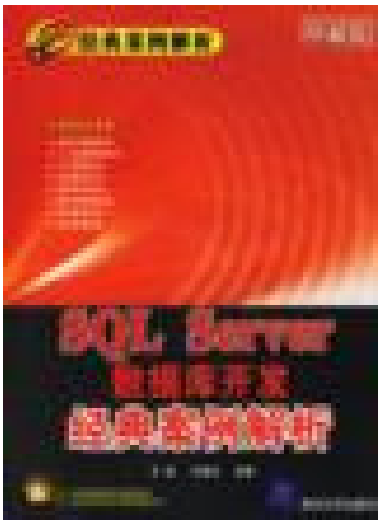
SQL Server数据库开发经典案例解析



清华大学出版社 2006 年 1 月 1 日

王　晟　马里杰 编著

48 元

ISBN：7302122660

前言

SQL Server 2000 着眼于 Internet 背景下网络数据库的应用与开发，它扩展了 SQL Server 7.0 的可靠性及易用性。它还设计许多新功能，以提高系统的执行效能，而且使数据库的管理工作变得更加轻松，这些功能进一步将 SQL Server 确立为 OLTP、数据仓库及电子商务应用程序的最佳数据库平台。

管理信息系统(Management Information System，MIS)是 一个由人、计算机及其他外围设备等组成的能进行信息的收集、传递、存储、加工、维护和使用的系统。其主要任务是最大限度地利用现代计算机及网络通信技术加 强企业的信息管理，通过对企业拥有的人力、物力、财力、设备、技术等资源的调查了解，建立正确的数据库，加工处理后编制成各种信息资料并及时提供给管理人 员，以便进行正确的决策，不断提高企业的管理水平和经济效益。

随着计算机应用在中国的普及，众多的企事业单位越来越重视管理信息系统的投入和应用，这为管理信息系统提供了广阔的市场空间，也对开发人员的数量和质量产生了巨大的需求。SQL Server 的 初学者，有些没有编程基础，有些具有一定的基础、做过一些小程序的开发。不管是哪一种，都非常缺乏实际工作经验。目前大部分计算机类的图书均是介绍计算机 编程工具的用法，即便是范例类的图书，也是以介绍知识点为出发点，读者看完后知道了如何使用这些编程工具，但往往到实际工作中却无从下手，因为尽管知道了 怎么做，却不知道需要做什么、可以做什么。例如读者知道了 SQL Server 的数据查询语言和 SQL Server 数据库的使用方法，也知道使用 SQL Server 数据库和开发语言可以制作 ERP 系统，但不知道 ERP 系统是什么东西，其中包含什么内容，用计算机如何管理数据库以及实际的工作流程如何在计算机上实现等。

鉴于此，作者组织了多个管理信息系统公司的项目小组，完成了本书的创作。本书的第 1～4 章分别介绍数据库基础、SQL Server 2000 应用基础、数据库创建及维护、Transact-SQL 程序设计；从第 5～8 章以进销存管理系统、医院管理系统、酒店管理系统和图书馆管理系统 4 个实用的信息管理系统为例，分别以 Delphi、PowerBuilder、Visual Basic .NET 和 Visual C# .NET 4 种开发工具为开发平台，详细地介绍了这些系统的需求分析及开发的过程和方法，对项目背景、业务需求分析、功能需求分析、数据库需求分析、数据库建模、系统开发、系统编译及系统发行等过程进行了详细的讲解；第 9～12 章简单介绍了教务管理系统、人力资源管理系统、生产管理系统和财务管理系统的业务需求分析和数据库建模。同时本书提供了所有系统完整的数据库建库脚本，用户在实际的工作中可以直接使用，并可在此基础上进行补充，从而大大减少系统数据库设计的工作量和时间。

本书 5～12 章中的系统需求分析部分由王晟完成，第 1 章～第 4 章由王晟编写，第 5 章由杨华编写，第 6 章由文芳编写，第 7 章和第 8 章由王淼编写，第 9～12 章由杨芸编写。

由于时间仓促，加之编者的水平有限，缺点和错误在所难免，恳请专家和广大读者不吝赐教，批评指正。

目录

第 1 章 数据库基础

数据库技术是现代信息科学与技术的重要组成部分，是计算机数据处理与信息管理系统的核心。数据库技术研究和解决了计算机信息处理过程中大量数据有效地组织和存储的问题，在数据库系统中减少数据存储冗余、实现数据共享、保障数据安全以及高效地检索数据和处理数据。

随着计算机技术与网络通信技术的发展，数据库技术已成为信息社会中对大量数据进行组织与管理的重要技术手段及软件技术，是网络信息化管理系统的基础。本章主 要介绍数据库技术的应用与发展、关系模型的基本概念、关系数据库的设计理论及数据库设计方法等内容，是学习和掌握现代数据库技术的基础。

1.1 数据库技术的发展与应用

从 20 世纪 60 年代末期开始到现在，数据库技术已经发展了 30 多年。在这 30 多年的历程中，人们在数据库技术的理论研究和系统开发上都取得了辉煌的成就，而且已经开始对新一代数据库系统的深入研究。数据库系统已经成为现代计算机系统的重要组成部分。

1.1.1 数据库技术与信息技术

信息技术(Information Technology，IT)是当今使用频率最高的名词之一，它随着计算机技术在工业、农业以及日常生活中的广泛应用，已经被越来越多的个人和企业作为自己赶超世界潮流的标志之一。而数据库技术则是信息技术中一个重要的支撑。没有数据库技术，人们在浩瀚的信息世界中将显得手足无措。

数据库技术是计算机科学技术的一个重要分支。从 20 世纪 50 年代中期开始，计算机应用从科学研究部门扩展到企业管理及政府行政部门，人们对数据处理的要求也越来越高。1968 年，世界上诞生了第一个商品化的信息管理系统 IMS(Information Management System)，从此，数据库技术得到了迅猛发展。在互联网日益被人们接受的今天，Internet 又使数据库技术、知识、技能的重要性得到了充分的放大。现在数据库已经成为信息管理、办公自动化、计算机辅助设计等应用的主要软件工具之一，帮助人们处理各种各样的信息数据。

1.1.2 数据库技术的应用及特点

数据库最初是在大公司或大机构中用作大规模事务处理的基础。后来随着个人计算机的普及，数据库技术被移植到 PC 机(Personal Computer，个人计算机)上，供单用户个人数据库应用。接着，由于 PC 机在工作组内连成网，数据库技术就移植到工作组级。现在，数据库正在 Internet 和内联网中广泛使用。

20 世纪 60 年代中期，数据库技术是用来解决文件处理系统问题的。当时的数据库处理技术还很脆弱，常常发生应用不能提交的情况。20 世纪 70 年代关系模型的诞生为数据库专家提供了构造和处理数据库的标准方法，推动了关系数据库的发展和应用。1979 年，Ashton-Tate 公司引入了微机产品 dBase Ⅱ，并称之为关系数据库管理系统，从此数据库技术移植到了个人计算机上。20 世纪 80 年代中期到后期，终端用户开始使用局域网技术将独立的计算机连接成网络，终端之间共享数据库，形成了一种新型的多用户数据处理，称为客户机/服务器数据库结构。现在，数据库技术正在被用来同 Internet 技术相结合，以便在机构内联网、部门局域网甚至 WWW 上发布数据库数据。

1.1.3 数据库技术发展历史

数据模型是数据库技术的核心和基础，因此，对数据库系统发展阶段的划分应该以数据模型的发展演变作为主要依据和标志。按照数据模型的发展演变过程，数据库技术从开始到现在短短的 30 年中，主要经历了三个发展阶段：第一代是网状和层次数据库系统，第二代是关系数据库系统，第三代是以面向对象数据模型为主要特征的数据库系统。数据库技术与 网络通信技术、人工智能技术、面向对象程序设计技术、并行计算技术等相互渗透、有机结合，成为当代数据库技术发展的重要特征。

1. 第一代数据库系统

第一代数据库系统是 20 世纪 70 年代研制的层次和网状数据库系统。层次数据库系统的典型代表是 1969 年 IBM 公司研制出的层次模型的数据库管理系统 IMS。20 世纪 60 年代末 70 年代初，美国数据库系统语言协会 CODASYL(Conference on Data System Language)下属的数据库任务组 DBTG(Data Base Task Group)提出了若干报告，被称为 DBTG 报告。DBTG 报告确定并建立了网状数据库系统的许多概念、方法和技术，是网状数据库的典型代表。在 DBTG 思想和方法的指引下数据库系统的实现技术不断成熟，开发了许多商品化的数据库系统，它们都是基于层次模型和网状模型的。

可以说，层次数据库是数据库系统的先驱，而网状数据库则是数据库概念、方法、技术的奠基者。

2. 第二代数据库系统

第二代数据库系统是关系数据库系统。1970 年 IBM 公司的 San Jose 研究试验室的研究员 Edgar F. Codd 发表了题为《大型共享数据库数据的关系模型》的论文，提出了关系数据模型，开创了关系数据库方法和关系数据库理论，为关系数据库技术奠定了理论基础。Edgar F. Codd 于 1981 年被授予 ACM 图灵奖，以表彰他在关系数据库研究方面的杰出贡献。

20 世纪 70 年代是关系数据库理论研究和原型开发的时代，其中以 IBM 公司的 San Jose 研究试验室开发的 System R 和 Berkeley 大学研制的 Ingres 为典型代表。大量的理论成果和实践经验终于使关系数据库从实验室走向了社会，因此，人们把 20 世纪 70 年代称为数据库时代。20 世纪 80 年代几乎所有新开发的系统均是关系型的，其中涌现出了许多性能优良的商品化关系数据库管理系统，如 DB2、Ingres、Oracle、Informix、Sybase 等。这些商用数据库系统的应用使数据库技术日益广泛地应用到企业管理、情报检索、辅助决策等方面，成为实现和优化信息系统的基本技术。

3. 第三代数据库系统

从 20 世纪 80 年代以来，数据库技术在商业上的巨大成功刺激了其他领域对数据库技术需求的迅速增长。这些新的领域为数据库应用开辟了新的天地，并在应用中提出了一些新的数据管理的需求，推动了数据库技术的研究与发展。

1990 年高级 DBMS 功能委员会发表了《第三代数据库系统宣言》，提出了第三代数据库管理系统应具有的三个基本特征：

- 应支持数据管理、对象管理和知识管理。
- 必须保持或继承第二代数据库系统的技术。
- 必须对其他系统开放。

面向对象数据模型是第三代数据库系统的主要特征之一；数据库技术与多学科技术的有机结合也是第三代数据库技术的一个重要特征。分布式数据库、并行数据库、工程数据库、演绎数据库、知识库、多媒体库、模糊数据库等都是这方面的实例。

1.1.4 数据库系统访问技术

目前访问数据库服务器的主流标准接口主要有 ODBC、OLE DB 和 ADO。下面分别对这三种接口进行概要介绍。

1. 开放数据库连接(ODBC)

开放数据库连接(Open Database Connectivity，ODBC)是由 Microsoft 公司定义的一种数据库访问标准。使用 ODBC 应用程序不仅可以访问存储在本地计算机的桌面型数据库中的数据，而且可以访问异构平台上的数据库，例如可以访问 SQL Server、Oracle、Informix 或 DB2 构建的数据库等。

ODBC 是一种重要的访问数据库的应用程序编程接口(Application Programming Interface,API)，基于标准的 SQL 语句，它的核心就是 SQL 语句，因此，为了通过 ODBC 访问数据库服务器，数据库服务器必须支持 SQL 语句。

ODBC 通过一组标准的函数(ODBC API)调用来实现数据库的访问,但是程序员不必理解这些 ODBC，API 就可以轻松开发基于 ODBC 的客户机/服务器应用程序。这是因为在很多流行的程序开发语言中，如 Visual Basic、PowerBuilder、Visual C++等，都提供了封装 ODBC 各种标准函数的代码层，开发人员可以直接使用这些标准函数。

ODBC 获得了巨大成功并大大简化了一些数据库开发工作。但是它也存在严重的不足，因此 Microsoft 公司又开发了 OLE DB。

2. OLE DB

OLE DB 是 Microsoft 公司提供的关于数据库系统级程序的接口(System-Level Programming Interface)，是 Microsoft 公司数据库访问的基础。OLE DB 实际上是 Microsoft 公司 OLE 对象标准的一个实现。OLE DB 对象本身是 COM(组件对象模型)对象并支持这种对象的所有必需的接口。

一般说来，OLE DB 提供了两种访问数据库的方法：一种是通过 ODBC 驱动器访问支持 SQL 语言的数据库服务器；另一种是直接通过原始的 OLE DB 提供程序。因为 ODBC 只适用于支持 SQL 语言的数据库，因此 ODBC 的使用范围过于狭窄，目前 Microsoft 公司正在逐步用 OLE DB 来取代 ODBC。

因为 OLE DB 是一个面向对象的接口，特别适合于面向对象语言。然而，许多数据库应用开发者使用 VBScript 和 JScript 等脚本语言开发程序，所以 Microsoft 公司在 OLE DB 对象的基础上定义了 ADO。

3. 动态数据对象(ADO)

动态数据对象(Active Data Objects，ADO)是一种简单的对象模型，可以被开发者用来处理任何 OLE DB 数据，可以由脚本语言或高级语言调用。ADO 对数据库提供了应用程序水平级的接口(Application-Level Programming Interface)，几乎使用任何语言的程序员都能够通过使用 ADO 来使用 OLE DB 的功能。Microsoft 公司声称，ADO 将替换其他的数据访问方式，所以 ADO 对于任何使用 Microsoft 公司产品的数据库应用是至关重要的。

1.1.5 网络数据库系统编程技术

在当今网络盛行的年代，数据库与 Web 技术的结合正在深刻改变着网络应用。有了数据库的支持，扩展网页功能、设计交互式页面、构造功能强大的后台管理系统、更新网站和维护网站都将变得轻而易举。随着网络应用的深入，Web 数据库技术将日益显示出其重要地位。在这里简单介绍一下 Web 数据库开发的相关技术。

1. 通用网关接口(CGI)编程

通用网关接口(Common Gateway Interface，CGI)是一种通信标准，它的任务是接受客户端的请求，经过辨认和处理，生成 HTML 文档并重新传回到客户端。这种交流过程的编程就叫做 CGI 编程。CGI 可以运行在多种平台上，具有强大的功能，可以使用多种语言编程，如 Visual Basic、Visual C++、Tcl、Perl、AppletScript 等，比较常见的是用 Perl 语言编写的 CGI 程序。但是 CGI 也有其致命的弱点，即速度慢和安全性差等。

2. 动态服务器页面(ASP)

动态服务器页面(Active Server Pages，ASP)是 Microsoft 公司推出的一种用以取代 CGI 的技术，是一种真正简便易学、功能强大的服务器编程技术。ASP 实际上是 Microsoft 公司开发的一套服务器端脚本运行环境，通过 ASP 可以建立动态的、交互的、高效的 Web 服务器应用程序。用 ASP 编写的程序都在服务器端执行，程序执行完毕后，再将执行的结果返回给客户端浏览器，这样不仅减轻了客户端浏览器的负担，大大提高了交互速度，而且避免了 ASP 程序源代码的外泄，提高了程序的安全性。

3. Java 服务器页面(JSP)

Java 服务器页面(Java Server Pages，JSP)是 Sun 公司发布的 Web 应用程序开发技术，一经推出，就受到了人们的广泛关注。JSP 技术为创建高度动态的 Web 应用程序提供了一个独特的开发环境，它能够适用于市场上大多数的服务器产品。

JSP 使用 Java 语言编写服务器端程序，当客户端向服务器发出请求时，JSP 源程序被编译成 Servlet 并由 Java 虚拟机执行。这种编译操作仅在对 JSP 页面的第一次请求时发生。因此，JSP 程序能够提供更快的交互速度，其安全性和跨平台性也很优秀。

1.2 关系数据库的设计理论

关系是数学上集合论中的一个重要概念。1970 年，E.F.Codd 发表了题为《大型共享数据库数据

的关系模型》的论文，把关系的概念引入了数据库，自此人们开始了数据库关系方法和关系数据理论的研究，在层次和网状数据库系统之后，形成了以关系数据模型为基础的关系数据库系统。

1.2.1 数据模型及其三要素

所谓信息，是客观事物在人类头脑中的抽象反 映。人们可以从大千世界中获得各种各样的信息，从而了解世界并且相互交流。但是信息的多样化特性使人们在描述和管理这些数据时往往力不从心，因此人们把表 示事物的主要特征抽象地用一种形式化的描述表示出来，模型方法就是这种抽象的一种表示。信息领域中采用的模型通常称为数据模型。

数据模型是实现数据抽象的主要工具。它决定了数据库系统的结构、数据定义语言和数据操纵语言、数据库设计方法、数据库管理系统软件的设计与实现。了解关于数据模型的基本概念是学习数据库的基础。

一般地讲，数据模型是严格定义的概念的集合。这些概念精确地描述系统的静态特性、动态特性和完整性约束条件。因此，数据模型通常由数据结构、数据操作和数据的完整性约束三部分组成。

1. 数据结构

数据结构是研究存储在数据库中的对象类型的集合，这些对象类型是数据库的组成部分。例如，在学校中要管理学生的基本情况(学号、姓名、出生年月、院系、班级、选课情况等)，这些基本情况说明了每一个学生的特性，构成在数据库中存储的框架，即对象类型。学生在选课时，一个学生可以选多门课程，一门课程也可以被多名学生所选，这类对象之间存在着数据关联，这种数据关联也要存储在数据库中。

数据库系统是按数据结构的类型来组织数据 的，因此数据库系统通常按照数据结构的类型来命名数据模型。例如，层次结构、网状结构和关系结构的模型分别命名为层次模型、网状模型和关系模型。由于采用 的数据结构类型不同，通常把数据库分为层次数据库、网状数据库、关系数据库和面向对象数据库等。

数据结构是对系统静态特性的描述。

2. 数据操作

数据操作是指对数据库中各种对象的实例允许执行的操作的集合，包括操作和有关操作的规则。例如插入、删除、修改、检索、更新等操作，数据模型要定义这些操作的确切涵义、操作符号、操作规则以及实现操作的语言等。

数据操作是对系统动态特性的描述。

3. 数据的完整性约束

数据的完整性约束条件是完整性规则的集合，用于限定符合数据模型的数据库状态，及状态的变化，以保证数据的正确、有效和相容。数据模型中的数据及其联系都要遵循完整性规则的制约。例如数据库的主键不能允许空值；每一个月的天数最多不能超过 31 天等。

另外，数据模型应该提供定义完整性约束条件的机制以反映某一应用所涉及的数据必须遵守的特定语义约束条件。例如在学生成绩管理中，本科生的累计成绩不得有三门以上不及格等。

数据模型是数据库技术的关键，它的三方面内容完整地描述了一个数据模型。实际数据库系统中所支持的主要数据模型是层次模型(Hierarchical Model)、网状模型(Network Model)和关系模型(Relational Model)。

1.2.2 关系模型

关系模型是三种数据模型中最重要的一种，数据库领域中当前的研究工作也都是以关系方法为基础的。本书把重点放在关系数据库上。

1. 关系模型的概念

在现实世界中，人们经常用表格形式表示数据信息。但是日常生活中使用的表格往往比较复杂，在关系模型中基本数据结构被限制为二维表格。因此，在关系模型中，数据在用户观点下的逻辑结构就是一张二维表。每一张二维表称为一个关系(Relation)。二维表中存放了两类数据：实体本身的数据和实体间的联系。这里的联系是通过不同的关系中具有相同的属性名来实现的。例如表 1.1 在学生

情况关系中存储了学生的学号、姓名、年龄和班级号等信息，表 1.2 在班级情况关系中存储了班级的班级号(假设班级号惟一)、专业、院系等信息。这两个关系通过班级号来实现二者之间一对多的联系。

表 1.1 学生情况表

学 号	姓 名	年 龄	班 级 号
2005001	张三	23	001
2005002	李四	22	001
2005003	王五	24	012
2005004	赵六	22	024
……	……	……	……

表 1.2 班级情况表

班 级 号	专 业	院 系
001	计算机应用	计算机系
003	车辆工程	车辆工程学院
续表		
班 级 号	专 业	院 系
012	电力系统	电气工程学院
024	经济管理	经济管理学院
……	……	……

2. 关系模型的特点

关系模型具有下列特点：

- 关系模型的概念单一。无论是实体还是实体之间的联系都用关系来表示。关系之间的联系通过相容的属性来表示，相容的属性即来自同一个取值范围的属性。在关系模 型中，用户看到的数据的逻辑结构就是二维表，而在非关系模型中，用户看到的数据结构是由记录及记录之间的联系所构成的网状结构或层次结构。当应用环境很复 杂时，关系模型就体现出其简单清晰的特点。
- 关系必须是规范化的关系。所谓规范化是指关系模型中的每一个关系模型都要满足一定的要求(或者称为规范条件)。最基本的一个规范条件是每一个分量都是一个不可分的数据项，即表中不允许还有表。
- 在关系模型中，用户对数据的检索操作就是从原来的表中得到一张新的表。
- 由于关系模型概念简单、清晰，用户易懂易用，有严格的数学基础，以及在此基础上发展的关系数据理论，简化了程序员的工作和数据库开发建立的工作，因而关系模型自诞生之日起迅速发展成熟起来，成为深受用户欢迎的数据模型。

1.2.3 实体类型的属性关系

关系数据库是以关系模型为基础的数据库，它利用关系和关系中的属性来描述现实世界。一个关系就是现实世界中的一个实体，而属性则代表了该实体的某种性质。例如，实体“学生”可以用关系 Student (SNO，SNAME，SEX，AGE，MAJOR，…)来描述，其中属性 SNO、SNAME、SEX、MAJOR 分别代表了学号、姓名、年龄、性别、专业等“学生”固有的性质。事实上，关系除了描述实体之外，还可以描述实体之间的联系。例如，男女之间“婚姻”关系，就可以用这样一个关系来抽象：MARRAY(MALE_NAME，FEMALE_NAME，…)。关系架起了现实数据与数学方法之间的桥梁。

实体的内部存在着一对一、一对多和多对多三种类型的联系，与之相应，关系的各属性之间也存在着 3 种联系模式。

1. 一对一关系(1:1)

在日常生活中，关系属性之间的一一对应关系是很常见的。比如，对于某个只有一个站台的车站来说，不同的火车车次肯定有不同的进站时间，则在关系“车次(车次号、到站时间、到站站台，……)”

中，属性“车次”和“到站时间”便是一对一的关系，记作 1:1。又如，网络设备中，网卡的硬件地址和网卡本身也是一一对应的。

在关系数据库设计理论中，以 1:1 对应的关系是这样定义的：A、B 为某个实体类型的两个属性的域(取值范围)，如果对于 A 中的任一具体值，B 中至少有一个值与之对应，B 中的任一具体值，A 中也至多有一值与之对应，则称 A、B 这两个属性之间为 1:1 对应的 关系。

2. 一对多关系(1:∞)

在一个实体类型中，如果它的一个实体属性集 A 中的一个值至多与另一个实体属性 B 中的一个值有关，而 B 中的一个值却与 A 中的多个值有关，则称 A、B 两属性之间的关系为一对多的关系。例如，如果一个人有多个住处，则这个人的身份证号与他的地址便满足一对多的关系。

3. 多对多关系(∞:∞)

在一个实体类型中，如果它的两个属性值集合中的任一值都与另一属性值集合中的多个值有关，则称两个属性之间是多对多的关系。例如，在大学中一名教师可以教授多门课程，而一门课程也可以由多名教师来教授，则教师的姓名和课程的名称之间便是多对多的关系。

1.3 数据库系统的设计过程

随着计算机技术的广泛应用，目前从小型的单项事务处理到大型的信息系统都采用数据库技术来保持数据的完整性和一致性，因此在应用系统的设计中，数据库搭建得是否合理变得日趋重要。

具体地说，数据库设计是指针对一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求(信息要求和处理要求)。数据库设计是数据库在应用领域的主要研究课题。

数据库设计包括静态数据模型(即模式与子模式)的设计，称为数据库的结构设计；以及在模型上的动态操作(即应用程序)设计称为数据设计。现代数据库设计十分强调数据库的结构设计与行为设计的统一。

目前所说的数据库设计大多是在一个现成的 DBMS 的支持下进行，即以一个通用的 DBMS 为基础开发数据库应用系统。

目前，数据库设计一般都遵循软件的生命周期理论，分为六个阶段进行，如图 1.1 所示，即需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施和数据库的运行与维护。其中，需求分析和概念结构设计独立于任何的 DBMS 系统，而逻辑结构设计和物理结构设计则与具体的 DBMS 有关。

一个完善的数据库设计不可能一蹴而就，在每一设计阶段完成后都要进行设计分析，评价一些重要的设计指标，与用户进行交流，如果不满足要求则进行修改。在设计过程中，这种评价和修改可能要重复若干次，以求得理想的结果。

由于数据库设计是一个复杂而烦琐的过程，近年来，许多软件厂商通过研究，开发了不少计算机辅助数据库开发工具，如 CA 公司的 ERWin 和 Sybase 公司的 PowerDesign。在设计过程中适当利用这些工具可以提高数据设计的效率和质量。

下面将详细讲解数据库设计过程中各阶段的设计内容与设计方法。

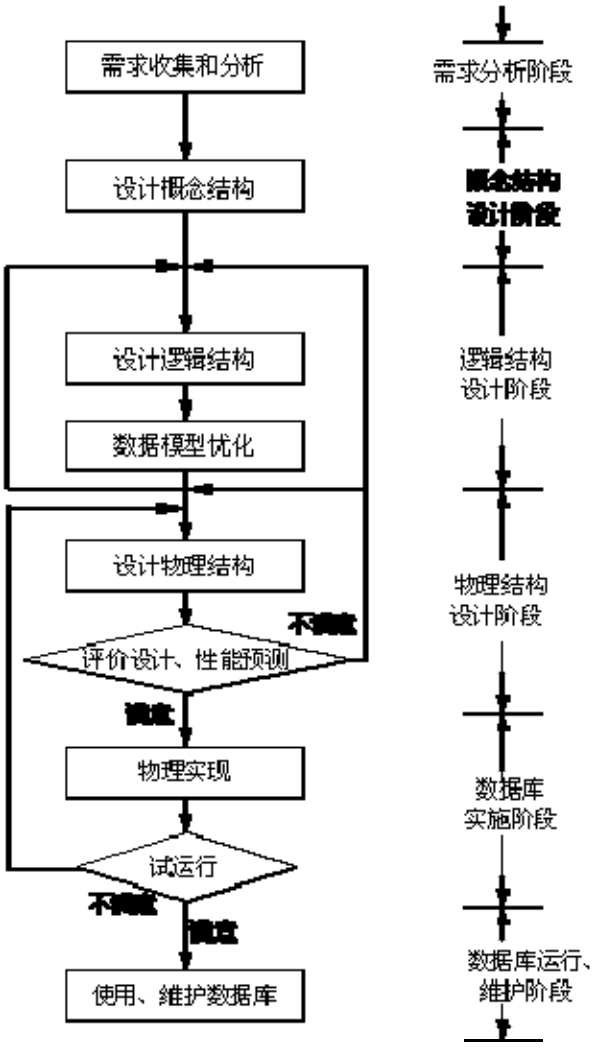


图 1.1 设计库的设计流程

1.3.1 需求分析

设计一个性能良好的数据库系统，明确应用环境对系统的要求是首要的和基本的。因此，应该把对用户需求的收集和分析作为数据库设计的第一步。

需求分析的主要任务是通过详细调查要处理的 对象，包括某个组织、某个部门、某个企业的业务管理等，充分了解原手工或原计算机系统的工作概况及工作流程，明确用户的各种需求，产生数据流图和数据字 典，然后在此基础上确定新系统的功能，并产生需求说明书。值得注意的是，新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

如图 1.2 所示，需求分析具体可按以下几步进行：

(1) 用户需求的收集。

(2) 用户需求的分析。

(3) 撰写需求说明书。

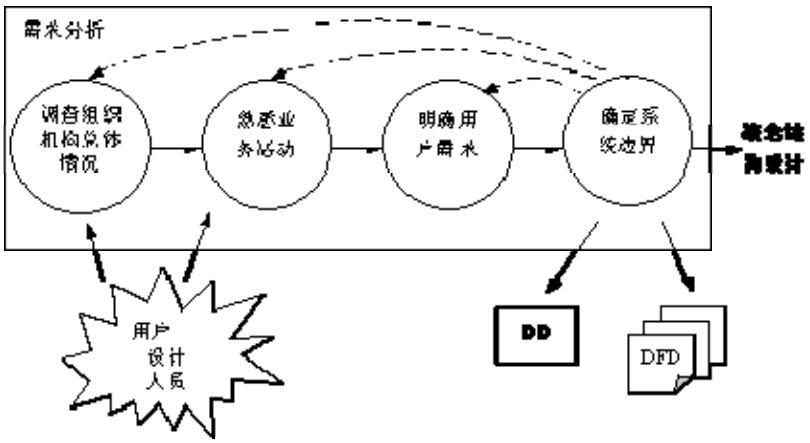


图 1.2 需求分析的过程

需求分析的重点是调查、收集和分析用户数据管理中的信息需求、处理需求、安全性与完整性要求。信息需求是指用户需要从数据库中获得的信息的内容和性质。由用 户的信息需求可以导出数据需求，即在数据库中应该存储哪些数据。处理需求是指用户要求完成什么处理功能，对某种处理要求的响应时间，处理方式指是联机处理 还是批处理等。明确用户的处理需求，将有利于后期应用程序模块的设计。

调查、收集用户要求的具体做法是：

(1)了解组织机构的情况，调查这个组织由哪些部门组成，各部门的职责是什么，为分析信息流程做准备。

(2)了解各部门的业务活动情况，调查各部门输入和使用什么数据，如何加工处理这些数据。输出什么信息，输出到什么部门，输出的格式等。在调查活动的同时，要注意对各种资料的收集，如票证、单据、报表、档案、计划、合同等，要特别注意了解这些报表之间的关系，各数据项的含义等。

(3)确定新系统的边界。确定哪些功能由计算机完成或将来准备让计算机完成，哪些活动由人工完成。由计算机完成的功能就是新系统应该实现的功能。

在调查过程中，根据不同的问题和条件，可采用的调查方法很多，如跟班作业、咨询业务权威、设计调查问卷、查阅历史记录等。但无论采用哪种方法，都必须有用户的积极参与和配合。强调用户的参与是数据库设计的一大特点。

收集用户需求的过程实质上是数据库设计者对 各类管理活动进行调查研究的过程。设计人员与各类管理人员通过相互交流，逐步取得对系统功能的一致的认识。但是，由于用户还缺少软件设计方面的专业知识， 而设计人员往往又不熟悉业务知识，要准确地确定需求很困难，特别是某些很难表达和描述的具体处理过程。针对这种情况，设计人员在自身熟悉业务知识的同时， 应该帮助用户了解数据库设计的基本概念。对于那些因缺少现成的模式、很难设想新的系统、不知应有哪些需求的用户，还可应用原型化方法来帮助用户确定他们的 需求。就是说，先给用户一个比较简单的、易调整的真实系统，让用户在熟悉使用它的过程中不断发现自己的需求，而设计人员则根据用户的反馈调整原型，反复验 证最终协助用户发现和确定他们的真实需求。

调查了解用户的需求后，还需要进一步分析和抽象用户的需求，使之转换为后续各设计阶段可用的形式。在众多分析和表达用户需求的方法中，结构化分析(Structured Analysis，SA)是一个简单实用的方法。SA 方法采用自顶向下，逐层分解的方式分析系统，用数据流图(Data Flow Diagram，DFD)、数据字典(Data Dictionary，DD)描述系统。

1. 使用数据流图分析信息处理过程

数据流图是软件工程中专门描绘信息在系统中流动和处理过程的图形化工具。因为数据流图是逻辑系统的图形表示，即使不是专业的计算机技术人员也容易理解，所以是极好的交流工具。图 1.3 给出了数据流图中所使用的符号及其含义。

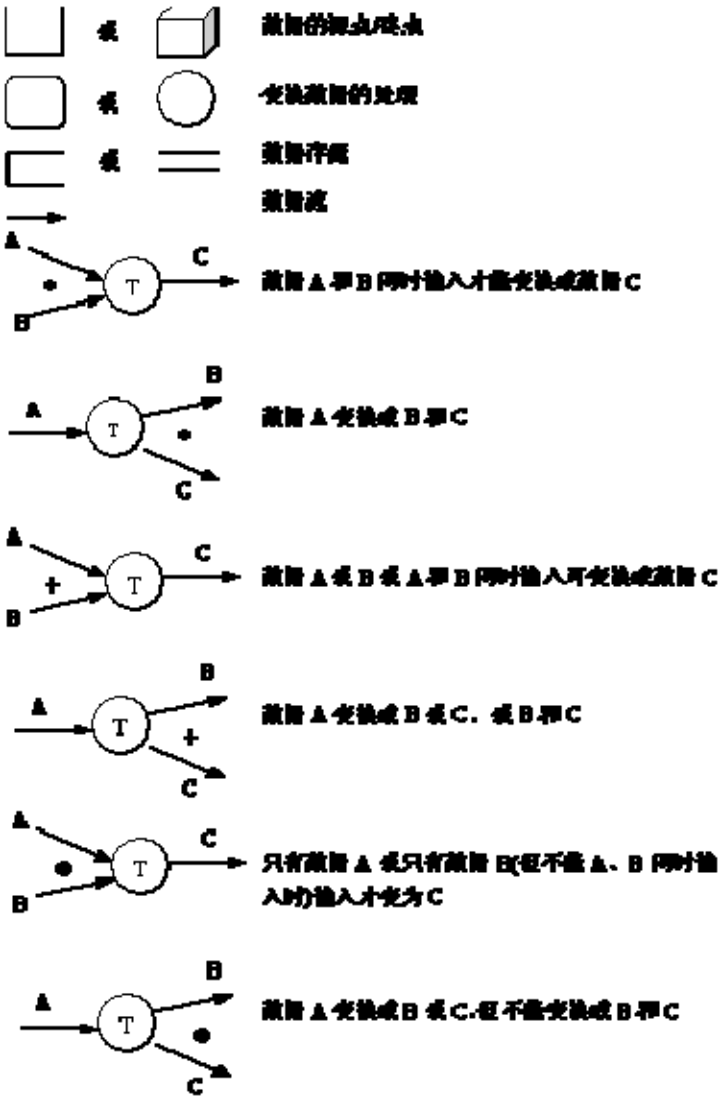


图 1.3　数据流图的符号

数据流图是有层次之分的，越高层次的数据流图表现的业务逻辑越抽象，越低层次的数据流图表现的业务逻辑则越具体。在 SA 方法中，我们可以把任何一个系统都抽象为　图 1.4 所示的形式。它是最高层次抽象的系统概貌，要反映更详细的内容，可将处理功能分解为若干子功能，每个子功能还可继续分解，直到把系统工作过程表示清楚为止。在处理功能逐步分解的同时，它们所用的数据也逐级分解，形成若干层次的数据流图，如图 1.5 所示。

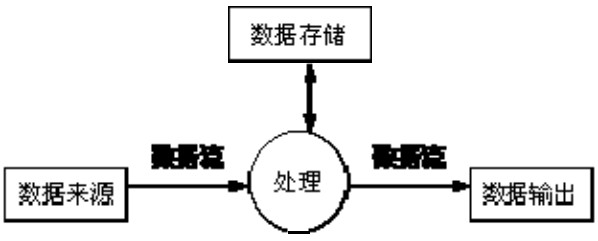


图 1.4　系统高层抽象图

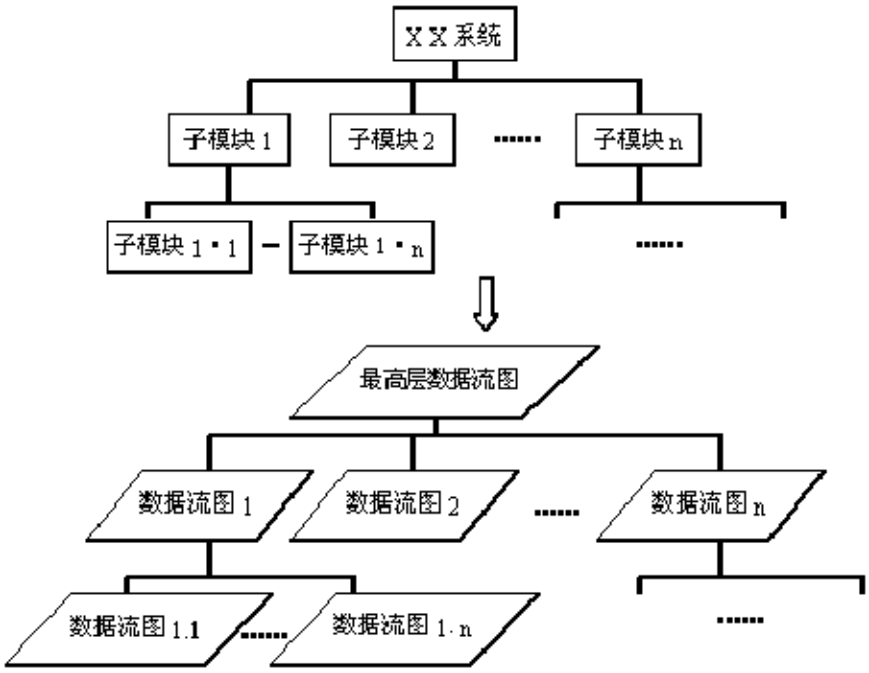


图 1.5 数据流图的建立

为了更好地说明 SA 方 法，下面举一个简单的设计实例。假定某工厂要设计一个数据库查询系统。其中，主管生产的部门要掌握产品的性能、各种零件的用料和每种产品的零件组成情况， 并据此编制工厂的生产计划。主管供应的部门需要了解产品的价格、各种零件的用料情况以及这些材料的价格与库存量，并根据这些资料提出材料的采购计划。

在调查、分析用户的业务活动并确定系统边界后，得到了如图 1.6 和图 1.7 所示的业务流程图。

根据实际应用环境对系统在功能上进行分解，如图 1.8 所示。

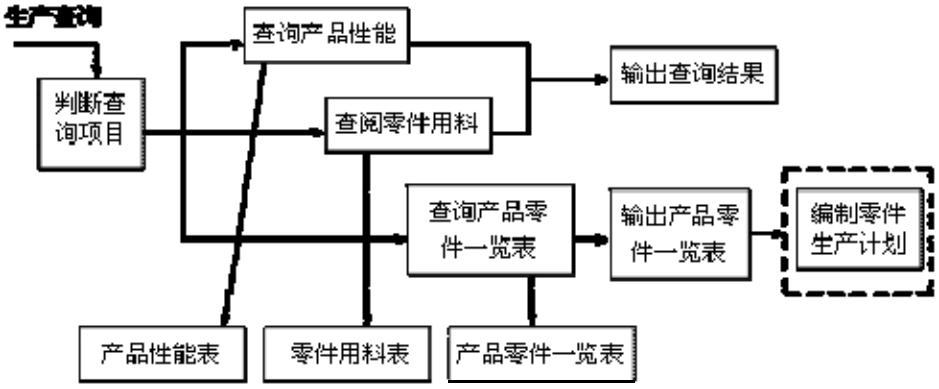


图 1.6 生产部门业务流程图

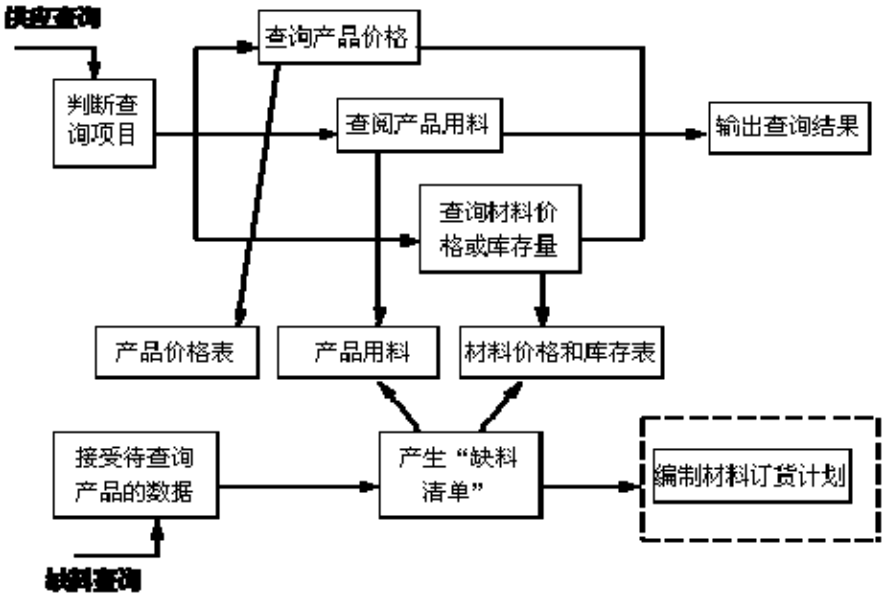


图 1.7　供应部门业务流程图

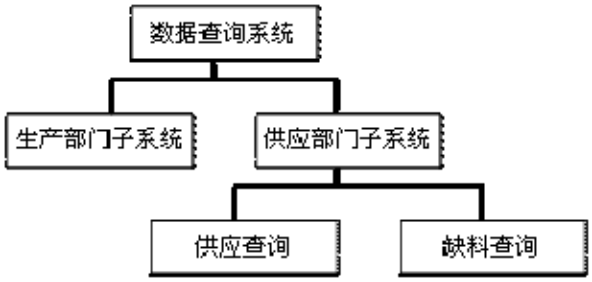


图 1.8　系统功能的分解

在功能分解的同时，不断细化数据流图。这里 以“缺料查询”为例，经过需求分析得到如下信息：系统一旦收到缺料查询请求，接受所查询产品的有关数据后，首先要查询单件产品的用料数据，并按产品生产数 量算出各种材料的需用量，然后从库存记录中减去这一需用量，即可找出缺料的名称和数量。根据语义，画出相应的数据流图，如图 1.9 所示。显然，每一种应用需要一张这样的流程图。

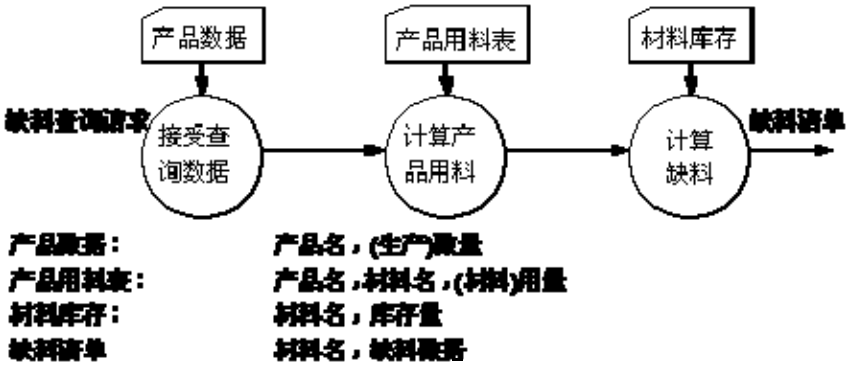


图 1.9　缺料查询数据流图

2. 使用数据字典汇总各类数据

数据字典是结构化设计方法的另一个工具，它用来对系统中的各类数据进行详尽的描述。对数据库设计来讲，数据字典是进行详细的数据收集和数据分析所获得的主要成果。数据字典中的内容在数据库设计过程中还要不断的修改、充实、完善。

数据字典是各类数据描述的集合，它通常包括以下 5 个部分。

- 数据项：数据项是数据最小的组成单位。
- 数据结构：是若干数据项有意义的集合。它反映了数据之间的组合关系。
- 数据流：可以是数据项，也可以是数据结构。表示某一处理过程的输入和输出。
- 数据存储：处理过程中存储的数据。常常是手工凭证、手工文档或计算机文件。
- 处理过程。

表 1.3 是一个数据字典的示例，它描述了缺料查询数据流图(图 1.9)中一些数据项。

表 1.3 数据字典示例

数 据 项	类 型	长 度	值 范 围
产品名	字符型	20	任何字母和数字
材料名	字符型	10	任何字母和数字
库存量	正整数	5	0～99 999
缺料数量	正整数	5	0～99 999
生产数量	正整数	3	0～999

1.3.2 概念结构设计

概念结构设计的任务是在需求分析阶段产生的需求说明书的基础上，按照特定的方法把它们抽象为一个不依赖于任何具体机器的数据模型，即概念模型。概念模型使设计者的注意力能够从复杂的实现细节中解脱出来，而只集中在最重要的信息的组织结构和处理模式上。

概念模型具有以下的特点：

1　概念模型是对现实世界的抽象和概括，它真实、充分地反映了现实世界中事物和事物之间的联系，能满足用户对数据的处理要求。

1　由于概念模型简洁、明晰、独立于计算机，很容易理解，因此可以用概念模型和不熟悉计算机的用户交换意见，使用户能积极参与数据库的设计工作，保证设计工作顺利进行。

1　概念模型易于更新，当应用环境和应用要求改变时，容易对概念模型修改和扩充。

1　概念模型很容易向关系、网状、层次等各种数据模型转换。

描述概念模型的有力工具是 E-R 图。E-R 模型是一个面向问题的概念模型，即用简单的图形方式(E-R 图)描述现实世界中的数据。这种描述不涉及数据在数据库中表示和存取方法，非常接近人的思维方式。后来又提出了扩展实体联系模型(Extend Entity-Relationship Model)，简称为“EER 模型”。EER 模型目前已经成为一种使用广泛的概念模型，为面向对象的数据库设计提供了有效的工具。

在 E-R 模型中，信息由实体型、实体属性和实体间的联系三种概念单元来表示。

实体型表示建立概念模型的对象，用长方形表示，在框内写上实体名。例如学生实体用图 1.10 表示。

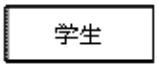


图 1.10 学生实体表示方法

实体属性是实体的说明。用椭圆形表示实体的属性，并用无向边把实体与其属性连接起来。例如学生实体有学号、姓名、年龄、性别、出生年月等属性，则其 E-R 图如图 1.11 所示。

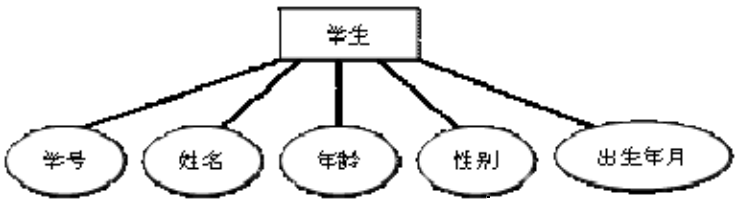


图 1.11 学生实体及其属性

实体间的联系是两个或两个以上实体类型之间的有名称的关联。实体间的联系用菱形表示，菱形

内要有联系名，并用无向边把菱形分别与有关实体相连接，在无向边旁标上联系的类型。例如可以用 E-R 图来表示某学校学生选课情况的概念模型，如图 1.12 所示。

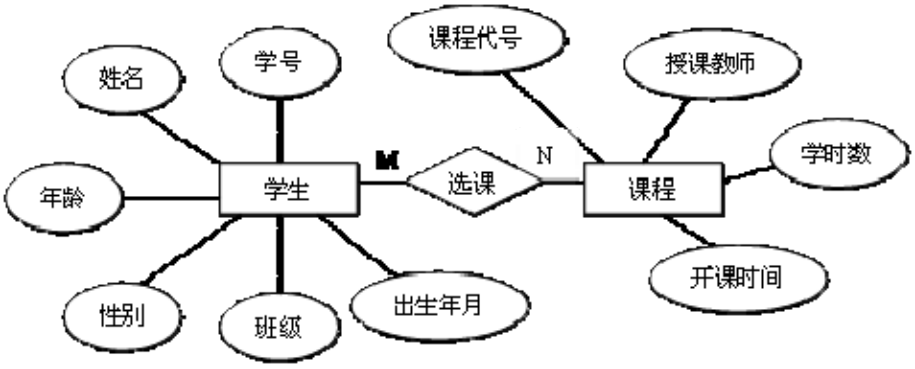


图 1.12　实体、实体属性及实体联系模型图

一个学生可以选修多门课程，一门课程也可以被多个学生选修，因此，学生和课程之间具有多对多的联系。

如果概念模型中涉及的实体带有较多的属性而使实体联系图非常不清晰，可以将实体联系图分成两部分，一部分是实体及其属性图，另一部分是实体及其联系图。如图 1.13 所示，只给出学生实体与课程实体的联系图，而二者的属性可以单独画出。

图 1.13　实体及其联系图

1.3.3 逻辑结构设计

概念结构设计所得的 E-R 模型是对用户需求的一种抽象的表达形式，它独立于任何一种具体的数据模型，因而也不能为任何一个具体的 DBMS 所支持。为了能够建立起最终的物理系统，还需要将概念结构进一步转化为某一 DBMS 所支持的数据模型，然后根据逻辑设计的准则、数据的语义约束、规范化理论等对数据模型进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式。这就是数据库逻辑设计所要完成的任务。

数据库逻辑结构的设计分为两个步骤：首先将概念设计所得的 E-R 图转换为关系模型；然后对关系模型进行优化，如图 1.14 所示。

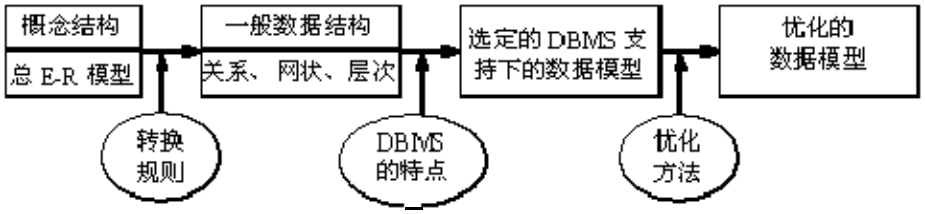


图 1.14　逻辑结构设计的过程

关系模型是由一组关系(二维表)的结合，而 E-R 模型则是由实体、实体的属性、实体间的关系三个要素组成。所以要将 E-R 模型转换为关系模型，就是将实体、属性和联系都要转换为相应的关系模型。下面具体介绍转换的规则。

1. 一个实体类型转换为一个关系模型

将每种实体类型转换为一个关系，实体的属性就是关系的属性，实体的关键字就是关系的关键字。例如，可将“学生”实体转换为一个关系模型，如图 1.15 所示。其中，带下划线的属性为主属性，该主属性为关系模型外键。

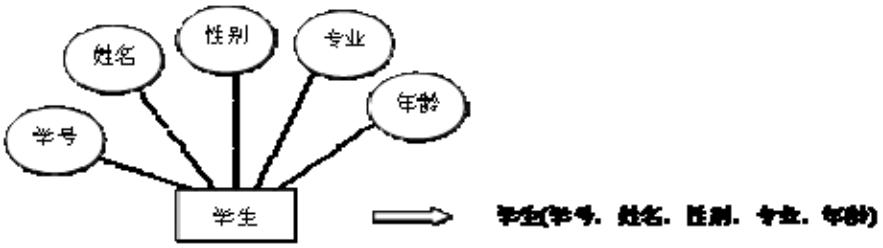


图 1.15　一个实体类型转换为一个关系模型

2. 一对一关系(1:1)的转换

一对一关系有以下两种转换方式：

l　　转换为一个独立的关系模型。联系名为关系模型名，与该联系相连的两个实体的关键字及联系本身的属性为关系模型的属性，其中每个实体的关键字均是该关系模型的候选键。

l　　与任意一端的关系模型合并。可将相关的两个实体分别转换为两个关系，并在任意一个关系的属性中加入另一个关系的主关键字。

例如，若某工厂的每个仓库只配备了一名管理员，那么仓库实体与管理员实体间便为 1:1 关系。根据以上介绍的原则，可以进行如图 1.16 所示的变换。

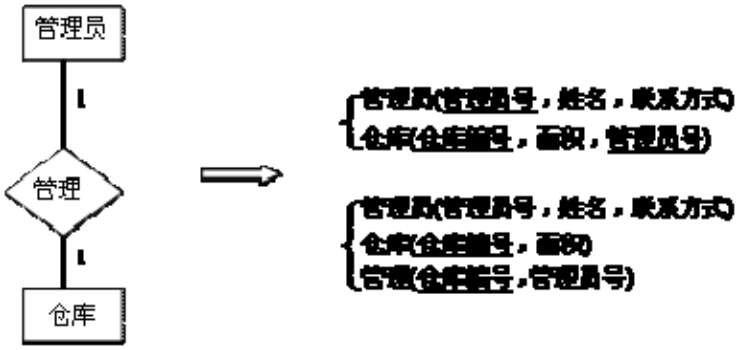


图 1.16　1:1 关系的转换

在实际设计中究竟采用哪种方案可视具体的应 用而定。如果经常要在查询仓库关系的同时查询此仓库管理员的信息，就可选用前一种关系模型，以减少查询时的连接操作。反之，如果在查询管理员时要频繁查询 仓库信息，则选用后一种关系模型。总之，在模型转换出现较多方案时，效率是重要的取舍因素。

3. 一对多关系(1:n)的转换

一对多关系也有两种转换方式：

- 将 1:n 关系转换为一个独立的关系模型。联系名为关系模型名，与该联系相连的各实体的关键字及联系本身的属性为关系模型的属性，关系模型的关键字为 n 端实体的关键字。
- 将 1:n 联系与 n 端关系合并。1 端的关键字及联系的属性并入 n 端的关系模型　即可。

在图 1.17 中，实体“专业”和“学生”之间的联系为 1:n，则两者可使用以上的原则进行关系模型的转换。

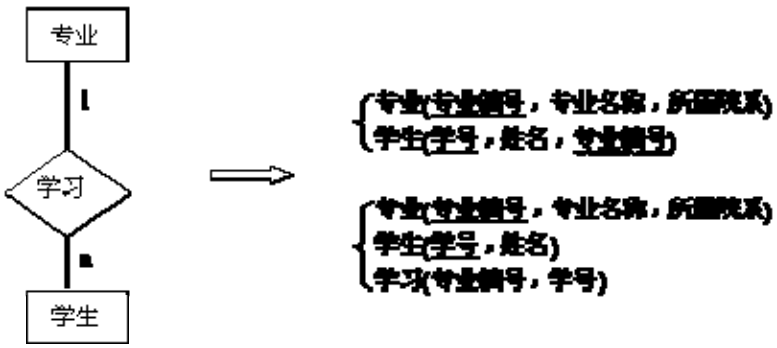


图 1.17　1:n 联系的转换

4. 多对多关系(m:n)的转换

关系模型名为关系名，与该关系相连的各实体的关键字及关系本身的属性为关系模型的属性，关系模型的关键字为关系中各实体关键字的并集。

例如，在学校中，一名学生可以选修多门课程，一门课程也可为多名学生选修，则实体“学生”与“课程”之间满足多对多的关系，其转换方法如图 1.18 所示。

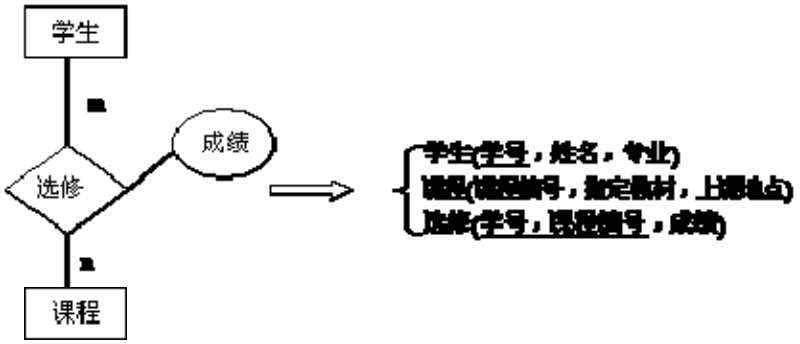


图 1.18　m:n 关系的转换

1.3.4 物理结构设计

数据库物理设计阶段的任务是根据具体计算机系统(DBMS 和硬件等)的特点，为给定的数据库模型确定合理的存储结构和存取方法。所谓的“合理”主要有两个含义：一个是要使设计出的物理数据库占用较少的存储空间，另一个对数据库的操作具有尽可能高的速度。

为了设计数据库的物理结构，设计人员必须充分了解所用 DBMS 的内部特征；充分了解数据系统的实际应用环境，特别是数据应用处理的频率和响应时间的要求；充分了解外存储设备的特性。数据库的物理结构设计大致包括：确定数据的存取方法、确定数据的存储结构。

物理结构设计阶段实现的是数据库系统的内模式，它的质量直接决定了整个系统的性能。因此在确定数据库的存储结构和存取方法之前，对数据库系统所支持的事务要进行仔细分析，获得优化数据库物理设计的参数。

对于数据库查询事务，需要得到如下信息：

- 要查询的关系。
- 查询条件(即选择条件)所涉及的属性。
- 连接条件所涉及的属性。
- 查询的投影属性。

对于数据更新事务，需要得到如下信息：

- 要更新的关系。
- 每个关系上的更新操作的类型。
- 删除和修改操作所涉及的属性。
- 修改操作要更改的属性值。

上述这些信息是确定关系存取方法的依据。除此之外，还需要知道每个事务在各关系上运行的频率，某些事务可能具有严格的性能要求。例如，某个事务必须在 20 秒内结束。这种时间约束对于存取方法的选择有重大的影响。需要了解每个事务的时间约束。

值得注意的是,在进行数据库物理结构设计时,通常并不知道所有的事务,上述信息可能不完全。所以，以后可能需要修改根据上述信息设计的物理结构，以适应新事务的要求。

1. 确定关系模型的存取方法

确定数据库的存取方法，就是确定建立哪些存储路径以实现快速存取数据库中的数据。现行的 DBMS 一般都提供了多种存取方法，如索引法、HASH 法等。其中，最常用的是索引法。

数据库的索引类似书的目录。在书中，目录允 许用户不必浏览全书就能迅速地找到所需要的位

置。在数据库中，索引也允许应用程序迅速找到表中的数据，而不必扫描整个数据库。在书中，目录就是内容和相应 页号的清单。在数据库中，索引就是表中数据和相应存储位置的列表。使用索引可以大大减少数据的查询时间。

但需要注意的是索引虽然能加速查询的速度，但是为数据库中的每张表都设置大量的索引并不是一个明智的做法。这是因为增加索引也有其不利的一面：首先，每个索引都将占用一定的存储空间，如果建立聚簇索引(会改变数据物理存储位置的一种索引)，占用需要的空间就会更大；其次，当对表中的数据进行增加、删除和修改的时候，索引也要动态地维护，这样就降低了数据的更新速度。

- 在创建索引的时候，一般遵循以下的一些经验性原则：
- 在经常需要搜索的列上建立索引。
- 在主关键字上建立索引。
- 在经常用于连接的列上建立索引，即在外键上建立索引。
- 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的。
- 在经常需要排序的列上建立索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询的时间。
- 在经常成为查询条件的列上建立索引。也就是说，在经常使用在 WHERE 子句中的列上面建立索引。

同样，对于某些列不应该创建索引。这时候应该考虑下面的指导原则：

- 对于那些在查询中很少使用和参考的列不应该创建索引。因为既然这些列很少使用到，有索引并不能提高查询的速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。
- 对于那些只有很少值的列不应该建立索引。例如，人事表中的“性别”列，取值范围只有两项：“男”或“女”。若在其上建立索引，则平均起来，每个属性值对应一半的元组，用索引检索，并不能明显加快检索的速度。
- 属性值分布严重不均的属性。例如学生的年龄往往集中在几个属性值上，若在年龄属性上建立索引，则在检索某个年龄的学生时，会涉及相当多的学生。
- 过长的属性，例如超过 30 个字节。因为在过长的属性上建立索引，索引所占的存储空间较大，而索引的级数也随之增加，有诸多不利之处。如果实在需要在其上建立索引，必须采取索引属性压缩的措施。
- 经常更新的属性或表。因为在更新时有关的索引需要做相应的修改。

最后举个简单的例子，说明究竟哪些情况下需要建立索引以提高效率。假设，某个大学需要建立一个学生成绩的数据库系统，整个系统包括三个数据表：课程信息表、学生信息表和学生成绩表。数据库的结构如下：

- 学生信息表(学号、姓名、出生日期、性别、系名、班号)
- 课程信息表(课程号、课程名、教师、学分)
- 学生成绩表(学号、课程号、成绩)

整个系统需要统计学生的平均分、某课程的平均分等，所以学生信息表中的属性“学号”，课程信息表中的属性“课程号”，学生成绩表中的属性“学号”、“课程号”将经常出现在查询条件中，可以考虑在上面建立索引以提高效率。

2. 确定数据库的存储结构

确定数据库的存储结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、日志、备份等的存储安排及存储结构，以及确定系统存储参数的配置。

确定数据存放位置是按照数据应用的不同将数据库的数据划分为若干类，并确定各类数据的大小和存放位置。数据的分类可依据数据的稳定性、存取响应速度、存取频度、数据共享程度、数据保密程度、数据生命周期的长短、数据使用的频度等因素加以区别。

确定数据存放的位置主要是从提高系统性能的角度考虑。由于不同的系统和不同应用环境有不同的应用需求，所以在此只列出一些启发性的规则。

- 在大型系统中，数据库的数据备份、日志文件备份等数据只在故障恢复时才使用，而且数据量很大，可以考虑放在磁带上。
- 对于拥有多个磁盘驱动器或磁盘阵列的系统，可以考虑将表和索引分别存放在不同的磁盘上，在查询时，由于两个磁盘驱动器分别工作，因而可以保证物理读写速度比较快。
- 将比较大的表分别存放在不同的磁盘上，可以加快存取的速度，特别是在多用户的环境下。
- 将日志文件和数据库对象(表、索引等)分别放在不同的磁盘可以改进系统的性能。

由于各个系统所能提供的对数据进行物理安排的手段、方法差异很大，因此设计人员应该在仔细了解给定的 DBMS 在这方面提供了什么方法、系统的实际应用环境的基础上进行物理安排。

3. 确定系统存储参数的配置

现行的许多 DBMS 都设置了一些系统的配置变量，供设计人员和 DBA(数据库管理员)进行物理的优化。在初始情况下，系统都为这些变量赋予了合理的初值。但是这些值只是从产品本身特性出发，不一定能适应每一种应用环境，在进行物理结构设计时，可以重新对这些变量赋值以改善系统的性能。以 Microsoft 公司的 SQL Server 2000 为例，它为用户提供的配置变量包括：同时使用数据库的用户数、同时打开的数据库对象数，使用的缓冲区长度、个数，数据库的大小，索引文件的大小，锁的数目等。

应该指出，在物理结构设计对系统配置变量的调整只是初步的，在系统运行时还需要根据系统实际的运行情况做进一步的调整，以获得最佳的系统性能。

1.3.5 数据库的实施、运行和维护

在进行概念结构设计和物理结构设计之后，设计者对目标系统的结构、功能已经分析得较为清楚了，但这还只是停留在文档阶段。数据系统设计的根本目的，是为用户提供一个能够实际运行的系统，并保证该系统的稳定和高效。要做到这点，还有两项工作，就是数据库的实施、运行和维护。

1. 数据库的实施

数据库的实施主要是根据逻辑结构设计和物理结构设计的结果，在计算机系统上建立实际的数据库结构、导入数据并进行程序的调试。它相当于软件工程中的代码编写和程序调试的阶段。

用具体的 DBMS 提供的数据定义语言(DDL)，把数据库的逻辑结构设计和物理结构设计的结果转化为程序语句，然后经 DBMS 编译处理和运行后，实际的数据库便建立起来了。目前的很多 DBMS 系统除了提供传统的命令行方式外，还提供了数据库结构的图形化定义方式，极大地提高了工作的效率。

具体地说，建立数据库结构应包括以下几个方面：

- 数据库模式与子模式，以及数据库空间的描述。
- 数据完整性的描述。
- 数据安全性描述。
- 数据库物理存储参数的描述。

此时的数据库系统就如同刚竣工的大楼，内部空空如也。要真正发挥它的作用，还有必须装入各种实际的数据。

2. 数据库的试运行

当有部分数据装入数据库以后，就可以进入数据库的试运行阶段，数据库的试运行也称为联合调试。数据库的试运行对于系统设计的性能检测和评价是十分重要的，因为某些 DBMS 参数的最佳值只有在试运行中才能确定。

由于在数据库设计阶段，设计者对数据库的评价多是在简化了的环境条件下进行的，因此设计结果未必是最佳的。在试运行阶段，除了对应用程序做进一步的测试之外，重点执行对数据库的各种操作，实际测量系统的各种性能，检测是否达到设计要求。如果在数据库试运行时，所产生的实际结果不理想，则应回过头来修改物理结构，甚至修改逻辑结构。

3. 数据库的运行和维护

数据库系统投入正式运行，意味着数据库的设计与开发阶段的基本结束，运行与维护阶段的开始。数据库的运行和维护是个长期的工作，是数据库设计工作的延续和提高。

在数据库运行阶段，完成对数据库的日常维护，工作人员需要掌握 DBMS 的存储、控制和数据恢复等基本操作，而且要经常性地涉及物理数据库、甚至逻辑数据库的再设计，因此数据库的维护工作仍然需要具有丰富经验的专业技术人员(主要是数据库管理员)来完成。

数据库的运行和维护阶段的主要工作有：

- 对数据库性能的监测、分析和改善。
- 数据库的转储和恢复。
- 维持数据库的安全性和完整性。
- 数据库的重组和重构。

第 2 章 SQL Server 2000 应用基础

SQL Server 2000 由一组数量众多的数据库组件组成。这些组件在功能上互相补充，在使用方式上彼此协调，以满足用户在数据存储和管理、大型 Web 站点支持和企业数据分析处理上的需求。本章将介绍 SQL Server 2000 的安装过程、体系结构和常用管理工具。

2.1 SQL Server 2000 的安装

以 SQL Server 2000 标准版为例介绍安装 SQL Server 2000 的全过程。需要注意的是，SQL Server 2000 标准版和服务器版只能安装在服务器版的操作系统中，而对 Windows 2000 专业版和 Windows XP 专业版的操作系统，只能安装个人版和开发版的 SQL Server 2000。

(1)将 SQL Server 2000 光盘放进光驱，这时安装程序会自动运行，安装画面如图 2.1 所示。

图 2.1 SQL Server 2000 的自启动安装画面

(2)选择【安装 SQL Server 2000 组件】，进入下一个安装界面。可以看见 SQL Server 2000 标准版提供的各种安装组件：数据库服务器、分析服务(Analysis Service，过去版本的 OLAP 组件)和英语查询(English Query)，如图 2.2 所示。

(3)选择【安装数据库服务器】，系统进入正式安装的画面，如图 2.3 所示。

(4)单击【下一步】按钮，系统弹出如图 2.4 所示的对话框，让用户选择是进行远程安装还是本地安装，默认的选择是进行本地安装。

图 2.2　安装 SQL Server 2000 的组件

图 2.3　正式安装系统

图 2.4　选择安装方式

(5)选择进行本地安装，然后单击【下一步】按钮。接下来安装程序会搜索这台计算机上已经安装的 SQL Server 组件。

(6)安装程序搜索完已安装组件以后会弹出如图 2.5 所示的对话框。在该对话框可以选择是创建一个新的实例还是只是对现有的实例进行升级或删除。如果用户选择【高级选项】单选按钮，则可以进行注册表重建等操作。

图 2.5 关于实例的选择使用

(7)选择【创建新的 SQL Server 实例，或安装“客户端工具”】单选按钮，单击【下一步】按钮，弹出【用户信息】对话框，这里填入用户姓名和所在的单位名称，然后单击【下一步】按钮。

(8)安装程序弹出对话框，询问用户是否同意软件的使用协议。选择同意后系统弹出【安装定义】对话框，如图 2.6 所示。在这里可以根据安装的目的进行选择。如果要安装的是数据库服务器，则必须选择【服务器和客户端工具】单选按钮；如果只是为了实现客户端应用程序和服务器的连接畅通，则可以选择【仅客户端工具】单选按钮或【仅连接】单选按钮。

这里选择【服务器和客户端工具】单选按钮。

图 2.6 提示选择安装类型

(9)单击【下一步】按钮，系统弹出如图 2.7 所示的对话框。可以选择安装为默认实例，也可以选择安装为命名实例。如果选择安装为命名实例则必须为实例取一个名字。

图 2.7 填写实例名称

(10)单击【下一步】按钮，弹出如图 2.8 所示的对话框。选择【自定义】安装，然后单击【下一

步】按钮。

图 2.8　选择正确的安装类型

(11)安装程序弹出对话框让用户选择要安装的组件，如图 2.9 所示。

图 2.9　要安装的组件

(12) 选择好要安装的组件后，单击【下一步】按钮，安装程序弹出如图 2.10 所示对话框。

图 2.10　选择启动服务的帐户

为了启动运行 SQL Server 2000 的 MS SQL Server 和 MS SQL Server 代理服务，必须指定有相应权限的用户和口令，可以选择对每个服务使用同一个帐户，也可以为每个服务设置各不相同的用户。

(13)单击【下一步】按钮，系统弹出如图 2.11 所示的对话框。可以选择是使用 Windows 身份验证模式，还是使用混合模式。如果是在 Windows NT 的环境下安装，建议使用 Windows 身份验证模式。如果选择混合模式，必须输入 sa(超级管理员)的密码；也可以使用空密码，只需选中【空密码(不推荐)】复选框就可以了，不过考虑到数据库的安全性，不推荐使用空密码。

图 2.11　身份验证模式

(14)单击【下一步】按钮，系统弹出如图 2.12 所示的对话框。这里必须对使用什么样的排序规则进行设置。一般情况下，安装程序会根据操作系统的类型自动选择正确的选项，而不用用户过多参与。

图 2.12　排序规则设置

(15)单击【下一步】按钮，系统弹出如图 2.13 所示的对话框，要求用户配置网络库，采用默认设置，单击【下一步】按钮，安装程序会弹出一个对话框提示用户：安装程序已经获得了足够的安装信息，可以自动进行安装工作了。

图 2.13　配置网络库

(16)安装程序在接下来的步骤里会提示用户选择客户许可协议方式，如图 2.14 所示。SQL Server 2000 支持两种协议许可方式：处理器许可证和每客户。处理器许可证方式要求每一个针对本服务器的连接都拥有一个处理器访问许可证。每客户方式要求每一个访问 SQL Server 2000 的计算机都拥有一个客户端访问许可证。

图 2.14　选择许可协议方式

单击【继续】按钮，安装程序便开始自动进行安装过程。

2.2 SQL Server 2000 的体系结构

从不同的应用和功能角度出发，SQL Server 2000 具有不同的系统结构分类。具体可以划分为以下几种。

- 客户机/服务器体系结构：主要应用于客户端可视化操作、服务器端功能配置以及客户端和服务器端的通信。
- 数据库体系结构：又划分为数据库逻辑结构和数据库物理结构。数据库逻辑结构主要应用于面向用户的数据组织和管理，如数据库的表、视图、约束、用户权限等；数 据库物理结构主要应用于面向计算机的数据组织和管理，如数据文件、表和视图的数据组织方式、磁盘空间的利用和回收、文本和图形数据的有效存储等。
- 关系数据库引擎体系结构：主要应用于服务器端的高级优化，如查询服务器(Query Processor)的查询过程、线程和任务的处理、数据在内存的组织和管理等。

SQL Server 2000 对大多数用户而言，首先是一个功能强大的具有客户机/服务器体系结构的关系数据库管理系统，所以理解客户机/服务器体系结构是非常有必要的。

2.2.1 客户机/服务器或浏览器/服务器

20 世纪 80 年代末到 20 世纪 90 年代初，许多应用系统从主机终端方式、文件共享方式向客户机/服务器方式过渡。客户机/服务器系统比文件服务器系统能提供更高的性能，因为客户机和服务器将应用的处理要求分开，同时又共同实现其处理要求(即分布式应用处理)。服务器为多个客户机管理数据库，而客户机发送请求并分析从服务器接收的数据。在一个客户机/服务器应用中，数据库服务器是智能化的,它只封锁和返回一个客户机请求的那些行,保证了并发性,网络上的信息传输减到最少，因而可以改善系统的性能。

典型客户机/服务器计算的特点：

- 服务器负责数据管理及程序处理。
- 客户机负责界面描述和界面显示。
- 客户机向服务器提出处理要求。
- 服务器响应后将处理结果返回客户机。
- 网络数据传输量小。

总体来说，客户机/服务器计算方式是一种两层结构的体系。随着技术的进步以及需求的改变，更多的层次划分出来。目前，在 Internet 应用体系结构中，事务的处理被划分为三层，即浏览器——Internet 服务器——数据库服务器。在这种体系结构中，业务的表达通过简单的浏览器来实现，用户通过浏览器提交表单，把信息传递给 Internet 服务器，Internet 服务器根据用户的请求，分析出要求数据库服务器进行的查询，交给数据库服务器去执行，数据库服务器把查询的结果反馈给 Internet 服务器，再由 Internet 服务器用标准的 HTML 语言反馈给浏览器。

使用浏览器/服务器最大的好处是对客户端的要求降到了最低，减少了客户端的拥有和使用成本，具有更大的灵活性。但是它也增加了潜在的复杂性，对小型应用程序而言，开发速度可能比较慢。

2.2.2 SQL Server 2000 的服务器端组件

SQL Server 2000 服务器端组件主要包括：SQL Server、SQL Server Agent、MS DTC(Microsoft Distributed Transaction Coordinator Service)、Microsoft Search(Microsoft Search Service)。通过 SQL Server 服务管理器可以浏览和设置服务器组件的状态，如图 2.15 所示。

图 2.15 SQL Server 服务管理器

1. SQL Server

SQL Server 是 SQL Server 2000 数据库管理系统的核心数据库引擎，是最重要的组成部分。在 Windows NT 或者 Windows 2000 操作系统中,SQL Server 以服务的形式实现,具体表现为 MS SQL Server Service。需要注意的是这里所指的 SQL Server 与在前面提到的 SQL Server 2000 数据库管理系统有所区别。SQL Server 2000 是一个功能完善的数据库管理系统，它包括了从数据库文件到数据库管理软件等一系列相关组件，所以称为 SQL Server 2000 系统。而 SQL Server 是 SQL Server 2000 系统里服务器端最核心的组成部分，是系统的一个服务器端组件。本书为防止混淆，以后都采用 MS SQL Server Service 来代替 SQL Server 组件。

MS SQL Server Service 从服务一启动就运行在 Windows NT 或 Windows 2000 服务器上，直到服务停止为止。MS SQL Server Service 管理着由该 SQL Server 2000 系统拥有的所有文件,MS SQL Server Service 是 SQL Server 2000 系统中惟一可以直接读取和修改数据的组件。客户对数据库的所有服务

请求，最终都会体现为一组 Transact-SQL 命令。MS SQL Server Service 的功能是负责协调和安排这些服务请求的执行顺序，然后逐一解释和执行 SQL 命令，并向提交这些服务请求的客户返回执行的结果。MS SQL Server Service 同时也可以支持分布式的数据库查询，并不把范围局限在本 SQL Server 2000 系统中。

MS SQL Server Service 的功能还包括监督客户对数据库的操作，实施企业规则，维护数据一致性等，具体体现在：

- 负责存储过程和触发器的执行。
- 对数据加锁，实施并发性控制，以防止多个用户同时修改同一个数据。
- 管理分布式数据库，保证不同物理地址上存放的数据的一致性和完整性。
- 加强系统的安全性。

2. SQL Server Agent

SQL Server Agent(SQL 服务器代理)在 Windows NT 或 Windows 2000 系统里以服务的形式存在和运行，体现为 SQL Server Agent Service。SQL Server Agent 提供 SQL Server 的调度服务，能够自动执行数据库管理员预先安排好的作业(JOB)，监视 SQL Server 事件并根据事件触发警报(Alert)或运行事先安排好的程序。

通过配置和使用 SQL Server Agent，可以实现数据库系统的定时与自动管理。例如，当数据库出现故障或者某一特定的事件发生时，自动执行一系列操作来进行事件的处理。SQL Server Agent 必须和 SQL Server 一同使用，它实现自动化管理的组件，包括作业、警报和操作员(Operator)。

3. MS DTC (分布式事务协调器)

随着网络普及，分布式数据库的应用也越来越 普遍。在分布式数据库中逻辑上作为一个整体的数据被存储在了多个服务器上。例如，一家大的商业银行完全有可能将客户的信用卡消费信息和支票消费信息存储在 不同的服务器上，但是用户的存款帐户只有一个，当用户用任何一种形式进行了消费以后，计算机必须同时对存储在不同服务器上的信息进行更新。

为了更好地协调和处理这种分布式事务，SQL Server 2000 使用了 MS DTC。MS DTC 也以 Windows NT 服务的形式存在和运行。MS DTC 是一个事务管理器，它允许客户的应用程序在一个事务中对分布在多个服务器上的数据源进行操作。MS DTC 通过两段式提交的方法来实施分布式事务，针对多个服务器的更新要么全部成功执行，要么全部不执　行，从而有效保证数据的一致性和完整性。

4. Microsoft Search

Microsoft Search 是一个全文搜索和查询服务，它是一个可选的组件，可以在标准 SQL Server 安装过程中或者安装完成后补充安装。Microsoft Search 为 SQL Server 2000 提供了更为复杂而强大的查询能力。它的作用分为索引支持和查询支持两方面的功能。索引支持提供了 SQL Server 2000 建立全文目录(full-text catalog)的能力，而查询支持使 SQL Server 2000 可以有效地响应全文搜索查询。

全文目录和索引与标准的数据库索引不同，它不是存储在 SQL Server 数据库里，而存放在专门由 Microsoft Search 管理的文件里。Microsoft Search 不能安装在 Windows 95、Windows 98、Windows NT Workstation 及 Windows 2000 Professional 系统里，但是运行在这些系统上的客户端应用程序可以利用安装在 Windows NT Server 和 Windows 2000 Advanced Server 系统上的 Microsoft Search 服务。

2.2.3 SQL Server 2000 的客户端组件

SQL Server 2000 提供的客户端组件包括：企业管理器、查询分析器、SQL Server 管理工具和向导、SQL Server 命令提示管理工具等。

1. 企业管理器

企业管理器是图形化的集成管理工具，提供了调用其他管理工具的简单途径，利用企业管理器可以实现 SQL Server 2000 服务器的有效配置和管理。

企业管理器采用的界面是 Microsoft 公司为 Microsoft BackOffice 服务器族统一开发的管理控

制台(Microsoft Management Console，MMC)界面，与【Windows 资源管理器】极为相似。

企业管理器按照树型结构的要求来管理多个彼此通过网络互联的 SQL Server 2000 服务器。通过在企业管理器中对 SQL Server 2000 服务器进行注册，任何访问 SQL Server 2000 的用户都可以使用这些服务器。通过企业管理器集成的各种管理工具，数据库管理员可以方便地管理服务器、数据库、数据库对象、用户登录和许可、复制、安全性、调度任务、生成 Web 标签、生成 SQL 脚本及其他多种事务。

2. 查询分析器

SQL Server 2000 提供了查询分析器作为编写 Transact-SQL 脚本程序的开发工具。查询分析器提供的是一个图形化编写和调试 Transact-SQL 程序的工作环境。它通过彩色代码编辑器和上下文敏感帮助提高了应用程序的可用性。

除此以外，查询分析器还可以完成以下几方面的工作：对 Transact-SQL 语句的执行计划显示一种图形化的描述；通过索引调整表，明确对特定表格采用什么样的索引才能达到性能的优化；显示关于 SQL 语句工作性能的统计。

3. SQL Server 管理工具和向导

SQL Server 2000 提供了许多管理工具和向导，以实现 SQL Server 2000 在某一具体方面的功能。这些工具包括以下几种。

- SQL Server 事件探查器(SQL Server Profiler)

SQL Server 事件探查器，用于监视和分析运行 SQL Server 2000 的 服务器的活动情况。数据库管理员在服务器出现性能问题时，一般首先查阅事件探查器。事件探查器可以实时抓取服务器中运行的连续活动，如批处理、数据锁定、 安全事件和数据库错误等。事件探查器可以过滤这些活动，以使用户专注于关心的事件。事件探查器可以在分析时逐步重演这些活动，以检测故障所在。事件探查器 能够以连续、断点或单步的方式执行 Transact-SQL 编写的脚本程序和存储过程。因此开发应用程序时，也可以利用它进行程序的调试和检测。

- SQL Server 性能监视器(SQL Server Performance Monitor)

SQL Server 性能监视器通过设置计数器(性能监视器可以统计的特定数据指标)使 SQL Server 2000 与 Windows NT 或 Windows 2000 系统的性能监视器紧密配合，从而使管理员可以利用性能监视器以图形化的方式监视 SQL Server 的运行性能。

- SQL Server 客户端网络实用工具(SQL Server Client Network Utility)

SQL Server 客户端网络实用工具主要提供客户端的网络连接配置，也可以用于测定网络库的版本信息。

- SQL Server 服务器端网络实用工具(SQL Server Server Network Utility)

SQL Server 服务器端网络实用工具主要用于配置服务器端的网络连接参数，以确保服务器端能够正常地接受来自客户端的访问。

- 服务管理器(Service Manager)

服务管理器是一个图形化的用于启动、暂停、停止 SQL Server 服务的实用程序。

- SQL Server 管理向导

Microsoft 公司在 SQL Server 2000 中提供了大量的管理向导(Wizards)来帮助管理员完成导入数据、导出数据、创建数据库维护计划，或者配置复制等复杂的管理工作，使管理工作变得更为简洁。

4. SQL Server 命令提示管理工具

SQL Server 命令提示管理工具允许输入 Transact-SQL 语句并执行脚本文件。比较常用的 SQL Server 命令提示程序有：BCP、ISQL、OSQL、TEXTCOPY 和 ODBCPING 等。其中 BCP 是一个命令行工具，主要用于从 SQL Server 2000 中导入和导出数据，它可以把数据库中的数据存储到一个文本文件或者二进制文件中；TEXTCOPY 是一个用来从 SQL Server 2000 中导入或者导出图像文件的命令行工具；ISQL 是利用 DB-Library 与 SQL Server 2000 进行通信的数据查询工具，而 OSQL 是利用 ODBC 来与 SQL Server 2000 进行通信的数据查询工具。

2.2.4 客户端应用程序与数据库服务器的通信

SQL Server 2000 采用多种方式实现客户端应用程序与数据库服务器之间的通信。具体可以划分为以下两种情况。

- 客户端应用程序与数据库服务器位于同一台计算机

在这种情况下，SQL Server 2000 利用 Windows 进程间通信组件，如本地命名管道(local named pipes)或共享内存等。

- 客户端应用程序与数据库服务器位于不同计算机

在这种情况下，SQL Server 2000 将使用网络进程通信组件(Interprocess Communication Component，IPC)进行客户端和服务器端的连接。

一个 IPC 通常由两个部分组成：

- API(Application Programming Interface，应用编程接口)

API 主要是一组已经定义好的函数，应用软件通过调用这些函数来向 IPC 发送查询请求，取回查询的结果。

- 协议(Protocol)

协议定义了两个利用 IPC 进行通信组件之间传递数据所使用的格式。当使用网络 IPC 进行通信时，协议定义了传递的分组数据格式。

由于 SQL Server 2000 具有强大的网络应用功能，所以客户端的通信方式比以往任何一个版本都要复杂。但是 SQL Server 2000 做了大量的简化工作，把复杂的通信方式屏蔽在用户的使用范围之外。SQL Server 2000 的客户端应用程序可以动态确定服务器的网络地址，所需要的仅仅是服务器计算机的网络名字。

在 SQL Server 2000 所使用的通信组件中，网络库(Net-Library)是最主要的。网络库的功能是按照适当的网络协议将数据库请求及传输结果进行打包。网络库必须在客户和服务器上安装。客户和服务器可以同时使用多个网络库，但它们必须都使用通用的网络库以便成功地进行通信。

2.3 企业管理器

SQL Server 2000 提供的客户端组件中，企业管理器是最主要的管理工具，绝大部分的数据库管理工作都可以在企业管理器中完成。企业管理器与 Windows NT 在功能上紧密集成，它以树型结构的形式来管理 SQL Server 数据库服务器、数据库以及数据库中的对象，能够在单一的控制界面上实现对位于同一企业网络结构中多个 SQL Server 数据库服务器的有效管理。

选择 Microsoft SQL Server 2000 程序组里的【企业管理器】项目，运行后界面如图 2.16 所示。

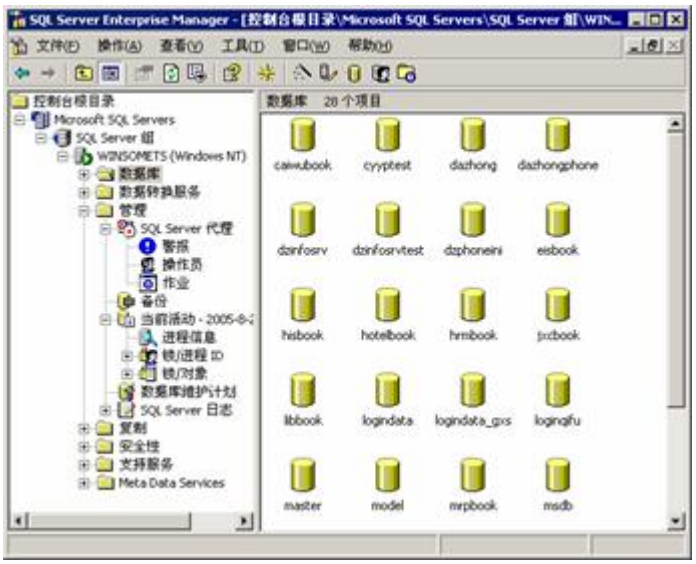


图 2.16 企业管理器

在窗口中可以看出，企业管理器使用了类似于【Windows 资源管理器】的树型结构。在左边的树型结构图上，根节点是【控制台根目录】，表示它是所有服务器控制台的根。在第一层节点上有一个默认的节点是 Microsoft SQL Servers，所有的 SQL Server 服务器组都是 Microsoft SQL Servers 节点的子节点。

用户可以在 Microsoft SQL Servers 节点下面自己定义新的服务器组。服务器组一般按照不同的用途和功能对服务器进行分类。例如，可以把所有执行 Web 数据库应用的服务器全都放在命名为 Web Servers 的服务器组下面。

按照模块化管理的原则，所有的服务器都必须按组进行分类。当安装完 SQL Server 2000 后，系统会默认提供一个【SQL Server 组】服务器组。刚安装成功的服务器就注册在该服务器组下面。当然可以将这条注册信息删除掉，对服务器进行重新注册。

在每个服务器下面是该服务器的所有管理对象和可以执行的管理任务，分为【数据库】、【数据转换服务】、【管理】、【复制】、【安全性】和【支持服务】六大类。在每一类下面又可以进一步细分。图 2.16 中的 Meta Data Services 不是标准类，因此这里不作介绍。

2.3.1 注册SQL Server服务器

在企业管理器中可以注册本地或者多个网络上的 SQL Server 服务器，从而通过企业管理器管理这些服务器。

(1)在【SQL Server 组】项目上右击，在弹出菜单中选择【新建 SQL Server 注册】菜单命令，出现【注册 SQL Server 向导】对话框，如图 2.17 所示。

图 2.17 注册 SQL Server 向导

(2)单击【下一步】按钮，弹出如图 2.18 所示的对话框。

在【可用的服务器】列表中列出了所有当前可以使用的服务器别名(图中 local 表示本机)， 选中打算使用的服务器别名,或者直接在【可用的服务器】文本框里输入需要注册的数据库服务器名称，并单击【添加】按钮，把该服务器别名转移到【添加的服务 器】列表框里。当【添加的服务器】列表框中存在可以使用的服务器别名时，对话框下方的【下一步】按钮将由灰白变为可用。

图 2.18　选择可以使用的服务器别名

(3)单击【下一步】按钮，弹出如图 2.19 所示的对话框，要求选择连接服务器时使用的身份验证模式。

SQL Server 提供了两种身份验证模式：一种模式集成使用 Windows NT 的安全认证机制，使用这种方式， 服务器所在的 Windows NT 系统会在连接过程中自动审查客户机登录时输入的帐号和密码，无须用户再次输入连接认证信息；另一种模式是使用 SQL Server 自身的安全认证机制，在这种情况下，用户必须是该 SQL Server 服务器的合法用户，拥有合法的帐号和密码，程序会在连接之前，弹出对话框，提示用户输入正确的 SQL Server 服务器上的帐号和密码。

图 2.19　选择连接时的身份验证模式

(4)选择好安全认证方式以后，单击【下一步】按钮，如果选择使用 SQL Server 自带的认证方式，必须事先在要连接的服务器上建立一个合法的帐户，并输入正确的认证信息。在输入帐号、密码的时候，可以选择“在连接时输入 SQL Server 帐户信息”单选按钮。这样，每次要使用连接访问服务器时都会要求用户输入认证信息。本例中选择使用 Windows NT 的安全认证机制。

(5)系统提示选择将容纳服务器的服务器组。可以根据将要连接的服务器的功能从本地服务器组中选择一个服务器组，连接成功后，连接上的服务器将会成为该服务器组下 面的一个节点；也可以选择重新建立一个新的服务器组来放置这个服务器，这种情况下，会弹出一个对话框让用户创建一个新的服务器组。

(6)单击【下一步】按钮，并确认输入的信息，单击【完成】按钮，进行连接操作。连接成功后，会出现服务器成功注册的说明。单击【关闭】按钮完成服务器的注册。

2.3.2　从企业管理器使用其他管理工具

在企业管理器提供的系统菜单中，【工具】菜单包含了 SQL Server 2000 提供的各种管理工具(客户端组件)，如图 2.20 所示。用户可以通过选择【工具】菜单中的命令，进入实现具体管理功能的管

理工具环境中。

图 2.20 【工具】菜单

SQL Server 2000 针对树型结构上不同的节点提供了不同的快捷菜单。使用快捷菜单是实现服务器管理功能的捷径。

2.4 查询分析器

SQL 查询分析器是一个图形化的数据库编程接口，是 SQL Server 2000 客户端应用程序的重要组成部分。SQL 查询分析器以自由的文本格式编辑 SQL 语句，对语法中的保留字提供彩色显示，并提供快速生成 SQL 代码的大量模板，同时使用图形化的方式显示执行 SQL 语句的逻辑步骤和 SQL 语句的执行效率评估。本节将简单介绍如何操作查询分析器，以便快速掌握 SQL 语言编写工具的使用方法。

用户启动查询分析器的方法有两种：一种是选择【开始】| Microsoft SQL Server |【查询分析器】命令；另一种方法是直接从企业管理器的【工具】菜单中选择【SQL 查询分析器】命令。

如果还没有连接上数据库服务器，那么系统执行命令时会弹出如图 2.21 所示的连接登录对话框。

在 SQL Server 下拉列表框中输入需要连接的数据库名称，选择身份认证的方法后单击【确定】按钮，进入查询分析器，如图 2.22 所示。

左边是查询分析器的对象浏览器和模板，右边是一个查询脚本编辑器，在这个窗口中，可以编写 Transact-SQL 语句，调用存储过程，进行查询优化，分析查询过程等。

为了方便输入，查询编辑器可以用不同的颜色显示特殊的关键字，例如，用蓝色显示标准的 SQL 命令字，如 SELECT、INSERT；用紫色显示全局变量名，如@@version 等，以确保语句输入的正确。

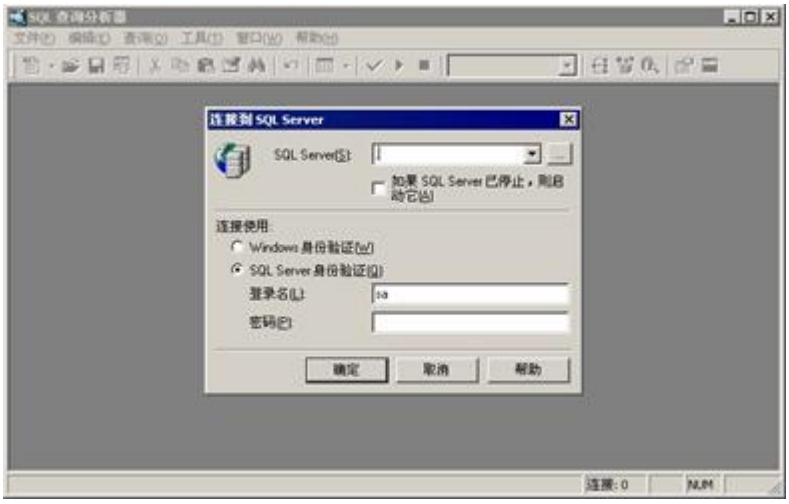


图 2.21 查询分析器登录界面

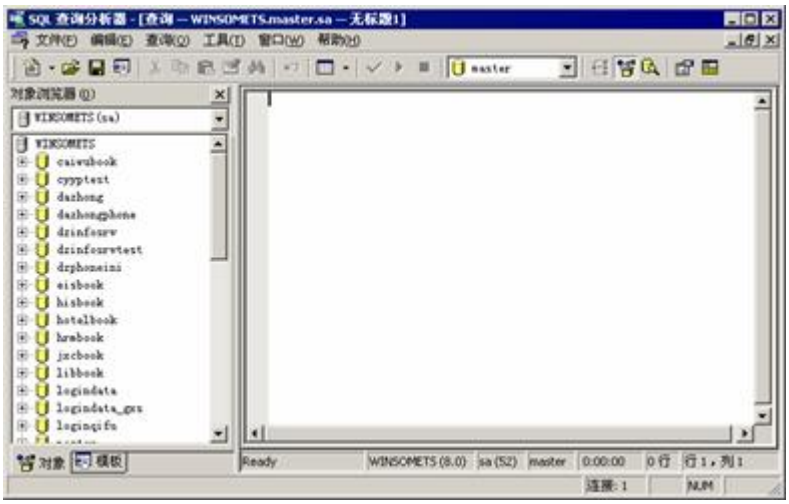


图 2.22 查询分析器的编辑界面

第 3 章 数据库创建及维护

客户端/服务器型数据库系统由程序和数据结构两部分组成，程序为基于客户端的用户数据访问提供界面；数据库结构管理和存储服务器上的数据。 例如，如果使用 SQL Server 2000 创建支票帐户应用程序，则必须建立一个数据库结构来管理帐户交易数据，还要建立一个应用程序充当数据库的用户界面，从而使用户得以访问支票帐户的信息。要创建能够满足业务需要的数据库，要求对如何设计、创建和维护各个组件有深刻的理解，这样才能确保数据库最佳地运行。

3.1 SQL Server 2000 的数据库组成部分

在 SQL Server 2000 中，数据库由存储特定结构化数据集的表集合组成。表中包含行(有时称作记录或元组)和列(有时称作特性)的集合。表中的每一列都设计为存储某种类型的信息(例如，日期、名称、美元金额或数字)。表上有几种控制(约束、规则、触发器、默认值和自定义用户数据类型)用于确保数据的有效性。表上可以有索引，利用索引可以快速地找到行。可将声明引用完整性 (DRI) 约束添加到表上，以确保不同表中相互关联的数据保持一致。数据库还可以存储过程，这些过程使用 Transact-SQL 编程代码对数据库中的数据进行操作，如存储对表数据提供自定义访问的视图。

3.1.1 文件和文件组

SQL Server 2000 使用一组文件映射数据库。数据库中的所有数据和对象(如表、存储过程、触

发器和视图)都存储在下列文件中。

- 主要数据文件

该文件包含数据库的启动信息，并用于存储数据。每个数据库都有一个主要数据文件。

- 次要数据文件

这些文件含有不能置于主要数据文件中的所有数据。如果主要数据文件可以包含数据库中的所有数据，那么数据库就不需要次要数据文件。有些数据库可能足够大，故需要多个次要数据文件，或使用位于不同磁盘驱动器上的辅助文件将数据扩展到多个磁盘。

- 事务日志文件

这些文件包含用于恢复数据库的日志信息。每个数据库都必须至少有一个日志文件。

文件组允许对文件进行分组，以便于管理数据的分配和放置。例如，可以分别在三个硬盘驱动器上创建三个文件(Data1.mdf、Data2.mdf 和 Data3.mdf)，并将这三个文件指派到文件组 Fgroup1 中。然后，可以明确地在文件组 Fgroup1 上创建一个表。对表中数据的查询将分散到三个磁盘上，因而性能得以提高。在 RAID(独立磁盘冗余阵列)条带集上创建单个文件也可以获得相同的性能改善。然而，文件和文件组使用户得以在新磁盘上轻易地添加新文件。

文件或文件组不能由一个以上的数据库使用。例如，文件 hospital.mdf 包含 Hospital 数据库中的数据和对象，任何其他数据库都不能使用这个文件。

3.1.2 事务日志

在 SQL Server 2000 中，数据库必须至少包含一个数据文件和一个事务日志文件。数据和事务日志信息从不混合在同一文件中，并且每个文件只能由一个数据库使用。

SQL Server 2000 使用各数据库的事务日志来恢复事务。事务日志是数据库中已发生的所有修改和执行每次修改的事务的一连串记录。事务日志记录每个事务的开始，它记录了在每个事务期间，对数据的更改及撤消所做更改(以后如有必要)所需的足够信息。对于一些大的操作(如 CREATE INDEX)，事务日志则记录该操作发生的事实。随着数据库中发生被记录的操作，日志会不断地增长。

在应用事务日志时，事务将前滚。SQL Server 2000 将每次修改后的映象复制到数据库中，或者重新运行语句(如 CREATE INDEX)。这些操作将按照其原始发生顺序应用。此过程结束后，数据库将处于与事务日志备份时相同的状态。

当收回未完成的事务时，事务将回滚。SQL Server 2000 将所有修改前的映象复制到 BEGIN TRANSACTION 后的数据库。如果遇到表示执行了 CREATE INDEX 操作的事务日志记录，则会执行与该语句逻辑相反的操作。这些前映象和 CREATE INDEX 操作的逆转将按照与原始顺序相反的顺序进行应用。

在检查点处，SQL Server 2000 确保所有已修改的事务日志记录和数据库页都写入磁盘。在重新启动 SQL Server 2000 时 所发生的各数据库的恢复过程中，仅在不知道事务中所有的数据修改是否已经从高速缓存中实际写入磁盘时才必须前滚事务。因为检查点强迫所有修改的页写入磁 盘，所以检查点表示启动恢复必须开始前滚事务的位置。因为检查点之前的所有修改页都保证在磁盘上，所以没有必要前滚检查点之前已完成的任何事务。

利用事务日志备份可以将数据库恢复到特定的即时点(如输入不想要的数据之前的那一点)或故障发生点。在媒体恢复策略中应考虑利用事务日志备份。

3.1.3 表

表是包含数据库中所有数据的数据库对象。表 定义为列的集合。与电子表格相似，数据在表中是按行和列的格式组织排列的。每行代表惟一的一条记录，而每列代表记录中的一个域。例如，在包含公司雇员数据 的表中每一行代表一名雇员，各列分别表示雇员的详细资料，如雇员编号、姓名、地址、职位及家庭电话号码等。

设计数据库时，应先确定需要什么样的表，各表中都有哪些数据，以及各个表的存取权限等。在创建和操作表的过程中，将对表进行更为细致的设计。

创建一个表最有效的方法是将表中所需的信息一次定义完成，包括数据约束和附加成分。也可以

先创建一个基础表，向其中添加一些数据并使用一段时间。这种方法使用户可以在添加各种约束、索引、默认设置、规则和其他对象形成最终设计之前，发现哪些事务最常用，哪些数据经常输入。

设计表时应注意以下的问题：

- 表所包含的数据的类型。
- 表的各列及每一列的数据类型(如果必要，还应注意列宽)。
- 哪些列允许空值。
- 是否要使用以及何时使用约束、默认设置或规则。
- 所需索引的类型，哪里需要索引，哪些列是主键，哪些是外键。

3.1.4 索引

数据库中的索引与书籍中的目录类似。在一本 书中，利用目录可以快速查找所需信息，无须阅读整本书。在数据库中，索引使数据库程序无须对整个表进行扫描，就可以在其中找到所需数据。数据库中的索引是 一个表中所包含的值的列表，其中注明了表中包含各个值的行所在的存储位置。可以为表中的单个列建立索引，也可以为一组列建立索引；索引采用 B 树结构。索引包含一个条目，该条目有来自表中每一行的一个或多个列(搜索关键字)。B 树按搜索关键字排序，可以在搜索关键字的任何子词条集合上进行高效搜索。例如，对于一个 A、B、C 列上的索引，可以在 A，A、B，A、B、C 上对其进行高效搜索。

在随 SQL Server 2000 提供的 pubs 示例数据库中，employee 表在 emp_id 列上有一个索引。当 SQL Server 执行一个语句，在 employee 中根据指定的 emp_id 值查找数据时，它能够识别 emp_id 列的索引，并使用该索引查找所需数据。如果该索引不存在，它会从表的第一行开始，逐行搜索指定的 emp_id 值。

SQL Server 2000 为某些类型的约束(如 PRIMARY KEY 和 UNIQUE 约束)自动创建索引。可以通过创建不依赖于约束的索引，进一步对表定义进行自定义。

不过，索引为性能所带来的好处却是有代价的。带索引的表在数据库中会占据更多的空间。另外，为了维护索引，对数据进行插入、更新、删除操作所花费的时间会更长。在设计和创建索引时，应确保对性能的提高程度大于在存储空间和处理资源方面的代价。

在考虑是否为一个列创建索引时，应考虑被索引的列是否以及如何用于查询中。索引对下列查询很有帮助：

- 搜索符合特定搜索关键字值的行(精确匹配查询)。精确匹配比较是指查询使用 WHERE 语句指定具有给定值的列条目。例如 WHERE emp_id = 'VPA30890F'。
- 搜索其搜索关键字值为范围值的行(范围查询)。范围查询是指查询指定其值介于两个值之间的任何条目。例如 WHERE job_lvl BETWEEN 9 and 12。
- 在表 T1 中搜索根据联接谓词与表 T2 中的某个行匹配的行(索引嵌套循环联接)。
- 在不进行显式排序操作的情况下产生经排序的查询输出，尤其是经过排序的动态游标。
- 在不进行显式排序操作的情况下，按一种有序的顺序对行进行扫描，以允许基于顺序的操作，如合并联接和流聚合。
- 以优于表扫描的性能对表中所有的行进行扫描，性能提高是由于减少了要扫描的列集和数据总量(该查询有覆盖索引可供使用)。
- 搜索插入和更新操作中重复的新搜索关键字值，以实施 PRIMARY KEY 和 UNIQUE 约束。
- 搜索已定义了 FOREIGN KEY 约束的两个表之间匹配的行。

在很多查询中，索引可以带来多方面的好处。例如，索引除了可以覆盖查询外，还使得可以进行范围查询。SQL Server 2000 可以在同一个查询中为一个表使用多个索引，并可以合并多个索引，以便搜索关键字共同覆盖一个查询。另外，SQL Server 会自动确定利用哪些索引进行查询，并且能够在表被改动时确保该表的所有索引都得到维护。

一个表如果建有大量索引会影响 INSERT、UPDATE 和 DELETE 语句的性能，因为在表中的数据更改时，所有索引都须进行适当的调整。另一方面，对于不需要修改数据的查询(SELECT 语句)，大量

索引有助于提高性能，因为 SQL Server 2000 有更多的索引可供选择，以便确定以最快速度访问数据的最佳方法。对小型表进行索引可能不会产生优化效果，因为 SQL Server 2000 在遍历索引以搜索数据时，花费的时间可能会比简单的表扫描还长。

3.1.5 视图

视图是一个虚拟表，其内容由查询定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。

对其中所引用的基础表来说，视图的作用类似 于筛选。定义视图的筛选可以来自当前或其他数据库的一个或多个表，或者其他视图。分布式查询也可用于定义使用多个异类源数据的视图。如果有几台不同的服务 器分别存储组织中不同地区的数据，而用户需要将这些服务器上相似结构的数据组合起来，这种方式就很有用。通过视图进行查询没有任何限制，通过它们进行数据 修改时的限制也很少。

视图通常用来集中、简化和自定义每个用户对数据库的不同认识。视图可用作安全机制，方法是允许用户通过视图访问数据，而不授予用户直接访问视图基础表的权限。从 SQL Server 2000 复制数据时也可使用视图来提高性能并分区数据。

视图可以简化用户操作数据的方式。可将经常 使用的联接、投影、联合查询和选择查询定义为视图，这样，用户每次对特定的数据执行进一步操作时，不必指定所有条件和限定。例如，一个用于报表目的，并执 行子查询、外联接及聚合以从一组表中检索数据的复合查询，就可以创建为一个视图。视图简化了对数据的访问，因为每次生成报表时无需写或提交基础查询，而是 查询视图。

视图允许用户以不同的方式查看数据，即使他们同时使用相同的数据时也如此。这在具有不同目的和技术水平的用户共享同一个数据库时尤为有利。例如，可定义一个视图以仅检索由客户经理处理的客户数据。视图可以根据使用该视图的客户经理的登录 ID 决定检索哪些数据。

可使用视图将数据导出至其他应用程序。例如，可能希望使用 pubs 数据库中的 stores 和 sales 表在 Excel 中分析销售数据。为此，可创建一个基于 stores 和 sales 表的视图。然后使用数据导入导出工具导出由视图定义的数据。

Transact-SQL UNION 集合运算符可在视图内使用，以将来自不同表的两个或多个查询结果组合成单一的结果集。这在用户看来是一个单独的表，称为分区视图。例如，如果一个表含有华盛顿的销售数据，另一个表含有加利福尼亚的销售数据，即可从 UNION 创建这两个表的视图。该视图代表了这两个区域的销售数据。使用分区视图时，首先创建几个相同的表，指定一个约束以决定可在各个表中添加的数据范围。视图即使用这些基表创建。当查询该视图时，SQL Server 自动决定查询所影响的表，并仅引用这些表。例如，如果一个查询指定只需要华盛顿特区的销售数据，则 SQL Server 只读取含有华盛顿特区销售数据的表，而并不访问其余的表。分区试图可基于来自多个异类源(如远程服务器)的 数据，而不仅仅局限于同一数据库中的表。例如，要将分别存储组织中不同区域数据的几台远程服务器上的数据组合起来，可以创建分布式查询，从每个数据源中检 索数据，然后基于这些分布式查询创建视图。所有查询都只从包含查询所请求数据的远程服务器上读取表中的数据，其他在视图中由分布式查询引用的服务器均不被 访问。

3.1.6 存储过程

在使用 SQL Server 2000 创建应用程序时，Transact-SQL 编程语言是应用程序和 SQL Server 数据库之间的主要编程接口。使用 Transact-SQL 程序时，可用两种方法存储和执行程序。可以在本地存储程序，并创建向 SQL Server 发送命令并处理结果的应用程序；也可以将程序在 SQL Server 中存储为存储过程，同时创建执行存储过程并处理结果的应用程序。

SQL Server 2000 中的存储过程与其他编程语言中的过程类似,利用存储过程可以完成以下任务。

- 接受输入参数并以输出参数的形式将多个值返回至调用过程或批处理。
- 包含执行数据库操作(包括调用其他过程)的编程语句。
- 向调用过程或批处理返回状态值，以表明成功或失败(以及失败原因)。

- 可使用 Transact-SQL EXECUTE 语句运行存储过程。存储过程与函数不同，因为存储过程不返回取代其名称的值，也不能直接用在表达式中。

使用 SQL Server 2000 中的存储过程代替存储在客户计算机本地的 Transact-SQL 程序有很多的好处。

- 允许模块化程序设计。只需创建过程一次并将其存储在数据库中，以后即可在程序中调用该过程任意次。存储过程可由在数据库编程方面有专长的人员创建，并可独立于程序源代码而单独修改。
- 允许更快执行。如果某操作需要大量 Transact-SQL 代码或需重复执行，存储过程将比 Transact-SQL 批代码的执行要快。将在创建存储过程时对其进行分析和优化，并可在首次执行该过程后使用该过程的内存中版本。每次运行 Transact-SQL 语句时，都要从客户端重复发送，并且在 SQL Server 2000 每次执行这些语句时，都要对其进行编译和优化。
- 减少网络流量。一个需要数百行 Transact-SQL 代码的操作由一条执行过程代码的单独语句就可实现，而不需要在网络中发送数百行代码。
- 可作为安全机制使用。即使对于没有直接执行存储过程中语句的权限的用户，也可授予他们执行该存储过程的权限。

3.1.7 触发器

SQL Server 2000 提供了两种主要机制来强制业务规则和数据完整性：约束和触发器。触发器是一种特殊类型的存储过程，它在指定的表中的数据发生变化时自动生效。唤醒调用触发器以响应 INSERT、UPDATE 或 DELETE 语句。触发器可以查询其他表，并可以包含复杂的 Transact-SQL 语句。触发器和触发它的语句要作为可在触发器内回滚的单个事务对待。如果检测到严重错误(例如磁盘空间不足)，则整个事务即自动回滚。

触发器可通过数据库中的相关表实现级联更改。触发器可以强制 CHECK 约束定义的约束更为复杂的约束。与 CHECK 约束不同，触发器可以引用其他表中的列。例如，触发器可以使用另一个表中的 SELECT 比较插入或更新的数据，以及执行其他操作，如修改数据或显示用户定义错误信息。触发器也可以评估数据修改前后的表状态，并根据其差异采取对策。一个表中的多个同类触发器(INSERT、UPDATE 或 DELETE)允许采取多个不同的对策以响应同一个修改语句。

约束和触发器在特殊情况下各有优势。触发器的主要好处在于它们可以包含使用 Transact-SQL 代码的复杂处理逻辑。因此，触发器可以支持约束的所有功能；但它在所给出的功能上并不总是最好的方法。

实体完整性总应在最低级别上通过索引进行强制，这些索引或是 PRIMARY KEY 和 UNIQUE 约束的一部分，或是在约束之外独立创建的。假设功能可以满足应用程序的功能需求，域完整性应通过 CHECK 约束进行强制，而引用完整性则应通过 FOREIGN KEY 约束进行强制。

3.1.8 用户定义函数

函数是由一个或多个 Transact-SQL 语句组成的子程序，可用于封装代码以便重新使用。SQL Server 2000 并不将用户限制在定义为 Transact-SQL 语言一部分的内置函数上，而是允许用户创建自己的用户定义函数。

可使用 CREATE FUNCTION 语句创建，使用 ALTER FUNCTION 语句修改，以及使用 DROP FUNCTION 语句除去用户定义函数。每个完全合法的用户定义函数名必须惟一。必须被授予 CREATE FUNCTION 权限才能创建、修改或除去用户定义函数。不是所有者的用户在 Transact-SQL 语句中使用某个函数之前，必须先被授予对该函数的适当权限。若要创建或更改在 CHECK 约束、DEFAULT 子句或计算列定义中引用用户定义函数的表，还必须具有函数的 REFERENCES 权限。

SQL Server 2000 支持三种用户定义函数：

- 标量函数
- 内嵌表值函数
- 多语句表值函数

用户定义函数采用零个或更多的输入参数并返回标量值或表。函数最多可以有 1 024 个输入参数。当函数的参数有默认值时，调用该函数时必须指定默认 DEFAULT 关键字才能获取默认值。该行为不同于在存储过程中含有默认值的参数，而在这些存储过程中省略该函数也意味着省略默认值。用户定义函数不支持输出参数。

标量函数返回在 RETURNS 子句中定义的类型的单个数据值。可以使用所有标量数据类型，包括 bigint 和 sql_variant。不支持 timestamp 数据类型、用户定义数据类型和非标量类型(如 table 或 cursor)。

表值函数返回 table。对于内嵌表值函数，没有函数主体；表是单个 SELECT 语句的结果集。对于多语句表值函数，在 BEGIN…END 块中定义的函数主体包含 Transact-SQL 语句，这些语句可生成行并将行插入将返回的表中。

3.2 数据库的创建

创建数据库有两种途径：通过 Transact-SQL 语句完成和直接在企业管理器中进行。

3.2.1 使用企业管理器创建数据库

利用企业管理器创建数据库的步骤如下。

(1)选中要建立数据库的服务器节点，双击展开该节点。

(2)选中【数据库】节点。在【操作】菜单或快捷菜单中选择【新建数据库】命令，弹出如图 3.1 所示的对话框。

图 3.1 数据库常规属性

(3)在【名称】文本框中输入正确的数据库名字，打开【数据文件】选项卡，如图 3.2 所示。

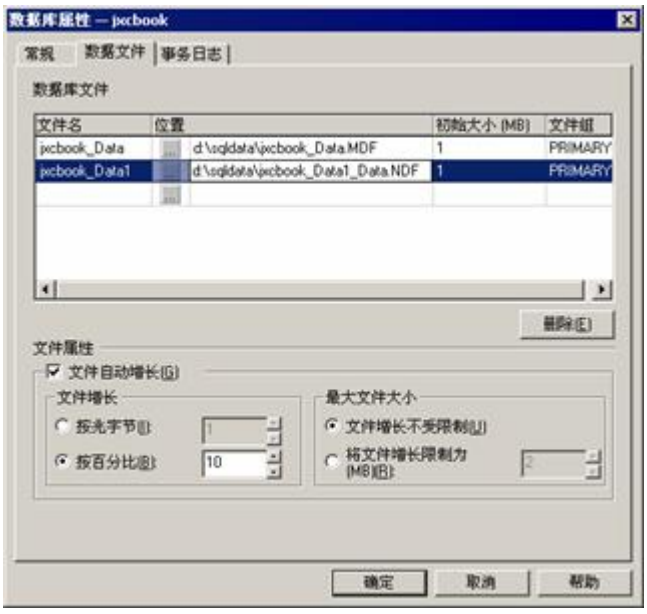


图 3.2 创建数据库

(4)在【文件名】栏中输入正确的数据文件名字。SQL Server 2000 在默认情况下自动为用户输入的文件名后面增加下划线和 Data 字样，并设置相应的文件扩展名。单击【位置】栏中的按钮，可以选择数据文件存放的位置。在【文件组】栏中可以输入文件所属的文件组名称，也可以使用默认值。

(5)打开【事务日志】选项卡，如图 3.3 所示。

图 3.3 创建日志文件

设置日志文件的方法与设置数据文件的方法类似。完成所有设置后单击【确定】按钮，完成数据库的创建。

3.2.2 使用Transact-SQL创建数据库

使用 Transact-SQL 创建数据库的语法如下。

```
CREATE DATABASE database_name
[ ON
  [ < filespec > [1,...n ] ]
  [ , < filegroup > [1,...n ] ]
]
```

```
[ LOG ON { < filespec > [1,...n ] } ]
[ COLLATE collation_name ]
[ FOR LOAD | FOR ATTACH ]
< filespec > ::=
[ PRIMARY ]
( [ NAME = logical_file_name , ]
  FILENAME = 'os_file_name'
  [ , SIZE = size ]
  [ , MAXSIZE = { max_size | UNLIMITED } ]
  [ , FILEGROWTH = growth_increment ] ) [1,...n ]
< filegroup > ::=
FILEGROUP filegroup_name < filespec > [1,...n ]
```

所有用[]括起来表示是可以省略的选项，[1,...n]表示同样的选项可以重复 1～n 遍；<>括起来表示是对一组若干选项的代替，实际编写语句时，应该用相应的选项来代替。类似 A|B 的语句，表示可以选择 A 也可以选择 B，但不能同时选择 A 和 B。

- database_name：表示为数据库取的名字，在同一个服务器内数据库的名字必须惟一。数据库的名字必须符合 SQL Server 2000 的标识符命名标准，即最大不得超过 128 个字符。
- ON：表示存放数据库的数据文件将在后面分别给出定义。
- PRIMARY：定义数据库的主数据文件。在 PRIMARY filegroup 中，第一个数据文件是主数据文件，如果没有给出 PRIMARY 关键字则默认文件序列中的第一个文件为主数据文件。
- LOG ON：定义数据库的日志文件。
- FOR LOAD：为了和过去的 SQL Server 版本兼容，FOR LOAD 表示计划将备份直接装入新建的数据库。
- NAME：定义操作系统文件的逻辑文件名。逻辑文件名只在 Transact-SQL 语句中使用，是实际磁盘文件名的代号。
- FILENAME：定义操作系统文件的实际名字，包括文件所在的路径。
- SIZE：定义文件的初始长度。
- MAXSIZE：定义文件能够增长到的最大长度，可以设置 UNLIMITED 关键字，使文件可以无限制增长，直到驱动器被填满。
- FILEGROWTH：定义操作系统文件长度不够时每次增长的速度。可以用 MB，KB，或使用%来设置增长的百分比。默认的情况下，SQL Server 2000 使用 MB 作为增长速度的单位，最少增长 1MB。

下面的代码是一个创建数据库的简单例子。

```
CREATE DATABASE Business
ON
( NAME = Business_dat,
FILENAME= 'e:\program files\Microsoft sql server\mssql\data\businessdat. mdf',
  SIZE = 10,
  MAXSIZE = 50,
  FILEGROWTH = 5 )
LOG ON
( NAME = 'Business_log',
FILENAME= 'e:\program files\Microsoft sql server\mssql\data\businessdat. ldf',
  SIZE = 5MB,
```

```
  MAXSIZE = 25MB,
  FILEGROWTH = 5MB )
GO
```

在查询分析器中运行上面的 Transact-SQL 语句后系统返回结果如下：

```
CREATE DATABASE 进程正在磁盘 'Business_dat' 上分配 10.00 MB 的空间。
CREATE DATABASE 进程正在磁盘 'Business_log' 上分配 5.00 MB 的空间。
```

这个例子创建了一个名为 Business 的数据库。由于在指定数据文件时，没有使用 PRIMARY 关键字，所以第一个文件 e:\programfiles\Microsoft sql server\mssql\data\ businessdat.mdf 被默认为主数据文件，本例没有指定二级数据文件。由于在指定数据文件的增长时没有指定单位，所以按默认值取为 MB，即文件增长为 5MB。

数据库建立后需要设计数据库对应的表、视图、存储过程等元素。这些元素可以通过企业管理器创建，也可以通过 Transact-SQL 语句创建，下面一章将介绍相关的 Transact-SQL 语句。

3.3 查看数据库的信息

可以使用 USE 子句选择要执行操作的数据库，例如要使用 pubs 数据库，可以在查询分析器中运行如下语句。

```
USE pubs
```

在没有再次使用 USE 语句来改变当前数据库以前，后面的每一行操作语句都将针对 pubs 数据库执行。使用系统存储过程 sp_helpdb 可以查看当前服务器上所有数据库的信息，如果指定了数据库的名字则返回指定数据库的信息。

使用系统存储过程 sp_databases 可以查看当前服务器上所有可以使用的数据库。

使用系统存储过程 sp_helpfile 可以查看当前数据库上所有文件(包括数据文件和日志文件)的信息。如果指定了文件的名字，则返回该文件的信息。

使用系统存储过程 sp_helpfilegroup 可以查看当前数据库上所有文件组，包括 Primary 文件组和 User_defined 文件组的信息。如果指定了文件组的名字，则返回该文件组的信息。

可以使用企业管理器来查看更详细的有关数据库的信息，方法是：

(1)选中要查看的数据库。

(2)从【操作】菜单或快捷菜单中选择【属性】命令，弹出如图 3.4 所示的对话框。从中可以查看或者修改数据文件、日志文件、文件组及使用权限等属性。

图 3.4 数据库的属性查看

3.4 管理数据库

创建好数据库，也许使用一段时间之后，发现此数据库的文件容量不敷使用、此数据库已经有一段时间不用了等相关事情发生时，就必须要针对数据库来进行管理。

3.4.1 修改数据库大小

SQL Server 2000 的数据文件可以自动扩充长度，所以数据库的大小也会自动增加。但是如果设置了最大文件长度，则数据库的扩充依然是有必要进行的操作。修改数据库的大小实质上是修改数据文件和日志文件的长度，或者增加/删除操作系统文件。这种操作可以通过下面的语法来实现：

```
ALTER DATABASE database
{   ADD FILE <filespec> [,...n] [TO FILEGROUP filegroup_name]
   | ADD LOG FILE <filespec> [,...n]
   | REMOVE FILE logical_file_name
   | ADD FILEGROUP filegroup_name
   | REMOVE FILEGROUP filegroup_name
   | MODIFY FILE <filespec>
   | MODIFY FILEGROUP filegroup_name filegroup_property
}
<filespec> ::=
(NAME = logical_file_name
  [, FILENAME = 'os_file_name' ]
  [, SIZE = size]
  [, MAXSIZE = { max_size | UNLIMITED } ]
  [, FILEGROWTH = growth_increment] )
```

下面的例子在 Company 数据库的默认文件组 Primary 文件组里，增加了一个数据文件。

```
ALTER DATABASE Company
ADD FILE
(
 NAME = Test1dat2,
 FILENAME = 'e:\program files\Microsoft sql server\mssql\data\t1dat2.ndf',
 SIZE = 5MB,
 MAXSIZE = 100MB,
 FILEGROWTH = 5MB
)
GO
```

下面的例子在 Company 数据库的 ComGroup1 文件组里增加了两个数据文件，并将该文件组设置为默认文件组。

```
ALTER DATABASE Company
ADD FILE
( NAME = test1dat3,
  FILENAME ='e:\program files\Microsoft sql server\mssql\data\t1dat3.ndf',
  SIZE = 5MB,
  MAXSIZE = 100MB,
  FILEGROWTH = 5MB),
( NAME = test1dat4,
  FILENAME = 'e:\program files\Microsoft sql server\mssql\data\t1dat4.ndf',
  SIZE = 5MB,
  MAXSIZE = 100MB,
  FILEGROWTH = 5MB)
TO FILEGROUP ComGroup1

ALTER DATABASE Company
MODIFY FILEGROUP ComGroup1 DEFAULT
GO
```

下面的例子将 Company 数据库中 test1dat3 数据文件的长度改变为 20MB。

```
ALTER DATABASE Company
MODIFY FILE
   (NAME = test1dat3,
   SIZE = 20MB)
GO
```

同样的操作可以在企业管理器中进行，方法如下。

(1)选中要查看的数据库。

(2)从主菜单或快捷菜单中选择【属性】命令，弹出如图 3.4 所示对话框。

(3)在【数据文件】选项卡中增删数据文件或对其属性进行修改。

(4)在【事务日志】选项卡上增删日志文件或对其属性进行修改。

3.4.2 收缩数据库

SQL Server 2000 数据库在长时间使用后数据文件和日志文件会非常庞大，同时删除了大量的数据后，数据文件的大小并没有自动变小。SQL Server 2000 提供了手段来缩小过于庞大的数据库，以回收没有使用的数据页。可以用手动的方法单独缩小某一个数据文件，也可以缩小整个文件组的长度。还可以设置数据库在达到一定大小之前自动执行缩小操作。

使用企业管理器完成缩小数据库的方法如下。

(1) 从指定服务器上展开数据库节点，选中要执行缩小操作的数据库。

(2) 从快捷菜单中选择【所有任务】|【收缩数据库】命令，弹出如图 3.5 所示的对话框。

图 3.5 缩小数据库

选择执行数据库缩小操作的方式：

- 【收缩后文件中的最大可用空间】微调框 在进行过数据库收缩后，文件中还可以利用的空间。
- 【在收缩前将页移到文件起始位置】复选框 把含有数据的数据页都移动到文件的开头。
- 【收缩文件】选项组 为了进行更精确的文件大小控制，针对每个单独的数据库文件进行收缩，具体方法是：单击【文件】按钮，弹出如图 3.6 所示的对话框。

图 3.6 针对文件进行收缩

可以在这个对话框中选择针对文件执行收缩操作的不同方式，也可以设置执行收缩操作的时间，把执行数据库收缩的时间安排在数据库存取不那么频繁的时段。

3.4.3 备份数据库

数据对于用户来说是非常宝贵的资产。数据是存放在计算机上的，但是即使是最可靠的硬件和软件，也会出现系统故障或产品故障。所以，应该在意外发生之前做好充分的准备工作，以便在意外发生之后有相应的措施能快速地恢复数据库的运行，并使丢失的数据量减少到最小。

通过企业管理器来备份数据库的操作如下：

(1)选中指定的数据库，选择【工具】|【备份数据库】命令，弹出如图 3.7 所示的对话框。

图 3.7 进行数据库备份

(2)在该对话框可以更换需要备份的数据库，输入备份的名字和对备份的描述，选择备份的类型。

(3)单击【添加】按钮选择要备份的设备，如图 3.8 所示。

图 3.8 选择备份设备

在这个对话框中可以创建新的备份设备。用户可以一次选择多个设备，将数据库备份到多个设备上。也可以将数据库备份到指定的文件。

(4)在图 3.7 所示的对话框的【重写】选项组里，选择在设备上进行操作的方式，可以选择【追加到媒体】单选按钮，将新的备份添加到备份设备中以前备份的后面(不影响原来的备份)；也可以选择【重写现有媒体】单选按钮，用新的备份覆盖原来的备份。

(5)选中【调度】复选框，用户可以拟订自动进行备份操作的时间，单击右边的按钮用户可以自主设计自动进行备份操作的时间，如图 3.9 所示。完成设置后单击【确定】按钮，回到数据库备份界面。

图 3.9 设置自动备份

(6)单击【确定】按钮，完成备份操作。

3.4.4 恢复数据库

使用 SQL Server 2000 可以很方便地实现数据库的恢复，步骤如下。

(1)选中指定的数据库，选择【工具】|【还原数据库】命令，弹出对话框如图 3.10 所示的对话框。

图 3.10 还原数据库

(2)可以选择要进行还原的数据库。在【参数】选项组中，用户可以选择利用哪个数据库备份来执行数据库的恢复操作。如果是利用事务日志来进行恢复，还可以选择恢复数据库到某一指定时刻的状态。如果是从设备(文件)恢复数据库，可以在【还原】选项中选择【从设备】单选按钮，如图 3.11 所示。

图 3.11　从设备还原数据库

单击【选择设备】按钮，如图 3.12 所示。单击【添加】按钮可以选择还原的文件名称。完成后单击【确定】按钮。

图 3.12　选择设备

(3)单击【确定】按钮完成数据库的恢复。

第 4 章 Transact-SQL 程序设计

不同的数据库供应商一般都会对 SQL 语言进行不同程度的扩展，主要是基于两方面的原因：一是数据库供应商开发的系统早于 SQL 语言标准的制定时间；二是不同的数据库供应商为了达到特殊性能和实现新的功能而对标准的 SQL 语言进行了扩展。

Transact-SQL 是 SQL Server 2000 提供的查询语言。使用 Transact-SQL 编写应用程序可以完成所有的数据库管理工作。任何应用程序，只要目的是向 SQL Server 2000 的数据库管理系统发出命令以获得数据库管理系统的响应，最终都必须体现为以 Transact-SQL 语句为表现形式的指令。对用户来说，Transact-SQL 是惟一可以和 SQL Server 2000 的数据库管理系统进行交互的语言。

4.1 Transact-SQL 语言概述

尽管 SQL Server 2000 提供了使用方便的图形化用户界面，但各种功能的实现基础是 Transact-SQL 语言，只有 Transact-SQL 语言可以直接和数据库引擎进行交互。Transact-SQL 语言是基于商业应用的结构化查询语言，是标准 SQL 语言的增强版本。

4.1.1 Transact-SQL语言特点

由于 Transact-SQL 语言直接来源于 SQL 语言，因此它也具有 SQL 语言的几个特点。

1. 一体化

Transact-SQL 语言集数据定义语言、数据操纵语言、数据控制语言和附加语言元素为一体。其中附加语言元素不是标准 SQL 语言的内容，但是它增强了用户对数据库操作的灵活性和简便性，从而增强了程序的功能。

2. 两种使用方式，统一的语法结构

两种使用方式，即联机交互式和嵌入高级语言的使用方式。统一的语法结构使 Transact-SQL 语言可用于所有用户的数据库活动模型，包括系统管理员、数据库管理员、应用程序员、决策支持系统管理人员以及许多其他类型的终端用户。

3. 高度非过程化

Transact-SQL 语言一次处理一个记录，对数据提供自动导航；允许用户在高层的数据结构上工作，可操作记录集，而不是对单个记录进行操作；所有的 SQL 语句接受集合作为输入，返回集合作为输出，并允许一条 SQL 语句的结果作为另一条 SQL 语句的输入。另外，Transact-SQL 语言不要求用户指定对数据的存放方法，所有的 Transact-SQL 语句使用查询优化器，用以指定数据以最快速度存取的手段。

4. 类似于人的思维习惯，容易理解和掌握

SQL 语言的易学易用性，而 Transact-SQL 语言是对 SQL 语言的扩展，因此也是非常容易理解和掌握的。如果对 SQL 语言比较了解，在学习和掌握 Transact-SQL 语言及其高级特性时就更游刃有余了。

4.1.2 Transact-SQL附加语言

Transact-SQL 附加语言元素不是 SQL 的标准内容，而是 Transact-SQL 语言为了编程方便而增加的语言元素。这些语言元素包括变量、运算符、函数、流程控制语句和注释等内容。

1. 变量

变量对于一种语言来说是必不可少的组成部分。Transact-SQL 语言允许使用两种变量：一种是用户自己定义的局部变量(Local Variable)，另一种是系统提供的全局变量(Global Variable)。

(1) 局部变量

局部变量使用户自己定义的变量，它的作用范围近在程序内部。通常只能在一个批处理中或存储过程中使用，用来存储从表中查询到的数据，或当作程序执行过程中暂存变量使用。局部变量使用 DECLARE 语句定义，并且指定变量的数据类型，然后可以使用 SET 或 SELECT 语句为变量初始化；局部变量必须以“@”开头，而且必须先声明后使用。其声明格式如下：

```
DECLARE @变量名 变量类型[,@变量名 变量类型…]
```

其中变量类型可以是 SQL Server 2000 支持的所有数据类型，也可以是用户自定义的数据类型。

局部变量不能使用“变量=变量值”的格式进行初始化，必须使用 SELECT 或 SET 语句来设置其初始值。初始化格式如下：

```
SELECT @局部变量=变量值
SET @局部变量=变量值
```

比如在pubs数据库中使用名为@find的局部变量检索所有姓以Ring开头的作者信息,代码如下：

```
USE pubs
DECLARE @find varchar(30)
SET @find = 'Ring%'
SELECT au_lname, au_fname, phone
FROM authors
WHERE au_lname LIKE @find
```

执行结果：

```
au_lname                au_fname       phone
--------------------------------------- -------------------- ------------
Ringer                  Albert         801 826-0752
Ringer                  Anne           801 826-0752
```

注意：如果声明字符型的局部变量，一定要在变量类型中指明其最大长度，否则系统认为其长度为 1。

(2)全局变量

全局变量是 SQL Server 2000 系统内部使用的变量，起作用范围并不局限于某一程序，而是任何程序均可随时调用。全局变量通常存储一些 SQL Server 2000 的配置设置值和效能统计数据。用户可在程序中用全局变量来测试系统的设定值或者 Transact_SQL 命令执行后的状态值。引用全局变量时，全局变量的名字前面要有两个标记符“@@”。不能定义与全局变量同名的局部变量。从 SQL Server 7.0 开始，全局变量就以系统函数的形式使用。全局变量的符号及其功能如表 4.1 所示。

表 4.1　全局变量及其功能

全局变量	功　能
@@CONNECTIONS	自 SQL Server 2000 最近一次启动以来登录或试图登录的次数
@@CPU_BUSY	自 SQL Server 2000 最近一次启动以来 CPU Server 的工作时间
@@CURRSOR_ROWS	返回在本次连接最新打开的游标中的行数
@@DATEFIRST	返回 SET DATEFIRST 参数的当前值
@@DBTS	数据库的惟一时间标记值
@@ERROR	系统生成的最后一个错误，若为 0 则成功
@@FETCH_STATUS	最近一条 FETCH 语句的标志
@@IDENTITY	保存最近一次的插入身份值
@@IDLE	自 CPU 服务器最近一次启动以来的累计空闲时间
@@IO_BUSY	服务器输入输出操作的累计时间
@@LANGID	当前使用的语言的 ID
@@LANGUAGE	当前使用语言的名称
@@LOCK_TIMEOUT	返回当前锁的超时设置
@@MAX_CONNECTIONS	同时与 SQL Server 2000 相连的最大连接数量
@@MAX_PRECISION	十进制与数据类型的精度级别
@@NESTLEVEL	当前调用存储过程的嵌套级，范围为 0～16
@@OPTIONS	返回当前 SET 选项的信息
@@PACK_RECEIVED	所读的输入包数量
@@PACKET_SENT	所写的输出包数量
@@PACKET_ERRORS	读与写数据包的错误数

@@RPOCID	当前存储过程的 ID
@@REMSERVER	返回远程数据库的名称
@@ROWCOUNT	最近一次查询涉及的行数
@@SERVERNAME	本地服务器名称
@@SERVICENAME	当前运行的服务器名称
@@SPID	当前进程的 ID
@@TEXTSIZE	当前最大的文本或图像数据大小

续表

全局变量	功　能
@@TIMETICKS	每一个独立的计算机报时信号的间隔(ms)数，报时信号为 31.25ms 或 1/32s
@@TOTAL_ERRORS	读写过程中的错误数量
@@TOTAL_READ	读磁盘次数(不是高速缓存)
@@TOTAL_WRITE	写磁盘次数
@@TRANCOUNT	当前用户的活动事务处理总数
@@VERSION	当前 SQL Server 的版本号

2. 流程控制语句

Transact-SQL 语言提供了一些可以用于改变语句执行顺序的命令，称为流程控制语句。流程控制语句允许用户更好地组织存储过程中的语句，方便地实现程序的功能。流程控制语句与常见的程序设计语言类似，主要包含以下几种。

(1)　IF…ELSE 语句

```
IF〈条件表达式〉
   〈命令行或程序块〉
[ELSE [条件表达式]
   〈命令行或程序块〉]
```

其中<条件表达式>可以是各种表达式的组合，但表达式的值必须是“真”或“假”。ELSE 子句是可选的。IF…ELSE 语句用来判断当某一条件成立时执行某段程序，条件不成立时执行另一段程序。如果不使用程序块，IF 或 ELSE 只能执行一条命令。IF…ELSE 可以嵌套使用，最多可嵌套 32 级。

(2)　BEGIN…END 语句

```
BEGIN
   〈命令行或程序块〉
END
```

BEGIN…END 用来设置一个程序块，该程序块可以被视为一个单元执行。BEGIN…END 经常在条件语句中使用，如 IF…ELSE 语句。如果当 IF 或 ELSE 子句为真时，想让程序执行其后的多条语句，这时就要把这多条语句用 BEGIN…END 括起来使之成为一个语句块。在 BEGIN…END 语句中可以嵌套另外的 BEGIN…END 语句来定义另一程序块。

(3) CASE 语句

```
CASE<运算式>
   WHEN<运算式>THEN<运算式>
   …
```

```
WHEN<运算式>THEN<运算式>
[ELSE<运算式>]
END
```

例如，在 pubs 数据库中查询每个作者所居住州的全名，可以使用如下代码实现：

```
SELECT au_fname, au_lname,
  CASE state
    WHEN 'CA' THEN 'California'
    WHEN 'KS' THEN 'Kansas'
    WHEN 'TN' THEN 'Tennessee'
    WHEN 'OR' THEN 'Oregon'
    WHEN 'MI' THEN 'Michigan'
    WHEN 'IN' THEN 'Indiana'
    WHEN 'MD' THEN 'Maryland'
    WHEN 'UT' THEN 'Utah'
     END AS StateName
FROM pubs.dbo.authors
ORDER BY au_lname
```

执行结果：

```
au_fname      au_lname                 StateName
-------------------- ---------------------------------------- ----------
Abraham       Bennet                   California
Reginald      Blotchet-Halls             Oregon
Cheryl        Carson                   California
Michel        DeFrance                   Indiana
Innes         del Castillo               Michigan
Ann          Dull                    California
…
```

(4) WHILE…CONTINUE…BREAK 语句

```
WHILE<条件表达式>
BEGIN
   <命令行或程序块>
   [BREAK]
   [CONTINUE]
   [命令行或程序块]
END
```

WHILE 语句在设置的条件为真时会重复执行命令行或程序块。CONTINUE 语句可以让程序跳过 CONTINUE 语句之后的语句，回到 WHILE 循环的第一行。BREAK 语句则让程序完全跳出循环，结束 WHILE 循环的执行。WHILE 语句也可以嵌套使用。

注意：如果嵌套了两个或多个 WHILE 循环，内层的 BREAK 语句将导致退出到下一个外层循环。首

先运行内层循环结束之后的所有语句，然后下一个外层循环重新开始执行。

3. 注释

在 Transact-SQL 语言中可使用两种注释符：行注释和块注释。

行注释符为“--”，这是 ANSI 标准的注释符，用于单行注释。

块注释符为“/*...*/”，“/*”用于注释文字的开头，“*/”用于注释文字的末尾。块注释符可在程序中标识多行文字为注释。

4.1.3 查询分析器的使用

第 2 章简单介绍了 SQL Server 2000 提供的查询分析器工具，这里详细说明一下它的使用方法。

查询分析器使用一个图形用户界面，用以交互地设计和测试 Transact-SQL 语句、批处理和脚本，其提供以下几项功能。

- 用于输入 Transact-SQL 语句的自由格式文本编辑器。
- 在 Transact-SQL 语法中使用不同的颜色，以提高复杂语句的易读性。
- 使用对象浏览器和对象搜索工具可以轻松查找数据库中的对象和对象结构。
- 模板可用于加快创建 SQL Server 对象的 Transact-SQL 语句的开发速度。模板是包含创建数据库对象所需的 Transact-SQL 语句基本结构的文件。
- 有用于分析存储过程的交互式调试工具。
- 以网格或自由格式文本窗口的形式显示结果。
- 显示计划信息的图形关系图，用以说明内置在 Transact-SQL 语句执行计划中的逻辑步骤。这使程序员得以确定在性能差的查询中，具体是哪一部分使用了大量资源。之后，程序员可以试着采用不同的方法更改查询，使查询使用的资源减到最小同时仍返回正确的数据。
- 使用索引优化向导分析 Transact-SQL 语句及其所引用的表，以了解通过添加其他索引是否可以提高查询的性能。

在启动查询分析器并连接到数据库后，用户就可以进行操作了。如图 4.1 所示，查询分析器主界面有三个主要部分，左边部分为对象浏览器，可以在这里看到所有的数据库及其对象信息；上半部分为 SQL 命令的输入窗格，可以在这里输入并调试 Transact-SQL 语句；下半部分为执行结果输出窗格。用户可以在命令输入窗格中输入 Transact-SQL 应用程序，程序执行后可以在结果输出窗格中看见程序执行的结果或错误信息。用户可以在主菜单中选择查询结果的输出方式，由于这里选择了结果以表格的形式输出，所以有关程序执行的消息可以切换到【消息】选项卡中进行查看。

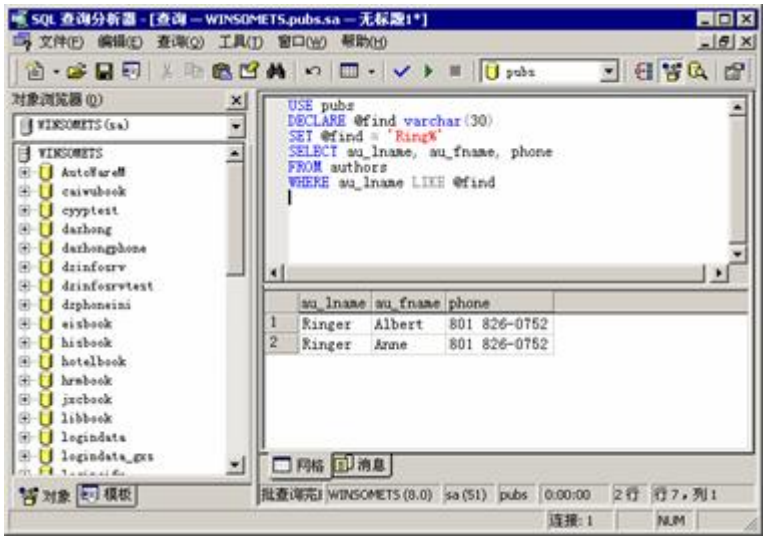


图 4.1 查询分析器的编辑界面

4.2 Transact-SQL 语言数据类型

SQL 语言是所有关系数据库通用的标准语言，Transact-SQL 语言在标准 SQL 语言的基础上进行了

功能上的扩充，由命令语句、基本数据类型、函数和运算符组成。SQL Server 2000 的 Transact-SQL 语言也有一些自己的特色，从而增加了用户对数据库操作的方便性和灵活性。

在 SQL Server 2000 中，每个变量、参数和表达式都有数据类型。所谓数据类型就是以数据的表现方式和存储方式来划分的数据的种类。SQL Server 2000 中提供多种基本数据类型，如表 4.2 所示。

其中 bigint 和 sql_variant 是 SQL Server 2000 中新增的数据类型。另外， SQL Server 2000 还新增了 table 基本数据类型，该数据类型可用于存储 SQL 语句的结果集。table 数据类型不适用于表中的列，而只能用于 Transact-SQL 变量和用户定义函数的返回值。

表 4.2 SQL Server 2000 的基本数据类型

binary	bigint	bit	char	datetime
decimal	float	image	int	money
nchar	ntext	nvarchar	numeric	real
smalldatetime	smallint	smallmoney	sql_variant	sysname
text	timestamp	tinyint	varbinary	varchar
uniqueidentifier				

4.2.1 整数数据类型

整数型数据包括 bigint 型、int 型、smallint 型和 tinyint 型。

- bigint型数据的存储大小为 8 个字节，共 64 位。其中 63 位用于表示数值的大小，1 位用于表示符号。bigint型数据可以存储的数值范围是-2^{63}～$2^{63}-1$，即 -9 223 372 036 854 775 808 ～9 223 372 036 854 775 807。
- int型数据的存储大小为 4 个字节，共 32 位。其中 31 位用于表示数值的大小，1 位用于表示符号。int型数据存储的数值范围是-2^{31}～$2^{31}-1$，即-2 147 483 648～ 2 147 483 647。
- smallint型数据的存储大小为 2 个字节，共 16 位。其中 15 位用于表示数值的大小，1 位用于表示符号。smallint型数据存储的数值范围是-2^{15}～$2^{15}-1$，即-32 768～ 32 767。
- tinyint型数据的存储大小只有 1 个字节，共 8 位，全部用于表示数值的大小，由于没有符号位，所以tinyint型的数据只能表示正整数。tinyint型数据存储的数值范围是-2^{7}～$2^{7}-1$，即-256～255。

4.2.2 浮点数据类型

浮点数据类型用于存储十进制小数。在 SQL Server 2000 中浮点数值的数据采用上舍入(Round up)的方式进行存储，也就是说，要舍入的小数部分不论其大小，只要是一个非零的数，就要在该数字的最低有效位上加 1，并进行必要的进位。由于浮点数据为近似值，所以并非数据类型范围内的所有数据都能精确地表示。

浮点数据类型包括 real 型、float 型、decimal 型和 numeric 型。

- real 型数据的存储大小为 4 个字节，可精确到小数点后第 7 位数字。这种数据类型的数据存储范围为从-3.40E+38～-1.18E-38，0 和 1.18E-38～3.40E+38。
- float 型的数据存储大小为 8 个字节，可精确到小数点后第 15 位数字。这种数据类型的数据存储范围为从-1.79E+308～-2.23E-308，0 和 2.23E+308～1.79E+308。

float 型的数据可写成 float[(n)]的形式。其中 n 是 1～15 之间的整数值，指定 float 型数据的精度。当 n 为 1～7 时，实际上用户定义了一个 real 型的数据，系统用 4 个字节存储；当 n 为 8～15 时，系统认为它是个 float 型的数据，用 8 个字节存储它。这样既增强了数据定义的灵活性，又节省了空间。

- decimal数据类型和numeric数据类型的功能完全一样，它们都可以提供小数所需要的实际存储空间，但也有一定的限制，用户可以用 2～17 个字节来存储数据，取值范围是$-10^{38}+1$～$10^{38}-1$。

decimal 型数据和 numeric 型数据的定义格式为 decimal[(p,[s])]和 numeric[(p,[s])]，其中 p 表示可供存储的值的总位数(不包括小数点)，默认值为 18；s 表示小数点后的位数，默认值为 0；参数之间的关系是 0≤s≤p。例如：decimal(15,5)表示共有 15 位数，其中整数 10 位，小数 5 位。

4.2.3 二进制数据类型

二进制数据类型用于存储二进制数据，包括 binary 型、varbinary 型和 image 型。

- binary 型是固定长度的二进制数据类型,其定义形式为 binary(n),其中 n 表示数据的长度，取值为 1～8 000。在使用时应指定 binary 型数据的大小，默认值为 1 个字节。binary 类型的数据占用 n+4 个字节的存储空间。

在输入数据时必须在数据前加上字符“0X”作为二进制标识。例如:要输入“abc”则应输入“0Xabc”。若输入的数据位数为奇数，则系统会自动在起始符号“0X”的后面添加一个 0。如上述输入“0Xabc”后，系统会自动变为“0X0abc”。

- varbinary 型是可变长度的二进制数据类型，其定义形式为 varbinary(n)，其中 n 表示数据的长度，取值为 1～8 000。如果输入的数据长度超出 n 的范围，则系统会自动截掉超出部分。

varbinary 型具有变动长度的特性,因为 varbinary 型数据的存储长度为实际数值长度+4 个字节。当 binnary 型数据允许 null 值时，将被视为 varbinary 型的数据。

一般情况下，由于 binary 型的数据长度固定，因此它比 varbinary 型的数据处理速度快。

- l　　image型的数据也是可变长度的二进制数据，其最大长度为 $2^{31}-1$(2 147 483 647)个字节。

4.2.4 逻辑数据类型

逻辑数据类型只有一种 bit 型。bit 数据类型只占用 1 个字节的存储空间，其值为 0 和 1。只要输入的值为非 0，系统都会当作 1 处理。另外，bit 型不能定义为 null 值。

4.2.5 字符数据类型

字符数据类型是使用最多的数据类型，它可以用来存储各种字母、数字符号、特殊符号等。一般情况下，使用字符类型数据时，须在数据的前后加上单引号或双引号。字符数据类型包括 char 型、nchar 型、varchar 型和 nvarchar 型。

- char 型是固定长度的非 Unicode 字符数据类型，在存储时每个字符和符号占用一个字节的存储空间。其定义形式为 char[(n)],其中 n 表示所有字符所占的存储空间,取值为 1～8 000，即可容纳 8000 个 ANSI 字符，默认值为 1。若输入的数据字符数小于 n 定义的范围，则系统自动在其后添加空格来填满设定好的空间；若输入的数据字符数超过 n 定义的范围，则系统自动截掉超出部分。
- nchar 型是固定长度的 Unicode 字符数据类型，由于 Unicode 标准规定在存储时每个字符和符号占用 2 个字节的存储空间,因此 nchar 型的数据比 char 型数据多占用一倍的存储空间。其定义形式为 nchar[(n)]，其中 n 表示所有字符所占的存储空间，取值为 1～4 000，即可容纳 4 000 个 Unicode 字符，默认值为 1。

使用 Unicode 标准字符集的好处是由于它使用两个字节作存储单位，使得一个存储单位的容量大大增加，这样就可以将全世界的语言文字都囊括在内。当用户在一个数据列中同时输入不同语言的文字符号时，系统不会出现编码冲突。

- varchar 型是可变长度的非 Unicode 字符数据类型。其定义形式为 varchar[(n)]。它与 char 型类似，n 的取值范围是 1～8 000。由于 varchar 型具有可变长度的特性，所以 varchar 型数据的存储长度为实际数值的长度。如果输入数据的字符数小于 n 定义的长度，系统也不会

像 char 型那样在数据后面用空格填充；但是如果输入的数据长度大于 n 定义的长度，系统会自动截掉超出部分。

一般情况下，由于 char 型的数据长度固定，因此它比 varchar 型数据的处理速度快。

- nvarchar 型是可变长度的 Unicode 字符数据类型，其定义形式为 nvarchar[(n)]。由于它采用了 Unicode 标准字符集，因此 n 的取值范围是从 1～4 000。nvarchar 型的其他特性与 varchar 类型相似。

4.2.6 文本和图形数据类型

文本和图形数据类型是用于存储大量的非 Unicode 和 Unicode 字符以及二进制数据的固定长度和可变长度数据类型，包括 text 型、ntext 型和 image 型。

- text型是用于存储大量非Unicode文本数据的可变长度数据类型，其容量理论上为 2^{31}-1(2 147 483 647)个字节。在实际应用时需要视硬盘的存储空间而定。

在 SQL Server 2000 以前的版本中，数据库中一个 text 对象存储的实际上是一个指针，它指向一个以 8KB 为单位的数据页。这些数据页是动态增加并被逻辑连接起来的。在 SQL Server 2000 中，则将 text 和 image 型的数据直接存放到表的数据行中，而不是存放到不同的数据页中。这样就减少了用于存储 text 和 image 类型的空间，并相应减少了磁盘处理这类数据的 I/O 数量。

- ntext型是用于存储大量Unicode文本数据的可变长度数据类型，其理论容量为 2^{30}-1(1 073 741 823)个字节。ntext型的其他用法与text型基本一样。
- image型是用于存储大量二进制数据的可变长度数据类型，其理论容量为 2^{31}-1 (2 147 483 647)个字节。Image型数据的存储模式与text型数据相同，通常用来存储图形等OLE对象。在输入数据时，与输入二进制数据一样，必须在数据前加上起始符号“0X”作为二进制标识。

4.2.7 日期和时间数据类型

日期和时间数据类型代表日期和一天内的时间，包括 datetime 型和 smalldatetime 型。

- datetime 型是用于存储日期和时间的结合体的数据类型。它可以存储从公元 1753 年 1 月 1 日零时起到公元 9999 年 12 月 31 日 23 时 59 分 59 秒之间的所有日期和时间，其精确度可达 3%秒。

datetime 型数据所占用的存储空间为 8 个字节，其中前 4 个字节用于存储 1900 年 1 月 1 日以前或以后的天数，数值分正负，正数表示在此日期之后的日期，负数表示在此日期之前的日期；后 4 个字节用于存储从此日零时起所指定的时间经过的毫秒数。如果在输入时省略了时间部分，则系统将默认为 12:00:00:000AM；如果省略了日期部分，系统将默认为 1900 年 1 月 1 日。

- smalldatetime 型与 datetime 型相似，但其存储的日期时间范围较小，从 1900 年 1 月 1 日到 2079 年 6 月 6 日。它的精度也较低，只能精确到分钟级，其分钟个位上的值是根据秒数并以 30 秒为界四舍五入得到的。

Smalldatetime 型数据所占用的存储空间为 4 个字节，其中前两个字节存储从基础日期 1900 年 1 月 1 日以来的天数，后两个字节存储此日零时起所指定的时间经过的分钟数。

4.2.8 货币数据类型

货币数据类型用于存储货币或现金值，包括 money 型和 smallmoney 型。在使用货币数据类型时，应在数据前加上货币符号，以便系统辨识其为哪国的货币，如果不加货币符号，则系统默认为“￥”。

- money型是一个有 4 位小数的decimal值，其取值从-2^{63}(-9 223 372 036 854 775 808) ～$2^{63}-1$(+9 223 372 036 854 775 807)，精确到货币单位的千分之十。存储大小为 8 个字节。
- smallmoney 型货币数据值介于-2 147 483 648～+2 147 483 647 之间，精确到货币单位的千分之十。存储大小为 4 个字节。

4.2.9 特定数据类型

SQL Server 2000 中包含了一些用于数据存储的特殊数据类型，包括 timestamp 型和 uniqueidentifier 型。

- timestamp 数据类型提供数据库范围内的惟一值，它相当于 binary(8)或 varbinary(8)，但当它所定义的列在进行更新或插入数据行操作时，此列的值会自动更新。每个数据库表中只能有一个 timestamp 型数据列。如果表中的列名为“timestamp”，则该列的类型将被自动定义为 timestamp 型。
- uniqueidentifier 数据类型用于存储一个 16 位的二进制数据，此数据称为全局惟一标识符(Globally Unique Identifier，GUID)。此数据由 SQL Server 2000 的 NEWID()函数产生，在全球各地的计算机经由此函数产生的数字不会相同。

4.2.10 用户自定义数据类型

sysname 数据类型是系统提供给用户的，便于用户自定义的数据类型。该数据类型被定义为 NVARCHAR(128)，即它可以存储 128 个 Unicode 字符或 256 个一般字符。

4.2.11 新增数据类型

前面讲到过 SQL Server 2000 新增了 3 种数据类型：bigint 型、sql_variant 型和 table 型。其中 bigint 型已经在整数类型中介绍过了。

- sql_variant 型是一种存储 SQL Server 支持的各种数据类型(text、ntext、timestamp 和 sql_variant 除外)值的数据类型。此数据类型大大方便了 SQL Server 的开发工作。
- table 型用于存储对表或视图处理后的结果集。table 数据类型不适用于表中的列，而只能用于 Transact-SQL 变量和用户定义函数的返回值。这一类型使得变量可以存储一个表，从而使函数或过程返回查询结果更加方便快捷。

4.3 Transact-SQL 语言运算符

运算符是一种符号，用来指定要在一个或多个表达式中指定的操作。SQL Server 2000 中使用如下几种运算符：算术运算符、赋值运算符、位运算符、比较运算符、逻辑运算符、字符串串联运算符和一元运算符。

1. 算术运算符

算术运算符用来在两个表达式上执行数学运算，这两个表达式可以是任意两个数字数据类型的表达式。算术运算符包括+(加)、-(减)、*(乘)、/(除)、%(模)五个。

在 Transact-SQL 中，“+”包含了三个方面的意义：

- 表示正号，即在数值前添加“+”号表示该数值是一个正数。
- 表示算术运算的加号，能将数值类型的两个数据相加。
- 连接两个字符型或 binary 型的数据，这时的“+”号叫做字符串串联运算符。

2. 赋值运算符

Transact-SQL 有一个赋值运算符，即等号(=)。

例如，下面的代码创建了@MyCounter 变量。然后赋值运算符将@MyCounter 设置成一个由表达式返回的值。

```
DECLARE @MyCounter INT
SET @MyCounter = 1
```

3. 位运算符

位运算符在两个表达式之间执行位操作，这两个表达式可以是任意两个整型数据类型的表达式。位运算符的符号及其定义如表 4.3 所示。

表 4.3 位运算符

运 算 符	含 义
& (按位 AND)	按位与(两个操作数)
| (按位 OR)	按位或(两个操作数)
^ (按位互斥 OR)	按位异或(两个操作数)
~ (按位 NOT)	按位取反(一个操作数)

位运算符的操作数可以是整型或二进制字符串数据类型中的任何数据类型(但 image 数据类型除外)，此外，两个操作数不能同时是二进制字符串数据类型中的某种数据类型。

4. 比较运算符

比较运算符用来测试两个表达式是否相同。除了 text、ntext 或 image 数据类型的表达式外，比较运算符可以用于所有的表达式。比较运算符的符号及其含义如表 4.4 所示。

表 4.4 比较运算符

运 算 符	含 义
=	等于
>	大于
<	小于
>=	大于等于
<=	小于等于
<>	不等于
!=	不等于(非 SQL-92 标准)
!<	不小于(非 SQL-92 标准)
!>	不大于(非 SQL-92 标准)

比较运算符的结果是布尔数据类型，它有三种值：TRUE、FALSE 和 NULL。那些返回布尔数据类型的表达式被称为布尔表达式。

和其他 SQL Server 数据类型不同，不能将布尔数据类型指定为表列或变量的数据类型，也不能在结果集中返回布尔数据类型。

当 SET ANSI_NULLS 为 ON 时,带有一个或两个 NULL 表达式的运算符返回 NULL。当 SET ANSI_NULLS 为 OFF 时，上述规则同样适用，只不过如果两个表达式都为 NULL，那么等号运算符返回 TRUE。例如，如果 SET ANSI_NULLS 是 OFF，那么 NULL=NULL 就返回 TRUE。

在 WHERE 子句中使用带有布尔数据类型的表达式，可以筛选出符合搜索条件的行，也可以在流控制语言语句(例如 IF 和 WHILE)中使用这种表达式。

5. 逻辑运算符

逻辑运算符用来对某个条件进行测试，以获得其真实情况。逻辑运算符和比较运算符一样，返回带有 TRUE 或 FALSE 值的布尔数据类型。逻辑运算符的符号及其含义如表 4.5 所示。

表 4.5 逻辑运算符

运 算 符	含 义
ALL	如果一系列的比较都为 TRUE，那么就为 TRUE
AND	如果两个布尔表达式都为 TRUE，那么就为 TRUE
ANY	如果一系列的比较中任何一个为 TRUE,那么就为 TRUE
BETWEEN	如果操作数在某个范围之内，那么就为 TRUE
EXISTS	如果子查询包含一些行，那么就为 TRUE
IN	如果操作数等于表达式列表中的一个，那么就为 TRUE
LIKE	如果操作数与一种模式相匹配，那么就为 TRUE
NOT	对任何其他布尔运算符的值取反
OR	如果两个布尔表达式中的一个为 TRUE,那么就为 TRUE
SOME	如果在一系列比较中，有些为 TRUE，那么就为 TRUE

6. 一元运算符

一元运算符只对一个表达式执行操作，这个表达式可以是数字数据类型中的任何一种数据类型，如表 4.6 所示。

表 4.6 一元运算符

运 算 符	含 义
+ (正)	数值为正
- (负)	数值为负

~ (按位 NOT)	返回数字的补数

4.4 数据定义语言

数据定义语言(DDL)用来定义数据的结构，如创建、修改或者删除数据库对象，常用的数据定义语言有 CREATE、ALTER、DROP 等。

4.4.1 创建表

在创建表时要遵循严格的语法定义。在 Transact-SQL 语言中，必须满足以下规定：

每个表有一个名称，称为表名或关系名。表名必须以字母开头，最大长度为 30 个字符。

一张表包含若干列，列名惟一，列名也成为属性名。

同一列的数据必须要有相同的数据类型。

表中的每一列数值必须为一个不可分割的数据项。

表中的一行称为一条记录。

创建一张新表可以使用 CREATE TABLE 命令，其格式如下：

```
CREATE TABLE
  [ database_name.[ owner ] .| owner.] table_name
  ( { < column_definition >| column_name AS computed_column_expression | < table_constraint > } [ ,...n ] )
  [ ON { filegroup | DEFAULT } ] [ TEXTIMAGE_ON { filegroup | DEFAULT } ]
< column_definition > ::= { column_name data_type } [ [ DEFAULT constant_expression ]| [ IDENTITY [ ( seed , increment ) [ NOT FOR REPLICATION ] ] ]] [ ROWGUIDCOL]
  [ < column_constraint > ] [ ...n ]
```

各参数说明如下：

- database_name 指定创建新表的数据库名称。该名称必须是现有数据库的名称。如果不指定数据库，database_name 默认为当前数据库。而且连接 SQL Server 的用户名在指定的数据库中有建立表格的权限时，建表操作才能顺利执行。
- owner 指定新表的所有者，如果不指定，系统认为建表人就是表的拥有者。只有管理员或数据库拥有者才能将表的拥有者指定为其他人。owner 必须是 database_name 所指定的数据库中的已经存在的用户名，owner 默认为与 database_name 所指定的数据库中的当前连接相关联的用户名。
- table_name 是新表的名称。表名必须符合标识符规则。数据库中的 owner.table_ name 组合必须惟一并且最多可包含 128 个字符。

l　column_name 是表中的列名。列名必须符合标识符规则，在表内必须惟一，并且必须指定。以 timestamp 数据类型创建的列可以省略列名，默认为 timestamp。

- computed_column_expression 指定一个表达式，用于定义计算列的产生规律。计算列是物理上并不存储在表中的虚拟列。计算列由同一表中的其他列通过表达式计算得到。例如，计算列可以这样定义：cost AS price * qty。表达式可以是非计算列的列名、常量、函数、变量，也可以是用一个或多个运算符连接的上述元素的任意组合。表达式不能为子查询。

计算列可用于选择列表、WHERE 子句、ORDER BY 子句或任何其他可使用常规表达式的位置，但下列情况除外：

- 计算列不能用作 DEFAULT 或 FOREIGN KEY 约束定义，也不能与 NOT NULL 约束定义一起使用。但是，如果计算列由具有确定性的表达式定义，并且索引列中允许计算结果的数据类型，则可将该列用作索引中的键列，或用作 PRIMARY KEY 或 UNIQUE 约束的一部分。

- 计算列不能作为 INSERT 或 UPDATE 语句的作用对象。
- ON {filegroup | DEFAULT}指定存储表的文件组。指定的文件组必须存在于数据库中。如果选择 DEFAULT，或者忽略该参数，则表存储在默认文件组中。
- TEXTIMAGE_ON 是表示 text、ntext 和 image 列存储在指定文件组中的关键字。如果表中没有 text、ntext 或 image 列，则不能使用 TEXTIMAGE ON。如果没有指定 TEXTIMAGE_ON，则 text、ntext 和 image 列将与表存储在同一文件组中。
- data_type 指定列的数据类型。可以是系统数据类型或用户自定义数据类型。
- DEFAULT 指定某列的默认值。如果某列定义了默认值，在输入数据时，如果不为该列输入专门的数据，则该列的值取默认值。
- IDENTITY(seed,increment)用于指定某列为标识列。seed 指定标识列的初始值，increment 指定递增量。如果某列被指定为标识列，则在该表中所有记录的该列的值都是惟一的。该列的值第一行取 seed 参数的指定值，以后每增加一行，该列的值取前一行的值加上 increment 的值。
- NOT FOR REPLICATION 指定列的 IDENTITY 属性在把从其他表中复制的数据插入到表中时不发生作用。
- ROWGUIDCOL 表示新列是行的全局惟一标识符列。对于每个表只能指派一个 uniqueidentifier 列作为 ROWGUIDCOL 列。ROWGUIDCOL 属性只能指派给 uniqueidentifier 列。ROWGUIDCOL 属性不会使列具有惟一性，也不会自动生成一个新的数值给插入的行。需要在 INSERT 语句中使用 NEWID()函数或指定列的默认值为 NEWID()函数。

另外，在创建表时，还要注意以下几点：

- 一个表至少有一列，最多不超过 1 024 列。
- 每个数据库中最多可以创建 200 万个表。
- 表在存储时使用的计量单位是盘区(Extent)。一个盘区分为 8 个数据项，每页 8KB 字节。在创建表时，会分配给它一个初始值为一个盘区的存储空间。当增加表的存储空间时，以盘区为单位增加。

例如下面的代码将建立一个学生表 student。

```
CREATE TABLE student
(
   ID_Card varchar(18) not null,
   Sname varchar(10) not null,
   Ssex char(2) not null,
   Sage int not null,
   Sschool_number char(6),
   CONSTRAINT stud_ID_card primary key(ID_Card)
)
```

在创建表的时候通常要定义一些约束，例如学生的身份证号是惟一的，在表中应该作为主键值，学生的年龄不超过 35 岁等等。在 SQL Server 中约束可以分为列约束(Column Constraint)和表约束(Tbale Constraint)。列约束作为列定义的一部分制作用于此列本身；表约束作为表定义的一部分，可以作用于多个列。

表约束中经常用到的是主键约束，它能惟一地指定一行记录。每个表中只能有一列被指定为主键，且被指定的主键列不能允许空值属性。

下面的代码建立一个贷款单表，其中贷款金额不得超过 10 万元。

```
CREATE TABLE loan
```

```
(
   loan_number char(6) not null,
   amount money not null,
   constraint pk_loan_num primary key(loan_number),
   constraint chk_amount check(amount<=100000)
)
```

4.4.2 删除表

删除表指的是删除表定义及该表的所有数据、索引、触发器、约束和权限规范。语法格式如下：

```
DROP TABLE table_name
```

其中 table_name 就是要删除的表名。例如要删除前面建立的 loan 表，代码如下：

```
DROP TABLE loan
```

表的所有者可以除去任何数据库内的表。除去表时，表上的规则或默认值将解除绑定，任何与表关联的约束或触发器将自动除去。如果重新创建表，必须重新绑定适当的规则和默认值，重新创建任何触发器并添加必要的约束。在系统表上不能使用 DROP TABLE 语句。

4.4.3 修改表

Alter Table 语句可以完成对现有表的修改。可以更改、添加、除去列和约束，或者启用或禁用约束和触发器。语法格式如下：

```
ALTER TABLE table
{ [ ALTER COLUMN column_name
  { new_data_type [ ( precision [ , scale ] ) ]
    [ COLLATE < collation_name > ]
    [ NULL | NOT NULL ]
    | {ADD | DROP } ROWGUIDCOL }
  ]
  | ADD
    { [ < column_definition > ]
    | column_name AS computed_column_expression
    } [ ,...n ]
  | [ WITH CHECK | WITH NOCHECK ] ADD
    { < table_constraint > } [ ,...n ]
  | DROP
    { [ CONSTRAINT ] constraint_name
      | COLUMN column } [ ,...n ]
  | { CHECK | NOCHECK } CONSTRAINT
    { ALL | constraint_name [ ,...n ] }
  | { ENABLE | DISABLE } TRIGGER
    { ALL | trigger_name [ ,...n ] }
}
```

通常表的结构可以用 CREATE TABLE 语句一次创建，但是当发现表的定义有不符合需要的情况或者是实际的业务需要建立新的约束，此时可以用 Alter Table 来调整表结构。例如我们给前面建立的 student 学生表添加一个民族字段，代码如下：

```
ALTER TABLE student ADD Smingzu char(10)
```

4.4.4 创建和管理视图

对于视图的管理包括创建、修改和删除视图。

1. 定义视图

SQL 语言用 CREATE VIEW 命令建立视图，其常用语法如下：

```
CREATE VIEW <视图名>[(<列名>[,<列名>]...)]
AS <子查询>
[WITH CHECK OPTION];
```

其中子查询可以是任意复杂的 SELECT 语句,但通常不允许含有 ORDER BY 子句和 DISTINCT 短语。

WITH CHECK OPTION 表示对视图进行 UPDATE、 INSERT 和 DELETE 操作时要保证更新、插入或删除的行满足视图定义中的谓词条件(即子查询中的条件表达式)。

如果 CREATE VIEW 语句仅指定了视图名，省略了组成视图的各个属性列名，则隐含该视图由子查询中 SELECT 子句目标列中的诸字段组成。

2. 修改视图定义

要改变一个已经创建的视图的定义，用 ALETER VIEW 语句，其常用语法如下：

```
CREATE VIEW <视图名>[(<列名>[,<列名>]...)]
AS <子查询>
[WITH CHECK OPTION];
```

3. 删除视图

用 DROP VIEW 语句从当前数据库中删除视图。删除视图时，将从 sysobjects、syscolumns、syscomments、sysdepends 和 sysprotects 系统表中删除视图的定义及其他有关视图的信息。其常用语法如下：

```
DROP TABLE table_name
```

4.4.5 创建和管理存储过程

对存储过程的管理包括创建、修改和删除存储过程。

1. 创建存储过程

SQL 语言用 CREATE PROCEDURE 命令建立视图，其常用语法如下：

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
  [ { @parameter data_type }
    [ VARYING ] [ = default ] [ OUTPUT ]
  ] [ ,...n ]

[ WITH
```

```
  { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]

[ FOR REPLICATION ]

AS sql_statement [ ...n ]
```

number 是可选的整数，用来对同名的过程分组，以便用一条 DROP PROCEDURE 语句即可将同组的过程一起删除。例如名为 orders 的应用程序使用的过程可以命名为 orderproc;1、orderproc;2 等。DROP PROCEDURE orderproc 语句将删除整个组。如果名称中包含定界标识符，则数字不应包含在标识符中，只应在 procedure_name 前后使用适当的定界符。

@parameter 用于定义存储过程的参数。在 CREATE PROCEDURE 语句中可以声明一个或多个参数。用户必须在执行过程时提供每个所声明参数的值(除非定义了该参数的默认值)。存储过程最多可以有 2 100 个参数。使用 @ 符号作为第一个字符来指定参数名称。参数名称必须符合标识符的规则。每个过程的参数仅用于该过程本身；相同的参数名称可以用在其他过程中。默认情况下，参数只能代替常量，而不能用于代替表名、列名或其他数据库对象的名称。

sql_statement 就是要建立的存储过程的主体,其中可以包含的任意数目和类型的 Transact-SQL 语句。

2. 修改存储过程

要改变一个已经创建的存储过程，用 ALETER PROCEDURE 语句，其常用语法如下：

```
ALTER PROC [ EDURE ] procedure_name [ ; number ]
  [ { @parameter data_type }
    [ VARYING ] [ = default ] [ OUTPUT ]
  ] [ ,...n ]

[ WITH
  { RECOMPILE | ENCRYPTION
    | RECOMPILE , ENCRYPTION
  }
]
[ FOR REPLICATION ]
AS
  sql_statement [ ...n ]
```

3. 删除视图

用 DROP PROCEDURE 从当前数据库中删除存储过程。其常用语法如下：

```
DROP PROCEDURE procedure _name
```

4.5 数据操作语言

建立了数据库和表后，就需要对这些表进行操作了，一个信息系统其实就是对信息数据进行增(加)、删(除)、(修)改和查(询)的操作。

4.5.1 向表中插入记录

INSERT 语句用来向表中增加数据，可以一次追加一条数据，也可以从另外的表或查询中一次追加多条数据，其常用语法如下：

```
INSERT [ INTO]
  { table_name WITH ( < table_hint_limited > [ ...n ] )
    | view_name
    | rowset_function_limited
  }

  {  [ ( column_list ) ]
    { VALUES
      ( { DEFAULT | NULL | expression } [ ,...n] )
      | derived_table
      | execute_statement
    }
  }
```

例如，向 student 表中插入数据，代码如下：

```
INSERT INTO student(ID_Card,Sname,Ssex,Sage,Sschool_number)
VALUES('11015019821228003','孙晓明','男',20,'C_20')
```

如果不指定字段的列表，那么就必须对表中出现的字段一一赋值，或者使用字段的默认值。所以上面的语句也可以写成：

```
INSERT INTO student
VALUES('11015019821228003','孙晓明','男',20,'C_20')
```

当有大量数据需要插入到表中时，可以使用 SELECT 语句实现大量数据的插入。例如，新建一个 student2 表，结构和 student 表一致并插入所有 student 表中的记录(已经向 student 表中输入了数据)，代码如下：

```
INSERT student2 SELECT * FROM student
```

4.5.2 从表中删除记录

当不再使用表中的记录时，可以使用 DELETE 语句将其删除。使用 DELETE 语句可以一次删除一条或多条记录，而且可以使用 WHERE 子句指定删除条件。其语法格式如下：

```
DELETE
  [ FROM ] { table_name WITH ( < table_hint_limited > [ ...n ] )
     | view_name
     | rowset_function_limited
    }
[ FROM { < table_source > } [ ,...n ] ]
  [ WHERE
    { < search_condition >
   }]
```

FROM 是可选的关键字，可用在 DELETE 关键字与目标 table_name、view_name 或 rowset_function_limited 之间。

table_name 是要从其中删除行的表的名称。

view_name 是视图名称。在视图的 FROM 子句中，view_name 引用的视图必须可更新且正确引用一个基表。

例如从 student 表中删除年龄为 25 的学生记录，代码如下：

```
DELETE FROM Student WHERE Sage=25
```

如果用户只想清除某个表中的所有数据，但不删除表结构，可以使用 TRUNCATE TABLE 语句。例如要删除 student 表中所有数据，代码如下：

```
TRUNCATE TABLE student
```

也可以用如下代码：

```
delete from student
```

4.5.3 修改和更新记录

可以使用 UPDATE 语句修改和更新表中已经存在的数据。UPDATE 语句既可以一次修改一条记录，也可以一次修改多条记录，甚至可以一次修改表中的全部数据行。

在 UPDATE 语句中使用 WHERE 子句可以指定要修改的行，使用 SET 子句给出新的数据。其语法格式如下：

```
UPDATE
    {table_name WITH ( < table_hint_limited > [ ...n ] )}
    SET{ column_name = { expression | DEFAULT | NULL }
WHERE < search_condition >
```

table_name 是需要更新的表的名称。如果该表不在当前服务器或数据库中，或不为当前用户所有，这个名称可用链接服务器、数据库和所有者名称来限定。

WITH (< table_hint_limited > [...n])指定目标表所允许的一个或多个表提示。需要有 WITH 关键字和圆括号。

SET 指定要更新的列或变量名称的列表。

column_name 含有要更改数据的列的名称。column_name 必须驻留于 UPDATE 子句中所指定的表或视图中。标识列不能进行更新。

Expression 是变量、字面值、表达式或加上括号的返回单个值的子 SELECT 语句。expression 返回的值将替换 column_name 中的现有值。

DEFAULT 指定使用对列定义的默认值替换列中的现有值。如果该列没有默认值并且定义为允许空值，这也可用来将列更改为 NULL。

例如把身份证号为 110105197405060012 的学生的贷款金额修改为 20 000，代码如下：

```
UPDATE Loan SET amount=20000
WHERE Loan_number=
(SELECT Loan_number FROM Borrower
```

```
WHERE ID_Card='110105197405060012')
```

4.5.4 按条件查询数据

可以使用 SELECT 语句来进行数据的查询。SELECT 语句只有三个关键字：SELECT、FROM 和 WHERE。但是 SELECT 语句中有很多子句，能够完成非常复杂的查询功能。SELECT 语句常用的语法如下：

```
SELECT select_list
[ INTO new_table ]
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

因为 SELECT 语句应用情况非常的复杂，下面以多个实例来演示实际中的应用，实例采用 SQL Server 2000 自带的 pubs 数据库。

1. 使用 SELECT 语句查询所有行和列

从 pubs 数据库内的 authors 表中返回所有行(没有指定 WHERE 子句)和所有列(使用*)，如图 4.2 所示。

2. 使用 SELECT 语句查询指定列

从 pubs 数据库内的 authors 表中返回所有行(没有指定 WHERE 子句)和指定列的一个子集(au_lname、au_fname、phone、city、state)。另外，还改变了列标题(列别名)。如图 4.3 所示。

3. 使用 DISTINCT 语句防止查询重复

使用 DISTINCT 语句防止检索重复的作者 ID 号，如图 4.4 所示。

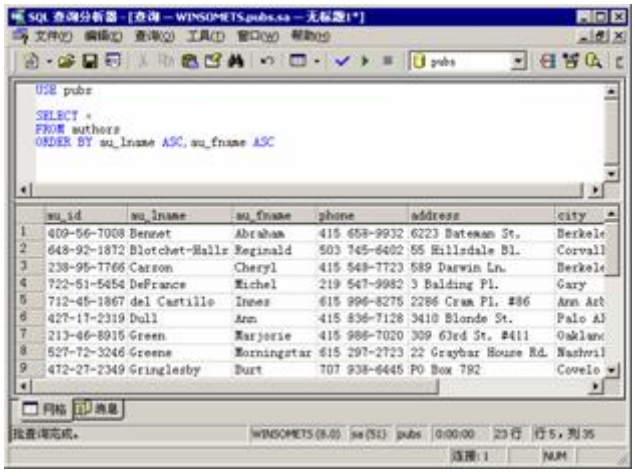


图 4.2 查询所有的数据

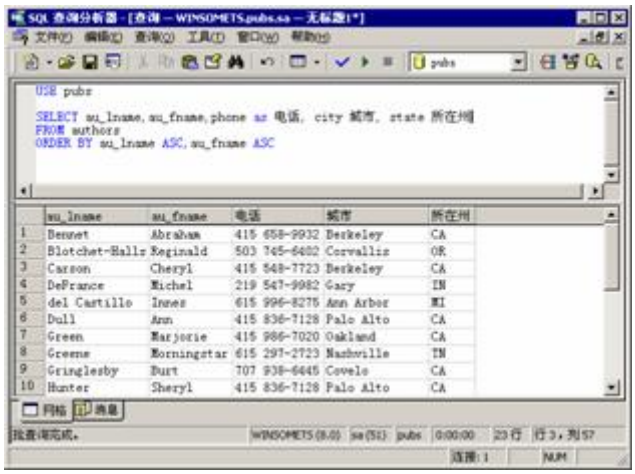


图 4.3 查询指定列

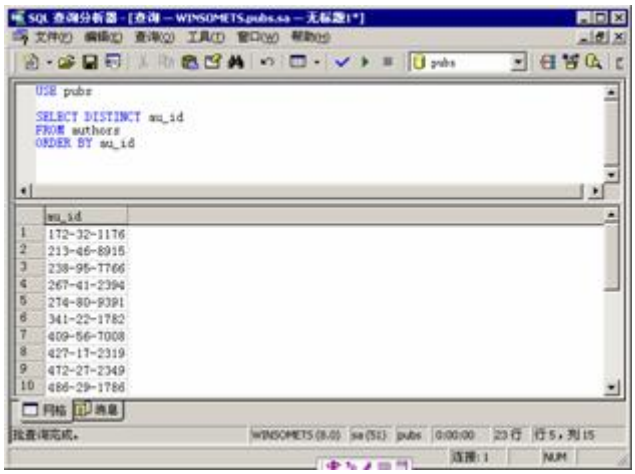


图 4.4 DISTINCT 防止检索重复

4. 带查询条件的查询

查找价格超过$20 的所有类型书籍的价格和预付款，如图 4.5 所示。

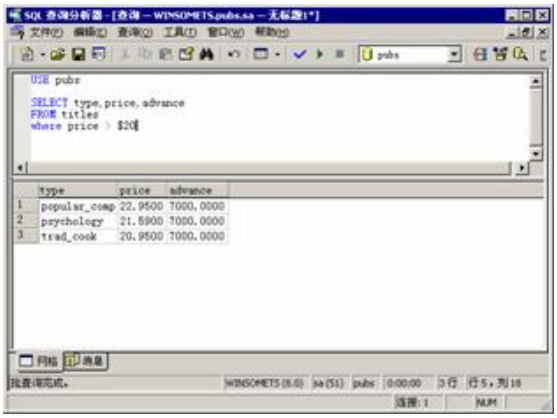


图 4.5 带查询条件的查询

5. 使用通配符

通配符一般是通过 LIKE 使用的。SQL Server 2000 中支持四种通配符，如表 4.7 所示。

表 4.7 通配符及其功能

运 算 符	含 义
%	代表零个或多个任意字符
_	代表一个任意字符
[ ]	指定范围内的任意单个字符
[^]	不在指定范围内的任意单个字符

例如，“AB%”表示以 AB 开始的任意字符串；“_cd”表示以 cd 结尾的三个字符的字符串；“[ef]%”表示以 e 或 f 开始的任意字符串；“[s-v]ing”表示开始是 s 到 v，结尾是 ing，长度为四个字符的字符串；“m[^c]%”表示以 m 开始且第二个字符不是 c 的任意字符串。

例如，使用通配符查询书的类型末尾有“cook”的书的信息，如图 4.6 所示。

6. 使用子查询

SELECT 语句可以嵌套在其他许多语句中，这些嵌套的语句称为子查询。子查询是一个 SELECT 查询，它返回单个值且嵌套在 SELECT、INSERT、UPDATE、DELETE 语句或其他子查询中。任何允许使用表达式的地方都可以使用子查询。子查询也称为内部查询或内部选择，而包含子查询的语句也称为外部查询或外部选择。当一个查询依赖于另外一个查询结果时，就可以使用子查询。

例如使用子查询检索书名为商业书籍的每个出版商名称，并且 titles 表和 publishers 表之间的出版商 ID 号要相匹配，如图 4.7 所示。

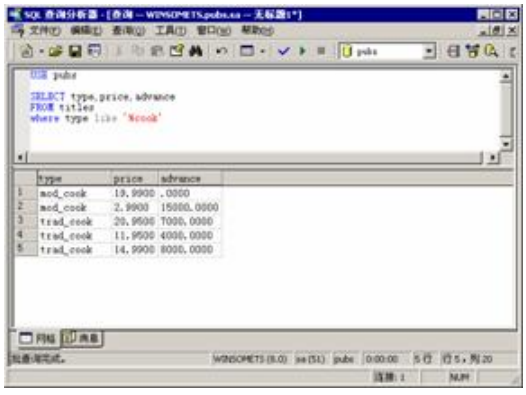


图 4.6 带通配符的查询

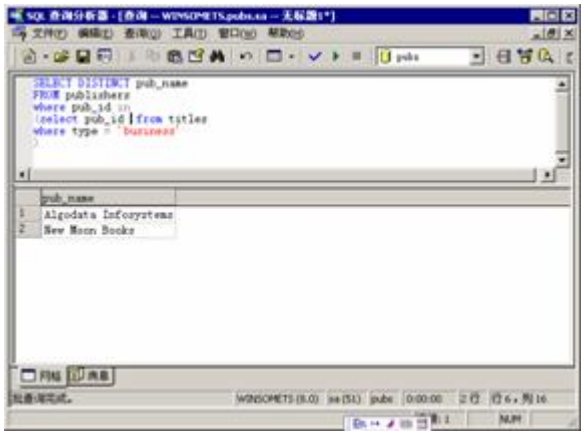


图 4.7 使用子查询

当然，上面的功能还可以通过不同的代码来实现，分别如图 4.8 和图 4.9 所示，其原理都是一样的。

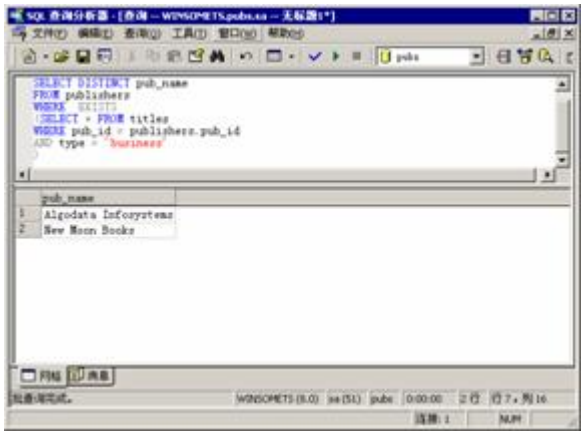


图 4.8 子查询的其他实现方法

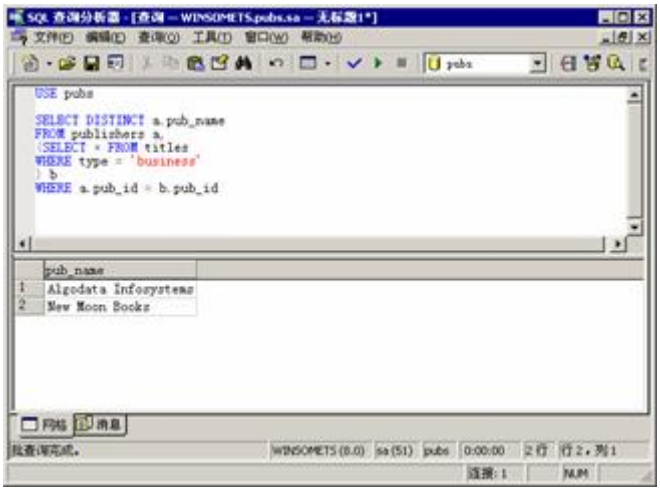


图 4.9 子查询的其他实现方法

相关子查询还可以用于外部查询的 HAVING 子句。例如要查找那些预付款最大金额是组平均值两倍以上的书籍类型，如图 4.10 所示。

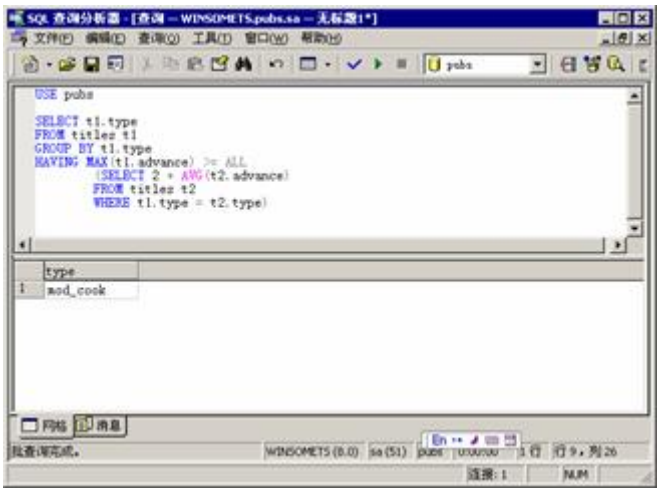


图 4.10　带 HAVING 子句的查询

4.5.5　数据连接多表查询

在实际的数据库操作中，往往需要同时从两个或两个以上的表中查询相关数据，连接就是满足这些需求的技术。使用连接技术查询时，其简单语法格式如下：

```
SELECT table_name.column_name,table_name.column_name,…
FROM table_name[join_type] JOIN table_name
ON search_conditions
WHERE search_conditions
```

在进行表之间的连接时，可以使用三种连接关键字。

- INNER JOIN 关键字，表示内连接，结果集中仅包含满足条件的行。
- CROSS JOIN 关键字，表示交叉连接，结果集中包含两个表中所有行的组合。
- OUTER JOIN 关键字，表示外连接，结果集中既包含那些满足条件的行，还包含某些不满足条件的数据。

例如，要查询在同一个城市的作者和出版商的信息，如图 4.11 所示。

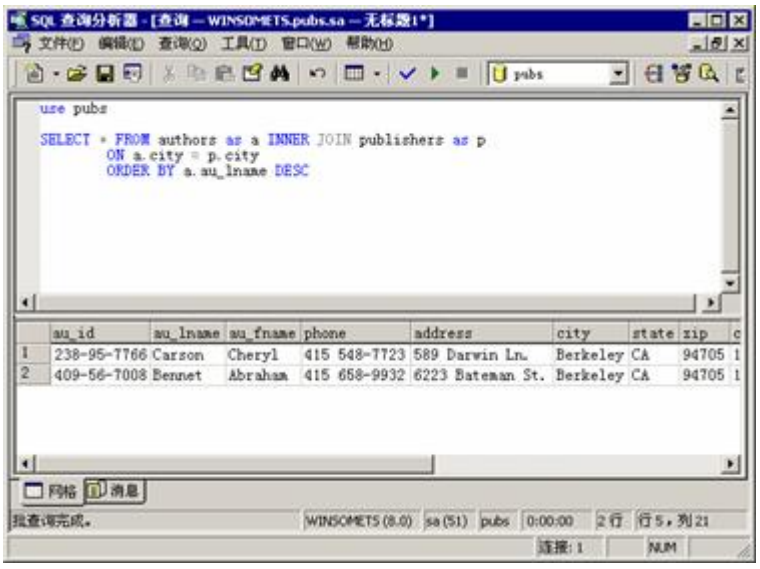


图 4.11　INNER JOIN 连接查询

INNER JOIN 连接查询可以采用另外一种语法实现，如图 4.12 所示，查询结果是一致的。

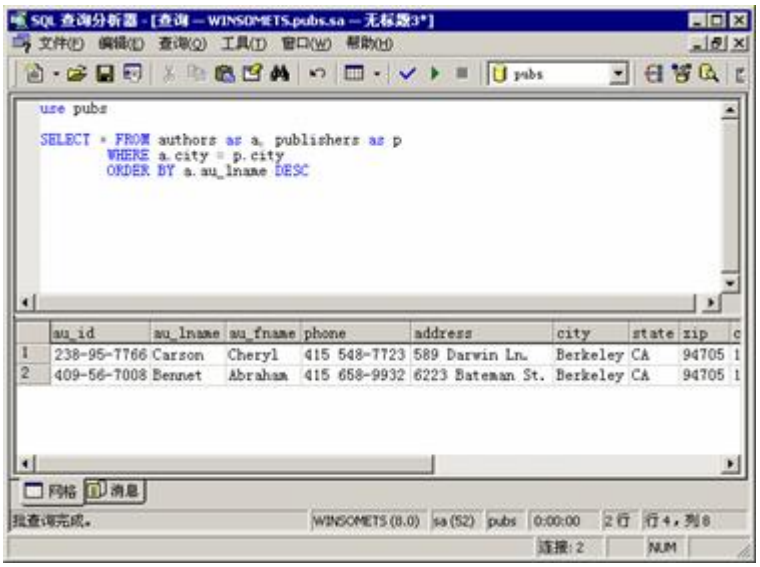


图 4.12　INNER JOIN 连接查询的另外一种写法

内连接是保证两个表中所有的行都要满足连接条件，而外连接返回 FROM 子句中提到的至少一个表的所有行，只要这些行符合任何 WHERE 或 HAVING 搜索条件。外连接包括以下三种形式：

- LEFT OUTER JOIN，左外连接，包括了左表中的全部行。
- RIGHT OUTER JOIN，右外连接，包括了右表中的全部行。
- FULL OUTER JOIN，完整外连接，包括了左表和右表中所有不满足条件的行。

将 authors 作为左表，publishers 作为右表，两表做左外连接，左表 authors 中的全部数据都显示出来了，就是查询结果的记录数和 authors 表的记录数一致。如果 publishers 表中有数据不满足连接条件的行，那么对应右表的数据是空值，如图 4.13 所示。

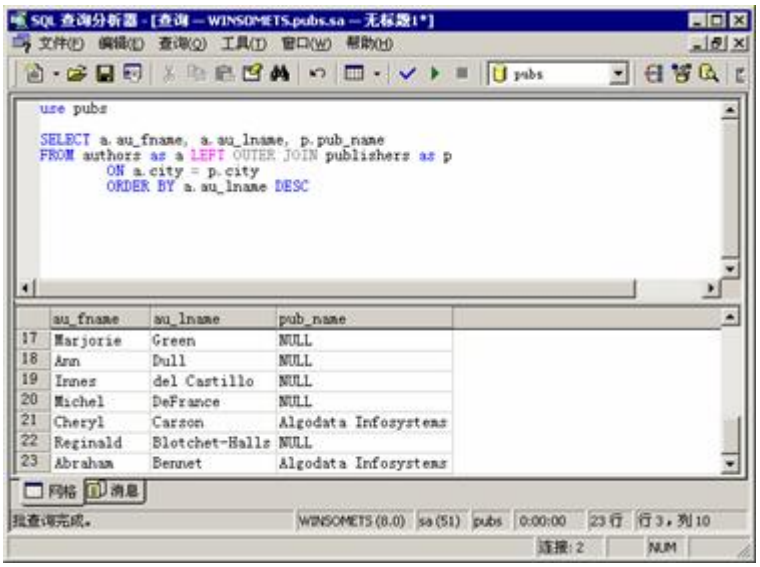


图 4.13　LEFT OUTER JOIN 左外连接查询

外连接查询的 OUTER 关键字可以省略，如图 4.14 所示，效果是一致的。

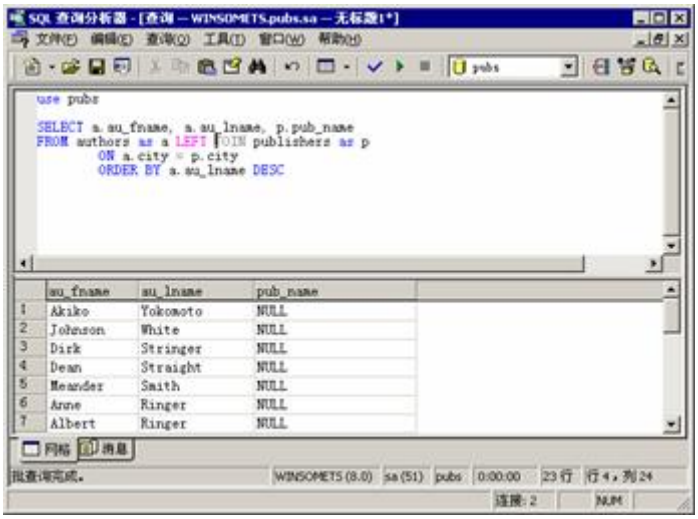


图 4.14　省略了 OUTER 关键字

4.5.6　对查询结果排序

排序技术就是使用 ORDER BY 子句排列查询结果的顺序。ORDER BY 子句按查询结果中的一列或多列对查询结果进行排序，用作排序依据的列总长度可达 8 060。其语法格式如下：

```
SELECT column_name,column_name,…
FROM table_name
ORDER BY column_name[ASC|DESC],…
```

在 ORDER BY 子句中，既可以使用列名，也可以使用相对列号。排序可以是升序的 (ASC)，也可以是降序的 (DESC)。如果没有特别指定，默认为 ASC。

例如，在 titles 表中查询 pub_id、type、title_id 三列的数据，并将查询结果按 pub_id 升序排列，如图 4.15 所示。

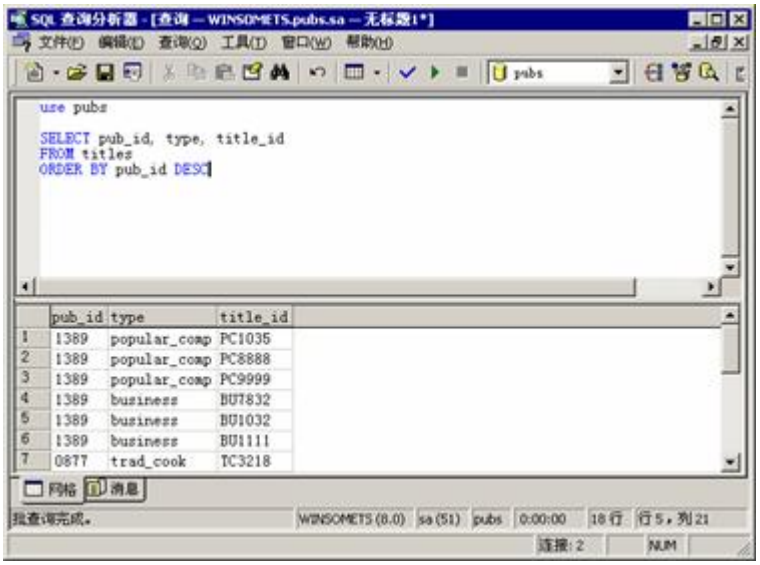


图 4.15　将查询结果排序

4.5.7　数据统计分组查询

在实际应用中，往往需要对表中的原始数据做一些数学处理。统计函数就是满足这些需求的最好工具。常用的统计函数如 AVG、SUM、COUNT、MAX 和 MIN 等，都必须用在 SELECT 语句中，且伴随 GROUP BY 关键字一起使用。

例如，按照图书统计截止到现在的销售情况，并按照销售数量由大到小排序，如图 4.16 所示。

需要注意得是，如果要在查询条件里使用统计函数，则需要使用 HAVING 关键字代替 WHERE 关键字，例如，在上面的查询中如果只想看总销售数量超过 30 册的图书，则代码与结果如图 4.17 所示。

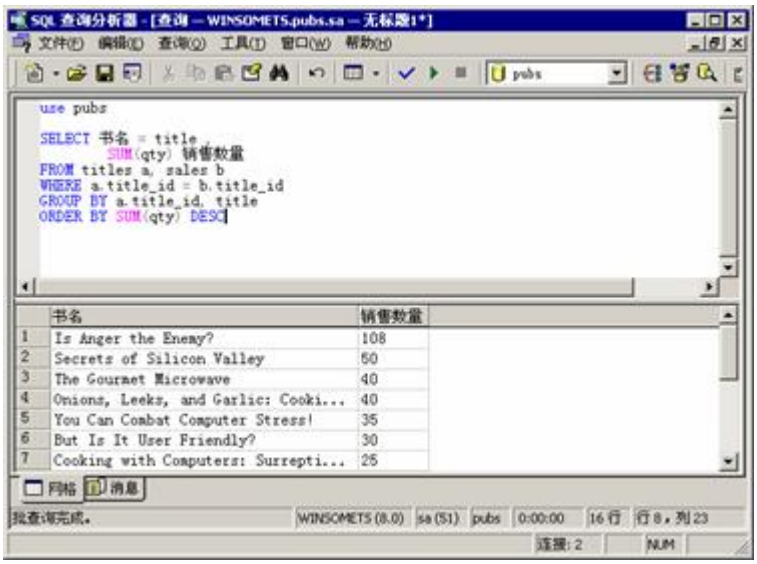


图 4.16 按照图书分组统计销售

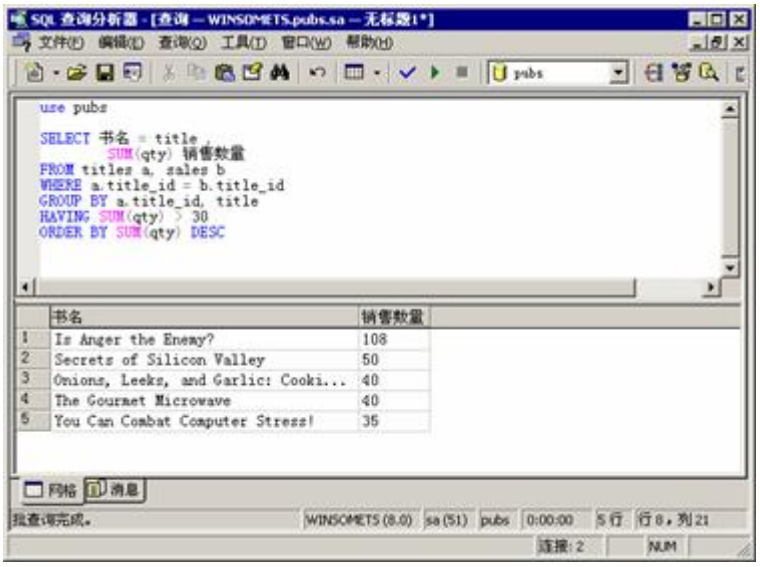


图 4.17 在查询条件中使用统计函数

4.6 游标

数据库的游标是类似于 C 语言指针一样的语言结构。通常情况下，数据库执行的大多数 SQL 命 令都是同时处理集合内部的所有数据。但是，有时侯用户也需要对这些数据集合中的每一行进行操作。在没有游标的情况下，这种工作不得不放到数据库前端，用高 级语言来实现。这将导致不必要的数据传输，从而延长执行的时间。通过使用游标，可以在服务器端有效地解决这个问题。游标提供了一种在服务器内部处理结果集 的方法，它可以识别一个数据集合内部指定的工作行，从而可以有选择地按行采取操作。

游标的功能比较复杂，要灵活使用游标需要花费较长的时间练习和积累经验。本节只介绍使用游标最基本和最常用的方法。如果想进一步地学习，可以参考数据库的相关 书籍。

4.6.1 声明游标

在使用游标之前首先要声明游标，Transact-SQL 在 ANSI 92 SQL 的基础上对游标的功能作了一

定的扩充，因此产生了与 ANSI 92 SQL 稍有不同的游标声明方法。

声明游标的语法如下：

```
DECLARE 游标名 [INSENSITIVE] [SCROLL] CURSOR
FOR SELECT 语句
[FOR READ ONLY | UPDATE [OF 列名 1，列名 2，列名 3 …]
```

游标名为声明的游标所取的名字，声明游标必须遵守 Transact-SQL 对标识符的命名规则。

使用 INSENSITIVE 定义的游标，把提取出来的数据放入一个在 tempdb 数据库创建的临时表里。任何通过这个游标进行的操作，都在这个临时表里进行。所以所有对基本表的改动都不会在用游标进行的操作中体现出来。如果忽略了 INSENSITIVE 关键字，那么用户对基本表所做的任何操作，都将在游标中得到体现。

使用 SCROLL 关键字定义的游标，具有包括如下所示的所有取数功能：

- FIRST 取第一行数据。
- LAST 取最后一行数据。
- PRIOR 取前一行数据。
- NEXT 取后一行数据。
- RELATIVE 按相对位置取数据。
- ABSOLUTE 按绝对位置取数据。

如果没有在声明时使用 SCROLL 关键字，那么所声明的游标只具有默认的 NEXT 功能。

READ ONLY 声明只读光标，不允许通过只读光标进行数据的更新。

UPDATE [OF 列名 1，列名 2，列名 3 …]定义在这个游标里可以更新的列。如果定义了[OF 列名 1，列名 2，列名 3 …]，那么只有列在表中的列可以被更新；如果没有定义[OF 列名 1，列名 2，列名 3 …]，那么游标里的所有列都可以被更新。

下面是两个声明游标的例子：

```
--这个游标可以在整个 authors 表中所有的数据行上进行操作
DECLARE authors_cursor CURSOR
  FOR SELECT *
  FROM authors

--这个例子声明了一个只读游标，并对游标可以处理的结果集进行了筛选和排序
DECLARE authors_cursor CURSOR FOR
SELECT au_id, au_fname, au_lname
FROM authors
WHERE state = "UT"
ORDER BY au_id
FOR READONLY
```

4.6.2 打开游标

在可以使用游标之前，必须首先打开游标。打开游标的语法如下：

```
OPEN cursor_name
```

当执行打开游标的语句时，服务器执行声明游标时使用的 SELECT 语句，如果使用了 INSENSITIVE 关键字，则服务器会在 tempdb 中建立一张临时表，以存放游标将要操作的数据集的副本。

在打开游标后，可以使用@@CURSOR_ROWS 全局变量来返回当前声明的游标可以操作数据行的数量。

4.6.3 关闭游标

在打开游标以后，SQL Server 服务器会专门为游标开辟一定的内存空间存放游标操作的数据结果集，同时游标的使用也会根据具体情况对某些数据进行封锁。所以，在不使用游标的时候，一定要关闭游标，以通知服务器释放游标所占用的资源。

关闭游标的语法如下：

```
CLOSE cursor_name
```

关闭游标以后，可以再次打开游标，在一个批处理中，也可以多次打开和关闭游标。

4.6.4 释放游标

游标结构本身也会占用一定的计算机资源，所以在使用完游标后，为了回收被游标占用的资源，应该将游标释放。

释放游标的语法是：

```
DEALLOCATE cursor_name
```

当释放完游标以后，如果要重新使用这个游标必须重新执行声明游标的语句。

4.6.5 使用游标取数

在打开游标以后，就可以打开游标提取数据了。使用游标提取某一行的数据应该使用下面的语法：

```
FETCH
   [ [  NEXT | PRIOR | FIRST | LAST
      | ABSOLUTE {n | @nvar}
      | RELATIVE {n | @nvar}
     ]
     FROM
   ]
    cursor_name
   [INTO @variable[,...n]]
```

在这个语法里，n 和@nvar 表示游标相对与作为基准的数据行所偏离的位置。

在使用 INTO 子句对变量赋值时，变量的数量和相应的数据类型必须和声明游标时使用的 SELECT 语句中引用到的数据列的数目、排列顺序和数据类型完全保持一致，否则服务器会提示错误。

事实上，使用游标取数的操作通常与 WHILE 循环紧密结合在一起。下面的代码演示了游标的使用方法：

```
USE pubs
GO

/*声明游标*/
DECLARE authors_cur CURSOR FOR
SELECT au_lname,au_fname FROM authors
WHERE state = "CA"
ORDER BY au_lname

/*打开一个游标*/
OPEN authors_cur

/*执行第一次取数操作*/
FETCH NEXT FROM authors_cur

/*检查@@FETCH_STATUS 以确定是否还可以继续取数*/
WHILE @@FETCH_STATUS = 0
BEGIN
  FETCH NEXT FROM authors_cur
END

/*关闭游标*/
CLOSE authors_cur
```

当游标移动到最后一行数据的时候，继续执行取下一行数据的操作，将返回错误信息，但这个信息只在@@FETCH_STATUS 中体现，同时返回空白的数据，根据判断条件，程序现在就终止循环。

下面的例子定义了一个滚动游标，从而可以实现更灵活的数据提取。

```
USE pubs
GO

/*首先执行一遍查询语句以提供滚动游标操作成功与否的对比*/
SELECT au_lname, au_fname
FROM authors
WHERE state = "CA"
ORDER BY au_lname, au_fname

-- 定义滚动游标
DECLARE authors_cur SCROLL CURSOR FOR
SELECT au_lname, au_fname FROM authors
WHERE state = "CA"
ORDER BY au_lname, au_fname

/*打开游标*/
OPEN authors_cur
```

```
/*提取数据集中的最后一行*/
FETCH LAST FROM authors_cur

/*提取当前游标所在行的上一行*/
FETCH PRIOR FROM authors_cur

/* 提取数据集中的第 4 行*/
FETCH ABSOLUTE 4 FROM authors_cur

/*提取当前行的前两行*/
FETCH RELATIVE -2 FROM authors_cur

/*关闭游标*/
CLOSE authors_cur

/*释放游标*/
DEALLOCATE authors_cur
GO
```

程序执行结果如下：

```
au_lname                au_fname
---------------------------------------- --------------------
Bennet                  Abraham
Carson                  Cheryl
Dull                   Ann
…
Yokomoto                Akiko

(所影响的行数为 16 行)

au_lname                au_fname
---------------------------------------- --------------------
Yokomoto                Akiko

(所影响的行数为 1 行)

…

(所影响的行数为 1 行)
```

下面的程序将所有取到的数据存放在变量里，再打印出来：

```
USE pubs
GO
/*定义变量*/
```

```
DECLARE @au_lname varchar(40), @au_fname varchar(20)

/*声明游标*/
DECLARE authors_cur CURSOR FOR
SELECT au_lname, au_fname FROM authors
WHERE state = "CA"
ORDER BY au_lname, au_fname

/*打开游标*/
OPEN authors_cur

/*执行第一次提取数据操作*/
FETCH NEXT FROM authors_cur
INTO @au_lname, @au_fname

/*检查上一次操作的执行状态*/
WHILE @@FETCH_STATUS = 0
BEGIN
  PRINT "作者:  " + @au_fname + " " +  @au_lname
  FETCH NEXT FROM authors_cur
  INTO @au_lname, @au_fname
END

/*关闭并释放游标*/
CLOSE authors_cur
DEALLOCATE authors_cur
GO
```

4.6.6 利用游标修改数据

要使用游标进行数据的修改，其前提条件是该游标必须被声明为可更新的游标。在进行游标声明时，没有带 READONLY 关键字的游标都是可更新的游标。

在游标声明过程中可以使用 SELECT 语 句对多个表中的数据进行访问，因此如果声明的是可更新游标，那么可以使用该游标对多表中的数据进行修改，但是这不是一个更改数据的好办法，因为这种不规范 更新数据的途径很容易造成数据的不一致。在计算机编程过程中，经常会遇到这样的情况，具有充分灵活性的语法总是难于操纵，易于出错。

使用游标更新数据的常用语法如下：

```
UPDATE table_name
{SET  column_name = expression}
[,...n]
WHERE  CURRENT OF  cursor_name
```

其中，CURRENT OF cursor_name 表示当前游标的当前数据行。CURRENT OF 子句只能使用在进行 UPDATE 和 DELETE 操作的语句中。

下面的例子声明了一个可更新的游标，并限定了可以更新的列，然后针对该列进行了更新运算。

```
USE pubs
GO

/*定义一个对价格可以进行更改的滚动游标*/
DECLARE a_ta_t CURSOR SCROLL FOR
SELECT au_lname , title ,price
FROM  authors a, titleauthor ta, titles t
WHERE  a.au_id   = ta.au_id
AND   ta.title_id = t.title_id
AND   a.state   = "CA"
FOR  UPDATE OF price

DECLARE  @fetch_status INT
/*打开一个游标*/
OPEN a_ta_t

/*取第 1 行数据*/
FETCH a_ta_t
SELECT @fetch_status = @@FETCH_STATUS

WHILE  @fetch_status = 0
BEGIN
UPDATE titles
SET   price = price / 1.5
    WHERE CURRENT OF a_ta_t

/*继续取下一数*/
FETCH a_ta_t
SELECT @fetch_status = @@FETCH_STATUS
END

/*关闭并释放游标*/
CLOSE    a_ta_t
DEALLOCATE a_ta_t
GO

/*再次取数进行验证*/
SELECT au_lname , title ,price
FROM  authors a, titleauthor ta, titles t
WHERE  a.au_id   = ta.au_id
AND   ta.title_id = t.title_id
AND   a.state   = "CA"
GO
```

这个例子查找来自加利福尼亚州的作者，并把他的书的价格减了价。由于这次声明的游标是用了

UPDATE OF 子句，所以只有包含在这个子句列表中的数据行可以被更新。

使用游标还可以进行数据的删除，语法是：

```
DELETE
FROM  table_name
WHERE  CURRENT OF  cursor_name
```

在使用游标进行数据的更新或删除之前，必须事先获得相应数据库对象的更新或删除的特权，这是进行这类操作的前提。

4.7 事务

SQL Server 作为典型的关系数据库，为事务控制提供了完善的编程结构。在 Transact-SQL 中，事务处理控制语句有以下 4 个：

- BEGIN TRAN[SACTION] [transaction_name]
- COMMIT [TRAN[SACTION]] [transaction_name]
- ROLLBACK [TRAN[SACTION]][transaction_name]
- SAVE TRAN[SACTION] [savepoint_name]

其对应的功能分别是开始、提交、回滚和保存事务。

保存点允许在一个事务处理内部做一些工作，在特定的条件下回滚这些工作。当回滚到保存点后，只有在保存点到回滚语句之间的操作被取消，其他的操作依然有效，而且，程序会接着从回滚的断点执行下去。

下面这个例子，定义了一个简单的事务：

```
BEGIN TRANSACTION
USE pubs
GO

UPDATE titles
SET advance = advance * 1.25
WHERE ytd_sales > 8000
GO

DELETE authors
WHERE  state = "MA"

COMMIT TRANSACTION
GO
```

从 BEGIN TRANSACTION 到 COMMIT TRANSACTION 只有两个操作，按照事务的定义，这两个操作要么都执行成功，要么都不执行。

下面这个例子中使用到了保存点：

```
--开始一个事务
BEGIN TRANSACTION exampletrans
 USE pubs
```

```
GO

--执行一次更新操作
  UPDATE titleauthor
   SET  royaltyper = 35
   FROM titleauthor, titles
   WHERE royaltyper = 25
    AND titleauthor.title_id = titles.title_id
    AND title = 'The Gourmet Microwave'
 GO

--设置保存点
SAVE TRANSACTION percentchange
--第二次更新操作
UPDATE titles
  SET price = price * 1.2
  WHERE title = 'The Gourmet Microwave'
  GO

  --回滚到保存点
ROLLBACK TRANSACTION percentchanged
PRINT "程序继续执行"
COMMIT TRANSACTION
```

这个例子一共执行了两次更新操作，第一次操作完成后，设置了一个保存点；第二次更新操作执行完后，程序执行了事务回滚，使得第二次更新操作被取消，但是第一次更新操作依然有效，程序继续从回滚处执行，打印出一行字。

4.8 常用函数

函数是一段特殊的程序代码，它能对查询结果进行一定的操作。函数的作用就是使用户不必书写太多的程序代码即可完成复杂的操作。

SQL Server 2000 提供了多种功能强大的函数，包括算术函数、字符串函数、系统函数、日期时间函数和文本图像函数等。由于系统函数非常多，下面简单列举常用的一些函数，具体可以在使用过程中参考 SQL Server 2000 的联机帮助。

4.8.1 日期和时间函数

SQL Server 2000 提供的日期和时间函数有如下几种。

- DATEADD：在向指定日期加上一段时间的基础上，返回新的日期值。
- DATEDIFF：返回跨两个指定日期的日期和时间边界数。
- DATENAME：返回代表指定日期的指定日期部分的字符串。
- DATEPART：返回代表指定日期的指定日期部分的整数。
- DAY：返回代表指定日期的天的日期部分的整数。
- GETDATE：返回当前系统日期和时间。
- GETUTCDATE：返回表示当前 UTC 时间(世界时间坐标或格林尼治标准时间)的值。
- MONTH：返回代表指定日期月份的整数。
- YEAR：返回表示指定日期中的年份的整数。

列出查询图书销售日期和当前日期的间隔，如图 4.18 所示。

图 4.18　日期函数的使用

4.8.2　聚合函数

聚合函数对一组值执行计算并返回单一的值，经常与 SELECT 语句的 GROUP BY 子句一同使用。主要的聚合函数有以下几种。

- AVG：取平均值。
- MAX：取最大值。
- MIN：取最小值。
- SUM：求和函数。
- COUNT：返回组中项目的数量。

关于聚合函数的使用方法在 4.5.7 小节已经介绍。

4.8.3　字符串函数

字符串函数对字符串输入值执行操作，返回字符串或数字值。

- ASCII：返回字符表达式最左端字符的 ASCII 代码值。
- CHAR：将 ASCII 代码转换为字符的字符串函数。
- SPACE：返回由重复的空格组成的字符串。
- REPLACE:用第 3 个表达式替换第一个字符串表达式中出现的所有第 2 个给定字符串表达式。
- STR：由数字数据转换来的字符数据。
- LEFT：返回从字符串左边开始指定个数的字符。
- SUBSTRING：返回字符串表达式的一部分。
- LEN：返回给定字符串表达式的字符(而不是字节)个数，其中不包含尾随空格。
- REVERSE：返回字符表达式的反转。
- LOWER：将大写字符数据转换为小写字符数据后返回字符表达式。
- RIGHT：返回字符串中从右边开始指定个数的字符。
- UPPER：返回将小写字符数据转换为大写的字符表达式。
- LTRIM：删除起始空格后返回字符表达式。
- RTRIM：截断所有尾随空格后返回一个字符串。

4.8.4　系统统计函数

系统统计函数返回系统的统计信息。

- @@CONNECTIONS：返回自上次启动 SQL Server 以来连接或试图连接的次数。
- @@CPU_BUSY：返回自上次启动 SQL Server 以来 CPU 的工作时间，单位为毫秒(基于系统计时器的分辨率)。
- @@TIMETICKS：返回一刻度的微秒数。

- @@IDLE：返回 SQL Server 自上次启动后闲置的时间，单位为毫秒(ms，基于系统计时器的分辨率)。
  - @@TOTAL_ERRORS：返回 SQL Server 自上次启动后，所遇到的磁盘读/写错误数。
- @@TOTAL_READ：返回 SQL Server 自上次启动后读取磁盘(不是读取高速缓存)的次数。
- @@TOTAL_WRITE：返回 SQL Server 自上次启动后写入磁盘的次数。

例如，显示自 SQL Server 启动后到当前日期和时间为止 SQL Server CPU 的工作时间，如图 4.19 所示。

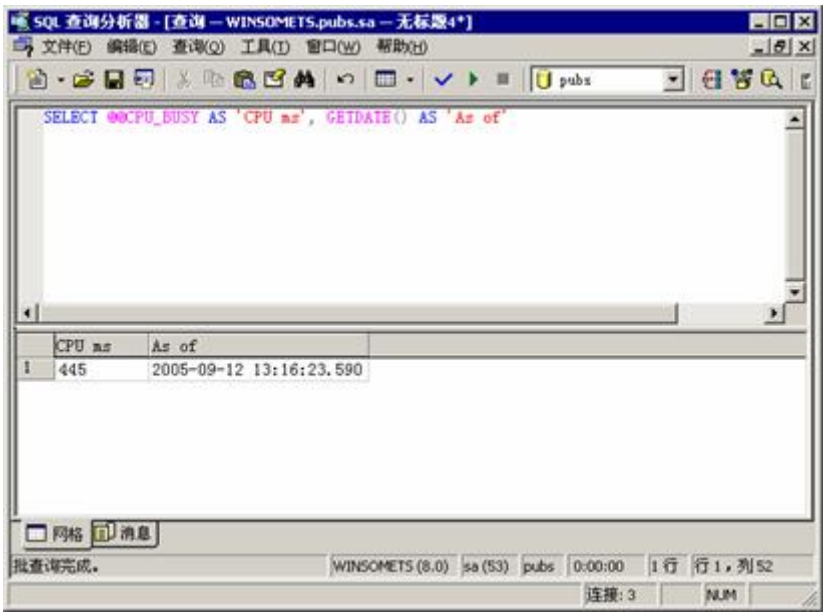


图 4.19　系统函数的使用

第 5 章 用 Delphi 完成进销存管理系统

电子商务的发展和市场竞争的加剧将企业推上了风口浪尖，中小企业除了积极迎接挑战之外，别无选择。网络的兴起与电子商务的发展带来了时空界限的突破、贸易方 式的变革和经济活动的革命，从某种意义上来讲，这些变化为中小企业创造了与大型企业、国外企业平等竞争的有利条件。然而，管理水平的差异会弱化这种平等的 实际意义。具体而言，中小企业在进、销、存等环节的管理上都存在着一定程度的不足，而这些不足无疑将使其在市场竞争中陷入被动的境地。在网络经济的时代背景下，进行有效的进销存管理已经成为中小企业存身立业的头等要事。然而对于一贯疏于管理的中小企业而言，实现有效的进销存管理必然存在着一定的难度，因此 必须借助现代化的管理方法和管理技术——计算机进销存管理系统。

本章将向用户全面剖析进销存管理的内容，并由此得出通用进销存管理系统的需求分析和数据建模，并最终演示如何利用 Delphi 完成系统的制作。同时还向用户介绍特殊行业进销存系统(如医药行业、服装行业等)与通用进销存系统的区别，从而使用户可以迅速掌握这些行业进销存系统的实现方法。

5.1　进销存管理

在市场经济中，销售是企业运作的重要环节， 为了更好地推动销售，不少企业建立分公司或代理制，通过分公司或代理把产品推向最终用户。这些分公司或代理商大多分布在全国各地，甚至是在国外，远距离频 繁的业务信息交流构成了这些企业业务活动的主要特点。在传统方式上，公司之间通常采用电传、电报、电话等方式传递订货、发货、到货、压货、换货、退货等信 息，总公司的商务部门在接到分公司或代理商传来的订单和银行汇款单据传真件后，开具产品出库通知，然后把相关

的进、销、存信息手工存档，再对这些信息进行 统计分析，才能了解到整个公司的生产、销售和库存情况。

这种信息传递和管理的方式不仅效率低，可靠 性、安全性和保密性都无法满足要求，而且数据统计时间严重滞后，往往是当领导了解到企业的“进、销、存”环节出现问题时，就已经远离了问题出现的时间和地 点。即便是没有分公司的企业，使用传统的手工方式管理也存在同样的问题。通过进销存管理系统，及时通过网络把决策信息传递给相关决策人，从而可以及时发现 问题、解决问题，从而更好地把握机会。

5.1.1　进销存管理的任务

进销存管理工作的主要任务有：

- 企业的采购管理
- 企业的销售管理(批发、零售、连锁)
- 企业各部门的商品配送管理
- 企业库存商品的管理
- 企业应收款、应付款的管理
- 企业经营状况分析与决策

由于企业经营的商品种类经常会很多，例如一个超市可能会经营上万种商品，因此按照传统的人工管理方式，会存在以下问题。

(1)各种物资种类繁多、品种齐全，保管管理困难。

有的物资超储备或库存积压现象严重，采购成本居高不下；有的物资严重短缺，导致销售时才发现没有库存；有的商品由于没有及时盘点，到月末时才发现库存缺失，却查不出原因；有的在入库、领用时找不到商品的存放地点等。

(2)无法进行准确及时的成本、毛利核算。

成本管理永远是企业管理的主题，特别是现在面对多变的市场环境，如何及时满足用户的多品种需求，进行科学合理的成本预测、成本分析及成本控制，及时、准确地为企业管理者提供经营决策信息，越来越显得至关重要。传统的成本核算方法(加权平均法和移动平均法)及核算工具只是粗放地进行成本核算及成本管理，很难满足管理的需要。至于成本核算到工序、核算到产品的思路(先进先出法)，在手工操作方式下更是无从谈起。

(3)管理信息相互独立，市场预测手段、方法落后，严重影响企业科学决策，建立科学的市场信息反馈系统已迫在眉睫。

手工财务数据相互独立，财务信息传递也只是 通过逐层地统计报表来完成的，因此常常出现数字不符、报表不详的情况，难以满足统计数据的及时性、准确性、相关性要求。集团决策层、领导层很难及时地把握 来自市场的准确信息，也就无法快速对市场作出正确的决策和预测。市场反馈信息系统已严重滞后于企业管理的需要。

(4)应收帐款居高不下，占用了大量资金，严重影响了企业资产运作效率，增加了企业的经营风险。

手工管理很难适应企业销售规模增大、销售业务复杂的形势，普遍存在着应收帐款管理不细、帐龄分析和催款通知不及时、不到位的问题；催款力度及相应催款政策不够。

(5)在生产和销售环节，不能准确地掌握质量的变化，尤其一些客户投诉或降级使用的产品，无法做到全程跟踪，从而影响市场的销售。

自动化程度低，信息不畅通，不能在整个企业共享资源，各种等级的成品的管理不能做到准确无误。缺乏质量跟踪、投诉处理监控机制。

(6)不能有效管理异地仓库和办事处销售业务。

办事处一般通过传真或电话的方式每月上报销售数据和异地仓库的收发存数据，工作量大，并且上报的信息存在信息延迟和不准确的现象，严重影响到企业领导层的决策工作。

5.1.2　进销存管理系统的作用

进销存管理系统是基于先进的软件和高速、大 容量的硬件基础上的新型进销存管理模式，通过集中式的信息数据库，将企业的进、销、调、存、转、赚等企业的经营业务有机地结合起来，达到数据共享、降低成 本、提高效率、改进服务等目的。一般来说，利用进销存管理系统可以在以下几方面提高企业管理的水平：

- 提高管理效率降低人工成本。
- 降低采购成本。
- 及时调整营销策略，防止价格流失。
- 防范陈呆死帐，降低应收帐款。
- 减少仓储面积，提高房产综合利用率。
- 降低储备资金占用。
- 加快资金周转实现的经济效益。
- 强化财务监控制实现的经济效益。
- 商业数据智能分析。
- 高效决策。

5.2 进销存管理系统需求分析

根据以上对进销存管理内容和进销存管理系统的分析，一个标准的进销存管理系统应该包括如图 5.1 所示的几大功能。

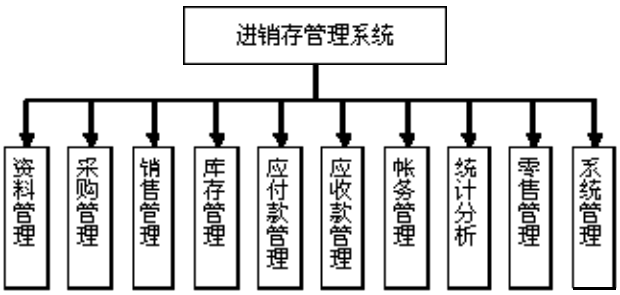


图 5.1 进销存管理系统应包括的基本功能

其中每个功能都由若干相关联的子功能模块组成。下面将对这些模块一一进行介绍。

5.2.1 资料管理

企业经营的基础资料是一个企业最基本、最重要的信息，脱离了基础资料(包括商品资料、供货商资料和客户资料等)，进销存系统就无法运行。“资料管理”功能模块就用于维护这些基础资料，其中所包含的子功能模块如图 5.2 所示。

“商品编码设置”用于设置商品编码的方法。进销存系统的每一条商品信息都具有惟一的一个编码，作为该商品信息的主键，一般来说，该编码具有一定的规律，例如药品进销存系统，商品的编码可能有两级，第一级为药品的类别(中药，西药、保健品、器械等)，第二级为药品的剂型(针剂、片剂、丸剂等)。该功能设置商品编码分类的方法，从而实现商品资料维护中自动生成编码的功能。例如该商品属于西药的丸剂，而西药的编码是 2，丸剂的编码是 02，该商品的编码就是 202****，后面部分由系统根据该类别里现有编码的最大值加 1 自动生成。

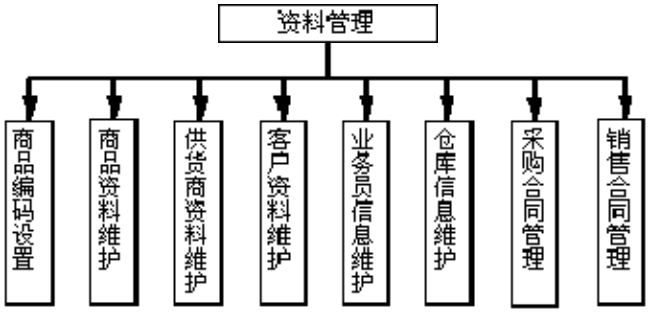


图 5.2 “资料管理”功能模块

“商品资料维护”用于维护(增加、修改、删除、查询)企业经营商品的基本信息，内容包括货号、条形码、商品名、拼音编码、规格、单位、产地、类别、进货价、销售价 1、销售价 2、最低售价等。其中拼音编码是商品名称的拼音简称，目的是使程序可以通过拼音编码方便地查询到所需要的商品。不同行业的进销存商品资料的属性差别很大，我们将在后面加以说明。

“供货商资料维护”用于维护企业供货商的基本信息，内容包括供货商号、拼音编码、简称、名称、地址、邮编、区号、地区、类型、电话、传真、电报、开户行、开户行邮编、银行帐号、税号、库房地址、库房电话、业务员、业务部门等。

“客户资料维护”用于维护企业客户的基本信息，内容包括客户编号、拼音编码、简称、名称、联系人、地址、邮编、区号、地区、电话、传真、电报、开户行、开户行邮编、银行帐号、税号、性质、业务员、业务部门、授信额度等。

“业务员信息维护”用于维护企业销售业务员的基本信息，内容包括业务员号、姓名、性别、电话、手机、地址、邮编、身份证号、类别等。在销售开票时，业务员属性可以直接从业务员清单中获取所有业务员的名字作为辞典供用户选择。

“仓库信息维护”用于维护企业的仓库信息，内容包括仓库号、仓库名、类别、备注等。企业的商品一般都是存放在不同的仓库或同一仓库的不同货位中，这样在填写商品进货单和销售单时用户需要指定入库或出库的是哪个仓库的货物。

“采购合同管理”用于维护企业与供货商签订的采购合同，内容包括供货商号、货号、进价、付款方式、帐期、签订日期、合同期限等。合同的内容对业务可以产生影响，例如合同规定了进价，在填写采购单时系统就会自动生成进价。

“销售合同管理”用于维护企业与客户签订的销售合同，内容包括客户编号、货号、售价、付款方式、帐期、签订日期、合同期限等。

小知识：数据库设计范式

数据库的设计有一定的规范，按照等级划分为 1NF、2NF、3NF、BCNF、4NF 和 5NF 这 6 个范式。这些范式主要用来减少数据库中的数据冗余，每个范式都有自己严格的数学定义，下面我们用通俗的语言解释第一范式和第二范式。1NF 指关系中的每个字数都必须是原子的，即每个字段都是不可再分的原子数据项，例如业务员信息表需要记录业务员的地址和邮编，如果使用字段 Address_Zip，这就是不符合第一范式的，应该分两个字段(Address 和 Zip)存放信息。2NF 定义为：如果关系模式 R(U,F)中的所有非主属性都完全依赖于任意一个候选关键字，则称关系 R 属于第二范式。举例来说，在系统中填写进货单时需要指定进货的商品，如果每次进货都填写进货商品的品名、单位等信息，就会造成很大的数据冗余。根据第二范式的要求，可以把商品的信息提炼出来，单独存放在一张表中，将商品的编码(货号)作为关键字，进货时只需要指定进货商品的货号就可以了，而其他信息可以通过关联的方法从商品表中获取。需要注意的是，并不是数据库设计时满足的范式层次越高数据库就越合理，一般满足第二范式就可以了，过于追求数据的精简反而增加操作的复杂程度。

5.2.2 采购管理

“采购管理”功能模块用于管理企业的采购业务，所包含的子功能模块如图 5.3 所示。

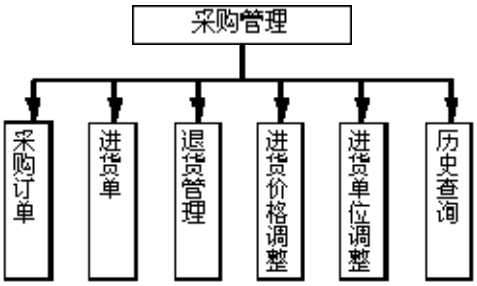


图 5.3 “采购管理”功能模块

“采购订单”用于录入企业的采购订单。一般 来说，采购订单、入库单、销售单等单据根据第二范式都分为主从两张表来存放数据，主表“采购订单”的内容包括编号、供货商号、订货日期、有效起日、有效止 日、业务员、制单人、税价合计、不含税价、税额等，从表“采购订单明细”的内容包括编号、订单号、货号、订货数量、进价、税价合计、扣率、税率、不含税 价、税额等。

“进货单”用于录入企业的进货单，也分为主 从两张表，主表“进货单”内容包括编号、供货商号、进货日期、业务员、制单人、验收员、保管员、税价合计、不含税价、税额、订单号等，从表“进货单明细” 内容包括编号、进货单号、货号、进货数量、进价、税价合计、扣率、税率、不含税价、税额、仓库、货物质量等。根据企业的规模和管理方法，可以直接作进货单 而不用作采购订单，但对于较大规模的企业，采购部门和库房部门一般是分开的，由采购部门填写采购订单，货物运输到库房后由库房验收人员填写进货单。填写进 货单后商品的库存才会增加并产生应付款。

“退货管理”用于录入企业进货时的退货单。进货的退货有两种方法，一是直接在进货单中填写负数的进货数量，另一种是填写进退货单，一般采用前一种方法。

“进货价格调整”用于调整历史进货单的价格，内容包括内部编号、编号、进货数量、原进价、新进价、调整日期、制单人等。

“进货单位调整”用于调整历史进货单的供货商，内容包括内部编号、编号、原供货商、新供货商、调整日期、制单人等。

“历史查询”用于查询商品采购、进货的历史。一般可以让用户按照任何条件查询，如按照供货商、日期、商品货号、商品拼音等。

5.2.3 销售管理

“销售管理”功能模块用于管理企业的销售业务，所包含的子功能模块如图 5.4 所示。

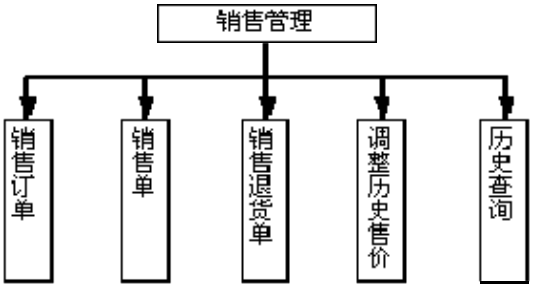


图 5.4 “销售管理”功能模块

“销售订单”用于录入企业的销售订单。主表 “销售订单”的内容包括编号、客户编号、销售日期、有效起日、有效止日、业务员、制单人、税价合计、不含税价、税额等，从表“销售订单明细”的内容包括编 号、订单号、货号、销售数量、销售价、税价合计、扣率、税率、不含税价、税额等。

“销售单”用于录入企业的销售单，主表“销售单”内容包括编号、客户编号、销售日期、业务员、制单人、保管员、税价合计、不含税价、税额、订单号等，从表“销售单明细”内容包括编号、销售单号、货号、销售数量、销售价、税价合计、扣率、税率、不含税价、税额、出货仓库等。

“销售退货单”用于录入企业销售退货单，内容包括编号、销售单编号、货号、退货数量、销售价、税价合计、扣率、税率、不含税价、税额、退货仓库等。

“调整历史售价”用于调整历史销售单的价格，内容包括内部编号、编号、销售数量、原销价、

新销进价、调整日期、制单人等。

“历史查询”用于查询商品销售的历史。一般可以让用户按照任何条件查询，如按照客户、日期、商品货号、商品拼音、业务员等信息查询。

5.2.4 库存管理

“库存管理”功能模块用于管理企业的库存信息，所包含的子功能模块如图 5.5 所示。

“库存查询”用于查询企业商品的库存，可以查询总库存和各分仓库库存。

“库存转库”用于将一个仓库的商品转移到另一个仓库，或者连锁店之间的商品调拨，内容包括编号、源部门、目的部门、货号、数量、单价、合计金额、调拨日期、制单人、调货原因等。

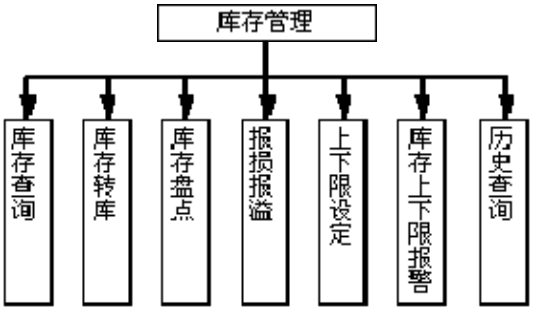


图 5.5 “库存管理”功能模块

“库存盘点”用于管理企业的库存盘点工作，将实际盘存的商品数量输入计算机，计算机自动与数据库中的库存数量进行核对，并产生盘盈盘亏统计信息。

“报损报溢”用于管理企业商品库存的损溢，内容包括编号、货号、仓库、数量、库存单价、金额、日期、责任人、制单人等。

“上下限设定”用于设置库存的上限和下限，当商品库存的数量超出指定的范围时系统可以自动报警，内容包括序号、仓库号、货号、上限、下限、最佳存量、备注等。

“库存上下限报警”根据当前商品库存和指定的库存上下限自动列出低于下限或高于上限的商品，并可以直接根据最佳库存量直接生成采购订单。

“历史查询”用于查询各种转库、盘点、损溢的历史数据。

5.2.5 应付款管理

“应付款管理”功能模块用于管理企业的应付款业务，包含的子功能模块如图 5.6 所示。

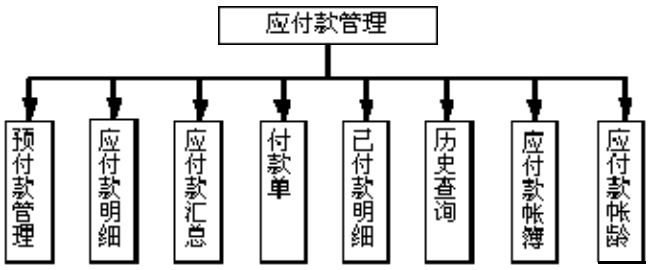


图 5.6 “应付款管理”功能模块

“预付款管理”用于管理企业的预付款信息，内容包括供货商号、预付款总额等。填写付款单时可以选择付款的方式，如果是“减预付款”，则“预付款”表中的“预付总额”将自动减少。

“应付款明细”用于查询企业所有的应付款，对应到每一笔进货的明细。

“应付款汇总”用于查询企业所有的应付款，对应到每一个供货商。

“付款单”用于录入企业支付货款的凭证和应付款，内容包括编号、发票号、填票日期、进货单号、货号、供货商号、数量、进货单价、金额、付款日期、详细说明、进货日期、状态、减预付等。

“已付款明细”用于查询企业历史的所有已经支付的款项。

“历史查询”用于查询付款单历史。

“应付款帐簿”用于查询应付款科目的明细分录，属于财务上的概念。每一笔进货或者付款发生

时，系统都会在帐簿中自动产生一笔记录，以借方、贷方、余额的形式存在，可以直接将查询的结果输出作为财务软件的凭证。

“应付款帐龄”用于分析应付款的帐龄，即在不同时间段内应付款分布的情况，作为付款依据。

5.2.6 应收款管理

“应收款管理”功能模块用于管理企业的应收款业务，包含的功能模块如图 5.7 所示。

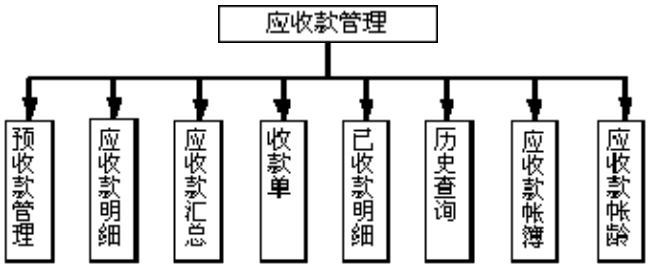


图 5.7 “应收款管理”功能模块

“预收款管理”用于管理企业的预收款信息，内容包括客户编号、预收款总额等。填写收款单时可以选择收款的方式，如果是“减预收款”则“预收款”信息表中的“预收总额”自动减少。

“应收款明细”用于查询企业所有的应收款，对应到每一笔销售的明细。

“应收款汇总”用于查询企业所有的应收款，对应到每一个客户。

“收款单”用于录入企业收回货款的凭证和应收款，内容包括编号、发票号、填票日期、销售单号、货号、客户编号、数量、销售价、金额、收款日期、详细说明、销售日期、状态、减预收等。

“已收款明细”用于查询企业历史的所有已收回的款项。

“历史查询”用于查询收款单历史。

“应收款帐簿”用于查询应收款科目的明细分录，每发生一笔销售或收款业务，系统都会在帐簿中自动产生一笔记录，以借方、贷方、余额的形式存在。可以直接将查询的结果输出作为财务软件的凭证。

“应收款帐龄”用于分析应收款的帐龄，即在不同时间段内应收款分布的情况，作为收款依据。

5.2.7 帐务管理

“帐务管理”功能模块用于管理企业的结帐业务，所包含的功能模块如图 5.8 所示。一般商业企业每个月都要进行一次结帐操作，确定该财务月份所有商品成本的进、销、结存情况，为财务管理提供数据。

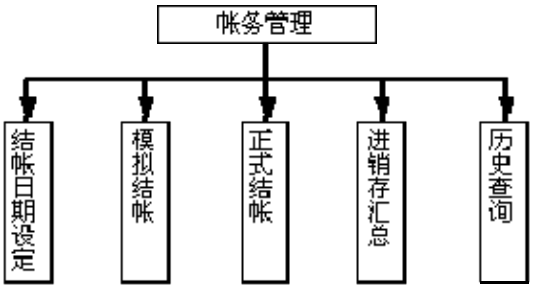


图 5.8 “帐务管理”功能模块

“结帐日期设定”用于设置企业每月结帐的日期，内容包括月份、全称、结帐日期等。一般商业企业每个月的结帐日期都是固定的，默认为 25 号。

“模拟结帐”用于进行模拟结帐操作。因为结帐操作不可逆，即正式结帐后商品进销存汇总的数据就无法更改了，因此在正式结帐前往往要进行模拟结帐，以查看结帐数据是否正确。

“正式结帐”用于进行结帐操作，用于计算进销存汇总报表。

“进销存汇总”用于查询本次结帐的结果，内容包括结帐编号、年月、货号、上期结帐数量、上期结帐金额、借方数量、借方金额、贷方数量、贷方金额、本期结存数量、本期结存金额、备注等。

“历史查询”用于查询历史结帐结果。

小知识：财务成本核算方法

财务上，为了统计企业的经营利润和报税，每个月都要对经营商品的成本进行核算，核算方法一般分为加权平均法和先进先出法(或者后进先出法)，下面来举例说明这两种方法的区别。

例如，1 月 10 日进了 100 双鞋子，进价为每双 50 元；1 月 15 日销售了其中的 80 双；1 月 20 日又进了 100 双，但进价变为每双 40 元了。采用加权平均法计算：当月销售的成本为 80 × (0 + 100 × 50 + 100 × 40) / (0 + 100 + 100) = 80 × 45 = 3 600 元；当月期末余额为 120 × 45 = 5 400 元。其中 0 表示当月该品种的期初数量和期初金额(上月的期末数量、金额)为 0，例如下月计算时取值为 5 400。由此可见，加权平均法计算公式为：销售成本 = 销售数量 × (上月期末金额 + 本月入库金额) / (上月期末数量 + 本月入库数量)。

而用先进先出法计算，因为销售的 80 双属于第一次进货的，其真实的成本应该是 80 × 50 = 4 000 元。由此可见两种不同的计算方法对企业当月的利润是不一样的，后一种增加了 400 元的成本，利润就降低了 400 元。 到这里，也许用户会问，这两种方法有什么区别？最终企业的利润总是一样的，只不过是先算还是后算的问题。首先，先进先出法可以更加准确的反映出企业的实际 经营状况；其次在不同的动机下会产生不同的结果，例如一般商品的价格总是下降的趋势，如果上市公司想增加本年度的利润，采用先进先出法就可以合法地提高公 司的利润。需要注意的是，先进先出法需要大量的运算工作，因此在手工操作模式下是不现实的，这也是加权平均法产生的根本原因。

5.2.8 统计分析

“统计分析”功能模块用于统计和分析企业的经营数据，供企业决策者作为决策依据。常用的功能如图 5.9 所示。

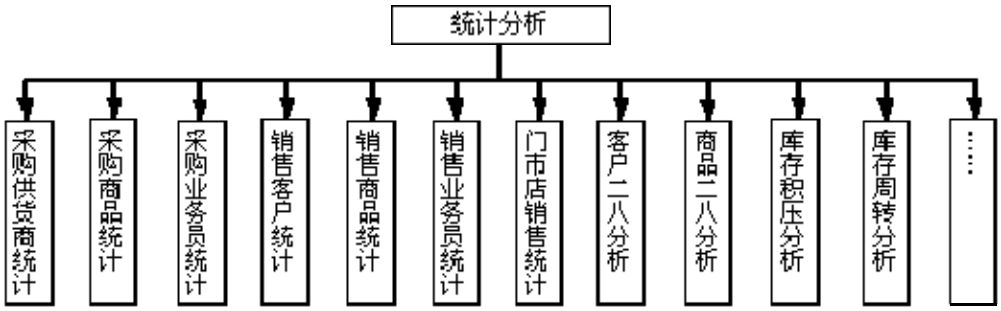


图 5.9 “统计分析”功能模块

“统计分析”功能模块对企业的历史数据进行统计分析，并将这些数据的分析结果以直观的形式表现出来，当然最理想的情况是以图表的形式表现出来，如图 5.10 和图 5.11 所示。以上列举的只是“统计分析”功能模块的一小部分功能，其他还有商品时段销售对比、门店时段销售对比等功能。

图 5.10　同一商品不同时段销售对比

图 5.11　不同门店销售分析对比

小知识：二八(ABC)分析方法

意大利经济学家帕列托于 19 世纪发现：社会约 80% 的财富集中在 20% 的人手里，而其余 80% 的人只拥有 20% 的社会财富。这种统计的不平衡性在社会经济及生活中无处不在，这就是二八法则，即 80% 的结果(如产出、酬劳、销售等)，往往源于 20% 的原因(如投入、努力、商品等)。对应到流通企业，就是 20% 的商品(客户)大致带来 80% 的销售额和毛利，因此需要找出这些重点品种和重点客户，从而更好地发挥它们的优势。ABC 分析与二八分析原理是一致的，只不过 ABC 分析把结果分为 3 个等级，而二八分析只分为两个等级。

5.2.9　零售管理

流通企业分为批发企业和零售企业，零售企业又分为单门店企业和连锁企业。对于零售企业，其自身的特点必然导致进销存系统与批发企业的系统不一致。最大的区别就是零售销售的界面，还有连锁门店的管理。“零售管理”功能模块包括的子功能模块如图 5.12 所示。

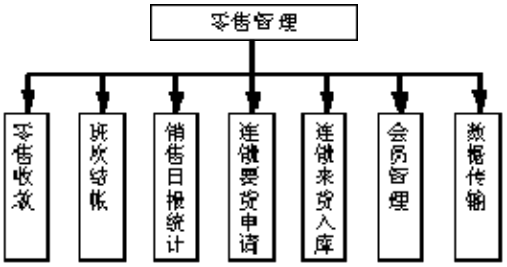


图 5.12 “零售管理”功能模块

“零售收款”用于门店的 POS 收款管理。正如前面所说的，零售企业不管把东西卖给了谁，也不会产生应收款。而由于顾客收款排队，需要保证最快的收款速度，因此“零售收款”的功能要简单便捷，往往只输入商品的货号、数量和收款的总金额即可。

“班次结帐”用于零售企业的结帐管理。零售企业一般都是分班次销售的，一天一个班次或两个班次，各班次的人员之间交班前需要将收款的金额点清，并和计算机统计的金额一致后，选择“班次结帐”功能模块，确定并清除该班次的收款信息，将其放入历史记录，从而实现顺利交班。

“销售日报统计”用于统计零售企业日销售的信息，内容包括日期、部门、组别、收款机号、货号、班次、销售数量、应收金额、实收金额、库存单价等。

“连锁要货申请”用于管理连锁门店向配送中心请求配货，内容包括编号、部门、组别、货号、数量、日期、申请人、申请说明、回复等。

“连锁来货入库”用于管理连锁门店在接收到配送中心的货物时进行入库确认。

“会员管理”用于管理企业的会员信息，可以增加新会员、维护现有会员信息、查询会员消费明细、设置会员折扣等。

“数据传输”用于管理连锁门店、配送中心和企业总部之间的数据传输，传输的操作应该简单安全。

5.2.10 系统管理

系统管理是每个系统都必须具备的功能，包括的子功能模块如图 5.13 所示。由于这部分功能比较通用，故在后面的实例中将不再详细加以介绍。

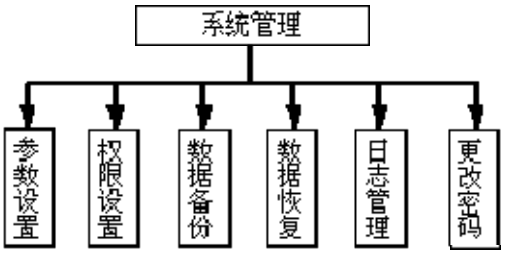


图 5.13 “系统管理”功能模块

“参数设置”用于设置系统定义的一些基础参数或开关。

“权限设置”用于设置各操作员使用系统的权限，为了方便设置，一般的应用系统都是可以将操作人员分组的，将通用的权限赋予整个组，个别的权限单独赋予个人，这样可以大大减少权限管理的工作量。

“数据备份”用于备份系统数据库。

“数据恢复”用于恢复系统数据库。

“日志管理”用于维护系统的使用日志。一个好的应用系统会对任何操作员进行的所有操作进行日志记录，“日志管理”可以查询、导出和删除历史的日志。

“更改密码”供用户更改自己的密码。

5.3 进销存管理系统数据库分析

根据以上需求分析，一个基本的进销存管理系统数据库中大致包括 60 多张表，分别存放相应子功能的数据信息，其中商品清单、供货商清单和客户清单都是关键表格，用于存放基础的数据信息。其他涉及商品、供货商和客户信息的表，都只记录这些元素的编号，根据作为外键的编号来对应。因此这三张表和其他表间的关系是 1∶N 的关系。

5.3.1 进销存管理系统E-R图

因为整个系统涉及的实体和属性较多，限于篇幅，这里不能也没有必要一一列举。图 5.14 为进销存管理系统关键实体的 E-R 图(即实体—关系图)。

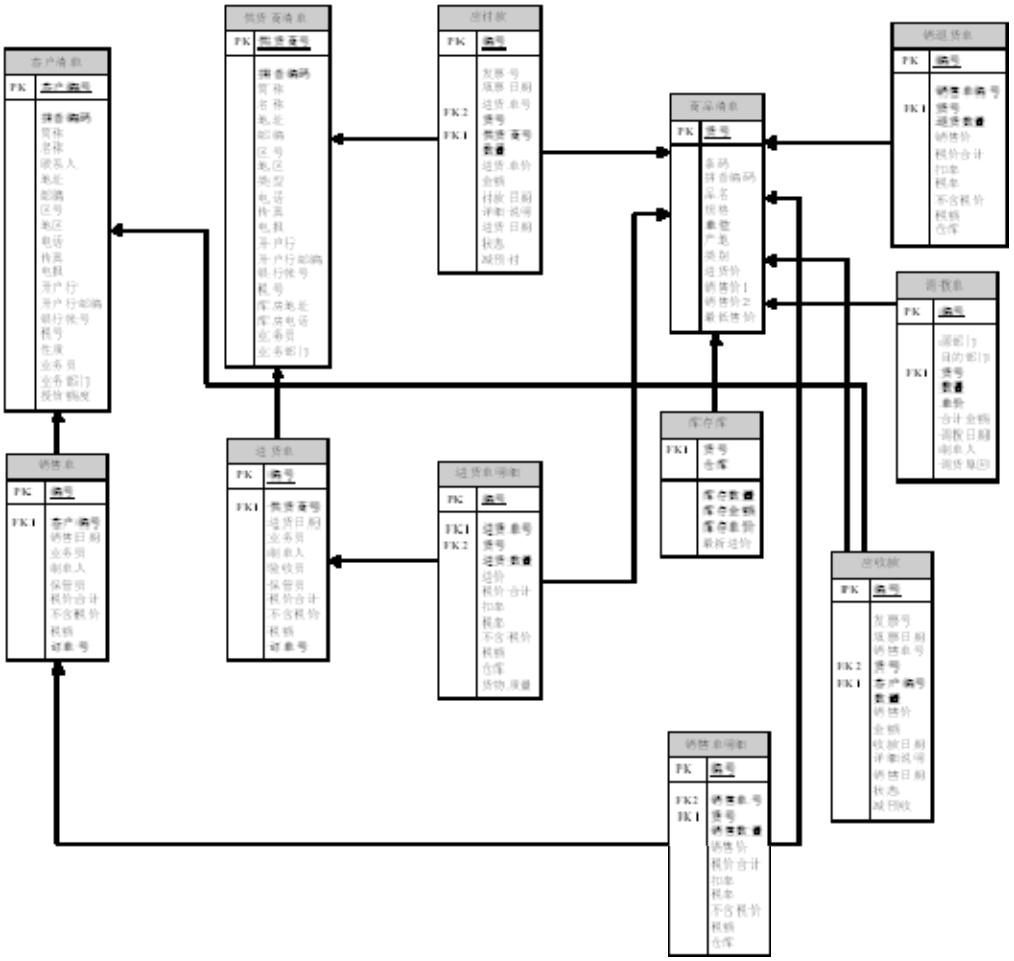


图 5.14 进销存管理系统 E-R 图

其他实体与基本信息表间的对应关系都是类似的，同时配书光盘的“\Chap5\建库脚本\进销存管理.sql”文件提供了创建数据库所有表的脚本，被省略的实体对象和实体属性用户完全可以参考这些脚本，也可以使用 Microsoft Visio 2002 自动生成全部实体和属性的 E-R 图。

5.3.2 进销存管理系统表清单

配书光盘中“\Chap1\建库脚本\进销存管理.sql”文件提供了创建数据库所有表的脚本，下面我们仅列出一些重要表的名称及其用途供用户参考(如表 5.1 所示)。关于各表所包含的字段，用户可以自己查看建库脚本和前面的需求分析。

表 5.1 进销存管理系统表清单

表 名 称	表 用 途
用户清单	保存系统使用者的信息
权限清单	保存系统使用者的权限信息，可以指定到菜单级权限
商品清单	保存企业经营商品的资料信息
供货商清单	保存企业供货商的资料信息

客户清单	保存企业客户的资料信息
业务员清单	保存企业业务员信息
仓库清单	保存企业仓库设置信息
采购合同	保存企业与供货商签订的采购合同
库存库	保存企业商品库存的数量、金额等信息
采购订单	采购订单主表，填写采购订单时使用
采购订单明细	采购订单从表
采购订单历史	保存采购订单历史，填写采购订单确认后单据导入历史
采购订单明细历史	保存采购订单明细历史
进货单	进货单主表
进货单明细	进货单从表
进货单历史	保存进货单历史
进货单明细历史	保存进货单明细历史
进价调整单	进价调整时保存进价调整的信息
销售订单	销售订单主表
销售订单明细	销售订单从表
销售订单历史	保存销售订单历史
销售订单明细历史	保存销售订单明细历史
销售单	销售单主表
销售单明细	销售单从表
销售单历史	保存销售单历史
销售单明细历史	保存销售单明细历史
销退货单	填写销售退货单时使用该表，确定后数据导入销退货单历史
销退货单历史	保存销售退货单历史
调拨单	填写调拨单时使用该表，确定后数据导入调拨单历史
调拨单历史	保存调拨单历史
报损报溢	填写报损报溢单时使用该表，确定后数据导入报损报溢历史
报损报溢历史	保存报损报溢历史
上下限对照表	保存各仓库的库存上、下限数量
应付款	保存企业应付款明细数据
应收款	保存企业应收款明细数据
预收款	保存企业预收款数据
预付款	保存企业预付款数据
每月结帐日期表	保存每个月份的结帐日期
结帐历史	保存每月结帐的历史
进销存汇总表	保存每月结帐的结果
销售日报	保存零售的销售数据
调货申请	门店填写调货申请时使用该表
盘点单	填写商品盘点单时使用该表，确认后数据导入盘点单历史
盘点单历史	保存商品盘点单历史

5.3.3 利用Microsoft Visio 2002 获取系统E-R图

本书所有 E-R 图都是使用 Microsoft Visio 2002 绘制的，利用 Microsoft Visio 2002 可以快速获取全系统 E-R 图。首先利用配书光盘中提供的创建数据库所有对象的脚本创建数据库，并建立一个指向该数据库的 ODBC 连接(不知道如何建立数据库和 ODBC 连接的用户参看随书光盘“\程序运行所需文件”目录下的文档)；然后进入 Microsoft Visio 2002，选择【新建】|【数据库】|【数据库模型

图】命令，新建文档，如图 5.15 所示。

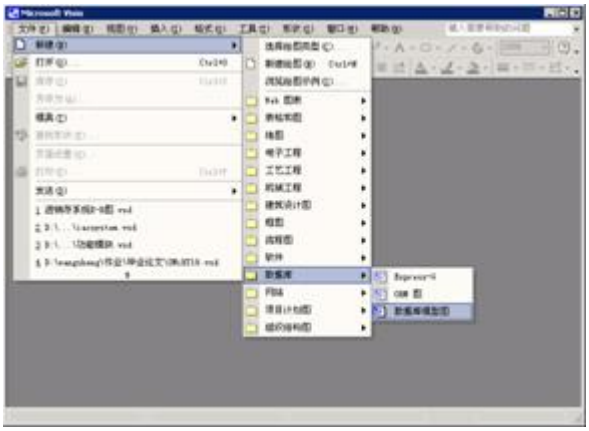


图 5.15　新建数据库模型图文档

随后选择【数据库】|【反向工程…】命令，弹出【反向工程向导】对话框，如图 5.16 所示。按照向导操作，选择前面的建立的 ODBC 数据连接、需要反向工程的对象类型和 E-R 图的实体，如图 5.17 和 5.18 所示，经过确认后就可以得到指定实体的 E-R 图了。

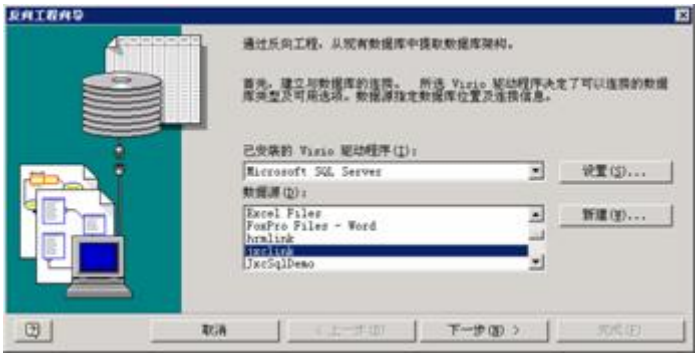


图 5.16　【反向工程向导】对话框

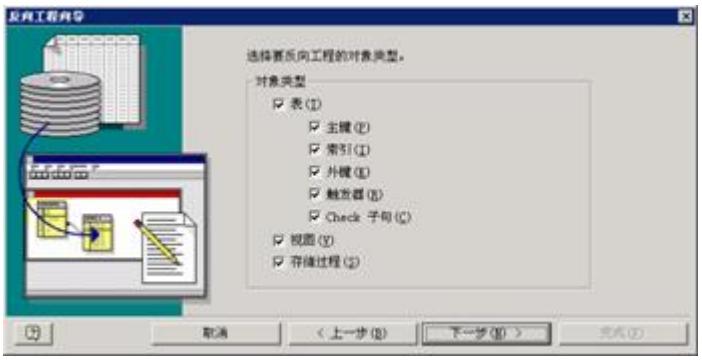


图 5.17　选择所需对象类型

图 5.18　选择所需实体

5.4 实例制作介绍

基于以上需求分析和数据库分析，用户对标准的进销存管理系统应该有了一个全面的认识。下面将通过实例说明如何利用 Delphi 完成系统的开发。

5.4.1 实例功能

由于篇幅有限，本实例将详细介绍如图 5.19 所示功能的开发过程，并简化其中各功能模块所包含的属性，其他功能用户完全可以参照这些功能的开发方法去实现。

需要强调的是，由于用户登录和权限管理的功能各个系统实现的方法是一致的，故在后面的实例中，将不再包含权限管理的功能。

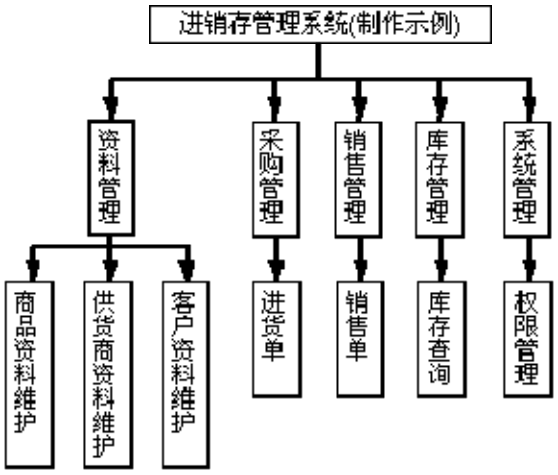


图 5.19 详细介绍的功能模块

5.4.2 系统流程图

系统流程图如图 5.20 所示。

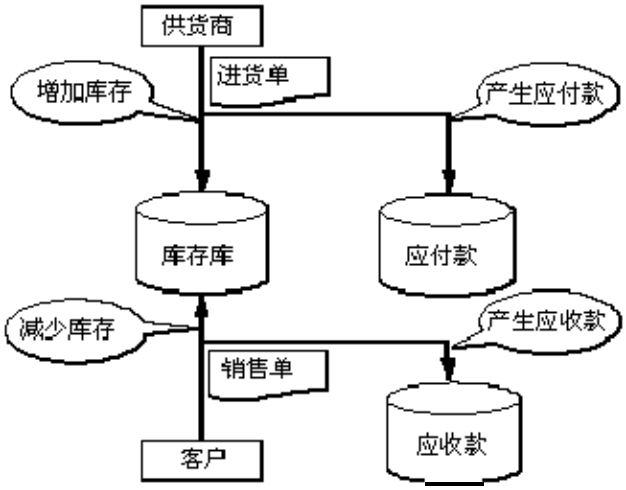


图 5.20 系统流程图

5.5 数据库设计

根据 5.4 节的介绍，本实例系统共需要 17 张表，用途分别如表 5.2 所示。

表 5.2 实例系统表及其用途

表 名 称	表 用 途
用户清单	保存系统使用者的信息

权限清单	保存系统使用者的权限信息，可以指定到菜单级权限
商品清单	保存企业经营商品的资料信息
供货商清单	保存企业供货商的资料信息
客户清单	保存企业客户的资料信息
仓库清单	保存企业库房的分类信息，可以用于数据辞典
业务员清单	保存企业的业务员信息，可以用于数据辞典
进货单	进货单主表
进货单明细	进货单从表
进货单历史	保存进货单历史
进货单明细历史	保存进货单明细历史
销售单	销售单主表
销售单明细	销售单从表
销售单历史	保存销售单历史
销售单明细历史	保存销售单明细历史
库存单	保存企业商品库存的数量、金额等信息
应付款、应收款	保存企业应付款明细、应收款明细

各表之间的 E-R 图可以参考前面的图 5.14。

5.5.1 创建数据库

打开 SQL Server 2000 企业管理器，新建一个数据库，名称为 jxcbook。利用配书光盘中的脚本代码“进销存管理.sql”(位于“\Chap5\建库脚本”目录下)文件创建数据库对象，完成数据库的设计。

在后面几小节中，将列出几个重点的表的建库脚本，其他表的脚本参考脚本文件。

小知识：使用 SQL Server 创建数据库

SQL Server 对一个数据库采用两个文件来管理，一个是数据文件，扩展名为.mdf；另一个是日志文件，扩展名为.ldf。新建数据库的方法有多种，一般使用 SQL Server 企业管理器来操作。通过 Windows 的【开始】菜单运行 SQL Server 的企业管理器，如图 5.21 所示。

图 5.21 运行 SQL Server 的企业管理器

进入企业管理器后单击服务器名称左边的＋号，一直将树型结构展开。在【数据库】项目上右击，在弹出的快捷菜单中选择【新建数据库】命令，如图 5.22 所示。

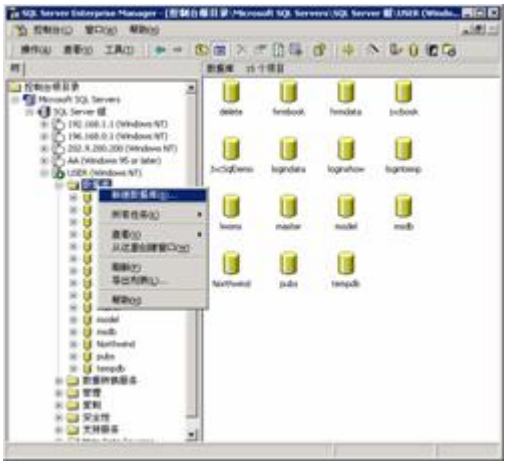


图 5.22 新建数据库

在随后出现的对话框中填写数据库的名称等信息，如图 5.23 所示。也可以在该对话框中修改数据文件和日志文件的存放位置。完成后单击【确定】按钮，即可完成数据库的创建工作。

图 5.23 设置新建数据库信息

通过上面的步骤，只是建立了一个空的数据库，还需要设计数据库中的表、存储过程等数据库元素。最简单的方法是利用 SQL Server 2000 的查询分析器打开建库脚本，直接运行它，系统的相关元素就可以自动创建。

5.5.2 创建“商品清单”表

创建“商品清单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[商品清单] (
   [货号] [char] (14) NOT NULL Primary Key,
   [条码] [char] (14) NULL ,
   [拼音编码] [char] (40)  NULL ,
   [品名] [varchar] (80)  NULL ,
   [规格] [varchar] (40)  NULL ,
   [单位] [char] (6)  NOT NULL ,
   [产地] [varchar] (50)  NULL ,
   [类别] [char] (20) NULL ,
   [进货价] [decimal] (28,6)  NULL default(0),
   [销售价 1] [decimal] (28,6)  NULL default(0),
   [销售价 2] [decimal] (28,6)  NULL default(0),
   [最低售价] [decimal] (28,6)  NULL default(0)
)
GO
```

5.5.3 创建“供货商清单”表

创建“供货商清单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[供货商清单] (
   [供货商号] [char] (10)  NOT NULL Primary Key,
   [拼音编码] [char] (40)  NOT NULL ,
   [简称] [varchar] (80)  NULL ,
   [名称] [varchar] (80)  NULL ,
   [地址] [varchar] (80)  NULL ,
   [邮编] [char] (6)  NULL ,
   [区号] [char] (6)  NULL ,
   [地区] [varchar] (12)  NULL ,
   [类型] [char] (10) NULL ,
   [电话] [varchar] (20)  NULL ,
   [传真] [varchar] (20)  NULL ,
   [电报] [varchar] (20)  NULL ,
   [开户行] [varchar] (40) NULL ,
   [开户行邮编] [char] (6)  NULL ,
   [银行帐号] [varchar] (20)  NULL ,
   [税号] [varchar] (20)  NULL ,
   [库房地址] [varchar] (40)  NULL ,
   [库房电话] [varchar] (20)  NULL ,
   [业务员] [char] (10)  NULL ,
```

```
    [业务部门] [varchar] (20)  NULL ,
)
GO
```

5.5.4 创建“客户清单”表

创建“客户清单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[客户清单] (
    [客户编号] [char] (10)  NOT NULL Primary Key,
    [拼音编码] [char] (20)  NOT NULL ,
    [简称] [varchar] (80)  NULL ,
    [名称] [varchar] (80)  NULL ,
    [联系人] [varchar] (30) NULL ,
    [地址] [varchar] (80)  NULL ,
    [邮编] [char] (6)  NULL ,
    [区号] [char] (6)  NULL ,
    [地区] [varchar] (12)  NULL ,
    [电话] [varchar] (20)  NULL ,
    [传真] [varchar] (20)  NULL ,
    [电报] [varchar] (20)  NULL ,
    [开户行] [varchar] (40) NULL ,
    [开户行邮编] [char] (6)  NULL ,
    [银行帐号] [varchar] (20)  NULL ,
    [税号] [varchar] (20)  NULL ,
    [性质] [varchar] (10)  NULL ,
    [业务员] [char] (10)  NULL ,
    [业务部门] [varchar] (20)  NULL ,
    [授信额度] [decimal] (28,6)  NULL,
)
GO
```

5.5.5 创建“进货单”和“进货单明细”表

创建“进货单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[进货单] (
    [编号] [char] (14) Not NULL Primary key,
    [供货商号] [char] (10)  NOT NULL ,
    [进货日期] [datetime] NULL,
    [业务员]  [char] (10)  NULL ,
    [制单人]  [char] (10)  NULL ,
    [验收员] [char] (10)  NULL ,
    [保管员] [char] (10)  NULL ,
    [税价合计] [decimal] (28,6)  NULL ,
    [不含税价] [decimal] (28,6)  NULL ,
    [税额] [decimal] (28,6) NULL,
    [订单号] [char] (14)  NULL
)
```

```
GO
```

创建“进货单明细”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[进货单明细] (
   [编号] [char] (14) Not NULL Primary key,
   [进货单号] [char] (14)  Not NULL ,
   [货号] [char] (14) NOT NULL ,
   [进货数量] [decimal] (28,6)  NOT NULL ,
   [进价] [decimal] (28,6) NULL ,
   [税价合计] [decimal] (28,6)  NULL ,
   [扣率] [decimal] (28,6) NULL ,
   [税率] [decimal] (28,6) NULL ,
   [不含税价] [decimal] (28,6)  NULL ,
   [税额] [decimal] (28,6) NULL ,
   [仓库] [char] (20) NULL ,
   [货物质量] [varchar] (50)  NULL
)
GO
```

5.5.6 创建“销售单”和“销售单明细”表

创建“销售单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[销售单] (
   [编号] [char] (14) Not NULL Primary key,
   [客户编号] [char] (10)  NOT NULL ,
   [销售日期] [datetime] NULL,
   [业务员]  [char] (10)  NULL ,
   [制单人]  [char] (10)  NULL ,
   [保管员] [char] (10)  NULL ,
   [税价合计] [decimal] (28,6)  NULL ,
   [不含税价] [decimal] (28,6)  NULL ,
   [税额] [decimal] (28,6) NULL,
   [订单号] [char] (14)  Not NULL
)
GO
```

创建“销售单明细”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[销售单明细] (
   [编号] [char] (14) Not NULL Primary key,
   [销售单号] [char] (14)  Not NULL ,
   [货号] [char] (14) NOT NULL ,
   [销售数量] [decimal] (28,6)  NOT NULL ,
   [销售价] [decimal] (28,6)  NULL ,
   [税价合计] [decimal] (28,6)  NULL ,
```

```
    [扣率] [decimal] (28,6) NULL ,
    [税率] [decimal] (28,6) NULL ,
    [不含税价] [decimal] (28,6) NULL ,
    [税额] [decimal] (28,6) NULL ,
    [仓库] [char] (20) NULL
)
GO
```

5.5.7 创建其他重要表

创建“库存库”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[库存库] (
    [货号] [char] (14) NOT NULL ,
    [仓库] [varchar] (20) NOT NULL ,
    [库存数量] [decimal] (28,6) NOT NULL ,
    [库存金额] [decimal] (28,6) NOT NULL ,
    [库存单价] [decimal] (28,6) NOT NULL ,
    [最新进价] [decimal] (28,6) NULL
)
GO
```

创建“权限清单”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[权限清单] (
    [权限序号] [int] IDENTITY (1, 1) NOT NULL ,
    [用户编号] [char] (6) NULL ,
    [部门] [char] (20) NULL ,
    [权限名称] [char] (6) NOT NULL
)
GO
```

5.5.8 创建外部关键字

以下为脚本代码为需要特别注意的外部关键字，其他外部关键字参考脚本文件。

```
-- 外键关联字段
ALTER TABLE [dbo].[进货单] ADD
    CONSTRAINT [进货单_供货商_fk] FOREIGN KEY
    (
      [供货商号]
    ) REFERENCES [dbo].[供货商清单] (
      [供货商号]
    )
GO
ALTER TABLE [dbo].[进货单明细] ADD
    CONSTRAINT [FK_进货单明细_进货单] FOREIGN KEY
    (
      [进货单号]
```

```
    )  REFERENCES [dbo].[进货单] (
      [编号]
    ),
    CONSTRAINT [进货单明细_货号_fk] FOREIGN KEY
    (
      [货号]
    )  REFERENCES [dbo].[商品清单] (
      [货号]
    )
GO
ALTER TABLE [dbo].[销售单] ADD
    CONSTRAINT [销售单_客户编号_fk] FOREIGN KEY
    (
      [客户编号]
    )  REFERENCES [dbo].[客户清单] (
      [客户编号]
    )
GO
ALTER TABLE [dbo].[销售单明细] ADD
    CONSTRAINT [FK_销售单明细_销售单] FOREIGN KEY
    (
      [销售单号]
    )  REFERENCES [dbo].[销售单] (
      [编号]
    ),
    CONSTRAINT [销售单明细_货号_fk] FOREIGN KEY
    (
      [货号]
    )  REFERENCES [dbo].[商品清单] (
      [货号]
    )
GO
```

5.5.9 创建存储过程

系统使用了两个存储过程，分别实现进货时加大库存、产生应付款和销售时减少库存、产生应收款的功能。

```
-- 存储过程
CREATE PROCEDURE sf_进货单 @记帐人 char(10) = NULL
AS
begin transaction
   -- 库存库中没有,增加记录
   INSERT INTO 库存库(货号,仓库,库存数量,库存金额,库存单价)
       SELECT DISTINCT j.货号, j.仓库, 0,0,0
       FROM 进货单明细 AS J left join 库存库 as k on ( j.仓库=k.仓库 and j.货号=k.货号 )
```

```
            where k.货号 is null
        -- 修改库存信息
        UPDATE 库存库 SET 库存单价=case when 库存数量<=0 or (库存数量+数量 ALL)<=0 then 进价
                    else (库存金额+税价合计 ALL)/(库存数量+数量 ALL)  end ,
            库存数量=库存数量+数量 ALL,
            库存金额=case when 库存数量<=0 or (库存数量+数量 ALL)<=0
              then 进价*(库存数量+数量 ALL)  else (库存金额+税价合计 ALL)  end ,
            最新进价=进价
          FROM
          (SELECT 仓库,货号,'数量 ALL'=sum(进货数量),'进价'= sum(税价合计)/sum(进货数量),
            '税价合计 ALL'=sum(税价合计)  FROM 进货单明细 GROUP BY 仓库,货号) AS LSJ
          WHERE 库存库.仓库=LSj.仓库 AND 库存库.货号=LSj.货号
        -- 加入应付款
        INSERT INTO 应付款(编号, 进货单号, 货号, 供货商号, 数量, 进货单价, 金额, 进货日期,
状态)
          SELECT   '付'+a.编号,b.编号,a.货号,b.供货商号,进货数量,进价,a.税价合计,进货日
期,'应付'
          FROM 进货单明细 as a, 进货单 as b
          where a.进货单号=b.编号
        -- 加入历史
        insert into 进货单历史 select * from 进货单
        insert into 进货单明细历史 select * from 进货单明细
        -- 清除进货单
        delete from 进货单明细
        delete from 进货单
    commit
    go

    CREATE PROCEDURE sf_销售单 @记帐人 char(10) = NULL
    AS
    begin transaction
        -- 修改库存信息
        UPDATE 库存库 SET 库存数量=库存数量-数量 ALL, 库存金额=库存单价*(库存数量-数量
ALL)
          FROM (SELECT 仓库,货号,'数量 ALL'=sum(销售数量)  FROM 销售单明细
          GROUP BY 仓库,货号) AS LSJ
          WHERE 库存库.仓库=LSj.仓库 AND 库存库.货号=LSj.货号
        -- 加入应收款
        INSERT INTO 应收款(编号, 销售单号, 货号, 客户编号, 数量, 销售价, 金额, 销售日期,
状态)
          SELECT '收'+a.编号,b.编号,a.货号,b.客户编号,销售数量,销售价,a.税价合计,销售日
期,'应收'
          FROM 销售单明细 as a, 销售单 as b
          where a.销售单号=b.编号
        -- 加入历史
```

```
    insert into 销售单历史 select * from 销售单
    insert into 销售单明细历史 select * from 销售单明细
    -- 清除销售单
    delete from 销售单明细
    delete from 销售单
commit
```

5.6 程序开发

5.6.1 程序运行结果

本实例在运行后，主窗体将如图 5.24 所示，单击相应按钮可以进入对应的功能。

图 5.24 实例的主窗体

(1) 运行程序，出现的是【用户登录】对话框，如图 5.25 所示。默认的系统管理员编号为“1”，密码为“sys”。在用户编号中输入“1”后回车，输入密码再回车，就可以登录进入系统主窗体。

图 5.25 用户登录

注意：如果是用脚本建立的数据库，由于还没有创建权限管理的功能，所以此时“用户清单”表是空的，没有用户信息。此时可以人为在表中输入用户信息；如果是使用随书光盘中的数据库，则可以使用用户号为 1 的用户。

(2) 单击【资料管理】按钮，进入【资料管理】功能窗体，在其中输入各种商品信息，如图 5.26 所示。这里一个【资料管理】窗体对应可以实现 3 个功能，通过菜单切换，单击【资料管理】按钮后默认的功能是商品资料管理。

图 5.26 商品信息维护功能

在【商品拼音】文本框中输入商品拼音后，可以单击【查询】按钮查找商品信息。通过上边的工具栏可以增加、修改和删除一条记录。

(3) 选择【资料管理】|【供货商资料维护】命令，进入【供货商资料维护】功能，如图 5.27 所示，其使用方法和商品资料维护功能一样。

图 5.27 供货商资料维护功能

(4) 选择【资料管理】|【客户资料维护】命令，进入【客户资料维护】功能，如图 5.28 所示，其使用方法和商品资料维护功能一样。

(5) 关闭【资料管理】窗体，返回到主窗体。单击【进货管理】按钮，进入【进货单】功能窗体，如图 5.29 所示。

单击【供货商】组合框的下拉按钮，弹出选择供货商的窗体，选择供货商。然后在下面的明细列表中输入进货的商品、数量及单价等信息，完成后单击【保存进货单】按钮，保存数据。单击【显示进货单报表】按钮可以查看(打印预览)进货单据的打印报表样式，如图 5.30 所示。

图 5.28 客户信息维护功能

图 5.29 【进货单】功能窗体

图 5.30 进货单报表

单击【打印进货单并记帐】按钮，系统将在打印机上打印进货单据，然后调用“sf_进货单”存储过程，完成增加商品库存、产生应付款以及将单据导入历史等功能。

(6) 关闭【进货单】窗体，返回到主窗体。单击【销售管理】按钮，进入【销售单】功能窗体，如图 5.31 所示。其使用方法和进货单基本一致，只是供货商变成了客户，进价变成了售价，区别在于选择商品时必须从有存货的仓库中选择。

图 5.31 【销售单】功能窗体

(7) 关闭【销售单】窗体，返回到主窗体。单击【库存管理】按钮，进入【库存查询】功能窗体，如图 5.32 所示，在这个窗体中可以根据商品拼音、货号及储存仓库对商品仓库中的商品进行查询。

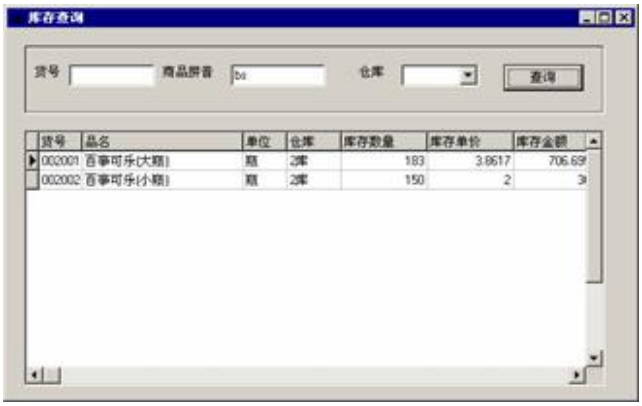


图 5.32 【库存查询】功能窗体

(8) 关闭【库存查询】窗体，返回到主窗体。单击【系统管理】按钮，进入【权限管理】功能窗体，如图 5.33 所示。这里可以修改用户清单中各个用户对于系统中各个子功能的访问权限。作为本实例中的系统管理员，用户 sys 的权限包括了所有功能，且是不可修改的。修改权限后，如果使用受限用户登录，则他们不能访问的功能菜单是灰色的

图 5.33 权限管理功能

5.6.2 创建工程

(1) 启动 Delphi 7.0，选择 File | New | Application 命令，创建一个新的工程，如图 5.34 所示。

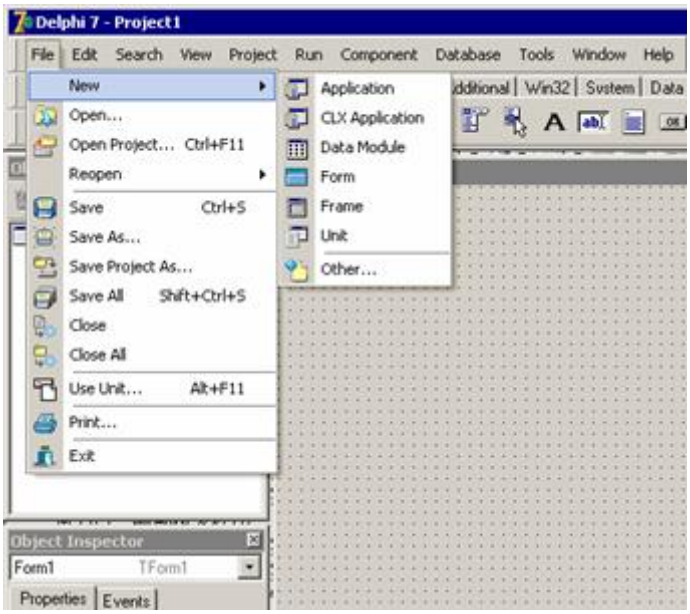


图 5.34 创建新的工程

(2) 根据程序的不同，可以自己为程序创建不同的图标。

选择 Project | Options 命令，弹出如图 5.35 所示的 Project Options for…对话框。在 Application 选项卡中修改工程的 Title 属性为“进销存管理系统”，并为项目选择一个可执行文件图标，本例中选择源代码目录下的“jxc.ico”文件。

图 5.35 Project Options for…对话框

(3) 创建新的工程之后，需要将其窗体文件和工程文件分别保存，这时候可以选择 File | Save All 命令，就会依次弹出两个对话框，提示保存窗体文件和保存工程文件，如图 5.36 和图 5.37 所示。

图 5.36　保存窗体文件

图 5.37　保存工程文件

这里需要说明的是，工程文件和窗体文件是完全不同的两个概念，前者是整个工程的描述，而后者仅为针对一个窗体的描述，在上述过程中，保存工程文件并将其命名为“jxc.dpr”，保存窗体文件并命名为“FormLogin.pas”。

小知识：　Delphi 中窗体的 name 属性、caption 属性与保存文件名的区别

caption 属性可以用于在窗体的标题栏上显示作者指定的字符，而 name 属性是一个窗体在程序代码中的名称，如果要在其他窗体中使用一个窗体，那么就需要在其他窗体的代码中加入对这个窗体的引用代码，例如，想在 form1 中设置 form2 的一个 button 控件——button1 的可视化属性，其代码为 form2.button5. visible:=true。文件名则是保存窗体时的名称，每一个窗体在储存时都被存为“窗体文件名.pas”。

5.6.3 系统登录功能的实现

用户登录窗体是在整个程序运行之前，首先呈现在用户面前的界面，必须通过它验证用户的合法性，即除非用户输入正确的用户名和密码，否则将无法使用进一步的功能。本节将演示利用 Delphi 实现系统登录和用户识别功能的方法。创建好的窗体如图 5.38 所示。

图 5.38 【用户登录】窗体

(1) 选择 New｜Form 命令，创建一个新窗体，该窗体包括 3 个 Edit 控件、1 个 Button 控件、3 个 Label 控件和 1 个 ADOQuery 组件，其布局如图 5.39 所示，将窗体保存为 FormLogin.pas。

图 5.39 用户登录窗体包含的组件

(2) 这些控件的取值如表 5.3 所示。

表 5.3 系统登录窗体组件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FormLogin	Caption	用户登录
Label	Label	Caption	用户编号
Label	Labe2	Caption	用户名
Label	Labe2	Caption	密码
Edit	Edit1	Text	1
Edit	Edit2	PasswordChar	*(隐藏输入的密码)
Button	Button1	Caption	登录

上表中没有提到 ADOQuery 组件，下面将对它进行单独的介绍。

小知识：ADO 组件

ADO(Microsoft ActiveX Data Objects)组件页共有 ADOConnection 组件、ADODataSet 组件、ADOTable 组件、ADOQuery 组件、ADOStoredProc 组件、ADOCommand 组件和 RDSConnection 组件等 7 个组件。ADO 是 Microsoft 公司提供的访问各种数据的高级接口。这一数据访问技术的应用数据库接口称为 OLEDB，OLEDB 加快了访问各种数据的速度。从 Delphi 5.0 开始，Delphi 中新的数据访问组件都采用了 ADO 技术，可以不通过 BDE 就能使用现行的数据控制部件如 DBGrid 和 DBEdit 对数据进行直接的访问，因此缩短了 ADO/OLE-DB 的运行时间。更重要的是，使用 ADO 组件可以避免用户在使用程序之前手动设置 BDE 和 ODBC 属性，以免产生不必要的错误。Delphi 7.0 中的 ADO 组件页如图 5.40 所示。

图 5.40 ADO 组件页

以后的程序中常常用到其中的 3 个组件，分别是 ADOTable 组件、ADOQuery 组件和 ADOCommand 组件。

ADOTable 组件用于检索或者操作由单一表生成的数据集。该组件可以直接连接到数据设备上或者通过 ADOConnection 组件连接到数据设备上。

ADOQuery 组件用于检索或操作由一个合法的 SQL 语句生成的数据集。该组件可以执行数据定义 SQL 语句，例如“select * from table”。该组件可以直接连接到数据设备上或者通过 ADOConnection 组件连接到数据设备上。

ADOCommand 组件主要用于执行命令(这些命令是一组不返回结果集的 SQL 语句，如果要得到返回结果集，则应该使用 ADOQuery 组件)，它一般与一个数据集支撑部件一起使用。可以实现从一个表中检索数据。该组件可以直接连接到数据设备上或者通过 ADOConnection 组件连接到数据设备上。

(3) 为使 ADOQuery 组件正常工作，必须先设置它的属性。

ConnectionString 属性是 ADO 组件共有的属性之一，也是最重要的属性之一。ConnectionString 属性描述了数据库连接建立的连接信息，在运行过程中可以赋予一个字符串来定义连接，也可以在设计时通过 ADO 系统的设计向导来产生正确的 ConnectionString 属性。下面将介绍通过 ADO 系统的设计向导生成 ConnectionString 的步骤和方法，以及相关的 ADO 连接的一些概念。

以【用户登录】窗体中用到的 ADOQuery 组件为例，要设置它的 ConnectionString 属性，可以在 Object Inspector 对话框中双击 ConnectionString 属性或者是单击 ConnectionString 属性右边的省略号，如图 5.41 所示。

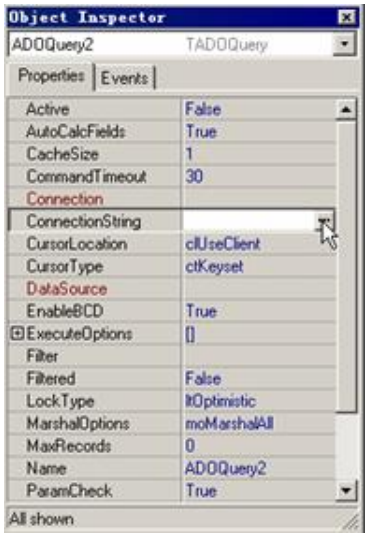


图 5.41 ADO属性设置--步骤 1

单击 ConnectionString 属性右边的省略号，出现如图 5.42 所示的描述连接字符串设置的对话框，在这个对话框中，可以指定 ADO 组件连接 ADO 数据仓库的连接字符串。也可以直接输入连接字符串，还可以使用 ADO 提供的对话框来生成这个字符串，还可以指定它为一个文件。在这里使用 ADO 提供的对话框来生成连接字符串。单击 Build 按钮，将显示 ADO 的【数据链接属性】设置对话框。

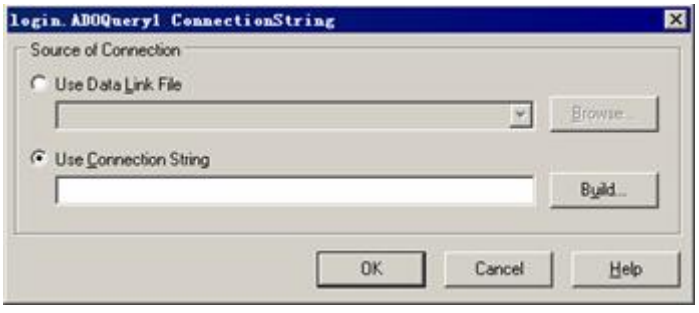


图 5.42 ADO属性设置--步骤 2

(4) 【数据链接属性】对话框由 4 个选项卡组成，分别是【提供程序】、【连接】、【高级】

和【所有】,【提供程序】选项卡中的列表框中显示了 ADO 从本机硬盘上扫描得到的所有 OLE DB Provider。在这里需要提醒大家的是，选择不同的 OLE DB Provider 所使用的 ConnectionString 参数是不同的，所以需要根据程序中所使用的数据库来确定选择哪一个 OLE DB 提供程序。OLE DB Provider 可以是 Microsoft 公司提供的，如 Microsoft OLE DB Provider for Oracle，Microsoft OLE DB Provider for Outlook Search,也可以是非 Microsoft 公司提供的。由于本程序所使用的后台数据库是 SQL Server，所以需要选择 Microsoft OLE DB Provider for SQL Server，然后单击【下一步】按钮，如图 5.43 所示。

图 5.43 ADO属性设置——步骤 3

(5) 设置好【提供程序】选项卡后，打开【连接】选项卡，在这里设置具体的服务器名称，登录数据库服务器的用户名和密码，以及在服务器上需要连接的数据库，设置完成后，还可以测试连接是否成功，如图 5.44 所示。首先，在【选择或输入服务器名称】下拉列表框中可以选择需要连接的数据库，本实例中由于使用的是本机数据库，所以选择默认参数即可。在【输入登录服务器的信息】下，需要指定登录服务器的用户名称和密码，在【用户名称】文本框输入 SQL Server 的登录用户名称，这里使用 SQL Server 默认用户名 sa，密码为设置 SQL Server 时为 sa 所配置的密码即可，用户可以在安装 SQL Server 自行配置，也可以在装好 SQL Server 后再修改(如果用户的数据库 sa 用户密码与本书不同，可以在 SQL Server 的查询分析器中输入以下 SQL 语句来修改密码：exec sp_password ‘旧密码’，‘新密码’)。一般来讲，为了设置好 ADO 组件之后能方便地访问数据库，可以选中【允许保存密码】复选框。【在服务器上选择数据库】下拉列表框用来确定 ADO 组件访问 SQL Server 中储存有程序需要使用的表格的数据库，这里选择本实例需要使用的 jxcbook 数据库。设置好以上选项后，单击【测试连接】按钮，如果设置正确则可以看到提示连接成功的对话框，如图 5.45 所示。

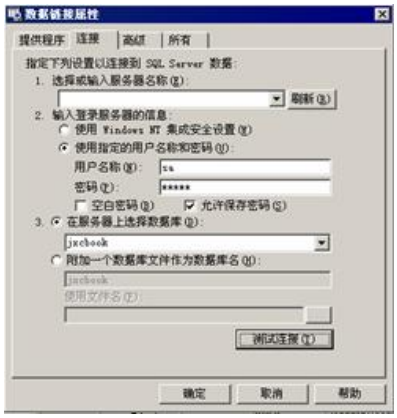


图 5.44 ADO 属性设置——步骤 4

图 5.45 ADO属性设置——测试成功

(6) 在成功设置 ADOQuery 的 ConnectionString 属性后，就可以利用它来完成【用户登录】窗体的用户和密码校验功能了。双击【用户登录】窗体中的【登录】按钮，为它添加事件处理代码，完

成用户登录功能。

```
//-------------------------输入编号自动显示用户名-----------
procedure Tlogin.Edit3Exit(Sender: TObject);
begin
  adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select 姓名 from 用户清单
where 用户编号='''+edit3.Text+'''');
 adoquery5.Open;
 edit5.Text:=adoquery5.fieldbyname('姓名').AsString;
end;
//---------------------输入编号后回车，跳到帐号输入框------------
procedure Tlogin.Edit3KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then { 判断是否按回车键}
//实际上也可以调用 edit3 的 onexit 方法来实现，是一样的
 begin
 adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select 姓名 from 用户清单
where 用户编号='''+edit3.Text+'''');
 adoquery5.Open;
 edit5.Text:=adoquery5.fieldbyname('姓名').AsString;
 end;
end;

//------------光标在密码框上时可按 Enter 键执行查询-----------
procedure Tlogin.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then //判断是否是按 Enter 键
button5.Click;
end;

//-----------------用户名和密码校验登录系统-----------------------
procedure Tlogin.Button1Click(Sender: TObject);
var
num,user,pass,temp:string;

begin
//------------------根据用户输入的密码和帐号进行查询----------
 //保存用户输入的帐号和密码
 user:=edit5.Text;
 pass:=Edit2.Text;
//使用 ADOQuery1 组件时要养成好的习惯，应先将其关闭，将原来的 SQL 语句清除后再添加新的 SQL 语句
```

```
ADOQuery5.Close;
ADOQuery5.SQL.Clear;
ADOQuery5.SQL.Text:='select 用户编号 from 用户清单
where 姓名='''+user+'''and 密码='''+pass+'''';
Adoquery5.Open;
temp:=adoquery5.FieldByName('用户编号').AsString;
if temp<>'' then
//如果输入了正确的用户名和密码，那么必然能查到用户编号，否则结果为空
begin

//--------根据用户获得的权限来确定能否操作相关窗体-----------------------
//先将所有按钮设置为不可用
  manage.Button5.Enabled:=false;
  manage.Button2.Enabled:=false;
  manage.Button3.Enabled:=false;
  manage.Button4.Enabled:=false;
  manage.Button5.Enabled:=false;
  info.N2.Enabled:=false;
  info.N3.Enabled:=false;
  info.N4.Enabled:=false;

//--------检查用户是否具有进货单权限，如是，则将【进货单】按钮置为可用----------
  adoquery5.Close;
  ADOQuery5.SQL.Clear;
  ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
where 用户编号='''+temp+'''and 权限名称=''进货单''';
  Adoquery5.Open;
  if adoquery5.FieldByName('权限序号').AsString<>'' then
  manage.Button2.Enabled:=true;

//-------检查用户是否具有销售单权限，如是，则将【销售单】按钮置为可用---------------
  adoquery5.Close;
  ADOQuery5.SQL.Clear;
  ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
where 用户编号='''+temp+'''and 权限名称=''销售单''';
  Adoquery5.Open;
  if adoquery5.FieldByName('权限序号').AsString<>'' then
  manage.Button3.Enabled:=true;

//------检查用户是否具有库存查询权限，如是，则将【库存管理】按钮置为可用--------
  adoquery5.Close;
  ADOQuery5.SQL.Clear;
  ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
where 用户编号='''+temp+'''and 权限名称=''库存查询''';
  Adoquery5.Open;
  if adoquery5.FieldByName('权限序号').AsString<>'' then
```

```
    manage.Button4.Enabled:=true;

  //----------检查用户是否具有权限管理权限，如是，则将【权限管理】按钮置为可用---------
    adoquery5.Close;
    ADOQuery5.SQL.Clear;
    ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
  where 用户编号='''+temp+'''and 权限名称=''权限管理''';
    Adoquery5.Open;
    if adoquery5.FieldByName('权限序号').AsString<>'' then
    manage.Button5.Enabled:=true;

  //-------检查用户是否具有商品资料维护权限如是，则将【资料维护】按钮和【商品资料维护】
菜单置为可用-------
    adoquery5.Close;
    ADOQuery5.SQL.Clear;
    ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
  where 用户编号='''+temp+'''and 权限名称=''商品资料维护''';
    Adoquery5.Open;
    if adoquery5.FieldByName('权限序号').AsString<>'' then
    begin
    info.N2.Enabled:=true;
    manage.Button5.Enabled:=true
    end;

  //----检查用户是否具有供货商资料维护权限，如是，则将【资料维护】按钮和【供货商资料维
护】菜单置为可用
    adoquery5.Close;
    ADOQuery5.SQL.Clear;
    ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
  where 用户编号='''+temp+'''and 权限名称=''供货商资料维护''';
    Adoquery5.Open;
    if adoquery5.FieldByName('权限序号').AsString<>'' then
    begin
    info.N3.Enabled:=true;
    manage.Button5.Enabled:=true
    end;

  //-----检查用户是否具有客户资料维护权限，如是，则将【资料维护】按钮和【客户资料维护】
菜单置为可用----
    adoquery5.Close;
    ADOQuery5.SQL.Clear;
    ADOQuery5.SQL.Text:='select 权限序号 from 权限清单
  where 用户编号='''+temp+'''and 权限名称=''客户资料维护''';
    Adoquery5.Open;
    if adoquery5.FieldByName('权限序号').AsString<>'' then
    begin
```

```
    info.N4.Enabled:=true;
    manage.Button5.Enabled:=true
    end;
  //---------登录成功，在管理窗体的状态栏上标示登录的用户名----------------
   manage.Visible:=true;
   manage.StatusBar5.Panels[0].Text:=user;

   login.Hide;
  end
  //---------登录失败，提示用户重新输入-----------------------------------
   else
   begin
    ShowMessage('用户名或密码错误，请检查后重新登录');
    end
  end;
  //----------登录对话框再次显示时，清空原来输入的用户名和密码--------------
  procedure Tlogin.FormShow(Sender: TObject);
  begin
  edit5.Clear;
  edit2.Clear;
  end;
```

到此已完成用户登录的功能。

5.6.4 管理主窗体的实现

本小节将演示如何创建一个方便而美观的管理主界面以供登录成功后的用户使用。

(1) 选择 New｜Form 命令，建立一个新的窗体并将其保存为 FormManage.pas，并在其上放置相应的控件，如图 5.46 所示。

图 5.46 主窗体设置

(2) 主窗体的控件属性设置如表 5.4 所示。

表 5.4 主窗体控件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FormManage	Caption	进销存管理系统
Button	Button1	Caption	资料管理
Button	Button2	Caption	进货管理

Button	Button3	Caption	销售管理
Button	Button4	Caption	库存管理
Button	Button5	Caption	系统管理
Image	Image1	Picture	光盘中的图片
StatusBa	StarTusbar	Panels[1].t	数据库开发经典案例解
r	1	ext	析 清华大学出版社

这是一个非常简单的窗体，它的主要作用是根据用户不同的选择打开不同的功能窗体，在用户使用完相应的功能窗体后又能回到该窗体执行其他功能操作。在窗体的下部有该实例的描述和登录用户的标示，如图 5.46 所示。

(3)　为该窗体添加代码如下。

```
//---------------------管理主界面窗体关闭时返回登录界面-----------------
procedure Tmanage.FormClose(Sender: TObject; var Action: TCloseAction);
begin
login.show;
end;
//---------------------打开【进货单】窗体并隐藏主窗体-----------------
procedure Tmanage.Button2Click(Sender: TObject);
begin
input.Visible:=true;
manage.Hide;
end;
//---------------------打开【信息管理】窗体并隐藏管理窗体-----------------
procedure Tmanage.Button1Click(Sender: TObject);
begin
info.Visible:=true;
info.Caption:='资料管理';
manage.Hide;
end;
//---------------------打开【库存查询】窗体并隐藏管理窗体-----------------
procedure Tmanage.Button4Click(Sender: TObject);
begin
manage.Hide;
store.visible:=true;
end;
//---------------------打开【权限管理】窗体并隐藏管理窗体-----------------
procedure Tmanage.Button5Click(Sender: TObject);
begin
rights.visible:=true;
manage.hide;
end;
//---------------------打开【销售单】窗体并隐藏管理窗体-----------------
procedure Tmanage.Button3Click(Sender: TObject);
begin
output.visible:=true;
```

```
manage.hide;
end;
```

图 5.47 添加事件处理程序

这里需要补充说明的是，为 Button1Click 和 FormClose 两个事件添加处理程序代码的方法有所不同，前者只需要在已经布置好控件的窗体上双击相应按钮，本例中是双击【资料管理】按钮，即可添加单击的响应处理程序 Button1Click。而后者需要在 Object Inspector 的事件属性框中自行设置。要设置窗体的 FormClose 事件处理程序，首先要在窗体空白处单击，然后在左边的【对象监视器】(即 Object Inspector)中选择 Events 选项，在下面的属性框中即可看到有许多事件可供添加，选择 OnClose 右边的空白栏并双击，系统自动将 FormClose 事件添加到代码中，这样就可以自行添加该时间的处理程序了，如图 5.47 所示。

这样就设计完成了管理主界面，它在程序中主要起到一个承前启后的作用，为后面各个功能的实现提供便利条件。

5.6.5 资料管理功能的实现

本小节将演示如何连接一个表，并在程序中对这个表进行修改、插入、删除等操作，同时还将演示菜单控件的使用。这一窗体的作用主要是实现商品资料维护，供货商资料维护和客户资料维护这 3 种功能，并提供在窗体中根据供货商、客户或商品的拼音进行模糊查询的功能。

(1) 选择 New | Form 命令，建立一个新的窗体，将它保存为 FormInfo.pas，并在窗体上布置如图 5.48 所示的控件。控件清单如表 5.5 所示。

图 5.48 【资料管理】窗体的实现

表 5.5 “资料管理”窗体控件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FormInfo	Caption	资料管理
Button	Button1	Caption	查询
DBNavigator	DBNavigator1	DataSource	DataSource1
Edit	Edit1		
Label	Label1	Caption	商品拼音
Label	Label1	AutoSize	false
MainMenu	MainMenu1	Items	下面将有详细叙述
DBGrid	DBGrid1	DataSource	DataSource1

(2) 为使这些控件能正常工作，需要进一步为其设置高级属性。

要设置菜单控件的属性，双击面板上的 MainMenu 控件，或者单击 Object Inspector 中 Items 的 Menu…属性，将会弹出菜单设置窗口，在标题栏下方有虚线框的空白处单击，并设置 Object Inspector 中的 Caption 属性，即可得到所需下拉菜单或其他主菜单。本实例所设置的菜单如图 5.49 所示。

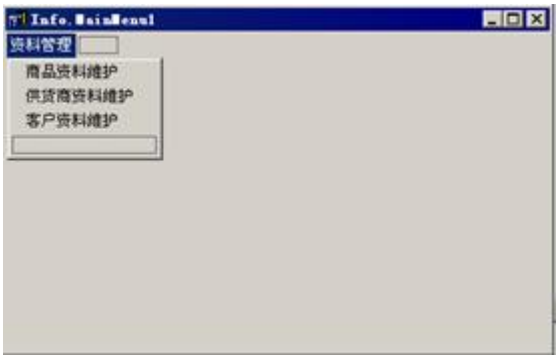


图 5.49 菜单控件的设置

(3) 设置好菜单控件的属性后，需要正确设置数据库控件的链接属性，这样才能使程序正常工作。

首先设置 ADOQuery1 控件的 ConnectionString 属性,将其正确链接到数据库上并测试是否成功。然后将 DataSource1 的 DataSet 属性指向 ADOQuery1 控件，最后将 DBGrid1 的 DataSource 属性设置为 DataSource1，这样就完成了 3 个数据库控件的链接。

本实例中只使用了一个 ADOQuery 控件、一个 DBGrid 控件、一个 DataSource 控件，然后在程序中动态指定它们的链接对象，以实现使用一套数据库控件对 3 个不同表进行查询和其他操作的功能。打开【资料管理】菜单并选择【商品资料维护】等 3 个子菜单，分别添加事件处理代码。

这样就实现了通过菜单管理资料的功能。上面在面板中加入了一个 DBNavigator 组件，它的主要作用是在数据集中进行记录导航和为用户操作数据集中的记录提供一组简单明了的控制按钮。DBNavigator 组件中包含一组控制按钮，用户单击其中的按钮可以向前(向后)移动记录指针、插入记录、修改现存记录、保存对记录的修改、取消修改、删除记录及刷新记录显示等。DBNavigator 组件如图 5.50 所示。

图 5.50 DBNavigator 组件

首记录：将当前记录指针移到数据库表中第一条记录处。

上一条：将记录指针移到当前记录的前一条记录处。

下一条：将记录指针移到当前记录的后一条记录处。

尾记录：将当前记录指针移动到数据库的最后一条记录处。

插入记录：调用数据集组件的 insert 方法，在当前记录的位置插入一条新记录。

删除当前记录：可以将 ConfirmDelete 属性设置为 true 以弹出确认删除的对话框。

编辑记录：将数据集设置为编辑状态，以便用户编辑当前记录。

提交：提交对当前记录的修改。

取消：取消对当前记录的修改。

刷新：清除数据控制组件的显示缓冲区，并用数据集组件的记录刷新。

操作表中的数据有多种方法,可以自己编写 SQL 语句,通过与数据库链接的数据库控件进行插入、修改、删除等，也可以直接使用 DBNavigator 组件实现上述功能。本实例中为简单起见，对一些简单的操作功能都直接使用 DBNavigator 组件来实现，在后面的程序中将会演示如何使用 SQL 语句来实现数据库操作的高级功能。例如，要对商品清单中的【百事可乐(大瓶)】的【条码】属性进行修改，只需单击图 5.51 中的箭头指示处，然后直接输入数据即可，注意到在修改过程中表的行前面会出现一个 标志，表示表中的数据正在被修改。修改数据完毕后，只需单击其他行或者单击 按钮，即可提交所做的修改。使用 DBNavigator 组件对表进行操作的界面如图 5.51 所示。

图 5.51　使用 DBNavigator 组件实现对表的操作

在打开一个表之后，可以利用查询功能来查询 其中的某条记录。由于实际使用过程中不可能也没有必要记住某个供货商或客户的编号，因此可以直接使用他们的拼音编码来进行查询。同时，人的记忆并不是百分 之百精确，因此在查询过程中使用模糊查询方法更有用，它可以从表中查找到与用户输入的拼音相近似的所有记录。

小知识：SQL 中的模糊查询

SQL 中如果使用“=”，“!=”等来确定给定字符串是否与指定的模式匹配。模式可以包含常规字符和通配符字符。模式匹配过程中，常规字符必须与字符串中指定的字符完全匹配。但可使用字符串的任意片段匹配通配符。与使用 =和 !=相比，使用通配符可使 LIKE 运算符更加灵活。如果任何参数都不属于字符串数据类型，那么在使用 LIKE 进行字符串比较时，模式字符串中的所有字符都有意义，包括起始或尾随空格。如果查询中的比较要返回包含“abc ”(abc 后有一个空格)的所有行，则将不会返回包含“abc”(abc 后没有空格)的列所在行。但是可以忽略模式所要匹配的表达式中的尾随空格。如果查询中的比较要返回包含“abc”(abc 后没有空格)的所有行，则将返回以“abc”开始且具有零个或多个尾随空格的所有行。如本例中，可以在用户输入的拼音编码前后都加入通配符“%”，输出所有与用户输入有相近拼音编码的数据记录。在后面的实例中，我们将讲到如何利用 Delphi 中的字符处理函数来更方便地完成相应的功能。

(4)　双击图 5.51 所示界面中的【查询】按钮，为按钮添加下面的事件处理程序，实现对所使用的表的动态判断和模糊查询。整个信息管理窗体的代码如下。

```
//----------------单击【商品信息管理】菜单，显示商品清单-----------------
procedure TInfo.N2Click(Sender: TObject);
begin
//查询商品信息
adoquery5.Close;
adoquery5.SQL.Clear;
adoquery5.SQL.Text:='select * from 商品清单';
adoquery5.Open;
//设置列的宽度，方便用户查看
dbgrid5.Columns[2].Width:=80;
dbgrid5.Columns[3].Width:=120;
dbgrid5.Columns[4].Width:=80;
dbgrid5.Columns[6].Width:=80;
//提示用户输入商品拼音以供查询
label5.Caption:='商品拼音';
```

```
end;

//---------------单击【供货商信息管理】菜单，显示供货商清单----------------
procedure TInfo.N3Click(Sender: TObject);
begin
//查询供货商信息
adoquery5.Close;
adoquery5.SQL.Clear;
adoquery5.SQL.Text:='select * from 供货商清单';
adoquery5.Open;
//设置列的宽度，方便用户查看
dbgrid5.Columns[1].Width:=80;
dbgrid5.Columns[2].Width:=80;
dbgrid5.Columns[3].Width:=80;
dbgrid5.Columns[4].Width:=120;
//提示用户输入供货商拼音以供查询
label5.Caption:='供货商拼音';
end;
//---------------单击【客户信息管理】菜单，显示客户清单----------------
procedure TInfo.N4Click(Sender: TObject);
begin
//查询客户信息
adoquery5.Close;
adoquery5.SQL.Clear;
adoquery5.SQL.Text:='select * from 客户清单';
adoquery5.Open;
//设置列的宽度，方便用户查看
dbgrid5.Columns[1].Width:=80;
dbgrid5.Columns[2].Width:=80;
dbgrid5.Columns[3].Width:=80;
dbgrid5.Columns[4].Width:=80;
dbgrid5.Columns[5].Width:=120;
//提示用户输入客户拼音以供查询
label5.Caption:='客户拼音';
end;

//----------------窗体关闭时返回的窗体选择---------------------------
procedure TInfo.FormClose(Sender: TObject; var Action: TCloseAction);
begin
manage.show;
//如果此时是从进货单功能中调用【信息管理】窗体，则不返回主窗体而返回【进货单】窗体
if (input.Enabled=false)  then
 begin
 manage.Hide;
 input.enabled:=true;
end
```

```
end;

//------------窗体显示时，默认显示商品清单-------------------------
procedure TInfo.FormShow(Sender: TObject);
begin
n2.Click;
end;

//---------------------实现查询功能--------------------------
procedure TInfo.Button1Click(Sender: TObject);
begin
//根据标签的文字进行判断应该对那个表进行查询
//查询商品清单中的信息
 if label5.Caption='商品拼音' then
  begin
  adoquery5.Close;
  adoquery5.SQL.Clear;
  adoquery5.SQL.Add('select * from 商品清单
where 拼音编码 like ''%'+edit5.text+'%''');
  adoquery5.Open;
  dbgrid5.Columns[2].Width:=80;
  dbgrid5.Columns[3].Width:=120;
  dbgrid5.Columns[4].Width:=80;
  dbgrid5.Columns[6].Width:=80
  end;

  //查询供货商清单中的信息
  if label5.Caption='供货商拼音' then
  begin
  adoquery5.Close;
  adoquery5.SQL.Clear;
  adoquery5.SQL.Add('select * from 供货商清单
where 拼音编码 like ''%'+edit5.text+'%''');
  adoquery5.Open;
  dbgrid5.Columns[1].Width:=80;
  dbgrid5.Columns[2].Width:=80;
  dbgrid5.Columns[3].Width:=80;
  dbgrid5.Columns[4].Width:=120
  end;

  //查询客户清单中的信息
  if label5.Caption='客户拼音' then
  begin
  adoquery5.Close;
  adoquery5.SQL.Clear;
  adoquery5.SQL.Add('select * from 客户清单
```

```
where 拼音编码 like ''%'+edit5.text+'%''');
 adoquery5.Open;
 dbgrid5.Columns[1].Width:=80;
 dbgrid5.Columns[2].Width:=80;
 dbgrid5.Columns[3].Width:=80;
 dbgrid5.Columns[4].Width:=80;
 dbgrid5.Columns[5].Width:=120
 end;
end;

//-------在【进货单】或【销售单】窗体中调用【信息管理】窗体-------------------------------
//-------实现通过单击 grid 控件向进货单或销售单中传送相应的数据---------------
procedure TInfo.DBGrid1DblClick(Sender: TObject);
var
num,name:string;
begin
//-----------------向进货单中传送商品或供货商信息------------------------
//判断是否是从进货单窗体中调用的【信息管理】窗体，以及调用的是哪个表
 if (input.Enabled=false)and(label5.Caption='供货商拼音') then
 begin
//将供货商的标号和名称传回“进货单”窗体
 num:=dbgrid5.Fields[0].AsString;
 name:=dbgrid5.Fields[3].AsString;
 input.ComboBox2.Text:=num;
 input.Edit3.Text:=name;
//成功后返回进货单
 input.Enabled:=true;
 info.Close;
 manage.Hide;
 end;

 if (input.Enabled=false)and(label5.Caption='商品拼音')  then
 begin
//将货号和商品名称传回“进货单”窗体
 num:=dbgrid5.Fields[0].AsString;
 name:=dbgrid5.Fields[3].AsString;
 if input.currentRow=0 then
 input.currentRow:=1;
 input.StringGrid5.Cells[1,input.currentRow]:=num;
 input.StringGrid5.Cells[2,input.currentRow]:=name;
//成功后返回进货单
 input.Enabled:=true;
 info.Close;
 manage.Hide;

 end;
```

```
//------------------向销售单中传送客户信息------------------
 if (output.Enabled=false)and(label5.Caption='客户拼音') then
 begin
//向将客户编号和名称传回【销售单】窗体
 num:=dbgrid5.Fields[0].AsString;
 name:=dbgrid5.Fields[3].AsString;
 output.ComboBox2.Text:=num;
 output.Edit3.Text:=name;
//成功后返回【销售单】窗体
 output.Enabled:=true;
 info.Close;
 manage.Hide;
 end;
end;
```

加入上述代码后，就可以实现程序的动态模糊查询，例如仅仅在【商品拼音】文本框中输入一个“s”，单击【查询】按钮后会显示所有拼音编码中有 s 的商品记录，并得到如图 5.52 中窗体所示的结果。如果不输入任何字符，那么查询结果将为表中的所有数据。

图 5.52　资料管理中的动态模糊查询

到此已完成了资料维护的功能，在这一功能中我们学习了如何使用 ADOQuery 组件从一个数据库中查询符合要求条件的数据，并将其显示在窗体中，还学习了数据库链接组件之间的连接，以及如何在程序中动态修改 ADOQuery 的 SQL 语句实现动态查询。需要注意的是，dbgrid 的双击操作代码是为了在操作进货单和销售单时调用本窗体来选择数据，这一点在介绍进货单和销售单时会详细讲述。

5.6.6 进货管理功能的实现

(1)　选择 New | Form 命令，建立一个新的窗体并将其保存为 FormInput.pas，并在其上放置相应的控件，如图 5.53 所示。

(2)　【进货单】窗体中的控件属性设置如表 5.6 所示。

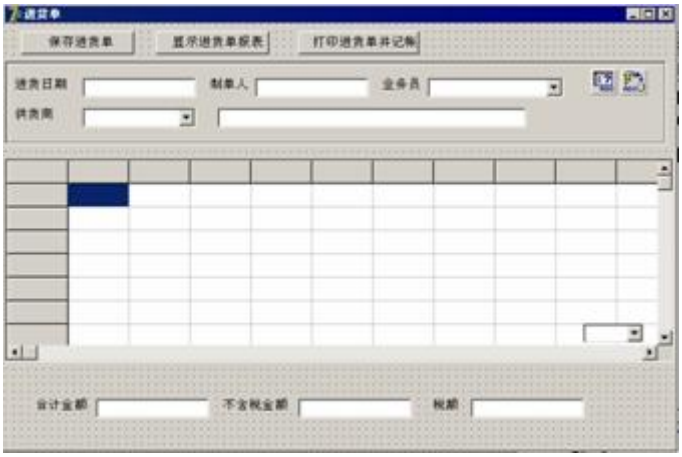


图 5.53 【进货单】窗体的控件布置

表 5.6 【进货单】窗体控件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FomInput	Caption	进货单
ListBox	Iistbox1		
Combobox	Combobox1		
Combobox	Combobox2		
Combobox	Combobox3	visible	false
StringGrid	StringGrid1	Options\goEditing	true
StringGrid	StringGrid1	RowCount	21
StringGrid	StringGrid1	ColCount	11
Button	Button1	Caption	保存进货单
Button	Button2	Caption	显示进货单报表
Button	Button3	Caption	打印进货单并记帐
Label	Label1	Caption	进货日期
Label	Label2	Caption	制单人

续表

控件类型	对 象 名	属 性	取值(说明)
Label	Label3	Caption	业务员
Label	Label4	Caption	供货商
Label	Label5	Caption	合计金额
Label	Label6	Caption	不含税金额
Label	Label7	Caption	税额
Edit	Edit1～6	Enable	将 Edit4～Edit6 共 3 个控件的 Enable 属性设为 false
ADOQuery	ADOQuery1	ConnectionString	如 5.6.4 小节所述
ADOCommand	ADOCommand1	ConnectionString	如 5.6.4 小节所述

实际进货中供货商和商品常用代号来表示，但这些号码往往没有办法记忆，因此需要按照拼音查询，并选择输入。借用前面创建的 3 个资料维护窗体来实现选择的功能。

(3) 本窗体所有的代码如下。

//设置选取的 stringgrid 单元成员的行，列值为 public，以供其他窗体中调用

```
  currentRow: integer;
  currentCol:integer;
  { Public declarations }
  end;
var
  input: Tinput;
  currentRow: integer;
  currentCol:integer;
implementation

uses FormManage, FormInfo, FormReport;

{$R *.dfm}

//-----------------关闭窗体时打开主窗体----------------------------
procedure Tinput.FormClose(Sender: TObject; var Action: TCloseAction);
begin
manage.show;
end;

//---------------窗体显示时设置当前值-------------------------
procedure Tinput.FormShow(Sender: TObject);
begin
//制表事件为当前日期
edit5.Text:=datetostr(date);
//制单人即登录用户
edit2.Text:=manage.StatusBar5.Panels[0].Text;
currentCol:=1;
currentRow:=1;
end;

//----------------动态添加供货商数据词典-------------------------
//--------【供货商】窗体中双击读入数据功能见【供货商】资料维护窗体的代码--------
procedure Tinput.ComboBox2DropDown(Sender: TObject);
begin
//显示供货商清单
 info.visible:=true;
 info.N3.Click;
//显示供货商清单时，不允许对【进货单】窗体进行操作
 input.Enabled:=false;
end;

//----------双击货号列读入商品信息----------------------
procedure Tinput.StringGrid1DblClick(Sender: TObject);
begin
//只允许在第一列由“(双击)”标示处双击
```

```
    if currentCol=1 then
    begin
    //显示商品清单
    info.Visible:=true;
    info.N2.Click;
    //显示商品清单时，不允许对【进货单】窗体进行操作
    input.Enabled:=false;
    end;
    end;

    //------------------设置选取的单元行、列值--------------------------
    //-----------------添加单价、数量等信息并根据这些信息进行计算---------
    //-------------------将统计结果显示在窗体下方的文本框中-----------------
    procedure Tinput.StringGrid1SelectCell(Sender: TObject; ACol,
      ARow: Integer; var CanSelect: Boolean);
    var
     price,number,tax,sum:double;
     j:integer;
    begin
    //读取选取的当前单元的行、列值
     currentCol:=ACol;
     currentRow:=ARow;
    //只有当选取仓库时，动态下拉列表才显示
     if (currentCol<>5)   then
      combobox3.Visible:=false;
    //只有商品信息不为空时才可以输入单位和默认税率
     if (currentCol=3)and(stringgrid5.Cells[2,currentRow]<>'')  then
       begin
      adoquery5.Close;
     adoquery5.SQL.Clear;
     adoquery5.SQL.Add('select  distinct  单 位   from  商 品 清 单   where  货 号
='''+stringgrid5.Cells[1,currentRow]+'''');
     adoquery5.Open;
     stringgrid5.Cells[3,currentRow]:=adoquery5.FieldByName('单位').AsString;
     stringgrid5.Cells[8,currentRow]:='17';
       end;

     //-----------当商品信息、数量、单价不为空时，才能进行计算----------------
                                                                          if
(currentRow<>0)and(stringgrid5.Cells[1,currentRow]<>'')and(stringgrid5.Cells[4,currentR
ow]<>'')and(stringgrid5.Cells[6,currentRow]<>'')and(stringgrid5.Cells[5,currentRow]<>'')
then
       begin
     //---------计算某条进货单明细的进货金额、税额和不含税金额-----------------------
       number:=strtofloat(stringgrid5.Cells[4,currentRow]);
       price:=strtofloat(stringgrid5.Cells[6,currentRow]);
```

```
  tax:=strtofloat(stringgrid5.Cells[8,currentRow])/100;
  stringgrid5.Cells[7,currentRow]:=floattostr(number*price);
   stringgrid5.Cells[9,currentRow]:=floattostr(tax*number*price/(1+tax));
  stringgrid5.Cells[10,currentRow]:=floattostr(number*price/(1+tax));
  //-----------------统计合计金额-----------------------
  sum:=0;
  for j:=1 to 20 do
  if  stringgrid5.Cells[7,j]<>'' then
   sum:=sum+strtofloat(stringgrid5.Cells[7,j]);
  edit4.Text:=floattostr(sum);
  //-----------------统计合计税额-----------------------
  sum:=0;
  for j:=1 to 20 do
  if  stringgrid5.Cells[9,j]<>'' then
   sum:=sum+strtofloat(stringgrid5.Cells[9,j]);
  edit6.Text:=floattostr(sum);
  //-----------------统计合计不含税额-----------------------
  sum:=0;
  for j:=1 to 20 do
  if  stringgrid5.Cells[10,j]<>'' then
   sum:=sum+strtofloat(stringgrid5.Cells[10,j]);
  edit5.Text:=floattostr(sum);

end;
end;

//--------窗体创建时，设置 stringgrid 的行列属性、宽度等信息--------------
//--------同时为业务员添加数据词典----------------------------------------
procedure Tinput.FormCreate(Sender: TObject);
var
i,j,num:integer;
begin
//设置 stringgrid 行、列名称
stringgrid5.Cols[0].Add('序号');
stringgrid5.ColWidths[0]:=32;
stringgrid5.Cols[1].Add('货号(双击)');
stringgrid5.ColWidths[1]:=80;
stringgrid5.Cols[2].Add('品名');
stringgrid5.ColWidths[2]:=128;
stringgrid5.Cols[3].Add('单位');
stringgrid5.ColWidths[3]:=32;
 stringgrid5.Cols[4].Add('数量');
 stringgrid5.ColWidths[4]:=64;
stringgrid5.Cols[5].Add('仓库');
stringgrid5.ColWidths[5]:=48;
 stringgrid5.Cols[6].Add('单价');
```

```
 stringgrid5.ColWidths[6]:=64;
stringgrid5.Cols[7].Add('金额');
stringgrid5.ColWidths[7]:=64;
 stringgrid5.Cols[8].Add('税率');
 stringgrid5.ColWidths[8]:=64;
stringgrid5.Cols[9].Add('税额');
stringgrid5.ColWidths[9]:=64;
stringgrid5.Cols[10].Add('不含税额');
stringgrid5.ColWidths[10]:=64;

for i:=1 to 20 do
 begin
 stringgrid5.Rows[i].Add(inttostr(i));
 stringgrid5.RowHeights[i]:=20;
 end;

//-------------------------设置业务员数据词典-------------------------
adoquery5.Close;
adoquery5.SQL.Add('select 姓名 from 用户清单 where 姓名!=''sys''');
adoquery5.Open;
combobox5.Clear;
while not adoquery5.Eof do
begin
 combobox5.Items.Add(adoquery5.fieldbyname('姓名').AsString);
 adoquery5.Next;
end;
end;

//----------------在选取仓库条件满足时，显示【仓库名】下拉列表--------------
//-------------------并设置仓库名数据词典-------------------------------
procedure Tinput.StringGrid1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin

//------------------【仓库名】下拉列表框显示在鼠标指针附近-----------------
if (currentCol=5)and(stringgrid5.Cells[2,currentRow]<>'')  then
  begin
  combobox3.Visible:=true;
  combobox3.Left:=X;
  combobox3.Top:=Y+120;

//------------------------添加数据词典-----------------------
 adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select distinct 仓库名 from 仓库清单');
 adoquery5.Open;
```

```
 combobox3.Items.Clear;
 while not adoquery5.Eof do
  begin

  combobox3.Items.Add(adoquery5.FieldByName('仓库名').AsString);
  adoquery5.Next;
  end;

  end;
end;


//---------------将选取的值添加到 stringgrid 单元，并隐藏列表框---------------
procedure Tinput.ComboBox3Select(Sender: TObject);
begin
//只有鼠标单击“仓库”列时，才会显示列表框
if currentCol=5 then
begin
 stringgrid5.Cells[5,currentRow]:=combobox3.Text;
 combobox3.Visible:=false;
 combobox3.Items.Clear;
end;
end;

//---------------预览报表--------------------------------
procedure Tinput.Button2Click(Sender: TObject);
begin
//-----------------先刷新报表中的数据源-----------------------
report.ADOQuery5.Active:=false;
report.ADOQuery5.Active:=true;
report.QuickRep5.Preview;
end;

//-----------------保存进货单、进货单明细数据--------------------------
procedure Tinput.Button1Click(Sender: TObject);
var
maxnum,maxnum2,temp,inputnum,inputnum2:string;
newnum,newnum2,i:integer;
begin
 //---------------为新插入的进货单和进货单明细数据计算编号-------------
//计算进货单中最大编号，以便在插入新的进货单数据时编号不会发生冲突
  adoquery5.Close;
  adoquery5.SQL.Clear;
  adoquery5.SQL.Add('select max(编号) 最大编号 from 进货单');
  adoquery5.Open;
  maxnum:=adoquery5.FieldByName('最大编号').asstring;
```

```
//对读出的编号进行截取并将其转换为整数值
 //防止插入第一条记录时出错
   if (maxnum='')or(maxnum=' ') then
  temp:='00000'
  else
  temp:=copy(maxnum,1,5);

//新插入的进货单编号为插入前的最大编号+1
  newnum:=strtoint(temp)+1;
//重新组合编码
if length(inttostr(newnum))=1 then
 inputnum:='0000'+inttostr(newnum);
if length(inttostr(newnum))=2 then
 inputnum:='000'+inttostr(newnum);
if length(inttostr(newnum))=3 then
 inputnum:='00'+inttostr(newnum);
if length(inttostr(newnum))=4 then
 inputnum:='0'+inttostr(newnum);
if length(inttostr(newnum))=5 then
 inputnum:=inttostr(newnum);

 //计算进货单明细最大编号以便在插入新的进货单明细数据时编号不会发生冲突
  adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select max(编号) 最大编号 from 进货单明细');
 adoquery5.Open;
 maxnum2:=adoquery5.FieldByName('最大编号').asstring;

//对读出的编号进行截取并将其转换为整数值，防止插入第一条记录时出错
   if (maxnum2='')or(maxnum2=' ')  then
  temp:='00000'
  else
  temp:=copy(maxnum2,1,5);
  newnum2:=strtoint(temp);


//由于进货单明细数据可能有很多条，因此在下面的进货单明细循环中再编号和组合编码
//----------插入新的进货单和进货单明细----------------------------
//如果供货商号为空或者时没有进货单明细数据，则取消插入
 if (combobox2.Text='')or(edit4.Text='') then
 showmessage('供货商号不能为空，且进货单明细数据必须完整')
 else
 begin
//插入新的进货单数据
adocommand5.CommandText:='insert into 进货单([编号],[供货商号],[进货日期],[业务
```

```
员 ],[ 制 单 人 ],[ 税 价 合 计 ],[ 不 含 税 价 ],[ 税 额])  values('''+inputnum+''','''+combobox2.Text+''','''+edit5.Text+''','''+combobox5.
Text+''','''+edit2.Text+''','''+edit4.Text+''','''+edit5.Text+''','''+edit6.
Text+''')';
      adocommand5.Execute;
    //根据进货单明细条目的数量，插入进货单明细数据
      for i:=1 to 20 do
       if stringgrid5.Cells[7,i]<>'' then
       //重新组合编码
       begin
       newnum2:=newnum2+1;
       if length(inttostr(newnum2))=1 then
        inputnum2:='0000'+inttostr(newnum2);
       if length(inttostr(newnum2))=2 then
        inputnum2:='000'+inttostr(newnum2);
       if length(inttostr(newnum2))=3 then
        inputnum2:='00'+inttostr(newnum2);
       if length(inttostr(newnum2))=4 then
        inputnum2:='0'+inttostr(newnum2);
       if length(inttostr(newnum2))=5 then
        inputnum2:=inttostr(newnum2);
    adocommand5.CommandText:='insert into 进货单明细([编号],[进货单号],[货号],[进货数量 ],[ 进 价 ],[ 税 价 合 计 ],[ 税 率 ],[ 不 含 税 价 ],[ 税 额 ],[ 仓 库])  values('''+inputnum2+''','''+inputnum+''','''+stringgrid5.Cells[1,i]+''','''+str
inggrid5.Cells[4,i]+''','''+stringgrid5.Cells[6,i]+''','''+stringgrid5.Cells[7,i]+''','
''+stringgrid5.Cells[8,i]+''','''+stringgrid5.Cells[10,i]+''','''+stringgrid5.Cells[9,i
]+''','''+stringgrid5.Cells[5,i]+''')';
       adocommand5.Execute;
       end;
    //通知用户，操作成功
      showmessage('进货单及明细保存成功');
     end;

    end;

    //-----------------预览和打印报表--------------------------------------
    procedure Tinput.Button3Click(Sender: TObject);
    var
    i,j:integer;
    begin
    if application.messagebox('是否现在打印进货单报表？','确认',MB_OKCANCEL)=IDOK then
     report.QuickRep5.Print;
    adocommand5.CommandText:='exec sf_进货单';
    adocommand5.Execute;
    //清空前面的输入
    for i:=1 to 10 do
```

```
    for j:=1 to 20 do
     stringgrid5.Cells[i,j]:='';
    combobox5.Text:='';
    combobox2.Text:='';
    edit3.Clear;
    edit4.Clear;
    edit5.Clear;
    edit6.Clear;
    end;

    end;
```

在本窗体显示时，【进货日期】文本框内会自动加入系统当时日期，在【制单人】文本框中自动设置为用户的登录帐号，而【业务员】下拉列表框中则为从用户清单中获取的数据。单击【供货商】下拉列表框的下拉箭头，会调出供货商的【资料管理】窗体，如图 5.54 所示。

双击供货商【资料管理】窗体的 dbgrid 控件中任意一栏，其中的【供货商号】和【名称】两项数据将自动添加到【进货单】窗体中，如图 5.55 所示。

同样，双击 stringgrid 控件窗体中的某行的第一列，也可以调出【资料管理】窗体，同样双击其中的某一行数据，可以把它自动添加到 stringgrid 中，如图 5.56 及图 5.57 所示。

图 5.54　在进货单中调用供货商的【资料管理】窗体

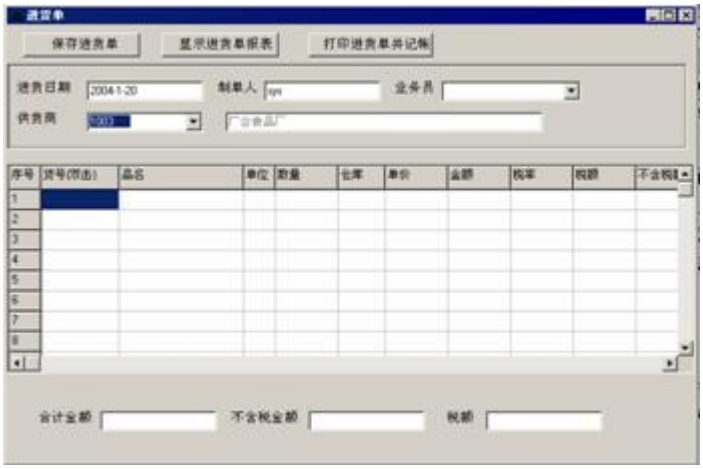


图 5.55　双击添加供货商数据

图 5.56 进货单中调用商品的【资料管理】窗体

图 5.57 双击添加商品数据

这样，可以为一个进货单添加最多 20 条进货单明细记录，实际使用时的窗体如图 5.58 所示。

图 5.58 【进货单】窗体

至此就实现了进货管理功能，在这个窗体中用到了一些相对复杂的技术，并练习了 stringgrid 的使用，用户可以结合代码体会它与 dbgrid 之间的不同。

5.6.7 编制报表程序

前面已经介绍了如何利用数据库访问、数据库控制及 ADO 等控件对数据库进行操作。利用这些组件可以编写一些数据库应用程序，提供查询、插入、修改等一系列的功能。但是，无论对数据库进行

怎样的操作，在实际使用过程中的最终目的都是为了生成一份实用、简洁而美观报表并打印出来。在 Delphi 中有现成的控件可以很好地完成这一功能，它就是 QuickReport 控件。本小节将以 5.6.6 小节所介绍的进货管理为例，介绍如何通过 QuickReport 来创建打印一个数据库报表。上面如图 5.58 的进货单生成的预览报表窗体如图 5.59 所示。

图 5.59 进货单报表预览

QuickReport 控件不是 Borland 公司自己制作的，而是挪威的 QuSoft AS 公司专门为 Delphi 设计的用于制作报表的一组控件。通过 QuickReport 控件可以很方便地把报表和代码联系起来，生成美观的图文报表。可以用 QReport 页上的控件给一个报表添加标题、页眉、页脚和概要等；还可以为各种数据源设计报表，如 Ttable、TQuery 等；还可以自动进行求和、计算均值等统计功能。

要使用 QuickReport 控件，首先要在 Delphi 中添加 QuickReport 控件包。

(1) 选择 Delphi 主菜单的 Project | Options 命令，在 Packages 选项卡的 Design packages 列表中如果可以看到 QuickReport Components 选项，那么说明 QuickReport 控件组已经被添加到 Delphi 中，如图 5.60 所示。如果没有，那么需要单击此窗体中的 Add 按钮，手动从 Delphi 安装目录 \Borland\Delphi7\Bin\dclqrt70.bpl 中添加 QuickReport 控件组，如图 5.61 所示。

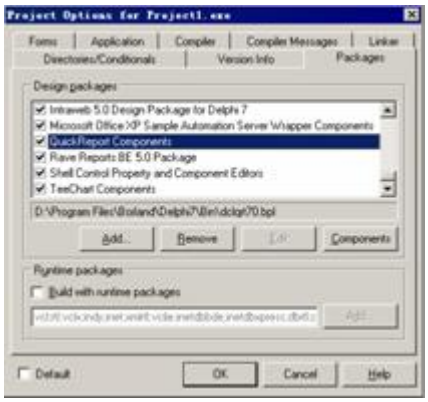


图 5.60 设置控件包 Quick Report

图 5.61　手动添加 QuickReport

(2)　加入 QuickReport 控件组后，在组件面板上将会有一个名为 QReport 的页面，它包含了所有 QuickReport 系列的控件，如图 5.62 所示。

图 5.62　所有 QuickReport 控件

QuickReport 控件组生成的报表是一个表带式的报表。表带就是指报表上的一个区域，在这个区域上可以利用组件来创建文本，图形等对象。表带是用来存放其他组件的容器，它将文本和图形带入报表中。

(3)　在 Delphi 中，生成报表最简单的方法就是使用 QuickReport Wizzard。启动 Delphi 后，在菜单栏中选择 File | New | Other 命令，在打开的 New Items 对话框中打开 Business 选项卡，双击 Quick Report Wizzard，在弹出的对话框中单击 Start Wizzard 按钮，即可进入生成报表的向导，然后根据向导提示操作即可，如图 5.63 所示。

图 5.63　选择 QuickReport Wizzard 向导

为了更好地说明 QuickReport 的使用，下面将从头开始重新设计，在一个报表中同时打印进货管理中【进货单】和【进货单明细】两个表的内容。

新建一个窗体，把它保存为 FormReport.pas 并加入工程，在其上放置一个 QuickRep 控件并拖到合适的位置，并放置如图 5.64 所示的控件。其中 Page Header Band 上放置两个 QRSysData 控件，Data 属性分别设置为 Date 和 PageNumber；Title Band 上放置一个 QRLabel 控件，Caption 设置为报表的标题；Column Header Band 上放置 10 个 QRLabel 控件和 3 个 QRDBText 控件，第一行的 3 个表明后面的数据内容,第二行的 7 个是下面 Detail Band 中明细数据的列名,3 个 QRDBText 控件的 DataField 属性分别设置为控件显示的字符；在 Detail Band 中放置 7 个 QRDBText 按钮件，用于显示进货列表；在 Summary Band 中放置一个 QRExpr 控件，用于计算合计金额，将其 Expression 属性设置为“SUM(ADOQuery5.税价合计)”。这样基本完成了进货单报表的制作，在报表上右击，在弹出的快捷

菜单中选择 Preview 命令，可以直接查看打印预览的结果。

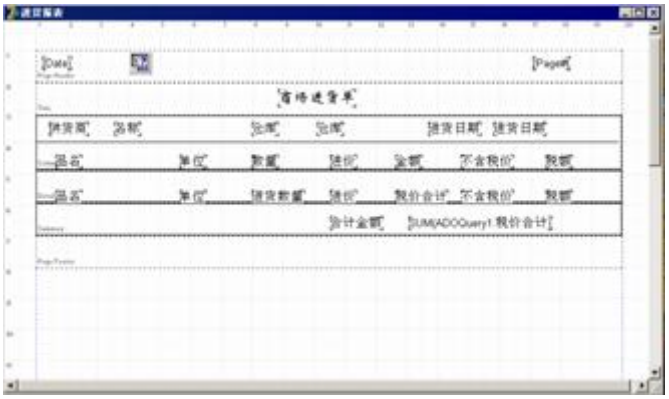


图 5.64 报表控件的布局

报表窗体中的控件属性设置如表 5.7 所示。

表 5.7 报表窗体中的其他控件清单

控件类型	对 象 名	属 性	取值(说明)
ADOQuery	ADOQuery1	ConnectionString	如前所述
ADOQuery	ADOQuery1	SQL	select a.供货商号,b.货号,a.进货日期,a.业务员,a.制单人,b.进货数量,b.进价,b.税价合计,b.税率,b.不含税价,b.税额,c.名称,d.品名,仓库,单位 from 进货单 as a,进货单明细 as b, 供货商清单 as c, 商品清单 as d where a.编号=b.进货单号 and a.供货商号=c.供货商号 and b.货号=d.货号
ADOQuery	ADOQuery1	Active	true
QuickRep	QuickRep1	DateSet	ADOQuery1
QuickRep	QuickRep1	Bands	Bands 属性下的所有子选项均设置为 true

另外，还可以为报表加上表线。首先展开 Column Header、Detail 和 Summary 这 3 个 Band 的 Frame 属性，设置 DrawBotton、DrawLeft 、DrawRight 及 DrawTop 属性为 true ，然后向各个 Band 中添加多个 QRShape 控件(直线)，移动各个控件到相应的位置即可。

到此就完成了报表的制作，调用 QuickRep 控件的 preview 函数可以生成预览，调用 print 函数可以打印报表，用户可以从前面【进货单】窗体中的【显示进货单报表】按钮和【打印进货单并记帐】按钮的代码中看到这两个函数的使用。

小知识：利用 QuickRep 制作报表的方法

(1)设置资料来源(使用 Table 或 Query 控件或 ADO 的相应控件)。

(2)设置报表的页码大小(使用 QuickRep 控件)。

(3)设置报表所含配件(Band)。

常用的 Band 有表头(Page Header Band)、标题(Title Band)、字段标题 (Column Header Band)、字段明细(Detail Band)、表尾(Page Footer Band)等。这些 Band 均可通过设置 QuickRep 控件的 Bands 属性而得到，用户无法重设其顺序。其中字段明细是用来放置字段变量，打印或打印预览时，会呈现用户表的所有记录，字段标题则是放置字段名称。

(4)设置所要打印的字段于 Detail Band。

若将报表字段对象放至非 Detail 区，则该报表字段对象只打印出第一笔记录一次。

(5)设置表头与表尾(Page Header Band 与 Page Fooder Band)。

表头与表尾可以放置一些日期、时间、记录笔数、报表页数及报表标题等资料，此项工作可由 QRSysData 控件完成，并设置 Data 属性为 qrsDate、qrsTime、rsPageNumber 或 qrsReportTitle 即可。

(6)设置报表标题(Title Band)与字段标题(Column Header Band)。

报表标题与字段标题，均可透过 QRLabel 放置一些报表标题与字段标题于标题区或字段标题区。

(7)预览打印。

单击选中 QuickRep 控件快捷菜单的 Preview 命令，即可在设计阶段预览打印结果。

5.6.8 销售管理功能的实现

销售和进货是两个类似的功能，销售是从库存 中出货，进货是向库存中添加商品。在本实例中它们所接触到的表是一一对应的，如“进货单”与“销售单”，“进货单明细”和“销售单明细”。这两者仅仅是操 作的对象和计算方法不一样，而实现方法是一样的。因此本小节将不再详细介绍销售功能的制作步骤和代码。需要注意的是：“进货单”中的货物可以选择储存仓 库，而“销售单”中的货物出货时只能从存有这个货物的仓库中出货。进货单中的进货数量在本例中没有限制，而销售单中的销售数量必须小于等于仓库中的储存数 量。对这两个个要求，在程序中都有详细的注释，请用户阅读时特别注意。

这个窗体的控件和进货管理的控件几乎完全一致，对比“进货单”与“销售单”、“进货单明细”与“销售单明细”表可以发现实体的属性是一致的，只是供货商变成了客户，进价变成了售价。从图 5.65 可以看到【销售单】窗体和【进货单】窗体几乎是一样的，运行时的操作也基本一致，因此本节不再详细叙述它的完成过程。用户可以从配书光盘中自行查阅。

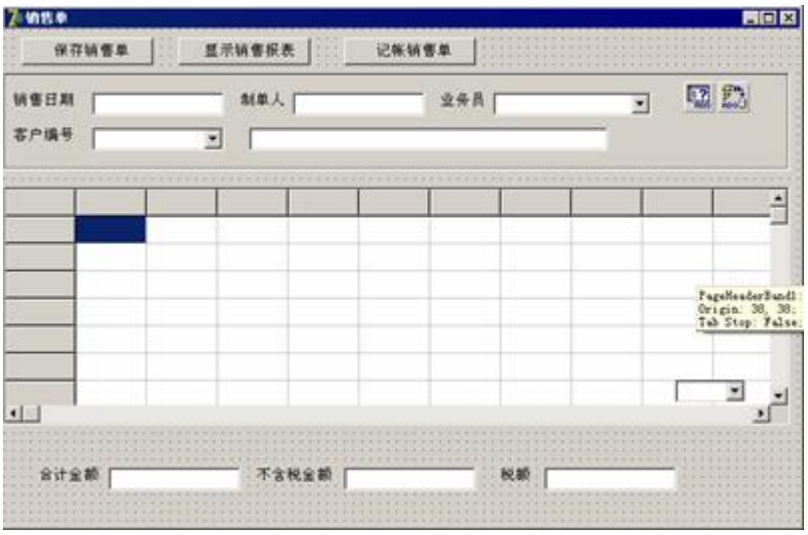


图 5.65 【销售单】窗体

(1) 新建一个窗体并保存为 FormOutput.pas。

(2) 完成【销售单】窗体的完整代码，如下所示。

```
unit FormOutput;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ADODB, DB, StdCtrls, Grids, ExtCtrls;

type
```

```
    Toutput = class(TForm)
     Label5: TLabel;
     Label6: TLabel;
     Label7: TLabel;
     Panel1: TPanel;
     Label1: TLabel;
     Label2: TLabel;
     Label3: TLabel;
     Label4: TLabel;
     Edit1: TEdit;
     Edit2: TEdit;
     ComboBox1: TComboBox;
     ComboBox2: TComboBox;
     Edit3: TEdit;
     StringGrid1: TStringGrid;
     ComboBox3: TComboBox;
     Button1: TButton;
     Edit4: TEdit;
     Edit5: TEdit;
     Edit6: TEdit;
     Button2: TButton;
     ADOQuery1: TADOQuery;
     ADOCommand1: TADOCommand;
     Button3: TButton;
     procedure FormCreate(Sender: TObject);
     procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
     procedure StringGrid1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
     procedure ComboBox3Select(Sender: TObject);
     procedure StringGrid1DblClick(Sender: TObject);
     procedure ComboBox2DropDown(Sender: TObject);
     procedure FormShow(Sender: TObject);
     procedure FormClose(Sender: TObject; var Action: TCloseAction);
     procedure Button1Click(Sender: TObject);
     procedure Button3Click(Sender: TObject);

    private
     { Private declarations }
   //-----设置选取的 stringgrid 单元成员的行、列值为 public，以供其他窗体中调用
-----------------
    public
    currentRow: integer;
    currentCol:integer;
     { Public declarations }
    end;
```

```
var
  output: Toutput;
  currentRow: integer;
  currentCol:integer;
implementation

uses FormInfo, FormManage;

{$R *.dfm}
//--------窗体创建时，设置 stringgrid 的行列属性、宽度等信息--------------
//------同时为业务员添加数据词典---------------------------------------
procedure Toutput.FormCreate(Sender: TObject);
var
i,j,num:integer;
begin
//-------------设置 stringgrid 行、列名称---------------------
stringgrid5.Cols[0].Add('序号');
stringgrid5.ColWidths[0]:=32;
stringgrid5.Cols[1].Add('货号(双击)');
stringgrid5.ColWidths[1]:=80;
stringgrid5.Cols[2].Add('品名');
stringgrid5.ColWidths[2]:=128;
stringgrid5.Cols[3].Add('单位');
stringgrid5.ColWidths[3]:=32;
stringgrid5.Cols[4].Add('数量');
stringgrid5.ColWidths[4]:=64;
stringgrid5.Cols[5].Add('仓库');
stringgrid5.ColWidths[5]:=48;
stringgrid5.Cols[6].Add('单价');
stringgrid5.ColWidths[6]:=64;
stringgrid5.Cols[7].Add('金额');
stringgrid5.ColWidths[7]:=64;
stringgrid5.Cols[8].Add('税率');
stringgrid5.ColWidths[8]:=64;
stringgrid5.Cols[9].Add('税额');
stringgrid5.ColWidths[9]:=64;
stringgrid5.Cols[10].Add('不含税额');
stringgrid5.ColWidths[10]:=64;

for i:=1 to 20 do
begin
stringgrid5.Rows[i].Add(inttostr(i));
stringgrid5.RowHeights[i]:=20;
end;
//-------------------------设置业务员数据词典------------------------
```

```
adoquery5.Close;
adoquery5.SQL.Add('select 姓名 from 用户清单 where 姓名!=''sys''');
adoquery5.Open;
combobox5.Clear;
while not adoquery5.Eof do
begin
combobox5.Items.Add(adoquery5.fieldbyname('姓名').AsString);
adoquery5.Next;
end;
end;

//------------------设置选取的单元行、列值--------------------------
//-----------------添加销售价、数量等信息并根据这些信息进行计算---------
//------------------将统计结果显示在窗体下方的文本框中-----------------
procedure Toutput.StringGrid1SelectCell(Sender: TObject; ACol,
ARow: Integer; var CanSelect: Boolean);
var
price,number,tax,sum:double;
j:integer;
begin
//读取选取的当前单元的行、列值
currentCol:=ACol;
currentRow:=ARow;
//只有当选取仓库时，动态下拉列表才显示
if (currentCol<>5) then
combobox3.Visible:=false;
//只有商品信息不为空时才可以输入单位和默认税率
if (currentCol=3)and(stringgrid5.Cells[2,currentRow]<>'')  then
begin
adoquery5.Close;
adoquery5.SQL.Clear;
adoquery5.SQL.Add('select distinct 单位 from 商品清单 where 货号一步
='''+stringgrid5.Cells[1,currentRow]+''''');
adoquery5.Open;
stringgrid5.Cells[3,currentRow]:=adoquery5.FieldByName('单位').AsString;
stringgrid5.Cells[8,currentRow]:='17';
end;

//--------当商品信息、数量、单价不为空时，才能进行计算-----------------------
if
(currentRow<>0)and(stringgrid5.Cells[1,currentRow]<>'')and(stringgrid5.Cells[4,currentR
ow]<>'')and(stringgrid5.Cells[6,currentRow]<>'')and(stringgrid5.Cells[5,currentRow]<>'')
then
  begin
//------计算某条销售单明细的进货金额、税额和不含税额--------------------------
//-------这里开始添加的代码是判断出货量的-----------------------
```

```
    //-------销售商品的数量不允许大于库存中的储存数量----------------------
     adoquery5.Close;
     adoquery5.SQL.Clear;
     adoquery5.SQL.Add('select 库 存 数 量 from 库 存 库 where 货 号
='''+stringgrid5.Cells[1,currentRow]+''''');
     adoquery5.Open;
     if strtoint(stringgrid5.Cells[4,currentRow])>adoquery5.FieldByName('库存数量').
    AsInteger then
      begin
      showmessage('该库中此种商品的最大储量小于您的输入,将按照最大储量出货');
      stringgrid5.Cells[4,currentRow]:=adoquery5.FieldByName('库存数量').AsString;
      end;

    //----------计算某条销售单明细的进货金额、税额和不含税额----------------------
      number:=strtofloat(stringgrid5.Cells[4,currentRow]);
      price:=strtofloat(stringgrid5.Cells[6,currentRow]);
      tax:=strtofloat(stringgrid5.Cells[8,currentRow])/100;
      stringgrid5.Cells[7,currentRow]:=floattostr(number*price);
      stringgrid5.Cells[9,currentRow]:=floattostr(tax*number*price/(1+tax));
      stringgrid5.Cells[10,currentRow]:=floattostr(number*price/(1+tax));

      //----------------统计合计金额----------------------
      sum:=0;
      for j:=1 to 20 do
      if stringgrid5.Cells[7,j]<>'' then
        sum:=sum+strtofloat(stringgrid5.Cells[7,j]);
      edit4.Text:=floattostr(sum);
      //----------------统计合计税额----------------------
      sum:=0;
      for j:=1 to 20 do
      if stringgrid5.Cells[9,j]<>'' then
        sum:=sum+strtofloat(stringgrid5.Cells[9,j]);
      edit6.Text:=floattostr(sum);
      //----------------统计合计不含税额----------------------
      sum:=0;
      for j:=1 to 20 do
      if stringgrid5.Cells[10,j]<>'' then
        sum:=sum+strtofloat(stringgrid5.Cells[10,j]);
      edit5.Text:=floattostr(sum);

    end;
    end;

    //---------------在选取仓库条件满足时，显示仓库名下拉列表--------------
    //-------------------并设置仓库名数据词典-------------------------------
    procedure Toutput.StringGrid1MouseDown(Sender: TObject;
```

```
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin

//------------------仓库名下拉列表框显示在鼠标指针附近-----------------
if (currentCol=5)and(stringgrid5.Cells[2,currentRow]<>'')  then
  begin
  combobox3.Visible:=true;
  combobox3.Left:=X;
  combobox3.Top:=Y+120;
 //-------------------------添加数据词典-------------------------
 //------------------注意和进货单的不同之处--------------------
 //-------------这里是根据商品的编号来查询存有这个商品的仓库-------
 adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select  distinct  仓 库  from  库 存 库  where  货 号
='''+stringgrid5.Cells[1,currentRow]+''''');
 adoquery5.Open;
 combobox3.Items.Clear;
 while not adoquery5.Eof do
  begin
  combobox3.Items.Add(adoquery5.FieldByName('仓库').AsString);
  adoquery5.Next;
  end;
  end;
end;

//----------将选取的值添加到 stringgrid 单元，并隐藏列表框--------------
procedure Toutput.ComboBox3Select(Sender: TObject);
begin
if currentCol=5 then
begin
 stringgrid5.Cells[5,currentRow]:=combobox3.Text;
 combobox3.Visible:=false;
 combobox3.Items.Clear;
end;
end;


//----------双击货号列读入商品信息----------------------
procedure Toutput.StringGrid1DblClick(Sender: TObject);

begin
//--------只允许在第一列有“(双击)”标示处双击----------
if currentCol=1 then
begin
//---------------显示库存清单--------------------------
```

```
manage.button4.click;
//-------------显示商品清单时，不允许对【进货单】窗体进行操作------
output.Enabled:=false;
end;
end;

//----------------动态添加客户数据词典-----------------------
//--------在【客户资料】窗体中双击读入数据功能见客户资料窗体的代码--------
procedure Toutput.ComboBox2DropDown(Sender: TObject);
begin
//显示客户清单
 info.visible:=true;
 info.N4.Click;
//显示客户清单时，不允许对进货单窗体进行操作
 output.Enabled:=false;

end;

//---------------窗体显示时设置当前值-------------------------
procedure Toutput.FormShow(Sender: TObject);
begin
//制表事件为当前日期
edit5.Text:=datetostr(date);
//制单人即登录用户
edit2.Text:=manage.StatusBar5.Panels[0].Text;
currentRow:=1;
currentCol:=1;
end;

//----------------关闭窗体时打开管理窗体-----------------------------
procedure Toutput.FormClose(Sender: TObject; var Action: TCloseAction);
var
i,j:integer;
begin
manage.show;
//-----------------保存销售单，销售单明细数据-------------------------
procedure Toutput.Button1Click(Sender: TObject);
var
maxnum,maxnum2,temp,inputnum,inputnum2:string;
newnum,newnum2,i:integer;
begin
 //计算“销售单”中最大编号，以便插入新的销售单数据时编号不冲突
 adoquery5.Close;
 adoquery5.SQL.Clear;
 adoquery5.SQL.Add('select max(编号) 最大编号 from 销售单');
 adoquery5.Open;
```

```
maxnum:=adoquery5.FieldByName('最大编号').asstring;

//对读出的编号进行截取并将其转换为整数值，字段太长时不能用 strtoint 来转换
//防止插入第一条记录时出错
  if (maxnum='')or(maxnum=' ') then
  temp:='00000'
  else
  temp:=copy(maxnum,1,5);
  //新插入的销售单编号为插入前的最大编号+1
  newnum:=strtoint(temp)+1;
//重新组合编码
if length(inttostr(newnum))=1 then
inputnum:='0000'+inttostr(newnum);
if length(inttostr(newnum))=2 then
inputnum:='000'+inttostr(newnum);
if length(inttostr(newnum))=3 then
inputnum:='00'+inttostr(newnum);
if length(inttostr(newnum))=4 then
inputnum:='0'+inttostr(newnum);
if length(inttostr(newnum))=5 then
inputnum:=inttostr(newnum);
//计算“销售单明细”最大编号以便插入新的销售单明细数据时编号不冲突
  adoquery5.Close;
  adoquery5.SQL.Clear;
  adoquery5.SQL.Add('select max(编号) 最大编号 from 销售单明细');
  adoquery5.Open;
  maxnum2:=adoquery5.FieldByName('最大编号').asstring;

//防止插入第一条记录时出错
  if (maxnum2='')or(maxnum2=' ')  then
  temp:='00000'
  else
  temp:=copy(maxnum2,1,5);
  newnum2:=strtoint(temp);

//由于“销售单明细”数据可能由很多条，因此在下面的“销售单明细”循环中再编号和组合编码

  //----------插入新的销售单和销售单明细------------------------------
  //如果客户号为空或者是没有销售单明细数据，则取消插入
  if (combobox2.Text='')or(edit4.Text='') then
  showmessage('客户号不能为空，且销售单明细数据必须完整')
  else
  begin
  //插入新的销售单数据
  adocommand5.CommandText:='insert into 销售单([编号],[客户编号],[销售日期],[业务
```

```
员 ],[ 制 单 人 ],[ 税 价 合 计 ],[ 不 含 税 价 ],[ 税 额])  values('''+inputnum+''','''+combobox2.Text+''','''+edit5.Text+''','''+combobox5.Text+'

    '','''+edit2.Text+''','''+edit4.Text+''','''+edit5.Text+''','''+edit6.Text+''')';
      adocommand5.Execute;
       //根据销售单明细条目的数量，插入销售单明细数据
      for i:=1 to 20 do
       if stringgrid5.Cells[7,i]<>'' then
       //重新组合编码
       begin
       newnum2:=newnum2+1;
       if length(inttostr(newnum2))=1 then
        inputnum2:='0000'+inttostr(newnum2);
       if length(inttostr(newnum2))=2 then
        inputnum2:='000'+inttostr(newnum2);
       if length(inttostr(newnum2))=3 then
        inputnum2:='00'+inttostr(newnum2);
       if length(inttostr(newnum2))=4 then
        inputnum2:='0'+inttostr(newnum2);
       if length(inttostr(newnum2))=5 then
        inputnum2:=inttostr(newnum2);
       adocommand5.CommandText:='insert into 销售单明细([编号],[销售单号],[货号],[销售数量 ],[ 销 售 价 ],[ 税 价 合 计 ],[ 税 率 ],[ 不 含 税 价 ],[ 税 额 ],[ 仓 库])  values('''+inputnum2+''','''+inputnum+''','''+stringgrid5.Cells[1,i]+''','''+stringgrid5.Cells[4,i]+''','''+stringgrid5.Cells[6,i]+''','''+stringgrid5.Cells[7,i]+''','''+stringgrid5.Cells[8,i]+''','''+stringgrid5.Cells[10,i]+''','''+stringgrid5.Cells[9,i]+''','''+stringgrid5.Cells[5,i]+''')';
        adocommand5.Execute;
        end;
      //通知用户，操作成功
       showmessage('销售单及明细保存成功');

      end;

    end;
    //------------------提交销售单-----------------------------------
    //---------报表预览及打印功能与进货单完全一致--------------------
    procedure Toutput.Button3Click(Sender: TObject);
    var
    i,j:integer;
    begin

    adocommand5.CommandText:='exec sf_销售单';
    adocommand5.Execute;
    showmessage('销售单处理成功');
```

```
//清空前面的输入
for i:=1 to 10 do
 for j:=1 to 20 do
  stringgrid5.Cells[i,j]:='';
combobox5.Text:='';
combobox2.Text:='';
edit3.Clear;
edit4.Clear;
edit5.Clear;
edit6.Clear;
end;
end;
```

5.6.9 库存查询功能的实现

“库存查询”主要用于查询商品的库存信息。这里要用到在 5.6.5 小节实现资料管理功能时曾经用到过的动态模糊查询技术，因此本小节的知识也相当简单。SQL 语句方面可以参照 5.6.5 小节的代码。新建一个窗体，并保存为 FormStore.pas。需要注意的是，窗体中显示的数据是从两张不同的表中查询得到的，因此添加静态字段时要格外注意对应关系。

(1) 在窗体上布置如图 5.66 所示的控件，并按照表 5.8 设置其属性。

图 5.66 【库存查询】窗体的控件布局

表 5.8 【库存查询】窗体中的控件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FormStore	Caption	库存查询
Label	Label1	Caption	货号
Label	Label2	Caption	商品拼音
Label	Label3	Caption	仓库
Edit	Edit1		
Edit	Edit2		
DBGrid	DBGrid1	DataSource	DataSource1
Combobox	Combobox	Items	1 库 2 库 3 库 4 库 5 库
Button	Button1	Caption	查询
Panel	Panel1	Caption	
DBGrid	DBGrid1	DataSource	DataSource1
DBGrid	DBGrid1	Columns	双击控件并添加所有字段

DataSource	DataSource1	DataSet	ADOQuery1
ADOQuery	ADOQuery1	ConnectionString	如前所述
ADOQuery	ADOQuery1	SQL	select 库存库.货号,商品清单.品名,商品清单.单位,库存库.仓库,库存库.库存数量,库存库.库存单价,库存库.库存金额 from 库存库,商品清单 where 库存库.货号=商品清单.货号
ADOQuery	ADOQuery1	Active	true

(2) 为窗体添加如下代码，完成库存查询的功能。

```
//----------------窗体关闭时返回的窗体的选择--------------------------
procedure Tstore.FormClose(Sender: TObject; var Action: TCloseAction);
begin
manage.show;
//如果此时是从销售单功能中调用【库存查询】窗体，则不返回管理主窗体而返回【销售单】窗体
if (output.Enabled=false)  then
 begin
 manage.Hide;
 output.enabled:=true;
 end
end;

//------------窗体显示时，从两张不同的表里面进行组合查询库存数据----------------
procedure Tstore.FormShow(Sender: TObject);
begin
adoquery5.Close;
adoquery5.SQL.Clear;
adoquery5.SQL.Text:='select 库存库.货号,商品清单.品名,商品清单.单位,库存库.仓库,库存库.库存数量,库存库.库存单价,库存库.库存金额 from 库存库,商品清单 where 库存库.货号=商品清单.货号';
adoquery5.Open;

end;

//-----------------------执行查询---------------------------------
procedure Tstore.Button1Click(Sender: TObject);
begin
 adoquery5.Close;
 adoquery5.sql.Clear;
 adoquery5.SQL.Add('select 库存库.货号,商品清单.品名,商品清单.单位,库存库.仓库,库存库.库存数量,库存库.库存单价,库存库.库存金额 from 库存库,商品清单');
 adoquery5.SQL.Add('where 库存库.货号=商品清单.货号 and ');
 adoquery5.SQL.Add('((商品清单.货号='''+edit5.Text+''')or');
 adoquery5.SQL.Add('(商品清单.拼音编码='''+edit2.Text+''')or');
```

```
adoquery5.SQL.Add('(库存库.仓库='''+combobox5.text+'''))');
adoquery5.Open;
//上面 SQL 语句的作用是从数据库中查询任意与输入的商品拼音、货号、仓库三者之一有相同值的记录，因此
//后面 3 个语句是或的关系，但是任何结果必须至少符合 3 个条件中的一个，所以又是一个与的关系
//这里我们没有使用前面所讲过的模糊查询，用户可以参照前面的章节自行修改程序，看看会有什么样的结果
end;

//---------------在【销售单】窗体中调用库存查询窗体-----------------------------
//-----------实现通过单击 grid 控件向销售单中传送相应的数据---------------
procedure Tstore.DBGrid1DblClick(Sender: TObject);
var
name,num:string;

begin
if (output.Enabled=false)  then
begin
 num:=dbgrid5.Fields[0].AsString;
name:=dbgrid5.Fields[1].AsString;
//将客户编号和名称传回【销售单】窗体
if output.currentRow=0 then
 output.currentRow:=1;
output.StringGrid5.Cells[1,output.currentRow]:=num;
output.StringGrid5.Cells[2,output.currentRow]:=name;
output.Enabled:=true;
//成功后返回【销售单】窗体
store.Close;
manage.Hide;
end;

end;
```

5.6.10 权限管理功能的实现

每一个系统都要对应不同的用户，而不同的用户就必然有不同的使用权限。因此，权限管理是每一个应用系统都必须具备的功能。本章将详细介绍权限管理的有关内容，以后各章的实例将不再包含该功能。

(1) 新建一个窗体，将其保存为 FormRights.pas，其控件布局如图 5.67 所示。

图 5.67 【权限管理】窗体的控件布局

(2) 按表 5.9 设置【权限管理】窗体的控件属性。

表 5.9 【权限管理】窗体控件清单

控件类型	对 象 名	属 性	取值(说明)
Form	FormRights	Caption	权限管理
Label	Label	Caption	选择要浏览权限的用户名
Label	Labe2	Caption	该用户所拥有的权限
Combobox	Combobox1		
ListBox	ListBox		
Button	Button1	Caption	确认修改
GroupBox	GroupBox1	Caption	用户权限修改
CheckBox	CheckBox1	Caption	商品资料维护
CheckBox	CheckBox2	Caption	供货商资料维护
CheckBox	CheckBox3	Caption	客户资料维护
CheckBox	CheckBox4	Caption	进货单
CheckBox	CheckBox5	Caption	销售单
CheckBox	CheckBox6	Caption	库存查询
CheckBox	CheckBox7	Caption	权限管理
ADOQuery	ADOQuery1	ConnectionString	如 5.6.4 小节所述
ADOQuery	ADOQuery2	ConnectionString	如 5.6.4 小节所述
ADOQuery	ADOQuery3	ConnectionString	如 5.6.4 小节所述
ADOQuery	ADOQuery1	SQL	select 姓名 from 用户清单

(3) 实现用户权限管理的代码如下。

```
//-----------关闭窗体时同时关闭对权限清单的插入许可功能-------------------
procedure Trights.FormClose(Sender: TObject; var Action: TCloseAction);
begin
//先关闭权限清单的插入功能
adocommand5.CommandText:='SET IDENTITY_INSERT 权限清单 off';
adocommand5.Execute;
```

```
    //关闭窗体是返回上一级窗体
    manage.show;
    end;

    //--------窗体显示时，为下拉列表框设置用户词典------------------------
    //--------不允许对管理员 sys 的权限进行设置-------------------------
    procedure Trights.FormShow(Sender: TObject);
    begin

     adoquery5.Close;
     adoquery5.SQL.Clear;
     adoquery5.SQL.Text:='select distinct 姓名 from 用户清单 where 姓名!=''sys''';
     adoquery5.Open;
     combobox5.Clear;
     while not adoquery5.Eof do
    begin
     combobox5.Items.Add(adoquery5.fieldbyname('姓名').AsString);
     adoquery5.Next;
     end
    end;

    //---------------将下拉列表中选中的用户所对应拥有的权限显示在列表框中---------
    procedure Trights.ComboBox1Change(Sender: TObject);
    var
     str,temp:string;
    begin
    adoquery2.Close;
    adoquery2.SQL.clear;
    adoquery2.SQL.Text:='select 权限名称 from 权限清单 where 用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+''''+')';
    adoquery2.Open;
    listbox5.Clear;

    while not adoquery2.Eof do
     begin
      temp:=adoquery2.fieldbyname('权限名称').AsString;
      listbox5.Items.Add(temp);
      adoquery2.Next;
      end;

    //----------------------根据用户的权限修改 checkbox 中的内容-----------
    //--------------------判断用户是否拥有商品清单的管理权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
```

```
='''+checkbox5.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox5.Checked:=true
     else
     checkbox5.Checked:=false;
    //--------------------判断用户是否拥有供货商资料的管理权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox2.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox2.Checked:=true
     else
     checkbox2.Checked:=false;
    //--------------------判断用户是否拥有客户资料的管理权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox3.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox3.Checked:=true
     else
     checkbox3.Checked:=false;
    //--------------------判断用户是否拥有进货单的管理权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox4.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox4.Checked:=true
     else
     checkbox4.Checked:=false;
    //--------------------判断用户是否拥有商品销售单的管理权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox5.Caption+''')');
    adoquery2.Open;
```

```
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox5.Checked:=true
     else
     checkbox5.Checked:=false;
    //--------------------判断用户是否拥有库存查询的管理权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+'''))and(权限名称='''+checkbox6.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox6.Checked:=true
     else
     checkbox6.Checked:=false;
    //--------------------判断用户是否拥有权限的管理的权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+'''))and(权限名称='''+checkbox7.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox7.Checked:=true
     else
     checkbox7.Checked:=false;
    end;

    //------------修改选择的用户的权限----------------------------------
     procedure Trights.Button1Click(Sender: TObject);
     var str:string;
     begin
    //获得与某个用户相对应的用户编号
    UserName:=combobox5.Text;
    adoquery3.Close;
    adoquery3.SQL.clear;
    adoquery3.SQL.Text:='select 用户编号 from 用户清单 where 姓名='''+UserName+'''';
    adoquery3.Open;
    UserNumber:=adoquery3.fieldbyname('用户编号').AsString;
    adoquery3.Close;

    //计算权限清单中的权限序号的最大值，以免其后输入的权限序号与已有的冲突
    adoquery3.Close;
    adoquery3.SQL.clear;
    adoquery3.SQL.Text:='select max(权限序号) maxnum from 权限清单 as max';
    adoquery3.Open;
```

```
MaxNumber:=adoquery3.fieldbyname('maxnum').AsString;
adoquery3.Close;

//删除该用户的所有权限记录以备下一步的修改
adocommand5.CommandText:='delete from 权限清单 where 用户编号='''+UserNumber+'''';
adocommand5.Execute;

//打开对用户权限记录表的插入权限，这是由建立数据库的时候就设置好的，用完后需要把它关闭
adocommand5.CommandText:='SET IDENTITY_INSERT 权限清单 on';
adocommand5.Execute;

//检查 checkbox1 状态，判断是否给该用户商品资料维护权限
if checkbox5.Checked then
begin
 order:=strtoint(MaxNumber)+1;
 str:=inttostr(order);

 adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名称])  values('''+str+''','+''''+UserNumber+''','+''''+checkbox5.Caption+''''+')';
 adocommand5.Execute;
end;

 //检查 checkbox2 状态，判断是否给该用户供货商资料维护权限
if checkbox2.Checked then
begin
 order:=strtoint(MaxNumber)+2;
 str:=inttostr(order);
adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名称])  values('''+str+''','+''''+UserNumber+''','+''''+checkbox2.Caption+''''+')';
 adocommand5.Execute;
end;

 //检查 checkbox3 状态，判断是否给该用户客户资料维护权限
if checkbox3.Checked then
begin
 order:=strtoint(MaxNumber)+3;
 str:=inttostr(order);

 adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名称])  values('''+str+''','+''''+UserNumber+''','+''''+checkbox3.Caption+''''+')';
 adocommand5.Execute;
end;

  //检查 checkbox4 状态，判断是否给该用户进货单访问权限
if checkbox4.Checked then
```

```
    begin
     order:=strtoint(MaxNumber)+4;
     str:=inttostr(order);

     adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名
称])   values('''+str+''','+''''+UserNumber+''','+''''+checkbox4.Caption+''''+')';
     adocommand5.Execute;
    end;

    //检查 checkbox5 状态，判断是否给该用户销售单访问权限
    if checkbox5.Checked then
    begin
     order:=strtoint(MaxNumber)+5;
     str:=inttostr(order);

     adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名
称])   values('''+str+''','+''''+UserNumber+''','+''''+checkbox5.Caption+''''+')';
     adocommand5.Execute;
    end;

    //检查 checkbox6 状态，判断是否给该用户库存查询权限
    if checkbox6.Checked then
    begin
     order:=strtoint(MaxNumber)+6;
     str:=inttostr(order);

     adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名
称])   values('''+str+''','+''''+UserNumber+''','+''''+checkbox6.Caption+''''+')';
     adocommand5.Execute;
    end;

     //检查 checkbox7 状态，判断是否给该权限“管理权限”
    if checkbox7.Checked then
    begin
     order:=strtoint(MaxNumber)+7;
     str:=inttostr(order);

     adocommand5.CommandText:='insert into 权限清单([权限序号],[用户编号],[权限名
称])   values('''+str+''','+''''+UserNumber+''','+''''+checkbox7.Caption+''''+')';
     adocommand5.Execute;
    end;

    //通知用户修改成功
      showmessage('修改成功!');

    //-------------根据用户的权限修改同步更新 checkbox 中的内容-----------
```

```
    //--------------------判断更新后用户是否拥有商品清单的管理权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox5.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox5.Checked:=true
     else
     checkbox5.Checked:=false;
    //--------------------判断更新后用户是否拥有供货商资料维护权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox2.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox2.Checked:=true
     else
     checkbox2.Checked:=false;
    //--------------------判断更新后用户是否拥有客户资料维护权限--------------
    adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox3.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox3.Checked:=true
     else
     checkbox3.Checked:=false;
    //--------------------判断更新后用户是否拥有进货单管理权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号
from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称
='''+checkbox4.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox4.Checked:=true
     else
     checkbox4.Checked:=false;
    //--------------------判断更新后用户是否拥有销售单管理权限--------------
     adoquery2.Close;
```

```
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称='''+checkbox5.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox5.Checked:=true
     else
     checkbox5.Checked:=false;
    //--------------------判断更新后用户是否拥有库存查询权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称='''+checkbox6.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox6.Checked:=true
     else
     checkbox6.Checked:=false;
    //--------------------判断更新后用户是否拥有权限管理权限--------------
     adoquery2.Close;
    adoquery2.SQL.Clear;
    adoquery2.SQL.Add('select 用户编号 from 权限清单 where (用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+'''))and( 权限名称='''+checkbox7.Caption+''')');
    adoquery2.Open;
    if adoquery2.FieldByName('用户编号').AsString<>'' then
     checkbox7.Checked:=true
     else
     checkbox7.Checked:=false;

    //-------------刷新更新后的列表框中的权限清单------------

    adoquery2.Close;
    adoquery2.SQL.clear;
    adoquery2.SQL.Text:='select 权限名称 from 权限清单 where 用户编号=(select 用户编号 from 用户清单 where 姓名='''+combobox5.Text+''''+')';
    adoquery2.Open;
    listbox5.Clear;
    while not adoquery2.Eof do
     begin
      listbox5.Items.Add(adoquery2.fieldbyname('权限名称').AsString);
      adoquery2.Next;
      end;
    end;
```

在实际使用中，管理员的权限是不允许修改的，因此在【选择要浏览的用户名】下拉列表框中是没有 sys 的，这和实际情况也符合。同时，对一个用户的权限所做的任何修改，都应该在【该用户所拥有的权限】列表框和【用户权限修改】选项组中反映出来，上面的代码很好地完成了目的。实际运行的结果如图 5.68 所示。

图 5.68 权限管理

根据权限管理的结果，可以在【用户登录】对 话框中对用户进行识别，登录系统后将该用户不具备的功能菜单变成灰色，成为不可操作菜单，从而控制用户的操作权限。如用户“王五”所拥有的权限为【客户资 料维护】、【销售单】和【库存查询】，那么用“王五”这个用户登录系统后，看到的管理界面将如图 5.69 所示。

图 5.69 受限用户的界面

5.7 系统发布

系统设计完成后要打包发布，需将应用系统制作成安装程序。制作安装程序的工具有很多，一般 Delphi 开发的应用系统可以使用 Install Shield 来完成系统的安装查询。Delphi 的安装光盘内就已经带有专门为 Delphi 量身定做的 Install Shield，用户在安装 Delphi 时一并安装即可，将 Delphi 安装光盘放入光驱后，安装界面如图 5.70 所示上将有 InstallShield 的安装提示出现。

图 5.70 Delphi 7 的安装界面

在制作安装程序时，用户就能发现我们一直坚持使用 ADO 数据链接组件而未使用 BDE 的好处。由于 ADO 和 SQL Server 以及 Windows 都是 Microsoft 公司的产品，程序运行时，ADO 控件在 Windows 系统的支持下直接与 SQL Server 进行链接，而不需要再通过 Borland 公司的 BDE 引擎，也就无须再安装 Borland 公司的动态链接库。因此只要 SQL Server 数据库设置正确，就可以直接将程序的.exe 文件复制到计算机中直接运行，不管该计算机中是否安装了 BDE 都是如此。

5.8 系统扩展

5.8.1 系统功能扩展

以上的实例演示了进销存系统基本功能设计，用户可以在此基础上利用随书光盘中提供的数据库脚本添加需求分析中的其他功能。例如销售退货功能和应付款管理功能的流程图分别如图 5.71 和图 5.72 所示。

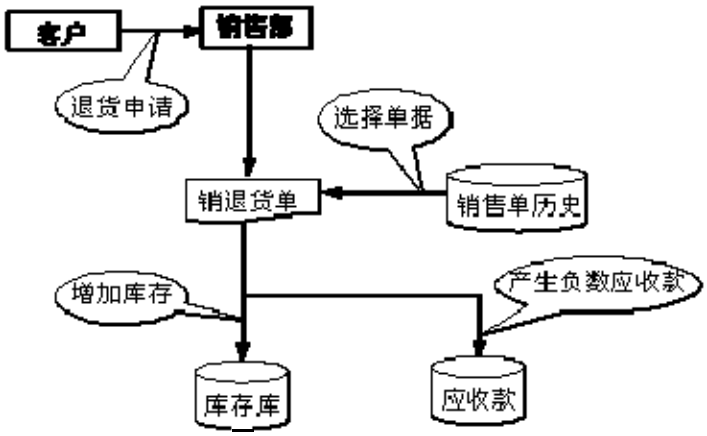


图 5.71 销售退货流程图

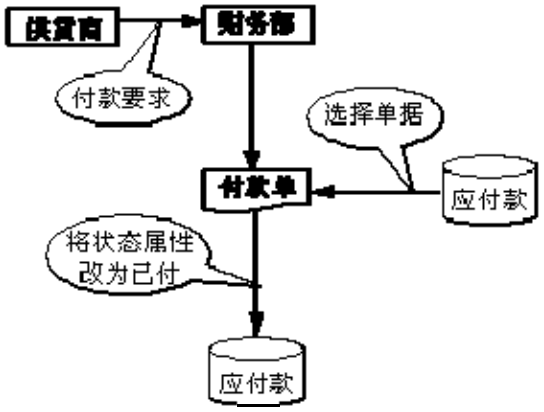


图 5.72 应付款管理流程图

5.8.2 系统向医药行业扩展

以上分析和演示的都是通用进销存系统，但往往在实际应用中不同的行业对系统的要求是不一样的，特别是一些特殊的行业，例如医药行业和服装行业。这两小节我们将向用户介绍这些行业的特点。

医药行业因为是与百姓性命攸关的特殊行业，因此国家药品监督管理局对医药流通企业有非常严格的要求，特别是近两年推出的 GSP 认证，将医药企业的管理大大提升了一个台阶。GSP(Good Supply Practice，良好供应规范)是 控制医药商品流通环节所有可能发生质量事故的因素从而防止质量事故发生的一整套管理程序，医药商品在其生产、经营和销售的全过程中，由于内外因素作用，随 时都有可能发生质量问题，必须在所有这些环节上采取严格的措施，才能从根本上保证医药商品的质量。因此，许多国家制定了一系列法规来保证药品质量，在实验 室阶段实行 GLP，新药临床阶段实行 GCP，在生产阶段实行 GMP，在流通阶段实行 GSP。根据 GSP 的要求，对药品的停售应有控制、首次经营药品必须有审批控制、首次经销药品企业应通过资质审批、药品应有有效期控制、药品检验入库应自动生成检验记录、药品能自动建档、建立药品养护、药品催销等功能。具体来说，医药行业进销存系统有以下最为重要的特性。

1. 商品的属性(商品清单表字段)

医药行业商品即药品，因其特殊性，在商品清单中会增加以下属性：剂型、处方类型、批发价、零售价、批准文号、商标、大包装单位、大包装数量、中包装单位、中包装数量、化学名、化学名拼音、GMP 认证、保质期、功能主治、储存条件等。

其中的每一个属性都会对系统产生影响。例 如，药品分为处方药和非处方药，在药品的属性中必须标明。又如药品有批发价和零售价的概念，批发价和零售价都是国家规定的价格，药品销售时一般批发销售都 是在批发价的基础上进行折扣，而零售一般都是按照零售价销售，当然现在出现的很多平价药店，也有低于零售价销售的；又如药品一般都有其化学名，在商品属性 中也需要指定，因为现在国家经常按照药品的成分(化学名)对药品进行降价，如果商品属性中没有标明商品的化学名，系统查询起来很不方便，同时化学名也是 GSP 要求的条款之一。

2. 批号、效期的管理

GSP 对药品批号、效期的管理有严格的要求，从进货、库存到销售都要按照效期严格管理。6 个月内到期的药品要催销、3 个 月内到期的药品要办退货及过期的药品禁止销售。因此信息系统需要能够将库存中的药品按照批号来管理，即输入进货单时要求输入药品的批号和效期；“库存库” 表中要增加“批号”字段，不同批号的商品需要不同记录来存放；开销售单时要指定销售的批号并严格按照开票的批号出库；系统应具有近效期药品的自动报警功 能。

3. GSP 流程管理

如上所述，GSP 要求企业经营的所有业务数据必须形成严格的记录，并至少保存 3 年以供查询，包括药品采购记录、药品入库验收记录、药品销售记录、药品退货记录、近效期药品催销表、药品报告记录、库存温湿度记录等。

5.8.3 系统向服装行业扩展

服装行业有两个最重要的行业特点。

1. 款式颜色尺寸的综合管理

服装行业的款式管理与颜色尺码的管理，与通常意义上的产品管理有很大的区别。

由于每件服装都是以款式、颜色、尺码定义的，使服装的经营管理很复杂。大量的款式、颜色、尺码管理使运作的复杂性成指数倍地上涨。许多服装企业每天要处理成百上千的款式、颜色、尺码。在这种复杂的经营管理中，精确的预测和分销管理就显得尤其重要。

产品开发对于服装行业是重要的一面，其取决于设计师的优秀设计与对市场销售的分析、预测。有些服装企业只核算商品的品名和货号，并不管理商品的颜色、款式、规格、版型。这样，产品开发就存在一定的盲目性。

服装企业要求管理款式、颜色、尺码的特点需要管理软件必须能够对服装进行款式、颜色、尺码的管理。普通的通用进销存系统是无法达到服装企业对款式、颜色、尺码的管理要求的。因为同样款式的服装一般商品编码(货号)是一样的，但它有很多种颜色和尺寸，这样在填写进货单时系统必须可以按照表格的形式输入同一商品不同颜色、不同尺寸的进货数量，库存和销售也应该分开管理。

2. 流行性和季节性

服装具有典型的流行性和季节性，并且随着市场竞争的加剧，服装新款式的准备期越来越长，而流行期却越来越短。在服装行业中，抢先生产出市场需要的产品是一个服装企业成为市场领导者的关键性因素，因此时机的选择和把握就成为了服装企业的关键之关键。

服装的易变性和循环性决定了服装业面临着种 种挑战，有些挑战是服装业要特殊面对的，而有些是与其他行业相同的。面对瞬息万变的市场需求，为了保持零售商的兴趣、刺激消费者购买，服装企业必须紧跟季 节更迭，不断地设计和开发新的产品线。服装企业发现，要跟上零售商的兴趣和消费者的需求，提高自身的灵活性和快速反应性的要求比以前更迫切了。

由于服装的流行性和季节性特点，企业必需要 做到“库存管理优化、信息反馈高效、市场反应灵敏”，才能在日趋激烈的市场竞争中立稳脚跟。因此，建立“小批量、多批次、多品种、快出货”的服装业现代化 经营管理模式，进一步缩短企业对于市场变化的响应时间，建立企业的快速反应体系已成为服装企业发展的必然趋势。

流行性和季节性的特点也使得公司要根据季节的变化和顾客的喜好对服装价格进行调整，实行各种促销政策。

5.9 小结

通过本章中的实例可以掌握以下知识和技巧：

- 进销存管理系统的需求。
- 关于财务成本核算、二八分析等小知识。
- 利用 Microsoft Visio 2002 绘制数据库 E-R 图。
- 利用 Microsoft Visio 2002 的“反向工程”功能生成 E-R 图。
- SQL Server 2000 数据库的创建方法。
- ADO 数据连接的创建方法。
- 利用 Delphi 进行数据库编程的多种方法。
- 利用 QuickReport 制作报表。
- 通用进销存系统向医药行业、服装行业的扩展。
- 应用系统发布的注意事项。

用户可以根据这些方法自行完成进销存管理系统的其他功能。

第 6 章 用 PowerBuilder 完成医院管理系统

6.1 医院信息化管理

医院信息系统(Hospital Information System，HIS)，指利用电子计算机和通信设备，为医院所属各部门提供病人诊疗信息和行政管理信息的收集、存储、处理、提取和数据交换的能力，并满足所有授权用户的功能需求。

医院管理信息系统(Hospital Management Information System，HMIS)的主要目标是支持医院的行政管理与事务处理业务，减轻事务处理人员的劳动强度，辅助医院管理，辅助高层领导决策，提高医院的工作效率，从而使医院能够以少的投入获得更好的社会效益与经济效益，如财务系统、人事系统、住院病人管理系统、药品库存管理系统等都属于 HMIS 的范围。

临床信息系统(Clinical Information System，CIS)的 主要目标是支持医院医护人员的临床活动，收集和处理病人的临床医疗信息，丰富和积累临床医学知识，并提供临床咨询、辅助诊疗及辅助临床决策，提高医护人员 的工作效率，为病人提供更多、更快、更好的服务。例如医嘱处理系统、病人床边系统、医生工作站室系统、实验室系统、药物咨询系统等都属于 CIS 范围。

一个完整的医院信息系统应该既包括医院管理信息系统(HMIS)，又包括临床医疗信息系统(CIS)，这是毫无疑义的。但是，无论外国还是中国，当一个医院的信息系统刚刚起步时，往往是首先建立 HMTS，这是因为：

(1) HMTS 所需要的资源较少，比较起来所需要的磁盘容量、工作站数量、网络传输能力、显示器质量均远远低于 CIS 的需求。

(2) 支持 HMTS 的计算机技术较为单纯和简单。由于 HMTS 以处理文字和数字类数据为主，较少涉及声音、图像、多媒体数据的动态传递等复杂需求，因此实现起来容易得多。

(3) CIS 在数据处理的实时性要求、相应速度、安全保密等方面一般要比 HMTS 有更苛刻的要求。

(4) 从投入与产出考虑，多数医院的决策者们均认为 HMIS 与 CIS 相比，能够使医院更直接、更明显、更迅速地获得系统的回报。就是说，以较少的投入获得较大的收益。

当然，HMIS 和 CIS 也不是截然分开的，HMIS 中常常会涉及一些病人的临床信息，特别是它所收集的病人主索引、病案首页等信息往往是 CIS 以病人为中心的临床医疗信息的基础。而 CIS 一旦建立，也往往会使 HMIS 工作得更准确和更有效率。

6.1.1 医院信息化管理的发展历史与现状

国际上医院信息化管理大致可以分为 3 个阶段。第一个十年，集中在开发医院行政管理的功能上，如财务收费管理、住院病人和门诊病人管理等，但到 1972 年 Collen 仍报告美国连一个成功的已完成的全面医院管理计算机系统都没有。第二个十年，在继续完成和实现医院管理信息计算机化的同时，开发者的努力已进入医疗信息的处理领域，如病人医疗处理系统，实验室系统等。到 1985 年，美国全国医院数据处理工作调查表明，100 张床位以上的医院 80%实现了计算机财务收费管理、70%的医院可支持病人挂号登记和行政事务管理，25%的医院有了较完整的 HIS。最后一个十年至今，研究者又把重点放在了病人床边系统(Bedside Information System)、医学影像处理系统(Picture Archiving and Communication System，PACS)、病人计算机化病案(Computer Based Patient Record，CPR)、统一的医学语言系统(Unified Medical Language System，UMLS)等方面。医院信息系统正在经历着小型化、智能化和集成化的改造过程。

刺激美国医院采用计算机系统的重要因素有：

- 为病人提供更好、更快的服务，提高医院在医疗服务市场上的竞争力，以吸引更多病人到本院就医。
- 人力工资的昂贵，迫使医院采用计算机以提高劳动效率。
- 国家和保险公司为施行预付款制度(Prospective Payment System，PPS)而要求医院及时上交复杂的有关 DRGS 的报告，迫使医院采用计算机技术。

DRGS(诊断相关分组，Diagnosis—Related Groups)自 1983 年开始在美国实行，其目的是为了控制不断上涨的老年医疗保险的费用。所谓 DRG 编码是在国际疾病分类的基础上，再参照疾病的严重程

度、个人状态和并发症做出一个编码。每一个编码有对应的费用，医疗保险单位根据这个编码付费。如果医院的费用超出了规定的费用，则由医院自己赔付，相反若低于规定费用则节约的部分归医院所有。

我国医院信息系统的研发工作，从 20 世纪 80 年代初期算起，至今也有十多年的历史，其中经历了单机单任务的阶段，多机多任务的阶段以及微机网络一体化的阶段，应该承认这期间我们有了很大进步。医院对信息的需求永远是 HIS 发展的原动力。刺激我国医院广泛采用计算机信息系统的因素主要有：

(1) 医院要强化自身的管理，逐步实现医疗价格评价与按成本收费，实现医院内按成本核算与全面财务计划，并体现在建立和优化医院内合理的分配制度上，为此每个医院都迫切需要建立自己的以财务管理为中心的医院管理信息系统。

(2) 医院要加强医疗服务质量的自我监督、自我控制的能力，为病人提供更快、更好的服务，必然要依赖于计算机信息系统。

(3) 医疗制度的改革正在把病人—医院的二元关系改变为病人—医院—保险机构—政府监督的多元关系。大量的有关病人的诊断、治疗、用药、资源消耗的信息不仅在院内而且要在许多部门之间流通、传递，这将是一件难以用手工完成的繁杂的任务。

(4) 每个医院的信息系统是全国或地区性医疗信息网络的基础，医疗信息网络即所谓“金卫”工程，是我国国民经济信息化的重要组成部分，必然要与“金桥”、“金卡”等国民经济信息网络相连接。

(5) 建立以计算机化病人病历 CPR 为核心的 HIS 会成为已初步建立医院管理信息系统的那些医院的下一个努力目标。

6.1.2 医院信息系统的特点

医院信息系统作为企业管理系统的一个子类，具有其自身很强的特点和复杂性。这是由医院本身的目标、任务和性质决定的，它不仅要同其他所有 MIS 系统一样追踪管理伴随人流、财流、物流所产生的管理信息，从而提高整个医院的运作效率，而且还应该支持以病人医疗信息记录为中心的整个医疗、科学、科研活动。

广义地说，HMIS 是管理系统(MIS)在医院环境的具体应用。因此它必定具有一些与其他 MIS 系统共有的特性。但是，医院信息系统的许多不同于一般 MIS 系统的独有的特点，为 HIS 的设计与实现带来更高的难度，更多的复杂性。

(1) 在许多情况下，它需要极其迅速的响应速度和联机事务处理能力。在一个急诊病人入院抢救的情况下，迅速、及时、准确地获得他的既往病史和医疗记录的重要性是显而易见的。

(2) 医疗信息的复杂性。病人信息是以多种数据类型表达出来的，不仅是文字类型的数据，而且经常需要图形、图表、影像等类型的数据。

(3) 信息的安全、保密性要求高。病人医疗记录是一种拥有法律效力的文件，它不仅在医疗纠纷案件中，而且在许多其他法律程序中均会发挥重要作用，有关人事的、财务的，乃至病人的医疗信息均有严格的保密性要求。

(4) 数据量大。任何一个病人的医疗记录都是一部不断增长的、图文并茂的书，而一个大型综合性医院拥有上百万份病人的病案是常见的。

(5) 缺 乏医疗信息处理的标准。这是另一个突出地导致医院信息系统开发复杂化的问题。目前医疗卫生界极少有医学信息表达、医院管理模式与信息系统模式的标准与规 范。计算机专业人员在开发信息系统的过程中要花费极大精力去处理自己并不熟悉的领域的信息标准化问题，甚至要参与制定一些医院管理的模式与算法。医学知识 表达的规范化，即如何把医学知识翻译成一种适合计算机的形式，是一个世界性的难题，而真正的病人电子化病历的实现有待于这一问题的解决。

(6) 高水平的信息共享需求。一个医生对医学知识(例如某种新药品的用法与用量，使用禁忌，某一种特殊病例的文献描述与结论等)、病人医疗记录(无论是在院病人还是若干年前已死亡的病人)的需求可能发生在他所进行的全部医、教、研的活动中，可能发生在任何地点。而一个住院病人的住

院记录摘要也可能被全院各有关临床科室、医技科室、行政管理部门所需要。因此信息的共享性设计、信息传输的速度与安全性、网络的可靠性等也是 HIS 必须保证的。

6.1.3 医院信息系统基本功能规范

卫生部于 1997 年 印发公布了《医院信息系统基本功能规范》，对于加快医院信息化基础设施建设，规范管理，提高医院信息系统软件质量，保护用户利益，推动医院计算机应用的健 康发展，起到了重要的指导作用。随着计算机网络技术的迅速发展，卫生部重大医改政策的实施及医疗模式的转变，原《医院信息系统基本功能规范》已不能适应新 形势的需要。为此，卫生部于 2001 年 3 月着手修订《医院信息系统基本功能规范》。

《医院信息系统基本功能规范》共分为 24 章，从以下各个方面规定了相关工作环节的信息管理规范。这些规范的详细内容用户可以参阅附书光盘“相关资料\医院信息系统基本功能规范”文件夹下的文件，每一章的内容实际上对应的就是医院信息管理系统的一个子模块，因此用户最好能够仔细阅读。

- 数据字典标准化
- 门诊医生工作站分系统功能规范
- 住院医生工作站分系统功能规范
- 护士工作站分系统功能规范
- 临床检验分系统功能规范
- 输血管理分系统功能规范
- 医学影像分系统功能规范
- 手术、麻醉管理分系统功能规范
- 药品管理分系统功能规范
- 门急诊挂号分系统功能规范
- 门急诊划价收费分系统功能规范
- 住院病人入、出、转管理分系统功能规范
- 住院收费分系统功能规范
- 物资管理分系统功能规范
- 设备管理分系统功能规范
- 财务管理分系统与经济核算管理分系统功能规范
- 病案管理分系统功能规范
- 医疗统计分系统功能规范
- 院长综合查询与分析分系统功能规范
- 病人咨询服务分系统功能规范
- 医疗保险接口功能规范
- 社区卫生服务接口功能规范
- 远程医疗咨询系统接口功能规范

6.2 医院管理系统需求分析

根据上面的介绍，总结《医院信息系统基本功能规范》的内容，可以将通用医院管理系统所必备的功能归纳如图 6.1 所示，其中每个功能都由若干相关联的子功能模块组成。除此之外系统还应包括信息系统必须具备的通用功能，例如权限设置、数据备份与恢复等。

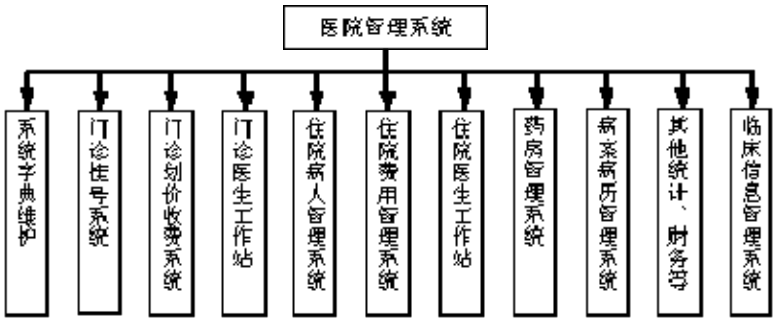


图 6.1 医院管理系统应包括的基本功能

因为随书光盘提供的《医院信息系统基本功能规范》，对这些系统的功能都进行了详细的需求描述，所以下面不再详细对医院管理系统进行需求分析，只是列举出每个系统对应的一些功能。

6.2.1 业务流程

医院管理的基本业务流程如图 6.2 所示。

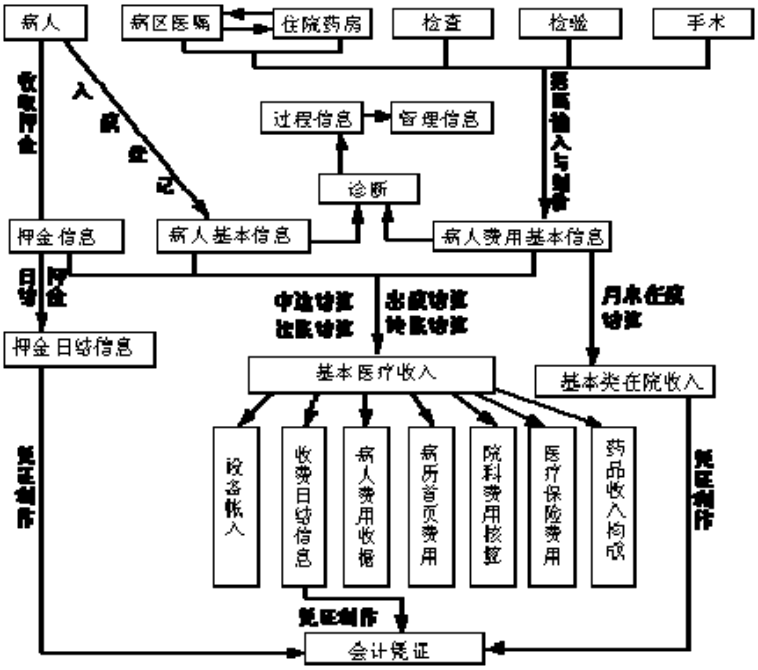


图 6.2 医院管理的基本业务流程

医院门诊业务流程如图 6.3 所示。

医院门诊药房管理的业务流程如图 6.4 所示。

病人住院业务流程如图 6.5 所示。

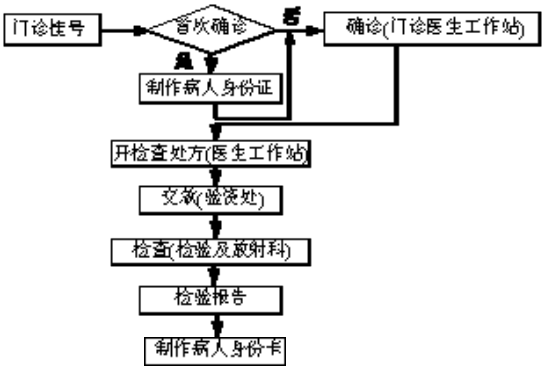


图 6.3 医院门诊业务流程

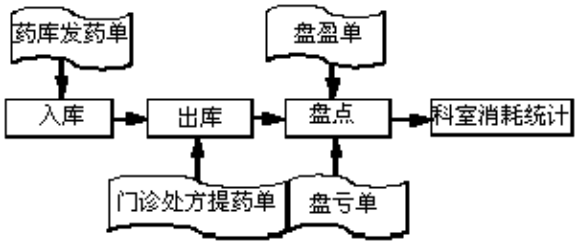


图 6.4 医院门诊药房管理业务流程

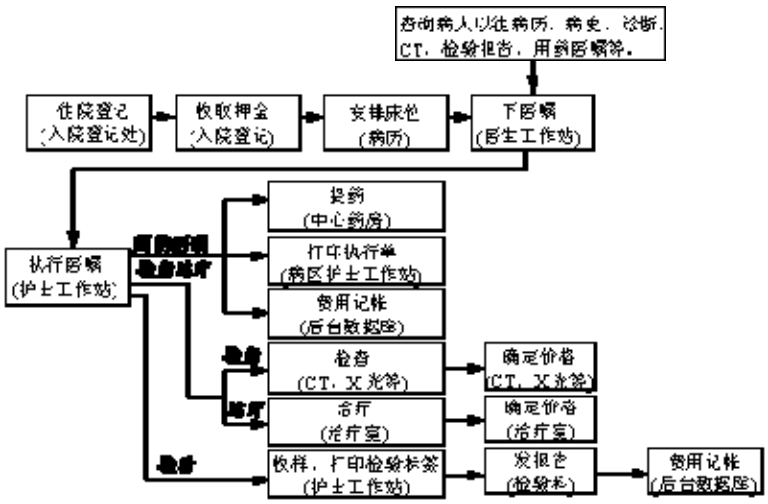


图 6.5 病人住院业务流程

病人出院结帐业务流程如图 6.6 所示。

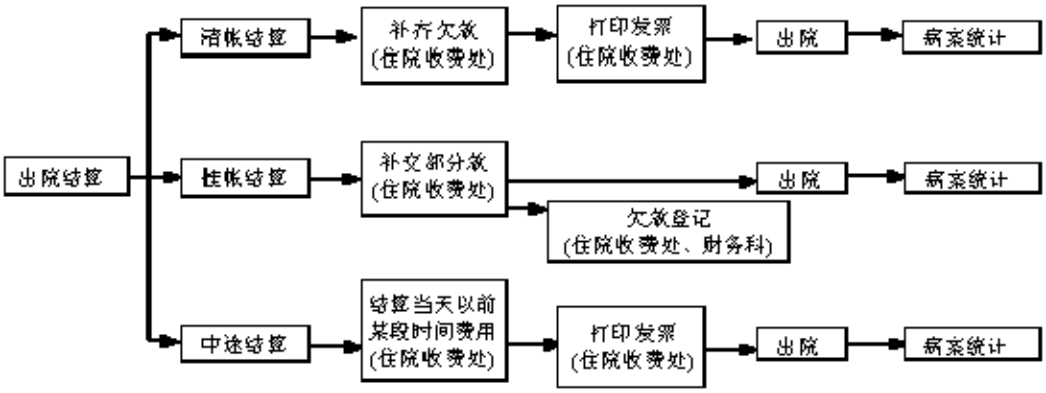


图 6.6 病人出院结帐业务流程

6.2.2 系统字典维护

“系统字典维护”功能模块用于设置医院管理系统的常用字典信息，包括如图 6.7 所示的子功能模块。

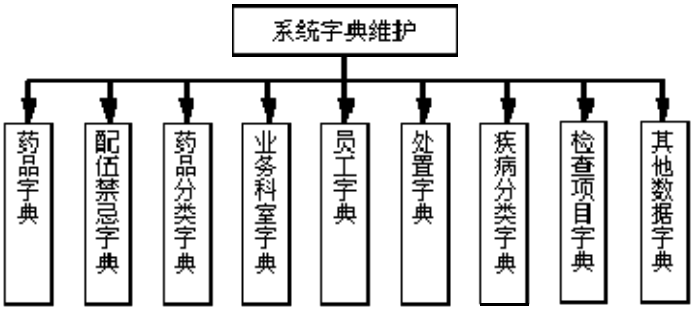


图 6.7 “系统字典维护”功能模块

6.2.3 门诊挂号系统

“门诊挂号系统”功能模块用于建立和维护病人的主索引信息，分配病人的 ID 号，确保病人信息的惟一性，为病人建立就诊卡，对门诊病人进行挂号或者预约号处理，为门诊病人的后续活动以及门诊工作量统计提供信息。病人首次就医时可办理 IC 卡、磁卡等，实现一卡通看病，持卡病人就诊时通过刷卡代替频繁的排队交费，可以大大提高效率，减少等待时间。挂号时计算机自动分配临时 ID 号，可选择输入病人姓名、挂号类别(普诊号、专家号等)及就诊科室等信息，打印产生门诊挂号单，挂号单上的条码号将是病人各环节就医的依据，实现划价收费、项目检查、药房取药的一体化流水作业。

6.2.4 门诊划价收费系统

“门诊划价收费系统”功能模块用于在门诊收费处记录病人的缴费信息，并执行相应的统计核算功能，包含的子功能模块如图 6.8 所示。

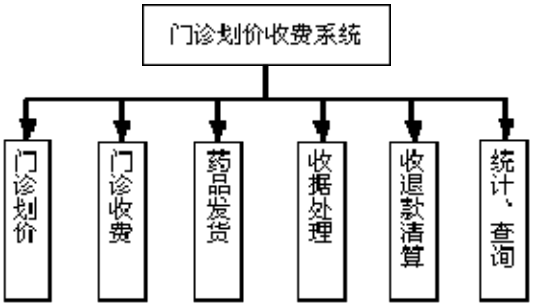


图 6.8 “门诊划价收费系统”功能模块

“门诊划价”用于完成门诊病人各种处方、检查申请、治疗申请等诊治费用的计价工作，各种药品、检查的价格信息在字典管理中维护。

“门诊收费”用于完成门诊病人各种诊治费用的收取工作，能依据划价单(或其他方法)查询病人划价信息，进行费用收取、收据打印处理，并保存操作记录备查。

“药品发货”用于药房预先打印需要发货的药品明细，并将药品准备好，这样病人取药时就可以直接给病人，避免拿到病人的交费单后再去找相应的药品，提高工作效率。

6.2.5 门诊医生工作站

“门诊医生工作站”功能模块是医院信息系统的关键模块，是医疗工作最主要的信息来源。主要功能是完成病历基本指标的填写，病情、病史的记载，医嘱的开立和实施，以及相关辅助功能。该功能模块实现了医生病历收发和医嘱作业的数字化，包含的子功能模块如图 6.9 所示。“门诊医生工作站”功能模块是医院管理系统中比较高层次的功能模块，一般医院的管理都达不到应用的要求，该模块应用后，医生的医嘱可以直接输入到计算机，而不是写在处方上，同时划价时可以直接调出电子医嘱，进行划价。

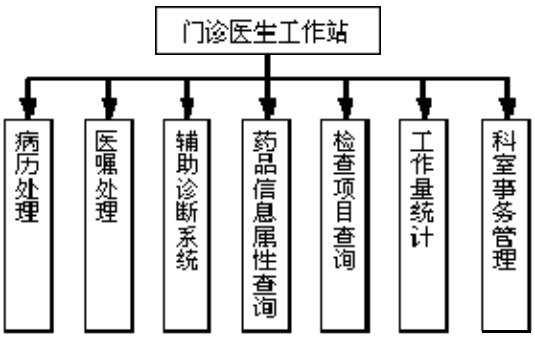


图 6.9 “门诊医生工作站”功能模块

6.2.6 住院病人管理系统

“住院病人管理系统”功能模块用于进行病人入院登记及病人基本情况信息的记录，以及病人在

住院过程中的流动控制和管理，为费用管理、病区护理站、医生工作站等模块提供病人的基本信息，包含的子功能模块如图 6.10 所示。

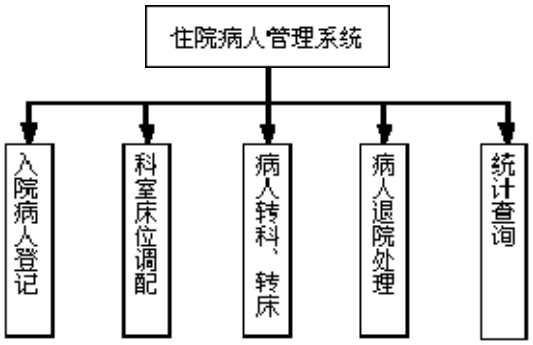


图 6.10 “住院病人管理系统”功能模块

病人办理住院手续时，给病人分配一个住院号，并建立病人住院首页。如果病人是首次住院，分配一个新的住院号；如果病人不是首次住院，系统通过检索病案管理系统，查找到病人所拥有的住院号，在此住院号下新建一份病案，并自动生成病案首页的相关内容。

6.2.7 住院费用管理系统

“住院费用管理系统”功能模块用于完成住院病人的费用和预收款的记录和监管，病人预收金的催缴，病人出院的费用清算及收据处理，包含的子功能模块如图 6.11 所示。

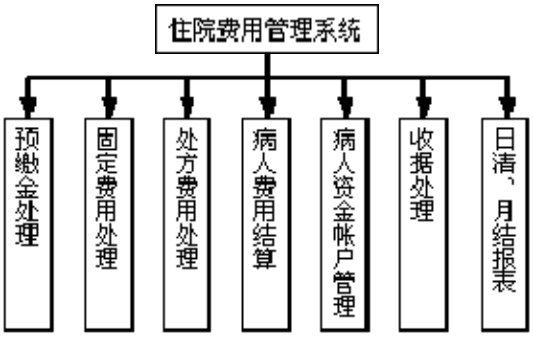


图 6.11 “住院费用管理系统”功能模块

6.2.8 住院医生工作站

“住院医生工作站”功能模块用于完成住院部医生病历基本指标的填写，病情和病史的记载，医嘱的开立、实施，以及相关辅助功能。该功能模块实现医生病历收发和医嘱作业的数字化，包含的子功能模块类似于“门诊医生工作站”。

6.2.9 药房管理系统

“药房管理系统”功能模块用于管理医院药房的采购、入库及出库等业务，包含的子功能模块如图 6.12 所示。

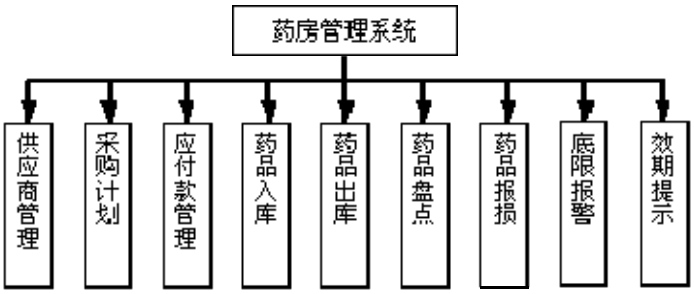


图 6.12 “药房管理系统”功能模块

6.2.10 病案病历管理系统

“病历”作为医院对患者进行诊疗全过程的完整记录，具有十分重要的作用，国家法律对医院病历的记录、保存等都有严格的要求，当发生医疗纠纷时，病历是调节纠纷的重要依据。

电子病历有两层含义，核心的含义是“病历”，而“电子”只是一个限定词，说明它的手段、过程与物质形式。因此，电子病历至少应达到以下几个要求：

- 能够准确地按时间序列真实反映患者自入院到出院期间的医疗与护理过程。
- 可操作性与易操作性。
- 有利于提高医疗质量，促进医务人员掌握相应的知识，提高医疗文书的质量。
- 能达到作为法律依据所要求的安全性。
- 能达到保护患者个人隐私的安全性要求。
- 信息的记录、传输、保存均以电子数据的形式，而从本质上区别于传统的物质形式(纸张、胶片等)。
- 输入后的数据不得修改，任何操作都应留有痕迹。

尤其是最后一条，因为电子数据不像纸张数据，如果系统存在漏洞，数据可以随意更改，则电子病历就失去了其存在的意义。

6.2.11 院长综合查询系统

“院长综合查询系统”功能模块用于从医院信息系统中加工处理出有关医院管理的医、教、研和人、财、物分析决策信息，以便为院长及各级管理者决策提供依据。

6.2.12 外部数据接口

医院管理信息系统的数据不是孤立的，需要和外部很多系统进行数据对接，例如，医保、社区卫生服务、远程医疗服务等都需要医院提供相应的信息。

“医疗保险数据接口”主要完成医院信息系统与医保部门进行信息交换的功能，包括下载、上传、处理医保病人在医院中发生的各种与医疗保险有关的费用，并做到及时结算。

“社区卫生服务数据接口”主要用于跟踪病人，提高出院后服务质量，为社区病人转上级医院提供快速、方便的服务，以及为各种医疗统计分析提供基础数据。

“远程医疗服务数据接口”主要用于保证远程医疗咨询系统所需的信息能及时、迅速的从医院信息系统中直接产生并读取，最大限度地避免信息的二次录入，使对方医院能够调阅原始的没有因各种处理带来误差的真实数据与信息。

6.3 医院管理系统数据库分析

根据以上需求分析，一个基本的医院管理系统数据库中大致包括 80 多张表，分别存放相应子功能的数据信息，其中“药品资料”、“病人资料”、“科室资料”等表都是系统的关键表，其他表与这些关键表间的关系是 N:1 的关系。

6.3.1 医院管理系统E-R图

因为整个系统涉及的实体和属性较多，限于篇幅不能也没有必要一一列举。图 6.13 为医院管理系统关键实体的 E-R 图。

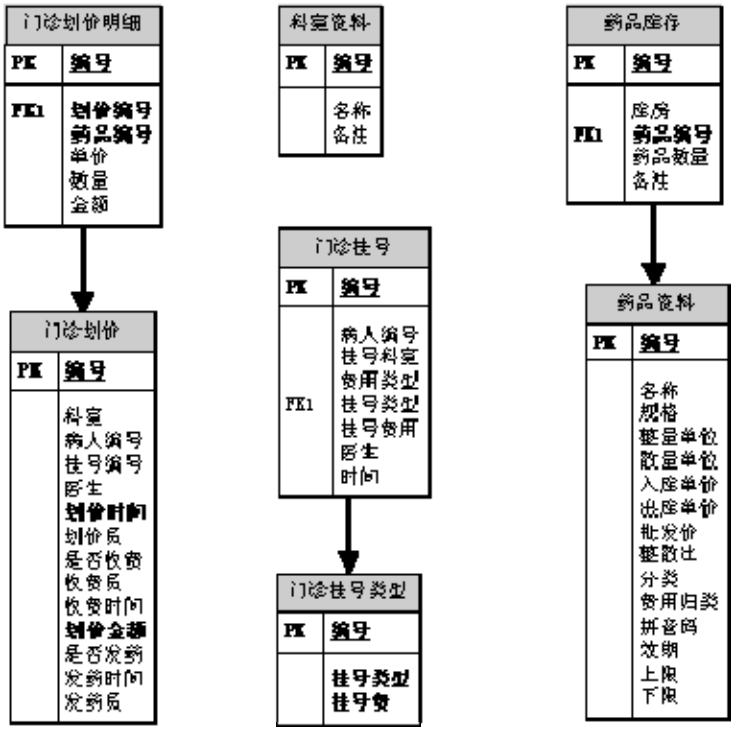


图 6.13 医院管理系统 E-R 图

其他实体与基本信息表间的对应关系都是类似的，同时随书光盘“建库脚本\医院管理系统.sql”文件提供了创建数据库所有表的脚本，被省略的实体对象和实体属性用户完全可以参考这些脚本，也可以按照第 5 章中提供的方法使用 Microsoft Visio 2002 自动生成全部实体和属性的 E-R 图。

6.3.2 医院管理系统表清单

附书光盘“建库脚本\医院管理系统.sql”文件提供了创建数据库所有表的脚本，下面仅列出一些重要表的名称及其用途供参考，如表 6.1 所示。其中每张表所包含的字段用户可以自己查看建库脚本和前面的需求分析。

表 6.1 医院管理系统表清单

表 名 称	表 用 途
药品资料	保存医院药品的基础信息，包括售价等
医生资料	保存医生信息，包括医生所属的科室
科室资料	保存科室分类信息，如分为内科、外科等
库房资料	保存库房分类信息，如分为仓库、中药房、西药房等
职务资料	保存人员的职务分类信息
病人信息库	保存病人的基本信息，以后可以重复使用
--门诊相关表	
门诊挂号	保存门诊病人挂号的信息
门诊挂号类型	保存门诊挂号类型分类信息及其挂号价格，如普通号、专家号等
门诊划价	门诊划价信息(主表)
门诊划价明细	门诊划价明细信息(从表)
门诊收费项目	保存门诊的收费项目及其价格信息，内容包括名称、类型、费用等
--住院相关表	
住院处方明细	保存住院部医生所开具的处方信息
住院处方预设	保存住院部的预设处方信息

住院记帐记录	保存医院来往的帐簿记录
住院收费单	住院部划价信息(主表)
住院收费记录	住院部划价明细信息(从表)
住院收费项目	保存住院部的收费项目及其价格信息
--药品相关表	
产地	保存药品产地属性的字典
调拨药品	保存药品库间调拨的信息
分销客户	保存药品批发销售的信息
供货单位	保存药品供货单位的信息
计量单位	保存药品的计量单位信息
药品分类	保存药品的分类信息
药品库存	保存药品的库存数量、金额信息
药品请领	保存药房向仓库申请药品的信息
药品销售	保存药品销售的信息

续表

表 名 称	表 用 途
库存盘点	保存药品的盘点信息
库存损耗	保存药品的损耗信息
协约单位	保存协约单位的基础信息
盘点明细	保存药品的盘点明细信息
采购计划	保存药品的采购计划信息
入库药品	保存药品的入库信息

6.4 实例制作介绍

基于以上需求分析和数据库分析，用户对一个标准的医院管理系统应该有了一个全面的认识。下面将用实例说明如何利用 PowerBuilder 完成系统的开发。

6.4.1 实例功能

由于篇幅有限，本实例将详细介绍如图 6.14 所示功能的开发过程，并简化其中各功能所包含的属性，其他功能用户完全可以参照这些功能的开发方法完成开发。

需要强调的是，由于用户登录和权限管理的功能各个系统实现的方法是一致的，这里就不再详细介绍，具体实现方法参看第 1 章。

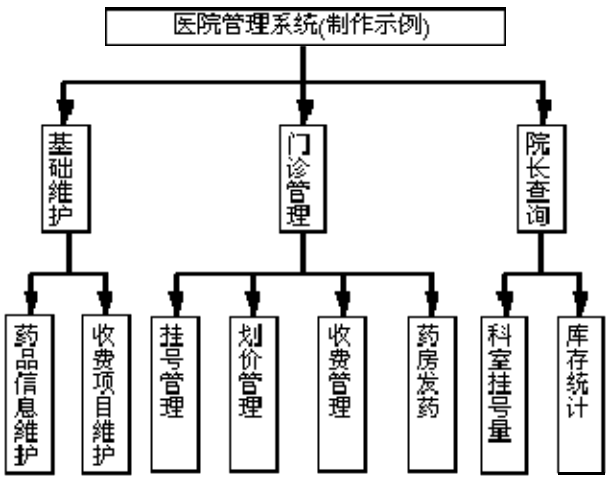


图 6.14　详细介绍的功能模块

6.4.2 系统流程图

系统流程图如图 6.15 所示。

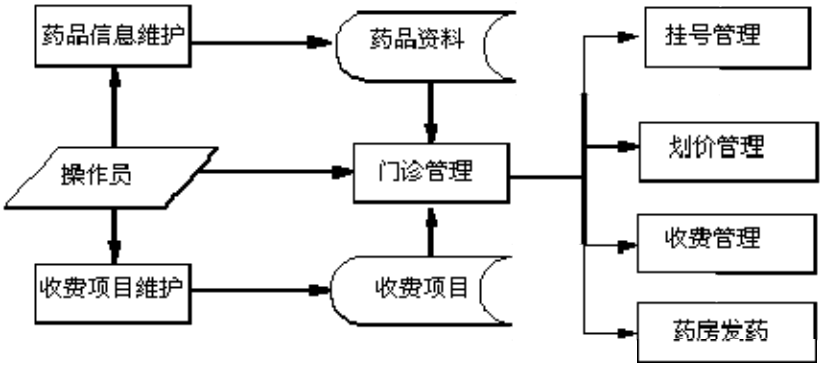


图 6.15　系统流程图

6.5 数据库设计

根据实例介绍，“药品资料”和“门诊收费项目”表是系统关键的表，其他各表均与之通过“编号”字段相对应进行多对一的关联。系统共需要 10 张表，用途分别如表 6.2 所示。

表 6.2　系统表及其用途

表 名 称	表 用 途
药品资料	保存医院药品的基础信息，包括售价等
医生资料	保存医生信息，包括医生所属的科室
科室资料	保存科室分类信息，如分为内科、外科等
病人信息库	保存病人的基本信息，以后可以重复使用
门诊挂号	保存门诊病人挂号的信息
门诊挂号类型	保存门诊挂号类型分类信息及其挂号价格，如普通号、专家号等
门诊划价	门诊划价信息(主表)
门诊划价明细	门诊划价明细信息(从表)
门诊收费项目	保存门诊的收费项目及其价格信息，内容包括名称、类型、费用等
药品库存	保存药品的库存数量、金额信息

6.5.1 创建数据库

打开 SQL Server 企业管理器，新建一个数据库，名称为 hisbook。利用光盘中的脚本代码文件“建库脚本\医院管理系统实例程序.sql”创建数据库对象，完成数据库的设计。也可以利用“建库脚本\hisbook.bak”文件直接恢复数据库，这样数据库中存在初始的药品、收费项目等信息。

后面几小节将列出几个重点表的建库脚本，其他表的脚本参考光盘的脚本文件。

6.5.2 创建“药品资料”表

创建““药品资料”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[药品资料] (
   [编号] [varchar] (20) NOT NULL ,
   [名称] [varchar] (150) NULL ,
   [规格] [varchar] (100) NULL ,
   [整量单位] [varchar] (50) NULL ,
   [散量单位] [varchar] (50) NULL ,
```

```
    [入库单价] [decimal](12, 2) NULL ,
    [出库单价] [decimal](12, 2) NULL ,
    [批发价] [decimal](12, 2)  NULL ,
    [整散比] [decimal](12, 2)  NULL ,
    [分类] [varchar] (100) NULL ,
    [费用归类] [varchar] (100) NULL ,
    [拼音码] [varchar] (50) NULL ,
    [效期] [int] NULL ,
    [上限] [decimal](12, 2) NULL ,
    [下限] [decimal](12, 2) NULL
) ON [PRIMARY]
GO
```

6.5.3 创建“病人信息库”表

创建“病人信息库”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[病人信息库] (
    [编号] [varchar] (15)  NOT NULL primary key,
    [姓名] [varchar] (30) NULL ,
    [性别] [varchar] (2)  NULL ,
    [年龄] [int]NULL ,
    [民族] [varchar] (20) NULL ,
    [费用类型] [varchar] (20) NULL ,
    [电话] [varchar] (15)  NULL ,
    [拼音码] [varchar] (5)  NULL
) ON [PRIMARY]
GO
```

6.5.4 创建“门诊挂号”表

创建“门诊挂号”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[门诊挂号] (
    [编号] [varchar] (15)  NOT NULL ,
    [病人编号] [varchar] (15)  NULL ,
    [姓名] [varchar] (30) NULL ,
    [性别] [varchar] (2)  NULL ,
    [挂号科室] [varchar] (30) NULL ,
    [费用类型] [varchar] (30) NULL ,
    [挂号类型] [varchar] (30) NULL ,
    [挂号费用] [decimal](12, 2) NULL ,
    [医生] [varchar] (30) NULL ,
    [时间] [datetime] NULL,
    [是否已划价] [varchar] (2)  default '否'
) ON [PRIMARY]
GO
```

6.5.5 创建“门诊划价”和“门诊划价明细”表

创建“门诊划价”表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[门诊划价] (
   [编号] [varchar] (15)  NOT NULL ,
   [科室] [varchar] (30) NULL ,
   [挂号编号] [varchar] (15)  NULL ,
   [医生] [varchar] (10)  NULL ,
   [划价时间] [datetime] NOT NULL ,
   [划价员] [varchar] (10) NULL ,
   [是否收费] [varchar] (2) NULL ,
   [收费员] [varchar] (10) NULL ,
   [收费时间] [datetime] NULL ,
   [划价金额] [money] NOT NULL ,
   [是否发药] [varchar] (2) NULL ,
   [发药时间] [datetime] NULL ,
   [发药员] [varchar] (10) NULL
) ON [PRIMARY]
GO
```

创建“门诊划价明细“表的 SQL 脚本如下：

```
CREATE TABLE [dbo].[门诊划价明细] (
   [编号] [int] IDENTITY (1, 1) NOT NULL ,
   [划价编号] [varchar] (15)  NOT NULL ,
   [药品编号] [varchar] (15)  NOT NULL ,
   [单价] [decimal](12, 2) NULL ,
   [数量] [decimal](12, 2) NULL ,
   [金额] [decimal](12, 2) NULL
) ON [PRIMARY]
GO
```

6.5.6 创建其他关键表

以下是其他关键表的创建脚本，剩余表可以参看附书光盘。

```
CREATE TABLE [dbo].[门诊挂号类型] (
   [编号] [int] IDENTITY (1, 1) NOT NULL ,
   [挂号类型] [varchar] (50) NOT NULL ,
   [挂号费] [decimal](12, 2)  NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[门诊收费项目] (
   [编号] [varchar] (20) NOT NULL ,
   [名称] [varchar] (100) NULL ,
   [拼音码] [varchar] (100) NULL ,
   [费用] [decimal](12, 2) NULL ,
```

```
    [费用分类] [varchar] (100) NULL ,
    [病种分类] [varchar] (100) NULL ,
    [备注] [varchar] (100) NULL
) ON [PRIMARY]
GO
```

6.5.7 创建主键及外键等表约束

在查询分析器中通过如下代码创建表的主键及外键等表约束。

```
-- 主键、外键关联字段
ALTER TABLE [dbo].[药品库存] WITH NOCHECK ADD
   CONSTRAINT [PK_药品库存] PRIMARY KEY  CLUSTERED
   (
     [编号]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[药品资料] WITH NOCHECK ADD
   CONSTRAINT [PK_药品资料] PRIMARY KEY  CLUSTERED
   (
     [编号]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[门诊划价] WITH NOCHECK ADD
   CONSTRAINT [PK_门诊划价] PRIMARY KEY  CLUSTERED
   (
     [编号]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[门诊划价明细] WITH NOCHECK ADD
   CONSTRAINT [PK_门诊划价明细] PRIMARY KEY  CLUSTERED
   (
     [编号]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[门诊挂号] WITH NOCHECK ADD
   CONSTRAINT [PK_门诊挂号] PRIMARY KEY  CLUSTERED
   (
     [编号]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[门诊挂号类型] WITH NOCHECK ADD
```

```
    CONSTRAINT [PK_门诊挂号类型] PRIMARY KEY  CLUSTERED
    (
      [编号]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[门诊收费项目] WITH NOCHECK ADD
    CONSTRAINT [PK_门诊收费项目] PRIMARY KEY  CLUSTERED
    (
      [编号]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[药品库存] ADD
    CONSTRAINT [FK_药品库存_药品资料] FOREIGN KEY
    (
      [药品编号]
    ) REFERENCES [dbo].[药品资料] (
      [编号]
    )
GO

ALTER TABLE [dbo].[门诊划价明细] ADD
    CONSTRAINT [FK_门诊划价明细_门诊划价] FOREIGN KEY
    (
      [划价编号]
    ) REFERENCES [dbo].[门诊划价] (
      [编号]
    )
GO

ALTER TABLE [dbo].[门诊挂号] ADD
    CONSTRAINT [FK_门诊挂号_门诊挂号类型] FOREIGN KEY
    (
      [挂号类型]
    ) REFERENCES [dbo].[门诊挂号类型] (
      [编号]
    )
GO
--其他外键、主键等约束参看附书光盘中的脚本。
```

6.5.8 创建相关视图

在查询分析器中通过如下代码创建“v 收费项目及药品”视图，该视图对“药品资料”和“门诊收费项目”两张表的数据进行合并操作，从而在划价时实现两张表同时查询的功能。

```
create view v 收费项目及药品
```

```
as
select 编号,名称,规格, 整量单位 as 单位, 出库单价 as 单价, 拼音码 from 药品资料
   union
select 编号,名称,费用分类 as 规格,病种分类 as 单位,费用 as 单价,拼音码
from 门诊收费项目
GO
```

6.5.9 创建存储过程

系统使用了“sf_药品发出”一个存储过程，实现划价收费后药房发出药品，减少库存的功能。具体计算方法在下面的脚本中有详细的注释，用户可以参考这些注释。

```
create proc sf_药品发出 @id varchar(15)
as
begin tran
   -- 划价收费后药房发货
   update 门诊划价 set 是否发药='是', 发药员='王五', 发药时间=getdate()
     where 编号=@id
   -- 减少库存,演示程序,直接减少'西药房'的库存
   -- 实际应用,应该根据药房减
   update 药品库存 set 药品数量=药品数量-b.数量 from
     药品库存 as a, 门诊划价明细 as b where a.药品编号=b.药品编号
       and 库房='西药房' -- 这里应该替换为实际的组别
       and 划价编号=@id
commit
GO
```

6.6.1 程序运行结果

本系统采用多文档窗体程序，每一功能对应一个子窗体。

本实例运行后的结果如图 6.16 所示。选择菜单中的各命令可以进入相应的功能。

图 6.16 实例运行结果

(1) 选择【字典维护】|【药品信息】命令，进入【药品信息维护】功能窗体，在其中输入医院相关的药品信息，如图 6.17 所示，其中“分类”属性表示药品所属的剂型，可从“药品分类”表中查询。需要新增或修改药品，单击相应的按钮，输入新信息后单击【保存】按钮即可。需要删除一条信息，选择该信息后单击【删除】按钮。

(2) 选择【字典维护】|【收费项目】命令，进入【收费项目维护】功能窗体，如图 6.18 所示，

在其中输入医院收费项目的信息。

图 6.17 【药品信息维护】功能窗体

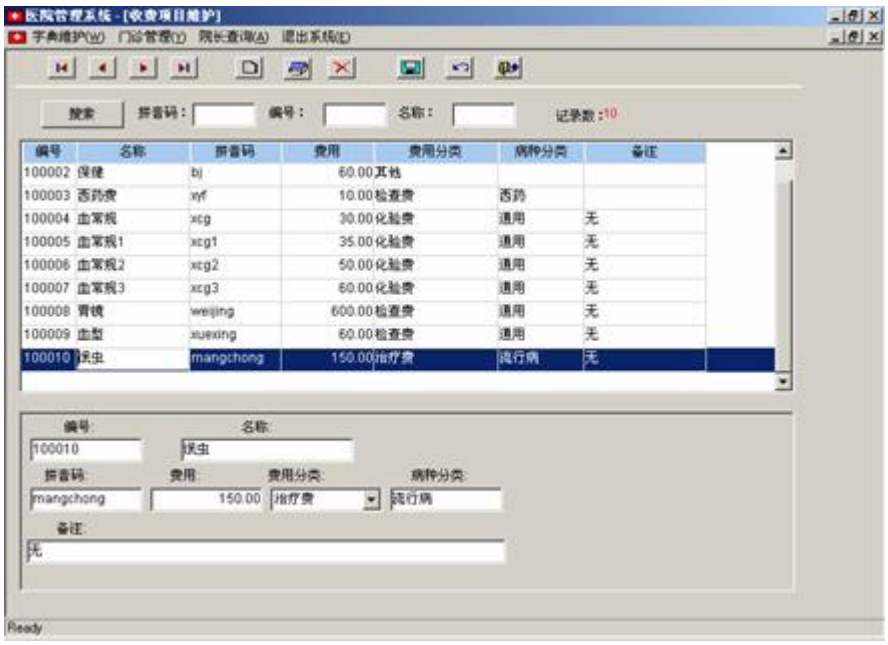


图 6.18 【收费项目维护】功能窗体

(3) 选择【门诊管理】|【门诊挂号】命令，进入【门诊挂号】功能窗体，在其中完成门诊病人挂号的功能，如图 6.19 所 示。其中【姓名】、【费用类型】、【挂号科室】、【医生】文本框为必填字段，这些信息是必须填写的，灰色的文本框是不可填写字段，内容由程序自动生成，如 【挂号费用】文本框的内容将根据所选择的挂号类型自动关联显示出来。填写相应的信息后单击【保存】按钮，数据库将信息写入“病人信息库”和“门诊挂号”两 张表。

图 6.19 【门诊挂号】功能窗体

(4) 选择【门诊管理】|【门诊划价】命令，进入【门诊划价管理】功能窗体。先选择划价对应的挂号单，如图 6.20 所示。选择后，划价单对应的姓名、科室、医生等信息自动关联出来。在 Grid 控件的【编号】栏中输入医生处方中的药品或收费项目的编号，回车后出现选择列表，选择相应的项目，并输入项目的数量，系统自动计算出相应的费用，如图 6.21 所示。完成相应的划价后单击【保存】按钮，系统将数据保存到“门诊划价”和“门诊划价明细”表。划价完成后病人需要到收费处交费。

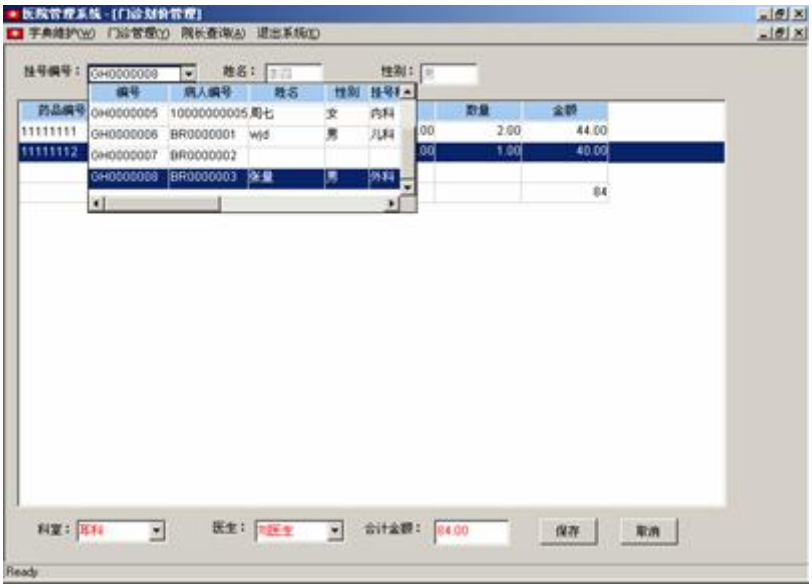


图 6.20 【门诊划价管理】功能窗体--选择挂号单

图 6.21 【门诊划价管理】功能窗体——划价

(5) 选择【门诊管理】|【门诊收费】命令，进入【门诊收费管理】功能窗体，如图 6.22 所示。在划价单主列表中选择收费的单据，从列表中将自动关联出该划价单对应的明细，单击【收费】按钮，弹出【门诊收费付款】功能窗体，如图 6.23 所示。在【收款】文本框中输入收款的金额，【找零】文本框自动计算出找零金额。单击【收款】按钮，完成该划价单的收款工作，“门诊划价”表对应的记录“是否收费”字段设置为“是”。

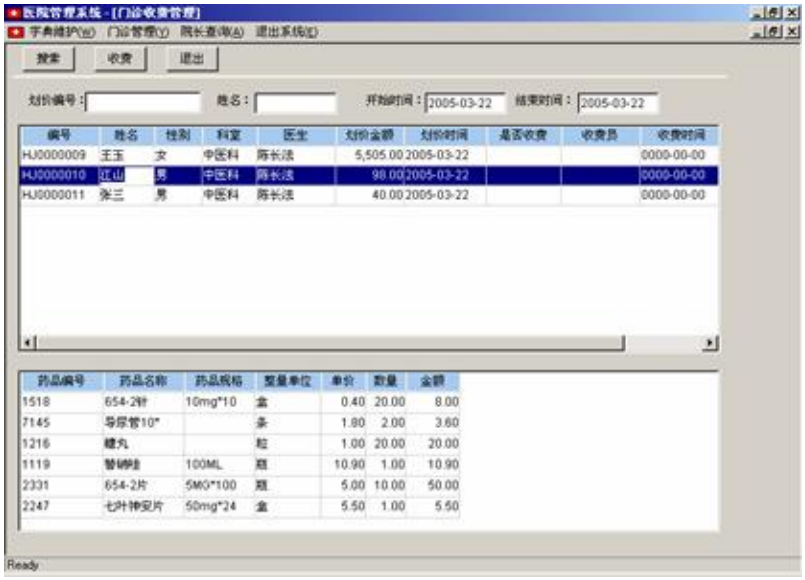


图 6.22 【门诊收费管理】功能窗体

图 6.23 【门诊收费付款】功能窗体

交费后病人凭打印出的单据到药房领取药品。

(6) 选择【门诊管理】|【药房发药】命令，进入【药房发药管理】功能窗体，如图 6.24 所示，在划价单主列表中选择单据，从列表中关联出该单据对应的药品明细，单击【发药】按钮，系统调用“sf_药品发出”存储过程，完成该划价单的发药领药工作，“门诊划价”表对应的记录“是否发药”字段设置为“是”，并减少药品明细对应的库存数量。

图 6.24 【药房发药管理】功能窗体

(7) 院长查询功能主要是为医院管理人员提供决策依据，实例制作了两个查询功能。选择【院长查询】|【科室挂号量】命令，进入【科室挂号量】功能窗体，如图 6.25 所示。输入需要统计的时间段，单击【搜索】按钮，可以统计出该时间段各个科室的挂号人数和挂号金额。

图 6.25 【科室挂号量】功能窗体

(8) 选择【院长查询】|【药品库存量】命令，进入【药品库存查询】功能窗体，如图 6.26 所示。输入需要查询的条件，单击【搜索】按钮，可以查询出相应药品的库存数量。

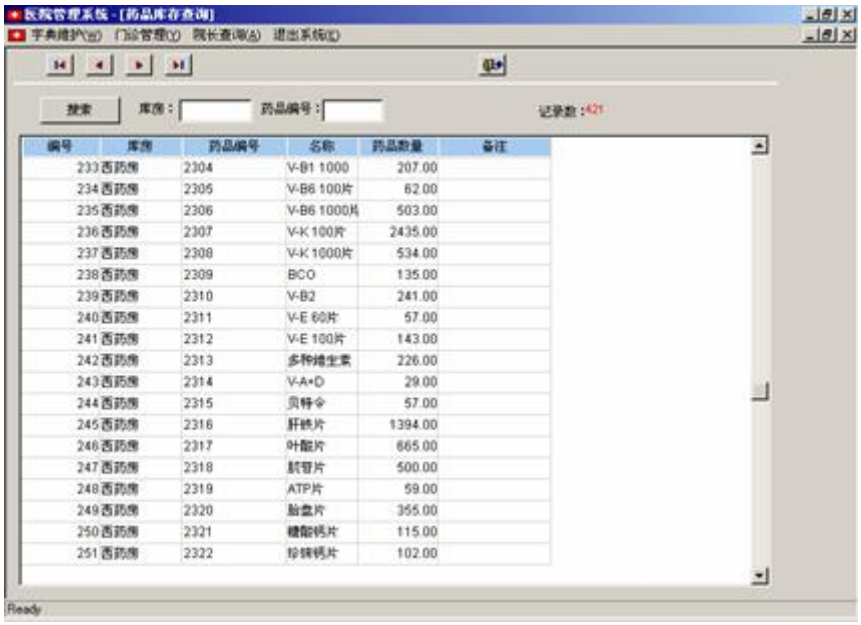


图 6.26 【药品库存查询】功能窗体

6.6.2 创建工程

(1) 启动 PowerBuilder，新建一个工作区，将工作区保存为“his.pbw”。因为 PowerBuilder 是工作在工作区上的，而工作区里可以建立多个工程，每个工程就相当于是一个项目，所以 PowerBuilder 里一个工作区可以建立多个工程。实际应用中，除非多个工程有很紧密的联系，一般都是一个工作区建一个工程。

(2) 建立一个工程。选择 File | New 命令，在弹出的 New 对话框中打开 Target 选项卡，如图 6.27 所示。

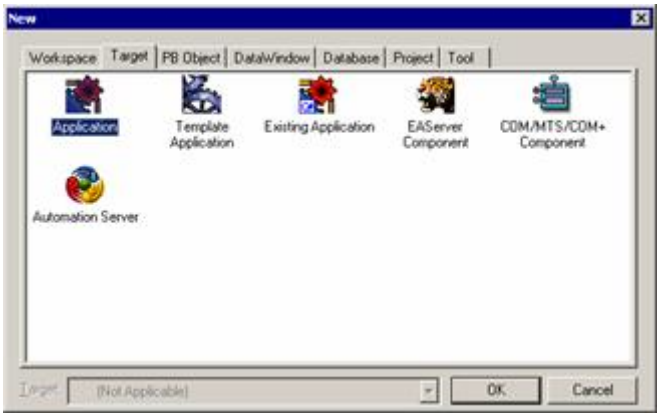


图 6.27 新建工程

(3) 选中 Application 选项，单击 OK 按钮，弹出如图 6.28 所示的对话框，在 Application Name 文本框中输入“his”，然后下面的库和工程将自动变为工作区所在的路径加上该工程。

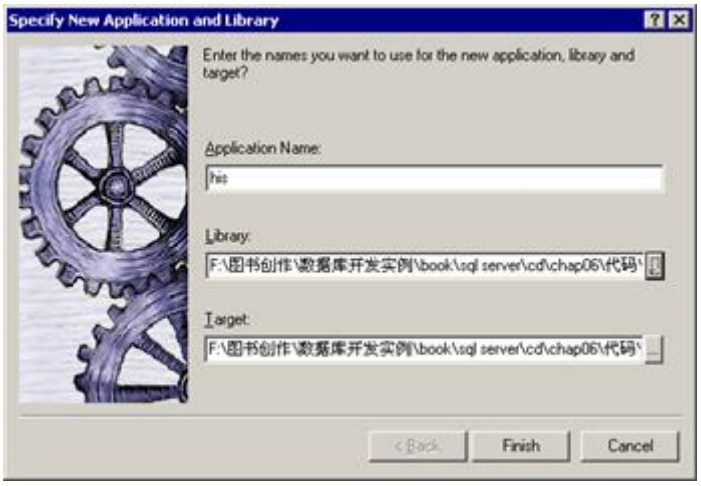


图 6.28　工程名图示

(4)　单击 Finish 按钮，建立名称为“his”的工程，并且包含一个名称为“his”的 pbl。这里，工程就是 PowerBuilder 的 Target，pbl 就是 PowerBuilder 的库，库下面才能建立窗口、按钮等项，所以 PowerBuilder 是以 pbl 来工作的。而 PowerBuilder 中一个工程里可以包含多个库，一般建议每个库文件不要太大，最好不要超过 2MB。在实际应用中，库文件的分类一般根据数据库结构或者模块来划分。

(5)　选择 View | Properties 菜单命令，在 Properties(属性)对话框的 General 选项卡中修改工程的 DisplayName 属性为“医院管理系统”，如图 6.29 所示。

(6)　单击 Additional Properties 按钮，在弹出的 Application 对话框中切换到 Icon 选项卡，为项目选择一个 Exe 程序的图标，本例中选择源代码目录下的“icon\医院.ico”文件，如图 6.30 所示。

图 6.29　修改 DisplayName 属性

图 6.30　设置应用程序图标

(7)　建立一个名称为 hislink 的 ODBC，连接数据库 hisbook，用于编写程序时连接数据库。

(8)　在 his.pbl 的 OPEN 事件中输入以下功能代码，通过 ODBC“hislink”连接数据库，连接成功后，打开系统登录窗体。

```
//--------------------------------------------------------------------
// Profile hislink
string Rtn
SQLCA.DBMS = "ODBC"
```

```
SQLCA.AutoCommit = False
SQLCA.DBParm = "ConnectString='DSN=hislink;UID=sa;PWD='"

Connect using SQLCA;//连接数据库
 //连接失败则返回
if SQLCA.sqlcode <> 0 then
   MessageBox("提示信息","连接 ODBC"liblink"失败！")
   return
end if
//打开登录窗体
open(w_login)
Rtn = message.stringparm
if Rtn = "login" then
   //打开主窗体
   open(w_main)
end if
//--------------------------------------------------------------------------
```

(9) 在 his.pbl 的 CLOSE 事件中输入以下功能代码。在程序关闭的时候断开与数据库的连接。

```
disconnect using sqlca; //断开数据库连接
```

现在，由于登录窗体和主窗体还没有制作，所以上面的程序还不能正常运行，下面将介绍主窗体和登录窗体的制作。

6.6.3 创建系统主窗体

(1) 选择 File | New 命令，在弹出如图 6.31 所示的 New 对话框中打开 PB Object 选项卡，然后选中 Window 对象，单击 OK 按钮，创建一个窗体，名称为 w_main。

图 6.31 创建 Window 对象

(2) 窗体 w_main 的控件如图 6.32 所示。

图 6.32　系统主窗体

(3)　程序主窗体 w_main 各控件属性设置如表 6.3 所示。

表 6.3　主窗体各控件属性设置

控件类型	对 象 名	属　性	取值(说明)
Window	w_main	Title	
Window	w_main	WindowType	Mdihelp!
Window	w_main	ICON	icon\医院.ico
GroupBox	gb_zidian	Text	字典维护
GroupBox	gb_mz	Text	门诊管理
GroupBox	gb_yz	Text	院长查询

续表

控件类型	对 象 名	属　性	取值(说明)
StaticText	st_ypxx	Text	药品信息
StaticText	st_sfxx	Text	收费项目
StaticText	st_mzgh	Text	门诊挂号
StaticText	st_mzhj	Text	门诊划价
StaticText	st_mzsf	Text	门诊收费
StaticText	st_yffy	Text	药房发药
StaticText	st_ksgh	Text	科室挂号量
StaticText	st_ypkc	Text	药品库存量

(4)　在文本框“st_ypxx”中加入如下代码，打开【药品信息维护】窗体。

```
//------------------------------------------------------------
//打开 sheet
opensheet(w_ypxx,w_main,1, Original!)
//调整 mdi_1 的位置和大小
w_main.MDI_1.Move(w_main.workspaceX(),w_main.workspaceY())
w_main.MDI_1.Resize(w_main.WorkSpaceWidth() , w_main.workspaceHeight())
//------------------------------------------------------------
```

其他文本框代码实现打开对应表单窗口的功能，跟 st_ypxx 相似，在后面的文本框中就不具体的介绍了(用户可以查看光盘中相应的代码)。

(5)　在窗体 w_main 的 TOOLBARMOVED 事件中加入如下代码，在工具栏变化时，重新设置主窗体中 MDI_1 的大小和位置。

```
//------------------------------------------------------------
w_main.MDI_1.Move(w_main.workspaceX(),w_main.workspaceY())
w_main.MDI_1.Resize(w_main.WorkSpaceWidth() , w_main.workspaceHeight())
w_main.MDI_1.Resize(1 , 1)
//------------------------------------------------------------
```

6.6.4 创建系统主菜单

(1) 选择 File | New 命令，在弹出如图 6.31 所示的 New 对话框中打开 PB Object 选项卡，然后选择 Menu 对象，单击 OK 按钮，创建一个窗体，名称为 m_main。

(2) 窗体的主菜单 m_main 属性设置如图 6.33 所示。

图 6.33 系统主菜单

(3) 窗体的主菜单 m_main 属性设置如表 6.4 所示。

表 6.4 窗体主菜单属性设置

菜单名称	属 性	取值(说明)
m_字典维护	text	字典维护(&W)
m_药品信息	text	药品信息(&Y)
m_收费项目	text	收费项目(&Z)
m_门诊管理	text	门诊管理(&Y)
m_门诊挂号	text	门诊挂号(&X)
m_门诊划价	text	门诊划价(&Z)
m_门诊收费	text	门诊收费(&Y)
m_药房发药	text	药房发药(&R)
m_院长查询	text	院长查询(&A)
m_科室挂号量	text	科室挂号量(&X)
m_药品库存量	text	药品库存量(&Y)
M_退出系统	text	退出系统(&E)
M_退出	text	退出(&E)

(4) 在菜单“m_药品信息”中加入如下代码，打开【药品信息维护】窗体。

```
//------------------------------------------------------------
//打开 sheet
opensheet(w_ypxx,w_main,1, Original!)
//调整 mdi_1 的位置和大小
w_main.MDI_1.Move(w_main.workspaceX(),w_main.workspaceY())
w_main.MDI_1.Resize(w_main.WorkSpaceWidth() , w_main.workspaceHeight())
//------------------------------------------------------------
```

其他菜单打开对应的表单窗口，代码跟“m_药品信息”相似，在后面的菜单中就不具体的介绍了(用户可以查看光盘中相应的代码)。

6.6.5 创建登录窗体

本小节将演示如何利用 PowerBuilder 完成登录系统功能的建立。在用户输入用户编号和密码后对用户的输入进行验证，并记录登录用户的姓名和用户编号，供系统以后使用。

(1) 创建一个窗体，保存名称为“w_logon”。

(2) 设置窗体 Window type 属性为 response!。

(3) 添加控件如图 6.34 所示。

图 6.34 登录窗体

各控件的属性设置如表 6.5 所示。

表 6.5 登录窗体的控件属性设置

控件类型	对 象 名	属 性	取值(说明)
Window	w_login	Title	登录系统
Window	w_login	Window type	response!
StaticText	st_1	text	用户编号：
StaticText	st_2	text	用户名：
StaticText	st_3	text	密码：
StaticText	st_4	text	请输入用户名和密码，登录系统…… 默认用户编号“1”，默认密码“sys”
Picture	p_1	picture	kp2003_2M.bmp
SingleLineEdit	sle_no		
SingleLineEdit	sle_user	DisplayOne	true
SingleLineEdit	sle_pass	Password	true
CommandButton	cb_ok	text	确定
CommandButton	cb_cancel	text	取消

(4) 在文本框 sle_no 的 MODIFIED 事件中加入以下功能代码，根据用户编号查找出用户名并显示在 sle_user 文本框中，如果输入的用户编号不存在，则将文本框 sle_no 的内容设置为空，并且将焦点聚集到文本框 sle_no 上，等待用户输入存在的用户编号。

```
//--------------------------------------------------------------------------------
string ls_no,ls_user
ls_no = trim(this.text)

 SELECT 用户清单.姓名
   INTO :ls_user
   FROM 用户清单
```

```
  WHERE 用户清单.用户编号 = :ls_no  ;

if sqlca.sqlcode <> 0 or ls_user = "" or isnull(ls_user) then
   sle_no.text = ""
   sle_user.text = ""
   sle_pass.text = ""
   sle_no.setfocus()
   return
end if

sle_user.text = ls_user
sle_pass.setfocus()
//--------------------------------------------------------------------------------
```

(5) 在按钮 cb_ok 的 CLICKED 事件中加入以下代码，判断如果用户名和密码全部正确，就登录系统。

```
//--------------------------------------------------------------------------------
string ls_user,ls_pass,ls_passdata
integer li_count

ls_user = trim(sle_user.text)
ls_pass = trim(sle_pass.text)

if isnull(ls_pass) then ls_pass = ""

select count(*) into :li_count from 用户清单 where 姓名 =:ls_user;
if li_count < 1 then
   messagebox("提示信息","请输入正确的用户名！")
   sle_user.setfocus()
   return
end if
//查询密码
select 密码 into :ls_passdata from 用户清单 where 姓名 =:ls_user;
if isnull(ls_passdata) then ls_passdata = ""
if ls_pass <> trim(ls_passdata) then
   messagebox("提示信息","请输入正确的密码！")
   sle_pass.setfocus()
   return
end if
closewithreturn(parent,"login")
//--------------------------------------------------------------------------------
```

(6) 在按钮 cb_cancel 的 CLICKED 事件中加入以下代码，返回 cancel 参数给系统，退出系统。

```
CloseWithReturn(Parent, "Cancel")
```

现在，系统的登录，连接数据库窗口都已经完成了，系统的大概模样就出来了。

6.6.6 完成祖先窗体创建功能

(1) 选择 File | New 命令，在弹出的 New 对话框中打开 PB Object 选项卡，然后单击 Window 对象，新建一个窗体，将其 Name 属性改为“w_sheet”，设置 Windows Type 的属性为 main!，并为其添加如图 6.35 所示的控件。

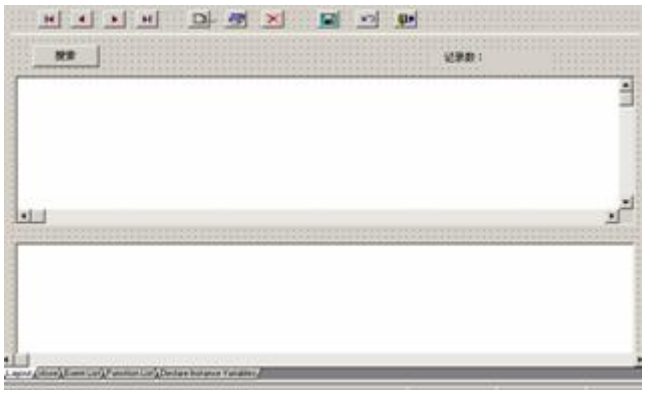


图 6.35　祖先窗体控件图

(2) 各控件的属性设置如表 6.6 所示。

表 6.6　祖先窗体控件属性设置

控件类型	对 象 名	属　性	取值(说明)
Window	w_sheet	Title	医院管理系统
Window	w_shee	WindowType	main!
Window	w_shee	ICON	\ICON\医院.ico
Picturebutton	pb_first	Picture_name	\ICON\首记录.bmp
Picturebutton	pb_prior	Picture_name	\ICON\上记录.bmp
Picturebutton	pb_next	Picture_name	\ICON\下记录.bmp
Picturebutton	pb_buttom	Picture_name	\ICON\尾记录.bmp
Picturebutton	pb_add	Picture_name	\ICON\新增.bmp

续表

控件类型	对 象 名	属　性	取值(说明)
Picturebutton	pb_modify	Picture_name	\ICON\修改.bmp
Picturebutton	pb_del	Picture_name	\ICON\删除.bmp
Picturebutton	pb_save	Picture_name	\ICON\保存.bmp
Picturebutton	pb_cancel	Picture_name	\ICON\取消.bmp
Picturebutton	pb_exit	Picture_name	\ICON\关闭.bmp
CommandButton	cb_query	text	搜索
statictext	st_1	text	记录数：
Singlelineedit	sle_record	text	
Datawindow	dw_1		
Datawindow	dw_2		

(3) 在窗体 w_sheet 中定义实例变量。

```
//定义实例变量
integer il_row,il_rowcount //当前行和记录总数
string is_sql  //数据窗体 dw_1 原始语法
```

(4) 在窗体 w_sheet 的 OPEN 事件中加入以下代码，将数据窗体连接到数据库，并得到数据窗体 dw_1 的原始语法，为后面的检索做准备。

```
//--------------------------------------------------------------------
dw_1.settransobject(sqlca) //连接数据库
dw_2.settransobject(sqlca)
is_sql = dw_1.GetSQLSelect() //得到数据窗体原始语法
//--------------------------------------------------------------------
```

(5) 在图片按钮 pb_first 的 CLICKED 事件中加入以下代码，移动数据窗体中的记录到第一行，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//没有记录则返回
if il_rowcount < 1 then return
il_row = 1 //当前行为 1
//取消选择
dw_1.selectrow(0,false)
//选择当前行
dw_1.selectrow(il_row,true)
//移到当前行
dw_1.scrolltorow(il_row)
//设置按钮状态
pb_first.enabled = false
pb_prior.enabled = false
pb_next.enabled = true
pb_buttom.enabled = true
//调用自定义事件
parent.event ue_search()
//--------------------------------------------------------------------
```

(6) 在图片按钮 pb_prior 的 CLICKED 事件中加入以下代码，移动数据窗体中记录到前一行，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//没有数据则返回
if il_rowcount < 1 then return
//当前记录为第一行则返回
if il_row <= 1 then return
il_row -= 1 //记录减一
//取消选择
dw_1.selectrow(0,false)
//选择当前行
dw_1.selectrow(il_row,true)
//移动到前一行
```

```
dw_1.scrolltorow(il_row)
//控制按钮状态
if il_row = 1 then
   pb_first.enabled = false
   pb_prior.enabled = false
end if
pb_next.enabled = true
pb_buttom.enabled = true
//调用窗口自定义事件
parent.event ue_search()
//--------------------------------------------------------------------
```

(7) 在图片按钮 pb_next 的 CLICKED 事件中加入以下代码，移动数据窗体中记录到下一行，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//记录为最后行则返回
if il_rowcount = il_row then return
il_row += 1 //记录加一
//取消选择
dw_1.selectrow(0,false)
//选择当前行
dw_1.selectrow(il_row,true)
//移动到下一行
dw_1.scrolltorow(il_row)
//控制按钮状态
if il_row = il_rowcount then
   pb_next.enabled = false
   pb_buttom.enabled = false
end if
if il_rowcount > 1 then
   pb_first.enabled = true
   pb_prior.enabled = true
end if
//调用窗口自定义事件
parent.event ue_search()
//--------------------------------------------------------------------
```

(8) 在图片按钮 pb_buttom 的 CLICKED 事件中加入以下代码，移动数据窗体中记录到最后行，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//记录为最后行则返回
if il_rowcount = il_row then return
il_row = il_rowcount //当前记录为最后行
//取消选择
```

```
dw_1.selectrow(0,false)
//选择当前行
dw_1.selectrow(il_row,true)
//移动到最后行
dw_1.scrolltorow(il_row)
//控制按钮状态
pb_next.enabled = false
pb_buttom.enabled = false
if il_rowcount > 1 then
   pb_first.enabled = true
   pb_prior.enabled = true
end if
//调用窗口自定义事件
parent.event ue_search()
//--------------------------------------------------------------------
```

(9) 在图片按钮 pb_add 的 CLICKED 事件中加入以下代码，在数据窗体 dw_1 中插入一行，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//插入一行
il_row = dw_1.insertrow(0)
//选择插入的行
dw_1.selectrow(0,false)
dw_1.selectrow(il_row,true)
dw_1.scrolltorow(il_row)
//控制按钮状态
dw_1.enabled = false
cb_query.enabled = false
cb_query.enabled = false
pb_add.enabled = false
pb_modify.enabled = false
pb_del.enabled = false
//--------------------------------------------------------------------
```

(10) 在图片按钮 pb_modify 的 CLICKED 事件中加入以下代码,修改数据窗体 dw_2 中选中的记录，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//在 d w__2 中修改数据
dw_1.enabled = false
dw_2.enabled = true
//查询按钮不可用
cb_query.enabled = false
pb_add.enabled = false
pb_modify.enabled = false
```

```
pb_del.enabled = false
//--------------------------------------------------------------------
```

(11) 在图片按钮 pb_save 的 CLICKED 事件中加入以下代码，并控制好按钮的状态。因为保存数据的情况每个窗口都不一样，所以保存数据的代码在每个窗口中单独编写。

```
//--------------------------------------------------------------------
//控制按钮状态
cb_query.enabled = true
pb_add.enabled = true
pb_modify.enabled = true
pb_del.enabled = true
//--------------------------------------------------------------------
```

(12) 在图片按钮 pb_cancel 的 CLICKED 事件中加入以下代码，取消在数据窗体 dw_1 中选中的记录，并控制好按钮的状态。

```
//--------------------------------------------------------------------
//回滚数据
rollback;
dw_1.retrieve()
dw_1.enabled = true
dw_2.enabled = false
//控制按钮状态
cb_query.enabled = true
pb_add.enabled = true
pb_modify.enabled = true
pb_del.enabled = true
pb_first.postevent(clicked!)
//--------------------------------------------------------------------
```

(13) 在图片按钮 pb_exit 的 CLICKED 事件中加入以下代码，提问是否保存数据，然后退出窗体。

```
//--------------------------------------------------------------------
integer li_rtn
//提问是否保存数据
if dw_1.enabled = false then
   li_rtn = messagebox("提示信息","数据没有保存，需要保存退出吗！",question!,yesno!)
   if li_rtn = 1 then
     pb_save.triggerevent(clicked!)
   end if
end if
close(parent)
//--------------------------------------------------------------------
```

(14) 在数据窗体 dw_1 的 CLICKED 事件中加入以下代码，选择单击的行，然后调用窗口的自定义事件 UE_SEARCH。

```
//--------------------------------------------------------------------
this.selectrow(0,false)
if row < 1 then return
//选择第一行
this.selectrow(row,true)
il_row = row
//调用窗体事件
parent.event ue_search()
//--------------------------------------------------------------------
```

(15) 在数据窗体dw_1的RETRIEVEEND事件中加入以下代码,选择第一行,赋值总行数和当前行,然后调用窗体的自定义事件UE_SEARCH。

```
//--------------------------------------------------------------------
this.selectrow(0,false)
if rowcount < 1 then
   dw_2.reset()
   dw_2.insertrow(0)
   return
end if
//选择第一行
this.selectrow(1,true)
il_row = 1
il_rowcount = rowcount
sle_record.text = string(il_rowcount)
if il_rowcount <= 1 then
   pb_next.enabled = false
   pb_buttom.enabled = false
end if
//调用窗体事件
parent.event ue_search()
//--------------------------------------------------------------------
```

现在，制作完了祖先窗体，后面的很多窗体就都可以从它来继承了。这样很多重复的代码在后代窗体中没必要重复的编写了，具体的后面将介绍。

6.6.7 完成药品信息维护功能

(1) 选择File｜Inherit命令，弹出如图6.36所示的Inherit from Object对话框，在Objects of Type下拉列表框中选择Windows选项，然后选中w_sheet对象。

图 6.36　用祖先继承窗体图

(2)　单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_ypxx”，设置窗体的属性为 main!，并为其添加如图 6.37 所示的控件。

图 6.37　【药品信息维护】窗体

(3)　各控件的属性设置如表 6.7 所示。

表 6.7　【药品信息维护】窗体控件属性设置

控件类型	对 象 名	属　性	取值(说明)
Window	w_ypxx	Title	药品信息维护
Window	w_ypxx	WindowType	main!
Window	w_ypxx	ICON	\ICON\医院.ico
Picturebutton	pb_first	Picture_name	\ICON\首记录.bmp
Picturebutton	pb_prior	Picture_name	\ICON\上记录.bmp
Picturebutton	pb_next	Picture_name	\ICON\下记录.bmp
Picturebutton	pb_buttom	Picture_name	\ICON\尾记录.bmp
Picturebutton	pb_add	Picture_name	\ICON\新增.bmp
Picturebutton	pb_modify	Picture_name	\ICON\修改.bmp
Picturebutton	pb_del	Picture_name	\ICON\删除.bmp
Picturebutton	pb_save	Picture_name	\ICON\保存.bmp
Picturebutton	pb_cancel	Picture_name	\ICON\取消.bmp
Picturebutton	pb_exit	Picture_name	\ICON\关闭.bmp
CommandButton	cb_query	text	搜索
Statictext	st_1	text	记录数:

Statictext	st_2	text	拼音码：
Statictext	st_3	text	编号：
Statictext	st_4	text	名称：

续表

控件类型	对 象 名	属 性	取值(说明)
Singlelineedit	sle_record	text	
Singlelineedit	sle_pym	text	
Singlelineedit	sle_bh	text	
Singlelineedit	sle_mc	text	
Datawindow	dw_1	dataobject	d_ypxx
Datawindow	dw_2	dataobject	d_ypxx _detail

由于实例变量在祖先窗体 w_sheet 中已经定义，在后代窗体中直接使用就可以，没必要再重复的定义。后代窗体也可以定义与祖先窗体不重复的实例变量，这里为了需要，在后代再定义实例变量。

```
string is_status
```

(4) 在窗体 w_ypxx 的 OPEN 事件中加入以下代码，在数据窗体 dw_2 中插入一个空行。因为数据窗体 dw_2 的对象为 Freeform 类型，当没有值时，显示为空白，所以需要插入一个空行。

```
//--------------------------------------------------------------------
dw_2.insertrow(0)
//--------------------------------------------------------------------
```

(5) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户在拼音码、编号或名称框中输入的内容检索药品信息的数据。

```
//--------------------------------------------------------------------
string ls_select,ls_txm,ls_bh,ls_xm

ls_txm = trim(sle_pym.text) // 拼音码
ls_bh = trim(sle_bh.text) // 编号
ls_xm = trim(sle_mc.text) // 名称

ls_select = ""
// 拼音码
if not isnull(ls_txm) and ls_txm <> "" then
   ls_txm = "%" + ls_txm + "%"
   ls_select = ls_select + " 拼音码 LIKE '" + ls_txm + "'"
end if
// 编号
if not isnull(ls_bh) and ls_bh <> "" then
   ls_bh = "%" + ls_bh + "%"
   if ls_select = "" then
```

```
        ls_select = " 编号 LIKE '" + ls_bh + "'"
      else
        ls_select = ls_select + " AND 编号 LIKE '" + ls_bh + "'"
      end if
    end if
    // 名称
    if not isnull(ls_xm) and ls_xm <> "" then
      ls_xm = "%" + ls_xm + "%"
      if ls_select = "" then
        ls_select = " 名称 LIKE '" + ls_xm + "'"
      else
        ls_select = ls_select + " AND 名称 LIKE '" + ls_xm + "'"
      end if
    end if

    //获得 SQL 语法
    if ls_select <> "" then
      ls_select = is_sql + " where " + ls_select
    else
      ls_select = is_sql
    end if
    //重新连接数据库
    dw_1.SetSQLSelect(ls_select)
    il_rowcount = dw_1.retrieve() //重新检索数据
    sle_record.text = string(il_rowcount)
    //--------------------------------------------------------------------
```

用户应该都知道，SQL 语句里，模糊查询都需要在检索条件中加上“%”，上面的检索中就用到了模糊查询。在实际的编程做项目时，除非有特别的限制，一般的查询都最好用模糊查询。

(6) 在按钮pb_add的CLICKED事件中加入以下代码,控制数据窗体的状态,并且在数据窗体dw_2中插入一行。因为在祖先窗体代码中已经在数据窗体 dw_1 中插入了一行，所以在这里没必要再编写代码，只需要做一些状态控制就可以了。

```
    //--------------------------------------------------------------------
    //设置数据窗体状态
    dw_1.enabled = false
    dw_2.enabled = true
    dw_2.reset()
    //插入一行
    dw_2.insertrow(1)
    //--------------------------------------------------------------------
```

(7) 在按钮 pb_save 的 CLICKED 事件中加入以下代码，保存数据窗体 dw_2 中修改的数据。

```
    //--------------------------------------------------------------------
    //没有数据则返回
```

```
if dw_2.rowcount() < 1 then return
//接收数据
dw_2.ACCEPTTEXT()
//给 dw_1 赋值
dw_1.setitem(il_row,"编号",dw_2.getitemstring(1,"编号"))
dw_1.setitem(il_row,"名称",dw_2.getitemstring(1,"名称"))
dw_1.setitem(il_row,"规格",dw_2.getitemstring(1,"规格"))
dw_1.setitem(il_row,"整量单位",dw_2.getitemstring(1,"整量单位"))
dw_1.setitem(il_row,"散量单位",dw_2.getitemstring(1,"散量单位"))
dw_1.setitem(il_row,"入库单价",dw_2.getitemdecimal(1,"入库单价"))
dw_1.setitem(il_row,"出库单价",dw_2.getitemdecimal(1,"出库单价"))
dw_1.setitem(il_row,"批发价",dw_2.getitemdecimal(1,"批发价"))
dw_1.setitem(il_row,"整散比",dw_2.getitemdecimal(1,"整散比"))
dw_1.setitem(il_row,"分类",dw_2.getitemstring(1,"分类"))
dw_1.setitem(il_row,"费用归类",dw_2.getitemstring(1,"费用归类"))
dw_1.setitem(il_row,"拼音码",dw_2.getitemstring(1,"拼音码"))
dw_1.setitem(il_row,"效期",dw_2.getitemnumber(1,"效期"))
dw_1.setitem(il_row,"上限",dw_2.getitemdecimal(1,"上限"))
dw_1.setitem(il_row,"下限",dw_2.getitemdecimal(1,"下限"))

//提交数据
if dw_2.update() = 1 then
   commit;
   dw_1.enabled = true
   dw_2.enabled = false
   messagebox("提示","保存数据成功!")
else
   //回滚数据
   rollback;
   messagebox("提示","保存数据失败!")
end if
il_rowcount = dw_1.rowcount()
sle_record.text = string(il_rowcount) //显示记录数
//--------------------------------------------------------------------
```

(8) 在窗口的自定义事件 UE_SEARCH 中加入以下代码，得到数据窗体中当前行的药品编号并在数据窗体 dw_2 中检索出相应的详细数据。

```
//--------------------------------------------------------------------
string ls_bh
//得到编号并检索数据
ls_bh = dw_1.getitemstring(il_row,"编号")
dw_2.retrieve(ls_bh)
//--------------------------------------------------------------------
```

(9) 在按钮 pb_del 的 CLICKED 事件中加入以下代码， 删除数据窗体 dw_1 中选择的记录，并重

新检索数据。

```
//--------------------------------------------------------------------
string ls_bh
if il_row > 0 and il_rowcount > 0 then
   //删除
   ls_bh = dw_1.getitemstring(il_row,"编号")

   DELETE FROM 药品信息
  WHERE 药品信息.编号 = :ls_bh ;

   //提交数据库
   if sqlca.sqlcode = 0 then
     commit;
     messagebox("提示","删除成功！")
   else
     rollback;
     messagebox("提示","删除失败！")
   end if
   if il_row >= 1 then
     il_row -= 1
   end if
   dw_1.retrieve()
   if il_row < 1 then return
   //焦点上移
   dw_1.selectrow(0,false)
   dw_1.selectrow(il_row,true)
   dw_1.scrolltorow(il_row)
   //调用窗口事件
   parent.event ue_search()
end if
//--------------------------------------------------------------------
```

由于按钮 pb_modify、pb_cancel 和 pb_exit 的代码沿用祖先窗体的就可以了，所以在这里不需要重复编写，程序执行的时候会直接执行祖先的代码。

需要新增药品信息，单击相应的按钮，输入新信息后单击【保存】按钮即可。需要删除一条信息，选中该信息后单击【删除】按钮。

这样，完成了药品信息维护功能的制作，下面将介绍收费项目维护功能的制作方法。

6.6.8 完成收费项目维护功能

(1) 选择 File｜Inherit 命令，弹出如图 6.36 所示的 Inherit from Object 对话框，在 Objects of Type 下拉列表中选择 Windows 选项，然后选中 w_sheet 对象。

(2) 单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_sfxx”，设置窗体的属性为 main!，并为其添加如图 6.38 所示的控件。

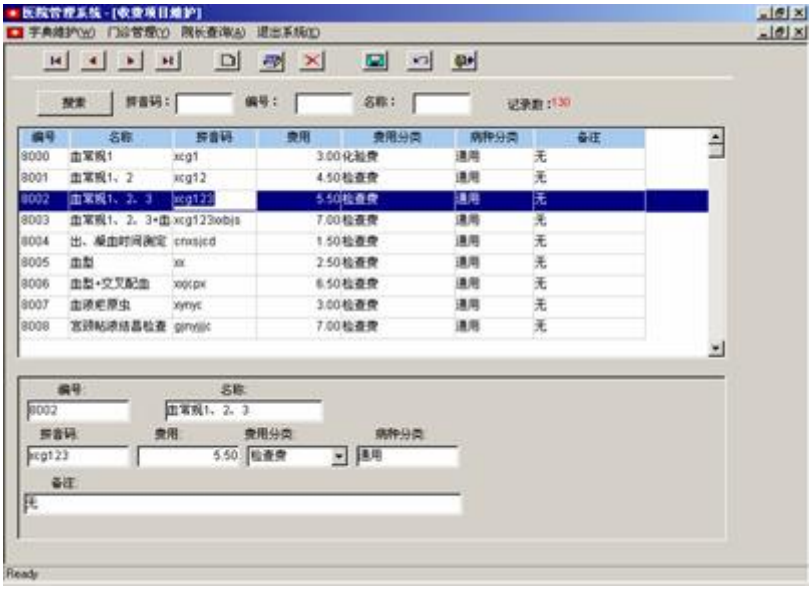


图 6.38 【收费项目维护】窗体

(3) 各控件的属性设置如表 6.8 所示。

表 6.8 【收费项目维护】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
Window	w_sfxx	Title	收费项目维护
Window	w_sfxx	WindowType	main!
Window	w_sfxx	ICON	\ICON\医院.ico
Picturebutton	pb_first	Picture_name	\ICON\首记录.bmp
Picturebutton	pb_prior	Picture_name	\ICON\上记录.bmp
Picturebutton	pb_next	Picture_name	\ICON\下记录.bmp
Picturebutton	pb_buttom	Picture_name	\ICON\尾记录.bmp
Picturebutton	pb_add	Picture_name	\ICON\新增.bmp
Picturebutton	pb_modify	Picture_name	\ICON\修改.bmp
Picturebutton	pb_del	Picture_name	\ICON\删除.bmp
Picturebutton	pb_save	Picture_name	\ICON\保存.bmp
Picturebutton	pb_cancel	Picture_name	\ICON\取消.bmp
Picturebutton	pb_exit	Picture_name	\ICON\关闭.bmp
CommandButton	cb_query	text	搜索
Statictext	st_1	text	记录数:
Statictext	st_2	text	拼音码:
Statictext	st_3	text	编号:
Statictext	st_4	text	名称:

续表

控件类型	对 象 名	属 性	取值(说明)
Singlelineedit	sle_record	text	
Singlelineedit	sle_pym	text	
Singlelineedit	sle_bh	text	
Singlelineedit	sle_mc	text	
Datawindow	dw_1	dataobject	d_sfxx
Datawindow	dw_2	dataobject	d_sfxx _detail

由于实例变量在祖先窗体“w_sheet”中已经定义，在后代窗体中直接使用就可以，没必要再重复的定义。

(4) 在窗体 w_sfxx 的 OPEN 事件中加入以下代码，在数据窗体 dw_2 中插入一个空行。因为数据窗体 dw_2 的数据窗体对象为 Freeform 类型，当没有值时，显示为空白，所以需要插入一个空行。

```
//--------------------------------------------------------------------
dw_2.insertrow(0)
//--------------------------------------------------------------------
```

(5) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户在【拼音码】、【编号】或【名称】文本框中输入的内容检索档案的数据。

```
//--------------------------------------------------------------------
string ls_select,ls_txm,ls_bh,ls_xm

ls_txm = trim(sle_pym.text) // 拼音码
ls_bh = trim(sle_bh.text) // 编号
ls_xm = trim(sle_mc.text) // 名称

ls_select = ""
// 拼音码
if not isnull(ls_txm) and ls_txm <> "" then
   ls_txm = "%" + ls_txm + "%"
   ls_select = ls_select + " 拼音码 LIKE '" + ls_txm + "'"
end if
// 编号
if not isnull(ls_bh) and ls_bh <> "" then
   ls_bh = "%" + ls_bh + "%"
   if ls_select = "" then
     ls_select = " 编号 LIKE '" + ls_bh + "'"
   else
     ls_select = ls_select + " AND 编号 LIKE '" + ls_bh + "'"
   end if
end if
// 名称
if not isnull(ls_xm) and ls_xm <> "" then
   ls_xm = "%" + ls_xm + "%"
   if ls_select = "" then
     ls_select = " 名称 LIKE '" + ls_xm + "'"
   else
     ls_select = ls_select + " AND 名称 LIKE '" + ls_xm + "'"
   end if
end if

//获得 SQL 语法
if ls_select <> "" then
```

```
    ls_select = is_sql + " where " + ls_select
else
    ls_select = is_sql
end if
//重新连接数据库
dw_1.SetSQLSelect(ls_select)
il_rowcount = dw_1.retrieve() //重新检索数据
sle_record.text = string(il_rowcount)
//--------------------------------------------------------------------
```

(6) 在按钮 pb_add 的 CLICKED 事件中加入以下代码，控制数据窗体的状态，并且在数据窗体 dw_2 中插入一行。因为在祖先窗体代码中已经在数据窗体 dw_1 中插入了一行，所以在这里没必要再编写代码，只需要做一些状态控制就可以了。

```
//--------------------------------------------------------------------
//设置数据窗体状态
dw_1.enabled = false
dw_2.enabled = true
dw_2.reset()
//插入一行
dw_2.insertrow(1)
//--------------------------------------------------------------------
```

(7) 在按钮 pb_save 的 CLICKED 事件中加入以下代码， 保存数据窗体 dw_2 中修改的数据。

```
//--------------------------------------------------------------------
//没有数据则返回
if dw_2.rowcount() < 1 then return
//接收数据
dw_2.ACCEPTTEXT()

//给 dw_1 赋值
dw_1.setitem(il_row,"编号",dw_2.getitemstring(1,"编号"))
dw_1.setitem(il_row,"名称",dw_2.getitemstring(1,"名称"))
dw_1.setitem(il_row,"拼音码",dw_2.getitemstring(1,"拼音码"))
dw_1.setitem(il_row,"费用",dw_2.getitemdecimal(1,"费用"))
dw_1.setitem(il_row,"费用分类",dw_2.getitemstring(1,"费用分类"))
dw_1.setitem(il_row,"病种分类",dw_2.getitemstring(1,"病种分类"))
dw_1.setitem(il_row,"备注",dw_2.getitemstring(1,"备注"))

//提交数据
if dw_2.update() = 1 then
    commit;
    dw_1.enabled = true
    dw_2.enabled = false
    messagebox("提示","保存数据成功!")
```

```
else
   //回滚数据
   rollback;
   messagebox("提示","保存数据失败!")
end if
il_rowcount = dw_1.rowcount()
sle_record.text = string(il_rowcount)  //显示记录数
//--------------------------------------------------------------------
```

(8) 在窗体的自定义事件 UE_SEARCH 中加入以下代码， 得到数据窗体中当前行的药品编号并在数据窗体 dw_2 中检索出相应的详细数据。

```
//--------------------------------------------------------------------
string ls_bh
//得到编号并检索数据
ls_bh = dw_1.getitemstring(il_row,"编号")
dw_2.retrieve(ls_bh)
//--------------------------------------------------------------------
```

(9) 在按钮 pb_del 的 CLICKED 事件中加入以下代码， 删除数据窗体 dw_1 中选择的记录，并重新检索数据。

```
//--------------------------------------------------------------------
string ls_bh
if il_row > 0 and il_rowcount > 0 then
   //删除
   ls_bh = dw_1.getitemstring(il_row,"编号")

   DELETE FROM 门诊收费项目
  WHERE 门诊收费项目.编号 = :ls_bh ;

   //提交数据库
   if sqlca.sqlcode = 0 then
     commit;
     messagebox("提示","删除成功！")
   else
     rollback;
     messagebox("提示","删除失败！")
   end if
   if il_row >= 1 then
     il_row -= 1
   end if
   dw_1.retrieve()
   if il_row < 1 then return
   //焦点上移
   dw_1.selectrow(0,false)
```

```
    dw_1.selectrow(il_row,true)
    dw_1.scrolltorow(il_row)
    //调用窗口事件
    parent.event ue_search()
end if
//--------------------------------------------------------------------
```

由于按钮 pb_modify、pb_del、pb_cancel 和 pb_exit 的代码沿用祖先的就可以，所以在这里不需要重复编写，程序执行的时候会直接执行祖先的代码。

需要新增收费项目信息，单击相应的按钮，输入新信息后单击【保存】按钮即可。需要删除一条信息，选中该信息后单击【删除】按钮。

这样，完成了收费项目维护功能，下面将介绍门诊挂号功能的制作方法。

6.6.9 完成门诊挂号功能

(1) 选择 File | New 命令，在弹出的 New 对话框中打开 PB Object 选项卡，然后单击 Window 对象，新建一个窗体，将其 Name 属性改为“w_mzgh1”，设置窗体的属性为 main!，并为其添加如图 6.39 所示的控件。

(2) 各控件的属性设置如表 6.9 所示。

图 6.39 【门诊挂号】窗体

表 6.9 【门诊挂号】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
window	w_mzgh1	Title	门诊挂号
window	w_mzgh1	WindowType	main!
window	w_mzgh1	ICON	\ICON\医院.ico
CommandButton	cb_add	text	新增
CommandButton	cb_save	text	保存
CommandButton	cb_cancel	text	取消
CommandButton	cb_exit	text	退出
groupbox	gb_1	text	挂号信息
statictext	st_1	text	姓名:
statictext	st_2	text	性别:
statictext	st_3	text	年龄:

statictext	st_4	text	民族：
statictext	st_5	text	费用类型：
statictext	st_6	text	挂号类型：
statictext	st_7	text	挂号科室：
statictext	st_8	text	医生：
statictext	st_9	text	挂号费用：
statictext	st_10	text	挂号时间：
statictext	st_11	text	必填
statictext	st_12	text	选填
statictext	st_13	text	自动
Singlelineedit	sle_xm	text	
Singlelineedit	sle_nl	text	
Dropdownlistbox	ddlb_xb	items	男/女
Dropdownlistbox	ddlb_mz	items	汉族/维族/藏族/蒙古族
Dropdownlistbox	ddlb_fylx	text	
Dropdownlistbox	ddlb_ghlx	text	
Dropdownlistbox	ddlb_ghks	text	
Dropdownlistbox	ddlb_ys	text	
Singlelineedit	sle_1	backcolor	fushsia
Singlelineedit	sle_2	backcolor	fushsia
editmask	em_1	editdatatype	Numericmask!
editmask	em_2	editdatatype	Numericmask!

(3) 在窗体 w_mzgh1 的 OPEN 事件中加入以下代码，先设置单个控件的状态，然后从数据库取得收费类型，挂号类型，科室资料，医生资料等信息添加到相应的下拉列表框中。

```
//--------------------------------------------------------------------
string ls_temp

//设置单个控件的状态
sle_xm.displayonly = true
ddlb_xb.enabled = false
sle_nl.displayonly = true
ddlb_mz.enabled = false
ddlb_fylx.enabled = false
ddlb_ghks.enabled = false
ddlb_ghlx.enabled = false
//////添加下拉列表框的值
//收费类型
 DECLARE lc_sflx CURSOR FOR
  SELECT DISTINCT(门诊收费项目.名称)
   FROM 门诊收费项目  ;
OPEN lc_sflx ;
do while sqlca.sqlcode = 0
   fetch lc_sflx into :ls_temp ;
   if sqlca.sqlcode = 0 then
```

```
      ddlb_fylx.additem(ls_temp)
    end if
loop
close lc_sflx ;

//挂号类型
 DECLARE lc_ghlx CURSOR FOR
  SELECT DISTINCT(门诊挂号类型.挂号类型)
   FROM 门诊挂号类型  ;
OPEN lc_ghlx ;
do while sqlca.sqlcode = 0
   fetch lc_ghlx into :ls_temp ;
   if sqlca.sqlcode = 0 then
     ddlb_ghlx.additem(ls_temp)
   end if
loop
close lc_ghlx ;

//科室资料
 DECLARE lc_ghks CURSOR FOR
  SELECT DISTINCT(科室资料.名称)
   FROM 科室资料  ;
OPEN lc_ghks ;
do while sqlca.sqlcode = 0
   fetch lc_ghks into :ls_temp ;
   if sqlca.sqlcode = 0 then
     ddlb_ghks.additem(ls_temp)
   end if
loop
close lc_ghks ;

//医生资料
 DECLARE lc_ghys CURSOR FOR
  SELECT DISTINCT(医生资料.姓名)
   FROM 医生资料  ;
OPEN lc_ghys ;
do while sqlca.sqlcode = 0
   fetch lc_ghys into :ls_temp ;
   if sqlca.sqlcode = 0 then
     ddlb_ys.additem(ls_temp)
   end if
loop
close lc_ghys ;
//赋值日期
em_2.text = string(today(),'yyyy-mm-dd')
//--------------------------------------------------------------------
```

(4) 在按钮 cb_add 的 CLICKED 事件中加入以下代码，重新设置各控件的状态，方便继续增加新的挂号。

```
//--------------------------------------------------------------------
sle_xm.text = ''
sle_nl.text = ''
//设置各控件的状态
sle_xm.displayonly = false
ddlb_xb.enabled = true
sle_nl.displayonly = false
ddlb_mz.enabled = true
ddlb_fylx.enabled = true
ddlb_ghks.enabled = true
ddlb_ghlx.enabled = true
ddlb_ys.enabled = true
//--------------------------------------------------------------------
```

(5) 在按钮 pb_save 的 CLICKED 事件中加入以下代码，保存输入的门诊挂号的信息，并将输入的病人信息添加到病人信息库表中。

```
//--------------------------------------------------------------------
string ls_xm,ls_xb,ls_mz,ls_fylx,ls_bh,ls_brbh
string ls_ghlx,ls_ghks,ls_ys
decimal ld_ghfy
date ld_ghsj
integer li_nl

//得到输入的值
ls_xm = sle_xm.text
ls_xb = ddlb_xb.text
ls_mz = ddlb_mz.text
li_nl = integer(sle_nl.text)
ls_fylx = ddlb_fylx.text
ls_ghlx = ddlb_ghlx.text
ls_ghks = ddlb_ghks.text
ls_ys  = ddlb_ys.text
ld_ghfy = Dec(em_1.text)
ld_ghsj = today()
//姓名不能为空
if isnull(ls_xm) or ls_xm = '' then
   messagebox("提示","姓名不能为空！")
   sle_xm.setfocus()
   return
end if
//费用类型不能为空
```

```
if isnull(ls_fylx) or ls_fylx = '' then
   messagebox("提示","费用类型不能为空！")
   ddlb_fylx.setfocus()
   return
end if
//挂号科室不能为空
if isnull(ls_ghks) or ls_ghks = '' then
   messagebox("提示","挂号科室不能为空！")
   ddlb_ghks.setfocus()
   return
end if
//医生不能为空!
if isnull(ls_ys) or ls_ys = '' then
   messagebox("提示","医生不能为空！")
   ddlb_ys.setfocus()
   return
end if

//得到病人信息库的编号
select max(编号)
into:ls_brbh
from 病人信息库;
//生产编号
if isnull(ls_brbh) then
   ls_brbh = "BR0000001"
else
   ls_brbh = "BR" + string(long(right(ls_brbh,7) ) + 1,'0000000')
end if
//插入数据到病人信息库
 INSERT INTO 病人信息库
     ( 编号,
      姓名,
      性别,
      年龄,
      民族,
      费用类型)
 VALUES ( :ls_brbh,
      :ls_xm,
      :ls_xb,
      :li_nl,
      :ls_mz,
      :ls_fylx
     ) ;
//插入失败则回滚数据并返回
 if sqlca.sqlcode <> 0 then
    messagebox("提示","插入病人信息失败！")
```

```
    rollback ;
    return
  end if

//得到门诊挂号的编号
select max(编号)
into:ls_bh
from 门诊挂号;
//生成门诊挂号编号
if isnull(ls_bh) then
  ls_bh = "GH0000001"
else
  ls_bh = "GH" + string(long(right(ls_bh,7) ) + 1,'0000000')
end if
//插入数据到门诊挂号表
INSERT INTO 门诊挂号
    ( 编号,
     病人编号,
     姓名,
     性别,
     挂号科室,
     费用类型,
     挂号类型,
     挂号费用,
     医生,
     时间,
     是否已划价 )
 VALUES ( :ls_bh,
     :ls_brbh,
     :ls_xm,
     :ls_xb,
     :ls_ghks,
     :ls_fylx,
     :ls_ghlx,
     :ld_ghfy,
     :ls_ys,
     :ld_ghsj ,
       '否') ;
//插入失败则返回
if sqlca.sqlcode <> 0 then
    messagebox("提示","插入挂号信息失败！")
    rollback ;
    return
end if
//提交数据
commit ;
```

```
messagebox("提示","挂号成功！")
//改变各个控件的状态
sle_xm.displayonly = true
ddlb_xb.enabled = false
sle_nl.displayonly = true
ddlb_mz.enabled = false
ddlb_fylx.enabled = false
ddlb_ghks.enabled = false
ddlb_ghlx.enabled = false
ddlb_ys.enabled = false
//--------------------------------------------------------------------
```

(6) 在按钮 cb_cancel 的 CLICKED 事件中加入以下代码，回滚数据。

```
//--------------------------------------------------------------------
rollback; //回滚数据
//--------------------------------------------------------------------
```

(7) 在按钮 cb_exit 的 CLICKED 事件中加入以下代码，关闭窗口。

```
//--------------------------------------------------------------------
close(parent)
//--------------------------------------------------------------------
```

需要新增门诊挂号信息，单击相应的按钮，输入新信息后单击【保存】按钮即可。这样，完成了门诊挂号功能，下面将介绍门诊划价功能的制作方法。

6.6.10 完成门诊划价功能

(1) 选择 File｜Inherit 命令，弹出如图 6.36 所示的 Inherit from Object 对话框，在 Object of Type 下拉列表框中选择 Windows 选项，然后选中 w_sheet 对象。

(2) 单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_mzhj”，设置窗体的属性为 main!，并为其添加如图 6.40 所示的控件。

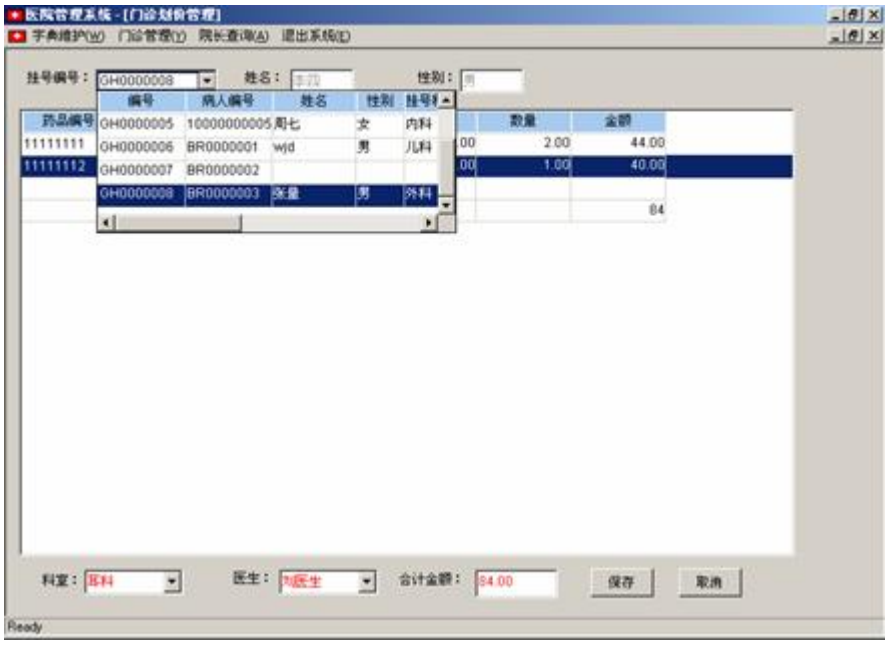


图 6.40 【门诊划价管理】窗体

(3) 各控件的属性设置如表 6.10 所示。

表 6.10 【门诊划价管理】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
window	w_mzhj	Title	门诊划价
window	w_mzhj	WindowType	main!
window	w_mzhj	ICON	\ICON\医院.ico
statictext	st_2	text	挂号编号：
statictext	st_3	text	姓名：
statictext	st_4	text	性别：
datawindow	dw_select	dataobject	d_mzhj_select
Singlelineedit	sle_xm	text	
Singlelineedit	sle_xb	text	

续表

控件类型	对 象 名	属 性	取值(说明)
datawindow	dw_1	dataobject	d_mzhj
datawindow	dw_2	dataobject	d_ypxx
statictext	st_5	text	科室：
statictext	st_6	text	医生：
statictext	st_7	text	合计金额：
dropdownlistbox	ddlb_1	text	
dropdownlistbox	ddlb_2	text	
editmask	em_1	maskedittype	Numericmask!
commandbutton	cb_save	text	保存
commandbutton	cb_cancel	text	取消

由于实例变量在祖先窗体“w_sheet”中已经定义，在后代窗体中直接使用就可以，没必要再重复的定义。但后代根据自己的需要，也可以定义自己需要的变量，这里定义变量来保存挂号编号。

```
string is_ghbh
```

(4) 在窗体w_mzhj的OPEN事件中加入以下代码,将数据窗体dw_select连接到数据窗体dw_2，利用游标得到医院科室和医生的信息添加到下拉列表框中。

```
//--------------------------------------------------------------------
string ls_ks,ls_ys

dw_select.settransobject(sqlca)
dw_select.retrieve()
dw_2.insertrow(0)

//增加科室下拉列表框
//定义游标
declare ldc_ks cursor for
  SELECT 名称  FROM 科室资料 ;
open ldc_ks; //打开游标
//打开成功
do while sqlca.sqlcode = 0
   //赋值
   fetch ldc_ks into :ls_ks ;
   if sqlca.sqlcode = 0 then
     ddlb_1.additem(ls_ks)
   end if
loop
close ldc_ks; //关闭游标

//增加医生下拉列表框
declare ldc_ys cursor for
  SELECT 姓名  FROM 医生资料 ;
open ldc_ys;
//取得游标值
do while sqlca.sqlcode = 0
   fetch ldc_ys into :ls_ys ;
   if sqlca.sqlcode = 0 then
     ddlb_2.additem(ls_ys)
   end if
loop
close ldc_ys;
//检索出药品信息
dw_2.retrieve()
//--------------------------------------------------------------------
```

(5) 在数据窗体 dw_select 的 ITEMCHANGED 事件中加入以下代码，在数据窗体 dw_select 的值发生变化后，得到数据窗体的基本信息并赋值给文本框和下拉列表框。

```
//--------------------------------------------------------------------
string ls_xm,ls_xb,ls_ghks,ls_ys,ls_sfyhj
```

```
//没有数据则返回
if row < 1 then return
this.selectrow(0,false)
this.selectrow(row,true)
//得到值

is_ghbh  = trim(data)
 SELECT 门诊挂号.姓名,
    门诊挂号.性别,
    门诊挂号.挂号科室,
    门诊挂号.医生,
    门诊挂号.是否已划价
  INTO :ls_xm,
    :ls_xb,
    :ls_ghks,
    :ls_ys,
    :ls_sfyhj
  FROM 门诊挂号
   where 编号 = :is_ghbh ;

if ls_sfyhj = '是' then
   messagebox("提示","该挂号单已经划价！")
   return
end if
//赋值
sle_xm.text = ls_xm
sle_xb.text = ls_xb
ddlb_1.text = ls_ghks
ddlb_2.text = ls_ys
//插入一行
dw_1.reset()
dw_1.insertrow(0)
//--------------------------------------------------------------------
```

(6) 在数据窗体 dw_select 的 RETRIEVEEND 事件中加入以下代码，在数据窗体 dw_select 检索完后，得到数据窗体的基本信息并赋值给文本框和下拉列表框。

```
//--------------------------------------------------------------------
string ls_xm,ls_xb,ls_ghks,ls_ys

//没有数据则返回
if rowcount < 1 then return
this.selectrow(0,false)
this.selectrow(1,true)
```

```
//得到数值
is_ghbh = this.getitemstring(1,"编号")
ls_xm = this.getitemstring(1,"姓名")
ls_xb = this.getitemstring(1,"性别")
ls_ghks = this.getitemstring(1,"挂号科室")
ls_ys = this.getitemstring(1,"医生")
//赋值
sle_xm.text = ls_xm
sle_xb.text = ls_xb
ddlb_1.text = ls_ghks
ddlb_2.text = ls_ys
dw_1.insertrow(0) //插入一行
//--------------------------------------------------------------------
```

(7) 在数据窗体 dw_1 的 ITEMCHANGED 事件中加入以下代码,在数据窗体 dw_1 的值发生变化后，如果数量列发生变化，则计算金额；如果药品列发生变化，则得到药品的基本信息并显示出来。

```
//--------------------------------------------------------------------
string ls_bh,ls_mc,ls_gg,ls_dw
decimal ld_dj,ld_sl,ld_je
string ls_filter

//数量列变化时
if dwo.name = "数量" then
   //得到数据和单价，计算金额
   ld_sl = dec(data)
   ld_dj = this.getitemdecimal(row,"单价")
   if isnull(ld_dj) then ld_dj = 0
   dw_1.setitem(row,"金额",ld_sl * ld_dj)
   ld_je = dec(dw_1.object.sum_je[row])
   em_1.text = string(ld_Je)
   dw_1.setcolumn("药品编号")
   dw_1.selectrow(0,false)
   dw_1.selectrow(il_row,true)
   dw_1.scrolltorow(il_row)
   dw_1.setcolumn("药品编号")
end if

//药品列发生变化时
if dwo.name = '药品编号' then
   //得到药品信息
   ls_bh = trim(data)
   //得到输入的拼音码，然后过滤数据
   if isnull(ls_bh) or ls_bh = '' then
     ls_bh = '%'
   else
```

```
      ls_bh = '%' + ls_bh + '%'
    end if
    il_row = row
    //设置过滤条件
    ls_filter = "拼音码 like '" + ls_bh + "'"
    //过滤
    dw_2.setfilter(ls_filter)
    dw_2.filter()
    //设置数据窗体状态
    dw_2.visible = true
    dw_2.selectrow(0,false)
    dw_2.selectrow(1,true)
    dw_2.setfocus()
    if dw_2.rowcount() < 1 then
      il_selectrow = 0
    else
      il_selectrow = 1
    end if
  end if
  //--------------------------------------------------------------------
```

(8) 在数据窗体dw_1中增加自定义事件UE_ENTER,选择PowerBuilder本身的事件ue_keydown，在自定义事件UE_ENTER中加入以下代码，在数据窗体dw_1的值发生变化后，如果数量列发生变化，则计算金额，并将光标移动到下一行，以便用户可以继续输入药品。

```
  //--------------------------------------------------------------------
  integer i,li_row
  decimal ld_sl,ld_dj,ld_je
  if key = KeyEnter! then
    i = dw_1.getcolumn()
    //当列为数量列时
    if i = 5 then
      li_row = il_row
      if li_row < 1 then return
      //得到数量和单价，计算金额
      ld_sl = this.getitemdecimal(li_row,"数量")
      ld_dj = this.getitemdecimal(li_row,"单价")
      //赋值金额
      dw_1.setitem(li_row,"金额",ld_sl * ld_dj)
      ld_je = dec(dw_1.object.sum_je[il_row])
      em_1.text = string(ld_Je)
      //判断是否要插入行
      if dw_1.rowcount() = li_row then
        il_row = dw_1.insertrow(0)
      end if
    end if
```

```
    //设置列为编号列
    dw_1.setcolumn(3)
end if
//--------------------------------------------------------------------
```

(9) 在数据窗体 dw_1 的 DOUBLECLICKED 事件中加入以下代码，在数据窗体 dw_1 中双击某一行数据时，得到该行的值赋值给数据窗体 dw_2。

```
//--------------------------------------------------------------------
string ls_bh,ls_mc,ls_gg,ls_dw
decimal ld_dj

//选择双击的行
if row < 1 then return
this.selectrow(0,false)
this.selectrow(row,true)
//得到需要的值
ls_bh = dw_2.getitemstring(row,"编号")
ls_mc = dw_2.getitemstring(row,"名称")
ls_gg = dw_2.getitemstring(row,"规格")
ls_dw = dw_2.getitemstring(row,"整量单位")
ld_dj = dw_2.getitemdecimal(row,"出库单价")
dw_2.visible = false
//赋值
dw_1.setitem(il_row,"药品编号",ls_bh)
dw_1.setitem(il_row,"名称",ls_mc)
dw_1.setitem(il_row,"规格",ls_gg)
dw_1.setitem(il_row,"单位",ls_dw)
dw_1.setitem(il_row,"单价",ld_dj)
dw_1.setitem(il_row,"数量",1)
dw_1.setcolumn("数量")
dw_1.setfocus()
//--------------------------------------------------------------------
```

(10) 在数据窗体 dw_2 中增加自定义事件 UE_ENTER,选择 PowerBuilder 本身的事件 ue_keydown，在自定义事件 UE_ENTER 中加入以下代码，在数据窗体 dw_2 中某一行上单击回车键时，得到该行的值赋值给数据窗体 dw_1，并使数据窗体 dw_2 隐藏。

```
//--------------------------------------------------------------------
string ls_bh,ls_mc,ls_gg,ls_dw
decimal ld_dj

//按向下的箭头时
if key = KeyDownArrow! then
   if il_selectrow = dw_2.rowcount() then return
   il_selectrow += 1
```

```
    dw_2.selectrow(0,false)
    dw_2.selectrow(il_selectrow,true)
    dw_2.scrolltorow(il_selectrow)
end if
//按向上的箭头时
if key = KeyUpArrow! then
    if il_selectrow = 1 then return
    il_selectrow -= 1
    dw_2.selectrow(0,false)
    dw_2.selectrow(il_selectrow,true)
    dw_2.scrolltorow(il_selectrow)
end if
//按回车键时
if key = KeyEnter! then
    if il_selectrow < 1 or il_row < 1 then
      dw_2.visible = false
      return
    end if
    //得到需要的值
    ls_bh = dw_2.getitemstring(il_selectrow,"编号")
    ls_mc = dw_2.getitemstring(il_selectrow,"名称")
    ls_gg = dw_2.getitemstring(il_selectrow,"规格")
    ls_dw = dw_2.getitemstring(il_selectrow,"整量单位")
    ld_dj = dw_2.getitemdecimal(il_selectrow,"出库单价")
    dw_2.visible = false
    //赋值
    dw_1.setitem(il_row,"药品编号",ls_bh)
    dw_1.setitem(il_row,"名称",ls_mc)
    dw_1.setitem(il_row,"规格",ls_gg)
    dw_1.setitem(il_row,"单位",ls_dw)
    dw_1.setitem(il_row,"单价",ld_dj)
    dw_1.setitem(il_row,"数量",1)
    dw_1.setcolumn("数量")
    dw_1.setfocus()
end if
//--------------------------------------------------------------------
```

(11) 在按钮 cb_save 的 CLICKED 事件中，加入以下代码，保存门诊划价信息。

```
//--------------------------------------------------------------------
string ls_bh,ls_ghbh,ls_ks,ls_ys,ls_ypbh,ls_sfyhj
integer li_row ,i,li_rtn
decimal ld_je
datetime ldt_time
datawindowchild ldwc_gh
```

```
//得到选择的行
li_row = dw_select.getselectedrow(0)
if li_row < 1 then return
//得到需要信息
ls_ghbh = is_ghbh
ls_ks = dw_select.getitemstring(li_row,"挂号科室")
ls_ys = dw_select.getitemstring(li_row,"医生")
ld_je = dw_1.object.sum_je[1]
ldt_time = datetime(today())
//检查是否已经划价
select 是否已划价
 into :ls_sfyhj
 from 门诊挂号
where 编号 = :ls_ghbh ;
//不能重复划价
if ls_sfyhj = '是' then
   messagebox("提示","该挂号单已经划价！")
   return
end if
//得到划价编号
ls_bh = dw_1.getitemstring(1,"划价编号")
//划价编号为空
if isnull(ls_bh) or ls_bh = '' then
   //得到划价编号+ 1
   select max(编号)
   into:ls_bh
   from 门诊划价;
   //生成划价号
   if isnull(ls_bh) then
     ls_bh = "HJ0000001"
   else
     ls_bh = "HJ" + string(long(right(ls_bh,7) ) + 1,'0000000')
   end if
end if
//赋值划价号
for i = 1 to dw_1.rowcount()
   ls_ypbh = dw_1.getitemstring(i,"药品编号")
   if ls_ypbh <> "" then
     dw_1.setitem(i,"划价编号",ls_bh)
   end if
next
//插入门诊划价表
 INSERT INTO 门诊划价
    ( 编号,
     科室,
     挂号编号,
```

```
        医生,
           划价时间,
        划价金额)
  VALUES ( :ls_bh,
        :ls_ks,
        :ls_ghbh,
        :ls_ys,
           :ldt_time,
        :ld_je) ;
//插入成功则提交
if sqlca.sqlcode <> 0 then
   //回滚
   rollback;
   messagebox("提示","保存数据失败!")
   return
end if
if  dw_1.update() <> 1 then
   //回滚
   rollback;
   messagebox("提示 1","保存数据失败!")
   return
end if
   update 门诊挂号
   set 是否已划价 = '是'
   where 编号 = :ls_ghbh ;
   if sqlca.sqlcode = 0 then
     commit ;
     messagebox("提示","保存数据成功!")
     li_rtn = dw_select.getchild("编号",ldwc_gh)
     if li_rtn = 1 then
       ldwc_gh.settransobject(sqlca)
       ldwc_gh.retrieve()
       dw_1.reset()
     end if
//    dw_select.retrieve()
   else
     //回滚
     rollback;
     messagebox("提示 2","保存数据失败!")
     return
   end if
//--------------------------------------------------------------------
```

(12)　在按钮 cb_cancel 的 CLICKED 事件中，加入以下代码，取消所作的修改。

```
//--------------------------------------------------------------------
```

```
    rollback ;
//--------------------------------------------------------------------
```

这样，完成了门诊划价功能，下面将介绍门诊收费的制作方法。

6.6.11 完成门诊收费功能

(1) 选择 File｜Inherit 命令，弹出如图 6.36 所示的 Inherit from Object 对话框，在 Objects of Type 下拉列表框中选择 Windows 选项，然后选中 w_sheet 对象。

(2) 单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_mzsf”，设置窗体的属性为 main!，并为其添加如图 6.41 所示的控件。

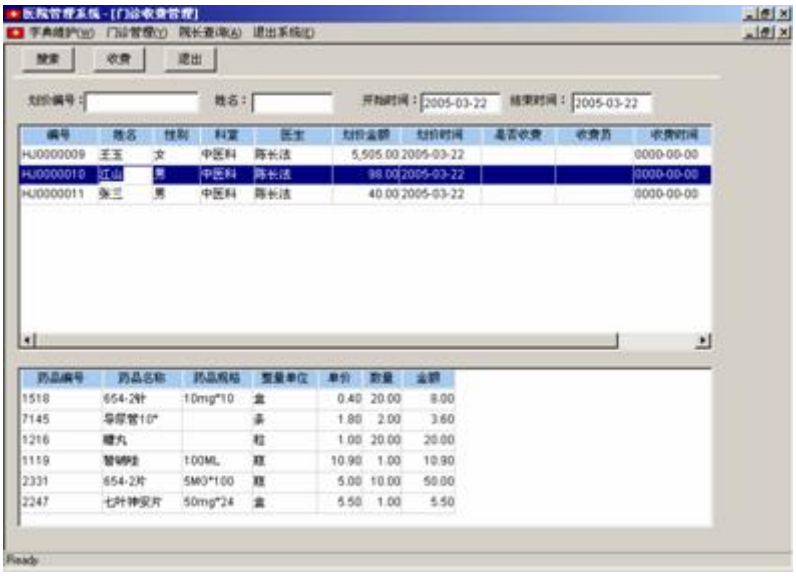


图 6.41 【门诊收费管理】窗体

(3) 各控件的属性设置如表 6.11 所示。

表 6.11 【门诊收费管理】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
Window	w_mzsf	Title	门诊收费管理
Window	w_mzsf	WindowType	main!
Window	w_mzsf	ICON	\ICON\医院.ico
statictext	st_2	text	划价编号：
statictext	st_3	text	姓名：
statictext	st_4	text	开始时间：
statictext	st_5	text	结束时间：
Singlelineedit	sle_bh	text	
Singlelineedit	sle_xm	text	
Editmask	em_start	maskdatatype	Datemask!
Editmask	em_start	Mask	Yyyy-mm-dd
Editmask	em_end	maskdatatype	Datemask!
Editmask	em_end	mask	Yyyy-mm-dd
commandbutton	cb_query	text	搜索
commandbutton	cb_sf	text	收费
commandbutton	cb_exit	text	退出
Datawindow	dw_1	dataobject	d_mzsf
Datawindow	dw_2	dataobject	d_mzsf_detail

(4) 在窗体 w_mzsf 的 OPEN 事件中加入以下代码，在数据窗体 dw_2 中插入一行。

```
//--------------------------------------------------------------------
dw_2.insertrow(0)
//--------------------------------------------------------------------
```

(5) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户在文本框中输入的划价编号、姓名或者输入的起始日期或结束日期等信息，检索出数据窗体 dw_1 中的已经划价而没有收费的数据。

```
//--------------------------------------------------------------------
string ls_select,ls_hjbh,ls_xm,ls_start,ls_end

ls_hjbh  = trim(sle_bh.text) // 编号
ls_xm   = trim(sle_xm.text) // 姓名
ls_start = trim(em_start.text) // 开始时间
ls_end   = trim(em_end.text) //结束时间

ls_select = ""
// 划价编号
if not isnull(ls_hjbh) and ls_hjbh <> "" then
   ls_hjbh = "%" + ls_hjbh + "%"
   ls_select = ls_select + " 门诊划价.编号 LIKE '" + ls_hjbh + "'"
end if
// 病人姓名
if not isnull(ls_xm) and ls_xm <> "" then
   ls_xm = "%" + ls_xm + "%"
   if ls_select = "" then
     ls_select = " 门诊挂号.姓名 LIKE '" + ls_xm + "'"
   else
     ls_select = ls_select + " AND 门诊挂号.姓名 LIKE '" + ls_xm + "'"
   end if
end if
// 开始时间
if not isnull(ls_start) and ls_start <> "" and lower(ls_start) <> 'none' then
   if ls_select = "" then
     ls_select = " convert(char(10)  ,门诊划价.划价时间,120) >= '"
+ ls_start + "'"
   else
     ls_select = ls_select + " AND  convert(char(10)   ,门诊划价.划价时间,120)
>= '" + ls_start + "'"
   end if
end if
// 结束时间
if not isnull(ls_end) and ls_end <> "" and lower(ls_end) <> 'none' then
   if ls_select = "" then
```

```
      ls_select = " convert(char(10)  ,门诊划价.划价时间,120) <= '"
+ ls_end + "'"
   else
      ls_select = ls_select + " AND  convert(char(10)   ,门诊划价.划价时间,120)
<= '" + ls_end + "'"
   end if
end if
//获得 SQL 语法
if ls_select <> "" then
   ls_select = is_sql + " and " + ls_select
else
   ls_select = is_sql
end if

//重新赋值 SQL 语句
dw_1.SetSQLSelect(ls_select)
il_rowcount = dw_1.retrieve() //重新检索数据
sle_record.text = string(il_rowcount)
//--------------------------------------------------------------------
```

(6)　在按钮 cb_sf 的 CLICKED 事件中加入以下代码，对已经划价的单据进行收费操作，更新门诊划价表的收费字段。

```
//--------------------------------------------------------------------
integer li_row
string ls_bh,ls_rtn
datetime ld_today

//判断是否选中行
li_row = dw_1.getselectedrow(0)
if li_row < 1 then return
ls_bh = dw_1.getitemstring(li_row,"编号")
//打开收费窗口
openwithparm(w_mzsf_sf,ls_bh)
//得到返回值
ls_rtn = message.stringparm
//赋值数据窗体
if ls_rtn = 'TRUE' THEN
   dw_1.retrieve()
END IF
//--------------------------------------------------------------------
```

(7)　在按钮 cb_exit 的 CLICKED 事件中加入以下代码关闭窗体。

```
//--------------------------------------------------------------------
close(parent) //关闭窗体
```

```
//--------------------------------------------------------------------
```

上面的程序中用到的窗体 w_mzsf_sf，该窗体是用来门诊收费付款功能的，下面将介绍该窗体的制作方法。

6.6.12 完成门诊收费付款功能

(1) 选择 File｜New 命令，在弹出的 New 对话框中打开 PB Object 选项卡，在 Objects of Type 下拉列表框选择 Window 选项，然后选择 w_sheet 对象，单击 OK 按钮新建一个窗体，将其 Name 属性改为“w_mzsf_sf”，设置窗体的属性为 response!，并为其添加如图 6.42 所示的控件。

图 6.42 【门诊收费付款】窗体

(2) 各控件的属性设置如表 6.12 所示。

表 6.12 【门诊收费付款】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
Window	w_mzsf_sf	Title	门诊收费付款
Window	w_mzsf_sf	WindowType	response!
Window	w_mzsf_sf	ICON	\ICON\医院.ico
Datawindow	dw_1	dataobject	d_mzsf_detail
statictext	st_1	text	合计金额：
statictext	st_2	text	收款：
statictext	st_3	text	找零：
editmask	em_hj	Maskdatatype	Numericmask!
editmask	em_hj	Displayonly	true
editmask	em_sk	Maskdatatype	Numericmask!
editmask	em_zl	Maskdatatype	Numericmask!
editmask	em_zl	Displayonly	true
CommandButton	cb_ok	text	收费
CommandButton	cb_cancel	text	取消

(3) 在窗体 w_mzsf_sf 中定义实例变量。

```
//定义实例变量
string is_bh
```

(4) 在窗体 w_mzsf_sf 的 OPEN 事件中加入以下代码，将数据窗体连接到数据库，并得到数据窗体 dw_1 的原始语法，为后面的检索作准备。

```
//--------------------------------------------------------------------
```

```
datetime ld_today
decimal ld_je

//连接数据库
dw_1.settransobject(sqlca)
//得到参数
is_bh = message.stringparm
if is_bh = '' or isnull(is_bh) then
   messagebox("提示","没有得到划价数据！")
   return
end if
//检索数据
DW_1.RETRIEVE(is_bh)
//得到总的金额
select sum(金额)
 into :ld_je
 from 门诊划价明细
 where 划价编号 = :is_bh ;
//显示金额
 em_hj.text = string(ld_je)
//--------------------------------------------------------------------
```

(5) 在编辑框 em_sk 的 MODIFIED 事件中加入以下代码，根据输入的收款金额，计算出应该找零的金额。

```
//--------------------------------------------------------------------
decimal ld_sk,ld_hj,ld_zl

ld_sk = dec(em_sk.text)
ld_hj = dec(em_hj.text)
ld_zl = ld_sk - ld_hj

if ld_zl < 0 then
   messagebox("提示","收款小于合计金额")
   return
end if
em_zl.text = string(ld_zl)
//--------------------------------------------------------------------
```

(6) 在按钮 cb_ok 的 CLICKED 事件中加入以下代码，更新门诊划价表的门诊收费字段，然后返回到【门诊收费付款】窗体。

```
//--------------------------------------------------------------------
datetime ld_today
decimal ld_hj,ld_sk
```

```
ld_hj = dec(em_hj.text)
ld_sk = dec(em_sk.text)
//付款小于合计金额，则不能付款
if ld_sk < ld_hj then
   messagebox("提示","收款小于合计金额，不能付款！")
   em_sk.setfocus()
   return
end if

ld_today = datetime(today())
//更新门诊划价表的收费信息
update 门诊划价
set 是否收费 = '是',
  收费员  = '收费员',
   收费时间 = :ld_today
where  编号 = :is_bh  ;
//判断是否更新成功
if sqlca.sqlcode <> 0 then
   messagebox("提示","收费失败！")
   rollback;//回滚
   return
end if
//提交
commit ;
messagebox("提示","保存成功")
closewithreturn(parent,'TRUE')
//--------------------------------------------------------------------
```

(7)　在按钮 cb_cancel 的 CLICKED 事件中加入以下代码，关闭门诊收费付款窗体。

```
//--------------------------------------------------------------------
CLOSE(PARENT)
//--------------------------------------------------------------------
```

这样，完成了门诊收费功能，下面将介绍药房发药的制作方法。

6.6.13 完成药房发药功能

(1)　选择 File｜Inherit 命令，弹出如图 6.36 所示的 Inherit from Object 对话框，在 Objects of Type 下拉列表框中选择 Windows 选项，然后选中 w_sheet 对象。

(2)　单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_yffy”，设置窗体的属性为 main!，并为其添加如图 6.43 所示的控件。

图 6.43 【药房发药管理】窗体

(3) 各控件的属性设置如表 6.13 所示。

表 6.13 【药房发药管理】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
window	w_yffy	Title	药房发药管理
window	w_yffy	WindowType	main!
window	w_yffy	ICON	\ICON\医院.ico
statictext	st_2	text	划价编号：
statictext	st_3	text	姓名：
statictext	st_4	text	开始时间：
statictext	st_5	text	结束时间：
Singlelineedit	sle_bh	text	

续表

控件类型	对 象 名	属 性	取值(说明)
Singlelineedit	sle_xm	text	
Editmask	em_start	maskdatatype	Datemask!
Editmask	em_start	Mask	Yyyy-mm-dd
Editmask	em_end	maskdatatype	Datemask!
Editmask	em_end	mask	Yyyy-mm-dd
commandbutton	cb_query	text	搜索
commandbutton	cb_fy	text	发药
commandbutton	cb_exit	text	退出
Datawindow	dw_1	dataobject	d_yffy
Datawindow	dw_2	dataobject	d_sffy_detail

(4) 在窗体 w_yffy 的 OPEN 事件中加入以下代码，在数据窗体 dw_2 中插入一行。

```
//--------------------------------------------------------------------
dw_2.insertrow(0)
```

```
//--------------------------------------------------------------------
```

(5) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户在文本框中输入的划价编号、姓名或者输入的起始日期或结束日期等信息，检索出数据窗体 dw_1 中的已经收费而没有发药的数据。

```
//--------------------------------------------------------------------
string ls_select,ls_hjbh,ls_xm,ls_start,ls_end

ls_hjbh  = trim(sle_bh.text) // 编号
ls_xm   = trim(sle_xm.text) // 姓名
ls_start = trim(em_start.text) // 开始时间
ls_end   = trim(em_end.text) //结束时间

ls_select = ""
// 划价编号
if not isnull(ls_hjbh) and ls_hjbh <> "" then
   ls_hjbh = "%" + ls_hjbh + "%"
   ls_select = ls_select + " 门诊划价.编号 LIKE '" + ls_hjbh + "'"
end if
// 病人姓名
if not isnull(ls_xm) and ls_xm <> "" then
   ls_xm = "%" + ls_xm + "%"
   if ls_select = "" then
     ls_select = " 门诊挂号.姓名 LIKE '" + ls_xm + "'"
   else
     ls_select = ls_select + " AND 门诊挂号.姓名 LIKE '" + ls_xm + "'"
   end if
end if

// 开始时间
if not isnull(ls_start) and ls_start <> "" and lower(ls_start) <> 'none' then
   if ls_select = "" then
     ls_select = " convert(char(10)  ,门诊划价.划价时间,120) >= '"
+ ls_start + "'"
   else
     ls_select = ls_select + " AND  convert(char(10)   ,门诊划价.划价时间,120)
>= '" + ls_start + "'"
   end if
end if

// 结束时间
if not isnull(ls_end) and ls_end <> "" and lower(ls_end) <> 'none' then
   if ls_select = "" then
     ls_select = " convert(char(10)  ,门诊划价.划价时间,120) <= '"
+ ls_end + "'"
   else
```

```
        ls_select = ls_select + " AND  convert(char(10)   ,门诊划价.划价时间,120)
<= '" + ls_end + "'"
      end if
   end if

   //获得 SQL 语法
   if ls_select <> "" then
      ls_select = is_sql + " and " + ls_select
   else
      ls_select = is_sql
   end if

   //重新赋值 SQL 语句
   dw_1.SetSQLSelect(ls_select)
   il_rowcount = dw_1.retrieve() //重新检索数据
   sle_record.text = string(il_rowcount)
   //--------------------------------------------------------------------
```

(6)　在按钮 cb_fy 的 CLICKED 事件中加入以下代码，对已经划价并收费的单据进行发药操作，更新门诊划价表的发药字段。

```
   //--------------------------------------------------------------------
   li_row = dw_1.getselectedrow(0)
   if li_row < 1 then return
   ls_bh = dw_1.getitemstring(li_row,"编号")
   ld_today = datetime(today())
   //更新门诊划价的发药信息
   //定义存储过程
   DECLARE sp_yffy PROCEDURE FOR sf_药品发出
   (:ls_bh);
   EXECUTE sp_yffy; //执行数据存储

   commit ; // 提交
   messagebox("提示","保存成功")
   dw_1.retrieve()
   //--------------------------------------------------------------------
```

(7)　在按钮 cb_exit 的 CLICKED 事件中加入以下代码关闭窗体。

```
   //--------------------------------------------------------------------
   close(parent) //关闭窗体
   //--------------------------------------------------------------------
```

这样，完成了药房发药功能，下面将介绍院长查询的一些报表的制作方法。

6.6.14 完成科室挂号量功能

本实例中，由于篇幅的关系，这里以科室挂号量和药品库存查询为例来介绍院长查询的报表，实际中的报表会多很多，本小节介绍科室挂号量的功能，下一小节介绍药品库查询功能。

(1) 选择 File | New 命令，在弹出 New 对话框中打开 PB Object 选项卡，在 Objecbs of Type 下拉列表框选择 Window，然后选择 w_sheet 对象，单击 OK 按钮新建一个窗体，将其 Name 属性改为“w_ksghl”，设置窗体的属性为 main!，并为其添加如图 6.44 所示的控件。

图 6.44 【科室挂号量】窗体

(2) 各控件的属性设置如表 6.14 所示。

表 6.14 【科室挂号量】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
Window	w_ksghl	Title	科室挂号量
Window	w_ksghl	WindowType	main!
Window	w_ksghl	ICON	\ICON\医院.ico
statictext	st_1	text	开始时间：
statictext	st_2	text	结束时间：
Editmask	em_start	maskdatatype	Datemask!
Editmask	em_start	mask	Yyyy-mm-dd
Editmask	em_end	maskdatatype	Datemask!
Editmask	em_end	mask	Yyyy-mm-dd
CommandButton	cb_query	text	搜索
Datawindow	dw_1	dataobject	d_ksghl

在窗体 w_ksghl 中定义实例变量，在保存数据窗体 dw_1 的原始语法，为以后数据检索做准备。

```
string is_sql
```

(3) 在窗体 w_ksghl 的 OPEN 事件中加入以下代码，将数据窗体连接到数据库并得到数据窗体的原始语法。

```
//--------------------------------------------------------------------
string ls_today,ls_curmonth

//将数据窗体连接到数据库
dw_1.settransobject(sqlca)
//得到原始语法，为检索做准备
is_sql = dw_1.getsqlselect()
```

```
//得到当天和当月 1 号
ls_today = string(today(),"yyyy-mm-dd")
ls_curmonth = left(ls_today,8)  + '01'
//赋值
em_start.text = ls_curmonth
em_end.text = ls_today
//--------------------------------------------------------------------.
```

(4) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户输入的开始和结束时间检索各科室的挂号量。

```
//--------------------------------------------------------------------
string ls_start,ls_end ,ls_select
string ls_left ,ls_right
long ll_group

//得到开始和结束时间
ls_start = em_start.text
ls_end  = em_end.text
ls_select = ''
// 开始时间
if not isnull(ls_start) and ls_start <> "" and lower(ls_start) <> 'none' then
    ls_select = ls_select + " convert(char(10)  ,门诊挂号.时间,120) >= '"
+ ls_start + "'"
end if
// 结束时间
if not isnull(ls_end) and ls_end <> "" and lower(ls_end) <> 'none' then
   if ls_select = "" then
     ls_select = " convert(char(10)  ,门诊挂号.时间,120) <= '" + ls_end + "'"
   else
     ls_select = ls_select + " AND  convert(char(10)  ,门诊挂号.时间,120)
<= '" + ls_end + "'  "
   end if
end if
//因为原语法中含有 GROUP，分解语法
ll_group = pos(is_sql,"GROUP")
ls_left = left(is_sql,ll_group - 1)
ls_right = mid(is_sql,ll_group)
//拼 SQL 语句
ls_select = ls_left + ' where ' + ls_select + ls_right
if ls_select <> '' then
   //赋值 SQL 语句
   dw_1.setsqlselect(ls_select)
end if
dw_1.retrieve() //检索数据
//--------------------------------------------------------------------
```

(5) 在数据窗体 dw_1 的 CLICKED 事件中加入以下代码，选中在当前数据窗体中单击的数据行。

```
//--------------------------------------------------------------------
if row < 1 then return
//选择单击的行
this.selectrow(0,false)
this.selectrow(row,true)
//--------------------------------------------------------------------
```

这样，完成了科室挂号量查询功能，下面将介绍药品库存查询的制作方法。

6.6.15 完成药品库存查询功能

本小节将以药品库存查询为例来介绍院长查询的报表，实际中的报表会多很多。

(1) 选择 File｜Inherit 命令，弹出如图 6.36 所示的 Inherit from Object 对话框，在 Objects of Type 下拉列表框中选择 Windows 选项，然后选中 w_sheet 对象。

(2) 单击 OK 按钮创建一个新窗体，将其 Name 保存为“w_ypkc”，设置窗体的属性为 main!，并为其添加如图 6.45 所示的控件。

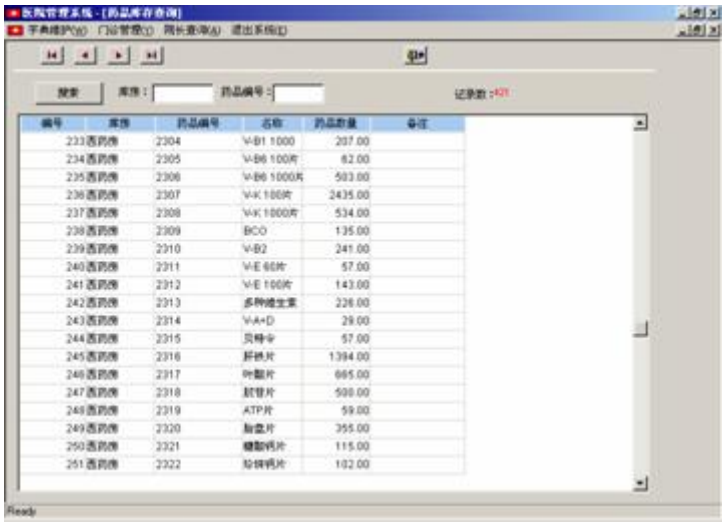


图 6.45 【药品库存查询】窗体

(3) 各控件的属性设置如表 6.15 所示。

表 6.15 【药品库存查询】窗体控件属性设置

控件类型	对 象 名	属 性	取值(说明)
window	w_ypkc	Title	药品库存查询
window	w_ypkc	WindowType	main!
window	w_ypkc	ICON	\ICON\医院.ico
statictext	st_1	text	开始时间：
statictext	st_2	text	结束时间：
Editmask	em_start	maskdatatype	Datemask!
Editmask	em_start	mask	Yyyy-mm-dd
Editmask	em_end	maskdatatype	Datemask!
Editmask	em_end	mask	Yyyy-mm-dd
CommandButton	cb_query	text	搜索
Datawindow	dw_1	dataobject	d_ksghl

在窗体 w_ksghl 中定义实例变量，保存数据窗体 dw_1 中的原始语法，为以后数据检索做准备。

```
string is_sql
```

(4) 在窗体 w_ypkc 的 OPEN 事件中加入以下代码，将数据窗体连接到数据库并得到数据窗体的原始语法。

```
//--------------------------------------------------------------------
string ls_today,ls_curmonth

//将数据窗体连接到数据库
dw_1.settransobject(sqlca)
//得到原始语法，为检索做准备
is_sql = dw_1.getsqlselect()
//得到当天和当月 1 号
ls_today = string(today(),"yyyy-mm-dd")
ls_curmonth = left(ls_today,8)  + '01'
//赋值
em_start.text = ls_curmonth
em_end.text = ls_today
//--------------------------------------------------------------------.
```

(5) 在按钮 cb_query 的 CLICKED 事件中加入以下代码，根据用户输入的开始和结束时间检索药品库存量。

```
//--------------------------------------------------------------------
string ls_start,ls_end ,ls_select
string ls_left ,ls_right
long ll_group

//得到开始和结束时间
ls_start = em_start.text
ls_end  = em_end.text
ls_select = ''
// 开始时间
if not isnull(ls_start) and ls_start <> "" and lower(ls_start) <> 'none' then
     ls_select = ls_select + " convert(char(10)  ,门诊挂号.时间,120) >= '" + ls_start
+ "'"
end if
// 结束时间
if not isnull(ls_end) and ls_end <> "" and lower(ls_end) <> 'none' then
   if ls_select = "" then
     ls_select = " convert(char(10)  ,门诊挂号.时间,120) <= '" + ls_end + "'"
   else
     ls_select = ls_select + " AND  convert(char(10)  ,门诊挂号.时间,120)
```

```
    <= '" + ls_end + "' "
       end if
    end if
    //因为原语法中含有 GROUP，分解语法
    ll_group = pos(is_sql,"GROUP")
    ls_left = left(is_sql,ll_group - 1)
    ls_right = mid(is_sql,ll_group)
    //拼 SQL 语句
    ls_select = ls_left + ' where ' + ls_select + ls_right
    if ls_select <> '' then
       //赋值 SQL 语句
       dw_1.setsqlselect(ls_select)
    end if
    dw_1.retrieve() //检索数据
    //--------------------------------------------------------------------
```

(6) 在数据窗体 dw_1 的 CLICKED 事件中加入以下代码，选中当前在数据窗体中单击的数据行。

```
    //--------------------------------------------------------------------
    if row < 1 then return
    //选择单击的行
    this.selectrow(0,false)
    this.selectrow(row,true)
//--------------------------------------------------------------------
```

6.6.16 编译并运行系统

至此完成了整个系统的制作，要将系统编译成为 exe 可执行文件，需要新建一个 Project，步骤如下。

(1) 选择 File | New 命令，在弹出的 New 对话框中打开 Project 选项卡，如图 6.46 所示。

这里，创建应用程序一般使用前面两个选项 Application Wizard 和 Application，由于 Application Wizard 是创建应用程序的向导，比较简单，这里主要以 Application 为例来介绍应用程序的建立。

图 6.46 选择应用程序类型

(2) 选中 Application，单击 OK 按钮，出现如图 6.47 所示的对话框。

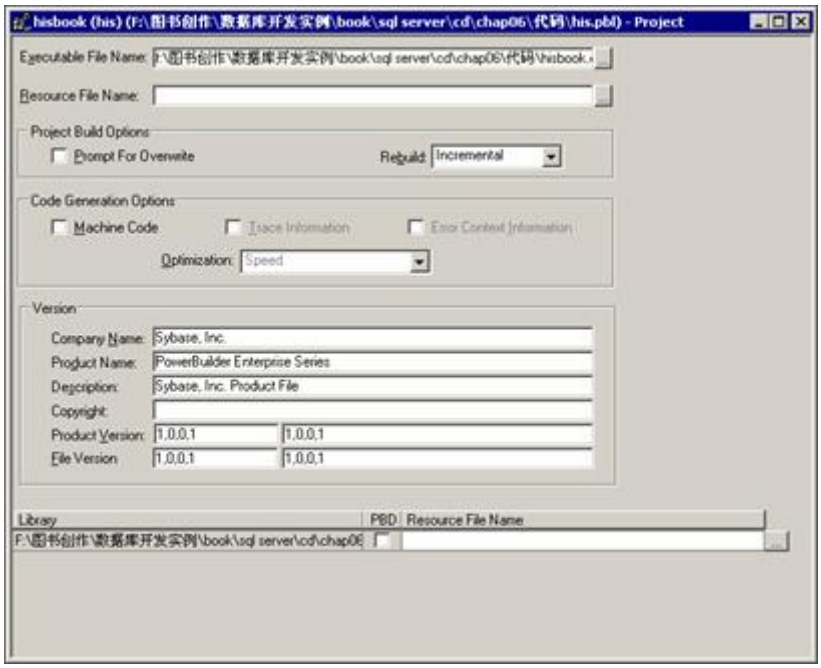


图 6.47 创建应用程序类型

- Executable File Name 为应用程序的路径及名称。
- Resource File Name 为源文件的路径及名称。这里一般不要填写，系统默认当前工程文件。
- Project Build Options 为程序编译选项。Rebuild 方式有两种，full 为完全编译，Incremental 为增量编译，即只编译已经修改的 pbl。
- Code Generation Options 为编译的方式，如果选中 Machine Code 则编译成 dll 方式，否则编译成 PowerBuilder 自己的 PBD 方式。
- Version 为编译应用程序的版本，这个对应用程序影响不大，这里就不具体介绍。
- library 为当前项目包含的库文件 pbl，一般是全选。

(3) 按图 6.47 所示输入信息。

(4) 保存后单击工具栏上的 Deploy 图标或者选择 Design | Deploy Project 命令，编译程序。

6.7 系统发布

系统设计完成后需要打包发布，将应用系统制作成安装程序。制作安装程序的工具有很多，一般 PowerBuilder 开发的应用系统可以使用 Install Shield 来完成系统的安装包。

6.8 小结

通过该实例可以掌握以下知识和技巧：

- 医院管理系统的需求。
- 利用 PowerBuilder 进行数据库编程的多种方法。
- PowerBuilder 中继承的用法。
- PowerBuilder 多窗体的制作方法。
- 利用 PowerBuilder 编写医院管理系统。

用户可以根据这些方法自行完成医院管理系统的其他功能。