|  |  |  |
| --- | --- | --- |
| 南农大  **数据库课程设计报告**  XH2 | | |
|  | 题 目: | 商品购买系统 |
|  | 组 长: | 张飒儒 学号: 19221220 |
|  | 组 员 | 任晨玮 学号: 9213010806 |
|  |  | 李明翰 学号: 19221201 |
|  | 学 院: | 人工智能学院 |
|  | 专 业: | 计算机科学与技术 |
|  | 班 级: | 计科212班 |
|  | 指导教师: | 黄芬 职称: 副教授 |
| 2023年 5 月 21 日 | | |

目录

**1开发技术、开发工具及技术路线**1

1.1开发技术及开发工具1

**1.1.1开发技术**1

1**.1.2开发工具**1

1.2技术路线2

**2功能分析与设计**3

2.1需求分析3

2**.1.1数据需求分析**4

2**.1.2功能模块划分**4

**2.1.3 其他性能需求**7

2.2系统设计4

2**.2.1系统整体流程设计**4

2**.2.2功能分析**5

2.3数据库概念结构设计6

2.4数据库逻辑结构设计8

2**.4.1 E-R图向关系模型的转换**9

2**.4.2 范式优化**9

2**.4.3最终关系模式的确立**9

2.5数据库物理结构设计10

2**.5.1确定存储结构**10

2**.5.2评价物理结构与优化**11

2.6模块划分12

**3实现**13

3.1前端界面实现11

3**.1.1用户界面设计**12

3**.1.3雇员/管理员界面设计以及其他界面**15

3.2后端代码实现13

3**.2.1 SQL Server建立数据库**18

3**.2.2设置索引**22

3**.2.3设置触发器**24

3**.2.4设置约束**28

3.3 **Java与SQL Server的连接和交互**36

3**.3.1 连接步骤**36

3**.3.2 功能代码**36

**4系统测试与应用**43

4.1系统测试运行环境43

4.2数据库测试与运行43

4**.2.1用户注册**43

4**.2.2商品分类页面**44

4**.2.3购物车页面**45

4**.2.4订单详情页面**46

4**.2.5订单处理页面**46

**5总结与思考**47

5.1数据库设计思路总结47

5.2不足与展望48

**5总结**49

1. 开发技术、开发工具及技术路线
   1. 开发技术及开发工具
      1. 开发技术

本系统是基于pyside6技术搭建，数据库管理系统为MicroSoft SQL Server,利用Java进行系统的前后端连接交互

**（a） 数据库管理系统（DBMS）：SQL Server 2022**

SQL　Server是由Microsoft公司开发的关系型数据库系统(RDBMS)，它是一款功能强大、可靠性高的数据库软件，广泛应用于企业级应用程序和数据中心。

SQL　Server支持多种数据存储方式，包括关系型表、XML 文档、空间数据、JSON文档等。它提供了一系列的工具和功能，用于管理和维护数据库，例如备份和恢复、性能监控、安全性管理、数据复制和同步等。

SQL Server还提供了一系列高级功能，例如支持分布式事务、数据压缩、实时数据分析、数据挖掘、内存优化等。此外，SQL　Server还支持多种编程语言的访问，包括C#、Java、Python等，以及各种应用程序框架和工具，例如 .NET、Visual Studio 等。

**(b) 前端技术：Python——PySide6**

PySide6是一款基于Python语言的开源图形用户界面（GUI）工具包，它是Ｑｔ库的绑定版本。PySide可以让开发人员使用Python语言来构建跨平台的桌面应用程序，具有良好的可移植性和易用性。

Pyside提供了一系列的模块和类，用于创建GUI界面、处理用户输入、实现事件处理、绘制图形等。它具有与Qt相同的面向对象的设计理念，使用了类似C++的信号和槽来实现事件处理和消息传递，支持多线程编程和国际化等功能。

PySide 可以与各种Python环境和开发工具集成，例如Python自带的IDLE编辑器、PyCharm、VS　Code等。此外，PySide还提供了一些额外的工具和库，例如Qt　Designer可视化设计器、Qt　Linguist国际化工具等，可以帮助开发人员更高效地创建和管理 GUI 应用程序。

**(c) 前后端交互技术：Java——JDBC 9.4.1**

JDBC (Java Database Connectivity)是Java语言中用于连接和操作关系型数据库的标注API。它提供了一组接口和类来访问各种数据库，包括MySQL、Oracle、SQL Server、PostgreSQL等。使用JDBC，开发人员可以编写Java应用程序来连接并与数据库进行交互，执行SQL查询、更新和删除操作等。

JDBC的工作原理如下：首先，使用JDBC驱动程序管理器加载数据库特定的驱动程序。然后，使用DriverManager类连接到数据库，并创建一个Connection对象表示数据库连接。接着，使用Statement或PreparedStatement对象执行SQL语句，获取ResultSet对象表示查询结果。最后，使用ResultSet对象获取查询结果，并对其进行处理。

JDBC提供了一种标准的数据库访问方式，使得Java应用程序可以与各种关系数据库进行交互，从而实现了数据库的访问和操作。

* + 1. 开发工具

1. 集成开发环境（IDE）:InteliJ IDEA(Java JDK 20)、PyCharm

* **InteliJ IDEA(Java JDK 20)**

IntelliJ IDEA是一款专业的Java开发IDE，拥有丰富的功能和工具，可以帮助开发者提高编码效率和代码质量。IntelliJ IDEA的代码编辑器具有强大的智能化功能，如代码提示、自动补全、代码格式化、代码重构等，能够帮助开发者快速编写高质量的代码降低开发成本、提高代码质量和可维护性.

* **PyCharm Edition 2021**

PyCharm是一款由JetBrains公司开发的Python集成开发环境（IDE），它提供了丰富的功能和工具，能够帮助开发者提高编码效率和代码质量。支持多种数据库和版本控制工具，如MySQL、PostgreSQL、Git等。PyCharm还支持插件和扩展机制，开发者可以根据自己的需求添加相应的插件和扩展，以满足不同的开发需求。

1. 数据库客户端：SQL Server Management Studio 19

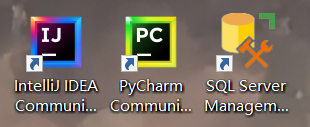
SQL Server Management Studio（SSMS）是Microsoft SQL Server的官方管理工具，它提供了一系列用于管理和开发SQL Server的工具和功能，包括查询、调试、备份和恢复、安全性管理、性能优化等。SSMS还提供了多种管理工具，包括数据库备份和还原、安全性管理、对象浏览器、任务计划等，能够帮助管理员轻松地管理和维护SQL Server数据库。

图1-1系统设计开发工具

* 1. 技术路线：

本系统利用PySide6技术来实现界面的设计，界面进行美化设计，使界面更加的清晰客观，具有更好的操作性和观赏性。

本系统着重分析了雇员和用户的功能实现、完成了例如：用户通过商品的类别来找到商品、根据商品价格进行下单、用户登陆一次界面可以进行多次下单；雇员对订单进行审核、合理化处理订单、对商品的库存量进行实时更新等功能，有效地规避雇员和用户的不正常操作。结合数据库安全思想和软件系统设计方法、利用JDBC数据库操作与Python语言的合理结合，设计实现了商品购物系统。

具体技术路线如图1-2：

图1-2 数据库技术路线图

概念结构设计

逻辑结构设计

数据库结构优化

前端界面设计

物理结构设计

数据库搭建

前后端交互连接

系统功能运行测试

系统功能完善

需求分析

1. 功能分析与设计
   1. 需求分析
      1. 数据需求分析

通过分析数据的具体需求，构建数据字典，如图2-1：

图2-1：数据字典

* + 1. 功能模块划分

根据系统用户调研，系统功能可以分为下面几个模块：

（a）登录界面：该界面登录和注册账户两大主要功能，细分为用户登录、雇员登录和用户注册、雇员注册四个功能，通过对用户和雇员的不同身份来进行后续功能界面的跳转。

(b) 用户界面：用户在此界面可以根据商品类别查看商品信息、进行商品的下单、用户可以通过购物车功能来查看自己所下单的商品、对购物车内的商品数量进行修改。

(c) 雇员界面：在此界面，管理员对用户的订单进行处理功能。管理员可以查看商品的所有信息、对商品的信息进行更新。

登录界面

雇员界面

用户界面

图2-2：系统功能大致模块

* + 1. 其他性能需求

① 客户ID必须为5字符的唯一标识

② 客户和管理人员具有不同权限，保证数据库安全性

③ 创建触发器，防止数据在存储过程中不一致

* 1. 系统设计
     1. 系统整体流程设计

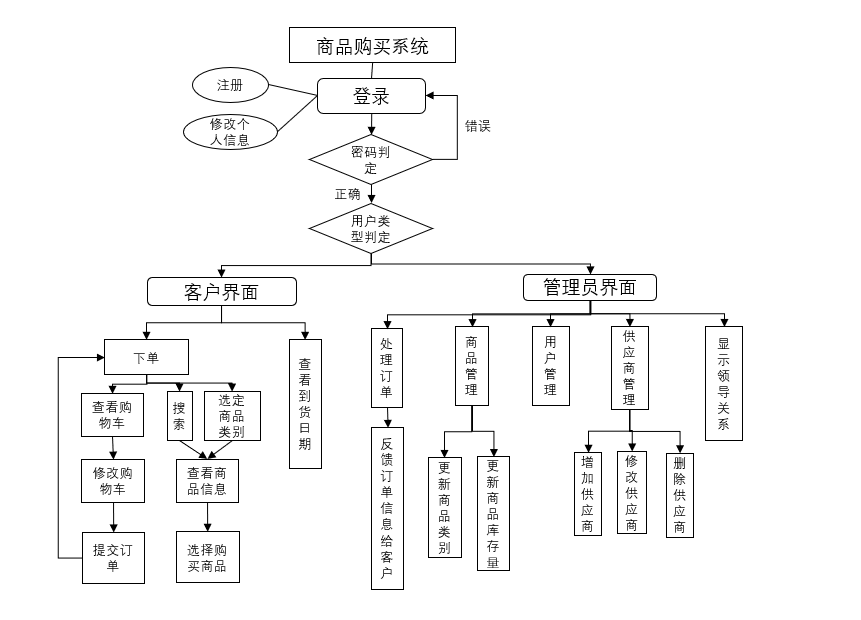
根据用户的需求分析以及功能模块的划分。系统整体流程如下图2-2所示：

图2-3：系统流程图

* + 1. 功能分析

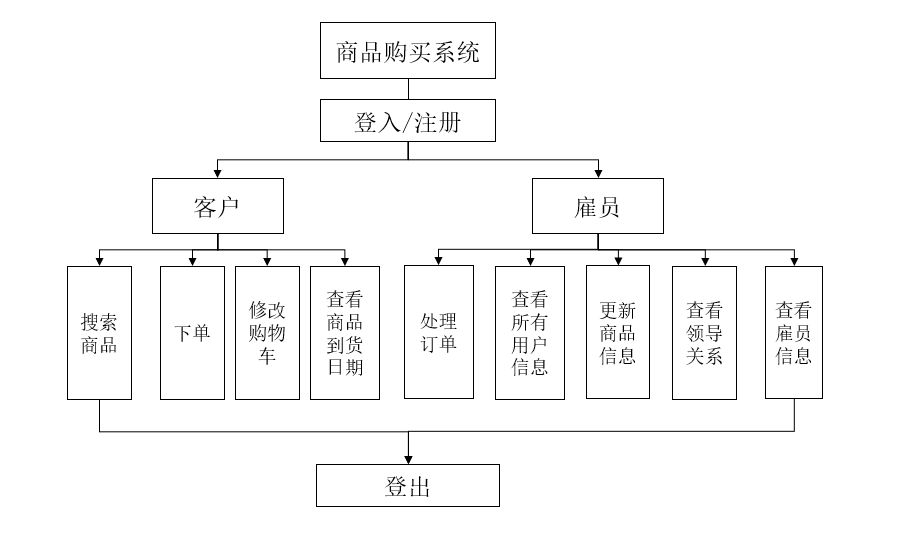
根据不同界面的不同功能进行系统功能的实现，具体如图2-3所示：

图2-4：系统功能图

* 1. 数据库概念结构设计

数据库概念结构设计是数据库设计的第一个阶段，主要是通过建立实体-关系图（E-R图）来描述系统中的实体、属性和它们之间的关系，从而形成一个抽象的、与具体实现无关的、逻辑上的数据库模型。这个模型是面向用户和需求的，主要目的是为了满足系统的需求和业务规则，以及为后续的逻辑结构设计和物理结构设计提供指导。

* + 1. 局部E-R图设计

对整体的数据字典进行分析，将整体关系模式分为供应属于关系和下单联系关系。

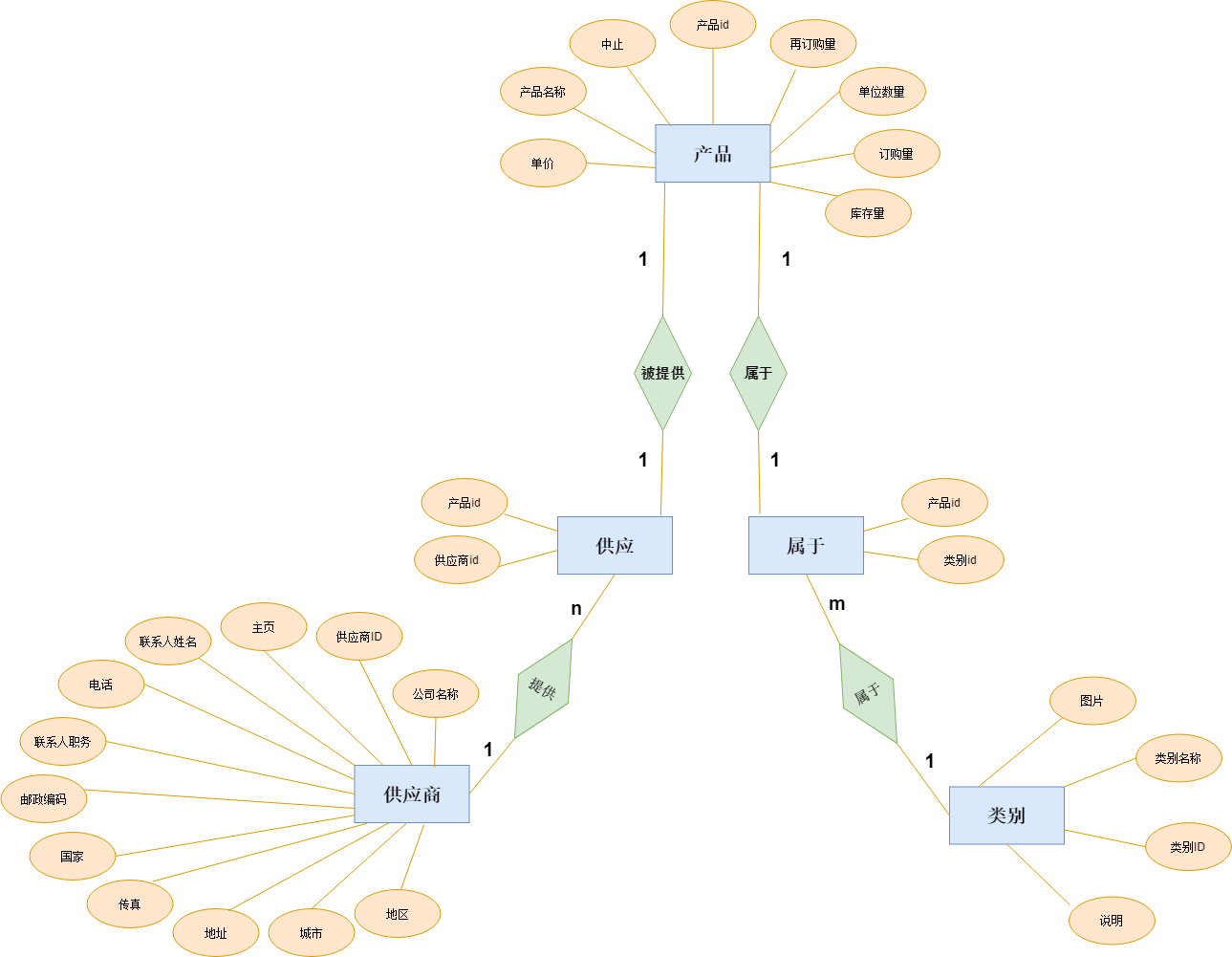
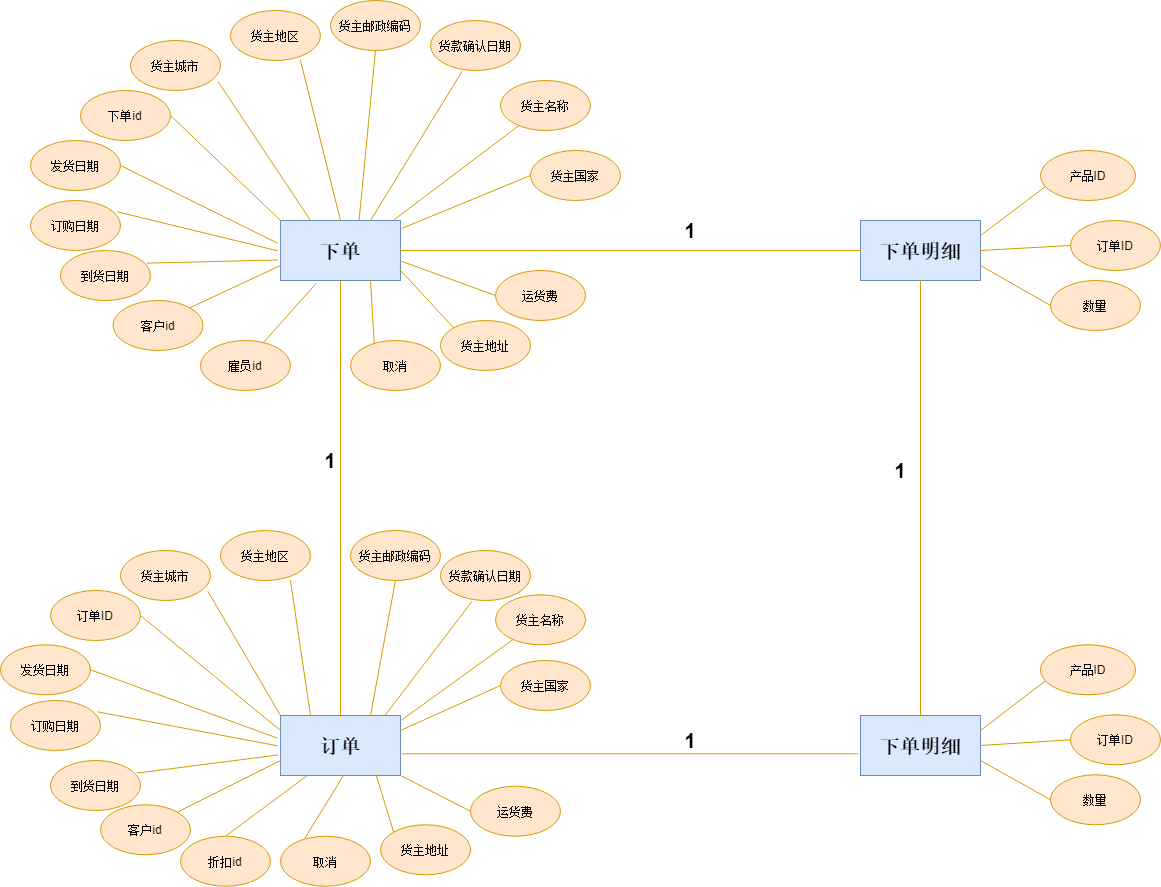
●供应属于关系E-R图（图2-5）:

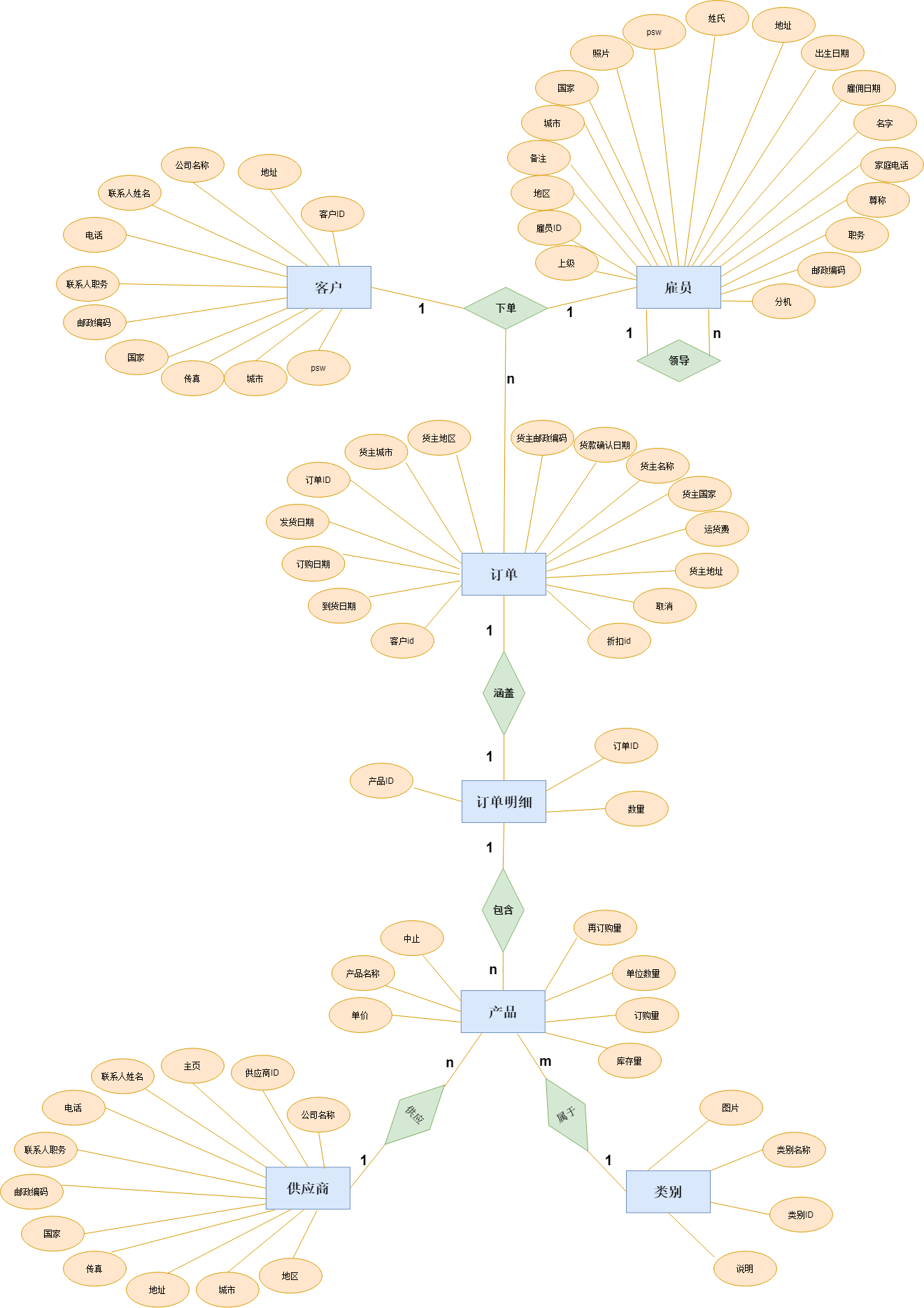
图2-5：供应属于关系E-R图

* 下单联系E-R图（图2-6）：

图2-6：下单联系E-R图

* + 1. 全局E-R图设计

将所有的局部E-R图进行合成，便于后续数据库物理结构设计和功能实现，满足系统的整体需求。如图2-7所示：

图2-7：全局E-R图

* 1. 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。本系统采用关系数据库，需要实现E-R图到关系模型的转换。

* + 1. E-R图向关系模式进行转换：

根据“实体的名称就是关系模式的名称、实体的属性就是关系的属性、实体的主码就是关系的”的原则，将E-R图中的每一个实体一一转换为关系模式：

* 客户（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）

雇员信息（雇员ID、姓氏、名字、职务、尊称、出生日期、雇用日期、地址、城市、地区、邮政编码、国家、家庭电话、分机、照片、备注）

* 类别（类别ID、类别名称、说明、图片）
* 供应商信息（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）
* 产品（产品ID、产品名称、单位数量、单价、库存量、订购量、再订购量、中止、图片）
* 订单（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、折扣ID、取消）
* 订单明细（订单ID、产品ID、数量）
  + 1. 利用数据依赖对关系模式进行分析与范式优化

由于所有关系模式中都只有一个主码，所以各个关系模式均至少满足第二范式。为进一步规范数据关系，则需要把关系模式继续优化到BCNF，需满足条件为：

① 所有非主属性都完全函数依赖于每个候选码；

② 所有主属性都完全函数依赖于每个不包含它的候选码；

③ 没有任何属性完全函数依赖于非码的任何一组属性

解决方法：将关系模式中所有非主属性设置为完全函数依赖于主码。

* + 1. 最终关系模式的确立（用三元组R<U,F>表示）：

客户（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）

雇员信息（雇员ID、姓氏、名字、职务、尊称、出生日期、雇用日期、地址、城市、地区、邮政编码、国家、家庭电话、分机、照片、备注）

类别（类别ID、类别名称、说明、图片）

供应商信息（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）

产品（产品ID、产品名称、单位数量、单价、库存量、订购量、再订购量、中止、图片）

订单（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、折扣ID、取消、***客户ID***）

订单明细（***订单ID***、***产品ID***、数量）

**注：**下划线字体为主码、斜体加粗字体为外码、每个关系模式中的所有非主属性全部完全函数依赖于主码

* 1. 数据库物理结构设计

数据库物理结构设计是数据库设计的最后一个阶段，主要是将概念结构设计转换为物理实现，包括确定数据存储的方式、存储结构、索引方式等，以及优化数据库的性能和可维护性。

本系统的数据库物理结构设计主要有两步：

① 选定数据库存储结构和存取，包括数据类型和长度、存取方式、存取系统、索引方式；

② 评价物理结构和优化，设置备份和维护策略和检测数据库的安全性能。

* + 1. 选定数据库存储结构和存取

① 确定数据类型和长度：

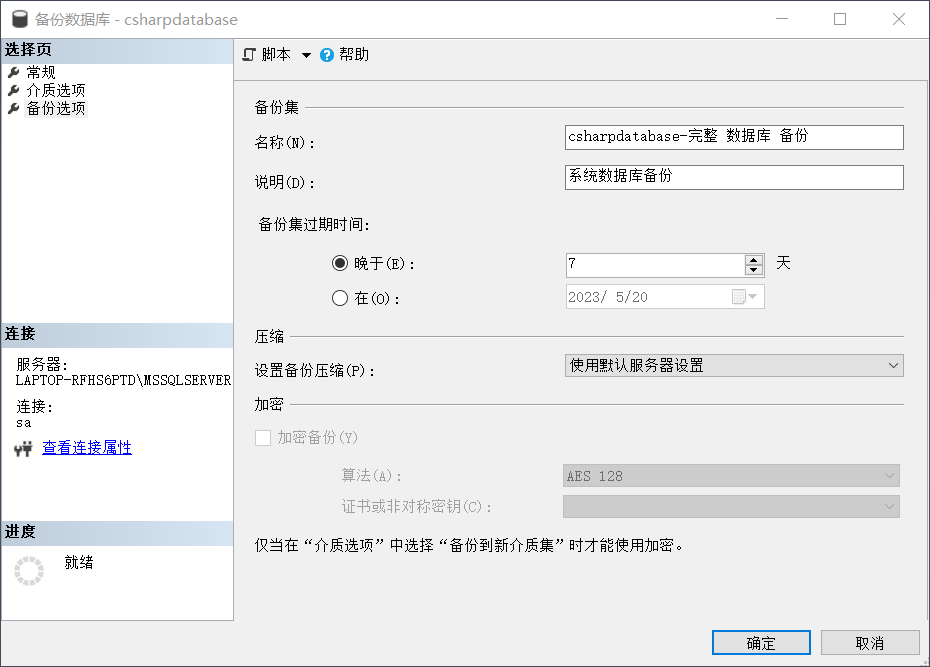
在概念结构设计中已经确定了每个属性的数据类型和长度，现在需要将其转换为实际的存储方式。字符类型使用固定长度或可变长度的方式存储，数字类型使用定点数或浮点数等方式存储。

② 存取结构选择：

本系统选择SQL Server 2022作为存储结构、数据库相关文件保存在计算机的磁盘中，分为主数据文件、次数据文件、日常事务文件:

1. 主数据文件：主数据文件通常使用扩展名为.mdf。主数据文件存储在操作系统的文件系统中，存储所有的表、索引和视图等数据库对象。主数据文件中存储着所有的数据记录和元数据信息、控制数据库的增长和空间分配、管理事务、日志和缓存等数据库操作;
2. 次数据文件：次数据文件通常使用扩展名为.ndf。可以用于扩展数据库的存储空间、提高数据库的性能、管理数据;
3. 日常事务文件：事务日志文件通常使用扩展名为.ldf。事务日志文件记录了所有的数据库操作，包括插入、更新、删除等操作，以及事务的开始和提交等信息。事务日志文件的主要作用是保证数据库的可恢复性和一致性。提高数据库的性能
   * 1. 评价物理结构并优化

① 设置备份和恢复策略：

选定系统数据库，进行备份策略的设置：完整恢复模式、完整备份、备份地址为“C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER01\MSSQL\DATA\BackUp”、备份集的过期时间为7天、备份设备中选定自己的设备。如图2-8、2-9所示：

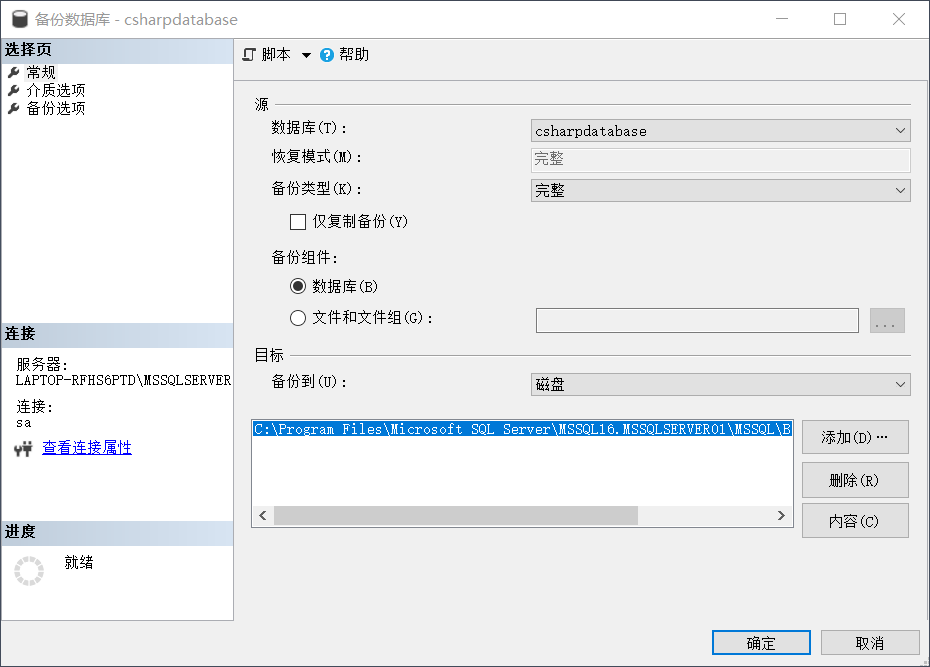
图2-8：数据备份

图2-9：备份恢复

② 检测数据库的安全性能：

i.检查备份日志：

以下使用Transact-SQL语句检查备份:

EXEC sp\_readerrorlog 0,1,N’Backup’

--过滤普通日志文件

RESTORE HEADERONLY FROM DISK = ’C:\ProgramFiles\MicrosoftSQLServer\MSSQL16.MSSQLSERVER01\MSSQL\DATA\BackUp\A1.dsk’

--查看备份文件头部信息

RESTORE VERIFYONLY FROM DISK = ' C:\ProgramFiles\MicrosoftSQLServer\MSSQL16.MSSQLSERVER01\MSSQL\DATA\BackUp\A1.dsk '

--确定备份的完整性

ii. 定期清理备份文件：

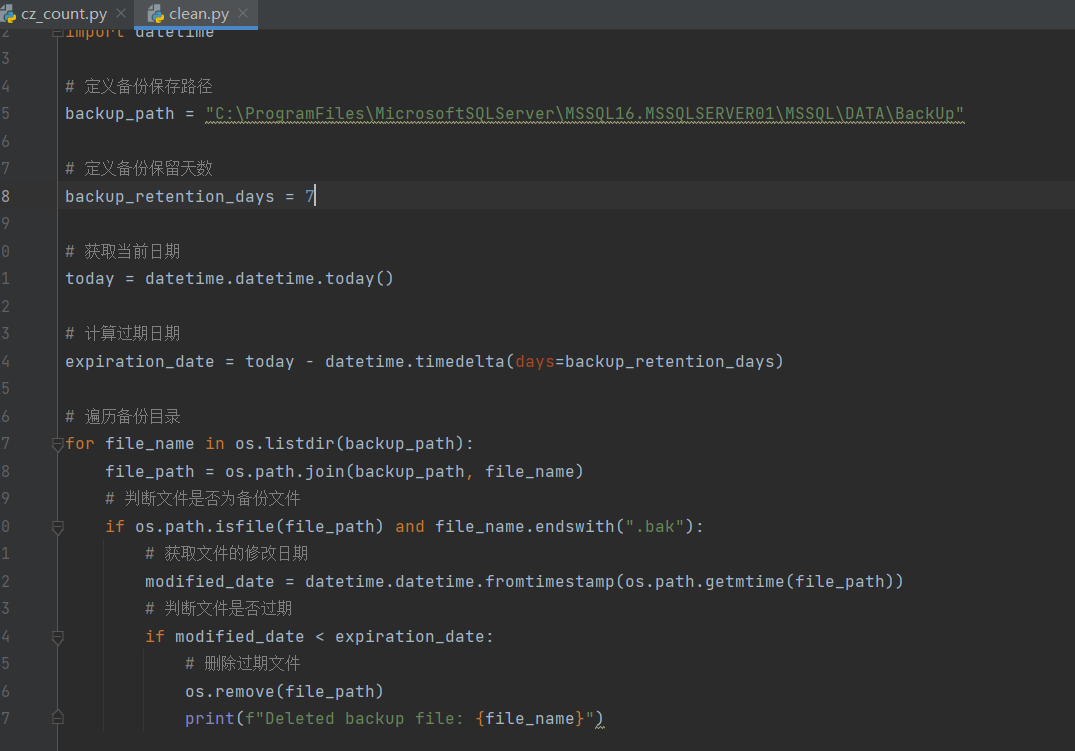
利用python代码执行备份清理脚本、清理存在超过7天（已经过期）的备份文件，以下是脚本代码（图2-10）：

图2-10：python删除过期备份文件脚本代码

* 1. 模块划分

经过前面数据库概念、逻辑、物理结构的设计，大致将系统分为三个模块：注册登录模块、客户购买模块、雇员管理模块。

* + 1. 注册登录模块：
* 客户注册：客户在此模块通过提供个人信息完成系统注册、有系统自动分配ID给客户；
* 客户登录：用ID、手机号、密码登录后，跳转至客户购买模块
* 雇员登录：雇员用ID、手机号、密码登录后、跳转到雇员管理模块
  + 1. 客户购买模块：
* 根据商品类别进行购买商品
* 购物车调整商品数量
* 查看之前订单
  + 1. 雇员管理模块：
* 查看订单并处理订单
* 查看所有用户信息
* 查看雇员信息
* 查看从属关系信息
* 查看所有类别和更新类别
* 查看所有供应商和更新供应商

1. 功能实现
   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PySide6和Qt Designer工具实现项目前端界面的设计和开发工作。

Qt Designer 是一个可视化的 GUI 设计工具，它可以帮助开发者创建漂亮、交互式的用户界面。它可以通过拖放控件、设置属性和布局等方式来创建用户界面，而无需编写任何代码。它支持多种控件和布局，并且可以与其他开发工具集成，提高了开发效率和代码可维护性。

PySide6 是一个 Python 绑定 Qt 库的版本，它提供了一种使用 Python 编程语言来创建跨平台的桌面应用程序的方式。它拥有强大的 GUI 库，易于学习和使用，并且是开源免费的。



图3-1-1

在Python与Java的交互中，使用了JPype，它是一个功能强大，易于使用的Python和Java之间的交互库，提供了Java虚拟机（JVM）封装，拥有完善的类型转换，方便实用。

图3-1-2：JVM封装

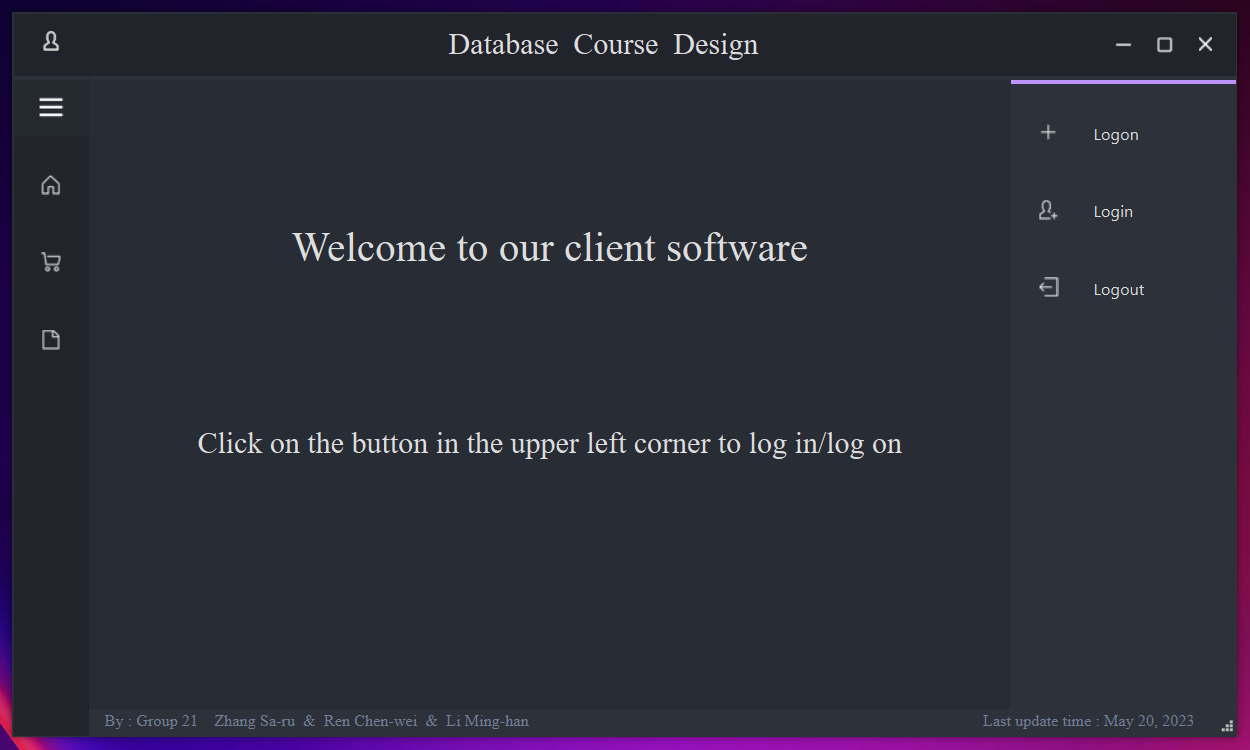
* + 1. 客户界面设计（如图3-1-3到3-1-9）

图3-1-3

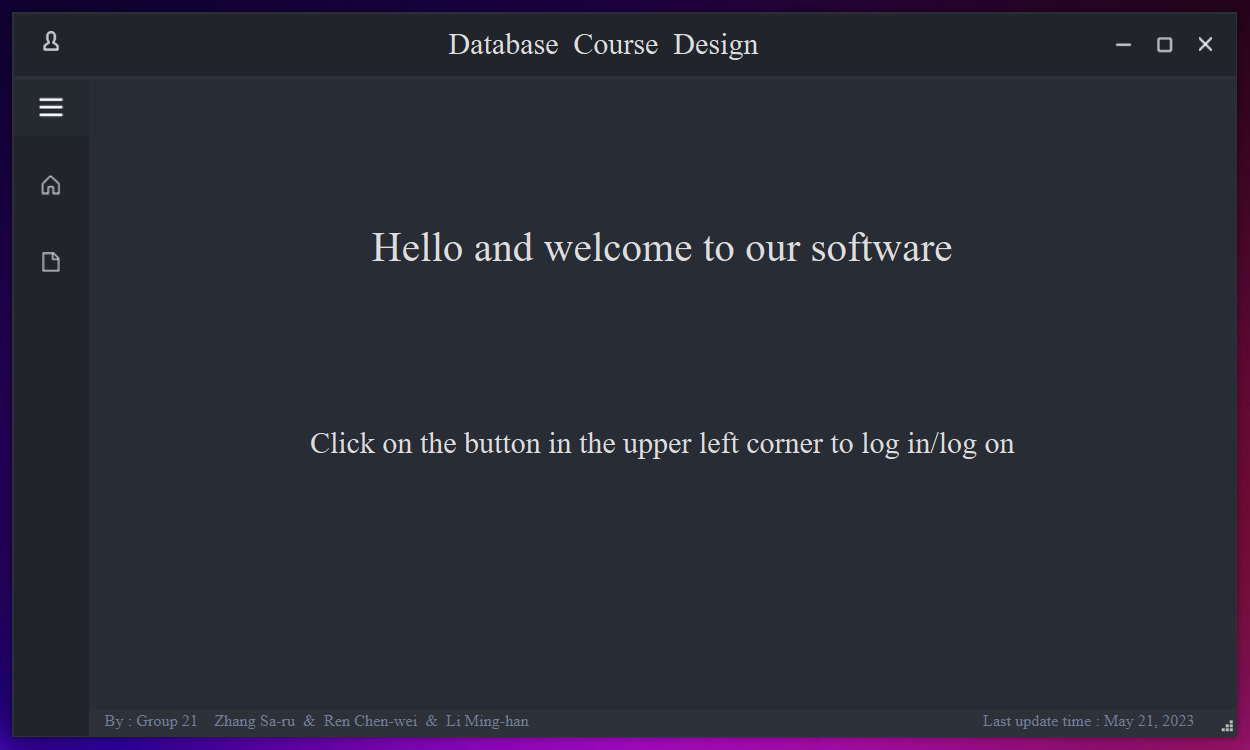


图3-1-4

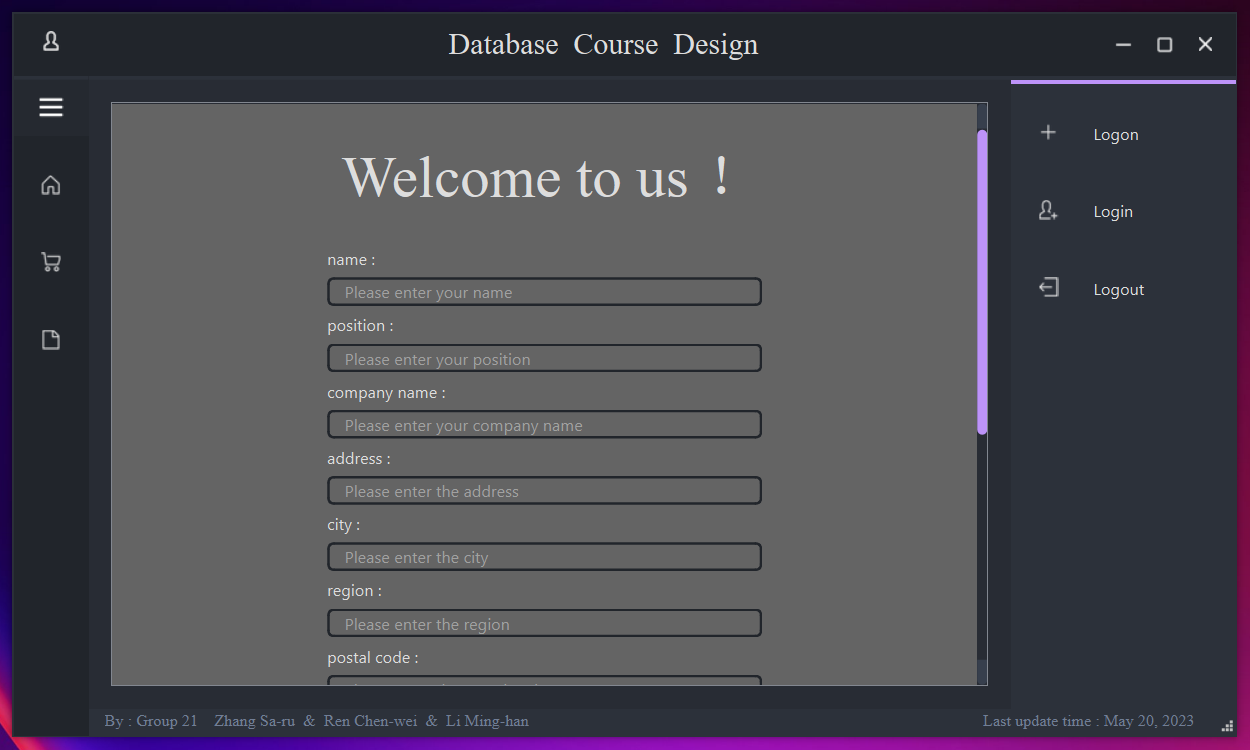


图3-1-5



图3-1-6



图3-1-7



图3-1-8

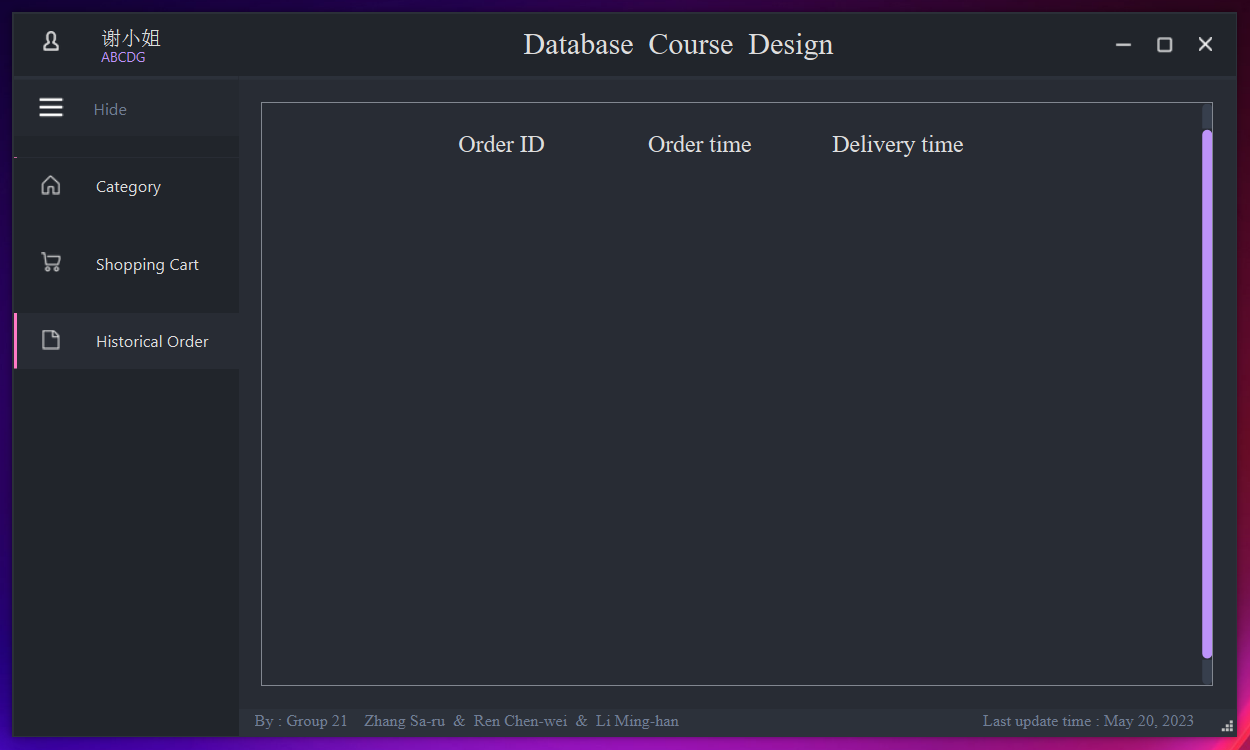


图3-1-9

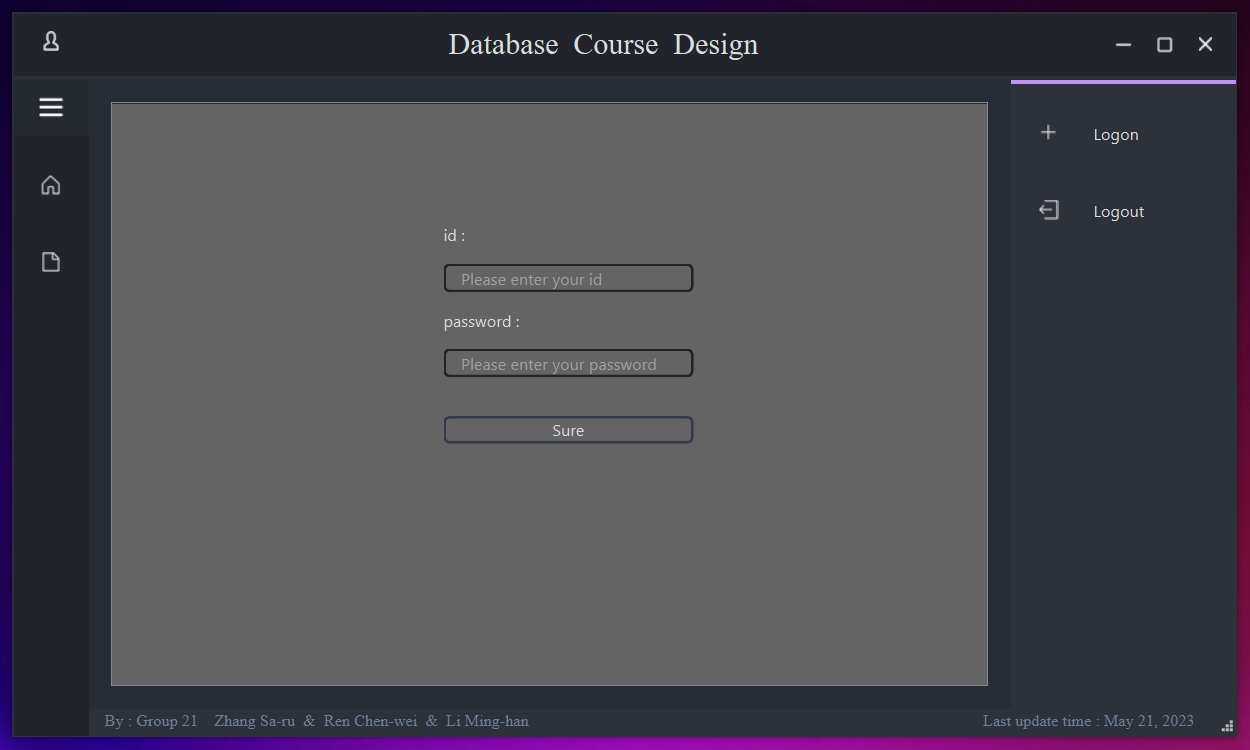
* + 1. 雇员界面设计（如图3-1-10、3-1-11）

图3-1-10

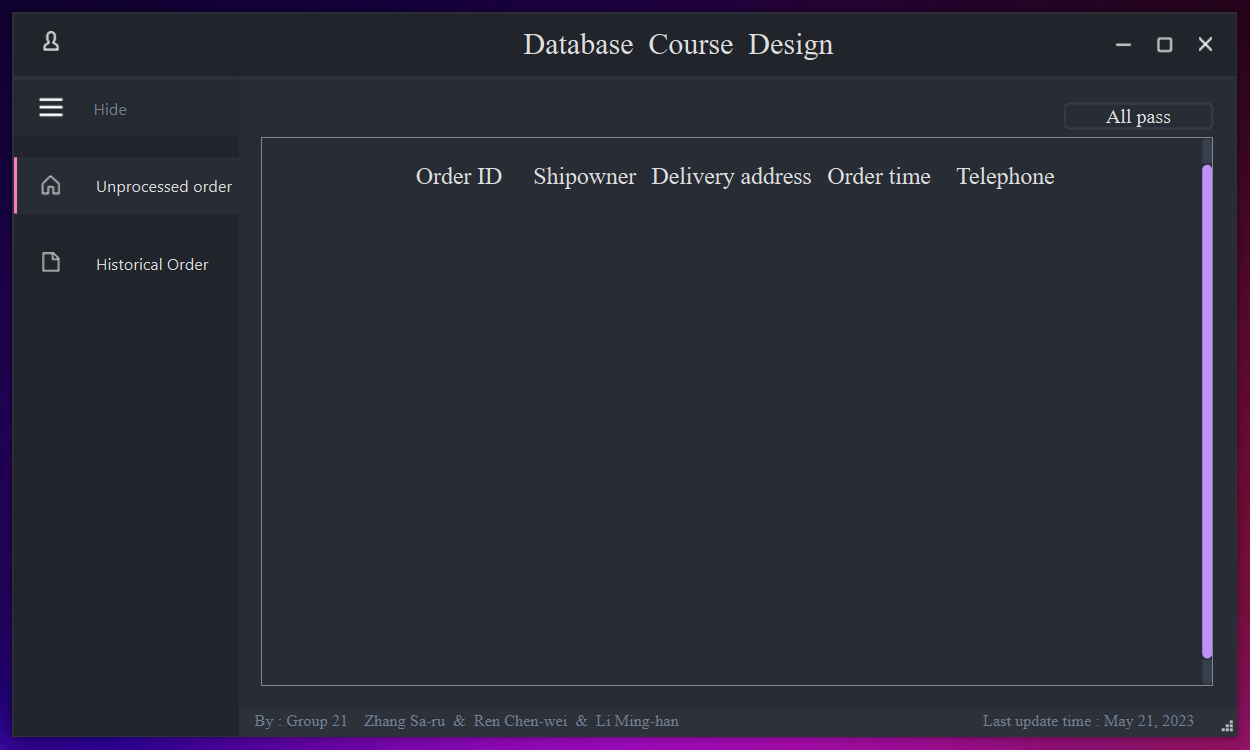


图3-1-11

* 1. 后端数据库搭建

用SQL Server 2022 进行数据库中表格的建立、触发器设置和导入测试数据。

* + 1. 利用SQL Server搭建数据库

① 建立表格（设置数据类型、主键、是否为空、具体说明等）

根据系统关系模式，建立如下12个表（表3-1至表3-12）：

* 表3-1：类别表（t\_class）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 类别ID | int |  | PK | 否 |
| 2 | 类别名称 | nvarchar | 30 |  | 是 |
| 3 | 说明 | ntext |  |  | 是 |
| 4 | 图片 | image |  |  | 是 |

* 表3-2：供应关系表（t\_gongying）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 产品ID | int |  | PK | 否 |
| 2 | 供应商ID | int |  | FK | 是 |

* 表3-3：领导关系表（t\_lingdao）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 雇员ID | int |  | PK,FK | 否 |
| 2 | 上级 | int |  |  | 是 |

* 表4-4：订单明细表（t\_odetails）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 订单ID | int |  | FK | 否 |
| 2 | 产品ID | int |  | FK | 否 |
| 3 | 数量 | smallint |  |  | 否 |

* 表4-5：订单表（t\_order）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 订单ID | int |  | PK | 否 |
| 2 | 订货日期 | datetime |  |  | 是 |
| 3 | 发货日期 | datetime |  |  | 是 |
| 4 | 到货日期 | datetime |  |  | 是 |
| 5 | 货款确认日期 | datetime |  |  | 是 |
| 6 | 运货费 | money |  |  | 是 |
| 7 | 货主名称 | nvarchar | 80 |  | 是 |
| 8 | 货主地址 | nvarchar | 120 |  | 是 |
| 9 | 货主城市 | nvarchar | 30 |  | 是 |
| 10 | 货主地区 | nvarchar | 30 |  | 是 |
| 11 | 货主邮政编码 | nvarchar | 20 |  | 是 |
| 12 | 货主国家 | nvarchar | 30 |  | 是 |
| 13 | 折扣ID | nvarchar | 20 |  | 是 |
| 14 | 取消 | bit |  |  | 是 |
| 15 | 客户ID | nvarchar | 10 | FK | 是 |

* 表4-6：产品表（t\_product）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 产品ID | int |  | PK | 否 |
| 2 | 产品名称 | nvarchar | 80 |  | 是 |
| 3 | 单位数量 | nvarchar | 40 |  | 是 |
| 4 | 单价 | money |  |  | 是 |
| 5 | 库存量 | smallint |  |  | 是 |
| 6 | 订购量 | smallint |  |  | 是 |
| 7 | 再订购量 | smallint |  |  | 是 |
| 8 | 中止 | bit |  |  | 是 |
| 9 | 图片 | image |  |  | 是 |

* 表4-7：供应商表（t\_provider）

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

* 表4-8：属于表（t\_shuyu）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 产品ID | int |  | PK,FK | 否 |
| 2 | 类别ID | Int |  | Fk | 是 |

* 表4-9：客户表（t\_user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 客户ID | nvarchar | 10 | PK | 否 |
| 2 | 公司名称 | nvarchar | 80 |  | 是 |
| 3 | 联系人 | nvarchar | 60 |  | 是 |
| 4 | 联系人职务 | nvarchar | 60 |  | 是 |
| 5 | 地址 | nvarchar | 120 |  | 是 |
| 6 | 城市 | nvarchar | 30 |  | 是 |
| 7 | 地区 | nvarchar | 30 |  | 是 |
| 8 | 邮政编码 | nvarchar | 20 |  | 是 |
| 9 | 国家 | nvarchar | 30 |  | 是 |
| 10 | 电话 | nvarchar | 48 |  | 是 |
| 11 | 传真 | Nvarchar | 48 |  | 是 |
| 12 | psw | nvarchar | 30 |  | 是 |

* 表4-10：雇员表（t\_worker）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 雇员ID | int |  | PK | 否 |
| 2 | 姓氏 | nvarchar | 40 |  | 是 |
| 3 | 名字 | nvarchar | 20 |  | 是 |
| 4 | 职务 | nvarchar | 60 |  | 是 |
| 5 | 尊称 | nvarchar | 50 |  | 是 |
| 6 | 出生日期 | datetime |  |  | 是 |
| 7 | 雇用日期 | datetime |  |  | 是 |
| 8 | 地址 | nvarchar | 120 |  | 是 |
| 9 | 城市 | nvarchar | 30 |  | 是 |
| 10 | 地区 | nvarchar | 30 |  | 是 |
| 11 | 邮政编码 | nvarchar | 20 |  | 是 |
| 12 | 国家 | nvarchar | 30 |  | 是 |
| 13 | 家庭电话 | nvarchar | 48 |  | 是 |
| 14 | 分机 | nvarchar | 8 |  | 是 |
| 15 | 照片 | nvarchar | 510 |  | 是 |
| 16 | 备注 | ntext |  |  | 是 |
| 17 | 上级 | int |  |  | 是 |
| 18 | psw | nvarchar | 20 |  | 是 |

* 表4-11：下单表（t\_xiadan）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | | 字段 | 数据类型 | 长度 | 码 | 是否可空 |
| 1 | 下单ID | | int |  | PK | 否 |
| 2 | 客户ID | | nvarchar | 10 | FK | 是 |
| 3 | 雇员ID | | int |  | FK | 是 |
| 4 | 客户电话 | | nvarchar | 48 |  | 是 |
| 5 | 下单时间 | | datetime |  |  | 是 |
| 6 | 通过时间 | | datetime |  |  | 是 |
| 7 | 货主名称 | | nvarchar | 60 |  | 是 |
| 8 | 货主城市 | | nvarchar | 30 |  | 是 |
| 9 | 货主邮政编码 | | nvarchar | 20 |  | 是 |
| 10 | 货主国家 | | nvarchar | 30 |  | 是 |
| 11 | 货主地址 | | nvarchar | 120 |  | 是 |
| 12 | 货主地区 | | nvarchar | 30 |  | 是 |

* 表4-12：下单明细表（t\_xiadandetail）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 数据类型 |  | 长度 | 码 | 是否可空 |
| 1 | 下单ID | int |  |  | FK | 否 |
| 2 | 产品ID | int |  | 30 | FK | 是 |
| 3 | 数量 | smallint |  |  |  | 是 |

② 数据库关系图

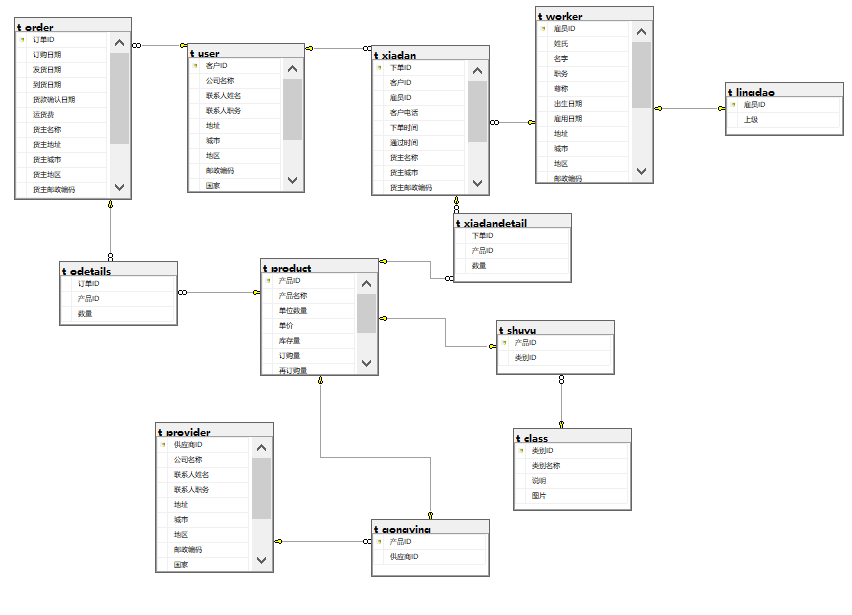
综合所建立的表格和设置的主键和外键，得到数据关系图如下图所示（图3-2-1）:

图3-2-2：数据库关系图

③ 表格建立流程

由于很多表格含有主键和外键，所以在建立表格时产生了相应的限制，具体建立流程如下：

T\_user->t\_worker->t\_product->t\_order->t\_odetails->t\_provider->t\_class

* + 1. 设置索引方式

在数据库中，可以通过设置索引来加速数据的查询和排序。索引是一种数据结构，它将表中的一列或多列映射到一个更快速访问的数据结构中。当你查询一个被索引的列时，数据库引擎可以通过索引直接定位到你需要的数据，而不需要扫描整个表。

本系统主要建立了主键索引和唯一索引用来快速访问数据。

① 主键索引：

本系统设置了如下主键索引（见具体代码）：

ALTER TABLE t\_class ADD PRIMARY KEY (类别ID);

ALTER TABLE t\_gongying ADD PRIMARY KEY (产品ID);

ALTER TABLE t\_lingdao ADD PRIMARY KEY (雇员ID);

ALTER TABLE t\_order ADD PRIMARY KEY (订单ID);

ALTER TABLE t\_product ADD PRIMARY KEY (产品ID);

ALTER TABLE t\_provider ADD PRIMARY KEY (供应商ID);

ALTER TABLE t\_shuyu ADD PRIMARY KEY (产品ID);

ALTER TABLE t\_user ADD PRIMARY KEY (客户ID);

ALTER TABLE t\_worker ADD PRIMARY KEY (雇员ID);

ALTER TABLE t\_xiadan ADD PRIMARY KEY (下单ID);

② 唯一索引：

本系统设置了如下唯一索引（见具体代码）：

CREATE UNIQUE INDEX t\_shuyu\_pid ON t\_shuyu (产品ID);



图3-2-3：数据库索引整体预览图

* + 1. 设置触发器

为了保证数据库中数据之间具有准确的对应关系，在关联属性增删改时需要对应的触发器在对应的表中进行相应的增删改操作，在当前数据库中，设置了如下触发器：

1. 雇员表t\_worker上设置增加，删除，修改雇员信息时候在对应的领导关系表中同步增加，删除，修改对应的雇员属性信息：

**CREATE** **TRIGGER** [dbo].[trigger\_lingdao\_insert]

**ON**  [dbo].[t\_worker]

**AFTER** **INSERT**

**AS**

**BEGIN**

**insert** **into** t\_lingdao(雇员ID,上级)

**select** 雇员ID,上级

**from** inserted

**END**

**CREATE** **TRIGGER** [dbo].[trigger\_lingdao\_id\_update]

**ON**  [dbo].[t\_worker]

**for** **update**

**AS**

**BEGIN**

**declare** @雇员ID **int**

**select** @雇员ID=雇员ID **from** inserted;

**declare** @雇员ID1 **int**;

**select** @雇员ID1=雇员ID **from** deleted;

if **update**(雇员ID)

**begin**

**update** t\_lingdao **set** t\_lingdao.雇员ID=@雇员ID **from** inserted

**where** t\_lingdao.雇员ID=@雇员ID1;

**end**

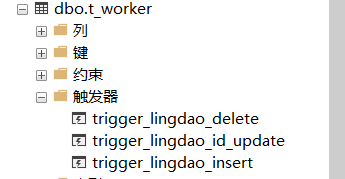
**END**  

图3-2-4：t\_worker表格触发器

1. 商品表上的t\_product设置触发器，总共有5个触发器，分别实现对供应，属于表同步增加更新信息，最后一个触发器完成在删除产品表中产品信息时候先将属于和供应表中信息删除，以免产生删除主表中数据后，属于和供应表外键没有对应到唯一主键而报错。

在商品增加时候，自动在供应和属于表中增加一行，初始对应的类别和供应商赋值为空，当更新时候，分别查找供应和属于表，将对应的商品id更新

实现删除的触发器是instead of触发器，但是因为一个表中因为数据库设计要求只能有一个instead of触发器，所以删除商品时候针对shuyu和gongying对应行的删除使用一个触发器实现使用了外连接语句：

**CREATE** **TRIGGER** [dbo].[trigger\_shuyu\_gongying\_delete]

**ON** [dbo].[t\_product]

**INSTEAD** **OF** **DELETE**

**AS**

**BEGIN**

--删除 Table2 中对应的行

**DELETE** t2

**FROM** t\_shuyu t2

JOIN deleted d **ON** t2.产品ID = d.产品ID; -- 根据外键关联条件删除对应行

--删除 供应 中对应的行

**DELETE** t3

**FROM** t\_gongying t3

JOIN deleted d **ON** t3.产品ID = d.产品ID; -- 根据外键关联条件删除对应行

--继续执行原始的删除操作

**DELETE** **FROM** t\_product **WHERE** 产品ID IN (**SELECT** 产品ID **FROM** deleted);

**END**;

1. 在订单表t\_order上设置了9个触发器

① 在插入之后判断，总共6个日期触发器，并且在订单表中增加日期错误1，日期错误2，日期错误3共3个属性，初值为false，分别代表发货时间，到货时间，货款确认时间是否错误。如果导入时候发现任意两个之间不满足下单时间<发货时间<到货时间<货款确认时间，就使用触发器更新判断错误的时间，并且在对应的日期错误属性栏设置为true，并且自动update修改时间以保证数据正确。例如下单时间比到货时间晚，那么插入改行时候，触发器触发更新改行的到货时间是下单时间的后10分钟，并且将改行的日期错误2设置为true，结合前端界面，当该用户取到该订单时候，判断这3个日期错误属性是否为false，如果为true，说明原始数据错误，系统自动更新为正确形式的数据，并在前端显示。

如下是其中一个示例：

**CREATE** **TRIGGER** [dbo].[CheckDateTimeTrigger11]

**ON** [dbo].[t\_order]

**AFTER** **INSERT**

**AS**

**BEGIN**

**UPDATE** o

**SET** 发货日期 = DATEADD(**MINUTE**, 5, i.订购日期),

日期错误1 = 'true'

**FROM** t\_order

**INNER** JOIN inserted i **ON** o.订单ID = i.订单ID

**WHERE** o.发货日期 <  i.订购日期

**END**

② 设置订单对应行删除后，订单明细中对应订单ID相同的所有行删除，使用instead of触发器，避免产生先删主表而订单明细中订单ID外键没有对应而异常情况

③ 库存量更新触发器，当订单的中止属性设置为True时，也就意味着订单被用户取消，该订单中购买的所有商品全部返还给商家，对应的产品表中对应商品的库存量相应恢复。实现方法，先根据当前订单ID连接查询订单明细中订单ID相等的所有行信息，在所有行信息中取出所有产品ID列信息和对应的购买数量信息，再级联查询到产品表中对应产品ID，将产品的数量返还给库存量。

**CREATE** **TRIGGER** [dbo].[update\_stock] **ON** [dbo].[t\_order]

**AFTER** **UPDATE**

**AS**

**BEGIN**

--检查是否更新了"取消"属性为true

IF **UPDATE**([取消]) AND EXISTS (**SELECT** 1 **FROM** inserted **WHERE** [消] = 1)

**BEGIN**

--获被取消的订单ID

**DECLARE** @canceledOrderIds **TABLE** (order\_id **INT**);

**INSERT** **INTO** @canceledOrderIds (order\_id)

**SELECT** 订单ID **FROM** inserted **WHERE** [取消] = 1;

**UPDATE** t\_product

**SET** 库存量 = 库存量 + t\_odetails.数量

**FROM** t\_product

**INNER** JOIN t\_odetails **ON** t\_product.产品ID = t\_odetails.产品ID

**INNER** JOIN @canceledOrderIds c **ON** t\_odetails.订单ID = c.order\_id

**END**;

**END**

④ 订单额外增加属性客户ID，为了前端通过客户ID快速寻找信息，在前端插入时候添加客户ID，通过excel导入时候使用级联查询，通过插入订单的联系人姓名信息，在客户表中搜索相同的联系人姓名，将其ID插入对应行。

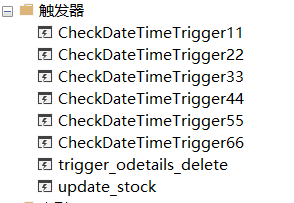


图3-2-5：t\_order表格触发器

1. 订单明细上有一个自动减少对应商品的库存量的触发器，当确认了一个订单之后，自动将订单明细中所有商品以及对应的数量在产品库存量中删除。

**CREATE** **TRIGGER** [dbo].[trigger\_product\_kcl]

**ON**  [dbo].[t\_odetails]

**AFTER** **INSERT**

**AS**

**BEGIN**

**UPDATE** t\_product **SET** 库存量 = 库存量 - i.数量

**FROM** t\_product

**INNER** JOIN inserted i **ON** t\_product.产品ID = i.产品ID

**END**

* + 1. 设置约束

1. 电话&传真类约束，要求区号加电话号码，可以为空

([电话] like '(\_\_\_)%' OR [电话] like '(\_\_\_\_)%' OR [电话] IS NULL)

([传真] like '(\_\_\_)%' OR [传真] like '(\_\_\_\_)%' OR [传真] IS NULL)

1. 用户id约束，要求为长度为5的字符串，

(len([客户ID])=(5))

1. 其他约束：产品id，雇员id，供货商id，订单id，下单id，类别id在增加时候设置自增，主要增加在操作时候实现，前端设置每次新增时查询当前id最大值，并加一作为新id。Excel导入数据库时。
2. excel订单表中运货费类别为datetime，无法正确插入，表中默认插入空值。设置默认值为10
   * 1. 测试数据的导入问题处理
3. excel数据导入过程及出现问题
4. Excel数据导入过程：

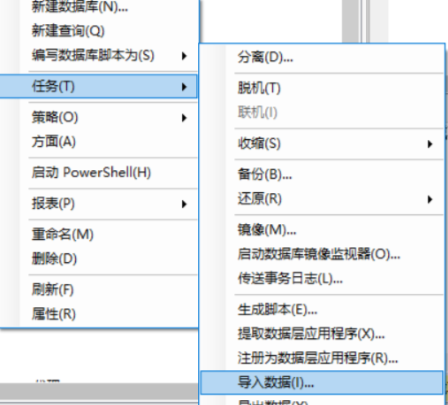
① 在 SQL Server Management Studio 中，右键单击目标数据库，并选择“任务” -> “导入数据”；

图3-2-6：选择任务、导入数据

②在“选择数据源”页面上，选择“Microsoft Excel”作为数据源，并指定 Excel 文件的路径；

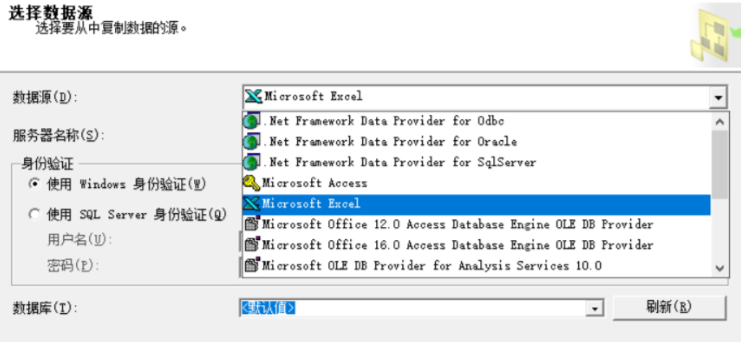


图3-2-7：选择数据源

③在“选择目标”页面上，选择要将数据导入的 SQL Server 数据库；

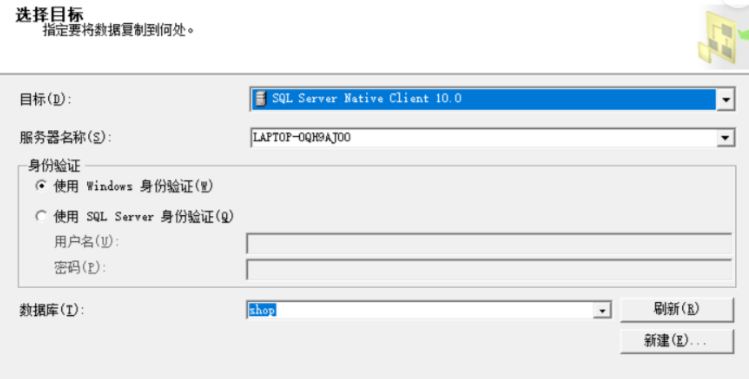


图3-2-8：选择导入目标

④在“映射列”页面上，将 Excel 文件中的列映射到 SQL Server 表中的列；

图3-2-9：编辑映射

⑤点击“完成”按钮，即可开始导入 Excel 数据。

1. 出现问题：

在向数据库中导入Excel表格时，出现了如下问题，具体可以分为三点：

① Excel表格数据格式和数据库中间的表格式冲突导入

② 图片无法正确处理；

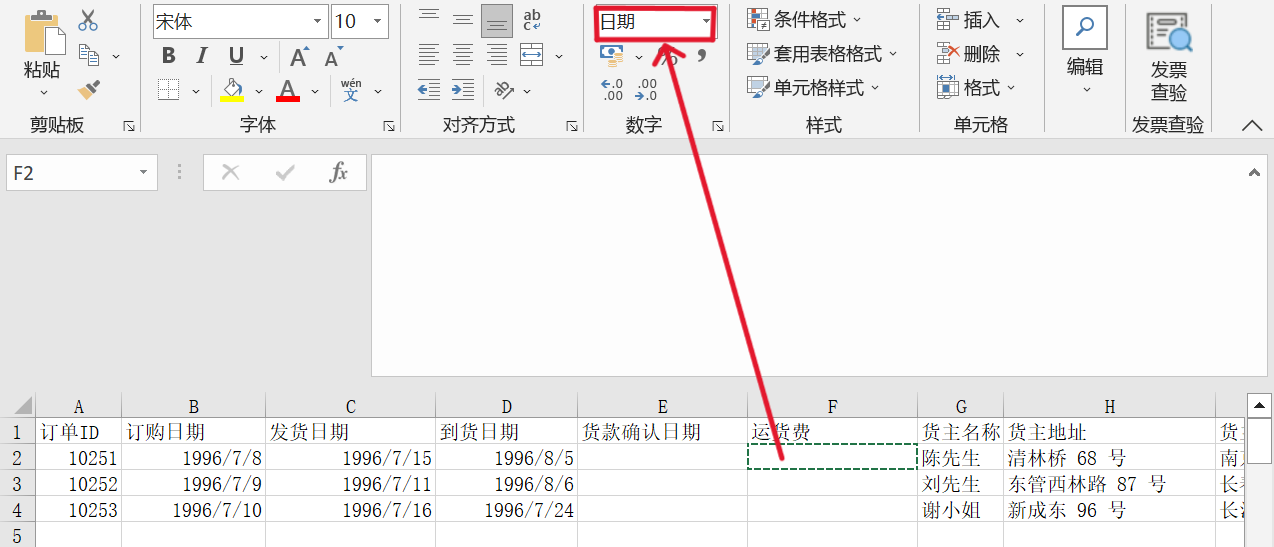
③ Excel 表格中有若干格式完全错乱的数据，如图：

图3-2-10：数据格式错乱

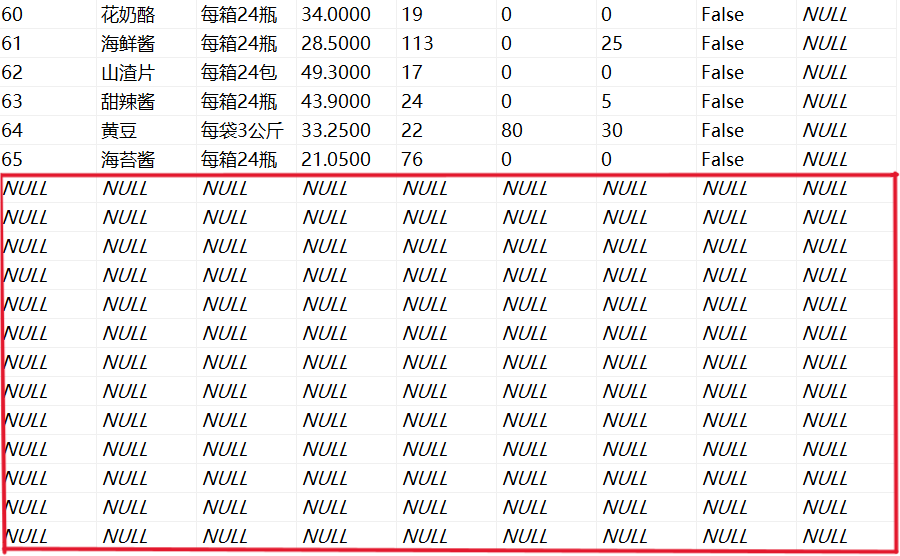
④ Excel 中有大量空行数据被导入，如图：

图3-2-11：大量空行数据被导入

1. Excel数据导入问题解决

①临时表法：

通过使用一个基本属性和目标表完全相同的表，先使用excel向临时表导入数据，方便对其中数据正确性进行判断，通过对其加触发器和限制以及对目标表增加同样的触发器和限制保证数据插入目标表中的一定是正确数据，执行插入语句后临时表中数据会被delete，减少数据库的冗余性。

对于原数据表格中数据类型错乱问题，可以在临时表中建立异常处理列，先将该表格这列所有属性导入异常处理列，之后再通过SQL语句将临时表（i\_table）的数据正常导入本表(t\_table)。

通过建立临时表并在表中设置触发器，在Excel数据导入时，临时表法的有效性远远强于只使用触发器的方法。Excel表格数据导入的流程较为繁琐，若出现错误，触发器会使得所有数据导入失败。若Excel表格有非常多的错误，只使用触发器查找错误点并修改数据效率低下，使用临时表结合触发器，可以减少Excel导入操作次数（一般就1次），并且解决触发器解决不了的问题（Excel表格中的数据类型错乱）。在导入目标表的时候，会根据目标表和临时表的触发器对导入数据正确性进行判断，最后插入目标表的数据一定是正确的，如果错误可以直接在临时表中根据触发器提示修改，速度快，判错准确，效率高。



图3-2-12：在临时表中设置异常处理列

② SQL语句合理解决大量空行问题

利用SQL语句，对临时表中的数据项进行筛选、导入本表中。在完成数据导入后，输出临时表，减少冗余数据。每一次从临时表导入目标表sql语句可以自行判断同时使用还是单独使用，判错能力强，针对触发器判错可以及时准确修改。具体代码如下（以订单表为例）：

**INSERT** **INTO** t\_order

**SELECT** 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,货地址,货主城市,货主地区,货主邮政编码,货主国家,折扣id,取消

**FROM** i\_order

**WHERE** i\_order.订单ID **IS** NOT NULL

**delete** **from** i\_order

1. 图片处理问题解决
2. 题目所给图片处理

①导入图片进数据库

原excel表中图片16进制数据存在冗余信息，利用java代码对其进行预处理后转换成ByteArray类型再插入到数据库image类型中，原始图片为.bmp格式，转换后输出的ByteArray可以存在数据库中

如下，先将excel中图片16进制数据存放在电脑上一个.txt文件中，然后对.txt文件预处理，将前面冗余信息截断删除再转换：

**public** **class** dts {

**public** **static** **void** main(String[] args) {

        String filePath = "C:\\Users\\rcw25\\Desktop\\picture\\final\_class\\444.txt";

**int** charactersToRemove = 158;

        String connectionString = "jdbc:sqlserver://localhost:1433;databaseName=csharpdatabase;user=sa;password=123456";

        String tableName = "t\_class";

        String columnName = "图片";

        //修改id

**int** id=2;

**try** {

            // 读取文本文件中的16进制数据

            String hexData = readHexDataFromFile(filePath);

            // 删除前面指定数量的16进制字符

            String trimmedHexData = trimHexData(hexData, charactersToRemove);

            // 保存修改后的16进制数据到原始文件中

            saveHexDataToFile(trimmedHexData, filePath);

            // 将修改后的16进制数据转换为字节数组

**byte**[] byteArray = hexToByteArray(trimmedHexData);

            // 将字节数组保存到SQL Server的image类型字段中

            saveByteArrayToSQLServer(byteArray, connectionString, tableName, columnName,id);

            System.out.println("数据处理完成！");

        } **catch** (IOException | SQLException e) {

            e.printStackTrace();

        }

    }

    // 读取文本文件中的16进制数据

**private** **static** String readHexDataFromFile(String filePath) **throws** IOException {

        StringBuilder hexData = **new** StringBuilder();

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(filePath))) {

            String line;

**while** ((line = reader.readLine()) != **null**) {

                hexData.append(line.trim());

            }

        }

**return** hexData.toString();

    }

    // 删除前面指定数量的16进制字符

**private** **static** String trimHexData(String hexData, **int** charactersToRemove) {

**return** hexData.substring(charactersToRemove);

    }

    // 保存修改后的16进制数据到原始文件中

**private** **static** **void** saveHexDataToFile(String hexData, String filePath) **throws** IOException {

**try** (FileWriter writer = **new** FileWriter(filePath, StandardCharsets.UTF\_8)) {

            writer.write(hexData);

        }

    }

    // 将16进制字符串转换为字节数组

**private** **static** **byte**[] hexToByteArray(String hexData) {

**int** len = hexData.length() / 2;

**byte**[] byteArray = **new** **byte**[len];

**for** (**int** i = 0; i < len; i++) {

            String byteStr = hexData.substring(2 \* i, 2 \* (i + 1));

            byteArray[i] = (**byte**) Integer.parseInt(byteStr, 16);

        }

**return** byteArray;

    }

    // 将字节数组保存到SQL Server的image类型字段中

**private** **static** **void** saveByteArrayToSQLServer(**byte**[] byteArray, String connectionString, String tableName, String columnName,**int** id) **throws** SQLException {

**try** (Connection connection = DriverManager.getConnection(connectionString);

             PreparedStatement statement = connection.prepareStatement("UPDATE " + tableName + " SET " + columnName + " = ? where 类ID = ?")) {

            statement.setBytes(1, byteArray);

            statement.setInt(2, id);

            statement.executeUpdate();

        }

    }

② 导出图片到本地和前端

从数据库image类型中取出原始二进制格式图片，将其先转换成16进制，再从16进制中恢复bmp图像，当前前端展示图像需要png类型，所以再转换输出一次即可。

**public** **static** **void** db\_to\_png(String outputFilePath,**int** cid) {

    String connectionString = "jdbc:sqlserver://localhost:1433;databaseName=csharpdatabase;user=sa;password=123456";

    String tableName = "t\_class";

    String columnName = "图片";

    String ssql="SELECT " + columnName + " FROM " + tableName+" wher 类别ID = "+cid;

**try** {

        // 从SQL Server中获取image类型数据

**byte**[] imageData = retrieveImageDataFromSQLServer(connectionString, tableName, columnName,ssql);

        // 将字节数组转换为16进制字符串

        String hexData = byteArrayToHex(imageData);

        // 从16进制字符串中恢复BMP图像

        BufferedImage image = restoreBMPImageFromHex(hexData);

        // 将BMP图像转换为PNG并保存到计算机上

        savePNGImage(image, outputFilePath);

        System.out.println("图像已保存到：" + outputFilePath);

    } **catch** (SQLException | IOException e) {

        e.printStackTrace();

    }

}

1. 本地图片处理

①商品图片

商品在excel中没有给图像，所以自己寻找图像存放于数据库，使用的jpg或者png图像先转换成16进制数据再转换成ByteArray类型存放在数据库：

**public** **static** **void** db\_to\_png(String outputFilePath,**int** cid) {

    String connectionString = "jdbc:sqlserver://localhost:1433;databaseName=csharpdatabase;user=sa;password=123456";

    String tableName = "t\_class";

    String columnName = "图片";

    String ssql="SELECT " + columnName + " FROM " + tableName+" where 类别ID = "+cid;

**try** {

        // 从SQL Server中获取image类型数据

**byte**[] imageData = retrieveImageDataFromSQLServer(connectionString, tableName, columnName,ssql);

        // 将字节数组转换为16进制字符串

        String hexData = byteArrayToHex(imageData);

        // 从16进制字符串中恢复BMP图像

        BufferedImage image = restoreBMPImageFromHex(hexData);

        // 将BMP图像转换为PNG并保存到计算机上

        savePNGImage(image, outputFilePath);

        System.out.println("图像已保存到：" + outputFilePath);

    } **catch** (SQLException | IOException e) {

        e.printStackTrace();

    }

}

②数据库中存放图片在本地计算机上的地址

将计算机上图像绝对路径存放在String字符串中，然后更新到数据库中，取出时获取到绝对路径，前端界面可以直接提取。

* 1. JDBC与SQL Server的连接与交互
     1. 连接步骤

为了更好的连接数据库和pyside的前端界面，本系统使用JDBC（Java Database Connectivity）来编写api。一般有以下步骤：

1. 导入JDBC驱动程序（JAR包）：JDBC驱动程序是连接Java应用程序与数据库之间的桥梁。需要将JDBC驱动程序的JAR包导入到Java项目中。
2. 加载JDBC驱动程序：通过Class.forName()方法加载JDBC驱动程序。
3. 建立数据库连接：使用DriverManager.getConnection()方法

建立与数据库的连接，其中需要指定数据库的URL、用户名和密码等信息。

1. 创建Statement对象：使用Connection.createStatement()方法创

建一个Statement对象，用于执行SQL语句。

1. 执行SQL语句：使用Statement对象的execute()、

executeQuery()或executeUpdate()方法执行SQL语句，其中execute()方法用于执行任意SQL语句，executeQuery()方法用于执行查询语句，executeUpdate()方法用于执行更新语句。

1. 处理查询结果：如果执行的是查询语句，可以通过ResultSet对象 获取查询结果。
2. 关闭连接：在完成数据库操作后，需要关闭数据库连接，释放资源。
   * 1. 功能代码展示（api代码介绍）

利用JDBC完成数据库的连接和操作，以下是功能的一些代码实现：

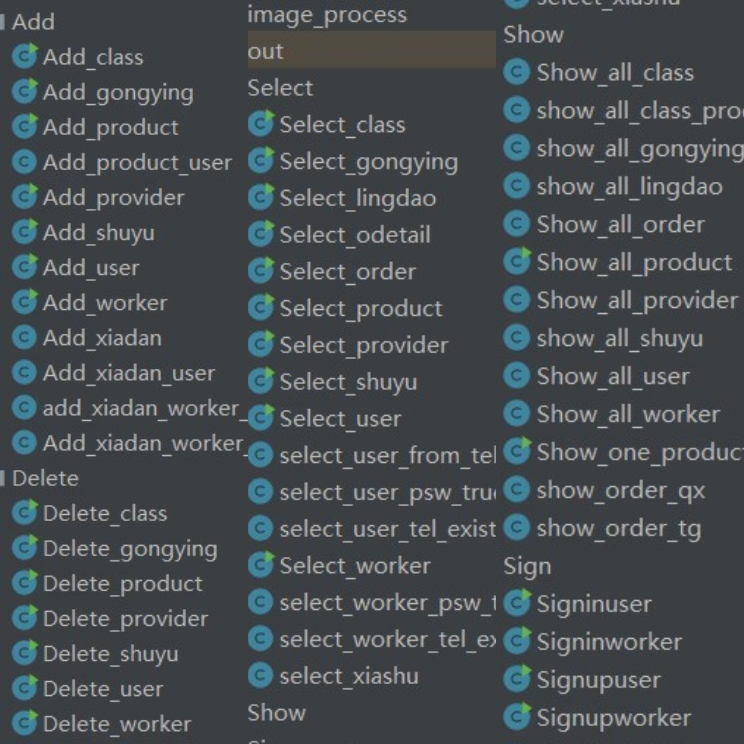


图3-3-1：功能代码总览

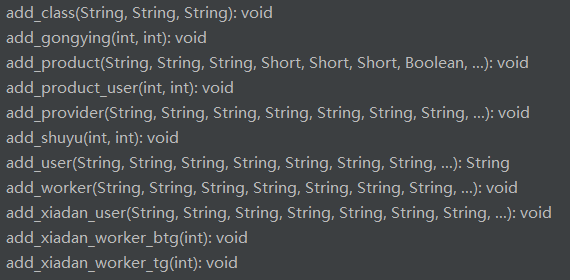
1. Add函数大类：

图3-3-2：add函数总览

● add\_class函数：

函数通过类别名称、类别说明、图片三个参数变量，自动生成类别ID，将类别ID、名称、说明、图片四个字段值增加到类别表中，实现新增类别功能；

● add\_gongying:

函数通过传入产品ID、供应商ID，在新建商品时，将产品ID和供应商ID插入供应表中，实现新增供应关系功能；

● add\_product:

函数通过传入产品ID、产品名称、单位数量、单价、库存量、订购量、再订购量、中止，在增加新商品时，同时增加供应商和类别、赋予新商品产品ID，实现新增商品功能；

● add\_product\_user:

函数通过传入产品ID和数量，给购物车（下单明细表）中添加商品和数量；

● add\_provider:

函数通过供应商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，自动分配给供应商ID，将供应商信息插入供应商表格中，实现新增供应商功能；

● add\_shuyu:

函数通过传入产品ID和类别ID，将新建商品的信息插入属于表（t\_shuyu），实现新增属于表元组功能；

● add\_user:

函数通过传入客户输入的所有信息，自动分配客户ID，将客户信息插入客户表中，实现用户注册功能；

● add\_worker:

函数通过雇员输入的所有信息，自动给新雇员分配ID，将雇员所有信息插入雇员表中，实现雇员注册功能

● add\_xiadan\_user:

函数通过传入客户ID、客户电话、下单时间、货主名称等参数变量，自动生成下单ID，将新建的购物车导入到下单表中，实现新增下单表功能。

● add\_xiadan\_worker\_btg和add\_xiadan\_worker\_btg:

函数通过传入下单ID,展示下单ID所对应的所有信息，雇员查看该订单、并且进行通过与否的判断。

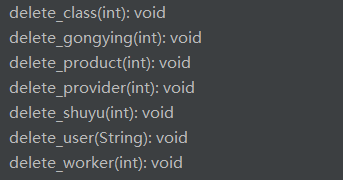
1. Delete函数大类：

图3-3-3：Delete函数总览

● delete\_class函数

函数通过传入类别ID参数变量，实现删除类别表的一个类别的功能。

● delete\_gongying函数

函数通过传入供应商ID参数变量，实现删除供应关系表格的一个供应关系的功能。

● delete\_product函数

函数通过传入产品ID参数变量，实现删除产品表的一个产品的功能。

● delete\_provider函数

函数通过传入供应商ID参数变量，实现删除供应商表的一个供应商的功能。

● delete\_shuyu函数

函数通过传入产品ID参数变量，实现删除属于关系表的一个属于关系的功能。

● delete\_user函数

函数通过传入客户ID参数变量，实现删除客户表的一个客户的功能。

● delete\_worker函数

函数通过传入雇员ID参数变量，实现删除雇员表的一个雇员的功能。

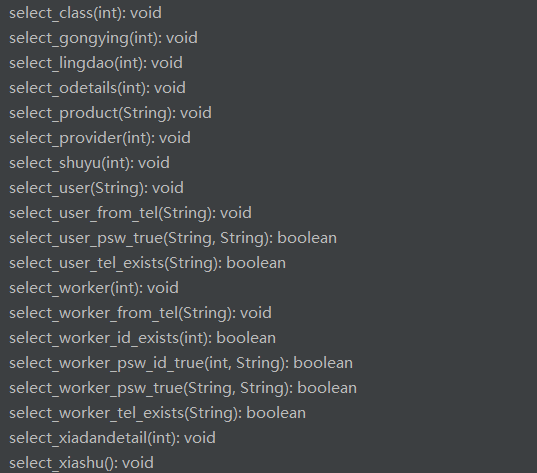
1. Select函数大类：

图3-3-4：select函数总览

● select\_class函数

函数通过传入类别ID参数变量，实现读出所有的类别信息的功能。

● select \_gongying函数

函数通过传入产品ID参数变量，实现查找商品的供应信息的功能。

● select \_lingdao函数

函数通过传入雇员ID参数变量，实现通过雇员查找其领导的功能。

● select \_odetails函数

函数通过传入订单ID参数变量，实现展示订单明细的功能。

● select \_product函数

函数通过传入商品名参数变量，实现查找商品的功能。

● select \_provider函数

函数通过传入供应商ID参数变量，实现查找供应商表的一个供应商的功能。

● select \_shuyu函数

函数通过传入产品ID参数变量，实现商品查找属于哪一个类别的功能。

● select \_user函数

函数通过传入客户ID参数变量，实现读出用户信息的功能。

● select \_user\_from\_tel函数

函数通过传入客户电话号码参数变量，实现通过用户电话号码查找用户的功能。

● select \_user\_psw\_true函数

函数通过传入电话参数变量，实现通过电话判断用户密码是否正确的功能。

● select \_user\_tel\_exists函数

函数通过传入客户电话号码参数变量，实现通过电话判断用户是否存在的功能。

● select \_worker\_from\_tel函数

函数通过传入雇员电话号码参数变量，实现通过雇员电话读出雇员所有信息的功能。

● select \_worker\_id\_exists函数

函数通过传入雇员ID参数变量，实现删除雇员表的一个雇员的功能。

● select \_worker\_psw\_id\_true函数

函数通过传入雇员ID参数变量，实现通过电话判断雇员密码是否正确的功能。

● select \_worker\_psw\_true函数

函数通过传入雇员电话号码参数变量，实现通过电话判断雇员密码是否正确的功能。

● select \_worker\_tel\_exists函数

函数通过传入雇员电话号码参数变量，实现通过电话判断用户是否存在的功能。

● select \_xiadan\_detail函数

函数通过传入下单ID参数变量，实现查看下单明细表的功能。

● select \_xiashu函数

函数可以实现雇员查找下属的功能。

1. Show函数大类：

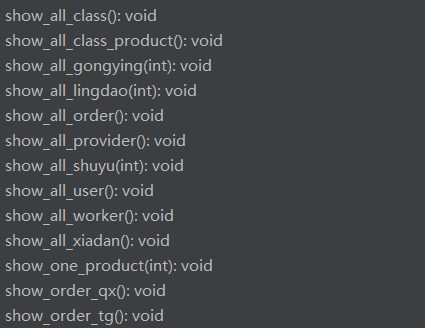


图3-3-5：show函数总览

● show\_all\_class函数

系统调用此函数，可以实现在前端界面主页展示所有类别的功能。

● show\_all\_class\_product函数

函数通过传入类别ID参数变量，实现展示所有商品的功能。

● show\_all\_gongying函数

函数通过传入供应商ID参数变量，实现展示所有供应关系的功能。

● show\_all\_lingdao函数

函数通过传入雇员ID参数变量，实现展示所有领导关系的功能。

● show\_all\_order函数

系统调用此函数，可以实现在前端界面主页展示所有历史订单的功能。

● show\_all\_provider函数

系统调用此函数，可以实现在前端界面主页展示所有供应商的功能。

● show\_all\_shuyu函数

函数通过传入类别ID参数变量，实现展示所有类别从属关系的功能。

● show\_all \_user函数

系统调用此函数，可以实现在前端界面主页展示所有用户信息的功能。

● show\_all\_worker函数

系统调用此函数，可以实现在前端界面主页展示所有雇员信息的功能。

● show\_all\_xiadan函数

系统调用此函数，可以实现在前端界面主页展示所有待通过订单信息的功能。

● show\_one\_product函数

函数通过传入产品ID参数变量，实现展示产品ID所对应的商品所有属性的功能。

● show\_order\_qx函数

系统调用此函数，可以实现在前端界面主页用户取消自己的订单的功能。

● show\_order\_tg函数

系统调用此函数，可以实现在前端界面主页展示用户的所有已经通过订单信息的功能。

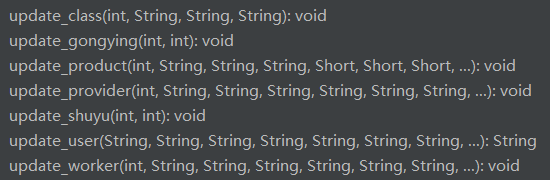
1. Update函数大类：

图3-3-6：update函数总览

● update\_class函数

函数通过传入类别ID、类别名称、说明、图片等参数变量，实现更新类别所有信息的功能。

● update \_gongying函数

函数通过传入产品ID等参数变量，实现更新商品的供应关系的功能。

● update \_product函数

函数通过传入商品名参数变量，实现查找商品的功能。

● update \_provider函数

函数通过传入供应商ID、公司名称等参数变量，实现更新供应商信息的功能。

● update \_shuyu函数

函数通过传入产品ID等参数变量，实现更行产品从属关系的功能。

● update \_user函数

函数通过传入客户ID、公司名称等参数变量，实现更新用户信息的功能。

● update \_worker函数

函数通过传入雇员ID、姓名等参数变量，实现更新雇员信息的功能。

1. 系统测试与应用
   1. 系统测试运行环境

测试系统软件环境：操作系统为Windows 11，数据库管理系统为 MicroSoft SQL Server 2022，开发工具为PyCharm 2023.1, jdk-20,eclipse 2023-03,

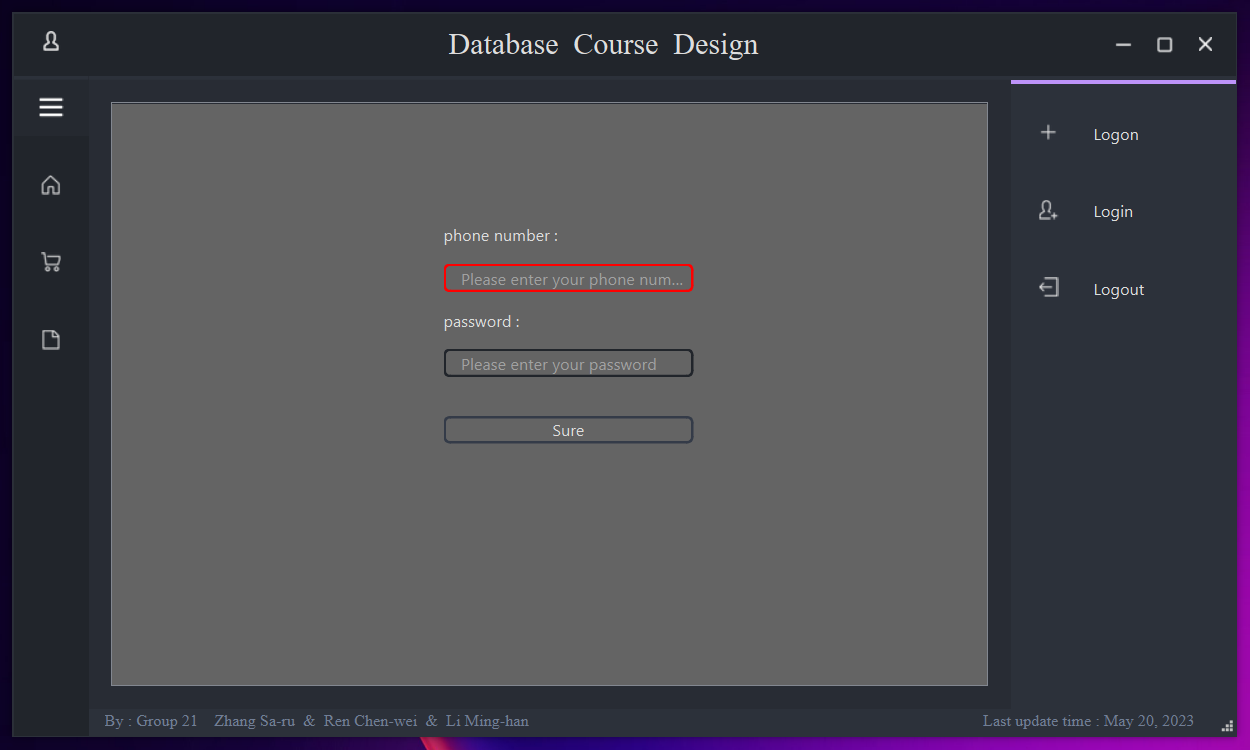
测试系统硬件配置：AMD R7 5800H处理器，16GB内存

* 1. 数据库测试与运行

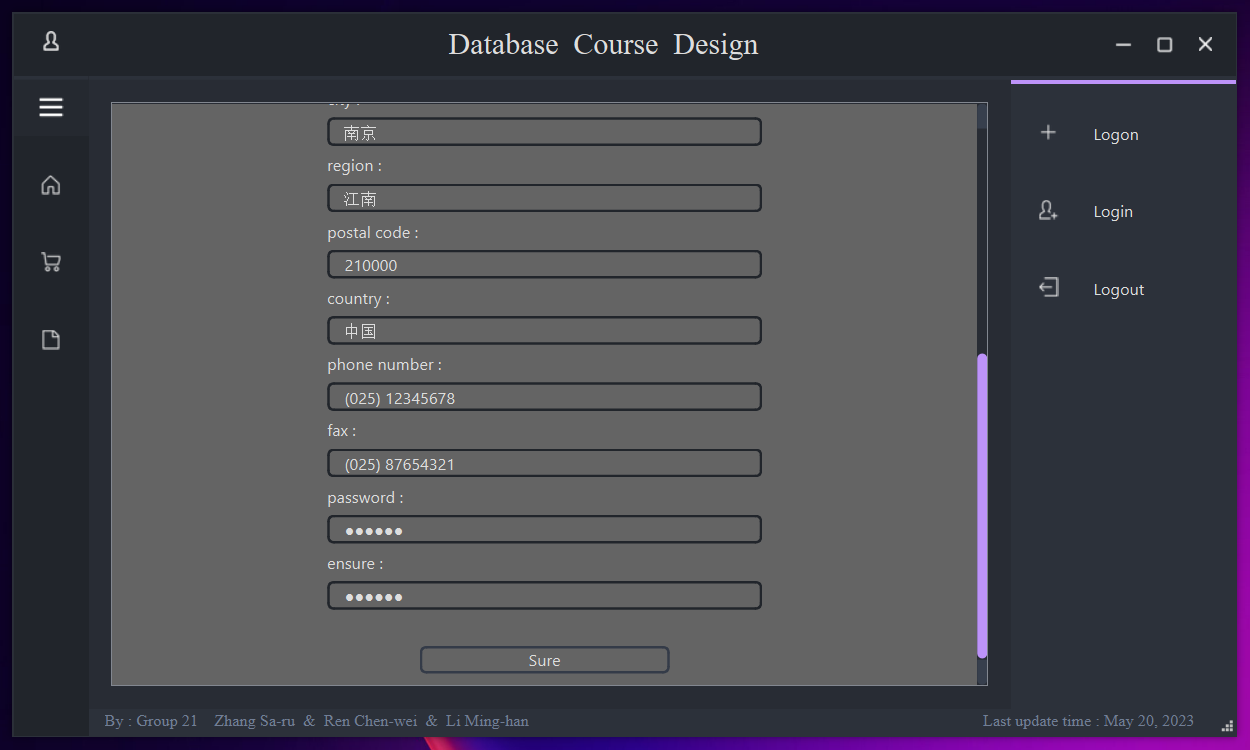
由于用户和雇员具有许多相似的操作，在此为避免冗余，只选取主要的功能展示

* + 1. 用户登录/注册

用户通过电话号码和密码登录，如果输入有误会给予准确简明的错误提示



用户注册时确定所有个人信息，不允许用户随意修改自己的个人信息。



* + 1. 商品分类页面、商品页面

提供可视化的类别选择，点击各个按钮即可进入对应分类的商品界面



用户在商品界面可以通过商品底部的按钮选择数量，并且加入购物车。



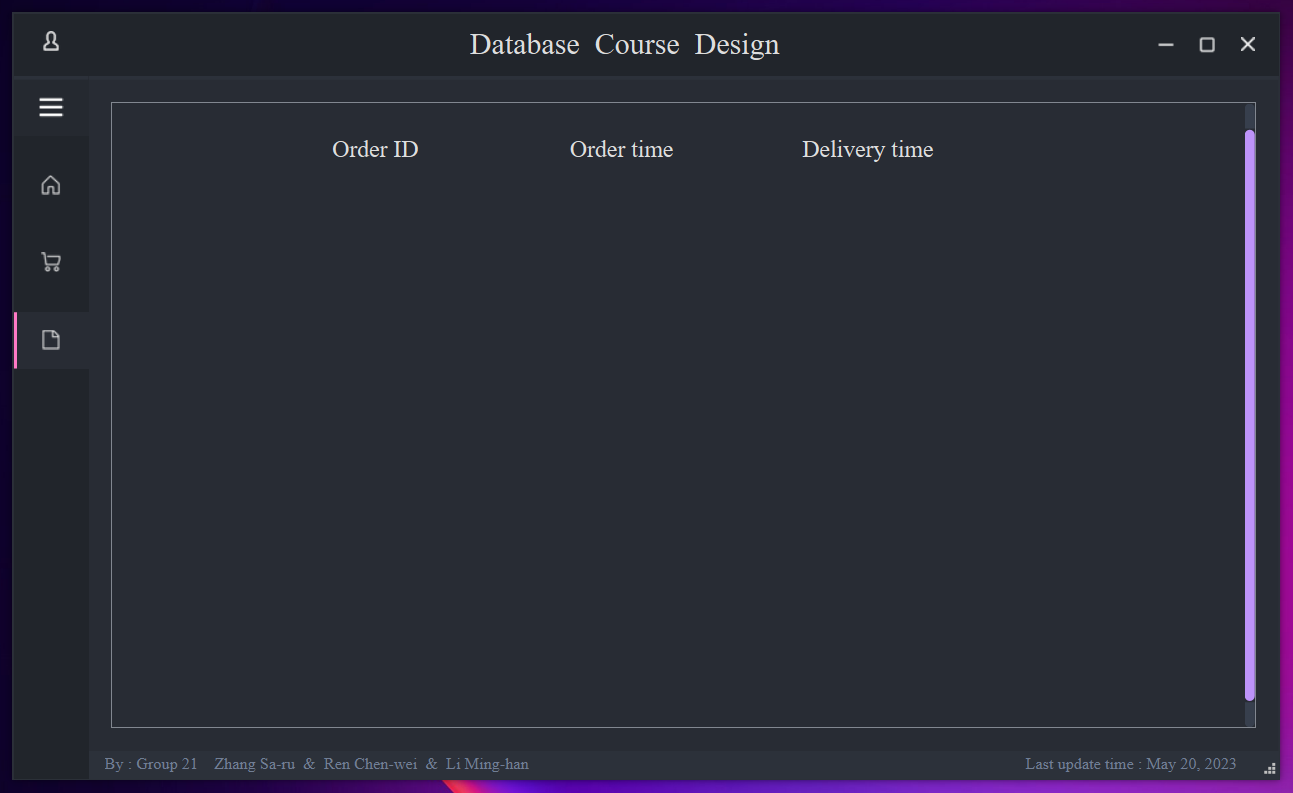
* + 1. 购物车页面

用户可以在购物车中查看当前已购买的商品名称和数量



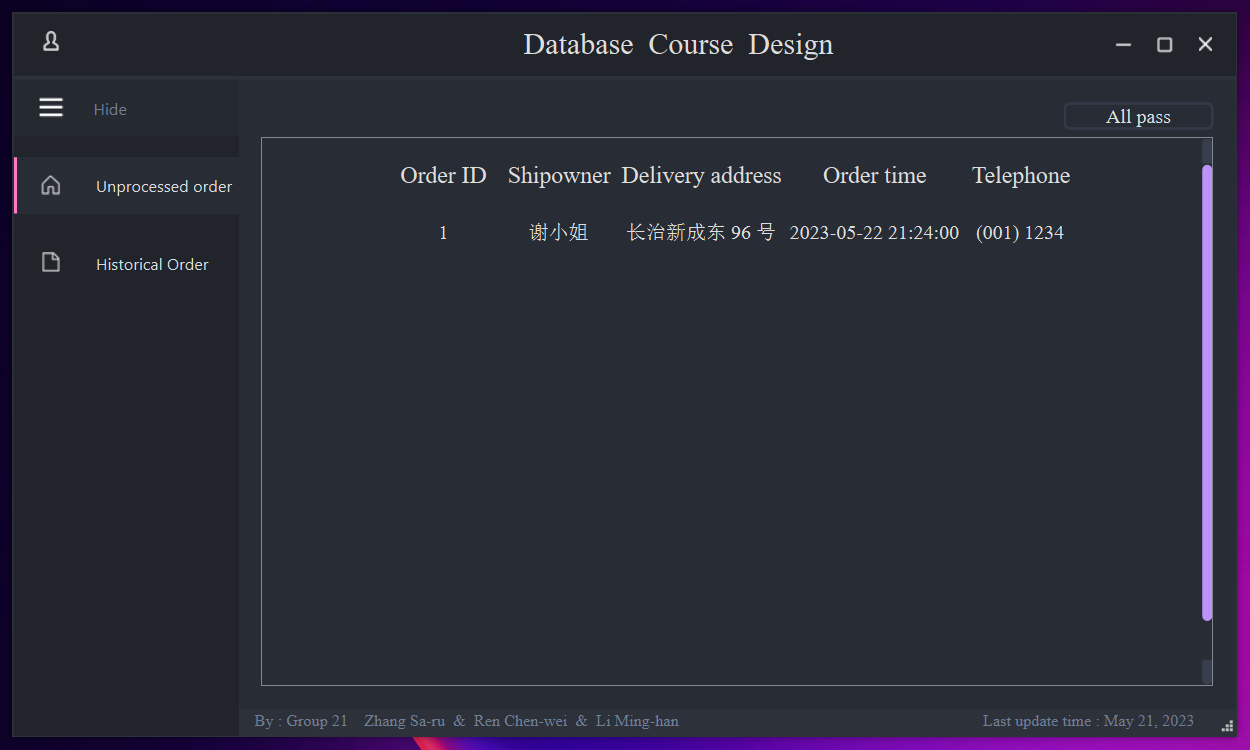
* + 1. 订单详情界面

用户可以查看订单ID、下单时间和发货时间



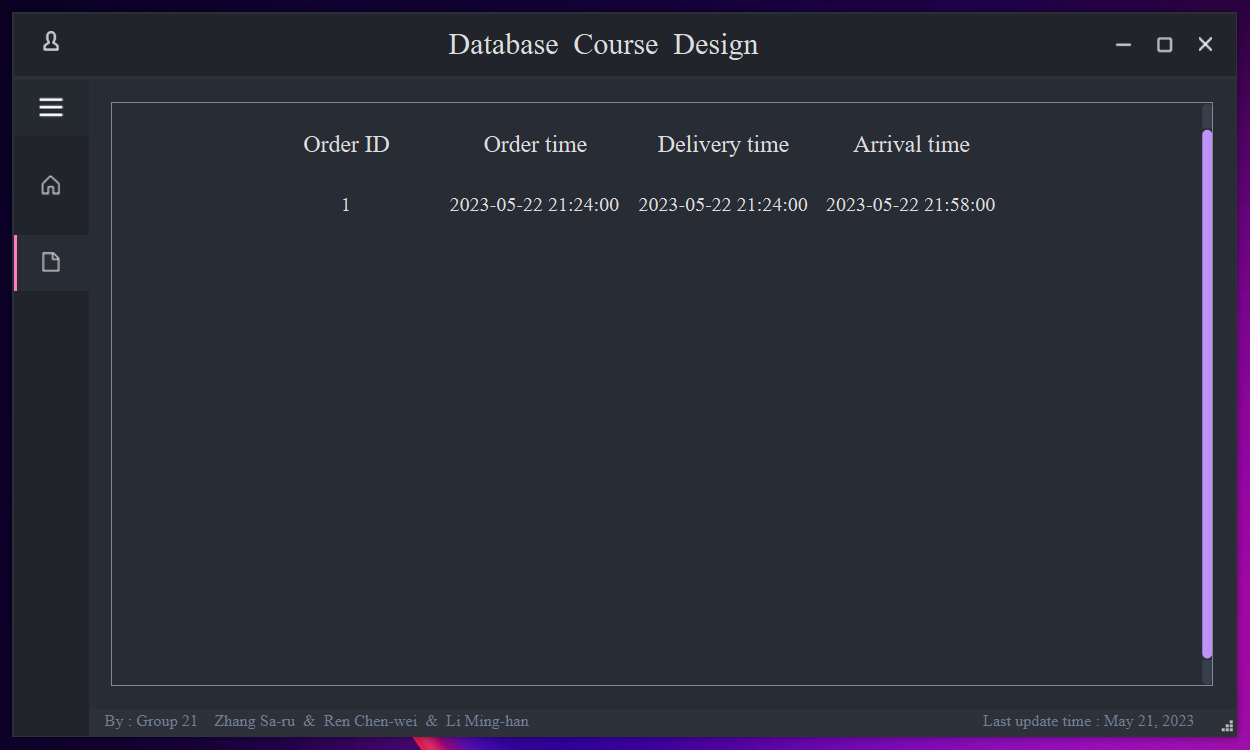
* + 1. 订单处理界面

雇员可以在此处看到所有尚未处理的订单，并且附有一键通过的按钮



* + 1. 历史订单界面

雇员可以在此处看到所有历史订单



1. 总结与思考
   1. 数据库设计思路总结
      1. 总体规划

该数据库设计通过设计关系表和触发器以及约束限制，完成客户下单，雇员通过订单这一购物流程。

数据库底层设计时候大致遵循以下思路：

1. 需要设计的实体以及其属性
2. 实体属性的约束和关系表之间联系
3. 操作过程中的级联控制关系
4. 数据导入的正确性判断与异常处理
5. 数据操作的正确性判断与出错处理

图5-1：初期设计思路

* + 1. 增加额外表格

根据实际需求，除了基础的几个基础实体，增加了以下几个实体用于满足实际需求：

1. 领导表：存储雇员和其领导对应关系；
2. 供应表：存储产品和供应商对应关系；
3. 属于表：存储产品和类别的对应关系；
4. 下单表和下单明细表：分别一一对应订单表和订单明细表，存储购物车中间数据，待购物车确认后插入订单表和订单明细表。
   * 1. 增加拓展属性

为了使得数据库功能更加强大，增加了若干扩展属性用于简化操作提高效率：

1. 订单表设计时考虑前端查询，增加客户ID方便查询；
2. 订单表增加日期错误1，2，3用于存储插入的发货日期，到货日期，货款确认日期的正确与否；
3. 下单表除了订单表属性外，增加雇员ID属性，便于接口级联查询；
4. 下单表增加客户电话属性，用户判断下单和通过不是同一个人，将电话作为“桥梁”连接客户和职工，简化查询；
5. 雇员表和客户表增加密码属性用于登录。
   * 1. 增加级联处理

为了保证外键为空或者对应主键值，所有外键设计时候全部对修改删除做级联处理。

* + 1. 处理Excel导入

为处理Excel中的数据导入SQL Server中出现的各种问题，我们通过建立临时表和结合select语句来完成数据的正确导入和处理冗余。

* 1. 不足与展望
     1. 不足

目前，本系统的核心功能都基本实现，但是系统的应用上来说还存在一些不足：

* 数据库安全性设计方面未考虑设立维护计划，以防止数据库发生故障hou 能够及时对数据进行恢复
* 用户在注册账户时，系统给用户分配的ID是随机的5个英文字符，不具有很好的辨识度和记忆性
  + 1. 展望

本系统只是我们对于商品购买系统的设计与实现的一次尝试，我们只做出来一个购买系统的雏形，希望在以后能在已有的基础上优化并拓展新的功能，如：客户和供应商进行聊天的功能，以及对数据库的维护计划进行更好的完善等。让该商品购买系统更好的用于现实场景。

1. 总结：

在数据库课设中，我学会了如何使用数据库来管理和处理数据，并且获得了许多宝贵的经验和体会。

俯观数据库课设之所获，固著实辛苦。然亦有所得之欢欣。细思基本理论，追推本源语言，开阔思维界限。修改重构难免不歇,更迭无常需慎思。夜未睡眠思路纷,尚犹有未解之谜。终于圆满课题要求,兢兢业业竟之成。指间之痕迹犹存,心有余悸难自辄。原来体察不尽理论之精髓,实践方知功夫之不易。书中线索难全盈我心,惟有努力求索不废止。虽然仓促功夫尚浅,而苦尝始终有成。唯愿借此契机,不断提炼技艺，期许再接再厉，养就高超本领。若有此盼,终归该为此学有所得之快慰。

最重要的是，通过数据库课设，我培养了良好的数据管理和分析能力。我学会了如何从实际需求出发，进行数据库的需求分析和设计，合理地组织和存储数据，以支持各种业务操作和决策。这使我对数据的重要性有了更深刻的认识，并且在实际项目中具备了更加实用的技能。