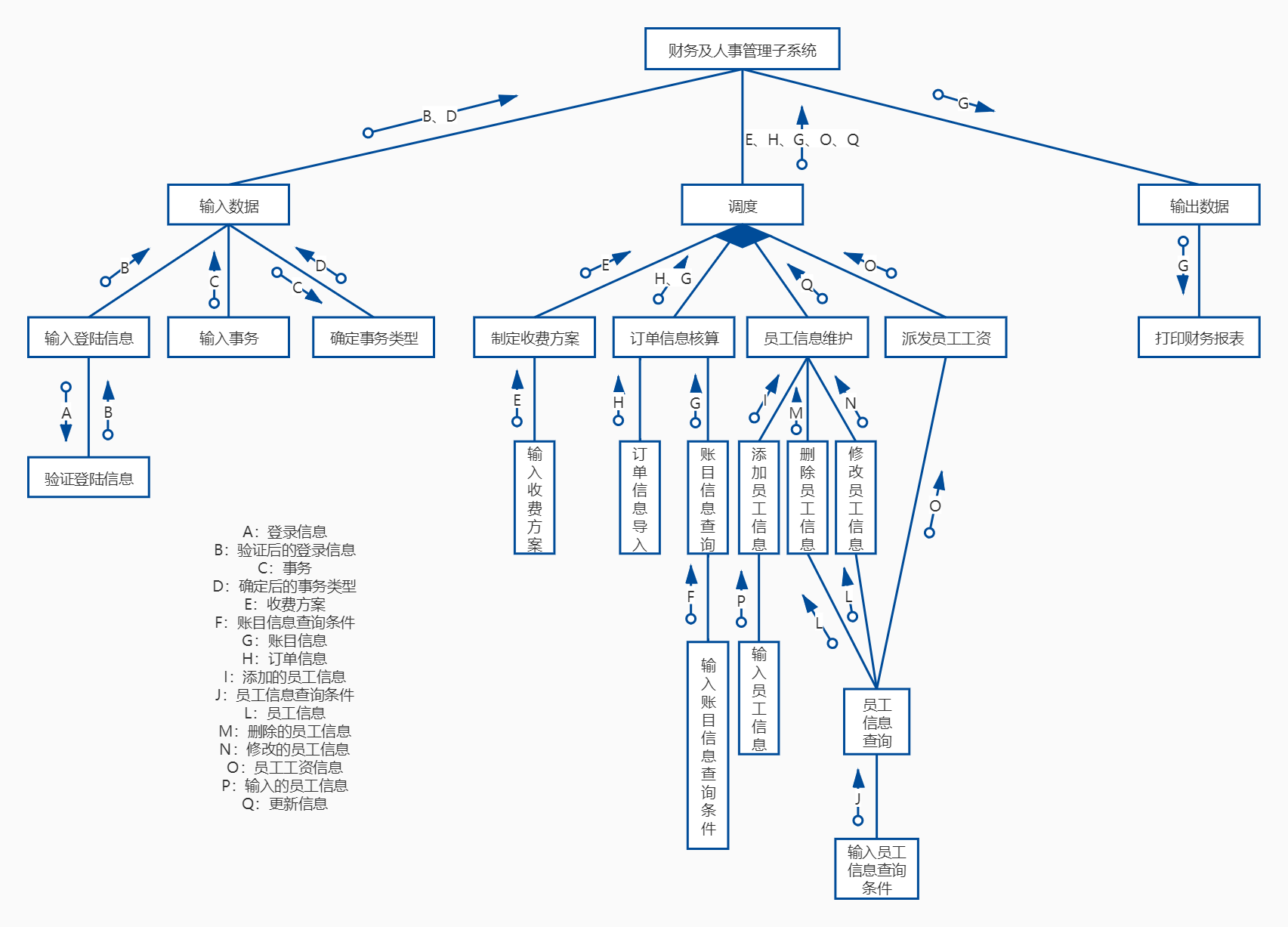
**（3）中转中心管理子系统的系统结构初始图**

****

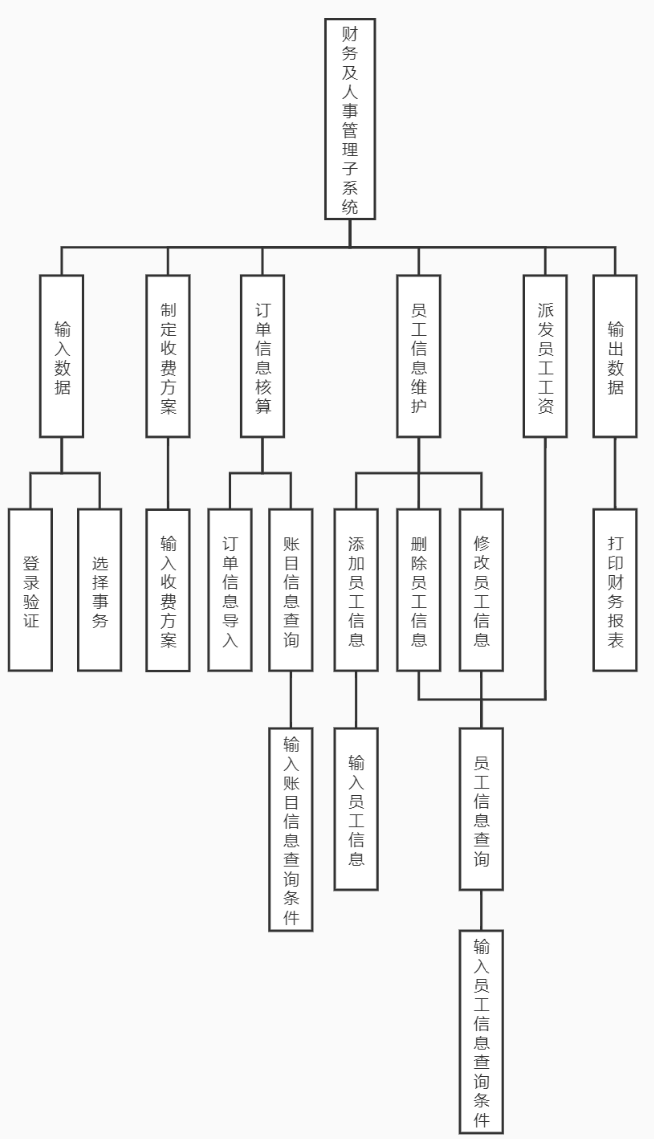
**优化后的系统结构图**

****

**（4）财务及人事管理子系统的系统结构初始图**

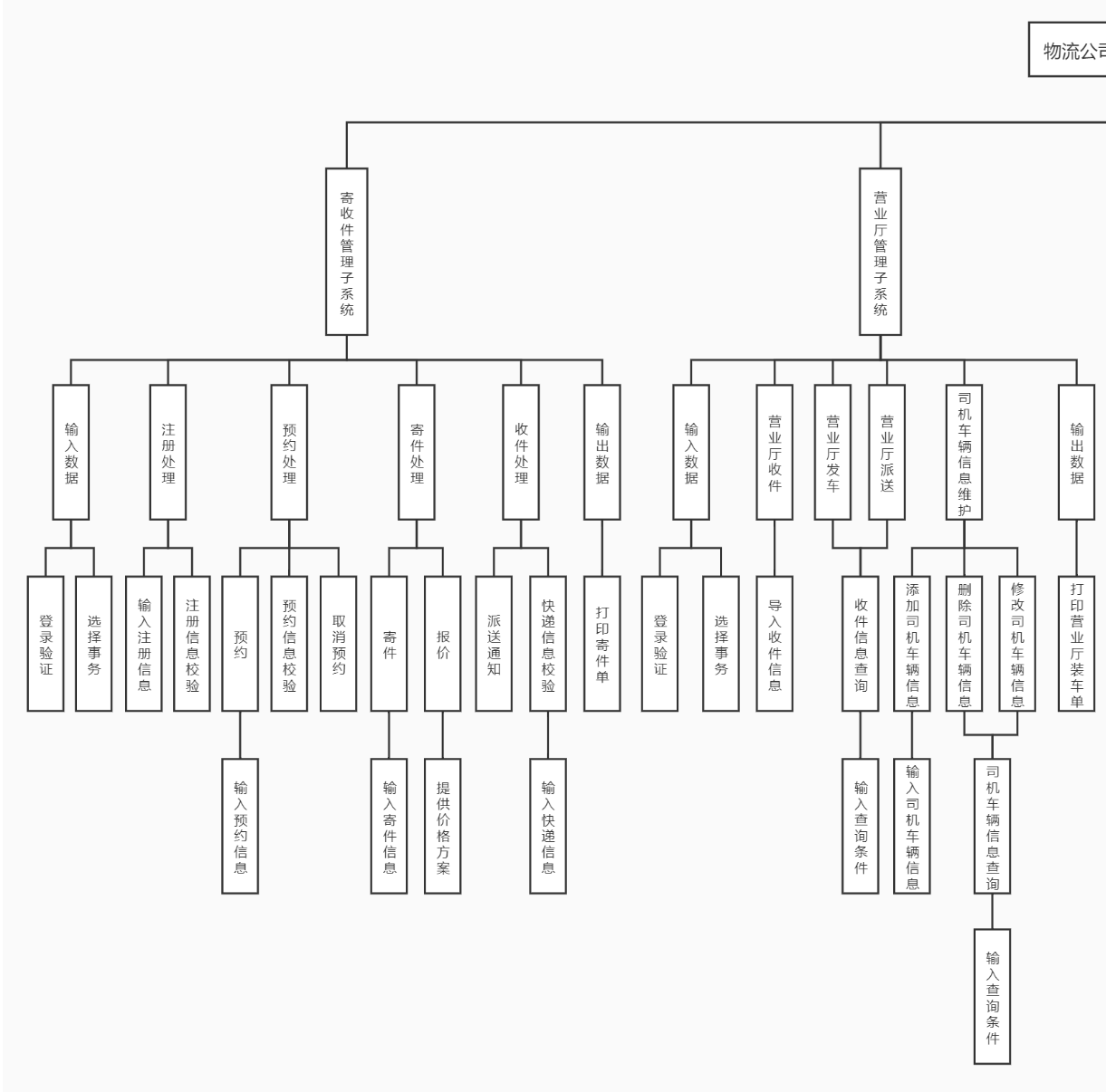
****

**优化后的系统结构图**

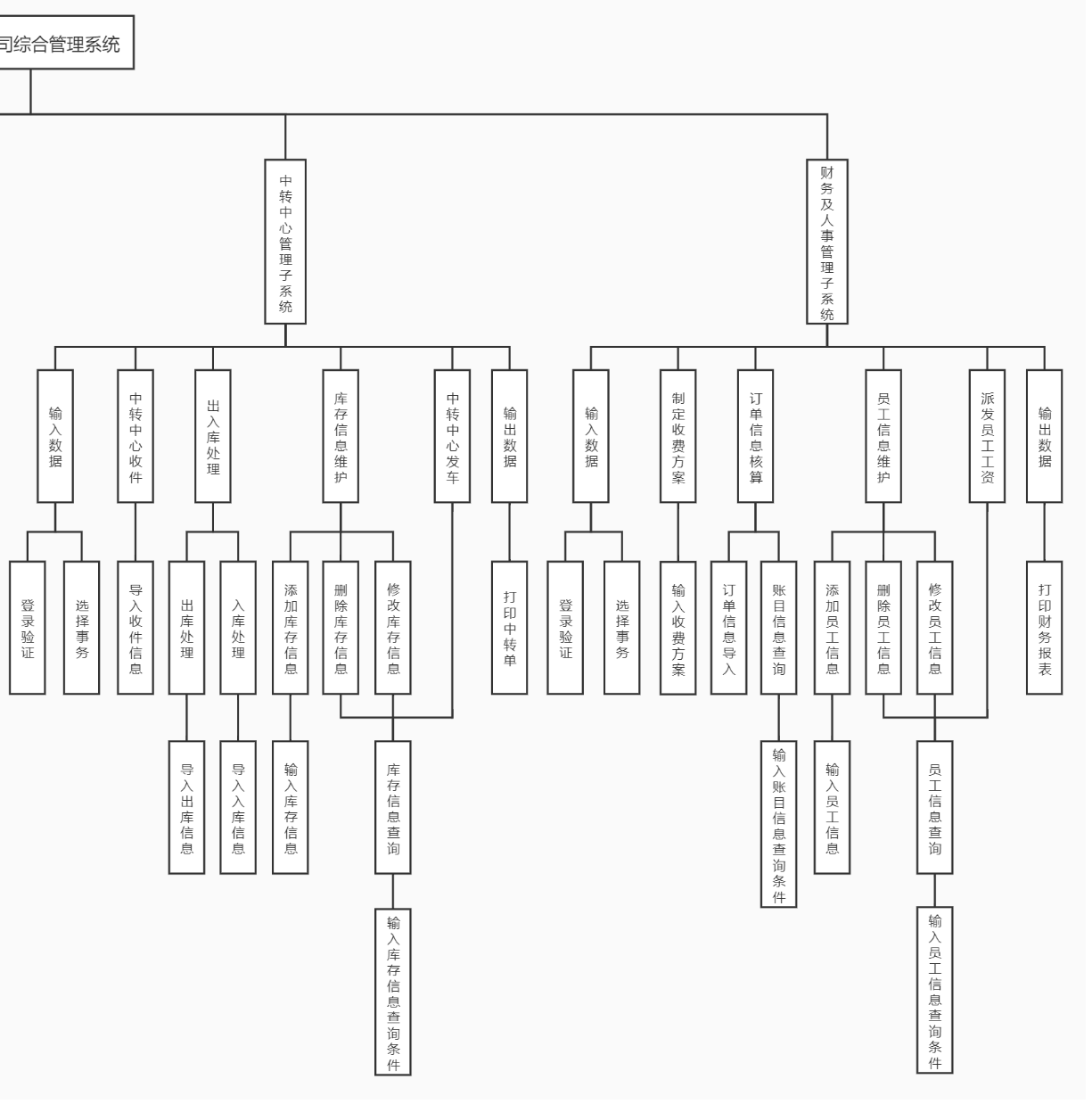
****

**（5）总体系统结构图**

**（左侧两个子系统）**

****

**（右侧两个子系统）**

****

# 结构设计

## 3.1 人机交互接口设计

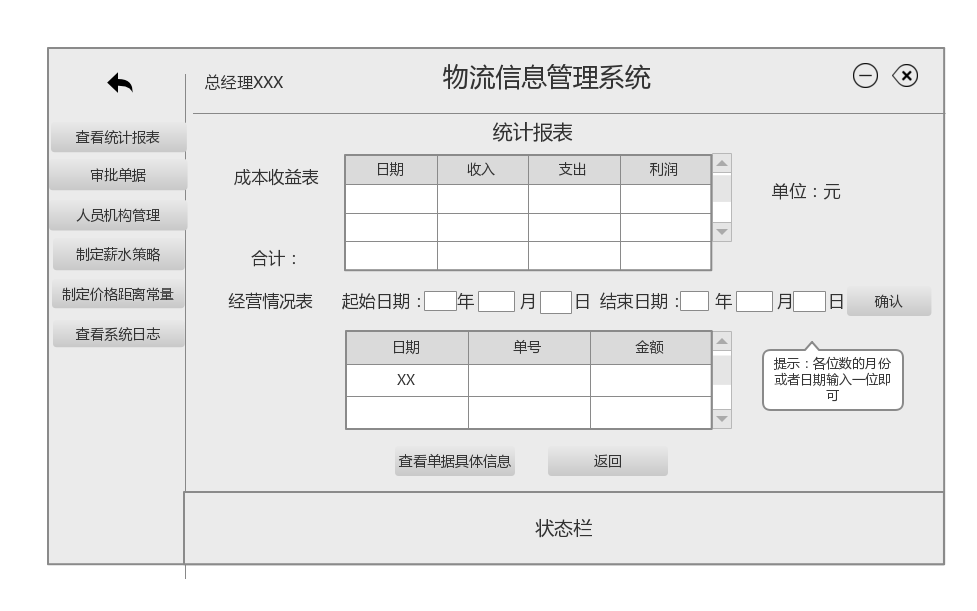
主界面设计：



Login界面设计：



统计报表查看界面：



系统日志查看界面：



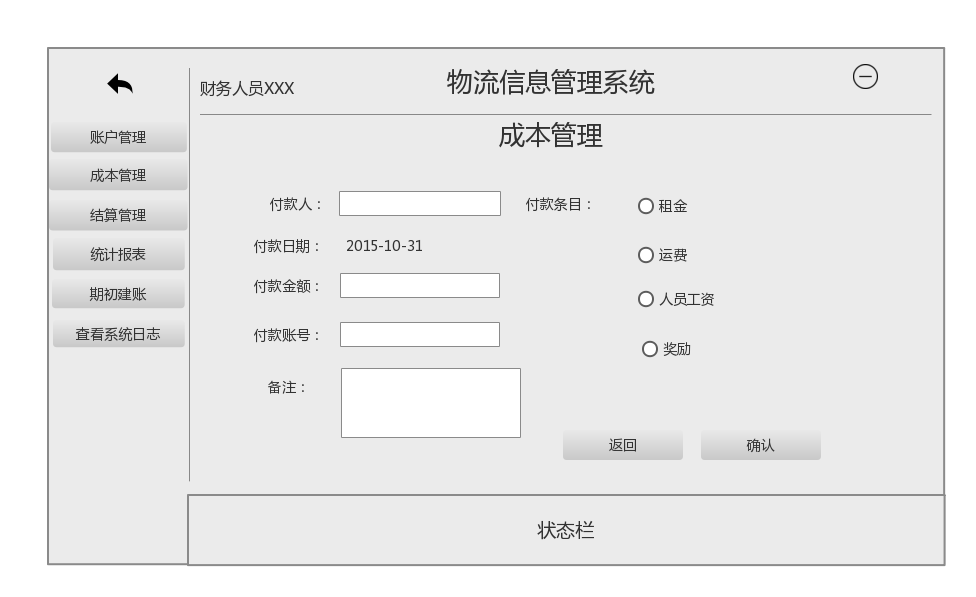
车辆信息管理界面：



车辆装车管理界面：



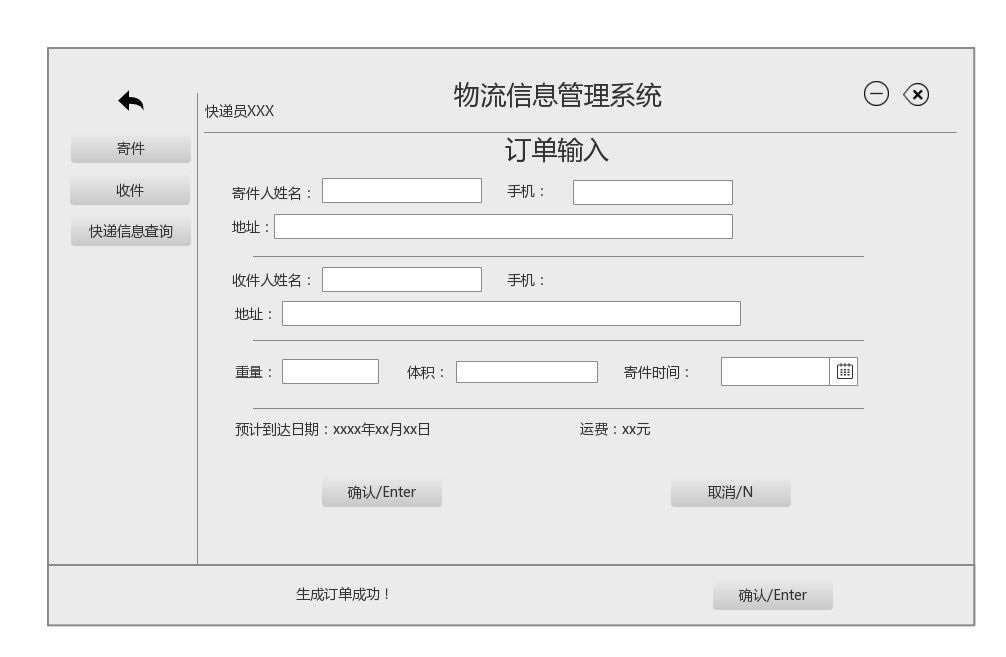
成本管理界面：



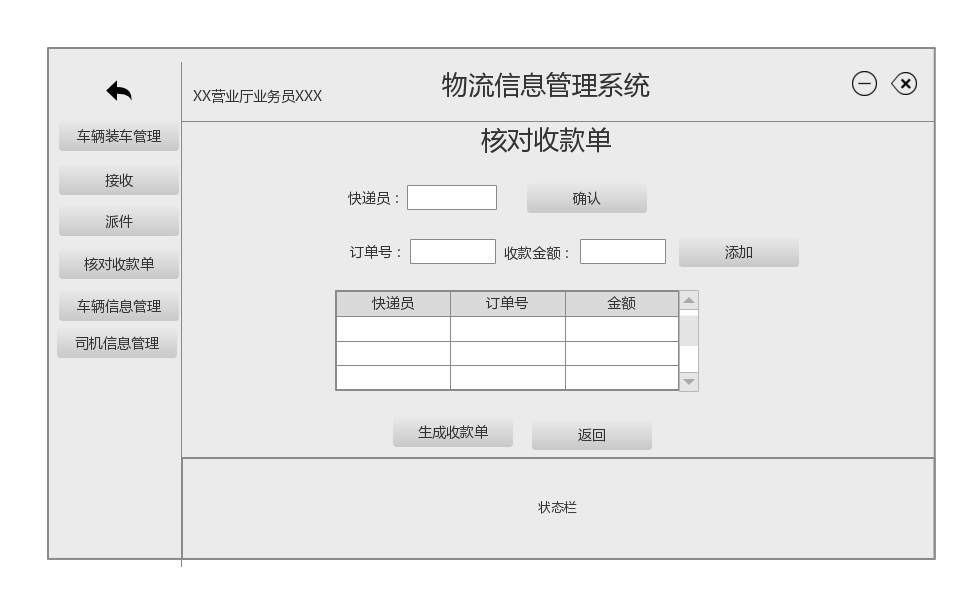
出库管理界面：



订单输入界面：



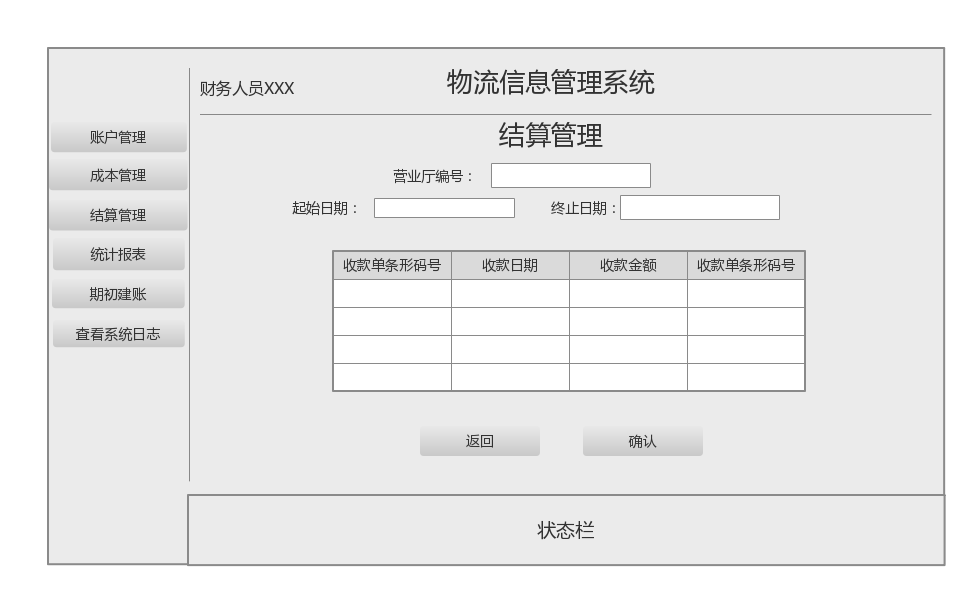
核对收款单界面：



接收界面：



结算管理界面：



库存查看界面：



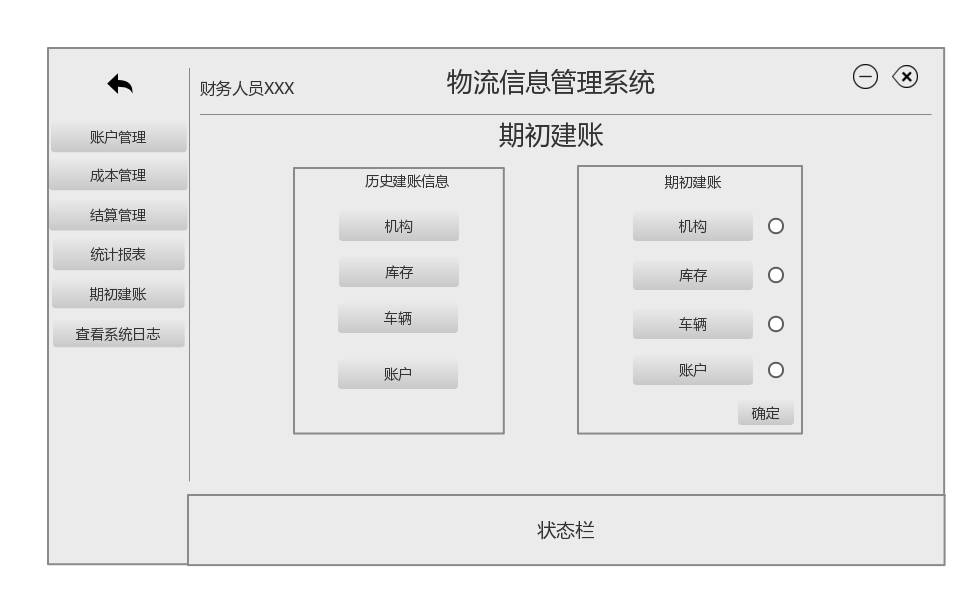
库存盘点界面：



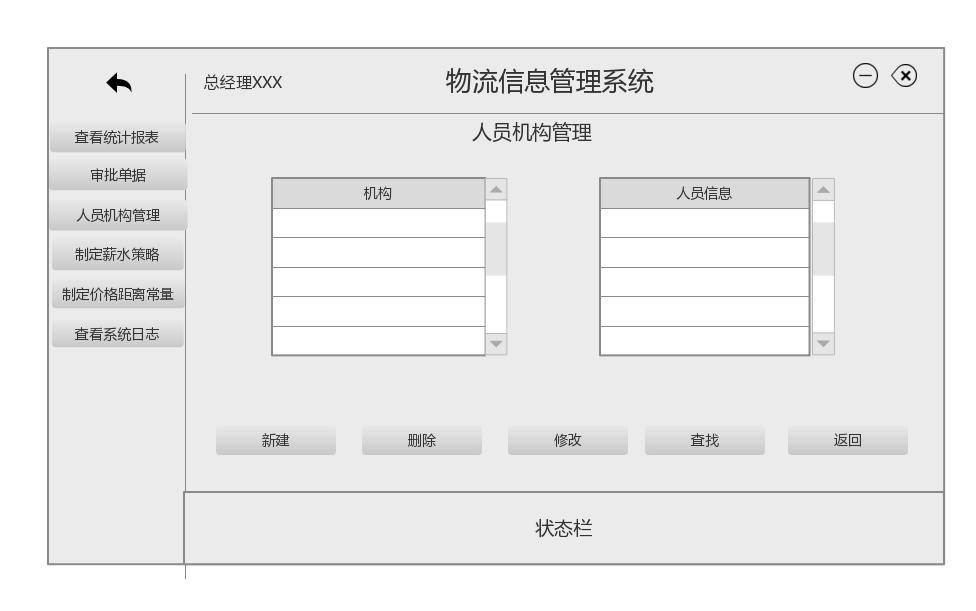
派件界面：



期初建账界面：



人员机构管理界面：



入库管理界面：



审批单据界面：



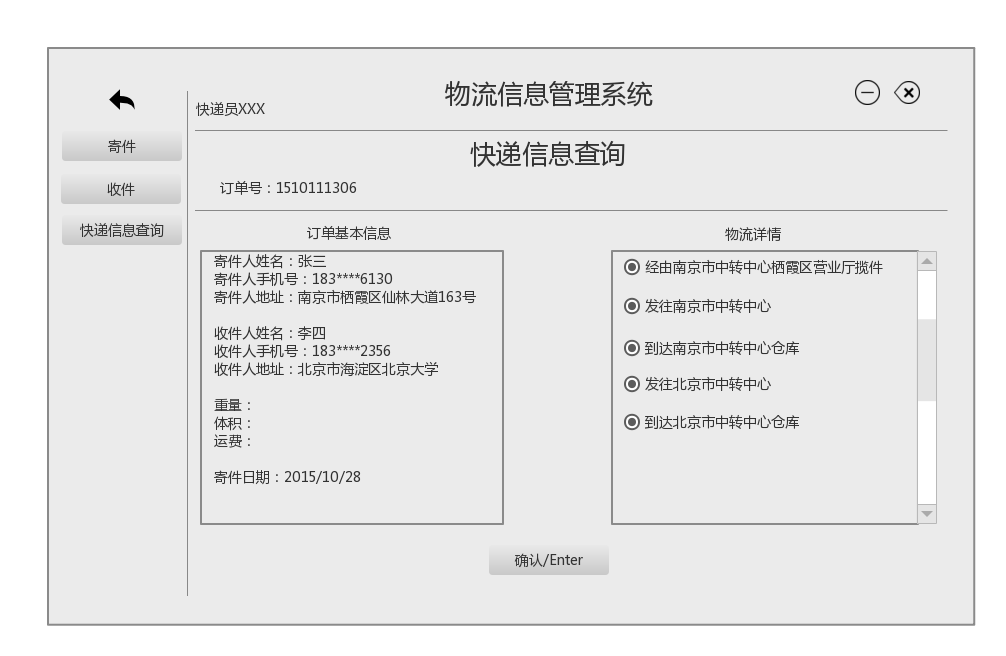
收件处理界面：



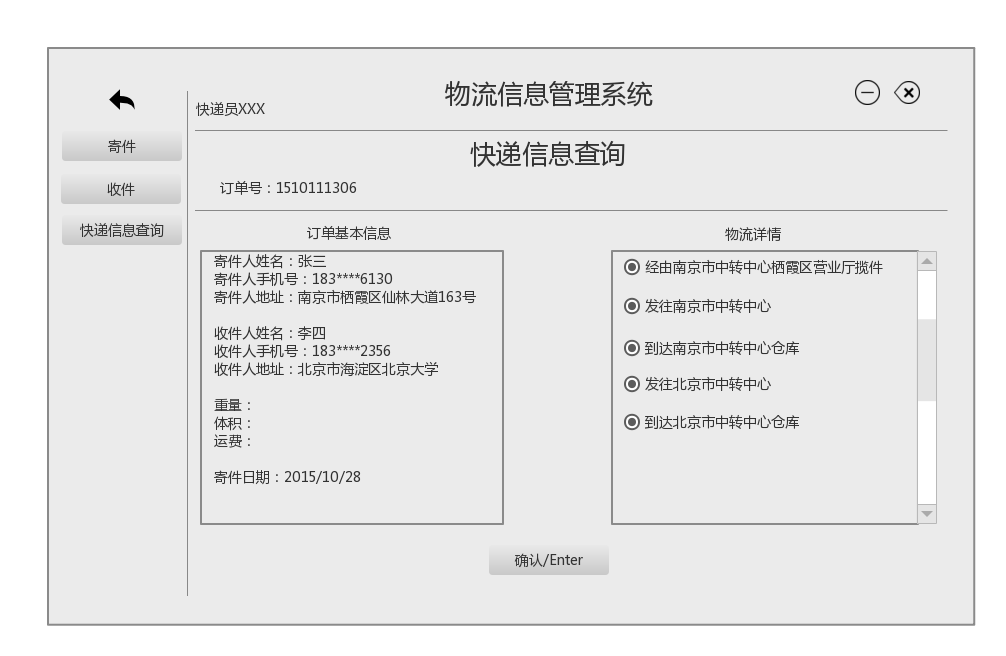
司机信息管理界面：



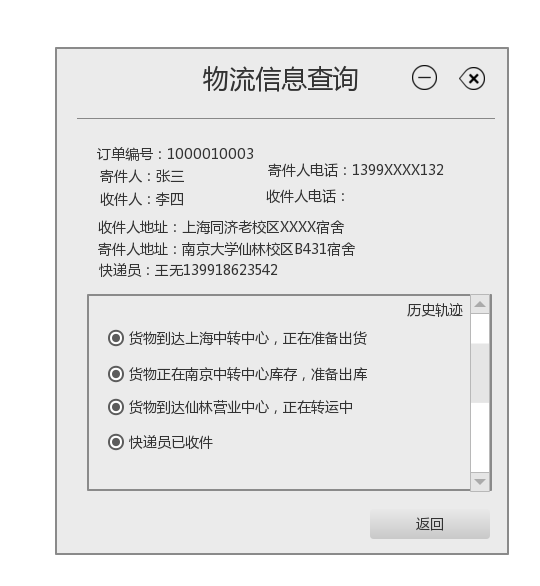
物流查询界面：



快递员物流信息查询界面：



寄件人物流信息查询界面：



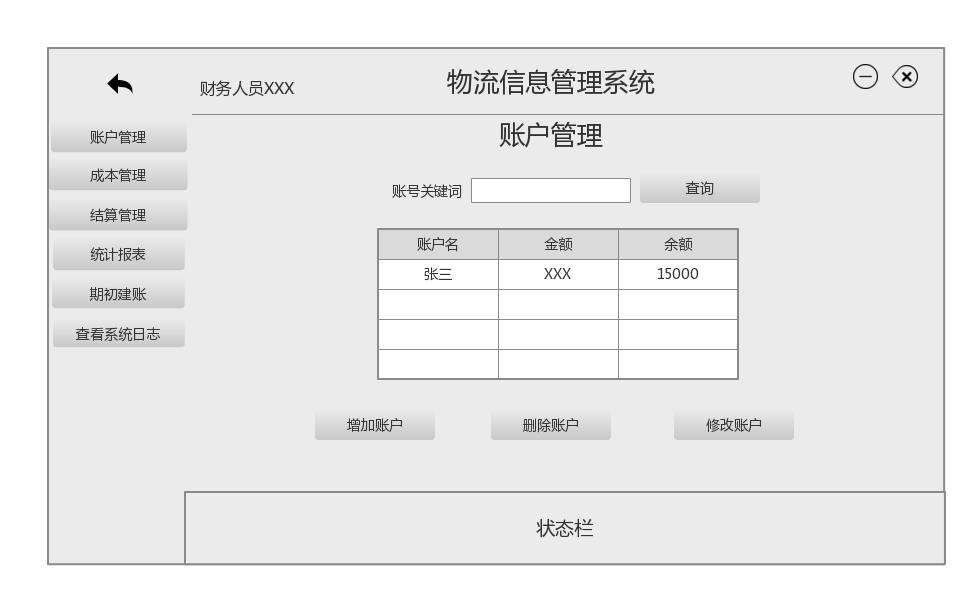
物流信息查询输入订单界面：



用户管理界面：



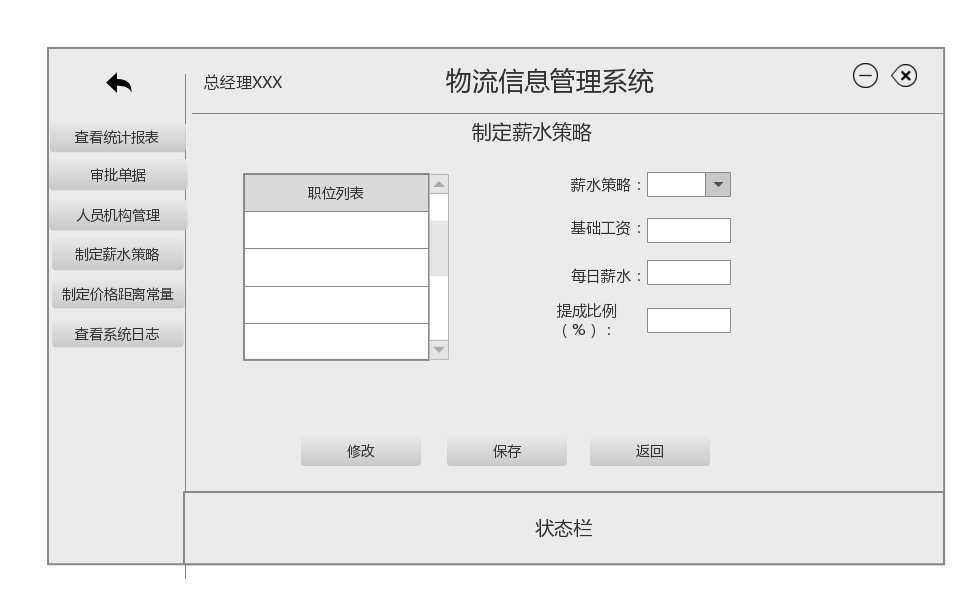
账户管理界面：



高层制定价格距离常量界面：



高层制定薪水策略界面：



中转接收界面：



装运管理界面：



## 模块间接口设计

### 3.2.1 数据库连接初始化

private static final String PORT = XMLReader.loadjdbcconfig().getPort();

private static final String NAME = XMLReader.loadjdbcconfig().getName();

private static final String ADMIN = XMLReader.loadjdbcconfig().getAdmin();

private static final String PASSWORD = XMLReader.loadjdbcconfig()

.getPassword();

private static Connection connect;

public static void init()

{

connect = DataJDBCConnection.getConnection();

try

{

PreparedStatement prestmt = connect

.prepareStatement("set names gbk");

prestmt.execute();

ServerUI.getFrame().myPanel.add("Success load Mysql Driver!\n");

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static Connection getConnection()

{

// 初始化

try

{

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

// 加载MYSQL JDBC驱动程序

connect = DriverManager.getConnection(

"jdbc:mysql://localhost:" + PORT + "/" + NAME

+ "?useUnicode=true&characterEncoding=UTF-8",

ADMIN, PASSWORD);

// ServerUI.getFrame().myPanel.add("Success load Mysql Driver!\n");

}

catch (Exception e)

{

ServerUI.getFrame().myPanel.add("Error load Mysql Driver!\n");

}

return connect;

}

### 3.2.2 调用数据库增删改功能

public static void update(String sql)

{

connect = DataJDBCConnection.getConnection();

try

{

PreparedStatement prestmt = connect.prepareStatement(sql);

prestmt.execute();

ServerUI.getFrame().myPanel.add("Execute :" + sql + "\n");

}

catch (SQLException e)

{

ServerUI.getFrame().myPanel

.add("Error :" + e.getMessage() + "\n");

}

}

### 3.2.3 数据库查找功能

public static ResultSet find(String sql)

{

connect = DataJDBCConnection.getConnection();

ResultSet rs = null;

try

{

Statement stmt = connect.createStatement();

rs = (ResultSet) stmt.executeQuery(sql);

ServerUI.getFrame().myPanel.add("Find :" + sql + "\n");

}

catch (SQLException e)

{

// TODO Auto-generated catch block

ServerUI.getFrame().myPanel

.add("Error :" + e.getMessage() + "\n");

}

return rs;

}

## 3.3 服务器与客户端间通信

服务器与数据库间通信使用mysql-connector-java-8.0.18.jar，位于libs文件夹下。

服务器与客户端间通信使用RMI通信端口。

public class RMIstart {

/\*\*

\* 端口号

\*/

private static final int PORT=XMLReader.loadipconfig().getPORT();

/\*\*

\* 服务器ip地址

\*/

private static final String IP=XMLReader.loadipconfig().getIP();

public static void init(){

if(IP==null){

System.out.println("error IP config!Please set your ip.");

System.exit(0);

}

try {

//注册RMI通信服务器端

System.setProperty("java.rmi.server.hostname",IP);

//注册通讯端口

LocateRegistry.createRegistry(PORT);

//创建实现类

ConnectedDataService ConnectedDataImpl=new ConnectedDataImpl();

AgencyDataService AgencyDataImpl=new AgencyDataImpl();

BankAccountDataService BankAccountDataImpl=new BankAccountDataImpl();

DriverDataService DriverDataImpl=new DriverDataImpl();

StaffDataService StaffDataImpl=new StaffDataImpl();

TruckDataService TruckDataImpl=new TruckDataImpl();

ConstantDataService ConstantDataImpl=new ConstantDataImpl();

DeliveringListDataService DeliveringListDataImpl=new DeliveringListDataImpl();

HallArrivalListDataService HallArrivalListDataImpl=new HallArrivalListDataImpl();

LoadListDataService LoadListDataImpl=new LoadListDataImpl();

OrderListDataService OrderListDataImpl=new OrderListDataImpl();

TransArrivalListDataService TransArrivalListDataImpl=new TransArrivalListDataImpl();

TransShipmentListDataService TransShipmentListDataImpl=new TransShipmentListDataImpl();

BaseDataSettingDataService BaseDataSettingDataImpl=new BaseDataSettingDataImpl();

MoneyInListDataService MoneyInListDataImpl=new MoneyInListDataImpl();

MoneyOutListDataService MoneyOutListDataImpl=new MoneyOutListDataImpl();

RepertoryInDataService RepertoryInDataImpl=new RepertoryInDataImpl();

RepertoryOutDataService RepertoryOutDataImpl=new RepertoryOutDataImpl();

RepertoryInfoDataService RepertoryInfoDataImpl=new RepertoryInfoDataImpl();

UserDataService UserDataImpl=new UserDataImpl();

SystemLogDataService SystemLogDataImpl=new SystemLogDataImpl();

//注册通讯路径，把所有的Impl实现类注册到RMI服务器上

Naming.rebind("rmi://"+IP+":"+PORT+"/ConnectedDataService", ConnectedDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/UserDataService", UserDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/AgencyDataService", AgencyDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/SystemLogDataService", SystemLogDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/BankAccountDataService", BankAccountDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/DriverDataService", DriverDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/StaffDataService", StaffDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/TruckDataService", TruckDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/ConstantDataService", ConstantDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/DeliveringListDataService", DeliveringListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/HallArrivalListDataService", HallArrivalListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/LoadListDataService", LoadListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/OrderListDataService", OrderListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/TransArrivalListDataService", TransArrivalListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/TransShipmentListDataService", TransShipmentListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/BaseDataSettingDataService", BaseDataSettingDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/MoneyInListDataService", MoneyInListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/MoneyOutListDataService", MoneyOutListDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/RepertoryInDataService", RepertoryInDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/RepertoryOutDataService", RepertoryOutDataImpl);

Naming.rebind("rmi://"+IP+":"+PORT+"/RepertoryInfoDataService", RepertoryInfoDataImpl);

} catch (RemoteException e) {

e.printStackTrace();

} catch (MalformedURLException e) {

e.printStackTrace();

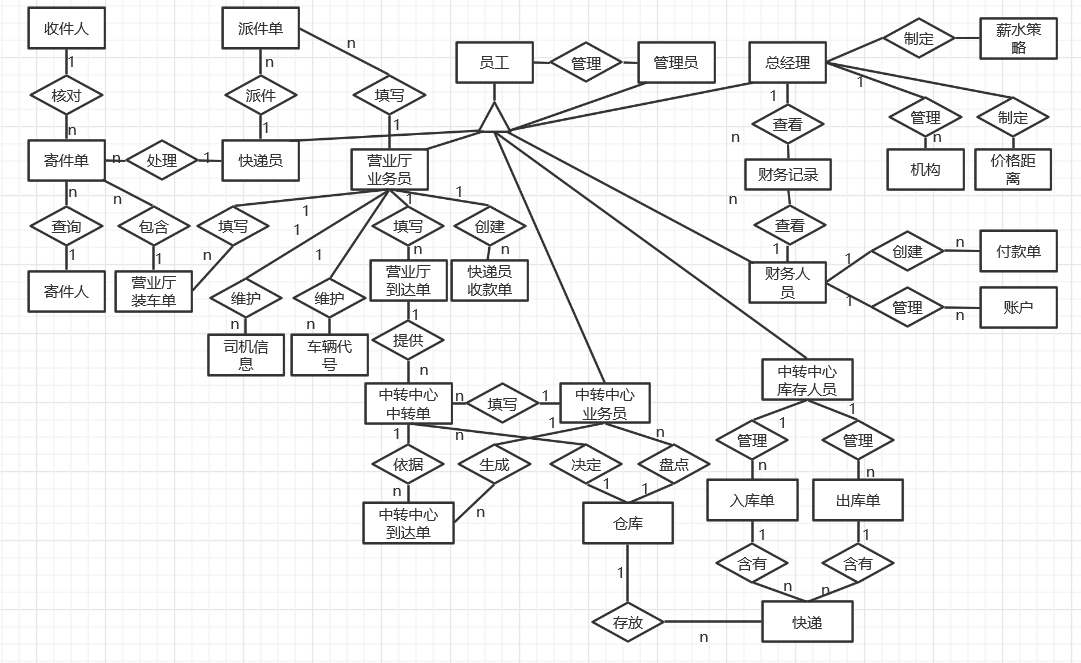
}

}

}

# 四．数据设计

## 4.1 调整后的实体-关系图



## 4.2 关系模式的建立

收件人表（身份证号，姓名，用户名，密码，手机号）

寄件人表（身份证号，姓名，用户名，密码，手机号）

寄件单表（编号，是否被审阅，出发日期，总价，货物信息，到达时间，包装费，是否被签收，订单类型，收件人信息，寄件人信息）

快递员表（工号，姓名，职位）

快递员收款单表（订单号，收款金额，快递员工号，收款日期）

营业厅业务员表（姓名，工号，职位）

派件单表（日期，派件订单号，快递员工号）

营业厅装车单（装车日期，装车号，运费，本次装箱所有订单号，押运员工号，监装员工号，车辆代号，到达营业厅编号，汽运编号，营业厅编号）

车辆代号表（城市编号，营业厅编号，车牌号，服役时间）

营业厅到达单（物品情况，营业厅编号，订单编号，到达日期）

司机信息表（编号，姓名，身份证号，性别，手机，行驶证期限）

中转中心中转单表（汽车车次，监装员，货柜号，中转出发日期，装箱所有订单号，中转单号，中转中心编号，运费，到达营业厅编号）

中转中心到达单表（货物状态，出发营业厅编号，中转单编号，中转中心编号，到达日期）

中转中心业务员表（工号，姓名，职位）

中转中心装车单表（装车日期，装车号，运费，本次装箱所有订单号，押运员工号，监装员工号，车辆代号，到达营业厅编号，汽运编号，营业厅编号）

中转中心库存管理人员表（工号，姓名，职位）

入库单表（快递编号，入库日期，目的地，区号，排位，架号，位号）

仓库表（区，排，架，位）

出库单（目的地，搬运形式，汽运编号，快递编号，出库日期）

账户表（户号，名称，金额）

总经理表（工号，姓名，职位）

付款单表（付款日期，付款金额，条目，备注，付款账号，付款单号）

快递表（快递编号，区号，排号，架号，位号）

单据表（草稿状态，审批后状态，提交状态，单据号）

员工表（工号，职位，密码，薪水，姓名）

价格距离表（价格，距离）

财务记录表（类型，发起单位，接受单位，时间，记录号，金额）

机构表（编号，名称）

管理员表（工号，姓名，职位）

## 4.3 数据库中表的实现

### 机构表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 编号 | varchar(25) | 否 | 主键 |
| 名称 | varchar(25) |  |  |

### 账户表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 户号 | varchar(25) | 否 | 主键 |
| 名称 | varchar(25) |  |  |
| 金额 | double |  |  |

### 收件人表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 身份证号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 用户名 | varchar(25) |  |  |
| 密码 | varchar(25) |  |  |
| 手机号 | varchar(25) |  |  |

### 寄件人表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 身份证号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 用户名 | varchar(25) |  |  |
| 密码 | varchar(25) |  |  |
| 手机号 | varchar(25) |  |  |

### 寄件单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 编号 | varchar(25) | 否 | 主键 |
| 是否被审阅 | bit |  |  |
| 出发日期 | datatime |  |  |
| 总价 | double |  |  |
| 货物信息 | varchar(25) |  |  |
| 到达时间 | datatime |  |  |
| 包装费 | double |  |  |
| 是否被签收 | bit |  |  |
| 订单类型 | varchar(25) |  |  |
| 收件人信息 | varchar(25) | 否 |  |
| 寄件人信息 | varchar(25) | 否 |  |

### 快递员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 快递员收款单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 订单号 | varchar(25) | 否 | 主键 |
| 收款金额 | double |  |  |
| 快递员工号 | varchar(25) |  |  |
| 收款日期 | datatime |  |  |

### 营业厅业务员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 派件单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 日期 | datatime |  |  |
| 派件订单号 | varchar(25) | 否 | 主键 |
| 快递员工号 | varchar(25) |  |  |

### 营业厅装车单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 装车号 | varchar(25) | 否 | 主键 |
| 本次装箱订单号 | varchar(25) |  |  |
| 装车日期 | datatime |  |  |
| 运费 | double |  |  |
| 押运员工号 | varchar(25) |  |  |
| 监装员工号 | varchar(25) |  |  |
| 车辆代号 | varchar(25) |  |  |
| 到达营业厅编号 | varchar(25) | 否 |  |
| 汽运编号 | varchar(25) |  |  |
| 营业厅编号 | varchar(25) | 否 |  |

### 车辆代号表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 车牌号 | varchar(25) | 否 | 主键 |
| 城市编号 | varchar(25) |  |  |
| 营业厅编号 | varchar(25) | 否 |  |
| 服役时间 | datatime |  |  |

### 营业厅到达单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 订单号 | varchar(25) | 否 | 主键 |
| 物品情况 | varchar(25) |  |  |
| 营业厅编号 | varchar(25) | 否 |  |
| 到达日期 | datatime |  |  |

### 司机信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 编号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 身份证号 | varchar(25) | 否 |  |
| 手机 | varchar(25) |  |  |
| 性别 | varchar(25) |  |  |
| 行驶证期限 | datatime |  |  |

### 中转中心中转单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 汽车车次 | varchar(25) |  |  |
| 监装员工号 | varchar(25) |  |  |
| 货柜号 | varchar(25) |  |  |
| 装箱订单号 | varchar(25) |  |  |
| 中转出发日期 | datatime |  |  |
| 中转单号 | varchar(25) | 否 | 主键 |
| 中转中心编号 | varchar(25) | 否 |  |
| 运费 | double |  |  |
| 到达营业厅编号 | varchar(25) | 否 |  |

### 中转中心到达单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 货物状态 | varchar(25) |  |  |
| 出发营业厅编号 | varchar(25) | 否 |  |
| 中转单编号 | varchar(25) | 否 | 主键 |
| 中转中心编号 | varchar(25) | 否 |  |
| 到达日期 | datatime |  |  |

### 中转中心业务员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 中转中心装车单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 装车号 | varchar(25) | 否 | 主键 |
| 本次装箱订单号 | varchar(25) |  |  |
| 装车日期 | datatime |  |  |
| 运费 | double |  |  |
| 押运员工号 | varchar(25) |  |  |
| 监装员工号 | varchar(25) |  |  |
| 车辆代号 | varchar(25) |  |  |
| 到达营业厅编号 | varchar(25) | 否 |  |
| 汽运编号 | varchar(25) |  |  |
| 营业厅编号 | varchar(25) | 否 |  |

### 中转中心库存管理人员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 付款单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 付款日期 | datatime |  |  |
| 付款金额 | double |  |  |
| 条目 | int |  |  |
| 备注 | varchar(25) |  |  |
| 付款账号 | varchar(25) | 否 |  |
| 付款单号 | varchar(25) | 否 | 主键 |

### 仓库表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 区 | varchar(25) | 否 | 主键 |
| 排 | varchar(25) | 否 | 主键 |
| 架 | varchar(25) | 否 | 主键 |
| 位 | varchar(25) | 否 | 主键 |

### 入库单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 快递编号 | varchar(25) | 否 | 主键 |
| 入库日期 | datatime |  |  |
| 目的地 | varchar(25) |  |  |
| 区号 | varchar(25) |  |  |
| 排号 | varchar(25) |  |  |
| 架号 | varchar(25) |  |  |
| 位号 | varchar(25) |  |  |

### 出库单表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 快递编号 | varchar(25) | 否 | 主键 |
| 出库日期 | datatime |  |  |
| 汽运编号 | varchar(25) |  |  |
| 搬运形式 | varchar(25) |  |  |
| 目的地 | varchar(25) |  |  |

### 总经理表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 快递表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 快递编号 | varchar(25) | 否 | 主键 |
| 区号 | varchar(25) |  |  |
| 排号 | varchar(25) |  |  |
| 架号 | varchar(25) |  |  |
| 位号 | varchar(25) |  |  |

### 单据表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 单据号 | varchar(25) | 否 | 主键 |
| 提交状态 | varchar(25) |  |  |
| 审批后状态 | varchar(25) |  |  |
| 草稿状态 | varchar(25) |  |  |

### 管理员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 姓名 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |

### 价格距离表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 价格 | double | 否 | 主键 |
| 距离 | double | 否 | 主键 |

### 员工表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 工号 | varchar(25) | 否 | 主键 |
| 密码 | varchar(25) |  |  |
| 职位 | varchar(25) |  |  |
| 姓名 | varchar(25) |  |  |
| 薪水 | double |  |  |

### 财务记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 主键/外键 |
| 记录号 | varchar(25) | 否 | 主键 |
| 类型 | varchar(25) |  |  |
| 发起单位号 | varchar(25) |  |  |
| 接受单位号 | varchar(25) |  |  |
| 金额 | double |  |  |
| 时间 | datatime |  |  |

## 4.4 建表语言

create table 收件人(

身份证号 varchar(25) not null primary key,

姓名 varchar(25),

用户名 varchar(25)

密码 varchar(25),

手机号 varchar(25),

)

create table 寄件人(

身份证号 varchar(25) not null primary key,

姓名 varchar(25),

用户名 varchar(25)

密码 varchar(25),

手机号 varchar(25),

)

Create table 寄件单(

编号 varchar(25) not null primary key,

是否被审阅 bit,

出发日期 varchar(25),

总价 double,

货物信息 varchar(25),

到达时间 datatime,

包装费 double,

是否被签收 bit,

订单类型 varchar(25),

收件人信息 varchar(25) not null,

寄件人信息 varchar(25) not null,

)

create table 快递员(

工号 varchar(25) not null primary key,

姓名 varchar(25),

职位 varchar(25),

)

create table 快递员收款单(

订单号 varchar(25) not null primary key,

收款金额 double,

快递员工号 varchar(25),

收款日期 datatime,

)

create table 营业厅业务员(

姓名 varchar(25),

工号 varchar(25) not null primary key,

职位 varchar(25),

)

create table 派件单(

日期 datatime,

派件订单号 varchar(25) not null primary key,

快递员工号 varchar(25),

)

create table 营业厅装车单(

装车日期 datatime,

装车号 varchar(25) not null primary key,

运费，本次装箱订单号 varchar(25),

押运员工号 varchar(25),

监装员工号 varchar(25),

车辆代号 varchar(25),

到达营业厅编号 varchar(25) not null,

汽运编号 varchar(25),

营业厅编号 varchar(25) not null,

)

create table 车辆代号(

城市编号 varchar(25),

营业厅编号 varchar(25) not null,

车牌号 varchar(25) not null primary key,

服役时间 datatime,

)

create table 营业厅到达单(

物品情况 varchar(25),

营业厅编号varchar(25) not null,

订单编号 varchar(25) not null primary key,

到达日期 datatime,

)

create table 司机信息(

编号 varchar(25) not null primary key,

姓名 varchar(25),

身份证号 varchar(25) not null,

性别 varchar(25),

手机 varchar(25),

行驶证期限 datatime,

)

create table 中转中心中转单(

汽车车次 varchar(25),

监装员 varchar(25),

货柜号 varchar(25),

中转出发日期 datatime,

装箱订单号 varchar(25),

中转单号 varchar(25) not null primary key,

中转中心编号 varchar(25) not null,

运费 double,

到达营业厅编号 varchar(25) not null,

)

create table 中转中心到达单(

货物状态 varchar(25),

出发营业厅编号 varchar(25) not null,

中转单编号 varchar(25) not null primary key,

中转中心编号 varchar(25) not null,

到达日期 datatime,

)

create table 中转中心业务员(

工号 varchar(25) not null primary key,

姓名 varchar(25),

职位 varchar(25),

)

create table 中转中心装车单(装车日期，

装车号 varchar(25) not null primary key,

运费 double,

本次装箱订单号 varchar(25),

押运员工号 varchar(25),

监装员工号 varchar(25),

车辆代号 varchar(25),

到达营业厅编号 varchar(25) not null,

汽运编号 varchar(25),

营业厅编号 varchar(25) not null,

)

create table 中转中心库存管理人员(

工号 varchar(25) not null primary key,

姓名 varchar(25),

职位 varchar(25),

)

create table 入库单(

快递编号 varchar(25) not null primary key,

入库日期 datatime,

目的地 varchar(25),

区号 varchar(25),

排位 varchar(25),

架号 varchar(25),

位号 varchar(25),

)

create table 仓库(

区 varchar(25),

排 varchar(25),

架 varchar(25),

位 varchar(25),

primary key(区,排,架,位),

)

create table 出库单(

目的地 varchar(25),

搬运形式 varchar(25),

汽运编号 varchar(25),

快递编号 varchar(25) not null primary key,

出库日期 datatime,

)

create table 账户(

户号 varchar(25) not null primary key,

名称varchar(25),

金额 double,

)

create table 总经理(

工号 varchar(25) not null primary key,

姓名 varchar(25),

职位 varchar(25),

)

create table 付款单(

付款日期 datatime,

付款金额 double,

条目 int,

备注 varchar(25),

付款账号 varchar(25),

付款单号 varchar(25) not null primary key,

)

create table 快递(

快递编号 varchar(25) not null primary key,

区号 varchar(25),

排号 varchar(25),

架号 varchar(25),

位号 varchar(25),

)

create table 单据(

草稿状态 varchar(25),

审批后状态 varchar(25),

提交状态 varchar(25),

单据号 varchar(25) not null primary key,

)

create table 员工(

工号 varchar(25) not null primary key,

职位 varchar(25),

密码 varchar(25),

薪水 double,

姓名 varchar(25),

)

create table 价格距离(

价格 varchar(25),

距离 varchar(25),

primary key(价格,距离),

)

create table 财务记录(

类型 varchar(25),

发起单位 varchar(25),

接受单位 varchar(25),

时间 datatime,

记录号 varchar(25) not null primary key,

金额 double,

)

create table 机构(

编号 varchar(25) not null primary key,

名称 varchar(25),

)

create table 管理员表(

工号 varchar(25) not null primary key,

姓名 varchar(25),

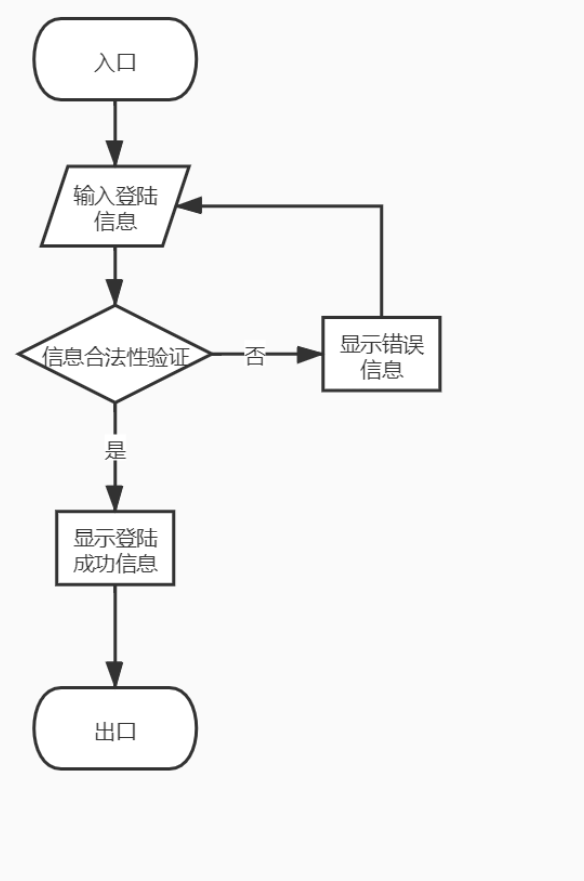
职位 varchar(25),

)

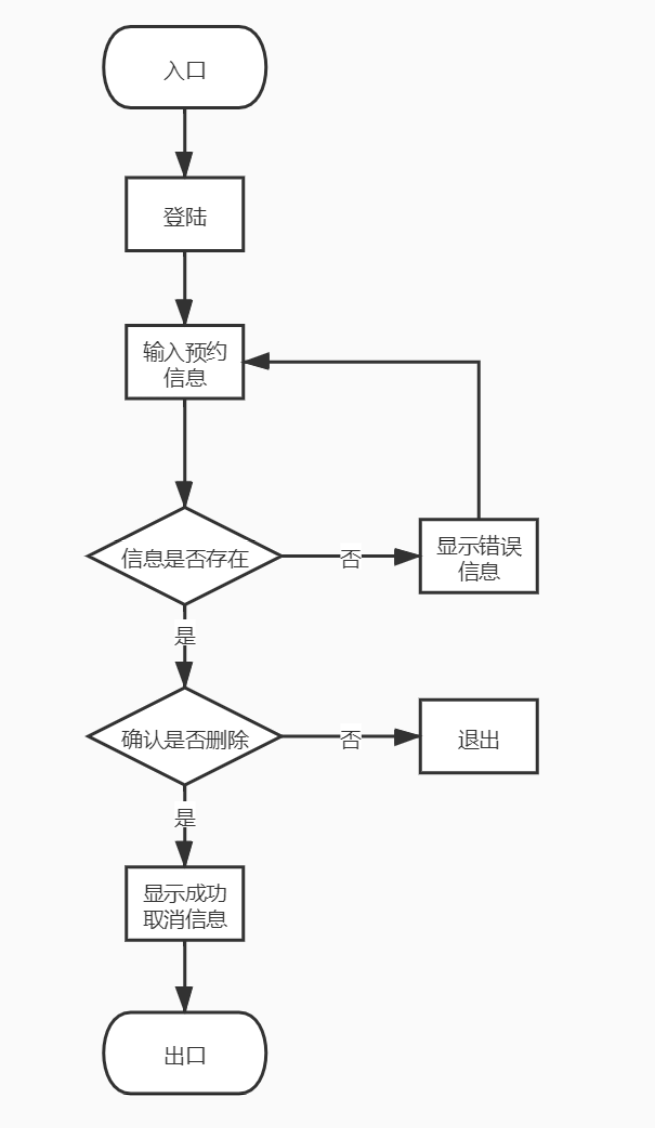
# 五．过程设计

## 5.1 主要功能模块的程序流程图

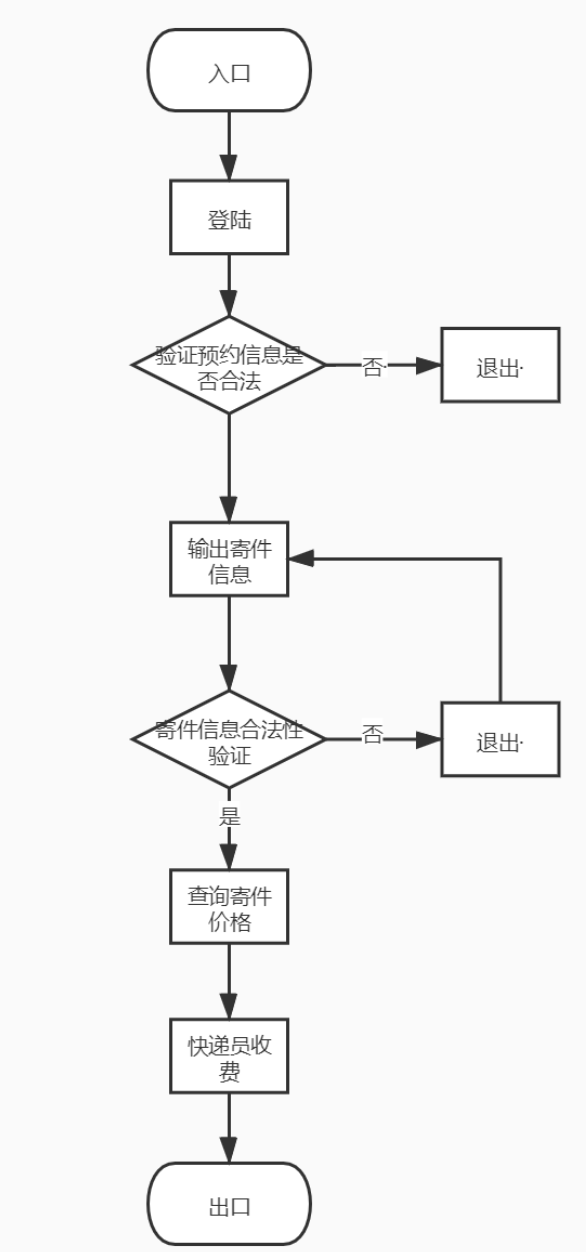
1. 登陆



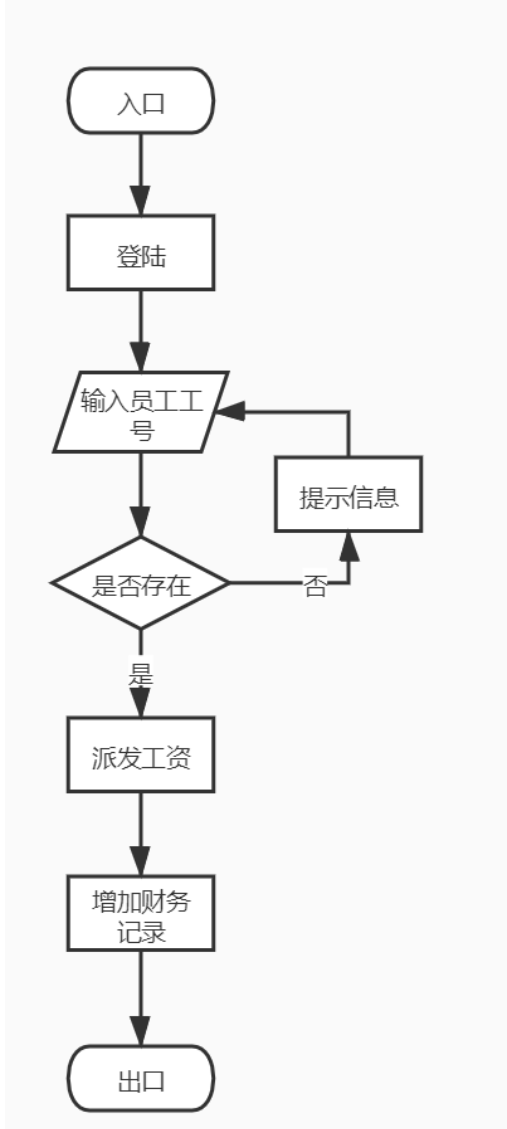
1. 取消预约



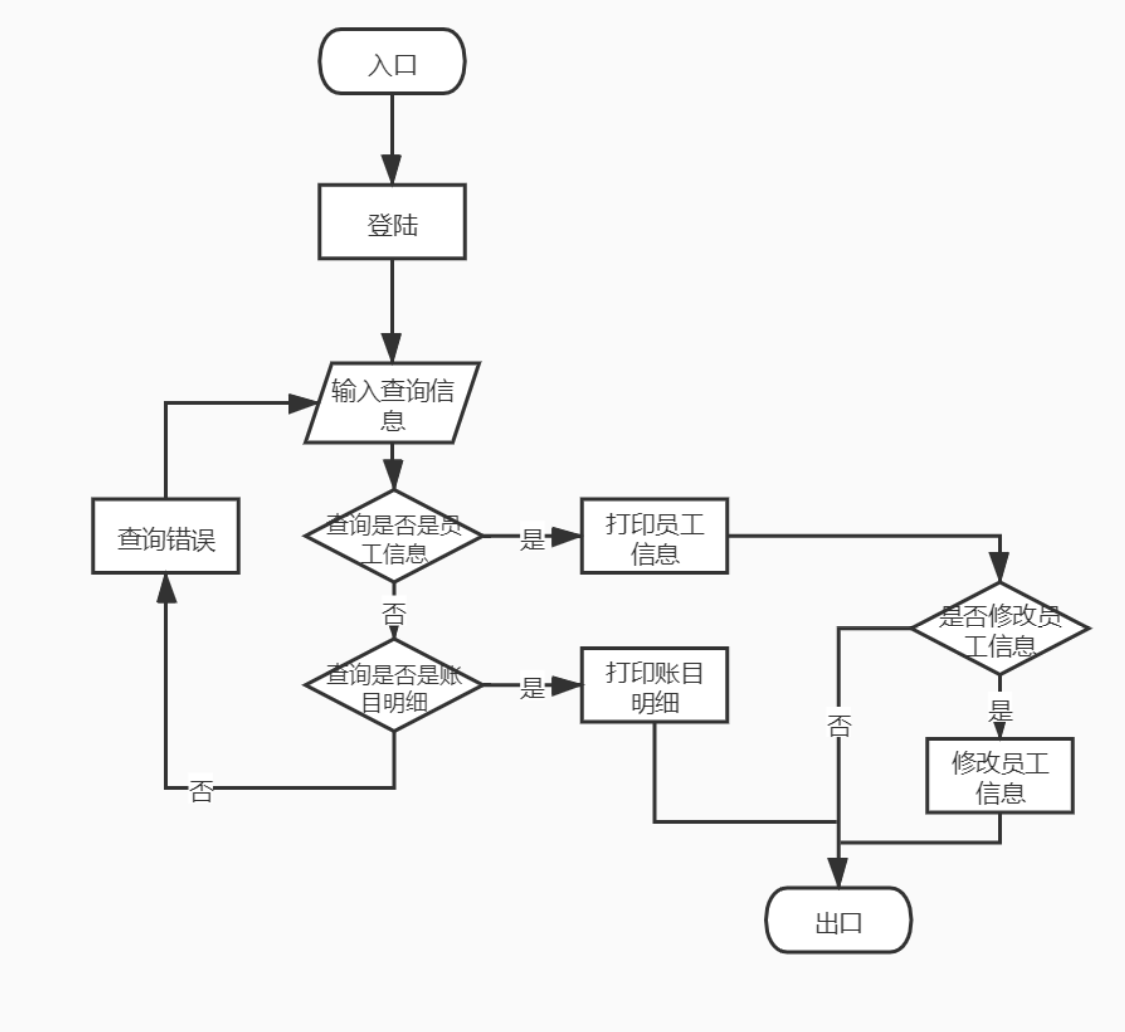
1. 寄件



1. 派发工资



1. 维护员工信息



## 5.2 重要算法的伪代码表述

更新城市间货运距离：

public void setDistance(String newDistance, String cityA, String cityB) {

// distancChart存储已有城市距离表

String[][] distanceChart=this.showDistanceChart().getDistanceChart();

// c1、c2找到所要修改的城市所在下标

int c1=-1,c2=-1;

for(int i=0;i<distanceChart[0].length;i++)

{

if(cityA.equals(distanceChart[0][i]))

c1=i;

if(cityB.equals(distanceChart[0][i]))

c2=i;

}

// 若c1、c2不为-1，说明输入城市存在，则修改可完成。

if(c1!=-1&&c2!=-1)

{

System.out.print("success");

// 修改表中信息

distanceChart[c1][c2]=newDistance;

distanceChart[c2][c1]=newDistance;

// 写入表

try{

FileWriter fw=new FileWriter(f1);

fw.write("");

fw.close();

fw=new FileWriter(f1,true);

int num=distanceChart[0].length;

for(int i=0;i<num;i++)

{

if(i<num-1){

for(int j=0;j<num;j++)

if(j<num-1)

fw.write(distanceChart[i][j]+" ");

else

fw.write(distanceChart[i][j]);

fw.write("\n");

}else

for(int j=0;j<num;j++)

if(j<num-1)

fw.write(distanceChart[i][j]+" ");

else

fw.write(distanceChart[i][j]);

}

fw.close();

}catch(Exception e){

e.printStackTrace();

}

}

展示银行账户表：

public ArrayList<BankAccountPO> showAll() {

// 用ArrayList存储

ArrayList<BankAccountPO> accountlist =new ArrayList<BankAccountPO>();

BankAccountPO bankaccount;

// sql字符串存储命令

String sql="select \* from bankaccount";

// 向DataJDBCConnection.find函数（上面已给出）注入sql语句

ResultSet rs=(ResultSet) DataJDBCConnection.find(sql);

try {

// 将得到的ResultSet写入accountlist

while(rs.next())

{

bankaccount=new BankAccountPO(rs.getString("name"), rs.getString("number"), Double.parseDouble(rs.getString("money")));

accountlist.add(bankaccount);

}

} catch (SQLException e) {

System.out.println("未找到");

return accountlist;

}

return accountlist;

}