|  |  |  |
| --- | --- | --- |
| 南农大  **数据库课程设计报告**  XH2 | | |
|  | 题 目: | 418物流订单管理系统 |
|  | 组 长: | 董自经 学号: 19222126 |
|  | 组 员 | 黄子琰 学号: 19222125 |
|  |  | 南昊冉 学号: 19222123 |
|  |  | 曾 瑞 学号: 19222128 |
|  | 学 院: | 人工智能学院 |
|  | 专 业: | 计算机科学与技术 |
|  | 班 级: | 计科221 |
|  | 指导教师: | 黄芬 职称: 副教授 |
| 2024年 4 月 20 日 | | |

目录

[**1 开发技术、开发工具及技术路线** 2](#_Toc164788362)

[1.1 开发技术及开发工具 2](#_Toc164788363)

[**1.1.1 开发技术** 2](#_Toc164788364)

[**1.2.2 开发工具** 3](#_Toc164788365)

[1.2 技术路线 4](#_Toc164788366)

[**2 功能分析与设计** 4](#_Toc164788367)

[2.1 需求分析 5](#_Toc164788368)

[**2.1.1 任务背景** 5](#_Toc164788369)

[**2.1.2 管理工作需求** 5](#_Toc164788370)

[2.2 数据字典 5](#_Toc164788371)

[**2.2.1 数据项和数据结构** 5](#_Toc164788372)

[**2.2.2 数据流** 6](#_Toc164788373)

[**2.2.3 系统任务分析** 6](#_Toc164788374)

[2.3 数据库概念结构设计 6](#_Toc164788375)

[**2.3.1各模块的ER图** 6](#_Toc164788376)

[**2.3.2 总体ER图** 8](#_Toc164788377)

[2.4 数据库逻辑结构设计 9](#_Toc164788378)

[**2.4.1 E-R图向关系模型的转换** 9](#_Toc164788379)

[**2.4.2 利用数据依赖的理论对关系模式进行分析** 11](#_Toc164788380)

[**2.4.3最终关系模式设计** 11](#_Toc164788381)

[2.5 数据库物理结构设计 13](#_Toc164788382)

[**2.5.1确定存储结构** 13](#_Toc164788383)

[**2.5.2评价物理结构** 13](#_Toc164788384)

[2.6 模块 13](#_Toc164788385)

[3.**实现** 14](#_Toc164788386)

[3.1 前端界面实现 14](#_Toc164788387)

[**3.1.1 界面设计** 14](#_Toc164788388)

[3.2 后端代码实现 15](#_Toc164788389)

[**3.2.1 VScode与SQL Server的连接和交互** 15](#_Toc164788390)

[**3.2.2 部分功能代码** 15](#_Toc164788391)

[**3.2.3 SQL Server建立数据库** 18](#_Toc164788392)

[**4 系统测试与应用** 25](#_Toc164788393)

[4.1 系统测试运行环境 25](#_Toc164788394)

[4.2 数据库测试与运行 25](#_Toc164788395)

[**5.总结与展望** 26](#_Toc164788396)

[**5.1总结** 26](#_Toc164788397)

[5.1.1用户中心： 26](#_Toc164788398)

[5.1.2购物中心 26](#_Toc164788399)

[5.1.3查询中心 26](#_Toc164788400)

[5.1.4创新点 27](#_Toc164788401)

[5.2不足 27](#_Toc164788402)

[5.3展望 27](#_Toc164788403)

418物流订单管理系统

**1 开发技术、开发工具及技术路线**

1.1 开发技术及开发工具

1.1.1 开发技术

本次所设计的物流订单管理系统采用Vue前端框架、ngrok跨域链接以及采用FastAPI来创建API；所使用的编程语言为Python、JavaScript、HTML以及CSS；数据库采用的是Microsoft SQL Server。

1. Python

Python是一种高级、通用、解释型编程语言，由Guido van Rossum于1991年创建。它具有简单易学、代码清晰、功能强大等特点，因此被广泛应用于各种领域，包括Web开发、数据分析、人工智能等。Python拥有丰富的标准库和第三方库，使得开发者可以轻松地实现各种功能。同时，Python还支持多种编程范式，包括面向对象、函数式编程等，使得开发者可以根据需要选择合适的编程风格。

1. Javascript

JavaScript是一种高级、动态、解释型编程语言，最初由Netscape公司开发，后被ECMA（欧洲计算机制造商协会）标准化。JavaScript通常用于在网页上实现交互性和动态效果，如表单验证、动画效果、DOM操作等。它也被广泛应用于前端开发、后端开发、移动应用开发等领域。JavaScript具有灵活的语法和强大的功能，支持面向对象、函数式编程等多种编程范式。不仅如此，得益于JavaScript丰富的第三方库和框架，诸如React、Angular、Vue等，让Javascript能够应用于复杂应用程序的开发。

1. HTML

HTML（HyperText Markup Language）是一种标记语言，用于创建网页结构和内容。HTML由一系列标签组成，每个标签都用来定义不同的元素或内容，如标题、段落、图像等。通过使用HTML标记语言，开发者可以将文本、图像、链接等内容组织起来，形成一个完整的网页。HTML是Web开发中最基础的技术之一，与CSS（层叠样式表）和JavaScript一起构成了Web开发的基础。HTML引入了许多新的元素和API，有利于创建丰富多样的网页和Web应用。

1. CSS

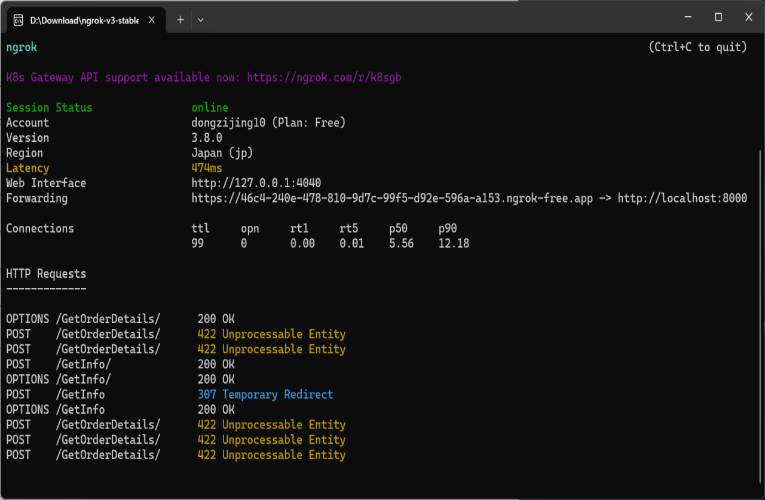
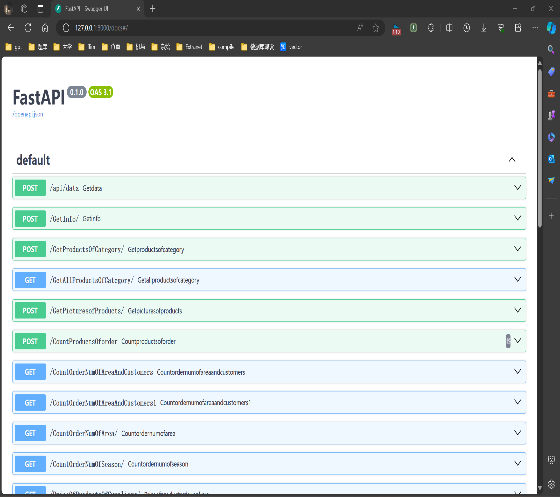
CSS（Cascading Style Sheets）是一种样式表语言，用于控制网页的外观和布局。通过使用CSS，开发者可以为网页中的各种元素指定样式，如颜色、字体、大小、间距、边框等，是实现网页美观和响应式设计的关键技术。CSS与HTML结合使用，可以实现更加丰富和美观的网页设计。CSS具有层叠性（Cascading）的特点，即可以通过不同的样式规则来控制同一个元素，使得开发者可以更灵活地管理和调整网页的样式。同时，CSS还支持响应式设计，使得网页可以适应不同设备和屏幕尺寸，提供更好的用户体验。随着Web技术的发展，CSS也不断更新，引入了新的特性和功能，如Flexbox、Grid布局等。

1. SQL Server

SQL Server是由微软公司开发的关系型数据库管理系统（RDBMS），用于存储、管理和查询数据。SQL Server使用结构化查询语言（SQL）作为数据操作的标准语言，开发者可以使用SQL语句来创建数据库、表格、索引，以及执行数据查询、插入、更新、删除等操作。SQL Server提供了高性能、安全可靠的数据库解决方案，广泛应用于企业级应用程序、Web应用程序、数据分析等领域。

SQL Server具有许多强大的功能，如事务处理、数据复制、备份与恢复、安全性控制等，能够构建强大的数据库系统。除此以外，SQL Server还支持存储过程、触发器、视图等高级数据库对象，能够实现复杂的业务逻辑和数据处理需求。

1. FastAPI

FastAPI 是一个现代的高性能 Web 框架，用于基于标准类型提示使用 Python 构建 API。FastAPI具有简单易用的语法和直观的API设计，使开发者能够快速构建高效的API服务。同时，FastAPI支持自动化的数据验证、依赖注入、异步请求处理等功能，帮助开发者简化开发流程，提高开发效率。如下图所示，FastApi可以直接在后端机上进行调试而不需要与前端进行连接。

1. Vue

Vue.js是一款流行的JavaScript框架，用于构建交互式的Web界面。它采用了MVVM（Model-View-ViewModel）架构模式，通过数据驱动和组件化的方式，使开发者能够更轻松地构建复杂的前端应用。Vue.js具有简洁、灵活的语法，易于学习和上手，同时拥有丰富的功能和生态系统，包括Vue Router、Vuex、Vue CLI等工具，帮助开发者快速构建现代化的单页面应用（SPA）和响应式网页。Vue.js还支持服务端渲染（SSR）、虚拟DOM等高级特性，提供了优秀的性能和用户体验。

1. Ngork

Ngrok是一个免费的、跨平台的反向代理工具，可以将本地服务器暴露到公共互联网，从而实现内网穿透。使用Ngrok，可以让后端机所提供的端口在前端机上被使用，而不需要将代码和数据库拷贝到前端机。Ngrok提供了简单易用的命令行工具和Web界面，支持HTTP和HTTPS协议，同时还可以进行自定义域名和密码保护等设置。Ngrok的强大功能和方便的操作，使得开发者能够快速搭建临时的公开访问通道，方便与他人共享项目进展或进行远程调试。

1.2.2 开发工具

本次设计的物流订单管理系统在Windows 11操作系统上开发，采用Visual Studio Code编辑器进行开发，数据库管理系统采用Microsoft SQL Server。

1. Visual Studio Code

Visual Studio Code（简称“VS Code”）是Microsoft在2015年4月30日[Build](https://baike.baidu.com/item/Build/15992854?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)开发者大会上正式宣布一个运行于[MacOSX](https://baike.baidu.com/item/Mac OS X/470629?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)、[Windows](https://baike.baidu.com/item/Windows/0?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)和[Linux](https://baike.baidu.com/item/ Linux/27050?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)之上的，针对于编写现代[Web](https://baike.baidu.com/item/Web/150564?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)和[云应用](https://baike.baidu.com/item/%E4%BA%91%E5%BA%94%E7%94%A8/476249?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)的跨平台[源代码编辑器](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81%E7%BC%96%E8%BE%91%E5%99%A8/16273015?fromModule=lemma_inlink)，可在桌面上运行，并且可用于[Windows](https://baike.baidu.com/item/Windows/165458?fromModule=lemma_inlink)，[macOS](https://baike.baidu.com/item/macOS/8654551?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)和[Linux](https://baike.baidu.com/item/Linux/27050?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)。它具有对[JavaScript](https://baike.baidu.com/item/JavaScript/321142?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)，[TypeScript](https://baike.baidu.com/item/TypeScript/4314718?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)和[Node.js](https://baike.baidu.com/item/Node.js/7567977?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)的内置支持，并具有丰富的其他语言（例如[C++](https://baike.baidu.com/item/C++/99272?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)，python，[Java](https://baike.baidu.com/item/Java/85979?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)）和运行时扩展的[生态系统](https://baike.baidu.com/item/%E7%94%9F%E6%80%81%E7%B3%BB%E7%BB%9F/457895?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/visual%20studio%20code/_blank)。

1. Microsoft SQL Server

SQL Server 是Microsoft 公司推出的关系型数据库管理系统。具有使用方便可伸缩性好与相关软件集成程度高等优点，可跨越从运行Microsoft Windows 98 的膝上型电脑到运行Microsoft Windows 2012 的大型多处理器的服务器等多种平台使用。

Microsoft SQL Server 是一个全面的数据库平台，使用集成的商业智能 (BI)工具提供了企业级的数据管理。Microsoft SQL Server 数据库引擎为关系型数据和结构化数据提供了更安全可靠的存储功能，使您可以构建和管理用于业务的高可用和高性能的数据应用程序。

1.2 技术路线

本系统着重分析当前购物平台的订单物流管理过程，设计并实现了一些用户功能，诸如注册用户信息、修改用户信息、在购物中心界面按类别查找产品、根据产品返回图片、查询供应商产品单价总金额、查询运货商承接订单量和根据产品类别查询供应商供应量；此外，还实现了对于订货量、地区运货量、季节运货量、供应商产品单价总金额查询结果的多种方式呈现，分别为柱状图、折线图和线性表。本操作系统的开发技术路线如图1.1所示。

图 1.1

图 1-1 开发路线图

**2 功能分析与设计**

2.1 需求分析

**2.1.1 任务背景**

随着科技的飞速发展，网上购物已成为一种日益流行的消费方式。网上购物打破了时间和空间的限制，消费者足不出户即可浏览和购买各种商品，省去了实体店购物的繁琐程序。在线零售商通常提供比实体店更广泛的产品选择，满足了消费者对多样性和个性化需求移动设备的普及和支付技术的创新，使得网上购物更加方便快捷。随着科技的持续进步和消费者习惯的不断变化，预计网上购物将在未来继续保持增长势头。我们现在模拟出一个供应商数据库系统，实现对用户订单以及货物基本信息的查询。

**2.1.2 管理工作需求**

根据任务背景，可以得到以下合理的工作需求。

（1）每个订单可涵盖一至多项产品，需记录每项产品订购数量

（2）一项产品由一家供应商供应，一家供应商可供应多项产品

（3）每项产品属于一种类别，一个类别下有多项产品；类别对商品进行归类

（4）一个产品可以对应多个生产信息，每个生产信息的记录对应一个产品。

（4）每个客户可发出由一家运货商运送的多个订单，每家运货商可为每个客户提交的多张订单运货，每个客户的每张订单仅由一家运货商运送。

（5）客户下单时需提交货主姓名、地址、地区、国家、邮编、电话等联系信息，随着订单的进展，需要记录订购日期、发货日期、到货日期。

（6）客户收货后确认收货以及确认支付款项后，需要记录客户货款确认日期。

2.2 数据字典

建立的表如以下所示：

（1）运货商信息表：运货ID，公司名称，电话，运输工具。

（2）客户信息表：客户ID，公司名称，联系人姓名，联系人职务，地址，城市，地区，邮政编码，国家，电话，传真。

（3）货品类别信息表：类别ID，名称，说明，图片，创建时间，更新时间。

（4）供应商信息表：供应商ID，公司名称，联系人姓名，联系人职务，地址，城市，地区，邮政编码，国家，电话，传真，主页。

（5）产品信息表：产品ID，名称，数量，单价，库存量个，订购量，再订购量，供货状态。

（6）订单信息表：订单ID，订购日期，发货日期，到货日期，货款确认日期，运货费，货主名称，货主地址，货主城市，货主地区，货主邮政编码，货主国家，支付方式，保险信息。

（7）生产信息表：产品批次ID，产品ID，生产日期，保质期。

（8）订单详细信息表：订单ID，产品ID，数量，备注。

（9）生产批次信息表：生产批次ID。

（10）运费规则信息表：规则ID，重量下限，花费，标准，重量上限。

**2.2.1 数据项和数据结构**

数据项：以“orderidID”为例。含义说明：唯一标识每一个订单；类型：int型；别名：订单ID

数据结构：以“订单”为例。订单是该系统中一个核心数据结构。订单；含义说明：是物流订单管理系统的主体数据结构定义了一个学生的有关信息；组成：订单ID，订购日期，发货日期，到货日期，货款确认日期，运货费，货主名称，货主地址，货主城市，货主地区，货主邮政编码，货主国家，支付方式，保险信息。

**2.2.2 数据流**

因为课程设计中的数据流大都相似，以查询不同地区客户的订购量为例。数据流来源是订单详情中的订购数量相加，数据流去向是返回到前端最终绘制成图表展示，说明：从表中筛选并汇总得到的不同地区订购量的总和

**2.2.3 系统任务分析**

我们小组计划为来实现的数据库系统可以实现一下目标：

（1）按照类别查询浏览可供货产品、单位数量、单价、库存量

（2）查询浏览同一类别下不同产品及图片

（3）根据用户请求（用户可输入确定的订单号，或设计合适友好的模糊查询界面帮助用户确定想查询的订单），制订相应订单的明细，含订单每项产品的编号、名称、单位、数量、单价、总价，以及整张订单的总金额、订单的订购日期、货主名称和地址

（4）统计不同地区客户的订货量，以表格、曲线图、柱图方式呈现

（5）统计发往不同地区的运货量，不同季节的运货量，以表格、曲线图、柱图方式呈现

（6）统计每家供应商供应产品的单价总金额

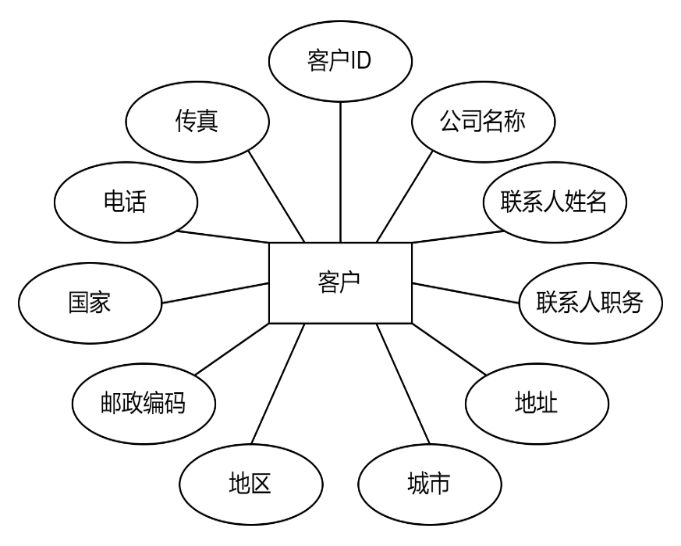
（7）统计各运货商承接的订单数量

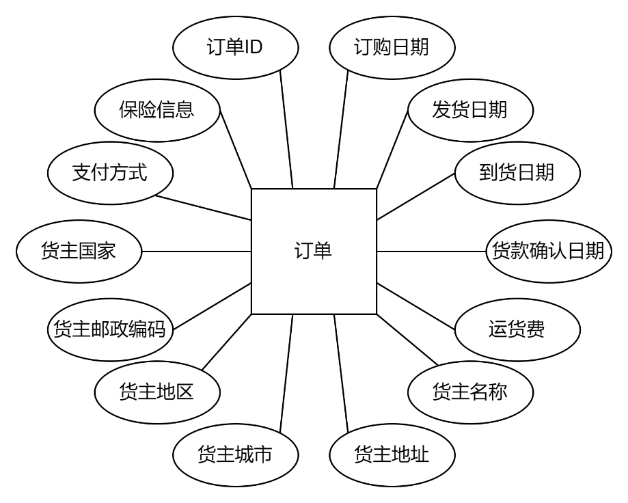
（8）统计同一类别下不同供应商供应的产品数量

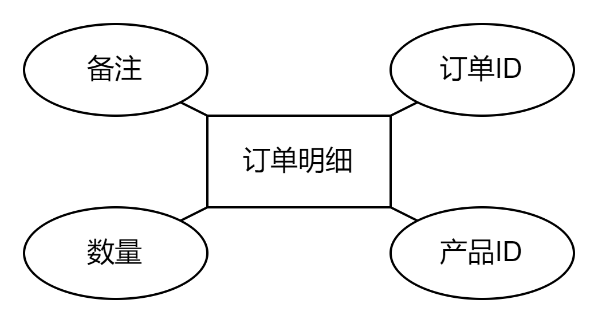
2.3 数据库概念结构设计

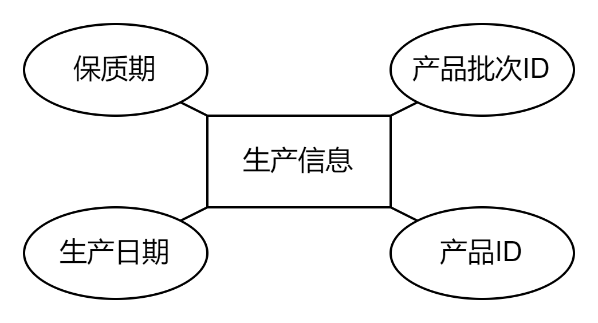
**2.3.1各模块的ER图**

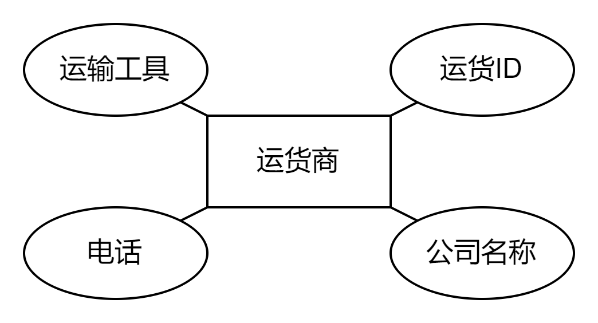
根据如上需求分析各模块的ER图如下：

（1）客户模块ER图如下：

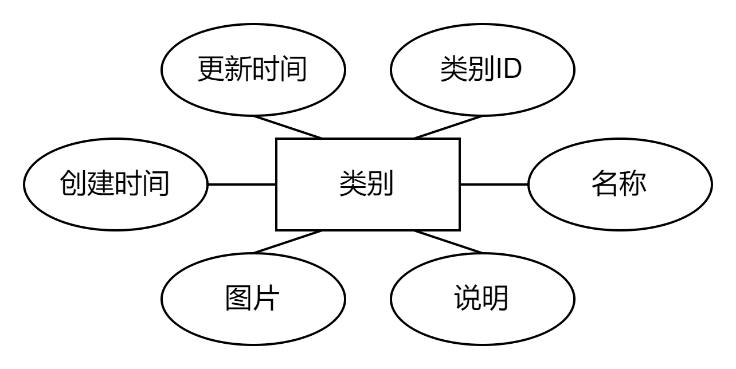
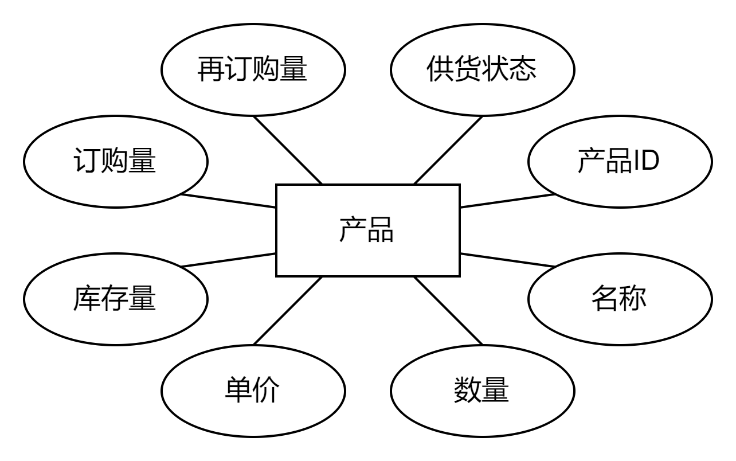
（2）订单模块ER图如下：

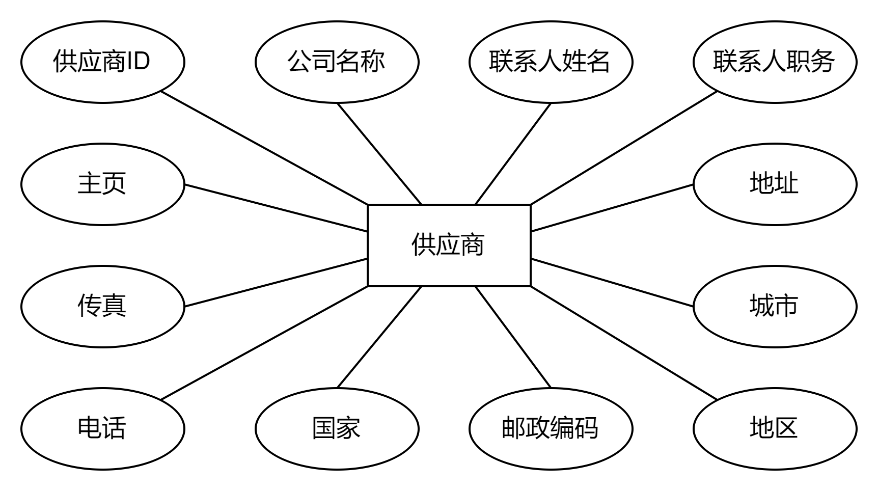
（3）订单明细模块ER图如下：

 （4）运货商模块ER图如下：

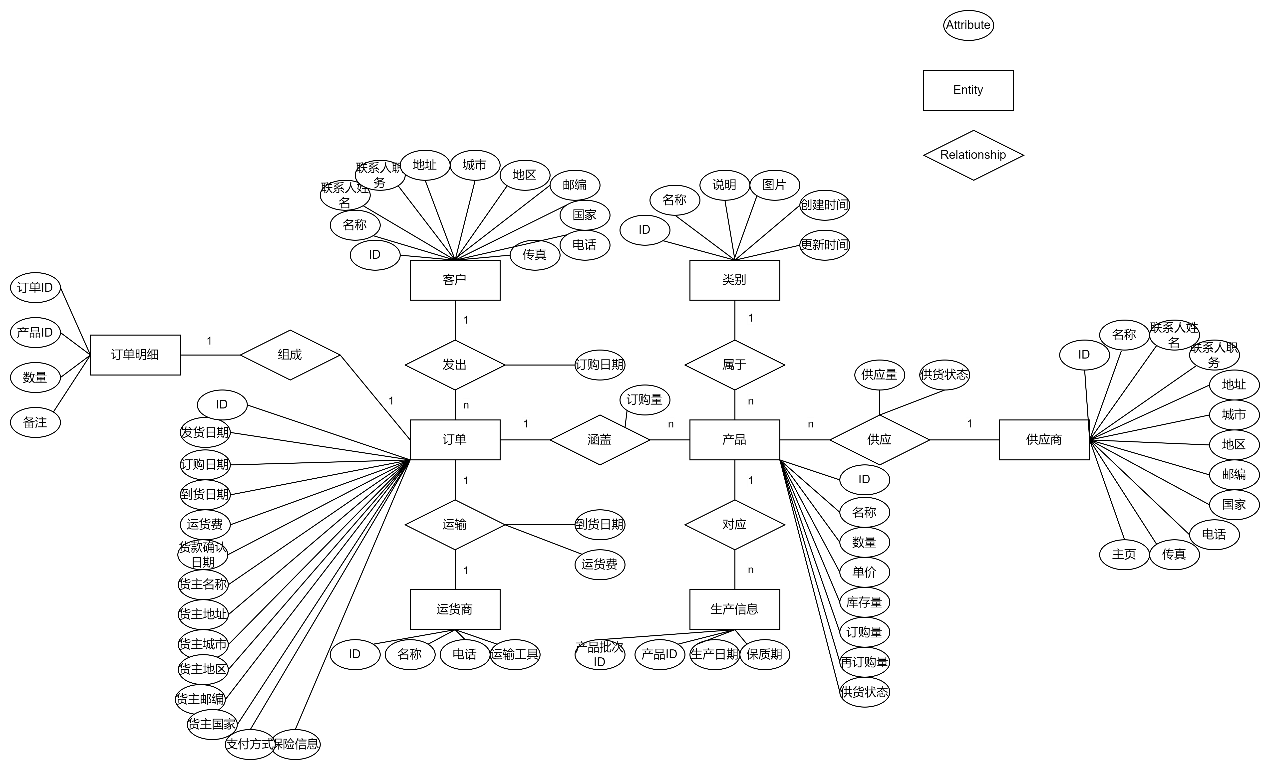
（5）生产信息模块ER图如下：

（6）产品信息模块ER图如下：

（7）类别模块ER图如下：

（8）供应商模块ER图如下：

**2.3.2 总体ER图**

具体物流订单管理系统的ER图如下所示：

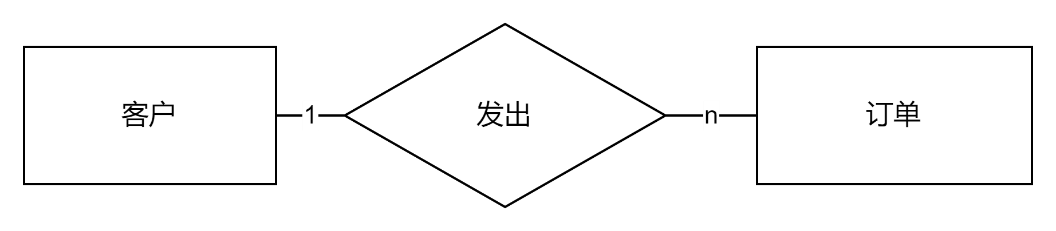
2.4 数据库逻辑结构设计

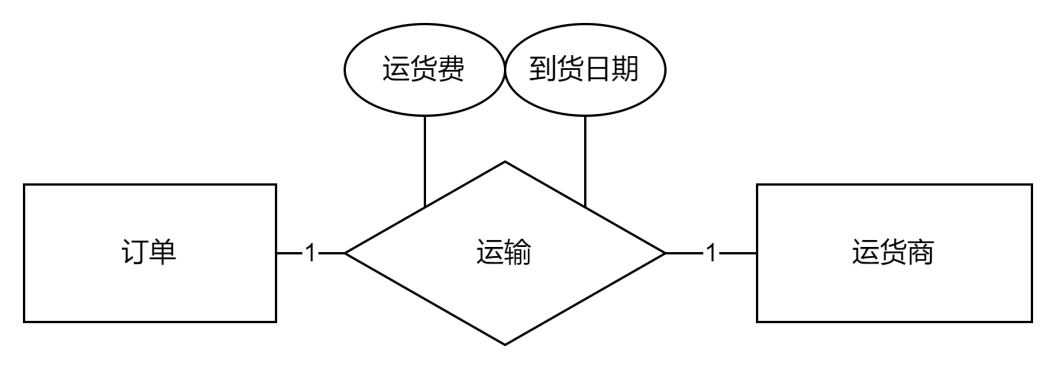
数据库逻辑结构设计即将概念结构转换为某个数据库管理系统所支持的数据模型，并对其进行优化。本课程设计采用关系数据库，需要实现E-R图到关系模型的转换。

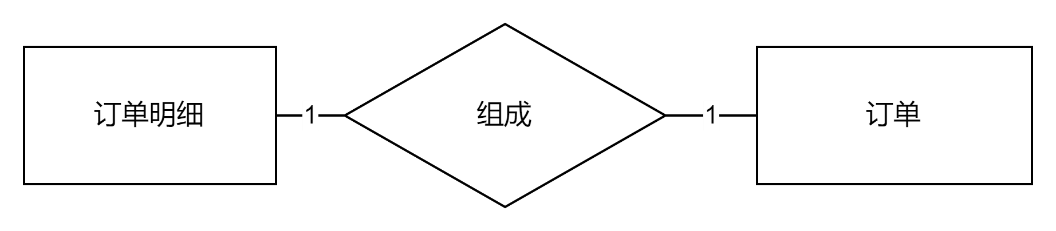
**2.4.1 E-R图向关系模型的转换**

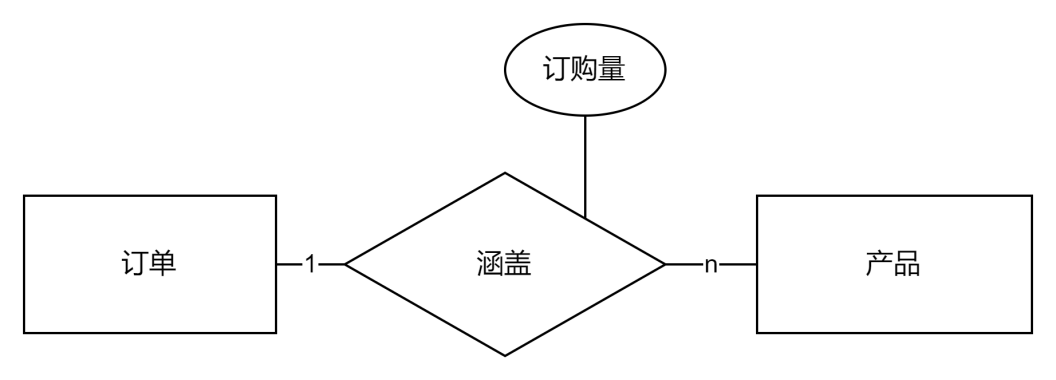
各个关系模型如下：

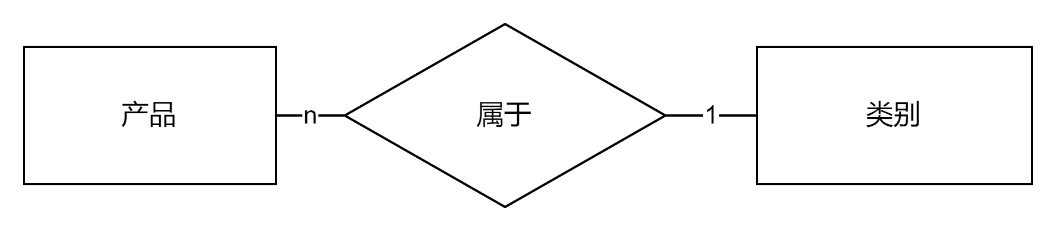
1. 发出（客户ID，订单ID，订购日期）：

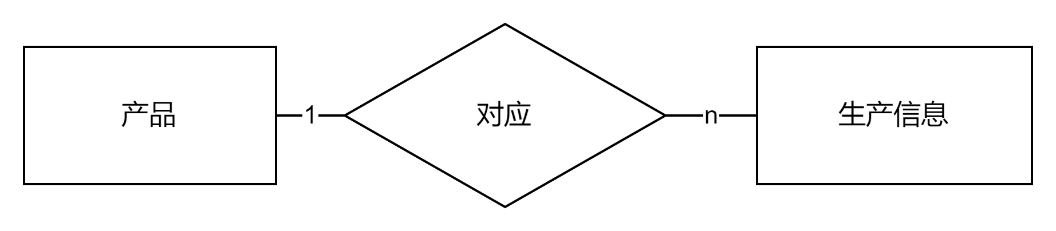


（2）运输（订单ID，运货商ID，到货日期，运货费）：

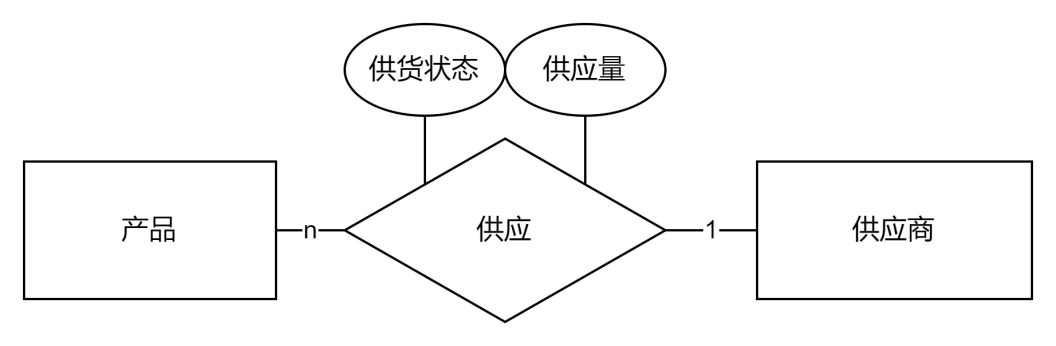
（3）（组成（订单ID）：

（4）涵盖（订单ID，产品ID，订购量）：

（5）属于（产品ID，类别ID）：

（6）供应（供应商ID，产品ID，供应量，供货状态）：

（7）对应（产品ID，产品批次ID）：

**2.4.2 利用数据依赖的理论对关系模式进行分析**

3NF定义：设关系模式R<U,F>是1NF，关系模式R中不存在这样的码X，属性组Y以及非主属性Z（Z不属于Y），是的X🡪Y，Y🡪Z成立，Y⇏X，则称关系模式R属于3NF。既关系模式中每一个非主属性既不传递依赖于码，也不部分依赖于码。

由于建立的关系模式较多，下面以两个关系模式为例，说明关系模式符合第三范式定义。

以orders为例，关系的主码是ID字段，根据ID字段可以唯一的确定该订单相应的客户ID，雇员ID，发货时间等等，没有属性是传递依赖于码或者部分依赖于码，符合了第三范式的定义。

再以供应关系为例，该关系中，产品ID和供应商ID一起组成主码，由主码唯一决定了相应的供货量以及供应状态，没有传递依赖和部分依赖，亦符合第三范式的定义。

**2.4.3最终关系模式设计**

一个完整的关系由5元组构成，一般写成如下模式：R(U,D,DOM,F,TF)，这其中，R是关系的名字，U是组成该关系的属性名集合，D属性组U中属性所来自的域，DOM是一个函数，使U中每个属性对应到它来自的域，F是属性间数据的依赖关系集合，TF是一个关系上的完整性约束条件的集合

最终的关系模式包含以下几个关系模式：

(1)客户—订单关系：发出

属性名集合：订单ID(order\_id)，客户ID(customer\_id)，订购日期(order\_date)

域：订单ID来自域orders(整数)，客户ID来自域cusrtomers(字符串)，订购日期来自域orders(日期)

DOM函数：DOM(order\_id) = orders; DOM(customer\_id) = customers; DOM(order\_date) =orders

数据依赖关系：订单ID决定该订单的客户ID和订购日期，(order\_id)(customer\_id,order\_date)

完整性约束：订单ID是主码，客户ID是外码，引用客户中的ID属性，订单日期不能为空(每个订单在下单时都有日期)

(2)订单—运货商关系：运输

属性名集合：订单ID(order\_id)，运货商ID(shippers\_id)，到货日期(arrival\_date)，运费(delivery\_cost)

域：订单ID来自域orders(整数)，运货商ID来自域shippers(整数)，到货日期来自域订单(日期)

DOM函数：DOM(order\_id)=orders, DOM(shippers\_id)=shippers, DOM(arrival\_date)=orders, DOM(delivery\_cost)=orders

数据依赖关系：订单ID决定该订单的运货商、到货日期以及运货费

(order\_id)(shippers\_id, arricval\_date, delivery\_cost)

完整性约束：订单ID是主码，运货商ID是外码，引用运货商中的ID属性，到货日期不为空(表中的订单均已经完成，都有到货日期)，运货费(订单都有运费)

(3)订单—订单明细关系：组成

属性名集合：订单ID(order\_id)，订单明细(order\_detail)

域：订单ID来自域orders，订单明细来自域orderdetail

DOM函数：DOM(order\_id) =orders, DOM(order\_detail)=orderdetail

数据依赖关系：订单ID决定订单明细，(order\_id)(order\_detail)

完整性约束：订单ID为主码，order\_detail不为空

(4)订单—产品关系：涵盖

属性名集合：订单ID(order\_id)，产品ID(product\_id)，订购量(order\_num)

域：订单ID来自域orders，产品ID来自域products，订购量来自域orderdetail

DOM函数：DOM(order\_id) =iorders, DOM(product\_id)=products, DOM(order\_num)=ordertail

数据依赖关系：订单ID和产品ID决定订购量，(order\_id,product\_id)(order\_num)

完整性约束：订单ID为主码，产品ID为外码，引用产品中的ID属性，order\_num不能为空(产品必须有相应的订购数量才能加入订单)

(5)产品—类别关系：属于

属性名集合：产品ID(product\_id)，类别ID(category\_id)

域：产品ID来自域products，类别ID来自域category

DOM函数：DOM(product\_id) =products, DOM(category\_id)=category

数据依赖关系：产品ID决定类别ID，(product\_id)(category\_id)

完整性约束：产品ID为主码，类别ID为外码，引用类别中的ID属性

(6)供应商—产品关系：供应

属性名集合：供应商ID(supplier\_id)，产品ID(product\_id)，供应量(order\_num)，供货状态(supplier\_state)

域：供应商ID来自域suppliers，产品ID来自域products，供应量来自域orderdetail，供货状态来自域products

DOM函数：DOM(supplier\_id) = suppliers, DOM(product\_id)=products, DOM(order\_num)=orderdetail, DOM(supplier\_state)=products

数据依赖关系：产品ID决定供应商ID、供应量、供货状态

(product\_id)(supplier\_id,order\_num,supplier\_state)

完整性约束：产品ID是主码，供应商ID是外码，引用category中的ID属性，供应量不为空(最小为0)，供应状态不为空(bool值变量为0或1)

(7)产品—产品批次关系：对应

属性名集合：产品ID(product\_id)，产品批次ID(pici\_id)

域：产品ID来自域products，产品批次ID来自pici

DOM函数：DOM(product\_id) =products, DOM(pici\_id)=pici

数据依赖关系：产品ID决定产品批次ID，(product\_id)(pici\_id)

完整性约束：产品ID是主码，产品批次是外码，引用pici中的ID属性

2.5 数据库物理结构设计

数据库的物理结构是数据库在物理设备上的存储结构与存取方法，这个结构依赖于选定的数据库管理系统。数据库的物理设计就是为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程。

数据库物理结构设计分为两个方面：

1.确定数据库的物理结构，也就是其在关系数据库中主要指存取方法和存储结构； 2.对物理结构进行评价，评价的关键点在于查询时间上的消耗和空间利用效率。

**2.5.1确定存储结构**

本平台采用的数据库平台是Microsoft SQL Server，Microsoft SQL Server的物理结构设计是非常经典和成熟的。通过主数据文件（.mdf）、次数据文件（.ndf）和事务日志文件（.ldf）的组合，SQL Server提供了一种灵活而可靠的方式来存储和管理数据库。

（1）主数据文件

每个数据库只能有一个主数据文件，.mdf文件，其中包含了数据库的启动信息、大部分数据，并记录数据库还拥有哪些文件。在对数据库进行操作时，要对主数据文件进行备份，以方式主数据文件被恶意破坏。

1. 次数据文件

一个数据库除了主文件以外还有其他的文件，这些文件统称为次要数据文件，.ldf文件。当数据库的数据文件过大时，次数据文件可以协助将数据库文件的大小进行有效管理，使其保持在一个相对合理的范围内。此外次数据文件还可以分散数据，使得查询的速度提高。

1. 事务日记文件。

事务日记文件是用来记录数据库更新情况的文件，可以记录针对数据库的任何操作，并将记录的结果保存到独立的文件中，这些记录和检查点一起构成了数据库恢复的功能。

1. 索引设计

在SQL Server中建立数据库时，数据库会自动建立以主码值为索引值的升序排列的索引结构

**2.5.2评价物理结构**

（1）可靠性；通过事务日志文件可以保证数据库事务持久化，能够再数据故 障时 正确的恢复数据

（2）易于管理；每种文件都有明确定义的作用和功能，使数据库变得更加简 单；

（3）扩展性；次数据文件允许用户将数据库的不同部分存储在不同物理位置 上， 提高了数据库的扩展性。

2.6 模块

按照上述分析将系统最终划分为如下5个模块：

（1）用户注册模块：

用户如果希望使用本平台的功能则必须完成注册，需要设定账号，密码，信息填写完整后即可提交注册申请并使用本平台；

（2）用户登录模块

输入用户的ID和密码可以登录到平台并实现相应的功能

（3）购买模块

在购物中心界面，客户可以根据自己的需求选择购买某种产品

（4）查询模块

在查询界面，用户可以查看不同地区的运货量、供货量，不同季节的运货量，不同产品的图片等等

（5）用户信息修改模块

在用户信息界面，用户可以更改自己的信息并提交到数据库

3.**实现**

3.1 前端界面实现

此次数据库课程设计的前端界面开发主要借助vue框架集成开发环境。开发界面如图4-1所示。



**3.1.1 界面设计**

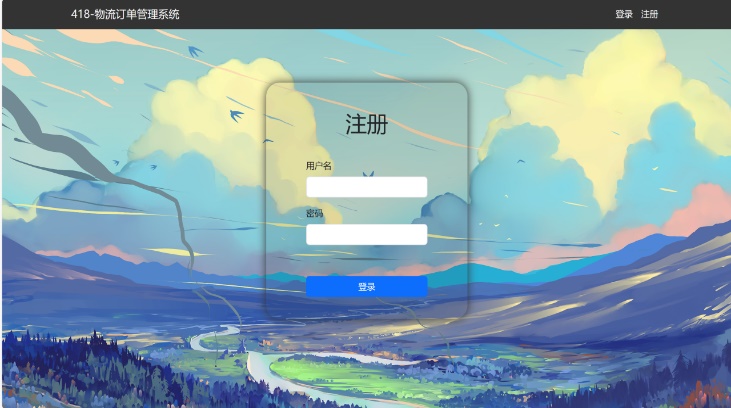
 用户界面如图4-2-1至图4-2-6所示。其中主要展示了登录界面、菜单界面、具体功能界面等主要界面设计。

图4-1-1 登录界面 图4-1-2 注册界面



图4-1-3 用户界面 图4-1-4 购物中心界面

3.2 后端代码实现

此次数据库课程设计的代码量较多且具有冗余性，为此仅选取核心和关键部分的代码进行说明和解释。

**3.2.1 VScode与SQL Server的连接和交互**

因为在我们的课程设计中，使用了vscode作为编译器，python作为主要的后端语言，为了实现vscode和数据库的各种交互式操作，引入了python中的pymssql包。现对该方法进行简介：

（1）首先要对数据库进行操作，使得数据库能够和编译器进行连接，打开数据库的用户名密码登录，同时阿静TCP/IP协议打开

（2）使用pip -install pymssql语句在电脑python解释器的路径下安装相应的pymssql包，之后在相应的python文件中使用import pymssql引入pymssql包等候使用

（3）connect对象：使用pymssql连接数据库要使用命令pymssql.connect命令进行连接，在连接中要为connect对象提供相应的主机名、用户名、密码、数据库名称

（4）cursor对象：connect对象的一个函数，使用这个函数对数据库进行访问和操作，在进行修改操作如更新删除时，要对connect对象进行commit

（5）获取cursor中的数据，使用两种方法：

* 1. fetchone：获取cursor中的第一条数据
  2. fetchall：获取cursor中的所有信息

（6）在结束对数据的所有操作后关闭游标和连接

**3.2.2 部分功能代码**

为了更好地展示此次数据库课程设计的具体实现，节选了一些基本功能的代码，来说明vscode是如何与SQL server进行交互的。

1. 后端代码

（1）.实现用户注册的功能

import json

import models

import pymssql

import datetime

from decimal import Decimal

connect = pymssql.connect( #连接数据库

server = 'ZIJING-PC',

user = 'sa',

password = '19222126',

database = 'test',

as\_dict = True

)

cursor = connect.cursor()

def RegisterCustomer(info:models.register\_customer): #执行数据库语句

cursor.execute('insert into customers values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%d)', (info.id,'s','s','s','s','s','s','s','s','s','s',info.password))

connect.commit()

return {"result": 'success'} #注册成功向前端返回success信息

（2）.查询不同地区客户的订货量

def CountOrderNumOfArea():

cursor.execute(''' #执行sql语句

SELECT suppliers.area, SUM(orderdetail.num) AS total\_num

FROM suppliers

JOIN products ON suppliers.ID = products.sid

JOIN orderdetail ON products.ID = orderdetail.productID

GROUP BY suppliers.area;

''')

rows=()

for row in cursor: #从游标中获取数据并以json格式返回给前端

rows+=(row,)

return json.dumps(rows,default = Convert)

2. 前端代码：用户登录

<template>

<div class="row justify-content-md-center">

<div class="header">

<h1>登录</h1>

</div>

<div class="col-8">

<form @submit.prevent="login">

<div class="mb-3">

<label for="username" class="form-label">用户名</label>

<input v-model="username" type="txet" class="form-control" id="username">

</div>

<div class="mb-3">

<label for="password" class="form-label">密码</label>

<input v-model="password" type="password" class="form-control" id="password">

</div>

<div class="error-message">{{ error\_message }}</div>

<button type="submit" class="btn btn-primary">登录</button>

</form>

</div>

</div>

<!-- <div id="liveAlertPlaceholder"></div>

<button type="button" class="btn btn-primary" id="liveAlertBtn">显示警告框（alert）</button> -->

</template>

<script>

import { useStore } from 'vuex';

import { ref } from 'vue';

import router from '@/router';

export default {

name: 'LoginView',

setup() {

const store = useStore();

let username = ref("");

let password = ref("");

let error\_message = ref("");

let login = () => {

error\_message.value = "";

store.dispatch("login", {

username: username.value,

password: password.value,

success(userID) {

console.log("success");

console.log(userID);

router.push({

name: 'profile',

params: { userID: userID },

});

},

error() {

error\_message.value = "用户或密码错误！";

console.log("failed");

},

});

console.log(username.value, password.value);

};

return {

username, password, error\_message, login,

};

},

}

</script>

**3.2.3 SQL Server建立数据库**

建立以下8个基本表（表4-1至表4-8）

表4-1 shippers（运货商表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 运货ID | int |  | PK | 是 |
| 2 | Sname | 公司名称 | nvarChar | 80 |  | 否 |
| 3 | Phone | 电话 | nvarChar | 48 |  | 否 |
| 4 | Tool | 运输工具 | nvarChar | 10 |  | 否 |

表4-2 customers（客户表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | cID | 客户ID | nvarchar | 20 | PK | 是 |
| 2 | cname | 公司名称 | nvarchar | 80 |  | 否 |
| 3 | pname | 联系人姓名 | nvarchar | 60 |  | 否 |
| 4 | pjob | 联系人职务 | nvarchar | 60 |  | 否 |
| 5 | caddress | 地址 | nvarchar | 120 |  | 否 |
| 6 | city | 城市 | nvarchar | 120 |  | 否 |
| 7 | area | 地区 | nvarchar | 30 |  | 否 |
| 8 | postcode | 邮政编码 | nvarchar | 20 |  | 否 |
| 9 | country | 国家 | nvarchar | 30 |  | 是 |
| 10 | phone | 电话 | nvarchar | 48 |  | 否 |
| 11 | fax | 传真 | nvarchar | 48 |  | 是 |
| 12 | password | 密码 | int |  |  | 是 |

表4-3 category（类别表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 类别ID | int |  | PK | 是 |
| 2 | cname | 名称 | nvarchar | 30 |  | 否 |
| 3 | explain | 说明 | ntext |  |  | 是 |
| 4 | picture | 图片 | image |  |  | 是 |
| 5 | setup | 创建时间 | datetime |  |  | 否 |
| 6 | updatetime | 更新时间 | datetime |  |  | 否 |

表4-4 suppliers（供应商表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 供应商ID | int |  | PK | 是 |
| 2 | sname | 公司名称 | nvarchar | 80 |  | 否 |
| 3 | pname | 联系人姓名 | nvarchar | 60 |  | 否 |
| 4 | pjob | 联系人职务 | nvarchar | 60 |  | 否 |
| 5 | caddress | 地址 | nvarchar | 120 |  | 否 |
| 6 | city | 城市 | nvarchar | 120 |  | 否 |
| 7 | area | 地区 | nvarchar | 30 |  | 否 |
| 8 | postcode | 邮政编码 | nvarchar | 20 |  | 否 |
| 9 | country | 国家 | nvarchar | 30 |  | 否 |
| 10 | phone | 电话 | nvarchar | 48 |  | 否 |
| 11 | fax | 传真 | nvarchar | 48 |  | 是 |
| 12 | homepage | 主页 | ntext |  |  | 是 |

表4-5 projects（产品表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 产品ID | int |  | PK | 是 |
| 2 | cnam | 名称 | nvarchar | 80 |  | 否 |
| 3 | num | 数量 | nvarchar | 40 |  | 否 |
| 4 | price | 单价 | money |  |  | 否 |
| 5 | inventory | 库存量 | smallint |  |  | 否 |
| 6 | ordernum | 订购量 | smallint |  |  | 否 |
| 7 | reordernum | 再订购量 | smallint |  |  | 否 |
| 8 | supplystate | 供货状态 | bit |  |  | 否 |
| 9 | sid | 供应商ID | int |  | FK | 是 |
| 10 | cid | 类别ID | int |  | FK | 是 |

表4-6 orders（订单表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 订单ID | Int |  | PK |  |
| 2 | customerID | 客户ID | nvarchar |  | FK |  |
| 3 | employeeID | 供应商ID | Int |  | FK |  |
| 4 | orderdate | 订购日期 | datetime |  |  | 否 |
| 5 | startdate | 发货日期 | datetime |  |  | 否 |
| 6 | arrivaldate | 到货日期 | datetime |  |  | 否 |
| 7 | confirmtime | 货款确认日期 | datetime |  |  | 否 |
| 8 | delivercost | 运货费 | money |  |  | 否 |
| 9 | oname | 货主名称 | nvarchar | 80 |  | 否 |
| 10 | addr | 货主地址 | nvarchar | 80 |  | 否 |
| 11 | city | 货主城市 | nvarchar | 30 |  | 否 |
| 12 | area | 货主地区 | nvarchar | 30 |  | 否 |
| 13 | postcode | 货主邮政编码 | nvarchar | 30 |  | 否 |
| 14 | country | 货主国家 | nvarchar | 20 |  | 否 |
| 15 | paymethod | 支付方式 | nvarchar | 20 |  | 否 |
| 16 | insurance | 保险信息 | int |  |  | 是 |

表4-7 orderdetails（订单详细表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | orderID | 订单ID | int |  | FK | 否 |
| 2 | productID | 产品ID | int |  | FK | 否 |
| 3 | num | 数量 | smallint |  |  | 否 |
| 4 | remark | 备注 | ntnext |  |  | 是 |

表4-8 proinfo（生产信息表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | piciID | 产品批次ID | int |  | FK |  |
| 2 | productID | 产品ID | int |  | FK |  |
| 3 | prodate | 生产日期 | datetime |  |  | 否 |
| 4 | expiration | 保质期 | datetime |  |  | 否 |

表4-9 pici（生产批次表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ID | 生产批次ID | int |  | PK |  |

表4-10 rules（运费规则表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | ruleID | 规则ID | int |  | PK |  |
| 2 | weight | 重量下限 | int |  |  | 是 |
| 3 | cost | 花费 | int |  |  | 是 |
| 4 | criterion | 标准 | int |  |  | 是 |
| 5 | weight1 | 重量上限 | int |  |  | 是 |

数据库关系图如图4-9所示：

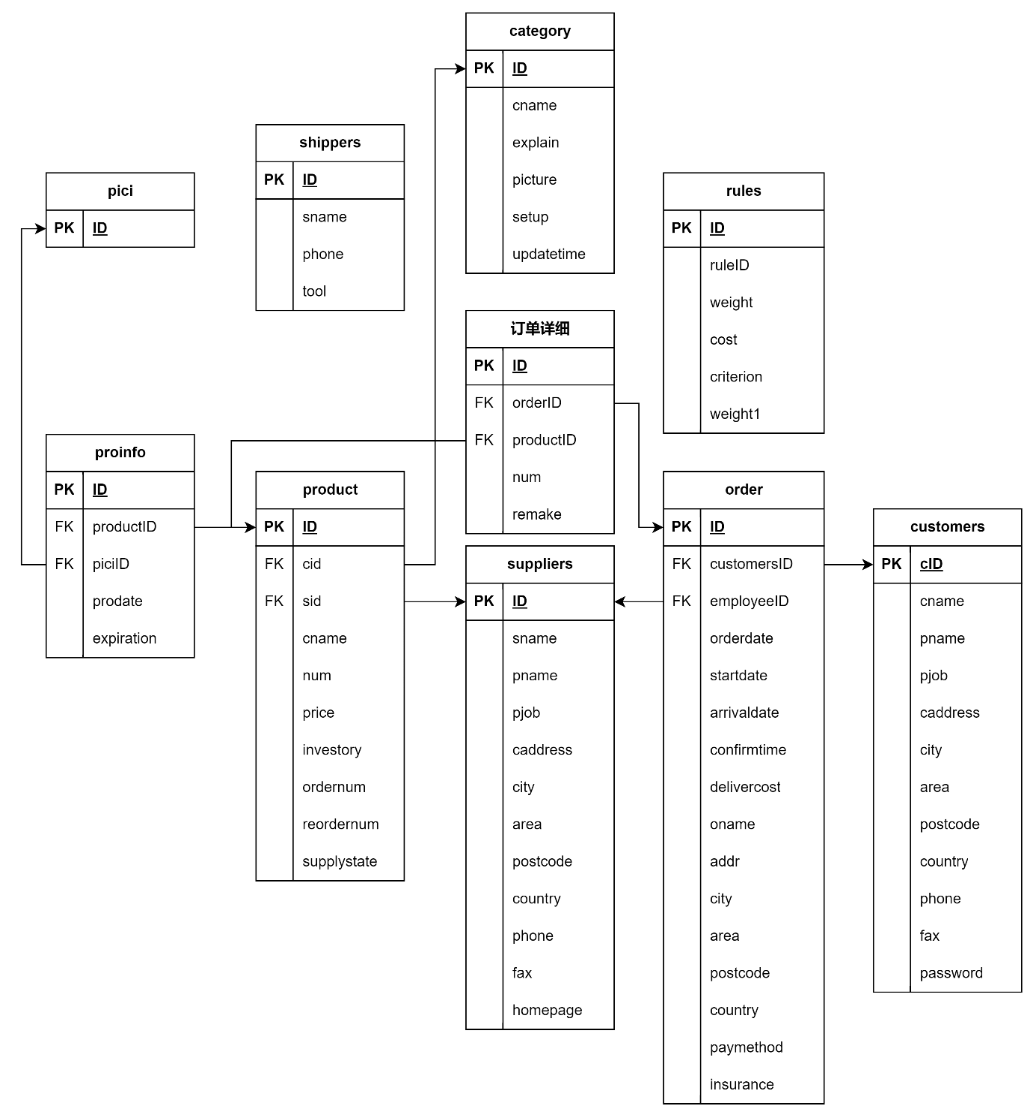


图4-9 数据库关系图

根据数据库逻辑结构设计，编写SQL语句建立数据库。

具体实现代码如下：

--数据库

Create database DatabaseCourseDesign

on(

name = DatabaseCourseDesign\_data,

filename = 'd:\sql\DatabaseCourseDesign.mdf')

log on(

name = DatabaseCourseDesign\_log,

filename = 'd:\sql\DatabaseCourseDesign.ldf'

)

go

--建立运货商

drop table if exists shippers

Create table shippers(

ID int primary key,

sname nvarCHAR(80) not NULL,

phone nvarchar(48) not NULL,

tool nvarchar(10) not NULL

)

--建立客户

drop table if exists customers

Create table customers(

cID nvarchar(20) primary key,

cname nvarCHAR(80) not NULL,

pname nvarchar(60) not NULL,

pjob nvarchar(60) not NULL,

caddress nvarchar(120)not NULL,

city nvarchar(120) not NULL,

area nvarchar(30) not NULL,

postcode nvarchar(20) not NULL,

country nvarchar(30),

phone nvarchar(48) not NULL,

fax nvarchar(48),

id int,

password int

)

--建立类别表

drop table if exists category

Create table category(

ID int primary key,

cname nvarCHAR(30) not NULL,

explain ntext,

picture image,

setup datetime not NULL,

updatetime datetime not NULL

best64\_img nvarchar[MAX]

)

--建立供应商表

drop table if exists suppliers

Create table suppliers(

ID int primary key,

sname nvarchar(80) not NULL,

pname nvarchar(60) not NULL,

pjob nvarchar(60) not NULL,

addre nvarchar(120) not NULL,

city nvarchar(30) not NULL,

area nvarchar(30) not NULL,

postcode nvarchar(20) not NULL,

country nvarchar(30) not NULL,

phone nvarchar(48) not NULL,

fax nvarchar(48),

homepage ntext

)

--建立产品表

drop table if exists products

Create table products(

ID int primary key,

cname nvarchar(80) not NULL,

num nvarchar(40) not NULL,

price money not NULL,

inventory smallint not NULL,

ordernum smallint not NULL,

reordernum smallint,

supplystate bit not NULL

sid int not NULL,

cid int not NULL

foreign key(sID) references suppliers(ID),

foreign key(cID) references cateory(ID),

)

--建立订单表

drop table if exists orders

Create table orders(

ID int primary key,

customerID nvarchar(20) not NULL,

employeeID int,

orderdate datetime not NULL,

startdate datetime not NULL,

arrivaldate datetime not NULL,

confirmtime datetime not NULL,

delivercost money not NULL,

oname nvarchar(80) not NULL,

addr nvarchar(80) not NULL,

city nvarchar(30) not NULL,

area nvarchar(30) not NULL,

postcode nvarchar(30) not NULL,

country nvarchar(20) not NULL,

paymethod nvarchar(20) not NULL,

insurance int not NULL

foreign key(customerID) references customers(cID),

foreign key(employeeID) references suppliers(ID),

)

--建立订单详细表

drop table if exists orderdetail

Create table orderdetail(

orderID int not NULL,

productID int not NULL,

num smallint not NULL,

remark ntext,

foreign key(productID) references products(ID),

foreign key(orderID) references orders(ID)

)

--建立生产批次表

drop table if exists pici

Create table pici(

ID int primary key

)

--建立生产信息表

drop table if exists proinfo

Create table proinfo(

piciID int,

productID int,

prodate datetime not NULL,

expiration datetime not NULL,

foreign key(productID) references products(ID),

foreign key(piciID) references pici(ID)

)

--建立运费规则表

drop table if exists rules

Create table rules(

ruleID int primary key,

weight int,

cost int,

criterion int,

Weight1 int

)

针对上述SQL语句的的说明如下：

1. primary key：将该字段设置为主码，同时也设置了该字段不能为空的约束，等同于某些列中的not NULL限制。
2. foreign key：将该字段设置为外码，将多个表组织成一个有效的关系数据库。提高了数据库的性能和可用性。

**4 系统测试与应用**

4.1 系统测试运行环境

测试系统软件环境：操作系统为Windows 11并且已经安装VScode和vue框架，同时已安装SQL Server Management Studio 2019。

测试电脑的硬件配置：处理器为Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz，机带RAM大小为16.0GB

4.2 数据库测试与运行

图4-2-1 根据产品类别查询供应商的供应量 图4-2-2 订单明细



图4-2-3 订单中心，模糊查询 图4-2-4 购物中心-购买界面



图4-2-5 依据产品类别返回图片 图4-2-6 订货量三种形式呈现



图4-2-7 地区运货量三种形式 图4-2-8 季节运货量三种形式



 图4-2-9供应商产品单价总金额查询 图4-2-10 运货商承接订单查询

**5.总结与展望**

**5.1总结**

本次设计的物流订单管理系统共实现了如下的功能：

5.1.1用户中心：

（1）用户注册系统：用户如果没有账号时可以点击主页右上角的“注册”键进 入注册界面，输入用户名和密码即可注册一个合法账户。

（2）用户登录系统：用户注册账号后可以点击主页右上角的“登录”键，输入 用户名和密码即可进入个人信息界面。

用户登录之后可以完成的功能有：

（a）用户信息修改：用户可以点击网页右上角的用户名进入个人信息界面，在 个人信息界面可以修改用户的姓名、职务、公司名称、地址、城市、地区、邮 政编码、国家、电话以及传真。修改完后点击最下方的保存按钮即可实现用户 信息的修改。

（b）购物：用户点击网页左上端的“购物中心”进入购物中心界面。用户需要 输入产品类别名筛选并购买产品。

（c）返回：返回登陆时的主界面。

5.1.2购物中心

（1）筛选产品：在购物中心右上角查询框输入产品类别ID,系统能够根据产品类 别ID筛选出指定产品，并显示在购物中心页面。

5.1.3查询中心

（1）查询订货量：将光标移动到查询中心并点击“订货量查询”，在右上角的 查询框输入产品类别ID，系统能够返回所选产品类别下物品的订货量并显示在查 询 中心界面。

（2）查询运货量

（a）查询地区运货量：将光标移动到查询中心并点击“地区运货量查询”， 系统能够自动生成地区运货量表并以折线图和柱状图的形式显示在查询中心界 面。

（b）查询季节运货量：将光标移动到查询中心并点击“季节运货量查询”， 系统能够自动生成季节运货量表并以折线图和柱状图的形式显示在查询中心界 面。

（3）查询供应商产品单价总金额：将光标移动到查询中心并点击“产品总金额 查询”，系统能够自动生成季节运货量表并以折线图和柱状图的形式显示在查 询中心界面。

（4）查询运货商承接订单量：将光标移动到查询中心并点击“承接订单量查 询”， 系统能够自动生成季节运货量表并以折线图和柱状图的形式显示在查询中 心界面。

（5）根据产品类别查询供应商的供应量：将光标移动到查询中心并点击“供应 量查询”，在右上角的查询框输入产品类别ID，系统能够返回所选产品类别下物 品 的供应量并显示在查询中心界面。

（6）查询图片：将光标移动到查询中心并点击“图片查询”，在右上角的查询 框输入产品类别ID，系统能够返回所选产品类别下物品的图片并显示在查询中心 界面。

5.1.4创新点

此次课程设计中，我们的一个创新点是在连接前后端时使用了ngrok作为连接工具，使得无需将两台电脑上的代码拷贝到一台电脑，随时可以进行交互，但也带来了时间上的问题，连接速度受网络影响较大

通过对本课题的研究以及该管理系统的设计与开发，我们对数据库的理论知识有了更进一步的理解，也能更加熟练地使用SQL语句。在整个数据库课设中，本小组成员分工明确，相互配合，积极讨论，团结互助，在整个过程中我们小组成员一起成长一起进步。在我们小组的团结合作下，通过查阅资料、网上搜索等方式共同完成了本次课程设计，最终我们都收获很多。

5.2不足

在本次的数据库系统设计过程中也存在着一些不足：

1. 订单只能一次插入一条信息。
2. 没有实现对相应订单的删除功能。
3. 没有实现管理员对相应账户的更改和维护。
4. 没有实现供应商和运货商对于订单的管理。
5. 由于使用了跨域链接，前端机向后端机请求数据时间较长
6. 没有向数据库中加入日志文件，不能再数据库发生故障后及时恢复数据

5.3展望

本次设计的物流订单管理系统面向供货商和运货商以及用户，该系统设计的初衷是希望能通过对数据库系统实现的实践中总结一些经验，当然更希望本系统能为大家带来一些实质上的用处。除此之外，通过本次数据库课设，我们不仅接触了不同的机器语言，还认识到各语言的使用特点。

总而言之，该物流管理系统也是一次难得可贵的尝试，在设计过程中同样遇见了诸多困难，并且最后的结果中也有一些不足，像是没有实现管理员对用户信息的维护以及保障数据库的安全性问题，相信在未来我们能够在此次设计基础上加以完善，实现诸如运货商、供应商对订单的增删改功能等，进一步推动418物流管理系统走向实际应用，满足用户的多方面要求。