

# 南京信息工程大学

## 《数据库系统原理》课程设计



题    目    小型超市商品管理系统设计与实现

---

学生姓名    鲁哲豪        王子荀

学    号    202083290400 202083290153

学    院    计算机与软件学院

专    业    计算机科学与技术

指导教师    马瑞

二〇二二 年 六 月 二十 日

# 目 录

<b>1 引言</b>	<b>1</b>
1.1 课题背景和意义	1
1.2 课题内容	1
<b>2 系统需求分析及相关技术介绍</b>	<b>1</b>
2.1 功能需求分析	1
2.2 可行性分析	2
2.3 系统运行环境	2
2.4 相关技术介绍	2
<b>3 系统总体设计</b>	<b>3</b>
3.1 系统功能结构设计	3
3.2 系统功能流程设计	3
3.2.1 主程序流程设计	3
3.2.2 查看数据流程设计	4
3.2.3 账号注册流程设计	4
3.2.4 账号删除流程设计	4
3.2.5 账号密码修改流程设计	4
3.2.6 增加信息流程设计	5
3.2.7 删除信息流程设计	5
3.2.8 修改信息流程设计	6
3.3 数据库设计	6
3.3.1 概念结构设计	6
3.3.2 逻辑结构设计	7
<b>4 系统详细设计</b>	<b>9</b>
4.1 账号模块详细设计	9
4.2 主菜单模块详细设计	10
4.3 查看模块详细设计	10
4.4 增加模块详细设计	10
4.5 删除模块详细设计	10
4.6 修改模块详细设计	10

<b>5 系统实现</b>	<b>11</b>
5.1 账号模块实现	11
5.2 主菜单模块实现	12
5.3 查看模块实现	14
5.4 增加模块实现	15
5.5 删除模块实现	16
5.6 修改模块实现	17
<b>6 总结</b>	<b>19</b>

# 小型超市商品管理系统设计与实现

鲁哲豪 王子荀

南京信息工程大学计算机与软件学院，江苏 南京 210044

## 1 引言

### 1.1 课题背景和意义

在计算机技术高速发展的今天，数据库技术也已经日趋完善。随着现代化管理理念的产生，计算机管理信息系统已经被广泛的应用在各个领域。使用计算机管理信息系统，可以减少人力的投入，加大信息的处理效率并且可以降低管理的难度。

商品管理系统可以对商品的所有信息进行统一的管理，这样就可以减少管理人员的工作时间，加大工作效率。商品销售管理系统不但可以对项目信息进行存储，还可以对项目信息进行修改、删除、查询等操作，计算机管理信息系统的保密性要远远高于手工管理，通过创建拥有有限权限的账户，可以避免信息被错误的修改，保障数据的安全性。

### 1.2 课题内容

本文详细介绍了关于小型超市商品管理系统设计与实现的相关概念，用到的开发技术的简要介绍，针对该系统的需求分析，系统总体结构设计方案，以及数据库结构的设计与实现和数据库应用程序的开发。

## 2 系统需求分析及相关技术介绍

### 2.1 功能需求分析

随着经济的高速发展以及互联网支付的普及，超市的信息化工作也在如火如荼地进行，小型超市由于规模较小，在运营开始就采用原始的手工操作方式进行商品的记录、管理，这种操作方式已经逐渐无法跟上互联网的时代浪潮，商品管理系统为超市对商品的管理提供了方便，提高了管理效率。

小型超市管理系统是为了方便管理员管理商品而设立的，是典型的信息管理系统(MIS)，本系统主要完成对小型的管理，包括进货管理，商品订单汇总，库存管理和客户管理四个方面。系统可以完成对各类信息的浏览、查询、增加、更改和删除。

该系统特点：

1. 通用性：适合小型超市对商品信息进行管理。
2. 界面友好：提供给管理员良好的操作界面，简单直观，方便操作。
3. 准确性：通过良好的用户界面，可以快速准确的实现信息查询。

## 2.2 可行性分析

技术可行性：本系统采用 MySQL 8 以及 Python3 进行开发。MySQL 能够处理大量数据并保障数据的安全性，其易用性，灵活性和低成本非常适合小型超市。Python 可以良好的支持跨平台、跨架构运行，满足客户不同的运行环境需求。

市场前景：目前多数小型超市仍在采用纸笔记录的方式进行商品管理，效率低下且数据以损坏。低成本的小型超市商品管理系统可以大幅提升商品管理效率，提高超市的库存周转率，降低超市等库存和运营成本。

目标群体：小型超市、社区便利店等

## 2.3 系统运行环境

系统平台：Windows 或 macOS

数据库版本：MySQL 8

开发环境：Python 3.9（或 Python3.10）

引用库：PyMySQL、

cryptography、

wxPython

## 2.4 相关技术介绍

MySQL：MySQL 是一个关系型数据库，使用 SQL 语言进行增删改查操作，目前属于 Oracle 旗下的产品。MySQL 数据库开源免费，能够跨平台，支持分布式，性能也不错，非常适合中小型企业作为数据库，本设计使用 MySQL 8.0 作为数据库。

Python：Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。

PyMySQL：PyMySQL 是在 Python3.x 版本中用于连接 MySQL 服务器的一个库，PyMySQL 遵循 Python 数据库 API v2.0 规范，并包含了 pure-Python MySQL 客户端库。

Python 并不包含 PyMySQL 模块，需要使用命令“pip install PyMySQL”安装。

cryptography：Cryptography 是一个标准 Python 加密库，支持 Python 2.6-2.7, Python 3.3+, and PyPy 2.6+。在这里主要给 PyMySQL 提供加密/解密服务。

Python 并不包含 cryptography 模块，需要使用命令“pip install cryptography”安装。

wxPython：wxPython 是 Python 语言的一套优秀的 GUI 图形库。允许 Python 程序员很方便的创建完整的、功能健全的 GUI 用户界面，它是基于 C++的函数库 wxWidgets 的封装。wxPython 是作为优秀的跨平台 GUI 库 wxWidgets 的 Python 封装和 Python 模块的方式提供给用户的。

Python 并不包含 wxPython 模块，Python3.9 版本需要使用“pip install wxPython”安装 4.1.1 版本，Python3.10 需要使用“pip install wxPython310”安装 4.1.2a2 版本。

### 3 系统总体设计

我们主要针对小型超市的需求进行了分析，总结出员工管理、供应商管理、库存管理和商品订单管理这四大模块。

#### 3.1 系统功能结构设计

系统功能模块划分如图 3-1 所示。系统的核心是库存与订单，订单的增加、删除、修改会直接影响库存。库存中的商品信息与供应商相关联，便于超市进行库存管理。同时系统有完整的账号管理系统，可以根据账号所拥有的权限等级开放对应的功能。

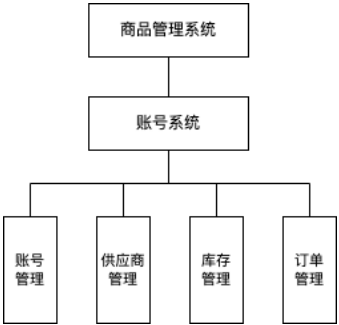


图 3-1 系统功能结构图

#### 3.2 系统功能流程设计

##### 3.2.1 主程序流程设计

主程序主要承担了验证账号的功能，在第一次使用软件的时候初始化数据库以及创建账号。成功验证账号后根据账号权限等级展示对应的功能。

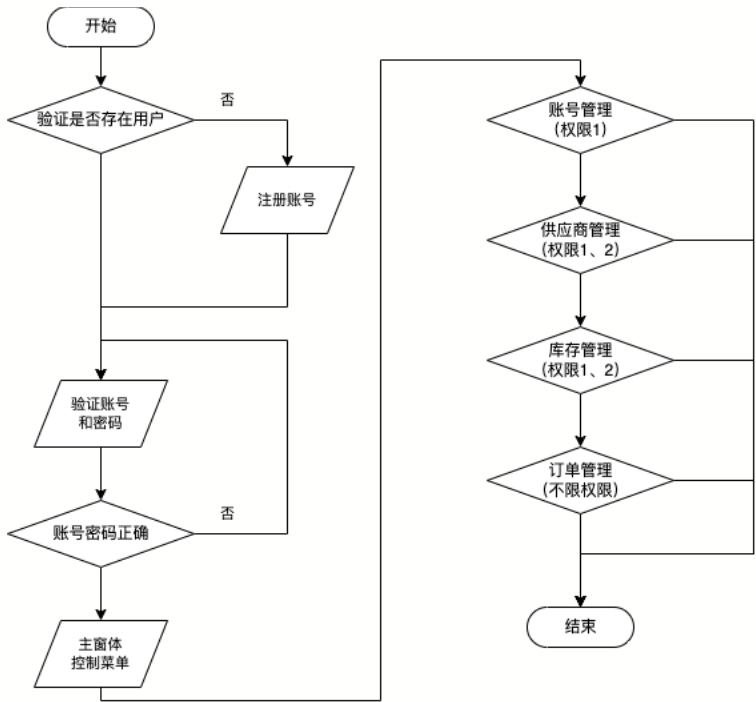


图 3-2 主程序流程图

3.2.2 查看数据流程设计

查看数据模块主要用于读取相应表数据，通过表格控件进行输出，把对应数据直观的呈现给用户。

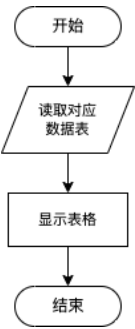


图 3-3 查看数据流程图

3.2.3 账号注册流程设计

账号注册模块主要用于管理员管理账号使用，也用于初次使用程序时的注册初始账号。

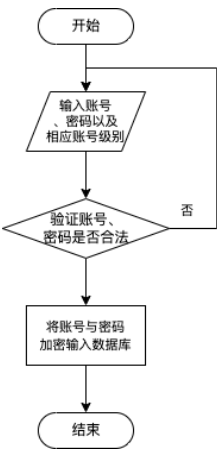


图 3-4 账号注册流程图

3.2.4 账号删除流程设计

账号删除模块主要用于管理员管理账号使用。

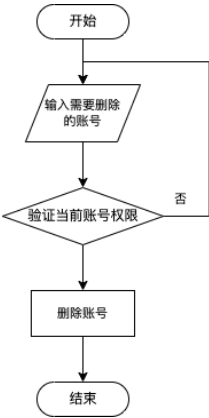


图 3-5 账号删除流程图

3.2.5 账号密码修改流程设计

账号密码修改模块主要用于管理员管理账号使用。

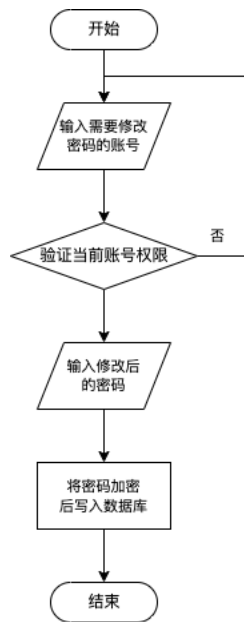


图 3-6 账号密码修改流程图

### 3.2.6 增加信息流程设计

增加信息模块可以用于添加商品信息、供应商信息和订单信息。

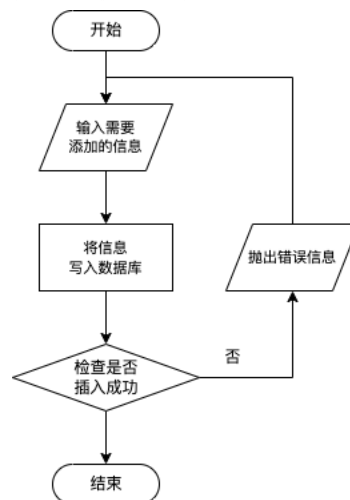


图 3-7 增加信息流程图

### 3.2.7 删除信息流程设计

删除信息模块可以用于删除商品信息、供应商信息和订单信息。



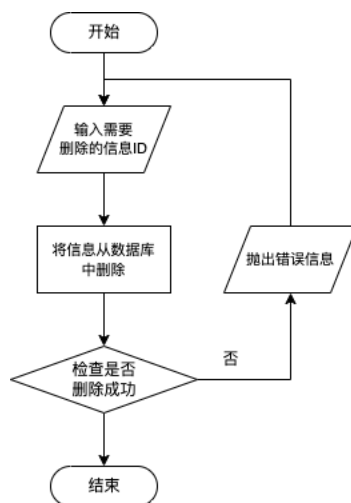


图 3-8 删除信息流程图

### 3.2.8 修改信息流程设计

修改信息模块可以用于修改商品信息、供应商信息和订单信息。

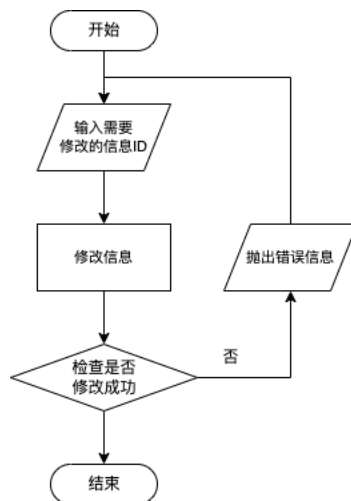


图 3-9 修改信息流程图

## 3.3 数据库设计

### 3.3.1 概念结构设计

#### (1) 供应商信息

供应商信息包含供应商编号、名称、所在地址、电话、联系人、邮件这几个属性。其中，供应商编号是区别供应商的关键信息，其余属性只是便于超市进行进货、联系供应商等需求。供应商实体如图 3-10 所示。

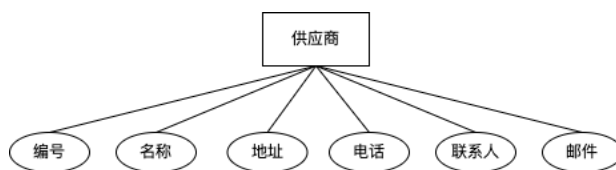


图 3-10 供应商实体属性图

## （2）库存信息

库存信息包含商品编号、名称、产地、价格、库存量、供应商编号这几个属性。其中，商品编号是区别商品的关键信息，供应商编号依赖于供应商信息，库存量会受到订单增删改的影响，其余属性为商品本身的固有属性。库存实体如图 3-11 所示。

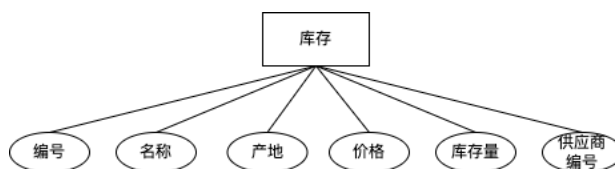


图 3-11 商品实体属性图

## （3）订单信息

订单信息包含订单编号、商品编号、数量、付款方式这几个属性。其中，订单编号是区别订单的关键信息，商品编号依赖于商品信息，数量的增删改会根据商品编号对应修改库存信息中的库存量，付款方式便于超市追溯款项，进行财务管理。订单实体如图 3-12 所示。

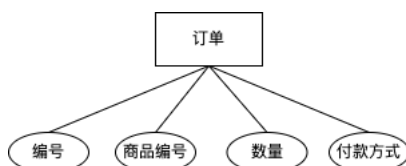


图 3-12 订单实体属性图

## （4）总体 E-R 图

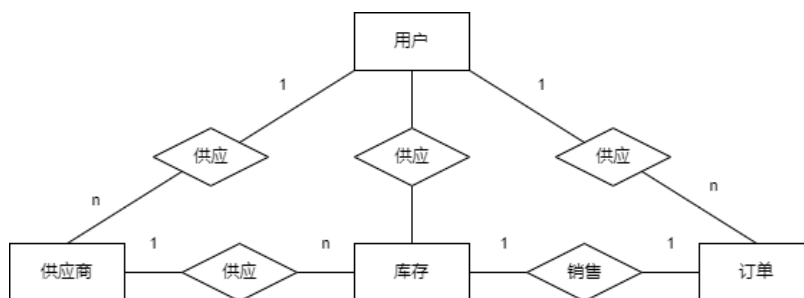


图 3-13 总体 E-R 图

### 3.3.2 逻辑结构设计

#### （1）用户表

用户表包含用户账号密码等数据。其中最重要的用户账号是主键，非空且唯一，在用户名为空的情况下会自动按自增分配账号。其他重要信息均有非空约束限制，添加时间默认系统当前时间。其属性有用户表如表 3-1 所示。

表 3-1 用户表结构说明

列名	数据类型	非空	主键	唯一	默认值	含义
id	INT UNSIGNED	Y	Y	Y	AUTO_INCREMENT	用户账号

name	varchar(255)	Y				用户名
password	varchar(255)	Y				密码（SHA256 加密）
level	INT UNSIGNED	Y				账号等级
add_time	DATETIME				CURRENT_TIMESTAMP	添加时间

## （2）供应商表

供应商表包含供应商编号、名称等数据。其中最重要的供应商编号是主键，非空且唯一，在编号为空的情况下会自动按自增分配。其他重要信息均有非空约束限制。操作用户为当前操作用户，受用户表的外键约束，方便回溯。添加时间默认系统当前时间。供应商表如表 3-2 所示。

表 3-2 供应商表结构说明

列名	数据类型	非空	主键	外键	默认值	含义
id	INT UNSIGNED	Y	Y		AUTO_INCREMENT	供应商编号
name	varchar(255)	Y				供应商名
address	varchar(255)					地址
telephon	varchar(255)	Y				电话
contact_person	varchar(255)	Y				联系人
email	varchar(255)					邮件
userid	INT UNSIGNED			user(id)		操作用户
add_time	DATETIME				CURRENT_TIMESTAMP	添加时间

## （3）库存表

库存表包含商品编号、名称等数据。其中最重要的商品编号是主键，非空且唯一，在编号为空的情况下会自动按自增分配。商品与供应商是一对多的关系，受供应商编号外键约束。其他重要信息均有非空约束限制。操作用户为当前操作用户，用户表的外键，方便回溯。添加时间默认系统当前时间。库存表如表 3-2 所示。

表 3-3 库存表结构说明

列名	数据类型	非空	主键	外键	默认值	含义
id	INT UNSIGNED	Y	Y		AUTO_INCREMENT	商品编号
name	varchar(255)	Y				商品名
place_of_pr	varchar(255)	Y				产地

oduction						
price	FLOAT	Y				价格
num	INT UNSIGNED	Y			0	库存
companyid	INT UNSIGNED			company(id)		供应商编号
userid	INT UNSIGNED			user(id)		操作用户
add_time	DATETIME				CURRENT_TIMESTAMP	添加时间

#### (4) 订单表

订单表包含订单编号、商品编号、数量等数据。其中最重要的订单编号是主键，非空且唯一，在编号为空的情况下会自动按自增分配。订单与商品是一对一的关系，受商品编号外键约束。其他重要信息均有非空约束限制。其中订单的增加、删除和数量的修改都会影响库存量，由触发器完成。操作用户为当前操作用户，用户表的外键，方便回溯。添加时间默认系统当前时间。订单表如表 3-2 所示。

表 3-3 库存表结构说明

列名	数据类型	非空	主键	外键	默认值	含义
id	INT UNSIGNED	Y	Y		AUTO_INCREMENT	订单编号
goodsid	varchar(255)			goods(id)		商品编号
num	varchar(255)	Y				数量
payment_type	FLOAT	Y				支付方式
userid	INT UNSIGNED			user(id)		操作用户
add_time	DATETIME				CURRENT_TIMESTAMP	添加时间

## 4 系统详细设计

### 4.1 账号模块详细设计

账号模块包含注册、登录两个模块。我们使用 wxFormBuilder 进行 GUI 构建（图 3-14），其他模块也均由此框架构建。所有窗口都可以通过图形化界面设计，非常方便。在账号加密方面我们使用 SHA256 模块加密密码，保障用户的数据安全。账号注册时，通过 GUI 界面读取到对应账号信息，加密后写入数据库的用户表中。登录时读取密码并进行 SHA256 编码，与数据库中的数据比对，成功后方可进入主页面菜单。

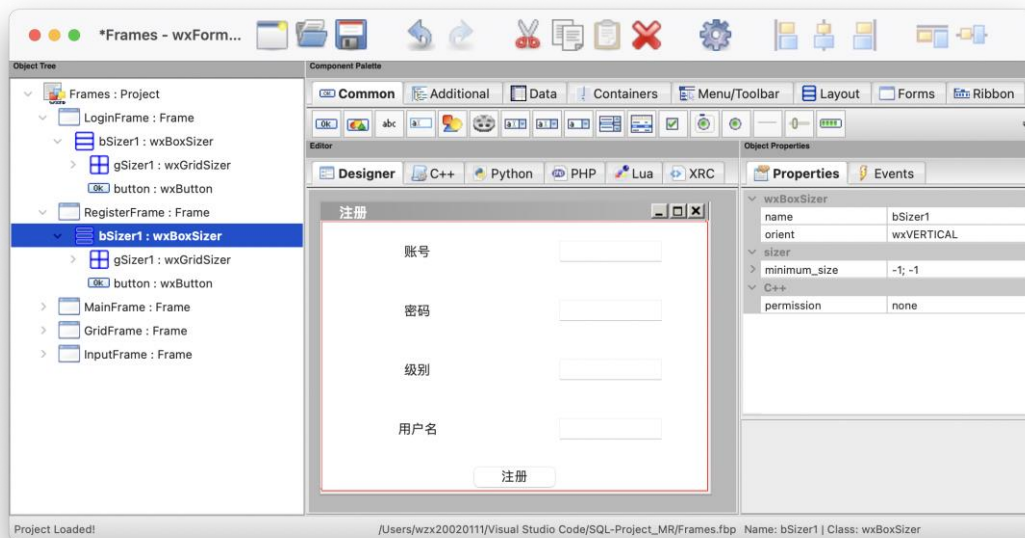


图 3-14 wxFormBuilder 设计界面

## 4.2 主菜单模块详细设计

主菜单模块包含账号管理、供应商管理、库存管理和订单管理。其中部分模块的可用与否收到账号的权限限制。1 级账号可以使用全部功能，2 级账号可以使用除账号管理的其余功能，3 级账号仅可使用订单管理功能。这个功能使用 `wxPython` 中的 `Hide()` 函数对对应按钮进行隐藏来达到效果。

## 4.3 查看模块详细设计

查看模块包含对所选数据的查看表格，以及对应的增删改入口。查看模块通过使用 `wx.Grid()` 控件，通过读取数据库数据，自适应展示对应表的数据，做到了一份代码，多表通用。

## 4.4 增加模块详细设计

增加模块的用途是给当前表增加数据。通过查看模块可以获取到需要增加到表名，通过读取表的列数及列名自动显示对应的输入文本框。针对异常以及非法输入有着完善的异常抛出机制，可以提示用户对数据进行检查。

## 4.5 删除模块详细设计

删除模块的用途是给当前表删除数据。通过查看模块可以获取到需要增加到表名，通过读取编号的窗体获取要删除的数据编号，根据编号向数据库发送 `SQL` 语句进行删除操作。

## 4.6 修改模块详细设计

修改模块的用途是给当前表修改数据。通过查看模块可以获取到需要增加到表名，通过读取表的列数及列名自动显示对应的输入文本框。仅对需要修改的数据进行修改，读取的空数据不做处理。修改完成后有对应的提示，针对异常以及非法输入有着完善的异常抛出机制，可以提示用户对数据进行检查。

# 5 系统实现

## 5.1 账号模块实现

账号模块承担了账号注册以及登录的功能。输入正确的账号密码，在成功登录后会有相应的提示。账号模块实现界面如图 4-1 到 3 所示。



图 4-1 账号功能实现界面-登录

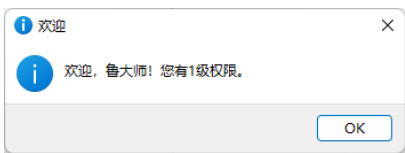


图 4-2 账号功能实现界面-登录成功提示

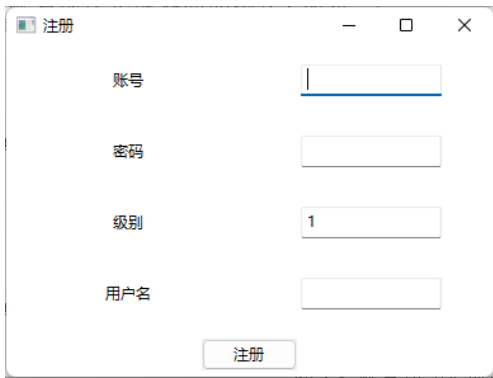


图 4-3 账号功能实现界面-注册

账号功能的核心代码如下：

```
class log(LoginFrame):
    def clicked(self, event):
        s='SELECT id,password FROM user;'
        users=dict(one_query(s))
        global uid,uname,level
        a=int(self.textCtrl1.Value)
        b=self.textCtrl2.Value
```

```

b=sha256(b.encode('utf-8')).hexdigest()
if a not in users:
    wx.MessageBox('无此账号','错误',wx.ICON_ERROR)
elif users[a]!=b:
    wx.MessageBox('密码错误','错误',wx.ICON_ERROR)
else:
    s='SELECT name,level FROM user WHERE id=%s;'%a
    uid=a
    uname,level=one_query(s)[0]
    self.Close()

class reg(RegisterFrame):
    def clicked(self, event):
        global uid,uname,level
        a=self.textCtrl1.Value
        b=self.textCtrl2.Value
        c=self.textCtrl3.Value
        d=self.textCtrl4.Value
        if a==' ' or b==' ' or c==' ' or d==' ':
            wx.MessageBox('不能留空','输入出错',wx.ICON_ERROR)
        elif not a.isdigit() and not c.isdigit():
            wx.MessageBox('账号和权限必须是数字','输入出错',wx.ICON_ERROR)
        else:
            uid,uname,level=a,d,c
            b=sha256(b.encode('utf-8')).hexdigest()
            s='INSERT INTO user (id,password,name,level) VALUES
(%s,"%s", "%s", %s);'%(a,b,d,c)
            one_query(s)
            self.Close()

```

## 5.2 主菜单模块实现

主菜单模块是整个系统的中枢，会根据不同的账号等级，展示不同的功能。点击不同的按钮可以到达对应的功能界面。主菜单模块实现界面如图 4-4 所示。



图 4-4 主菜单功能实现界面

主菜单功能的核心代码如下：

```
class dashboard(MainFrame):
    def act(self, event):
        if level>1:
            self.m_button1.Hide()
            self.m_button5.Hide()
        if level>2:
            self.m_button2.Hide()
            self.m_button3.Hide()
        s='SELECT COUNT(*) FROM user'
        data=str(one_query(s)[0][0])
        self.m_staticText1.Label='共有 '+data+' 个员工'
        s='SELECT COUNT(*) FROM company'
        data=str(one_query(s)[0][0])
        self.m_staticText2.Label='共有 '+data+' 个供应商'
        s='SELECT COUNT(*) FROM goods'
        data=str(one_query(s)[0][0])
        self.m_staticText3.Label='共有 '+data+' 种商品'
        s='SELECT COUNT(*) FROM orders'
        data=str(one_query(s)[0][0])
        self.m_staticText4.Label='共有 '+data+' 个订单'
    def user_click(self, event):
        col_names=('id','name','password','level','add_time')
        table_name='user'
        show(col_names,table_name)
    def company_click(self, event):
        col_names=('id','name','address','telephone','email','contact_person','userid','add_time')
        table_name='company'
        show(col_names,table_name)
    def goods_click(self, event):
        col_names=('id','name','place_of_production','price','num','companyid','userid','add_time')
        table_name='goods'
        show(col_names,table_name)
    def orders_click(self, event):
        col_names=('id','goodsid','num','payment_type','userid','add_time')
        table_name='orders'
        show(col_names,table_name)
    def reg_click(self, event):
        app2 = wx.App()
```



```

frame = reg(None)
frame.textCtrl3.Value = '1'
frame.Show()
app2.MainLoop()

```

### 5.3 查看模块实现

查看模块用于查看不同表中的数据，同时也是增删改功能的入口。查看模块实现界面如图4-5、6所示。

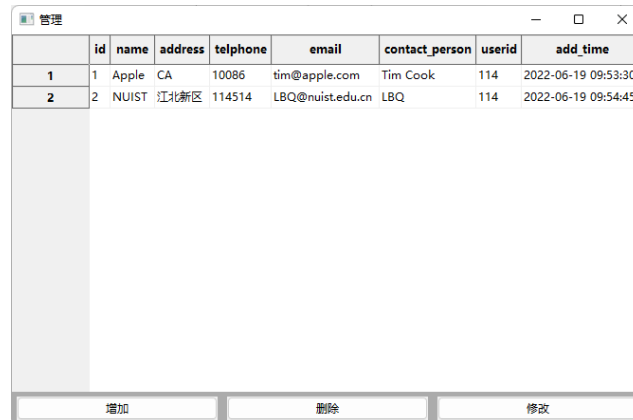


图 4-5 查看功能实现界面-供应商

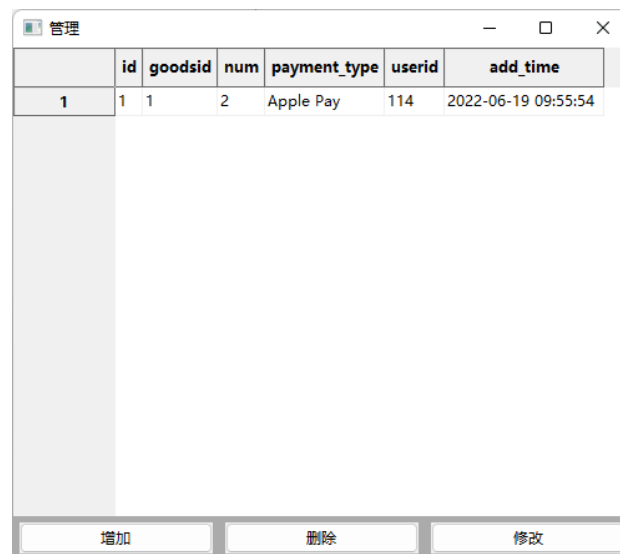


图 4-6 查看功能实现界面-订单

查看功能的核心代码如下：

```

def show(col_names:tuple[str],table_name:str):
    apps=wx.App()

    frame=Grid(None)

    frame.col_names=col_names

```

```

frame.table_name=table_name
s = 'SELECT '+','.join(col_names)+' FROM '+table_name+';'
data=one_query(s)

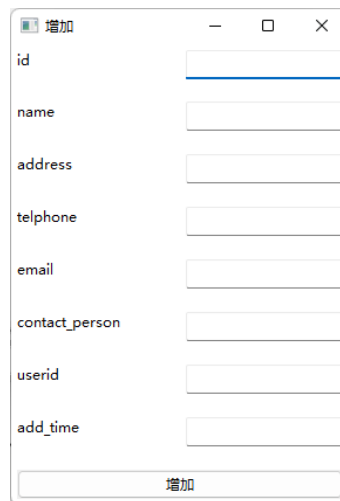
table = Table(head=col_names, body=data, changeable=True)
frame.m_grid.SetTable(table=table, takeOwnership=True)
frame.m_grid.AutoSize()

frame.Show()

```

#### 5.4 增加模块实现

增加模块用于给当前正在查看的表增加数据。点击增加即可增加数据，添加成功会有相应提示并刷新表格，对于非法数据会有对应的错误提示。增加模块实现界面如图 4-7 到 9 所示。



The dialog box '增加' contains the following fields and a button:

- id:
- name:
- address:
- telephone:
- email:
- contact\_person:
- userid:
- add\_time:
- 增加:

图 4-7 增加功能实现界面

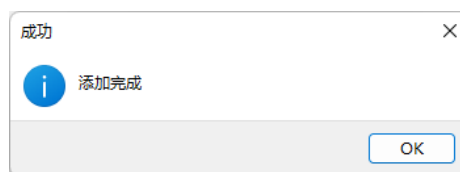


图 4-8 增加功能实现界面-成功提示



图 4-9 增加功能实现界面-报错提示

增加功能的核心代码如下：

```

def add(self, event):
    wx.Yield()
    x=get_item(self.col_names).split(',')
    col=[]
    inp=[]
    for i in range(len(self.col_names)):
        if x[i].strip()!='':
            col.append(self.col_names[i])
            inp.append(''+x[i]+'')
    s='INSERT INTO '+self.table_name+' ('+','.join(col)+') VALUES ('+','.join(inp)+');'
    try:
        one_query(s)
        wx.MessageBox('添加完成', '成功', wx.ICON_INFORMATION)
        s = 'select '+','.join(self.col_names)+' from '+self.table_name+';'
        data=one_query(s)
        table = Table(head=self.col_names, body=data, changeable=True)
        self.m_grid.SetTable(table=table, takeOwnership=True)
        self.m_grid.AutoSize()
        self.Layout()
    except:
        wx.MessageBox('数据有误，请重试', '失败', wx.ICON_WARNING)

```

## 5.5 删除模块实现

删除模块用于删除正在查看的表中的数据。输入想要删除的数据编号即可删除，有约束的情况会有对应的错误提示。查看模块实现界面如图 4-10 到 12 所示。



图 4-10 删除功能实现界面

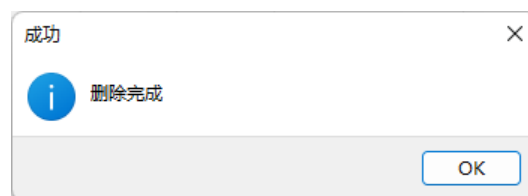


图 4-11 删除功能实现界面-成功提示

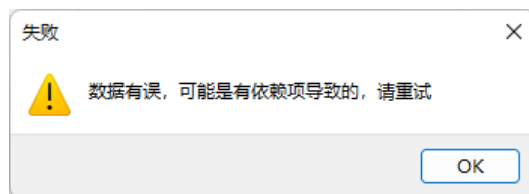


图 4-12 删除功能实现界面-报错

查看功能的核心代码如下:

```
def delete(self, event):
    wx.Yield()
    i=get_input()
    s='DELETE FROM '+self.table_name+' WHERE id = '+i
    try:
        one_query(s)
        wx.MessageBox('删除完成','成功',wx.ICON_INFORMATION)
        s = 'select '+'+'.join(self.col_names)+' from '+self.table_name+';'
        data=one_query(s)
        table = Table(head=self.col_names, body=data, changeable=True)
        self.m_grid.SetTable(table=table, takeOwnership=True)
        self.m_grid.AutoSize()
        self.Layout()
    except:
        wx.MessageBox('数据有误, 可能是有依赖项导致的, 请重试','失败',wx.ICON_WARNING)
```

## 5.6 修改模块实现

修改模块用于修改正在查看的表中的数据。输入对应的编号即可即可跳转到修改界面。成功修改后有相应的提示, 修改失败会有对应的错误提示。修改模块实现界面如图 4-13 到 15 所示。



图 4-13 修改功能实现界面-获取编号

图 4-14 修改功能实现界面

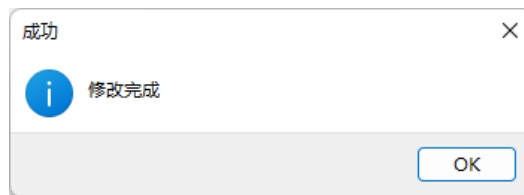


图 4-15 修改功能实现界面

查看功能的核心代码如下：

```
def mod(self, event):
    wx.Yield()
    i=get_input().strip()
    s='SELECT * FROM '+self.table_name+' WHERE id= '+i+';'
    # try:
    col=self.col_names[:-1]
    data=one_query(s)[0][:-1]
    data=mod_item(col,data)
    # print(data)
    data=['']+x.strip()+'' for x in data.split(',')
    t=','.join([x+'='+y for x,y in zip(col,data)])
    s='UPDATE '+self.table_name+' SET '+t+',add_time=CURRENT_TIMESTAMP
WHERE id = '+i+';'
    print(s)
    one_query(s)
```

```

wx.MessageBox('修改完成','成功',wx.ICON_INFORMATION)
s = 'select '+'+'.join(self.col_names)+' from '+self.table_name+';'
data=one_query(s)
table = Table(head=self.col_names, body=data, changeable=True)
self.m_grid.SetTable(table=table, takeOwnership=True)
self.m_grid.AutoSize()
self.Layout()

```

## 6 总结

本次课程设计总体进展相对顺利。首先我们查阅相关资料以及论文，确定了工作方向和大致框架，如表的设计，程序设计的大题思路等。在这个阶段我们根据自身的实际情况，确定了分工、技术路线和协作工具，为后面工作的开展打下了良好的基础。在设计和开发中，我们结合同学间的交流，增加了账号系统，根据不同账号的权限来限制对应的功能。在与指导老师的交流中，我们意识到在多表联动中还有欠缺，在后续的开发中通过触发器弥补了这一缺陷。

目前这套系统已经可以实现了基本的商品库存管理功能，包括供应商管理、库存管理和订单管理，已经经过多次测试，可以在不同平台下稳定运行。目前系统还有诸多不足，在进货和出货方面我们在设计中一起合并到订单功能里，仅靠账号标志区分，这可能造成潜在的管理混乱。同时，软件部分界面的适配还没有做到自适应，部分窗口需要调整大小才能完全显示。

在未来的工作中，首要目标是细化个表的功能，根据实际需求添加表。其次就是改进系统界面的外观，做到易用与美观两方面。最后在安全性方面也需要加以考虑，防范 SQL 注入等操作。

最后，感谢指导老师以及相关同学给予我们的指导和帮助。