[任务书 3](#_Toc160103501)

[摘要 5](#_Toc160103502)

[Abstract. 5](#_Toc160103503)

[第一章 概述 6](#_Toc160103504)

[1.1 项目背景 6](#_Toc160103505)

[1.2 编写目的 6](#_Toc160103506)

[1.3 软件定义 6](#_Toc160103507)

[1.4 开发环境 6](#_Toc160103508)

[1.4.1 QT 6](#_Toc160103509)

[1.4.2 SQL Server 7](#_Toc160103510)

[第二章 需求分析 7](#_Toc160103511)

[1.5 数据流图 8](#_Toc160103512)

[第三章 软件概要设计 9](#_Toc160103513)

[1.6 系统目标 9](#_Toc160103514)

[1.7 软件功能图 9](#_Toc160103515)

[第四章 数据库设计 16](#_Toc160103516)

[1.8 ER关系图 16](#_Toc160103517)

[1.9 数据库表的创建 17](#_Toc160103518)

[1.9.1 系统用户表 18](#_Toc160103519)

[1.9.2 员工信息表 19](#_Toc160103520)

[1.9.3 进货厂商信息表 19](#_Toc160103521)

[1.9.4 进货信息表 19](#_Toc160103522)

[1.9.5 销售信息表 20](#_Toc160103523)

[1.9.6 退货信息表 20](#_Toc160103524)

[1.9.7 现货信息表 21](#_Toc160103525)

[第五章 详细设计 21](#_Toc160103526)

[1.10 连接SQLserver数据库 21](#_Toc160103527)

[1.10.1 配置ODBS数据源 21](#_Toc160103528)

[1.10.2 QT连接数据源配置 23](#_Toc160103529)

[1.11 QT如何访问数据库 24](#_Toc160103530)

[1.12 登录系统 24](#_Toc160103531)

[1.13 SQL注入攻击 24](#_Toc160103532)

[1.14 QT中可使用预编译来防止sql注入 25](#_Toc160103533)

[1.14.1 登录 25](#_Toc160103534)

[1.14.2 添加用户 27](#_Toc160103535)

[1.14.3 修改密码 29](#_Toc160103536)

[1.15 交易管理 30](#_Toc160103537)

[1.15.1 进货登记 31](#_Toc160103538)

[1.15.2 销售登记 34](#_Toc160103539)

[1.15.3 退货登记 37](#_Toc160103540)

[1.16 库存信息查询 40](#_Toc160103541)

[1.17 进货统计 44](#_Toc160103542)

[1.18 销售统计 47](#_Toc160103543)

[1.19 业绩查看 48](#_Toc160103544)

[1.20 查看数据表 50](#_Toc160103545)

[1.21 员工表与厂商表 55](#_Toc160103546)

[第六章 用户使用说明书 57](#_Toc160103547)

[1.22 连接数据库 57](#_Toc160103548)

[第七章 结束语 64](#_Toc160103549)

[1.23 创意点 64](#_Toc160103550)

任务书

**一、课程设计目的**

《数据库与信息系统基础》课程设计是《数据库与信息系统基础》课程的后续实践性课程，是一门独立开设的实践性课程，也是毕业设计等后续课程的基础。《数据库与信息系统基础》课程设计对于巩固数据库知识，加强学生的实际动手能力和提高学生综合素质十分必要。

《数据库与信息系统基础》课程设计的主要目标是：

1、加深对数据库系统原理及程序设计的理论知识的理解和应用。

2、通过设计实际的数据库系统课题，进一步熟悉数据库管理系统的操作技术，提高动手能力，提高分析问题和解决问题的能力。

3、学习基本的数据库编程方法。掌握利用某种语言环境进行程序设计与调试的方法。（注：软件开发语言为C#或ASP或python或C++或Java或JavaScript，后台数据库为SQL server等）。

**二、选题依据**

本课程设计是对《数据库与信息系统基础》课堂教学的一个重要补充，与理论学习起着相辅相成的作用，是实施《数据库与信息系统基础》教学的一个重要组成部分。

**三、课程设计的题目要求**

能体现《数据库与信息系统基础》基本内容。

**四、课程设计的任务**

背景资料：

无论是一个销售公司还是一个销售商店，都有很多的销售数据需要管理。手工管理这些繁琐复杂的数据方式已经不适应时代的潮流，实现数据的规范化、自动化的计算机管理，是一个管理规范、运作高效的企事业单位的必然要求。本课程设计就是为了实现给销售单位高效管理而设计的。

本系统主要考虑了最为普遍的销售管理需求：

（1）进货时入库登记，以便日后查看和统计进货情况；

（2）销售时把相应数量的商品从现存商品库中减去，并记录于另外的一个销售库中，以便日后查看和统计销售信息；

（3）是销售就难免会有顾客退货情况发生，因此还需建立一个相应的退货管理，把退回来的货加入现存商品库，再从销售库中减去，进行退货登记。

（4）还要能够随时查看各种信息表，如员工信息、进货信息、销售信息、退货信息。

（5）除了以上一些基本的需求外，实际中公司用来进行决策和管理分析的数据才是用得最多的数据。因此还需要有一些相应的关于以上数据的统计信息。如希望查看各个时间段（如今日、本月、本年度等）的进货和销售情况，以便对以后的运作进行一个准确的预测和正确的决断。

（6）为了调动员工的积极性，员工的薪水应该和他的销售情况挂钩的，因此还需要能够查看各员工的销售情况，以便对他们的业绩进行考核和评比。

以上数据如果用传统的方法管理，则显得繁琐且容易出错，透明度也不高，容易使人有机可乘，造成公司账目混乱。

本设计正是针对一般销售公司都有的以上需求而设计的，可以有效管理数据。主要功能有

交易管理，包括进货登记管理、销售登记管理、退货登记管理；进货统计管理，其中包括有关进货的各种统计数据表，如今日进货统计、本月进货统计、本季度进货统计、本年度销售统计等：销售统计管理，其中包括有关销售的各种统计数据表，如今日销售统计、本月销售统计、本季度销售统计、本年度销售统计等；员工销售业绩的考核，可以很方便地查看各员工的销售情况，以便对各员工的业绩进行考核比较。

2、内容：学生根据所学的数据库原理与程序设计的知识，针对销售管理信息系统，进行系统的需求分析，系统设计，数据库设计，编码，测试等，完成题目要求的功能，从而达到掌握开发一个小型数据库的目的。

3、操作过程：

1）进行系统调查，搜集资料。

2）进行系统需求分析和系统设计，写出系统分析和系统设计报告。

3）设计数据库模型并进行优化，确定数据库结构、功能结构和系统安全性和完整性要求。

3）完成数据库定义工作，实现系统数据的数据录入和数据处理。

4）实现应用程序的设计、编程、优化功能，实现数据安全性、数据完整性和并发控制技术等功能，并针对具体课题问题提出解决方法。

5）对系统的各个应用程序进行集成调试和测试，进一步优化系统性能，改善系统用户界面。

摘要

本次课设旨在设计和开发一款销售管理系统，以帮助企业实现销售流程的数字化、自动化管理，提升销售效率和客户满意度。本系统包括登录系统、交易管理、进货统计、销售统计、业绩和数据表查看和库存信息模块，涵盖了销售业务全流程的需求且功能齐全并且考虑到数据库安全性方面采用预编译与参数化查询技术可有效防止SQL注入。我首先进行了需求分析，明确了系统功能和性能需求，并根据我的测试不断进行完善和调整。随后选择了QT Creator和SQLserver等技术来实现系统的前后端开发和数据库存储。最终，我成功完成了销售管理系统的开发总的来说，本次销售管理系统课设为我们提供了宝贵的实践经验，让我们更深入地理解了软件开发的流程和技术应用，也提升了我们项目管理能力。希望我们的努力能为企业管理者带来实际的商业效益和见证我们的成长与进步。

Abstract.

This course aims to design and develop a sales management system to help enterprises realize digital and automated management of sales process, improve sales efficiency and customer satisfaction. The system includes login system, transaction management, purchase statistics, sales statistics, performance and data table view and inventory information modules, covering the needs of the whole process of sales business and complete functions, and considering the database security using precompilation and parametric query technology can effectively prevent SQL injection. I first carried out a requirement analysis, clarified the system function and performance requirements, and continuously improved and adjusted according to my tests. Then we choose QT Creator and SQLserver to realize the backend development and database storage of the system. Finally, I successfully completed the development of the sales management system. In general, the sales management system course provided us with valuable practical experience, which allowed us to have a deeper understanding of the process and technical application of software development, and also improved the ability of team cooperation and project management. We hope that our efforts can bring actual business benefits to enterprise managers and witness our growth and progress.

第一章 概述

* 1. 项目背景

无论是一个销售公司还是一个销售商店，都有很多的销售数据需要管理。手工管理这些繁琐复杂的数据方式已经不适应时代的潮流，实现数据的规范化、自动化的计算机管理，是一个管理规范、运作高效的企事业单位的必然要求。本课程设计就是为了实现给销售单位高效管理而设计的。

* 1. 编写目的

为了解决好这个销售管理系统，也更好的能够维护销售管理系统中出现的问题，同时也能够让用户能够正确的了解本销售管理系统。所以设计本系统。

本系统主要根据用户的需求做出分析，让用户更好的在线查看销售管理系统信息等，管理员后台管理系统数据等功能。从这个系统的操作来说，能够有效的进行信息的添加、修改、查询、删除一些销售管理系统信息，在一定的程序上能够实现了自动化。设计该系统的主要目的是为实现通过网络来减少人力和财力的投入，不断提高工作效率。最终我们希望通过销售管理系统可以达到提高销售管理系统的管理效率，实现管理上的井井有条。

* 1. 软件定义

无论是一个销售公司还是一个销售商店，都有很多的销售数据需要管理。手工管理这些繁琐复杂的数据方式已经不适应时代的潮流，实现数据的规范化、自动化的计算机管理，是一个管理规范、运作高效的企事业单位的必然要求。本课程设计就是为了实现给销售单位高效管理而设计的。

本设计总体的系统功能是实现销售公司销售管理的系统化、规范化和自动化。

* 1. 开发环境
     1. QT

Qt是一个1991年由Qt Company开发的跨平台C++图形用户界面应用程序开发框架。它既可以开发GUI程序，也可用于开发非GUI程序，比如控制台工具和服务器。Qt是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）以及一些宏，Qt很容易扩展，并且允许真正地组件编程。2008年，Qt Company科技被诺基亚公司收购，Qt也因此成为诺基亚旗下的编程语言工具。2012年，Qt被Digia收购。2014年4月，跨平台集成开发环境Qt Creator 3.1.0正式发布，实现了对于iOS的完全支持，新增WinRT、Beautifier等插件，废弃了无Python接口的GDB调试支持，集成了基于Clang的C/C++代码模块，并对Android支持做出了调整，至此实现了全面支持iOS、Android、WP,它提供给应用程序开发者建立艺术级的图形用户界面所需的所有功能。基本上，Qt 同 X Window 上的 Motif，Openwin，GTK 等图形界面库和 Windows 平台上的 MFC，OWL，VCL，ATL 是同类型的东西。

Qt使用“一次编写，随处编译”的方式为开放跨平台的图形用户界面应用提供了一个完整的C++应用程序开发框架。Qt允许程序开发人员使用应用程序中单一源代码树来构建可以运行在不同平台下的应用程序下的不同版本，这些平台包括从window98到vista、Mac Os X、Linux、HP-UX以及其他版本的基于X11的UNIX。

1、优良的跨平台特性:

Qt支持下列操作系统: Microsoft Windows 95/98， Microsoft Windows NT， Linux， Solaris， SunOS， HP-UX， Digital UNIX (OSF/1， Tru64)， Irix， FreeBSD， BSD/OS， SCO， AIX， OS390，QNX 等等。

2、面向对象

Qt 的良好封装机制使得 Qt 的模块化程度非常高，可重用性较好，对于用户开发来说 是非常方便的。 Qt 提供了一种称为 signals/slots 的安全类型来替代 callback，这使得 各个元件之间的协同工作变得十分简单。

3、丰富的 API

Qt 包括多达250个以上的 C++ 类，还提供基于模板的 collections， serialization， file， I/O device， directory management， date/time 类。甚至还 包括正则表达式的处理功能,支持 2D/3D 图形渲染，支持 OpenGL。

* + 1. SQL Server

SQL Server 是由微软公司开发的一种关系型数据库管理系统（RDBMS），用于存储和检索数据。它提供了一个可扩展的、安全的和可靠的数据存储和管理解决方案。

SQL Server 主要用于构建企业级应用程序，支持大规模数据处理，并拥有强大的数据分析和报告功能。它提供了一套强大的工具和功能，可以有效地管理和操作数据库，包括数据模型设计、数据导入和导出、事务处理、数据备份和恢复。

SQL Server 支持结构化查询语言（SQL），这是一种用于管理和操纵数据库的标准语言。通过使用 SQL，用户可以轻松地创建、修改和查询数据库中的表、视图、存储过程和触发器。

通过学习SQL server，可以深入了解SQL语言和SQL server数据库的基础知识，并学习数据安全性和防御机制的最佳实践。您将学习如何编写安全的查询语句、使用参数化查询和存储过程来防止SQL注入攻击。此外，了解SQL server的配置和权限管理也是确保数据库安全的重要方面。

第二章 需求分析

（1）系统登录：包括用户登录、添加用户、修改密码、退出系统等。

（2）交易管理：包括进货登记、销售登记、退货登记等。其中进货登记中还包含进货厂商的登记，如果是新的进货厂商，可以在此登记入库。进货和销售的主要交易都是在此进行：每次交易(包括进货、销售和退货)都需要对商品名、生产厂商、商品型号、单价、数量、总金额、交易日期、经手业务员编号等各项进行记录。

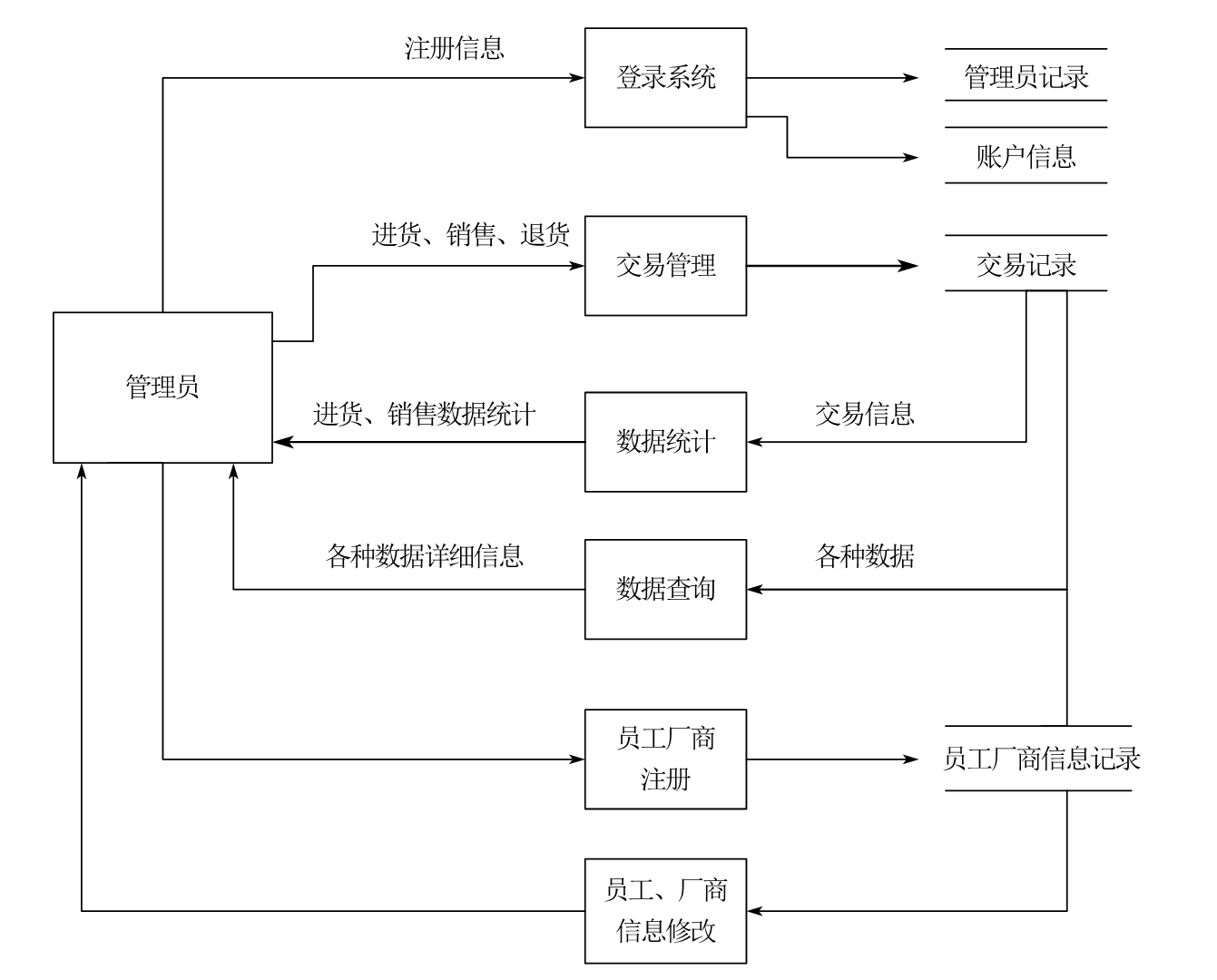
（3）进货统计：包括今日进货统计、本月进货统计、本季度进货统计、本年度进货统计等。其中每次进货统计都是按进货厂商和进货金额顺序排列的。使用一个表格显示所选时间段的全部进货数据，包括进货编号、商品名、生产厂商、产品型号、单价、数量、总金额、进货日期、业务员编号等；再使用另一表格显示分别从各个厂商进货的金额，还有一个文本框用于显示此时间段的总进货金额。

（4）销售统计：包括今日销售统计、本月销售统计、本季度销售统计、本年度销售统计等。其中每次销售统计都是按产品厂商和销售金额顺序排列的，用户可以一目了然地看出各种型号产品的销售额。使用一个表格显示所选时间段的全部销售数据，包括销售编号、商品名、生产厂商、产品型号、单价、数量、总金额、销售日期、业务员编号等；再使用另一表格分别显示各个厂商产品的销售金额，还有一个文本框用于显示此时间段的总销售金额。

（5）业绩查看：实现各员工销售情况的统计，可以一次性显示所有员工的销售情况，也可以根据员工编号查看各个员工的销售情况。

（6）查看数据表：实现各种数据表的浏览，包括进货表、销售表、退货表、员工表、进货商表。浏览时此数据表的所有信息项全部显示。如果是员工信息或进货商信息有变化，还可以对员工表和进货商表进行修改，其他的表则不能进行修改，防止有人为造假数据。

* 1. 数据流图



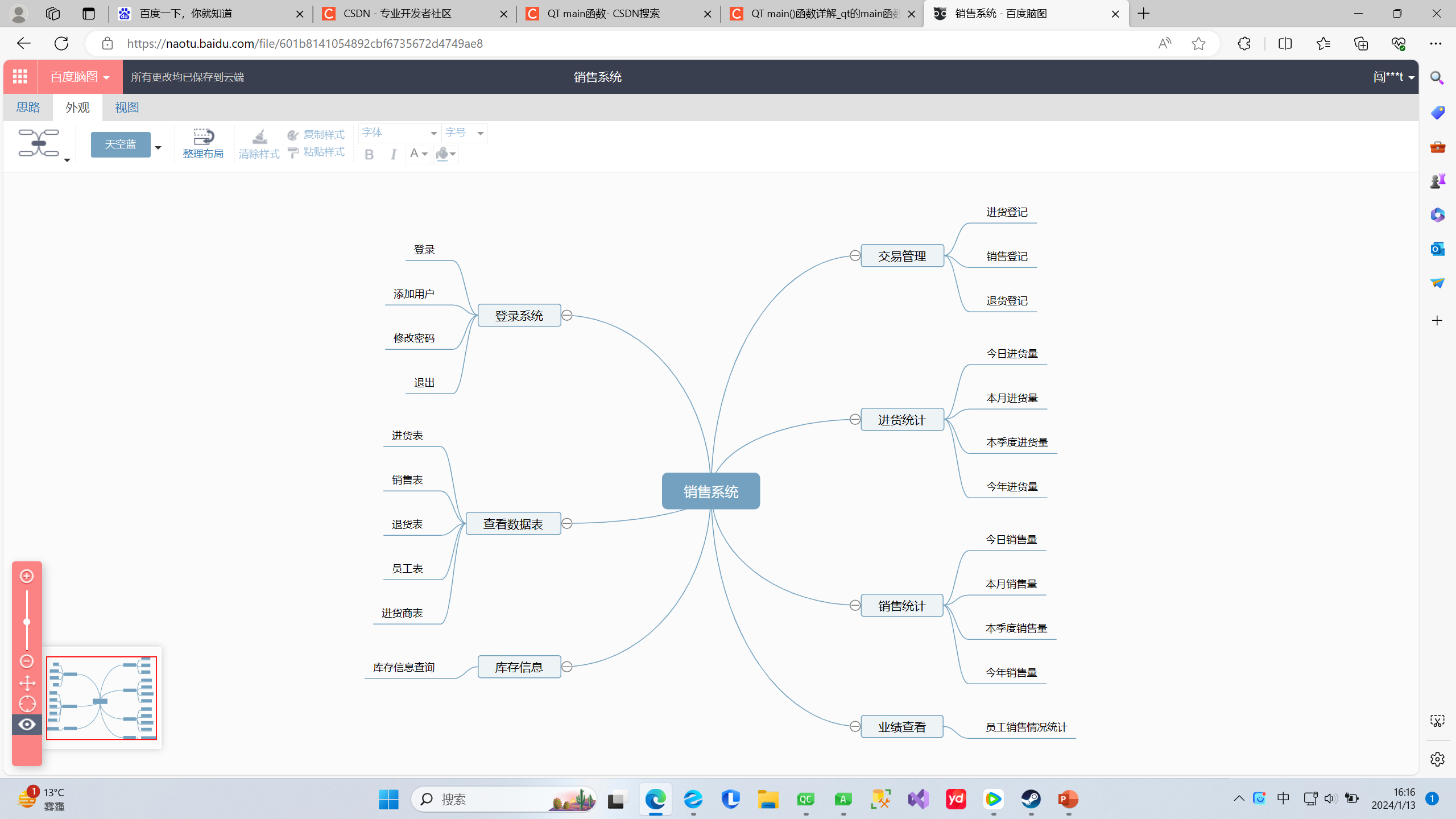
1. 数据流图

第三章 软件概要设计

* 1. 系统目标

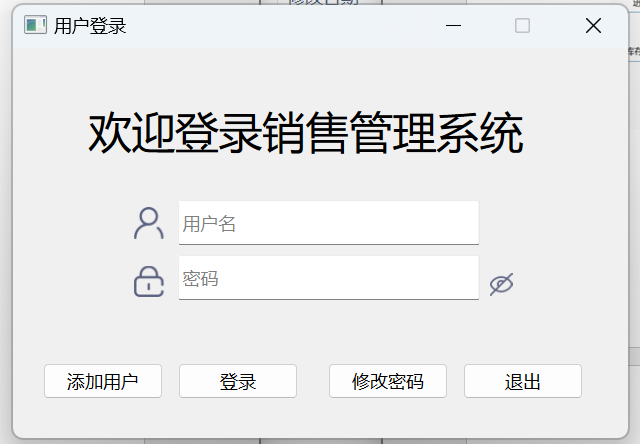
完成任务书的所有任务。

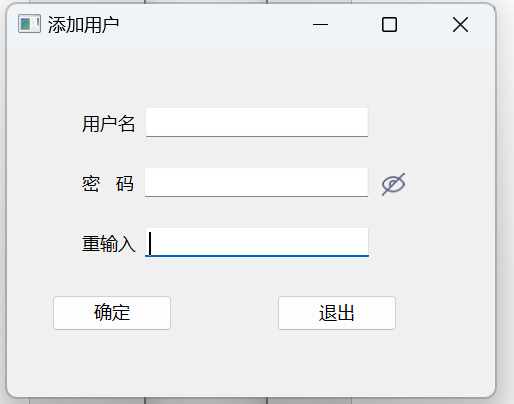
* 1. 软件功能图

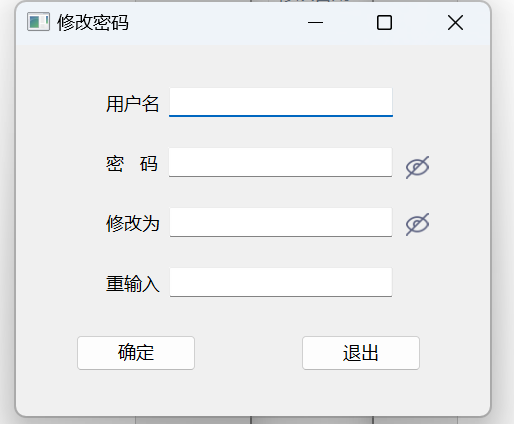


1. 软件功能图
2. 登录系统

包括用户登录、添加用户、修改密码、退出系统等。

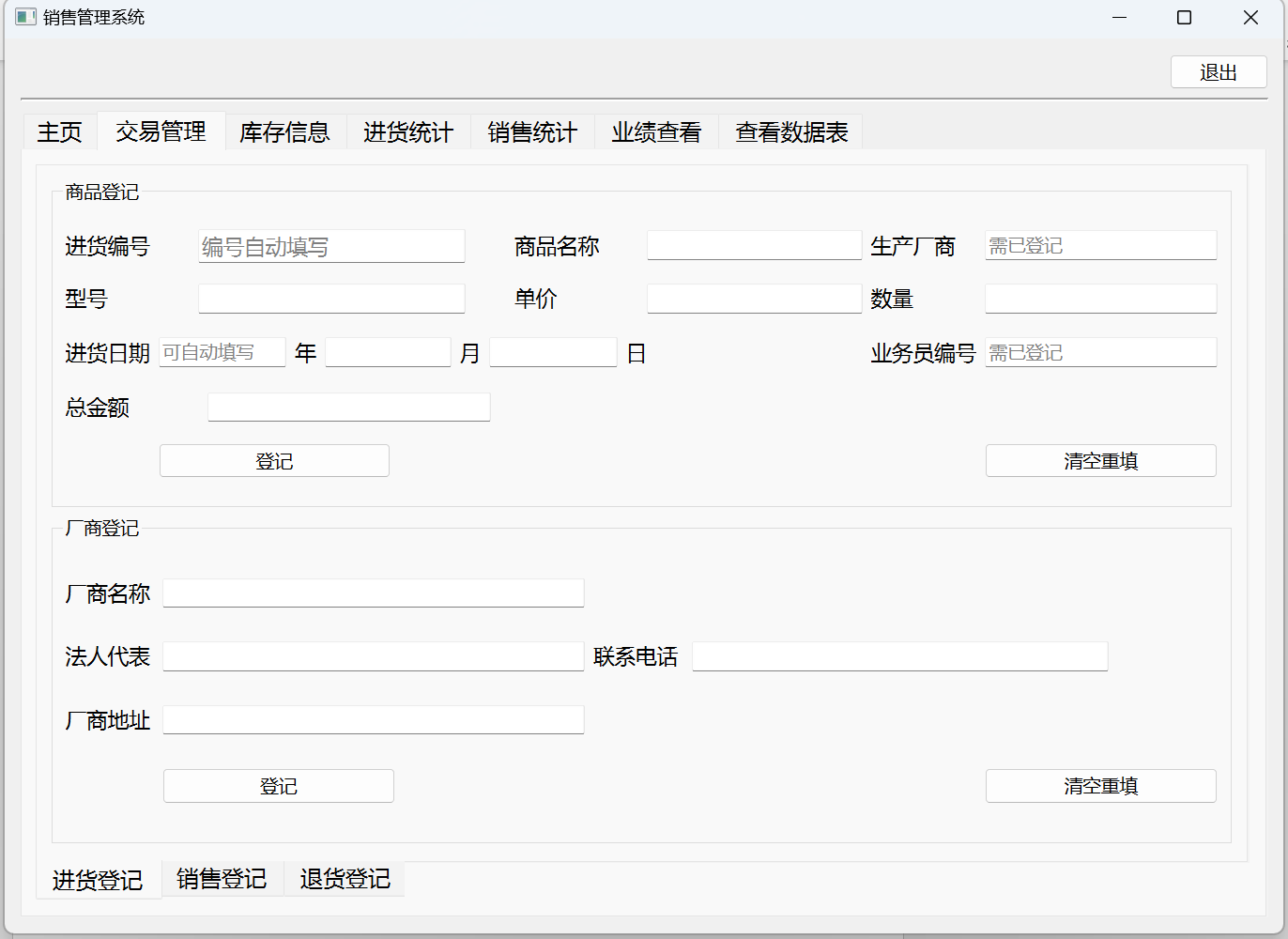






2.交易管理

包括进货登记、销售登记、退货登记等。其中进货登记中还包含进货厂商的登记，如果是新的进货厂商，可以在此登记入库。进货和销售的主要交易都是在此进行：每次交易(包括进货、销售和退货)都可对商品名、生产厂商、商品型号、单价、数量、总金额、交易日期、经手业务员编号等各项进行记录。



1. 库存信息查询

可查询库存信息表中的所有信息，并且可以通过界面内的信息筛选框筛选出指定条件条目。



3.进货统计

包括今日进货统计、本月进货统计、本季度进货统计、本年度进货统计等。其中每次进货统计都是按进货厂商和进货金额顺序排列的。使用一个表格显示所选时间段的全部进货数据，包括进货编号、商品名、生产厂商、产品型号、单价、数量、总金额、进货日期、业务员编号等；再使用另一表格显示分别从各个厂商进货的金额，还有一个文本框用于显示此时间段的总进货金额。



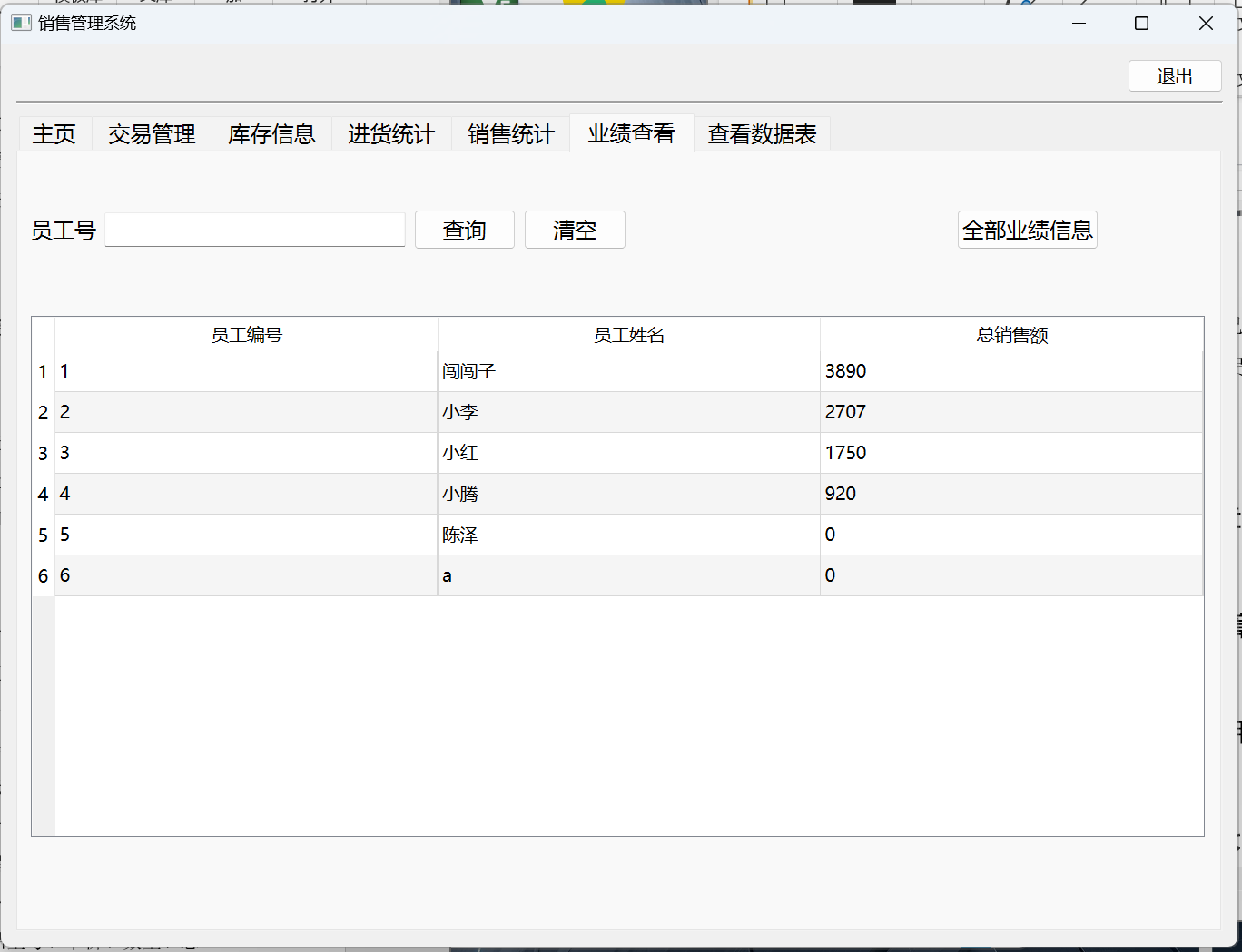
4.销售统计

包括今日销售统计、本月销售统计、本季度销售统计、本年度销售统计等。其中每次销售统计都是按产品厂商和销售金额顺序排列的，用户可以一目了然地看出各种型号产品的销售额。使用一个表格显示所选时间段的全部销售数据，包括销售编号、商品名、生产厂商、产品型号、单价、数量、总金额、销售日期、业务员编号等；再使用另一表格分别显示各个厂商产品的销售金额，还有一个文本框用于显示此时间段的总销售金额。



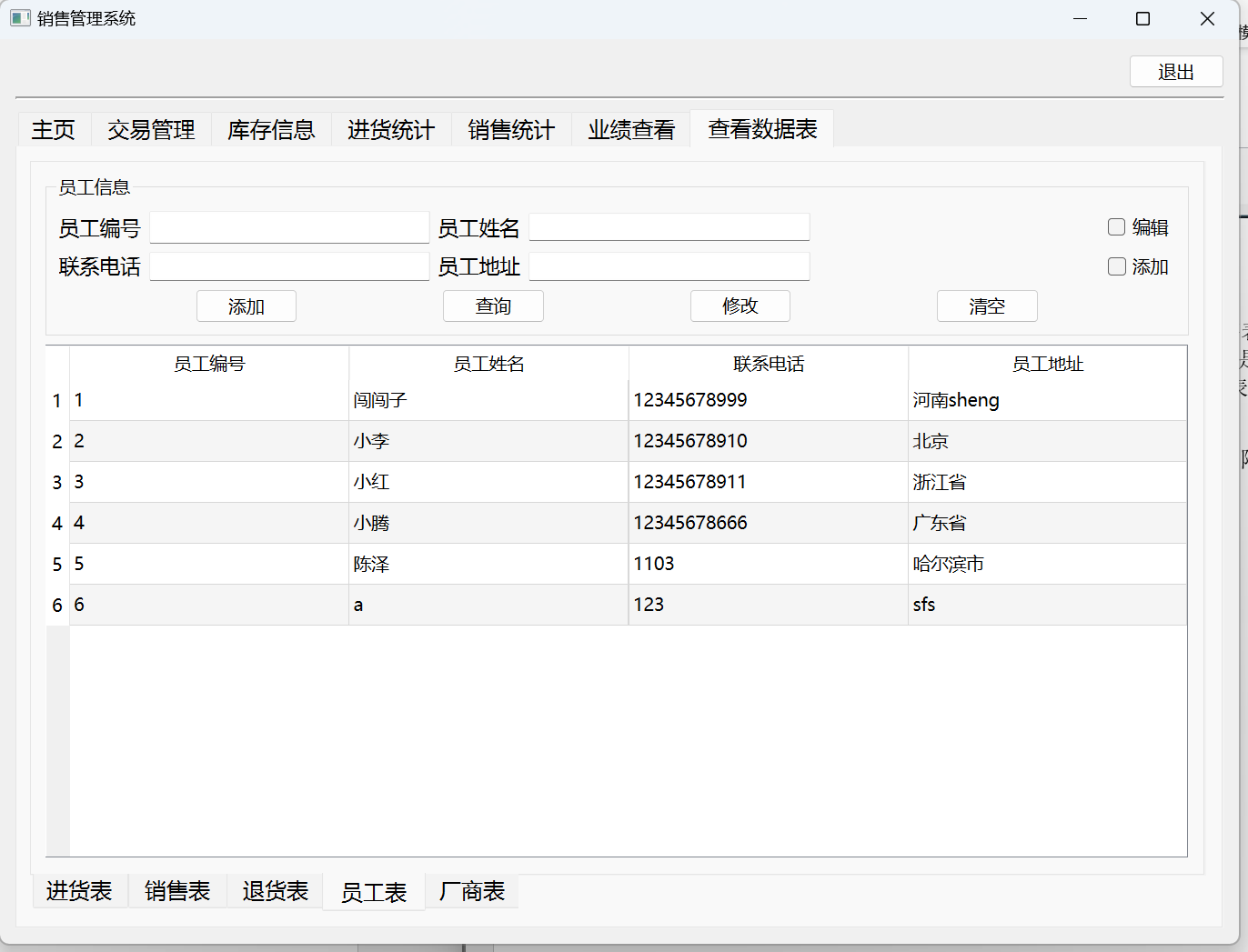
5.业绩查看

业绩查看：实现各员工销售情况的统计，可以一次性显示所有员工的销售情况，也可以根据员工编号查看各个员工的销售情况。



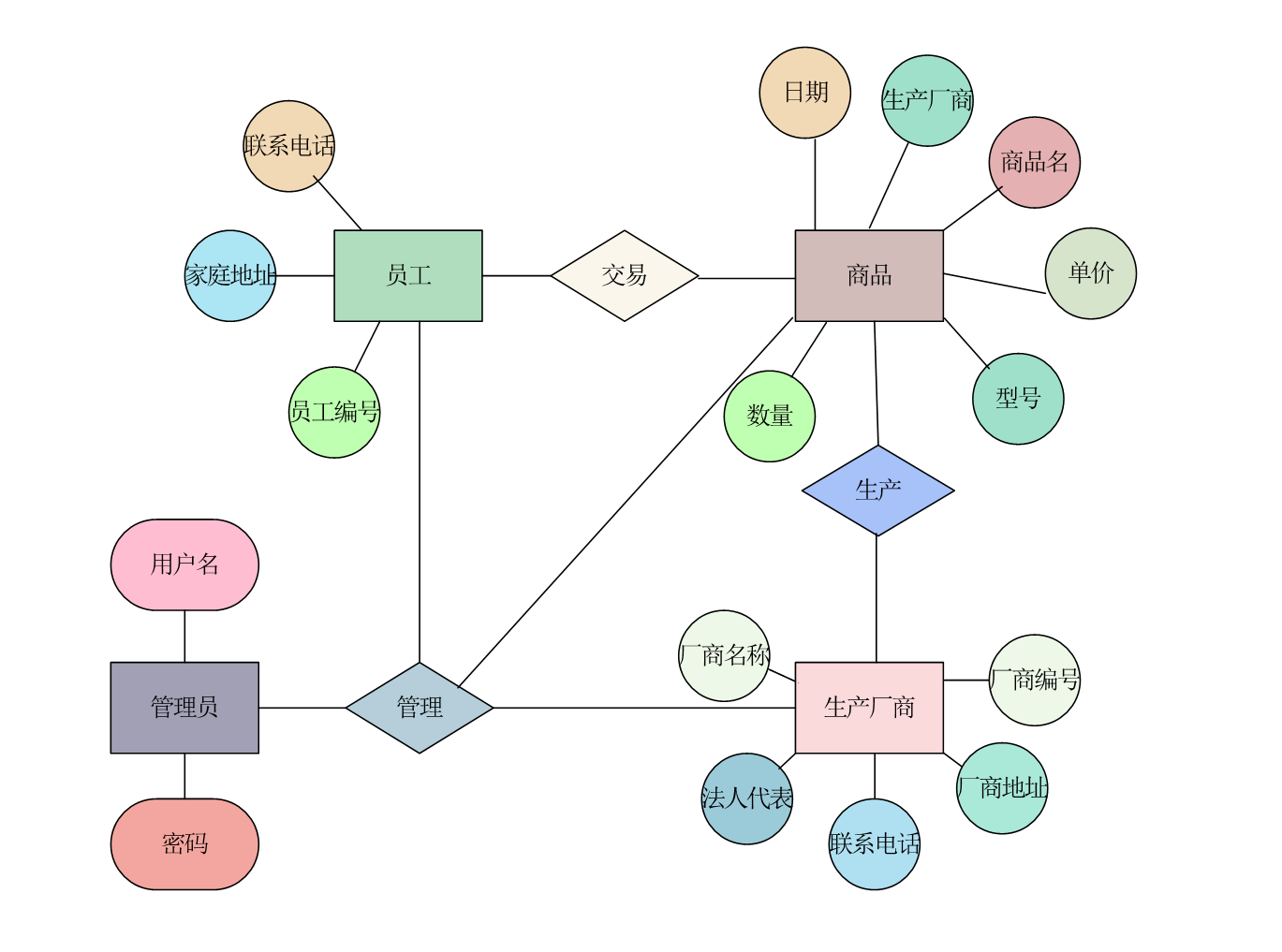
6.查看数据表

查看数据表：实现各种数据表的浏览，包括进货表、销售表、退货表、员工表、进货商表。浏览时此数据表的所有信息项全部显示。如果是员工信息或进货商信息有变化，还可以对员工表和进货商表进行修改，其他的表则不能进行修改。



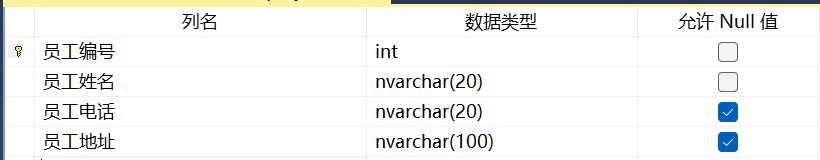
第四章 数据库设计

* 1. ER关系图



1. ER关系图
   1. 数据库表的创建

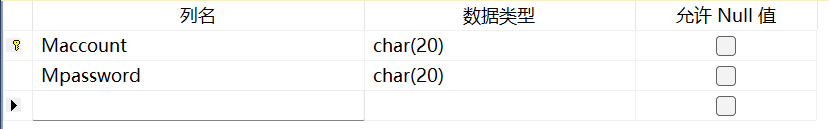
Employee表



Goods表



Manager表



Manufacture表



Retreat表



Sell表



Stock表



* + 1. 系统用户表

主键编号自增，用户依靠用户名和密码登录，用户名称应加唯一约束，或将编号取消将用户名称设为唯一约束

dbcc checkident(dbo.book,reseed,5),更新identity的值

creat table Manager  
(Msnb int identity(1,1) primary key,--identity(a,b),本行自动增加开始为a增幅为b  
Maccont char(20) primary key not null,  
Mpassword char(20) not null  
)   
​

* + 1. 员工信息表

use sellsystem  
go  
create table employee  
(  
员工编号 int identity(1,1) primary key not null,  
员工姓名 nvarchar(20) collate chinese\_prc\_ci\_as not null,  
员工电话 nvarchar(20) collate chinese\_prc\_ci\_as null,  
员工地址 nvarchar(100) collate chinese\_prc\_ci\_as null,  
)  
go

* + 1. 进货厂商信息表

constraint[pk\_manufacturer] primary key clustered  
(  
厂商名称  
)on primary  
​  
constraint[ix\_manufacturer\_1] unique nonclustered  
(  
厂商名称  
)on primary

* + 1. 进货信息表

use sellsystem  
go  
create table goods  
(  
商品编号 int identity(1,1) primary key not null,  
生产厂商 nvarchar(20) collate chinese\_prc\_ci\_as null,  
商品名 nvarchar(20) collate chinese\_prc\_ci\_as null,  
型号 nvarchar(20) collate chinese\_prc\_ci\_as null,  
单价 money null constraint[DF\_goods\_单价] default(0),  
数量 numeric(10,0) null constraint[DF\_goods\_数量] default(0),  
总金额 money null,  
进货年 smallint null constraint[DF\_goods\_进货年] default(datepart(year,getdate())),  
进货月 smallint null constraint[DF\_goods\_进货月] default(datepart(month,getdate())),  
进货日 smallint null constraint[DF\_goods\_进货日] default(datepart(day,getdate())),  
业务员编号 int not null,  
constraint[fk\_goods\_employee] foreign key  
(  
业务员编号  
)references[employee]  
(  
员工编号  
),  
constraint[fk\_goods\_manufacturer] foreign key  
(  
生产厂商  
)references[manufacturer]  
(  
厂商名称  
)  
)  
go

* + 1. 销售信息表

use sellsystem  
go  
create table sell  
(  
商品编号 int identity(1,1) primary key not null,  
生产厂商 nvarchar(20) collate chinese\_prc\_ci\_as not null,  
商品名 nvarchar(20) collate chinese\_prc\_ci\_as null,  
型号 nvarchar(20) collate chinese\_prc\_ci\_as null,  
单价 money null constraint[DF\_sell\_单价] default(0),  
数量 int null constraint[DF\_sell\_数量] default(0),  
总金额 money null constraint[DF\_sell\_总金额] default(0),  
销售年 smallint null constraint[DF\_sell\_进货年] default(datepart(year,getdate())),  
销售月 smallint null constraint[DF\_sell\_进货月] default(datepart(month,getdate())),  
销售日 smallint null constraint[DF\_sell\_进货日] default(datepart(day,getdate())),  
业务员编号 int not null,  
constraint[fk\_sell\_employee] foreign key  
(  
业务员编号  
)references[employee]  
(  
员工编号  
),  
constraint[fk\_sell\_manufacturer] foreign key  
(  
生产厂商  
)references[manufacturer]  
(  
厂商名称  
)  
)  
go

* + 1. 退货信息表

use sellsystem  
go  
create table retreat  
(  
退货编号 int identity(1,1) primary key not null,  
厂商 nvarchar(20) collate chinese\_prc\_ci\_as null,  
商品名 nvarchar(20) collate chinese\_prc\_ci\_as not null,  
型号 nvarchar(20) collate chinese\_prc\_ci\_as null,  
单价 money null constraint[DF\_retreat\_单价] default(0),  
数量 int null constraint[DF\_retreat\_数量] default(0),  
总金额 money null constraint[DF\_retreat\_总金额] default(0),  
退货年 smallint null,  
退货月 smallint null,  
退货日 smallint null,  
业务员编号 int not null,  
constraint[fk\_retreat\_employee] foreign key  
(  
业务员编号  
)references[employee]  
(  
员工编号  
),  
constraint[fk\_retreat\_manufacturer] foreign key  
(  
厂商  
)references[manufacturer]  
(  
厂商名称  
)  
)  
go

* + 1. 现货信息表

use sellsystem  
go  
create table stock  
(  
商品编号 int identity(1,1) primary key not null,  
生产厂商 nvarchar(20) collate chinese\_prc\_ci\_as null,  
商品名 nvarchar(20) collate chinese\_prc\_ci\_as not null,  
型号 nvarchar(20) collate chinese\_prc\_ci\_as null,  
  单价 money null constraint[DF\_goods\_单价] default(0),  
数量 numeric(10,0) null constraint[DF\_goods\_数量] default(0),  
修改年 smallint null,  
修改月 smallint null,  
修改日 smallint null,  
)  
go

第五章 详细设计

* 1. 连接SQLserver数据库
     1. 配置ODBS数据源

ODBC(Open DataBase Connectivity，开放数据库连接）是一种开放的标准接口，用于通过SQL（Structured Query Language，结构化查询语言）访问数据库，并提供了跨平台、跨数据库的数据访问能力。ODBC数据源是ODBC标准定义的一种数据源，它允许应用程序通过0DBC驱动程序连接和访问各种类型的数据存储。

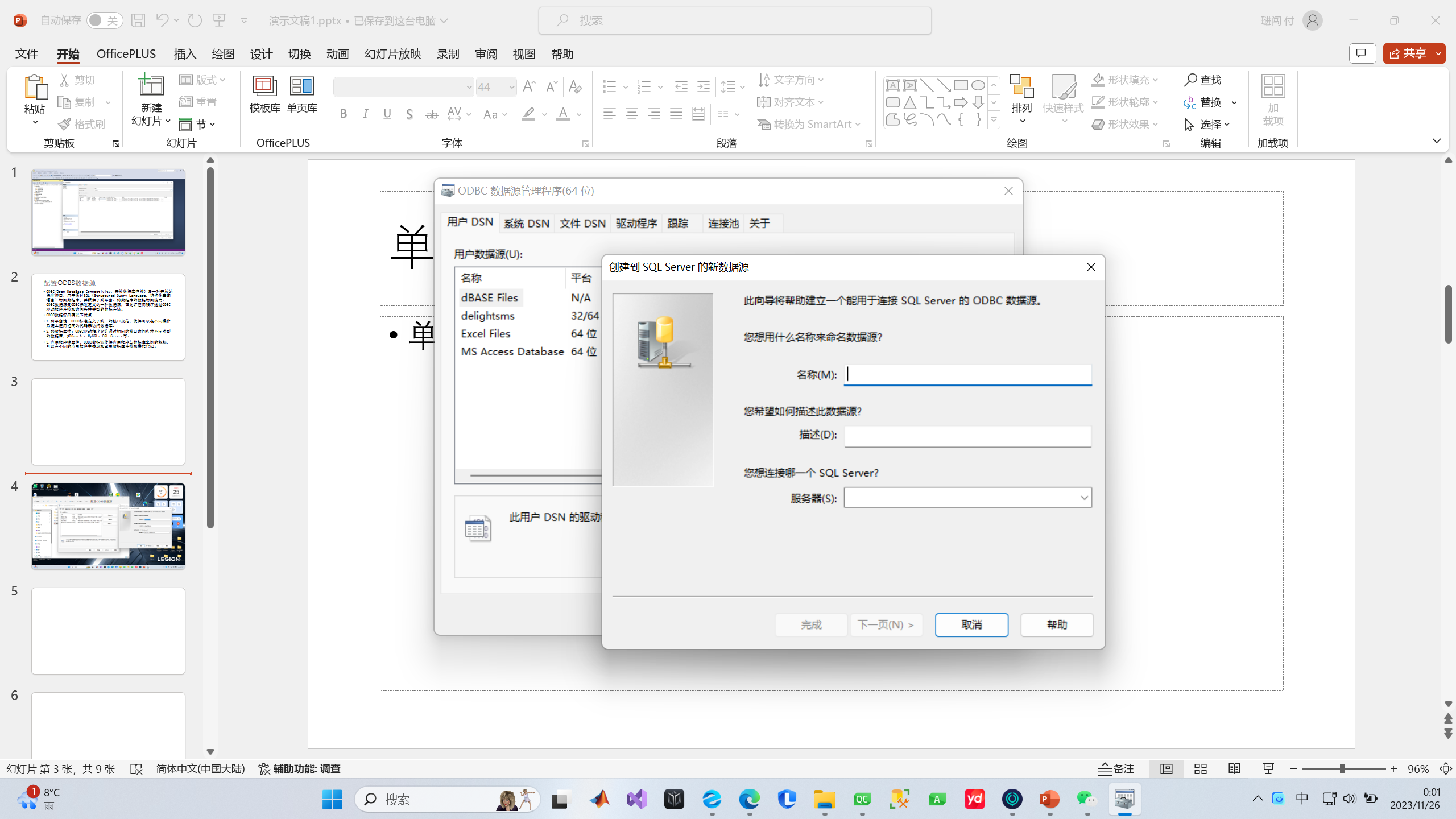
ODBC数据源具有以下优点：

1.跨平台性：ODBC标准定义了统一的接口规范，使得可以在不同操作系统上使用相同的代码来访问数据库。

2.跨数据库性：ODBC驱动程序允许通过相同的接口访问多种不同类型的数据库，如Orac1e、MySQL、SQL Server等。

3.应用程序独立性：ODBC数据源使得应用程序与数据库之间的解耦，可以在不同的应用程序中共享和重用数据库连接和操作代码。

在创建完数据库后打开ODBC数据源管理程序，创建一个能用于连接SQL Server的ODBC数据源

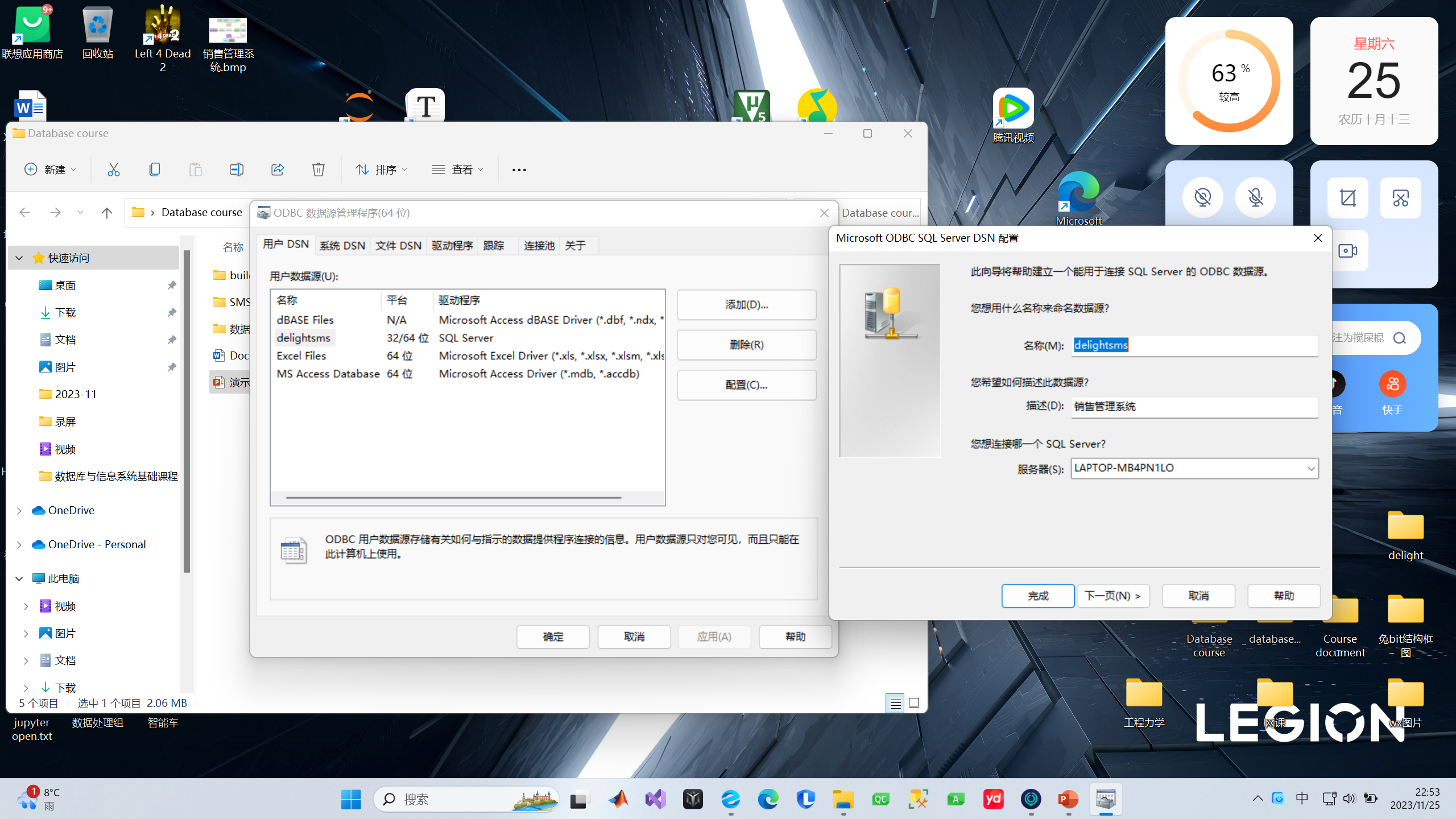


数据源名称（可任意）

服务器（SQLserver连接的服务器）

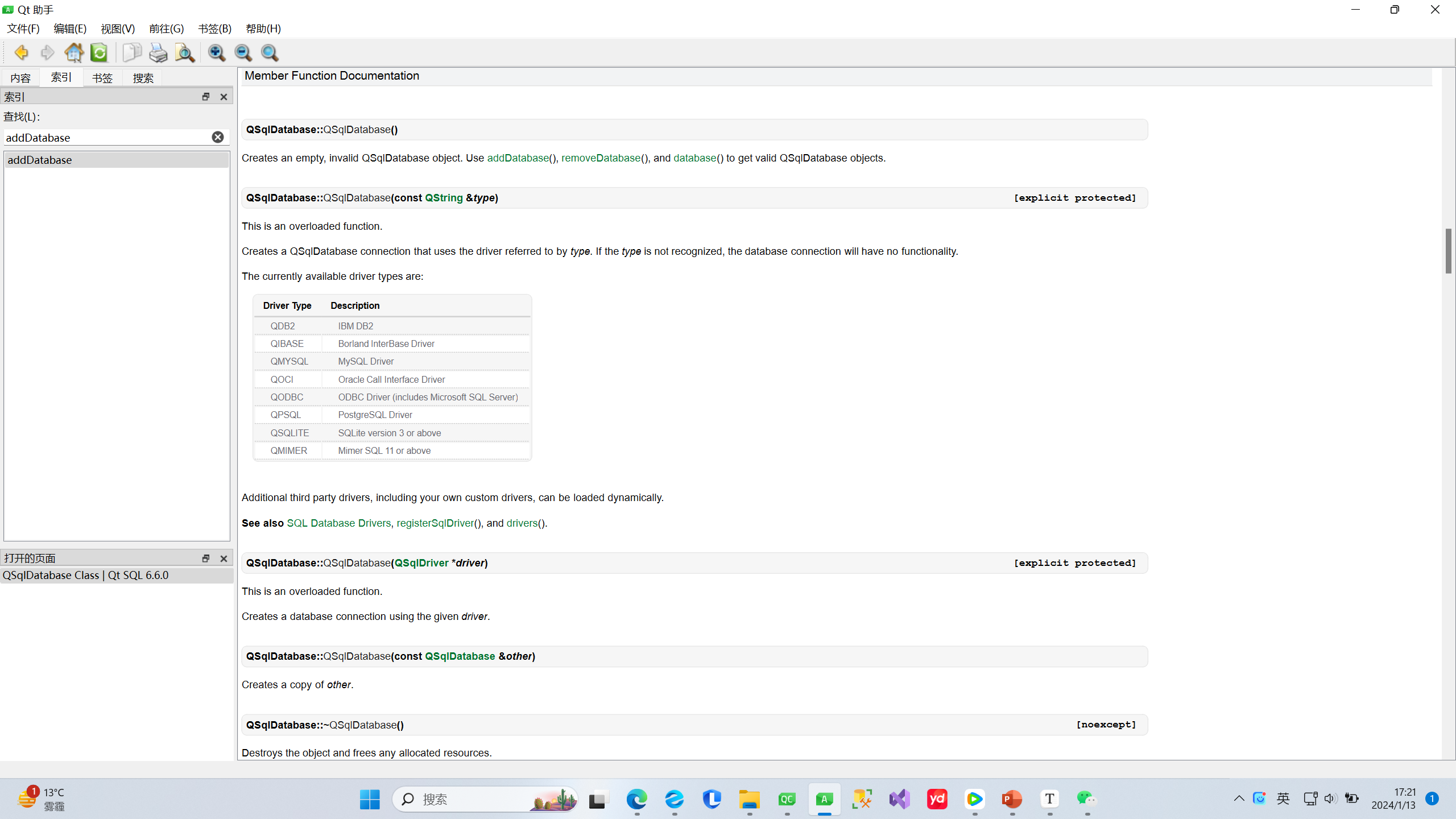
Sqlserver中需要连接的数据库

选择sqlserver用户密码验证（SSMS中添加，且有访问数据库的权限）



* + 1. QT连接数据源配置

QT支持很多种数据库，以下为具体支持的数据库类型



QT程序连接数据库需创建一个使用按类型引用的驱动程序的 QSqlDatabase 连接。如果无法识别类型，则数据库连接将不起作用。即实例化一个QSqlDatabase对象

代码如下

   QSqlDatabase db = QSqlDatabase::addDatabase("QODBC");  
   db.setHostName("LAPTOP-MB4PN1LO"); //服务器  
   db.setDatabaseName("delightsms");  
   db.setUserName("delight");  
   db.setPassword("123456");

其中用户名和密码是在SSMS中创建的用户名和密码，且用户名和密码必须是被授予权限能够访问此ODBC数据源的用户名和密码

* 1. QT如何访问数据库

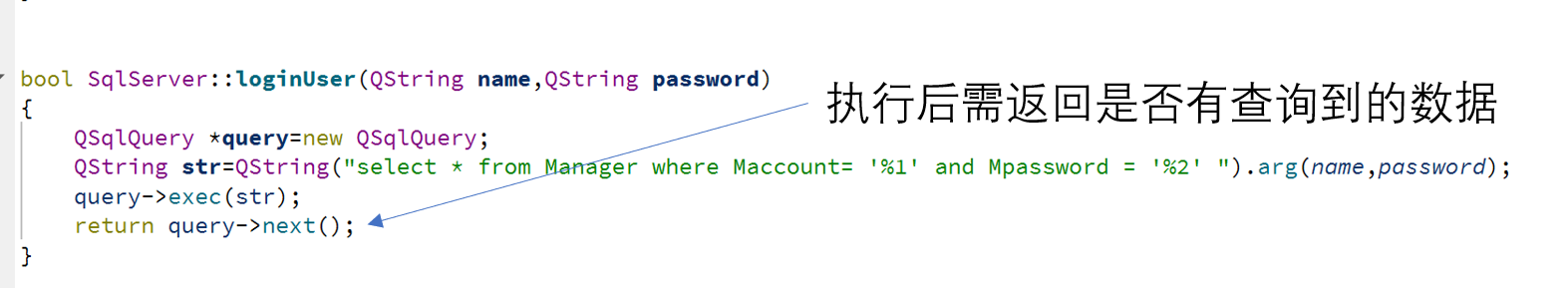
QSqlQuery 类提供了一种执行和操作 SQL 语句的方法。

QSqlQuery 封装了在 QSqlDatabase 上执行的 SQL 查询中创建、导航和检索数据所涉及的功能。它可用于执行 DML（数据操作语言）语句，如 SELECT、INSERT、UPDATE 和 DELETE，以及 DDL（数据定义语言）语句，如 CREATE TABLE。它还可用于执行非标准 SQL 的特定于数据库的命令。

成功执行的 SQL 语句将查询的状态设置为活动（isActive() 返回 true）。

使用以下功能执行**导航记录**：

next() previous() first() last() seek()



* 1. 登录系统

包括用户登录、添加用户和修改密码

* 1. SQL注入攻击

1.SQL注入即是指应用程序对用户输入数据的合法性没有判断或过滤不严，攻击者可以在应用程序中事先定义好的查询语句的结尾上添加额外的SQL语句，在管理员不知情的情况下实现非法操作，以此来实现欺骗数据库服务器执行非授权的任意查询，从而进一步得到相应的数据信息。

2.用拼接字符串方式构造SQL语句比较容易犯sql注入问题

一般sql注入攻击主要是修改原sql语句的判断条件。例如减少sql语句判断条件，或增加一个绝对true的语句，使sql最终执行结果不为空。

例如：

1.通过#注释掉一个判断条件，只要有用户名就可以登录。

用户名’#’

2.通过增加一个true的条件，来屏蔽掉密码的判断条件。

用户名’ or '1=1

3.通过增加一个true的条件，不用输入任何信息也可以登录。

’ or '1=1

以上这些注入攻击方式都改变了sql语句原本的判断条件，即修改了sql语句的骨架。

数字注入

字符串类型注入

* 1. QT中可使用预编译来防止sql注入

PreparedStatement对象创建的一个必要条件就是先传输一个sql骨架，输入参数用？占位符替代。这样sql语句的骨架已确定（比如判断条件中的and，or…已固定），任何由外部注入的，使sql语句骨架发生变化的操作都会判定为无效输入。从而防止了sql注入攻击。

* + 1. 登录



当用户点击“登录”按钮时，触发该按钮与clicked信号所绑定的槽函数，在槽函数中需先获取用户输入在输入框中的用户名和密码，代码如下，使用字符串来接收

QString name=ui->let\_name->text();

QString password=ui->let\_passwd->text();

同时可判断是否输入为空，提醒用户输入完整

if(name==nullptr || password==nullptr)

{

QMessageBox::information(this,"登录","账号或密码输入为空\n请输入完整");

}

else if(sql->loginUser(name,password))

{

Widget \*w=new Widget();

w->show();

this->hide();

}

else

{

QMessageBox::information(this, "登录", "账号或密码输入错误\n请重新输入");

}

其中sql是自定义的sqlserver类的一个对象，用于执行特定的数据库操作

class SqlServer

{

public:

SqlServer();

bool initSql1();

bool initSql(QString servername,QString databasename,QString usename,QString password);

bool registerUser(QString name,QString password);

bool loginUser(QString name,QString password);

bool insertStock(QString mc,QString cs,QString xh,QString dj,QString sl,QString year,QString moon,QString day,QString ywybh,QString zje);

bool chpd(QString name,QString password,QString password1);

bool insertmfr(QString mfrname,QString faren,QString number,QString mfradd);

bool insertsell(QString cs,QString mc,QString xh,QString dj,QString sl,QString zje,QString year,QString moon,QString day,QString ywybh);

bool retreat(QString cs,QString mc,QString xh,QString dj,QString sl,QString zje,QString year,QString moon,QString day,QString ywybh);

bool jhinsertstock(QString sccs,QString spmc,QString xh,QString sl,QString year,QString moon,QString day);

QSqlQuery \*query;

QSqlDatabase db;

};

bool SqlServer::loginUser(QString name,QString password)

{

QSqlQuery \*query=new QSqlQuery;

QString str=QString("select \* from Manager where Maccount= '%1' and Mpassword = '%2'").arg(name,password);

query->exec(str);

return query->next();

// QSqlQuery \*query=new QSqlQuery;

// query->prepare("select \* from Manager where Maccount= ? and Mpassword = ?");//限定查询结构

// query->addBindValue(name);

// query->addBindValue(password);

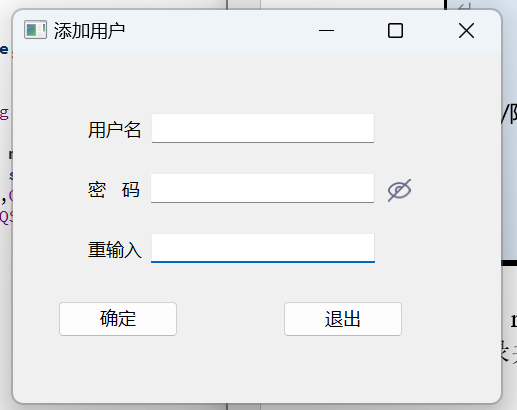
// query->exec();

// return query->next();

}

return query->next();此函数会返回查询结果的下一条记录，有记录为true即可登录并跳转到主界面，没有记录则为false提示用户登录失败

* + 1. 添加用户



以下为“确定”按钮和click信号连接的槽函数的逻辑代码，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作

void regis::on\_btn\_reg\_clicked()

{

SqlServer \*sql=new SqlServer;

QString name=ui->let\_name->text();

QString password1=ui->let\_passwd1->text();

QString password2=ui->let\_passwd2->text();

int i=QString::compare(password1,password2);

if(name==nullptr||password1==nullptr||password2==nullptr)

{

QMessageBox::information(this,"注册","注册失败，请输入完整");

delete sql;

return;

}

else if(i)

{

QMessageBox::information(this,"注册","注册失败，两次密码输入不同");

delete sql;

return;

}

else if(sql->registerUser(name,password1))

{

QMessageBox::information(this,"注册","注册成功");

inlog \*w=new inlog();

w->show();

this->hide();

delete sql;

return;

}

else

{

QMessageBox::critical(this,"注册","注册失败，账户已存在");

delete sql;

return;

}

}

以下为添加用户数据库操作部分代码，实例化一个QSqlQuery类的指针对象，通过该对象执行sql插入语句，将用户名和密码插入到管理员表中

bool SqlServer::registerUser(QString name,QString password)

{

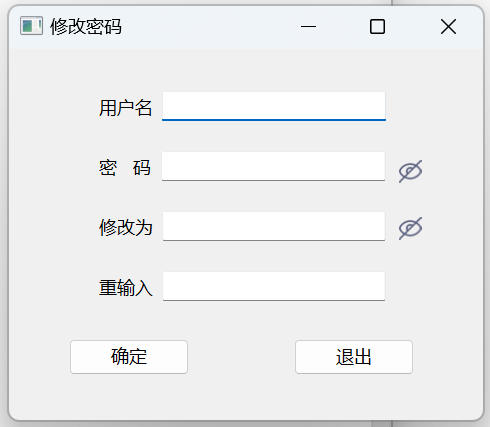
QSqlQuery \*query=new QSqlQuery;

QString str = QString("insert into Manager(Maccount,Mpassword) values('%1','%2')").arg(name,password);

return query->exec(str);

}

* + 1. 修改密码



1. 修改密码

以下为“确定”按钮和clicked信号连接的槽函数的代码实现，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作，首先查询账号密码在数据库中是否有记录，无记录则无法修改。

void chpd::on\_btn\_reg\_clicked()

{

SqlServer \*sql=new SqlServer;

QString name=ui->let\_name->text();

QString password1=ui->let\_passwd1->text();

QString password2=ui->let\_passwd2->text();

QString password3=ui->let\_passwd3->text();

int i=QString::compare(password2,password3);

if(name==nullptr || password1==nullptr || password2==nullptr || password3==nullptr)

{

QMessageBox::information(this,"修改密码","请输入完整");

delete sql;

return;

}

else if(i)

{

QMessageBox::information(this,"修改密码","两次密码输入不同");

delete sql;

return;

}

else if(sql->loginUser(name,password1))

{

if(sql->chpd(name,password1,password2))

{

QMessageBox::information(this,"修改密码","修改成功");

inlog \*w=new inlog();

w->show();

this->hide();

delete sql;

return;

}

else

{

QMessageBox::critical(this,"修改密码","修改失败");

}

}

else

{

QMessageBox::critical(this,"修改密码","修改失败,账号或密码输入错误");

delete sql;

return;

}

}

以下为修改密码数据库操作部分代码，实例化一个QSqlQuery类的指针对象，通过该对象执行修改信息的sql语句，并返回执行结果。

bool SqlServer::chpd(QString name,QString password,QString password1)

{

QSqlQuery \*query=new QSqlQuery;

QString str=QString("update Manager set Mpassword='%1' where Maccount= '%2' and Mpassword ='%3'").arg(password1,name,password);

return query->exec(str);

}

* 1. 交易管理

包括进货登记、销售登记和退货登记

* + 1. 进货登记



1. 进货登记

进货登记中还包含进货厂商的登记，如果是新的进货厂商，可以在此登记入库。

以下为商品登记组“登记”按钮和clicked信号连接的槽函数的代码实现，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作，将条目信息插入到进货表与库存表中。

void Widget::on\_btn\_addw\_clicked()

{

QString mc=ui->let\_mc->text();

QString cs=ui->let\_cs->text();

QString xh=ui->let\_xh->text();

QString dj=ui->let\_dj->text();

QString sl=ui->let\_sl->text();

QString year=ui->let\_year->text();

QString moon=ui->let\_moon->text();

QString day=ui->let\_day->text();

QString ywybh=ui->let\_ywybh->text();

QString zje=ui->let\_zje->text();

if(mc==nullptr||cs==nullptr|| xh==nullptr|| dj==nullptr || sl==nullptr || ywybh==nullptr || zje==nullptr)

{

QMessageBox::information(this,"进货登记","必填项未填写完整");

}

else

{

if(year==nullptr && moon==nullptr && day==nullptr)

{

int ok=QMessageBox::question(this,"进货登记","确认登记");

if(ok)

{

QDateTime datetime = QDateTime::currentDateTime();

year=QString("%1").arg(datetime.date().year());

moon=QString("%1").arg(datetime.date().month());

day=QString("%1").arg(datetime.date().day());

SqlServer \*sql=new SqlServer();

if(sql->insertStock(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh) && sql->jhinsertstock(cs,mc,xh,sl,year,moon,day))

{

QMessageBox::information(this,"进货登记","登记成功");

}

else

{

QMessageBox::critical(this,"进货登记","登记失败");

}

}

else

{}

}

else if((!ui->let\_year->text().isEmpty())&&(!ui->let\_moon->text().isEmpty())&&(!ui->let\_day->text().isEmpty()))

{

int ok=QMessageBox::question(this,"进货登记","确认登记");

if(ok)

{

SqlServer \*sql=new SqlServer();

if(sql->insertStock(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh) && sql->jhinsertstock(cs,mc,xh,sl,year,moon,day))

{

QMessageBox::information(this,"进货登记","登记成功");

}

else

{

QMessageBox::critical(this,"进货登记","登记失败");

}

}

}

else

{

QMessageBox::information(this,"进货登记","日期请填写完整");

}

}

}

以下为进货登记数据库操作部分代码，实例化一个QSqlQuery类的指针对象，通过该对象执行修改信息的sql语句，并返回执行结果。

bool SqlServer::insertStock(QString cs,QString mc,QString xh,QString dj,QString sl,QString zje,QString year,QString moon,QString day,QString ywybh)

{

QSqlQuery \*query=new QSqlQuery;

QString str = QString("insert into goods values('%1', '%2', '%3','%4','%5','%6','%7','%8','%9','%10')").arg(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh);

if(!query->exec(str))

return false;

return true;

}

厂商登记部分逻辑代码与进货登记类似

以下为厂商登记组“登记”按钮和clicked信号连接的槽函数的代码实现，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作，将条目信息插入到指定数据表中。

void Widget::on\_btn\_addmfr\_clicked()

{

QString mfrname=ui->let\_mfrname->text();

QString farname=ui->let\_faren->text();

QString number=ui->let\_number->text();

QString mfradd=ui->let\_mfradd->text();

SqlServer \*sql=new SqlServer();

if(mfrname==nullptr || farname==nullptr || number==nullptr || mfradd==nullptr )

{

QMessageBox::information(this,"厂商登记","请填写完整");

delete sql;

}

else

{

int ok=QMessageBox::question(this,"厂商登记","确认登记");

if(ok)

{

if(sql->insertmfr(mfrname,farname,number,mfradd))

{

QMessageBox::information(this,"厂商登记","登记成功");

}

else

{

QMessageBox::critical(this,"进货登记","登记失败");

}

}

delete sql;

}

}

以下为厂商登记数据库操作部分代码，实例化一个QSqlQuery类的指针对象，通过该对象执行修改信息的sql语句，并返回执行结果。

bool SqlServer::insertmfr(QString mfrname,QString faren,QString number,QString mfradd)

{

QSqlQuery \*query=new QSqlQuery;

QString str = QString("insert into manufacturer values('%1', '%2', '%3','%4')").arg(mfrname,faren,number,mfradd);

if(!query->exec(str))

{

return false;

}

else

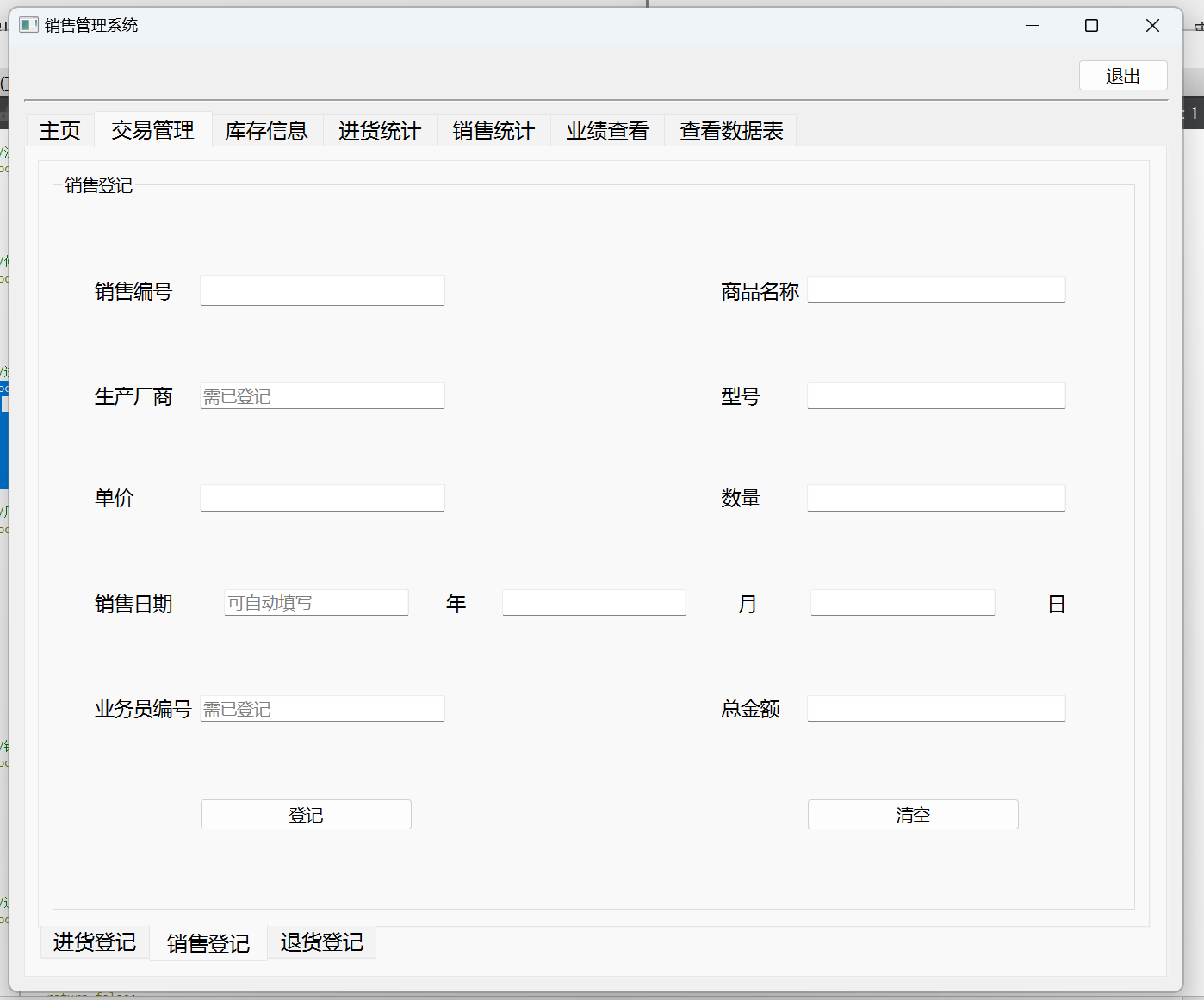
{

return true;

}

}

* + 1. 销售登记



1. 销售登记

销售登记以下为销售登记组“登记”按钮与clicked信号连接的槽函数的代码实现

首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作

因销售表创建时生产厂商与业务员编号与厂商表和业务员信息表存在外键约束，所以需首先需按照用户输入的厂商名称与业务员编号分别在厂商表与员工表中查询，若无记录则不可登记，并提示用户原因。

此函数存在自动填写日期的功能，默认填写的是今日时间

以上步骤通过后按照主键查询库存表库存数量，若数量小于销售数量则拒绝销售登记，否则登记成功，并刷新库存表中的商品数量与修改时间

void Widget::on\_btn\_addw\_2\_clicked()

{

QString cs=ui->let\_cs\_2->text();

QString mc=ui->let\_mc\_2->text();

QString xh=ui->let\_xh\_2->text();

QString dj=ui->let\_dj\_2->text();

QString sl=ui->let\_sl\_2->text();

QString zje=ui->let\_zje\_2->text();

QString year=ui->let\_year\_2->text();

QString moon=ui->let\_moon\_2->text();

QString day=ui->let\_day\_2->text();

QString ywybh=ui->let\_ywybh\_2->text();

if(cs==nullptr||mc==nullptr||zje==nullptr||ywybh==nullptr)

{

QMessageBox::information(this,"销售登记","请填写完整");

}

else

{

if(year==nullptr && moon==nullptr && day==nullptr)

{

int ok=QMessageBox::question(this,"销售登记","确认登记");

if(ok)

{

QDateTime datetime = QDateTime::currentDateTime();

year=QString("%1").arg(datetime.date().year());

moon=QString("%1").arg(datetime.date().month());

day=QString("%1").arg(datetime.date().day());

SqlServer \*sql=new SqlServer();

QString sql1=QString("select \* from stock where 生产厂商='%1' and 商品名='%2' and 型号='%3'").arg(cs,mc,xh);

QSqlQuery \*query1=new QSqlQuery;

if(query1->exec(sql1))

{

if(query1->first())

{

int sl1=query1->value("数量").toInt();

int sl2=sl.toInt();

if(sl1>=sl2)

{

QSqlQuery \*query2=new QSqlQuery;

sl1=sl1-sl2;

QString sl3=QString("%1").arg(sl1);

//QMessageBox::information(this,"",sl3);

QString sql2=QString("update stock set 数量='%1',修改年='%2',修改月='%3',修改日='%4' where 生产厂商= '%5' and 商品名 = '%6' and 型号='%7'").arg(sl3,year,moon,day,cs,mc,xh);

query2->exec(sql2);

if(sql->insertsell(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh))

{

QMessageBox::information(this,"销售登记","登记成功");

}

else

{

QMessageBox::critical(this,"销售登记","登记失败");

}

}

else

{

QMessageBox::information(this,"销售登记","登记失败，商品库存不足");

delete sql;

}

}

else

{

QMessageBox::information(this,"销售登记","登记失败，暂无此商品信息");

delete sql;

}

}

else

{

QMessageBox::critical(this,"销售登记","登记失败");

delete sql;

}

}

else

{}

}

else if((!ui->let\_year\_2->text().isEmpty())&&(!ui->let\_moon\_2->text().isEmpty())&&(!ui->let\_day\_2->text().isEmpty()))

{

int ok=QMessageBox::question(this,"销售登记","确认登记");

if(ok)

{

SqlServer \*sql=new SqlServer();

QString sql1=QString("select \* from stock where 生产厂商='%1' and 商品名='%2' and 型号='%3'").arg(cs,mc,xh);

QSqlQuery \*query1=new QSqlQuery;

if(query1->exec(sql1))

{

if(query1->first())

{

int sl1=query1->value("数量").toInt();

int sl2=sl.toInt();

if(sl1>=sl2)

{

QSqlQuery \*query2=new QSqlQuery;

int sl1=sl.toInt();

int sl2=query1->value("数量").toInt();

sl1=sl1-sl2;

QString sl3=QString("%1").arg(sl1);

QString sql2=QString("update stock set 数量='%1',修改年='%2',修改月='%3',修改日='%4' where 生产厂商= '%5' and 商品名 = '%6' and 型号='%7'").arg(sl3,year,moon,day,cs,mc,xh);

query2->exec(sql2);

if(sql->insertsell(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh))

{

QMessageBox::information(this,"销售登记","登记成功");

}

else

{

QMessageBox::critical(this,"销售登记","登记失败");

}

}

else

{

QMessageBox::information(this,"销售登记","登记失败，商品库存不足");

delete sql;

}

}

else

{

QMessageBox::information(this,"销售登记","登记失败，暂无此商品信息");

delete sql;

}

}

else

{

QMessageBox::critical(this,"销售登记","登记失败");

delete sql;

}

}

}

else

{

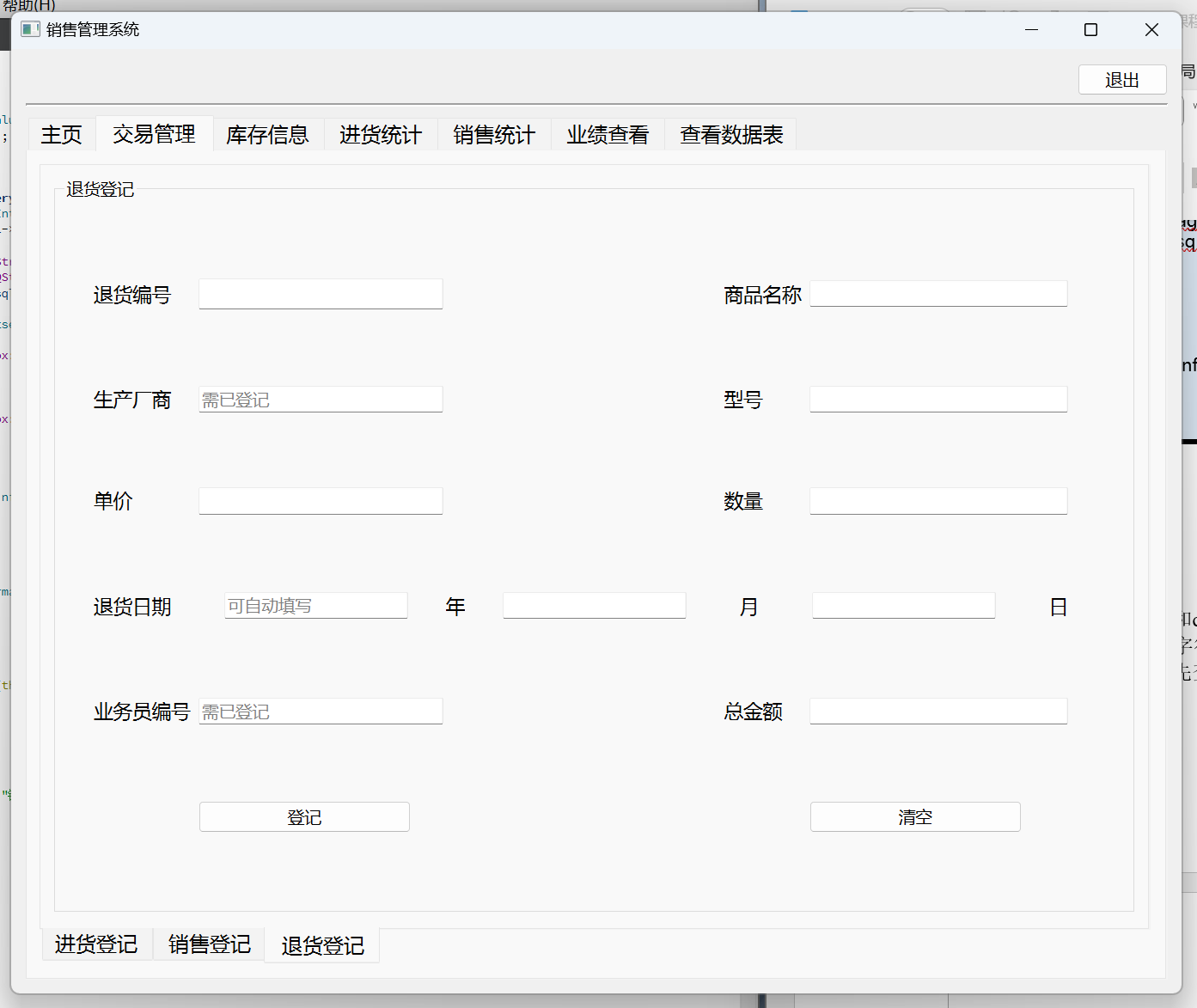
QMessageBox::information(this,"销售登记","日期请填写完整");

}

}

}

* + 1. 退货登记



1. 退货登记

以下为退货登记组“确定”按钮和clicked信号连接的槽函数的代码实现，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后判断是否输入完整，输入完整则执行数据库操作，将信息插入退货表中

void Widget::on\_btn\_addw\_3\_clicked()

{

QString cs=ui->let\_cs\_3->text();

QString mc=ui->let\_mc\_3->text();

QString xh=ui->let\_xh\_3->text();

QString dj=ui->let\_dj\_3->text();

QString sl=ui->let\_sl\_3->text();

QString zje=ui->let\_zje\_3->text();

QString year=ui->let\_year\_3->text();

QString moon=ui->let\_moon\_3->text();

QString day=ui->let\_day\_3->text();

QString ywybh=ui->let\_ywybh\_3->text();

//SqlServer \*sql=new SqlServer();

if(cs==nullptr||mc==nullptr||zje==nullptr||ywybh==nullptr)

{

QMessageBox::information(this,"退货登记","请填写完整");

}

else

{

if(year==nullptr && moon==nullptr && day==nullptr)

{

int ok=QMessageBox::question(this,"退货登记","确认登记");

if(ok)

{

QDateTime datetime = QDateTime::currentDateTime();

year=QString("%1").arg(datetime.date().year());

moon=QString("%1").arg(datetime.date().month());

day=QString("%1").arg(datetime.date().day());

// QMessageBox::information(this,"",year);

// QMessageBox::information(this,"",moon);

// QMessageBox::information(this,"",day);

SqlServer \*sql=new SqlServer();

if(sql->retreat(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh))

{

QMessageBox::information(this,"退货登记","登记成功");

}

else

{

QMessageBox::critical(this,"退货登记","登记失败");

}

}

else

{}

}

else if((!ui->let\_year\_3->text().isEmpty())&&(!ui->let\_moon\_3->text().isEmpty())&&(!ui->let\_day\_3->text().isEmpty()))

{

int ok=QMessageBox::question(this,"退货登记","确认登记");

if(ok)

{

SqlServer \*sql=new SqlServer();

if(sql->retreat(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh))

{

QMessageBox::information(this,"退货登记","登记成功");

}

else

{

QMessageBox::critical(this,"退货登记","登记失败");

}

}

}

else

{

QMessageBox::information(this,"退货登记","日期请填写完整");

}

}

}

以下为退货登记数据库操作部分代码，实例化一个QSqlQuery类的指针对象，通过该对象执行退货登记的sql语句，并返回执行结果。

bool SqlServer::retreat(QString cs,QString mc,QString xh,QString dj,QString sl,QString zje,QString year,QString moon,QString day,QString ywybh)

{

QSqlQuery \*query=new QSqlQuery;

QString str = QString("insert into retreat values('%1', '%2', '%3','%4','%5','%6','%7','%8','%9','%10')").arg(cs,mc,xh,dj,sl,zje,year,moon,day,ywybh);

if(!query->exec(str))

return false;

return true;

}

* 1. 库存信息查询



1. 库存信息

以下为“查询”按钮和clicked信号连接的槽函数的代码实现，首先获取输入框中用户输入文本，使用字符串类型变量接收，

1. 首先通过字符串拼接来获取sql查询语句，此函数存在自动填写日期功能，日期不填写默认为所有记录
2. 获取查询语句后执行
3. 执行完成后返回查询记录，使用导航函数将数据显示到表格中

void Widget::on\_btn\_kcbcx\_clicked()

{

ui->tw\_kcb->setRowCount(0);

//获取sql查询语句

int flag=1;

QString kcbh1=QString("商品编号='%1'").arg(ui->let\_kcbbh->text());

QString sccs1=QString("生产厂商='%1'").arg(ui->let\_kcbsccs->text());

QString spmc1=QString("商品名='%1'").arg(ui->let\_kcbspmc->text());

QString xh1=QString("型号='%1'").arg(ui->let\_kcbxh->text());

QString year1=QString("修改年='%1'").arg(ui->let\_kcbyear->text());

QString moon1=QString("修改月='%1'").arg(ui->let\_kcbmoon->text());

QString day1=QString("修改日='%1'").arg(ui->let\_kcbday->text());

QString endsql1="select \* from stock";

QString endsql2="";

QString endsql3="";

if(!ui->let\_kcbbh->text().isEmpty())

{

endsql3.append(" where ");

endsql2.append(kcbh1);

}

if(!ui->let\_kcbsccs->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(sccs1);

}

if(!ui->let\_kcbspmc->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(spmc1);

}

if(!ui->let\_kcbxh->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(xh1);

}

//日期\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if((ui->let\_kcbyear->text().isEmpty()) && (ui->let\_kcbmoon->text().isEmpty()) && (ui->let\_kcbday->text().isEmpty()))

{

}

else if((!ui->let\_kcbyear->text().isEmpty()) && (!ui->let\_kcbmoon->text().isEmpty()) && (!ui->let\_kcbday->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

endsql2.append(" and ");

endsql2.append(moon1);

endsql2.append(" and ");

endsql2.append(day1);

}

else if((!ui->let\_kcbyear->text().isEmpty()) && (!ui->let\_kcbmoon->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

endsql2.append(" and ");

endsql2.append(moon1);

}

else if((!ui->let\_kcbyear->text().isEmpty())&&(ui->let\_kcbmoon->text().isEmpty())&&(ui->let\_kcbday->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

}

else

{

QMessageBox::information(this,"库存信息","日期请填写完整");

flag=0;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

QString endsql4=" order by 生产厂商,数量";

endsql1.append(endsql3);

endsql1.append(endsql2);

endsql1.append(endsql4);

//QMessageBox::information(this,"",endsql1);

//获取查询数据

if(flag)

{

QSqlQuery query;

QString kcbh2;

QString sccs2;

QString spmc2;

QString xh2;

QString sl2;

QString year2;

QString moon2;

QString day2;

if(query.exec(endsql1))

{

if(query.first())

{

query.previous();

while (query.next())

{

int count=ui->tw\_kcb->rowCount();

ui->tw\_kcb->setRowCount(count+1);

ui->tw\_kcb->setRowCount(count+1);

kcbh2=query.value("商品编号").toString();

sccs2=query.value("生产厂商").toString();

spmc2=query.value("商品名").toString();

xh2=query.value("型号").toString();

sl2=query.value("数量").toString();

year2=query.value("修改年").toString();

moon2=query.value("修改月").toString();

day2=query.value("修改日").toString();

QTableWidgetItem \*spbh=new QTableWidgetItem(kcbh2);

QTableWidgetItem \*sccs=new QTableWidgetItem(sccs2);

QTableWidgetItem \*spm=new QTableWidgetItem(spmc2);

QTableWidgetItem \*xh=new QTableWidgetItem(xh2);

QTableWidgetItem \*sl=new QTableWidgetItem(sl2);

QTableWidgetItem \*year=new QTableWidgetItem(year2);

QTableWidgetItem \*moon=new QTableWidgetItem(moon2);

QTableWidgetItem \*day=new QTableWidgetItem(day2);

ui->tw\_kcb->setItem(count,0,spbh);

ui->tw\_kcb->setItem(count,1,sccs);

ui->tw\_kcb->setItem(count,2,spm);

ui->tw\_kcb->setItem(count,3,xh);

ui->tw\_kcb->setItem(count,4,sl);

ui->tw\_kcb->setItem(count,5,year);

ui->tw\_kcb->setItem(count,6,moon);

ui->tw\_kcb->setItem(count,7,day);

}

}

else

{

QMessageBox::information(this,"库存信息","暂无库存");

}

}

else

{

QMessageBox::critical(this,"库存信息","库存信息获取失败");

}

}

}

* 1. 进货统计



1. 进货统计

今日进货、本月进货、本季度进货和今年进货仅仅时间间隔不同其他均相同，此处仅介绍今日进货功能的实现

以下为“今日进货”按钮与此按钮的clicked信号连接的槽函数的代码实现，

首先通过QT内置的时间类提供的获取系统时间的方法获取今日时间

获取后执行查询语句，因查询结果还需将每个厂商的进货总金额显示到另一张表中，则需要遍历并判断，从而将相同厂商的条目金额相加并显示

执行完成后返回查询记录，使用导航函数将数据显示到表格中

void Widget::on\_btn\_day\_clicked()

{

ui->tw\_hw->setRowCount(0);

ui->tw\_cs->setRowCount(0);

QSqlQuery query;

QString sql1="select \* from goods where 进货年=datepart(year,getdate()) and 进货月=datepart(month,getdate()) and 进货日=datepart(day,getdate()) order by 生产厂商,总金额";

QString spbh1;

QString sccs1;

QString spm1;

QString xh1;

QString dj1;

QString sl1;

QString year1;

QString moon1;

QString day1;

QString zje1;

QString ywybh1;

if(query.exec(sql1))

{

if(query.first())

{

QString sccsstock=query.value("生产厂商").toString();

query.last();

QString bhlast=query.value("商品编号").toString();

query.first();

query.previous();

int endzje1=0;

int endzje2=0;

int csbzje1=0;

//QMessageBox::information(this,"",sccsstock);

while (query.next())

{

int count=ui->tw\_hw->rowCount();

ui->tw\_hw->setRowCount(count+1);

spbh1=query.value("商品编号").toString();

sccs1=query.value("生产厂商").toString();

spm1=query.value("商品名").toString();

xh1=query.value("型号").toString();

dj1=query.value("单价").toString();

sl1=query.value("数量").toString();

year1=query.value("进货年").toString();

moon1=query.value("进货月").toString();

day1=query.value("进货日").toString();

ywybh1=query.value("业务员编号").toString();

zje1=query.value("总金额").toString();

endzje1=query.value("总金额").toInt();

endzje2+=endzje1;

QTableWidgetItem \*zje=new QTableWidgetItem(zje1);

QTableWidgetItem \*spbh=new QTableWidgetItem(spbh1);

QTableWidgetItem \*sccs=new QTableWidgetItem(sccs1);

QTableWidgetItem \*spm=new QTableWidgetItem(spm1);

QTableWidgetItem \*xh=new QTableWidgetItem(xh1);

QTableWidgetItem \*dj=new QTableWidgetItem(dj1);

QTableWidgetItem \*sl=new QTableWidgetItem(sl1);

QTableWidgetItem \*year=new QTableWidgetItem(year1);

QTableWidgetItem \*moon=new QTableWidgetItem(moon1);

QTableWidgetItem \*day=new QTableWidgetItem(day1);

QTableWidgetItem \*ywybh=new QTableWidgetItem(ywybh1);

ui->tw\_hw->setItem(count,0,spbh);

ui->tw\_hw->setItem(count,1,sccs);

ui->tw\_hw->setItem(count,2,spm);

ui->tw\_hw->setItem(count,3,xh);

ui->tw\_hw->setItem(count,4,dj);

ui->tw\_hw->setItem(count,5,sl);

ui->tw\_hw->setItem(count,6,zje);

ui->tw\_hw->setItem(count,7,year);

ui->tw\_hw->setItem(count,8,moon);

ui->tw\_hw->setItem(count,9,day);

ui->tw\_hw->setItem(count,10,ywybh);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(sccs1==sccsstock)

{

//QMessageBox::information(this,"","1");

csbzje1+=endzje1;

}

else

{

//QMessageBox::information(this,"","1");

int count1=ui->tw\_cs->rowCount();

ui->tw\_cs->setRowCount(count1+1);

QString csbzje2=QString("%1").arg(csbzje1);

QTableWidgetItem \*year5=new QTableWidgetItem(year1);

QTableWidgetItem \*moon5=new QTableWidgetItem(moon1);

QTableWidgetItem \*day5=new QTableWidgetItem(day1);

QTableWidgetItem \*zjecsb=new QTableWidgetItem(csbzje2);

QTableWidgetItem \*sccs2=new QTableWidgetItem(sccsstock);

ui->tw\_cs->setItem(count1,0,sccs2);

ui->tw\_cs->setItem(count1,1,year5);

ui->tw\_cs->setItem(count1,2,moon5);

ui->tw\_cs->setItem(count1,3,day5);

ui->tw\_cs->setItem(count1,4,zjecsb);

csbzje1=endzje1;

sccsstock=sccs1;

}

if(spbh1==bhlast)

{

int count1=ui->tw\_cs->rowCount();

ui->tw\_cs->setRowCount(count1+1);

QString csbzje2=QString("%1").arg(csbzje1);

QTableWidgetItem \*year5=new QTableWidgetItem(year1);

QTableWidgetItem \*moon5=new QTableWidgetItem(moon1);

QTableWidgetItem \*day5=new QTableWidgetItem(day1);

QTableWidgetItem \*zjecsb=new QTableWidgetItem(csbzje2);

QTableWidgetItem \*sccs2=new QTableWidgetItem(sccsstock);

ui->tw\_cs->setItem(count1,0,sccs2);

ui->tw\_cs->setItem(count1,1,year5);

ui->tw\_cs->setItem(count1,2,moon5);

ui->tw\_cs->setItem(count1,3,day5);

ui->tw\_cs->setItem(count1,4,zjecsb);

csbzje1=0;

sccsstock=sccs1;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

QString endzje3=QString("%1").arg(endzje2);

ui->let\_jhzje->setText(endzje3);

}

else

{

QMessageBox::information(this,"进货统计","今日暂未进货");

}

}

else

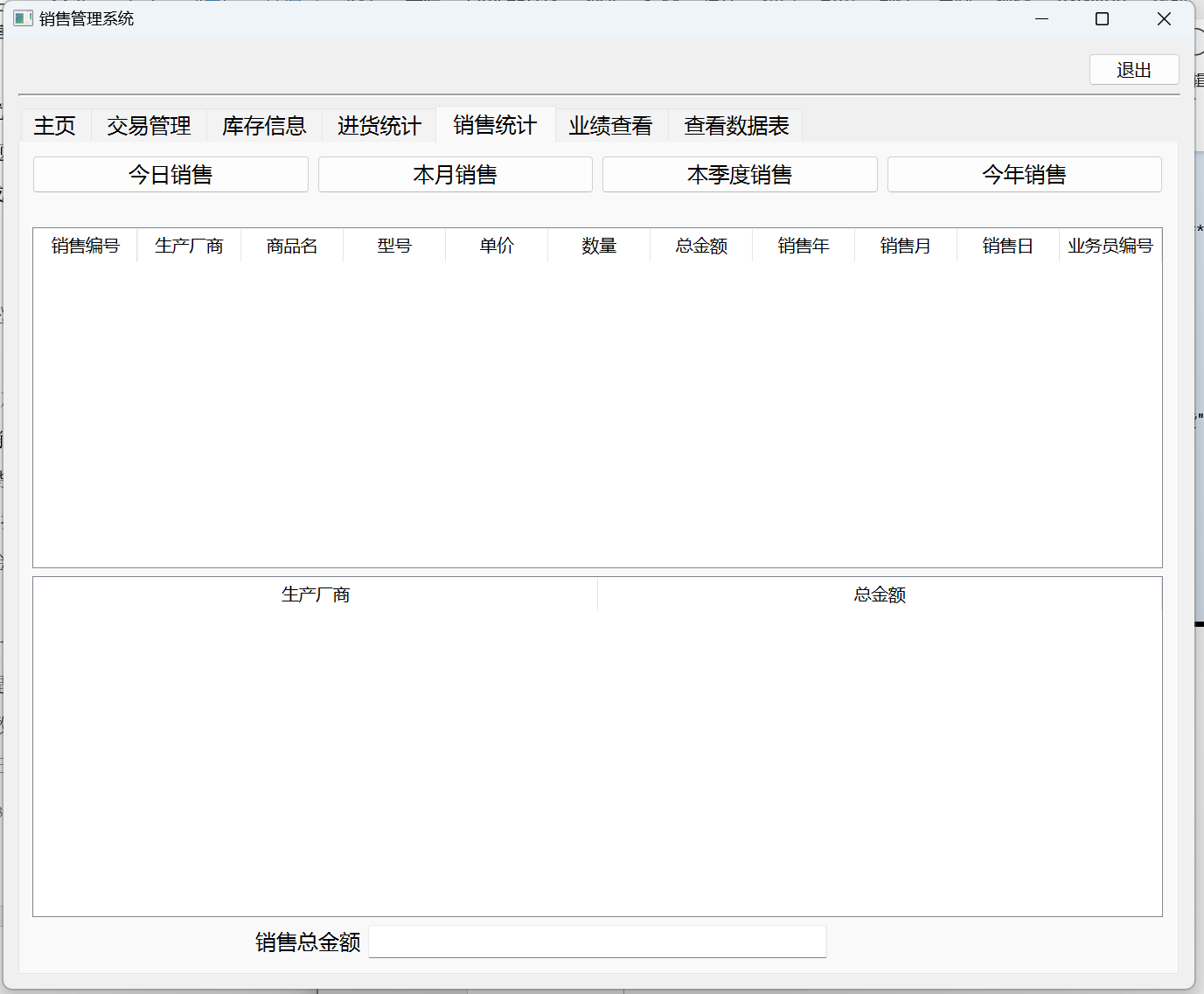
{

QMessageBox::critical(this,"进货统计","进货数据获取失败");

}

}

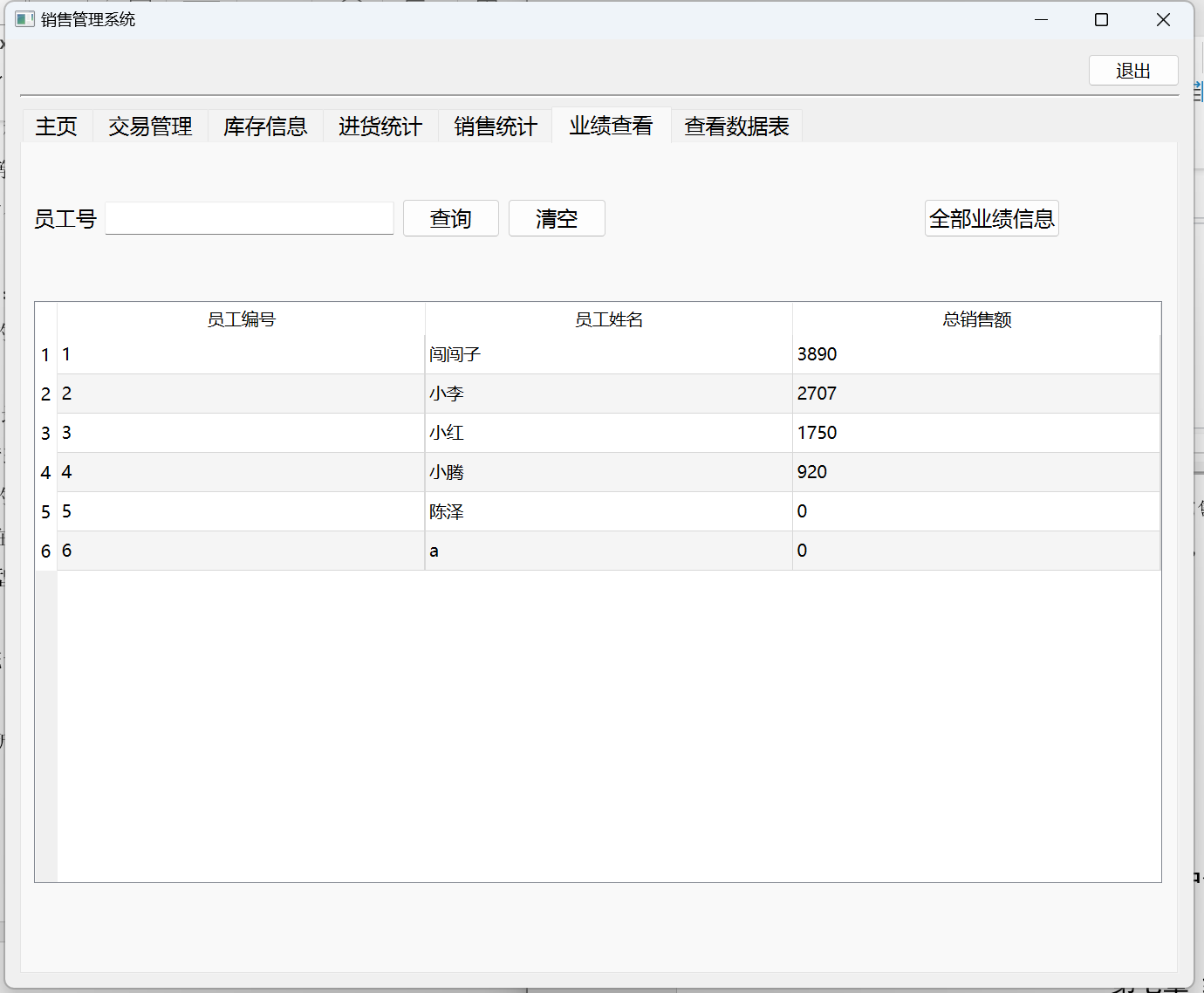
* 1. 销售统计



1. 销售统计

销售统计实现代码与进货统计类似，仅sql语句以及字段名有所差别，就不再重复介绍

* 1. 业绩查看



1. 业绩查看



1. 业绩查看

首先获取输入框中用户输入文本，使用字符串类型变量接收，然后执行数据库操作，执行完成后返回查询记录，使用导航函数将数据显示到表格中

以下为“查询”按钮与此按钮的clicked信号连接的槽函数的代码实现，

void Widget::on\_btn\_ygyj\_clicked()

{

ui->tw\_ygyj->setRowCount(0);

QString ygbh1=ui->let\_ygh->text();

QSqlQuery query1;

QSqlQuery query2;

QString sql2=QString("select \* from sell where 业务员编号='%1'").arg(ygbh1);

QString sql1=QString("select \* from employee where 员工编号='%1'").arg(ygbh1);

QString ygxm1;

int je=0;

int zje1=0;

if(query1.exec(sql1))

{

if(query1.first())

{

if(query2.exec(sql2))

{

if(query2.first())

{

//QMessageBox::information(this,"1","1111");

query2.previous();

while (query2.next())

{

je=query2.value("总金额").toInt();

zje1=zje1+je;

}

int count=ui->tw\_ygyj->rowCount();

ui->tw\_ygyj->setRowCount(count+1);

ygxm1=query1.value("员工姓名").toString();

QString zje2=QString("%1").arg(zje1);

zje1=0;

je=0;

QTableWidgetItem \*ygbh=new QTableWidgetItem(ygbh1);

QTableWidgetItem \*ygxm=new QTableWidgetItem(ygxm1);

QTableWidgetItem \*zje=new QTableWidgetItem(zje2);

ui->tw\_ygyj->setItem(count,0,ygbh);

ui->tw\_ygyj->setItem(count,1,ygxm);

ui->tw\_ygyj->setItem(count,2,zje);

}

else

{

//QMessageBox::information(this,"1","222222");

int count=ui->tw\_ygyj->rowCount();

ui->tw\_ygyj->setRowCount(count+1);

zje1=0;

QString zje2=QString("%1").arg(zje1);

ygxm1=query1.value("员工姓名").toString();

QTableWidgetItem \*ygbh=new QTableWidgetItem(ygbh1);

QTableWidgetItem \*ygxm=new QTableWidgetItem(ygxm1);

QTableWidgetItem \*zje=new QTableWidgetItem(zje2);

ui->tw\_ygyj->setItem(count,0,ygbh);

ui->tw\_ygyj->setItem(count,1,ygxm);

ui->tw\_ygyj->setItem(count,2,zje);

}

}

else

{

QMessageBox::critical(this,"业绩查看","员工销售数据获取失败");

}

}

else

{

QMessageBox::information(this,"业绩查看","此编号员工不存在");

}

}

else

{

QMessageBox::critical(this,"业绩查看","员工销售数据获取失败");

}

}

* 1. 查看数据表



1. 查看数据表

查看数据表的逻辑代码与前文的库存信息查询相似，不在重复介绍

以下为“查询”按钮与此按钮的clicked信号连接的槽函数的代码实现，

void Widget::on\_btn\_jhbcx\_clicked()

{

ui->tw\_jhb->setRowCount(0);

//获取sql查询语句

int flag=1;

QString spbh1=QString("商品编号='%1'").arg(ui->let\_jhbbh->text());

QString cs1=QString("生产厂商='%1'").arg(ui->let\_jhbsccs->text());

QString spm1=QString("商品名='%1'").arg(ui->let\_jhbmc->text());

QString xh1=QString("型号='%1'").arg(ui->let\_jhbxh->text());

QString year1=QString("进货年='%1'").arg(ui->let\_jhbyear->text());

QString moon1=QString("进货月='%1'").arg(ui->let\_jhbmoon->text());

QString day1=QString("进货日='%1'").arg(ui->let\_jhbday->text());

QString ywybh1=QString("业务员编号='%1'").arg(ui->let\_jhbywybh->text());

QString endsql1="select \* from goods";

QString endsql2="";

QString endsql3="";

if(!ui->let\_jhbbh->text().isEmpty())

{

endsql3.append(" where ");

endsql2.append(spbh1);

}

if(!ui->let\_jhbsccs->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(cs1);

}

if(!ui->let\_jhbmc->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(spm1);

}

if(!ui->let\_jhbxh->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(xh1);

}

//日期\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if((ui->let\_jhbyear->text().isEmpty()) && (ui->let\_jhbmoon->text().isEmpty()) && (ui->let\_jhbday->text().isEmpty()))

{

}

else if((!ui->let\_jhbyear->text().isEmpty()) && (!ui->let\_jhbmoon->text().isEmpty()) && (!ui->let\_jhbday->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

endsql2.append(" and ");

endsql2.append(moon1);

endsql2.append(" and ");

endsql2.append(day1);

}

else if((!ui->let\_jhbyear->text().isEmpty()) && (!ui->let\_jhbmoon->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

endsql2.append(" and ");

endsql2.append(moon1);

}

else if((!ui->let\_jhbyear->text().isEmpty())&&(ui->let\_jhbmoon->text().isEmpty())&&(ui->let\_jhbday->text().isEmpty()))

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(year1);

}

else

{

QMessageBox::information(this,"进货表","日期请填写完整");

flag=0;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(!ui->let\_jhbywybh->text().isEmpty())

{

if(endsql3.isEmpty())

{

endsql3.append(" where ");

}

if(!endsql2.isEmpty())

{

endsql2.append(" and ");

}

endsql2.append(ywybh1);

}

endsql1.append(endsql3);

endsql1.append(endsql2);

//QMessageBox::information(this,"",endsql1);

//获取查询数据

if(flag)

{

QSqlQuery query;

QString spbh2;

QString sccs2;

QString spm2;

QString xh2;

QString dj2;

QString sl2;

QString zje2;

QString year2;

QString moon2;

QString day2;

QString ywybh2;

if(query.exec(endsql1))

{

if(query.first())

{

query.previous();

while (query.next())

{

int count=ui->tw\_jhb->rowCount();

ui->tw\_jhb->setRowCount(count+1);

ui->tw\_jhb->setRowCount(count+1);

spbh2=query.value("商品编号").toString();

sccs2=query.value("生产厂商").toString();

spm2=query.value("商品名").toString();

xh2=query.value("型号").toString();

dj2=query.value("单价").toString();

sl2=query.value("数量").toString();

zje2=query.value("总金额").toString();

year2=query.value("进货年").toString();

moon2=query.value("进货月").toString();

day2=query.value("进货日").toString();

ywybh2=query.value("业务员编号").toString();

QTableWidgetItem \*spbh=new QTableWidgetItem(spbh2);

QTableWidgetItem \*sccs=new QTableWidgetItem(sccs2);

QTableWidgetItem \*spm=new QTableWidgetItem(spm2);

QTableWidgetItem \*xh=new QTableWidgetItem(xh2);

QTableWidgetItem \*dj=new QTableWidgetItem(dj2);

QTableWidgetItem \*sl=new QTableWidgetItem(sl2);

QTableWidgetItem \*zje=new QTableWidgetItem(zje2);

QTableWidgetItem \*year=new QTableWidgetItem(year2);

QTableWidgetItem \*moon=new QTableWidgetItem(moon2);

QTableWidgetItem \*day=new QTableWidgetItem(day2);

QTableWidgetItem \*ywybh=new QTableWidgetItem(ywybh2);

ui->tw\_jhb->setItem(count,0,spbh);

ui->tw\_jhb->setItem(count,1,sccs);

ui->tw\_jhb->setItem(count,2,spm);

ui->tw\_jhb->setItem(count,3,xh);

ui->tw\_jhb->setItem(count,4,dj);

ui->tw\_jhb->setItem(count,5,sl);

ui->tw\_jhb->setItem(count,6,zje);

ui->tw\_jhb->setItem(count,7,year);

ui->tw\_jhb->setItem(count,8,moon);

ui->tw\_jhb->setItem(count,9,day);

ui->tw\_jhb->setItem(count,10,ywybh);

}

}

else

{

QMessageBox::information(this,"进货表","暂无进货数据");

}

}

else

{

QMessageBox::critical(this,"进货表","进货数据获取失败");

}

}

}

* 1. 员工表与厂商表

员工表与厂商表中存在编辑与添加功能，查询功能与其他表类似

厂商表与员工表实现代码类似，此处仅介绍员工表代码实现

需要编辑时，首先将“编辑”单选框选中选中后单击所要修改的条目，软件会获取选中的信息，并显示到“员工信息框”中的文本框中，直接在文本框中修改完成后点击修改即可修改完成



以下为“编辑”单选框与选中信号连接的槽函数的实现代码，首先获取单选框选中状态，然后连接表格控件选中信号与槽函数

void Widget::on\_checkBox\_clicked(bool checked)

{

if(checked)

{

connect(ui->tw\_ygb, &QTableWidget::itemClicked,this,&Widget::onitemclicked);

ui->let\_ygbygbh->setReadOnly(true);

ui->check\_ygbtj->setCheckable(false);

}

else if(ui->check\_ygbtj->isChecked())

{

}

else

{

ui->let\_ygbygxm->clear();

ui->let\_ygbygbh->clear();

ui->let\_ygblxdh->clear();

ui->let\_ygbygdz->clear();

disconnect(ui->tw\_ygb, &QTableWidget::itemClicked,this,&Widget::onitemclicked);

ui->let\_ygbygbh->setReadOnly(false);

ui->check\_ygbtj->setCheckable(true);

}

}

以下为“修改“按钮与clicked信号连接的槽函数的实现代码，首先获取输入框中用户输入文本，使用字符串类型变量接收，然后执行数据库操作，修改信息

void Widget::on\_btn\_ygbxg\_clicked()

{

if(ui->checkBox->isChecked())

{

int ok=QMessageBox::question(this,"员工表","确认修改？");

if(ok==QMessageBox::Yes)

{

QString ygbh=ui->let\_ygbygbh->text();

QString ygxm=ui->let\_ygbygxm->text();

QString yglxdh=ui->let\_ygblxdh->text();

QString ygdz=ui->let\_ygbygdz->text();

QSqlQuery query1;

QSqlQuery query2;

QSqlQuery query3;

QString sql1=QString("update employee set 员工姓名='%1' where 员工编号= '%2'").arg(ygxm,ygbh);

QString sql2=QString("update employee set 员工电话='%1' where 员工编号= '%2'").arg(yglxdh,ygbh);

QString sql3=QString("update employee set 员工地址='%1' where 员工编号= '%2'").arg(ygdz,ygbh);

if(query1.exec(sql1) && query2.exec(sql2) && query3.exec(sql3))

{

//QMessageBox::information(this,"员工表","修改成功");

int count=ui->tw\_ygb->currentRow();

QTableWidgetItem \*ygbh1=new QTableWidgetItem(ygbh);

QTableWidgetItem \*ygxm1=new QTableWidgetItem(ygxm);

QTableWidgetItem \*lxdh1=new QTableWidgetItem(yglxdh);

QTableWidgetItem \*ygdz1=new QTableWidgetItem(ygdz);

ui->tw\_ygb->setItem(count,0,ygbh1);

ui->tw\_ygb->setItem(count,1,ygxm1);

ui->tw\_ygb->setItem(count,2,lxdh1);

ui->tw\_ygb->setItem(count,3,ygdz1);

}

else

{

QMessageBox::information(this,"员工表","修改失败");

}

}

else

{

}

}

else

{

QMessageBox::critical(this,"员工表","拒绝修改");

}

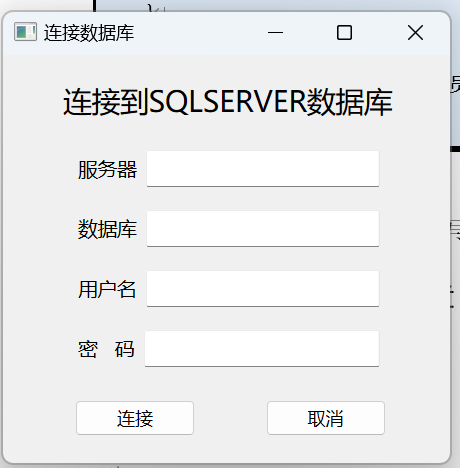
}

添加时需将添加单选框选中，选中后系统会自动获取下一编号，然后用户仅需在输入框中填写完整信息，填写完成后点击“添加“按钮，即可添加完毕。

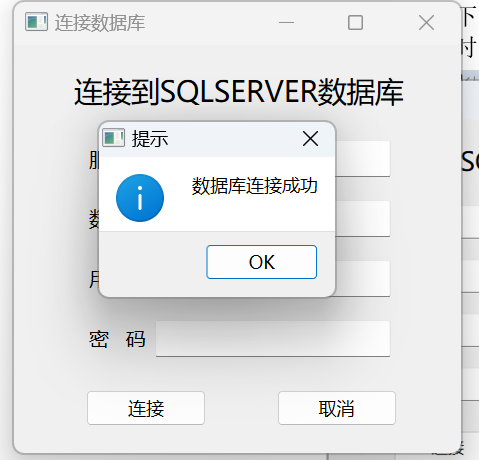
第六章 用户使用说明书

* 1. 连接数据库

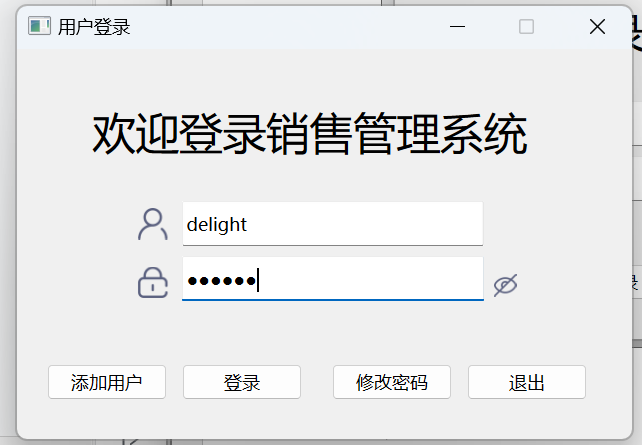
打开软件后首先需首先在以下界面依次填写服务器名、数据库名、用户名与密码；注意此处的用户名和密码时获取数据库访问权限的用户名和密码



连接成功后会有信息提示，如下图



点击OK跳转到登录界面，在输入框中填写用户名与密码，此处为有登录此系统权限的用户名与密码，即管理员



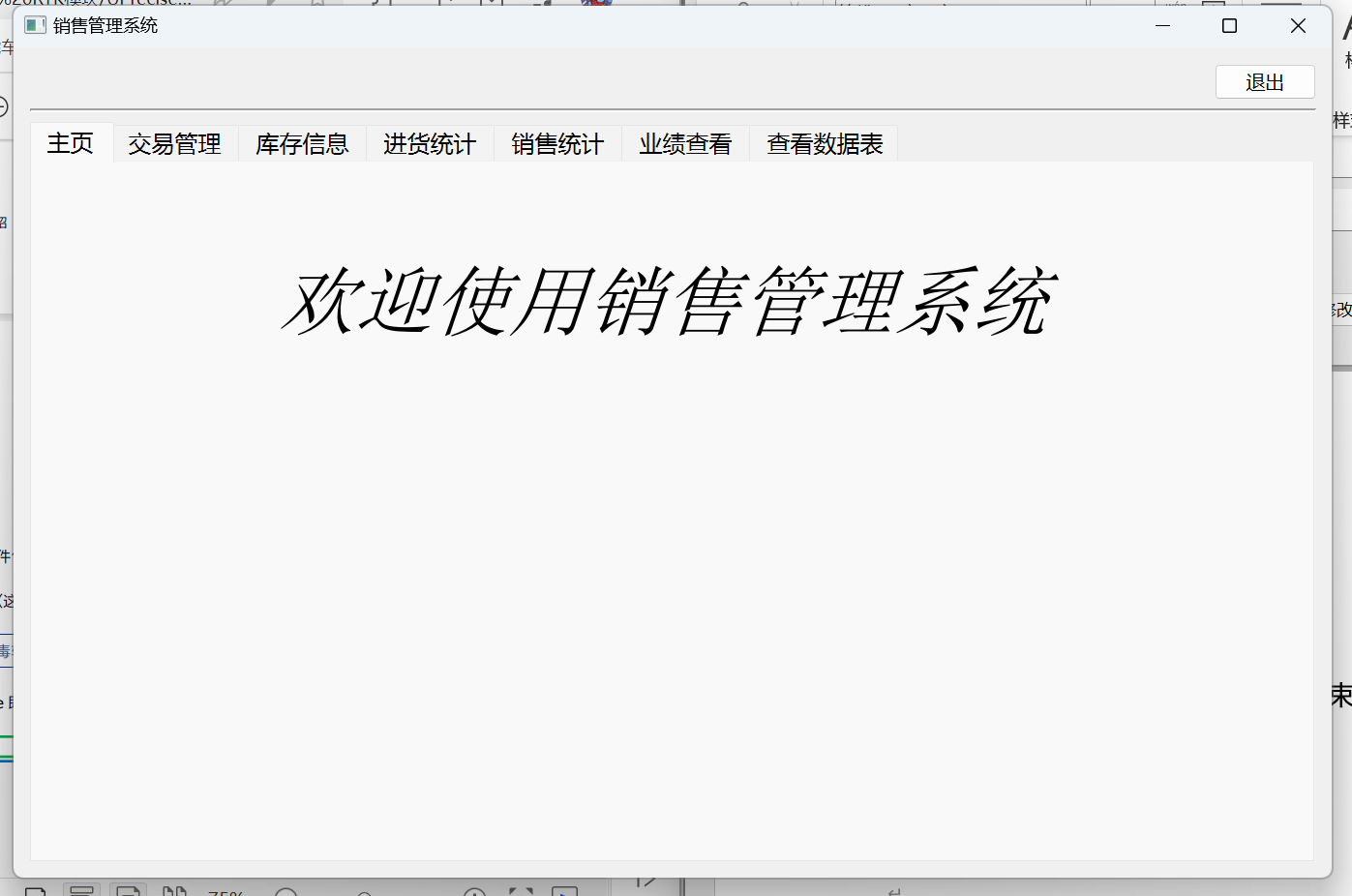
1. 登录界面

点击右侧小眼睛按钮可将密码显示，如下图，箭头所指为功能选择键



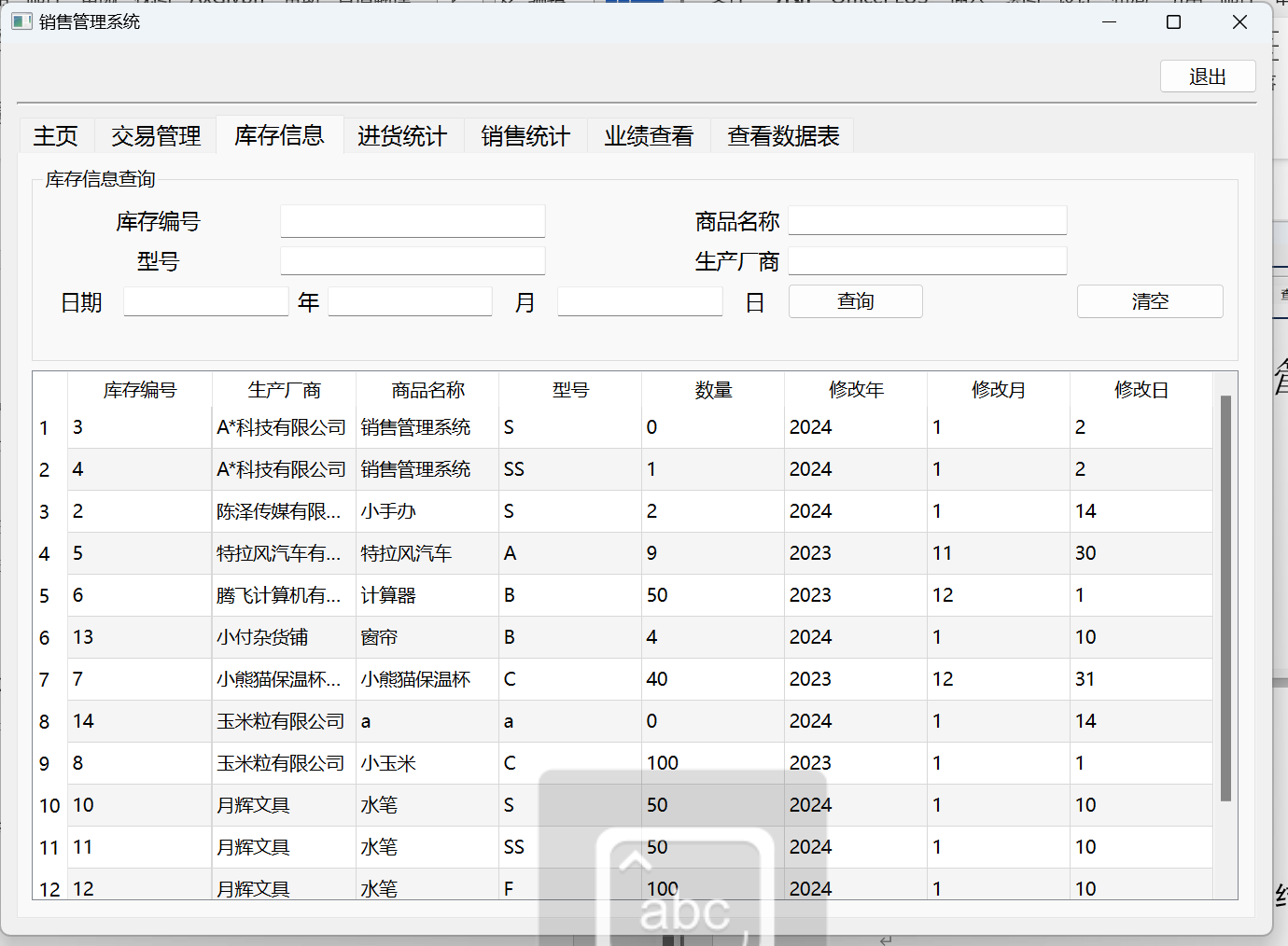
不同功能选项卡

点击登录后跳转到主页

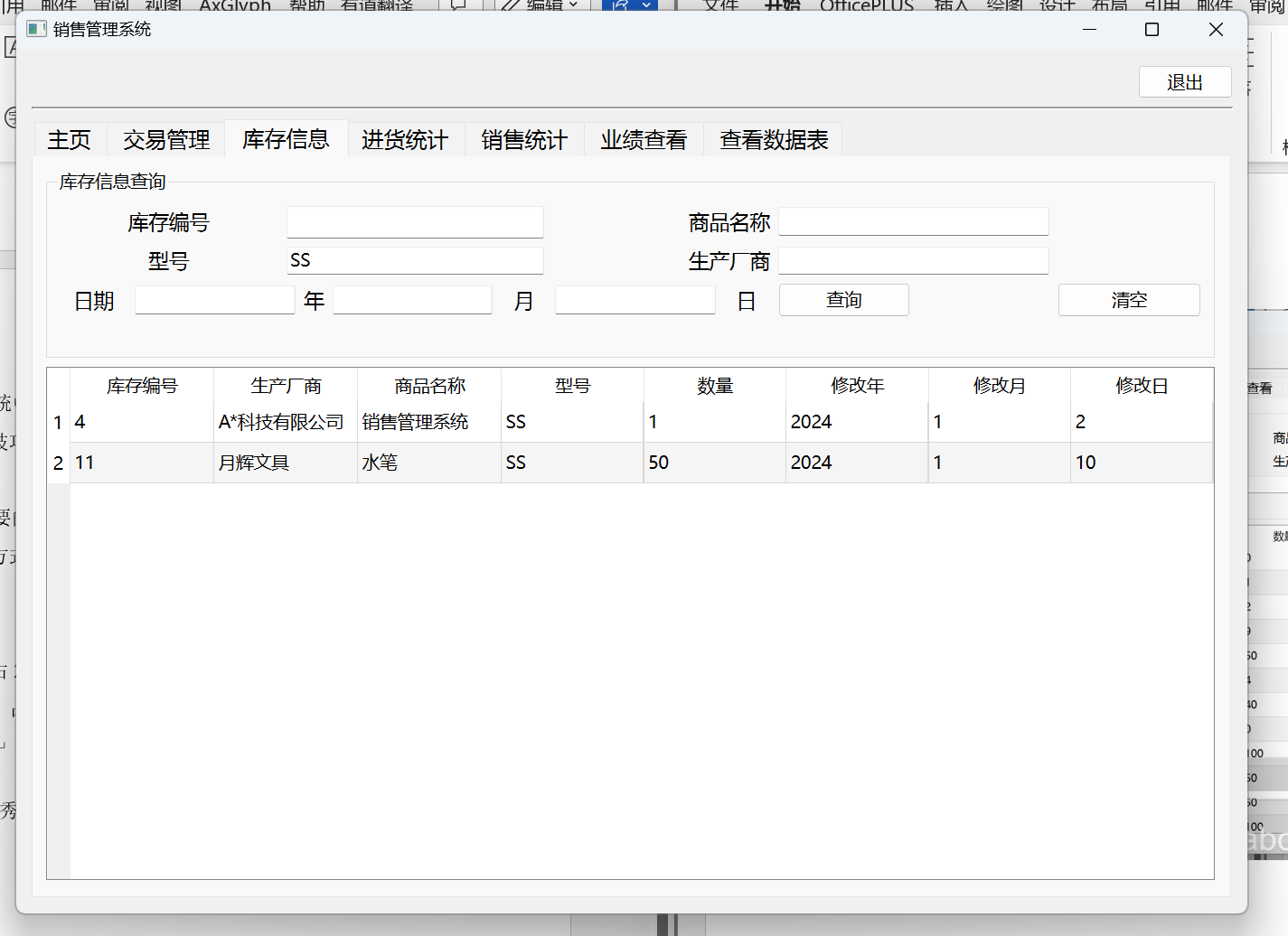


1. 主页

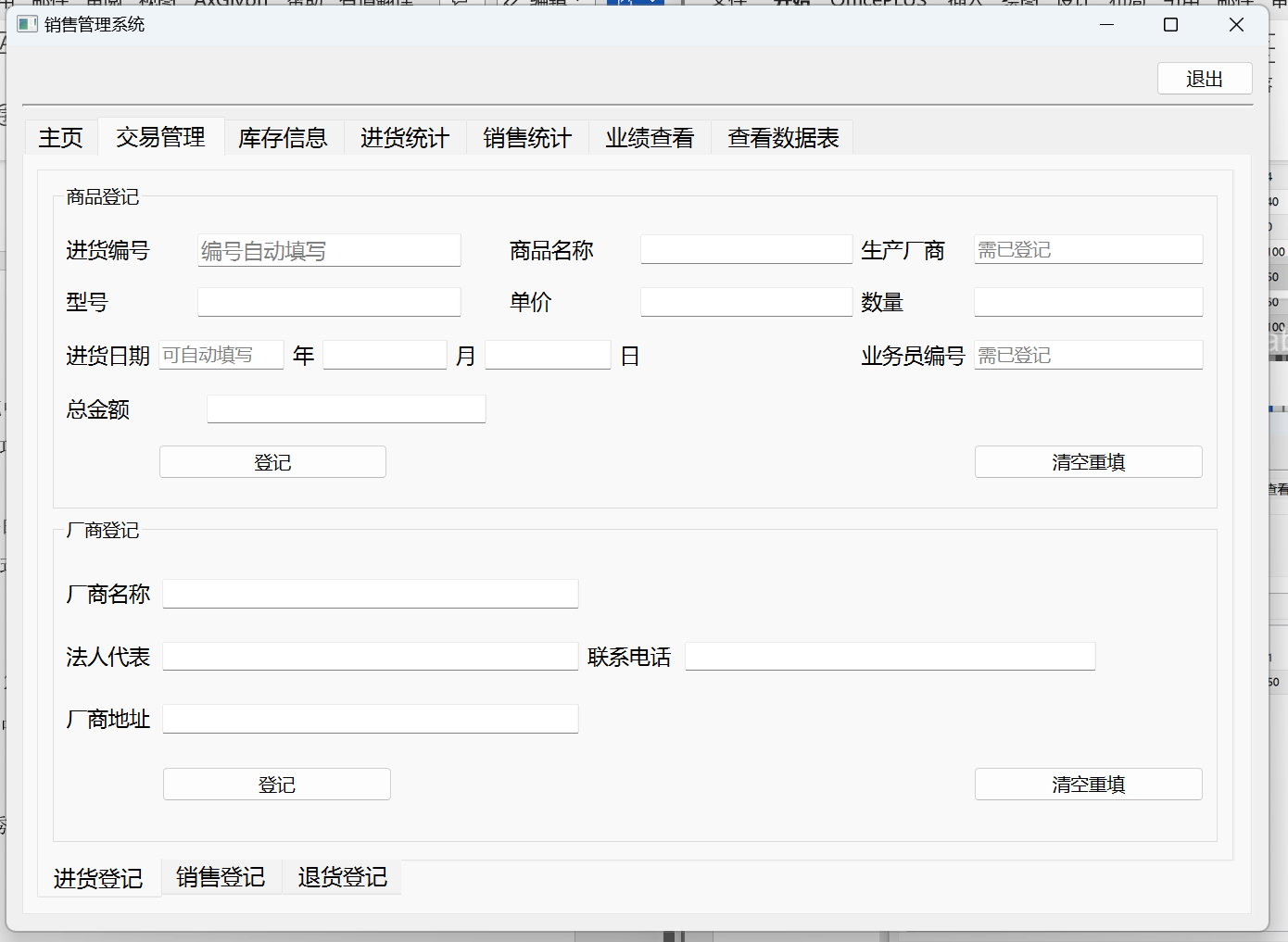
以下为不同功能界面展示



1. 库存信息查询



1. 库存信息查询



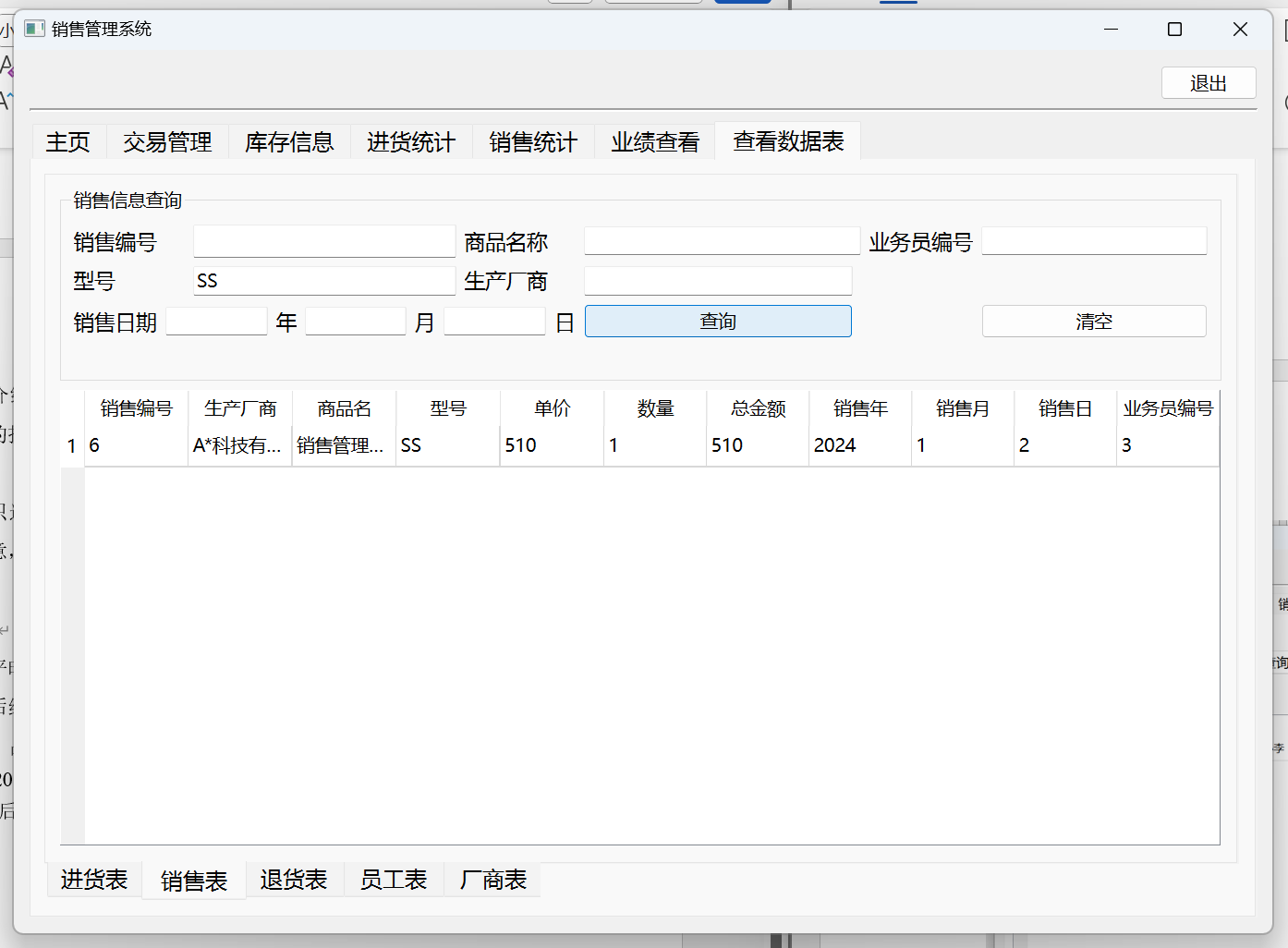
1. 交易管理



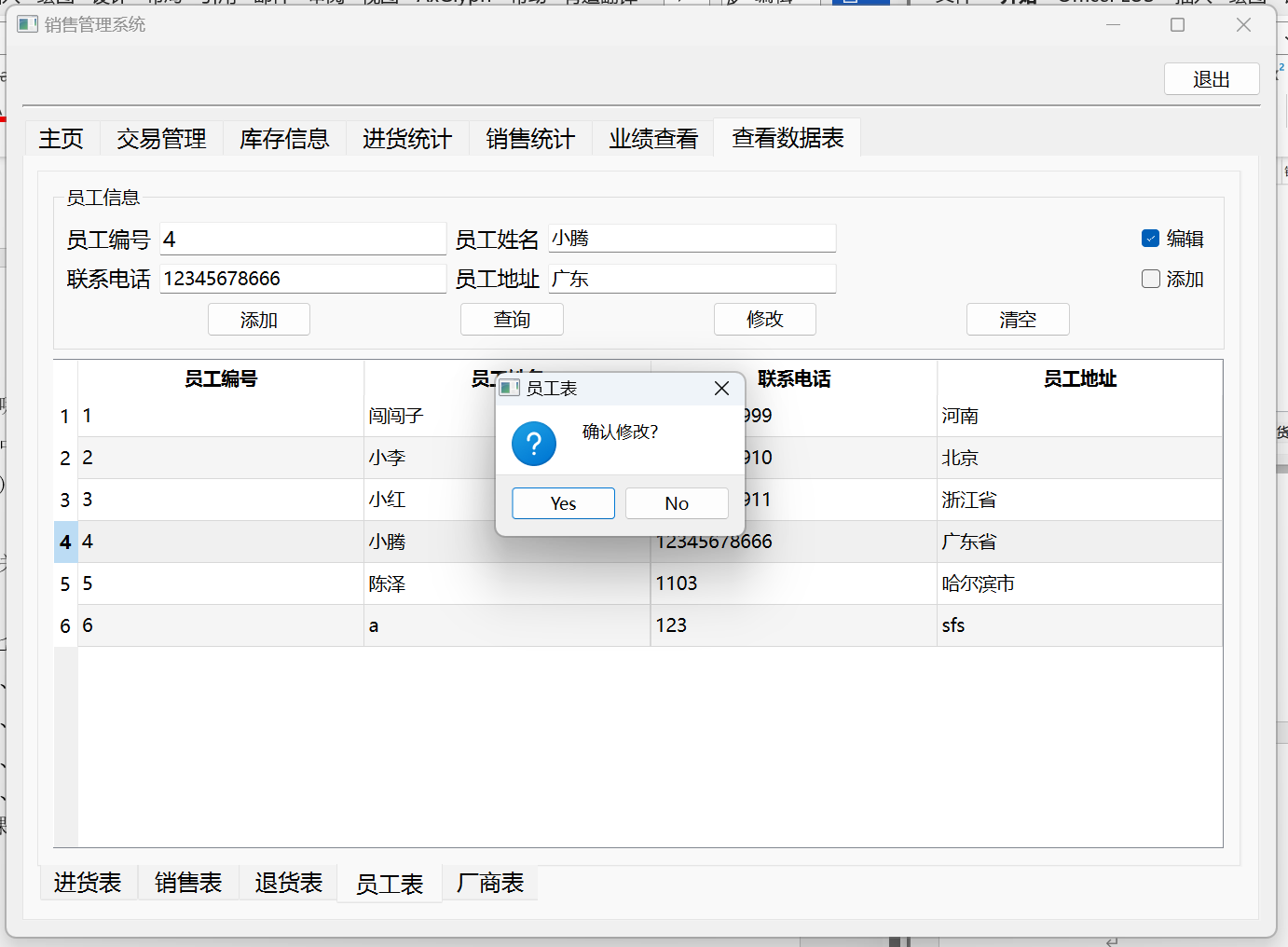
1. 进货统计



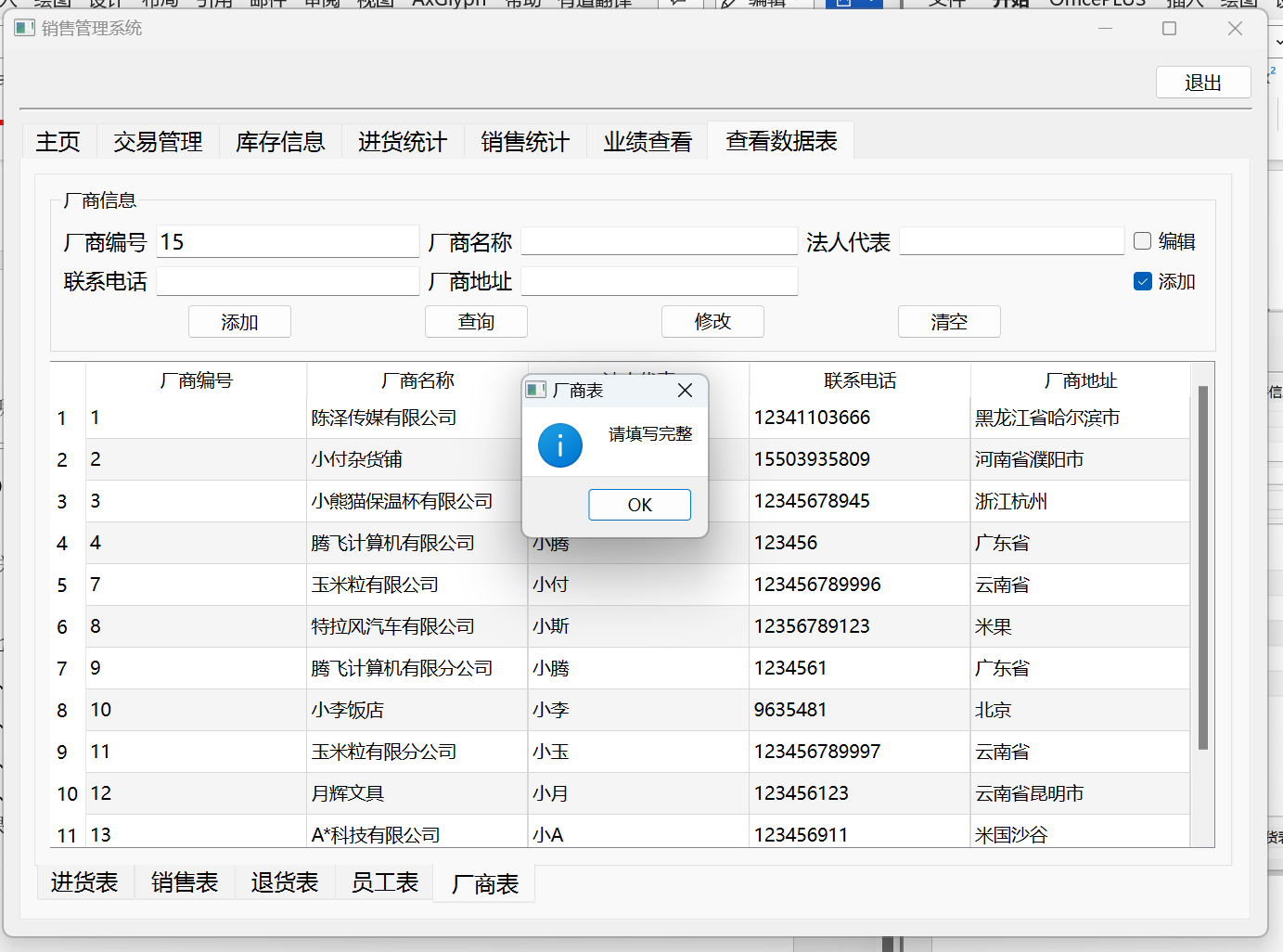
1. 业绩查看



1. 查看数据表



1. 员工表



1. 厂商表

第七章 结束语

回顾整个销售管理系统数据库课设的设计过程。在这个过程中，我学习了数据库设计的基本原理和流程，了解了如何规划数据库结构、设计数据表和建立关系等重要内容。通过实际操作和实践，加深了对数据库系统原理及程序设计的理论知识的理解和应用，并深入了解了数据库管理系统的功能和作用，提升了自己的实际操作能力。

在课设过程中，我遇到了一些挑战，比如在设计数据库时需要考虑到各种业务需求，保证数据的一致性和完整性；在编写 SQL 查询语句时要保证语法正确并实现所需功能等。为了解决这些挑战，我积极查阅资料、请教老师和同学，不断学习和提升自己的技能。

通过这个课设项目，我不仅掌握了数据库设计和管理的基本知识，还提高了自己的问题解决能力。在未来，我希望能够继续深入学习数据库相关知识，不断提升自己的能力，为实际工作做好充分准备。同时，我也希望能够将所学到的知识运用到实际项目中，为企业的发展和管理提供更好的支持和帮助。

总的来说，销售管理系统数据库课设报告的心得是我在这个项目中的成长和收获，是我对所学知识的总结和反思，也是我对未来发展的展望和规划。通过这样的学习和实践，我相信我会更加成熟和自信地面对未来的挑战，为自己的职业发展打下坚实的基础。

* 1. 创意点

此系统在进行数据库查询时采用参数化获取查询语句并结合预编译技术，即PreparedStatement对象。

PreparedStatement对象创建的一个必要条件就是先传输一个sql骨架，输入参数用？占位符替代。这样sql语句的骨架已确定（比如判断条件中的and，or…已固定），任何由外部注入的，使sql语句骨架发生变化的操作都会判定为无效输入。从而防止了sql注入攻击。这样可提高数据库与系统安全性。