

《数据库原理实践》

课程设计报告

——外卖管理系统的设计与实现

|  |  |  |
| --- | --- | --- |
|  | 组 长 | 组 员 |
| 姓 名： |  |  |
| 学 号： |  |  |
| 班 级： |  |  |
| 编 号： |  |  |
| 指导教师： |  | |
| 提交日期： |  | |

# 目 录

目 录 2

（一）概述 4

1.1项目背景 4

1.2课程设计选题及分析 4

（二）需求分析 5

2.1数据流图 5

2.2数据字典 8

2.3系统功能模块图 9

（三）概念结构设计 10

3.1 菜品管理模块 10

3.1.1 模块功能描述 10

3.1.2 模块分E-R图 11

3.2 员工管理模块 12

3.2.1 模块功能描述 12

3.2.2 模块分E-R图 13

3.3 订单管理模块 14

3.3.1 模块功能描述 14

3.3.2 模块分E-R图 15

3.4 物流信息模块 15

3.4.1 模块功能描述 15

3.4.2 模块分E-R图 16

3.5 店铺管理模块 16

3.5.1 模块功能描述 16

3.5.2 模块分E-R图 17

3. 6总体E-R图 18

（四）逻辑结构设计 18

4.1 转换为关系数据模型 21

4.2 关系模式的优化 23

4.3 视图设计 27

（五） 数据库物理设计与实施 30

5.1 数据库应用的硬件、软件环境介绍 30

5.2 物理结构设计 30

5.3 物理结构优化 33

1. 索引设计 33

2. 视图设计 36

3. 存储过程设计 37

4. 触发器设计 42

5. 权限控制设计 45

5.4 数据库实施 45

1. 环境准备 45

2.数据库创建 45

3.数据导入 48

4. 索引和视图的创建 49

5. 存储过程和触发器的实现 51

（六） 应用系统的设计与实施 53

6.1 连接数据库 53

6.2 界面设计 54

6.2.1 前端查询功能的实现 54

6.1.2 前端查询方法实现 55

6.1.3前端删除功能的实现 55

6.1.4前端菜品口味管理功能 56

6.1.5前端口味选择功能 56

6.1.5前端导出功能 57

6.3 功能实现 57

6.3.1 菜品管理模块 57

6.3.2 员工管理模块 59

6.3.3 订单管理模块 60

6.3.4 店铺管理模块 61

6.3.5 物流信息模块 63

（七） 数据库测试 64

7.1 加载测试数据 64

7.2 功能测试 66

7.2.1 菜品模块功能测试 66

7.2.2 员工管理模块功能测试 67

7.2.3 订单管理模块功能测试 69

7.2.4 店铺管理模块功能测试 71

7.2.5 物流管理模块功能测试 72

7.3 性能测试 74

7.3.1 内存消耗、CPU消耗 74

7.3.2 时间消耗 76

（八） 收获和体会 77

（九） 参考文献 78

# （一）概述

## 1.1项目背景

随着移动互联网技术的发展，人们的生活方式发生了巨大变化，其中最显著的一个方面就是餐饮消费模式的变化。外卖服务作为现代生活的一部分，已经成为许多人日常生活不可或缺的一部分。为了更好地满足用户需求，提高商家的服务效率和用户体验，构建一个高效、便捷且功能完善的外卖管理系统显得尤为重要。

若依（RuoYi）是一个开源的企业级快速开发平台，它使用了Spring Boot + MyBatis Plus + Shiro + Layui等技术栈，提供了丰富的基础功能模块和良好的扩展性，非常适合用于企业级应用的快速开发。本项目将基于若依框架进行二次开发，旨在创建一个专为外卖业务定制的管理平台，通过优化原有系统架构来提升系统的实用性与易用性。

## 1.2课程设计选题及分析

选题为外卖管理系统，设计外卖管理以及相关订单管理中的管理内容，包括菜品管理、订单、员工管理、订单管理以及相关的增删查改、统计功能，简化的系统数据以及相关功能如下:  
(1)每个员工都有唯一的员工号(employee\_id),要反映出员工姓名(employee\_name)、员工账号(account)、手机号(phone\_number)、以及账号状态是否启用(account\_status)、最后的操作时间(last\_operation\_time)

1. 关于用户，其外卖订单信息则由订单表反映，每张订单表有如下字段:订单编号(order\_id)、店铺名称(shop\_name)、订单金额(order\_money)、以及订餐方式的选择(order\_way)、顾客的电话(cons\_phone)、顾客姓名(cons\_name)、顾客地址(cons\_addre)、以及审核状态(checked)、创建时间(create\_time)
2. 调度员信息则有调度员编号(dispatcher\_id)、调度员姓名(dispatcher\_name)、调度员电话(dispatcher\_phone)
3. 物流信息则包括物流编号(wuliu\_id)、店铺名称(shop\_name)、调度员编号(dispatcher\_id)
4. 员工管理则包括员工编号(employee\_id)、员工姓名(employee\_name)、员工账号(account)、手机号(phone\_number)、账号启动状态(account\_status)、最后操作时间(last\_operation\_time)

(6)店铺管理的信息则包括店铺ID(shop\_id)、店铺名称(shop\_name)、店铺地址(shop\_address)、店铺电话(shop\_phone)、营业状态(shop\_status)

# （二）需求分析

## 2.1数据流图

（1）顶层数据流图：外部实体有菜品、员工、订单、物流信息、店铺共五个。

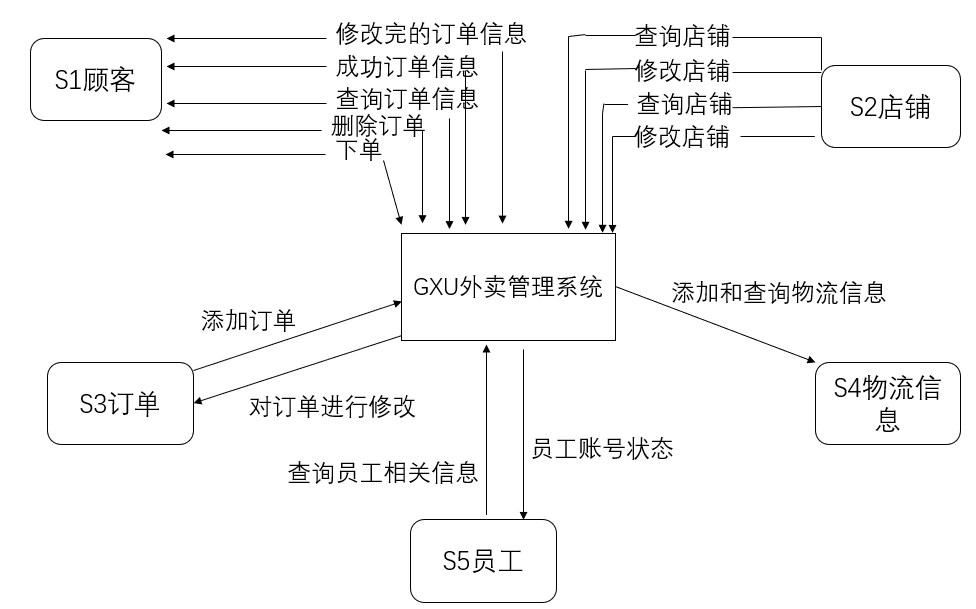


图2-1 顶层图

（2）零层数据流图：分解原来庞大的系统，增加数据存储和对机加工和外部实体的编号。笔者采用使用“订单处理功能”、“菜品处理功能”、“员工处理功能”、“物流处理功能”替代顶层数据流图中的“GXU管理系统”。存储信息：订单信息，菜品金额，物流信息情况。如下图2-2所示。

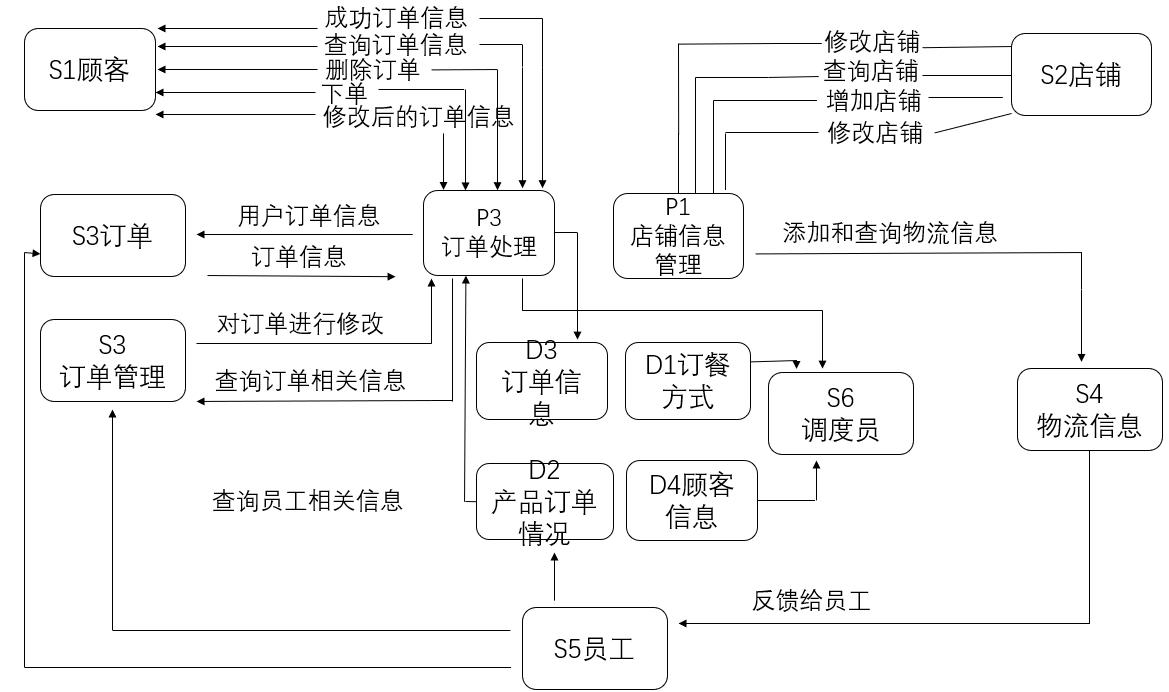


图2-2 零层图

经过细化后从图可以看出，在顾客与后台管理之间主要存在五个方面的数据流出入，分别是：菜品的信息查询、订单的处理、物流处理、员工管理和店铺管理，五个方面分别对应五个系统功能，下面分别就这五个方面绘制数据流图。

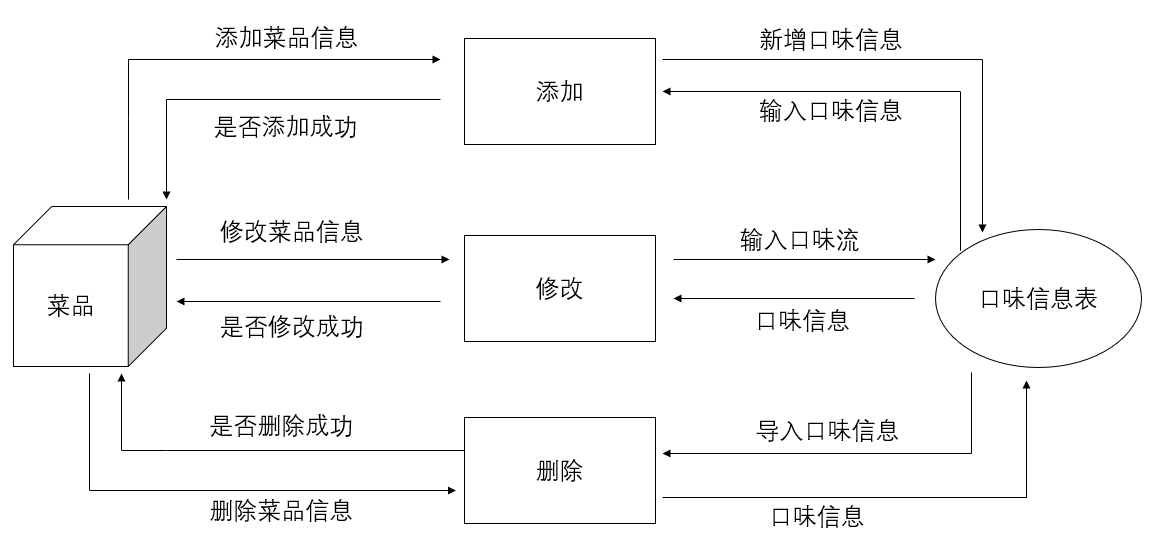


图2-3 管理员对菜品和口味修改数据流

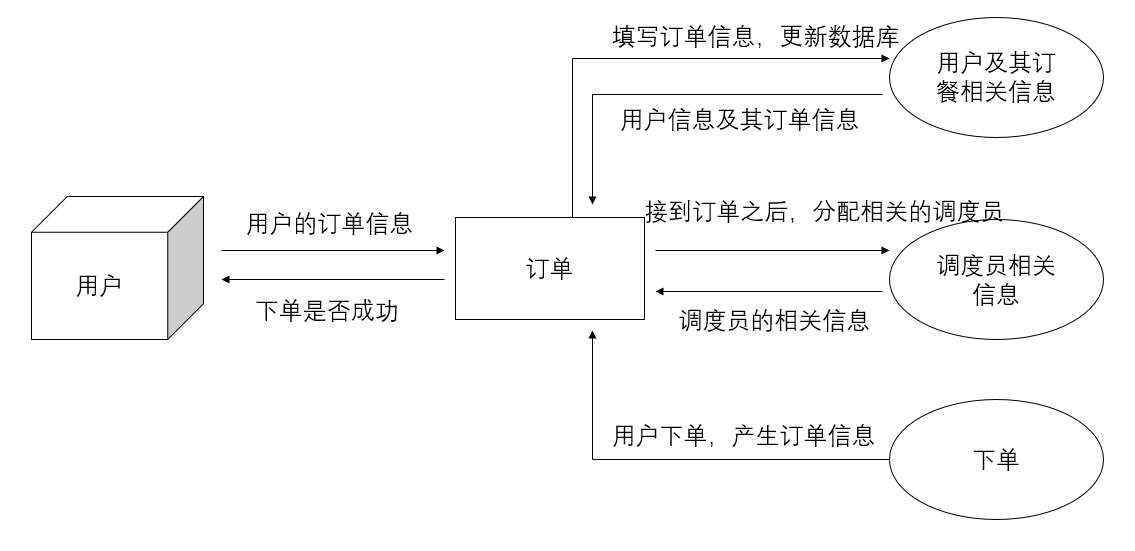


图2-4 订单数据流

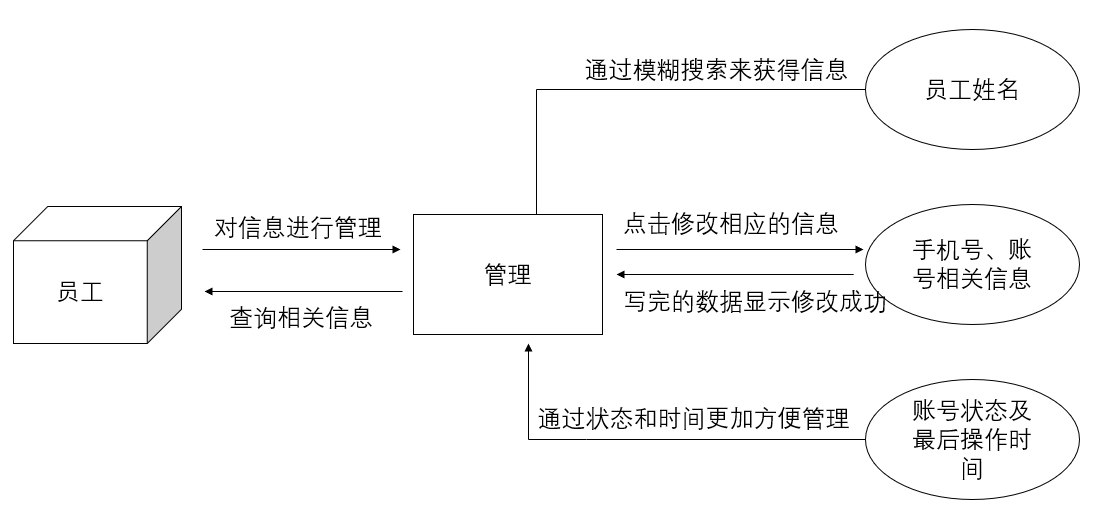


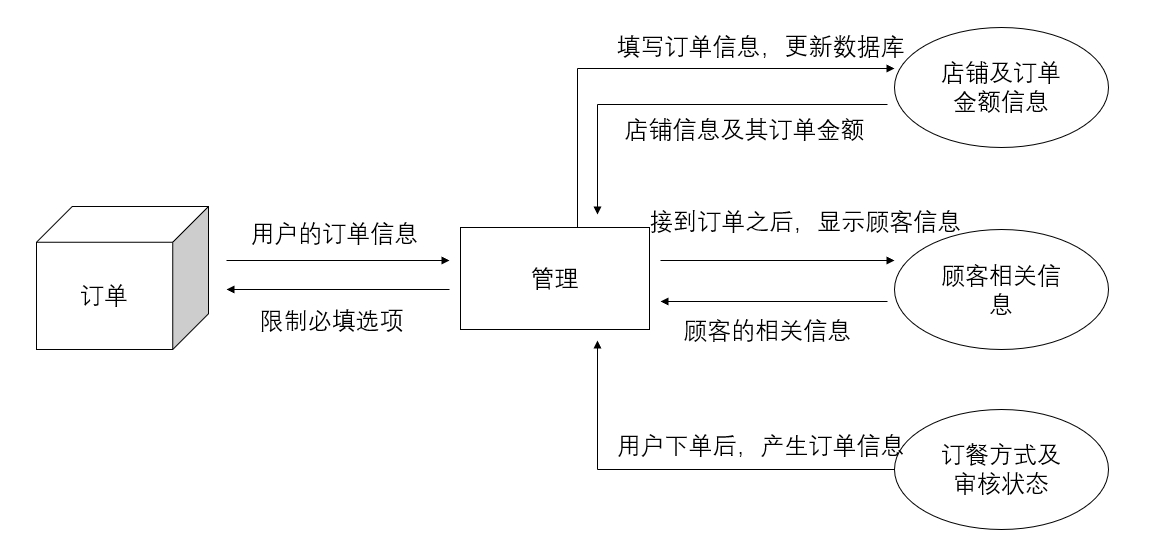
图2-5 员工数据流

图2-6 订单管理数据流

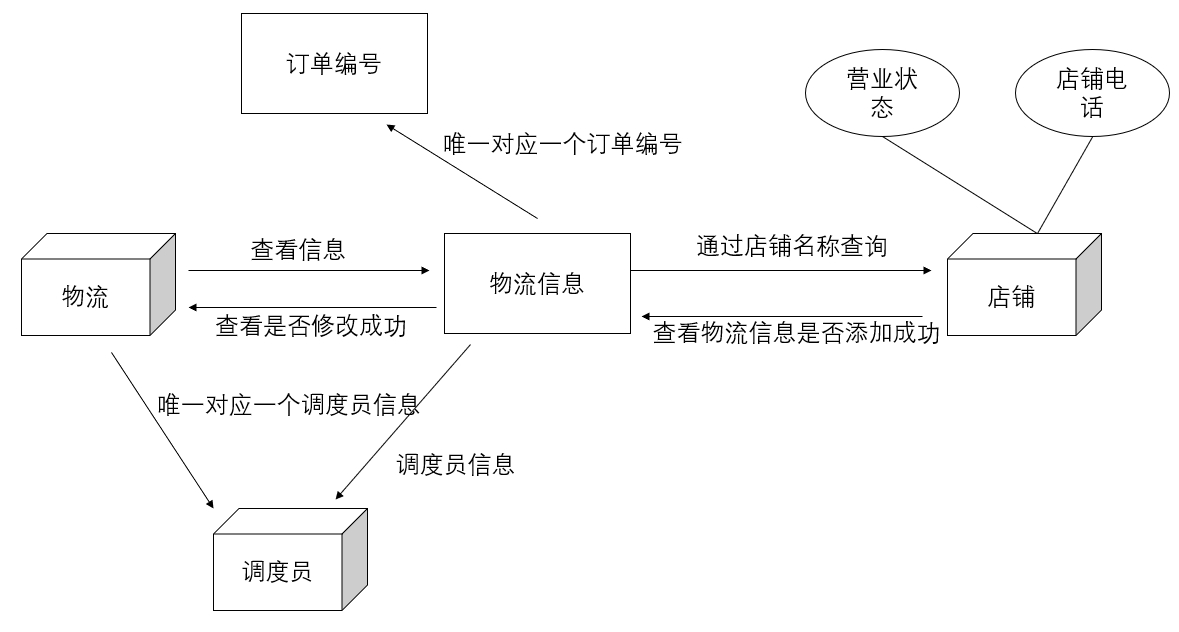


图2-7 物流信息和店铺信息数据流

## 2.2数据字典

系统的数据字典包括数据存储字典、数据流字典、数据处理字典、数据项字典、数据结构。

表2-1菜品相关的数据字典

|  |
| --- |
| （1）处理菜品“添加”描述 |
| **处理过程**：添加 |
| **说明**：在菜品管理界面进行操作 |
| **输入**：菜品名称、相关金额、售卖状态、口味信息 |
| **输出**：菜品信息，新增菜品是否成功信息 |
| **说明**：根据输入的菜品名称、相关金额、售卖状态、口味信息，若菜品名称未出现过，则将菜品信息记录在菜品信息表中，新增菜品成功。 |
| （2）处理菜品“删除”描述 |
| **处理过程**：登录 |
| **说明**：选中指定的菜品之后点击删除 |
| **输入**：确认是否删除 |
| **输出**：菜品是否删除成功的信息 |
| **说明**：根据删除的菜品信息搜索，判断删除是否成功。 |
| （3）处理菜品“修改”描述 |
| **处理过程**：修改菜品信息 |
| **说明**：修改菜品信息 |
| **输入**：修改之后的菜品相关信息 |
| **输出**：菜品信息以及是否修改成功 |
| **说明**：根据修改后的菜品信息进行搜索，来判断是否修改成功。 |

如图2.4-2.7为分别为用户信息、员工信息、订单信息、物流和店铺信息的数据流图，涉及到的对应的处理过程描述如下：

表2-2用户、物流、店铺等的数据流图

|  |
| --- |
| （1）用户订单信息描述 |
| **处理过程**：订单 |
| **说明**：客户订单 |
| **输入**：用户的订单信息、调度员的信息、用户信息 |
| **输出**：下单是否成功 |
| **说明**：填写信息是若有错误，则无法填写，成功与否均有提示 |
| （2）员工信息描述 |
| **处理过程**：增删查改 |
| **说明**：对员工信息进行修改 |
| **输入**：查询相关信息 |
| **输出**：搜索的结果 |
| **说明**：根据员工姓名、账号、手机号、账号状态来搜索 |
| （3）物流和店铺信息描述 |
| **处理过程**：查找 |
| **说明**：通过店铺名称和物流查询 |
| **输入**：相应的字段名 |
| **输出**：物流信息是否修改成功、店铺信息是否修改成功 |
| **说明**：根据修改后的信息进行管理。 |

## 2.3系统功能模块图

外卖管理系统的功能模块图如下:

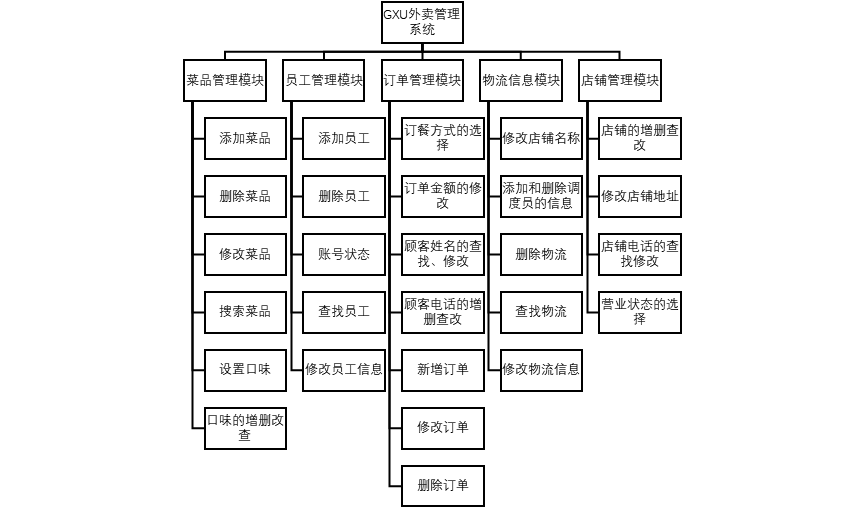


图2-3 系統功能模块图

# （三）概念结构设计

## 3.1 菜品管理模块

### 3.1.1 模块功能描述

该模块的功能主要有菜品相关信息的增删差改、菜品的添加、更新和删除、查询。以及通过菜品的名称搜索相应的菜品，以及导出功能，导出的是excel表，前端界面可以清晰的看到菜品的具体信息，更新时间，编辑和删除可以通过左上角的修改和删除进行，也可以在菜品行中最右侧的两个按钮进行。如下图3-1所示。

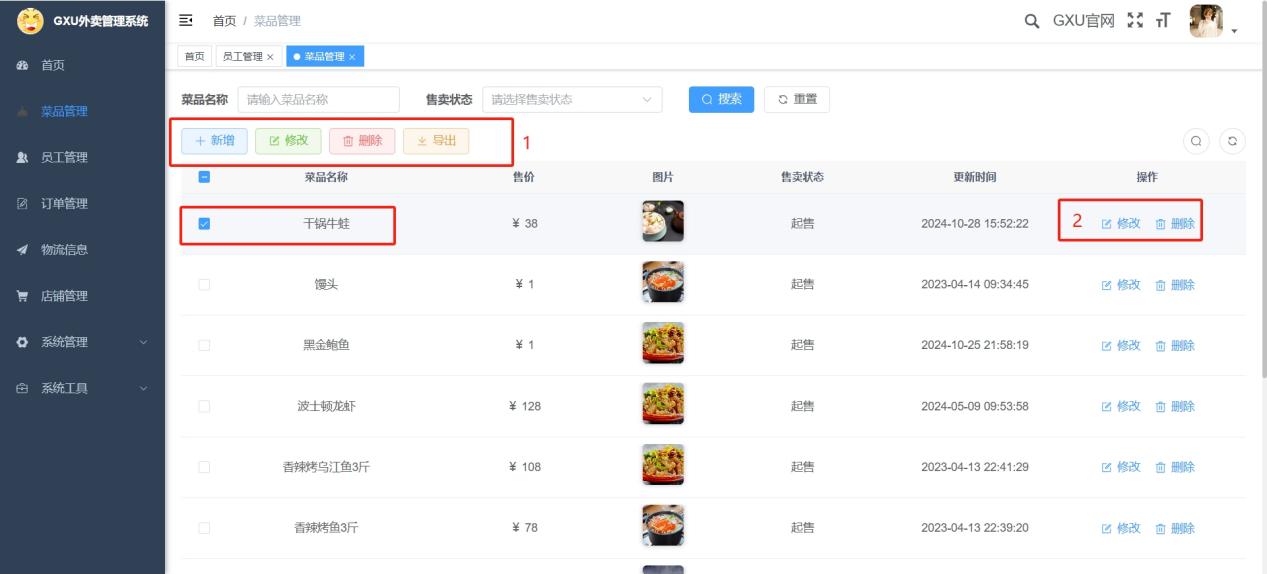


图3-1 菜品功能模块图

在添加菜品的时候，需要输入菜品的名称，售价，并且还可以上传图片。可以输出描述信息。还可以设置售卖状态，同时由于菜品模块内菜品表与口味表相关联，因此可以选择口味信息，点击添加之后，便可以选择口味名称与口味列表。如下图

3-2所示。



图3-2 添加菜品功能图

综上所述，菜品管理模块涉及的实体有口味、菜品，其中菜品与口味的关系是一对多，及1:n，下面是此模块的分E-R图。

### 3.1.2 模块分E-R图

菜品管理模块的分E-R图如图3-3所示：

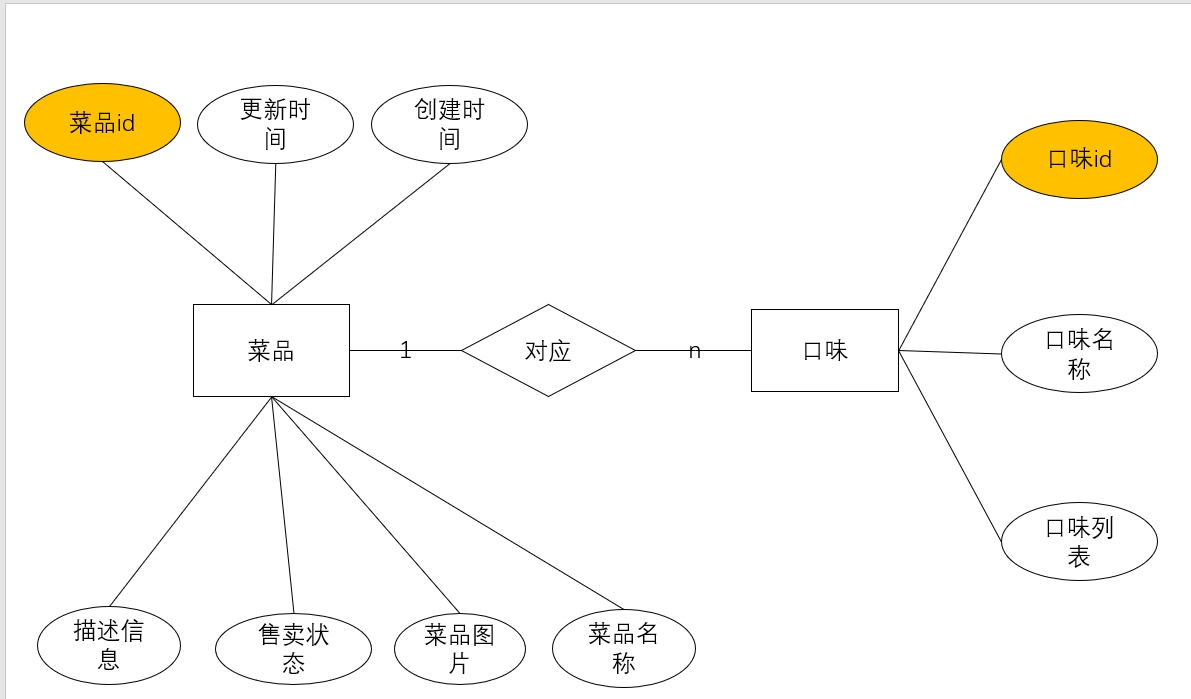


图3-3 菜品管理模块分E-R图

## 3.2 员工管理模块

### 3.2.1 模块功能描述

员工管理模块主要涉及的实体有员工、调度员。这里主要是对员工进行相应的信息管理。包括有员工工号、员工姓名、员工编号、手机号、账号状态、最后的操作时间等等。同时提供了增删查改功能。可以通过员工姓名进行模糊查询。通过手机账号进行查询，通过员工账号进行查询，还可以通过账号状态、最后操作时间进行查询。除了查询功能，还提供了重置功能。使得信息全部显示出来。员工与调度员是1对1的关系。

相关界面如下图3-4所示，查询功能如下图3-5所示。

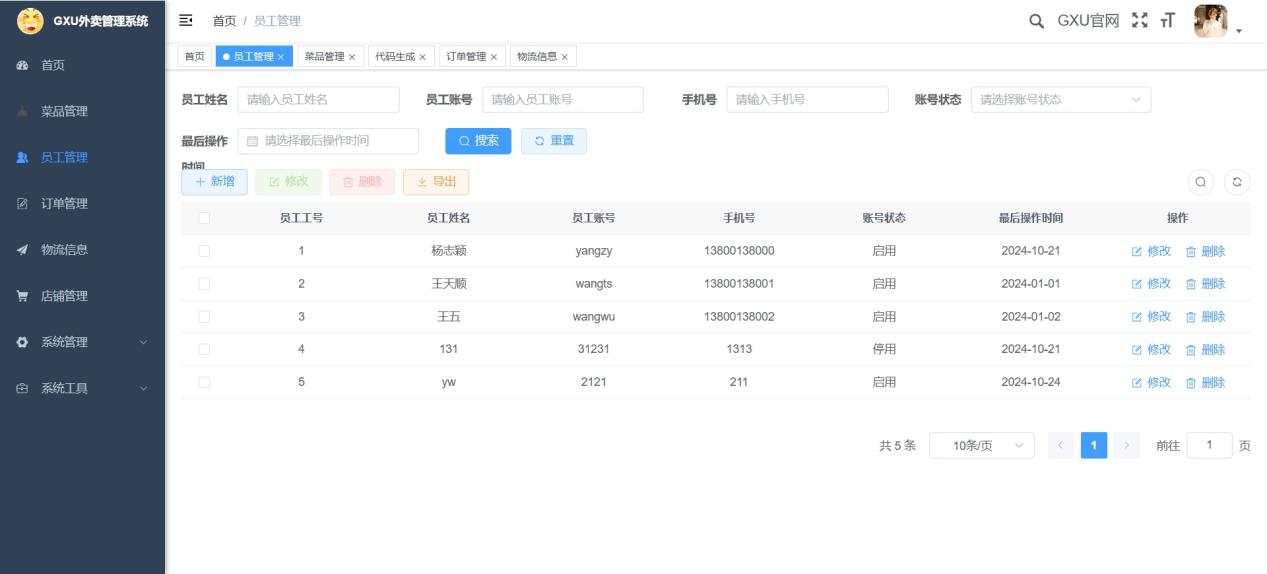


图3-4 员工管理模块图

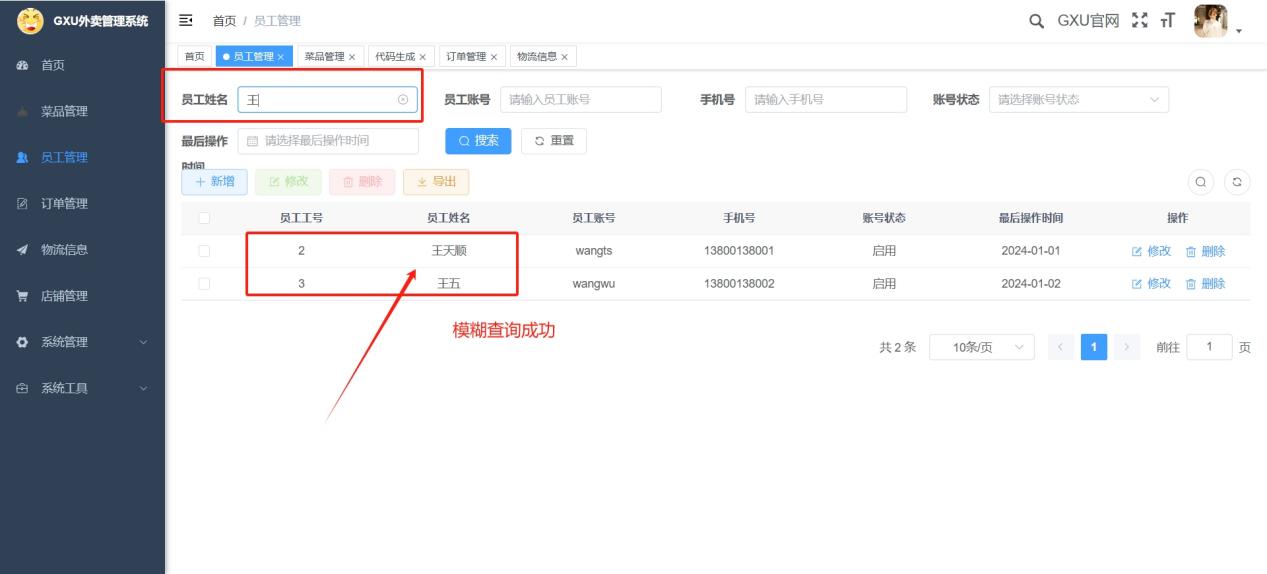


图3-5 员工管理搜索功能图

### 3.2.2 模块分E-R图

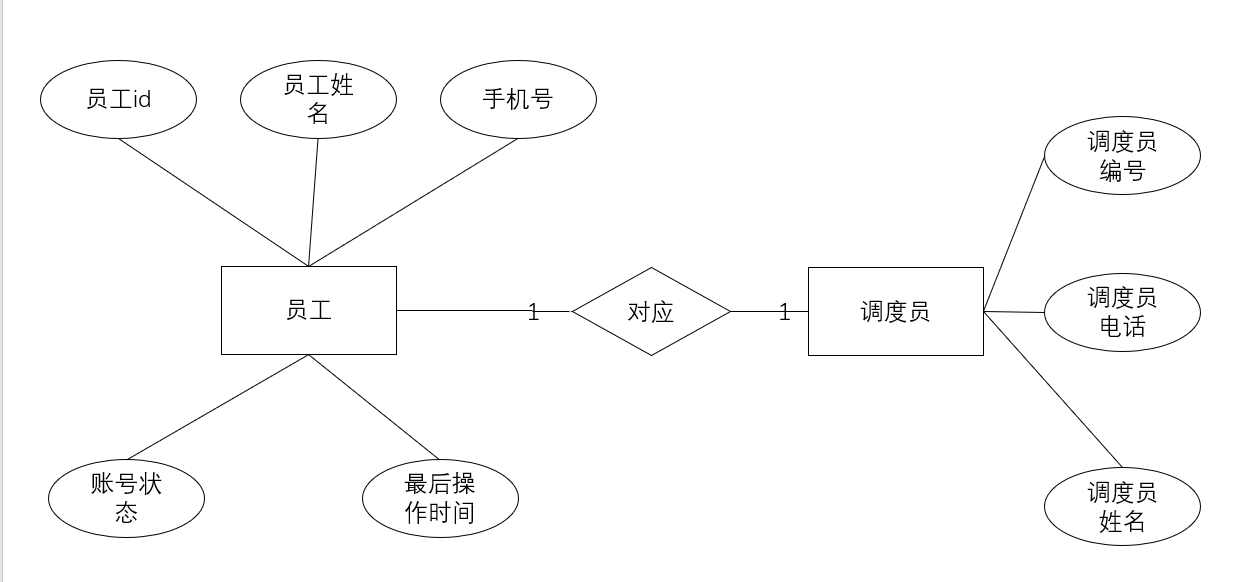


图3-6 员工管理模块ER图

## 3.3 订单管理模块

### 3.3.1 模块功能描述

相关界面如下图3-7所示，添加功能如下图3-8所示。

图3-7 订单管理模块图

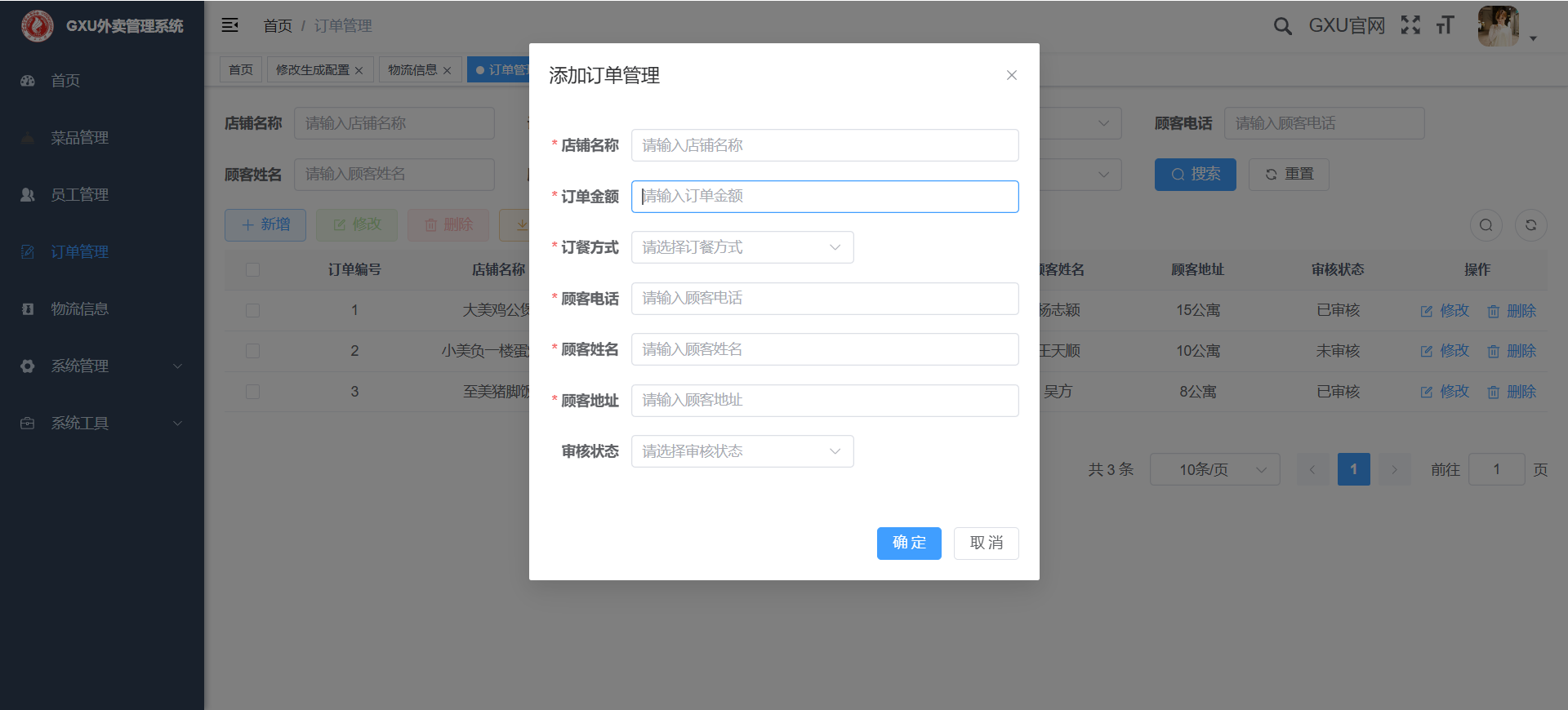


图3-8 订单管理模块操作图

· **订单创建与管理**：支持创建新的订单，包括设置店铺名称、订单金额、订餐方式（线上或线下）、顾客信息（电话、姓名、地址）等。

· **订单审核功能**：支持订单的审核状态管理，可标记订单为已审核或未审核。

· **订单查询与筛选**：根据订单编号、店铺名称、订餐方式或审核状态等条件查询订单信息。

· **订单历史记录**：记录每个订单的创建时间，并支持按时间范围查询历史订

### 3.3.2 模块分E-R图

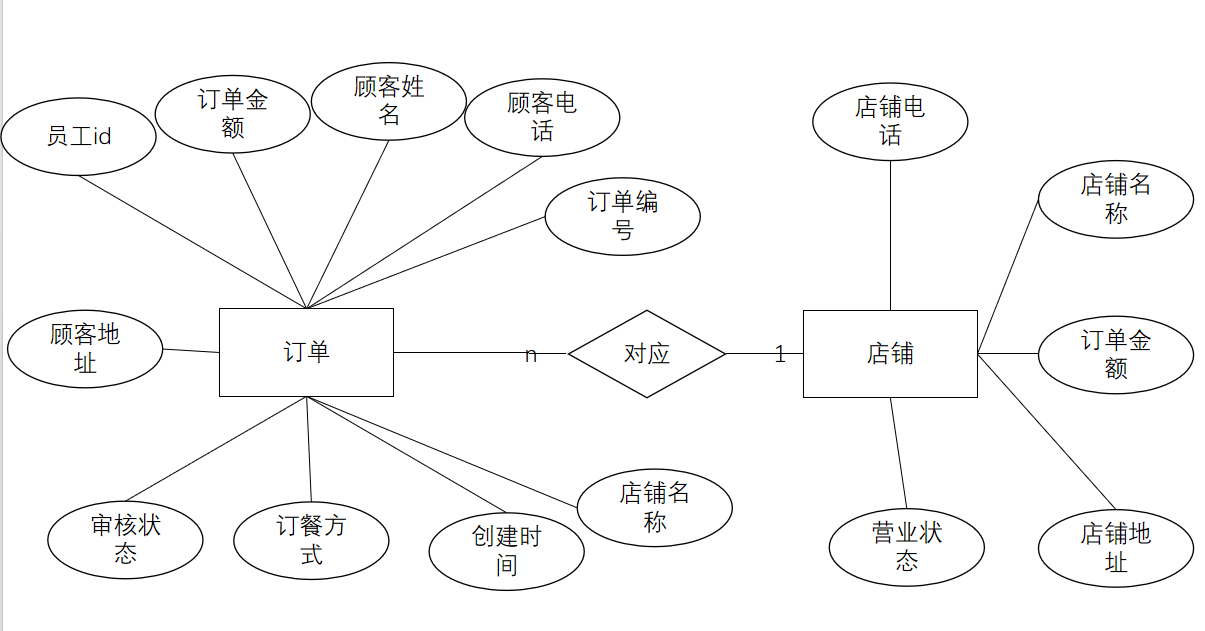
****

图3-9 订单管理模块ER图

## 3.4 物流信息模块

### 3.4.1 模块功能描述

相关界面如下图所示。

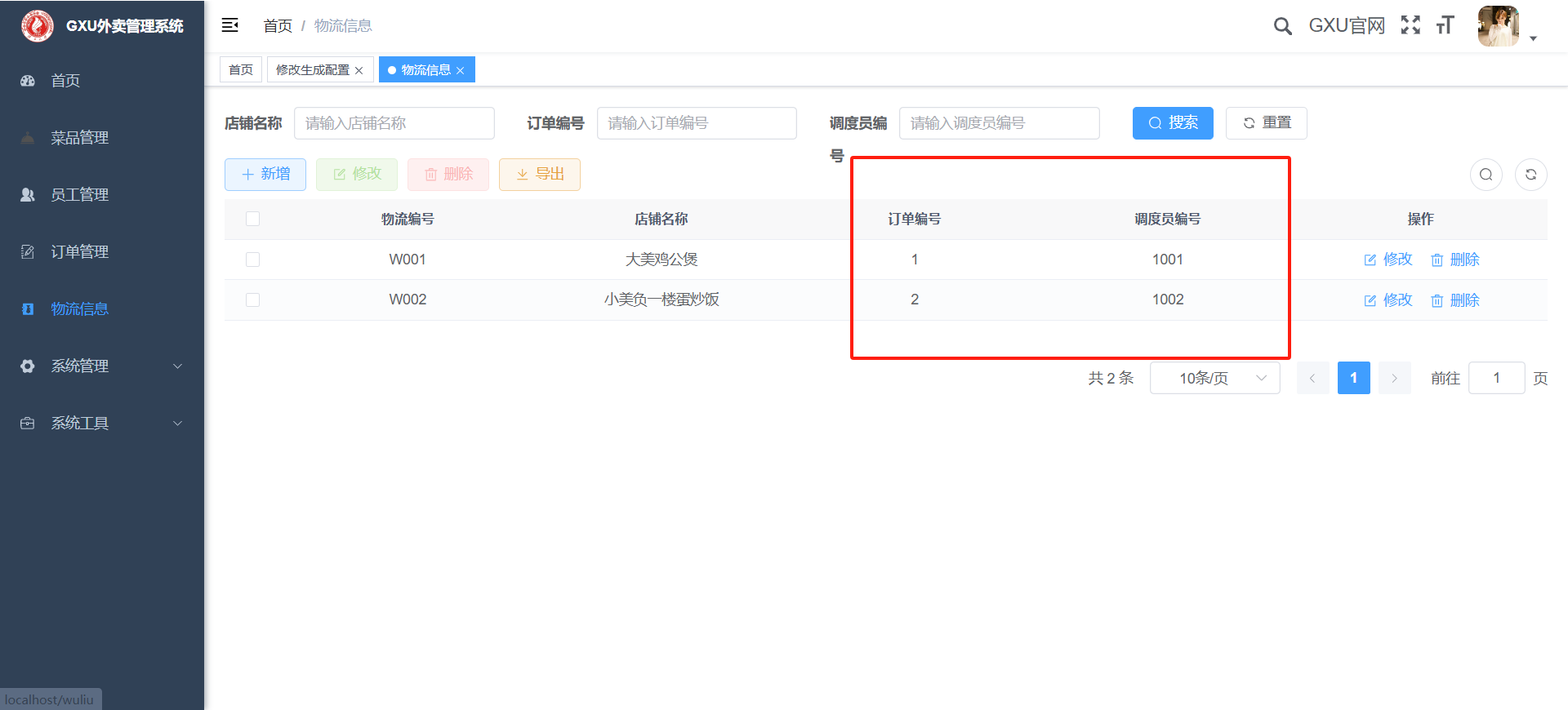


图3-10 物流管理模块图

· **物流记录管理**：管理每个订单的物流信息，包括物流编号、店铺名称、订单编号和调度员编号等。

· **调度员信息管理**：记录调度员的基本信息，如编号、姓名和电话，并支持添加、删除和更新调度员信息。

· **订单物流跟踪**：通过订单编号查询对应的物流状态，包括调度员的分配情况。

· **多条件查询**：根据物流编号、订单编号或调度员编号等条件查询物流信息。

### 3.4.2 模块分E-R图

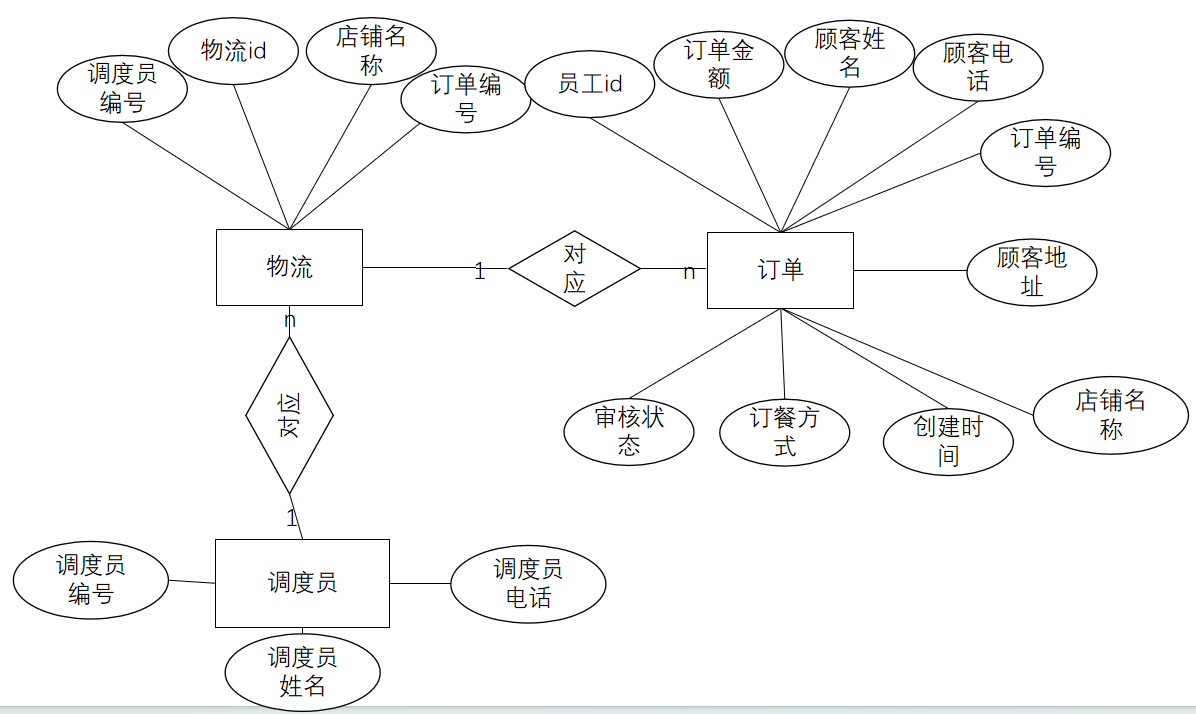
****

图3-11 物流管理模块ER图

## 3.5 店铺管理模块

### 3.5.1 模块功能描述

相关界面如下图所示。

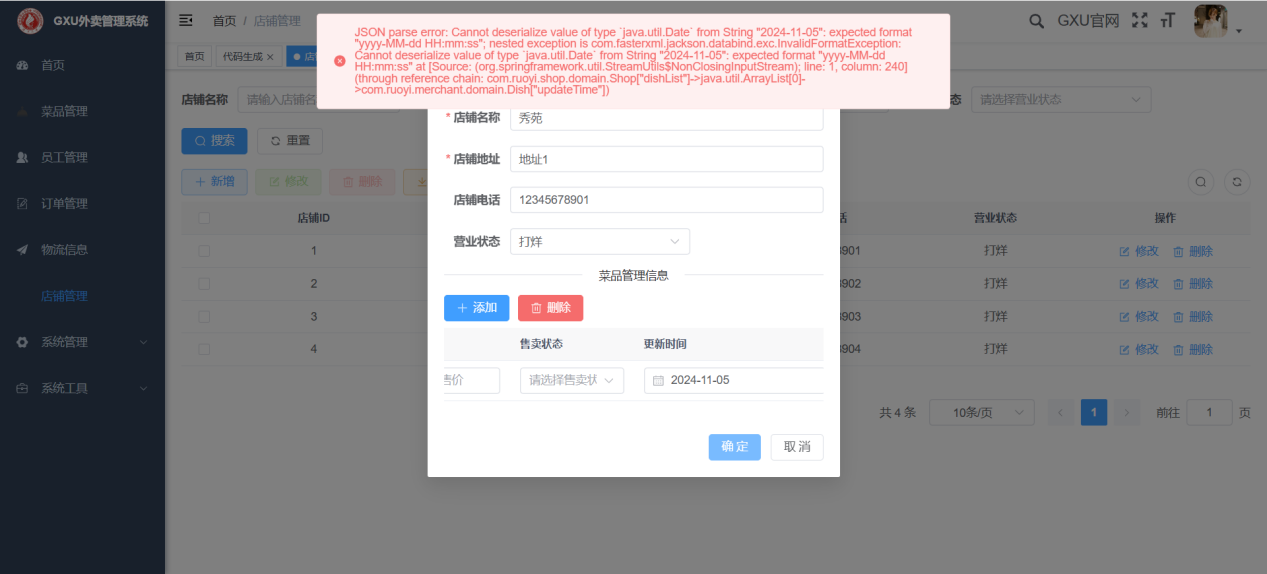


图3-12 店铺管理模块图

· **店铺基本信息维护**：管理店铺的名称、地址、电话等信息，并支持添加、修改和删除店铺。

· **店铺营业状态管理**：控制店铺的营业状态，可设置为“营业中”或“打烊”。

· **店铺查询与筛选**：根据店铺名称或状态查询店铺信息。

· **店铺与菜品关联管理**：关联每个店铺的菜品信息，支持查看店铺所供应的菜品列表。

### 3.5.2 模块分E-R图

模块分ER图如下图所示。

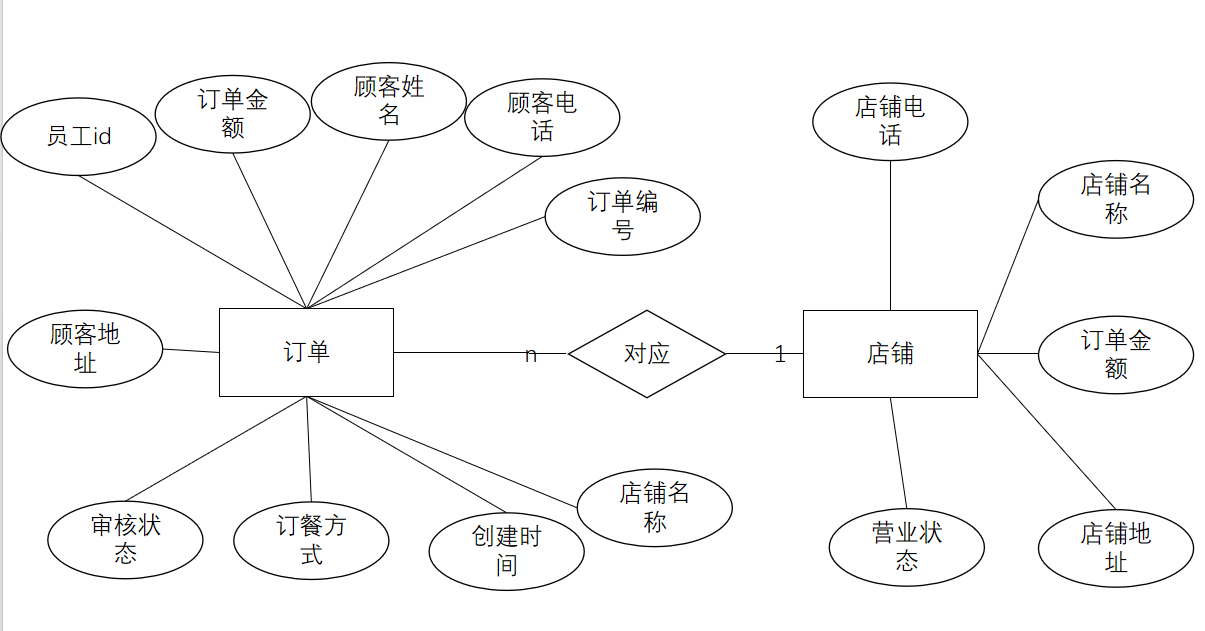
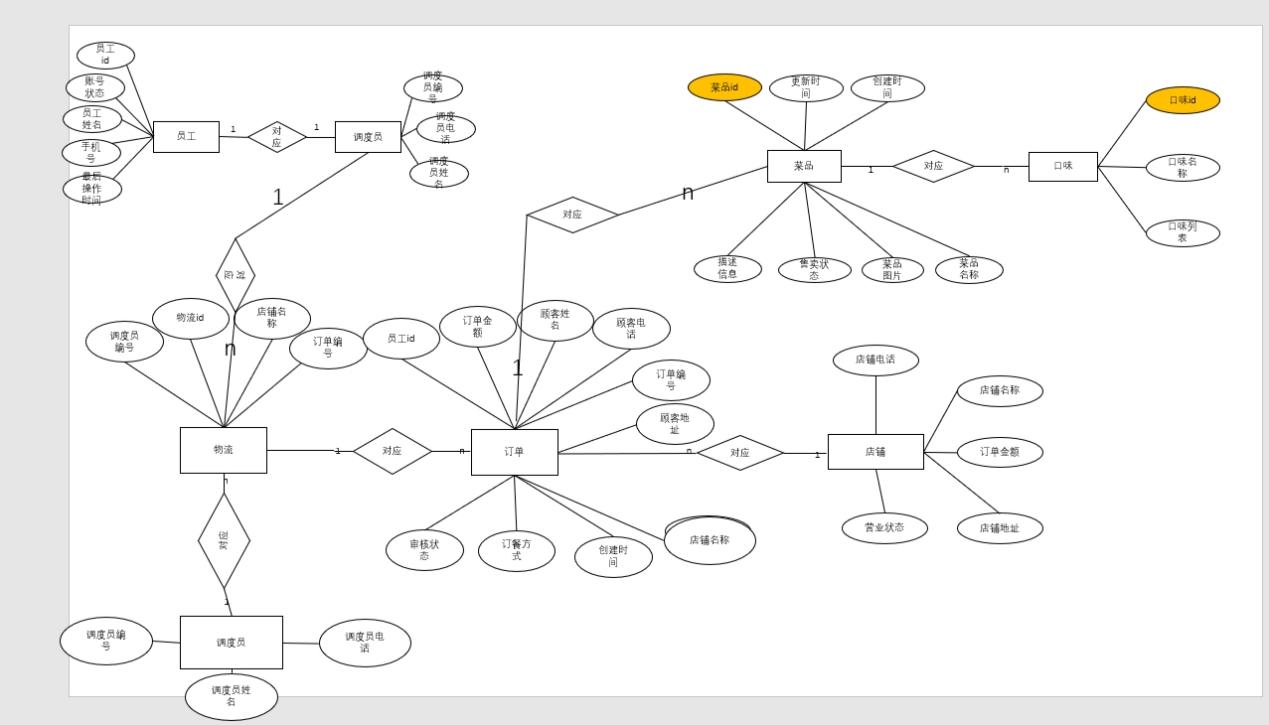
****

图3-6 店铺管理模块ER图

## 6总体E-R图



# （四）逻辑结构设计

数据库表：

表2.1 菜品表：tb\_dish

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 是否为空 | 备注 |
| id | Int | 10 | 否 | 自增(主键) |
| name | Varchar | 50 | 否 | 用户名 |
| price | Varchar | 50 | 否 | 价格 |
| image | Varchar | 50 | 否 | 图片 |
| m\_sale\_v | Int | 10 | 否 | 销售量 |
| status | Int | 1 | 否 | 状态0停售1起售 |
| shop\_id | Int | 10 | 否 | 外键  关联店铺 |

表2.2 口味表：tb\_dish\_flavor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 是否为空 | 备注 |
| id | Int | 10 | 否 | 自增(主键) |
| dish\_id | Int | 10 | 否 | 外键  关联菜品 |
| name | Varchar | 32 | 否 | 口味名 |
| value | json | 50 | 否 | 口味列表 |

表2.3 用户+管理员账户表：user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 是否为空 | 备注 |
| id | Int | 10 | 否 | 自增(主键) |
| username | Varchar | 50 | 否 | 用户名 |
| password | Varchar | 50 | 否 | 密码 |
| telephone | Varchar | 50 | 否 | 电话 |
| role | Int | 1 | 否 | 0表示用户，1表示管理员 |

表2.4 用户基本信息表：user\_msg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| id | Int | 10 | 否 | 外键，与user的主键id对应 |
| real\_name | Varchar | 50 | 否 | 真实姓名 |
| sex | Varchar | 50 | 否 | 性别 |
| age | Int | 10 | 否 | 年龄 |
| mail | Varchar | 50 | 否 | 邮箱 |
| phone | Varchar | 50 | 否 | 电话(主键) |
| user\_name | Varchar | 50 | 否 | 用户名 |

表2.5 店铺数据表：tb\_shop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| shop\_name | Varchar | 50 | 否 | 店铺名 |
| shop\_id | Int | 10 | 否 | 店铺号（主键） |
| shop\_address | Varchar | 20 | 否 | 店铺地址 |
| shop\_phone | Varchar | 20 | 否 | 店铺电话 |
| shop\_status | Int | 2 | 否 | 店铺状态  Default 1（1营业 0打烊） |

表2.6 送餐员数据表：dispatcher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| dispatcher\_id | Varchar | 50 | 否 | 送餐员编号(主键) |
| dispatcher\_name | Varchar | 50 | 否 | 送餐员姓名 |
| dispatcher\_phone | Varchar | 50 | 否 | 送餐员电话 |

表2.7 订餐方式数据表：orderway

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| orderway\_name | Varchar | 50 | 否 | 订餐方式(主键) |
| count | Int | 10 | 否 | 该种方式的订餐数量 |

表2.8 订单数据表：oorder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| order\_id | Int | 10 | 否 | 订单编号(主键) |
| shop\_name | Varchar | 50 | 否 | 店铺名称 |
| order\_money | Int | 10 | 否 | 订单金额 |
| order\_way | Varchar | 50 | 否 | 订餐方式 |
| cons\_phone | Varchar | 50 | 否 | 顾客电话 |
| cons\_name | Varchar | 50 | 否 | 顾客姓名 |
| cons\_addre | Varchar | 50 | 否 | 取餐地址 |
| checked | Int | 10 | 否 | 0表示未派发  1表示正在派发  2表示完成派发 |
| create\_time | Varchar | 50 | 否 | 下单时间 |

表2.9 物流数据表：wuliu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 备注 |
| order\_id | Int | 10 | 否 | 订单编号(主键) |
| cons\_phone | Varchar | 50 | 否 | 顾客电话 |
| disp\_id | Varchar | 50 | 否 | 送餐员编号 |
| deliver\_time | Varchar | 50 | 否 | 预计送餐时间 |
| ended | Int | 1 | 否 | 是否送餐结束 |

## 4.1 转换为关系数据模型

· 菜品表（tb\_dish）

关系模型: tb\_dish(id, name, price, image, m\_sale\_v, status, shop\_id)

主键: id

外键: shop\_id 引用 tb\_shop(shop\_id)

· 口味表（tb\_dish\_flavor）

关系模型: tb\_dish\_flavor(id, dish\_id, name, value)

主键: id

外键: dish\_id 引用 tb\_dish(id)

· 用户+管理员账户表（user）

关系模型: user(id, username, password, telephone, role)

主键: id

· 用户基本信息表（user\_msg）

关系模型: user\_msg(id, real\_name, sex, age, mail, phone, user\_name)

主键: id

外键: id 引用 user(id)

· 店铺数据表（tb\_shop）

关系模型: tb\_shop(shop\_id, shop\_name, shop\_address, shop\_phone, shop\_status)

主键: shop\_id

· 送餐员数据表（dispatcher）

关系模型: dispatcher(dispatcher\_id, dispatcher\_name, dispatcher\_phone)

主键: dispatcher\_id

· 订餐方式数据表（orderway）

关系模型: orderway(orderway\_name, count)

主键: orderway\_name

· 订单数据表（oorder）

关系模型: oorder(order\_id, shop\_name, order\_money, order\_way, cons\_phone, cons\_name, cons\_addre, checked, create\_time)

主键: order\_id

外键: shop\_name 可以连接到 tb\_shop(shop\_name)，order\_way 可连接到 orderway(orderway\_name)

· 物流数据表（wuliu）

关系模型: wuliu(order\_id, cons\_phone, disp\_id, deliver\_time, ended)

主键: order\_id

外键: order\_id 引用 oorder(order\_id), disp\_id 引用 dispatcher(dispatcher\_id)

**关系数据模型概要**

菜品表和店铺表之间通过 shop\_id 建立连接，表示每个菜品属于某个店铺。

订单表和订餐方式表通过 order\_way 建立连接，表示每个订单对应一种订餐方式。

物流表和订单表之间通过 order\_id 建立连接，表示每个物流记录与订单关联，同时通过 disp\_id 与送餐员表关联。

用户基本信息表与用户表通过 id 连接，提供用户的详细信息。

## 4.2 关系模式的优化

表结构分析与BCNF检测

· **菜品表 (tb\_dish)**

原始结构：

字段：id (主键)、name、price、image、m\_sale\_v、status、shop\_id（外键）

分析：

id 是主键，能够唯一标识每个菜品。其余字段都依赖于 id，没有部分依赖或传递依赖，因此该表符合BCNF。

结果：符合BCNF，无需进一步分解。

· **口味表 (tb\_dish\_flavor)**

原始结构：

字段：id (主键)、dish\_id（外键）、name、value（JSON存储口味列表）

分析：

主键为 id，字段 dish\_id 为外键，指向 tb\_dish 表的 id。name 和 value 完全依赖于 id。

因此，这张表符合BCNF，无需进一步分解。

结果：符合BCNF。

· **用户+管理员账户表 (user)**

原始结构：

字段：id (主键)、username、password、telephone、role

分析：

id 是主键，所有字段都依赖于 id，没有部分依赖或传递依赖。

该表符合BCNF。

结果：符合BCNF。

· **用户基本信息表 (user\_msg)**

原始结构：

字段：id（外键，与 user 表的 id 对应）、real\_name、sex、age、mail、phone（主键）、user\_name

分析：

phone 是主键，所有其他字段依赖于 phone。在此设计中，id 并非主键。

可将 id 作为主键使用，消除 phone 作为主键的依赖关系，以确保符合BCNF。

优化结果：

重新设定 id 为主键，将 phone 改为非主键字段。这样，该表符合BCNF。

· **店铺数据表 (tb\_shop)**

原始结构：

字段：shop\_id (主键)、shop\_name、shop\_address、shop\_phone、shop\_status

分析：

shop\_id 是主键，其他字段都依赖于 shop\_id，没有部分依赖或传递依赖。

该表符合BCNF。

结果：符合BCNF。

· **送餐员数据表 (dispatcher)**

原始结构：

字段：dispatcher\_id (主键)、dispatcher\_name、dispatcher\_phone

分析：

dispatcher\_id 是主键，其他字段完全依赖于 dispatcher\_id，符合BCNF。

结果：符合BCNF。

· **订餐方式数据表 (orderway)**

原始结构：

字段：orderway\_name (主键)、count

分析：

orderway\_name 是主键，count 依赖于 orderway\_name，符合BCNF。

结果：符合BCNF。

· **订单数据表 (oorder)**

原始结构：

字段：order\_id (主键)、shop\_name、order\_money、order\_way、cons\_phone、cons\_name、cons\_addre、checked、create\_time

分析：

主键为 order\_id，所有字段都依赖于 order\_id，符合BCNF。

结果：符合BCNF。

· **物流数据表 (wuliu)**

原始结构：

字段：order\_id (主键)、cons\_phone、disp\_id、deliver\_time、ended

分析：

order\_id 是主键，其他字段都依赖于 order\_id，符合BCNF。

结果：符合BCNF。

## 4.3 视图设计



图4-1 所有视图

视图用于简化查询、提供数据安全性和增强系统的易用性。以下是外卖管理系统中的一些视图设计示例：

**视图1：v\_active\_dishes**

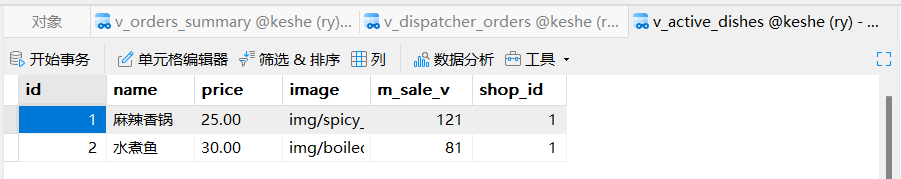
****

图4-2 v\_active\_dishes视图

目的：

v\_active\_dishes 视图用于获取所有当前处于“起售”状态的菜品信息。该视图仅包含有效销售的菜品数据，便于查询和显示在用户界面中。

SQL 语句：

CREATE VIEW v\_active\_dishes AS

SELECT id, name, price, image, m\_sale\_v, shop\_id

FROM tb\_dish

WHERE status = 1;

说明：

tb\_dish 表中 status 字段为 1 表示该菜品处于起售状态。

视图筛选出所有 status = 1 的记录，返回菜品的 id、name、price、image、m\_sale\_v (月销量) 和 shop\_id。

通过此视图，前端可以直接获取当前在售菜品信息，简化了条件查询。

**视图2：v\_orders\_summary**

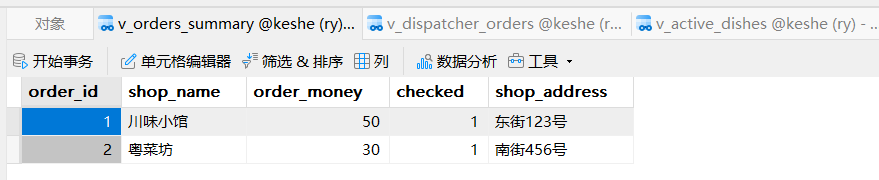
****

图4-3 v\_orders\_summary视图

目的：

v\_orders\_summary 视图提供订单和店铺基本信息的汇总，便于查询订单的总金额、状态及其关联的店铺地址。该视图适合用于订单管理和订单信息查询的场景。

SQL 语句：

CREATE VIEW v\_orders\_summary AS

SELECT oorder.order\_id, oorder.shop\_name, oorder.order\_money, oorder.checked, tb\_shop.shop\_address

FROM oorder

JOIN tb\_shop ON oorder.shop\_name = tb\_shop.shop\_name;

说明：

oorder 表和 tb\_shop 表通过 shop\_name 字段关联，连接查询显示了订单的编号、店铺名称、订单金额、订单派发状态和店铺地址。

该视图提供订单和店铺位置的汇总信息，使得查询订单基本信息时不必进行多表连接操作，从而简化查询逻辑。

checked 字段的值可用来判断订单的派发状态，0 表示未派发，1 表示正在派发，2 表示派发完成。

**视图3：v\_dispatcher\_orders**

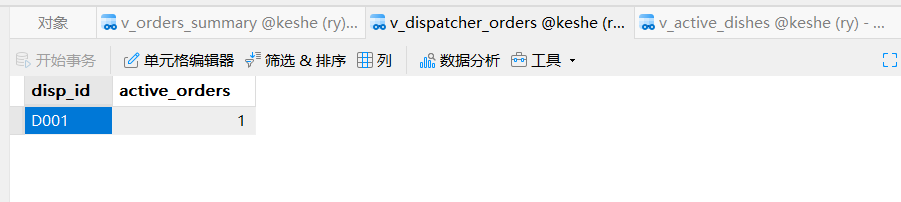
****

图4-4 v\_dispatcher\_orders视图

目的：

v\_dispatcher\_orders 视图用于获取每个送餐员的活跃订单数量，方便管理人员快速查询各送餐员的工作量。

SQL 语句：

CREATE VIEW v\_dispatcher\_orders AS

SELECT disp\_id, COUNT(order\_id) AS active\_orders

FROM wuliu

WHERE ended = 0

GROUP BY disp\_id;

说明：

wuliu 表中，ended 字段为 0 表示订单仍在进行中。

视图按照 disp\_id 对 order\_id 进行分组并统计未结束的订单数量，COUNT(order\_id) 返回每位送餐员的活跃订单数。

此视图适用于工作量分析，帮助管理员分配任务，合理调度送餐员。

# 数据库物理设计与实施

## 5.1 数据库应用的硬件、软件环境介绍

本系统数据库的硬件环境为PC端，系统运行在PC端。本系统数据库是MySQL，数据库的图形工具是Navicat，运行的操作系统环境为Windows11，软件使用IntelliJ IDEA和Visual Studio Code集成环境进行开发。

## 5.2 物理结构设计

（1）菜品表

菜品表的表名为tb\_dish，用于保存菜品的相关信息，菜品表的各字段设计与解释如下表5-1所示：

表5-1 菜品表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| id | 菜品id | bigint | / | 否 | 主键 |
| name | 菜品名称 | VarCHAR | 32 | 否 |  |
| price | 菜品价格 | decimal(10,2) |  | 否 |  |
| image | 图片 | varchar | 255 | 否 |  |
| description | 描述信息 | varchar | 255 | 否 |  |
| status | 售卖状态 | int | / | 否 |  |
| Create\_time | 创建时间 | datetime | / | 是 |  |
| Update\_time | 更新时间 | datetime | / | 是 |  |

（2）口味表

口味表的表名为tb\_dish\_flavor，用于保存菜品口味的相关信息，口味表的各字段设计与解释如下表5-2所示：

表5-2 菜品口味表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| id | 口味id | bigint | / | 否 | 主键 |
| dish\_id | 菜品id | bigint | / | 否 |  |
| name | 口味名称 | varchar | 32 | 否 |  |
| Value | 口味列表 | varchar | 255 | 否 |  |

（3）员工管理表

员工管理表的表名为employee\_management，用于保存员工管理的相关信息，员工管理表的各字段设计与解释如下表5-3所示：

表5-3 员工管理表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| employee\_id | 员工id | bigint | / | 否 | 主键 |
| employee\_name | 员工姓名 | varchar | 100 | 否 |  |
| account | 员工账号 | varchar | 50 | 否 |  |
| phone\_number | 手机号 | varchar | 15 | 否 |  |
| account\_status | 账号状态 | int | / | 否 |  |
| last\_operation\_time | 最后操作时间 | datetime |  | 否 |  |

（4）订单管理表

订单管理表的表名为oorder，用于保存订单管理的相关信息，订单管理表的各字段设计与解释如下表5-4所示：

表5-4 订单管理表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| order\_id | 订单编号 | int | / | 否 | 主键 |
| shop\_name | 店铺名称 | varchar | 50 | 否 |  |
| order\_money | 订单金额 | int | / | 否 |  |
| order\_way | 订餐方式 | int | / | 否 |  |
| cons\_phone | 顾客电话 | varchar | 50 | 否 |  |
| cons\_name | 顾客姓名 | varchar | 50 | 否 |  |
| cons\_addre | 顾客地址 | varchar | 50 | 否 |  |
| checked | 审核状态 | int | / | 否 |  |
| create\_time | 创建时间 | datetime | / | 否 |  |

（5）店铺管理表

店铺管理表的表名为tb\_shop，用于保存店铺管理的相关信息，店铺管理表的各字段设计与解释如下表5-5所示：

表5-5 店铺管理表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| shop\_id | 店铺ID | bigint | / | 否 | 主键 |
| shop\_name | 店铺名称 | varchar | 50 | 否 |  |
| shop\_address | 店铺地址 | varchar | 255 | 否 |  |
| shop\_phone | 店铺电话 | varchar | 20 | 否 |  |
| shop\_status | 营业状态 | int | / | 否 |  |

（6）物流管理表

物流管理表的表名为wuliu，用于保存物流管理的相关信息，物流管理表的各字段设计与解释如下表5-6所示：

表5-6 物流管理表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| wuliu\_id | 物流编号 | varchar | 50 | 否 | 主键 |
| shop\_name | 店铺名称 | varchar | 50 | 否 |  |
| order\_id | 订单编号 | Int | / | 否 |  |
| dispatcher\_id | 调度员编号 | varchar | 50 | 否 |  |

（7）调度员表

调度员表的表名为dispatcher，用于保存调度员的相关信息，调度员表的各字段设计与解释如下表5-7所示：

表5-7 调度员表各字段设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段意义 | 字段类型 | 字段长度 | 允许为空 | 完整性约束 |
| dispatcher\_id | 调度员编号 | varchar | 50 | 否 | 主键 |
| dispatcher\_name | 调度员姓名 | varchar | 50 | 否 |  |
| dispatcher\_phone | 调度员电话号 | varchar | 50 | 否 |  |

## 5.3 物理结构优化

为外卖管理系统设计合理的索引、函数、存储过程、触发器、事务和权限控制，以提高数据库的性能、安全性和维护性。以下是每个部分的详细设计。

### 1. 索引设计

创建适当的索引可以加快查询速度，尤其是在频繁查询的列上。

**tb\_dish 表：**

索引：idx\_shop\_id，字段 shop\_id，用于加速根据店铺ID查询菜品。

索引：idx\_status，字段 status，加速菜品状态查询，如查询停售或在售的菜品。



图5-1 tb\_dish 表索引

**tb\_dish\_flavor 表：**

索引：idx\_dish\_id，字段 dish\_id，用于加速菜品与口味的关联查询。



图5-2 tb\_dish\_flavor 表索引

**user 表：**

索引：idx\_username，字段 username，用于加速用户名查找（登录时）。

索引：idx\_role，字段 role，用于加速用户角色的查询，区分用户与管理员。

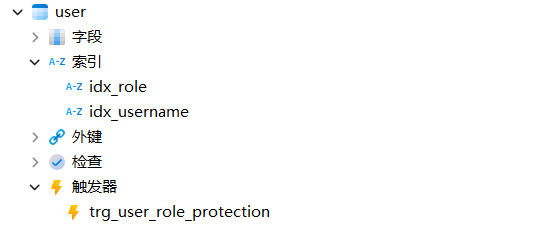


图5-2 user 表索引

**user\_msg 表：**

索引：idx\_phone，字段 phone，用于加速通过电话查找用户信息。

索引：idx\_user\_name，字段 user\_name，用于快速查找用户名。



图5-3 user\_msg 表索引

**tb\_shop 表：**

索引：idx\_shop\_status，字段 shop\_status，加速查询店铺状态，如查询营业或打烊的店铺。



图5-4 tb\_shop 表索引

**oorder 表：**

索引：idx\_shop\_name，字段 shop\_name，用于根据店铺名称查询订单。

索引：idx\_order\_way，字段 order\_way，用于加速不同订餐方式的订单查询。

索引：idx\_cons\_phone，字段 cons\_phone，用于加速通过顾客电话查询订单记录。

索引：idx\_checked，字段 checked，加速查询订单派发状态。

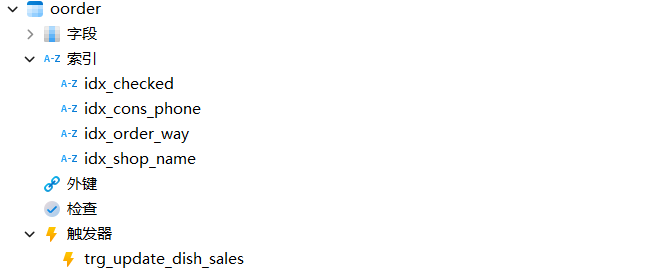


图5-5 oorder 表索引

**wuliu 表：**

索引：idx\_disp\_id，字段 disp\_id，用于加速查询送餐员的物流信息。

索引：idx\_ended，字段 ended，用于加速是否送餐结束的查询。

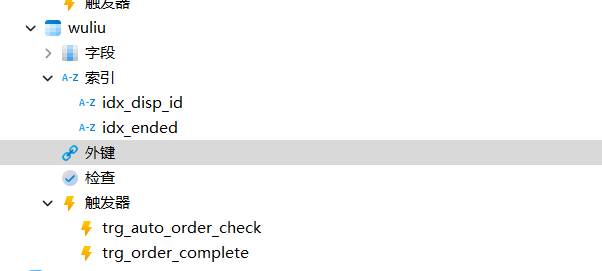


图5-6 wuliu 表索引

### 2. 视图设计

视图可以简化复杂的查询，并为管理者提供易于理解的数据展示。

视图：v\_active\_dishes

用途：显示所有在售的菜品。

SQL：

CREATE VIEW v\_active\_dishes AS

SELECT id, name, price, image, m\_sale\_v, shop\_id

FROM tb\_dish

WHERE status = 1;

视图：v\_orders\_summary

用途：显示订单的概要信息，包括店铺名、订单金额和派发状态。

SQL：

CREATE VIEW v\_orders\_summary AS

SELECT oorder.order\_id, oorder.shop\_name, oorder.order\_money, oorder.checked, tb\_shop.shop\_address

FROM oorder

JOIN tb\_shop ON oorder.shop\_name = tb\_shop.shop\_name;

视图：v\_dispatcher\_orders

用途：显示每位送餐员当前派发的订单数量。

SQL：

CREATE VIEW v\_dispatcher\_orders AS

SELECT disp\_id, COUNT(order\_id) AS active\_orders

FROM wuliu

WHERE ended = 0

GROUP BY disp\_id;

### **3. 存储过程设计**

存储过程用于封装复杂的操作，可以提高执行效率和安全性。

DELIMITER //

CREATE PROCEDURE AddNewOrder(

IN p\_shop\_name VARCHAR(50),

IN p\_order\_money DECIMAL(10, 2),

IN p\_order\_way VARCHAR(50),

IN p\_cons\_phone VARCHAR(50),

IN p\_cons\_name VARCHAR(50),

IN p\_cons\_addre VARCHAR(50),

IN p\_checked INT,

IN p\_create\_time DATETIME

)

BEGIN

DECLARE new\_order\_id INT;

-- 插入订单记录到 oorder 表

INSERT INTO oorder (shop\_name, order\_money, order\_way, cons\_phone, cons\_name, cons\_addre, checked, create\_time)

VALUES (p\_shop\_name, p\_order\_money, p\_order\_way, p\_cons\_phone, p\_cons\_name, p\_cons\_addre, p\_checked, p\_create\_time);

-- 获取新插入的订单 ID

SET new\_order\_id = LAST\_INSERT\_ID();

-- 返回订单 ID

SELECT new\_order\_id AS order\_id;

END //

DELIMITER ;

**3. 触发器设计**

触发器用于在数据插入、更新或删除时自动执行特定操作，以维护数据一致性和自动化操作。

**触发器：trg\_update\_dish\_sales**

用途：在 tb\_dish 表中每次插入新订单时，自动更新菜品的销售量。

SQL：

CREATE TRIGGER trg\_update\_dish\_sales

AFTER INSERT ON oorder

FOR EACH ROW

BEGIN

UPDATE tb\_dish

SET m\_sale\_v = m\_sale\_v + 1

WHERE id = NEW.order\_id;

END;

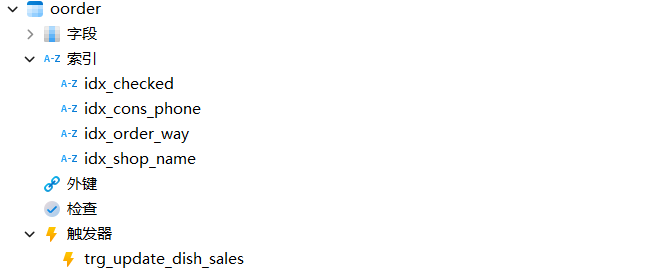


图5-7 trg\_update\_dish\_sales 触发器

**触发器：trg\_auto\_order\_check**

用途：在 wuliu 表中插入新物流信息时，自动更新订单的派发状态为“正在派发”。

SQL：

CREATE TRIGGER trg\_auto\_order\_check

AFTER INSERT ON wuliu

FOR EACH ROW

BEGIN

UPDATE oorder

SET checked = 1

WHERE order\_id = NEW.order\_id;

END;

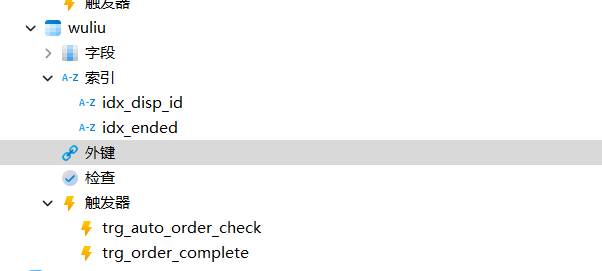


图5-8 trg\_auto\_order\_check 触发器

**触发器：trg\_order\_complete**

用途：当物流信息中的送餐结束标志 (ended) 更新为1时，自动将订单状态更新为“完成派发”。

SQL：

CREATE TRIGGER trg\_order\_complete

AFTER UPDATE ON wuliu

FOR EACH ROW

BEGIN

IF NEW.ended = 1 THEN

UPDATE oorder

SET checked = 2

WHERE order\_id = NEW.order\_id;

END IF;

END;

**触发器：trg\_user\_role\_protection**

用途：防止直接将普通用户的角色字段更新为管理员。

SQL：

CREATE TRIGGER trg\_user\_role\_protection

BEFORE UPDATE ON user

FOR EACH ROW

BEGIN

IF OLD.role = 0 AND NEW.role = 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot change user role to admin directly';

END IF;

END;

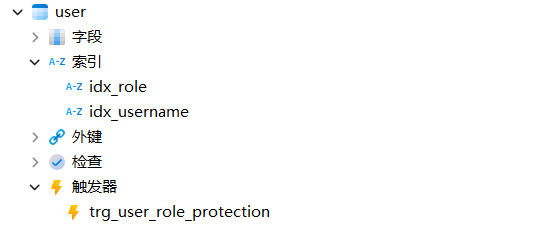


图5-9 trg\_user\_role\_protection 触发器

### 4. 触发器设计

触发器用于在数据插入、更新或删除时自动执行特定操作，以维护数据一致性和自动化操作。

**触发器：trg\_update\_dish\_sales**

用途：在 tb\_dish 表中每次插入新订单时，自动更新菜品的销售量。

SQL：

CREATE TRIGGER trg\_update\_dish\_sales

AFTER INSERT ON oorder

FOR EACH ROW

BEGIN

UPDATE tb\_dish

SET m\_sale\_v = m\_sale\_v + 1

WHERE id = NEW.order\_id;

END;

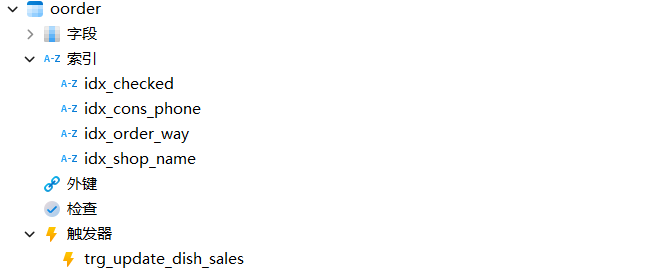


图5-7 trg\_update\_dish\_sales 触发器

**触发器：trg\_auto\_order\_check**

用途：在 wuliu 表中插入新物流信息时，自动更新订单的派发状态为“正在派发”。

SQL：

CREATE TRIGGER trg\_auto\_order\_check

AFTER INSERT ON wuliu

FOR EACH ROW

BEGIN

UPDATE oorder

SET checked = 1

WHERE order\_id = NEW.order\_id;

END;

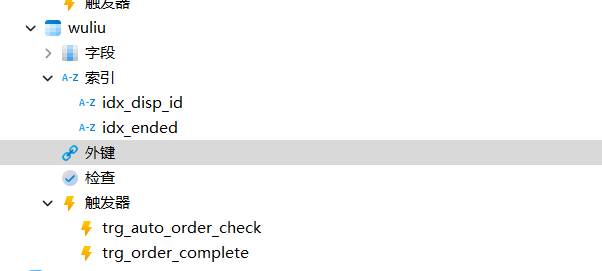


图5-8 trg\_auto\_order\_check 触发器

**触发器：trg\_order\_complete**

用途：当物流信息中的送餐结束标志 (ended) 更新为1时，自动将订单状态更新为“完成派发”。

SQL：

CREATE TRIGGER trg\_order\_complete

AFTER UPDATE ON wuliu

FOR EACH ROW

BEGIN

IF NEW.ended = 1 THEN

UPDATE oorder

SET checked = 2

WHERE order\_id = NEW.order\_id;

END IF;

END;

**触发器：trg\_user\_role\_protection**

用途：防止直接将普通用户的角色字段更新为管理员。

SQL：

CREATE TRIGGER trg\_user\_role\_protection

BEFORE UPDATE ON user

FOR EACH ROW

BEGIN

IF OLD.role = 0 AND NEW.role = 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot change user role to admin directly';

END IF;

END;

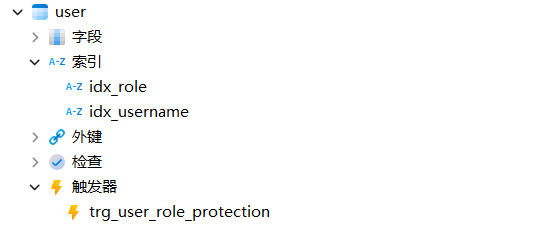


图5-9 trg\_user\_role\_protection 触发器

### 5. 权限控制设计

权限控制用于确保不同角色的用户只能执行他们被授权的操作。根据角色定义权限策略：

* **管理员**：具有所有权限，包括用户管理、订单管理、店铺管理等。
* **用户**：只能查看和操作自己的订单。

-- 为用户授予对其订单的查询权限GRANT SELECT ON Orders TO 'user'@'localhost';

通过这些设计，可以提高外卖管理系统的数据库性能和安全性，确保数据的完整性和用户权限的合理分配。

## 5.4 数据库实施

### 1. 环境准备

* 硬件要求：至少 8GB RAM, 4 CPU cores。
* 软件要求：
  + 操作系统：Linux Ubuntu 20.04 或 Windows 10。
  + 数据库管理系统：MySQL 8.0。

### 2.数据库创建

-- 菜品表

CREATE TABLE tb\_dish (

    id INT(10) AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(50) NOT NULL,

    price VARCHAR(50) NOT NULL,

    image VARCHAR(50) NOT NULL,

    m\_sale\_v INT(10) NOT NULL,

    status INT(1) NOT NULL COMMENT '状态0停售1起售',

    shop\_id INT(10) NOT NULL

);

-- 口味表

CREATE TABLE tb\_dish\_flavor (

    id INT(10) AUTO\_INCREMENT PRIMARY KEY,

    dish\_id INT(10) NOT NULL,

    name VARCHAR(32) NOT NULL,

    value JSON NOT NULL

);

-- 用户+管理员账户表

CREATE TABLE user (

    id INT(10) AUTO\_INCREMENT PRIMARY KEY,

    username VARCHAR(50) NOT NULL,

    password VARCHAR(50) NOT NULL,

    telephone VARCHAR(50) NOT NULL,

    role INT(1) NOT NULL COMMENT '0表示用户，1表示管理员'

);

-- 用户基本信息表

CREATE TABLE user\_msg (

    id INT(10) NOT NULL PRIMARY KEY,

    real\_name VARCHAR(50) NOT NULL,

    sex VARCHAR(50) NOT NULL,

    age INT(10) NOT NULL,

    mail VARCHAR(50) NOT NULL,

    phone VARCHAR(50) NOT NULL,

    user\_name VARCHAR(50) NOT NULL

);

-- 店铺数据表

CREATE TABLE tb\_shop (

    shop\_name VARCHAR(50) NOT NULL,

    shop\_id INT(10) AUTO\_INCREMENT PRIMARY KEY,

    shop\_address VARCHAR(20) NOT NULL,

    shop\_phone VARCHAR(20) NOT NULL,

    shop\_status INT(2) NOT NULL DEFAULT 1 COMMENT '1营业 0打烊'

);

-- 送餐员数据表

CREATE TABLE dispatcher (

    dispatcher\_id VARCHAR(50) PRIMARY KEY,

    dispatcher\_name VARCHAR(50) NOT NULL,

    dispatcher\_phone VARCHAR(50) NOT NULL

);

-- 订餐方式数据表

CREATE TABLE orderway (

    orderway\_name VARCHAR(50) PRIMARY KEY,

    count INT(10) NOT NULL

);

-- 订单数据表

CREATE TABLE oorder (

    order\_id INT(10) AUTO\_INCREMENT PRIMARY KEY,

    shop\_name VARCHAR(50) NOT NULL,

    order\_money INT(10) NOT NULL,

    order\_way VARCHAR(50) NOT NULL,

    cons\_phone VARCHAR(50) NOT NULL,

    cons\_name VARCHAR(50) NOT NULL,

    cons\_addre VARCHAR(50) NOT NULL,

    checked INT(10) NOT NULL COMMENT '0表示未派发，1表示正在派发，2表示完成派发',

    create\_time VARCHAR(50) NOT NULL

);

-- 物流数据表

CREATE TABLE wuliu (

    order\_id INT(10) PRIMARY KEY,

    cons\_phone VARCHAR(50) NOT NULL,

    disp\_id VARCHAR(50) NOT NULL,

    deliver\_time VARCHAR(50) NOT NULL,

    ended INT(1) NOT NULL COMMENT '是否送餐结束'

);

### 3.数据导入

-- 插入 tb\_shop 表数据 (店铺数据)

INSERT INTO tb\_shop (shop\_name, shop\_id, shop\_address, shop\_phone, shop\_status) VALUES

('川味小馆', 1, '东街123号', '12345678901', 1),

('粤菜坊', 2, '南街456号', '12345678902', 1),

('湘菜馆', 3, '西街789号', '12345678903', 0);

-- 插入 tb\_dish 表数据 (菜品数据)

INSERT INTO tb\_dish (name, price, image, m\_sale\_v, status, shop\_id) VALUES

('麻辣香锅', '25.00', 'img/spicy\_pot.jpg', 120, 1, 1),

('水煮鱼', '30.00', 'img/boiled\_fish.jpg', 80, 1, 1),

('清蒸鱼', '35.00', 'img/steamed\_fish.jpg', 50, 0, 2);

-- 插入 tb\_dish\_flavor 表数据 (口味数据)

INSERT INTO tb\_dish\_flavor (dish\_id, name, value) VALUES

(1, '辣度', JSON\_ARRAY('微辣', '中辣', '重辣')),

(2, '咸度', JSON\_ARRAY('少盐', '正常', '多盐'));

-- 插入 user 表数据 (用户+管理员账户)

INSERT INTO user (username, password, telephone, role) VALUES

('user1', 'pass123', '13812345678', 0),

('admin1', 'adminpass', '13887654321', 1);

-- 插入 user\_msg 表数据 (用户基本信息)

INSERT INTO user\_msg (id, real\_name, sex, age, mail, phone, user\_name) VALUES

(1, '张三', '男', 25, 'zhangsan@example.com', '13812345678', 'user1'),

(2, '管理员', '男', 30, 'admin@example.com', '13887654321', 'admin1');

-- 插入 dispatcher 表数据 (送餐员数据)

INSERT INTO dispatcher (dispatcher\_id, dispatcher\_name, dispatcher\_phone) VALUES

('D001', '王小明', '13712345678'),

('D002', '李小红', '13787654321');

-- 插入 orderway 表数据 (订餐方式)

INSERT INTO orderway (orderway\_name, count) VALUES

('堂食', 50),

('外带', 70),

('外卖', 100);

-- 插入 oorder 表数据 (订单数据)

INSERT INTO oorder (order\_id, shop\_name, order\_money, order\_way, cons\_phone, cons\_name, cons\_addre, checked, create\_time) VALUES

(1, '川味小馆', 50, '外卖', '13812345678', '张三', '东街123号', 0, '2024-10-25 12:00:00'),

(2, '粤菜坊', 30, '堂食', '13887654321', '李四', '南街456号', 1, '2024-10-26 13:00:00');

-- 插入 wuliu 表数据 (物流数据)

INSERT INTO wuliu (order\_id, cons\_phone, disp\_id, deliver\_time, ended) VALUES

(1, '13812345678', 'D001', '2024-10-25 12:30:00', 0),

(2, '13887654321', 'D002', '2024-10-26 13:30:00', 1);

### 索引和视图的创建

-- tb\_dish 表索引

CREATE INDEX idx\_shop\_id ON tb\_dish (shop\_id);

CREATE INDEX idx\_status ON tb\_dish (status);

-- tb\_dish\_flavor 表索引

CREATE INDEX idx\_dish\_id ON tb\_dish\_flavor (dish\_id);

-- user 表索引

CREATE INDEX idx\_username ON user (username);

CREATE INDEX idx\_role ON user (role);

-- user\_msg 表索引

CREATE INDEX idx\_phone ON user\_msg (phone);

CREATE INDEX idx\_user\_name ON user\_msg (user\_name);

-- tb\_shop 表索引

CREATE INDEX idx\_shop\_status ON tb\_shop (shop\_status);

-- oorder 表索引

CREATE INDEX idx\_shop\_name ON oorder (shop\_name);

CREATE INDEX idx\_order\_way ON oorder (order\_way);

CREATE INDEX idx\_cons\_phone ON oorder (cons\_phone);

CREATE INDEX idx\_checked ON oorder (checked);

-- wuliu 表索引

CREATE INDEX idx\_disp\_id ON wuliu (disp\_id);

CREATE INDEX idx\_ended ON wuliu (ended);

-- 视图：v\_active\_dishes

CREATE VIEW v\_active\_dishes AS

SELECT id, name, price, image, m\_sale\_v, shop\_id

FROM tb\_dish

WHERE status = 1;

-- 视图：v\_orders\_summary

CREATE VIEW v\_orders\_summary AS

SELECT oorder.order\_id, oorder.shop\_name, oorder.order\_money, oorder.checked, tb\_shop.shop\_address

FROM oorder

JOIN tb\_shop ON oorder.shop\_name = tb\_shop.shop\_name;

-- 视图：v\_dispatcher\_orders

CREATE VIEW v\_dispatcher\_orders AS

SELECT disp\_id, COUNT(order\_id) AS active\_orders

FROM wuliu

WHERE ended = 0

GROUP BY disp\_id;

### 5. 存储过程和触发器的实现

-- 触发器：trg\_update\_dish\_sales

DELIMITER //

CREATE TRIGGER trg\_update\_dish\_sales

AFTER INSERT ON oorder

FOR EACH ROW

BEGIN

    UPDATE tb\_dish

    SET m\_sale\_v = m\_sale\_v + 1

    WHERE id = NEW.order\_id;

END;

//

DELIMITER ;

-- 触发器：trg\_auto\_order\_check

DELIMITER //

CREATE TRIGGER trg\_auto\_order\_check

AFTER INSERT ON wuliu

FOR EACH ROW

BEGIN

    UPDATE oorder

    SET checked = 1

    WHERE order\_id = NEW.order\_id;

END;

//

DELIMITER ;

-- 触发器：trg\_order\_complete

DELIMITER //

CREATE TRIGGER trg\_order\_complete

AFTER UPDATE ON wuliu

FOR EACH ROW

BEGIN

    IF NEW.ended = 1 THEN

        UPDATE oorder

        SET checked = 2

        WHERE order\_id = NEW.order\_id;

    END IF;

END;

//

DELIMITER ;

-- 触发器：trg\_user\_role\_protection

DELIMITER //

CREATE TRIGGER trg\_user\_role\_protection

BEFORE UPDATE ON user

FOR EACH ROW

BEGIN

    IF OLD.role = 0 AND NEW.role = 1 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = 'Cannot change user role to admin directly';

    END IF;

END;

//

DELIMITER ;

# 应用系统的设计与实施

## 6.1 连接数据库

1、**数据源配置**

数据源类型：使用 Druid 数据源，类名为 com.alibaba.druid.pool.DruidDataSource。

1. **数据库驱动**

使用 MySQL JDBC 驱动，类名为 com.mysql.cj.jdbc.Driver。

3、**主库配置**：

URL：连接到数据库的 URL 为 jdbc:mysql://localhost:3306/ry-vue?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT%2B8。

用户名：root。

密码：123456。

4、**从库配置**：

目前未启用，从库的相关信息为空（enabled: false），如果需要配置从库，需填入其 URL、用户名和密码。

5、**连接池参数：**

初始连接数：5

最小连接池数量：10

最大连接池数量：20

连接获取等待超时：60000 毫秒（1分钟）

连接超时时间：30000 毫秒（30秒）

网络超时时间：60000 毫秒（1分钟）

6、**连接检测配置**：

间隔检测时间：60000 毫秒（1分钟）

最小生存时间：300000 毫秒（5分钟）

最大生存时间：900000 毫秒（15分钟）

验证查询：SELECT 1 FROM DUAL

7、**其他配置**：

启用网页统计过滤器和管理控制台。

设置了访问控制（白名单），控制台管理的用户名和密码为 ruoyi 和 123456。

记录慢 SQL，并设定慢 SQL 的阈值为 1000 毫秒（1秒）。

## 6.2 界面设计

前端用到的技术有Vue、Vuex、Element-ui、Axios、Sass、Quill。这里由于模块众多，并且模块之间的格式差不多，因此只需分析一个即可，下面是对菜品管理模块的界面设计进行的分析。

### 6.2.1 前端查询功能的实现

<el-form :model="queryParams" ref="queryRef" :inline="true" v-show="showSearch" label-width="68px">

<el-form-item label="菜品名称" prop="name">

<el-input

v-model="queryParams.name"

placeholder="请输入菜品名称"

clearable

@keyup.enter="handleQuery"

/>

</el-form-item>

<el-form-item label="售卖状态" prop="status">

<el-select v-model="queryParams.status" placeholder="请选择售卖状态" clearable>

<el-option

v-for="dict in dish\_status"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="Search" @click="handleQuery">搜索</el-button>

<el-button icon="Refresh" @click="resetQuery">重置</el-button>

</el-form-item>

</el-form>

* v-model：双向绑定表单输入与 queryParams 对象，方便获取用户输入的搜索条件。
* @keyup.enter：监听用户在输入框中按下回车键，调用 handleQuery() 方法触发查询。
* el-select：通过下拉框选择售卖状态，动态加载状态字典。

### 6.1.2 前端查询方法实现

/\*\* 查询菜品管理列表 \*/

function getList() {

loading.value = true; // 开始加载

listDish(queryParams.value).then(response => {

dishList.value = response.rows; // 更新菜品列表

total.value = response.total; // 更新总数

loading.value = false; // 结束加载

}).catch(error => {

loading.value = false; // 结束加载

console.error("查询失败:", error);

});

}

· loading：使用 loading 变量显示加载状态，改善用户体验。

· API 调用：通过 listDish(queryParams.value) 调用后端 API，传递查询参数，获取菜品数据。

· 错误处理：使用 catch 方法处理请求失败的情况，向控制台输出错误信息。

### 6.1.3前端删除功能的实现

/\*\* 删除按钮操作 \*/

function handleDelete(row) {

const \_ids = row.id || ids.value; // 获取要删除的菜品 ID

proxy.$modal.confirm('是否确认删除菜品管理编号为"' + \_ids + '"的数据项？').then(function() {

return delDish(\_ids); // 调用删除 API

}).then(() => {

getList(); // 重新加载列表

proxy.$modal.msgSuccess("删除成功"); // 提示用户删除成功

}).catch(() => {

// 处理取消操作或其他错误

});

}

· 确认删除：使用 proxy.$modal.confirm() 弹出确认框，确保用户确认删除操作。

· API 调用：调用 delDish() 方法，传入要删除的 ID，删除成功后更新列表。

### 6.1.4前端菜品口味管理功能

/\*\* 菜品口味关系添加按钮操作 \*/

function handleAddDishFlavor() {

let obj = { name: "", value: "" }; // 创建新的口味对象

dishFlavorList.value.push(obj); // 添加到口味列表

}

/\*\* 菜品口味关系删除按钮操作 \*/

function handleDeleteDishFlavor() {

if (checkedDishFlavor.value.length == 0) {

proxy.$modal.msgError("请先选择要删除的菜品口味关系数据");

} else {

const dishFlavors = dishFlavorList.value;

const checkedDishFlavors = checkedDishFlavor.value;

dishFlavorList.value = dishFlavors.filter(item => {

return checkedDishFlavors.indexOf(item.index) == -1; // 过滤掉已选中的口味

});

}

}

· 动态添加：允许用户动态添加新的口味关系，提升用户交互灵活性。

· 口味关系删除：先检查是否选择了口味，确保用户选择后再进行删除操作。

### 6.1.5前端口味选择功能

<el-select v-model="scope.row.name" placeholder="请选择口味名称" @change="changeFlavorName(scope.row)">

<el-option

v-for="dishFlavor in dishFlavourListSelect"

:key="dishFlavor.name"

:label="dishFlavor.name"

:value="dishFlavor.name"

/>

</el-select>

**动态更新**：根据用户选择的口味名称，动态更新可选口味列表。

### 6.1.5前端导出功能

/\*\* 导出按钮操作 \*/

function handleExport() {

proxy.download('merchant/dish/export', {

...queryParams.value // 传递查询参数

}, `dish\_${new Date().getTime()}.xlsx`) // 生成文件名

}

下载导出：通过调用 proxy.download() 方法，触发下载功能，将当前的菜品数据导出为 Excel 文件。

## 6.3 功能实现

### 6.3.1 菜品管理模块

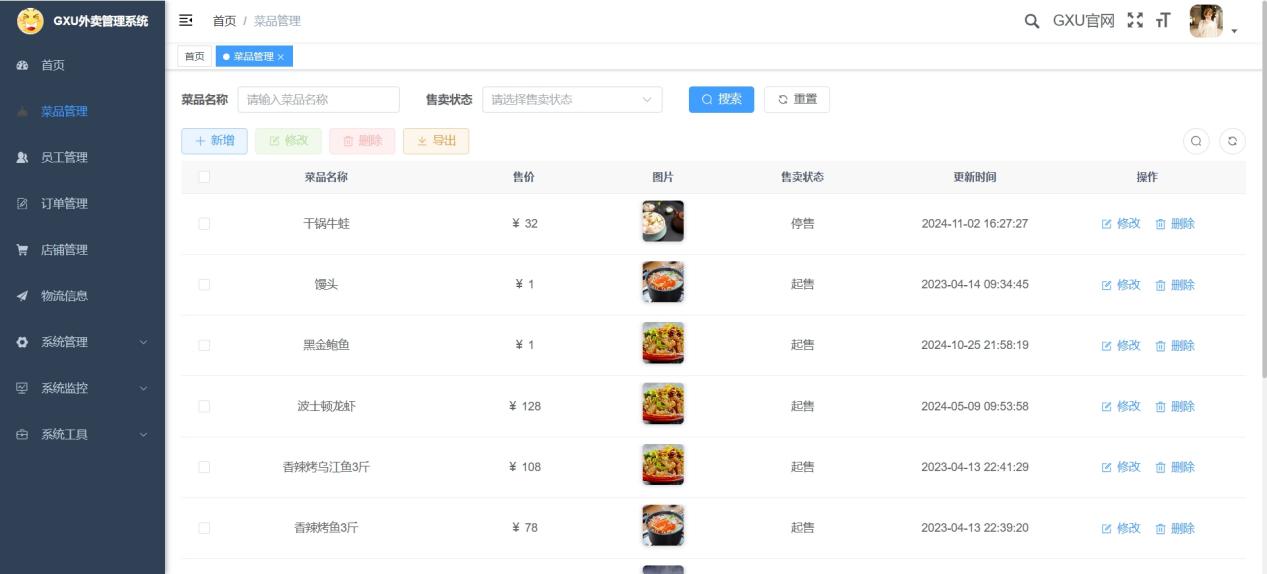
菜品管理的界面如下:  


图6-1 菜品管理演示图

该界面的Vue部分代码如下:

<el-form :model="queryParams" ref="queryRef" :inline="true" v-show="showSearch" label-width="68px">

      <el-form-item label="菜品名称" prop="name">

        <el-input

          v-model="queryParams.name"

          placeholder="请输入菜品名称"

          clearable

          @keyup.enter="handleQuery"

        />

      </el-form-item>

      <el-form-item label="售卖状态" prop="status">

        <el-select v-model="queryParams.status" placeholder="请选择售卖状态" clearable>

          <el-option

            v-for="dict in dish\_status"

            :key="dict.value"

            :label="dict.label"

            :value="dict.value"

          />

        </el-select>

      </el-form-item>

      <el-form-item>

        <el-button type="primary" icon="Search" @click="handleQuery">搜索</el-button>

        <el-button icon="Refresh" @click="resetQuery">重置</el-button>

      </el-form-item>

    </el-form>

**表单结构：**

· <el-form>：这是表单的容器，使用 :model 绑定到 queryParams 数据对象，表单的状态和输入值会自动同步。

ref="queryRef"：为表单提供了一个引用，可以通过这个引用在 Vue 组件内调用表单相关的方法。

:inline="true"：将表单设置为行内模式，控件会在一行中排列。

v-show="showSearch"：根据 showSearch 的值来控制表单的显示与隐藏。

**表单项：**

· <el-form-item>：每个表单项包含一个标签和一个输入控件。

第一个表单项用于输入菜品名称，包含一个 <el-input> 组件，支持清除输入和按回车键提交查询。

第二个表单项用于选择售卖状态，使用 <el-select> 组件，并通过 v-for 渲染状态选项，选项的内容来自 dish\_status 数组。

**按钮：**

· 表单的最后一部分包含两个按钮：

一个搜索按钮，类型为 primary，点击时会调用 handleQuery 方法。

一个重置按钮，点击时会调用 resetQuery 方法，通常用于清空表单中的输入。

· 交互逻辑：

· handleQuery 方法负责处理查询逻辑，可以在该方法中根据 queryParams 对象中的值进行搜索。

resetQuery 方法用于重置表单，通常会将 queryParams 中的属性重置为初始值。

### 6.3.2 员工管理模块

该模块的效果图如下:

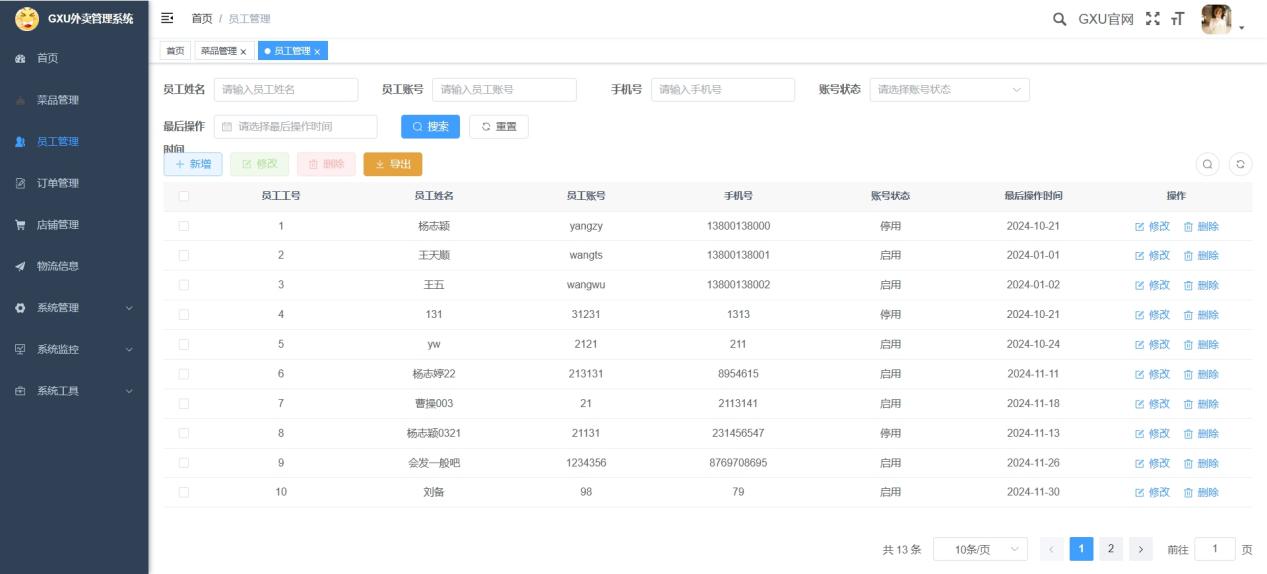


图6-2 员工管理演示图

由于前端界面已经在6.2界面设计说过，因此之后模块都介绍重点将在SQL语句的实现。下面对员工管理的EmployeeMapper.xml文件进行分析。

**Mapper 命名空间：**

· <mapper namespace="com.ruoyi.employee.mapper.employeeMapper">：定义了映射器的命名空间，方便在代码中引用。

结果映射：

· <resultMap> 定义了如何将数据库查询结果映射到 Java 对象的属性，确保查询的列名与对象的属性名正确对应。

SQL 语句定义：

· <sql id="selectemployeeVo">：定义了一个基本的查询语句，选择员工管理表中的相关字段，后续可以通过 include 引用。

· 查询操作：

· <select id="selectemployeeList">：实现员工列表的查询，支持根据多个条件（如姓名、账号、电话、状态和最后操作时间）进行筛选。使用 if 标签动态构建查询条件。

<select id="selectemployeeByEmployeeId">：根据员工 ID 查询特定员工的信息。

· 插入操作：

· <insert id="insertemployee">：用于插入新员工信息。使用 <trim> 标签动态处理要插入的字段和对应值，确保只有非空字段被插入。

· 更新操作：

· <update id="updateemployee">：更新员工信息，类似于插入操作，使用 trim 标签动态构建要更新的字段。

· 删除操作：

· <delete id="deleteemployeeByEmployeeId">：根据员工 ID 删除特定员工。

<delete id="deleteemployeeByEmployeeIds">：支持批量删除，通过传入一组员工 ID 来进行操作。

### 6.3.3 订单管理模块

效果图如下:

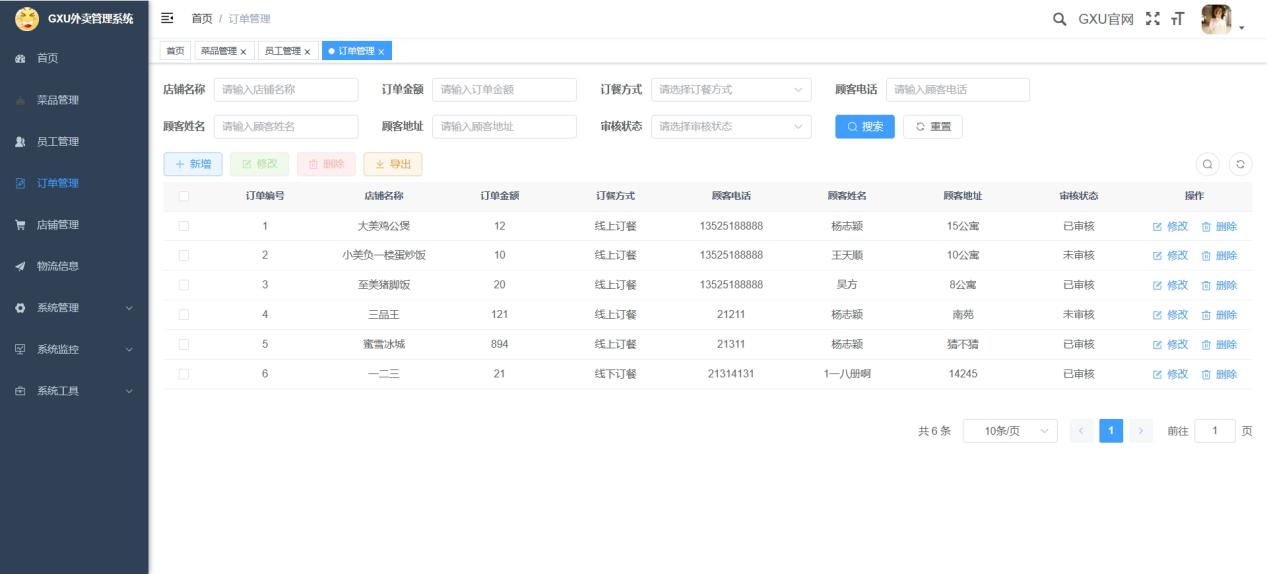


图6-3 订单管理演示图

**Mapper 命名空间**：

· <mapper namespace="com.ruoyi.order.mapper.OorderMapper">：定义了该映射器的命名空间，以便在 Java 代码中引用。

**结果映射**：

· <resultMap> 定义了如何将数据库查询结果映射到 Oorder Java 对象的属性，确保查询的列名与对象的属性名对应。

**SQL 语句定义：**

· <sql id="selectOorderVo">：定义了一个基本的查询语句，用于从 oorder 表中选择多个字段，以便后续通过 include 引用。

· 查询操作：

· <select id="selectOorderList">：实现订单列表的查询，支持根据多个条件（如店名、订单金额、支付方式、联系电话、联系人姓名、地址和检查状态）进行筛选，使用 if 标签动态构建查询条件。

<select id="selectOorderByOrderId">：根据订单 ID 查询特定订单的信息。

· 插入操作：

· <insert id="insertOorder">：用于插入新订单信息，使用 <trim> 标签动态处理要插入的字段和对应值，确保只有非空字段被插入。

· 更新操作：

· <update id="updateOorder">：更新订单信息，使用 trim 标签动态构建要更新的字段。

· 删除操作：

· <delete id="deleteOorderByOrderId">：根据订单 ID 删除特定订单。

<delete id="deleteOorderByOrderIds">：支持批量删除，通过传入一组订单 ID 来进行操作。

### 6.3.4 店铺管理模块

效果图如下：

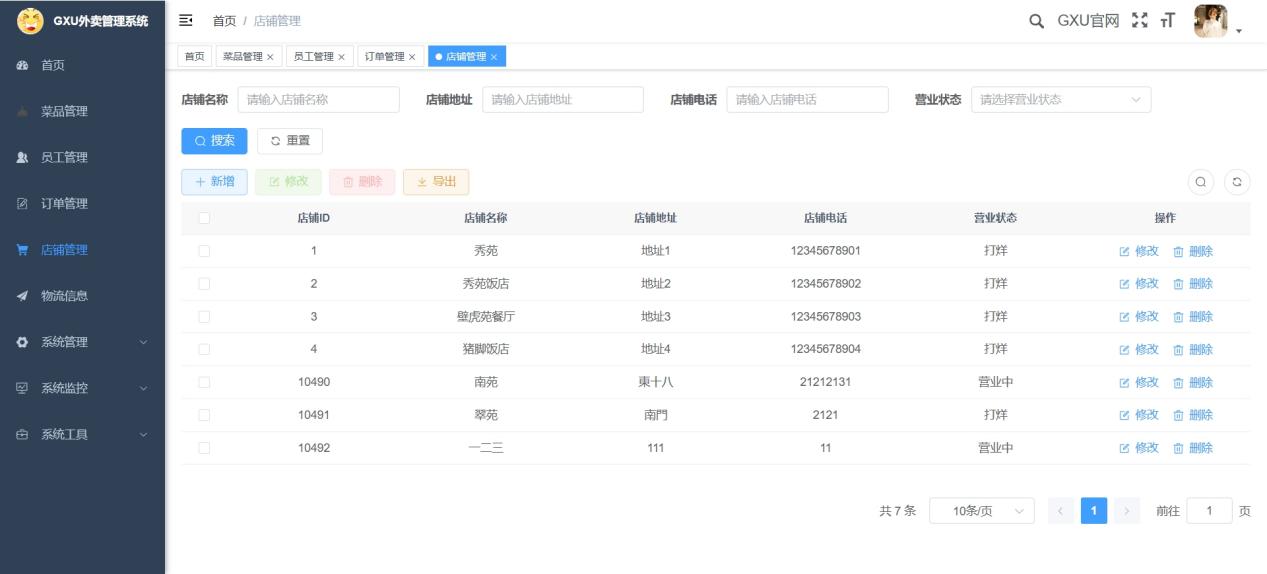


图6-4 店铺管理演示图

**Mapper 命名空间：**

· <mapper namespace="com.ruoyi.shop.mapper.ShopMapper">：定义了映射器的命名空间，便于在 Java 代码中引用。

**结果映射：**

· <resultMap type="Shop" id="ShopResult">：定义了如何将数据库查询结果映射到 Shop 对象的属性。

<resultMap id="ShopDishResult" type="Shop" extends="ShopResult">：扩展了 ShopResult，添加了一个名为 dishList 的集合，使用 selectDishList 查询该店铺的所有菜品。

<resultMap type="Dish" id="DishResult">：定义了如何将菜品的查询结果映射到 Dish 对象的属性。

**SQL 语句定义：**

· <sql id="selectShopVo">：定义了基本的查询语句，从 tb\_shop 表中选择字段，以便后续通过 include 引用。

· 查询操作：

· <select id="selectShopList">：用于查询店铺列表，支持根据多个条件（如店名、地址、电话和状态）进行筛选。

<select id="selectShopByShopId">：根据店铺 ID 查询特定店铺的信息，返回的结果包括店铺的菜品列表。

<select id="selectDishList">：根据菜品名称查询菜品列表。

· 插入操作：

· <insert id="insertShop">：用于插入新店铺信息，动态处理要插入的字段和对应值。

<insert id="batchDish">：用于批量插入菜品信息，支持通过列表传入多个菜品。

· 更新操作：

· <update id="updateShop">：更新店铺信息，动态构建要更新的字段。

· 删除操作：

· <delete id="deleteShopByShopId">：根据店铺 ID 删除特定店铺。

<delete id="deleteShopByShopIds">：支持批量删除，通过传入一组店铺 ID。

<delete id="deleteDishByNames">：根据菜品名称批量删除菜品。

<delete id="deleteDishByName">：根据菜品名称删除特定菜品。

### 6.3.5 物流信息模块

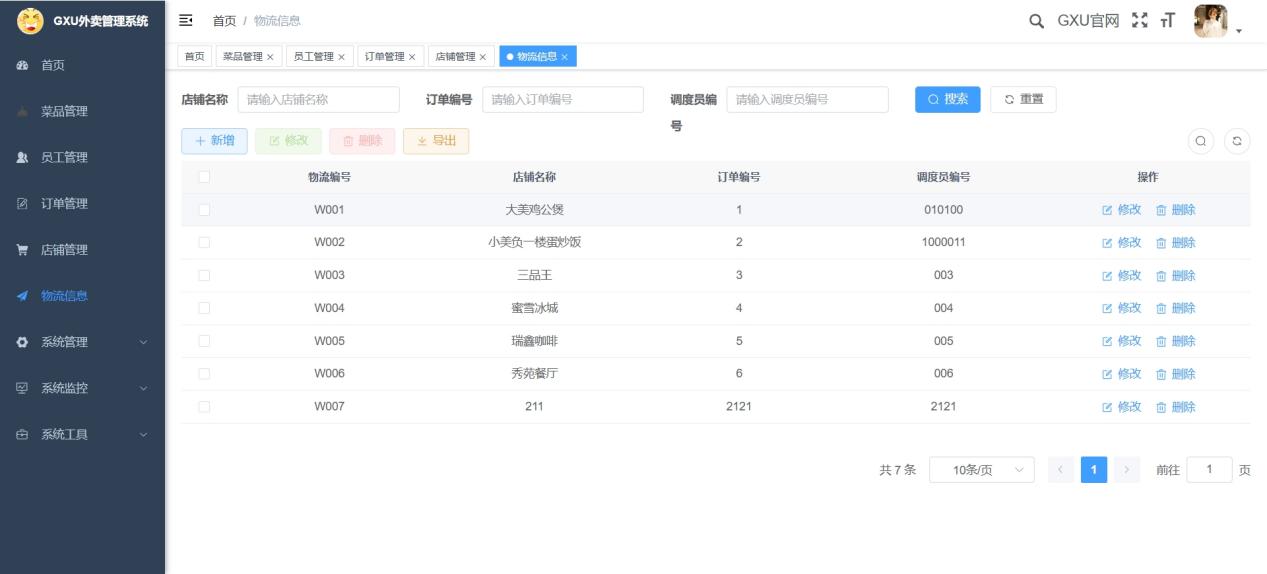
效果图如下:  


图6-5物流信息演示图

**结果映射：**

<resultMap type="Wuliu" id="WuliuResult">：定义了将查询结果映射到 Wuliu 对象的方式。

<resultMap id="WuliuDispatcherResult" type="Wuliu" extends="WuliuResult">：扩展了 WuliuResult，包含一个 dispatcherList 属性，该属性通过 selectDispatcherList 查询与该物流项相关的调度员。

<resultMap type="Dispatcher" id="DispatcherResult">：定义了将调度员的查询结果映射到 Dispatcher 对象的方式。

**SQL 语句定义**：

<sql id="selectWuliuVo">：定义了从 wuliu 表中选择字段的基本查询语句，以便在其他查询中引用。

查询操作：

<select id="selectWuliuList" parameterType="Wuliu" resultMap="WuliuResult">：用于查询物流列表，支持按多个条件（如店名、订单 ID、调度员 ID）进行筛选。

<select id="selectWuliuByWuliuId" parameterType="String" resultMap="WuliuDispatcherResult">：根据物流 ID 查询特定的物流项，包括其相关的调度员信息。

<select id="selectDispatcherList" resultMap="DispatcherResult">：根据调度员 ID 查询调度员的详细信息。

插入操作：

<insert id="insertWuliu" parameterType="Wuliu">：插入新的物流信息，动态处理要插入的字段和对应值。

<insert id="batchDispatcher">：用于批量插入调度员信息，支持通过列表传入多个调度员。

更新操作：

<update id="updateWuliu" parameterType="Wuliu">：更新物流信息，动态构建要更新的字段。

删除操作：

<delete id="deleteWuliuByWuliuId" parameterType="String">：根据物流 ID 删除特定的物流项。

<delete id="deleteWuliuByWuliuIds" parameterType="String">：支持批量删除，通过传入一组物流 ID。

<delete id="deleteDispatcherByDispatcherIds" parameterType="String">：根据调度员 ID 批量删除调度员。

<delete id="deleteDispatcherByDispatcherId" parameterType="String">：根据调度员 ID 删除特定的调度员。

# 数据库测试

## 7.1 加载测试数据

菜品相关信息的数据来自于百度，其余表格的数据是在以现实生活中实际情况为参考的前提下编造的。其中菜品表tb\_dish有25条数据(表示当前系统中总共有25种菜)。

员工管理employee\_management表有13条数据，订单管理oorder部分有5条数据(表示当前有5条数据)，店铺管理tb\_shop则有6条数据(目前系统中存在的店铺共有6家)，物流wuliu信息是有6条数据。一些数据的示例如下所示:  
INSERT INTO `tb\_dish` VALUES (30,'干锅牛蛙',38.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/56395e06-6d86-4d16-8d0e-18b2da085b8a.jpg','干锅牛蛙',1,'2023-04-13 23:14:12','2023-04-16 22:04:25'),(50,'馒头',1.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/dadaf543-b305-4139-9147-4e9f19f4c84c.jpg','优质面粉',1,'2022-06-10 09:34:28','2023-04-14 09:34:45'),(74,'黑金鲍鱼',68.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/833cf1ae-0835-4278-a374-00395cd4cbe9.jpg','新西兰黑金鲍鱼',1,'2023-02-16 09:48:28','2023-04-13 22:44:08'),(75,'波士顿龙虾',128.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/833cf1ae-0835-4278-a374-00395cd4cbe9.jpg','2斤重 波斯顿龙虾',1,'2023-02-16 09:50:06','2024-05-09 09:53:58'),(76,'香辣烤乌江鱼3斤',108.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/833cf1ae-0835-4278-a374-00395cd4cbe9.jpg','香辣烤乌江鱼3斤',1,'2023-02-16 09:52:30','2023-04-13 22:41:29'),(77,'香辣烤鱼3斤',78.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/dadaf543-b305-4139-9147-4e9f19f4c84c.jpg','香辣烤鱼3斤 草鱼',1,'2023-02-17 15:27:02','2023-04-13 22:39:20'),(80,'宽粉',8.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/c04d2598-0b96-44d7-8ea9-72792b9be66a.jpg','宽粉',0,'2023-04-13 22:48:57','2023-04-15 13:45:03'),(81,'青笋',10.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/3685a52b-46ca-4e2a-8fd2-d90205fa9405.jpg','青笋',1,'2023-04-13 22:49:21','2023-04-13 22:49:21'),(82,'鲜豆皮',8.00,'https://yjy-rjwm-oss.oss-cn-hangzhou.aliyuncs.com/c04d2598-0b96-44d7-8ea9-72792b9be66a.jpg','鲜豆皮',1,'2023-04-13 22:49:52','2023-04-13 22:49:52')

INSERT INTO `oorder` VALUES (1, '大美鸡公煲', 12, 0, '13525188888', '老三', '15公寓', 0, '2022-12-16 14:35:17');

INSERT INTO `oorder` VALUES (2, '小美负一楼蛋炒饭', 10, 1, '13525188888', '吴方', '10公寓', 1, '2022-12-16 14:35:26');

INSERT INTO `oorder` VALUES (3, '至美猪脚饭', 20, 0, '13525188888', '吴方', '8公寓', 0, '2022-12-16 14:35:35');

INSERT INTO `employee\_management` VALUES

(1, '杨志颖', 'yangzy', '13800138000', 0, '2024-01-01 10:00:00'),

(2, '王天顺', 'wangts', '13800138001', 0, '2024-01-01 10:30:00'),

(3, '王五', 'wangwu', '13800138002', 0, '2024-01-02 09:00:00');

INSERT INTO `tb\_shop` (`shop\_id`, `shop\_name`, `shop\_address`, `shop\_phone`, `shop\_status`) VALUES

(1, '鸡公煲店', '地址1', '12345678901', 1),

(2, '秀苑饭店', '地址2', '12345678902', 1),

(3, '壁虎苑餐厅', '地址3', '12345678903', 1),

(4, '猪脚饭店', '地址4', '12345678904', 1);

LOCK TABLES `wuliu` WRITE;

INSERT INTO `wuliu` VALUES ('W001', '大美鸡公煲', 1, '010100');

INSERT INTO `wuliu` VALUES ('W002', '小美负一楼蛋炒饭', 2, '1000011');

UNLOCK TABLES;

LOCK TABLES `dispatcher` WRITE;

INSERT INTO `dispatcher` VALUES ('010100', '摇摆羊', '13365789765');

INSERT INTO `dispatcher` VALUES ('010101', '小亮', '15878977898');

INSERT INTO `dispatcher` VALUES ('1000011', '老八', '13526777887');

INSERT INTO `dispatcher` VALUES ('10111', '赵三金', '15965578765');

UNLOCK TABLES;

## 7.2 功能测试

### 7.2.1 菜品模块功能测试

菜品管理模块的功能测试结果如下表7-1所示以及图7-1、图7-2所示。

表7-1 菜品管理模块测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 详细操作 | 预期效果 | 实际效果 |
| 菜品搜索 | 在菜品名称框输入查询的菜品名称，输入后点击最右侧的搜索 | 搜索成功 | 和预期效果相符合 |
| 通过售卖状态进行查询 | 选择相关的售卖状态，点击查询按钮 | 查询成功 | 和预期效果相符合 |
| 导出 | 在点击相关要导出的菜品之后，点击“导出”按钮 | 导出表格成功，如图7-2所示。 | 和预期效果相符 |
| 删除功能测试 | 在选择要删除的菜品之后，点击删除按钮 | 删除成功，在菜品栏和数据库中没有该菜品。 | 和预期效果相符合 |
| 修改功能测试 | 在勾选好要修改的数据之后，点击修改按钮 | 修改成功，可修改的信息包括菜品名称、售价图片、描述信息、售卖状态以及相关的口味信息。 | 和预期效果相符合 |
| 新增功能模块测试 | 点击新增按钮 | 填入相关的菜品信息，并且会提示相关不能为空的字段信息。 | 和预期效果相符合 |
| 重置按钮测试 | 当我们搜索完菜品，点击重置按钮，即可 | 会回到显示所有菜品的页面， | 和预期效果相符合 |

根据表7-1对菜品管理模块的测试结果可知，菜品管理模块的基本功能都已满足需求。

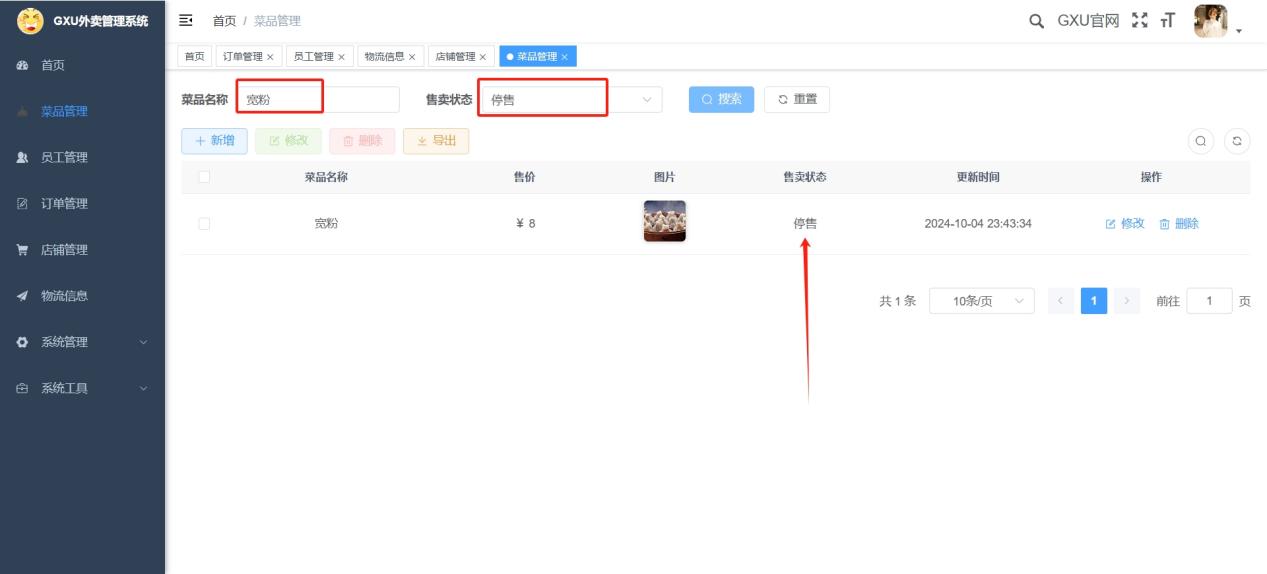


图7-1 菜品管理搜索模块功能演示图

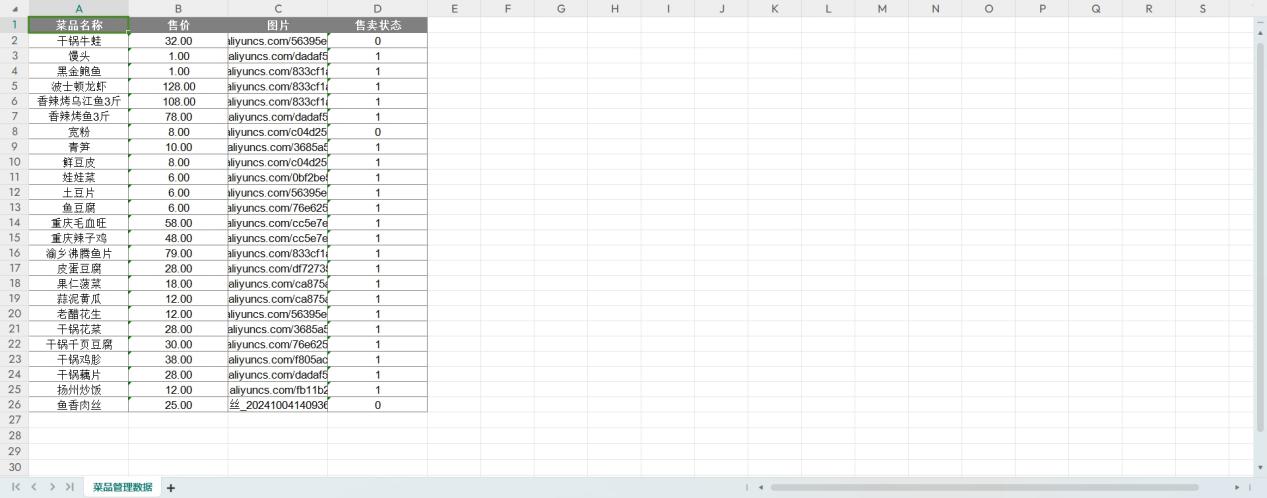


图7-2 菜品管理导出模块功能演示图

、

### 7.2.2 员工管理模块功能测试

员工管理模块的功能测试结果如下表7-2所示，部分效果图如图7-3和图7-4所示。

表7-2 员工管理模块测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 详细操作 | 预期效果 | 实际效果 |
| 根据员工姓名查询 | 输入已经存在的员工姓名，点击右侧的搜索按钮 | 显示该员工的相关信息 | 和预期效果相符合 |
| 根据员工账号进行查询 | 在员工账号输入框输入员工账号，输入完之后点击搜索按钮 | 能够显示相关数据，若无则显示暂无数据 | 和预期效果相符合 |
| 根据员工的手机号进行查询操作 | 在手机号框内输入想要查询员工的手机号，输入之后点击搜索 | 根据手机号搜索成功，成功显示员工的相关信 | 和预期效果相符 |
| 根据账号状态进行相关查询 | 在售卖状态一栏选择相对应的账号状态进行搜索 | 搜索成功 | 和预期效果相符合 |
| 根据最后操作时间进行搜索 | 点击时间框，选择好相应的时间之后，进行搜索 | 在当前界面显示该时间内操作过的员工信息 | 和预期效果相符合 |
| 新增员工信息 | 点击新增按钮 | 输入员工的相关信息，点击确定 | 和预期效果相符合 |
| 导出相关的员工信息 | 选中要导出的员工信息之后，点击导出按钮 | 可以看到相关员工信息被导出 | 和预期效果相符合 |
| 修改员工信息 | 在要进行修改的员工信息的修改按钮处点击 | 可以输入新的信息 | 和预期效果相符合 |
| 删除员工信息 | 对应删除的员工信息处点击删除 | 员工信息成功删除 | 和预期效果相符合 |

根据表7-2对员工管理模块的测试结果可知，员工管理模块的基本需求都已实现。

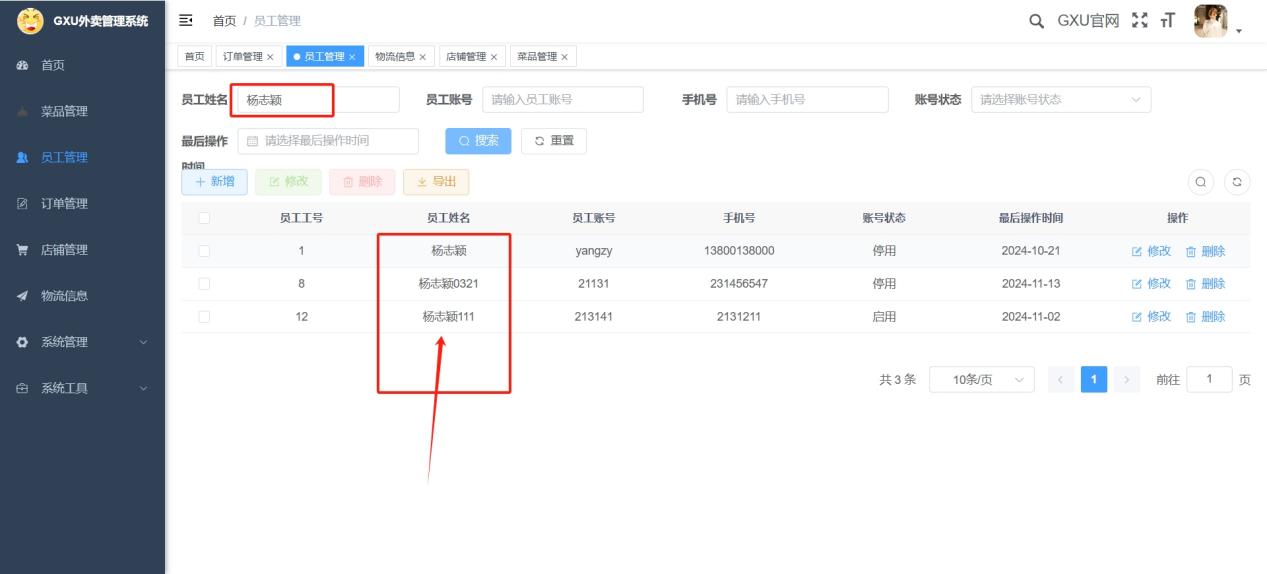


图7-3 员工管理根据姓名进行模糊查询功能演示图

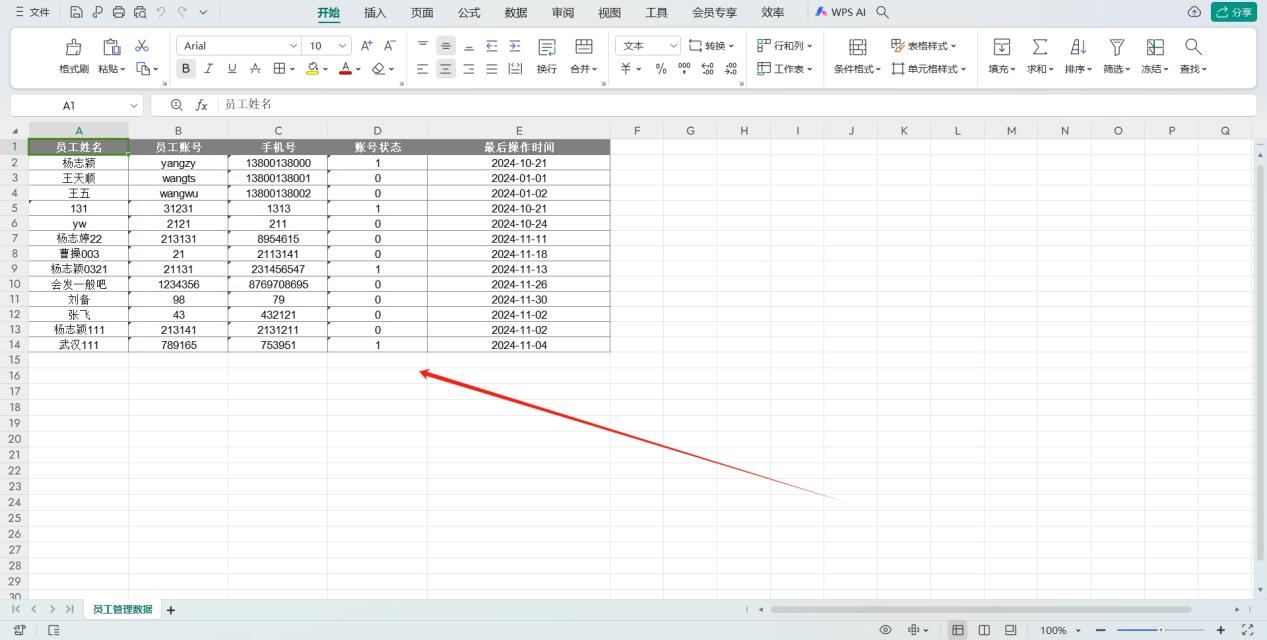
****

图7-4 员工管理导出功能演示图

### 7.2.3 订单管理模块功能测试

订单管理模块的功能测试结果如下表7-3所示。部分效果图如图7-5和7-6所示。

表7-3 订单管理模块测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 详细操作 | 预期效果 | 实际效果 |
| 根据店铺名称进行搜索 | 输入要查询的店铺名称 | 点击搜索后，可以看到对应店铺的相关信息 | 和预期效果相符合 |
| 根据订单金额进行查询 | 输入要查询的订单的金额 | 点击搜索按钮 | 和预期效果相符合 |
| 通过订餐方式查看对应订单 | 选择好订餐方式之后点击搜索 | 能够正确显示相关信息 | 和预期效果相符 |
| 由顾客电话进行相关搜索 | 输入顾客的电话 | 显示出对应的订单信息 | 和预期效果相符 |
| 通过顾客名字来进行相关搜索 | 输入顾客的姓名 | 显示出被搜索的顾客以及订单的相关信息 | 和预期效果相符合 |
| 通过顾客地址来进行相关搜索 | 输入顾客的地址 | 显示出被搜索的顾客以及订单的相关信息 | 和预期效果相符合 |
| 通过审核状态进行搜索 | 选择相关的审核状态 | 正确显示相关订单信息 | 和预期效果相符合 |
| 新增订单信息 | 点击新增按钮 | 输入相关的订单信息 | 和预期效果相符合 |
| 导出 | 选中要导出的订单信息 | 能够正确显示相应的订单信息 | 和预期效果相符合 |

根据表7-3对订单管理模块的测试结果可知，订单管理模块的基本需求都已实现。

****

图7-5 订单管理新增功能演示图

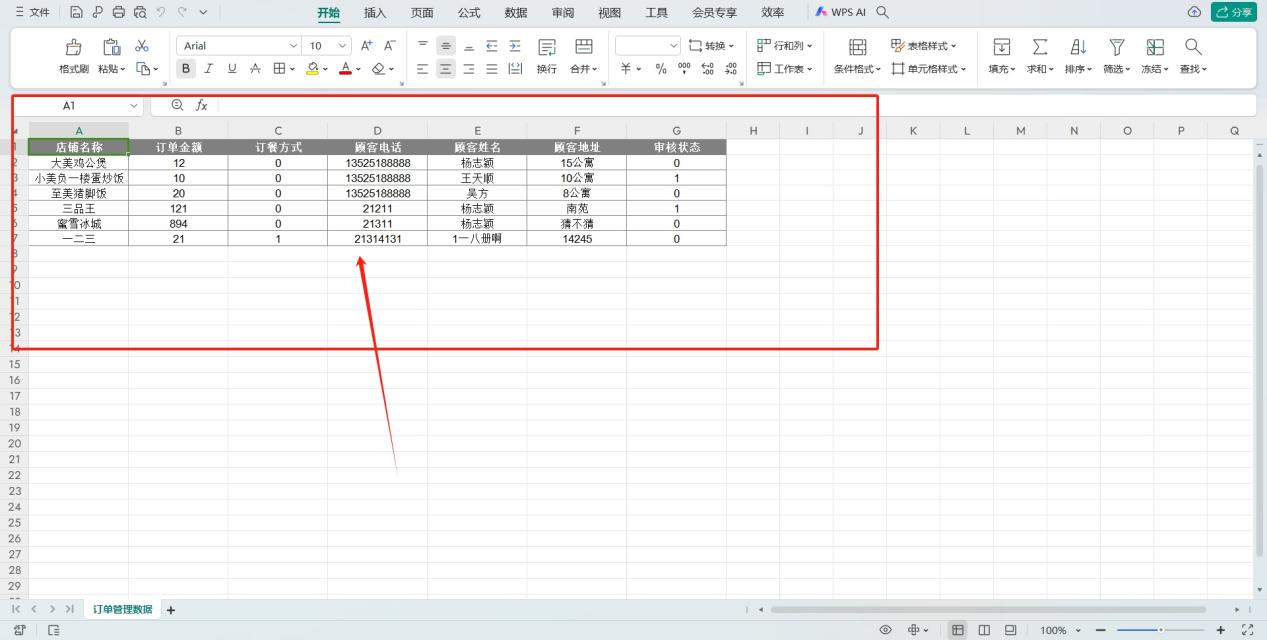
****

图7-6 订单管理导出功能演示图

### 7.2.4 店铺管理模块功能测试

店铺管理模块的功能测试结果如下表7-4所示。部分效果图如图7-7和7-8所示。

表7-4 店铺管理模块测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 详细操作 | 预期效果 | 实际效果 |
| 根据店铺名称进行搜索 | 输入要查询的店铺名称 | 点击搜索后，可以看到对应店铺的相关信息 | 和预期效果相符合 |
| 根据店铺地址进行查询 | 输入要查询的店铺地址 | 点击搜索按钮 | 和预期效果相符合 |
| 通过店铺电话查看相关信息 | 输入好之后点击查询 | 能够正确显示相关信息 | 和预期效果相符 |
| 由营业状态进行查询 | 输入对应店铺的营业状态 | 显示出对应的店铺信息 | 和预期效果相符 |
| 新增 | 点击新增按钮 | 能够成功添加相关信息 | 和预期效果相符合 |
| 删除 | 点击删除按钮 | 相关信息被删除 | 和预期效果相符合 |
| 导出 | 点击导出按钮 | 能够正确导出 | 和预期效果相符合 |

根据表7-4对店铺管理模块的测试结果可知，店铺管理模块的基本需求都已实现。

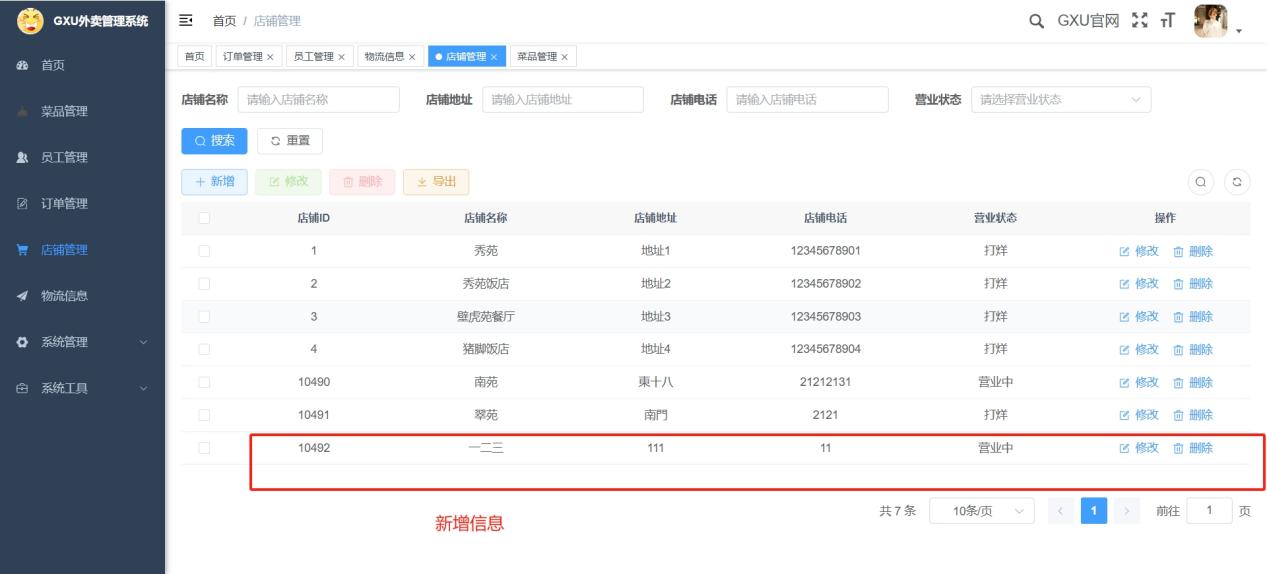
****

图7-7 店铺管理新增功能演示图

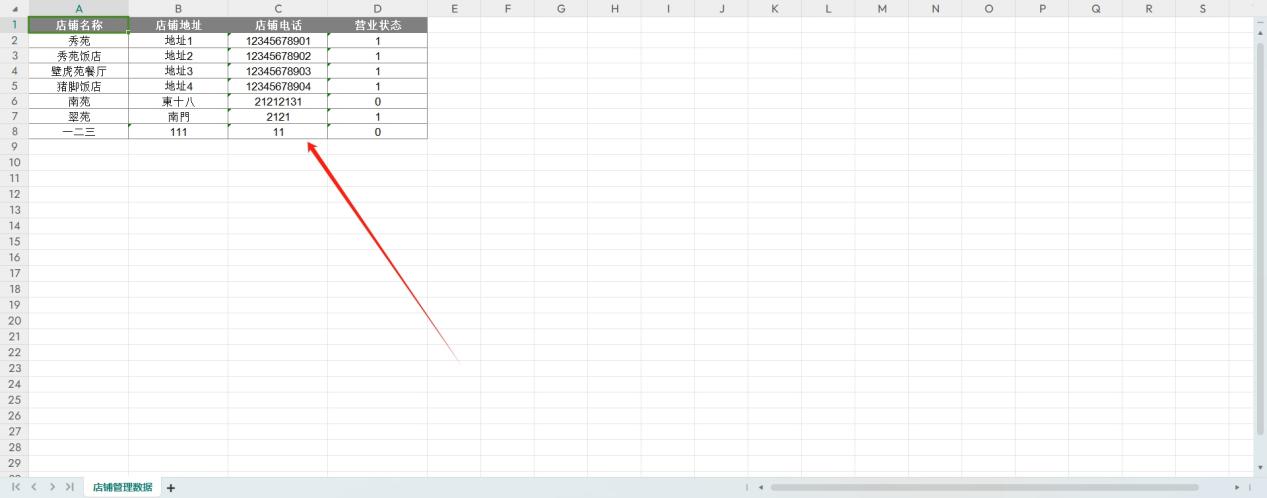


图7-8 店铺管理导出功能演示图

### 7.2.5 物流管理模块功能测试

物流管理模块的功能测试结果如下表7-5所示。部分效果图如图7-9、7-10所示。

表7-5 物流管理模块测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 详细操作 | 预期效果 | 实际效果 |
| 修改物流信息 | 点击修改，可修改店铺名称、订单编号、调度员编号、调度员的相关信息 | 点击确定后再次点修改，发现相关信息已修改 | 和预期效果相符合 |
| 删除物流信息 | 点击界面的删除按钮 | 点击重置按钮，可以发现对应数据已被删除 | 和预期效果相符合 |
| 通过店铺名称来对物流信息进行查询 | 输入店铺名称之后点击搜索 | 能够正确显示相关信息 | 和预期效果相符 |
| 通过订单编号来对物流信息进行查询 | 输入订单编号 | 显示出对应信息 | 和预期效果相符 |
| 通过调度员编号来对物流信息进行查询 | 输入之后点击搜索 | 显示出相关信息 | 和预期效果相符合 |

根据表7-5对物流信息模块的测试结果可知，物流管理模块的基本需求都已实现。

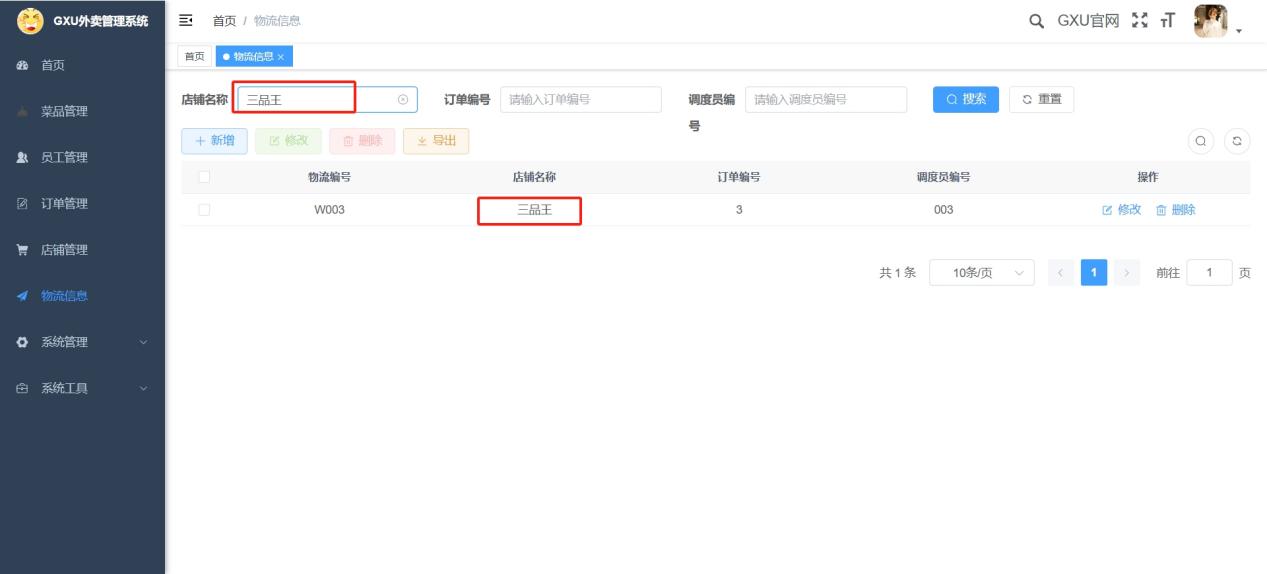


图7-9物流管理搜索功能演示图

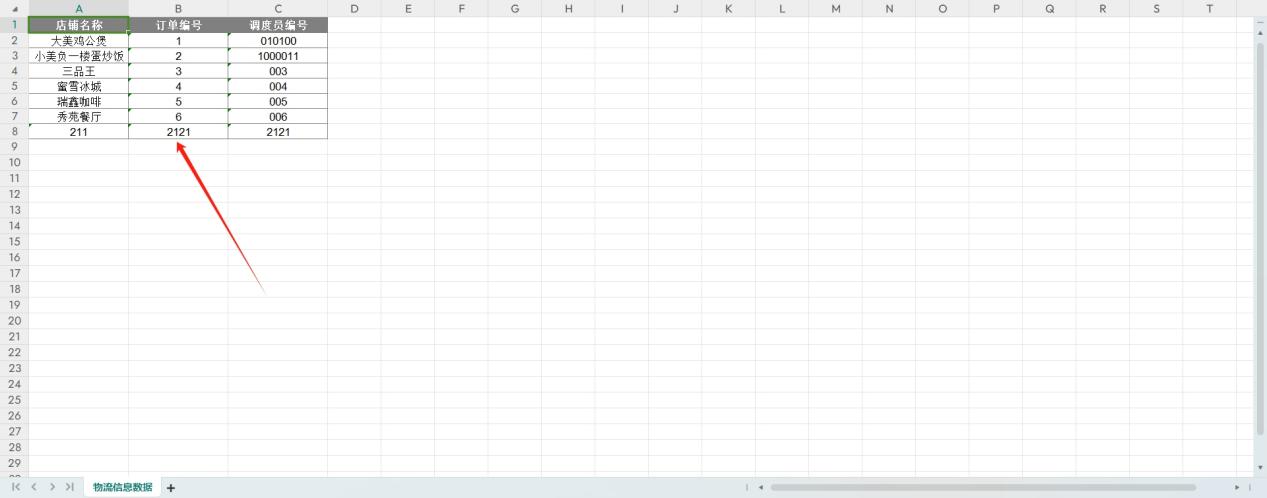


图7-10物流管理导出功能演示图

## 7.3 性能测试

### 7.3.1 内存消耗、CPU消耗

利用主流系统性能测试工具（例如YellowLabTools）测试数据库的访问性能。

由于YellowLabTools工具我试了几次均不行，如下图7-11于是我使用的性能监测工具是java自带的jconsole.exe，来查看程序运行时的峰值内存消耗。

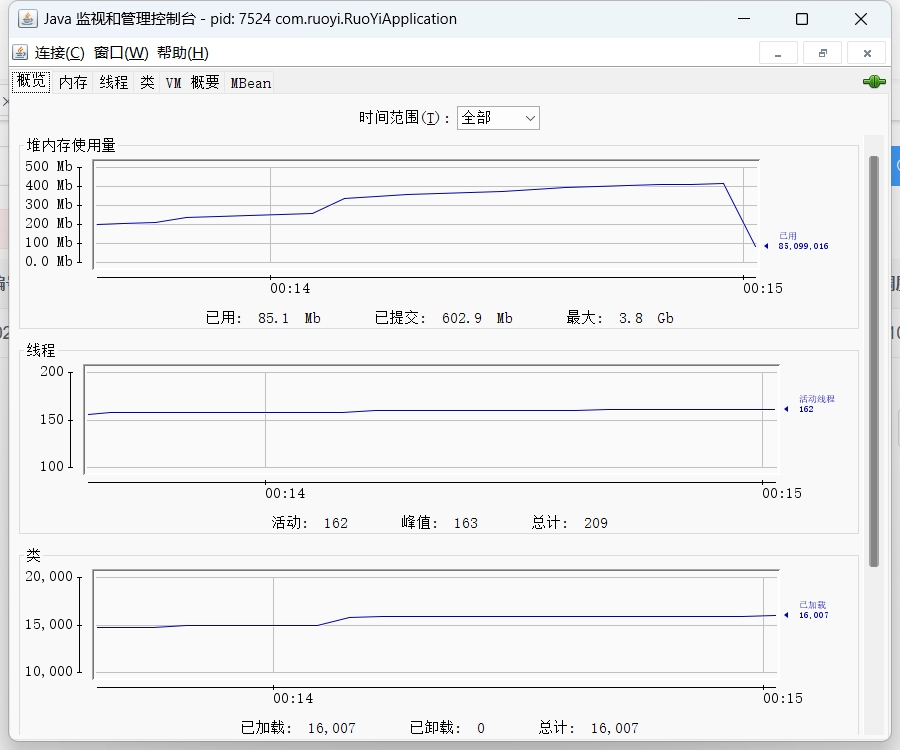


图7-12内存使用量、线程、类的使用情况

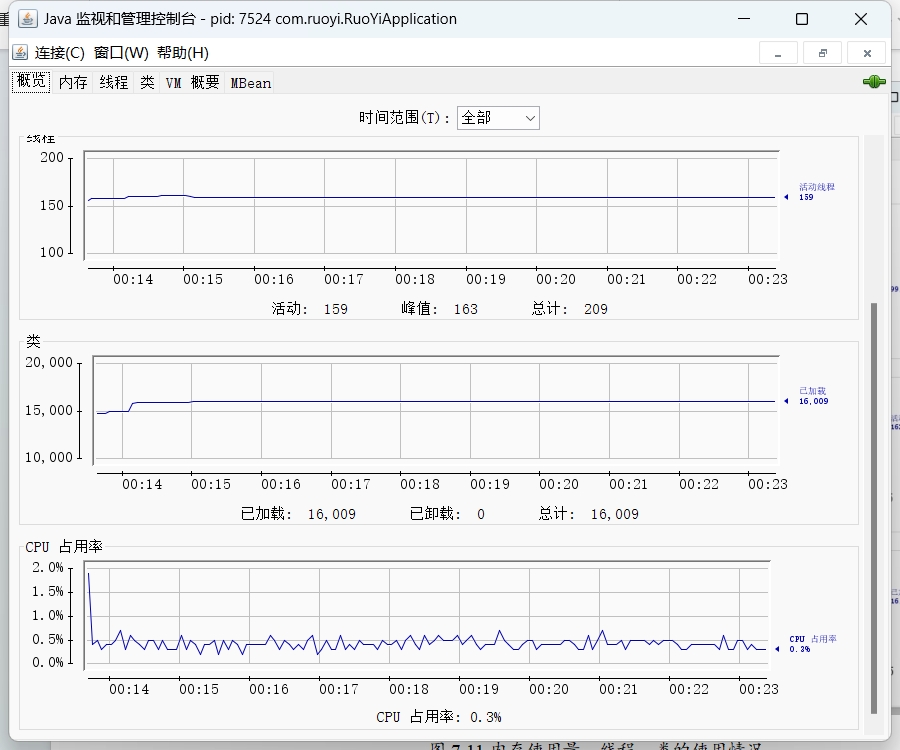


图7-13 CPU占用率使用情况

由于系统功能较多，这里没有一一测试，选择了较为主要的功能进行测试。这里我主要测试的是各个模块的导出功能以及搜索功能以及新增功能，可以看到在CPU的使用率界面，刚开始最高，这里是因为我首先测试的是菜品管理模块，此模块涉及内容较多，同时数据也是最多的一个模块，因此此时是CPU的峰值，占比2%，同时虽然这个模块数据量较大，但是搜索时间没有受到影响。同时在菜品管理模块都堆占用内存也可以看到最高，同时之后的其他模块由于没有菜品模块复杂，因此CPU消耗较少，同时堆占用内存也较少。同时由于若依框架已经自带了性能监测工具，如下图7-14。可以在图7-12、7-13中看到CPU的峰值利用率才是2%，堆内存最高300MB，本系统在CPU消耗、内存消耗方面性能良好。



图7-14 缓存监控使用情况

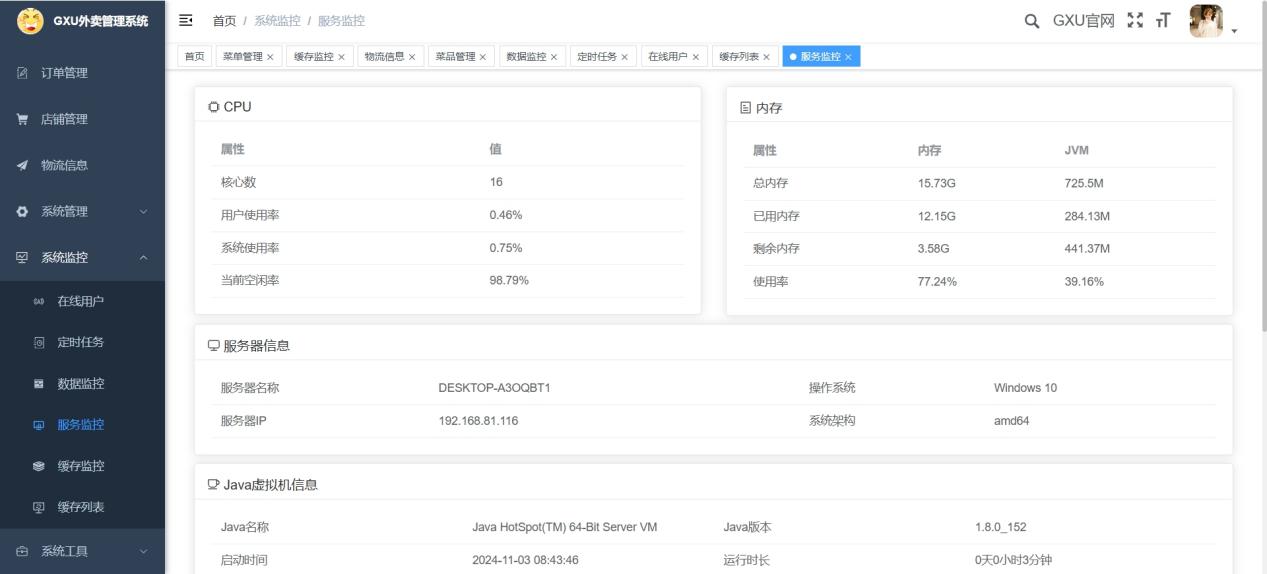


图7-15 服务监控使用情况

### 7.3.2 时间消耗

为评估系统各功能的时间消耗，我们采用JDK自带的时间测试方法，对8个主要功能的运行时间进行了测试。用于测试的代码如下：

long StartTime = System.currentTimeMillis(); // 获取当前时间// 执行需要测试的功能long EndTime = System.currentTimeMillis();

System.out.println("程序运行时间:" + (EndTime - StartTime) + "ms");

图7-2 展示了8个主要功能的时间测试性能平均值，单位为毫秒（ms）。在测试过程中，以订单生成功能为例，由于用户的输入可能不同，每次的查询和处理结果也会有所不同，因此每个功能进行了30次时间消耗测试，最终取平均值，结果如下：

* 注册程序 — 50ms
* 登录程序 — 48ms
* 菜品搜索程序 — 65ms
* 订单生成程序 — 20ms
* 订单修改程序 — 30ms
* 员工信息修改程序 — 27ms
* 店铺查询程序 — 22ms
* 物流信息处理程序 — 24ms

根据以上测试结果，在测试的程序中，时间消耗的平均值最高为65ms，说明本系统在时间消耗方面也表现良好。

# 收获和体会

通过课程设计，我们不仅复习和研究了老师讲授的SQL数据库知识（如触发器、索引、存储过程等），还锻炼了编程能力，深入实践了从需求分析到系统实现的全过程。整个项目涉及数据库的设计与优化、功能开发和前后端整合，使我在多个方面得到了提高。

在数据库设计和实现过程中，熟悉了MySQL数据库的使用，并通过索引优化查询效率，利用视图扩展了数据的展示能力，通过触发器实现表间联动操作，确保了数据的完整性和一致性。同时，我根据实际需求设计了多种存储过程和函数，简化了复杂业务逻辑的处理。通过这些措施，我设计的数据库不仅能满足系统的基本需求，还具备较好的性能表现。

此外，在实现外卖管理系统的过程中，我们使用了Spring Boot作为后端开发框架，并完成了数据库和前端页面的连接。Spring Boot的使用让我们对Java Web开发的整体流程有了清晰的理解和熟练的掌握。学会了如何通过编程实现页面功能，如订单管理、店铺管理、物流信息的展示与更新等功能，提升了自己的全栈开发技能。

通过这次课程设计，我们不仅加深了对数据库理论的理解，还在动手实践中体会到了网站设计和开发的实际流程。从数据库的需求分析、架构设计到实现和优化，对整个网站开发过程有了更深入的认识。这次项目让我们对未来的开发工作充满信心，同时也激发了我们在数据库优化、系统开发等方面继续深入学习和探索的兴趣。

组内人员分工和完成情况详见下表：

|  |  |  |
| --- | --- | --- |
| **姓名** | **所负责的任务** | **完成情况** |
|  | 完成概述和需求分析，概念结构设计的菜品管理、员工管理模块.  总体的ER图设计。 | 完成良好 |
| 数据库测试、物理设计与实施。 | 完成良好 |
| 应用系统的设计与实施。 | 完成良好 |
|  | 根据题目要求完成需求分析，得出关系模式，优化关系模式。 | 完成良好 |
| 完成概念结构设计的店铺管理、订单管理、物流管理模块。 | 完成良好 |
| 设计物理结构，索引、视图、触发器等。 | 完成良好 |

# 参考文献

[1]王珊，萨师煊.数据库系统概论[M].北京：高等教育出版社，2014：206-267，293-300.

[2]姜承尧. MySQL技术内幕：InnoDB存储引擎[M].北京：机械工业出版社，2011：17-63.

[3]赵文婷.图书管理信息系统的设计与实现 [J]. 中国信息化, 2020(10):87-88.

[4]Elmasri, R., & Navathe, S. B. Fundamentals of Database Systems. 7th ed. Boston: Pearson, 2016.

[5]Silberschatz, A., Korth, H. F., & Sudarshan, S. Database System Concepts. 6th ed. New York: McGraw-Hill, 2011.

[6]廖明辉, 王小刚. 《数据库技术与应用》. 上海: 上海交通大学出版社, 2020.

[7]Date, C. J. An Introduction to Database Systems. 8th ed. Boston: Addison-Wesley, 2003.

[8]陈明. 《SQL优化与性能调优实战指南》. 北京: 清华大学出版社, 2020.

[9]康睿. 《数据库系统优化设计及应用实践》. 广州: 华南理工大学出版社, 2021.

[10]Connolly, T., & Begg, C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Boston: Pearson, 2014.

[11]McLaughlin, M. Oracle Database 12c PL/SQL Programming. 2nd ed. New York: McGraw-Hill Education, 2014.

[12]黄红, 王峰. 《数据库查询优化与存储过程设计》. 武汉: 武汉大学出版社, 2019.