**数据库简单仓库管理系统项目设计文档**

课 程　 数据库课程设计

系统名称　 简单仓库管理系统

专 业 软件工程

班 级 1602

学 号　 161060009

学生姓名 黄佳俊

嘉应学院计算机学院

2018 年 12月 23日

摘要：仓库全人为管理的方法现存在管理时间约束，管理效率低,信息处理速度低而且准确率难以令人满意等缺点。 为了提高仓库管理效率, 减轻人员的劳动强度,提高信息处理速度和准确性;需要为仓库管理员提供一种更科学的管理仓库的。设计实现的管理系统能够正确有效地完成仓库管理的日常入库出库等工作,极大提高了管理工作的效率和正确性, 并能够根据需求进行灵活的查询, 完成月底盘点等功能;系统与数据库管理系统结合, 提供了功能较为强大的数据安全功能,防止了越权操作、误操作,并支持灾难性恢复;系统界面友好,操作简单,效率高易掌握, 并且让计算机对仓库进行自动管理, 仓库管理员可以直接在计算机上实现仓库的信息管理, 并能在一定程度上实现自动化。采用MYSQL Ver 8.0.12的数据库技术作为后台数据库，PYTHON作为开发语言，通过目前最新的模块PYQT5(pyqt5.9.2)和QT5.7.1技术实现窗口界面，主要完成货物管理，用户管理等功能，基本实现python语言的图形化界面编程，满足用户与管理员的使用需求，但在某些方面还存在一些问题，如在切换窗口时在切换多个窗口时会卡住，这应该是线程问题，需要改进。

关键词：仓库管理，MYSQL Ver 8.0.12，PYTHON，PYQT5(pyqt5.9.2)

Abstract: The whole-person warehouse management method exists problems such as management of time constraints, management efficiency is low, information processing speed is low and accuracy is difficult to satisfy the shortcomings.In order to improve the efficiency of warehouse management, reduce the labor intensity of personnel, improve the speed and accuracy of information processing,it’s time to provide a more scientific warehouse management for warehouse managers.The design and implementation of the management system can correctly and effectively complete the warehouse management of the daily warehousing in and out of the warehouse and other work, greatly improve the efficiency and accuracy of the management work, and according to the needs of flexible query, complete the monthly inventory and other functions.Combined with the database management system, the system provides powerful data security functions, prevents unauthorized operation and misoperation, and supports catastrophic recovery.The system interface is friendly, the operation is simple, the efficiency is high,easy to learn and lets the computer carry on the automatic management to the warehouse, the warehouse manager may realize the warehouse information management directly on the computer, and may realize the automation to a certain extent.Using MYSQL Ver 8.0.12 database technology as the background database, PYTHON as a development language, by now the latest module PYQT5 (pyqt5.9.2) and QT5.7.1 technical implements window interface, mainly finished goods management, user management, and other functions, basic implementation of PYTHON language graphical interface programming, meet the demands of users and administrators, but some problems still exist in some aspects, such as the switching window but when switching multiple Windows the system will get stuck, it should be a thread problem, and I need to update and improve it.

Keywords: warehouse management, ，MYSQL Ver 8.0.12，PYTHON，QT5.7.1 PYQT5(pyqt5.9.2)

目录

摘要·····································································································1

Abstract·····························································································1

1前言··································································································3

2开发环境及相关技术······································································3

2.1开发环境及技术·····························································4

2.2相关技术介绍·································································5

3系统需求分析··················································································6

3.1数据字典·········································································7

3.2 uml图··············································································7

4 系统设计·························································································8

4.1概念设计·········································································8

4.2逻辑设计·········································································8

4.3物理设计········································································11

4.4用户模式设计································································12

4.5安全性设计····································································14

4.6事务设计········································································14

4.7用户接口界面设计························································15

5 测试································································································18

6 开发总结························································································28

7 参考文献························································································28

**1 前言**

随着计算机及网络技术的飞速发展, InternetIntranet应用在全球范围内日益普及,当今社会正快速向信息化社会前进,信息系统的作用也越来越大。仓库是供应链的核心环节, 仓库管理水平的高低, 会直接影响到物流运作的效率。当仓库由静态的储存转化为动态的物流, 在物流与供应链概念的基础上, 对仓库的功能进行重新定义, 便显得尤为重要。数据库中包括货物出入库信息和商品销售等信息。该系统包括系统登录、货物出入库管理、用户管理和购物车管理等功能，通过这些功能实现对库存中货物信息和销售信息进行管理。

系统主要实现以下目标：

 实现系统登录及修改用户密码的功能。

 对库存货物的出入库信息进行管理。

 对用户的信息进行管理。

用户对购物车等进行管理。

**2开发环境及相关技术**

**2.1开发环境及所用技术：**

系统开发环境：pycharm2018.3

系统开发语言：python

运行平台：win10

数据库：mysql Ver 8.0.12 for Win64 on x86\_64 (MySQL Community Server - GPL)

界面开发技术：pyqt5.9.2 QT5.7.1

**2.2相关技术介绍：**

**Introduction To PYTHON**

Python is a general-purpose, dynamic, object-oriented programming language. The design purpose of the Python language emphasizes programmer productivity and code readability. Python was initially developed by Guido van Rossum. It was first released in 1991. Python was inspired by ABC, Haskell, Java, Lisp, Icon, and Perl programming languages. Python is a high-level, general purpose, multiplatform, interpreted language. Python is a minimalistic language. One of its most visible features is that it does not use semicolons nor brackets. It uses indentation instead. There are two main branches of Python currently: Python 2.x and Python 3.x. Python 3.x breaks backward compatibility with previous releases of Python. It was created to correct some design flaws of the language and make the language more clean. Python is maintained by a large group of volunteers worldwide. Python is open source software. Python is an ideal start for those who want to learn programming.Python supports object-oriented and procedural programming. There is also a limited support for functional programming.

**Introduction To PYQT5**

PyQt5 is a set of Python bindings for Qt5 application framework from Digia. It is available for the Python 2.x and 3.x. This tutorial uses Python 3. Qt library is one of the most powerful GUI libraries. The official home site for PyQt5 is [www.riverbankcomputing.co.uk/news](http://www.riverbankcomputing.co.uk/news). PyQt5 is developed by Riverbank Computing.

PyQt5 is implemented as a set of Python modules. It has over 620 classes and 6000 functions and methods. It is a multiplatform toolkit which runs on all major operating systems, including Unix, Windows, and Mac OS. PyQt5 is dual licensed. Developers can choose between a GPL and a commercial license.

**Introduction To MYSQL**

The MySQL™ software delivers a very fast, multithreaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

**3需求分析**

**3.1数据字典**

问题的提出:为了高效率的完成仓库的管理,决定开发仓库管理系统。

需求及描述包括：入出库登记：记录对应货物id的单价和进货数量，记录供应商信息，登记出入库时间，便于检查出错等情况，管理库内信息 ：删除修改查询库内信息，产品分类管理：将货物分类，简洁高效 ，登录退出系统：通过账号密码获取权限登录，退出时会有记录进退出时间的日志，预警管理：对货物保质期，库存剩余量进行提醒

1. 抽象出系统实体

货物(货物编号、货物名称，货物数量，货物价格，货物类别，货物生产日期，货物保存天数)

用户(用户编号，用户姓名，用户电话，用户密码)

购物车(用户编号，购买的商品编号，对应商品数量，对应商品的价格，总价格)

管理员(账号，密码)

**3.2 uml图**

用户

用户编号

用户姓名

用户电话

用户密码

货物

货物编号

货物名称

货物数量

货物价格

货物类别

货物生产日期

货物保存天数

购买

管理员

账号

密码

用户

用户编号

用户姓名

用户电话

用户密码

被管理于

购物车

用户编号

货物编号

货物数量

货物价格

总价格

货物

货物编号

货物名称

货物数量

货物价格

货物类别

货物生产日期

货物保存天数

记录

记录

用户

用户编号

用户姓名

用户电话

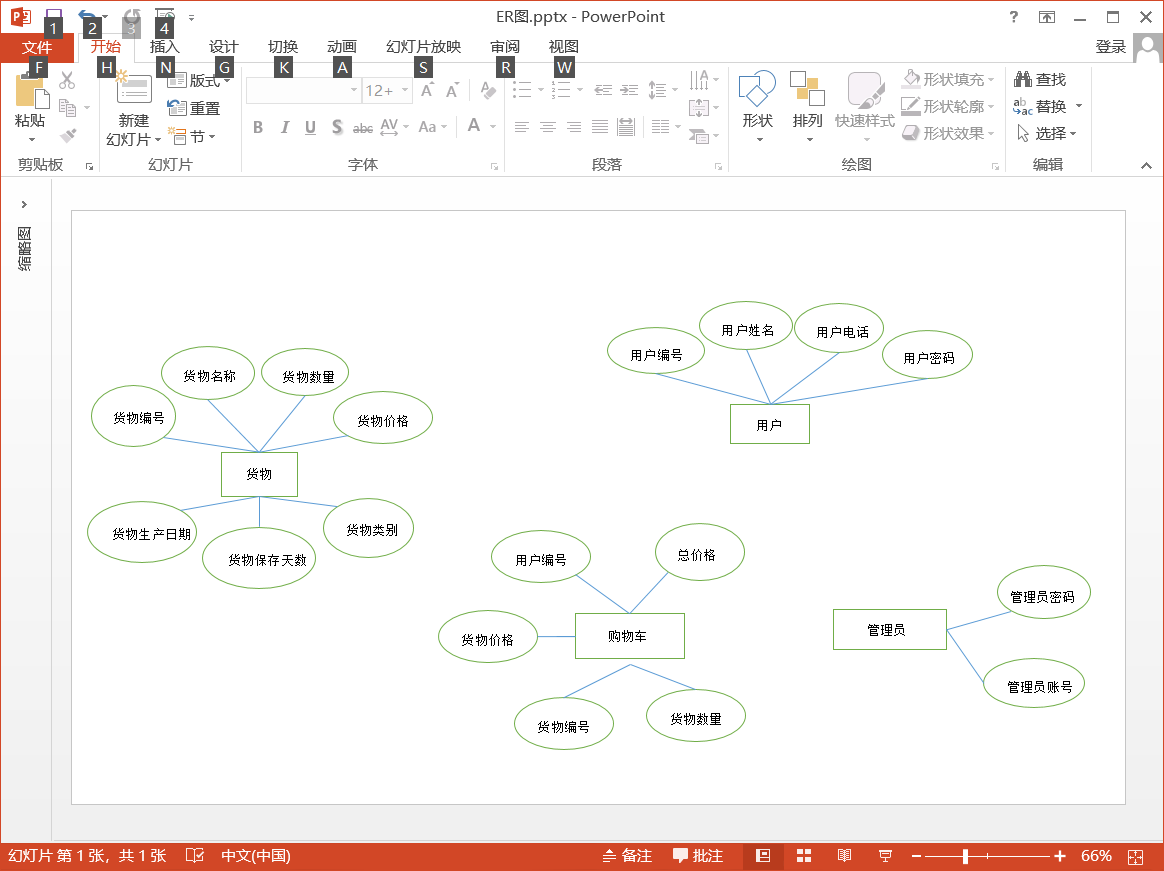
用户密码

**4系统设计**

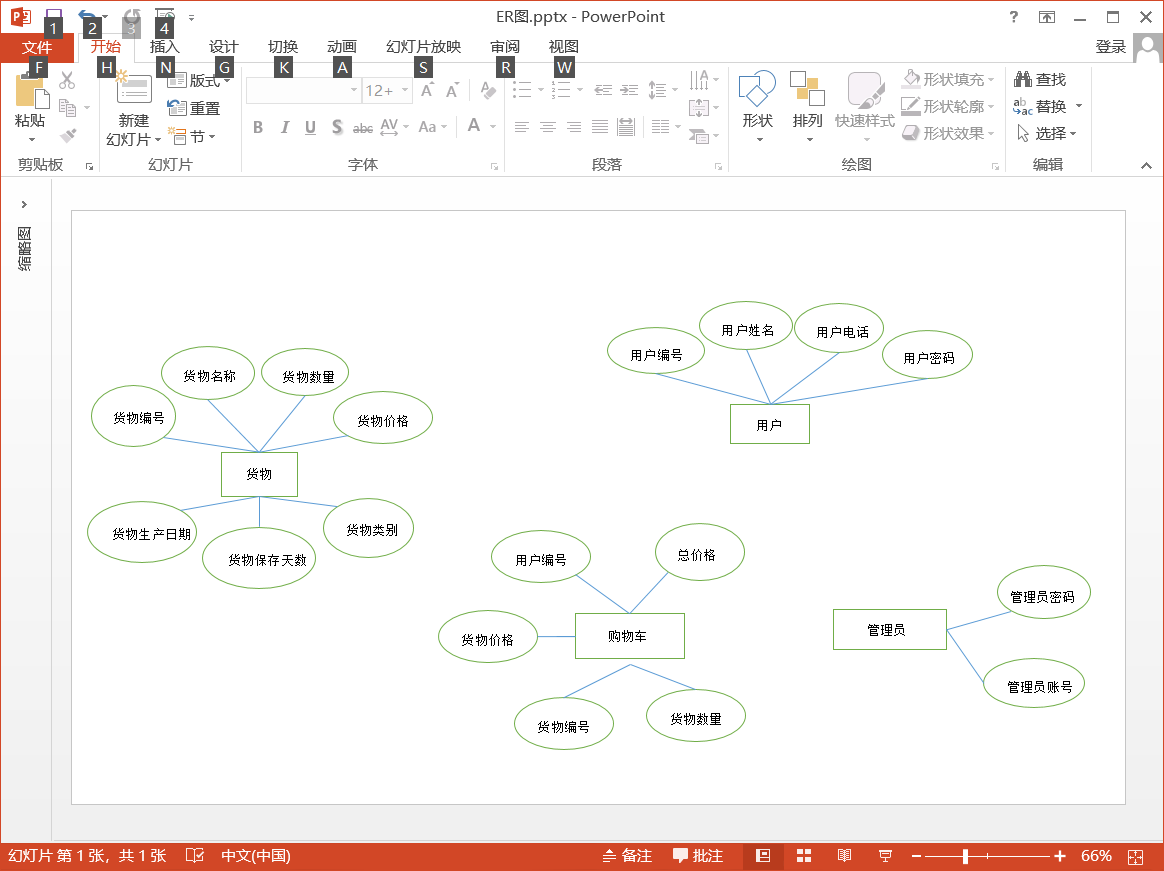
**4.1概念设计**

系统管理模块：

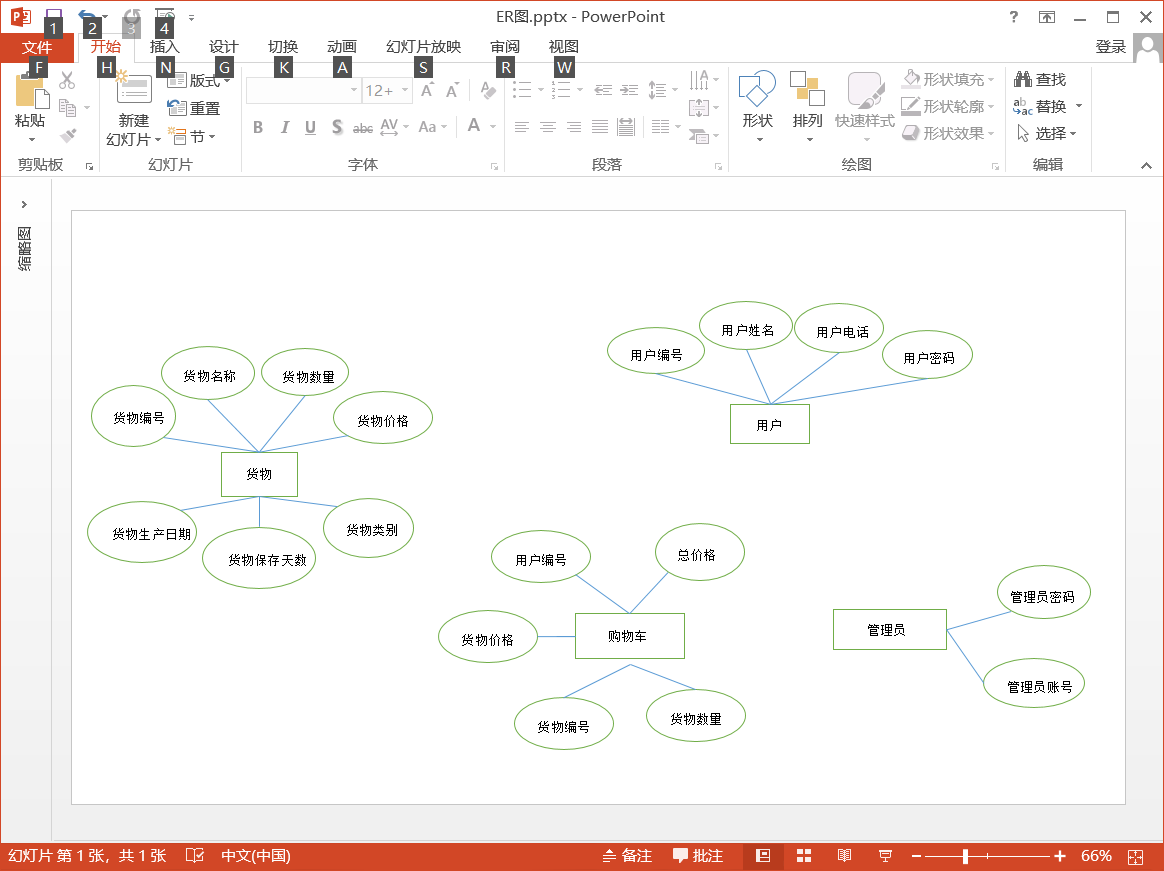
各个实体属性图（6个）



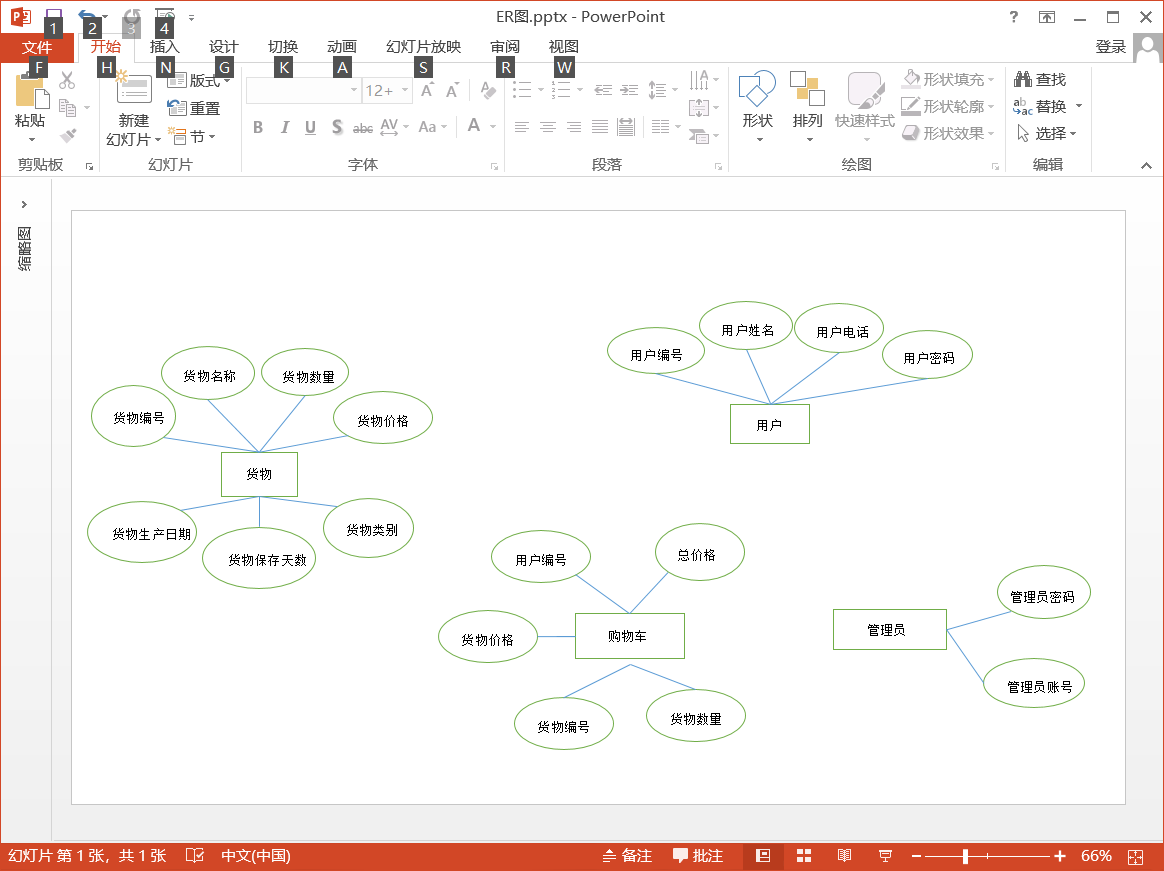
**货物实体属性图**



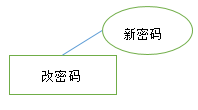
**用户实体属性图**



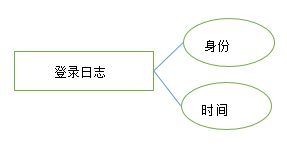
**购物车实体属性图**



**管理员实体属性图**



**改密码实体属性图**

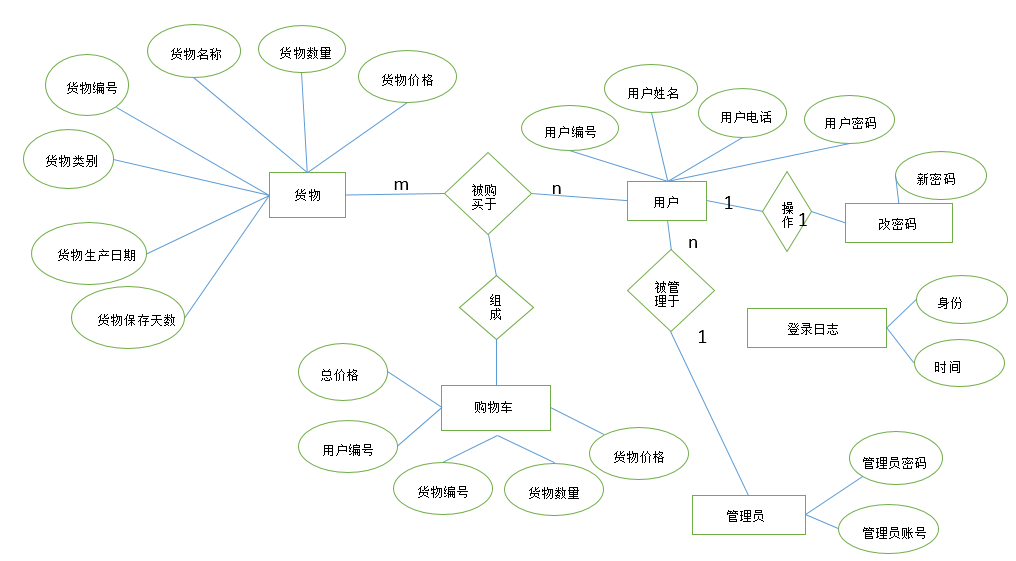


**登录日志实体属性图**

实体之间的联系

根据需求分析,归结出合适的联系:1、一个用户能够选多个货物,一个货物可以被多个用户选2 、一个管理员能管理多个用户 3、购物车可以记录多个用户，购物车可以记录多个货物，一个货物可以被多个购物车记录，一个用户可以被多个购物车记录

实体关系（ER）图:



**4.2逻辑设计**

关系模式（表）设计

根据需求分析中的E-R图,通过对实体的属性和之间的联系的分析,我们将其由概念模型（ER图）向关系模型转化

数据库中包含9个表,即goods(货物表), date(日期表), client(客户表), Client\_pwd(客户密码表),

manager(管理员表), trade(购物车表), user\_trade(用户购物车表),user\_log(用户日志表)，manager\_log(管理员日志表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| goods(货物表) | | | | |
| 列名 | | 数据类型 | 空/非空 | 约束条件 |
| 货号(stock\_id) | INT(10) | | NOT NULL | PRIMARY KEY |
| 货物名字(stock\_name) | VARCHAR(20) | | NULL |  |
| 货物数量(stock\_num) | INT(10) | | NULL |  |
| 货物价格(stock\_price) | FLOAT | | NULL |  |
| 货物种类(stock\_kind) | VARCHAR(10) | | NULL |  |
| 货物生产日期(stock\_expiration\_dates) | DATE | | NULL |  |
| 货物保存日期(stock\_days) | INT | | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| date(日期表) | | | |
| 列名 | 数据类型 | 空/非空 |  |
| 货号(stock\_id) | INT(10) | NOT NULL | PRIMARY KEY |
| 货物生产日期(stock\_expiration\_dates) | DATE | NULL |  |
| 货物保存日期(stock\_days) | INT | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| client(客户表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 客户编号(c\_id) | INT(10) | NOT NULL | PRIMARY KEY |
| 客户名字(c\_name) | VARCHAR(10) | NULL | UNIQUE |
| 客户电话(c\_phone) | VARCHAR(20) | NULL | UNIQUE |
| 客户密码(c\_pwd) | VARCHAR(10) | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| client\_pwd(客户密码表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 客户编号(c\_id) | INT(10) | NOT NULL | PRIMARY KEY |
| 客户密码(c\_pwd) | VARCHAR(10) | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| manager(管理员表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 管理员名字(m\_name) | VARCHAR(20) | NULL | UNIQUE |
| 管理员密码(m\_pwd) | INT(10) | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| trade(购物车表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 客户编号(c\_id) | INT(10) | NOT NULL | PRIMARY KEY |
| 货物编号(stock\_id) | VARCHAR(20) | NOT NULL |  |
| 货物数量(stock\_num) | INT(10) | NULL |  |
| 货物价格(stock\_price) | FLOAT | NULL |  |
| 总价格(total\_price) | VARCHAR(10) | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| user\_trade(用户购物车表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 客户姓名(c\_name) | VARCHAR(20) | NOT NULL |  |
| 货物名字(stock\_name) | VARCHAR(20) | NOT NULL |  |
| 货物数量(stock\_num) | INT(10) | NULL |  |
| 货物价格(stock\_price) | FLOAT | NULL |  |
| 总价格(total\_price) | VARCHAR(10) | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| user\_log(用户日志表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 用户编号(c\_id) | INT(10) | NOT NULL |  |
| 登录时间(u\_log\_ datetimes) | DATETIME | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| manager\_log(管理员日志表) | | | |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| 管理员名字(m\_name) | VARCHAR(20) | NOT NULL |  |
| 登录时间(log\_ datetimes) | DATETIME | NULL |  |

**4.3物理设计**

索引设计，数据库完整性设计和触发器设计。

索引设计：

use mydb create unique non clustered index index stock\_id on goods(stock\_id)

数据库完整性设计

约束条件，参照完整性，属性值限制

create database mydb charset gbk;

use mydb;

create table goods(

stock\_id int(10) primary key auto\_increment,

stock\_name varchar(10) unique,

stock\_num int(10) not null,

stock\_price float not null,

stock\_kind varchar(10) not null,

stock\_expiration\_dates date not null,

stock\_days int(10) not null

)charset utf8;

create table date(

stock\_id int(10) primary key auto\_increment,

stock\_expiration\_dates date null,

stock\_days int(10) null

)charset utf8;

create table client(

c\_id int(10) primary key auto\_increment,

c\_name varchar(10) unique null,

c\_phone varchar(20) unique null,

c\_pwd varchar(10) not null

)charset utf8;

create table client\_pwd(

c\_id int(10) primary key auto\_increment,

c\_pwd varchar(10) null

)charset utf8;

create table manager(

m\_name varchar(10) unique,

m\_pwd varchar(10)

)charset utf8;

create table trade(

c\_id int(10) not null,

stock\_id int(10) not null,

stock\_num int(10) not null,

stock\_price float not null,

total\_price float

)charset utf8;

create table user\_trade(

c\_name varchar(20) not null,

stock\_name varchar(20) not null,

stock\_num int(10) not null,

stock\_price float not null,

total\_price float

)charset utf8;

create table user\_log(

c\_id int(10) not null,

u\_log\_datetimes datetime not null

)charset utf8;

create table manager\_log(

m\_name varchar(20) not null,

log\_datetimes datetime null

)charset utf8;

触发器设计

在数据库的goods表中,定义一个触发器,当一个货物库存为零时,把该货物的编号添加到delgoods表中。

创建一个触发器

CREATE TABLE delgoods SELECT stock\_id FROM goods WHERE stock\_num = 0;

创建一个空表用于接受删除后的数据

CREATE TRIGGER trig\_goods //创建goods表的触发器

AFTER DELETE ON goods FOR EACH ROW

INSERT INTO delgoods stock\_id values old. stock\_id ;

验证

DELETE FROM goods WHERE stoci\_id=2;//删除一个学生,验证触发器是否触发

**4.4用户模式设计**

视图的实现

1. 创建查询类别为水果的id，一列多行（列子查询：子查询得到的结果是一列数据（一列多行））的视图

CREATE VIEW [goods]

AS

select stock\_id from mydb.goods where stock\_kind in (select stock\_kind from mydb.goods where stock\_kind='水果');

查询定义视图 [goods]

查询语句：select \* from [goods];

1. 创建[client]视图

CREATE VIEW [client]

AS

SELECT c\_id,c\_name,c\_phone

FROM client;

查询定义视图[client]

查询语句：select \* from [client];

1. 创建[manager]视图

CREATE VIEW [manager]

AS

SELECT m\_name,m\_pwd

FROM manager;

查询定义视图[manager]

查询语句：SELECT m\_name,m\_pwd FROM manager;

存储过程的实现：

（1）创建一个存储过程，从goods表检索所有类别为水果的id，数量

DELIMITER // /\*将MYSQL的结束符设置为//\*/

CREATE PROCEDURE proc\_goods()

READS SQL DATA

BEGIN

select stock\_id,stock\_num from mydb.goods where stock\_kind in (select stock\_kind from mydb.goods where stock\_kind='水果');

END //

DELIMITER; /\*将结束符恢复成分号\*/

执行存储过程 : CALL proc\_goods;

（2）建立一个叫stock\_num\_120存储过程，统计stock\_num为120的水果有哪些

DELIMITER // /\*将MYSQL的结束符设置为//\*/

CREATE PROCEDURE stock\_num\_120()

BRFADS SOL DATA

BEGIN

select stock\_name from goods where stock\_num =120;

END //

DELIMITER; /\*将结束符恢复成分号\*/

执行存储过程 : CALL stock\_num\_120;

**4.5安全性设计**

设计两种角色（管理员、用户），分配角色的数据库对象的操作权限。

(1)管理员权限

赋予全部操作权限给管理员，并允许管理员师将权限赋予其他用户

GRANT ALL PRIVILEGES

ON TABLE goods,client,manager,trade

TO manager

WITH GRANT OPPTION;

1. 普通用户

赋予select权限给所有用户

GRANT SELECT

ON TABLE goods,client,manager,trade

TO PUBLIC;

**4.6事务设计**

MySQL中事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

设计2个事务。

（1）Begin transaction dem

select \* from goods

insert into goods values ('1', '苹果', 100,20,'水果','1900-01-01',100)

save transaction sd1

insert into goods values('2', '火龙果',130,20,'水果','1900-01-01',100)

select \* from goods

rollback transaction sd1

select \* from goods

rollback transaction dem

（2）Begin transaction dem

select \* from goods

insert into goods values('14', '西红柿',111,20,'蔬菜','1900-01-01',100)

save transaction sd2

insert into goods values('15', '西蓝花', 111,20,'蔬菜','1900-01-01',100)

select \* from goods

rollback transaction sd2

select \* from goods

rollback transaction dem

（3）Begin transaction dem

select \* from goods

insert into goods values('10', '无线鼠标', 120,20,'办公','1900-01-01',100)

save transaction sd2

insert into goods values('13', '无线键盘', 210,20,'办公','1900-01-01',100)

select \* from goods

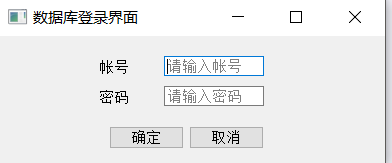
rollback transaction sd2

select \* from goods

rollback transaction dem

**4.7用户接口界面设计**

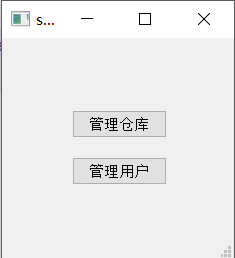
采用PYTHON语言和PYQT5编写可视化的用户接口。



**登录界面**

**登录之后会根据用户和管理员权限进入不同界面**

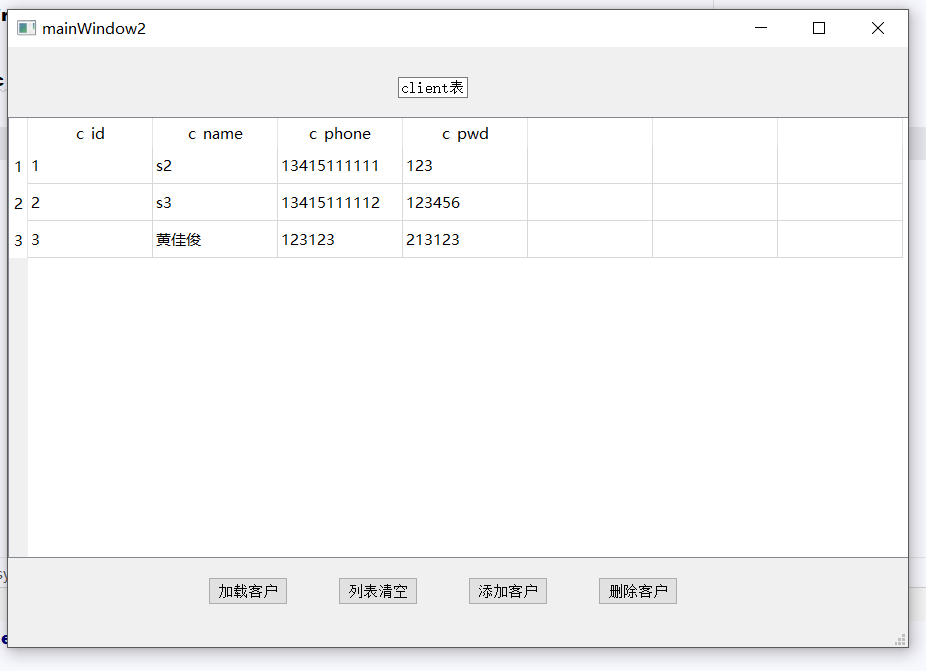
**管理员**



**管理员系统界面**

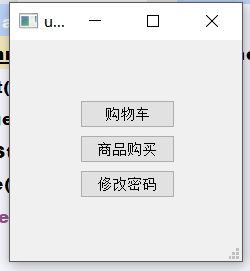


**管理员管理仓库界面**

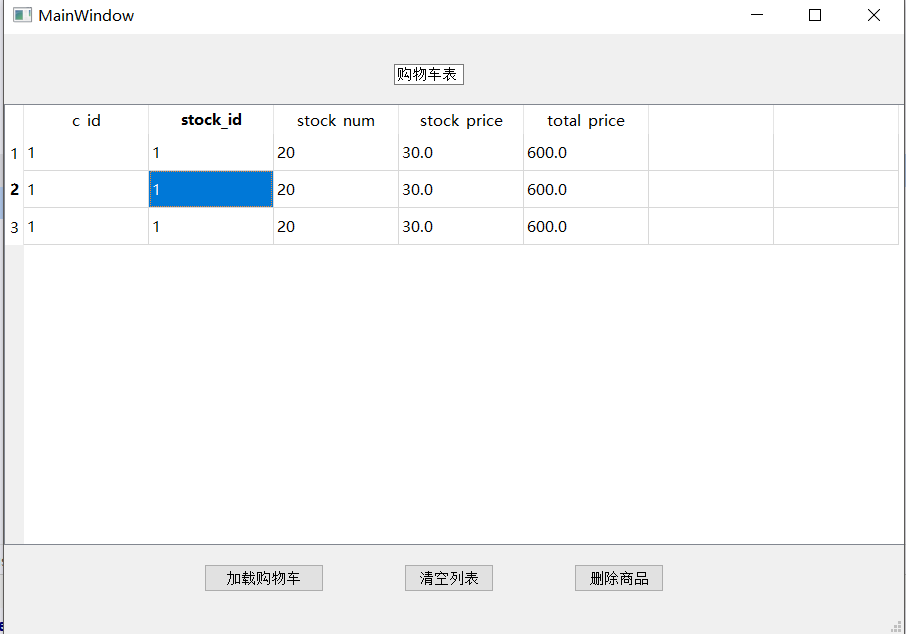


**管理员管理用户界面**

用户



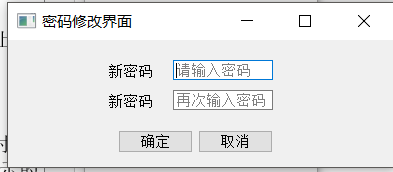
**用户系统界面**



**用户购物车界面**



**用户购买货物界面**



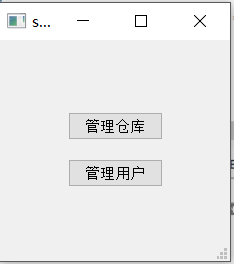
**用户修改密码界面**

**5测试**

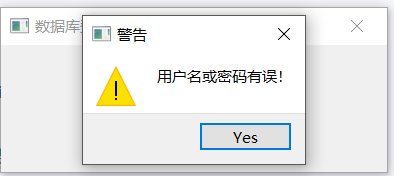
**登录测试**



登录admin键入正确密码，按确定成功弹出如下窗口，按取消退出窗口，



输入错误账号或密码时会弹出



定义的函数如下

**def word\_get**(self)**:** login\_user **=** self.lineEdit.text()  
 login\_password **=** self.lineEdit\_2.text()  
 **if** login\_user **== 'admin' and** login\_password **== '123456':** ui\_system.show()  
 MainWindow.close()  
 **elif** login\_user **== 's2' and** login\_password **== '123456':** user\_system.show()  
 MainWindow.close()  
  
 **else:** QMessageBox.warning(self,  
 **"警告"**,  
 **"用户名或密码有误！"**,  
 QMessageBox.Yes)  
 self.lineEdit.setFocus()

发现问题：

别人可以直接在输入密码框右键菜单复制非明文密码，直接盗用密码

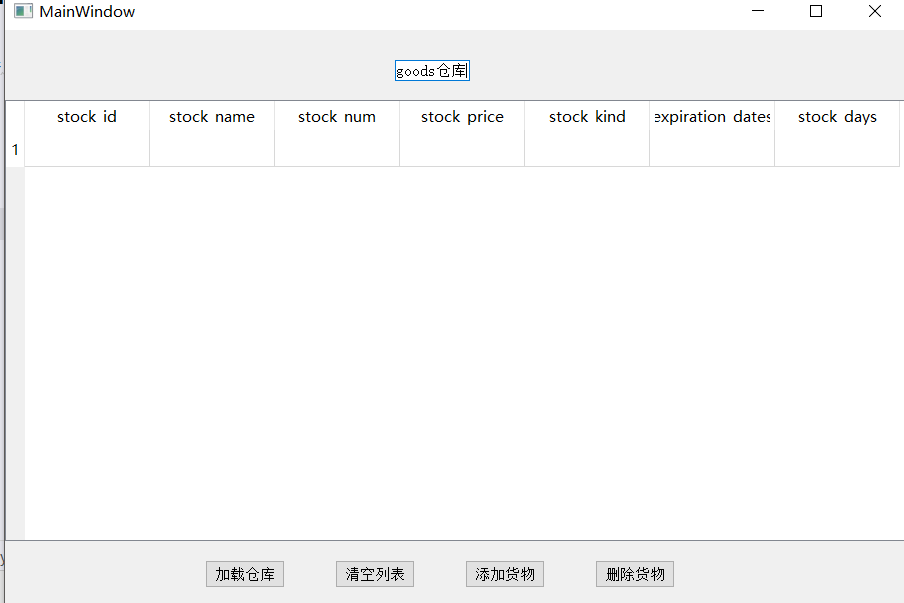
解决方案：

为了防止别人直接在输入密码框右键菜单复制非明文密码我设置QLineEdit对象的上下文菜单的策略,使得密码输入框无法右键菜单复制，该函数如下

self.lineEdit\_2.setContextMenuPolicy(Qt.NoContextMenu)

**管理测试**

接下来点击管理仓库，成功弹出窗口如下



点击加载仓库，成功加载数据



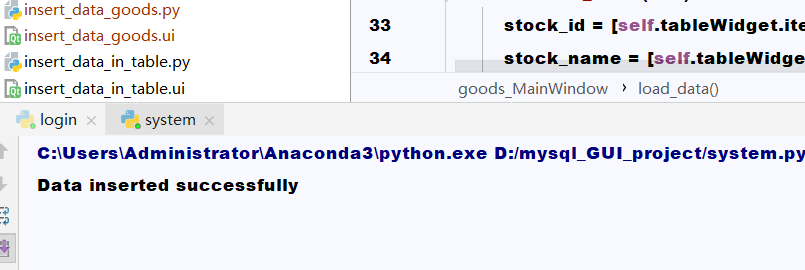
定义的如下

**def load\_data**(self)**:** conn **=** pymysql.connect(**"localhost"**,**"root"**,**"hjj111"**,**"mydb"**)  
 **with** conn**:** cursor **=** conn.cursor()  
 query **= "select \* from goods"** cursor.execute(query)  
 result1 **=** cursor.fetchall()  
 self.tableWidget.setRowCount(0)  
  
 **for** stock\_id,row\_data **in** enumerate(result1)**:** self.tableWidget.insertRow(stock\_id)  
 **for** column\_number,data **in** enumerate(row\_data)**:** self.tableWidget.setItem(stock\_id,column\_number,QtWidgets.QTableWidgetItem(str(data)))

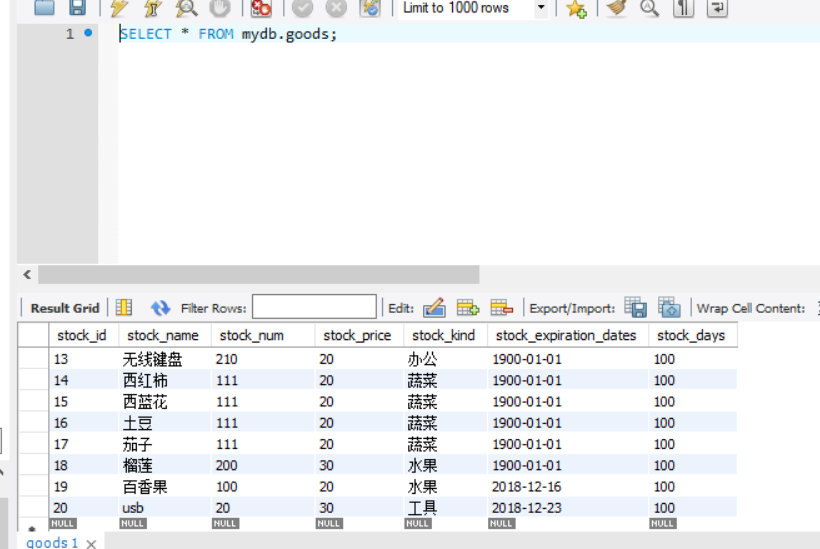
点击清空列表，便于添加货物，添加货物数据如下第二个图，点击添加







由上图可以知道数据插入成功此时用mysql workbench查看如下

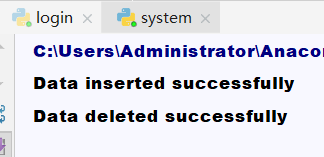


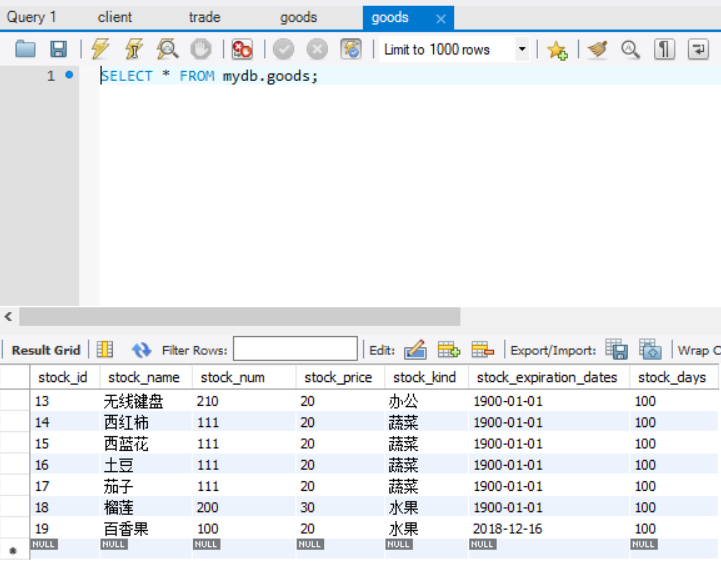
插入数据到数据库成功

插入函数如下

**def insert\_data**(self)**:** stock\_id **=** [self.tableWidget.item(row, 0).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_name **=** [self.tableWidget.item(row, 1).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_num **=** [self.tableWidget.item(row, 2).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_price **=** [self.tableWidget.item(row, 3).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_kind **=** [self.tableWidget.item(row, 4).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_expiration\_dates**=** [self.tableWidget.item(row, 5).text() **for** row **in** range(self.tableWidget.rowCount())]  
 stock\_days**=** [self.tableWidget.item(row, 6).text() **for** row **in** range(self.tableWidget.rowCount())]  
  
 con **=** pymysql.connect(**'localhost'**, **'root'**, **'hjj111'**, **'mydb'**)  
 **with** con**:** cursor **=** con.cursor()  
 cursor.execute(**"insert into goods values('%s' ,'%s', '%s', '%s', '%s', '%s', '%s')" %** (**''**.join(stock\_id),  
 **''**.join(stock\_name),**''**.join(stock\_num),**''**.join(stock\_price),**''**.join(stock\_kind),**''**.join(stock\_expiration\_dates),  
 **''**.join(stock\_days)))  
 print(**'Data inserted successfully'**)

现在进行删除货物操作，点击删除货物按钮，把刚才的插入的数据删除，同理看如下图：

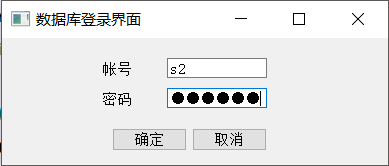


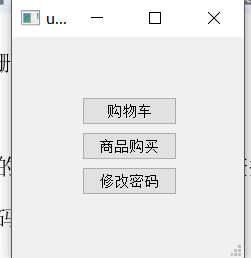


说明删除操作成功

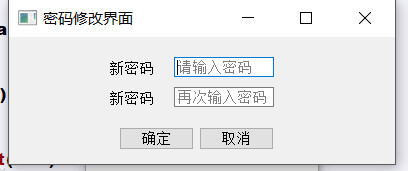
用户的操作大致一样，增删查操作已经有了，现在实现改操作，对用户密码进行更改，

登录s2用户账号，进入用户系统

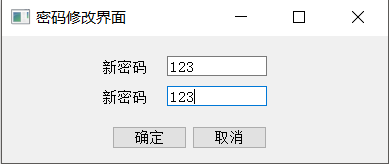


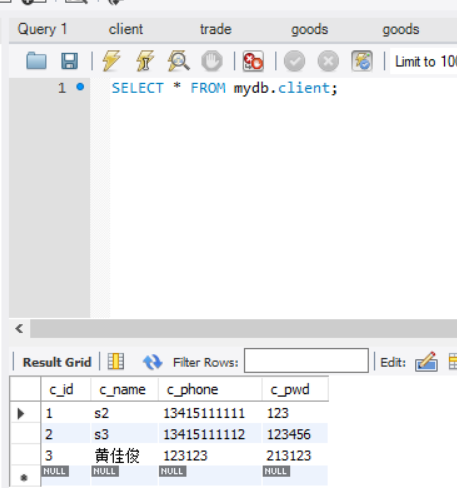


前面购物车和商品操作就不予展示，直接点击修改密码，进入该界面



把密码设置为新密码123，用mysql workbench查看





由以上知道密码修改成功，此时已基本完成测试，定义的函数和mysql语句如下：

**def word\_get**(self)**:** c\_pwd1**=** self.lineEdit.text()  
 c\_pwd2 **=** self.lineEdit\_2.text()  
 **if** c\_pwd1 **==** c\_pwd2**:** con **=** pymysql.connect(**'localhost'**, **'root'**, **'hjj111'**, **'mydb'**)  
 **with** con**:** myCursor **=** con.cursor()  
 sql **= "update client set c\_pwd=%s where c\_name='s2'" %''**.join(c\_pwd1)  
 myCursor.execute(sql)  
 print(**'Data updated successfully'**)  
 **else:** QMessageBox.warning(self,  
 **"警告"**,  
 **"两次输入不一致！"**,  
 QMessageBox.Yes)  
 self.lineEdit.setFocus()

**6开发总结**

通过这次课程设计，感觉有很多不足之处，做一个项目需要到网上查找资料，大多数文档都是英文的，有些看不懂的只能通过翻译软件翻译解释，熟练掌握很多方法和概念如

系统需求分析，数据字典，uml图，实体属性图，概念设计，实体关系（ER）图，逻辑设计，包括：关系模式（表）设计，，物理设计。包括：索引设计，数据库完整性设计和触发器设计。用户模式设计，包括视图、存储过程，安全性设计，熟悉了各种开发流程和学会如何针对一个问题提出解决方案，在学习pyqt5时是通过看一个西班牙语和英语结合的教学视频，发现即使语言不通，作为程序员的我也能从中获取很多知识，虽然6级过了，我依然会努力学习好外语，多读英文技术文档，努力达到更高的层次。

**7参考文献**

PyQt5 tutorial ZetCode:: last modified January 10, 2018 © 2007 - 2018 Jan Bodnar

https://dev.mysql.com/doc/refman/8.0/en/

MySQL 8.0 Reference Manual