高等教育系列教材（计算机与信息管理类）

Visual FoxPro 应用及程序设计教程

熊李艳
吴　昊　编著

电子科技大学出版社

图书在版编目 (CIP) 数据

Visual FoxPro **应用及程序设计教程/熊李艳,吴昊编著.**
- **成都 :电子科技大学出版社,**2003.8
高等教育系列教材 (计算机与信息管理类)
ISBN 7-81094-032-5

Ⅰ.V… Ⅱ.①**熊…** ②**吴…** Ⅲ.**关系数据库-数据库管理系统,**Visual FoxPro-**高等学校-教材** Ⅳ.TP311.138

中国版本图书馆 CIP **数据核字 (**2003**) 第** 057037 **号**

高等教育系列教材 (计算机与信息管理类)

Visual FoxPro **应用及程序设计教程**

熊李艳
吴 昊 **编著**

出 版 :电子科技大学出版社 (成都市建设北路 2 **段** 4 **号 邮编** 610054**)**
责任编辑 :张致强
发 行 :新华书店
印 刷 :安徽省蚌埠市方达印刷厂
开 本 :787×960 1/16 **印张 :**24.5 **字数 :**438 **千**
版 次 :2003 **年** 8 **月第一版**
印 次 :2003 **年** 8 **月第一次印刷**
书 号 :ISBN 7-81094-032-5/TP·19
印 数 :1 - 5000 **册**
定 价 :35.00 **元**

编审说明

随着我国高校教育体制改革的不断深入，高校非计算机专业计算机系列课程三个层次的教学也在不断推陈出新。在各校转变教育观念，深化教育体制改革，全面推行素质教育，突出创新精神和创新能力培养之时，作为全国高校非计算机专业必修的计算机文化基础系列课程，对提高同学综合素质以及计算机在各个专业中的应用都将起到十分重要的作用。

本书理论联系实际，主要讲述 Visual FoxPro6.0 概述、项目管理器、数据库和表的基本操作、表的基本应用、VFP 程序设计基础、VFP6 表单设计、数据库和表的高级应用、视图和查询、设计报表和标签、设计菜单、多用户与共享技术。

本书和《Visual FoxPro 程序设计实践教程》配套，适于作为各高等学校非计算机专业基础教学用书，也可作为各类培训班的培训教材和一般计算机应用人员自学用书。

本书共分十二章，其中第一、第二章由周美玲老师负责编写，第三、第四、第五、第八、第九、第十、第十一、第十二章由熊李艳编写，第六、第七章由吴昊编写。张邦明、谢昕、雷莉霞、杜玲玲、吴慧蕴、范萍、李秋珍等同仁做了大量的工作，参与了该书的大纲制订和程序调试工作。为保证教材质量，刘觉夫、汤彬同志负责整套书的审阅。全书由熊李艳负责统稿。

本书在编写过程中得到了华东交通大学计算机基础教研室全体同仁的大力支持和帮助，在此表示由衷的感谢！

由于编写时间仓促和编者水平有限，书中难免存在不妥之处，敬请广大读者提出宝贵意见！

高等教育系列教材编审指导委员会

2003 年 8 月

目　录

§1.1 数据库系统简介

1.1.1 从文件管理到数据库管理

随着计算机技术的发展,计算机的主要应用已从科学计算逐渐转变为事务处理。在进行事务处理时,并不需要进行复杂的科学计算,而主要是对大量数据的存储、查找、统计等工作。为了有效地保存和管理计算机应用中出现的大量数据,必须采用一整套合理的数据处理方法,即数据管理。

在数据库出现以前,计算机用户使用数据文件来存放数据。常用高级语言从早期的 FORTRAN 到今天的 C 语言,都支持数据文件。这种以数据文件的形式对数据进行管理的方法称为文件管理。文件管理是 20 世纪 50 年代后期到 60 年代中期普遍采用的管理数据的有效手段。数据的存取基本上以记录为单位,一个记录又包含若干数据项。用户通过对文件的访问实现对记录的存取。数据文件基本上对应于某个应用程序,当不同的应用程序所需要的数据有部分相同时,必须建立各自的文件,而不能共享相同的数据,因此数据冗余度大,浪费存储空间,并且由于相同数据重复存储,各自管理,给数据的修改和维护带来了困难 数据和程序缺乏独立性。数据文件是为某一特定的应用程序服务的,要想对现有的数据再增加一些新的应用是很困难的,系统不容易扩充,一旦数据的逻辑结构改变,必须修改应用程序,修改数据文件的定义。而应用程序的改变,也将影响文件的数据结构的改变。

随着计算机的应用越来越广泛,用于管理的规模越来越庞大,数据量急剧增长,而且数据的共享要求越来越强,文件管理系统采用的一次最多存取一个记录的访问方式,以及在不同数据文件之间缺少相互联系的结构,越来越不能适应管理大量数据的需要。于是,数据库管理系统应运而生。数据库技术的萌芽从 60 年代中期开始,到 60 年代末 70 年代初,数据库技术日益成熟,并有了坚实的理论基础。1969 年 IBM 公司研制开发了第一个数据库管理系统的商品化软件 IMS (Information Management System)。

1.1.2 数据库系统的特点

使用数据库来管理,具有下列特点:

1. 面向全组织的复杂的数据结构

数据库系统中的数据是相互关联的,这种关联不仅表现在记录内部,更重要的是记录类型之间的相互关联。整个数据库是以一定的形式结构而成的。这种结构不仅能将数据组织起来,还能充分反映现实世界中数据间的联系,满足用户的不同需要。数据的结构化是数据库主要特征之一,是数据库系统与文件管理系统的根本区别。

例如在教学管理系统中,将学生表、课程表两者通过成绩表联系在一起。在数据库中可将其组织成如图 1-1 所示的结构。使用这种结构,对诸如查询"李明各科学习成绩"、"选修 C 语言的学生名单"等,就可通过不同的路径来完成。

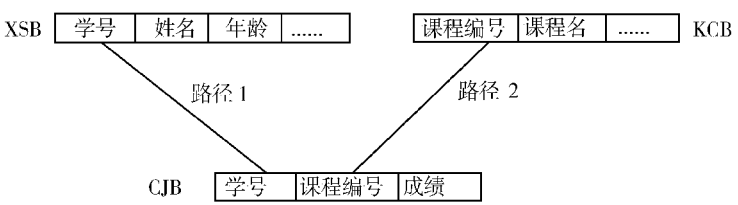


图 1-1 数据库中的数据结构

2. 可减少冗余

在非数据库系统中,每个应用程序使用自己的数据来处理,经常会造成数据的重复建立,而且彼此之间的数据格式也不相同,无法交互应用。而在数据库系统中,仅建立共同的数据库,其他的应用程序都使用这个数据库,因此冗余性可大大减少。

例如,公司员工的数据可用于工资计算系统和人事管理系统,若分开建立两个数据库,不仅数据可能不一致,也会浪费存储空间。若能利用数据库来管理,则只需储存一份数据即可,冗余性可减少。

3. 可避免不一致

在上述情形中,当相同的数据存于不同的系统中时,若数据需要变更,两者的更改时间可能不同步,造成两者的不一致,若用数据库系统来管理,则仅需更改同一份数据,不一致性可以消除。

4. 数据可以共享

所行程序都存取同一份数据库,数据完全共享。

5. 数据独立性

应用程序不需要了解数据实际的存取方式,通过数据库系统的存取指令,就可得到需要的数据,当数据的存储结构变更时,仅需更改数据库系统的内部程序,外部应用程序不需更改。表 1-1 列出了数据库管理系统和文件管理系统的主要性能差别。

表 1-1　数据库管理系统与文件管理系统性能对照表

文件管理系统	数据库管理系统
文件中的数据由特定的用户专用	库内数据由多个用户共享
每个用户拥有自己的数据,导致数据重复存储	原则上可以消除重复。为方便查询允许少量数据重复存储,但冗余度可以控制
数据从属于应用程序,二者相互依赖	数据独立于应用程序,强调数据的独立性
各数据文件彼此独立,从整体看为“无结构”的	各文件的数据相互联系,从整体上看是“有结构”的

1.1.3　数据模型

从理论上讲,数据模型是指反映客观事物及客观事物间联系的数据组织的结构和形式。客观事物是千变万化的,各种客观事物的数据模型也是千差万别的,但也有其共同性。常用的数据模型有如下三种:

1. 层次模型

层次模型 (Hierarchical Model) 表示数据间的从属关系结构,是一种以记录某一事物类型为根结点的有向树结构。层次模型示例如图 1-2 所示,各个结点间具有父子关系,每个父结点可以有多个子结点,但每个子结点仅能有一

个父结点。层次模型结构示意如图 1-3 所示。

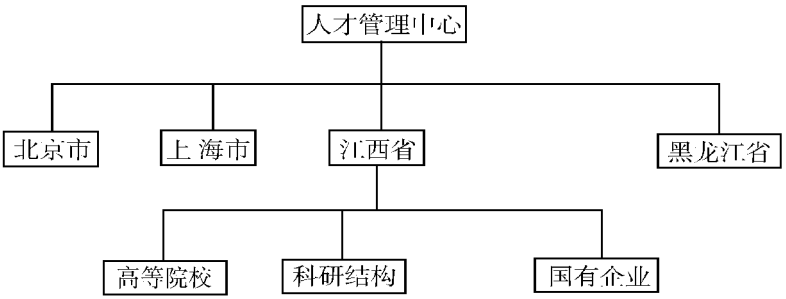


图 1-2　层次模型示例

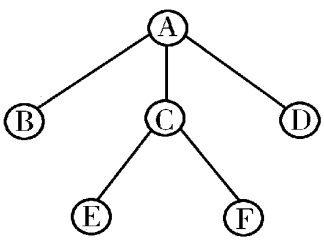


图 1-3　层次模型结构示意

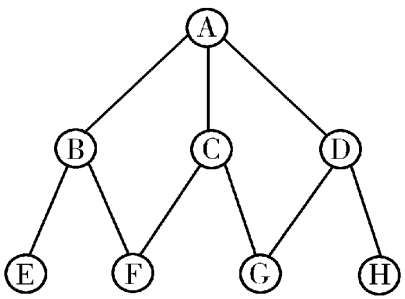


图 1-4　网状模型结构示意

2. 网状模型

网状模型 (Network Model) 是层次模型的扩充,它表示多个从属关系的层次结构,呈现一种交叉的网络结构。网状模型是以记录为结点的网络结构。其主要特征如下：

① 有一个以上的结点无双亲。

② 至少有一个结点有多个双亲。

网状模型可以表示较复杂的数据结构,但这种模型在概念上和结构上都比较复杂,操作上也有很多不便。网状模型结构示意如图 1-4 所示。

3. 关系模型

关系模型 (Relational Model) 的所谓“关系”,是指那种虽具有相关性但非从属性的平行的数据之间按照某种序列排列的集合关系。

例如,有数据记录如下:甲,女,20 岁;乙,女,45 岁;丙,男,35 岁;丁,男,55 岁。这四组数据之间是平行的,不存在从属关系,但若他们是同一部门的员工,我们可以建立一张二维表(一个关系)(见表 1-2)。

表 1-2　某部门专门人才基本情况表

姓名	性别	年龄
甲	女	20
乙	女	45
丙	男	35
丁	男	55

表中的这些数据是平行的,不构成从属关系,但它们构成了某部门工作人员的属性关系结构,故可用一个关系来表示。同样,假设上述四人不是来自同一部门,且从事的专业也不相同,便构成了某地区专门人才基本情况表,又可以建立一个关系(一张二维表)(见表 1-3)。

表 1-3　某地区专门人才基本情况表

部门	专业	姓名	性别	年龄
科研机构	计算机	甲	女	20
高等院校	国际会计	乙	女	45
国有企业	国际贸易	丙	男	35
高等院校	信息管理	丁	男	55

以上两表就是关系模型的两个简单例子。表格中的每一数据可看成独立的数据项,它们共同构成了该关系的全部内容。表格中的每一行称为一个记录。每一条记录由若干数据项组成。表格中的每一列称为一个字段。字段表示关系模型中各种数据项的类型,每一字段由若干相同类型的数据项组成。

从总体上讲,由多个字段组成的若干条记录的集合构成了一个关系模型或称为一个关系。从某种意义上可以说,关系模型就是一张二维表。

1.1.4　数据库系统的分代

经过 30 多年的发展,数据库系统已走过了第一、第二两代——非关系型数据库系统和关系型数据库系统,正在向第三代——对象——关系型数据库前进。

1. 非关系型数据库系统

非关系型数据库系统是对第一代数据库系统的总称,其中包括层次数据库系统和网状数据库系统两种类型。数据库的性质是由其中的数据模型决定的。数据库中的数据如果依照层次模型进行存储,则为层次数据库;若依照网状模型进行数据存储,则为网状数据库。这一代数据库系统具有以下共同特征:

⑴ 采用"记录"为基本的数据结构,即记录为其数据模型中的结点。结点之间的联系由数据模型的类型所决定。

⑵ 无论层次模型或网状模型,一次查询只能访问数据库中的一个记录,存取效率不高。对于具有复杂联系的系统,还需用户详细描述对数据的访问路径,增加使用的麻烦。所以,自关系数据库兴起后,非关系数据库系统已逐渐被关系数据库系统所取代。

2. 关系型数据库系统(Relational DataBase System,简称 RDBS)

1970 年,科德在一篇论文中提出"关系模型"的概念。70 年代中期,商品化的 RDBS 问世,数据库系统进入了第二代。80 年代后,RDBS 在各种类型计算机上分别实现,目前 PC 机上使用的数据库系统主要是第二代关系型数据库系统。

与第一代数据库系统相比,关系型数据库系统具有以下优点:

⑴ 采用人们经常使用的表格作为基本的数据结构,通过公共的关键字段来实现不同二维表之间的联系,简单明了,使用方便。

⑵ 一次查询仅用一条命令就可访问整个关系(二维表),查询效率较高。通过多表联合操作,还可实现对有联系的若干二维表的关联查询。而第一代数据库每次仅能访问一个记录。

⑶ 层次数据库和网状数据库必须依照数据间的关系来建立,建立数据库不容易。而关系型数据库的数据之间的关系是平行的,修改起来也比较容易。

⑷ 搜寻速度较快,表格式的搜寻比线形结构更容易。

⑸ 层次和网状结构很容易转换成表格结构,因此关系型可视为万用形式的结构。

3. 对象——关系数据库系统(Object - Relational DataBase System,简称 ORDBS)

随着多媒体应用的扩大,对数据库提出了新的要求,要求数据库系统能够存储图形、图像、声音等多媒体信息,并能实现对这些复杂对象的复杂行为。

将数据库技术与面向对象技术相结合,便成了数据库技术的新的研究方向,也构成了第三代数据库系统的基础。

20 世纪 80 年代中期以来,对“面向对象的数据库系统”(OODBS)和“对象——关系数据库系统”的研究都十分活跃。1989 年和 1990 年先后发表了《面向对象数据库系统宣言》和《第三代数据库系统宣言》,后者主要介绍 ORDBS。由于 ORDBS 是建立在 RDBS 技术之上的,可以直接利用 RDBS 的原有技术和用户基础,所以发展比 OODBS 更顺利,正在成为第三代数据库系统的主流。

根据《第三代数据库系统宣言》提出的原则,第三代数据库系统除应包含第二代数据库系统的功能外,还应支持正文、图形、图像、声音等新的数据类型,支持类、继承、函数/方法等丰富的对象机制,并能提供高度继承的、可支持客户机/服务器应用的用户接口。虽然 ORDBS 目前还处于发展阶段,在技术上和应用上还有许多工作要做,但是已经显示出光明的发展前景,一些数据库厂商已经推出了可供实用的 ORDBS 产品。

1.1.5 数据库管理系统和数据库应用系统

1. 数据库管理系统

数据库的建立和查询,都是通过特定的数据库语言进行的。由于查询是数据库语言的最主要功能,所以数据库语言有时也称为查询语言。被国际标准化组织(ISO)确定为关系数据库语言标准的 SQL 语言,就是“结构化查询语言”(Structured Query Language)的缩写。

正如使用高级语言需要解释/编译程序的支持一样,使用数据库语言也需要特定软件的支持,这个软件就是“数据库管理系统”DBMS(DataBase Management System)。一般地,DBMS 应包含数据定义、数据操作以及控制和管理功能。DBMS 能向用户提供“数据定义语言”(DDL)用于描述数据库的结构,“数据操作语言”(DML)用于支持用户对数据库中的数据进行查询、增加、删除、修改等操作。DBMS 还提供必要的控制和管理功能,其中包括:并发控制 安全性检查 数据的备份、恢复和转储功能 对数据库运行情况的监控和报告等。

需要指出的是,数据库系统有层次、网状、关系和对象——关系等多种数据模型,一种 DBMS 只能支持一种模型的数据库系统。目前在 PC 机上使用的 DBMS 都是关系数据库管理系统,简称 RDBMS。从较早的 dBASE 和 FoxBASE,到今天广泛流行的 FoxPro 和 Oracle,都是 RDBMS。

2. 数据库应用系统

数据库应用系统 (DataBase Application System,简称 DBAS) 专指基于数据库的应用系统。实际上数据库应用系统是一个比较复杂的系统,它是由硬件、操作系统、数据库管理系统、编译系统、用户应用程序和数据库等组成的。

⑴ 计算机硬件

硬件资源需要有足够大的内、外存空间,用来运行操作系统、DBMS 核心模块、数据缓冲区和应用程序,以及存储数据库数据。此外,还要求系统具有较高的通道能力,以提高数据传送率。按照目前的一些应用,还应包含与计算机网络相关的硬件。

⑵ 数据库

所谓数据库,就是以一定的组织方式将相关的数据组织在一起存放在计算机存储器上的、能为多个用户共享的、与应用程序相互独立的一组相关数据的集合。数据库的性质是由其中的数据模型决定的。在数据库中的数据如果依照层次模型来存储,则为层次数据库;若依照网状模型来存储,则为网状数据库;若依照关系模型来存储数据,则该数据库为关系数据库。Visual FoxPro 数据库管理系统所管理的数据,都是依照关系模型来建立和存储的,因此其数据库是关系数据库。

⑶ 数据库管理系统

数据库应用系统的核心软件,起着管理、操作控制等作用。

⑷ 软件部分

包括操作系统、编译系统、用户应用程序及计算机网络软件等。

1.1.6 数据库系统的分类

1987 年,著名的美国数据库专家厄尔曼 (J. D. Ullman) 教授在一篇题为《数据库理论的过去和未来》的论文中,把数据库理论概括为 4 个分支:关系数据库理论、分布式数据库理论、演绎数据库和面向对象数据库。今天,关系数据库理论已愈趋成熟,在 PC 机上获得普遍的应用;ORDBS 已成为第三代数据库系统的主流。其余两个分支——分布式数据库和智能数据库也在过去 10 多年间取得了不小的进展,扩大了应用范围,现将部分变化简述如下:

1. 单用户数据库和多用户数据库

早期的数据库都是单用户系统,只能供一个人使用。随着局域网应用的扩大,供网络用户共享的多用户数据库开始流行。Visual FoxPro 就是一种多

用户数据库系统。在它以前,已有 dBASE Ⅲ +,FoxBASE +,FoxPro 等多用户数据库供用户选用。

多用户数据库的关键是保证“并发存取”(Concurrent Access)的正确执行。例如火车订票系统允许乘客在多个售票点订票。当两位乘客在不同的售票点同时向某一车次预订相同时间的车票时,若缺乏相应的并发存取机制,在数据库中可能仅仅反映一个乘客的订票,从而发生两人同订一票的错误。

2. 集中式数据库和分布式数据库

集中和分布,是指数据存放地点而言的。集中式数据库是指把数据集中存放在一台计算机上或一个地方。分布式数据库则是把数据分散存储在网络的多个节点上,彼此用通信线路连接。

20 世纪 70 年代中期以来,由于计算机网络和通信的迅速发展,以及地理上分散的公司、团体和组织对于数据库更为广泛应用的需求,在集中式数据库系统成熟技术的基础上产生和发展了分布式数据库系统。分布式数据库是数据库技术和网络技术两者相互渗透和有机结合的结果。

例如,一个银行有众多储户,如果他们的数据存放在一个集中式数据库中,所有的储户在存、取款时都要访问这个数据库,访问量必然很大。若改用分布式数据库,将众储户的数据分散存储在离各自住所最近的储蓄所,则大多数时候数据就可就近存取,仅有少量数据需远程调用,从而大大减少网上的数据传输量。对一个设计良好的数据库,用户在存取数据时不需指明数据的存放地点。换句话说,它能使用户像对集中式数据库访问时一样方便。

分布式数据库和多用户数据库都是在网络上使用的,但多用户数据库并非都是分布存储的。

3. 传统数据库和智能数据库

传统数据库存储的数据都代表已知的事实,智能数据库则除了存储事实外还能存储用于逻辑推理和判断的原则。所以后者也称为“基于规则的数据库”。

例如,某智能数据库存储有“科长领导科员”的规则。如果它同时存有“甲是科长”、“乙是科员”等数据,它就能推理得出“甲领导乙”的新事实。随着人工智能不断走向实用化,对智能数据库的研究将日趋活跃,演绎数据库、专家数据库和知识库系统都属于智能数据库的范畴。将它们的共同关键是逻辑推理,如果逻辑模式出了问题,就可能导出荒谬的结果。

§1.2 Visual FoxPro 6.0 发展概述

1.2.1 Xbase 数据库的发展

在 PC 机关系数据库中，Xbase 家族占有重要的地位。从 dBASE 到 FoxBASE，FoxPro，Visual FoxPro，这一系列数据库始终独占鳌头，拥有最广大的用户群。

计算机技术的发展带动着数据库系统的发展。20 世纪 70 年代后期，数据库理论的研究已较为成熟。当 IBM - PC 及其兼容机于 80 年代初逐步普及时，美国 Ashton - Tate 公司于 1982 年推出了适合 8 位微机的 dBASE Ⅱ 关系数据库管理系统。由于它简单、易学、易用，数据库处理能力大大优于其他语言，因此很快随微机的普及而风靡全球。但随着 16 位微机的出现，dBASE Ⅱ 很快显示出它的不足，如数据类型不够丰富、精度不高、能同时打开的数据表文件数过少等。因此，Ashton——Tate 公司于 1984 年 6 月推出了更新版本 dBASE Ⅲ。dBASE Ⅲ 增加了日期型、备注型两种数据类型，将数据精度提高到 16 位，增加了 20 多条命令和 10 多个函数，改善了报表功能和屏幕输出格式。

dBASE Ⅲ 虽然有许多优点，但其最突出的缺点是它只能解释执行，从而导致应用程序无法加密，运行速度太慢。1986 年 Ashton - Tate 公司推出了改进型 dBASE Ⅲ Plus，它增加了 30 多条命令和 30 多个函数，提供了更友好的用户界面和新的数据目录处理方法。dBASE Ⅲ Plus 网络版具有在局域网上运行所需的管理工具，如文件和记录加锁、提供安全保密等。

1.2.2 从 FoxBASE 到 FoxPro

dBASE 数据库虽然优点很多，但仍然存在不少缺点，如速度慢、不带编译器、人机界面差、命令和函数有限等。从事数据库工作之一的美国 Fox Software 公司预见到微型计算机数据库系统应用的巨大潜力，于 1984 年推出了与 dBASE 完全兼容的 FoxBASE，其速度大大快于 dBASE，且第一次引入了编译器。1986 年，推出了与 dBASE Ⅲ Plus 兼容的 FoxBASE +，它提供了编译功能、支持数组、命令和函数更加丰富。1987 年推出了 FoxBASE + 2.0。1988 年，Ashton - Tate 公司推出了 dBASE Ⅳ，它除了可以对程序进行编译外，还支持 SQL 语句，具有网络功能，并提供了安全保密功能。

随着软件技术的发展，要求数据库管理系统提供更友好的界面、更丰富的工具。为了顺应这一发展趋势，FoxPro 诞生了。1989 年下半年，FoxPro 1.0 正

式推出。它采用了用户接口非常友好的图形用户界面 GUI,支持鼠标,操作方便,是一个与 dBASE、FoxBASE 全兼容的集成环境式的数据库开发环境。FoxPro 1.0 功能强大,运行速度快,要比 dBASEⅣ快 8 倍,比 dBASEⅢ快 16 倍,比 FoxBASE +2.10 快两倍。FoxPro 1.0 还增加了大量的函数和命令。1991 年 7 月,FoxPro 2.0 推出。由于使用了 Rushmore 查询优化技术、先进的关系查询与报表技术,以及第 4 代语言 4GL (Fourth Generation Language) 工具,FoxPro 2.0 第一次引入了 SQL 以及直观的查询设计器,增加了屏幕生成器、菜单生成器、报表生成器和项目管理器等强大的工具,在性能上大幅度提高了。它面向对象与事件,其扩展版更是一个真正的 32 位产品。它增加了 100 多条全新的命令和函数,从而使得 FoxPro 逐步成为 Xbase 语言的标准。

1992 年,Fox 软件公司被 Microsoft 公司收购。当年推出了 FoxPro 2.5。它可运行在 MS - DOS、Windows、Machitosh、Unix 操作系统环境下,并保持了对每一级用户拥有相同的图形用户界面、工具和语言。它以其优越的性能、最快的速度而领先于任何其他数据库管理软件,被用户公认为是首选的数据库产品。FoxPro 2.6 是对 FoxPro 2.5 的补充,它使用户可以很容易地管理目录文件,提供了向导工具 Wizards,增强了与 dBASEⅣ的兼容性,扩展了 FoxPro 2.5 的性能。

1.2.3　Visual FoxPro 的推出

1995 年 9 月,Microsoft 公司推出了最新的 FoxPro 版本 Visual FoxPro 3.0。它是多年来出现的关系数据库方面最重要的产品,是继 Visual C + +、Visual Basic 后又一可视化产品。Visual FoxPro 是一个 32 位的数据库开发系统,可运行于 Windows 95 和 Windows NT 操作系统,它既具有 Visual 系列的功能强大、直观易用、面向对象等优点,又兼有 Windows 和 FoxPro 的长处,提供了向导、设计器和生成器等工具,使数据库的管理工作变得更加容易。

与 FoxPro 2.x 比较,Visual FoxPro 3.0 在以下几个方面的特性有所增强:

- 快速创建应用程序的能力。
- 更强大的开发能力。
- 开发客户机/服务器解决方案的能力。
- 与其他 Microsoft 应用程序互相作用的能力。

后来,Microsoft 公司又推出了 Visual FoxPro 5.0,它增加或加强了以下功能:

- 增强了项目管理器和数据库管理。
- 改善了调试工具。
- 更加简单的表设计方法和扩展的数据字典。

- 增强的查询和视图设计。
- 增加了表单功能。
- 有更多和更好的向导。
- 更紧密的 OLE 和 ActiveX 集成。

1998 年,Microsoft 公司发布了 Visual FoxPro 6.0。Visual FoxPro 6.0 增加或增强了以下功能:

1. Access 和 Assign 方法程序。利用这两种用户自定义的方法程序,可在查询或想要更改属性值时执行所需的代码。
2. 组件管理库。组件管理库能帮助用户将类库、表单、按钮等对象进行分组,并组成对象、项目、应用程序或其他分组。
3. 代码范围分析器应用程序。
4. 对 GIF 和 JPEG 图形的支持。
5. HTML 帮助。
6. OLE 拖放。
7. 新增或改进了向导和生成器。
8. 对 2000 年日期的支持。

§1.3 Visual FoxPro 6.0 系统概述

1.3.1 Visual FoxPro 6.0 的特点

1. 简单、易学、易用

- Visual FoxPro 6.0 提供了“向导”、“生成器”和“设计器”这 3 种工具,使用户能最简单而又最快速地完成数据操作任务。
- 改进了用户界面,其主窗体与其他 Microsoft 产品 (如 Word、Excel 等) 更趋于一致。另外,Visual FoxPro 6.0 还提供了工具栏,使得用户操作更加容易。
- 提供的“表单设计器”是一种功能强大的工具,用户可以不编程或使用少量的代码来实现友好的交互式应用程序界面。
- 可以通过“项目管理器”集中地管理数据、文档、类库、源代码等各种资源。

2. 功能更强大

- 真正的数据库概念。XBASE 软件中称. DBF 文件为数据库,容易使人产生一个数据库就是一个二维表的错误认识。在 Visual FoxPro 6.0 中,原来

的.DBF 文件变成了数据库中的一个表,不属于任何数据库的表称为自由表。数据库是由若干表、表之间的关系和触发程序的集合,合理地体现了关系数据库的思想,与关系数据库理论统一了起来。

- 可视化编程技术。可视化编程技术给人一种所见即所得的感受,在您编辑表单、报表、菜单时,可以直接运行,不必来回调试,极为方便。
- 具有面向对象编程的能力。Visual FoxPro 6.0 支持 XBASE 传统的结构化的编程方式,也提供了面向对象编程能力。Visual FoxPro 6.0 提供了近 30 个基类,包括表单、工具栏、页格式等,还允许用户自定义类。
- 新增很多命令和函数,SQL 语句更加丰富。Visual FoxPro 6.0 新增了 7 种数据类型:整型、货币型、日期时间型、双精度型、通用型、二进制字符型和二进制备注型。
- 使用快速查询技术。快速查询 (Rushmore) 技术能够迅速地从数据库中选择出用户需要的记录,它将数据查询所需的时间从几小时缩短到几分钟。
- 使用 32 位方式,运算速度、存储能力大大提高。

3. 支持客户机/服务器结构

Visual FoxPro 6.0 提供了支持客户机/服务器结构所需的各种特性:多功能的数据词典、本地和远程视图、空值 NULL 的支持、事务处理、对各种 ODBC 数据资源的访问。

Visual FoxPro 6.0 数据库提供了一个数据词典,通过这个数据词典,用户可以对数据库中的每一个数据表添加规则、视窗、触发器、永久关系和连接。

4. 其他软件的高度兼容性

- 同其他软件共享数据。Excel 和 Word 都可以通过相应的方法共享 Visual FoxPro 6.0 的数据。
- 导入和导出数据。用户能够在 Visual FoxPro 6.0 和其他软件之间输入和输出数据,即导入和导出。这种导入和导出是通过不同的文件格式的转换来实现的,不同的文件格式包括文本、电子表格和表。
- 使用自动 OLE 控制其他软件。Visual FoxPro 6.0 可以通过编程来运行其他的软件。

1.3.2　Visual FoxPro 6.0 系统的安装

我们以 Visual FoxPro 6.0 中文版为例。在安装之前,要充分做好安装前的准备工作。

1. 安装环境

(1) 硬件环境

- PC 兼容机,具有 80486 50MHz 以上处理器;
- 鼠标
- 内存在 16MB 以上;
- 硬盘的最小空间为 15MB;用户自定义安装需要 100MB 空间;完全安装所有联机文档需要 240MB 硬盘空间;
- 推荐使用 VGA 或更高分辨率的显示器。

(2) 软件环境

可以在 Windows 95/98 或 Windows NT 以上操作系统中安装。

2. Visual FoxPro 6.0 的安装

中文 Visual FoxPro 6.0 可以从 CD - ROM 或网络上安装。若从网络上安装,则安装准备中还应有一个网络环境。从 CD - ROM 安装的步骤如下:

① 将 Visual FoxPro 6.0 系统安装光盘放入 CD - ROM 驱动器中,找到光盘中的 setup.exe 文件,双击该文件运行,将出现"Visual FoxPro 6.0 安装向导"对话框,如图 1-5 所示。

图 1-5

② 单击"下一步"按钮,出现"最终用户许可协议"对话框,如图 1-6 所示,阅读协议之后,选择"接受协议",单击"下一步"按钮。

③ 在下一个界面中输入正确的产品 ID 号和用户信息,单击"下一步"按钮,如图 1-7 所示。

图 1-6

图 1-7

④ 按照安装向导的提示操作，单击“继续”或“确定”按钮，进入选择安装类型的对话框，如图 1-8 所示。

在该对话框中，用户可以选择 Visual FoxPro 6.0 安装的文件夹和安装类型。系统提供了两种安装类型：

图 1-8

- 典型安装：安装系统默认的最典型的组件；
- 自定义安装：用户可以选择自己需要的组件进行安装。

通过“自定义安装”,用户可以实现“最小安装”和“完全安装”。单击“自定义安装”按钮,将进入“自定义安装”对话框,如图 1-9 所示。

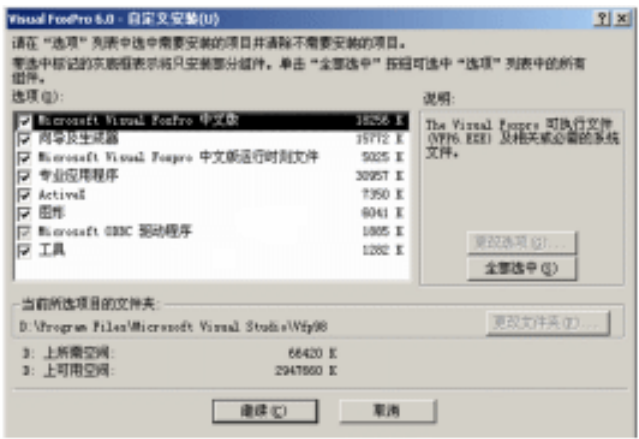


图 1-9

单击“全部选中”按钮,即可实现“完全安装”。若只选择安装第一项“Microsoft Visual FoxPro 中文版”,则可实现“最小安装”。

⑤ 选择好要安装的组件之后,单击“继续”按钮,系统将进入安装过程,安装过程需要几分钟。安装完成后,系统会询问你是否“安装 MSDN”联机帮助以及是否“现在注册”。安装 MSDN 联机帮助需要在光驱中插入 MSDN 光盘,进行注册需要因特网的支持。你可以选择暂时不安装,以后再注册。最后将

显示“Visual FoxPro 6.0 安装程序已成功安装”的对话框，如图 1-10 所示，单击“确定”按钮即完成了整个安装过程。一旦安装完毕，“Microsoft Visual FoxPro 6.0”将加入到 Windows“开始”菜单的“程序”菜单中。

图 1-10

当安装 Visual FoxPro 6.0 或 MSDN 时，若没有完全安装，如选择“典型安装”方式或在“自定义安装”时只安装了部分组件，在以后的使用过程中，可能还需要再安装一些组件。这时可再次执行 Setup.exe 程序（针对 Visual FoxPro 6.0 或 MSDN 分别在不同的目录有各自的 Setup.exe 程序），进入安装程序之后有三种安装方式可供选择：

添加/删除：为当前安装添加新组件，或删除已有的安装组件；

重新安装：重复上一次的安装，恢复丢失的文件和重新设置系统；

全部删除：删除已有的组件。

1.3.3　Visual FoxPro 6.0 的启动和退出

1. Visual FoxPro 6.0 的启动

启动 Visual FoxPro 6.0 有多种方法，我们介绍其中的一种。

从“开始”菜单中选择“程序”选项，在下拉菜单中选择“Microsoft Visual FoxPro 6.0”选项，再选择“Microsoft Visual FoxPro 6.0”，单击左键，就可以进入 Microsoft Visual FoxPro 6.0 系统。其界面如图 1-11 所示。

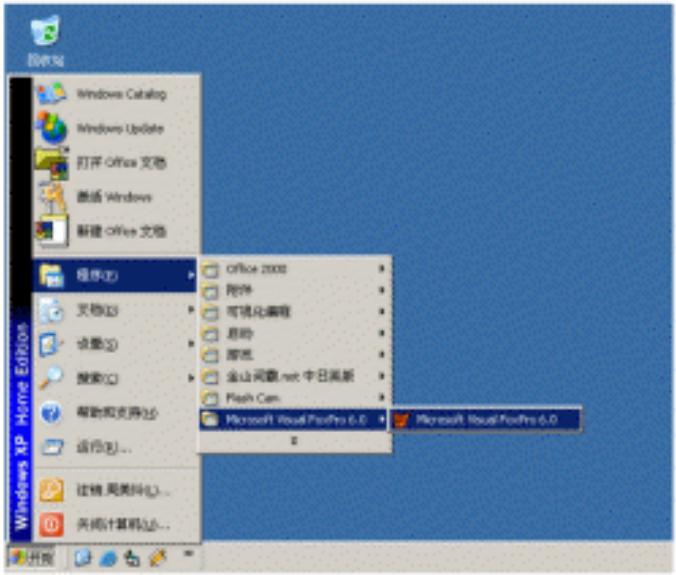


图 1-11

启动之后的界面如图 1-12 所示。

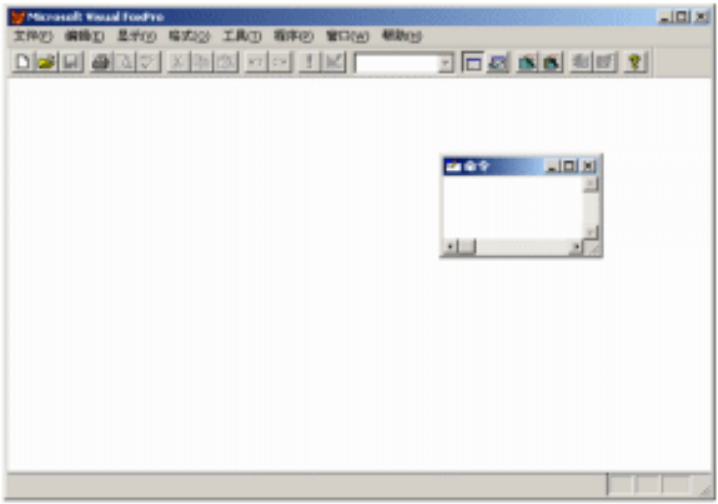


图 1-12

从界面中可以看出，Visual FoxPro 6.0 有两个窗口：主窗口和命令窗口。

2. Visual FoxPro 6.0 的退出

要退出 Visual FoxPro 6.0 系统，可以采用以下几种方法：

① 单击主窗口右上角的关闭按钮。

② 在 Visual FoxPro 6.0 主菜单中，打开“文件”菜单，选择“退出”命令。

③ 用组合键 Alt + F4 退出。

④ 当程序停止响应时，按组合键 Ctrl + Alt + Del，进入“关闭程序”窗口，按“结束任务”退出。

§1.4　Visual FoxPro 基本概念

1.4.1　Visual FoxPro 6.0 的项目

Visual FoxPro 6.0 的项目是一个包含广泛的概念，简单地说，项目是指文件、数据、文档和各种对象的集合。项目被保存在以 .pjx 为扩展名的文件中。项目是用“项目管理器”来管理的。

项目和 Visual FoxPro 应用程序之间有密切的关系。一个 Visual FoxPro应用程序会包含很多种文件，Visual FoxPro 6.0 把这些文件放到项目管理器中统一组织和管理。

1.4.2　Visual FoxPro 6.0 的文件

Visual FoxPro 6.0 提供的文件类型有很多种（见表 1-4）。

表 1-4　Visual FoxPro 6.0 的文件类型

文件类型	扩展名	说　明
项目文件	.pjx	实现对项目中其他类型文件的组织
项目备注文件	.pjt	包含相应的.pjx 的相关文件
表文件	.dbf	存储表结构及记录
表备注文件	.fpt	存储相应的.dbf 文件的备注与通用字段的实际内容
数据库文件	.dbc	数据库表的集合及一个数据库内完整性的定义、检验规则等
数据库备注文件	.dct	存储相应.dbc 的相关信息
索引、压缩索引文件	.idx	只有单索引的标准索引及压缩索引文件
复合索引文件	.cdx	有若干索引标识符的复合结构索引文件
程序文件	.prg	又称命令文件，用于存储用 Visual FoxPro 6.0 语言编写的程序

续 表

文 件 类 型	扩展名	说 明
编译过的程序文件	. fxp	对. prg 文件进行编译后产生的文件
生成的查询程序	. qpr	存储通过查询设计器设置的查询条件和查询输出要求等
编译后的查询程序	. qpx	对. qpr 文件进行编译后产生的文件
表单文件	. scx	用于存储表单的内容
表单备注文件	. sct	存储相应. scx 文件的有关信息
生成的屏幕程序	. spr	由表单定义而生成的程序文件,只适用于 FoxPro 以前的版本
编译后的屏幕程序	. spx	对. spr 编译后生成的
菜单文件	. mnx	存储菜单的内容
菜单备注文件	. mnt	存储相应的. mnx 文件的有关信息
生成的菜单程序文件	. mpr	根据菜单格式文件而自动生成的程序文件
编译后的菜单程序	. mpx	对. mpr 进行编译而产生的文件
报表文件	. frx	存储报表的定义数据
报表备注文件	. frt	存储相应的. frx 文件的有关信息
标签文件	. lbx	存储标签定义的内容
标签备注文件	. lbt	存储相应的. lbx 文件的有关信息
内存变量文件	. mem	存储已定义的内存变量,以便需要时可从中恢复它们
FoxPro 2. x 视图文件	. vue	存储程序运行环境的设置,以备需要时恢复所设置的环境
文本文件	. txt	用于供 Visual FoxPro 6. 0 与其他高级语言间进行数据交换
可视类库	. vcx	用于存储一个或多个类定义
生成的应用程序	. app	可在 Visual FoxPro 6.0 环境支持下,用 DO 命令直接运行
可执行程序	. exe	与. app 类似,可脱离 Visual FoxPro 6. 0 环境而独立运行
ActiveX 控件	. ocx	将. ocx 并到 Visual FoxPro 6.0 后,可像基类一样使用其中的对象
Windows 动态链接库	. dll	包含能被 Visual FoxPro 6. 0 和其他 Windows 应用程序使用的函数
FoxPro 动态链接库	. fll	与. dll 类似,包含专为 Visual FoxPro 6.0 内部调用建立的函数

第2章 Visual FoxPro 6.0的基本操作方法

§2.1 Visual FoxPro 6.0 系统菜单的使用

2.1.1 Visual FoxPro 6.0 用户界面

正常启动 Visual FoxPro 6.0 **后,首先进入的是** Visual FoxPro 6.0 **系统的主屏幕界面,如图** 2-1 **所示。**

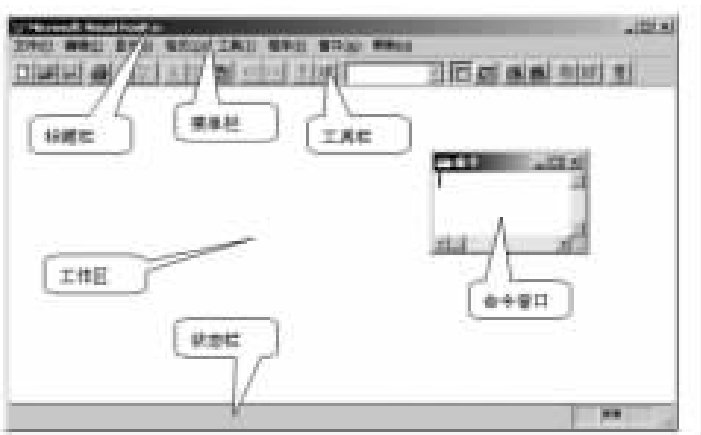


图 2-1 Visual FoxPro **用户界面**

从图中可以看出，Visual FoxPro 6.0 的界面是由标题栏、菜单栏、工具栏、工作区、状态栏和命令窗口组成的。

标题栏位于界面的第一行，它包含系统程序图标、主屏幕标题、最小化按钮、最大化按钮和关闭按钮 5 个对象。

下面对相关内容加以简要介绍。

1. 系统程序图标

单击 Microsoft Visual FoxPro 系统程序图标，可以打开窗口控制菜单，在窗口控制菜单下，分别有还原、移动、大小、最小化、最大化、关闭等菜单项可供选择 双击系统程序图标，可以关闭 Visual FoxPro 系统。

2. 主屏幕标题

主屏幕标题是系统定义的该窗口的名称，可以使用_SCREEN 或_VFP 命令，根据自己的兴趣改变它的内容。

如在命令窗口中输入以下命令，将主屏幕标题依次改为“abcdefg”、“1234567”、“Microsoft Visual FoxPro”：

```
_SCREEN. caption = "abcdefg"
_VFP. caption = "1234567"
_SCREEN. caption = "Microsoft Visual FoxPro"
```

3. 其他部分

命令窗口是 Visual FoxPro 6.0 系统命令编辑、执行的窗口。在命令窗口中，可以输入命令实现对数据库的操作和管理 也可以用各种编辑工具对操作命令进行修改、插入、删除、剪切、拷贝、粘贴等操作 还可以在此窗口建立命令文件并运行命令文件。命令窗口的显示可以通过主菜单的“窗口”菜单来控制。在“窗口”菜单下，选择“隐藏”，可以关闭命令窗口 选择“命令窗口”，可以弹出命令窗口。

在工具栏和状态栏之间的一大块空白区域是系统工作区，各种工作窗口将在这里展开。

状态栏位于屏幕的最底部，用于显示某一时刻的管理数据的工作状态。通过“SET STATUS”命令可以设置状态行：若使用“Set status off”命令，屏幕上不出现状态行 若“Set Status”是“ON”状态，屏幕上出现状态行。如果当前工作区中没有表文件打开，状态行的内容是空白的 如果当前工作区中有表文件打开，则状态行显示表名、表所在的数据库名、表中当前记录的记录号、表中的记录总数、表中当前记录的共享状态等信息。

菜单栏和工具栏的内容较多，我们将用专门的章节进行讲解。

2.1.2　主菜单栏

主菜单又称系统菜单，它包含：文件、编辑、显示、格式、工具、程序、窗口和帮助共 8 个菜单选项。当单击其中一个菜单选项时，可以打开一个对应的下拉式菜单，在该下拉式菜单下，通常还有若干子菜单选项，当选择其中一个子菜单选项时，就可以执行一个操作。

下面介绍常用的菜单选项，以及对应的下拉式菜单所包含的子菜单选项的内容和功能。

1. “文件”菜单选项

在“文件”菜单中，包含各种与文件有关的子菜单选项。这些选项的内容及功能见表 2-1。

表 2-1　文件菜单选项及功能

菜单选项	功　　能
新建	打开新建对话框，对话框中的选项用于创建新的项目、数据库、表格、查询、连接、视图、表单、报表、标签、程序、类、文本文件和菜单
打开	打开“打开”对话框，用于打开新建中所列的所有文件
关闭	关闭活动窗口，如果按下 Shift 并打开文件菜单，此选项将变为全部关闭，即关闭所有窗口
保存	用于保存当前活动窗口中的文件，对于没有文件名的新文件，此选项提示用户输入一个文件名
另存为	提示用户在存储文件前输入一个新文件名或改变存储路径
还原	在当前编辑会话期间取消对当前文件所做的修改
导入	引入一个 Visual FoxPro 6.0 文件或由其他应用程序格式化的文件，还可用此选项启动 Import Wizard (引入向导)
导出	将 Visual FoxPro 6.0 文件以另一个应用程序的文件格式输出
页面设置	为报表改变页面布局和打印机设置
打印预览	在窗口中预览要打印页，与打印出来的形式相同
打印	打印当前窗口中的一个文件或 Visual FoxPro 6.0 剪贴板的内容
发送	用于发送 E-mail
打开项目	为重新打开最近的 4 个项目提供快捷方式
退出	退出 Visual FoxPro 6.0，选择此项与在命令窗口输入 QUIT 命令的效果相同

2. “编辑”菜单选项

“编辑”菜单选项包含用于编辑程序、表格和报表文件等的子菜单选项，这些命令分别是：撤消、重做、剪切、复制、粘贴、选择性粘贴、清除、全部选定、查找、再次查找、替换、定位行、插入对象、对象、链接和属性，这些命令的功能见表 2-2。

表 2-2 编辑菜单选项及功能

菜单选项	功 能
撤消	恢复在最后一次存盘前所做的修改
重做	恢复上一次被还原的修改
剪切	删除当前文档中被选定的文本和对象，并将其存放到剪贴板
复制	将所选定的文本和对象拷贝到剪贴板上
粘贴	将剪贴板上当前的内容拷贝到当前光标处
选择性粘贴	用于将其他应用程序中的 OLE 对象插入到常规字段中
清除	删除所选文本
全部选定	选择当前活动窗口中所有的对象
查找	显示查找对话框，用于定位文件中的文本字符串
再次查找	从当前插入点位置开始重复上次的搜索，而不是从文档开始处查找
替换	显示替换对话框，用于定位并替换文件中的文本字符串
定位行	用于在程序调试时将光标移至程序文件中的指定行
插入对象	将对象嵌入常规类型字段
对象	为编辑选定的 OLE 对象提供各种选项
链接	打开被链接的文件 (OLE) 并允许用户编辑该链接
属性	打开编辑属性窗口

3. “显示”菜单选项

在“显示”菜单选项中，子菜单选项的内容是由当前操作环境决定的。当用户尚未打开要显示的文件时，子菜单选项中只有一项 (工具栏)；当打开表、表单或报表等文件时，子菜单选项将附加一些与打开文件对应的菜单选项，这些选项的内容及功能见表 2-3。

表 2-3　显示菜单选项及功能

菜单选项	功　　能
编辑	改变、查看和修改记录的编辑布局风格
浏览	改变、查看和修改记录的浏览布局风格
追加模式	在表格的末尾添加一条空白记录并将记录指针移至其中的第一字段
Tab 键次序	允许用户在表单中设置 Tab 顺序
数据环境	设置用于表单、表单集或报表中的表格和关系
属性	显示表单及控件的属性对话框
代码	当对象编辑方法时打开代码窗口
表单控件工具栏	在使用表单设计器时，打开表单控件工具栏
报表控件工具栏	显示报表控件工具栏，用于向报表中加入控件
布局工具栏	打开布局工具条，帮助用户对齐控件
调色板工具栏	打开调色板工具栏，以便用户为控件选择前景色和背景色
数据库设计器	打开数据库设计器，用于维护数据库中存储的表格、视图和关系
表单设计器	打开表单设计器
网格线	对是否显示网格线进行选择
工具栏	显示含有 Visual FoxPro 6.0 所使用的每一个工具栏列表的对话框

4. “格式”菜单选项

“格式”菜单选项包含控制字体格式、文本缩进及空格控制等子菜单选项，这些命令分别是：字体、放大字体、缩小字体、1 倍行距、1.5 倍行距、两倍行距、缩进、撤消缩进、注释、撤消注释，这些命令的功能见表 2-4。

表 2-4　格式菜单选项及功能

菜单选项	功　　能
字体	选择字体及显示风格
放大字体	放大当前窗口中使用的字体
缩小字体	缩小当前窗口中使用的字体
1 倍行距	当前窗口中的文字按单行分隔
1.5 倍行距	当前窗口中的文字每两行间空 1.5 行
两倍行距	当前窗口中的文字每两行间空 2 行
缩进	缩进当前窗口中的当前行或所选行
撤消缩进	取消当前窗口中的当前行或所选行的缩进
注释	注释所选行
撤消注释	取消所选行的注释

5. “工具”菜单选项

“工具”选项包含向导菜单选项，另外还提供编程工具、程序调试器、系统环境设置等子菜单选项。这些选项的内容及功能见表 2-5。

表 2-5 工具菜单选项及功能

菜单选项	功 能
向导	Visual FoxPro 6.0 中向导的列表及对它们的访问
拼写检查	主要用于文本字段和备注字段的拼写检查
宏	定义并维护键盘宏
类浏览器	检查任何类的内容以查看它的属性和方法
修饰	加入缩进的大写字符以及对程序文件重新格式化
调试器	打开调试器窗口
选项	提供了使用 Visual FoxPro 6.0 配置选项的方法

6. “程序”菜单选项

包含与程序编译、运行有关的子菜单选项，这些选项的内容及功能见表 2-6。

表 2-6 程序菜单选项及功能

菜单选项	功 能
运行	运行从对话框中选定的程序
取消	取消当前的程序
继续	恢复处于挂起状态的当前程序的运行
挂起	挂起当前程序的执行，但不将其从内存中删除
编译	将源文件编译成目标代码
执行	执行当前程序文件

7. “窗口”菜单选项

“窗口”菜单选项中包含对已打开的窗口进行管理的若干子菜单选项，这些选项的内容及功能见表 2-7。

表 2-7　窗口菜单选项及功能

菜单选项	功　　能
全部重排	以不互相重叠的方式排列所有打开的窗口
隐藏	将活动窗口隐藏，但不将其从内存中删除
消除	从应用程序的工作空间或当前输出窗口中消除文本
循环	从一个打开的窗口移到下一个窗口，使下一个窗口显示在最前面
命令窗口	打开一个命令窗口，激活命令窗口并显示在最前面
数据工作期	打开一个数据工作期窗口，激活数据工作期窗口并显示在最前面

8. “帮助”菜单选项

使用 Visual FoxPro 6.0 的帮助系统，用户可以快速查询到有关 Visual FoxPro 6.0 设计工具和程序语言的信息。如果对某个窗口或某个对话框的内容不理解，只要按 < F1 > 键，就可以显示出有关该窗口或对话框的上下文相关的帮助信息。

选择“帮助”菜单的相关命令，可以得到 Visual FoxPro 6.0 联机帮助的内容概述。“帮助”菜单选项提供了如何获得帮助信息的若干子菜单选项，这些选项的内容及功能见表 2-8。

表 2-8　帮助菜单选项及功能

菜单选项	功　　能
帮助主题	以分级的形式显示帮助信息
文档	打开 Visual FoxPro 6.0 联机文档
示例应用程序	描述了由 Visual FoxPro 6.0 提供的例子应用程序
技术支持	列出了 Microsoft 提供的可选资源列表，这些资源提供了附加的帮助
关于 Microsoft Visual FoxPro	显示本产品版权屏幕、授权人、版本日期、资源文件名及缺省目录。其他特性还包括列出系统信息和运行其他程序功能

在 Visual FoxPro 6.0 的联机文档中带有非常详细的帮助内容，如安装指南、用户指南、开发指南、语言参考等。在语言参考中，有 Visual FoxPro 6.0 全部的语句和函数，用户应该学会通过联机文档学习 Visual FoxPro 6.0。若要在联机文档中查找有关特定术语或主题的帮助内容，选择帮助中的“索引”选项卡，在其中输入索引关键字进行搜索。

2.1.3 工具栏

1. 工具栏种类

Visual FoxPro 6.0 系统提供的工具栏有如下 11 种：常用、布局、表单控件、表单设计器、查询设计器、视图设计器、数据库设计器、报表控件、报表设计器、调色板和打印预览工具栏。

2. 激活工具栏

默认情况只有“常用”工具栏可见。当使用一个设计器工具如表单设计器时，该设计器将显示使用它工作时常用的工具栏。我们可以在任何需要时激活一个工具栏，方法是：从“显示”菜单选择“工具栏”，在弹出的对话框中进行选择，如图 2-2 所示。在希望激活的工具栏上单击，使工具栏左边方框打上“×”标记，最后单击“确定”。

图 2-2

3. 清除工具栏

若不想使用某个工具栏，可在如图 2-2 所示的对话框中清除欲使之不活动的工具栏左边的“×”标记，单击“确定”按钮。

4. 重新设置工具栏

打开如图 2-2 所示的对话框，先单击“重置”按钮，然后单击“定制”按钮，

进入“定制工具栏”对话框，就可以根据系统工具栏提供的工具，配置图 2-2 所选定的“表单设计器”工具栏了。定制工具栏的界面如图 2-3 所示。

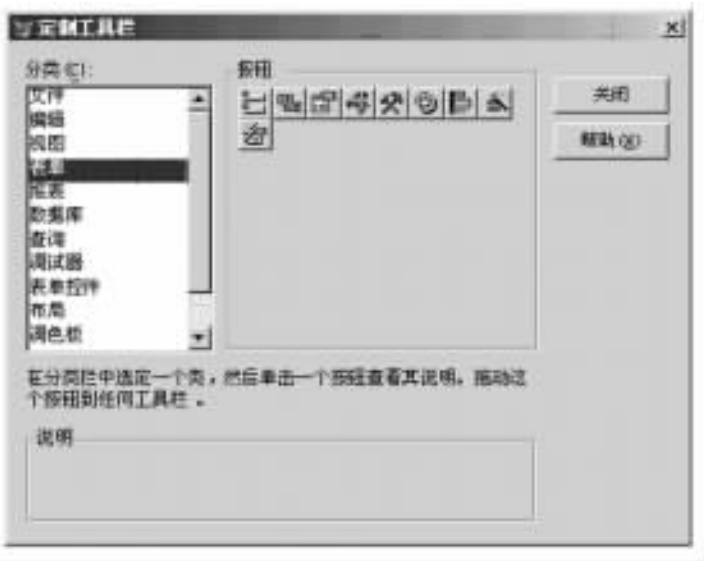


图 2-3

5. 建立新工具栏

在图 2-2 所示的对话框中，单击“新建”按钮，进入“新工具栏”对话框。在该对话框中要求输入新建工具栏名称，如图 2-4 所示，输入“我的工具栏”，然后单击“确定”按钮，进入“定制工具栏”对话框。选择工具栏种类，把所需要的按钮拖到新建的工具栏上，这样就建立了一个新的工具栏——“我的工具栏”，如图 2-5 所示。为了建立如图 2-5 所示的工具栏，要把“文件”分类中的“新建”、“打开”和“保存”按钮，“编辑”分类中的“剪切”、“复制”和“粘贴”按钮拖到新建的工具栏上。

图 2-4

图 2-5

2.1.4 配置 Visual FoxPro 6.0

安装 Visual FoxPro 6.0 后，可以根据需要定制开发环境。环境设置包括主窗口标题、默认目录、项目、编辑器、调试器及表单工具选项、临时文件存储、拖放字段对应的控件和其他选项。用户既可以用交互式，也可以用编程的方法配置 Visual FoxPro 6.0，甚至可以使 Visual FoxPro 6.0 启动时调用用户自建的配置文件。

1. **使用“选项”对话框配置环境**

可以在命令窗口中使用 SET 命令设置环境，也可以在主菜单下，选择“工具”，再选择“选项”，打开“选项”对话框进行设置。“选项”对话框如图 2-6 所示。

在“选项”对话框中，有 12 种不同类别的环境选项卡，每一个选项卡又有相应的设置信息的对话窗口，用户可以根据需要修改每一个选项卡的每一个参数，从而定制开发环境。可以把在“选项”中所做设置保存为在当前工作期有效或者 Visual FoxPro 6.0 默认（永久）设置。

图 2-6

① 在“选项”对话框中选择好各种设置，单击“确定”按钮，则当前设置保存为仅在当前工作期有效。当前设置一直起作用直到退出 Visual FoxPro 6.0（或直到再次更改它们）。

② 要永久保存所做更改，请把它们保存为默认设置，这将把它们存储在 windows 注册表中。把当前设置保存为默认设置的步骤为：在“选项”对话框中更改设置，选择“设置为默认值”按钮。对当前设置做更改之后，“设置为默认值”按钮才被激活为可用状态。

表 2-9 介绍了 12 种选项卡的特性。

表 2-9　“选项”对话框各选项卡特性列表

选项卡	选项卡功能
显示	界面选项，确定是否显示状态栏、时钟、命令结果或系统信息
常规	数据输入与编程选项，设置警告声音，确定是否记录编译错误，是否自动填充新记录 ;使用什么定位键，确定调色板使用什么颜色以及改写文件之前是否警告等
数据	表选项，确定是否使用 Rushmore 优化，是否使用索引强制惟一性，查找的记录计数器间隔以及使用什么锁定选项

续 表

选项卡	选 项 卡 功 能
远程数据	远程数据访问选项,确定连接超时限定,一次获取记录数据以及如何使用 SQL 更新
文件位置	确定 Visual FoxPro 6.0 默认目录路径,帮助文件以及临时文件等存储的位置
表单	表单设计器选项,确定网格面积,所用刻度单位,最大设计区域以及使用何种类模板
项目	项目管理器选项,确定是否提示使用向导,双击时运行或修改文件以及源代码管理选项
控件	确定是否使用“表单控件”工具栏中的“查看类”按钮,所提供的有关可视类库和 OLE 控件选项
区域	确定日期、时间、货币以及数字格式
调试	调试器显示及跟踪选项,确定使用什么字体与颜色
语法着色	确定区分程序元素 (注释与关键字) 所用的字体与颜色
字段映象	确定从数据环境设计器、数据库设计器或项目管理器中向表单拖动表或字段时创建何种控件

2. 恢复 Visual FoxPro 6.0 环境

如果希望关闭所有操作返回 Visual FoxPro 6.0 启动时的状态,在命令窗口或在退出 Visual FoxPro 6.0 之前最后调用的程序中,按下列顺序运行如下命令:

```
CLEAR ALL
CLOSE ALL
CLEAR PROGRAM
```

CLEAR ALL 命令从内存中移去所有的对象,按顺序关闭所有私有数据工作期以及其中的临时表。

CLOSE ALL,在 CLEAR ALL 正确执行后,关闭 Visual FoxPro 6.0 默认数据工作期,即数据工作期 1 中的所有数据库、表以及临时表。

CLEAR PROGRAM 清除最近执行程序的程序缓冲区。CLEAR PROGRAM 迫使 Visual FoxPro 6.0 从磁盘而不是从程序缓冲区中读取文件。

§2.2 Visual FoxPro 6.0 辅助设计工具

为了加快 Visual FoxPro 6.0 应用程序的开发,减轻用户的程序设计工作量,Visual FoxPro 6.0 提供了 3 类辅助设计工具:向导、设计器和生成器。

2.2.1　向导

向导是交互式的程序，能帮助你快速完成一般性的任务，如创建表单、设置报表格式、建立查询等。通过在向导的一系列屏幕提示中回答问题或选择选项，可以让向导建立一个文件或者根据选择完成一项任务。

Visual FoxPro 6.0 有 20 余种向导工具。从创建表、视图、查询等数据文件，到建立报表、标签、图表、表单等 Visual FoxPro 6.0 文档，直至创建 Visual FoxPro 6.0 的应用程序、SQL 服务器上的数据库等操作，均可使用相应的向导工具来完成。表 2-10 列出了 Visual FoxPro 6.0 提供的 21 种向导的名称及简明用途说明。

表 2-10　Visual FoxPro 6.0 向导一览表

向导名称	用途
表向导	创建一个表
查询向导	创建查询
本地视图向导	创建一个视图
远程视图向导	创建一个远程视图
交叉表向导	创建一个交叉表查询
文档向导	格式化项目和程序文件中的代码并从中生成文本文件
图表向导	创建一个图表
报表向导	创建报表
分组/总计报表向导	创建具有分组和总计功能的报表
一对多报表向导	创建一个一对多报表
标签向导	创建邮件标签
表单向导	创建一个表单
一对多表单向导	创建一个一对多表单数据库
透视表向导	创建数据库透视表
邮件合并向导	创建一个邮件合并文件
安装向导	从发布树中的文件创建发布磁盘
升迁向导	创建一个 Oracle 数据库，使之尽可能多地重复 Visual FoxPro 6.0 数据库的功能
SQL 升迁向导	创建一个 SQL Server 数据库，使之尽可能多地重复 Visual FoxPro 6.0 数据库的功能
导入向导	导入或追加数据
应用程序向导	创建一个 Visual FoxPro 6.0 应用程序
WWW 搜索页码向导	创建 Web 页面，使该页的访问者可以从 Visual FoxPro 6.0 表中搜索及检索记录

1. 启动向导

启动向导有两种方法：

① 选择“文件”菜单的“新建”命令，在“新建”对话框中选择要新建的文件类型，然后单击“向导”按钮，相应的向导就会启动。

② 利用“工具”菜单中的“向导”命令，可以访问大多数的向导。

2. 定位向导屏幕

启动向导后，要依次回答每一屏幕所提出的问题。在准备好进行下一个屏幕的操作时，可选择“下一步”按钮。如果操作中出现错误或原来的想法发生了变化，可选择“上一步”按钮来查看前一屏幕的内容，以便进行修改。选择“取消”将退出向导而不会产生任何结果。到达最后一屏时，可以选择“完成”按钮退出向导。还可以选择“完成”按钮直接走到向导的最后一步，跳过中间所要输入的选项信息，而使用向导提供的默认值。

3. 保存向导结果

根据所用向导的类型，每个向导的最后一屏都会要求提供一个标题，并给出保存、浏览、修改或打印结果的选项。

使用“预览”选项，可以在结束向导的操作前查看向导的结果。如果需要作出不同的选择来改变结果，可返回到前边重新进行选择。对向导的结果满意后，则可选择“完成”按钮。

2.2.2 设计器

设计器一般比向导具有更强的功能，可用来创建或者修改 Visual FoxPro 6.0 应用程序所需要的构件，如使用表设计器来定义或修改表、使用表单设计器来定义或修改表单等等。表 2-11 列出了 Visual FoxPro 6.0 中的 9 种设计器的用途。

表 2-11 Visual FoxPro 6.0 设计器一览表

设 计 器	用 途
表设计器	创建表和设置表中的索引
查询设计器	在本地表中运行查询
视图设计器	在远程数据源上运行视图 ;创建可更新的视图
表单设计器	创建表单，用以查看并编辑表中数据
报表设计器	创建报表，显示及打印数据
标签设计器	创建标签布局及打印标签
数据库设计器	建立数据库 ;在不同的表之间创建并查看关系
连接设计器	为远程视图创建连接
菜单设计器	创建菜单或快捷菜单

若要用设计器创建新文件,可通过“文件”菜单的“新建”命令,在“新建”对话框中,选择文件类型,然后单击“新建文件”按钮,相应的设计器就会打开。

若要使用设计器修改文件,可通过“文件”菜单的“打开”命令打开某个文件,一般地,相应的设计器会随着文件的打开而自动打开。

2.2.3　生成器

生成器的主要功能是在 Visual FoxPro 6.0 应用程序的构件中生成并加入某类控件,如生成一个组合框或生成一个列表框等等。生成器是带有选项卡的对话框,用于简化对表单、复杂控件和参照完整性代码的创建和修改过程。每个生成器显示一系列选项卡,用于设置选中对象的属性。可使用生成器在数据库表之间生成控件、表单、设置控件格式和创建参照完整性。

Visual FoxPro 6.0 提供的生成器见表 2-12。

表 2-12　Visual FoxPro 6.0 生成器一览表

生成器名称	功　　能
组合框生成器	生成组合框
命令按钮生成器	生成命令按钮
编辑框生成器	生成编辑框
表单生成器	生成表单
表格生成器	生成表格
列表框生成器	生成列表框
选项按钮生成器	生成选项按钮
文本框生成器	生成文本框
自动格式化生成器	格式化控件组
参照完整性生成器	在数据库表间创建参照完整性

若要生成一个控件,可从“表单控件”工具栏中,选择“生成器锁定”按钮。每次向表单添加新控件,Visual FoxPro 6.0 会显示一个适当的生成器。或者,从表单上选择控件,接着在快捷菜单中选择“生成器”命令。这样做的前提是 Visual FoxPro 6.0 对要添加的控件提供了生成器。

若要使用“表单生成器”,可以从“表单”菜单中,选择“快速表单”命令。

如果表单中已经有多个控件,可以使用“自动格式生成器”,同时设置它们的格式。

§2.3 Visual FoxPro 6.0 的数据类型和数据存储

2.3.1 数据类型

与其他程序设计语言一样，Visual FoxPro 6.0 提供了多种数据类型，可以将数据存入各种类型的表、变量、数组或其他存储容器中。Visual FoxPro 6.0 的数据类型分为两大类：一类用于变量和数组；另一类用于表中的字段。

1. 字符型 (Character)

由字母 (汉字)、数字、空格等任意 ASCII 码组成。字符数据的长度为 0 ~ 254，每个字符占 1 个字节。

2. 数值型 (Numeric)

用来表示数量，它由数字 0 ~ 9、一个符号 (+ 或 -) 和一个小数点 (.) 组成。数值型数据的长度为 1 ~ 20，每个数据占 8 个字节。

数值型数据的取值范围是 - .9999999999E +19 ~ .9999999999E +20。

3. 货币型 (Currency)

在使用货币值时，可以使用货币型来代替数值型。默认格式是“ $ 数值量”，货币型数据的取值范围是

- 922 337 203 685 477.580 7 ~ 922 337 203 685 477.580 7。

小数位数超过 4 位时，系统将进行四舍五入。每个货币型数据占 8 个字节。

4. 日期型 (Date)

用以保存不带时间的日期值。日期型数据的存储格式为“ yyyymmdd ”，其中 yyyy 为年，占 4 位；mm 为月，占 2 位；dd 为日，占 2 位。

日期型数据的表示有多种格式，最常用的格式为 mm/dd/yyyy。

日期型数据的取值范围是：公元 0001 年 1 月 1 日 ~ 公元 9999 年 12 月 31 日。

5. 日期时间型 (DateTime)

用以保存日期和时间值。日期时间型数据的存储格式是“ yyyymmddhhmmss ”，其中 yyyy 为年，占 4 位；mm 为月，占 2 位；dd 为日，占 2 位；hh 为时间中的小时，占 2 位；mm 为分钟，占 2 位；ss 为秒，占 2 位。

日期时间型数据中可以只包含一个日期或只包含一个时间值，缺省日期值

时,系统自动加上 1999 年 12 月 31 日 缺省时间值时,则自动加上午夜零点。

6. 逻辑型 (Logical)

存入的值只有真 (.T.) 和假 (.F.) 两种状态,占 1 个字节。

以下数据类型只能被用于数据表中的字段:

7. 双精度型 (Double)

用于取代数值型,以便能提供更高精度的数值,只能用于数据表中字段的定义。它采用固定存储长度的浮点数形式。每个双精度型数据占 8 个字节。

双精度型数据的取值范围是 +/- 4.94065645841247E - 324 ~ +/- 8.9884656743115E307。

8. 浮点型 (Float)

浮点型在功能上与数值型等价,包含此类型是为了提供兼容性,只能用于数据表中字段的定义。

9. 通用型 (General)

用于存储 OLE 对象,只能用于数据表中字段的定义。该字段包含了对 OLE 对象的引用,而 OLE 对象的具体内容可以是一个电子表格、一个字处理器的文本、图片等,是由其他软件建立的。通用型字段占 4 个字节,用来引用它的实际内容,实际内容存放在与表文件同名的备注文件中。

10. 整型 (Integer)

用于存储无小数部分的数值,只能用于数据表中字段的定义。整型字段占 4 个字节,取值范围是 - 2147483647 ~ 2147483647。

整型以二进制形式存储,不像数值型那样需要转换成 ASCII 字符存储。

11. 备注型 (Memo)

用于字符型数据块的存储,只能用于数据表中字段的定义。备注型字段占 4 个字节,并用这 4 个字节来引用备注的实际内容。实际备注内容存储在相应的备注文件中,故备注型字段的大小仅受限于现有的磁盘空间。

12. 字符型 (二进制)

用于存储任意不经过代码页修改而维护的字符数据,只能用于数据表中字段的定义。

13. 备注型 (二进制)

用于存储任意不经过代码页修改而维护的备注型数据,只能用于数据表中字段的定义。

2.3.2 常量

常量是一个命名的数据项,是在命令或程序中直接引用的实际值,其特征是在所有的命令操作或程序运行过程中其值不变。常量有数值型、浮点型、字符型、逻辑型、日期型、日期时间型六种。

1. 数值型常量

由数字(0~9)、小数点和正负号组成。

例如:-123.56,768,+32567.67。

2. 浮点型常量

是数值型常量的浮点格式。

例如:-123E+12,-3645E-89。

3. 字符型常量

由汉字和 ASCII 字符集中可打印字符组成的字符串,使用时必须用定界符(" "或' ')括起来。

例如:"ABCDE"、'清华大学'等。

4. 逻辑型常量

有"真"和"假"两种值。

例如:.t. 或 .T.,.f. 或 .F.。

5. 日期型常量

其规定格式为{mm/dd/yyyy}。

例如:{04/12/1982},{05/04/2002}等。

6. 日期时间型常量

其规定格式为{mm/dd/yyyy hh:mm:ss}。

例如:{04/12/1982 10:30:00}。

2.3.3 变量及基本操作

1. 变量的分类

Visual FoxPro 6.0 中有 3 种形式的变量:内存变量、数组变量和字段变量。

内存变量是存放单个数据的内存单元。

数组变量是存放多个数据的内存单元组,我们将在后面的章节详细介绍。

字段变量是数据库系统中的一个重要概念。字段和记录一纵一横构成了

数据表的基本结构。一个数据库文件是由若干相关的数据表组成，一个数据表是由若干个具有相同属性的记录组成，而每一个记录又是由若干个字段组成。字段变量就是指数据表中已定义的任意一个数据项。字段变量必须依附于表，随着表的打开和关闭而在内存中存储和被释放。字段变量是一多值变量，其值为表的当前记录在该字段的分量值。

Visual FoxPro 6.0 内存变量又可分为系统变量和用户定义的内存变量。

① 系统变量是 Visual FoxPro 6.0 自带的变量，由系统命名创建并维护。系统变量以“_”开头，不能被删除。

② 用户定义的内存变量是一种临时工作单元，它独立于数据库文件的存在，常用来保存在命令或程序执行中临时用到的输入、输出或中间数据，由用户根据需要设置或删除。内存变量的类型有字符型、逻辑型、日期型、数值型等，由赋值给它的数据决定。

通常所说的内存变量仅指用户定义的内存变量。

每个变量都有一个名称，称为变量名。Visual FoxPro 6.0 通过相应的变量名来引用该变量。变量名的命名规则是：以字母或下划线开头，由字母、数字及下划线组成，长度为 1 ~ 128 个字符，不能使用 Visual FoxPro 6.0 的保留字。在中文 Visual FoxPro 6.0 中，可以以汉字开头并包含汉字，每个汉字占 2 个字符。

2. 内存变量的赋值命令

命令格式：格式 1　　STORE　表达式 TO　内存变量名清单

　　格式 2　　内存变量名 = 表达式

功能：格式 1 可以给一组内存变量赋相同的值，格式 2 只能给一个内存变量赋值。

例如：STORE　“Visual FoxPro 6.0”　To ss

　　ss = “Visual FoxPro 6.0”

两者完全等价。

内存变量的类型和值由所赋数据的类型和值决定，并以最近一次所赋的值为准。内存变量名与字段变量名同名时，字段变量被优先引用。若要引用内存变量，可在内存变量名前加前缀 M. ，以示区别。

3. 内存变量的主要操作命令

```
DISPLAY / LIST MEMORY        && 显示当前内存中的内存变量
SAVE MEMORY …                && 将当前内存中的内存变量保存到指定
                                的内存变量文件中
```

```
RESTORE FROM MEMORY …   && 内存变量文件中的内存变量恢复到内存
CLEAR MEMORY            && 清除当前内存中的内存变量,释放存储空间
?变量名                 && 显示变量的值
例如:
STORE 0 TO A1,A2,A3     && 建立内存变量 A1,A2,A3,并赋数值型值 0
?A1,A2,A3               && 显示 A1,A2,A3 的值
0  0  0                 && 系统返回?命令执行的结果
A1 = "ABC123"           && 给 A1 赋值,字符型,值为"ABC123"
? A1                    && 显示 A1 的值
ABC123
USE 学生                && 打开数据表"学生.dbf"
GO 3                    && 将记录指针移至第 3 条记录
DISP FIELDS xm          && 显示当前记录 xm 字段的值:李明
xm = "张三"             && 给内存变量 xm 赋值
? xm                    && 显示字段变量 xm 的当前值:李明
? M.xm                  && 显示内存变量 xm 的值:张三
```

4. 内存变量的作用域

变量的作用域指的是变量的有效范围。变量的作用域包括定义它的过程以及该过程所调用的子过程范围。在 Visual FoxPro 6.0 中,还可以使用 LOCAL、PRIVATE 和 PUBLIC 命令强制规定变量的作用范围。

用 LOCAL 创建的变量只能在创建它们的程序中使用和修改,不能被更高层或更低层的程序访问。

PRIVATE 用于定义私有变量,它用于定义当前过程的变量,并将以前程序定义的同名变量保存起来,在当前程序中使用私有变量而不影响这些同名变量的原始值。

PUBLIC 用于定义全局变量。在本次 Visual FoxPro 6.0 运行期间,所有程序都可以使用这些全局变量。

§2.4 运算符与表达式

运算是对数据进行加工的过程,描述各种不同运算的符号称为运算符,而

参与运算的数据称为操作数。表达式用来表示某个求值规则，它由运算符和配对的圆括号将常量、变量、函数、对象等操作数以合理的形式组合而成。

表达式用来执行运算、操作字符或测试数据，每个表达式都产生惟一的值。表达式的类型由运算符的类型决定。在 Visual FoxPro 6.0 中有 5 类运算符和表达式：算术运算符和算术表达式、字符串运算符和字符串表达式、日期时间运算符和日期时间表达式、关系运算符和关系表达式、逻辑运算符和逻辑表达式。

2.4.1　算术运算符和算术表达式

算术表达式也称数值表达式，它是由算术运算符和数值型常量、数值型内存变量、数值型数组、数值类型字段、返回数值型数据的函数组成。算术表达式的运算结果是常数。

算术表达式的格式为：

＜数值 1＞＜算术运算符 1＞＜数值 2＞[＜算术运算符 2＞＜数值 3＞…]

1. 算术运算符

Visual FoxPro 6.0 提供的算术运算符见表 2-13。在这 6 个算术运算符中，除取负“ - ”是单目运算符外，其他都是双目运算符。运算的含义与数学中基本相同。

表 2-13　算术运算符

运算符	名　称	表达式	表达式值
+	加	3 +2 +1	6
-	减	19 - 2	17
*	乘	5 * 6	30
/	除	78 /2	39
或 * *	乘方	2 * * 8	256
%	取余 (模运算)	10%4	2

在进行算术表达式计算时，要遵循：先括号，在同一括号内，按先乘方，再 * 和 / ，再% ，后 + 和 - 的运算原则。若同处一个级别则按从左到右的顺序计算。

2. 表达式的书写规则

算术表达式与数学中的表达式写法有所区别，应当注意：

① 每个符号占 1 格，所有符号都必须一个一个地并排写在同一横线上，不能在右上角或右下角写方次或下标，如 2^3 要写成 2 3，$x_1 + x_2$ 要写成 x1 + x2。

② 原来在数学表达式中省略的内容必须重新写上，如 2x 要写成 2 * x。

③ 所有括号都用小括号 ()，括号必须成对出现，如：3 [x + 2 (y + z)] 必须写成 3 * (x + 2 * (y + z))。

④ 要把数学表达式中的有些符号，改成 Visual FoxPro 6.0 中可以表示的符号，如要把 2πr 改成 2 * pi * r。

2.4.2 字符串运算符和字符串表达式

字符串表达式是由字符串运算符和字符型常量、字符型内存变量、字符型数组、字符型类型的字段、返回字符型数据的函数组成。字符串表达式的运算结果是字符型常量或逻辑型常量。

字符串表达式的格式为：

<字符串 1> <字符串运算符 1> <字符串 2> [<字符串运算符 2> <字符串 3>…]

Visual FoxPro 6.0 提供的字符串运算符有 3 个，其优先级相同（见表 2-14）。

表 2-14 字符串运算符

运算符	名 称	说 明
+	连接	将字符型数据连接起来
-	空格移位连接	两字符型数据连接时，将前一数据尾部的空格移到后面数据的尾部
$	包含测试	一个字符串是否在另一个字符串中

例如：

“ABC123”+“666xyz” && 连接后的结果为：“ABC123666xyz”

“计算机”+“世界” && 连接后的结果为：“计算机世界”

“12345”+“abcd”+“xyz” && 连接后的结果为：“12345abcd xyz”

“ABC”-“DEFG” && 连接后的结果为：“ABCDEFG”

“计算机”$“计算机软件” && 运算结果为：.T.

在字符串中嵌入引号，只需将字符串用另一种引号括起来即可。例如：

?“abc”+‘ ”’　　&& 运算结果为：abc”

?“abc”+“‘ ”　　&& 运算结果为：abc’

2.4.3　日期时间运算符和日期时间表达式

日期型表达式由算术运算符“ +、- ”、算术表达式、日期型常量、日期型变量、日期型数组、返回日期型数据的函数组成。日期型数据是一种特殊的数值型数据，它们之间只能进行加“ + ”、减“ - ”运算。有以下 3 种情况：

1. 两个日期型数据可以相减，结果是一个数值型数据（两个日期相差的天数）。

例如：

{1999/12/19} - {1999/11/16}　　&& 结果为数值型数据：33

2. 一个表示天数的数值型数据可加到日期型数据中，其结果仍然为一日期型数据（向后推算日期）。

例如：

{1999/11/16} +33　　&& 结果为日期型数据：{1999/12/19}

3. 一个表示天数的数值型数据可从日期型数据中减掉它，其结果仍然为一日期型数据（向前推算日期）。

例如：

{1999/12/19} - 33　　&& 结果为日期型数据：{1999/11/16}

日期时间运算符中（+）运算结果，是把已给的日期时间再加多少秒。

日期时间运算符中（-）运算结果，是计算已给的两个日期时间相差多少秒。

2.4.4　关系运算符和关系表达式

关系表达式是指用关系运算符将两个表达式连接起来的式子（例如 2 +3 >4）。关系运算符又称比较运算符，用来对两个表达式的值进行比较，比较的结果是一个逻辑型常量。

Visual FoxPro 6.0 提供了 7 种关系运算符（见表 2-15）。

表 2-15　关系运算符

运算符	名　称	表 达 式	表达式值
<	小于	8 <9	.T.

< =	小于或等于	4 < = 3	. F.
>	大于	3 + 5 > 10	. F.
> =	大于或等于	4 > = 1	. T.
=	等于	“AA” = “AB”	. F.
< >、#、！ =	不等于	“ab” < > “AB”	. T.
= =	等同于	“123” = = “123”	. T.

说明：

1. 关系运算符的优先级相同，按从左到右的顺序依次进行。

2. 关系运算符两侧的值或表达式的类型应该一致。

3. 数学不等式 a≤x≤b，在 Visual FoxPro 6.0 中不能写成 a < = x < = b。

4. 字符型数据应按其 ASCII 码的值进行比较。在比较两个字符串时，首先比较两个字符串的第一个字符，其中 ASCII 码值较大的字符所在的字符串大。如果第一个字符相同，则比较第二个……依次类推。

5. “ = = ”表示“等同于”，用于精确匹配。

6. 关系运算符两边的表达式只能是数值型、字符串型、日期时间型，不能是逻辑型的表达式或值。

2.4.5 逻辑运算符和逻辑表达式

逻辑表达式是由逻辑运算符、逻辑型常量、逻辑型变量、逻辑型数组、返回逻辑型数据的函数和关系表达式组成的。逻辑表达式的运算结果仍然是逻辑型常量。

Visual FoxPro 6.0 提供的逻辑运算符有 3 种（见表 2-16）。

表 2-16 逻辑运算符

运算符	名 称	表达式	说 明
AND	逻辑与	(4 > 5) AND (3 < 4)	值为. F. ，两个表达式的值均为真，结果才为真
OR	逻辑或	(4 > 5) OR (3 < 4)	值为. T. ，两个表达式中只要有一个为真，结果就为真；只有两个表达式的值均为假，结果才为假
NOT	逻辑非	NOT (1 > 0)	值为. F. ，由真变假或由假变真，进行取反操作

逻辑运算的运算规则见表 2-17。

表 2-17 逻辑运算规则

a	b	a AND b	a OR b	NOT a
.T.	.T.	.T.	.T.	.F.
.T.	.F.	.F.	.T.	.F.
.F.	.T.	.F.	.T.	.T.
.F.	.F.	.F.	.F.	.T.

说明：

不等式 a≤x≤b **可以表示为** a <= x AND x <= b**。**

进行逻辑表达式的计算时，要遵循以下优先顺序：先括号，再 NOT**，再** AND**，后** OR**。**

在早期的版本中，逻辑运算符的两边必须使用点号，如.AND.**，**.OR.**，**.NOT.**，在** Visual FoxPro 6.0 **中，两者可以通用。**

2.4.6　运算符的优先顺序

在同一个表达式中进行多种操作时，Visual FoxPro 6.0 **会按一定的顺序进行求值，称这个顺序为运算符的优先顺序。运算符的优先顺序见表** 2-18**。**

表 2-18　**运算符的优先顺序**

优先顺序	运算符类型	运　算　符
1	算术运算符	-（取负）
2		（乘方运算）
3		*、/（乘法和除法）
4		%（取模运算）
5		+、-（加法和减法）
6	字符串运算符	+、-（字符串连接）
7	关系运算符	<、<=、>、>=、=、<>、==（优先级相同）
8	逻辑运算符	NOT
9		AND
10		OR

说明：

1. **同级运算按照它们从左到右出现的顺序进行计算。**
2. **可以用括号改变优先顺序，强制表达式的某些部分优先运行。**

3. 括号内的运算总是优先于括号外的运算，在括号内，运算符的优先顺序不变。

例如：

设变量 x = 4，y = - 3，a = 6.5，b = - 7.2，求下列表达式的值：

x + y > a + b AND NOT y < b

解：① 先作算术运算 1 > - 0.7 AND NOT y < b

② 再作关系运算 .T. AND NOT .F.

③ 再作逻辑非运算 .T. AND .T.

④ 最后得 .T.

2.4.7 名表达式

在 Visual FoxPro 6.0 中，还允许用户给命令和函数定义一个名字。

将这一名字存入到内存变量和数组元素中，就可以在引用命令和函数时用内存变量和数组元素来代替，这样会给程序开发带来很多便利。

存于内存变量和数组元素中的命令和函数名，用户可以通过间接引用和宏替换这两种方法来使用它们。

间接引用方式，是首先把命令和函数名赋给内存变量和数组元素，然后再取内存变量和数组元素值。

例如：

```
STORE "c:\test\student.dbf" To name
USE (name)
LIST
```

以上命令将打开数据表 student.dbf，并显示出来。

注意：第二条命令中的圆括号不能省略。

宏替换方法是首先把命令和函数名赋给内存变量和数组元素，然后再利用宏替换函数取内存变量和数组元素值。

例如：

```
STORE "?5 3" To ss
? &ss
```

以上命令将在屏幕上输出常数：125。

再例如：

```
x = "Fox"
? "Visual &x. Pro6.0"。
```

以上命令将在屏幕上输出"Visual FoxPro 6.0"。

2.4.8　类与对象运算符

类与对象运算符专门用于实现面向对象的程序设计。有以下两种：

.——点运算符，确定对象与类的关系，以及属性、事件和方法与其对象的从属关系。

::——作用域运算符，用于在子类中调用父类的方法。

§2.5　函　数

对于用户来说，Visual FoxPro 6.0 中的函数和其他程序设计语言的函数没有什么区别，使用函数要有参数，可以从函数得到一个返回值。从程序设计的角度来看，函数是子程序的一种，它能完成一种特定的运算。

2.5.1　函数的分类

Visual FoxPro 6.0 的函数有两种：一种是用户自定义的函数；另一种是系统函数。自定义函数由用户根据需要自行编写，系统函数则是由 Visual FoxPro 6.0 提供的内部函数，用户在需要时可随时调用。

Visual FoxPro 6.0 提供的系统函数大约有 380 多个，主要分为：数值函数、字符处理函数、表和数据库函数、日期时间函数、类型转换函数、调试函数、菜单函数、窗口函数、数组函数、SQL 查询函数、位运算函数、对象特征函数、文件管理函数以及系统调用函数等 14 类。通过查阅“帮助”中的“语言参考”可以了解到函数参数的类型、函数返回值的类型以及函数的使用方法。

2.5.2　函数的要素

函数的一般形式为：

函数名（[参数 1]［,参数 2]…）

函数有函数名、参数和函数值 3 个要素。

1. 函数名起标识的作用。
2. 参数是自变量，一般是表达式，写在括号内。
3. 函数运算后返回一个值，称为函数值。函数值因参数值而异。

有的函数缺省参数，但仍有返回值，如函数 DATE() 能返回系统当前日期。

2.5.3 函数的类型

所谓函数的类型就是函数值的类型。在表达式中嵌入函数时须了解函数值的类型,以免发生数据类型不一致的错误。

使用 TYPE 函数能返回表达式的类型,也能测出函数的类型。

例如:

```
? type ("date ()")          && 显示 D,表明函数 Date () 是日期型函数
x = 21
? Type ("x")                && 显示 N,表明变量 x 是数值型变量
```

注意:括号内的双引号不能省略。

2.5.4 常用函数

Visual FoxPro 6.0 提供了大量的系统函数供编程人员使用,下面仅列出一些常用的函数。

1. 数学函数 (见表 2-19)

表 2-19 数学函数

函 数	功 能	示 例
ABS (表达式)	求表达式的绝对值	? ABS (- 3) &&3
INT (表达式)	返回表达式的整数部分	? int (- 7.8) && - 7
EXP (表达式)	e 指数函数	? EXP (2) &&7.39
LOG (表达式)	求表达式的自然对数	? LOG (4) &&1.39
RAND (表达式)	产生 0 ~ 1 之间的随机数	? RAND () &&0.85
SQRT (表达式)	求平方根	? SQRT (9) &&3
SIGN (表达式)	求表达式的正负符号	? SIGN (- 9) && - 1
PI ()	求圆周率 π	? PI () &&3.14
MOD (表达式 1,表达式 2)	求表达式 1 除以表达式 2 的余数	? MOD (13,5) &&3
ROUND (表达式 1,表达式 2)	按表达式 2 指定的小数位数求表达式 1 四舍五入后的值	? ROUND (3.1514,3) &&3.152
MAX (表达式 1,表达式 2,…)	求各表达式中的最大值	? MAX (3,4,5) &&
SIN (表达式)	正弦函数	? SIN (5) && - 0.96

2. 字符串函数（见表 2-20）

表 2-20　字符串函数

函　数	功　能	示　例
ASC（表达式）	求字符串表达式中最左边一个字符的 ASCII 码值	?ASC（"abcd"）　&&97
CHR（数值表达式）	返回表达式表示的 ASCII 码	?CHR（65）　&&"A"
LEN（字符表达式）	求字符串表达式长度	?LEN（"计算机"）　&&6
LEFT（字符表达式，n）	从字符串表达式的左边取长度为 n 的子串	?LEFT（"程序设计"，4）&&"程序"
RIGHT（字符表达式，n）	从字符串表达式的右边取长度为 n 的子串	?RIGHT（"程序设计"，4）&&"设计"
UPPER（表达式）	将字符串表达式中的小写字母转换为大写字母，其余不变	?UPPER（"8hg2ab"）&&"8HG2AB"
LOWER（表达式）	将字符串表达式中的大写字母转换为小写字母，其余不变	?LOWER（"*RETY?"）&&"*rety?"
ALLTRIM（表达式）	去掉字符串表达式的前导空格和尾随空格	?ALLTRIM（"　ab　c　"）&&"ab　c"
SPACE（n）	返回 n 个空格	?SPACE（3）+"a"　&&"　a"
AT（表达式 1 表达式 2，n）	返回表达式 1 在表达式 2 中第 n 次出现的开始位置（n 缺省为 1）	?AT（"123"，"abc123"）&&4
SUBSTR（表达式，n［，m］）	从字符串表达式中提取从 n 开始的 m 个字符的子串，若省略 m，则取 n 开始的所有字符	?SUBSTR（"abcdefg"，3，2）&&"cd"
VAL（数字字符表达式）	将数字字符串转换成数值	?VAL（"123"）　&&123.00

3. 日期处理函数（见表 2-21）

表 2-21　日期函数

函　数	功　能	示　例
DATE（）	返回系统当前日期	?DATE（）　&&05/17/02
TIME（）	返回系统当前时间	?TIME（）　&&15：39：18
DATETIME（）	返回系统当前日期和时间	?DATETIME（）&&05/17/02 03：39：18 PM
DOW（表达式）	取日期表达式的星期号（1 为星期天）	?DOW（DATE（））　&&6，当天为星期五
YEAR（表达式）	取日期表达式的年份值	?YEAR（{2002-05-17}）　&&2002
MONTH（表达式）	取日期表达式的月份值	?MONTH（{2002-05-17}）　&&5

续 表

函 数	功 能	示 例
DAY (表达式)	取日期表达式在月份中的天数值	? DAY ({ 2002 - 05 - 17}) &&17
HOUR (表达式)	取时间表达式中的小时数	? HOUR ({2002 - 5 - 17 10 :20 :30}) &&10
MINUTE (表达式)	取时间表达式中的分钟数	? MINUTE ({ 2002 - 5 - 17 10 :20 :30}) &&20
SEC (表达式)	取时间表达式中的秒数	? SEC ({2002 - 5 - 17 10 :20 :30}) &&30
CTOD (表达式)	将字符串表达式转换为日期	? CTOD ("10 /1 /02") &&10 /01 /02
DTOC (表达式)	将日期表达式转换为字符	? DTOC ({ 2002 /3 /24}) &&"03 /24 /02"
DTOS (表达式)	将日期表达式转换为 YYYYMMDD 格式字符串	? DTOS ({ 2002 /3 /24}) &&"20020324"

§2.6 数 组

利用数组,可以灵活地组织和使用数据。在许多场合,使用数组可以缩短和简化程序。

2.6.1 数组的概念

数组是一种特殊的内存变量,它是用一个统一的名称表示的、由一系列数据值组成的有序列。每一个数据称为一个元素,可以用数组名及下标来惟一地标识一个数组元素,因此数组元素又称下标变量。

可以用数组名及下标惟一地标识一个数组元素,如 B (3) 表示名称为 A 的数组中下标为 3 的那个数组元素。Visual FoxPro 6.0 系统中可以有不同类型的元素,即在同一个数组里可以包含有任何一种类型的数据,这些数据可以是数值型、字符型、日期型,也可以是逻辑型等。

使用数组时要注意以下几点:

1. 数组的命名规则与简单变量的命名规则相同。

2. 下标必须用括号括起来,不能把 A (3) 写成 A3,A3 是一个简单变量。

3. 下标可以是常量、变量或表达式,还可以是数组元素。如 C (B (2)),若 B (2) =9,则 C (B (2)) 就是 C (9)。

4. 下标若不为整数,则会被自动取整(舍去小数部分),如 A(3.8)将被视为 A(3)。

5. 数组的下标从 1 开始。

2.6.2　数组的维数

在 Visual FoxPro 6.0 中,允许定义一维数组或二维数组。

如果一个数组的元素只有一个下标,则这个数组为一维数组。例如,数组 M 有 10 个元素:M(1)、M(2)、M(3)、…、M(10),分别用来保存 10 位同学的一门课程的成绩,则数组 M 为一维数组。一维数组中的各个元素又称为单下标变量。

如果要保存 10 位同学的 3 门课的成绩,则必须使用两个下标,如第 i 个学生的第 j 门课的成绩可以使用 M(i,j)来表示。其中 i 为行下标,用来表示学生号;j 为列下标,用来表示课程号。像这样有两个下标的数组称为二维数组,二维数组元素又称为双下标变量。

若定义了一个二维数组 N(4,5),则该二维数组共有 $4 \times 5 = 20$ 个元素,表示了一个 4 行 5 列的表格。该数组各元素的排列见表 2-22。

表 2-22　二维数组 N(4,5)各元素排列

N(1,1)	N(1,2)	N(1,3)	N(1,4)	N(1,5)
N(2,1)	N(2,2)	N(2,3)	N(2,4)	N(2,5)
N(3,1)	N(3,2)	N(3,3)	N(3,4)	N(3,5)
N(4,1)	N(4,2)	N(4,3)	N(4,4)	N(4,5)

2.6.3　数组的定义

数组必须遵循先定义后使用的原则。定义数组的语法格式如下:

格式:

DIMENSION | DECLARE < 数组名 1 >(< 数字表达式 1 > [,< 数字表达式 2 >]) [,< 数组名 2 > (< 数字表达式 3 > [,< 数字表达式 4 >]) …]

说明:

DIMENSION 和 DECLARE 可以选择其中一个来定义,二者是等价的。

可以同时定义若干个一维或二维数组。当只选择 < 数组名 1 > 和 < 数字表达式 1 > 时,定义的是一维数组。例如,Dimension X(20),Y(30),表示

定义了两个一维数组,一个数组名为 X,另一个数组名为 Y。

当选择 < 数字表达式 2 > 时,表示定义的是二维数组。例如,Declare C (2,3),D (3,4),表示定义了两个二维数组,一个数组名为 C,另一个数组名为 D。

在一条定义语句中,可以同时定义一维数组和二维数组。不管是一维数组还是二维数组,都是内存变量。

定义数组时,可以使用方括号代替圆括号,即 Declare A (3) 和 Declare a [3] 是等价的,都是合法的命令。

由于 Visual FoxPro 6.0 中,同一数组可以存放不同类型的数据,因此,数组定义时不必指定数组的类型。

执行该命令后,所建立的数组中的所有元素系统都将初始化为逻辑值".F.",但是其值随着以后赋给数组元素的数据类型的变化而变化。

例如:

```
Dimension S (4),T (5,6)
```

功能:建立有 4 个元素的一维数组 S 和建立有 5 行 6 列数据的二维数组 T,并将所有数组元素初始化为.F.。

我们还可以从如下程序中清楚地看到数组的定义和简单使用。

源程序清单:

```
Set talk off                    && 关闭人机对话
Clear                           && 清除屏幕上显示的内容
Clear memory                    && 清除所有的内存变量
Dimension  m (3,4) , n (4)      && 定义了一个二维数组 M 和一个一维数
                                   组 N
M (2,3) = 123                   && 把整数 123 赋值给数组元素 m (2,3)
? M (2,3)                       && 在屏幕上显示数组元素 m (2,3) 的值,
                                   显示为 123
m (2,3) = "abc"                 && 把字符常量"abc" 赋值给数组元素 m
                                   (2,3)
? M (2,3)                       && 在屏幕上显示数组元素 m (2,3) 的值,
                                   显示为"abc"
? N (2)                         && 在屏幕上显示数组元素 n (2) 的值,显示
                                   为 .F.
Display memory                  && 查看内存变量的值
Cancel                          && 程序终止运行,并返回到命令窗口
```

运行结果如图 2-7 所示：

图 2-7

程序运行后，在屏幕上出现如图 2-7 所示的内容。

从以上的运行结果中可以看出：

① 一个数组只用一个内存变量。

② 数组必须先定义后使用。

③ 数组元素的类型不是一成不变的。

④ 对于没有赋值的数组元素，系统都将初始化为逻辑值“. F. ”。

定义数组还可以和 LOCAL、PRIVATE 和 PUBLIC 这三个用来指定内存变量作用域的关键字结合起来。其格式如下：

格式：

PUBLIC | PRIVATE | LOCAL　[ARRAY] < 数组名 1 >（< 数字表达式 1 > [,< 数字表达式 2 >]）[,< 数组名 2 >（< 数字表达式 3 >

[,< 数字表达式4 >]）…]

说明：

关键字 ARRAY 是可选的。

例如：

Public R(3,4),E(5)

功能：定义两个全局变量数组，一个是二维数组 R，另一个是一维数组 E，并把所有的数组元素初始化为.F.。

全局变量数组在整个 Visual FoxPro 6.0 工作期中可以被任何程序访问。若使用 LOCAL 关键字，则表示定义的数组只能在创建它们的程序中使用，不能被更高层或更低层程序访问；使用 PRIBVATE 关键字表示定义私有数组，其含义和简单内存变量前使用 PRIVATE 含义相同。

2.6.4 数组的赋值

数组在定义之后，每个元素被默认地赋予逻辑值.F.。

对数组或单个数组元素赋值可用下面两种方式：

STORE <表达式> TO <数组名>

<数组名> = <表达式>

例如：

Store "abcd" To A(2,3)

或

A(2,3) = "abcd"

两者是等价的，都是将字符串"abcd"赋给数组 A 的第二行第三列元素。

又如：

Store 123 To A

或

A = 123

两者也是等价的，功能是把数组 A 的所有元素全部初始化为整数 123。

2.6.5 数组的使用

1. 重新定义数组的维数

重新执行 DIMENSION 命令可以改变数组的维数和大小。数组的大小可以增加或减少，一维数组可以重新定义为二维数组，二维数组可以重新定义为一维数组。

若重新定义后的数组只是元素的个数增加了，则将原数组中所有元素的

内容复制到重新定义过的数组中,增加的数组元素初始化为逻辑值.F.。

2. 数组变量的释放

使用 RELEASE 命令可以从内存中释放变量和数组。格式如下：

RELEASE <变量列表> | <数组列表>

其中各变量或数组名之间用逗号隔开。

二维数组的双下标变量可以用一维数组的单下标变量来表示。

对于二维数组 A(4,5),其各个下标变量的存储顺序为：A(1,1)、A(1,2)、A(1,3)、A(1,4)、A(1,5)、A(2,1)、…、A(4,4)、A(4,5),双下标变量 A(2,3)是其中的第 8 个元素,因此,双下标变量 A(2,3)也可以用单下标变量 A(8)来表示。

对于二维数组 A(m,n),若用一维数组来表示,则其元素 A(i,j)在一维数组中对应的下标可以通过以下公式来计算：

下标 = (i - 1) * n + j

或使用 AELEMENT()函数也能获得一维数组表示法中的元素下标,即

下标 = AELEMENT(数组名,行数 m,列数 n)

3. 数组与内存变量

数组是一种特殊的内存变量。一个数组只用一个内存变量,但是数组中的第一个元素可以当做一个内存变量来使用,这个内存变量称为下标变量。

第3章 项目管理器

项目是文件、数据、文档以及 Visual FoxPro 对象的集合，项目文件以 .pjx 扩展名来保存。我们可以使用“项目管理器”组织和管理项目中的文件。当激活“项目管理器”窗口时，Visual FoxPro 在菜单栏中显示“项目”菜单。本章主要介绍项目的创建及其基本使用。

§3.1 创建一个项目

3.1.1 新建项目

我们现在就以创建工资管理系统为例，创建一个项目“gzgl.pjx”用于管理文件。

方法一：利用 Visual FoxPro 系统主菜单。

其操作步骤为：

在 Visual FoxPro 系统主菜单下，单击“文件”菜单中的“新建”。

选择“项目”，然后选择“新文件”按钮。

在创建对话框中（见图 3-1）输入项目的名称“gzgl”，并选择保存项目的目录，再单击“保存”。

这样我们就建立好了一个项目“gzgl”，如图 3-2 所示。从图中我们可以看到，在“项目管理器”中，以类似于大纲的形式组织各项，可以展开或折叠它们。在项目中，如果某类型数据项有一个或多个数据项，则在其标志前有一个加号。单击标志前的加号可查看此项的列表，单击减号可折叠展开的列表。

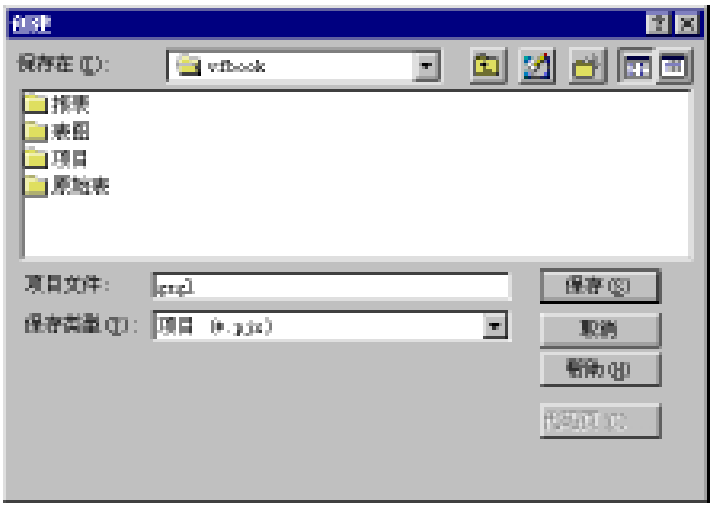


图 3-1　创建对话框

图 3-2　“项目管理器”

方法二：利用命令方式，其步骤如下：

在命令窗口输入：CREATE PROJECT

其后面的操作同方法一。

3.1.2 打开和修改已存在的项目

方法一：在 Visual FoxPro 系统主菜单下，单击“文件”菜单中的“打开”，在打开对话框中输入项目名，在文件类型中选择“pjx”。

方法二：利用命令方式：modify project。

方法三：利用资源管理器，找到该文件，双击它。

打开项目的方法还有很多，我们不再一一叙述。

3.1.3 关闭项目

在 Visual FoxPro 系统主菜单下，单击“文件”菜单中的“关闭”，或者单击其窗口右上角的关闭按钮。

与工具栏类似，您可以将“项目管理器”拖动到屏幕顶部，或双击标题栏，从而停放“项目管理器”。“项目管理器”停放后，被自动折叠，只显示选项卡（如图 3-3 所示）。

图 3-3 选项卡

§3.2 项目管理器的使用

在选项卡（图 3-3）中，分类显示各数据项。当“项目管理器”折叠时，您可以从“项目管理器”中拖下选项卡。如果要重新放置一个选项卡，只需将其拖回原来的位置或单击“关闭”框；也可单击图钉按钮，使选项卡保持在屏幕上。下面我们来逐一进行介绍。

3.2.1 窗口选项

1. 展开/折叠按钮

选项卡的最右边是展开/折叠按钮，用来展开和折叠“项目管理器”。当“项目管理器”折叠时，把鼠标指针放到选项卡上，并将其从“项目管理器”拖走，可以拖下选项卡。

2. 项列表(图 3-4)

以大纲方式列出包含在项目中的项。项左边的图标用来区分项的类型。

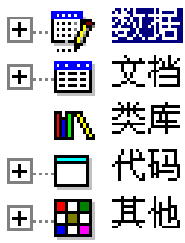


图 3-4　项列表

当文件包含在项目中时,编译后将形成单个 .app 文件。所有的包含文件在运行时都是只读的,在“程序”、“表单”、“查询”或“菜单”组中,主程序文件用黑体表示。

3.2.2 “项目管理器”按钮

1. 新建

创建一个新文件或对象。此按钮与“项目”菜单的“新建文件”命令作用相同。新文件或对象的类型与当前选定项的类型相同。

应强调一点,从“文件”菜单中创建的文件不会自动包含在项目中。使用“项目”菜单的“新建文件”命令 (或“项目管理器”上的“新建”按钮) 创建的文件自动包含在项目中。

2. 添加

把已有的文件添加到项目中。此按钮与“项目”菜单的“添加文件”命令作用相同。

3. 修改

在合适的设计器中打开选定项。此按钮与“项目”菜单的“修改文件”命令作用相同。

4. 浏览

在“浏览”窗口中打开一个表。此按钮与“项目”菜单的“浏览文件”命令作用相同,且仅当选定一个表时可用。

5. 关闭

关闭一个打开的数据库。此按钮与“项目”菜单的“关闭文件”命令作用相同,且仅当选定一个表时可用。如果选定的数据库已关闭,此按钮变为“打开”。

6. 打开

打开一个数据库。此按钮与“项目”菜单的“打开文件”命令作用相同,并且仅当选定一个表时可用。如果选定的数据库已打开,此按钮变为“关闭”。

7. 移去

从项目中移去选定文件或对象。Visual FoxPro 会询问您是仅从项目中移去此文件,还是同时将其从磁盘中删除。此按钮与“项目”菜单的“移去文件”命令作用相同。

8. 连编

连编一个项目或应用程序,还可以连编可执行文件或自动服务程序 (Automation Server)。此按钮与“项目”菜单的“连编”命令作用相同。

9. 预览

在打印预览方式下显示选定的报表或标签。当选定“项目管理器”中的一个报表或标签时可用。此按钮与“项目”菜单的“预览文件”命令作用相同。

10. 运行

执行选定的查询、表单或程序。当选定项目管理器中的一个查询、表单或程序时可用。此按钮与“项目”菜单的“运行文件”命令作用相同。

3.2.3 “项目”菜单

“项目”菜单包含创建和修改项目的菜单项,下面我们来介绍一下:

1. 新建文件

显示“新建文件”对话框,在“项目管理器”中选定的文件类型基础上,您可以创建新文件。当您在“项目管理器”中选定一个文件或一种文件类型时,该命令可用。“新建文件”命令与在“项目管理器”中选择“新建”按钮效果相同。

2. 添加文件

显示“打开”对话框,从中可以向项目添加一个现有文件。该命令与在“项目管理器”中选择“添加”按钮效果相同。当在“项目管理器”中选定一个

要添加的文件类型时,该命令可用。“打开”对话框中显示的文件以及文件扩展名反映出您的选择。

3. 移去文件

从项目中移去所选定的文件,或者移去并从硬盘上删除该文件。“移去”命令把文件从项目中删除,但不从硬盘上删除。“删除”命令把文件从项目及硬盘中同时删除。“移去文件”命令与“项目管理器”中的“移去”按钮效果相同。

4. 修改文件

打开设计器或编辑窗口,从而修改文件。当在“项目管理器”中选定一个数据库、自由表、查询、表单、标签、类库、程序、菜单或文本文件时,该命令可用。“修改文件”命令与“项目管理器”中的“修改”按钮效果相同。

5. 打开文件

打开选定的数据库。当打开“项目管理器”然后选定一个数据库时,该命令可用。它与“项目管理器”中的“打开”按钮效果相同。数据库打开之后,该命令即变成“关闭文件”。

6. 关闭文件

关闭一个数据库。当打开“项目管理器”,并选定一个已经打开的数据库时,该命令可用。它与“项目管理器”中的“关闭”按钮效果相同。数据库关闭之后,该命令即变成“打开文件”。

7. 运行文件

运行选定的查询、表单或程序,在“项目管理器”中选定查询、表单或程序时,此命令可用。它与“项目管理器”中的“运行”按钮效果相同。

8. 浏览文件

在“浏览”窗口中显示所选定的表,查看和编辑其内容。“浏览”命令与“项目管理器”中的“浏览”按钮效果相同。

9. 预览文件

用“打印预览”(文件菜单)方式显示所选定的报表或标签。在“项目管理器”中选定报表或标签时,此命令可用。“打印预览”命令与“项目管理器”中的“预览”按钮效果相同。

10. 重命名文件

显示“重命名文件”对话框,从中可以重命名选定的文件。当在“项目管

理器”中选定某个文件时，该命令可用。

11. 包含

在一个项目中包含以前排除在外的文件。仅当选定一个排除文件时，该命令可用。所有的包含文件都以只读方式被编译进 . app 或 . exe 文件中。如果希望在运行时写入一个文件，应把它们标记为排除文件。

12. 排除

在一个项目中排除以前包含在内的文件。排除文件并不编译进应用程序。仅当您在“项目管理器”中选定一个包含文件时，该命令可用。“项目管理器”在排除的文件名称前显示一个中间带斜杠的圆。排除文件列在“项目管理器”中供您参考，但如果应用程序需要它们时，您必须人工发布它们。

13. 设置主文件

把所选定的程序或表单指定为系统主程序，该程序在编译后的应用程序中执行。必须在“项目管理器”中选择一个程序、表单或菜单，此命令才可用。在选定主程序之后，每当在“项目管理器”中选择该主程序时，此命令的旁边将显示一个对号。

14. 编辑说明

显示“说明”对话框。可在框中修改文本，该文本作为文件说明出现在“项目管理器”窗口的底部。

15. 将项目加到源代码管理器中

显示源代码管理器的一个对话框，它允许您把打开的项目添加到源代码管理器中。如果在“选项”对话框内“项目”选项卡上选定了一个源代码管理器，该菜单选项出现。

16. 源代码管理

显示“源代码管理”子菜单，它允许您在 Visual FoxPro 中访问与源代码管理有关的对话框。仅当打开一个源代码管理项目时，这个“项目”菜单选项可用。

17. 项目信息

显示“项目信息”对话框，从中可以查看和编辑有关项目及其文件的信息。快捷键：CTRL + J。

18. 错误

显示编译应用程序期间生成的错误日志文件。

19. 连编

显示“连编选项”对话框，从而可以创建自定义应用程序并更新已有的项目。

20. 刷新

刷新项目。

21. 清理项目

通过运行 PACK 命令删除带有删除标记的文件，来减小项目（.pjx）文件的大小。当使用“移去文件”命令从项目中移去一个文件时，相关记录仍保留在 .pjx项目文件中，但带有删除标记。如果移去几个文件，可以选择“清理项目”来删除带有删除标记的记录。

3.2.4 “项目管理器”快捷菜单

项目管理器可提供多种快捷菜单，其包含了“项目管理器”中许多有用命令的快捷方式。用鼠标右按钮单击“项目管理器”的不同位置，显示在快捷菜单上的内容也会不同。下面我们来逐一介绍。

1. 全部展开

展开“项目管理器”中全部折叠的文件组，您可以看到文件组中的全部文件。该命令只有通过快捷菜单才能在“项目管理器”中使用。

2. 排除

排除项目中的一个文件，被排除的文件不被编译进应用程序。只有在“项目管理器”中选中一个包含的文件时，该项才被激活。在“项目管理器”内，被排除的文件显示时，文件名前标有一带斜杠的圆，这些文件列在“项目管理器”窗口内供您参考，但若应用程序需要它们，则必须手工来分配。该命令对应于“项目”菜单中的“排除”命令。

3. 包含

将被排除的文件包含进项目。只有在选中被排除文件时，该项才被激活。所有的包含文件都编译进 .app 或 .exe 文件内，且使其变为只读文件。若想在运行期间写文件，应将其标志为排除文件。该命令对应于“项目”菜单中的“包含”命令。

4. 设置主文件

将选中的程序、菜单、查询或表单指定为应用程序的主程序。要激活该命令，须在“项目管理器”中选中某个程序、表单或菜单。不管什么时候在“项目管理器”中选中了主程序，此命令都会在这个主程序旁显示一个复选标志，并

且主程序名在“项目管理器”中用粗体显示。此命令对应于“项目”菜单中的“设置主文件”命令。

5. 重命名

显示“重命名文件”对话框。在该对话框内,可以修改选中文件的文件名。在“项目管理器”中,如果有选中的文件,则该命令有效。此命令对应于“项目”菜单的“重命名文件”命令。

6. 编辑说明

显示“说明”对话框,从中可以对“项目管理器”窗口底部的内容进行修改,该内容用来作为文件的描述。此命令对应于“项目”菜单的“编辑说明”命令。

7. 项目信息

显示“项目信息”对话框,从中可以查看和编辑项目和文件的有关信息。在“项目管理器”窗口选中文件时,该命令被激活。此命令对应于“项目”菜单的“项目信息”命令。

8. 代码页

为跨平台数据共享显示“代码页”对话框。若要启用此命令,须在“项目管理器”选中文本文件、查询、图标或位图文件名。

9. 显示说明/路径

显示或隐蔽“项目管理器”底部的描述信息和路径信息。用鼠标右按钮单击“项目管理器”的“说明/路径”区域,就会出现此命令。只有用鼠标右按钮单击“说明/路径”区域或“项目管理器”标题栏,才可以在“项目管理器”中通过快捷菜单访问此命令。

10. 移动

利用箭头键移动“项目管理器”窗口。只有用鼠标右按钮单击“说明/路径”区域或“项目管理器”标题栏,才可以在“项目管理器”窗口使用快捷菜单访问该命令。

11. 大小

利用箭头键调整“项目管理器”的大小。只有用鼠标右按钮单击“说明/路径”区域或“项目管理器”标题栏,才可以在“项目管理器”窗口使用快捷菜单访问该命令。

12. 关闭

关闭“项目管理器”窗口。只有用鼠标右按钮单击“说明/路径”区域或

“项目管理器”标题栏,才可以在“项目管理器”窗口访问该命令。此命令对应“文件”菜单的“关闭”命令。

13. 拖走

取消停放模式的“项目管理器”。只有用鼠标右按钮单击停放的“项目管理器”,才可以在“项目管理器”窗口使用快捷菜单访问该命令。

14. 生成器

运行“应用程序生成器”。

15. 连编

显示“连编选项”对话框,从中可以建立一个用户应用程序,或者对现有的项目进行更新。只有“项目管理器”处于停放模式下,才可以使用快捷菜单访问该命令。当取消停放模式的“项目管理器”时,此命令对应“项目”菜单的“连编”命令,或者对应“项目管理器”窗口内的“连编”按钮。

16. 帮助

显示某选定项的“帮助”信息。

表(Table)是 Visual FoxPro 中处理数据和建立关系型数据库及应用程序的基本单元。涉及表的操作包括处理当前存储于表中的信息、定制已有的表、或者创建自定义的表来存储数据。可以使用索引对数据进行排序及加快处理。数据库(database)文件具有 .dbc 扩展名,可以包含一个或多个表、视图、到远程数据源的连接和存储过程。通过本章的学习您将了解如何设计表和使用表,以及如何用 Visual FoxPro 创建数据库和数据库的使用。在设计表时应确保表的结构符合应用程序的要求,表中数据类型与索引的选择是决定应用程序设计成功与否的重要因素,以及如何用 Visual FoxPro 创建数据库和数据库的使用。

§4.1 创建数据表

4.1.1 设计表结构

在前面的学习中我们了解到 Visual FoxPro 是基于关系代数的数据模型。表 4-1 是某校职工基本情况表,也就是我们平常所说的二维表。数据表和二维表之间存在一一对应的关系。在这张表中我们可以把它分成两部分,一部分是表头,我们在 Visual FoxPro 中称之为表结构。另一部分就是表的内容。从这张表中可以看到:每一行代表了一个职工的全部的信息,我们把这一行的数据称之为记录。每一列代表了各个职工的某一项信息,是不可分割的信

息最小单位,我们把这一列的信息称之为字段。字段是指包含在记录中的数据项。一个或多个字段组成表中的一个记录,一个或多个记录构成一个表。在这张表当中我们可以知道它共有 8 条记录、9 个字段。

表 4-1　职工基本情况表

编号	姓名	性别	出生年月	职称	部门代号	党员否	简历	照片
100001	赵一同	男	01/23/68	讲师	1	.T.	Memo	Gen
100002	王小小	女	08/12/75	助教	4	.F.	Memo	Gen
100003	周玖月	女	09/19/60	副教授	1	.T.	Memo	Gen
100004	吴建国	男	07/09/50	教授	2	.T.	Memo	Gen
100005	孙颖颖	女	11/23/70	讲师	5	.F.	Memo	Gen
100006	肖丽丽	女	06/01/65	副教授	2	.F.	Memo	Gen
100007	陈若无	男	02/12/66	副教授	1	.T.	Memo	Gen
100008	万维妙	女	06/24/81	助教	3	.T.	Memo	Gen

在 Visual FoxPro 中,由于日常的二维表就是一张数据表,也就是我们所说的表文件,因此要建立这样的一张表首先就要建立表的结构。

建立数据表结构,就是要定义表当中的字段个数、字段名、字段类型和字段宽度以及是否建立索引等。

字段名是用来标识字段的,它的命名规则和变量的命名规则相似,以字母或汉字开头,后面可以跟字母、汉字和数字以及下划线。不能够在字段名称中使用 .、"、/、h、[、]、:、|、<、>、+、=、;、*、?和空格。字段名的长度要小于等于 10。

字段类型反映的是字段的数据类型,表中的每一个字段都有特定的数据类型。可以将字段的数据类型设置为表 4-2 中的任意一种。字段的数据类型应与将要存储在其中的信息类型相匹配。在设计中,如编号、学号等看上去是数字,但是为了操作方便,我们常常把它定义成字符型,因为组成编号或学号的数字都包含了许多信息,通过前面介绍的字符型函数,能够很快查询到我们所需要的信息。

字段宽度对于日期型、逻辑型是固定不变的,分别是 8、1。备注型和通用型宽度为 4。备注型字段主要用于存放较长的字符型数据,这里的宽度 4 是用来存放数据指针的,因为备注型数据是存放在和表文件名同名的备注文件当中,其扩展名为.fpt,只要表文件当中包含备注型字段,该文件就会自动生成,并且该文件会随表的打开而自动打开。要注意的是如果备注文件不小心被删除或丢失,那么表文件也就打不开了。

通用型字段用来存放图像、电子表格等多媒体数据，其数据也是存放在和表文件名同名的备注文件当中。字段的宽度要足够容纳将要显示的信息内容。不过考虑到存储空间，在定义宽度时，字符型、数值型等应选择能够存储该数据的最小宽度，一个字符占 1 个字节，一个汉字占 2 个字节。

对于有小数的数值型和浮动型字段，应为“数值型”或“浮点型”字段设置正确的小数位数。注意：小数点本身也占一位。对于纯小数，其小数位数应比其字段宽度小 1。

如果想让字段接受 NULL 值，请选中 NULL (null value)，它代表无明确的值。NULL 值不等同于零或空格。一个 NULL 值不能认为比某个值 (包括另一个 NULL 值) 大或小、相等或不同。

表 4-2 列出了系统中字段的数据类型和字段宽度。

表 4-2 数据类型说明

数据类型	代号	说 明	字段宽度	数 据 范 围
字符型	C	字母、数字型文本	1 个字符占 1 个字节	小于等于 254 个字符
货币型	Y	货币单位	8 个字节	- 922337203685477. 5807 to 922337203685477. 5807
数值型	N	整数或小数	8 个字节， 小于等于 20 位	- . 9999999999E + 19 to. 9999999999E + 20
浮动型	N	同“数值型”		
日期型	D	年，月，日	8 个字节	{0001 - 01 - 01} to {9999 - 12 - 31}
日期 时间型	T	年，月，日，时，分，秒	8 个字节	{0001 - 01 - 01} to {9999 - 12 - 31} plus00 00 00a. m. to11 59 59p. m.
双精度型	N	双精度数值	8 个字节	+ / - 4. 94065645841247E - 324 to + / - 8. 9884656743115E307
整型	N	不带小数点的数值	8 个字节	- 2147483647 to 2147483647
逻辑型	L	真或假	1 个字节	真或假，用. T. 和. F. 来表示
备注型	M	较长的字母数字文本	4 个字节	只受存储空间限制
通用型	G	OLE 对象(链接与嵌入)	4 个字节	只受存储空间限制
字符型 (二进制)	C	同前述“字符型”相同，但当代码更改时字符值不变		保存在表中的用户密码，用于不同国家/地区
备注型 (二进制)	M	同前述“备注型”相同，但当代码页更改时备注不变		用于不同国家/地区的登录脚本

根据以上原则,我们可以把上述的表结构定义成如下结构：

字段	字段名	类　型	宽度	小数位
1	编号	字符型	6	
2	姓名	字符型	8	
3	性别	字符型	2	
4	出生年月	日期型	8	
5	职称	字符型	6	
6	部门代号	字符型	1	
7	党员否	逻辑型	1	
8	简历	备注型	4	
9	照片	通用型	4	

4.1.2　创建表结构

打开“项目管理器”gzgl 如图 4-1,在数据选项卡中选定“自由表”,然后单击“新建”按钮。弹出“新建表”对话框,建立新表包括利用“表向导”及直接使用“表设计器”这两种方法。

图 4-1　新建表

方法一：利用表设计器。

在“新建表”窗口中，单击“新建表”按钮，出现如图 4-2 所示的“创建”窗口，要我们指出该表保存的位置，我们在这里把它保存在“d：\vfbook”当中，文件名命名为“JBQK”，表的扩展名为“.dbf”，然后单击“保存”按钮。随即又弹出如图 4-3 所示的“表设计器”窗口。

图 4-2 “创建”窗口

图 4-3 “表设计器”窗口

表设计器共有三个选项卡，其中字段选项卡在可滚动表格中显示表字段。索引选项卡包含用于定义索引的可滚动表格。表选项卡显示有关表的信息，允许指定触发器和记录级规则。

选择“表设计器”的“字段”选项卡

在“字段名”区域键入第一个字段的名称“编号”。

在“类型”区域中，选择列表中的某一字段类型“字符型”。

在“宽度”列中，设置以字符为单位的列宽“6”。

注意：

如果“类型”是“数值型”或“浮点型”，请设置“小数位数”框中的小数点位数。

如果希望为字段添加索引，请在“索引”列中选择一种排序方式。索引方法请参见索引这一章。

如果想让字段接受 null 值，选中“NULL”。

依次输入各个字段的名称、类型、宽度，如图 4-4 所示，然后单击“确定”按钮。屏幕显示如图 4-5 所示的窗口，此时可以选择“是”立即开始输入记录，选择“否”退出表结构设计，在此之后再打开表进行输入。

图 4-4　依次输入各个字段的数据

方法二：利用表向导。

“表向导”是基于典型的表结构创建表。“表向导”允许从样表中选择满足需要的表。在一步步经过向导的过程中，可以定制表的结构和字段。也可以在向导保存表之后修改表。

图 4-5　询问窗口

如果有一个或多个数据库打开,表向导自动将新建表添加到当前数据库中。如果没有数据库打开,表向导将创建一个自由表。

若要运行表向导,可在如图 4-1 所示的情况下,单击“表向导”。也可以单击“工具”菜单内“向导”子菜单,选择“表”命令。出现“表向导”窗口,如图 4-6 所示。

图 4-6　“表向导”窗口

如果想要向“样表”中添加自己的表,选择“加入”,出现“打开”窗口(见图 4-7),把我们刚刚建立的表“JBQK. DBF”加入到样表中。其操作步骤为:

步骤 1　字段选取

单击“样表”中的一个表“JBQK. DBF”,显示可以加入表中的字段,我们也可以在多个表中间选择字段。在表向导中,可以通过以下四个按钮选择字段:

从可用字段中选取某一个字段到选定字段。

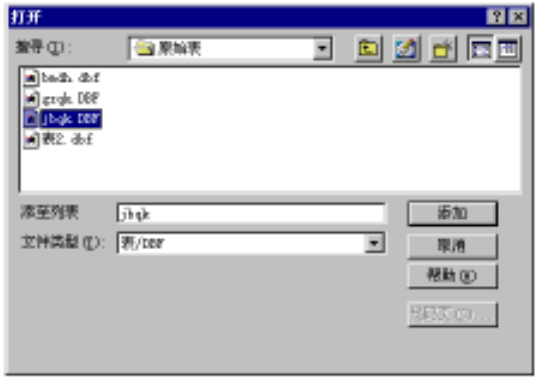


图 4-7　打开窗口

从可用字段中选取所有字段到选定字段。

从选定字段删除某一字段。

从选定字段删除所有字段。

我们选取样表中的编号、出生年月、职称来构成另一张表的结构（见图 4-8）。单击“下一步”。

图 4-8　选取字段

表向导的步骤 1　选择数据库

用来说明是创建自由表还是数据库中的表。如果是创建数据库中的表，“表向导”将在下一步提供自动的和自定义的格式选项。也可以为新表指定一个友好的或具有说明性的名称。如果创建的是自由表，则不能更改字段标题。我们选择“创建独立的自由表”按钮（见图 4-9），然后单击“下一步”按钮，进入“表向导”的步骤 2。

步骤 2　修改字段设置

图 4-9　创建独立的自由表

在这一步中，我们可以改变正在创建的表字段的设置。例如，您可以改变一个字段的最大宽度。可以指定字段是否包含 Null 值，或者更改字段标题或类型。如果在数据库中创建表，也可以为每个字段类型选择预定义数据输入掩码，或者创建自定义输入掩码。“格式”窗口显示所选择的掩码和格式。如果创建的是自由表，则不能更改字段标题（见图 4-10）。修改完后单击“下一步”按钮。

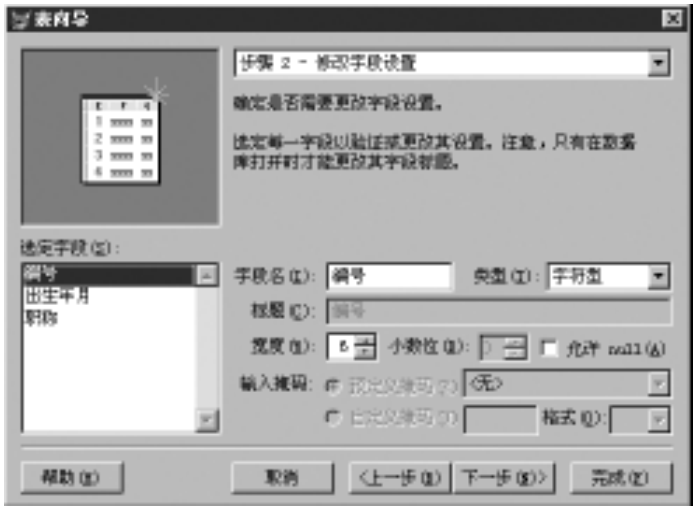


图 4-10　修改字段设置

步骤 3　为表建索引

选择在新表中成为主索引关键字的字段。同时可以指定其他字段为候选索引关键字 (见图 4-11)。

图 4-11　创建成索引

步骤 3a　建立关系

如果创建的是数据库中的表，可以在新表的字段和数据库中已有表的字段之间建立关系。

突出显示要建立关系的字段，然后单击“关系”按钮。

若要定义关系，可执行以下操作：

- 单击表示关系类型的单选按钮。
- 选择与数据库中的表相关的字段。如果选择 < newfield > ，请输入字段名。
- 单击“确定”。
- 关闭“关系”对话框，并返回步骤 3，此处的列表中会显示出新的关系。

步骤 4　完成

在如图 4-12 所示界面中，选择“保存表以备将来使用”选项，单击“完成”，出现如图 4-13 所示对话框，保存表后，以后可以像其他表一样在“表设计器”中打开或修改它。

图 4-12　完成窗口

图 4-13

请注意：这时命令窗口将会出现一条命令“CREAT**”，这就是建立新表的** VFP **命令，将光标移至这一行，按** Enter **键，将再次弹出“表设计器”对话框。** VFP **的大部分菜单操作都会将相应的命令显示在“命令”窗口中，请您可千万别错过这个学习命令的好机会。**

在不打开数据库的情况下，使用 CREATE TABLE **命令。**

例如，以下代码可以创建一个名为 smalltbl **的自由表，该表只有一个名为** name **的字段：**

```
CLOSE DATABASES
CREATE TABLE smalltbl FREE (name c (50))
```

其格式为：

```
CREATE TABLE | DBF TableName1 [NAME LongTableName] [FREE]
(FieldName1 FieldType [(nFieldWidth [, nPrecision])]
[NULL | NOT NULL]
```

用同样的方法我们建立其他两个表：

工资情况表：gzqk. dbf

表结构：

字段名	类型	宽度	小数位
编号	字符型	6	
姓名	字符型	8	
基本工资	数值型	7	2
地区补贴	数值型	7	2
其他	数值型	7	2

水电费	数值型	7	2
公积金	数值型	7	2

部门代号表：bmdh. dbf

表结构：

字段名	类型	宽度	小数位
部门代号	字符型	1	
部门名称	字符型	12	

§4.2 表的基本操作

4.2.1 添加记录

设计并创建了表的结构之后，就可以在表中添加新记录以存储数据，随后，可以更改或删除已有的记录，这些任务中的每一个都可以通过界面或命令来完成。

1. 创建表时立即添加记录

在图 4-5 中，单击“是”，即在定义好了表结构之后，立即可以进入如图 4-14所示的界面，进行数据输入。

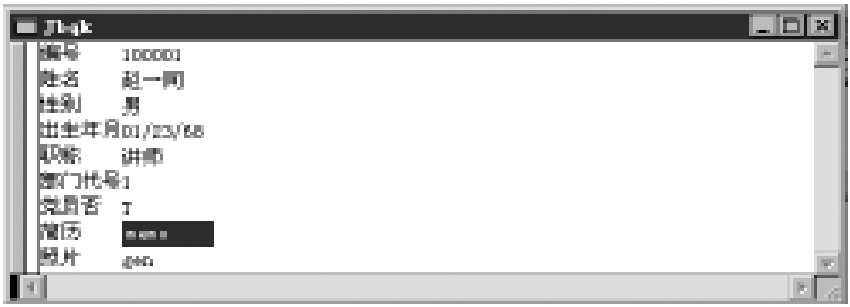


图 4-14 数据输入

表中数据输入时，我们在编辑框中逐一输入各个字段的值。数值型和字符型数据原样输入，逻辑型数据只能接受 T、Y、F、N 这四个字母中的一个（不区分大小写）。日期型数据必须与其相对应的日期格式相匹配。系统本身默认格式为美国格式 mm/dd/yy。如果要设置中国格式 yy. mm. dd，只需在命令窗口键入命令 SET DATE ANSI。如果要回到美国格式，其命令为 SET DATE

AMERCAN。

若要输入备注型字段和通用型字段的数据，可把光标放置在备注型或通用型字段的 memo 或 gen 所在位置，用鼠标双击或用键盘命令 Ctrl + PgDn 打开相应的字段编辑窗口，如图 4-15 所示。备注型数据可以直接输入，通用型字段可通过系统编辑菜单中的“插入对象”。输入完后，保存数据的方法是用鼠标单击该窗口的关闭按钮，或利用键盘命令存盘 Ctrl + W，放弃存盘用 Ctrl + Q。

图 4-15　备注型数据输入

如果某记录中备注型字段或通用型字段非空时，其字段标志的第一个字母将大写，即显示为 Memo 或 Gen。

依次输入 8 条记录，其结果如图 4-16 所示。

Jbqk

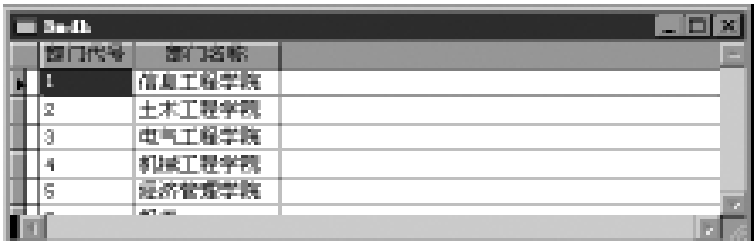
编号	姓名	性别	出生年月	职称	部门代号	党员否	简历	照片
100001	赵一同	男	01/23/68	讲师	1	T	Memo	Gen
100002	王小小	女	08/12/75	助教	4	F	Memo	Gen
100003	周玖月	女	09/19/60	副教授	1	T	Memo	Gen
100004	吴建国	男	07/09/50	教授	2	T	Memo	Gen
100005	孙颖颖	女	11/23/70	讲师	5	F	Memo	Gen
100006	肖丽丽	女	06/01/65	副教授	2	F	Memo	Gen
100007	陈若无	男	02/12/66	副教授	1	T	Memo	Gen
100008	万维妙	女	06/24/81	助教	3	T	Memo	Gen

图 4-16　8 条记录

同时把 gzqk. dbf 和 bmdh. dbf 两个表的内容输入计算机中（如图 4-17、4-18所示）。

编号	姓名	基本工资	地区补贴	其它	水电费	公积金
100001	赵一同	446.00	223.00	440.00	12.30	52.00
100002	王小小	345.00	187.00	580.00	3.50	30.00
100003	周明月	530.00	321.00	780.00	32.80	85.00
100004	吴建国	643.00	366.00	934.00	62.50	107.00
100005	孙晓晓	488.00	321.00	510.00	69.20	84.00
100006	肖强强	594.00	402.00	720.00	59.60	90.00
100007	陈若无	564.00	342.00	795.00	48.50	86.00
100008	万雅妙	311.00	150.00	420.00	10.60	25.00

图 4-17　gzqk. dbf 表内容



部门代号	部门名称
1	信息工程学院
2	土木工程学院
3	电气工程学院
4	机械工程学院
5	经济管理学院

图 4-18　bmdh. dbf 表内容

2. 在已经存在的表中添加记录

如果表已经存在，要往表中添加记录，首先必须打开表，然后进行操作。其操作步骤为：

- 在“项目管理器”中选择表的名称。
- 选择“浏览”。
- 从“显示”菜单中选择“追加方式”。
- 在“浏览”窗口中输入新的记录。

如果愿意在一条分隔线上查看每个字段，可以从“显示”菜单中选择编辑来切换到编辑方式。

4.2.2　修改表结构

在数据处理过程中，我们会发现表结构在设计时可能不是那么准确，需要修改表结构。利用“表设计器”，可以改变已有表的结构，如增加或删除字段，设置字段的数据类型及宽度，查看表的内容以及设置索引来排序表的内容。

如果正在进行修改的表是数据库的一部分，那么还可以得到额外的与数据库有关的字段和表的属性。这些属性的使用将来在“将表加入数据库”中

进行介绍。

若要修改表,其操作步骤为：

在“项目管理器”中选定表名,然后选择“修改”。

表的结构将显示在如图 4-19 所示的“表设计器”中。

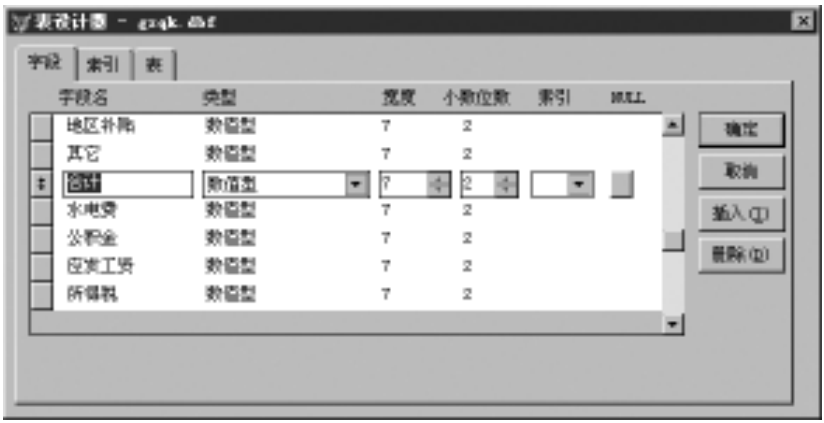


图 4-19　修改表结构

也可在 Visual FoxPro 系统环境下,通过“文件”菜单,打开一个已经存在的表。然后单击“显示”菜单下的“表设计器”按钮。由于我们生成的是自由表,因此此选项卡仅包括基本的字段名、类型和格式选项。

选项卡在可滚动表格内显示表字段,每一行包括字段名、数据类型、字符宽度、小数位数、索引,并支持 null 值。单击表格内的单元格可以修改字段。

在字段选项卡选项中,我们逐一介绍它们的用途：

1. 移动按钮

这是位于该行最左侧的双向箭头按钮,用户输入两或三行后,使用此移动钮可以在列表内上下移动某一行。

2. 在表中增加字段

- 在“表设计器”中选择“插入”。
- 在“字段名”列中,键入新的字段名。
- 在“类型”列中,选择字段的数据类型。
- 在“宽度”列中,设置或输入字段宽度。

如果使用的数据类型为“数值型”或“浮点型”,还需要设置“小数位数”列的小数位数。

如果想让表接受“null”值,请选中“NULL”列。

选择“确定”,弹出如图 4-20 所示对话框。

选择“是”,改变表的结构。

3. 删除表中的字段

选定该字段,并选择“删除”。

我们在如图 4-19 所示的位置增加“合计”、“应发工资”、“所得税”、“实发工资”四个字段。出现如图 4-20 所示对话框,单击“是”按钮,表结构就修改好了。

图 4-20 询问窗口

4.2.3 数据的显示和修改

通过前面的学习,表的基本结构和数据已输入完毕。用户可以通过“表设计器”来查看表的结构,也可以利用表“浏览”或表“编辑”两种方式对表中的数据进行输入。在实际操作中,我们也可以通过这两种方式来显示和修改表中记录。

1. “浏览”窗口中的表

表以行和列的格式存储数据,类似于电子表格。每一行代表一个记录,每一列代表记录中的字段。查看表内容的最快方法是使用“浏览”窗口。“浏览”窗口中显示的内容是由一系列可以滚动的行和列组成的。

若要浏览一个表 bmdh. dbf:

从“文件”菜单中选择“打开”,选定想要查看的表名。

从“显示”菜单中选择“浏览”,其结果如图 4-18 所示。也可以从“项目管理器”中选择表的名称,然后选择“浏览”按钮。

2. “ 编辑”窗口中的表

我们可以看到为方便输入,可以把“浏览”窗口设置为“编辑”方式,其结果如图 4-21 所示。在“编辑”方式下,列名显示在窗口的左边。若要将“浏览”窗口改为编辑方式,我们可以从“显示”菜单中选择“编辑”。

编辑方式下的表 ,在任何一种方式下,都可以滚动记录,查找指定的记

录,以及直接修改表的内容。

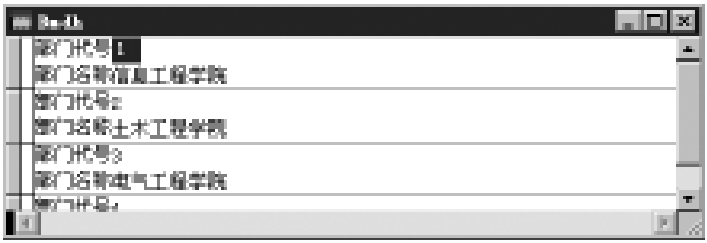


图 4-21　浏览窗口

使用滚动条可以来回移动表,显示表中不同的字段和记录。也可以用箭头键和 Tab 键进行移动。

3. 定制“浏览”窗口

若要按照不同的需求定制“浏览”窗口,可以重新安排列的位置、改变列的宽度、显示或隐藏表格线或把“浏览”窗口分为两个窗格。

(1) 重新安排列

可以重新安排“浏览”窗口中的列,使它们按照需要的顺序进行排列,但这并不影响表的实际结构。

我们以“jbqk. dbf”为例,在“浏览”窗口中重新安排列。

其操作过程是:

打开表“jbqk. dbf”,进入表“浏览”窗口。

在表“浏览”窗口,将列标头拖到新的位置,如图 4-22 所示。

或者也可以从“表”菜单中选择“移动字段”,然后用上下箭头键移动列,最后按 Enter 键。

图 4-22　重新安排列

⑵ 改变列的宽度

可以在“浏览”窗口中改变列的宽度。这种尺寸调整不会影响到字段的长度或表的结构。

如果您想改变字段的实际长度，应使用“表设计器”修改表的结构（参阅本章稍后的“修改表”部分）。

我们以“jbqk. dbf”为例，改变列宽。

打开表“jbqk. dbf”，进入表“浏览”窗口。

在表“浏览”窗口，在列标头中，将鼠标指针指向两个字段之间的结合点，拖动鼠标调整列的宽度，如图 4-23 所示。

也可以先选定一个字段，然后从“表”菜单中选择“调整字段大小”，并用左、右箭头键来调整列宽，最后按 Enter 键。

图 4-23 调整列的宽度

⑶ 打开或关闭网格线

在“浏览”窗口，我们看到记录与记录之间、字段与字段之间都有网格线，我们可以隐藏“浏览”窗口中的网格线。

我们以“jbqk. dbf”为例，隐藏网格线。

打开表“jbqk. dbf”，进入表“浏览”窗口。

在表“浏览”窗口，从“显示”菜单中选择“网格线”，网格线前面的钩就去掉了，其结果如图 4-24 所示。用同样的方法我们也可以给记录加上网格线。

图 4-24　隐藏网格线

⑷ 拆分“浏览”窗口

通过拆分“浏览”窗口，可以很方便地查看同一表中的两个不同区域，或者同时在“浏览”和“编辑”方式下查看同一记录。

我们以“jbqk. dbf”为例，拆分“浏览”窗口。

打开表“jbqk. dbf”，进入表“浏览”窗口。

在表“浏览”窗口，将鼠标指针指向窗口左下角的拆分条（黑色的小竖条）。

向右方拖动拆分条，将“浏览”窗口分成两个窗格，其结果如图 4-25 所示。

图 4-25　拆分“浏览”

或者，从“表”菜单中选择“调整分区大小”。光标变成双箭头型。左右箭头键移动拆分条。

若要调整拆分窗格的大小，将指针指向拆分条。向左或向右拖动拆分条，改变窗格的相对大小。

也可以从“表”菜单中选择“调整分区大小”。按左箭头或右箭头键移动

拆分条，再按 Enter 键。

默认情况下，“浏览”窗口的两个窗格是相互链接的，即在一个窗格中选择了不同的记录，这种选择会反映到另一个窗格中。取消“表”菜单中“链接分区”的选中状态，可以中断两个窗格之间的联系，使它们的功能相对独立。这时，滚动某一个窗格时，不会影响到另一个窗格中的显示内容。

在图 4-25 中，两个窗口都是以浏览的方式来显示记录的，我们也可以用不同的方式来显示记录，选中其中的右窗口，单击“显示”菜单下的“编辑”，其结果如图 4-26 所示。

图 4-26 不同的方式显示记录

4. 编辑字段

若要改变“字符型”字段、“数值型”字段、“逻辑型”字段、“日期型”字段或“日期时间型”字段中的信息，可以把光标设在字段中并编辑信息，或者选定整个字段并键入新的信息。

若要编辑“备注型”字段，可在“浏览”窗口中双击该字段或按下 Ctrl + PgDn。这时会打开一个“编辑”窗口，其中显示了“备注型”字段的内容。

“通用型”字段包含一个嵌入或链接的 OLE 对象。通过双击“浏览”窗口中的“通用型”字段，可以编辑这个对象，您可以直接编辑文档（如 Microsoft Word 文档或 Microsoft Excel 工作表），也可以双击对象打开其父类应用程序（如 Microsoft 画笔对象）。

5. 在表中快速加入新记录

可以将“浏览”和“编辑”窗口设置为“追加方式”。在“追加方式”中，文件底部显示了一组空字段，您可以在其中填入来建立新记录。

在新记录中填充字段，用 Tab 键可以在字段间进行切换。每完成一条记

录,在文件的底端就会又出现一条待输入的新记录。

4.2.4　查看不同的记录

当我们向表输入数据时,系统本身会按其输入的先后顺序,给每一个记录提供一个数字,用于区别其记录输入的先后,我们称这个数字为记录号。输入的第一条记录为 1 号记录,输入的第 2 条记录为 2 号记录,依此类推。

在 Visual FoxPro 系统内部,有一个记录指针,用以确定当前正在操作的记录,记录指针指向哪一条记录,这条记录就是当前记录,一般说来,我们的操作就是针对当前记录。当表一打开,记录指针就指向第一条记录。

在实际操作过程中,我们会在活动表或视图中查找匹配指定条件的记录。当您想要在一索引字段或未索引字段上逐条记录地搜索一个表或视图,或者想要在一个表中搜索多条记录时,就是要确定哪一条是当前记录。

我们以"jbqk. dbf"为例,要确定当前记录,其操作步骤为:

打开表"jbqk. dbf",进入表"浏览"窗口。

在表"浏览"窗口,打开"表"菜单,选择"转到记录"子菜单,这个菜单包含把记录指针定位到指定记录选项。

选择"转到记录"子菜单命令,这里包含了多个选项,用于把指针指向相应的记录:

第一个	把指针放置在表或视图的第一个记录上。
最后一个	把指针放置在表或视图的最后一个记录上。
下一个	把指针放置在紧跟当前记录之后的记录上。
上一个	把指针放置在当前记录的上一个记录上。
记录号	显示"转到记录"对话框,从中可以指定一个记录号。
定位	显示"定位记录"对话框,从中可以查找表或视图中的索引字段或未索引字段,或者查找多个记录。

我们通过具体操作来说明其用法:

1. 确定最后一条记录为当前记录

打开表"jbqk. dbf",进入表"浏览"窗口。

在表"浏览"窗口,打开"表"菜单,选择"转到记录"子菜单,选择"最后一个"选项,指针自动移到最后一条记录,其结果如图 4-27 所示。

图 4-27 确定最后一条记录为当前记录

2. 确定第四条记录为当前记录

打开表“jbqk. dbf”,进入表“浏览”窗口。

在表“浏览”窗口,打开“表”菜单,选择“转到记录”子菜单,选择“记录号”选项,出现“转到记录”对话框,如图 4-28 所示。指定记录指针到底放在哪条记录上。在“记录号”框中键入记录编号 4,或者视情况使用向上或向下箭头直到要求的编号出现在“记录号”框中。选择“确定”回到“浏览”窗口,这时,记录指针就放到了指定记录上,其结果如图 4-29 所示。

图 4-28 “转到记录”对话框

图 4-29 运行结果

3. 确定第一个性别为女的记录为当前记录

打开表“jbqk. dbf”,进入表“浏览”窗口。

在表“浏览”窗口,打开“表”菜单,选择“转到记录”子菜单,选择“定位”选项,出现定位记录对话框,如图 4-30 所示。

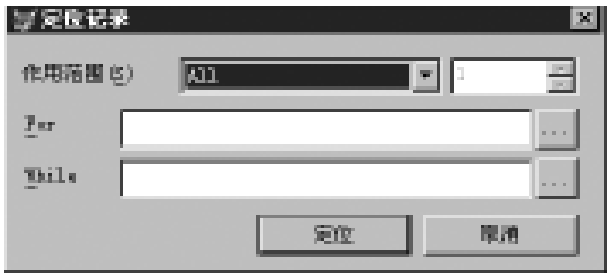


图 4-30　定位记录对话框

在该对话框中,有几个框用于我们进行条件输入:

作用范围下拉框用于指定 Visual FoxPro 要操作的记录。从下拉列表中选择“All”、“Next”、“Record”或者“Rest”。若从“作用范围”下拉列表中选择了“Next”或者“Record”,需从右边的微调控制项内再选择一个记录数。

For 文本框用于显示“表达式生成器”对话框,可在其中创建一个逻辑表达式,每条被命令影响的记录必须满足该表达式。

While 文本框用于显示“表达式生成器”对话框。While 表达式指定仅当该逻辑表达式为“真”时,操作影响记录。一旦表达式取值为“假”,操作即停止而不考虑其余记录。

定位按钮用于使用“作用域”、“For”和“While”值确定后,执行命令。

我们单击“For”文本框右边的按钮,出现表达式生成器对话框,如图 4-31 所示,在“表达式生成器”对话框中,允许创建并编辑表达式。一个表达式可以简单得像一个字段名,也可以像一个包括 IF 函数,级连和数据类型转换的计算一样复杂。“表达式生成器”的主要目标是通过提供方法中每一步骤的合适选项的列表使创建表达式更容易。该对话框可从设计器、窗口、生成器和向导中访问。

若要创建表达式,可直接在表达式框中键入,或者从对话框中的函数下拉列表中选取,并将其粘贴到表达式框中。

图 4-31 “表达式生成器”对话框

在表达式中处理字符串时，表 4-3 以下函数非常有用：

表 4-3 常用函数

操 作	使用以下函数
从字符表达式中删除前导空格和后继空格	ALLTRIM()
删除前导空格	LTRIM()
删除后继空格	RTRIM()
在字符串的左侧、右侧或两侧添加指定字符	PADL(),PADR(),PADC()
在比较中只处理部分字符串	SUBSTR()
使用从字符串的左侧开始指定数目的字符	LEFT()
使用从字符串的右侧开始指定数目的字符	RIGHT()
切换大小写字母	UPPER(),LOWER()
将字符串转换成大写	PROPER()
将一个数值型字段转换成一个字符串	STR()

在对话框选择项中，在函数下拉列框，包含四种函数类型的列表框。当从四种类型的某一种中选择一个函数时，Visual FoxPro 自动将其粘贴至表达式框内。在建立远程视图的表达式时，Visual FoxPro 仅列出特定于目标后台数

据库的函数。

- 字符串下拉列框,列出可用的字符串函数。
- 逻辑下拉列框,列出可用的逻辑函数。
- 数学下拉列框,列出可用的数学函数。
- 日期下拉列框,列出可用的日期和时间函数。
- 表达式滚动条,显示正在创建或编辑的表达式。

字段滚动条,列出当前表或视图中的可用字段。若要将字段粘贴入“表达式”框,既可双击该字段也可选定该字段并按 Enter 键。若要显示不同表中的字段,应在“来源于表”框中选定另外的表。

来源于表下拉列框,列出打开的表和视图。可以选择表或视图来更新“字段”框。

变量滚动条,列出系统内存变量、数组和已创建的内存变量。

若要将一个变量粘贴入“表达式”框,既可双击该变量也可选定该变量并按 Enter 键。

检验按钮,如果相应的表已打开,检查表达式框中表达式的语法。如果表达式是有效的,“表达式有效”显示于状态栏中。如果表达式是无效的或是相应的表没有打开,Visual FoxPro 显示一条错误信息。该选项不可用于远程视图。注意,如果用户在表达式中包括一个用户自定义的函数调用,则“检验”将指示一条错误,但在运行中表达式求值时不必指示错误。

选项按钮,显示“表达式生成器选项”对话框,可在其中设置“表达式生成器”的参数。

要确定第一个性别为女的记录为当前记录,我们在表达式生成器对话框中,在字段选项中选择“性别”,在“逻辑”选项中我们选择“ = ”,在字符串中我们选择“文本”,然后在字符串边界符内输入“女”,然后单击“确定”按钮,出现如图 4-32 所示对话框,然后单击“定位”,其结果如图 4-33 所示。

图 4-32　定位记录对话框

图 4-33 第一个性别为女的记录

4.2.5 删除和恢复表中记录

1. 逻辑删除表中的记录

在 Visual FoxPro 中,删除表中的记录共有两个步骤。首先是单击要删除记录的左边的小方框,标记要删除的记录,我们称之为逻辑删除。这种删除并没有真正从磁盘中把数据删除掉,只是加上了一个删除标志。

我们以“jbqk. dbf”为例,逻辑删除表中非党员的数据,其操作步骤为:

打开表“jbqk. dbf”,进入表“浏览”窗口。

方法一:把指针移到要进行逻辑删除的记录上,对准其删除标志栏单击鼠标左键,标志栏中就会出现一个黑色的小块,这就是逻辑删除的标志。其结果如图 4-34 所示。

图 4-34 逻辑删除的标志

方法二:打开表“jbqk. dbf”,进入表“浏览”窗口,打开“表”菜单,选择“删

除记录”，进入“删除”窗口，如图 4-35 所示。

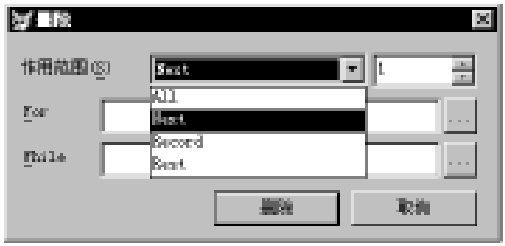


图 4-35　“删除”窗口

在“作用范围”对话框，指定 Visual FoxPro 对当前表或视图中的哪些记录起作用。从下拉列表中选择“All”、“Next”、“Record”或者“Rest”。若从“作用范围”下拉列表中选择了“Next”或“Record”，须再从右边的微调控制项内选择一个记录数。其中：

All 代表全部，可对表或视图中的全部记录起作用。

Next 代表后续，可对某一范围的记录起作用，作用范围从当前记录开始，持续指定数目的记录。键入 1 将只对当前记录起作用。

Record 代表记录号，可对指定记录号的记录起作用。

Rest 代表其余，可对某一范围的记录起作用，作用范围从当前记录开始，到表或视图的最后一条记录为止。

For 和 While 都代表条件，都将显示“表达式生成器”对话框，其作用和前面介绍的用法一样，这里不再重复。

我们在作用范围内选择“All”，然后单击“For”条件右边的按钮，在表达式生成器对话框中，在“逻辑”列表框中选择“NOT”，在字段选项中选择“党员否”，单击“确定”，出现如图 4-36 所示对话框 再单击“删除”，其结果如图 4-34 所示。

图 4-36　加上删除条件

2. 恢复表中带有删除标志的记录

带有删除标志的记录,系统默认情况下,都参与表中的操作,也就是说系统本身并没有删除记录。值得注意的是,如果我们在命令窗口执行了 Set Deleted On 命令,带有删除标志的记录将不参与任何操作。

在实际操作过程中,我们也可以给加上删除标志的记录去掉删除标志,即可对其进行恢复记录操作。

打开表“jbqk. dbf”,进入表“浏览”窗口。

方法一:把指针移到要进行恢复逻辑删除的记录上,对准其删除标志栏单击鼠标左键,标志栏中黑色的小块 (这就是逻辑删除的标志) 就消失了。

方法二:打开表“jbqk. dbf”,进入表“浏览”窗口,打开“表”菜单,选择“恢复记录”,进入“恢复记录”窗口,如图 4-37 所示。

图 4-37 “恢复记录”窗口

在“作用范围”对话框,指定 Visual FoxPro 对当前表或视图中的哪些记录起作用。在条件框输入条件,然后,单击“恢复记录”,就可以把满足条件的记录恢复了,这里就不再具体举例了。

3. 删除带有删除标记的记录

有删除标记记录并不等于删除记录。要想真正地删除记录,应从“表”菜单中选择“彻底删除”。这个过程将删除所有标记过的记录,并重新构造表中余下的记录。我们把这种真正从磁盘中删除记录称之为物理删除。应该注意的是要想物理删除记录,必须首先给这个记录做上删除标志。删除做过删除标记的记录时会将表关掉,因此若要继续工作,必须重新打开该表。

若要从表中删除记录,从“表”菜单中选择“彻底删除”。出现如图 4-38 所示对话框,当出现提示,问您是否想从表中移去已删除的记录,选择“是”,其结果如图 4-39 所示。

图 4-38　“彻底删除”窗口

图 4-39　删除的记录后的表

4.2.6　控制字段和记录的访问

在对表的操作过程中,有些情况下,我们不一定要对所有记录或者所有字段进行访问,在 Visual FoxPro 系统下,我们可以通过过滤器来限制记录的操作范围或者字段的个数。

1. 限制对字段的访问

打开表“jbqk. dbf”,进入表“浏览”窗口。从“表”菜单上选中“属性”时,会出现如图 4-40 所示对话框。

在对话框选项中:

临时表:用来显示当前打开的表的标题。

允许数据缓冲:表示缓冲区设置为打开。buffer (缓冲区) 是指在内存中供信息临时存储的一片区域。Visual FoxPro 以两种锁定方式提供缓冲:保守式缓冲和开放式缓冲。锁定方式决定了何时锁定一个或多个记录,这些记录如何以及何时被释放。

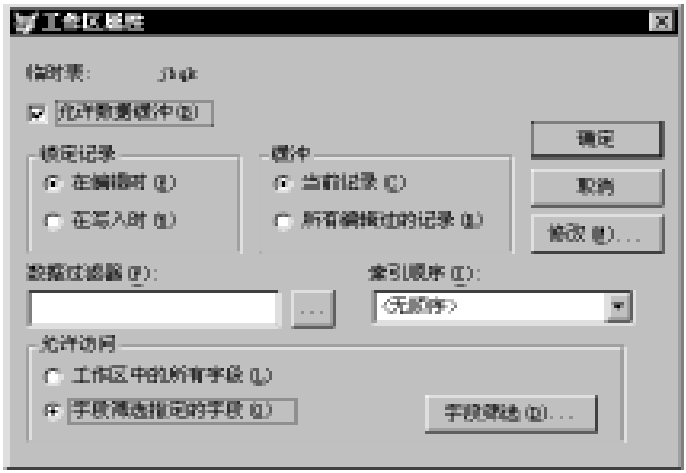


图 4-40 “属性”窗口

锁定记录中的两个选项：

在编辑时：代表当您编辑记录时防止其他用户访问，这也称为保守式缓冲。

在写入时：代表允许其他用户访问，直到记录更新为止，这也称为开放式缓冲。

缓冲中的两个选项如下：

当前记录：代表仅缓冲当前记录，这也称为记录缓冲。

所有编辑过的记录：代表缓冲所有编辑过的记录，这也称为表缓冲。注意，当您选择缓冲“当前记录”时，Multilocks 自动设置为“打开”(如果它们当前是关闭着的话)。但是当您清除缓冲“当前记录”选项或选择缓冲“所有编辑过的记录”时，并不把它设置为“关闭”。可在“选项”对话框中的“数据”选项卡内为新表设置默认值。

数据过滤器：它提供键入数据筛选表达式的文本框。另外，也可以用对话框按钮显示“表达式生成器”对话框，从中可以指定活动表中哪些记录可被处理，对应于 SET FILTER 命令。

索引顺序：指定表如何进行索引排序。在“数据库设计器”内，索引列表中表的主索引旁边显示一个钥匙符号，对应于 SET ORDER 命令。

允许访问中的选项：

工作区中的所有字段：代表指定用户能够访问工作区中的全部字段，对

应于 SET FIELDS 命令。

字段筛选指定的字段：代表指定只有匹配“字段选择器”对话框中选定字段的记录才被显示或能够编辑。

字段筛选：代表显示“字段选择器”对话框，可在其中控制活动字段的数目。

修改按钮：显示“表设计器”，可在其中修改数据库、自由表、字段和索引。

我们在“工作区属性”窗口，在“允许访问”中我们选择“字段筛选指定的字段”，单击“字段筛选”按钮，进入“字段选择器”窗口，如图 4-41 所示。

图 4-41　“字段选择器”窗口

“字段选择器”对话框，主要是限制一个表中活动字段的数目，从而定制哪些显示、哪些不显示。不能访问、检查或修改不活动的字段。表中的数据本身不受影响。除非您在“工作区属性”对话框中选择了“字段筛选指定的字段”，否则忽略此对话框中所做的选择。

在对话框中，您可以选择所有字段，显示当前表中的全部字段；也可以选定某些字段，在“所有字段”框中选定并且用“添加”按钮移入的字段名。

我们从字段中选取“姓名、性别、出生年月、党员否”四个字段加入到“选定字段”中，然后单击“确定”，返回“工作区属性”对话框后，再单击“确定”，结果如图 4-42 所示。

图 4-42　字段筛选后的表

2. 限制对记录的访问

在对记录的访问过程中,我们可以限制记录的使用范围,在某一时间内,只显示我们所需要的数据。

例如我们只想对男性记录进行操作:

打开表"jbqk. dbf",进入表"浏览"窗口。从"表"菜单上选中"属性"时,会出现如图 4-40 所示对话框。

在"工作区属性"窗口,选择"数据过滤器"边上按钮,进入"表达式生成器"窗口,在"表达式生成器"窗口中输入条件(性别 ="男"),单击"确定",返回"工作区属性"窗口(见图 4-43);再单击"确定",出现如图 4-44 所示结果。

图 4-43　数据过滤器

图 4-44　过滤后结果

应提醒大家的是：不管是对字段还是对记录加以限制，都是在某一时间范围内有效的，不具备永久性，更没有把字段或记录从表中删除。

从“选定字段”框中移去全部字段。

暂时筛选数据：

可用 SET FILTER 命令暂时筛选数据，而不用生成专门的筛选索引。若要指定一个暂时的条件，使表中只有满足该条件的记录才能访问时，这个命令特别有用。如果要关闭当前表的筛选条件，可以执行不带表达式的 SET FILTER TO 命令。例如，可以使用以下命令筛选 customer 表，只显示德国 (Germany) 的顾客：

```
USE customer
SET FILTER TO country = "Germany"
BROWSE
```

SET FILTER 命令接受任何一个有效的 Visual FoxPro 逻辑表达式作为筛选条件。如果使用 SET FILTER 命令，在表中只有满足筛选条件的记录才可访问，所有访问表的命令都遵守 SET FILTER 条件。可为每个打开的表设置独立的筛选条件。

§4.3　建立数据库

数据库提供了如下的工作环境：存储一系列的表，在表间建立关系，设置属性和数据有效性规则使相关联的表协同工作。数据库文件保存为带 .dbc 扩展名的文件。

在 Visual FoxPro 中的术语“数据库”和“表”不是同义词。“数据库”(.dbc文件) 指的是关联的数据库，它是一个或多个表 (.dbf 文件) 或视图信

息的容器。

4.3.1 数据库设计

如果您使用一个可靠的数据库设计过程,就能迅速、高效地创建一个设计完善的数据库,为您访问所需信息提供方便。在设计时打好坚实的基础,设计出结构合理的数据库,会节省日后整理数据库所需的时间,并使您能够更快地得到精确结果。下面是设计数据库的步骤:

1. 确定建立数据库的目的

这有助于确定 Visual FoxPro 保存哪些信息。

2. 确定需要的表

在明确了建立数据库的目的之后,就可以着手把信息分成各个独立的主题,例如 Employee 或 Orders 等。每个主题都可以是数据库中的一个表。

3. 确定所需字段

确定在每个表中要保存哪些信息。在表中,每类信息称作一个字段,浏览表时在表中显示为一列。例如,在 Employee 表中,可以有这样两个字段:Last_name 和 Hire_date。

4. 确定关系

分析每个表,确定一个表中的数据和其他表中的数据有何关系。必要时,可在表中加入字段或创建一个新表来明确关系。

5. 设计求精

对设计进一步分析,查找其中的错误。创建表,在表中加入几个示例数据记录,看能否从表中得到想要的结果。需要时可调整设计。

在最初的设计中,不要担心发生错误或遗漏东西。这只是一个初步方案,您可在以后对设计方案进一步完善。在完成初步设计后,可利用示例数据对表单、报表的原型进行测试。Visual FoxPro 很容易在创建数据库时对原设计方案进行修改。可是在数据库中输入了数据或连编表单和报表之后,再要修改这些表就困难得多。正因如此,在连编应用程序之前,应确保设计方案已经考虑得比较全面。

数据库设计完成后,就可通过用户界面或编程语言创建数据库。您可能还希望将已存在的表加入数据库,使它们也能享受到 Visual FoxPro 数据字典提供的各种功能。如果您已经在“项目管理器”中打开了一个项目,新创建的表将自动加入到该项目中。

4.3.2　创建数据库

创建数据库实际上就是将多个表收集到一个集合中，在这里，它们可以享受到数据字典（data dictionary，是包含数据库中所有表信息的一个表。存储在数据字典中的信息称之为元数据，换言之，其记录是关于数据的数据，例如长表名或字段名、有效性规则和触发器，以及有关数据库对象的定义，如视图和命名连接等）的各种功能。它的功能将在第八章讲述。

把表收集入数据库，需创建一个数据库容器以容纳所有与组成数据库的表相关联的对象，如视图、连接和存储过程。下面我们来创建一个新数据库：

方法一：利用“新建”创建数据库。

其操作步骤如下：

在“项目管理器”中，从列表中选择“数据”选项卡，然后选择“数据库”。

选择“新建”按钮，出现如图 4-45 所示对话框。选择“新建数据库”按钮，出现保存文件对话框，选择数据库保存的位置，并给数据库命名为“gzsj”，单击“保存”按钮，将显示一个空的“数据库设计器”窗口，如图 4-46 所示，与此同时，“数据库设计器”工具栏将变为有效。

图 4-45　新建数据库窗口

一个新的数据库创建好之后，里面是空的，没有包含任何相关表或其他对象。

方法二：使用“数据库向导”。

使用“数据库向导”帮助您创建数据库。向导提供模板并提出一系列问题，然后根据您的回答帮助您建立数据库。其操作步骤为：

在“项目管理器”中选择“数据”选项卡，然后选择“数据库”。

选择“新建”。

图 4-46 数据库设计器

选择“数据库向导”按钮。

按照屏幕上向导的指示操作。

也可以从菜单中访问“数据库向导”,方法是:从“文件”菜单中选择“新建”命令,再选择“数据库”,然后选择“向导”选项。

向导提供了表、视图、主关键字以及关系的模板,您可以选取或编辑它们。在这里我们不再介绍它的具体操作,同学们可以自己上机动手一试!

注:在“数据库设计器”中创建一个新的数据库可以使用 CREATE DATABASE 命令。

下面的代码创建并以独占方式打开了一个叫做 Sample 的新数据库。

```
CREATE DATABASE Sample
```

4.3.3 打开数据库

数据库可以单独使用,也可以将它们合并成一个项目,用“项目管理器”进行管理。

当数据库设计器窗口被激活时,将伴随出现“数据库”菜单和“数据库设计器”。

数据库必须在打开后才能访问它内部的表,常用方法如下:

- 可通过项目管理器来打开
- 从系统菜单下的“文件”中选“打开”
- 用下面的代码打开 testdata 数据库

```
OPEN DATABASE testdata
```

4.3.4 关闭数据库

可以使用“项目管理器”或 CLOSE DATABASE 命令关闭一个已打开的数据库。

若要关闭数据库,可以在“项目管理器”中,选定要关闭的数据库并选择“关闭”按钮。

或者在命令窗口使用 CLOSE DATABASE 命令。

§4.4　数据库的使用

Visual FoxPro 中包含有两种表：一种是自由表 (free table),它是指不包含在数据库中的表。所有由 FoxPro 早期版本创建的 . dbf 文件在被加入数据库之前,都是自由表。另一种就是数据库表 (database table),它是指包含在一个数据库中的表。

单独使用表,可以为我们存储和查看信息提供很多帮助。但是如果把若干表组织到一个数据库中,您就可以充分地利用 Visual FoxPro 提供的强大功能。通过把表放入数据库,可减少冗余数据的存储,保护数据的完整性。例如不必对已有的每一个客户订单的客户姓名和地址重复存储。可在一个表中存储用户的姓名和地址并把其关联到存储在另一个表中的订单上。如果客户的地址改变了,只需改变一个记录,可控制字段怎样显示或键入字段中的值。也可添加视图并连接到一个数据库中,用来更新记录或扩充访问远程数据的能力。向数据库中添加表实际上是建立表文件与数据库容器之间的双向链接关系：在数据库中保存指向表文件的前链,在表中保存指向数据库容器的后链。

4.4.1　向数据库添加数据表

建立数据库的第一步是向数据库中添加表,您可以在一个打开的数据库中创建表,或向数据库中添加已有的表,把表和数据库联系起来。您可以选定目前不属于任何数据库的表。因为一个表在同一时间内只能属于一个数据库,所以必须将表先从旧的数据库中移去后才能将它用于新的数据库中。

若要向 gzsj 数据库中添加表 jbqk,其操作步骤为：

从“数据库”菜单中选择“添加表”。

在“打开”对话框中选定表 jbqk,然后选择“确定”。

除用“数据库”菜单的方法外,还可以用：

- 利用项目管理器选中数据库 gzsj,选中“表”,然后单击右边的按钮“添加”把文件添加进数据库。
- 在“数据库设计器”窗口单击鼠标右键,选择“添加表”。

用以上方法把 gzqk. dbf 和 bmdh. dbf 添加至数据库中。其结果如图 4-47 所示。

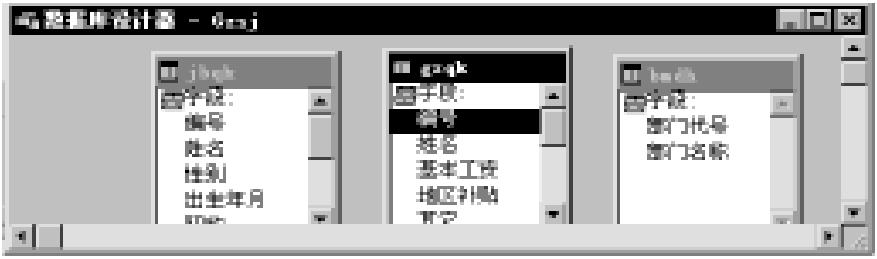


图 4-47 向数据库中添加表

- 或者 -

使用 ADD TABLE 命令。

例如,下面的代码打开 testdata 数据库,并向其中添加 orditems 表:

```
OPEN DATABASE testdata
ADD TABLE orditems
```

只有明确地把一个已有的自由表添加到数据库中,才能使它成为数据库的一部分。即使在打开数据库后,执行 MODIFY STRUCTURE 命令修改自由表的结构,也不能使 Visual FoxPro 把自由表添加到数据库中。

一个表只能加入到一个数据库中。但是,即便不把一个表加到数据库中,您仍可以使用该 .dbf 文件的数据。

4.4.2 若要从数据库中移去表

在"项目管理器"中选定表名,然后选择"移去"按钮。

- 或者 - 从"数据库设计器"中选定表,并从"数据库"菜单中选择"移去"。

- 或者 - 使用 REMOVE TABLE 命令。

例如,下面的代码打开了 testdata 数据库并移去 orditems 表:

```
OPEN DATABASE testdata
REMOVE TABLE orditems
```

从数据库中移去表不能自动删除该表文件。如果想要从数据库中移去表,同时从磁盘上删除该表的 .dbf 文件,请使用 REMOVE TABLE 命令的 DELETE 子句或 DROP TABLE 命令。例如,以下代码打开 testdata 数据库,从盘上删除 orditems 表:

```
OPEN DATABASE testdata
```

REMOVE TABLE orditems DELETE

以下代码也打开 testdata 数据库，从盘上删除 orditems 表，但不在 Windows 回收站中保存副本：

OPEN DATABASE testdata

DROP TABLE orditems NORECYCLE

在“数据库设计器”中调整表的大小，可以看到其中更多（或更少）的字段；也可以折叠视图，只显示表的名称。当数据库中包含多个表时，这一方法十分有用。

4.4.3　若要展开或折叠表

若要展开或折叠一个表，将鼠标指针指向“数据库设计器”中的一个表，单击鼠标右键，选择“展开”或“折叠”。

若要展开或折叠所有表，将鼠标指针指向“数据库设计器”窗口，单击鼠标右键，选择“全部展开”或“全部折叠”。

数据库中折叠后的表，可以更改显示在“数据库设计器”中的表的布局。例如，完成数据库操作后，您也许想让这些表恢复为默认的高度和宽度，或想对齐表以改进布局。

4.4.4　若要重排数据库的表

从“数据库”菜单中选择“重排”，再从“重排表和视图”对话框中选择适当的选项（参见表 4-3）。

在“数据库设计器”中，选中的表标题栏将加亮显示。

表 4-4　重排数据库的表的操作

操　作	选　择
按名称字母顺序重排表	按名称
按类型重排表	按类型
行对齐表	水平置中
列对齐表	垂直置中
将表恢复到原来的大小	调整对象大小到默认的高度和宽度

在本书第四章中，我们介绍了用窗口界面的方法建立、修改库结构，对记录进行追加、修改和删除等，本章我们将介绍用命令方式进行以上操作，同时将介绍索引的建立和使用以及一些常用的表的命令。

§5.1 表操作的常用命令

5.1.1 表文件的结构操作

1. 建表命令格式

格式1：CREATE [路径] [数据表名]

功能：定义表文件的结构，包含表文件名及表的字段名、类型、宽度、小数位等参数的设置。

说明：

省略 [数据表名]，系统会提示请用户输入表名。

省略扩展名，系统默认为. dbf。

库文件名已存在，系统询问用户是否将其覆盖。

当你创建一个表时如果一个表已经打开，表将自动添加至库中。

可直接指定表文件所在路径，省略 [路径] 表文件建立在当前盘当前目录下。

可用 SET DEFAULT TO 盘符设置默认盘。在实际操作过程中，我们对文件存放的位置老是要指明，用此命令，我们能够只设置一次，就能把文件放在

你所想放的位置。

例：SET DEFAULT TO d：\vfbook　把文件存放在 D 盘的 VFBOOK 子目录里。在系统环境下，我们也可以设置，其操作如下：在“工具”下拉菜单中选择“选项”，在“选项”对话框中，选择“文件位置”，在文件位置列表框中选择“默认目录”，单击右下角的“修改”可以根据自己的要求进行修改。

格式 2：CREATE TABLE <表名>（<字段名 1> <字段类型>[(<字段宽度>[,<小数位数>])][,<字段名 2>…..]]

功能：通过命令的执行，生成一个包含指定字段的表文件。

2. 表结构的显示

对于已存在的表文件，可以随时查看其结构的定义，这就要用到显示结构的命令 有时对已有的结构不满意，又可用修改结构的命令，修改字段的名字、类型、长度等属性，或者增删一些字段。

格式 1：DISPLAY STRUCTURE [TO PRINTER | TO FILE <file>]

功能：显示出当前正在使用表文件的结构。

说明：

选项 [TO PRINTER] 将显示结果输出到打印机上。

选项 [TO FILE] 将显示结果输出到文件名为 <file>，其后缀为. txt 的文本文件中。

格式 2：LIST STRUCTURE [TO PRINTER | TO FILE <file>]

其功能与格式 1 相同，只是 LIST 命令不能分屏显示。而 DISPLAY 命令可以分屏显示。

3. 表结构的修改

格式：MODIFY STRUCTURE

功能：修改当前表文件的结构。

说明：

可增减字段，也可改变字段名、字段类型、字段宽度及小数位数。修改完后可以根据 Ctrl + W 存盘，可用 Ctrl + Q 放弃存盘。

4. 表结构的复制

(1) 复制表的结构

如果要建立两个以上的表文件，而当它们的结构（部分）相同或相似时，则可以先建立其中一个表，通过复制命令把它的整个结构或其中的一部分复

制到其他的表。我们就可以在这个新表结构的基础上,只需作少量的修改,而免去重复输入的麻烦。

⑵ 复制表结构的命令

格式:COPY STRUCTURE TO <文件名> [FIELDS <字段列表>]

功能:复制一个同原表结构一样或相似的新的表结构。

说明:

TO 后面跟新表文件名,它不能同原表同名。实际上,文件名前面往往要加上路径,表示文件所存位置。

利用 FIELDS <字段列表>子句,可以指定将表中的一些字段复制到新表当中。新表的字段次序将按照字段名表所列的字段名次序排列。

5.1.2 表的数据录入

除了我们在建立表结构后,系统会弹出对话框询问"是否现在输入记录?"时按 [Y] 我们可以输入数据,数据录入还可以用下列方式进行:

1. 插入命令

格式:INSERT [BLANK]

功能:用来在表中插入记录。

说明:

如果跟 BLANK 子句表示在文件指针所指位置(当前记录)后面追加一条空白记录,等用户以后再输入数据。没有 BLANK 子句会弹出编辑窗口,进入编辑状态,用户可以在窗口输入数据。使用 [BEFORE] 子句,表示在文件指针所指位置(当前记录)前插入新记录。

2. 追加命令

命令 1:APPEND [BLANK]

功能:向已打开的表文件末尾加入新记录。

说明:

如果跟 BLANK 子句表示在文件尾追加一条空白记录,等用户以后再输入数据。没有 BLANK 子句会弹出编辑窗口,进入编辑状态,用户可以在窗口输入数据。

命令 2:INSERT - SQL 命令

INSERT INTO <表名> [(<字段 1> [,<字段 2> [,...]])] VALUES (<表达式 1> [,<表达式> [,...]])

功能：该命令用于将数据以新记录的形式直接添加到表的尾部。

说明：

选项 [< 表名 >] 为要添加数据的表名，该命令在使用之前不必打开。

选项 [(< 字段 1 > [, < 字段 2 > [,. . .]])] 为被追加记录的字段名。

选项 [VALUES (< 表达式 1 > [, < 表达式 2 > [,. . .]])] 为被追加记录的内容。

字段个数与表达式个数要相同，字段类型与表达式类型相同。

命令 3：

APPEND FROM 文件名 [FIELDS 字段名表] [FOR 条件] [[TYPE] [DELIMITED [WITH 定界符 | WITH BLANK | WITH TAB] | SDF | XLS]

功能：在当前表的末尾追加一批记录，这些记录来自一个文件中。

说明：

源文件的类型可以是表，也可以是系统数据格式、定界格式等文本文件，或是 microsoft excel 文件。如果是表文件，命令当中就不能用 TYPE 子句。

若源文件是 EXCEL 文件，TYPE 子句中必须用 XLS 如果源文件是文本文件，TYPE 子句必须用 DELIMITED 或 SDF。

5. 1. 3　表的打开和关闭

1. 打开表命令格式

USE [< 表文件名 >] [IN < expN1 >] [AGAIN] [ALIAS < alias >] [EXCLUSIVE] [SHARE] [NOUPDATE]

功能：在当前工作区内打开已存在的表文件及相关的索引文件。

说明：

如果文件不在当前驱动器的磁盘上，则应在文件名前指明它所在的驱动器名。

选项 [IN < exp N1 >] 指定工作区号，数值表达式 < exp N1 > 的值即为工作区号。

选项 [AGAIN] 在不同的工作区内同时打开一个表。

选项 [ALIAS < alias >] 指定表的别名，若无此选项，表的别名即为原文件名。

选项 [EXCLUSIVE] 及 [SHARE] 指定表打开方式是独占还是共享。

选项 [NOUPDATE] 以非修改方式打开表。

表打开后，与其同名的. FPT 文件自动打开。记录指针同时指向第一条记录。

打开一个表时，该工作区中原来已打开的表自动关闭。

2. 关闭表命令

命令 1：USE [IN <工作区号> | <工作区别名>]

功能：不带任何参数的 USE 命令，可以关闭当前工作区中已打开的表文件和索引文件。在同一工作区中打开另一个表文件，会同时关闭原来打开的表文件。若带 IN 子句，表示在不退出当前工作区的情况下关闭指定了工作区号或工作区别名的那个工作区中的表。

命令 2：

格式 1：CLOSE ALL

关闭所有的表，并选择工作区 1，从内存释放所有内存变量及用户自定义的菜单和窗口，但不释放内存变量。

格式 2：CLOSE DATABASES [ALL] 关闭当前数据库和表，并选择工作区 1，选 ALL 则关闭所有数据库和表。如果无数据库，则关闭的是自由表。

格式 3：CLOSE TABLES [ALL] 关闭当前数据库中的表，并选择工作区 1，选 ALL 则关闭所有数据库中的表。如果无数据库，则关闭的是自由表。

命令 3：QUIT

退出 FOXPRO 时也会关闭所有文件。

5.1.4 显示表的记录

命令 1：DISPLAY [OFF] [<范围>] [FIELDS <字段列表>] [FOR <条件>] [TO PRINTER | TO FILE <文件名>]

功能：在指定范围内显示表文件中的满足条件的记录。

说明：

命令后面没有子句时，则只显示当前记录。

选项 [OFF] 将不显示记录的编号。

选项 [<范围>] 为指定范围内的记录，其选择为：

ALL：所有记录

RECORDn：第 n 个记录

NEXT n：从当前记录开始的 n 个记录

REST：从当前记录开始到文件结束止的所有记录。

选项 [FIELDS < field list >] 指定字段名表。

选项 [FOR < 条件 1 >] 只显示满足条件的记录。

选项 [TO PRINT | TO FILE] 则把显示结果送往打印机或指定文件名的那个文件。

命令 2：LIST [OFF] [<范围>] [FIELDS < field list >] [FOR < 条

件 1 >] [TO PRINTER | TO FILE < file >]

功能：LIST 命令与 DISPLAY 相同。不同之处在于 LIST 后面无任何子句时，将显示所有记录。

命令 3：

BROWSE [FIELDS < 字段名表 >] [FOR < 条件 1 >] [LOCK < expN. >] [NOAPPEND] [NODELETE] [NOEDIT | NOMODIFY] [TITLE < expC >] [VALID [：F] < 条件 >] [ERROR < expC >]

功能：浏览表内容，它具有很强的全屏幕编辑功能。

说明：

选项 [LOCK < expN >] 锁定浏览窗口中左边显示 expN 个字段。

选项 [NOAPPEND] 禁止在编辑窗中追加记录

选项 [NODELETE] 禁止在编辑窗中删除记录。

选项 [NOMODIFY] 禁止对表内容进行修改。

5.1.5 记录指针定位

1. 绝对定位

命令 1：GO | GOTO TOP | BOTTOM [IN < 工作区号 > | < 别名 >]

功能：把记录指针直接定到指定的记录上。

说明：

TOP 把记录指针指向第一条记录。

BOTTOM 把记录指针指向最后一条记录。

IN 子句表示被移动的记录指针不是当前工作区打开的表。

命令 2：[GO | GOTO] < 数值表达式 > [IN < 工作区号 > | < 别名 >]

功能：把记录指针指向记录号等于 < 数值表达式 > 的值的记录。

例：

```
USE JBQK          && 打开数据表
?RECNO()          && 显示记录号，RECNO() 为测试当前记录号函数
1                 && 屏幕显示：1
GO BOTTOM         && 记录指针指向最后一条记录
?RECNO()          && 显示记录号
8                 && 屏幕显示 8
GO 6              && 记录指针指向 6
?RECNO()          && 显示记录号
```

```
6                    && 屏幕显示：6
```

2. **相对定位**

SKIP [< **数值表达式** >] [IN < **工作区号** > | < **别名** >]

功能：从当前记录开始，将记录指针向前或向后移若干条记录。

说明： < **数值表达式** > **的值为移动的记录数，其值为正数前移（文件头方向），为负数后移（文件尾方向）。省略该子句，则约定为** 1，**即** SKIP **等价于** SKIP1。

例：

```
USE JBQK
? RECNO ()           && 显示记录号
1                    && 屏幕显示：1
? BOF ()             &&BOF () 为文件起始函数
. F.                 && 屏幕显示：F.
SKIP - 1             && 记录指针向文件头移动一个记录
?RECNO ()            && 显示记录号
1                    && 屏幕显示：1
? BOF ()
. T.                 && 屏幕显示：T.
GO BOTTOM
? EOF ()             &&EOF () 为文件结束函数
. F.                 && 屏幕显示：F.
SKIP                 && 记录指针向文件尾移动一个记录
? RECNO ()           && 显示记录号
9                    && 屏幕显示：9
? EOF ()
. T.                 && 屏幕显示：T.
GO 5
SKIP - 1
?RECNO ()
4                    && 屏幕显示：4
```

5.1.6 表记录内容的修改

记录修改命令如 EDIT、CHANGE、BROWSE **等，可对已输入的数据记录内容进行修改。**

1. 编辑命令 EDIT

格式：EDIT　[< 范围 >]　[FIELDS　< field list >]　[FOR　< 条件 1 >]　[FREEZE < field name >]　[NOEDIT | NOMODIFY]

功能：对指定的记录进行修改的全屏幕编辑命令。

说明：

选项 [FIELDS < field list >] 可选定任意组合的字段，而且各字段的显示编辑方式还可有如下的选项加以限制：

< fields > [：R] [：V = < 索引关键字 1 > [：F] [：E = < expC1 >]] [：P = < expC2 >] [：B = < 索引关键字 2 >, < 索引关键字 3 > [，F]] [：H = < expC3 >] [：W = < 条件 1 >] [, < field2 > [：R] . . .]

：R　只读不能修改

：V　对字段数据校验

：F　不论字段是否被修改都进行校验

：E　用 < expC1 > 代替系统错误信息" Invalid input"

：P　定义字段的输入格式 < expC2 >

：B　限制字段数据的取值范围

：H　用 < expC3 > 的内容代替缺省的字段名

：W　用 < 条件 1 > 逻辑表达式为假时禁止到该字段

选项 [FREEZE] 限制可以修改的字段。

2. 改换命令 CHANGE

格式：CHANGE　[< 范围 >]　[FIELDS　< field list >] [FOR　< 条件 1 >]　[FREEZE < field name >]　[NOEDIT | NOMODIFY]

功能：同 edit 命令

选项 [FREEZE] 限制可以修改的字段。

3. 浏览命令 BROWSE

在前面我们已经介绍。

4. 替换命令 REPLACE

REPLACE　[[< 范围 >] < 字段名 1 > WITH < 表达式 1 > [ADDITIVE] [, < 字段名 2 > WITH < 表达式 2 > [ADDITIVE]] . . . [FOR < 条件 >] [WHILE　< 条件 >]

功能：用于成批地快速修改满足条件的一批记录。

说明：

能直接用此命令将字段的值用表达式的值来代替，常用于程序设计。

替换时是将 <表达式 1 > 的值替换其前面的 <字段 1 > 原内容,两者的数据类型必须一致。

选项 [ADDITIVE] 在替换备注字段内容时起作用,替换内容将加在备注字段原内容之后,缺省时替换内容取代备注字段原内容。

例:

```
USE GZQK
REPLACE 基本工资 WITH 基本工资 +100        && 对所有职工的基本工资都增加 100
APPEND BLANK
REPLACE 姓名 WITH "张曹",基本工资 WITH 300,地区补贴 with;
200&& 新增人员姓名为张曹,基本工资为 300,地区补贴为 200
```

5.1.7 表记录的删除和恢复

在实际工作中,对不再需要的记录,可用记录删除命令 (如 DELETE) 删除 对不小心作错删除标记的记录,又可用恢复命令 RECALL 恢复。

1. 删除命令 DELETE

DELETE [<范围 >] [FOR <条件 >] [WHILE <条件 >]

功能:给当前表中所指定的记录作删除标记 (*)。

例:删除性别为女的同志。

```
USE JBQK                                    && 打开表文件
DELETE FOR 性别 = "女"                      && 给女同志加上删除标志
list FIELDS 编号,姓名,性别,出生年月   && 显示记录的编号,姓名,性别,出生年月
```

记录号	编号	姓名	性别	出生年月
1	100001	赵一同	男	01/23/68
2 *	100002	王小小	女	08/12/75
3 *	100003	周玖月	女	09/19/60
4	100004	吴建国	男	07/09/50
5 *	100005	孙颖颖	女	11/23/70
6 *	100006	肖丽丽	女	06/01/65
7	100007	陈若无	男	02/12/66
8 *	100008	万维妙	女	06/24/81

2. 取消删除标记命令 RECALL

RECALL [<范围 >] [FOR <条件 >] [WHILE <条件 >]

功能：恢复当前表文件中已作了删除标记的记录，即取消删除标记（*）

例：恢复出生年月在 1970 年以后出生的同志。

RECALL ALL FOR YEAR（出生年月）>1970

list FIELDS 编号，姓名，性别，出生年月　　&& 显示记录的编号，姓名，性别，出生年月

记录号	编号	姓名	性别	出生年月
1	100001	赵一同	男	01/23/68
2	100002	王小小	女	08/12/75
3 *	100003	周玖月	女	09/19/60
4	100004	吴建国	男	07/09/50
5 *	100005	孙颖颖	女	11/23/70
6 *	100006	肖丽丽	女	06/01/65
7	100007	陈若无	男	02/12/66
8	100008	万维妙	女	06/24/81

3. **永久性删除命令** PACK

在发出 PACK **命令之前，带有删除标记的记录并没有在物理上被删除。如果** SET DELETED **设置为** OFF**，那么在“浏览”窗口中查看表时，可以看到每个要删除的记录都已标识了删除标记，但记录在表中仍可见。如果** SET DELETED **设置为** ON**，则带有删除标记的记录在“浏览”窗口中不可见。**

SET DELETED **命令的设置也同样影响了操作记录的命令能否作用于带有删除标记的记录。**

格式：PACK

例：删除所有带 * 号的记录。

PACK

list FIELDS 编号，姓名，性别，出生年月

结果：

记录号	编号	姓名	性别	出生年月
1	100001	赵一同	男	01/23/68
2	100002	王小小	女	08/12/75
3	100004	吴建国	男	07/09/50
4	100007	陈若无	男	02/12/66
5	100008	万维妙	女	06/24/81

功能：清除当前表文件中已作了删除标记的记录。

4. 全部删除命令 ZAP

格式：ZAP

功能：删除当前表文件中所有的记录。

说明：

此命令只剩下表结构，相当于执行了 DELETE ALL 和 PACK 命令。

§5.2 索引和排序

在实际操作过程中，若要按特定的顺序定位、查看或操作表中记录，可以使用索引。Visual FoxPro 使用索引作为排序机制，为开发应用程序提供灵活性。根据应用程序的要求，可以灵活地对同一个表创建和使用不同的索引关键字，使您可按不同顺序处理记录。也能根据这些索引创建自定义表间关系，使您能准确地访问想要的记录。

Visual FoxPro 索引是由指针构成的文件，这些指针逻辑上按照索引关键字的值进行排序。索引文件和表的. dbf 文件分别存储，并且不改变表中记录的物理顺序。实际上，创建索引是创建一个由指向. dbf 文件记录的指针构成的文件。若要根据特定顺序处理表记录，可以选择一个相应的索引，使用索引还可以加速对表的查看和访问。

对于已经建好的表，可以利用索引对其中的数据进行排序，以便加速检索数据的速度。可以用索引快速显示、查询或者打印记录。还可以选择记录、控制重复字段值的输入并支持表间的关系操作。

Visual FoxPro 中的索引和书中的索引类似。书中的索引是一份页码的列表，指向书中的页号。表索引是一个记录号的列表，指向待处理的记录，并确定了记录的处理顺序。表索引存储了一组记录指针。

5.2.1 索引文件的类型

1. 根据扩展名来分

⑴ 包含单一关键字的索引文件（. IDX）

这种索引文件只能按关键字段的值升序排序。它分为非压缩的. IDX 文件和压缩的. IDX 文件。

⑵ 包含多重关键字的复合索引文件（. CDX）

这种索引文件中可以存储多个索引，对每一个索引都取一个特殊的名字，称为一个标记。它分为：

① 结构化复合索引文件 (Structure compound index file)：结构化复合索引文件的文件名是自动分配的，与相关的表文件名一样，它是在 Visual FoxPro 数据库中最普通也最重要的一种索引文件。索引保存在一个结构复合索引文件中，每次表打开时自动打开，而且在表修改时自动更新的索引文件。

② 非结构化复合索引文件 (non - Structure compound index file)：非结构化复合索引文件的文件名与表的名字不同，当表打开时这种索引文件不会自动打开，必须由用户打开它。

2. 根据索引功能来分

⑴ 主索引

主索引可确保字段中输入值的唯一性并决定了处理记录的顺序，可以为数据库中的每一个表建立一个主索引。如果某个表已经有了一个主索引，可以继续添加候选索引。

⑵ 候选索引

候选索引像主索引一样要求字段值的惟一性并决定了处理记录的顺序。在数据库表和自由表中均可为每个表建立多个候选索引。

⑶ 普通索引

普通索引也可以决定记录的处理顺序，但是允许字段中出现重复值。在一个表中可以加入多个普通索引。

⑷ 唯一索引

为了保持同早期版本的兼容性，还可以建立一个惟一索引，以指定字段的首次出现值为基础，选定一组记录，并对记录进行排序。

能打开的索引文件 (. idx 或. cdx) 数目只受内存及系统资源的限制。

表 5-1 是这三种索引类型的总结，包括它们如何命名、可以包含的关键字数目、每种索引的字符限制。

独立索引文件基于单关键字表达式，并保存在. idx 文件中。. cdx 文件可以存储多关键字表达式，. idx 索引只存储单关键字表达式。

表 5-1 三种索引类型总结

索引类型	描 述	关键字数目	限 制
结构. cdx	使用和表文件名相同的基本名,随表的打开自动打开	多关键字表达式,称为标识	有效表达式限制在 240 个字符之内
非结构. cdx	必须明确地打开,使用和表名不同的基本名	多关键字表达式,称为标识	有效表达式限制在 240 个字符之内
独立. idx	必须明确地打开. idx	文件的基本名由用户定义	单关键字表达式。有效表达式限制在 100 个字符之内

5.2.2 索引文件的建立

要在表中建立索引,可以利用表设计器,其操作步骤如下:

在“项目管理器”中选择待建索引的表,然后选择“修改”。

在“表设计器”中,选择“索引”选项卡。

“表设计器”中的“索引”选项卡在“索引名”框中,键入索引名。

从“类型”列表中,选定索引类型。

有关索引类型的详细内容,请参阅后面的“选择索引类型”部分。

在“表达式”框中,键入作为记录排序依据的字段名。在实际操作过程中如果要用到表达式,则通过选择表达式框后面的对话按钮,显示“表达式生成器”来建立表达式。

若想有选择地输出记录,可在“筛选”框中输入筛选表达式,或者选择该框后面的按钮来建立表达式。

选择“确定”。

建好表的索引后,便可以用它来为记录排序。

前面我们所介绍的是利用界面操作的,实际上在软件开发中我们都会通过程序来操作。下面我们通过命令来创建索引:

格式:INDEX ON〈索引关键字〉TO〈单索引文件名〉| TAG〈索引标识名〉 [OF〈.复合索引文件名〉] [FOR〈条件〉] [ASCENDING | DESCENDING] [UNIQUE] [ADDITIVE] [COMPACT]

功能:建立索引文件或给表加上索引标识。

说明:

TO 子句用于建立单索引文件,其名字由单索引文件名标识。

TAG〈索引标识名〉用于建立复合索引文件和索引标识。

ON〈索引关键字〉指定索引字段的组合形式,可由单一字段组成,也可由多个字段组成。

选项 [OF〈.复合索引文件名〉]　指定建立非结构化复合索引文件。

选项 [ASCENDING | DESCENDING]项是选择升序还是降序

选项 [UNIQUE]　只保留索引字段相同的多条记录中的第一条记录。

选项 [ADDITIVE]建立一个新索引文件时,不关闭原已打开的索引文件。

选项 [COMPACT]指定建立的单一索引文件以压缩型方式存储。

例：建立单索引文件示例

根据出生年月生成一单索引文件 IDX1

```
USE d：hvfbook hjbqk. dbf EXCLUSIVE
BROWSE LAST
INDEX ON 出生年月 TO IDX1
LIST FIELDS 编号,姓名,性别,出生年月,部门代号,党员否
```

结果如下：

记录号	编号	姓名	性别	出生年月	部门代号	党员否
4	100004	吴建国	男	07/09/50	2	.T.
3	100003	周玖月	女	09/19/60	1	.T.
6	100006	肖丽丽	女	06/01/65	2	.F.
7	100007	陈若无	男	02/12/66	1	.T.
1	100001	赵一同	男	01/23/68	1	.T.
5	100005	孙颖颖	女	11/23/70	5	.F.
2	100002	王小小	女	08/12/75	4	.F.
8	100008	万维妙	女	06/24/81	3	.T.

按照基本工资建立一按降序排列的单索引文件 IDX2

```
USE d：hvfbook hGZqk. dbf EXCLUSIVE
INDEX ON - 基本工资 TO IDX2
LIST FIELDS 编号,姓名,基本工资
```

结果如下：

记录号	编号	姓名	基本工资
4	100004	吴建国	643.00
6	100006	肖丽丽	584.00
7	100007	陈若无	564.00
3	100003	周玖月	530.00
5	100005	孙颖颖	488.00

1	100001	赵一同	446.00
2	100002	王小小	345.00
8	100008	万维妙	311.00

为 JBQK.DBF 建立一个复合索引文件,其共包括两个索引标识:

根据出生年月建立一升序索引,其索引标识为 CDX1

```
USE d:\vfbook\JBqk.dbf EXCLUSIVE
INDEX ON 出生年月 TAG CDX1 DESCENDING
LIST FIELDS 编号,姓名,性别,出生年月,部门代号,党员否
```

结果如下:

记录号	编号	姓名	性别	出生年月	部门代号	党员否
8	100008	万维妙	女	06/24/81	3	.T.
2	100002	王小小	女	08/12/75	4	.F.
5	100005	孙颖颖	女	11/23/70	5	.F.
1	100001	赵一同	男	01/23/68	1	.T.
7	100007	陈若无	男	02/12/66	1	.T.
6	100006	肖丽丽	女	06/01/65	2	.F.
3	100003	周玖月	女	09/19/60	1	.T.
4	100004	吴建国	男	07/09/50	2	.T.

根据部门代号和出生年月建立一索引,其索引标识为 CDX2

```
INDEX ON 部门代号 + DTOC(出生年月) TAG CDX2
LIST FIELDS 编号,姓名,性别,出生年月,部门代号,党员否
```

结果如下:

记录号	编号	姓名	性别	出生年月	部门代号	党员否
1	100001	赵一同	男	01/23/68	1	.T.
7	100007	陈若无	男	02/12/66	1	.T.
3	100003	周玖月	女	09/19/60	1	.T.
6	100006	肖丽丽	女	06/01/65	2	.F.
4	100004	吴建国	男	07/09/50	2	.T.
8	100008	万维妙	女	06/24/81	3	.T.
2	100002	王小小	女	08/12/75	4	.F.
5	100005	孙颖颖	女	11/23/70	5	.F.

通过上面的例题,我们可以总结一下创建索引时应注意的问题:

(1) 自由表不能够建立主索引,因为只有在数据库中的表才能够建立主索引。

⑵ 若要创建降序索引,可在“表设计器”的“索引”选项卡中选择“索引名”框左边的箭头按钮,使箭头向下。或者使用 INDEX ON 命令的 DESCENDING 子句,创建降序索引。

⑶ 使用简单表达式进行索引。简单索引表达式基于单字段、或者基于两个以上字段组成的多字段关键字。例如,可以根据下面的表达式:

顾客表的索引:国家 + 地区 + 编号

当浏览根据这个索引标识排序的顾客表现时,可以看到顾客已根据国家/地区进行排序,然后是地区,最后根据顾客的编号进行排序。

⑷ 使用复杂表达式进行索引。也可以根据更复杂的表达式创建索引。Visual FoxPro 索引关键字表达式可以包括 Visual FoxPro 函数、常量或用户自定义函数。

创建的表达式对于独立(.idx)索引标识来说不能超过 100 个字符,对于.cdx 索引标识来说不能超过 240 个字符。在一个标识中,通过把表达式的单个组件转换成字符类型,可以一起使用不同的数据类型。

⑸ 在索引标识中可以使用 Visual FoxPro 函数。例如,可以使用 STR()函数把一个数值型值转换成字符串。如果想在一个表中创建一个将字符型 cust_id 字段和一个数值型 maxordamt 字段组合起来的索引标识,可以使用以下代码,将 maxordamt 字段有两个小数位的 8 字符字段:

```
INDEX ON cust_id + STR(maxordamt,8,2) TAG custmaxord
```

若要减小包含整数的字段索引的大小,可用 BINTOC()函数将整数转换为其对应二进制数所代表的字符,也可用 CTOBIN()函数做相反的转换。

如果想创建一个根据日期对表进行排序的索引,可以使用 DTOS()函数把日期字段转换成字符串。若要根据日期型字段 hire_date 和字符型字段 emp_id 访问 employee 表,可以使用以下代码创建这个索引关键字表达式:

```
INDEX ON DTOS(hire_date) + emp_id TAG id_hired
```

5.2.3　索引文件的使用

除了最普通的索引(结构.cdx 索引),Visual FoxPro 支持其他两种索引文件:非结构的.cdx 索引和独立的.idx 索引。非结构的.cdx 索引用于很少使用的多关键字标识 独立的.idx 索引用于暂时的或不经常使用的单关键字索引,独立的.idx 索引主要是为了满足向后兼容性。

通常可以使用独立索引作为临时索引,在需要它们时再创建或重新编排这些索引。例如,有一个只用于每季度或每年度总结报表的索引。如果把这个不经常使用的索引包括到结构.cdx 文件中,便要求在每次使用表时都要维

护它,因此最好创建一个独立. idx 索引。对一个特定的表,可创建任意多个. idx 文件。

由于特殊的目的,有时想创建多个索引标识,但又不想在运行过程中维护这些索引,因为这样会增加应用程序的负担。此时,非结构. cdx 索引很有用。例如,您的应用程序中有一组专门的报表,分析一些字段的数据,但这些字段并没有建立正常索引。应用程序就可以创建一个包含所需索引标识的非结构. cdx 索引,然后运行这些专门的报表,最后再删除非结构. cdx 文件。

通过建立和使用索引,可以提高完成某些重复性任务的工作效率,例如对表中的记录排序,以及建立表之间的关系等。根据所建索引类型的不同,可以完成不同的任务:

若要排序记录,以便提高显示、查询或打印的速度,使用普通索引、候选索引或主索引。

在字段中控制重复值的输入并对记录排序,对数据库表使用主索引或候选索引,对自由表使用候选索引。

通常在界面环境下要用索引对记录排序,其操作步骤为:

在"项目管理器"中选择已建好索引的表。

选择"浏览"。

从"表"菜单中选择"属性"。

在"索引顺序"框中,选择要用的索引。

选择"确定"。

显示在"浏览"窗口中的表将按照索引指定的顺序排列记录。选定索引后,通过运行查询或报表,还可对它们的输出结果进行排序。

在命令窗口打开索引文件,具体操作如下:

1. 打开索引文件

格式 1: SET INDEX TO〈索引文件名表〉[ADDITIVE]

功能:对已打开的表文件,再打开相应的索引文件。

例:

```
USE JBQK                    && 打开 JBQK 数据表
SET INDEX TO IDX1,IDX2      && 打开 IDX1. IDX 和 IDX2. IDX
LIST                        && 记录按出生年月升序排列,即 IDX1. IDX 为主控索引
```

说明:

索引文件在刚建立之后,索引文件已经打开。

除结构复合索引之外,其他文件必须用命令才能显式打开。

在任何时候都只有一个索引文件起作用，虽然可以同时打开多个索引文件，或者是一个复合索引文件但包括多个索引标识。当前起作用的索引文件称作主控索引文件。当前起作用的索引标识称作主控索引。

索引文件中表中的第一个索引文件称之为主控索引文件。

省略 [ADDITIVE]，打开索引文件时，除结构复合索引文件以外的任何索引文件均关闭。

格式 2：　USE　〈文件名〉　INDEX　TO〈索引文件名表〉

功能：打开表文件的同时也打开相应的索引文件。

例：

```
USE JBQK INDEX TO IDX1,IDX2    && 打开表时同时打开 IDX1.IDX 和 IDX2.IDX
```

2. 选择主索引

SET　ORDER　TO〈数值表达式〉〈单索引文件名表〉hTAG〈索引标识〉[　ASCENDING　|　DESCENDING]

功能：用来确定主控索引和主控索引文件。

说明：

SET ORDER TO 或 SET ORDER TO 0 用于取消主控索引文件和主控索引。

例：

```
USE JBQK INDEX IDX1                 && 打开表时同时打开 IDX1.IDX
USE GZQK INDEX IDX2                 && 打开表时同时打开 IDX2.IDX
LIST                                && 记录按基本工资降序排列
SET ORDER TO TAG CDX1               && 打开复合索引文件 JBQK.CDX 的
                                       索引标识 CDX1
LIST                                && 记录按出生年月降序排列
SET ORDER TO 1                      && 指定 IDX1.IDX 为主控索引
LIST                                && 记录按出生年月升序排列
SET ORDER TO 3                      && 指定 JBQK.CDX 中的 CDX1 主控
                                       索引
LIST                                && 记录按出生年月降序排列
```

3. 关闭索引文件

格式：　SET　INDEX　TO

功能：关闭当前表的索引文件。

例：

```
USE JBQK INDEX IDX1        && 打开表时同时打开 IDX1.IDX
USE GZQK INDEX IDX2        && 打开表时同时打开 IDX2.IDX
SET INDEX TO               && 关闭所有 IDX 文件,取消主控索引
LIST                       && 记录按物理顺序排列
```

4. **重新索引**

REINDEX

功能：对已打开的表按对应的索引文件自动进行重新索引。

说明：

索引文件必须打开并确定主控索引文件或主控索引，数据发生变化时才能自动更新。

除此之外，我们还可以用 INDEX ON **重新索引。**

5. **复合索引文件的删除**

方式 1：DELETE ALL TAG〈**索引标识名** 1〉[〈**索引标识名** 2〉]…

功能：用于可选择性地删除复合索引文件中的某一索引标识符名。

方式 2：**用文件删除命令删除。**

5.2.4 排序

1. **排序与索引的异同**

排序和索引都是对表按一定的顺序重新组织，实现的目的相同，但手段和方法有很大的差异。

排序是重新建立一个表文件，排序完成后，出现两个表，原表和新的排序表。要使用新的排序后的表，必须打开这个表。

索引是建立索引文件，索引文件中仅包含排了序的关键字段内容及其相应记录号并与原表同时使用。

2. **排序文件的建立**

SORT TO <**新文件名**> ON <**字段** 1> [/A] [/C] [/D] [,<**字段** 2>] [/A][/C][/D]...] [<**范围**>] [FIELDS <fields list>] [FOR <**条件**>][WHILE <**条件**>]

功能：按某一些字段建立一个新的排序文件。

说明：

TO <**新文件名**> **指定排序文件名。**

ON <**字段** 1> **指定排序的字段。**

选项 [/A] 表示升序 ASCENDING,省略为升序。

选项 [/C] 表示按字符型字段排序时,大小写字母无区别。

选项 [/D] 表示降序 DESCENDING。

范围和条件缺省表示对所有记录进行操作。

§5.3　表记录的查询

所谓查询,是指人们按照用户给定的条件在表中查找满足条件的记录。根据查询方式可分为顺序查询和索引查询。顺序查询是指按照记录的物理顺序来进行的。而索引查询则是利用了数学当中的二分法进行查询,由于采用了索引,加快了查询速度。

5.3.1　顺序查询 (LOCATE 命令)

Locate　[〈范围〉]　For < 条件 > [WHILE < 条件 >]

功能:按指定条件在一定范围中进行顺序查询。

说明:

范围缺省,系统默认为 ALL。

FOR　< 条件 > 查找所有使条件表达式为真的记录。

若找到,记录指针会指向第一个满足条件的记录,若没有满足条件的记录,指针指向表文件尾,主屏幕的状态栏中显示“已到定位范围末尾”。

若有多条满足条件的记录,要继续查找下一条满足条件的记录,必须用 CONTINUE 命令。

例:

```
use jbqk
LOCATE FOR 性别 = "女" . AND. 职称 = "副教授"
DISPLAY 姓名,出生年月,职称
```

结果如下:

记录号	姓名	出生年月	职称
3	周玖月	09/19/60	副教授

```
CONTINUE
DISPLAY 姓名,出生年月,职称
```

结果如下:

记录号	姓名	出生年月	职称
6	肖丽丽	06/01/65	副教授

CONTINUE && 再执行该命令

状态栏中显示“已到定位范围末尾”。

5.3.2 索引查询

和顺序查询相比,索引查询速度快,但是要先建立索引,然后才能查询,其中 SEEK 和 FIND 命令都可进行索引查询。

命令 1:SEEK〈索引关键字〉

功能:在已打开索引的表中根据关键字的值查找满足条件的第一个记录。并把指针指向该记录。

说明:

若找到,记录指针会指向第一个满足条件的记录,若没有满足条件的记录,指针指向表文件尾,主屏幕的状态栏中显示“没有找到”。

执行该命令时,主索引文件的关键值可以是字符型的,也可以是数字型的。

如果是字符型,命令中的字符串可以是关键数值的全部,也可以是从它左边开始的任一部分。

如果是数值,就必须把数字确切地列出来。

在这个命令中,字符串必用引号括起来。

如果该字符串是内存变量的数值,可用内存变量名来代替。

例:查找 1965 年 6 月 5 日出生的人。

```
USE JBQK INDEX IDX1
SET STRICTDATE TO 0              设置系统为常用日期方式
SEEK {06/01/65}
DISPLAY 姓名,出生年月,职称
```

结果如下:

记录号	姓名	出生年月	职称
6	肖丽丽	06/01/65	副教授

在 GZQK.DBF 表中查找工资为 446 元的职工。

```
USE d:\vfbook\hgzqk.dbf EXCLUSIVE
set index to idx2
seek -446        && 根据 - 基本工资索引,所以用 - 446
disp 编号,姓名,基本工资, 地区补贴
```

结果如下:

记录号	编号	姓名	基本工资	地区补贴
1	100001	赵一同	446.00	223.00

例：查找姓“孙”的同学。

```
USE d：\vfbook\hgzqk. dbf EXCLUSIVE
INDEX ON 姓名 TO IDX3
X = "孙"
SEEK X
disp 编号,姓名,基本工资,地区补贴
```

结果如下：

记录号	编号	姓名	基本工资	地区补贴
5	100005	孙颖颖	488.00	321.00

命令 2：　FIND 〈索引关键字〉

功能：在已打开索引的表中根据关键字的值查找满足条件的第一个记录，并把指针指向该记录。

说明：

若找到，记录指针会指向第一个满足条件的记录，若没有满足条件的记录，指针指向表文件尾，主屏幕的状态栏中显示“没有找到”。

执行该命令时，主索引文件的关键值可以是字符型的，也可以是数字型的。

如果是字符型，命令中的字符串可以是关键数值的全部，也可以是从它左边开始的任一部分。

如果是数值，就必须把数字确切地列出来。

在这个命令中，字符串不必用引号括起来，除了该关键字最前面包含若干空格。

如果该字符串是内存变量的数值，可用内存变量名来代替，但必须用宏代换函数。

例：查找姓“孙”的同学。

```
USE d：\vfbook\hgzqk. dbf EXCLUSIVE
INDEX ON 姓名 TO IDX3
X = "孙"
FIND &X
disp 编号,姓名,基本工资,地区补贴
```

结果如下：

记录号	编号	姓名	基本工资	地区补贴
5	100005	孙颖颖	488.00	321.00

5.3.3 使用 SET FILTER 查询

如果要执行一串命令，仅对满足某个相同条件的记录进行操作，这时需要在每条命令中都使用 FOR 子句，显然，这是很麻烦的，为此 Visual FoxPro 提供了一个过滤器。建立了过滤器之后，不满足条件的记录被过滤掉，留下可以操作的记录都是满足条件的，这样就不必为逐条命令增加 FOR 条件子句了。

格式：SET FILTER TO [<条件>]

功能：在当前表中按[<条件>]条件滤掉所有不满足条件的记录。

说明：

命令后不接[<条件>]，则关闭该命令。

例：查看基本工资大于 500 元的职工。

```
set index to
set filter TO 基本工资 >500
LIST 编号,姓名,基本工资,地区补贴
```

记录号	编号	姓名	基本工资	地区补贴
3	100003	周玖月	530.00	321.00
4	100004	吴建国	643.00	366.00
6	100006	肖丽丽	584.00	402.00
7	100007	陈若无	564.00	342.00

```
set filter TO                    && 关闭过滤器
LIST 编号,姓名,基本工资,地区补贴
```

记录号	编号	姓名	基本工资	地区补贴
1	100001	赵一同	446.00	223.00
2	100002	王小小	345.00	187.00
3	100003	周玖月	530.00	321.00
4	100004	吴建国	643.00	366.00
5	100005	孙颖颖	488.00	321.00
6	100006	肖丽丽	584.00	402.00
7	100007	陈若无	564.00	342.00
8	100008	万维妙	311.00	150.00

5.3.4 用 RQBE 查询

结构化查询语言 SQL 是一种为维护表而开发的强有力的语言。SELECT

命令是 SQL 用来进行表查询的主要工具。在任何 Visual FoxPro 程序中都可以使用 SQL 语句,也可以像输入其他任何命令一样将 SQL 语句输入到 Command 窗口中。

SELECT [ALL | DISTINCT] < select_item > [AS < colum_name >] [, < select_item > [AS < colum_name >] …] FROM < table > [, < table > …]

功能:SQL 语言语句保留字。

说明:

选项 [ALL | DISTINCT] 中 DISTINCT 是消除重复项和记录,ALL 是缺省情况,在输出中包含重复项。

< select_item > 项可以是包含在某张表中的字段名、常量值和表达式。用“ * ”表示将 FROM 子句中的所有列表中的所有字段包含在输出列表中。

选项 [AS < colum_name >] 选项使用户能够指定任何输出字段的名字,在用户查看查询结果的 Browse 窗口中,这个新的字段名也可充当列标题。

FROM < table > 子句是告诉 SQL 在查询中包含哪些表。

§5.4　表中数值参数的统计

5.4.1　求和

SUM [< 范围 >] [< 表达式列表 >] [TO < 内存变量名列表 >] [FOR < 条件 1 >] [WHILE < 条件 2 >]

功能:对当前表文件的数值型字段求和。

说明:

范围缺省,代表 ALL。

缺省“表达式列表”,表示对表文件中所有数值型字段进行操作。

选项 [< 表达式列表 >] 用于指定求和的数值字段表,省略为对所有数值型字段求和。

选项 < 内存变量名列表 > 指定内存变量表。

USE d:\hvfbook\hgzqk.dbf EXCLUSIVE

SUM 基本工资,地区补贴 TO A1,A2

?A1,A2

结果如下:

3911.00 2312.00

5.4.2 平均值

AVERAGE [<范围>] [<表达式列表>][TO <内存变量名列表>] [FOR <条件1>][WHILE <条件2>]

功能：计算当前表文件数值型字段的平均值。

说明：

范围缺省，代表 ALL。

缺省"表达式列表"，表示对表文件中所有数值型字段进行操作。

选项[<表达式列表>]**用于指定求平均的数值字段表，省略为对所有数值型字段求平均。**

选项<**内存变量名列表**>**指定内存变量表。**

USE d：\vfbook\hgzqk.dbf EXCLUSIVE

aver **基本工资** to ab

?ab

488.88

aver **水电费** + **公积金** to bb

104.75

5.4.3 计数

COUNT [<范围>]][TO <内存变量名>] [FOR <条件1>][WHILE <条件2>]

功能：计算当前表文件中满足指定条件的记录个数。

说明：

范围缺省，代表 ALL。

USE d：\vfbook\hjbqk.dbf EXCLUSIVE

count for **性别** = "**女**" to ab

?ab

5

count for **党员否** to bc

? bc

5

5.4.4　汇总

TOTAL　ON　<索引关键字> TO <目标文件名>　[<范围>] [FIELDS　<字段名列表>] [FOR　<条件 1>] [WHILE　<条件 2>]

功能：由当前数据表产生一个新的数据目的表，按关键字段相同的记录的汇总，也称为同类项合并。

说明：

ON　<索引关键字>中<索引关键字>为分类求和关键字。执行该命令时要求当前表必须按关键字段作索引，或打开相应的索引文件。

执行此命令时，系统会对表文件中“关键字段”的字段值相同的记录进行求和操作。同时，根据这几个关键字段值相同的记录，在目的表文件中生成一个记录。凡未求和的字段，生成的字段值等于这几个记录中第一个记录的字段值，在求和字段上，生成的字段值等于求和的结果。

我们假设一商店中表“XS”记载了一天的销售情况：

```
USE XS
LIST
```

记录号	产品名	单价	数量	总价
1	巧克力	8.00	3	24.00
2	矿泉水	2.00	9	18.00
3	面包	1.05	6	6.30
4	巧克力	8.00	4	32.00
5	矿泉水	1.05	3	3.15
6	巧克力	8.00	1	8.00

我们来统计一天的销售情况，其操作如下：

```
INDEX ON 产品名 TO CP
TOTAL ON 产品名 TO ZJ FIELDS 数量,总价
USE ZJ
LIST
```

记录号	产品名	单价	数量	总价
1	矿泉水	2.00	12	21.15
2	面包	1.05	6	6.30
3	巧克力	8.00	8	64.00

§5.5 文件操作命令

5.5.1 文件复制命令——COPY

1. 任何文件的整体复制

COPY FILE〈源文件〉 TO〈目的文件名〉

功能：产生一个和源文件完全相同的文件。

说明：

源文件和目标文件名都必须写完全，即要写扩展名。

操作时，源文件不能够打开。

如果复制的是表文件，且该文件含有备注型字段，则在拷贝该文件时，还要用命令把和该文件名同名的 FPT 文件拷贝过来，否则，表文件打不开。

例：复制表 JBQK，其结构和内容完全一样。

```
SET DEFA TO D：hVFBOOK                  && 设置默认路径为 D：hVFBOOK
COPY FILE JBQK.DBF TO JBQK1.DBF         && 拷贝扩展名为 DBF 的文件
USE JBQK1.DBF                           && 打开刚新建文件
```

屏幕出现提示 FPT 文件缺少或无效。

```
COPY FILE JBQK.FPT TO JBQK1.FPT         && 拷贝 FPT 文件
USE JBQK1.DBF                           && 打开刚新建文件
```

如果已经建立索引文件，可以再次拷贝索引文件，如果选择忽略，文件直接被打开。

```
BROWSE LAST                             && 浏览表内容，和原表完全一样
MODIFY STRUCTURE                        && 浏览表结构，和原表完全一样
```

2. 表文件部分内容的拷贝

在实际使用过程中，我们可能只会用到某表文件的部分内容以及部分字段的值，这时我们不必再创建表文件，只要通过命令来复制。

COPY TO <目标表文件名> [<范围>] [FIELDS <字段列表>] [FOR <条件 1>] [WHILE <条件 2>]

功能：把源表文件中满足条件的记录复制到目标表文件名中，目标文件所包含的字段由命令中的字段列表列出。

说明：

源文件必须打开,目标文件的扩展名可省略。如果拷贝的文件包含备注型字段,系统会自动生成备注型文件,不必再拷贝。

例:

```
use jbqk              && 打开表 JBQK
copy to jbqk2         && 把它复制成和原表完全一样的一个表 JBQK2
use jbqk2             && 打开表 JBQK2
list                  && 显示内容和原表完全一样
copy to jbqk3 fields  编号,姓名,性别 FOR 性别 = "男"
&& 把原表中男性职工的编号、姓名和性别拷贝至另一个表 JBQK3
use jbqk3             && 打开表 JBQK3
list                  && 显示表内容
```

结果如下:

记录号	编号	姓名	性别
1	100001	赵一同	男
2	100004	吴建国	男
3	100007	陈若无	男

3. 复制表文件结构

COPY　STRUCTURE TO　<目标表文件名>　[FIELDS <字段列表>]

功能:根据字段列表给定的字段把源表文件结构复制给目标表文件名。

说明:

该命令复制的只是空结构,不包括表内容。

源文件必须处于打开状态。

目标文件名中的扩展名可省略。

例:创建一个表结构,它包含原表中编号、姓名、性别三个字段。

```
use jbqk
copy stru  to jbqk4 fields  编号,姓名,性别
use jbqk4
list stru
表结构:                    D: hVFBOOKhJBQK4. DBF
数据记录数:                0
最近更新的时间:            03/23/02
代码页:                    936
```

字段	字段名	类型	宽度	小数位
1	编号	字符型	6	
2	姓名	字符型	8	
3	性别	字符型	2	
**	总计	**	17	

4. **把表文件的结构复制成表文件**

COPY STRUCTURE EXTENDED TO <**目标表文件名**>

功能：

把表结构复制到一新表中。

说明：

新表中共包含 16 **个字段，其中我们常用的是前四个字段，其字段名和长度都是固定的。其字段名分别是** FIELD_NAME，FIELD_TYPE，FIELD_LEN **和** FIELD_DEC，**各字段的长度分别是** 128，1，3，3。

新表的内容就是源表的结构，也就是把源表中每一个字段的字段名、类型、长度和小数点位数当做新表的一个记录。

例：把表 JBQK **的结构复制成表文件** JBQK5。

```
USE JBQK
COPY STRU EXTENDED TO JBQK5
USE JBQK5
LIST FIELDS FIELD_NAME,FIELD_TYPE,FIELD_LEN, FIELD_DEC
&& 显示表内容的前四个字段
```

记录号	FIELD_NAME	FIELD_TYPE	FIELD_LEN	FIELD_DEC
1	编号	C	6	0
2	姓名	C	8	0
3	性别	C	2	0
4	出生年月	D	8	0
5	职称	C	6	0
6	部门代号	C	1	0
7	党员否	L	1	0
8	简历	M	4	0
9	照片	G	4	0

```
LIST STRUCTURE      && 显示新表的表结构
```

表结构： D：hVFBOOKhJBQK5. DBF

数据记录数：　　9
最近更新的时间：　　03/23/02
备注文件块大小：　　64
代码页：　　936

字段	字段名	类型	宽度	小数位	索引	排序	Nulls
1	FIELD_NAME	字符型	128				否
2	FIELD_TYPE	字符型	1				否
3	FIELD_LEN	数值型	3				否
4	FIELD_DEC	数值型	3				否
5	FIELD_NULL	逻辑型	1				否
6	FIELD_NOCP	逻辑型	1				否
7	FIELD_DEFA	备注型	4				否
8	FIELD_RULE	备注型	4				否
9	FIELD_ERR	备注型	4				否
10	TABLE_RULE	备注型	4				否
11	TABLE_ERR	备注型	4				否
12	TABLE_NAME	字符型	128				否
13	INS_TRIG	备注型	4				否
14	UPD_TRIG	备注型	4				否
15	DEL_TRIG	备注型	4				否
16	TABLE_CMT	备注型	4				否
**	总计 **		302				

```
APPEND BLANK                && 在表 JBQK5 中新增一个新记录
REPL FIELD_NAME WITH "工资",FIELD_TYPE WITH "N",FIELD_LEN WITH 9,FIELD_DEC WITH 2
                            && 给新记录赋值
LIST                        && 在新表中多了一条 FIELD_NAME 为“工资”,FIELD_TYPE 为“N”,FIELD_LEN WITH 为 9,FIELD_DEC 为 2 的记录
```

5. **建立表文件结构命令**——CREATE FROM

通常我们都是通过 CREATE **命令用全屏幕编辑方式来建立一个表文件结构,而在上面的例子中我们可以看到,我们把表结构可以生成一个表文件,每一个字段就是一个记录。因此我们可以通过增加记录的方式增加一个字段,然后通过** CREATE FROM **又把表文件重新生成为一个新表。**

格式：CREATE　<**新建表文件名**>　FROM　<**源表文件名**>

功能：建立表结构

说明：

常和前面所介绍的第四条命令合用，不通过全屏幕编辑方式就可用于增加或删除字段。

例：

CREATE JBQK6 FROM JBQK5

USE JBQK6

LIST && **无记录，是一个空表**

LIST STRU && **显示表结构**

表结构： D：hVFBOOKhJBQK6. DBF

数据记录数： 0

最近更新的时间： 03/24/02

备注文件块大小： 64

代码页： 936

字段	字段名	类 型	宽度	小数位	索引	排序	Nulls
1	编号	字符型	6				否
2	姓名	字符型	8				否
3	性别	字符型	2				否
4	出生年月	日期型	8				否
5	职称	字符型	6				否
6	部门代号	字符型	1				否
7	党员否	逻辑型	1				否
8	简历	备注型	4				否
9	照片	通用型	4				否
10	工资	数值型	9				否
**	总计 **		50				

5.5.2 文件重命名命令

对于一个文件，一个好的名字对于程序设计和程序阅读都有帮助。在操作过程中我们可以用命令重新给文件命名。

格式：RENAME <**原文件名**> FROM <**新文件名**>

功能：更改一个文件的名字，注意其内容不改变。

说明：

原文件不能打开。

原文件和新文件名在命令行中必须带扩展名。
如果是表文件且带有备注型字段,则需再一次对 FPT 文件重命名。
例:把表 JBQK2 重新命名为 JBQK22。

```
RENAME JBQK2. DBF TO JBQK22. DBF
USE JBQK22                          && 屏幕出现 JBQK22. FPT 文件缺少或无效
RENAME JBQK2. FPT TO JBQK22. FPT    && 对备注型文件重新命名
USE JBQK22                          && 文件能正常打开
```

5.5.3　删除文件命令

对于一个无用的文件,放在磁盘当中浪费空间,因此我们可通过命令来删除。
格式 1:DELETE　FILE　< 文件名 >
格式 2:ERASE　< 文件名 >
功能:删除一个文件。
说明:
可用两种格式中的任何一种。
被删除文件不能打开,且应带上扩展名。
例:

```
ERASE JBQK6. DBF
ERASE JBQK6. FPT
```

5.5.4　查看文件目录

格式:DIR/DIRECTORY [< 驱动器号: >] [< 路径 >] [< 文件名 >] [TO PRINT]
功能:用于显示指定的磁盘上欲查找的文件名。
说明:
若驱动器号和路径都省略,则显示当前盘当前路径的文件。
可使用通配符“?”和“*”
例:显示 D:hVFBOOK 下的所有表文件:

```
set defa to d:hvfbook          && 设置默认路径
dir *. dbf                     && 显示 d:hvfbook 下的所有表文件
```

§6.1 VFP 程序设计的概念

学习 Visual FoxPro 的目的，主要是需使用它的命令来组织和处理数据、完成一些具体的任务。而许多任务是无法靠一条命令完成的，需要通过执行若干条命令来完成。不过采用在命令窗口中逐条输入命令的方式进行，不仅麻烦，而且容易出错。特别是当该任务需要反复执行或所包含的命令有很多的时候，这种采用逐条输入命令执行的方式几乎是不可行的，这时就应采用程序设计的方式来解决问题。

程序是能够完成一定任务的命令的有序集合。这组命令被存放在一个称为程序文件或命令文件的文本文件中。当运行程序时，系统会根据一定的次序自动执行包含在程序文件中的命令。这种方式与在命令窗口内逐条输入命令相比，具有如下优点：

⑴ 可以有效地利用编辑器，方便地输入、修改和保存程序。

⑵ 可以用多种方式、多次运行程序。

⑶ 可以在一个程序中调用另一个程序。

随着程序设计技术的发展，程序设计语言也变得越来越高级。总体分为两大类：面向过程的程序设计和面向对象的程序设计。它们是两种不同的应用程序设计方法，目前，我们基本上都在使用面向对象的程序设计方法进行应用程序的设计，但是它继承了面向过程程序设计方法的一些思想和方法。

所谓面向过程的程序设计方法，是用结构化程序设计语句来编写程序。它把一个复杂的程序分解成若干个较小的过程，每个过程都可以单独地设计、修改、调试。其程序流程完全由程序员控制，用户只能按照程序员设计好的程序处理问题。

面向对象的程序设计 OOP (Object - Oriented Programming) 是近年来新起的一种编程方法，它克服了面向过程的程序设计方法的缺陷，是目前程序设计方法的主流，也是程序设计在思维和方法上的一次巨大进步。面向对象的程序设计方法是以对象的数据结构为中心，而不是以过程和操作为中心。在这种程序设计方法中，用对象表现事物，用消息传递表现事物之间的联系，用方法表现处理事物的过程。其基本特征是封装性、继承性和多态性。因此，在利用这种技术进行应用程序的设计过程中，工作的重点不再是单纯考虑从代码的第一行到最后一行的程序的编写，而是将考虑的重点放在如何创建对象、如何利用对象简化程序设计、提供代码的可用性等方面。

VFP 经历了从 Dbase、FoxBase、FoxPro、Visual FoxPro，编程的方式和思路也发生了很大的改变。在 VFP 的程序设计中将过程化程序设计与面向对象程序设计结合在一起，帮助用户创建出功能强大、灵活多变的应用程序。从概念上讲，程序设计就是为了完成某一具体任务而编写一系列指令；从深层次来看，VFP 程序设计涉及对存储数据的操作。

§6.2　VFP 程序设计步骤

应用程序是为了完成某项任务所需执行的命令序列，这些命令按照一定的结构有机地组合在一起，并以文件的形式存储在磁盘上，程序执行时，按照其文件说明由磁盘调入内存，从第一条命令开始连续执行直至完毕。其命令序列在执行过程中，一般不需要人为干预。

VFP 语言规定了编写 VFP 程序的规则，VFP 程序在不同的开发阶段有不同的存在形式和用途：

源程序：也称为源代码，或简称程序，以文本文件的格式保存，默认文件扩展名为 .Prg。因此，可以使用各种文本编辑程序建立或修改源程序，并在 VFP 中直接运行。VFP 源程序是用户编写和修改程序的惟一形式。

目标程序：也称为程序的中间文件，通过编译源程序产生，以二进制文件格式保存，文件扩展名为 .Fxp，不可显示和编辑，不可运行，但可以连编为应用程序。一个源程序经编译产生一个对应的目标程序。

执行程序：连编项目所属的所有目标程序将产生一个可以运行的程序。

VFP 允许用户根据需要选择连编产生的结果，即产生只能在 VFP 环境下运行的默认扩展名为. App 的应用程序文件，或者产生在操作系统环境下运行的扩展名为. Exe 的可执行程序文件。两种连编结果均可以运行，但运行时需要的环境不同。

VFP 应用程序一般由以下几部分组成：

- 前言：多为一组注释语句，用以指出程序的名称、功能、作者以及编辑修改的有关信息。
- 设置区：该区在前言之后，用以指出设置程序的运行环境，多用 Set 命令设置程序运行时的系统状态和参量初值。
- 程序体：这一部分包含程序功能的所有命令序列，一般包含数据的输入输出、数据的处理以及结果输出等有关命令。
- 整理部分：每个程序在完成其预定任务之后，都需要做一些整理工作，如关闭各种文件，使系统状态恢复到其标准预定值。
- 程序的退出：程序在做完整理工作后，即可设置有关命令关闭文件返回到系统的命令窗口状态或操作系统状态。

总体来讲，VFP 应用程序设计的步骤包括：应用程序的建立、应用程序的运行和应用程序的修改调试。

6.2.1 应用程序的建立

将应用程序的命令序列依次逐条输入到计算机中，进行必要的编辑，然后存入磁盘的过程称为程序的建立。

1. 菜单方式下应用程序的建立

其操作步骤为如下：

- 在 VFP 的主屏幕画面中单击“文件”菜单中的“新建”选项，弹出“新建”对话框。
- 单击“程序”选项，再按下“新文件”按钮，便进入到程序编辑窗口，如图 6-1 所示。
- 在程序编辑窗口中，输入并编辑程序内容，输入完毕后，可以单击“文件”菜单下的“关闭”选项退出，或按 Ctrl + Esc 键放弃当前编辑的文件退出，也可按 Ctrl + W 或 Ctrl + End 键存盘退出。

2. 命令方式下应用程序的建立

格式：Modify Command < 程序名 >

说明：

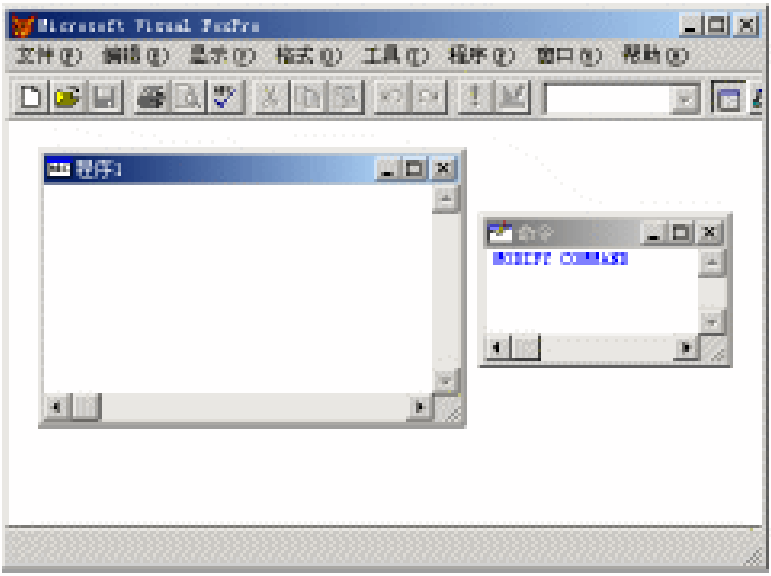


图 6-1　程序编辑窗口

＜程序名＞为主文件名，其扩展名的默认值为. Prg。

当在＜程序名＞中键入新文件名时，则是建立一个新命令文件。

例 1　在 hvfp 文件夹中建立名为 vfp601. Prg 的应用程序，使其能够打开表文件“hvfp h学生表. Dbf”，根据键入的学生姓名在该表文件中查找并输出该学生的基本情况，然后关闭该文件。在命令窗口中键入如下命令：

```
Modify Command    hvfp hvfp601. Prg
```

打开程序编辑窗口，输入应用程序的内容如下：

```
* vfp601. Prg
Set Talk Off
Use    hvfp h学生表. Dbf
Accept "请输入学生姓名：" To xm
Locate For 姓名 = xm
?学号，姓名，性别，班级
Use
Set Talk On
Return
```

程序内容输入完毕，按 Ctrl + W 键存盘退出。

6.2.2 应用程序的运行

1. 菜单方式下应用程序的运行

单击“程序”菜单下的“运行”选项，显示“运行”窗口，如图 6-2 所示。

在“执行文件”文本框中输入要执行的应用程序名。

单击执行窗口中的“运行”按钮，即可执行该应用程序。

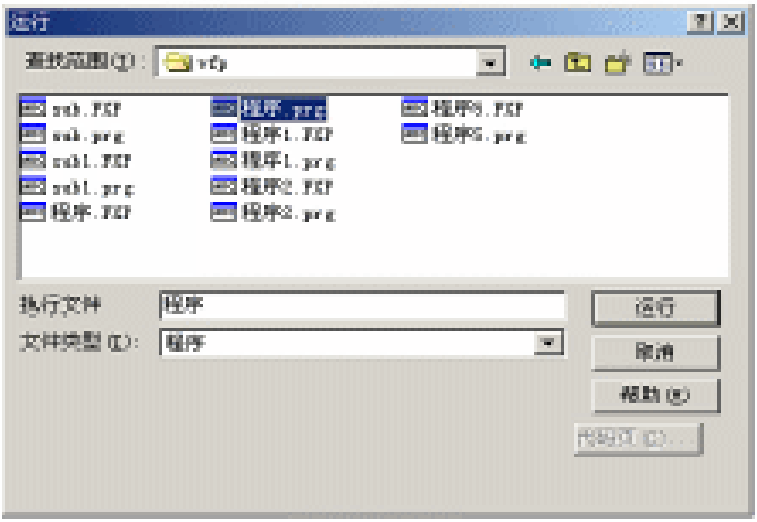


图 6-2 “运行”窗口

2. 在命令方式下应用程序的运行

格式：Do ＜应用程序名＞ [With ＜参数表＞]

功能：将指定文件调入内存并运行。

说明：

- 文件名中扩展名的默认值为. Prg。
- 可选项 With ＜参数表＞的功能将在以后进行介绍。
- 在程序运行过程中执行到下述命令时将会中断运行而退出。它们退出后返回的系统状态分别是：

▲ Cancel：命令窗口状态。

▲ Quit：操作系统状态。

▲　Return：调用点或命令窗口状态。

在程序运行过程中，若按下 Esc 键可使运行强行中断。这时系统会显示警告，并给用户 3 种选择：

▲　取消：中断程序的运行，此为缺省选项。

▲　继续执行：忽略 Esc 的中断作用，继续程序的执行。

▲　挂起：暂时中断程序的运行，返回命令窗口。当再次运行时可以从中断处继续运行。

6.2.3　应用程序的修改

1. 菜单方式下应用程序的修改

使用菜单方式修改应用程序的步骤如下：

✧ 使用“文件”菜单中的“打开”选项或单击常用工具栏中的“打开”按钮，出现“打开”对话框。

✧ 在“打开”对话框中，选定程序所在的文件夹及文件名，单击“确定”按钮，则需要修改的应用程序便出现在编辑窗口中。

✧ 在编辑窗口中对应用程序作必要的修改，然后使用“文件”菜单下的“保存”选项，或单击常用工具栏中的“保存”按钮，保存修改后的程序即可。

2. 命令方式下应用程序的修改

命令格式与建立应用程序的命令相同。

系统执行本命令后，将指定的文件从磁盘调入内存并显示在程序编辑窗口中。这时，用户就可以用全屏幕编辑方式对程序进行编辑修改。修改完毕，按下 Ctrl + W 或 Ctrl + End 键存盘退出。同时系统自动将修改前的程序内容复制到一个同名的备份文件（.Bak）中。若不打算将修改后的程序内容重新存盘，就按下 Esc 键，即放弃本次修改操作，这时程序内容仍然保留修改前的状态。

§6.3　VFP 程序设计语言基础

6.3.1　VFP 语言的特点

VFP 语言与其他程序设计语言（如 C、C + +、BASIC 等）相比，具有以下的特点：

1. 语言类型

VFP 语言既是一种解释型语言，也是一种编译型语言。VFP 的源程序可以直接运行而不必经过编译和连编生成执行程序后运行。例如，在“命令”窗口中单步执行 VFP 的命令，或者在项目管理器中，选择一个源程序，然后单击“运行”按钮，则一次运行程序的所有命令。这种运行程序的方法称之为解释型执行程序。这种方法适合于结构和功能简单、代码量小、与其他程序关系不紧密的程序运行。通常我们将程序中的命令称之为命令语句或简称语句，有些命令语句不能在命令窗口中执行（如 If_EndIf 等）。

要开发较复杂的 VFP 应用程序，一般需要许多的源程序。必须分别编译多个源程序，以得到对应的目标程序，然后将其连编（类似于传统语言的连接）成为一个应用程序后运行。各个程序文件在应用程序运行过程中根据需要被调用执行。

2. 程序结构

在 VFP 程序中既可使用结构化的程序结构，也可使用面向对象的程序结构。结构化的程序段中可以包含面向对象的程序段，面向对象的程序段中也可以包含结构化的程序段。这样编写的程序可以同时得到两种程序结构编程的优点，弥补了单独使用其中一种编程方法的不足。

3. 程序功能

由于 VFP 提供了大量的命令、系统变量和函数以及预装的类库，使得程序在进行与数据库应用有关的各种处理方面显得非常方便和完善。主要的程序功能可以概括为：

- ▲ 具有传统编程语言的数据库的数据处理功能
- ▲ 具有面向对象语言的类、对象和类库功能
- ▲ 具有对多种数据库的多方面的处理功能
- ▲ 具有面向数据库应用的用户界面处理功能
- ▲ 具有国际标准的 SQL 语言命令调用功能
- ▲ 具有 Windows 98 及 Windows 2000 系统管理和文件管理功能
- ▲ 具有 OLE 控件和 ActiveX 控件的调用功能

4. 兼容性

在源程序结构方面，VFP6 与以前的版本兼容，可以编译、运行 VFP3. X、FoxPro2. X 的源程序 在数据库访问接口方面，通过 VFP 提供的 SQL 命令实现主要的 SQL 功能 在数据库接口方面，可以使用 VFP 以前的版本的数据库文件、各种 xBASE 数据库的. Dbf 文件、Excel 电子表数据文件、Lotus1-2-3 电子表

数据文件、Paradox 数据库文件和文本文件的数据;可以将数据输出到上述文件以及 Oracle 数据库中。

6.3.2　源程序的组成

VFP 的源代码以程序行为单位,每行的结束标志是回车换行符。

程序的行分为语句行和注释行两种。命令语句行执行命令功能,注释行只起说明作用,在程序运行时被忽略。注释行的标志是行首为“*”字符。一个命令语句行最多只能容纳一条命令,而一条命令却可以占据多个程序行。在命令行的尾部,以“&&”字符串开始的部分也为程序的注释部分。

VFP 命令语句的长度几乎不受限制,当一个命令语句过长,不利于显示和编辑时可以分为多个命令行;除最后一行外,其他行都要在行尾加分号“;”表示续行。在编译或执行命令语句时,分号及其后的回车换行符将被忽略。

1. 源程序的开头部分

不带参数的程序,可以用任何语句开头,而带参数的程序必须以 Parameter <参数表>语句开头。VFP 最多可以定义 27 个参数,也可以不定义任何参数。

Parameter 语句必须放在过程或函数的首行,且 <参数表> 中的参数的个数不能少于程序调用命令 Do 中 With <参数表> 中的个数,两条命令语句中 <参数表> 中对应位置上的参数的数据类型必须匹配。若 With <参数表> 中有内存变量,则过程执行结束返回时,将把更新后的新值返回给上级调用者。过程返回值的含义表现于此。

2. 源程序的结束部分

结束部分可以有,也可以没有。当没有结束部分时,程序在最后一个语句之后自动作结束处理。但在很多情况下需要有结束部分,其理由是:

▲　有时希望在程序中部的某一处结束执行,而不在程序的结尾处结束。

▲　希望本程序结束后有不同的去向。

控制源程序结束的主要语句及功能如下:

Return 命令的功能:将程序的控制权返回给调用的程序中。如果是在程序中调用的程序或过程,则调用后,程序的控制转向执行调用语句的下一条语句,否则返回到交互模式下。

Return To Master 命令功能:将程序控制权直接返回给最高层调用程序。

Cancel 命令功能:中断当前 VFP 程序文件的执行。

Quit 命令功能:返回宿主操作系统。

Retry 命令功能：将程序的控制权返回到调用的语句处，即重新执行先前调用该程序或过程的语句处，常用于程序的错误处理操作。

6.3.3 应用程序中的常用命令

一个程序一般包含数据输入、数据处理和数据输出三个部分。数据输入和数据输出代码设计是编写许多程序都要面临的工作。在此首先介绍一些简单的常用的命令，在练习编写简单的、小的程序时是非常有用的。

1. 简单的输入输出命令

(1) Input 命令

格式：Input ［<提示信息>］To <内存变量>

功能：暂停程序的执行，等待用户从键盘上键入表达式并将表达式的值赋给指定的内存变量，待按回车键后，继续运行程序。

说明：

- 如果选用 <提示信息>，则系统会在屏幕上首先显示提示信息的内容，以作为提示（提示信息也可以以字符表达式的形式出现，这时，则系统会在屏幕上首先显示出此字符表达式的值，以此作为提示信息）。
- <内存变量>的类型决定于键入的数据类型，但是不能是 M 型数据。
- 输入的数据可以是常量、变量，也可以是表达式。若是表达式，则系统将先计算出表达式的值，然后将值赋给 <内存变量>。但是不能不输入任何内容直接按回车键，否则系统会给出“句法错”的提示信息。
- 输入字符串时，必须带定界符；输入逻辑常量时，也需带定界符（如 .T.，.F.）；输入日期时间常量时需使用花括号。

例：试运行下列交互输入命令：

```
Input "请输入学生的姓名：" To xm
Input "请输入其班级：" To bj
Input "此人的性别为：" To xb
```

运行上述命令时，屏幕上的提示信息及用户键入的数据如下：

```
请输入学生的姓名："王　刚"
请输入其班级："计算机 00 - 1"
此人的性别为："男"
```

检查上述命令的执行结果：

```
? xm,bj,xb
王　刚　　计算机 00 - 1　　男
```

⑵ Accept 命令

格式：Accept ［<提示信息>］To <内存变量>

功能：暂停程序的运行，等待用户从键盘上输入字符型常量以赋给指定的内存变量。

说明：

- <提示信息>为可选项，它是一个用于提示说明的字符型表达式，当程序执行到该可选项时，系统将会先计算该表达式的值并将此值在屏幕上显示出来，以作为提示。
- 该命令只能接收字符型常量。用户在输入字符型常量时不需加定界符，否则系统会将定界符也作为字符型常量的一部分。
- 本命令能够接收键入的字符的总个数不得超过 254 个。
- 如果不输入任何内容而直接按回车键，则系统会将一个空字符串赋给指定的内存变量。

例：试运行下列交互命令：

```
Accept "请输入需打开的表文件名："  To bm
Accept "请输入学生姓名："  To xm
```

执行上述命令后，屏幕上显示的提示信息及用户键入的数据如下：

```
请输入需打开的表文件名：学生表. Dbf
请输入学生姓名：王　刚
```

检查上述命令执行的结果：

```
?bm,xm
学生表. Dbf　　王　刚
```

若执行上述命令后，屏幕上显示的提示信息及用户键入的数据如下：

```
请输入需打开的表文件名："学生表. Dbf"
请输入学生姓名："王　刚"
```

检查上述命令执行的结果：

```
?bm,xm
"学生表. Dbf "　　"王　刚"
```

从上述两种不同的处理结果可以看出，Accept 命令接受的数据是用户在键盘上输入的全部字符，一个字符串的定界符是不需要且不能键入的。

⑶ Wait 命令

运用 Accept 命令，用户可以为内存变量键入多个字符的数据。但是在某些情况下，只需应答一个字符即可。此时为了减少击键的次数，提高反馈的速度，可以使用系统提供的单字符输入命令，即等待命令。

格式：Wait ［<提示信息>］［To <内存变量>］［Window［At <行>，

<列>]] [Nowait]

[Clear | Noclear] [Timeout <数值表达式>]

功能：暂停程序的执行，等待用户从键盘上单击某键或单击鼠标后继续程序的执行。

说明：

- <提示信息>为可选项，若选用，则先计算出此字符型表达式的值，并在屏幕上显示出来，以作为提示；若未选用，则系统将在屏幕上显示默认的提示信息“按任意键继续…”，以作为提示。
- To <内存变量>也是可选项，它用于接收用户在键盘上键入的单个字符，其类型是字符型。若选用，并且用户按下的是回车键或单击了鼠标，则<内存变量>中保存的将是空串；若未选用可选项，则输入的单字符将不保留。
- 一般情况下，提示信息将被显示在 VFP 主窗口或当前用户自定义窗口中。但若指定了 Window 子句，则会出现在一个 Wait 提示窗口，用于提示信息。提示窗口一般定位于主窗口的右上角，也可选用 At 子句，以指定提示窗口在主窗口中的位置。
- 若选用 [Nowait] 子句，则系统将不等待用户按键，直接往下执行。
- 若选用 [Noclear]，则不关闭提示窗口，直到用户执行下一条 Wait Window 命令或 Wait Clear 命令为止。
- Timeout 子句用于设定等待的时间（以秒为单位）。一旦超过规定的时间，则将不等待用户按键，系统自动往下执行。

例 1　试运行下面的 Wait 命令。

```
Wait "输入无效，请重新输入…" Window Timeout 5
```

执行上述命令，屏幕上的显示和执行情况如下：

命令执行时，在主窗口右上角出现一个提示窗口，其中显示“输入无效，请重新输入…”之后，程序暂停执行。当用户按任意键或等待的时间超过 5 秒时，提示 5 窗口关闭，程序继续执行（图 6-3）。

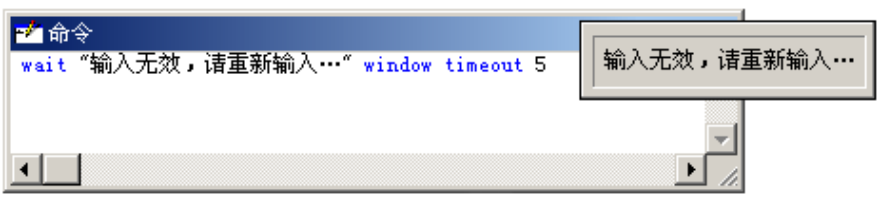


图 6-3　Wait 命令执行结果

本命令的特点：不仅在人机交互应答时可减少击键次数，提高响应速度，而且在程序的多个输出语句之间，适当地插入本命令可使运行结果分段显示出来，等待击键干预后再继续执行程序。

例 2　试编写一个程序，实现能够根据键入的系的代号（单字符）在不同系的档案表文件中查找某一职工的情况。

```
* vfp602.prg
Set Talk Off
Clear
Wait "请输入系的代号：" To dh
xh = "dh" + dh
Use &xh
Accept "请输入职工的姓名：" To xm
Locate For 姓名 = xm
?"职工姓名：" + 姓名
?"性别：" + 性别
?"年龄：" + Str (年龄,2)
……
Wait
?"职称：" + 职称
?"任教课程：" + 任教课程
?"教研室：" + 教研室
……
Wait
Set Talk On
Return
```

2. 常用的辅助命令

在 VFP 程序代码中，经常需对程序进行一些辅助性的处理，如程序代码的提示说明、保证程序正常运行的环境配置等等，它们对整个程序的正常运行起着重要的作用。

(1) 注释命令

在程序代码中，往往要对本程序的名称、功能作一些说明，或对某些语句作一些解释性的注释说明，以帮助其他用户读者了解程序的结构及命令的功能，提高程序的可读性或作为程序设计人员的备忘标记。

格式 1：Note　<注释内容>

格式 2：*　<注释内容>

功能：作为一个独立的语句行注明程序的名称、功能、提示说明或其他备忘标记。

例：

```
Note 本程序将配置应用系统所需的环境
……
```

格式 3：…… && < 注释内容 >

功能：置于某命令行之后，是系统中惟一能够与另一条命令写在同一自然行中的命令，用于注明本命令的意义、功能、说明信息或特殊信息。

例：

```
Set Console Off  && 取消窗口的输出显示
……
```

说明：

- 注释命令是非执行语句。系统执行到该语句时，不做任何响应的操作。
- < 注释内容 > 可以是系统能够输出的任何字符，但不得置于任何定界符内。
- 若注释内容的最后一个字符是分号（;），则系统将认为下一行内容仍属注释内容。
- 不能在命令语句行续行的分号后面加入 && 和注释。

(2) 环境命令

为保证应用程序的正常运行，需要为其设置一定的运行环境。VFP 系统提供了 Set 命令，就是用来设置程序运行环境状态的。这些命令相当于一个状态转换开关，当命令置为“On”时，开启指定的某种状态；而置为“Off”时则关闭该种状态。常用的 Set 命令见表 6-1，其中大写的“ON”或“OFF”是该命令的系统默认值。

表 6-1 常用的环境设置命令

设　置	说　明
Set Talk ON/Off	设置是否将所有命令的执行结果显示到主窗口的状态栏中
Set Console ON/Off	设置是否将输出信息在窗口上显示
Set Printer On/OFF	设置是否在打印机上输出信息
Set Safety ON/Off	设置在改写文件时，VFP 是否显示对话框以确认改写有效
Set Heading ON/Off	设置在执行 List、Display 等命令时是否显示字段名
Set Status ON/Off	设置是否显示屏幕下端的状态行 Set Default To < 盘符 > 设置默认的驱动器
Set Device To Screen/Printer	将输出信息发送到 VFP 的窗口屏幕或打印机

其中 Set Talk ON/Off 是应用程序中使用频率最高的一条辅助命令，它的系统默认值为“ON”，在此状态下，系统每执行一条命令，几乎都要在屏幕上显示命令执行的结果，这种环境虽然便于用户及时了解程序运行情况，但却使屏幕显示的画面比较混乱，而且降低了程序运行速度。因此多在程序的首部设置一条 Set Talk Off 命令用来关闭人机对话状态，而在程序的尾部再设置一条 Set Talk On 命令以恢复系统默认值。

应当注意的是，Set 命令不仅可以用于程序中的命令语句，同时也可以在命令窗口中作为单命令交互执行。其他的 Set 命令的格式与功能请参阅有关书籍。

⑶ 清除命令

格式 1：Clear

功能：清除当前屏幕上的所有信息，并将光标置于屏幕的左上角，同时从内存中释放指定项。

格式 2：Clear All

功能：关闭所有文件，释放所有内存变量，将当前工作区置于 1 号工作区。

格式 3：Clear Typethead

功能：清除键盘缓冲区，以便正确地接收用户键入的数据。

⑷ 关闭文件命令

格式 1：Close All

功能：关闭所有工作区中已打开的数据库、表以及索引文件，将工作区置于 1 号工作区。

格式 2：Close <文件类型>

功能：关闭 <文件类型> 指定的所有文件。

说明：<文件类型> 的可选标识符及其所代表的文件类型，如表 6-2 所示。

表 6-2　文件类型

设　置	说　明
Database	数据库文件、索引文件、格式文件
Index	当前工作区的索引文件
Format	当前工作区的格式文件
Procedure	当前工作区的过程文件
Alternate	文本输出文件

⑸ 运行中断和结束命令

VFP 的应用程序可以根据需要中断运行返回到命令窗口状态，返回到操作系统，也可以返回到调用它的上一级程序或最高级主程序。

格式 1：Quit

功能：关闭所有文件，结束当前 VFP 工作器，将控制权交还给操作系统。

格式 2：Cancel

功能：中断程序运行，关闭所有文件，释放所有局部变量，将控制权交还给命令窗口或操作系统。

格式 3：Return [To master]

功能：将程序控制权交还给调用它的上一级程序或最高一级的主程序。

格式 4：Release <THISFORM>

功能：终止表单的运行。

说明：

- 在 FoxPro 的早期版本中，终止命令程序的执行一般使用 Cancel 语句命令，而在 VFP 中，Cancel 命令不能终止表单的运行，要终止表单的运行可以使用 Release 语句或 Release 方法。
- Release 语句不会激发 QueryUnload 事件，而直接激发 Unload 事件从内存中释放表单或表单集。

⑹ 文本显示命令

格式：Text

<文本内容>

Endtext

功能：将 <文本内容> 原样显示输出。

说明：

- <文本内容> 中的信息无论表面上是什么，系统都将它们作为文本内容的提示信息，而不将它们作为特定的语句处理。

例：

……

```
Text
  x = 2
  y = "华东交通大学"
  ? x,y
Endtext
```

……

执行上述程序,屏幕上的显示结果如下:

x = 2

y = "华东交通大学"

? x,y

从以上的执行结果可以看出,< 文本内容 > 中的所有内容都被看成是普通的字符信息,不会因它表面上是一条可执行命令语句而执行响应的操作。

(7) 定位输出命令

格式:@ < 行,列 > Say < 数据 >

功能:该命令功能齐全、容易使用,是旧版本 FoxPro 中使用频率最高的命令之一,不过在 VPF 中则基本上被 Label 控件取代。它主要是确定在指定的行、列位置输出 < 数据 > 。

§6.4　顺序结构

VFP 系统提供的命令非常丰富,且功能强大,将这些命令和一些程序设计语句有效地组织在一起,形成实现某一特定功能的程序,就能更充分地体现 VFP 系统的特点。

VFP 系统的程序有两个特点:一是程序控制流模式,由顺序、选择分支、循环三种基本结构构成。每一个基本结构可以包含一个或多个语句。二是面向对象可视化的结构程序模块,在每个模块的内部也是由程序控制流组成。

程序结构是指程序中命令或语句执行的流程结构。VFP 语言体现了结构化程序设计的基本特征,它的语句没有标号,也没有无条件转移语句,程序的控制是靠逻辑结构来决定的。VFP 程序的结构有三种:顺序结构、选择 (分支) 结构和循环结构。

顺序结构是在程序执行时,根据程序中语句的书写顺序依次执行的命令序列。VFP 系统中的大多数命令都可以作为顺序结构中的语句。

例 3　试在表文件中查看某学生的有关情况。

程序清单如下:

```
* vfp603. prg
Set Talk Off
形 Clear
Use 学生表. dbf
Locate For 姓名 = "王　刚"
```

```
Display
Use
Set Talk On
Return
```

例 4　根据键入的基本人口数量 a 和人口的年增长率 b，计算 20 年后人口总数 c。

程序清单如下：

```
* vfp604.prg
Set Talk Off
Input "基本人口数量 a = ：" To a
Input "年平均增长率 b = ：" To b
c = a * (1 + b) * *20
?"20 年后的人口总数 c = " + Str (c,10)
Set Talk On
Return
```

运行本程序：

```
Do vfp604.prg
基本人口数量 a = ：1.2E +08
年平均增长率 b = ：0.05
20 年后的人口总数 c = 318395725
```

例 5　如下程序完成将一个数据库表打开并显示该数据库表的结构。

程序清单如下：

```
* vfp605.prg
Set Talk Off                    && 关闭系统对话
Use data.dbf                    && 打开数据库表
Clear                           && 清除屏幕
Display Structure               && 显示数据库表结构
Wait "请按下任意一键返回"       && 等待用户响应
Clear                           && 清除屏幕
Set Talk On                     && 打开系统对话
Return                          && 返回调用程序
```

从以上程序中可以看出，程序的顺序结构设计并不需要特殊的编程技巧，只需将所需完成的工作一步一步地分析清楚，再将其翻译成相应的 VFP 命令即可。

§6.5　选择(分支)结构

一般情况下,应用程序在进行数据处理时需要根据不同的条件采用不同的操作。在 VFP 程序中引用了判断或选择命令之后,程序的流向既可在一部分区间按照语句排列的顺序执行,也可在一定条件下从一个语句跳跃到另一个语句,从一个段落转移到另一个段落,即形成分支结构程序。

所谓分支结构程序,就是按照一定条件由判断语句或选择语句构成的双重或多重流向的程序。分支结构可分成双分支和多分支两种不同的结构形式,分别由 If 语句和 Do Case 语句实现。

6.5.1　双分支结构

双分支结构是分支结构的基本形式,它包括两种不同的处理情况:

1. 单分支结构(简单分支)

格式:If <条件表达式>
　　<语句序列>
　Endif

功能:首先计算 <条件表达式> 的值,然后对其值进行判断,若其值为真(.t.),则顺序执行 <语句序列>;若其值为假(.f.),则跳过 <语句序列>(即不执行 <语句序列>),执行 Endif 语句之后的后续语句(参见图 6-4)。

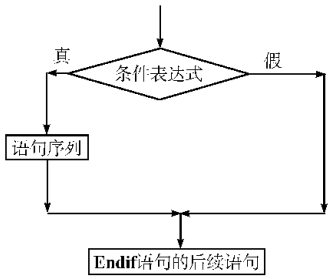


图 6-4　单分支结构流程图

说明:

- If、Endif 必须各占一行,且 If 与 Endif 必须配对出现。
- <语句序列> 中还可以出现另一个分支结构,即出现分支的嵌套,但是每次嵌套中的 If 语句必须与相应的 Endif 成对出现。

例 6　某地至北京的邮路里程为 1 043km,通过邮政局向北京城区寄交

“特快专递”邮件，应在 24 小时内到达，计费标准每克为 0.05 元，但超过 100 克后，超出数每克为 0.02 元。试编写程序实现计算邮费（参见图 6-5、图 6-6）。

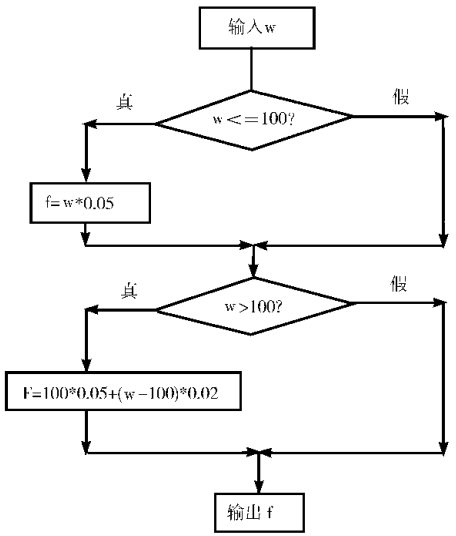


图 6-5　例 6 程序流程图

首先根据题意可得到邮费的计算公式：

$$f=\begin{cases} w*0.05 & (w<=100) \\ 100*0.05+(w-100)*0.02 & (w>100) \end{cases}$$

程序清单如下：

```
* vfp606.prg
Set Talk Off
Input "请输入邮件重量 w = :" To w
If w <= 100
   f = w * 0.05
Endif
If w > 100
   f = 100 * 0.05 + (w - 100) * 0.02
Endif
?"邮费为 :",f
Set Talk On
```

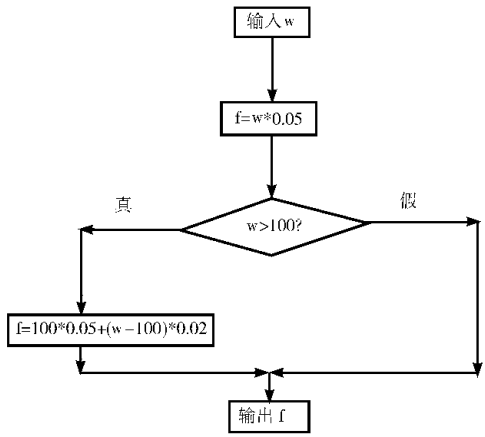


图 6-6　例 6 程序流程图

```
Return
Set Talk Off
Input "请输入邮件重量 w = :" To w
f = w * 0.05
If w > 100
   f = 100 * 0.05 + (w - 100) * 0.02
Endif
?"邮费为 :",f
Set Talk On
Return
```

例 7　修改“职工表. dbf”表中的数据，将编号为“gz05002”的职称由“副教授”改为“教授”。

程序清单如下：

```
* vfp607. prg
Use "f :\vfp 教材\vfp 例题\职工表. dbf"  Exclusive
Browse Last
Locate All For 编号 = "gz05002"
If 职称 = "副教授"
   Replace 职称 With "教授"
Endif
Browse Last
```

```
Use
Return
```

程序运行结果：先显示原数据表中的内容，再显示修改后数据表的内容，若编号为“gz05002”的职工的职称为“副教授”，则自动将职称“副教授”改为“教授”；若此职工的职称不是副教授，则不修改他的职称。

2. **双分支结构**(**参见图** 6-7)

格式：

```
If <条件表达式>
    <语句序列 1>
Else
    <语句序列 2>
Endif
```

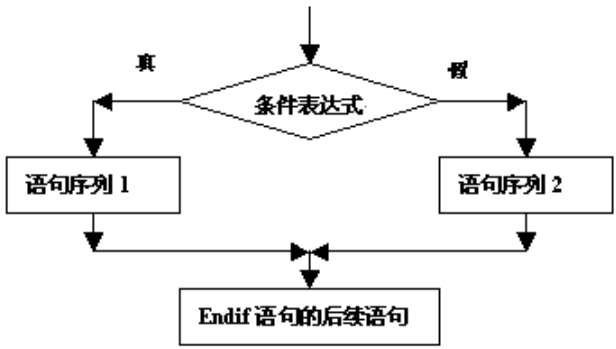


图 6-7　**双分支结构流程图**

功能：先计算 <**条件表达式**>，**然后对其值进行判断，若其值为真**（.t.），**则顺序执行** <**语句序列** 1>，**然后执行** Endif **语句之后的后续语句；若其值为假**（.f.），**则顺序执行** <**语句序列** 2>，**然后执行** Endif **语句之后的后续语句。**

说明：

- **为了养成良好的程序设计风格和习惯，应尽量将** <**语句序列**1> **和** <**语句序列** 2> **采用缩进格式处理。**
- **若必须设立空分支时，应该将它设在选择条件为假的相应分支中。**
- **若** <**语句序列** 1> **和** <**语句序列** 2> **中包含有其他的分支结构（分支结构的嵌套），那么应保证** If、Else、Endif **的准确配对，否则，有可能与设计时的要求不相符。**
- **嵌套必须清楚，不得交叉。**

例 8　在表文件“分数. dbf”中查找学号为 99020202 的学生成绩,若其成绩满 60 分,则显示其学号、课程号和成绩;否则显示此人的学号、课程号并通知其补考(参见图 6-8)。

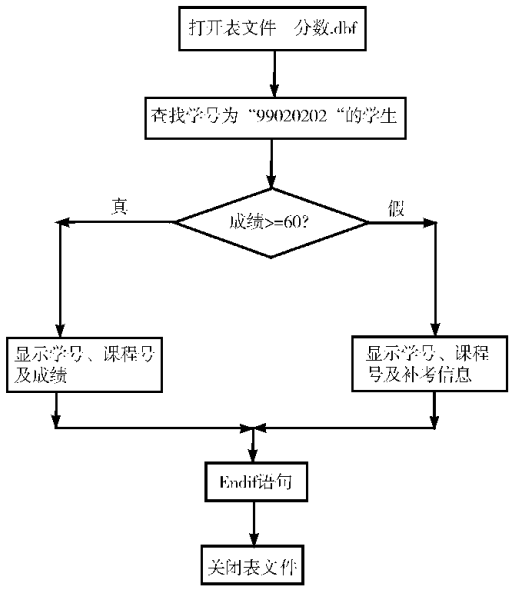


图 6-8　例 8 程序流程图

程序清单如下:

```
* vfp608. prg
Set Talk Off
Use 分数. dbf
Clear
Locate For 学号 = "99020202"
If 成绩 > = 60
   ?"学号 :" + 学号
   ?"课程号 :" + 课程号
   ?"成绩 :" + Str (成绩,6,2)
Else
   ?"学号为" + 学号 + "的学生应参加"
   ??"补考的课程号为 :" + 课程号
Endif
```

```
Use
Set Talk On
Return
```

例 9　利用双分支结构改写例 6（参见图 6-9）。

```
* vfp609. prg
Set Talk Off
```

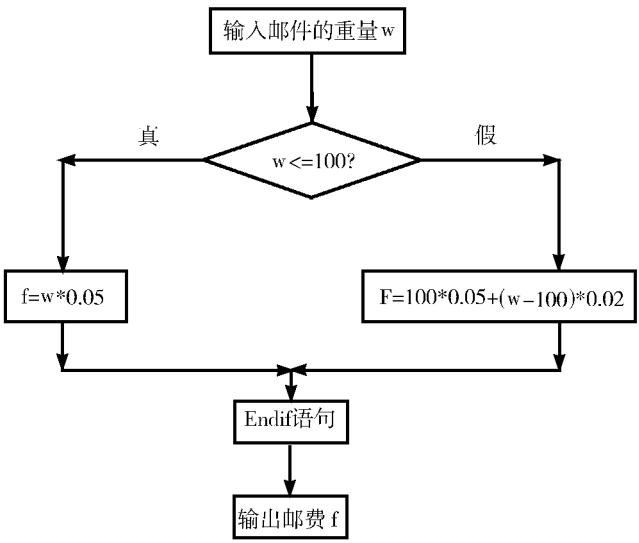


图 6-9　例 9 程序流程图

```
Input "请输入邮件的重量 w = " To w
If w < = 100
  f = w * 0. 05
Else
  f = 100 * 0. 05 + (w - 100) * 0. 02
Endif
? "邮费为:",f
Set Talk On
Return
```

例 10　根据键入的 x 值，计算下面分段函数的值，并显示结果。

$$y=\begin{cases}5x^2+6x-1 & (x\leqslant 0)\\ x^2-4x+1 & (0<x\leqslant 20)\\ 3x^2+1 & (x>20)\end{cases}$$

程序清单如下：

```
* vfp610. prg
Set Talk Off
Clear
Input "请输入 x 的值：" To x
If x > 0
  If x > 20
    y = 3 * x 2 + 1
  Else
    y = x * x - 4 * x + 1
  Endif
Else
  y = 5 * x 2 + 6 * x - 1
Endif
?"分段函数的值为："
??Str (y)
Set Talk On
Return
```

例 10 给出了一个 If 双重嵌套的示例。系统允许使用的 If 嵌套有多种形式，但是无论何种形式，都应遵守完全包含的原则，即一个 If 结构应被另一个 If 结构完全包含，不能出现嵌套的交叉。

6.5.2　多分支结构

虽然用条件语句的嵌套结构可以解决程序中的多重选择问题，但是在处理这类问题时，程序的结构显得复杂，层次较多，并且容易出错，使用也不方便。为此，VFP 系统提供了一种多分支结构语句。

格式：

```
Do Case
Case <条件表达式 1 >
<语句序列 1 >
Case <条件表达式 2 >
<语句序列 2 >
……
```

Case <条件表达式 n>
<语句序列 n>
[Otherwise
<语句序列 n+1>]
Endcase

功能：依次判断多个条件表达式，选择执行第一个逻辑值为真的 <条件表达式> 所对应的语句序列。

该语句具体的执行过程是：系统依次判断每个 Case 语句中的 <条件表达式>，遇到第一个逻辑值为真的表达式时，则执行该条件下的语句序列，之后其他语句被忽略，而转去执行 Endcase 语句之后的语句。若所有的 Case 语句之后的条件表达式的值均为假时，在没有可执行选项 Otherwise 语句的情况下，也将执行 Endcase 语句之后的语句；在有可执行选项的情况下，则执行 Otherwise 语句下的语句序列，然后再执行 Endcase 语句以后的语句。其执行过程如图 6-10 所示。

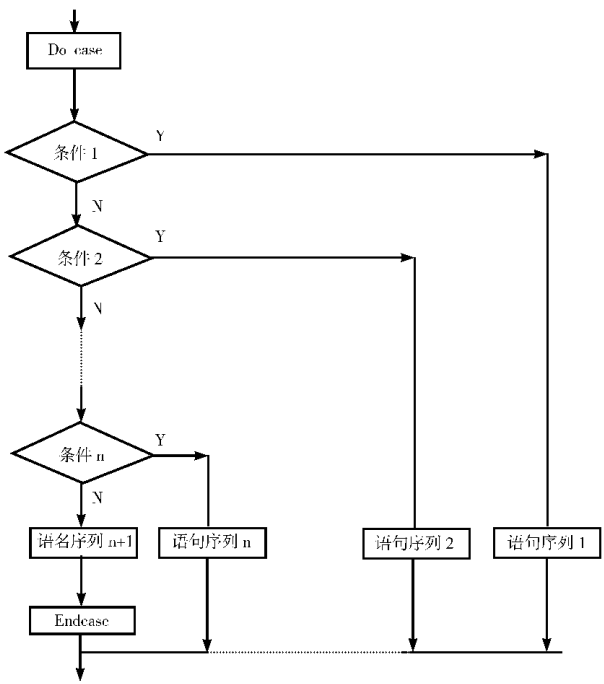


图 6-10 多分支结构执行过程流程图

说明：

- Do Case、Case、Otherwise、Endcase 必须各占一行。每个 Do Case 必须有一个 Endcase 与之对应，即 Do Case 与 Endcase 必须配对出现。
- 各个 < 条件表达式 > 的值必须是一个逻辑值，若 < 条件表达式 > 的值有多个为真时，系统只执行第一个结果为真的条件表达式之下的语句序列。
- 各个语句序列中可以嵌套各种控制结构的命令语句。

例 11　计算分段函数值：

$$y=f(x)=\begin{cases}2x-1 & (x<0)\\ 3x+5 & (0\leqslant x<3)\\ x+1 & (3\leqslant x<5)\\ 5x-3 & (5\leqslant x<10)\\ 7x+2 & (x\geqslant 10)\end{cases}$$

程序清单如下：

```
* fvp611.prg
Set Talk Off
Clear
Input "请输入自变量 x 的值：" To x
Do case
Case x<0
y=2*x-1
Case x<3
y=3*x+5
Case x<5
y=x+1
Case x<10
y=5*x-3
Otherwise
y=7*x+2
Endcase
? "y=f(",x,")=",y
Set Talk On
Return
```

从上例中可以看出：多分支语句可以使程序结构清晰、格式整齐、言简意明、容易编写。同时，摈弃了无条件转移语句，防止了程序的无规则转移，保证了程序结构的规律性，因此它是编写结构化多分支程序很好的工具。

例 12　利用多分支语句结构，编写程序实现显示当前季节。

```
* vfp612.prg
```

```
Set Talk Off
Clear
yue = Month (Date ( ) )   && 获取当前月份
Do Case
Case Inlist (yue,3,4,5)
season = "春"
Case Inlist (yue,6,7,8)
season = "夏"
Case Inlist (yue,9,10,11)
season = "秋"
Case Inlist (yue,12,1,2)
season = "冬"
Endcase
Wait season Window   && 当前季节显示在 Wait 提示窗口内
Set Talk On
Return
```

§6.6 循环结构

在程序设计过程中,经常遇到某一类问题的计算和处理方法完全一样,只是要求重复进行计算多次,而每次使用的数据都按照一定的规律在变化。类似这样的问题,如果使用顺序结构和选择分支结构来进行编程,那么代码将会比较长,效率也将会较低,此时,就应该采用循环结构。

程序设计中的循环结构是指在程序中,从某处开始有规律地反复执行某一段程序的现象。被重复执行的程序段称为循环体,循环体的执行与否及次数多少视循环类型与条件而定。当然,无论何种类型的循环结构,其共同的特点是必须确保循环体的重复执行能被终止(即不能是无限循环)。

VFP 具有一般程序设计语言都有的 While 条件循环语句和 For 计数循环语句,此外还有专门用于对表进行处理的 Scan 扫描循环语句。

6.6.1 条件循环(While 循环)

格式: Do While <条件表达式>
<语句序列>
[Exit]
[Loop]

< 语句序列 >

Enddo

功能：首先计算 < 条件表达式 > 的值，若其值为真（.t.），则执行循环体。遇到循环终端语句（或 Loop），就返回循环起始语句重新计算和判断 < 条件表达式 > 的值，若其值仍为真（.t.），则重复上述操作，直至其值为假（.f.）或遇到（Exit 语句）为止，则退出循环而执行循环终端语句的后续语句。

说明：

- 循环起始语句（Do While < 条件表达式 >）的作用是判断循环的条件是否满足。满足时则执行循环体，否则退出循环。语句中的 < 条件表达式 > 的值必须是逻辑型的数据。
- 循环终端语句的作用是标明循环的终点。它必须与循环起始语句成对出现。
- 循环体是循环程序段的主要组成部分，是被多次重复执行的语句序列，一般用来完成某种功能操作。
- 循环短路语句（Loop）为可选项，其作用是迫使程序不再执行其后至 Enddo 语句之间的语句序列，而返回到循环起始语句使循环流程短路。在程序中必须与判断语句或多分支语句联用。
- 循环断路语句（Exit）也是可选项，其作用是无条件地迫使程序中断而转去执行 Enddo 语句的后续语句。该语句必须与 Loop 一样安排在循环体的某个分支上。

Do While 型循环的执行流程图如图 6-11 所示。

从流程图中可以看出：循环起始语句和终端语句本身均没有改变 < 条件表达式 > 值的功能。因此，仅依靠这两个语句不能保证循环程序正常地退出或终止。例如，在 Do While 语句中将 < 条件表达式 > 设置为逻辑常量.t.，就构成了一个永远都不会自己结束的绝对循环（即死循环）。为了在完成循环体的既定任务之后能自动退出循环，则需在循环体的某个分支中选用 Exit、Cancel 或 Return 等语句。在一般情况下，执行不到这类语句时，循环就会周而复始地运行下去，一旦设置的某种条件满足而执行了该类语句时即可退出循环。这种循环的执行次数是随机的。

图 6-11 中的虚线部分是可选部分。

例 13　计算 $S=\sum_{i=1}^{100} i=1+2+3+\cdots+99+100$

该程序要使用循环结构，解题的思路归纳为两点：

引进变量 i 和 s，s 用来保存累加的结果，初值为 0；i 既作为被累加的数

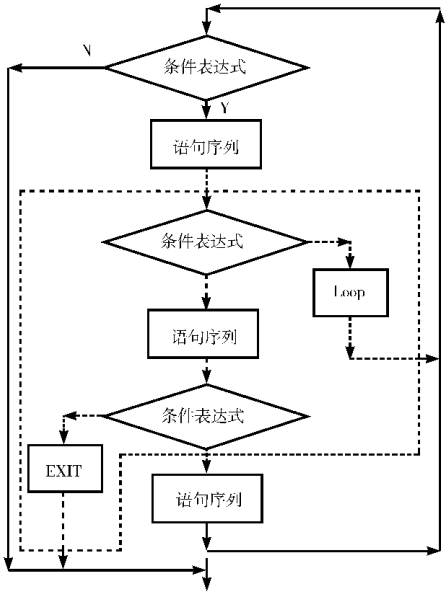


图 6-11 DO While 型循环的执行流程图

据,同时也作为控制循环条件是否成立的变量,初值为 1。

重复执行命令 s = s + i 和 i = i + 1,直至 i 的值超过 100。每一次执行,s 的值增加 i,i 的值增加 1。

程序清单如下:

```
* vfp613. prg
Set Talk Off
Clear
s = 0
i = 1
Do While i < = 100
   s = s + i
   i = i + 1
Enddo
?" s = " ,s
Set Talk On
Return
```

例 14 逐条输出 1982 年出生的学生记录 (假如有一个数据库 xsxk. dbc,

在 xsxk. dbc 中有一个表文件为：学生. dbf)。

方法 1：不使用索引。先用 Locate 命令将记录指针定位于满足条件的第一条记录上，然后进入循环语句。每次执行循环体，先显示当前记录的内容，然后用 Continue 命令将记录指针定位于满足条件的下一条记录上。

程序清单如下：

```
* vfp614. prg - - - 1
Set Talk Off
Clear
Open Database xsxk
Use 学生 In 0
Select 学生
Locate For Year (出生日期) = 1982
Do While . Not. Eof ( )
   Display
   Wait
   Continue
Enddo
Close Database
Set Talk On
Return
```

方法 2：使用索引。先用 Seek 命令将记录指针定位于满足条件的第一条记录上，然后进入循环语句。每次执行循环体，先显示当前记录的内容，然后用 Skip 命令将记录指针移动到下一条记录上。注意，在索引表达式上取值相同的记录 (1982 年出生的记录) 一定是排在一起的。

程序清单如下：

```
* vfp614. prg - - - 2
Set Talk Off
Clear
Open Database xsxk
Use 学生 In 0
Select 学生
Index On Year (出生日期) Tag csnf
Seek 1982
Do While Year (出生日期) = 1982
```

```
    Display
    Wait
    Skip
Enddo
Close Database
Set Talk On
Return
```

例 15　试用 Do While 循环语句设计一个有关档案管理的主控程序，用户通过键盘选择菜单，以实现不同功能模块的调用或退出运行。

程序清单如下：

```
* vfp615. prg
Set Talk Off
Do While . t.
    Clear
    Text    && 文本输出语句开始处
          计算中心档案管理系统
      查询 - - - - - 1                    统计 - - - - - 2
      修改 - - - - - 3                    退出 - - - - - 0
    Endtext
    Wait "请键入功能选择代号（0 ~ 3）：" To dh
    Do Case
Case dh = "1"
    Do 查询        && 调用“查询”子程序
Case dh = "2"
    Do 统计        && 调用“统计”子程序
Case dh = "3"
    Do 修改        && 调用“修改”子程序
Case dh = "0"
    Exit
    Otherwise
    Wait "选择错误，按任意键重新选择！"
EndCase
Enddo
Set Talk On
```

Return

6.6.2　计数型循环 (For 型循环)

在循环结构中,还可以建立固定次数的循环,即所谓计数型循环。构造这种循环的要点是:首先要设置一个循环控制变量,这种变量一般是由数值型内存变量来充当;然后为其设置初值、终值、步长,则循环体的执行次数即被固定。计数型循环可以根据给定的次数重复执行循环体。

格式:For <循环控制变量> = <初值> To <终值> [Step <步长>]
　　<语句序列>
　　[Loop]
　　<语句序列>
　　[Exit]
　　<语句序列>
　　EndFor/Next [注释]

功能:按照设置好的循环变量参数,执行固定次数的循环体的操作。

说明:

- 格式中的 For 语句称为循环说明语句,语句中所设置的初值、终值与步长决定了循环体的执行次数 r。r = Int ((终值 - 初值)/步长) + 1。
- 当步长为 1 时,子句 Step 1 可以省略。
- Endfor (或 Next) 语句成为循环终端语句,其作用是标明循环程序段的终点,同时使循环变量的当前值增加一个步长值。它必须与循环说明语句成对出现。
- 循环体指 For 语句与 Endfor 语句之间被循环执行的语句序列,用来完成多次重复操作功能。
- 循环短路语句 Loop 和循环断路语句 Exit 的作用与在 Do While 型循环中的作用相同。
- 该种循环语句结构的执行过程是:先为循环变量赋初值,然后判断其值是否超过终值;若不超过则执行循环体,遇到循环终端语句使控制变量增加一个步长值;再判断循环变量当前值是否超过终值,不超过时再执行循环体,直至循环体当前值超过终值或执行到 Exit 语句,程序才退出循环执行 Endfor 语句之后的语句。

这里所说的"超出"的含义是:当步长的值为正时,循环控制变量的当前值大于终值;当步长的值为负时,循环控制变量的当前值小于终值。

例 16　从键盘上输入 10 个数据,找出其中的最大值和最小值。

思路：假定已经找出 n - 1 个数中的最大值 max（或最小值 min），现在再读入第 n 个数据 a，那么 a 和 max（或 min）中较大的（或较小的）就是 n 个数据中的最大值（或最小值）。

程序清单如下：

```
* vfp616prg
Set Talk Off
Clear
Input "请输入第 1 个数据：" To a
Store a To Max,min
For i = 2 To 10
   Input "请输入第" + Trim(Str(i)) + "个数" To a
   If max < a
      max = a
   Endif
   If min > a
      min = a
   Endif
Endfor
? "最大值为：",max
? "最小值为：",min
Set Talk Off
Return
```

例 17　用下列级数的前 21 项之和计算自然对数之底 e 的近似值。

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{20!}$$

思路：假设内存变量 e 存放各项累加和，内存变量 t 依次存放 1 ~ 20 的阶乘，内存变量 x 为循环控制变量，根据题意其初值、终值和步长分别设置为 1、20 和 1。

程序清单如下：

```
* vfp617. prg
Set Talk Off
Clear
Store 1 To e,t
For x = 1 To 20
```

```
    t = t * x
  e = e + 1/t
  Endfor
  ? "该级数前 21 项之和为:" + Str(e,7,5)
  Set Talk On
  Return
```

例 18　Exit **的应用。该程序在执行过程中判断数据表记录指针是否已经指向尾部,如果是,则无条件地退出循环而不去理会计数是否完成。**

程序清单如下:

```
  * vfp618.prg
  Set Talk Off
  Use 学生.dbf
  Go Top
  For i = 1 To 8000
    If Eof()
      Exit
    Endif
    Goto i
    Display
  Endfor
  Use
  Set Talk On
  Return
```

例 19　Loop **的应用。该程序在执行过程中如果发现学生是 1982 年出生的,则不显示任何信息,直到循环执行完毕。**

程序清单如下:

```
  * vfp619.prg
  Set Talk Off
  Clear
  Use 学生.dbf
  Go Top
  For i = 1 To Count()
    If Year(出生日期) = 1982
      Loop
```

```
    Endif
    Display
  Endfor
  Use
  Set Talk On
  Return
```

6.6.3 指针型循环 (Scan 型)

“指针”型循环控制语句,即根据用户设置的表中的当前记录指针,决定循环体内语句的执行次数。这是一种专门用于数据处理的循环命令,不仅可以缩短程序,节约存储空间,而且可以提高程序的执行速度。

格式: Scan [<范围>] [For <条件表达式 1>] [While <条件表达式 2>]

<语句序列>
[Loop]
<语句序列>
[Exit]
<语句序列>
EndScan [注释]

功能:在当前表中从首记录开始自动、逐个移动记录指针扫描全部记录,对于符合条件的记录执行循环体规定的操作。

说明:

- **在循环起始语句** Scan **中,[<范围>]子句指明了扫描记录的范围,其默认值为** All **;**For **子句说明只对使** <条件表达式 1> **的值为真的记录进行相应的操作;**While **子句则指定只对使** <条件表达式 2> **的值为真的记录进行相应的操作,直至使其值为假的记录为止,即不再执行本程序。**
- **循环终端语句** Endscan **标明了循环程序段的结束,它必须与** Scan **语句成对出现。**
- **循环体中的语句说明了在扫描记录时,对于符合条件的记录所进行的功能操作。其中的** Loop **语句和** Exit **语句与其他循环结构中的该语句的作用相同。**

Scan **语句循环执行的过程是:** Scan **语句执行时,按记录范围、**While **和** For **优先次序判断,在限定范围内和条件为真时,执行该语句序列,遇到** End-scan **语句时,记录指针指向记录范围内的下一条记录,程序控制返回** Scan **语**

句继续进行判断，当记录指针超出记录范围到达文件尾部时，转而执行 Endscan 语句之后的语句命令。

例 20　统计专业代码（学号的第 3、4 位）为 03 的男、女学生人数。

程序清单如下：

```
* vfp620.prg
Set Talk Off
Clear
Open Database xsxk
Use 学生 In 0
Select 学生
Store 0 To m,w
Scan For Subs(学号,3,2) = "03"
    If 性别 = "男"
       m = m + 1
    Else
       w = w + 1
    Endif
Endscan
? "男生人数:",m
? "女生人数:",w
Close Database
Return
```

“指针”型循环结构的特点在于它能够自动、依次移动记录指针，当程序执行到 Endscan 或 Loop 语句时，会对条件表达式进行判断，若条件成立则自动将记录指针移到下一个符合条件的记录上。如：

```
Use 学生.dbf
Scan For 性别 = "女"
    Display
Endscan
```

其功能等价于下述 Do While 型循环结构的程序：

```
Use 学生.dbf
Go Top
Do While 性别 = "女" .And. !Eof()
    Display
```

```
    Skip
Enddo
```

通过比较可以发现：后者需用一个 Go Top 命令将记录指针移到表的顶部，然后才能从首记录依次向后扫描，否则将从当前记录开始扫描而丢掉前面的记录；同时还必须使用 Skip 命令将记录指针逐个向后移动，而这些命令在“指针”型循环结构中均不需要。

6.6.4 多重循环结构

若一个循环结构的循环体中完整地包含另一个循环结构，则称此循环结构为循环的嵌套。较为复杂的问题往往需要用多重循环来处理。下面为使用循环嵌套关系的示例。

示例 1：

```
Do While <条件表达式 1>
    …
    Do While <条件表达式 2>
       …
    Enddo
    …
Enddo
```

示例 2：

```
Do While <条件表达式>
    …
    For <循环控制变量> = <初值> To <终值> [<Step 步长>]
        …
    Endfor
    …
Enddo
```

示例 3：

```
Do While <条件表达式 1>
    …
    If <条件表达式 2>
        …
    Else
        …
```

```
    Endif
    …
Enddo
```

说明：

在 VFP 系统中，循环的嵌套层次不限，但内层循环的所有语句必须完全被包含在外层循环之中。否则就会出现循环的交叉，造成逻辑上的混乱。

循环结构与选择分支结构允许混合嵌套使用，但不允许交叉。其入口语句与其相应的出口语句必须成对出现。

下面的使用方法是错误的：

示例 1：

```
Do While <条件表达式>
    …
    For <循环控制变量> = <初值> To <终值> [<Step 步长>]
        …
    Enddo
    …
Endfor
```

示例 2：

```
Do While <条件表达式 1>
    …
    If <条件表达式 2>
        …
    Else
        …
    Enddo
    …
Endif
```

例 21　试用固定次数的循环的嵌套格式编写一个打印九九乘法口诀表的程序。

设置内存变量 x、y 和 z 分别存放乘法口诀表中的被乘数、乘数和乘积。

程序清单如下：

```
* vfp621.prg
Set Talk Off
```

```
Clear
For x = 1 To 9
    For y = 1 To 9
        Z = x * y
        ?? Str(x,1) + " * " + Str(y,1) + " = " + Str(z,2) + Space(2)
    Endfor y
    ?
Endfor x
Set Talk On
Return
```

例 22　输出 3 ~ 100 之间的所有素数。

思路：要判断一个数 m 是否为素数，最直观的方法是：用 3 到 m - 1 之间的各个整数一个一个地去除 m，若除不尽，则 m 就是素数；只要有一个能整除，则 m 就不是素数。若需提高效率，就不必除到 m - 1，只需除到 Int (Sqrt (m)) 即可。

程序清单如下：

```
* vfp622.prg
Set Talk Off
Clear
For m = 3 To 100 Step 2
    n = Int(Sqrt(m))
    For i = 3 To n
        If Mod(m,i) = 0
            Exit
        Endif
    Endfor i
    If i > n
        ?? m
    Endif
Endfor m
?
Set Talk On
Return
```

例 23　求出任意一个二维数组中某个元素：此元素在所在行最大、在所

在列最小(此类元素可能不止一个,也可能没有)。对于各种情况都应该输出相应的信息。

思路:考虑数组中的每个元素,先判断它在所在行上是否最大,再判断它在所在列上是否最小。若两次判断都满足,则表明此元素就是符合要求的一个元素。

程序清单如下:

```
* vfp623.prg
Set Talk Off
Clear
* 定义数组并输入数据
Input "m = " To m
Input "n = " To n
Dimension da(m,n)
For i = 1 To m
    For j = 1 To n
        Input "da(" + Str(i,2) + "," + Str(j,2) + ") = " To da(i,j)
    Endfor j
Endfor i
* 处理
num = 0                    && 存放共有多少个元素符合条件
For i = 1 To m             && 考虑每一行
    For j = 1 To n         && 考虑 i 行的每个元素
        For k = 1 To n     && 判断当前元素是否在所在行上最大
            If da(i,j) < da(i,k)
                Exit
            Endif
        Endfor k
        If k < = n         && 若当前元素不是最大,考虑下一个元素
            Loop
        Endif
        For h = 1 To m     && 判断当前元素是否在所在列上最小
            If da(i,j) > da(h,j)
                Exit
            Endif
```

```
            Endfor h
            If h < = m              && 若当前元素不是最小,考虑下一个元素
                Loop
            Endif
            ? "da (" + Str (i,2) + "," + Str (j,2) + ") = ",da (i,j)
            num = num + 1
        Endfor j
    Endfor i
    ? "共有" + Str (num,2) + "个元素符合条件"
    Set Talk On
    Return
```

例 24　编制一个查询学生情况的程序。要求根据给定的学号找出并显示学生的姓名及各门功课的成绩。

程序清单如下：

```
* vfp624. prg
Set Safety Off          && 当产生已存在的文件或索引标记时,直接覆盖,不给出确认对话框
Open Database xsxk
Use 选课. dbf In 0
Use 学生. dbf In 0
Select 选课
Index On 学号 Tag xh
Select 学生
Index On 学号 Tag xh
Set Relation To 学号 Into 选课          && 建立一对多关联
Set Skip To 选课
Do While . t.
    Clear
    Accept "请输入学号 :" To mxh
    Seek mxh
    If ! Eof ()             && 找到学生记录
        mes = "学号 :" + 学号 + "姓名 :" + 姓名
        Do While 选课. 学号 = mxh    && 找出该学生的所有选课记录
            mes = mes + Chr (10) + Chr (13) + "课程号 :" ;
```

```
                + 选课. 课程号 + "成绩 :" + Str (选课. 成绩,5,1)
                Skip
            Enddo
            mes = mes + Chr (10) + Chr (13) + "按任意键继续……"
        Else                    && 未找到学生记录
            mes = "查无此人,按任意键继续……"
        Endif
    * Chr (10) 和 Chr (13) 分别是换行符和回车符,
    * 使用它们的目的是为了让各门功课的信息分行输出
        Wait mes Window
        Wait "继续查询吗 ? (Y/N)" To p
        If Upper (p) < > "Y"
            Exit
        Endif
    Enddo
    Close Database
    Set Safety On
    Return
```

§6.7　过程 (多模块) 程序

在应用系统中,一般是根据实际需要将整个系统划分为若干个模块,然后在主控模块的控制之下,调用各个功能模块以实现系统的各种功能操作。通常将这些可以被调用的功能模块或能完成某种特定功能的独立程序称之为过程或子程序。在 VFP 系统中,程序是独立存放在磁盘上的程序文件,使用时用户通过文件名调用并执行。为了实现用户的某一目的,很可能需要由多个 (子) 程序完成。因此,常常将一个程序文件看成是整个操作的一个过程,而程序文件的建立和调用也可以看成是过程的建立和调用。

6.7.1　过程、函数的定义

在 VFP 系统中,过程是具有特定功能的命令文件,享有与主程序相同的待遇。它可以用 Modify Command 命令建立和修改,用 Do 命令执行,具有相同的扩展名. Prg,并以同样的文件格式存储在磁盘上,所不同的是在每个过程中至少要有一个返回语句。

1. 建立过程和函数文件的命令格式

Modify Command <过程文件名> [.Prg]

2. 返回语句的格式

Return [To Master]

功能：结束过程的运行，使程序的控制权交还到调用它的上一级或最高级主控程序。

说明：

- Return 语句通常作为过程的出口设置在过程的尾部（虽然在一个过程中可以设置多个返回语句以形成多个出口，但是根据结构化程序设计的要求，应尽量将过程设计成单入口和单出口）。
- 可选项 To Master 的作用是使系统流程返回到最高一级的主控程序。
- 出现在主控程序或单一程序末尾处的 Return 语句，将使系统返回到命令窗口状态。在某一程序中设置 Do 语句来运行另一个程序（即过程）称为过程的调用。

3. 过程、函数定义的基本格式

(1) 过程定义的格式

```
Procedure <过程名>
    [Parameters <参数名表>]
    <语句序列>
    [Return]
[Endproc]
```

(2) 函数定义的格式

```
Function <函数名>
    [Parameters <参数名表>]
    <语句序列>
    Return <表达式>
[Endfunc]
```

(3) 说明

- 过程名和函数名是过程或函数的标识，只能由字母、数字或下划线等字符组成，字符的个数不能超过 128 个，并且首字符必须为字母或下划线，不能与程序保留字同名。
- 参数名表中的参数与参数之间以逗号分隔，在过程或函数被调用时，

可以接受调用程序输入的值;在定义过程或函数时,通过 Parameters 语句,指定接受这些输入值的数据类型和顺序。

- 若过程调用语句中有 With <参数表>,则过程中的第一条语句必须是 Parameters 语句。若函数调用时指定自变量,则函数的第一条语句必须是 Parameters 语句。
- 过程的返回无特殊的要求,Return 语句可以出现在语句序列中的任何需要的位置,也可以省略;但函数返回时,Return 语句中需指定 <表达式>,<表达式> 的值是通过执行语句序列产生的结果,并作为定义函数的函数值予以返回。对于省略了 Return 语句的用户自定义函数,其返回值为. t. 。
- 若在过程或函数中未执行 Return 语句时 (可能是因为程序代码中无 Return 语句,也可能是因为程序代码中有 Return 语句但程序执行流程中未执行此语句),如果遇到 Endproc 或 Endfunc 语句,则结束过程或函数的执行。此时如果程序代码中又省略了 Endproc 或 Endfunc 语句,则当过程或函数代码执行至遇到另一个 Procedure 或 Function 语句或程序文件结束时,结束过程或函数的执行。

6.7.2　调用过程、函数

过程是作为一个文件独立存储在磁盘上的,因此,每调用一次过程都要打开一个磁盘文件。一个应用程序如果需要多个过程则要打开多个磁盘文件。这样一来,一则增加了打开文件的个数,再则频繁地访问磁盘势必影响系统的运行速度。为了解决这一矛盾,VFP 系统提供了关于过程文件的概念及有关操作。

过程文件即过程的集合。VFP 在一个过程文件中包含过程的个数未作限制,其中过程用 Procedure <过程名> 给予标识。当某一程序需要调用过程文件中的过程时,只需一次性地打开该过程文件,然后按照过程名调用其中的某个过程,从而大大减少访问磁盘的次数,提高了调用的速度。

过程文件中的过程又称为内部过程,它不能作为一个命令文件而单独存盘或运行。它的入口语句必须是带有其名称的标识语句:Procedure <过程名>。而独立存储在磁盘上的过程文件则与其他文件一样,可以单独存盘和运行 (称为外部过程)。

过程文件的一般格式:

```
Procedure <过程名 1>
    <过程名 1 的语句序列>
Return
```

Procedure <过程名 2>
<过程名 2 的语句序列>
Return
…
…
Procedure <过程名 n>
<过程名 n 的语句序列>
Return

若过程和函数组织在一个过程文件中,则调用过程前首先必须打开过程文件。打开过程文件的语句格式为:

Set Procedure To <过程文件名表> [Additive]

功能:打开命令中<过程文件名表>中指定的各个过程文件。

说明:

- 若不含任何参数,则关闭所有以前用此命令打开的过程文件。
- 若使用了参数 Additive,则在不关闭以前打开的过程文件的同时,打开当前命令指定的过程文件。

1. 调用过程

命令格式:

Do <过程名> [In <程序文件名>] [With <参数表>]

功能:调用<过程名>指定的过程。

Do 命令的具体执行顺序是:

▲ 若 Do 命令中有 In 子句和参数,则 VFP 在"程序文件名"指定的程序文件中查找指定的过程定义,"程序文件名"参数的默认扩展文件名是".Prg",默认路径是当前路径,即 Set Default To 命令设置的路径。

▲ 若 Do 命令中无 In 子句和参数,则 VFP 在 Do 命令所在的程序文件中查找指定名称的过程定义。

▲ 若 Do 命令中无 In 子句和参数,且在 Do 命令所在的程序文件中不存在指定名称的过程定义,则 VFP 在 Set Procedure To 命令打开的过程文件中查找指定名称的过程定义。

▲ 若上述过程中仍未找到指定名称的过程或函数,则 VFP 从最近执行过的程序文件开始,向前搜索每个执行过的程序文件,直至找到为止。

▲ 若仍未找到指定名称的过程或函数的定义,则 VFP 在当前文件夹中的尚未执行的程序文件中搜索,直至找到为止;若仍未找到,则 VFP 报告错误。

说明：

- 过程除了可以使用带有 With 传递参数的 Do 命令调用外，还可以按函数的方式调用。

示例：

设 ss. prg 为一个接受两个传递参数的过程。

按过程调用的形式为：

```
Do ss With a,b
```

按函数方式调用的形式为：

```
= ss (a,b)
```

2. 调用函数

函数作为表达式的一部分，其调用的格式为：

< 函数名 >（< 参数表 >）

说明：

- 函数定义也可以通过 Do 命令调用，但此时，函数定义中 Return < 表达式 > 语句中的函数返回值被忽略。
- 在表达式中调用函数与用 Do 命令调用的区别是直接书写函数名和一对圆括号，圆括号内书写与 Do 命令的 With 子句起同样作用的参数序列。

VFP 调用函数定义的顺序是：

▲　在当前程序文件中查找指定名称的函数定义。

▲　在 Set Procedure To 命令打开的过程文件中查找指定名称的函数定义。

▲　从最近执行过的程序文件开始，向前搜索最近执行过的程序文件，直至找到为止。

▲　在当前文件夹中的尚未执行的程序文件中搜索，直至找到为止。

▲　将其作为一个 VFP 内部的函数，若该函数不存在，则报告错误。

示例：

设计自定义函数，求一元一次方程 ax + b = 0 的根。

分析：因为方程有 a，b 两个参数，所以函数格式可以设计为 Root（< 数值表达式 1 >，< 数值表达式 2 >）。其中 Root 是建立函数时定义的函数名，< 数值表达式 1 > 表示方程的一次项系数，< 数值表达式 2 > 表示常数项。

程序清单如下：

```
Parameters a,b
If a = 0
   x = "无解"
```

```
Else
   x = - b/a
Endif
Return
```

用此函数来解方程 $4x+1=0$,可在命令窗口的命令表达式中或其他程序语句中的表达式中使用。具体调用格式如下：

▲ 键入命令：

```
?"x = ",Root (4,1)
```

▲ 用户函数存在于调用程序中：

```
…
?"x = ",Root (4,1)
Return
Function Root
   Parameters a,b
   Return Iif (a = 0, "无解", - b/a)
Endfunc
```

▲ 用户函数存在于过程文件中,设过程文件名为 gc. prg

```
…
* 以下是过程文件 gc. prg 的程序清单
Procedure <过程名 1 >
   <语句序列 1 >
Endproc
…
Function Root
   Parameters a,b
   Return Iif (a = 0, "无解", - b/a)
Endfunc
```

调用函数：

```
Set Procedure To gc
? "x = ",Root (4,1)
```

例 25 设计一个计算圆面积的程序,并要求在主程序中带参数调用。

程序清单如下：

```
* 主程序 vfp625m. prg
ymj = 0
```

```
Input "请输入圆的半径 r = :" To r
Do js With bj,ymj
? "ymj = ",ymj
Return
* 子程序 vfp625s.prg
Parameters r,s
s = PI() * r * r
Return
```

在调用子程序前,调用语句中的参变量都赋了值;在调用子程序时,调用语句的 bj 值传送给子程序中的参数 r,子程序计算面积之后返回主程序时变量 s 的值回送给参变量 ymj。

若将程序改为过程调用:

```
ymj = 0
Input "请输入圆的半径 r = :" To r
Do js With bj,ymj
? "ymj = ",ymj
Return
Procedure js              && 过程 js 开始语句
Parameters r,s
s = PI() * r * r
Return
```

若将程序改为调用过程文件中的过程:

```
* 主程序 vfp625m.prg
ymj = 0
Input "请输入圆的半径 r = :" To r
Do js With bj,ymj In js1          && 调用过程文件 js1 中的过程
? "ymj = ",ymj
Return
* 过程文件 js1.prg
Procedure js
Parameters r,s
    s = PI() * r * r
    Return
```

3. 参数传递

过程可以接收调用程序传递过来的参数，并能根据接收到的参数控制程序流程或对接收到的参数进行处理，从而大大提高过程程序功能设计的灵活性。

接收参数的命令格式：

Parameters <虚参表>

或

Lparameters <虚参表>

说明：

- Parameters 命令声明的虚参被看做是过程程序中建立的私有变量，Lparameter 命令声明的虚参被看做是过程程序中建立的局部变量，除此之外，两条命令无区别。
- 这两条命令都应是过程程序的第一条可执行命令。

相应的，调用过程程序命令的格式：

格式 1：Do <文件名> | <过程名> With <实参表>

格式 2：<文件名> | <过程名> With（<实参表>）

说明：

- 实参可以是常量、变量、表达式。调用过程程序时，系统会将实参传递给对应的虚参。虚参的数量不能少于实参的数量，否则系统会产生运行错误。若虚参的数量多于实参的数量，则多余的虚参赋初始值. f. 。
- 采用格式 1 调用过程时，若实参是常量或表达式，系统会计算出实参的值，并将它们赋值给相应的虚参，即按值传递；若实参是变量，则传递的将不是变量的值，而是变量的地址，这时虚参和实参实际上是同一个变量（尽管它们的名字可能不同），在过程中对虚参值的改变，同样会造成对实参值的改变，即按引用传递。
- 采用格式 2 调用过程时，默认以按值传递。若实参是变量，可以通过命令 Set Udfparms 命令重新设置参数传递的方式。该命令的格式为：

Set Udfparms To Value|Reference

其中：To Value 表示为按值传递，To Reference 表示为按引用传递。

例 26 按值传递和按引用传递示例。

程序清单：

```
* vfp626.prg
Set Talk Off
Clear
```

```
Store 100 To x1,x2
Set Udfparms To Value                    && 设置按值传递
Do p4 With x1,(x2)                       &&x1 按引用传递,x2 按值传递
? "第一次",x1,x2
Store 100 To x1,x2
P4 (x1,(x2))                             &&x1、x2 都按值传递
? "第二次",x1,x2
Set Udfparms To Reference                && 设置按引用传递
Do p4 With x1,(x2)                       &&x1 按引用传递,x2 按值传递
? "第三次",x1,x2
Store 100 To x1,x2
P4 (x1,(x2))                             &&x1 按引用传递,x2 按值传递
? "第四次",x1,x2
* 过程 p4
Procedure p4
Parameters x1,x2
Store x1 + 1 To x1
Store x2 + 1 To x2
Endproc
```

(x2) 用一对圆括号将一个变量括起来使其变成一般形式的表达式,因此不管什么情况,总是按值传递,它并不受 Udfparms 值的影响。

另外还可以在调用程序和被调用程序之间传递数组。当实参是数组元素时,总是采用按值传递方式传递元素值。当实参是数组名时,若传递方式是按值传递,那么就传递数组的第一个元素值给虚参;若传递方式是按引用传递,那么传递的将是整个数组。

例 27　传递整个数组示例。

程序清单:

```
* vfp627.prg
Dimension s (10)
For i = 1 To 10
   s (i) = i
Endfor
Do p5 With s
? s (1),s (2),s (3),s (4),s (5)
```

```
? s(6),s(7),s(8),s(9),s(10)
Return
Procedure p5
Parameters x
For i = 1 To 5
  t = x(i)
  x(i) = x(11 - i)
  x(11 - i) = t
Endfor
Return
```

执行结果：

```
10        9        8        7        6
 5        4        3        2        1
```

§6.8 变量的作用域

在程序设计中离不开变量的使用，一个变量除了类型和取值之外，还有一个重要的特性就是变量的作用域。变量的作用域指的是变量在什么范围内是有效的或能被访问的。在 VFP 中，若按变量的作用域来分类，则内存变量可分为公共变量、私有变量和局部变量三类。

6.8.1 公共变量

在任何模块中都可以使用的变量称为公共变量。公共变量在使用前需声明，可以用 Public 命令来声明建立。

命令格式：

Public <**公共变量表**>

或

Public <**数组名**>(<**数值表达式** 1>[,<**数值表达式** 2>])…

功能：建立公共的内存变量或数组，并为它们赋初值.f.。

说明：

- **公共变量一旦建立就一直有效，即使程序运行结束返回到命令窗口也不会消失，只有当执行 Clear Memory、Release、Quit 等命令后，公共变量才被释放。**
- **在命令窗口中直接使用的而由系统自动隐含建立的变量也是公共变**

量,但是这样定义的变量不能在程序方式下利用。

✣ 若下层模块中建立的内存变量要提供给上层模块使用,或某模块中建立的内存变量要提供给并列模块使用,则必须将这种变量定义为公共变量。

例 28　在过程调用中,运用公共变量传递数据。

程序清单:

```
* vfp628m.prg
* main program
Set Talk Off
Clear All
Public a
A = 1
Do sub
? "返回主程序:a,b,c,d = ",a,b,c,d
Set Talk On
Return
* sub.prg
Public b,c
B = 2
D = 3
? "在过程中:a,b,c,d = ",a,b,c,d
Return
```

执行结果:

```
在过程中:a,b,c,d = 1          2          .f.          3
返回主程序:a,b,c,d = 1        2          .f.
变量未找到
```

从输出结果可以看出:公共变量 a 和 b,都可以在主程序与过程之间相互传递数据,但是系统默认未赋值的公共变量 c 的值为逻辑常量.f.,在过程中建立的局部变量 d 赋值为 3,但是该值不能传递到主程序。因而在主程序中输出变量 d 的值时,系统显示"变量未找到"的提示信息。

6.8.2　私有变量

在程序中直接使用(未通过 Public 和 Local 命令事先声明)的、由系统自动隐含建立的变量都称之为私有变量。

说明:

✧ 私有变量仅在定义它的模块及其下层模块中有效,而在定义它的模块运行结束时自动清除。

✧ 私有变量允许与上层模块的变量同名,但是此时为分清两者是不同的变量,需要采用暂时屏蔽上层模块变量的方法,以使这些变量在子程序中暂时无效,可以在需要屏蔽上层变量的模块中使用 Private 命令来实现。

命令格式:

Private [<内存变量表>] [All [Like|Except <通配符>]]

功能:声明私有变量并屏蔽上层模块中的同名变量,直至声明它的程序、过程或自定义函数执行结束,才恢复使用先前隐藏的变量。

说明:

✧ "声明"与"建立"是不一样的。前者仅指变量的类型,后者包括类型与值。Public 命令除了声明变量的类型之外还赋给了初值(故称为建立);而 Private 命令并不自动对变量赋值,仅是声明而已。

✧ 若应用程序是由多个设计人员开发,难免因变量名相同而造成失误,如果各人将自己所用的变量用 Private 命令来声明,就可避免发生错误。

✧ 在程序模块调用时,参数接收命令 Parameters 声明的实参也是私有变量,与 Private 命令作用相同。

例 29 变量隐藏与恢复示例。

程序清单:

```
* vfp629.prg
Set Talk Off
Clear
arameters sj        &&sj 为私有变量,程序调用前的 sj 被隐藏起来
Private mj          &&mj 为私有变量,程序调用前的同名变量 mj 被隐藏起来
mj = 3.14 * sj * sj
? "程序执行时的变量清单:"
List Memory
Return
```

在命令窗口中键入如下命令:

```
Release All                 && 清除用户定义的所有内存变量
mj = 0                      && 在命令窗口设置的变量为公共变量
bj = 3
?"程序执行前的变量清单:"
List Memory Like ?j                 && 显示变量清单
Do vfp629 With bj                   &&bj 传入 vfp629.prg
```

```
? "程序执行后的变量清单:"        && 显示变量清单
List Memory Like ? j              && 程序执行结束时,被屏蔽的变量
                                     mj,bj 被恢复
```

命令及程序的执行结果:

程序执行前的变量清单:

mj	Pub	N	0	(0.00000000)	
bj	Pub	N	3	(3.00000000)	

程序执行时的变量清单:

mj	(hid)	N	0	(0.00000000)	
bj	(hid)	N	3	(3.00000000)	
sj	Priv	bj			
mj	Priv	N	28.26	(28.26000000)	vfp629

程序执行后的变量清单:

mj	Pub	N	0	(0.00000000)	
bj	Pub	N	3	(3.00000000)	

6.8.3　局部变量

所谓局部变量,是指只能在建立它的模块中使用,不能在它的上层或下层模块中使用。当建立它的模块程序运行结束时,局部变量就自动释放。建立局部变量可用命令 Local 实现。

命令格式:

Local <内存变量表>

功能:建立指定的局部内存变量,并为它们赋初值.f.。

说明:

- 局部变量需先建立后使用。

例 30　公共变量、私有变量、局部变量及其作用域示例。

程序清单:

```
* vfp630.prg
Set Talk Off
Clear
Public x1                    && 建立公共变量 x1,初值为.f.
Local x2                     && 建立局部变量 x2,初值为.f.
Store "f" To x3              && 建立私有变量 x3,初值为"f"
```

```
Do vfp630s1
? "主程序中…"                    && 三个变量在主程序中都可以使用
? "x1 = ",x1
? "x2 = ",x2
? "x3 = ",x3
Return
* vfp630s1
Procedure vfp630s1
? "子程序中…"                    && 公共变量和私有变量在子程序中都可以使用
?' x1 = ",x1
? "x3 = ",x3
Return
```

在命令窗口中键入下面命令：

```
Release All                      && 清除所有用户定义的内存变量
Do vfp630
? "返回命令窗口时…"              && 程序运行结束时公共变量仍然有效
?' x1 = ",x1
```

程序与命令执行的结果如下：

```
子程序中…
x1 = .f.
x3 = .f.
主程序中…
x1 = .f.
x2 = .f.
x3 = .f.
返回命令窗口时…
x1 = .f.
```

§6.9 面向对象的程序设计

面向对象的程序设计（Object Oriented Programming，简称 OOP）与编程技术不同于标准的结构化程序设计。在进行面向对象程序设计时，首先要考虑为实现某种目标而创建的具有某种功能且操作使用便捷的控件、对象和控件的使用参数及外观，以及为实现具体功能应选用的事件及数据环境并设计好

相应的方法程序模块。

6.9.1　面向对象的基本概念

对象是由数据及可以施加在这些数据上的可执行操作所构成的统一体，是代码和数据的组合，它可以作为一个完整的、独立的单位模块来处理。面向对象程序设计方法是一种以数据和信息为主线、将数据和处理相结合的方法。设计人员并不将程序看成是工作在数据上的一系列过程或函数的集合，而是将程序看成是相互协作而又相互独立的有工作能力的对象的集合。一个对象可以是应用程序的一部分（如控件或窗体），也可以是整个应用程序。对象所调用的方法程序可以是以结构化为基础设计的完成某种操作功能的程序模块，过程可以是面向对象程序设计中的一部分准备工作，并且可以被对象所调用。不同的功能模块采用适当的连接方式并配备与其吻合的交互式操作界面以形成整体功能模块，这就是面向对象程序设计。由此可见，面向对象程序设计是结构化程序设计的结构化。

在进行面向对象程序设计时，首先要考虑的是如何创建对象，其次是对象的功能和可以进行的操作。其中应该包括以下几点：

- 希望对象能够达到反映用户意图的目标。
- 为实现这一目标，对象应具备的环境、状态、条件（数据环境）。
- 以这一目标为中心，对象应该具有的可以实施的功能及配套参数。
- 作为一个完整的整体所应具备的最佳结构体系。
- 为用户使用方便提供最佳接口、交互式操作界面。

那么什么是对象呢？

对象就是人们接洽、观测、研制、利用形成的客观世界实体目标，它是客观世界中事物的抽象，是反映客观事物属性及行为特征的可运作实体。在 OOP 中，将对象作为一个变量来处理，对象包括数据和用来处理这些数据的方法和工具。对象是构成程序的基本单位和运行实体，是应用程序的组装模块。

一般说来，对象 = 属性 + 控件 + 事件 + 数据环境 + 方法程序，是一种模块的组合体。控件是显示数据和执行操作的基本工具对象；属性是对象所具有的物理性质及其特性标识符；事件是对象所能识别和响应的某些行为和操作；数据环境是对象运行生存所依据的数据信息范围；方法程序是对象在事件触发时的行为和动作。

6.9.2　对象的属性、事件和方法

对象（Object）在现实生活中是很常见的，（如：一个人是一个对象，一台

PC 即是一个对象。若将一台 PC 机拆开来看便有“显示器、机箱、软盘驱动器、硬盘、鼠标……”,每一个又都是一个对象,即 PC 机对象是由多个“子”对象组成的。此时 PC 机称为一个包容器对象 (Container) 对象。在可视化程序设计中,常见的对象有:表单、文本框、列表框等。

从可视化编程的角度来看,对象是一个具有属性 (数据) 和方法 (行为方式) 的实体。一个对象建立之后,其操作就通过与该对象有关的属性、事件和方法来描述。

1. 对象的属性

属性 (Property) 是指对象的一项描述内容,用于描述对象的一个特性。不同的对象具有不同的属性,而每个对象又都由若干属性来描述。属性是对象的特征,是对象某一方面的行为参数,它描述了一个对象,描述了对象的状态或某一方面的行为功能,说明了对象可以完成的工作,但是还没有说明如何去完成任务。状态是对象在其生命周期中的某个特定阶段所处的某种境域、所表现的形态;行为是指对象在某种状态下所做的一系列处理、操作和反应。

在可视化编程中,常见的属性有标题 (Caption)、名称 (Name)、背景色 (BackColor)、字体大小 (FontSize)、是否可见 (Visible) 等。对象的每个属性都具有一定的含义,用户可以对其进行设置、赋值,以及定义对象的特征和行为。所有这些对象的状态和特点统称为属性,用户可以通过控制这些属性来“操作”对象,改变对象的属性就可以改变对象的特定内容。

属性值的设置或修改可以通过属性窗口来进行,也可以通过编程的方式在程序运行阶段来改变对象的属性。但是有的属性可在设计时通过属性窗口来设置而无需编写程序代码;而有的属性在设计阶段是不可用的,这些属性必须通过代码在运行阶段才能设置和修改;有的属性只能在代码中设置或修改,而不能在设计阶段通过属性窗口来设置。

在程序代码中设置属性的格式:

表单名. 对象名. 属性名 = 属性值

表 6-3 是 VFP 控件的常用属性。

表 6-3 VFP 控件的常用属性

属性名称	简 要 说 明
ActiveControl	引用对象上的当前活动控件
ActiveForm	引用活动表单对象或 Screen 对象,设置其属性
AlwaysOnTop	防止其他窗口覆盖本对象、表单窗口

属性名称	简　要　说　明
Application	提供对某个对象的引用程序对象的引用
BackColor	指定背景颜色。供对象中显示文本时使用
Caption	指定在对象的标题中显示的文本
Closable	确定表单的控制菜单是否出现“关闭”以关闭菜单

续　表

属性名称	简　要　说　明
ColorSource	决定怎样设置控件的颜色
ControlBox	决定表单的控制菜单是否显示
Controls	为访问容器对象中的控件创建的数组
CurrentX	指定绘图方法的行坐标 (x)
CurrentY	指定绘图方法的列坐标 (y)
Top	指定控件上边界与包含它的容器上边界的距离
Left	指定控件左边界与包含它的容器左边界的距离
Hight	指定控件上、下边界之间的高度
Name	指定控件名称
Width	指定控件左、右边界之间的距离

2. 对象的事件

事件 (Event) 是由 VFP 系统预先定义好的、能够被对象识别的动作 (如：单击事件、双击事件、装入事件、鼠标移动事件等)，不同的对象能识别的事件也不完全一样。事件作用于对象，对象识别事件并能作出反应。事件可以由系统引发 (如：生成对象时，系统就引发一个 Init 事件，对象识别该事件，并执行相应的 Init 事件代码)；事件也可以由用户引发 (如：用户用鼠标单击程序界面上的一个命令按钮就引发了一个 Click 事件，命令按钮识别该事件并执行相应的 Click 事件)。

对象的事件是固定的，用户不能建立新的事件。为此 VFP 提供了丰富的内部事件，这些事件足以应付 Windows 中的绝大多数操作要求。

事件过程 (Event Procedure) 是为处理特定事件而编写的一段程序代码，当事件由用户触发 (如 Click) 或由系统触发 (如 Load) 时，对象就会对该事件作出响应 (Respond)。响应某个事件后所执行的程序代码就是事件过程。一个对象能识别一个或多个事件，因此可以使用一个或多个事件过程对用户或系统的事件作出响应。

虽然一个对象可以拥有多个事件过程，但是在程序代码中要使用多少事件过程，则完全由设计人员根据程序的具体要求而定。对于必须响应的事件需要编写该事件的事件过程代码，而不必理会的事件则无需编写事件过程代码，只要交给 VFP 系统的默认处理程序即可。表 6-4 是 VFP 系统中的核心事件。

表 6-4　核心事件

事　件	触发事件的操作
Click	按下并释放鼠标左键、或改变某个控件的值、或单击表单的空白区域
DblClick	双击鼠标左键,选择列表框或组合框中的选项并按回车键
Destroy	当释放对象时
GotFocus	当接收到焦点 Focus 时
Init	当创建对象时
KePress	当用户按下并释放一个键时
Load	在创建对象之前
LostFocus	当对象失去焦点 (Focus) 时
MouseDown	当用户按下鼠标左键时
MouseMove	当用户移动鼠标到对象上时
MouseUp	当用户释放鼠标键时
ProgrammatiChange	当程序代码中改变控件的值时
RightClick	当用户在控件中按下并释放鼠标右键时
Unload	释放对象时

3. 对象的方法

方法 (Method) 是与对象相关联的过程,但又不同于一般的 VFP 过程。方法程序紧密地和对象连接在一起,并且与一般 VFP 过程的调用方式有所不同。

与事件过程类似,VFP 的方法也属于对象的内部函数,只是方法用于完成某种特定的功能而不一定响应某一事件 (如:添加对象方法、绘制矩形方法、释放表单方法等)。方法也被"封装"在对象之中,不同的对象具有不同的内部方法。VFP 系统提供了百余个内部方法供不同的对象调用。

与事件过程不同的是:根据需要可以由用户自行建立新的方法。事件过程由事件的激发而调用其代码,也可以在运行阶段由程序调用其代码,而方法的代码只能在运行阶段由程序调用。

在程序中调用对象方法的格式:

Parent. Object. Method

其中:Parent 表示父对象,Object 表示对象,Method 表示方法程序。

表 6-5 给出了 VFP 中常用的方法程序。

表 6-5 常用方法程序

常用方法程序	用 途
AddColumn	在表格控件中添加一个列对象
AddObject	在表单对象中添加一个对象
Box	在表单对象中画一个矩形
Circle	在表单对象中画一个圆形或椭圆形
Clear	清除控件中的内容
Cls	清除表单上的图形和文本
Draw	重画表单对象
Hide	隐藏表单、表单组或工具
Line	在表单上画一条线
Move	移动对象
Point	返回表单上指定的红绿蓝 3 种颜色
Print	在表单上打印一个字符串
PrintForm	打印当前表单的屏幕内容
Pset	在表单上画点
ReadExpression	返回保存在一个属性单中的表达式字符串
ReadMethod	返回一个方法中的文本
Refresh	重画表单或控件,并刷新所有数据
Release	从内存释放表单或表单组
RemoveObject	在运行时从容器对象中删除指定的对象
ResetDefault	将 Time 控件复位,使它从零开始计数
Saveas	将对象保存为. scx 文件
SaveasClass	将对象的实例作为类定义保存到类库中
SetAll	为容器对象中的所有控件或某类控件指定属性设置
SetFocus	使指定控件获得焦点
Show	显示表单并且决定表单是模态还是非模态
TextHeight	按照当前字体中的显示,返回文本串的高度
TextWidth	按照当前字体中的显示,返回文本串的宽度
WriteExpression	将一个表达式写入属性
WriteMethod	将指定的文本写入指定的方法中

6.9.3 VFP 中的类

VFP 可视化编程的最大特点就是在可视化环境下以最快的速度和效率开发具有良好用户界面的应用程序,其实质就是利用 VFP 所提供的图形构件快速开发应用程序的输入输出屏幕界面。

控件 (Control) 是某种图形构件的统称 (如:标签控件、文本框控件、列表

框控件等），利用控件所创建的对象则是某一个赋有名称的控件。

1. 常用控件和内部对象

常用控件由 VFP 的基类提供，共有 21 个，每个控件用“Form Controls（表单控件）”工具栏中的一个图形按钮表示（见表 6-6）。

表 6-6　常用控件

按钮	按钮名称	说明
	标签控件（Label）	创建一个标签对象，用于保存不希望用户改动的文本
	文本框控件（TextBox）	创建用于单行数据输入的文本框对象，用户可在其中输入或更改文本
	编辑框控件（EditBox）	创建用于多行数据输入的编辑框对象，用户可在其中输入或更改多行文本
	命令按钮控件（CommandButton）	创建命令按钮对象，用于执行命令
	命令按钮组控件（CommandGroup）	创建命令按钮组对象，用于将相关的命令编成组
	选项按钮组控件（OptionGroup）	创建选项按钮组对象，用于显示多个选项，用户只能从中选择一个
	复选框控件（CheckBox）	创建复选框对象，允许用户选择开关状态或显示多个选项，用户可从中选择多于一项
	组合框控件（ComboBox）	创建组合框或下拉列表框对象，用户可从列表项中选择一项或人工输入一个值
	列表框控件（ListBox）	创建列表框对象，用于显示供用户选择的列表框。当列表项很多，不能同时显示时，列表可以滚动
	微调控件（Spinner）	创建微调对象，用于接受给定范围之内的数值输入
	表格控件（Grid）	创建表格对象，用于在电子表格样式的表格中显示数据
	图像控件（Image）	创建图像对象，在表单上显示图像
	计时器控件（Timer）	创建计时器对象，以设定的间隔捕捉计时器事件。此控件在运行时不可见
	页框控件（PageFrame）	创建页框对象，显示多个页面
	OLE 容器控件（OLE Container）	创建 OLE 容器对象，向应用程序中添加 OLE 对象
	OLE 绑定型控件（OLE Bound）	创建 OLE 绑定型对象，可用于向应用程序中添加 OLE 对象。与 OLE 容器对象不同的是，OLE 绑定型对象绑定在一个通用字段上

续 表

按钮	按钮名称	说明
	线条控件 (Line)	创建线条对象,设计时用于在表单上画出各种类型的线条
	形状控件 (Shape)	创建形状对象,设计时用于在表单上画出各种类型的形状。可以画出矩形、圆角矩形、正方形、圆角正方形、椭圆或圆
	容器控件 (Container)	创建容器对象,在容器中可以包含其他控件
	分隔符控件 (Separafor)	创建分隔符对象,在工具栏的控制间加上空格
	超级链接对象 (HyperLink)	使用“超级链接”可以跳转到 Internet 或 Intranet 的一个目标地址上

可以利用这 21 个常用控件创建用户自己需要的对象,也可以设计新的控件。同时 VFP 还提供了一些内部对象 (如:表单对象、表单集对象、页对象和工具栏对象等)。内部对象一般是可以直接被调用的,但是某些对象是要在建立某对象之后才能被使用。

2. VFP 中的类

对象和类是面向对象方法中的两个最基本的概念。

类和对象关系密切,但并不相同。类是一类相似对象的性质的描述,这些对象具有相同的性质、相同种类的属性以及方法。类就好像是一类对象的模板,有了类之后,基于类就可以生成这类对象中任何一个对象。这些对象虽然采用相同的属性来表示状态,但是它们在属性上的取值完全可以不同。这些对象一般有着不同的状态,且彼此间相对独立 (如:为学生创建一个类。在“学生”类的定义中,需要描述的属性可能包括“学号”、“姓名”、“性别”、“出生年月”等,需要描述的方法可能有“注册”、“考试”、“毕业”等。基于“学生”类,可以生成任何一个学生对象。对生成的每个学生对象,都可以为其设置相应的属性值)。

在类的定义中,也可以为某个属性指定一个值,这个值将作为基于该类生成的每个对象在该属性上的默认值。通常将基于某个类生成的对象称为这个类的实例,因此任何一个对象都是某个类的一个实例。不过需要注意的是,方法尽管定义在类中,但执行方法的主体是对象,同一个方法,如果由不同的对象去执行,一般会产生不同的结果。

在 VFP 环境下,要进行面向对象的程序设计或创建应用程序时,必然需使用 VFP 系统提供的基础类 (即 VFP 基类)。

(1) 基类

VFP 基类是系统本身内含的、并不存放在某个类库中,用户可以基于基类

生成所需的对象，也可扩展基类创建自己的类。表 6-7 是 VFP 派生子类或创建对象的基类，这些控件只能用 VFP 基类以编程的方式创建。

表 6-7　VFP 基类

类　名	含　义	类　名	含　义
Column	列对象	CheckBox	复选框控件
ListBox	列表框控件	ComboBox	组合框控件
CommandButton	命令按钮控件	EditBox	编辑框控件
CommandGroup	命令按钮组控件	Control	控件对象
Custom	自定义对象	Cursor	临时表对象
Form	表单对象	FormSet	表单集对象
Grid	表格控件	Header　(*)	标头对象
Image	图像控件	Label	标签控件
Line	线条控件	DataEnvironment	数据环境对象
Page　(*)	页面对象	PageFrame	页框对象
Separator	分割符对象	OLEBoundControlOLE	绑定形对象
OLEContainerControlOLE	容器对象	OptionGroup	选项组控件
OptionButton　(*)	选项按钮控件	Shape	形状控件
Spinner	微调控件	TextBox	文本框控件
Timer	计时器控件	ToolBar	工具栏对象
Class	该类属于何种类型	BaseClass	该类由何种基类派生
ClassLibrary	该类从属于哪种类库	ParentClass	对象所基于的类
Container	容器对象		

注：(*) 这些类是父容器的集成部分，在“类设计器”中不能进行子类化。

(2) 容器类

容器类对象可以包含其他对象，并且允许访问这些对象 (如：表单集、表单、表格等)。

控件类对象只能包含在容器对象之中，而不能包含其他对象 (如：命令按钮、复选框等)。当一个容器包含一个对象时，称该对象为容器的子对象，而容器称为该对象的父对象。

容器类所能包含的对象见表 6-8。

表 6-8 各种容器类及其所包含的对象

容 器	基 类	包 含 对 象
OLE 容器控件	OLEControl	任意控件
表单	Form	页框、任意控件、容器或自定义对象
表单集	FormSet	表单、工具栏
命令按钮组	CommandGroup	命令按钮
控制	Control	任意控件
表格	Grid	网格列
表格列	Column	标头对象及除表单(集)、工具栏、计时器和其他列对象之外的任意对象
选项按钮组	OptionGroup	选项按钮
页框	PageFrame	页面
工具栏	ToolBar	任意控件、容器和自定义对象
自定义	Custom	任意控件、页框、容器和自定义对象

(3) 控件类

控件类的封装比容器更为严密,但是也因此损失了一些灵活性,非容器类对象在设计阶段和运行阶段都可以被当做一个整体来处理,而控件对象的组件不能单独被修改和操作。

3. 对象的引用

对象的引用是指如何在程序设计和程序运行时对这些对象进行控制和操作。若需引用一个对象,就应该知道它相对于容器层次的关系(如:要在表单集中处理一个表单的控件,就应引用表单集、表单和控件)。对象的引用包括以下两种引用方式:

(1) 绝对引用

通过提供对象的完整容器层次来引用对象就称为绝对引用。

图 6-12 表示了一种可能的容器嵌套方式。若要使表列中的控件无效,则需:

FormSet. Form. PageFrame. Page. Grid. Column. Control. Enabled = . f.

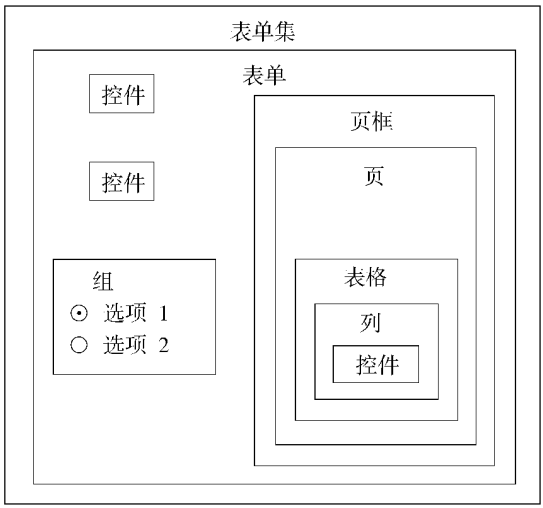


图 6-12　嵌套容器

⑵ 相对引用

在容器层次中引用对象时，可以通过快捷方式指明所要处理的对象，即相对引用。

表 6-9 列出了对象的一些属性和关键字，这些属性和关键字允许更方便地从对象层次中引用对象。

表 6-9　从对象层次中引用对象时用到的一些属性和关键字

属性或关键字	引　　用
Parent	该对象的直接容器（父对象）
THIS	该对象
THISFORM	包含该对象的表单
THISFORMSET	包含该对象的表单集

注：只能在方法程序或事件过程中使用 THIS、THISFORM、THISFORMSET。

6.9.4　创建对象和类

在进行面向对象的程序设计时，经常需要创建类与对象，然后再在这些对象的基础上设计应用程序。

1. 创建与定义类

创建类有两种常用的方式，以一个例子说明创建的过程。

例 31　建立一个新类“命令按钮组”。

方式 1：使用菜单方式创建：

✧ **选定“文件”菜单的“新建”选项，然后在弹出的“新建”对话框中选定“类”单选项，再选定“新建文件”按钮，屏幕上将显示“新建类”对话框，如图 6-13 所示。**

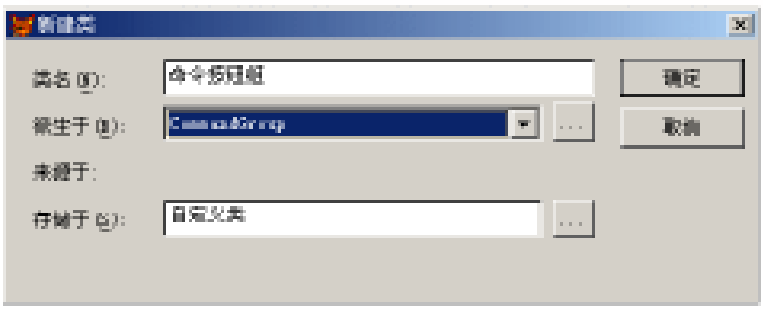


图 6-13 “新建类”对话框

✧ **在“新建类”对话框中需要指定新建类所需要的类库、基类与类名，详细说明如下：**

■ **“存储于”文本框：用于指定新类库名或已有类库的名字。类库名可包含路径，若未指明路径表示使用默认路径（本例中输入新类库名为“自定义类”）。**

■ **“派生于”下拉文本框：用于指定派生基类的子类。点击“派生于”文本框右侧按钮，在出现的基类名称中选择 CommandGroup，点击“确定”按钮，进入“类设计器”窗口，如图 6-14 所示。**

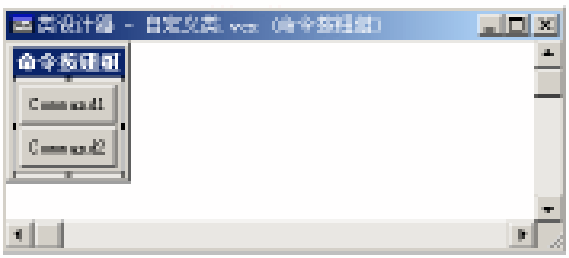


图 6-14 “类设计器”窗口

■　类名文本框：用于指定类名，该类是基类的子类（本例在类名文本框中输入“命令按钮组”）。

✣ 在“类设计器”窗口中，可以使用系统菜单中的“类”菜单项，在属性窗口中可以查看和编辑类的属性，在代码编辑窗口中可以编写各种事件和方法程序的代码。

方式 2：使用命令方式进行创建，命令格式如下：

格式：Create Class <类名>

或　Create Class <类名> Of <类库名>

使用 Create Class 命令可以打开“新建类”对话框，后续操作与方式 1 相同。

2. 创建对象

对象变量的创建可以使用 CreateObject（）函数进行，其功能是从一个类定义或一个 OLE 对象中创建一个对象，运行的函数返回对象的类型。

格式：CreateObject（<类名>）[,<参数表达式>[,<参数表达式>…]]

其中：<类名>可以是用户自定义的类，也可以是系统提供的类。

在容器对象中添加对象可以使用 AddObject（）方法，其命令格式如下：

格式：<对象名>. AddObject（<对象名>,<类名>[,<参数 1>[,<参数 2>…]]）

该方法有 3 个特性：添加对象可用在一般程序和类的方法中，但是不能用在类的定义中；向容器对象添加对象时，Visible 属性的缺省设置为. f.，即对象不可见，如果要使其可见则改其属性为. t.；可将参数传递给对象的 Init 方法并触发 Init 方法。

3. 设置属性

⑴ 属性的设置

当类创建之后，新类已继承了基类或父类的全部属性。同时系统也允许修改基类、父类的原有属性或设置类的新属性。现举例说明其操作方法。

例 32　修改已有新类“命令按钮组”的属性，将原 Caption 属性的默认值 Command1、Command2 分别改为“运行”与“结束”。

操作步骤如下：

✣ 在如图 6-14 所示的状态下，打开系统菜单的“显示”菜单，选择“属性”选项，弹出“属性”窗口，如图 6-15 所示。

✣ 用鼠标单击命令按钮组中的 Command1 按钮。

✣ 在“属性”对话框中用鼠标单击 Caption，然后输入“运行”。

✣ Command2 的修改与 Command1 相同。

✧ 关闭“属性”对话框,修改操作结束。

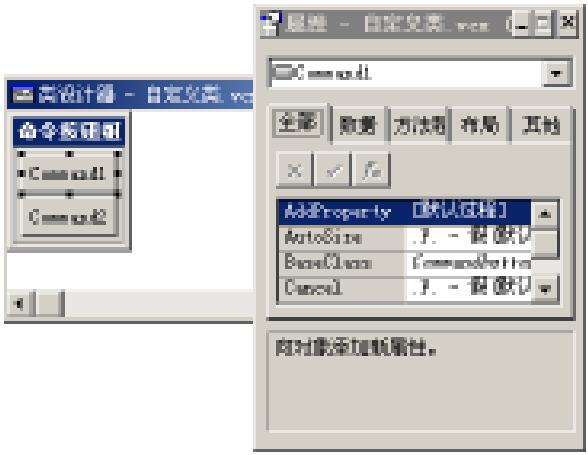


图 6-15 按钮 Command1 的属性窗口

有时需要添加新属性。可选择类菜单的“新建属性”,进入“新建属性”窗口,在该窗口中输入信息后,再按“添加”按钮,则新属性被加入到“属性”窗口。

例 33 在已有新类“命令按钮组”中添加名为“ ”的新属性。

操作步骤如下:

✧ 在如图 6-14 所示的状态下,打开“类”菜单,选择“新建属性”,进入“新建属性”窗口,如图 6-16 所示。

✧ 在“新建属性”窗口中输入如下信息:

■ 在“名称”文本框中,输入“ ”。

■ 在“可视性”下拉列表框中,选择“公共”。

■ 在“说明框”中输入“新属性”。

再按“添加”按钮,将会看到新添加的属性已加入到“属性”窗口中。

如果再弹出“属性”窗口,新属性已可使用。

⑵ 对象的属性设置

对象的属性设置分为直接设置和结构化设置。直接设置使用“对象名.属性名 = 值”的格式 结构化设置使用 With … EndWith 命令,该命令的格式如

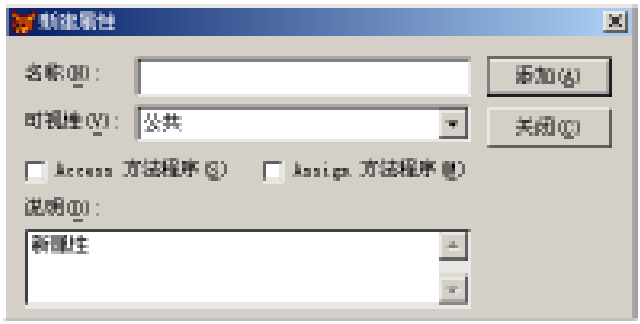


图 6-16　“新建属性”窗口

下：

格式：With　<**对象名**>　　**或：**With　<**路径**>
　　[<**语句序列**>]　　<**属性**>
　　EndWith　　EndWith

设置对象属性还经常使用如下格式：

Parent. Object. Property = Value

说明：

父对象. 对象. 属性 = 属性值

例 34　**为一个表单对象“**form1**”设置属性值。**

程序代码如下：

```
form1. Caption = "我的表单"
form1. BackColor = RGB (128,128,0)
form1. Name = "form1"
form1. Left = 20
form1. Top = 10
form1. Height = 225
form1. Width = 500
```

或：

```
With form1
   . Caption = "我的表单"
   . BackColor = RGB (128,128,0)
   . Name = "form1"
   . Left = 20
   . Top = 10
```

```
. Height = 225
. Width = 500
EndWith
```

4. 类的方法和事件的定义

当类创建完成之后，虽然已继承了基类或父类的全部方法和事件，但是多数时候还是需要修改基类、父类原有的方法和事件，或加入新的方法。操作方法如下：

- 在 VFP 系统的主菜单下，打开某一基类，进入“类设计器”窗口。
- 在系统菜单的“显示”菜单中，选择“代码”，进入“代码编辑”窗口。在该窗口，可在“对象”下拉框中选择对象，在“过程”窗口下拉框中确认继承下来的方法和事件，或修改继承的方法和事件。
- 在“代码编辑”窗口中，“过程”窗口下的下拉框中列出的方法如果不能满足对类的定义，用户可以自己添加新的方法：打开“类”菜单，选择“新方法程序”，进入“新方法程序”窗口。
- 在“新方法程序”窗口，输入如下信息：
  - 在“名称”文本框中，输入要创建的新方法名。
  - 在“可视性”下拉列表框中，选择方法属性。
  - 在“说明”框中，输入对新方法的说明。

再按“添加”按钮，进入“代码编辑”窗口，新方法被加入到“代码编辑”窗口中。

例 35　给已有的新类“自定义类”，添加“Click”事件的过程代码。

“Click”事件的过程代码内容如下：

```
a = MessageBox ("你真的要结束，退出系统吗？",4 + 16 + 0,"对话窗口")
If a = 6
    Release THISFORM
Endif
```

5. 通过编程定义类

在 VFP 系统中，定义类除了在类设计器中进行之外，还可通过 Define Class 命令改变来实现。

命令格式：

```
Define Class ClassName As ParentClass
[object.] Property = Expression
[Add Object ObjectName As ClassName
    With Propertylist
    [Procedure Name
```

```
        Statements
    Endprocedure]
EndDefine
```

例 36　定义一个带命令按钮的新的容器“myform”,并确定其自身属性和所包含的控件“comm1”的属性及控件的“Click”事件代码。

程序代码：

```
Define Class myform As Form
Visible = .t.
BackColor = RGB(128,128,0)
Caption = "我的表单"
Left = 20
Top = 10
Height = 250
Width = 500
Add Object comm1 As CommandButton;
With Caption = "关闭",;
Left = 300,;
Top = 150,;
Height = 40,;
Width = 60
Procedure comm1.Click
A = MessageBox("你真的要结束,退出系统吗?",4 + 16 + 0,"对话窗口")
If a = 6
Release THISFORM
Endif
EndProcedure
EndDefine
```

第7章 VFP6 表单设计

VFP 为用户提供了可以交互式操作的数据信息、可以设计的 Windows 窗口界面——表单（表单也可以称为窗体）。

表单是 VFP 提供的最常见的数据交互式操作界面工具，各种对话框和窗口是表单的不同表现形式。表单是 VFP 常用的并具有自己的控件、属性、事件、方法程序的容器对象，各种对话框和窗口都是表单不同的外观表达形式，为尽可能方便、直观地完成数据信息管理工作提供了条件。

VFP 为用户提供了设计交互式操作界面的工具——表单设计器，它是可视化的面向对象程序设计的工具。在 VFP 的每一个表单或表单集中都有一个数据环境，在表单的设计、运行中需要使用数据环境（与表单相配合的表或字段）。通过将与表单相关的表或视图放进表单的数据环境中，可以很容易地将表单、新控件与表或视图中的字段关联在一起，形成一个完整的构造体系。数据环境的设置在每一个表单设计中几乎都是必不可少的。

表单中使用的控件是提供给用户的基于标准化图形界面的多功能、多任务操作工具，可以创建、完成信息的输入与输出。在表单中可以使用的 Windows 交互式操作界面常用的 15 种标准控件分别是：复选框、组合框、编辑框、文本框、命令按钮、线条、形状控件、图像、微调控件、计时器、标签、ActiveX 绑定控件、ActiveX 控件、超级链接，另外还有命令按钮组、选项按钮组、页面框等 4 种容器。

进行 VFP 表单设计时经常使用的设计工具主要有：表单设计器、表单向导、属性窗口、生成器、数据环境设计器、对话框、控件工具栏、布局工具栏、调色板工具栏、代码设计窗口、浏览器等。表单设计器如图 7-1 所示。

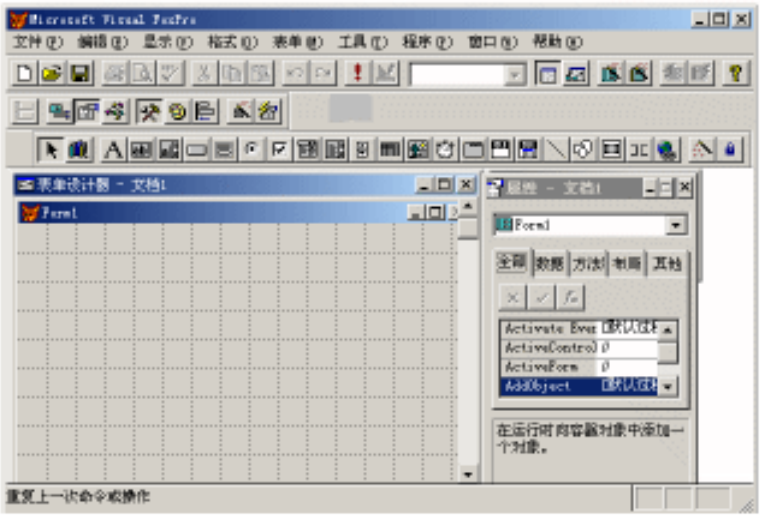


图 7-1　表单设计器窗口

“属性”窗口显示了添加到表单中的控件所具有的全部属性，可以为每一个控件和表单设置属性，选择与需求、操作相适应的事件，并配备方法程序。

在 VFP 中，可以使用以下任意一种方法生成表单：

- 使用表单向导。
- 通过选择“表单”菜单上的“快速表单”选项，可以创建一个通过添加用户自己的控件来定制的简单表单。
- 使用“表单设计器”创建或修改已有的表单。
- 使用命令：Create Form。

§7.1　设计表单

创建表单一般有两种途径：

- 使用表单向导创建即用表单。
- 使用表单设计器创建、设计新的表单或修改已有的表单。

7.1.1　使用表单向导创建表单

VFP 提供了两种表单向导来帮助用户创建表单：

- “表单向导”适合于创建基于一个表的表单。

✧“一对多表单向导”适合于基于两个具有一对多关系的表的表单。

利用表单向导创建表单的方法和步骤如下：

✧在“项目管理器”窗口中，选择“文档”选项卡，选择其中的“表单”图标。或在系统菜单中选择“文件”下的“新建”，然后选择“表单”。

✧单击“新建”按钮，系统弹出“新建表单”对话框，如图 7-2 所示。

✧单击“表单向导”图标按钮，打开“向导选取”对话框，如图 7-3 所示。

✧从列表框中选择要使用的向导，然后单击“确定”按钮。

图 7-2 “新建表单”对话框

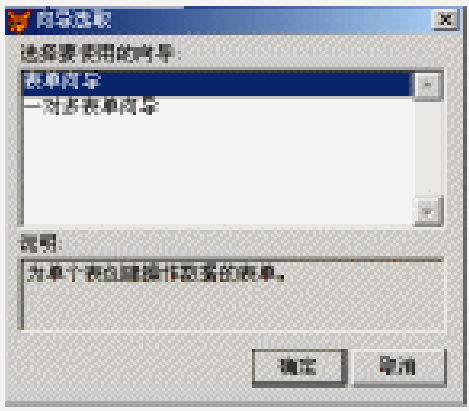


图 7-3 “向导选取”对话框

无论调用哪种表单向导，系统都会打开响应的对话框，逐步向用户询问一些简单的问题，并根据用户的回答自动创建表单。创建的表单将包含一些控件，用以显示表中的记录和字段中的数据，表单还会包含一组按钮。用户通过这组按钮，可以实现对表中的数据进行浏览、查找、添加、编辑、删除以及打印等操作。

此外，还可用以下方法调用表单向导：在“文件”菜单中选择“新建”命令，然后在打开的“新建”对话框中选择表单文件类型并按“向导”按钮，或者在“工具”菜单的“向导”子菜单中选择“表单”命令。

例 1 创建一对一的显示学生基本情况的表单。

操作步骤如下：

① 从“文件”菜单中选择“新建”选项，在“新建”对话框中选择表单文件类型，然后在“新建”对话框中选择“向导”按钮；也可单击“工具”菜单中的“向导”菜单选项引导子菜单中的“表单”项，选中表单向导；或在“项目管理

器”窗口中选择“文档”选项卡的“表单”项，单击“新建”按钮，在出现的“新建表单”窗口中单击“表单向导”按钮。“向导选取”对话框如图 7-4 所示。

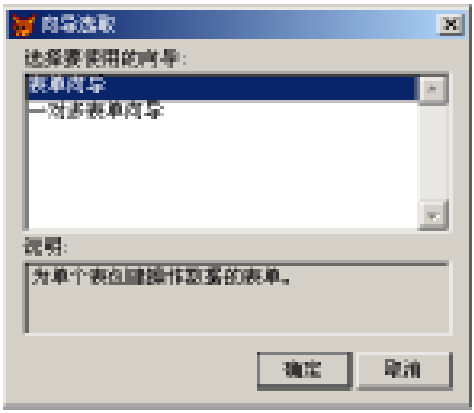


图 7-4　“向导选取”对话框

选择“表单向导”，单击“确定”按钮进入“表单向导”窗口（步骤 1），如图 7-5所示。

图 7-5　“表单向导”窗口（步骤 1）

在这一步骤中可先选用表：点击 ... 按钮，确定表单的数据源为教学数据库。在“数据库和表”的下拉列表中输入名称，也可单击其右边的浏览按钮选择表。用户可双击选中所需要的可用字段或按“选取”按钮选定字段，选定表的字段显示在可用字段中。字段选定的前后顺序决定了向导在表达方式中安排字段的顺序以及表单的缺省标签顺序，其顺序可通过选定字段列表框中前面的按钮进行调整。字段选好之后可单击“下一步”按钮，进入“表单向导”窗口（步骤 2），如图 7-6 所示。

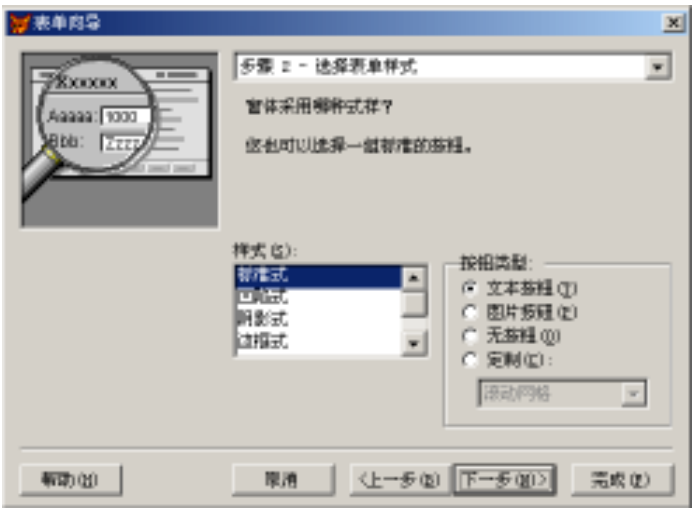


图 7-6 “表单向导”窗口（步骤 2）

在步骤 2 的选择表单样式中，若只打算做一般的表单，则可选择“标准格式”，此时可在“按钮类型”中选择“文本按钮”单选项，单击“下一步”按钮，进入下一个窗口，如图 7-7 所示。

在步骤 3 的排序次序中，允许按照每组记录的排序顺序选择字段，也可选择索引标识，还可定义一个所选列表的顺序视图，将排列顺序定义为升序或降序，此时需注意排序和索引的关系。在本例中，选定“学号”字段进行排序。

在步骤 4 的“完成”窗口（见图 7-8）中，可键入表单的标题，并可选择建立好表单后的动作。表单运行的结果如图 7-9 所示。

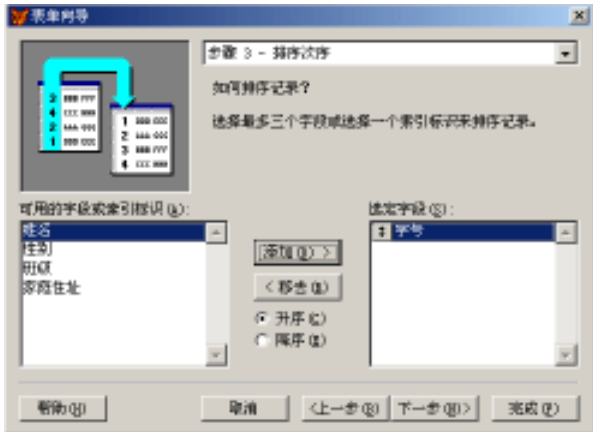


图 7-7　“表单向导”窗口 (步骤 3)

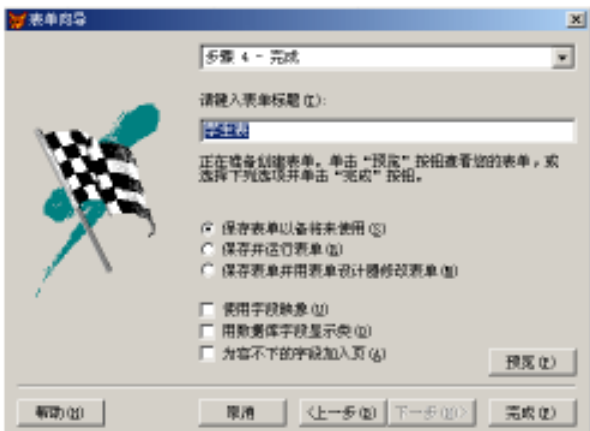


图 7-8　“表单向导”窗口 (步骤 4)

图 7-9　表单运行结果

注：表单向导自动按所选择按钮的形式，添加了 10 个常用的按钮。每一个按钮的功能用名称表述清晰。与其有关的属性、方法程序完好无损。

例 2　使用表单向导创建一对多表单。

一对多表单的创建不同于一对一表单的创建，在使用一对多表单向导进行创建表单时，字段既要从主（父）表中选取，也要从子表中选取，还要建立两表之间的有效关系。一对多表单一般使用文本框来表述父表，使用表格来表述子表。

操作步骤如下：

在“一对多表单向导”窗口（步骤 1）（见图 7-10）中，选择主表为“学生”，并且“学号”、“姓名”、“班级”3 个子字段出现在“选定字段”列表框中，然后单击“下一步”按钮。

图 7-10　“一对多表单向导”窗口（步骤 1）

在步骤 2 窗口（见图 7-11）中，选择子表为“分数”，并选定字段“学号”、“姓名”、“数学”、“体育”、“英语”。选好之后单击“下一步”按钮，进入步骤 3 窗口（见图 7-12），以建立表之间的关系。

在步骤 3 窗口中，在每个表中选定一个匹配字段即“学号”，建立表之间的关系，以便在建成的表单中可以显示相匹配的数据信息。

图 7-11　“一对多表单向导”窗口 (步骤 2)

图 7-12　“一对多表单向导”窗口 (步骤 3)

在步骤 4 窗口 (见图 7-13) 中，选择表单的样式可以产生不同的视频效果 (在本例仍然选择标准式和文本按钮)。

在步骤 5 窗口 (见图 7-14) 中，排序次序的选取与输出的结构密切相关，选择“学号”以每位学生的课程来划分，依次排列顺序。

在步骤 7 窗口 (见图 7-15) 中，输入表单标题，确定保存并运行表单项。运行结果如图 7-16 所示。

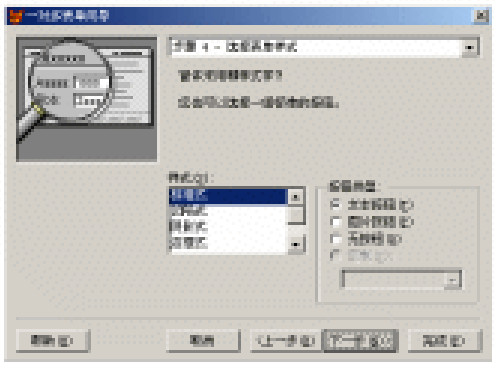


图 7-13 “一对多表单向导”窗口（步骤 4）

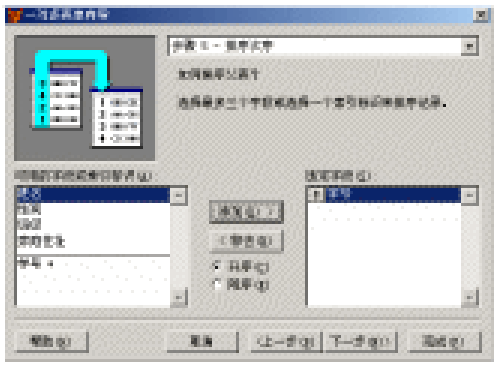


图 7-14 “一对多表单向导”窗口（步骤 5）

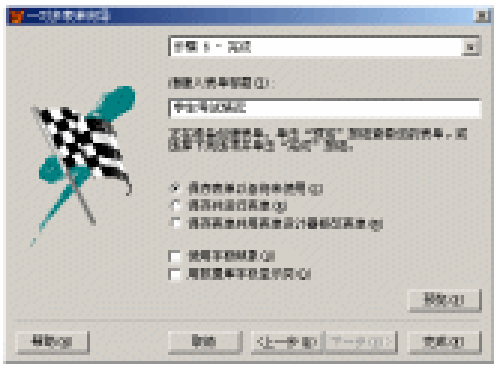


图 7-15 “一对多表单向导”窗口（步骤 6）

图 7-16　运行效果图

7.1.2　使用表单设计器创建表单

可以使用下面 3 种方法中的任何一种调用表单设计器：

方法 1：在项目管理器环境下调用。

① 在“项目管理器”窗口中选择“文档”选项卡，然后选择其中的“表单”图标。

② 单击“新建”按钮，系统弹出“新建表单”对话框。

③ 单击“新建表单”图标按钮。

方法 2：菜单方式调用。

① 单击“文件”菜单中的“新建”命令，打开“新建”对话框。

② 选择“表单”文件类型，然后单击“新建文件”按钮。

方法 3：命令方式调用。

在命令窗口中输入命令：

```
Create Form
```

以上任何一种方法都可以打开“表单设计器”，开始设计表单。图 7-17 为进入表单设计器时的初始画面。

表单设计器中包含一个新创建的表单或待修改的表单，可在其上添加和修改控件。表单可在表单设计器内移动或改变其大小。

如果用户的屏幕上未出现“表单设计器”工具栏，可将鼠标移到标准工具条上的任意位置，单击鼠标右键，从弹出的快捷菜单中选择“表单设计器”；或从“显示”菜单中选择“工具栏”，如图 7-18 (a) 所示；在“工具栏”对话框中选择“表单”设计器，然后单击“确定”按钮，如图 7-18 (b) 所示，也可得到表单设计器工具栏。

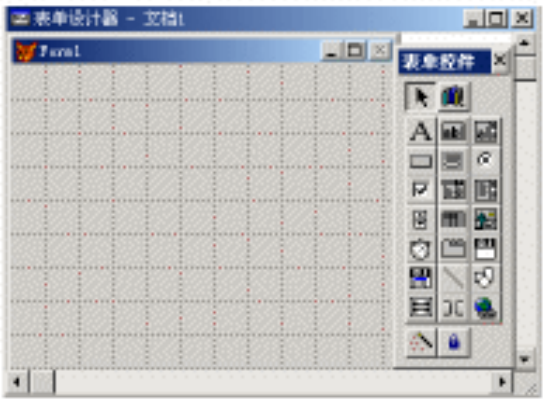
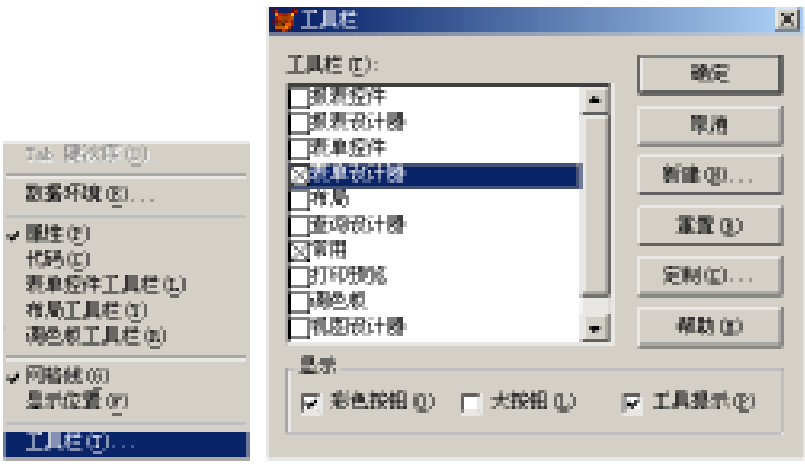


图 7-17　表单设计器窗口



(a)　　(b)

图 7-18　“显示”菜单与“工具栏”对话框

表单设计器工具栏中包括了设计表单时的所有工具。

将鼠标指针移到工具栏的某个按钮上，就会出现该工具按钮的名称，如图 7-19 所示。

图 7-19　表单设计器工具栏

各个工具按钮的功能说明如下：

设置 Tab 次序——在表单的设计过

程中，单击这个按钮，可以显示当按动 < Tab > 键时，光标在表单的各个控件上的移动顺序。用键盘上的 < Shift > 键加上鼠标左键可以重新设置光标移动的顺序。

数据环境窗口——在表单设计过程中，单击此按钮，可以结合用户界面同时设计一个依附的数据环境。

属性窗口——在表单设计过程中，单击此按钮，可以启动或关闭属性窗口，以便在属性窗口中查看和修改各个控件的属性。

代码窗口——在表单设计过程中，单击此按钮，可以启动或关闭代码窗口，以便在代码窗口中编辑各对象的方法及事件代码。

表单控件工具栏——在表单设计过程中，单击此按钮，可以启动或关闭表单控件工具栏，以便于利用各控件进行用户界面的设计。

调色板工具栏——在表单设计过程中，单击此按钮，可以启动或关闭调色板工具栏，利用调色板工具栏可以进行各对象前景与背景颜色的设置。

布局工具栏——在表单设计过程中，单击此按钮，可以启动或关闭布局工具栏，利用布局工具栏可以针对对象进行位置配置和对齐设置。

表单生成器——启动表单生成器，直接以填表的方式进行相关对象的各项设置，以方便用户快速建立表单。

自动格式——在表单设计过程中，单击此按钮，可以启动或关闭自动格式生成器，以对各控件进行格式控制。

7.1.3　添加控件

单击“表单设计器”工具栏上的“表单控件工具栏”按钮，屏幕上出现“表单控件”工具栏，可以将它拖到适当的位置，如图 7-20 所示。

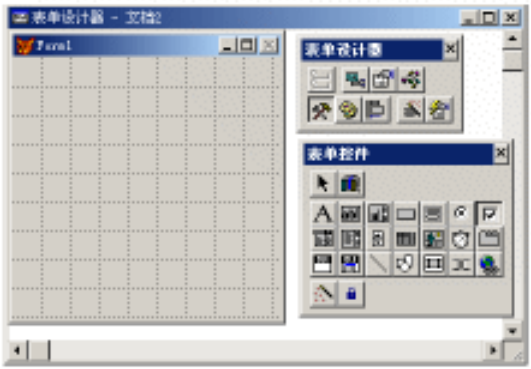


图 7-20　带有“表单控件”工具栏的表单设计器窗口 (步骤 1)

“表单控件”工具栏中提供了 VFP 可视化编程的各种控件，利用这些控件，可以创建出所需要的对象。除了各种控件之外，工具栏中还有以下按钮：

选定对象——选定一个或多个对象，移动和改变控件的大小。在创建一个对象之后，“选择对象”按钮被自动选定，除非按下了“按钮锁定”按钮。

查看类——单击激活，使用户可以选择显示一个已注册的类库。在选择一个类后，工具栏只显示选定类库中类的按钮。

生成器锁定——生成器锁定方式可以自动显示生成器，为任何添加到表单上的控件打开一个生成器。

按钮锁定——按钮锁定方式可以添加多个同种类型的控件，而不需要多次按此控件的按钮。

下面将开始设计一个表单。首先在表单上增加一个控件：

- 单击“表单控件”工具栏中的“命令按钮”。
- 将鼠标指向表单的右下部，按下鼠标左键并拖动鼠标的十字指针画出一个矩形框，松开左键即画出一个“命令按钮”，如图 7-21 所示。按钮内自动标有“Command1”，序号将自动增加。

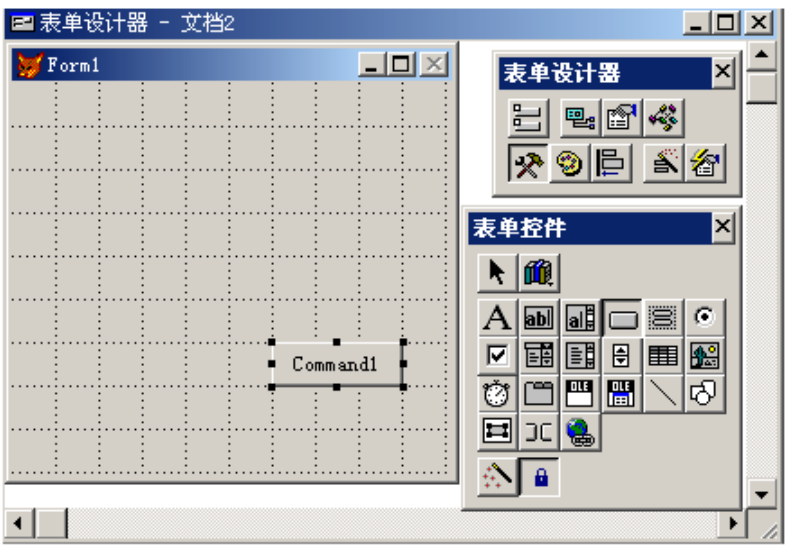


图 7-21　增加一个“命令按钮”

7.1.4　修改属性

在设计时修改或设置属性，一般是在“属性窗口”中进行的。单击鼠标右键，在弹出的快捷菜单中选取“属性”（见图 7-22）或单击“表单设计器”工具栏中“属性窗口”按钮，就可打开“属性”窗口（见图 7-23）。

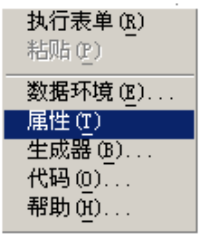


图 7-22　选取“属性”

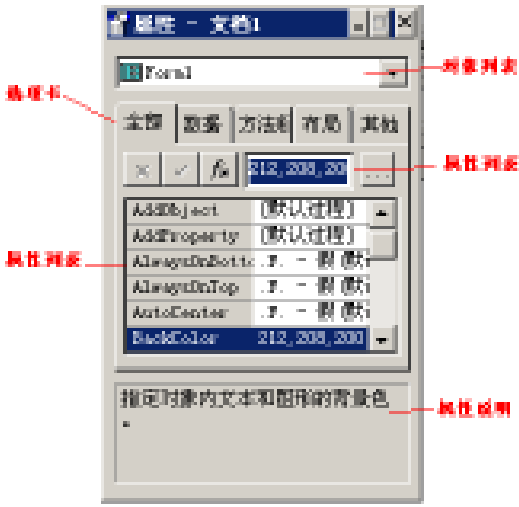


图 7-23　“属性”窗口

属性窗口包括选定对象（表单或控件）的属性、事件和方法列表。可以在设计或编程时对这些属性值进行设置或更改。属性窗口从上至下依次包括：

- 对象下拉列表框：标识当前选定的对象。单击右端的向下箭头，可看到包括当前表单（或表单集）及其所包含的全部对象的列表。从列表中可选择要更改其属性的表单或对象。
- 选项卡：按分类显示所选对象的属性、事件和方法。依次为：

  全部：显示全部对象的属性、事件和方法。

  数据：显示所选对象如何显示或怎样操纵数据的属性。

  方法程序：显示方法和事件。

  布局：显示所有的布局属性。

  其他：显示其他和用户自定义的属性。
- 属性设置框：可以更改属性列表中选定的属性值。若选定的属性需要预定义设置值，则在右边出现一个向下的箭头。若属性设置需要指定一个文件名或一种颜色，则在右边出现 3 点按钮。单击“接受”按钮（√符号）来确

认对此属性的更改。单击“取消”按钮（×符号）则取消更改，恢复以前的值。有些属性（如背景色）显示一个 3 点按钮，允许从一个对话框中设置属性。单击“函数（fx）”按钮，可打开表达式生成器。属性可以设置为原值或由函数或由表达式返回的值。

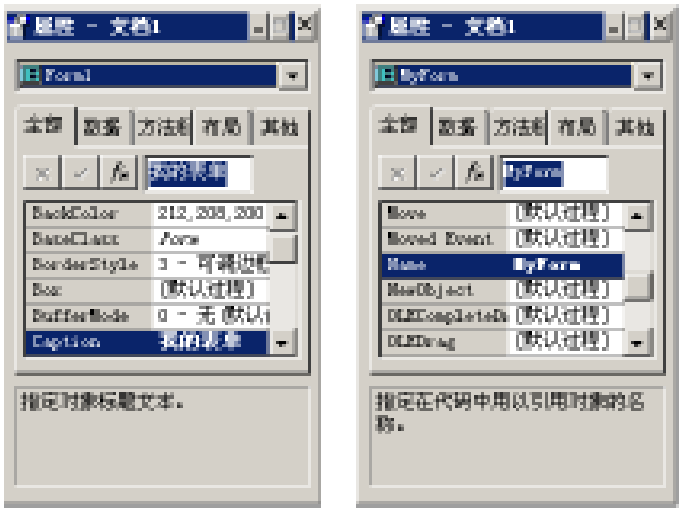
- 属性列表：它包含两列列表，显示所有可以在设计时更改的属性和它们的当前值。对于具有预定值的属性，在“属性”列表中双击属性名可遍历所有可选项。对于具有两个预定值的属性，在“属性”列表中双击属性名可在两者之间切换。选择任何属性并按 < F1 > 键可得到此属性的帮助信息。对于以表达式作为设置的属性，它的前面具有等号“ = ”。只读的属性、事件和方法以斜体显示。

在属性窗口中以上各项之外处单击鼠标右键，将弹出快捷菜单，如图 7-24 所示。在该快捷菜单中选择相应项，可改变“属性”窗口的外观。

图 7-24 快捷菜单

设置和修改属性的步骤如下：

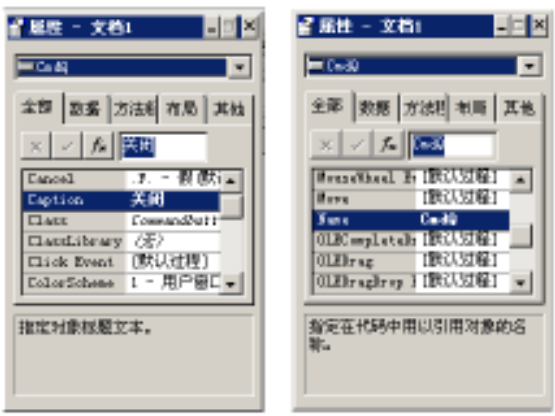
首先，“对象”下拉列表框中显示的对象名为：“Form1”。在“布局”选项卡中找到标题属性 Caption，将其改为“我的表单”（原值为 Form1），如图 7-25（a）所示。在“其他”选项卡中找到表单名属性 Name，将其改为“MyForm”（原值为 Form1），如图 7-25（b）所示。



(a)　　(b)

图 7-25　修改表单 Forml 的属性

其次，在表单上用鼠标单击命令按钮“Command1”或在“对象”下拉列表框中选择对象“Command1”，将其标题属性 Caption 改为“关闭”（原值为 Command1），如图 7-26 (a) 所示。将其名称属性 Name 改为“CmdQ”（原值为 Command1），如图 7-26 (b) 所示。修改后的表单画面如图 7-27 所示。



(a)　　(b)

图 7-26　修改命令按钮“Command1”的属性

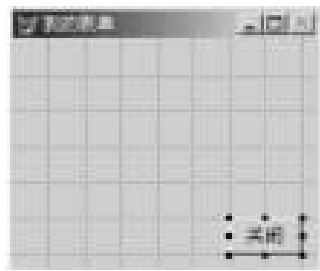


图 7-27　修改后的表单画面

7.1.5 编写代码

编写代码就是为对象编写事件过程或方法。编写代码是在“代码(Code)”窗口中进行的。打开代码窗口的方法为：

✧ 在表单中用鼠标右键单击需要编写代码的对象，在弹出的快捷菜单中选择“代码…”，如图 7-28 所示；或单击表单设计器中的“代码”按钮。

✧ 用鼠标左键双击需要编写代码的对象。打开的代码窗口如图 7-29 所示。

图 7-28 在快捷菜单中选择“代码”

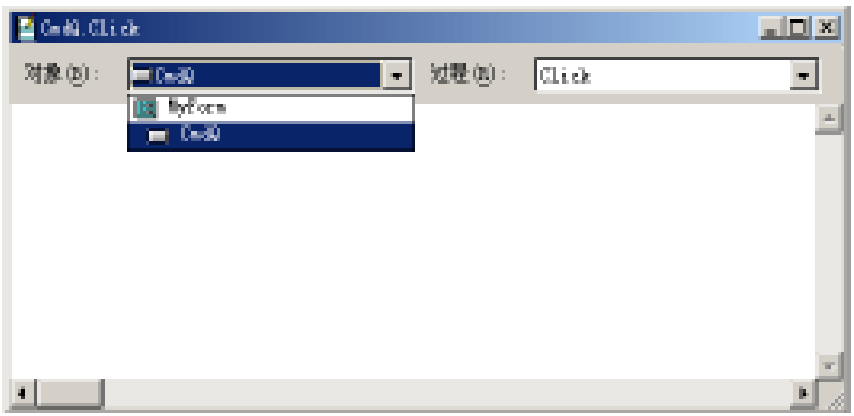


图 7-29 打开的代码窗口

窗口中的“对象”下拉列表框中列出当前表单及所包含的所有对象名：MyForm、CmdQ。其中 CmdQ 对象前的缩进表示对象的包容关系。

“过程”下拉列表框中可以列出所选对象的所有方法及事件名。

✧ 在“对象”下拉列表框中选择“CmdQ”对象，在“过程”下拉列表框中选择“Click”，并在代码窗口中输入代码：Release ThisForm，如图 7-30 所示。

其中：Release 是 VFP 命令，用于从内存中清除变量或引用的对象。上述代码表示：当用户单击(Click)命令按钮(CmdQ)时，清除该表单。

也可调用 VFP 预置的表单方法 Release()来清除表单：ThisForm.Release()

图 7-30　输入代码

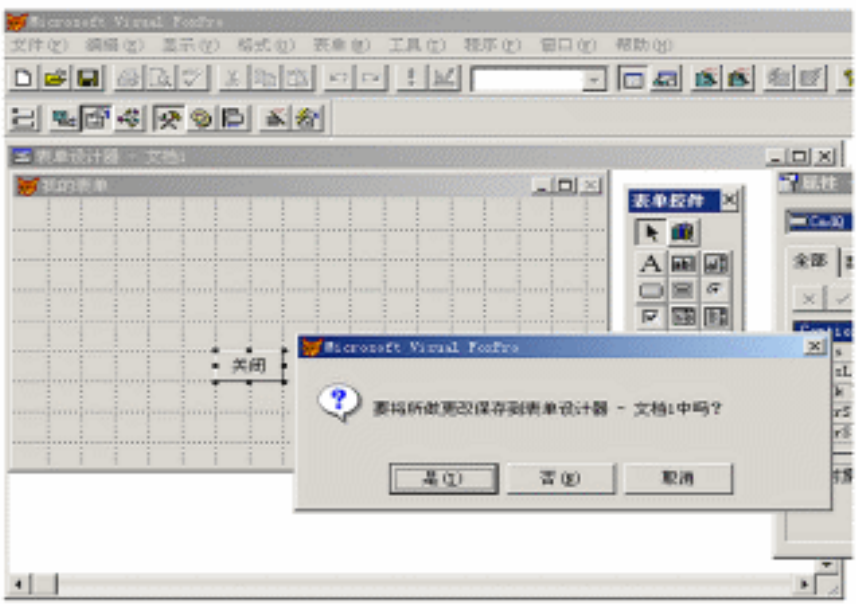


图 7-31　确认保存对话框

✧ 单击代码窗口右上角的关闭按钮，关闭代码窗口。然后单击“表单设计器”窗口右上角的“关闭”按钮关闭表单设计器，此时，系统提示是否保存所做的改变，如图 7-31 所示。选择“是”，则打开“另存为”对话框（如图 7-32 所示），选择表单名为：MyForm，则系统将以表单文件 MyForm. scx 存盘；选择“否”，则对刚才的修改不保存。

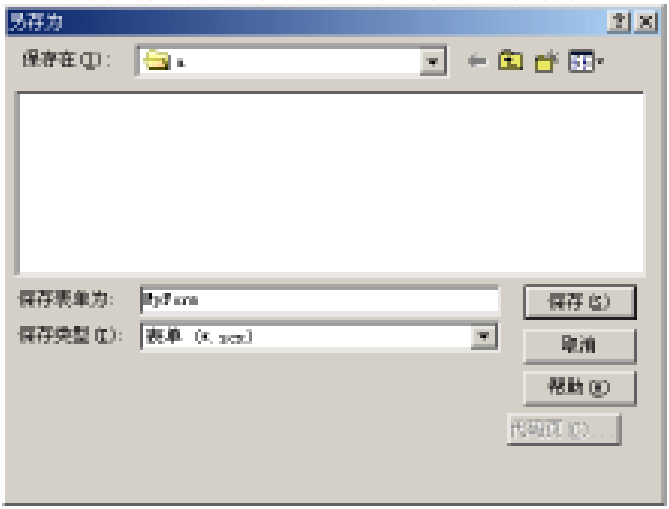


图 7-32 “另存为”对话框

7.1.6 运行表单

运行表单的方法有多种：

- 在命令窗口中键入：Do Form <表单名>
- 在程序代码中加入命令：Do Form <表达名>
- 在未退出“表单设计器”时，单击“常用工具栏”中的“运行”按钮 ! 。

在上述示例中只编写了具有一行代码的“程序”，运行后具有标准的 Windows 风格：

图标、标题、最大化按钮、最小化按钮、关闭按钮、移动栏及其周围的边框（试一试“最大化”按钮、“最小化”按钮；试一试移动表单到屏幕的其它位置；最后用鼠标单击“关闭”按钮。若是用上述第 3 种方式运行表单的，此时将返回“表单设计器”，否则将返回 VFP 主窗口）。

7.1.7 修改表单

修改表单有以下 3 种方法：

- 在“文件”菜单中选择“打开”或单击常用工具栏上的“打开”按钮，在“打开”对话框的“文件类型”下拉列表框中选择“表单（*.scx）”，然后单击在列出的表单文件中选择所需的表单名，如图 7-33 所示。

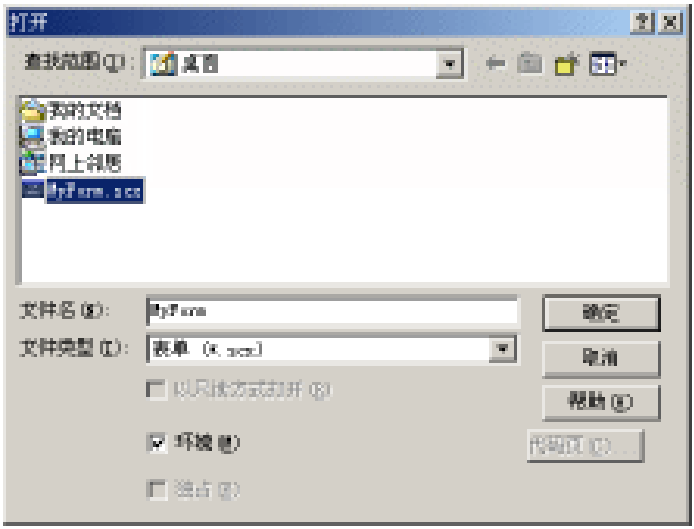


图 7-33　“打开”对话框

- 在命令窗口中使用命令：Modify Form <表单名>。
- 在项目管理器中选择所需要修改的表单名称，再按“修改”键，如图 7-34所示。

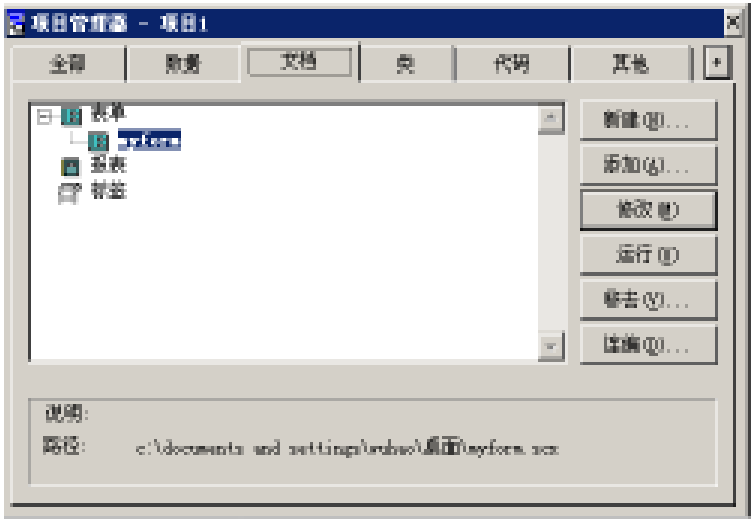


图 7-34　项目管理器

上述方法均可再次进入“表单设计器”。

下面以刚刚创建的表单 MyForm 为例，对其进行修改，使其具有一个快捷键 Q，如图 7-35 所示，即当从键盘上键入 < Alt > + < Q > 时，可以实现关闭表单。修改表单 MyForm 的步骤如下：

图 7-35 只有快捷键的命令按钮

- 若“属性窗口”处在关闭状态，则先打开属性窗口。
- 在“对象”下拉列表框中选择对象：“CmdQ”，在“布局”选项卡中选择 Caption 属性，将其值改为：h < Q 关闭。（其中“CmdQ”表示热键 Q）
- 在“布局”选项卡中选择 FontBold（粗体字）属性，将其值改为：T. ——真。
- 单击常用工具栏上的“运行”按钮，运行表单（屏幕显示表单如图 7-35 所示）。

从键盘上单击 < Q >，将关闭表单返回表单设计器。

7.1.8 将表单保存为“类”

若将上述表单保存为“类”，则可以避免许多重复的工作，每次进行新的程序设计时，只需将此“类”拿来即可使用。将表单保存为类的步骤如下：

- 在“文件”菜单中选择“另存为类”，打开“另存为类”对话框。
- 在“保存”选项组中选择“当前表单”，然后在“类定义”选项组的“类名”与“文件名”选项中都填入：forli（或选择其他类名与文件名），按“确定”键存盘，如图 7-36 所示。
- 在“工具”菜单中选择“选项”，打开“选项”对话框。“选项”对话框由 12 个选项卡组成，通过选项卡上方的标签来表示。单击“表单”标签显示“表单”选项卡，如图 7-37 所示。

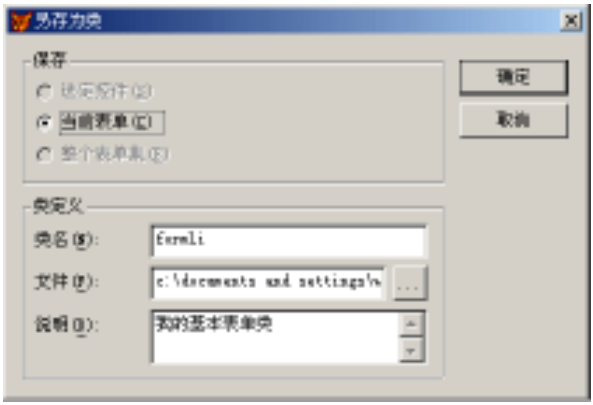


图 7-36　“另存为类”对话框

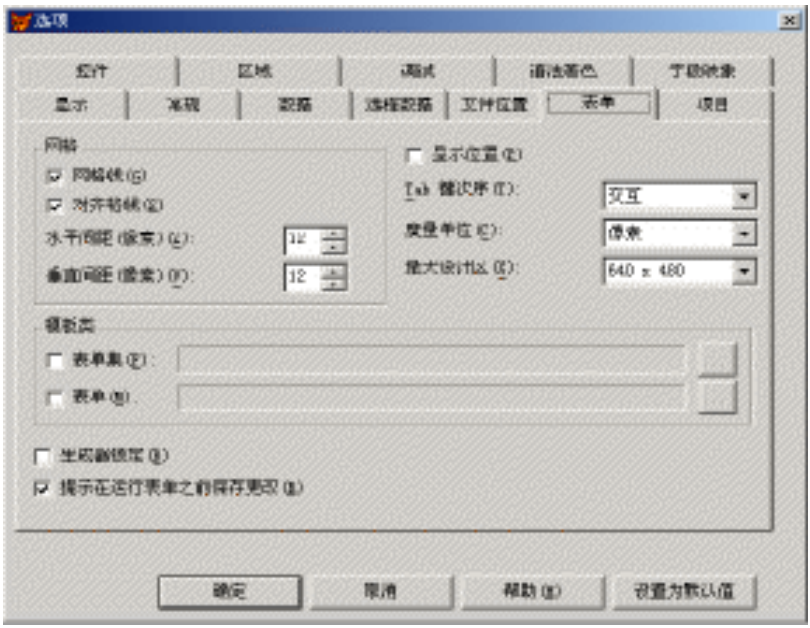


图 7-37　“选项”对话框

✧ 在“模板类”框中选择“表单”，此时将打开“表单模板”对话框，如图 7-38所示，选择刚刚保存的类文件 forli，并按“确定”返回到“表单”选项卡。单击“选项”对话框底部的“设置为默认值”按钮，将使以后再次新建的表单界面都已经具备前面刚刚所创建的表单的所有特征——对象、属性和方法。

✧ 单击“选项”对话框的“确定”按钮，返回到 VFP 主窗口。

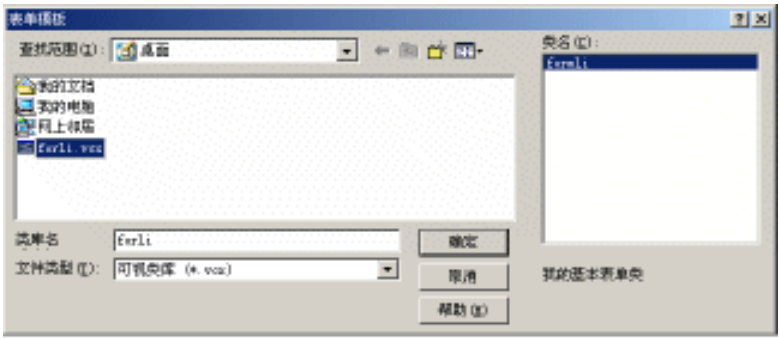


图 7-38 “表单模板”对话框

7.1.9 创建有图形按钮的表单

命令按钮除了可以具有快捷键外，还可以设计成图形按钮，如图 7-39 所示。设计步骤如下：

- 再次进入表单设计器，修改表单 MyForm。
- 若“属性”窗口处于关闭状态，则先打开属性窗口。
- 在“对象”下拉列表框中选择对象：“CmdQ”，在“布局”选项卡中选择 Caption 属性，将其值删为空，改为：（无）。
- 在“对象”下拉列表框中选择对象：“CmdQ”，在“布局”选项卡中选择 Picture 属性，如图 7-40 所示。
- 单击属性设置框右边的 3 点按钮 ...，屏幕显示“打开”对话框。选择文件类型为图标。并在系统目录 VFP 中找到图标文件 Trffc19a.ico（见图 7-41）。按下“确定”按钮后，表单中的命令按钮上会出现该图标，适当调整按钮的大小，即可得到所需的图形按钮表单。

图 7-39 图形按钮表单

图 7-40 选择 Picture 属性

图 7-41　选择图标文件

7.1.10　表单上控件的创建

在设计用户界面时，经常需要在表单上利用 VFP 提供的可视化控件画出各种所需要的对象。为了与在表单运行时由程序添加的对象相区别，将由控件创建的对象仍然称之为控件，且将由控件创建对象的过程称之为“画控件”。画控件时，经常需要进行下列常见的操作：

1. 在表单上画一个控件

在表单上画一个控件有以下两种方法：

- 第 1 种方法在前面已经作过介绍，即单击“表单控制工具栏”中的某个图标，然后在表单适当的位置拖动鼠标即可画出控件。
- 第 2 种方法是单击“表单控制工具栏”中的某个图标，然后在表单适当的位置单击鼠标右键即可在表单的相应位置画出该控件。

与第 1 种方法不同的是，用第 2 种方法所画出的控件大小是固定的。

2. 控件的缩放和移动

在前面画控件的过程中，可以看到，刚刚画出的控件其边框上有 8 个黑色小方块，表明该控件是“活动”的，活动控件也称“当前控件”（见图 7-42）。当表单上有多个控件时，一般只有一个控件是活动的（除非进行了多重选定），对控件的所有操作都是针对活动控件进行的。因此，为了对一个不活动的控件进行指定的操作，必须将其变为活动控件。单击不活动控件（鼠标指向其内部）可使该控件成为活动控件，而单击活动控件的外部，则可使该控件变为不活动控件。

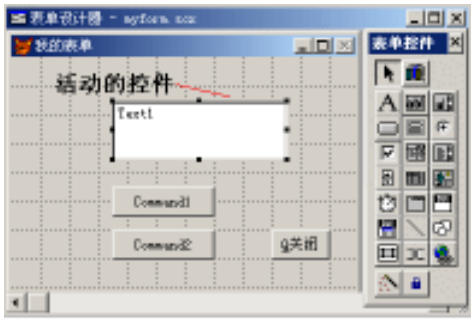


图 7-42　活动的控件

用鼠标拖拉活动控件边框上的小方块，则可以使控件在相应的方向放大与缩小。按住 < Shift > 键，用左右方向键可以调整控件的宽度，用上下方向键可以调整控件的高度。

当控件为活动控件时，用键盘的方向键可以使控件向相应的方向移动；也可将鼠标指向控件的内部，拖动控件到表单的任何位置。

除了以上方法外，还可通过修改某些属性来改变控件的大小与位置。有 4 种属性与表单及控件的大小与方向有关，即 Width、Height、Top、Left。其中：Left，Top 是表单或控件左上角的坐标，Width 是其宽度，Height 是其高度。坐标的原点是在 Windows 窗口或表单的左上角，单位由 ScaleMode 属性确定。在属性窗口的“Layout”栏中可以找到这些属性。

3. 控件的复制与删除

可以对控件进行复制与删除的操作。先将所需要操作的控件变为“活动控件”，按 < Ctrl > + < C > 键可将该控件拷贝到 Windows 的剪贴板中，按 < Ctrl > + < V > 键可以在表单中得到该控件的复制副本。

对于活动控件，只需按 < Delete > 键即可删除该控件。另外还可通过“编辑”菜单中的相应命令，或常用工具栏上的相应按钮，对控件进行复制与删除的操作。最快的方法是：用鼠标右键单击需要操作的控件，打开快捷菜单选取需要的项。

4. 在表单上画多个同类控件

若须在表单上画出多个同类的控件，则可以利用“按钮锁定”功能。在表单控件工具栏中选择“按钮锁定”按钮（见图 7-43），然后单击表单控件工具栏中的某个所需控件的图标，就可在表单上连续画出控件（不必每画一个，单击一次图标），直至用鼠标再次单击“按钮锁定”按钮而取消该功能。

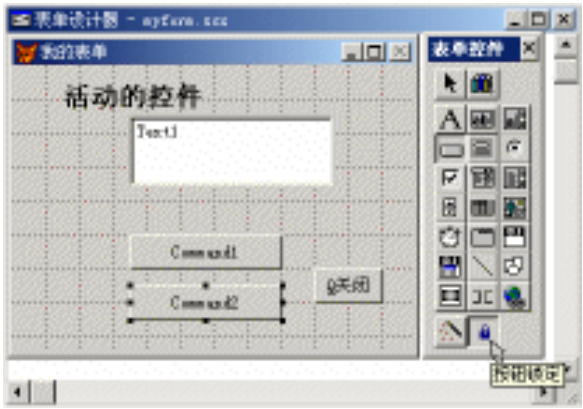


图 7-43　运行界面

5. 布局工具栏

当在表单上有多个控件时，可以使用“布局工具栏”对控件进行各种形式的对齐操作。在表单设计器工具栏中单击“布局工具栏”按钮可打开“布局”工具栏（如图 7-44 所示）。此时布局工具栏中的按钮全部处于不可用的状态，原因是为选定控件。

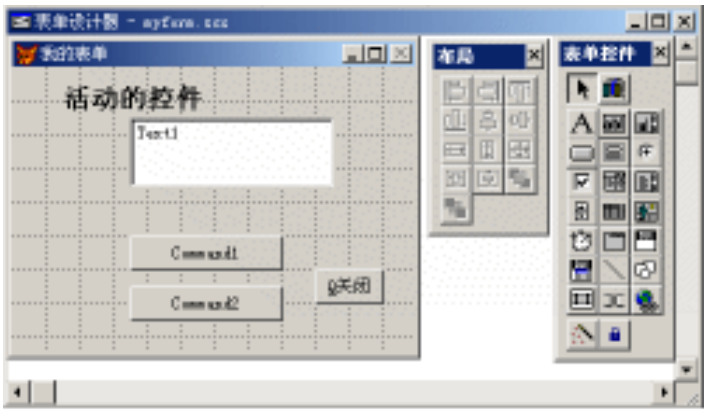


图 7-44　打开“布局”工具栏

①多重选定

使一组控件同时被选定称为多重选定。经多重选定的控件才可调整其相

互之间的位置。多重选定时,应先按住 < Shift > 键,再用鼠标单击所需选择的控件 或用鼠标在表单上拉出一个矩形,凡是与此矩形相交的控件均被选定(如图 7-45 所示)。

图 7-45 多重选定

②布局按钮介绍

布局工具栏有 13 个工具按钮,分别为:

- 左边对齐——被选择的控件靠左边对齐 (如图 7-46 所示)。

图 7-46 靠左边对齐

- 右边对齐——被选择的控件靠右边对齐。
- 顶边对齐——被选择的控件靠顶边对齐。
- 底边对齐——被选择的控件靠底边对齐。
- 垂直居中对齐——被选择的控件往垂直的中心对齐。

- 水平居中对齐——被选择的控件往水平的中心对齐。
- 相同宽度——被选择的控件设置相同的宽度。
- 相同高度——被选择的控件设置相同的高度。
- 相同大小——被选择的控件设置相同的大小。
- 水平居中——被选择的控件按表单的水平中心线对齐。
- 垂直居中——被选择的控件按表单的垂直中心线对齐。
- 置前——被选择的控件设置为前景显示。
- 置后——被选择的控件设置为背景显示（如图 7-47 所示）。

对控件的置前、置后操作在设计表单时非常有用。如当表单上的控件设计完成之后，希望用“方框”（形状控件）将其分组，但是后画的形状控件将覆盖原有的控件，按“置后”按钮即可将形状控件放置到原有控件的背后（如图 7-48 所示）。

图 7-47　置控件为背景显示

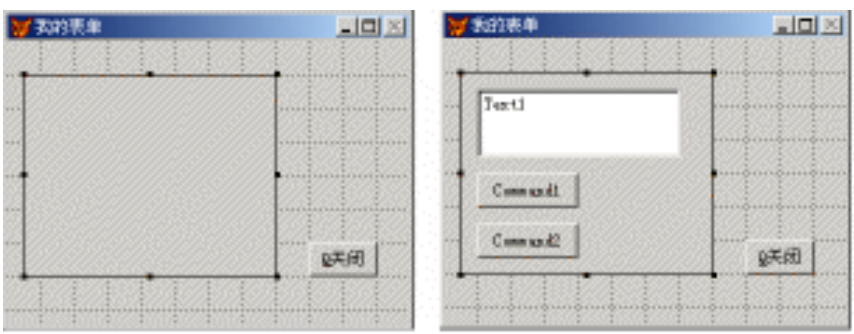


图 7-48　将形状控件放置到原有控件的背后

7.1.11 表单的常用属性

表单属性大约有 100 个左右，但绝大多数很少使用。表 7-1 中列出了常用的一些属性，这些属性规定了表单的外观和行为，经常在设计阶段进行设计。

表 7-1 表单常用属性

属 性	描述	默认值
AlwaysOnTop	指定表单是否总是位于其它打开窗口之上	.F.
AutoCenter	指定表单初始化时是否自动在 VFP 主窗口内居中显示	.F.
BackColor	指明表单窗口的颜色	255,255,255
BorderStyle	指定表单边框的风格。取默认值 (3) 时，采用系统边框，用户可改变表单大小	3
Caption	指明显示于表单标题栏上的文本	Form1
Closable	指定是否可通过单击关闭按钮或双击关闭框来关闭表单	.T.
DataSession	指定表单里的表是在全局能访问的工作区打开 (设置值为 1)。还是在表单自己的工作区打开 (设置值为 2)	1
MaxButton	确定表单是否有最大化按钮	.T.
MinButton	确定表单是否有最小化按钮	.T.
Movable	确定表单是否能够移动	.T.
Scrollbars	指定表单的滚动条类型。可取值为：0 (无)、1 (水平)、2 (垂直)、3 (既水平又垂直)	0
WindowState	指明表单的状态：0 (正常)、1 (最小化)、2 (最大化)	0
WindowType	指定表单是模式表单 (设置值为 1) 还是非模式表单 (设置值为 0)。在一个应用程序中，若运行了一个模式表单，那么在关闭该表单之前不能访问应用程序中的其它界面元素	0

7.1.12 表单的常用事件和方法

1. Init 事件

在对象建立时引发。在表单对象的 Init 事件引发之前，将先引发它所包

含的控件对象的 Init 事件，因此在表单对象的 Init 事件代码中能够访问它所包含的所有控件对象。

2. Destroy 事件

在对象释放时引发。表单对象的 Destroy 事件在它所包含的控件对象的 Destroy 事件引发之前引发，因此在表单对象的 Destroy 事件代码中能够访问它所包含的所有控件对象。

3. Error 事件

当对象方法或事件代码在运行过程产生错误时引发。事件引发时，系统会将发生的错误类型和错误发生的位置等参数传递给事件代码，事件代码可以据此对错误进行相应的处理。

4. Load 事件

在表单对象建立之前引发，即运行表单时，先引发表单的 Load 事件，再引发表单的 Init 事件。

5. Unload 事件

在表单对象释放时引发，是表单对象释放时最后一个要引发的事件（如在关闭包含一个命令按钮的表单时，先引发表单的 Destroy 事件，然后引发命令按钮的 Destroy 事件，最后引发表单的 Unload 事件）。

6. GotFocus 事件

当对象获得焦点时引发。对象可能会由于用户的动作（如鼠标单击）或代码中调用 SetFocus 方法而获得焦点。

7. Click 事件

用鼠标单击对象时引发。引发该事件的常见情况有：

- 鼠标单击复选框、命令按钮、组合框、列表框和选项按钮。
- 在命令按钮、选项按钮或复选框获得焦点时，按空格键。
- 当表单中包含一个确认按钮（Default 属性值为. T.）时，按 Enter 键，引发确认按钮的 Click 事件。
- 按控件的热键。
- 单击表单的空白处，引发表单的 Click 事件。但单击表单的标题栏或窗口边界不会引发 Click 事件。

8. DblClick 事件

用鼠标双击对象时引发。

9. RightClick 事件

用鼠标右键单击对象时引发。

10. InteractiveChange 事件

当通过鼠标或键盘交互式改变一个控件的值时引发。

11. Release 方法

将表单从内存中释放(清除)。如表单有一个命令按钮,若希望单击该命令按钮时关闭表单,就可以将该命令按钮的 Click 事件代码设置为 ThisForm.Release。

12. Refresh 方法

重新绘制表单或控件,并刷新它的所有值。当表单被刷新时,表单上的所有控件也都

被刷新。当页框被刷新时,只有活动页被刷新。

13. Show 方法

显示表单。该方法将表单的 Visible 属性设置为.T.,并使表单成为活动对象。

14. Hide 方法

隐藏表单。该方法将表单的 Visible 属性设置为.T.。

15. SetFocus 方法

让控件获得焦点,使其成为活动对象。若一个控件的 Enabled 属性值或 Visible 属性值.F.,将不能获得焦点。

7.1.13 数据环境

可以为表单建立数据环境,数据环境中能够包含与表单有联系的表和视图以及表之间的关系。通常情况下,数据环境中的表或视图会随着表单的打开或运行而打开,并随着表单的关闭或释放而关闭。可以用数据环境设计器来设置表单的数据环境。

1. 数据环境的常用属性

数据环境是一个对象,有自己的属性、方法和事件。常用的两个数据环境属性是

AutoOpenTables 和 AutoCloseTables,它们的设置情况如表 7-2 所示。

表 7-2　常用的数据环境属性

属　性	含　义	默认值
AutoOpenTables	当运行或打开表单时,是否打开数据环境中的表和视图	.T.
AutoCloseTables	当释放或关闭表单时,是否关闭由数据环境指定的表和视图	.T.

2. 打开数据环境设计器

在表单设计器环境下,单击“表单设计器”工具栏上的“数据环境”按钮,或选择“显示”菜单中的“数据环境”命令,即可打开“数据环境设计器”窗口(如图 7-49 所示),进入数据环境设计器环境。此时,系统菜单栏上将出现“数据环境”菜单。

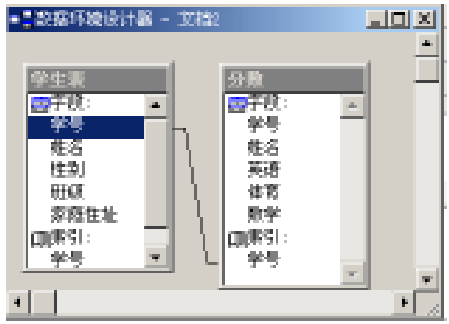


图 7-49　“数据环境设计器”窗口

3. 向数据环境添加表或视图

在数据环境设计器环境下,按下列方法向数据环境添加表或视图:

- 选择“数据环境”菜单中的“添加”命令,或右键单击“数据环境设计器”窗口,然后在弹出的快捷菜单中选择“添加”命令,打开“添加表或视图”对话框(如图 7-50 所示)。如果数据环境原来是空的,那么在打开数据环境设计器时,该对话框会自动出现。
- 选择需要添加的表或视图并单击“添加”按钮。如果单击“其他”按钮,将调出“打开”对话框,用户可以从中选择需要的表。如果数据环境原来是空的且无打开的数据库,那么在打开的数据环境设计器时,“打开”对话框会自动出现。

图 7-50　“添加表或视图”对话框

4. 从数据环境移去表或视图

在数据环境设计器环境下,按下列方法从数据环境移去表或视图:

• 在“数据环境设计器”窗口中,单击选择需要移去的表或视图。

• 选择“数据环境”菜单中的“移去”命令。

也可用鼠标右键单击需要移去的表或视图,然后在弹出的快捷菜单中选择“移去”命令。当表从数据环境中移去时,与这个表有关的所有关系也将随之消失。

5. 在数据环境中设置关系

如果添加到数据环境的表之间具有在数据库中设置的永久关系,这些关系也会自动添加到数据环境中。如果表之间没有永久关系,可以根据需要在数据环境设计器下为这些表设置关系。设置关系的方法较简单,只需将主表的某个字段(作为关联表达式)拖动到子表的相匹配的索引标记上即可。如果子表上没有与主表字段相匹配的索引,也可将主表字段拖动到子表的某个字段上,这时应根据系统提示确认创建索引。

要解除表之间的关系,可先单击选定表示关系的连线,然后按 Delete 键。

6. 在数据环境中编辑关系

关系是数据环境中的对象,它有自己的属性、方法和事件。编辑关系主要通过设置关

系的属性来完成。要设置关系属性,可先单击表示关系的连线,然后在“属性”窗口中选择关系属性并设置。常用的关系属性如表 7-3 所示。

表 7-3 常用的关系属性

属 性	含 义
RelationlExpr	用于指定基于主表的关联表达式
ParentAlias	用于指明主表的别名
ChildAlias	用于指明子表的别名
ChildOrder	用于指定与关联表达式相匹配的索引
OneToMany	用于指明关系是否为一对多关系

7. 向表单添加字段

利用“表单控件”工具栏可以很方便地将一个标准控件放置到表单上,但是多数情况下,需通过控件来显示和修改数据(如用一个文本框来显示或编辑一个字段数据,此时就需为该文本框设置 ControlSource 属性。

VFP 提供了更好的方法,允许用户从“数据环境设计器”窗口、“项目管理器”窗口或“数据库设计器”窗口直接将字段、表或视图直接拖入表单,系统将

产生相应的控件并与字段相联系。

默认情况下,若拖动的是字符型字段,将产生文本框控件;若拖动的是备注型字段,将产生编辑框控件;若拖动的是表或视图,将产生表格控件。但用户可以选择“工具”菜单中的“选项”命令,打开“选项”对话框,然后在“字段映象”选项卡中修改这种映象关系。

§7.2　常用的表单控件

VFP 在面向对象开发环境中,是通过在表单上放置控件来完成信息的输入设计工作的。在设计表单时可以使用两类控件:与表中数据捆绑的控件和不与数据捆绑的控件。所谓捆绑控件是指当输入或选择的值要保存或者被引用时,就需为该控件设置一个数据源,数据源可以是表中的字段或变量。对于数据源是变量或字段的控件,则需要设置控件的 ControlSource 属性;若数据源是整个表中的数据,则需要设置 RecordSource 属性。而非捆绑控件则不与数据源直接捆绑。

控件是有一个专门的浮动工具栏组成,当用户开始设计表单时,它的外观显示如图 7-51 所示。

图 7-51　表单控件浮动工具栏

7.2.1　标签控件

标签控件  :它是一个显示文本图形的控件,在设计时可以直接修改其中的文本,标签具有自己的一套属性、事件和方法,能够响应绝大多数鼠标事件,可以在运行时动态地改变标签文本。可以使用 TabIndex 属性为标签指定一个 Tab 次序,但标签并不能获得焦点,而是将焦点传递给 Tab 键次序中紧跟着标签的下一个控件。

常用的标签属性:

① Caption 属性

用于指定标签的标题文本,最多可包含的字符数量为 256。标签文本显示在屏幕上以帮助用户识别各个对象。用户在产生表单或控件对象时,系统给予对象的 Caption 属性值和 Name 属性值是相同的,此时用户应特别注意它们的区别。

用户在为控件设置 Caption 属性时,可以将其中的某个字符作为热键,方法是在该字符前插入一个反斜杠(\)和一个小于号(<)。例如,下面代码在

为标签设置的 Caption 属性的同时，指定了一个热键“X”：

ThisForm. MyLabel. Caption = "选择项目(h<X)"

对于一般的控件，按下相应的热键，将激活该控件，使该控件获得焦点。而对于标签，按下相应的热键，将把焦点传递给 Tab 键次序中的紧跟着标签的下一个控件。着在有些场合是比较有用的。例如，在某个列表框的上方放置一个标签，并将列表框的 Tab 键次序安排在标签之后，这样当按下标签的热键时，其下方的列表框将获得焦点。

给属性在设计和运行阶段都可用，除了标签，还适用于表单、复选框、选项按钮、命令按钮等许多控件。

② AutoSize 属性

该属性用于确定是否根据标题的长度来调整标签的大小。

③ Alignment 属性

该属性指定标题文本在控件中显示时的对齐方式。对于不同的控件，该属性的设置是有区别的。对于标签，该属性的设置值如表 7-4 所示。

表 7-4 标签的 Alignment 属性的设置值

设置值	说 明
0	(默认值) 左对齐，文本显示在区域的左边
1	右对齐，文本显示在区域的右边
2	中央对齐，将文本居中排放，使左右两边的空白相等

该属性在设计和运行时均可用。

④ BackStyle 属性

该属性用于确定标签是否透明。当标签控件完全处于另一个控件的上方时，标签将遮住下方的控件，此时若标签控件的 BackStyle 属性设置为 0，则下方的控件将透过标签显示出来。

⑤ WordWrap 属性

该属性用于确定标签上显示的文本能否换行。前提是 AutoSize 属性值应设置为. T. 。

⑥ FontSize 属性

该属性确定标签上显示的文本字体的大小。

⑦ ForeColor 属性

该属性用于确定标签上显示的文本字体的颜色。

例 3 表单中有 3 个标签，如图 7-52 所示。当用鼠标单击任何一个标签

时，都使其它两个标签的标题文本互换。

假定 3 个标签的名称（Name 属性值）分别是 Label1、Label2、Label3，它们可以从属性窗口中获得。

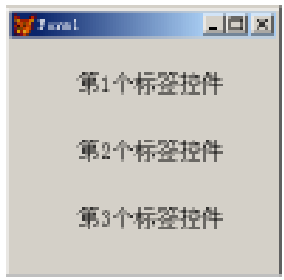


图 7-52　例 3 运行界面

① 标签 Label1 的 Click 事件代码为：

```
t = thisform. Label2. Caption
thisform. Label2. Caption = ；
thisform. Label3. Caption
thisform. Label3. Caption = t
```

② 标签 Label2 的 Click 事件代码为：

```
t = thisform. Label1. Caption
thisform. Label1. Caption = thisform. Label3. Caption
thisform. Label3. Caption = t
```

③ 标签 Label3 的 Click 事件代码为：

```
t = thisform. Label1. Caption
thisform. Label1. Caption = thisform. Label2. Caption
thisform. Label2. Caption = t
```

操作步骤：

- 创建表单，然后在表单中添加 3 个标签控件。
- 分别为 3 个标签控件设置 Caption 属性（如图 7-52 所示）。
- 分别为 3 个标签控件设置 Click 事件代码。

例 4　交换两个变量中的数据（如图 7-53 所示）。

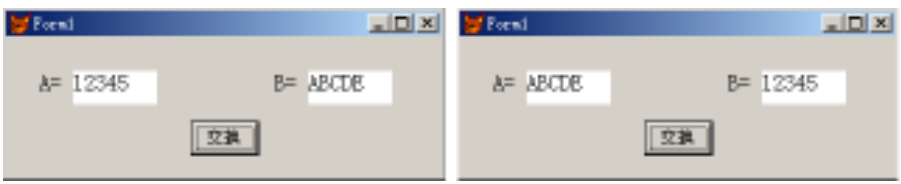


图 7-53　交换两个变量的值（交换前和交换后）

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加一个命令按钮 Command1，4 个标签 Label1、Label2、Label3、Label4（如图 7-54 左所示）。

图 7-54　建立应用界面

② 设置对象属性（如表 7-5），设置完成之后的界面如图 7-54 右所示。

表 7-5　属性设置

对　象	属　性	属性值	说　明
Label1	Caption	A =	标签的内容
Label2	Caption	B =	标签的内容
Label3	Caption	ABCDE	标签的内容
	BackColor	（白色）	标签的背景颜色
Label4	Caption	12345	标签的内容
	BackColor	（白色）	标签的背景颜色
Command1	Caption	交换	按钮的标题

③ 编写程序代码。

编写命令按钮 Command1 的 Click 事件代码：

```
t = thisform. label3. caption
thisform. label3. caption = thisform. label4. caption
thisform. label4. caption = t
```

运行程序，单击“交换”按钮，即可看到两个白框中的数据相互交换。

7. 2. 2　命令按钮控件

命令按钮一般是用于启动某一动作的事件（如关闭表单、移动记录等）。这些事件的操作代码通常是放置在命令按钮的 Click 事件中。当用户需要完成某项特定的操作时，可单击此命令按钮。

常用属性：

① Dfault 属性和 Cancel 属性

Default 属性值为. T. 的命令按钮称为“确认”按钮。命令按钮的 Default 属性的默认值为. F. 。一个表单内只能有一个“确认”按钮，当用户将某个命令按钮设置为“确认”按钮时，先前存在的“确认”按钮将自动变为“非确认”

按钮。

“确认”按钮的行为要受到 KEYCOMP 设置 (DOS 或 WINDOWS) 的影响。在“确认”按钮所在的表单激活的情况下,“确认”按钮的行为如表 7-6 所示。

表 7-6　KEYCOMP 设置值及其对“确认”键的影响

设置值	效　果
DOS	按 Ctrl + Enter,选择“确认”按钮、执行 Click 事件代码
WINDOWS	当焦点不在命令按钮上时,按 Enter,选择“确认”按钮、执行 Click 事件代码

Cancel 属性值为. F. 的命令按钮称为“取消”按钮。命令按钮的 Cancel 属性默认值为. F. 。在“取消”按钮所在的表单激活的情况下。按 Esc 键即可激活“取消”按钮,执行该按钮的 Click 事件代码。

这两个属性在设计和运行阶段都可使用,主要适用于命令按钮。

② Enabled 属性

指定表单或控件能否响应由用户引发的事件。默认值为. T. ,即对象是有效的,能被选择,能响应用户引发的事件。

Enabled 属性使得用户 (程序) 可以根据应用的当前状态随时决定一个对象是有效的还是无效的,也可限制一个对象的使用 (如用一个无效的编辑框来显示只读信息) 。

值得注意的是,若一个容器对象的 Enabled 属性值为. F. ,那么它所包容的所有对象也都不会响应用户引发的事件,而无论这些对象的 Enabled 属性值如何。

该属性在设计和运行阶段都可用,适用于绝大多数控件。

③ Visible 属性

指定对象是可见还是隐藏。在表单设计器中,默认值为 T. ,即对象是可见的 在程序代码中,默认值为. F. ,即对象是隐藏的。但是一个对象即使是隐藏的,在代码中仍可以访问它。

当一个表单由活动变为隐藏时,最近活动的表单或其它对象将成为活动的。当一个表单的 Visible 属性由. F. 设置成. T. 时,表单将成为可见的,但是并不成为活动的。若要使一个表单成为活动的,可以使用 Show 方法。Show 方法在使表单成为可见 (Visible 属性设置为. T.) 的同时,使其成为活动的。

该属性在设计和运行阶段可用,适用于绝大多数控件。

④ Caption 属性

该属性决定在命令按钮上显示的标题文本。

⑤ DisabledPicture 属性

该属性指定当此命令按钮失效时,所显示的“. bmp”文件。

⑥ DownPicture 属性

该属性指定当此命令按钮按下时,所显示的“. bmp”文件。

⑦ Picture 属性

该属性决定显示在此命令按钮上的“. bmp”文件。

例 5　设计一个显示当前日期的简单程序。按“显示”按钮,则显示当前日期 按“关闭”按钮,则结束程序运行。

设计步骤如下:

① 建立应用程序用户界面。选择“新建”表单,进入表单设计器,增加 2 个命令按钮 Command1、Command2,一个标签 Label1 (如图 7-55 左所示)。

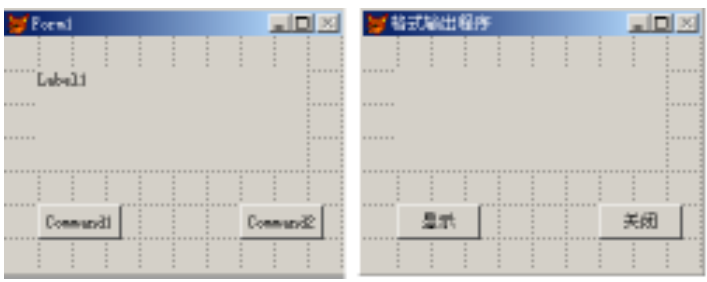


图 7-55　应用界面设计

② 设置对象属性 (如表 7-7),设置完成之后如图 7-55 右所示。

表 7-7　属性设置

对　象	属　性	属性值
Form1	Caprtion	格式输出程序
Command1	Caption	显示
Command2	Caption	关闭
Label	Caption	(无)

③ 编写程序代码

编写 Command1 的 Click 事件代码:

thisform. label1. caption = '今天是:' + str (year (date ( )),4) + '年' + str

(month (date ( )) ,2) + ;

'月' + str (day (date ( )) ,2) + '日'

编写 Command2 **的** Click **事件代码：**

thisform. release ( )

运行表单，得到的运行界面如图 7-56 **所示。**

图 7-56　**运行界面**

例 6　**利用命令按钮以实现显示状态的切换 (如图** 7-57 **所示)。**

图 7-57　**运行界面**

设计步骤如下：

① **建立应用程序用户界面：**

选择“新建”表单，进入表单设计器，增加 2 **个命令按钮** Command1、Command2，**一个标签** Label1 **(如图** 7-58 **左所示)。**

图 7-58　应用界面设计

② 设置对象属性（如表 7-8），设置完成之后如图 7-58 右所示。

表 7-8　属性设置

对　象	属　性	属性值
Label1	Caption	欢迎使用　Visual FoxPro 6.0
	FontName	隶书
	FontSize	14
Command1	Caption	h<H 欢迎
Command2	Caption	h<Q 关闭

③ 编写程序代码。

编写 Command1 的 Click 事件代码：

```
if this.caption = "h<H 欢迎"
  thisform.label1.caption = "欢迎使用" + chr(13) + "Visual FoxPro 6.0"
  this.caption = "h<D 日期"
else
  thisform.label1.caption = "今天是：" + chr(13) + str(year(date()),4)
+'年'；
 + str(month(date()),2) + '月' + str(day(date()),2) + '日'
  this.caption = "h<H 欢迎"
endif
```

编写 Command2 的 Click 事件代码：

```
thisform.release()
```

运行表单，得到的运行界面如图 7-57 所示。

7.2.3　文本框控件

文本框控件是 VFP 中一种常用的控件。用户利用它可以在内存变量、数组元素或非备注型字段中输入或编辑数据。所有标准的 VFP 编辑功能（如剪切、复制和粘贴），在文本框内都可使用，文本框一般包含一行数据，文本框可以编辑任何类型的数据（如字符型、数值型、逻辑型、日期型或日期时间型等）。如果编辑的是日期型或日期时间型数据，则在整个内容被选定的情况下，按“ + ”或“ - ”，就可使日期增加一天或减少一天。

常用属性：

① Value 属性

该属性返回文本框的当前内容。它的默认值为空串。

② PasswordChar 属性

该属性指定文本框控件内是显示用户输入的真实字符还是显示占位符。该属性的默认值为空串，此时没有占位符，文本框内显示用户输入的真实字符；当为该属性指定一个字符（即占位符，通常为 *）后，文本框内将显示占位符，而不会显示用户输入的实际字符。这通常用于在设计登录口令框时使用。此属性不会影响 Value 属性的设置，Value 属性总是包含用户输入的实际内容。

该属性在设计和运行阶段时可用，仅适用于文本框。

③ InputMask 属性

该属性指定在一个文本框如何输入和显示数据。

InputMask 属性值是一个字符串。该字符串通常由一些所谓的模式符组成，每个模式符都规定了相应位置上数据的输入和显示行为。各种模式符的功能如表 7-9 所示。

表 7-9　模式符及其功能

模式符	功　能
x	允许输入任何字符
9	允许数字和正负号
#	允许输入数字、空格和正负号
$	在固定位置上显示当前货币符号（由 Set Currency 命令指定）
$ $	在数值前面相邻的位置上显示当前货币号（浮动货币符）
*	在数值左边显示星号 *
.	指定小数点的位置
,	分隔小数点左边的数字串

InputMask 属性值中也可包含其它字符，这些字符在文本框内将回原样显示。

该属性在设计和运行阶段时可用。除了文本框，此属性还适用于组合框、列表框等控件。

注：SetFocu ( ) 方法可以为对象设置焦点。

例 7　利用文本框输入圆的半径，计算圆的面积。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加两个“文本框”控件 Text1 和 Text2，两个标签控件 Label1 和 Label2，两个命令按钮 Command1 和 Command2（如图 7-59 左所示）。

图 7-59　属性设置

② 设置对象属性（如表 7-10 所示），设置完成之后如图 7-59 右所示。

表 7-10　属性设置

对　象	属　性	属性值
Form1	Caption	计算圆的面积
Text1	InputMask	999.99
	Value	0
Text2	ReadOnly	.T.
	InputMask	9999999.99
	TabStop	.F.
	Value	0

续　表

对　象	属　性	属性值
Label1	Caption	请输入圆的半径
	AutoSize	. T.
	FontName	黑体
	FontBold	. T.
	FontSize	14
Label2	Caption	圆的面积为：
	AutoSize	. T.
	FontName	黑体
	FontBold	. T.
	FontSize	14
Command1	Caption	h < C 计算
	FontBold	. T.
Command2	Caption	h < Q 关闭
	FontBold	. T.

③ 编写程序代码。

编写表单 Form1 的 Activate 事件代码：

```
this. text1. setfocus
```

编写 Command1 的 Click 事件代码：

```
a = thisform. text1. value
thisform. text2. value = a * a * 3. 14159
thisform. text1. setfocus
```

编写 Command2 的 Click 事件代码：

```
thisform. release
```

图 7-60　运行界面

④ 运行表单，得到的运行界面如图 7-60 所示。

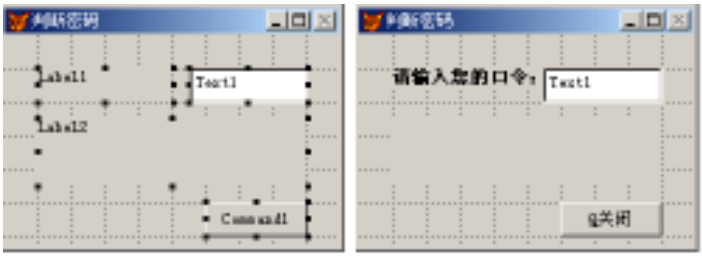
例 3　设计一个接受口令的表单，屏幕只显示相同个数的“ * ”号，并控制用户输入口令的次数。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加两个标签控件 Label1 和 Label2，一个命令按钮 Command1，一个文本框控件 Text1，如图 7-61 (a) 所示。

② 设置对象属性 (见表 7-1)，设置完成之后，如图 7-61 (b) 所示。



(a) (b)

图 7-61 应用界面设计

表 7-1 属性设置

对象	属 性	属性值
Form1	Caption	判断密码
Command1	Caption	h < Q 关闭
	TabStop	. F.
Label1	AutoSize	. T.
	Caption	请输入您的口令：
	FontBold	. T.
	FontSize	10
Label2	Caption	(无)
	FontBold	. T.
	FontSize	14
Text1	PasswordChar	*
	Value	(无)

③ 编写事件代码。

编写表单 Form1 的 Activate 事件代码：

```
public n                    && 定义全局变量
n = 0                       && 赋初值
this. command1. enabled = . f.
```

编写命令按钮 Command1 的 Click 事件代码：

```
thisform. release
```

编写文本框 Text1 的 Valid 事件代码：

```
a = lower (this. value)
if a = " abcdef"
thisform. label2. caption = "欢 迎 使 用 !"
```

```
thisform. command1. tabstop = . t.
else
thisform. label2. caption = "对不起,口令错 !"
this. value = " "
n = n + 1
 if n = 3
   thisform. label2. caption = "对不起,您无权使用 !"
   thisform. command1. tabstop = . t.
   this. enabled = . f.
   thisform. command1. enabled = . t.
 endif
endif
```

④ 运行表单,得到的运行界面如图 7-62 所示。

图 7-62　运行界面

7.2.4　计时器控件

计时器控件允许在指定的时间间隔内执行操作和检查数据。计时器控件与用户的操作相互独立。它对时间作出反应,可以让计时器以一定的时间间隔重复执行某种操作。计时器通常用于检查系统时钟,确定是否到了应该执行某一任务的时间,对于其他一些后台处理,计时器也很有用。在运行时,计时器是不可见的,但它一直在后台计时。

常用属性如下：

1. Enabled 属性

该属性用于确定计时器是否开始计时。当属性值为. F. 时,计时器停止计时 ;当属性值为. T. 时,计时器开始计时。

2. Interval 属性

该属性确定计时器 Timer 事件的两次引发的时间间隔。它的单位是毫秒。间隔并不能保证经历的时间的精确度。系统每秒产生 18 次时钟跳动,因

此间隔的真正精度不会超过 1/18s。若应用程序向系统提交繁重的任务（如很长的循环、大量的计算、或磁盘、网络、端口的访问等），则应用程序不能按 Interval 属性指定的频率来引发计时器事件。

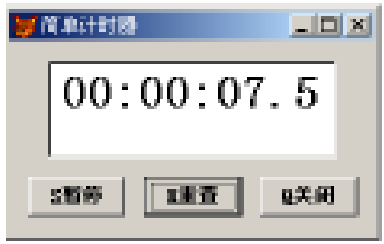


图 7-63　运行界面

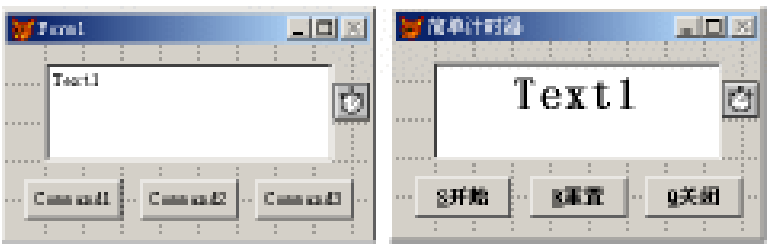
例 4　设计一个简单的计时器，如图7-63所示。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加一个文本框控件 Text1，三个命令按钮 Command1、Command2、Command3，一个计时器 Timer1（其中计时器控件可以放置在表单的任何位置），如图 7-64（a）所示。

② 设置对象属性（见表 7-2），设置完成之后，如图 7-64（b）所示。



（a）　　（b）

图 7-64　应用界面设计

表 7-2　属性设置

对　象	属　性	属　性　值
Form1	Caption	简单计时器
Command1	Caption	h < S 开始
	FontBold	. T.
Command2	Caption	h < R 重置
	FontBold	. T.
Command3	Caption	h < Q 关闭
	FontBold	. T.
Text1	Alignment	2
	FontSize	16
	Value	00：00：00.0
Timer1	Interval	100
	Enabled	. F.

③ 编写事件代码。

编写表单 Form1 的 Init 事件代码：

```
public b
b = 0
```

编写命令按钮 Command1 的 Click 事件代码：

```
if this. caption = " h < S 暂停"
  this. caption = " h < S 继续"
  thisform. timer1. enabled = . f.
else
  this. caption = " h < S 暂停"
  if thisform. text1. value = 00:  00:  00. 0
    b = second ( )
  endif
  thisform. timer1. enabled = . t.
endif
```

编写命令按钮 Command2 的 Click 事件代码：

```
b = second ( )
thisform. text1. value = "00:  00:  00. 0"
if thisform. command1. caption = " h < S 继续"
  thisform. command1. caption = " h < S 开始"
endif
```

编写命令按钮 Command3 的 Click 事件代码：

```
thisform. release
```

编写计时器 Timer1 的 Timer 事件代码：

```
tim = second ( ) - b
a0 = allt (str (int ( (tim * 10) ) % 10) )
time0 = tim
a1 = iif (time0%60 > 9,allt (str (time0%60) ) ,"0" + allt (str (time0%60) ) )
time1 = int (time0 /60)
a2 = iif (time1%60 > 9,allt (str (time1%60) ) ,"0" + allt (str (time1%60) ) )
time2 = int (time1 /60)
a3 = iif (time2%60 > 9,allt (str (time2%60) ) ,"0" + allt (str (time2%60) ) )
thisform. text1. value = a3 + ":" + a2 + ":" + a1 + "." + a0
```

④ 运行表单，得到的运行界面如图 7-63 所示。

例 5　设计一个电子动感标题板，使“2008 年北京奥运会”的标题字样在表单的黄色区域内从左至右运动。当单击“暂停”按钮时，使标题停止移动，

按钮变为“继续”;当单击“继续”按钮时,标题继续移动,按钮又变为“暂停”(见图 7-65)。

图 7-65 运行界面

设计步骤如下:

① 建立应用程序用户界面。

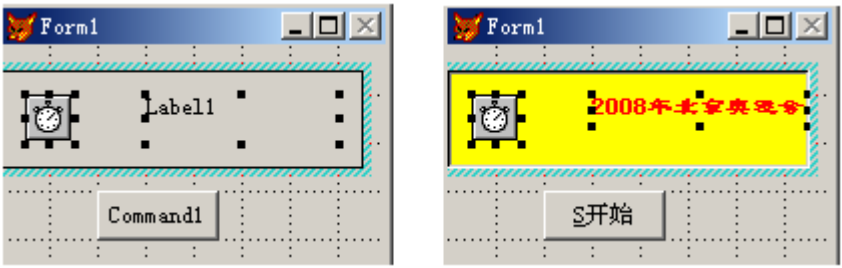
选择“新建”表单,进入表单设计器,增加一个命令按钮 Command1,一个容器控件 Container1,用鼠标右键单击容器控件,在弹出的快捷菜单中选择“编辑”,开始对容器进行设计。在容器中增加一个标签控件 Label1 和一个计时器控件 Timer1,如图 7-66 (a) 所示。

② 设置对象属性 (见表 7-3),设置完成之后,如图 7-66 (b) 所示。

表 7-3 属性设置

对 象	属 性	属性值	说 明
Container1	BackColor	255,255,0	容器的背景颜色为黄色
	SpecialEffect	1	设置容器具有凹陷效果
Command1	Caption	h < S 开始	
Timer1	Interval	100	
	Enabled	. F.	计时器停止计时
Label1	Caption	2008 年北京奥运会	标题内容
	AutoSize	. T.	标签自动适应标题内容的大小
	FontBold	. T.	粗体
	FontName	隶书	字体名称
	BackStyle	0	标签的背景类型 (透明)
	ForeColor	255,0,0	标签的字体颜色 (红色)

③ 编写程序代码。



(a)　　(b)

图 7-66　应用界面设计

编写 Command1 的 Click 事件代码：

```
if this. caption = " h < S 暂停"
  this. caption = " h < S 继续"
  thisform. container1. timer1. enabled = . f.
else
  this. caption = " h < S 暂停"
  thisform. container1. timer1. enabled = . t.
endif
```

编写 Timer1 的 Timer 事件代码：

```
if this. parent. label1. left + this. parent. label1. width > 0
  this. parent. label1. left = this. parent. label1. left - 5
else
  this. parent. label1. left = this. parent. width
endif
```

④ **运行表单，得到的运行界面如图 7-65 所示。**

7. 2. 5　编辑框控件 (EditBox)

在 VFP 中文本框只能处理单行的文本数据，处理多行文本数据则需要使用编辑框控件来完成。与文本框一样，编辑框也可用于输入、编辑数据，但是它也有自己的特点：

- **编辑框实际上是一个完整的字处理器，利用它可选择、剪切、粘贴和复制文本；可以实现自动换行；可以有自己的垂直滚动条；可用箭头键在正文内移动光标。**
- **编辑框只能输入、编辑字符型数据，包括字符型内存变量、数组元素、字段及备注型字段内的内容。**

常用属性：

1. AllowTabs 属性

指定编辑框控件中能否使用 Tab 键。其属性值的设置见表 7-4。该属性在设计和运行阶段均可使用，但它仅适用于编辑框。

表 7-4 AllowTabs 属性的设置值

设置值	说 明
.T.	编辑框中允许使用 Tab 键，按 Ctrl + Tab 键时焦点将移出编辑框
.F.	(默认值) 编辑框中不能使用 Tab 键，按 Tab 键时焦点移出编辑框

2. HideSelection 属性

指定当编辑框失去焦点时，编辑框中选定的文本是否仍显示为选定状态。其属性值的设置见表 7-5。该属性在设计和运行阶段均可用。除了编辑框，还可适用于文本框、组合框等控件。

表 7-5 HideSelection 属性的设置值

设置值	说 明
.T.	(默认值) 失去焦点时，编辑框中选定的文本不显示为选定状态。当编辑框再次获得焦点时，选定文本重新显示为选定状态
.F.	失去焦点时，编辑框中选定的文本仍显示为选定状态

3. ReadOnly 属性

指定用户能否编辑编辑框中的内容。其属性值的设置如表 7-6 所示。ReadOnly 属性与 Enabled 属性是有区别的。尽管在 ReadOnly 为 .T. 和 Enabled 为 .F. 两种情况下，都使编辑框具有只读的特点，但是在前种情况下，用户仍能移动焦点到编辑框上并使用滚动条，而后种情况则不能。该属性在设计时可用，在运行时可读写。除了编辑框，还适用于文本框、表格等控件。

表 7-6 ReadOnly 属性的设置值

设置值	说 明
.T.	不能编辑编辑框中的内容
.F.	(默认值) 能编辑编辑框中的内容

4. ScrollBars 属性

指定编辑框中是否具有滚动条。当属性值为 0 时，编辑框无滚动条；当属

性值为 2 时 (默认值),编辑框具有垂直滚动条。

该属性在设计时可用,在运行时可读写。除了编辑框,还适用于表单、表格等控件。

5. SelStart 属性

返回用户在编辑框中选定文本的起始点位置或插入点位置 (无文本选定时)。也可用于指定需要选定文本的起始位置或插入点位置。属性的有效取值范围在 0 与编辑区中字符的总数之间。

该属性在设计时不可用,在运行时可读写。除了编辑框,还适用于文本框、组合框等控件。

6. SelLength 属性

返回用户在控件的文本输入区中所选定字符的数目,或指定需要选定的字符数目。属性的有效取值范围在 0 与编辑区中的字符总数之间,若小于 0,将产生一个错误。

该属性在设计时不可用,在运行时可读写。除了编辑框,还适用于文本框、组合框等控件。

7. SelText 属性

返回用户编辑区中选定的文本,若无选定的任何文本,则返回空串。该属性在设计时不可用,在运行时可读写。除了编辑框,还适用于文本框、组合框等控件。

注:

SelStart、SelLength 和 SelText 属性配合使用,可以完成诸如设置插入点位置、控制插入点的移动范围、选择字符、清除文本等一些处理。

在使用这些属性时,需要注意以下几个方面:

- 若将 SelLength 属性值设置为小于 0,将产生一个错误。
- 若 SelStart 属性的设置值大于文本总字符数,系统将自动将其调整为文本的总字符数,即插入点位于文本末尾。
- 若改变了 Seltext 属性的值,系统将自动把 SelLength 属性值设置为 0。
- 若将 SelText 属性设置为一个新值,那么这个新值就会去置换编辑区中所选文本并将 SelLength 属性置为 0。若 SelLength 属性值本来就是 0,则新值就会被插入到插入点处。

例 6　设计一个文本文件的编辑器,可以实现新建或打开文件,并能在编辑后保存该文件,如图 7-67 所示。

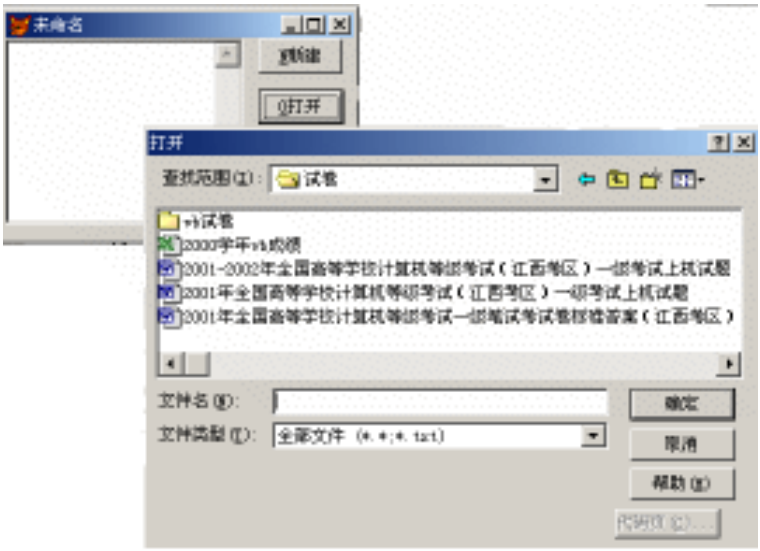


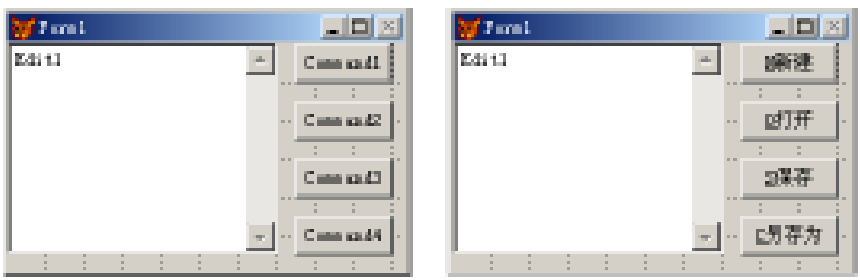
图 7-67　运行界面

设计步骤如下：

① 建立应用程序用户界面。

选择新建表单，进入表单设计器，增加一个编辑框控件 Edit1 **和** 4 **个命令按钮** Command1、Command2、Command3、Command4，**如图** 7-68 (a) **所示。**

② 设置对象属性 (见表 7-7)，**设置完成之后的界面，如图** 7-68 (b) **所示。**



(a)　　(b)

图 7-68　应用界面设计

表 7-7　属性设置

对　象	属　性	属　性　值

Command1	Caption	h < N **新建**
Command2	Caption	h < O **打开**
Command3	Caption	h < S **保存**
Command4	Caption	h < C **另存为**
Edit1	ControlSource	Thisform. tag

③ **编写程序代码。**

编写表单 Form1 **的** Activate **事件：**

```
with this. edit1
  . top = 0
  . left = 0
endwith
set exact on
this. caption = "未命名"
this. edit1. setfocus
```

编写命令按钮 Command1 **的** Click **事件：**

```
thisform. edit1. value = " "
thisform. refresh
thisform. caption = "未命名"
thisform. edit1. setfocus
thisform. command2. enabled = . t.
thisform. command3. enabled = . f.
thisform. command4. enabled = . t.
```

编写命令按钮 Command2 **的** Click **事件：**

```
cfile = getfile ("")
nhandle = fopen (cfile)
nend = fseek (nhandle,0,2)
 = fseek (nhandle,0,0)
thisform. edit1. value = fread (nhandle,nend)
thisform. caption = cfile
 = fclose (nhandle)
thisform. edit1. setfocus
thisform. refresh
thisform. command3. enabled = . t.
```

编写命令按钮 Command3 **的** Click **事件：**

```
cfile = thisform. caption
nhandle = fopen (cfile,1)
 = fwrite (nhandle,thisform. edit1. value)
 = fclose (nhandle)
thisform. refresh
thisform. edit1. setfocus
```

编写命令按钮 Command4 **的** Click **事件：**

```
cfile = putfile ("")
nhandle = fcreate (cfile,0)
cc = fwrite (nhandle,thisform. edit1. value)
 = fclose (nhandle)
thisform. edit1. setfocus
thisform. refresh
thisform. command3. enabled = . t.
```

④ **运行表单，运行界面，如图** 7-67 **所示。**

7.2.6 微调器控件

微调器控件 (Spinner) **可以在一定范围内控制数据的变化。除了能用鼠标单击控件右边向上和向下的箭头来增加和减少数值以外，还能像编辑框那样直接输入数值数据。**

常用属性如下：

Value **属性：该属性表示微调器控件的当前值。**

KeyBoardHighValue **属性：该属性指定微调器控件从键盘接受输入的数值的最大值。**

KeyBoardLowValue **属性：该属性指定微调器控件从键盘接受输入的数值的最小值。**

SpinnerHighValue **属性：该属性指定按钮微调数值的高限。**

SpinnerLowValue **属性：该属性指定按钮微调数值的低限。**

Increment **属性：设定按一次箭头按钮的递增 (递减) 值，默认为** 1.00。**若设置为** 1.50，**则递增 (递减) 数** 1.5。

InputMask **属性：设置输入掩码。微调器控件默认带** 2 **位小数，若只要整数可用输入掩码来限定 (如** 99999 **表示** 5 **位整数)。若微调器控件绑定到表的字段，则输入掩码位数不能小于字段宽度，否则将显示一串“** * **”号。**

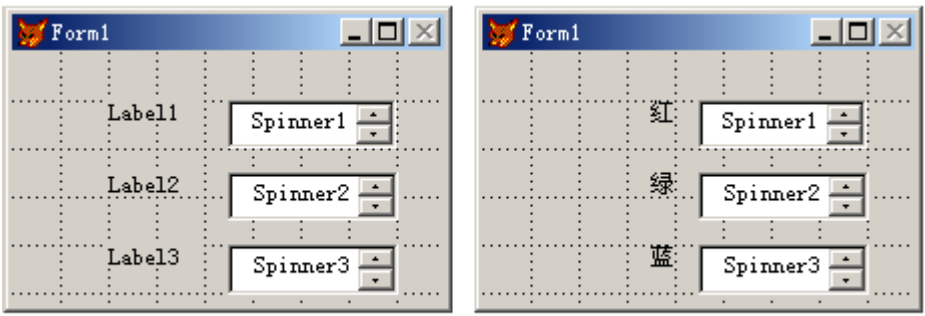
例 7　设计用 3 个微调器控件实现对表单背景颜色的改变。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加 3 个标签控件 Label1、Label2、Label3，**3 个微调器控件** Spinner1、Spinner2、Spinner3，**如图** 7-69 (a) **所示。**

② 设置对象属性 (见表 7-8)，**设置完成之后，如图** 7-69 (b) **所示。**



(a)　　(b)

图 7-69　**应用界面设计**

表 7-8　**属性设置**

对　象	属　性	属　性　值
Label1	Caption	**红**
	BackStyle	0
Label2	Caption	**绿**
	BackStyle	0
Label3	Caption	**蓝**
	BackStyle	0
Spinner1	Increment	1.00
	KeyBoardHighValue	255
	KeyBoardLowValue	0
	SpinnerHighValue	255
	SpinnerLowValue	0
	Value	100

续　表

对　象	属　性	属　性　值

Spinner2	Increment	1.00
	KeyBoardHighValue	255
	KeyBoardLowValue	0
	SpinnerHighValue	255
	SpinnerLowValue	0
	Value	100
Spinner3	Increment	1.00
	KeyBoardHighValue	255
	KeyBoardLowValue	0
	SpinnerHighValue	255
	SpinnerLowValue	0
	Value	100

③ **编写程序代码。**

编写微调器控件 Spinner1 **的** Click **事件代码：**

thisform. backcolor = rgb (thisform. spinner1. value,thisform. spinner2. value, ;
thisform. spinner3. value)

编写微调器控件 Spinner2 **的** Click **事件代码：**

thisform. backcolor = rgb (thisform. spinner1. value,thisform. spinner2. value, ;
thisform. spinner3. value)

编写微调器控件 Spinner3 **的** Click **事件代码：**

thisform. backcolor = rgb (thisform. spinner1. value,thisform. spinner2. value,;
thisform. spinner3. value)

④ **运行表单,运行界面如图** 7-70 **所示。**

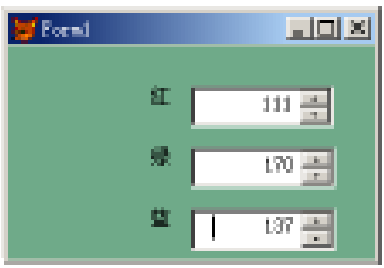


图 7-70 **运行界面**

7.2.7　复选框控件 (CheckBox)

复选框控件用于处理在两种状态之间的切换。当复选框被选中时,复选框中出现一个“√”,当需取消选中状态时,只需将复选框中的“√”取消即可。

常用属性:

Caption 属性:该属性用于指定显示在复选框旁边的文字信息。

Value 属性:该属性用于指明复选框的当前状态。复选框的 Value 属性值的设置有 3 种情况 (见表 7-9)。

表 7-9　Value 属性的设置值

属　性　值	说　　明
0 或 F.	默认值,未被选中
1 或. T.	被选中
2 或. Null.	不确定,只在代码中有效

ControlSource 属性:该属性用于指明与复选框建立联系的数据源。作为数据源的字段变量或内存变量,其类型可以是逻辑型或数值型。对于逻辑型变量,值. F. ,. T. 和. Null. 分别对应复选框未被选中、被选中和不确定。对于数值型变量,值 0,1 和. Null. 分别对应复选框未被选中、被选中和不确定。用户对复选框操作结果会自动存储到数据源变量以及 Value 属性中。

注:复选框的不确定状态与不可选状态 (Enabled 属性值为. F.) 是不同的。不确定状态只表明复选框的当前状态值不属于两个正常状态值中的一个,但是用户仍能对其进行选择操作,并使其变为确定状态。而不可选状态则表明用户现在不适合对它作出某种选择。在屏幕上,不确定状态复选框以灰色显示,标题文字正常显示,而不可选状态复选框的标题文字显示的颜色由 DisabledBackColor 和 DisabledForeColor 属性值决定,通常是浅色。

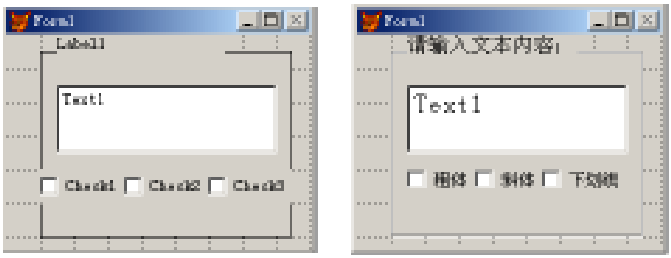
例 8　利用复选框来控制输入或输出文本的字体风格 (见图 7-71)。

设计步骤如下:

① 建立应用程序用户界面。

选择“新建”表单,进入表单设计器,增加一个文本框控件 Text1、一个标签控件 Label1、一个形状控件 Shape1、3 个复选框控件 Check1、Check2、Check3,如图 7-72 (a) 所示。

图 7-71　运行界面



(a)　　(b)

图 7-72　应用界面设计

② 设置对象属性（见表 7-10），设置完成之后如图 7-72 (b) 所示。

表 7-10　属性设置

对　象	属　性	属性值
Shape1	SpecialEffect	0
Label1	Caption	请输入文本内容：
Text1	FontSize	14
Check1	Caption	粗体
	AutoSize	.T.
Check2	Caption	斜体
	AutoSize	.T.
Check3	Caption	下划线
	AutoSize	.T.

③ 编写程序代码。

编写表单 Form1 的 Activate 事件代码：

```
this. text1. setfocus
```

编写 Check1 的 Click 事件代码：

```
thisform. text1. fontbold = this. value
```

编写 Check1 的 Click 事件代码：

```
thisform. text1. fontitalic = this. value
```

编写 Check1 的 Click 事件代码：

```
thisform. text1. fontunderline = this. value
```

④ 运行表单，运行界面如图 7-71 所示。

7.2.8　形状控件（Shape）

形状控件有利于可视方式将表单中的组件归成组。将相关项联系起来有助于用户设计出精美的用户界面，使应用程序更加完美，主要用于在表单上画出各种类型的形状（包括圆、椭圆、正方形、圆角正方形、矩形、圆角矩形等）。

常用属性：

Curvature 属性：属性值为 0（直角）～99（圆或椭圆）的一个值。

形状类型将由 Curvature、Width、Height 属性决定（见表 7-11）。

表 7-11　形状控件的形状设置

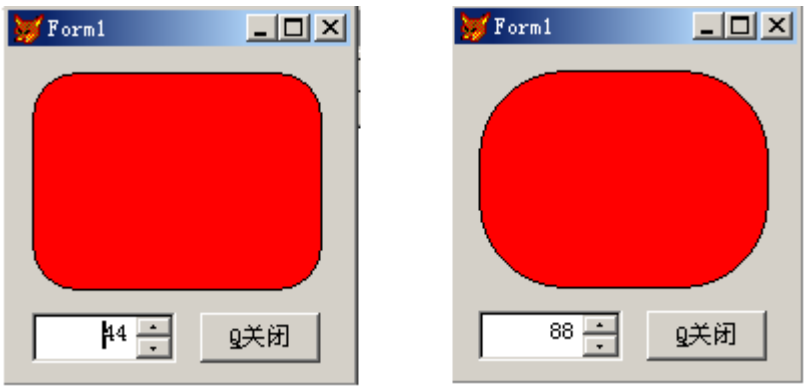
Curvature	Width 与 Height 相等	Width 与 Height 不相等
0	正方形	矩形
1～99	小圆角正方形→大圆角正方形→圆	小圆角矩形→大圆角矩形→椭圆

形状控件创建时若 Curvature 属性值为 0，Width 属性值与 Height 属性值也不相等，显示一个矩形。若要画出一个圆，应将 Curvate 属性值设置为 99，并使 Width 属性值与 Height 属性值相等。

FillStyle 属性：确定形状控件是透明的还是具有一个指定的背景填充方案。

确定形状控件是平面的还是三维的。仅当 Curvature 属性设置为 0 时才有效。

例 9　利用微调器改变图形的形状，如图 7-73 所示。用鼠标单击向上箭头时，四角逐渐变成圆形；反之，逐渐变方。



(a) (b)

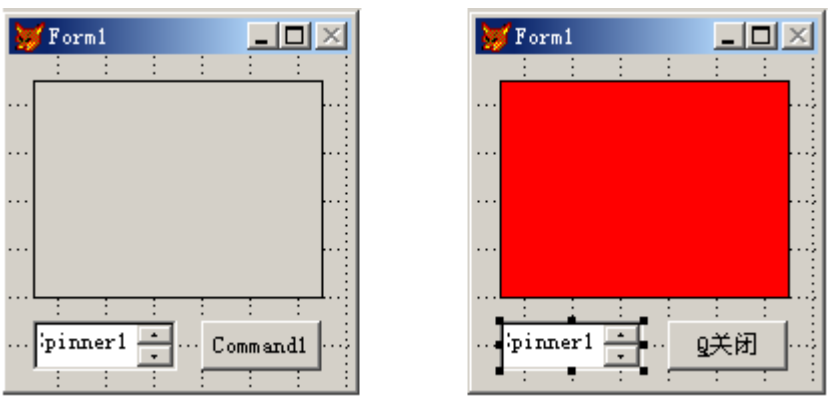
图 7-73 运行界面

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加一个形状控件 Shape1，一个微调器控件 Spinner1 和一个命令按钮 Command1，如图 7-74 (a) 所示。

② 设置对象属性 (见表 7-12)，完成之后如图 7-74 (b) 所示。



(a) (b)

图 7-74 应用界面设计

表 7-12　属性设置

对　象	属　　性	属　性　值
Command1	Caption	h < Q 关闭
Shape1	BackColor	255,0,0　(红)
Spinner1	Spinnerhigh	99
	Spinnerlow	0
	Keyboardhighvalue	99
	Keyboardlowvalue	0
	InputMask	99

③ 编写程序代码。

编写 Spinner1 的 InteractiveChange 事件代码：

```
thisform. shape1. curvature = this. value
```

编写 Command1 的 Click 事件代码：

```
thisform. release
```

④ 运行表单，运行界面如图 7-73 所示。

7. 2. 9　线条控件

线条控件主要用于在表单上画各种类型的线条（包括斜线、水平线和垂直线）。

1. 斜线

- 线条控件创建时，默认自控件区域的左上角至右下角显示一条斜线。
- 斜线倾斜角度由控件区域宽度与高度来决定，可以拖动控件区域的控制点来改变控件区域的宽度和高度，或改变宽度属性 Width 与高度属性 Height。
- 斜线走向可用 LineSlant 属性来指定，键盘字符“ h”表示左上角至右下角，而“ / ”表示右上角至左下角。

2. 水平线与垂直线

若需显示水平线或垂直线，可通过调节线条控件区域使对应边重合（表 7-13 列出了交互式与属性设置两种方式）。

表 7-13　线条控件水平线与垂直线的表示

线条类型	控 件 区 域 操 作	属 性 设 置
水平线	拖动控制点至上下重合	Height 设置为 0
垂直线	拖动控制点至左右重合	Width 设置为 0

常用属性：

BorderWidth 属性：该属性决定线条的宽度为多少像素点。

LineSlant 属性：该属性指定当线条不为水平或垂直时，线条的倾斜角度。

7.2.10 选项组控件（OptionGroup）

选项组又称为选项按钮组，是包含选项按钮的一种容器。一个选项组往往包含若干个选项按钮，但是用户只能从中选择一个按钮。当用户选择某个按钮选项按钮时，该按钮即成为被选中状态，而该选项组中的其他选项按钮，无论原来是何种状态，都将变为未选中状态。被选中的选项按钮中会出现一个圆点。

常用属性：

ButtonCount 属性：该属性指定选项按钮组中选项按钮的数目。在表单中创建一个选项按钮组时，ButtonCount 属性的默认值为 2（即包含 2 个选项按钮）。可以通过改变 ButtonCount 属性的属性值来重新设置选项按钮组中包含的选项按钮数目。

Value 属性：该属性用于指定选项组中哪个选项被选中。该属性值的类型可以是数值型的，也可以是字符型的。若为数值型 n，则表示选项组中第 n 个选项按钮被选中；若为字符型 c，则表示选项组中 Caption 属性值为 c 的选项按钮被选中。

ControlSource 属性：该属性指明与选项组建立联系的数据源。作为选项组数据源的字段变量或内存变量，其类型可以是数值型或字符型，用户对选项组的操作结果会自动存储到数据源变量以及 Value 属性中。

Buttons 属性：该属性用于存取选项组中每个按钮的数组。用户可以利用该属性为选项组中的按钮设置属性或调用其方法。例如，下面这行代码可以放在与选项组 myoptionG 处于同一个表单中的某个对象的方法或事件代码中，为选项组中第 3 个按钮设置 Caption 属性：

```
thisform. myoptionG. buttons (3) . caption = "Goodbye"
```

例 10 利用选项按钮组控制文本的对齐方式与字体，如图 7-75 所示。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加一个文本框 Text1，3 个标签 Label1、Label2、Label3，2 个选项按钮组控件 OptionGroup1、OptionGroup2，如图 7-76（b）所示。

② 设置对象属性（见表 7-14），完成之后的界面如图 7-76（a）所示。

图 7-75　运行界面

(a)　　(b)

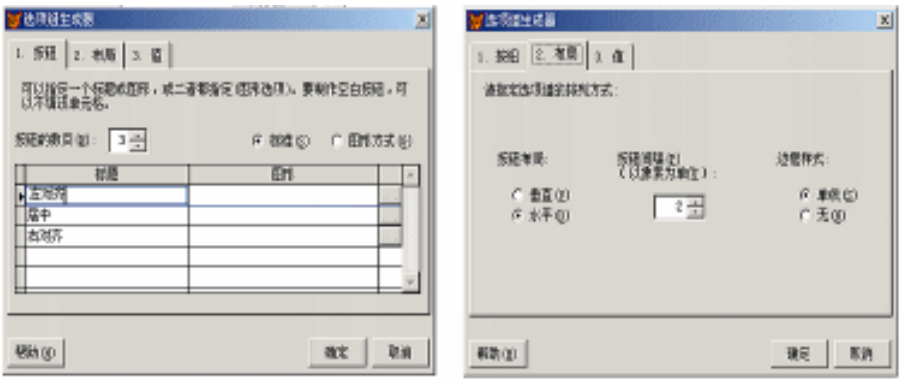
图 7-76　应用界面设计

表 7-14　属性设置

对　象	属　性	属 性 值
Label1	Caption	请输入文本内容：
Label2	Caption	对齐方式：
Label3	Caption	文本字体：
Text1	FontSize	12
OptionGroup1	ButtonCount	3
OptionGroup2	ButtonCount	3

用鼠标右键单击选项组控件 OptionGroup1，在弹出的快捷菜单中选择“生成器”，如图 7-76 (b) 所示。在“选项组生成器”的“按钮”选项卡中，修改“按钮的数目”为 3，这相当于在属性窗口中修改 ButtonCount 属性为 3。分别修改 3 个按钮的标题 Caption 属性为左对齐、居中、右对齐，如图 7-77 (a) 所示。在“选项组生成器”的“布局”选项卡中，修改“按钮布局”为水平，并适当设置按

钮的间隔，如图 7-77 (b) 所示，然后按“确定”退出“选项组生成器”。



(a) (b)

图 7-77 在生成器中设计按钮组

③ 编写程序代码。

编写表单 Form1 的 Activate 事件代码：

```
this. text1. setfocus
```

编写 OptionGroup1 的 Click 事件代码：

```
n = this. value
do case
  case n = 1
    thisform. text1. alignment = 0
  case n = 2
    thisform. text1. alignment = 2
  case n = 3
    thisform. text1. alignment = 1
endcase
```

编写 OptionGroup2 的 Click 事件代码：

```
n = this. value
do case
  case n = 1
    thisform. text1. fontname = "宋体"
  case n = 2
    thisform. text1. fontname = "隶书"
  case n = 3
```

```
    thisform. text1. fontname = "楷体_gb2312"
endcase
```

④ 运行表单，运行界面如图 7-75 所示。

7. 2. 11　命令组控件（CommandGroup）

命令组控件是包含一组命令按钮的容器控件，用户可以单个或作为一组来操作其中的按钮。

在表单设计器中，为了选择命令组中的某个按钮，以便为其单独设置属性、方法和事件，可以采用以下两种方法：一是从属性窗口的对象下拉式组合框中选择所需的命令按钮；二是用鼠标右键单击命令组，然后从弹出的快捷菜单中选择“编辑”命令，这样命令组就进入了编辑状态，用户可以通过鼠标单击来选择某个具体的命令按钮。这种编辑操作方法对其他容器类控件同样适用。

常用属性如下：

1. ButtonCount 属性

该属性指定命令组中命令按钮的数目。在表单中创建命令组数，ButtonCount 属性的默认值为 2，即包含 2 个命令按钮。可以通过改变 ButtonCount 属性的值来重新设置命令组中的命令按钮的数目。新增命令按钮的名称（Name）由系统自动给定（如 Command3、Command4 等），但是用户可以重新设置。该属性在设计和运行阶段可用。

2. Buttons 属性

该属性用于存取命令组中各按钮的数组。该属性数组在创建命令组时建立，用户可以利用该数组为命令组中的命令按钮设置属性或调用其方法。属性数组下标的取值范围应该在 1 至 ButtonCount 属性值之间。该属性在设计时不可用。

3. Value 属性

该属性指定命令组的当前状态。该属性的类型可以是数值型的（默认情况），也可以是字符型的。若为数值型 n，则表示命令组中的第 n 个命令按钮被选中；若为字符型 c，则表示命令组中 Caption 属性值为 c 的命令按钮被选中。

若命令组中的某个按钮有自己的 Click 事件，则一旦单击该按钮，就会优先执行为它单独设置的代码，而不会执行命令组的 Click 事件代码。该属性在设计和运行阶段可用。

例 11　演示命令组控件的使用特点。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加一个 Text1、一个命令组控件 CommandGroup1，如图 7-78 (a) 所示。

② 设置对象属性，完成之后如图 7-78 (b) 所示。

图 7-78　应用界面设计

③ 编写程序代码。

编写 CommandGroup1 的 Click 事件代码：

```
do case
  case this.value = 1
    thisform.text1.value = "指针移到顶部命令：go top"
  case this.value = 2
    thisform.text1.value = "指针上移命令：skip  - 1"
  case this.value = 3
    thisform.text1.value = "指针下移命令：skip  "
  case this.value = 4
    thisform.text1.value = "指针移到底部命令：go bottom"
endcase
```

④ 运行表单，运行界面如图 7-79 所示。

图 7-79　运行界面

7.2.12　图像控件 (Image)

图像控件允许在表单中显示图片 (.bmp 文件) 图像控件与他控件一样，具有一整套的属性、事件和方法程序。图像控件在运行时可以动态地改变它。用户可以用单击、双击和其他方式以交互地使用图像 (图像文件的类型可以是.bmp、.ico、.gif、.jpg 等)。

常用属性如下：

Picture 属性：该属性指定将要显示的图片 (.bmp 文件)。

BorderStyle 属性：该属性决定图像是否具有可见的边框。

Stretch 属性：该属性指定图像的显示方式：若该属性设置为 0 (剪裁)，则超出图像控件范围的那部分图像将不显示；若该属性设置为 1 (恒定比例)，则图像控件将保留图片的原有比例，并在图像控件中显示最大可能的图片；若该属性设置为 2 (伸展)，则将图片调整到正好与图像控件的高度和宽度相匹配。

例 12　利用图像控件设计“红绿灯”演示程序，如图 7-80 所示。

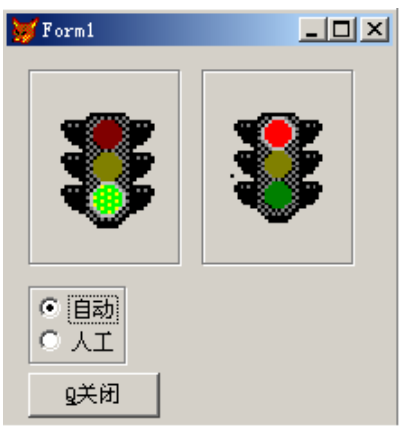


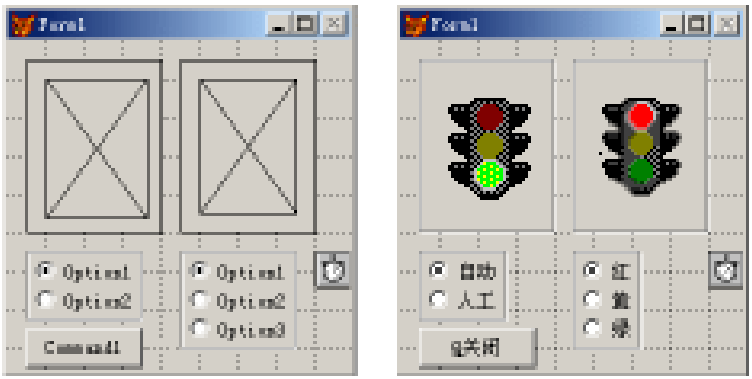
图 7-80　运行界面

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器，增加两个形状控件 Shape1、Shape2，两个图像控件 Image1、Image2，一个计时器 Timer1，一个命令按钮 Command1，两个选项按钮组 OptionGroup1、OptionGroup2，如图 7-81 (a) 所示。

② 设置对象属性（见表 7-15），完成之后如图 7-81（b）所示。



（a） （b）

图 7-81 应用界面设计

表 7-15 属性设置

对 象	属 性	属 性 值
Command1	Caption	h < Q 关闭
Timer1	Interval	500
Shape1、Shape2	SpecialEffect	0
Image1	BackStyle	0
	Picture	Traffc10a. ico
	Stretch	1
Image2	BackStyle	0
	Picture	Traffc10c. ico
	Stretch	1

选项按钮组 OptionGroup1、OptionGroup2 的属性可通过“选项组生成器”来进行设置修改。

③ 编写程序代码。

编写表单 Form1 的 Activate 事件代码：

```
public t
t = 0
```

编写 Command1 的 Click 事件代码：

```
thisform. release
```

编写选项按钮组 OptionGroup1 的 InteractiveChange 事件代码：

```
thisform. timer1. enabled = iif (this. value = 1 , . t. , . f. )
thisform. optiongroup2. visible = iif (this. value = 1 , . f. , . t. )
```

编写选项按钮组 OptionGroup2 **的** InteractiveChange **事件代码：**

```
do case
  case this. value = 3
    thisform. image1. picture = trffc10a. ico
    thisform. image2. picture = trffc10c. ico
  case this. value = 2
    thisform. image1. picture = trffc10b. ico
  case this. value = 1
    thisform. image1. picture = trffc10c. ico
    thisform. image2. picture = trffc10a. ico
  endcase
```

编写 Timer1 **的** Timer **事件代码：**

```
t = t + 1
do case
  case t = 1
    thisform. image1. picture = trffc10a. ico
  case t = 12
    thisform. image1. picture = trffc10b. ico
  case t = 15
    thisform. image1. picture = trffc10c. ico
  case t = 16
    thisform. image2. picture = trffc10a. ico
  case t = 27
    thisform. image2. picture = trffc10b. ico
  case t = 30
    thisform. image2. picture = trffc10c. ico
  case t = 31
    t = 0
endcase
```

说明：3 **个图标文件** trffc10a. ico、trffc10b. ico、trffc10c. ico **所在的目录为“**c：hprogram files hmicrosoft visual studio hcommon hgraphics hicons htraffic**”，应先将它们复制到当前目录之下。**

④ 运行表单,运行界面如图 7-80 所示。

7.2.13 列表框控件(ListBox)

列表框控件提供一组条目(数据项),用户可以从中选择一个或多个条目。在一般情况下,列表框显示其中的若干条目,用户可以通过滚动条实现浏览其他条目。

1. 常用属性

(1) RowSourceType 属性与 RowSource 属性

RowSourceType 属性指明了列表框中条目数据源的来源,RowSource 属性指定了列表框的条目数据源。RowSourceType 属性的取值范围及其含义见表 7-16。

表 7-16 RowSourceType 属性的设置值

属性值	说 明
0	无(默认值)。在运行时,通过 Additem 方法添加列表框条目,通过 RemoveItem 方法移去列表框条目
1	值。通过 RowSource 属性手工指定具体的列表框条目
2	别名。将表中的字段值作为列表框的条目。ColumnCount 属性指定需要取的字段数目,即列表框的列数。指定的字段总是表中最前面的若干字段
3	SQL 语句。将 SQL Select 语句的执行结果作为列表框条目的数据源
4	查询(.qpr),将.qpr 文件执行产生的结果作为列表框条目的数据源
5	数组。将数组中的内容作为列表框条目的来源
6	字段。将表中的一个或几个字段作为列表框条目的数据源
7	文件。将某个驱动器和目录下的文件名作为列表框的条目。在运行时,用户可以选择不同的驱动器和目录。可以利用文件名框架指定一部分文件
8	结构。将表中的字段名作为列表框的条目,由 RowSource 属性指定表。RowSource 属性值为空,则列表框显示当前表中的字段名清单
9	弹出式菜单。将弹出式菜单作为列表框条目的数据源

(2) List 属性

该属性用以存取列表框中数据条目的字符串数组。该属性在设计时不可用,在运行时可读写。除了列表框,还适用于组合框。

如:

读取列表框中的第 3 个条目第 1 列上的数据项

var = thisform. mylist. list (3,1) 或 var = thisform. mylist. list (3)

重新将列表框中第 3 个条目第 2 列上的数据项设置为“yes”

thisform. mylist. list (3,2) = yes

(3) ListCount 属性

该属性指明列表框中数据条目的数目。该属性在设计时不可用,在运行时只读。除了列表框,还适用于组合框。

(4) ColumnCount 属性

该属性指定列表框的列数。对于列表框和组合框,该属性在设计和运行时可用。除了列表框和组合框之外,还适用于表格。对于表格,该属性在设计时可用,在运行时可读写。

(5) Value 属性

该属性返回列表框中被选中的条目。该属性可以是数值型、字符型。若为数值型,则返回的是被选条目在列表框中的次序号;若为字符型,则返回的是被选条目的本身内容,若列表框不止 1 列,则返回由 BoundColumn 属性指明的列上的数据项。对于列表框和组合框,该属性为只读。

(6) ControlSource 属性

该属性在列表框中的用法与在其他控件中的用法有所不同。在列表框中,用户可通过该属性指定一个字段或变量用以保存用户从列表框中选择的结果。

(7) Selected 属性

该属性指定列表框中的某个条目是否处于选定状态。也可用此属性在多选列表框中找出所有被选的条目。该属性在设计时不可用,在运行时可读写。除了列表框之外,还适用于组合框。

(8) MultiSelect 属性

该属性指定用户能否在列表框控件中进行多重选定。该属性的设置情况见表 7-17。该属性在设计时可用,在运行时可读写,但仅适用于列表框。

表 7-17　MultiSelect 属性的设置值

属性值	说　明
0 或. F.	默认值,不允许多重选择
1 或. T.	允许多重选择。为选择多个条目,按住 Ctrl 键并同时用鼠标单击条目

2. 常用方法

AddItem 方法：该方法为 RowSourceType 属性为 0 的列表添加 1 项。

Clear 方法：该方法用于清除列表中的所有条目。

RemoveItem 方法：该方法从 RowSourceType 属性为 0 的列表中删除 1 项。

Requery 方法：该方法指明当 RowSource 中的值改变时更新列表。

例 13　求从 2000 年至 2100 年之间的所有闰年。

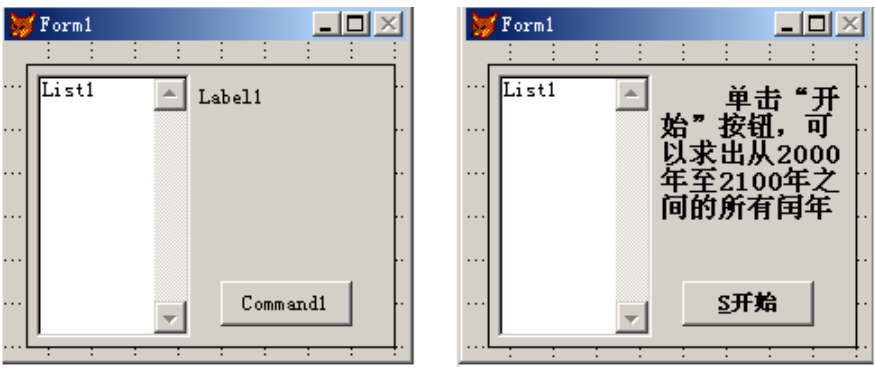
n 为闰年的条件是：若 n 能被 4 整除 若 n 不能被 100 整除或 n 能被 400 整除。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器。增加一个形状控件 Shape1、一个列表框控件 List1、一个标签控件 Label1、一个命令按钮 Command1，如图 7-82 (a) 所示。

② 设置对象属性 (见表 7-18)，完成之后如图 7-82 (b) 所示。



(a)　　(b)

图 7-82　应用界面设计

表 7-18　对象属性设置

对象	属　性	属性值
Command1	Caption	h < S 开始
Label1	Caption	单击“开始”按钮，可以求出从 2000 年至 2100 年之间的所有闰年
	WordWrap	. T.

列表框 List1 的属性保持默认值。

③ 编写事件代码。

编写命令按钮 Command1 的 Click 事件代码：

```
thisform. list1. clear
for n = 2000 to 2100
  if n%4  = 0
    if n%100 ! = 0 or n%400  = 0
      l = 1
    else
      l = 0
    endif
  else
    l = 0
  endif
  if l = 1
    thisform. list1. additem (allt (str (n)))
  endif
endfor
```

④ 运行表单，运行界面如图 7-83 所示。

图 7-83　运行界面

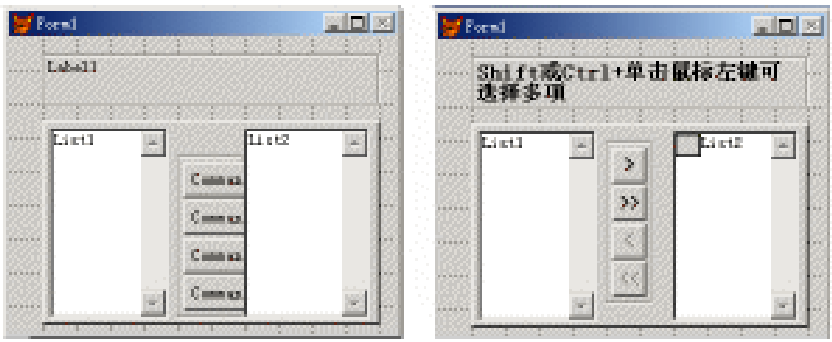
例 14　利用循环结构和列表框控件设计一个“选项移动”表单。所谓“选项移动”表单，是指由 2 个列表框控件和 4 个命令按钮所构成的界面。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器。增加一个容器控件 Container1、一个形状控件 Shape1、一个标签控件 Label1，在容器中增加两个列表框控件 List1、List2 和一个命令按钮组 CommandGroup1，并将命令按钮组的个数属性 ButtonCount 设置为 4，如图 7-84 (a) 所示。

② 设置对象属性 (见表 7-19)，完成之后如图 7-84 (b) 所示。



(a) (b)

图 7-84 应用界面设计

表 7-19 对象属性设置

对 象	属 性	属 性 值
Label1	Caption	Shift 或 Ctrl + 单击鼠标左键可选择多项
	WordWrap	. t.
Container1	SpecialEffect	0
List1	MultiSelect	. t.
List2	MultiSelect	. t.
	MoverBas	. t.
Container1. CommandGroup1	SpecialEffect	0
Command1	Caption	>
Command2	Caption	> >
Command3	Caption	<
	Enabled	. f.
Command4	Caption	< <
	Enabled	. f.

③ 编写程序代码。

编写容器 Container1 的 Init 事件代码：

```
this. list1. additem ("one")
this. list1. additem ("two")
this. list1. additem ("three")
this. list1. additem ("four")
this. list1. additem ("five")
this. list1. additem ("six")
this. list1. additem ("seven")
this. list1. additem ("eight")
this. list1. additem ("nine")
this. list1. additem ("ten")
```

编写容器控件中的命令按钮组 CommandGroup1 的 Click 事件代码：

```
do case
  case this. value = 1
    i = 0
    do while i < = this. parent. list1. listcount
      if this. parent. list1. selected (i)
        this. parent. list2. additem (this. parent. list1. list (i))
        this. parent. list1. removeitem (i)
      else
        i = i + 1
      endif
    enddo
  case this. value = 2
    do while this. parent. list1. listcount > 0
      this. parent. list2. additem (this. parent. list1. list (1))
      this. parent. list1. removeitem (1)
    enddo
  case this. value = 3
    i = 0
    do while i < = this. parent. list2. listcount
      if this. parent. list2. selected (i)
        this. parent. list1. additem (this. parent. list2. list (i))
        this. parent. list2. removeitem (i)
```

```
      else
        i = i + 1
      endif
    enddo
  case this. value = 4
    do while this. parent. list2. listcount > 0
      this. parent. list1. additem (this. parent. list2. list (1))
      this. parent. list2. removeitem (1)
    enddo
endcase
if this. parent. list2. listcount > 0
  this. command3. enabled = . t.
  this. command4. enabled = . t.
else
  this. command3. enabled = . f.
  this. command4. enabled = . f.
endif
if this. parent. list1. listcount = 0
  this. command1. enabled = . f.
  this. command2. enabled = . f.
else
  this. command1. enabled = . t.
  this. command2. enabled = . t.
endif
thisform. refresh
```

图 7-85 运行界面

④ 运行表单，运行界面如图 7-85 所示。

7.2.14 组合框

组合框与列表框类似，也是用于提供一组条目供用户选择，列表框具有的属性、方法，组合框也同样具有（除 MultiSelect 外），并且具有相类似的含义和使用方法。组合框与列表框的主要区别在于：

- 对于组合框来说，通常只有一个条目是可见的，用户可以单击组合框上的下箭头按钮打开条目列表，以便从中选择。因此，组合框能节省表单中的显示空间。
- 组合框不提供多重选择的功能，无 MultiSelect 属性。

✧ 组合框有 2 种形式：下拉组合框和下拉列表框。可通过设置 Style 属性选择所需要的组合框形式（见表 7-20）。

表 7-20　Style 属性的设置值与组合框的类型

属性值	说　　明
0	下拉组合框。用户既可从列表中选择，也可在编辑区内输入，在编辑区内输入的内容可以从 Text 属性中获得。
2	下拉列表框。用户只能从列表中选择。

在下拉列表中，只能看到一个条目，用户可以单击向下按钮来显示滚动的下拉列表框。

常用属性如下：

InputMask 属性：该属性用于下拉组合框，指定允许键入的数值类型。

Style 属性：该属性指定组合框是下拉组合框还是下拉列表框。

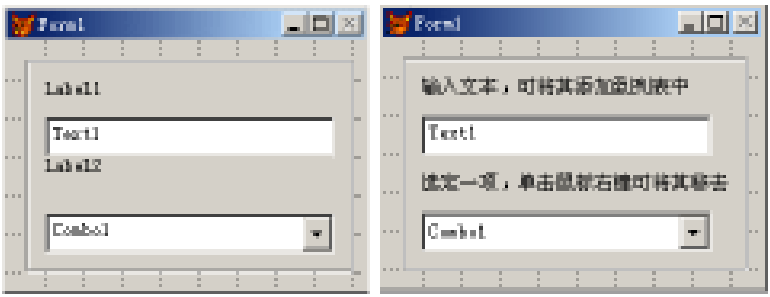
例 15　在文本框中输入数据，按回车键将其添加到列表框中，在列表框中选定项目，按回车键后可移去选定项。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器。增加一个形状控件 Shape1、一个文本框控件 Text1、2 个标签控件 Label1、Label2，一个组合框 Combo1，如图7-86 (a) 所示。

② 设置对象属性（见表 7-21），完成之后如图 7-86 (b) 所示。



(a)　　(b)

图 7-86　应用界面设计

表 7-21 对象属性设置

对 象	属 性	属性值
Label1	Caption	输入文本,可将其添加到列表中
Label2	Caption	选定一项,单击鼠标右键可将其移去
Text1	Text	(空)
Combo1	Style	2-下拉列表框

③ 编写应用程序代码。

编写表单 Form1 的 Activate 事件代码:

```
public a
a = 1
this. text1. setfocus
```

编写 Text1 的 KeyPress 事件代码:

```
LPARAMETERS nKeyCode, nShiftAltCtrl
if nkeycode = 13
  if ! empty (this. value)
    thisform. combo1. additem (this. value)
    thisform. combo1. displayvalue = this. value
  endif
  this. selstart = 0
  this. sellength = len (rtrim (this. text))
  a = 0
endif
```

编写 Text1 的 Valid 事件代码:

```
if a = 1
  return . t.
else
  a = 1
  return 0
endif
```

编写 Combo1 的 RightClick 事件代码:

```
if this. listindex > 0
  thisform. text1. value = this. list (this. listindex)
  this. removeitem (this. listindex)
```

```
  this. value = 1
endif
```

④ 运行表单，运行界面如图 7-87 所示。

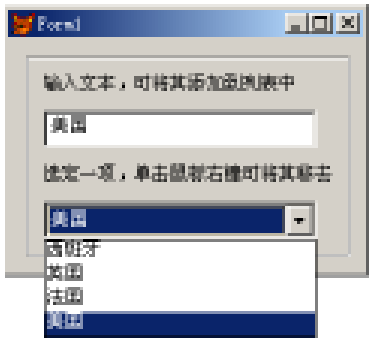


图 7-87　运行界面

7. 2. 15　表格控件 (Grid)

表格控件是一种容器对象，其外形与 Browse 窗口相似，按行和列的形式显示数据。一个表格对象由若干列对象 (column) 组成，每个列对象包含一个标头对象 (Header) 和若干控件。在此，表格、列、标头和控件都有自己的属性、事件和方法。

1. 表格设计基本操作

一旦指定了表格的列的具体数目 (表格的 ColumnCount 属性值不是 - 1)，就可以有两种方法来调整表格的行高和列宽：一是通过设置表格的 Header-Height 和 RowHeight 属性调整行高，通过设置列对象的 Width 属性调整列宽；二是让表格处于编辑状态，然后通过鼠标拖动操作可视地调整表格的行高和列宽。

若需切换到表格编辑状态，可选择表格快捷菜单中的“编辑”命令，或在属性窗口中的对象框中选择表格的 1 列。

在表格编辑状态下，表格的周围有一个粗框。若需切换出表格设计方式，可以选择表单或其他控件。

在表格编辑状态下，可视地调整列宽的方法是：

- 将鼠标指针置于两表格列的标头之间，此时鼠标指针变为水平双箭头的形状。

- 拖动鼠标，调整列至所需的宽度。

在表格编辑状态下，可视地调整行高的方法是：

- 将鼠标指针置于表格左侧的第一个按钮和第二个按钮之间，此时，鼠标指针变为垂直双箭头的形状。
- 拖动鼠标，调整行至所需的高度。

表格设计也可调用表格生成器来进行。通过表格生成器能交互地快速设置表格的有关属性，创建所需的表格。使用表格生成器生成表格的步骤是：先通过“表单控件”工具栏在表单上设置一个表格，然后用鼠标右键单击表格并在弹出的快捷菜单中选择“生成器”命令，打开“表格生成器”对话框，如图 7-88 所示，再在对话框内设置有关的选项参数。当设置完成之后按“确定”键关闭对话框返回时，系统会根据指定的选项参数设置列表框的属性。

图 7-88　在数据环境窗口中添加表

表格对话框包含 4 个选项卡，其作用如下：

- “表格项”选项卡：指明需在表格内显示的字段。
- “样式选项卡”：指定表格的样式（如标准型、专业型、账务型等）。
- “布局选项卡”：指明各列的列宽和控件类型、调整各列列宽。
- “关系选项卡”：设置一个一对多的关系，指明父表中的关键字段与子表中的相关索引。

2. 常用的表格属性

(1) RecordSourceType 属性与 RecordSource 属性

RecordSourceType 属性指明表格数据源的类型，RecordSource 属性指定表格数据源。

RecordSourceType 属性的取值范围及含义说明见表 7-22。

表 7-22　RecordSourceType 属性的设置值

属性值	说　明
0	表。数据来源于由 RecordSource 属性指定的表，该表能被自动打开
1	(默认值) 别名。数据来源于已打开的表，由 RecordSource 属性指定该表的别名
2	提示。运行时，由用户根据提示选择表格数据源
3	查询 (.qpr)。数据来源于查询，由 RecordSource 属性指定一个查询文件 (.qpr 文件)
4	SQL 语句。数据来源于 SQL 语句，由 RecordSource 属性指定一条 SQL 语句

设置了表格的 RecordSource 属性之后，就可通过 ControlSource 属性为表格中的一列指定所需显示的内容，若未指定，则该列将显示表格数据源中下一个还未显示的字段。这两个属性在设计时可用，在运行时可读写，但都仅适用于表格。

(2) ColumnCount 属性

该属性指定表格的列数，即一个表格对象所包含的列对象的数目。该属性的默认值为 - 1，此时表格将创建足够多的列来显示数据源中的所有字段。该属性在设计时可用，在运行时可读写。

(3) LinkMaster 属性

该属性用于指定表格控件中所需显示的子表的父表名称。使用该属性在父表和表格中显示的子表 (由 RecordSource 属性指定) 之间建立一对多的关联关系。需在两个表之间建立这种一对多关系，除了需设置该属性之外，还应使用 ChildOrder 和 RelationalExpr 两个属性。该属性在设计时可用，在运行时可读写，但是仅适于表格。

(4) ChildOrder 属性

该属性用于指定为建立一对多的关联关系，子表所需用到的索引。该属性类似于 Set Order 命令。该属性在设计时可用，在运行时只读。

(5) RelationalExpr 属性

该属性确定基于主表 (由 LinkMaster 属性指定) 字段的关联表达式。当主表中的记录指针移至新位置时，系统首先会计算出关联表达式的值，然后再从子表中找出在索引表达式 (当前索引可由 ChildOrder 属性指定) 上的取值与该值相匹配的所有记录，并将它们显示于表格中。该属性在设计时可用，在运行时可读写。

3. 常用的列属性

(1) ControlSource 属性

该属性指定需在列中显示的数据源，常见的是表中的一个字段。若未设

置该属性,列中将显示表格数据源(由 RecordSource 属性指定)中下一个还未显示的字段。

(2) CurrentCortrol 属性

该属性指定列对象中的一个控件,该控件用以显示和接收列中活动单元格的数据。列中非活动单元格的数据将在缺省的 TextBox 中显示。

缺省情况下,表格中的一个具体列对象包含一个标头对象(名称为 Header1)和一个文本框对象(名称为 Text1),而 CurrentControl 属性的默认值即是文本框 Text1。用户可以根据需要向列对象中添加所需的控件,并将 CurrentCortrol 属性设置为其中的某个控件(如可以用复选框来显示和接收逻辑型字段的数据)。

可以在表单设计器环境下,交互式地向表格列中添加控件,方法如下:

- 先从属性窗口的对象框中选择表格中需添加的列(也可以是其他表格元素)。此时表格四周出现粗框,表格处于编辑状态。
- 再从"表单控件"工具栏中选择所需控件,然后再用鼠标单击表格中需要添加该控件的列。

新添加的控件可在属性窗口的对象框中看到。若添加的控件是复选框,则通常将复选框的 Caption 属性设置为空串。若需将复选框指定为列的 CurrentControl 属性值,则一般是将列的 Sparse 属性设置为. F. 。

也可从表格列中移去控件,方法是:

- 从属性窗口的对象框中选择不需要的控件。
- 单击"表单设计器"窗口标题栏,激活表单设计器,此时,若属性窗口可见,控件的名称仍应显示在对话框中。
- 按 Delete 键。

该属性在设计时可用,在运行时可读写,仅适用于列。

(3) Sparse 属性

该属性用于确定 CurrentControl 属性是影响列中的所有单元格还是只影响活动单元格,若属性值为. T. (默认值),则只有列中的活动单元格使用 CurrentControl 属性指定的控件显示和接收数据,其他单元格的数据用缺省的 TextBox 显示;若属性值为. F. ,则列中所有单元格都使用 CurrentControl 属性指定的控件显示数据,活动单元格可接收数据。该属性在设计时可用,在运行时可读写,仅适用于列。

4. 常用的标头(Header)属性

(1) Caption 属性

该属性指定标头对象的标题文本,显示于列的顶部。

(2) Alignment 属性

该属性指定标题文本在对象中显示的对齐方式。对标头对象和列对象，其设置值见表 7-23。在默认方式 (属性值为 3) 下，对齐方式基于控件数据源的数据类型：数值型数据右对齐，其他类型数据左对齐。

表 7-23　Alignment 属性的设置

属性值	说　明	属性值	说　明
0	居中靠左	1	居中靠右
2	居中	3	(默认值) 自动
4	左上对齐	5	右上对齐
6	中上对齐	7	左下对齐
8	右下对齐	9	中下对齐

例 16　设计一个操作数据表的表单，使之具有按记录浏览、编辑的功能。

设计步骤如下：

① 创建数据环境。

选择“新建”表单，进入表单设计器。打开“数据环境设计器”窗口，在“数据环境”窗口中单击鼠标右键，在快捷菜单中选择“添加”，添加表单所需控制的数据表：学生表. dbf，如图 7-88 所示。

② 建立应用程序用户界面。

增加一个标签控件 Label1、一个命令按钮组 CommandGroup1、一个表格控件 Grid1，将表文件：学生表. dbf 中的字段“学号”、“姓名”、“班级”、“性别”、“照片”依次用鼠标拖拉至表单中，如图 7-89 所示。其中 CommandGroup1 的 ButtonCount 属性设置为 4。

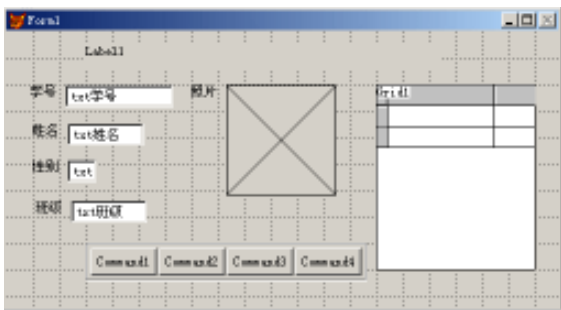


图 7-89　应用界面设计 (1)

表格控件 Grid1 的设计如下：

✧ 用鼠标右键单击表单控件 Grid1，在弹出的快捷菜单中选择“生成器”，打开“表格生成器”。用鼠标单击“数据库和表”右边的按钮 ...，如图 7-90 (a) 所示，选择表：学生表. dbf。

✧ 选择“可用字段”中的“学号”、“姓名”、“性别”、“班级”、“照片”，按添加按钮 ›，将其添加到“选定字段”列表中，如图 7-90 (b) 所示。

(a) (b)

图 7-90 “表格生成器”对话框

✧ 在“布局”页中，用鼠标指向标题行的分隔线可调整列标题的宽度。

✧ 按“确定”退出表格生成器。

③ 设置对象属性 (见表 7-24)，完成之后如图 7-91 所示。

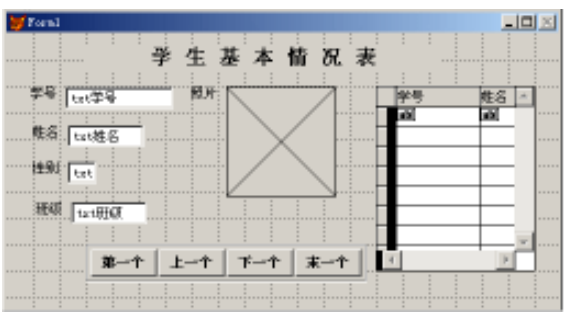


图 7-91 应用界面设计 (2)

表 7-24　对象属性设置

对　象	属　性	属　性　值
Label1	Caption	学生基本情况表
Command1	Caption	第一个
Command2	Caption	上一个
Command3	Caption	下一个
Command4	Caption	末一个

④ 编写程序代码。

编写自定义方法程序 butt ( ) —— 用于控制 4 个按钮是否可用：

```
lparameters l
this. commandgroup1. buttons (1) . enabled = l
this. commandgroup1. buttons (2) . enabled = l
this. commandgroup1. buttons (3) . enabled = not l
this. commandgroup1. buttons (4) . enabled = not l
```

编写命令按钮组 CommandGroup1 的 Click 事件代码：

```
n = this. value
do case
  case n = 1
    go top
    thisform. butt (. f.)
  case n = 2
    skip  - 1
    if bof ( )
      go top
      thisform. butt (. f.)
    endif
    this. buttons (3) . enabled = . t.
    this. buttons (4) . enabled = . t.
  case n = 3
    skip
    if eof ( )
      go bottom
      thisform. butt (. t.)
    endif
    this. buttons (1) . enabled = . t.
```

```
    this. buttons (2) . enabled = . t.
  case n = 4
    go bottom
    thisform. butt (. t.)
endcase
thisform. refresh
thisform. grid1. setfocus
```

编写表格 Grid1 **的** AfterRowColChange **事件代码：**

```
LPARAMETERS nColIndex
thisform. refresh
```

⑤ **运行表单，运行界面如图** 7-92 **所示。**

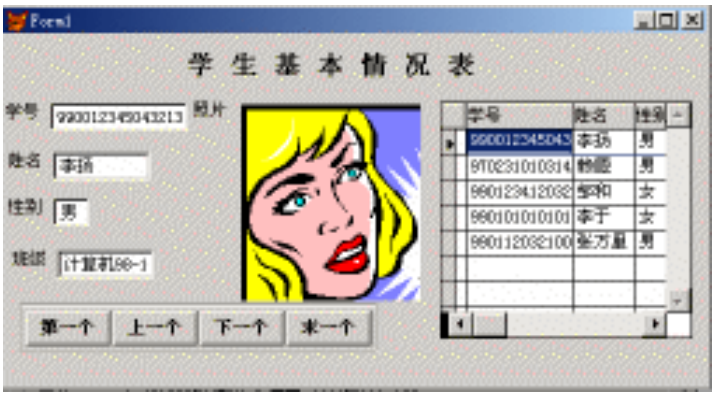


图 7-92　**运行界面**

7. 2. 16　页框控件 (PageFrame)

页框控件是包含页面 (page) 的容器对象，而页面本身也是一种容器，其中可以包含所需的控件。用页框、页面和相应的控件就可以构建选项卡对话框。这种对话框包含若干选项卡，其中的选项卡就对应着这里所说的页面。

页框定义了页面的总体特性，包括大小、位置、边界类型以及哪页是活动的等等。页框中的页面相对于页框的左上角定位，并随着页框在表单中移动而移动。

在表单设计器环境下，向表单添加页框的方法与添加其他控件的方法相同。默认情况下，添加的页框包含两个页面，它们的标签文本分别是 Page1 **和**

Page2 (与它们的对象名称相同)。用户可以通过设置页框的 PageCount 属性重新指定页面数目,通过设置页面的 Caption 属性重新指定页面的标签文本。

若需向页面中添加控件,可按以下步骤进行:

- 用鼠标右键单击页框,在弹出的快捷菜单中选择"编辑"命令,然后再单击相应页面的标签,使该页面成为活动的。也可从属性窗口的对象框中直接选择相应的页面,此时,页框四周出现粗框。
- 在"表单控件"工具栏中选择需要的控件,并在页面中调整其大小。

注:第一步的作用是将页框切换到编辑状态,并选择相应的页面。在添加控件前,若未将页框切换到编辑状态,则控件将会被添加到表单而不是添加到页框的当前页面中,即使看上去好像是在页面中。

常用属性:

PageCount 属性:该属性用于指明一个页框对象所包含的页 (Page) 对象的数量。PageCount 属性的最小值是 0,最大值是 99。当用户递减设置 PageCount 属性时,超出页框页数新设置的所有页及其中的对象将丢失。该属性在设计和运行时可用,仅适用于页框。

Pages 属性:该属性是一个数组,用于存取页框中的某个页对象。例如需将页框 myPageFrame 中的第 2 页的 Caption 属性设置为"列表项",可用代码:

```
thisform. myPageFrame. Page (2) . Caption = 列表项
```

页面的 Caption 属性值显示于页面标签上。当然,也可用页面的名称来引用某个具体的页面对象。该属性仅在设计时可用,适用于页框。

Tabs 属性:该属性指定页框中是否显示页面标签栏。若属性值为. T. (默认值),则页框中包含页面标签栏;若属性值为. F.,则页框中不显示页面标签栏。该属性在设计和运行时可用,仅适用于页框。

TabStretch 属性:若页面标题 (标签) 太长,标签栏无法在指定的宽度的页框内显示出来,可以通过 TabStretch 属性指明其行为方式,该属性的设置情况见表 7-24。该属性仅在 tabs 属性值为. T. 时有效,在设计和运行时可用,仅适用于页框。

表 7-24　TabStretch 属性的设置

属性值	说　　明
0	多重行。标签栏可根据需要分几行显示,所有的标签文本都被显示出来
1	单行 (默认值)。标签栏在一行中显示,太长的标签文本被截断

ActivePage 属性:返回页框中活动页的页号,或使页框中的指定页成

为活动页。该属性在设计时可用，在运行时可读写，仅适用于页框。

例 17　在例 16 中使用页面技术。

设计步骤如下：

① 建立应用程序用户界面。

选择“新建”表单，进入表单设计器。增加一个标签控件 Label1、一个命令按钮组控件 CommandGroup1（CommandGroup1 的 ButtonCount 属性设置为 4，用鼠标单击此命令按钮组，选择“生成器”，在“布局”选项卡中选择“水平”）、增加一个页框控件 PageFrame1（适当调整其大小与位置），如图 7-93 所示。

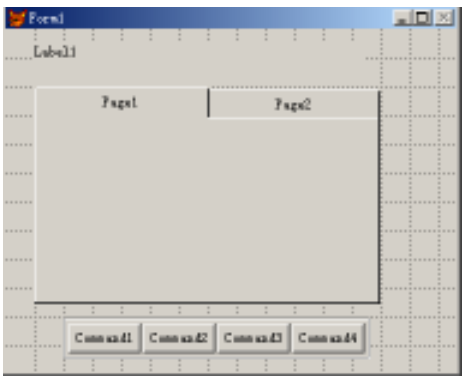


图 7-93　应用界面设计（1）

② 设置对象属性，设置完成之后如图 7-94 所示。

(a)

(b)

图 7-94　应用界面设计（2）

- Label1 的 Caption 属性设置为：学生基本情况表。
- CommandGroup1 中的 Command1、Command2、Command3、Command4 的 Caption 属性分别设置为：第一个、上一个、下一个、末一个。
- 对于页框 PageFrame1：

编辑第一页。单击鼠标右键，在快捷菜单中选择“编辑”，此时页框周围出现浅色边界，开始编辑第一页。在表单空白处单击鼠标右键，选择“数据环境”，然后在“数据环境”窗口中选择表文件：学生表. dbf。并将字段“学号”、“姓名”、“性别”、“班级”、“家庭住址”、“照片”逐一拖进 Page1 中，并调整位置。将 Page1 的 Caption 属性设置为：学生基本情况表。

编辑第二页。单击标题 Page2，进入第 2 页。单击“数据环境”按钮，打开“数据环境”窗口。将鼠标指针指向表文件名：学生表，然后按下鼠标左键，将其拖进至 Page2 中。此时 Page2 中自动出现一个包含了“学生表. dbf”中所有字段的表格控件“grd 学生表”，关闭：“数据环境”窗口，并调整表格的大小和位置，然后开始修改“grd 学生表”。用鼠标右键单击“grd 学生表”，在弹出的菜单中选择“编辑”，此时“grd 学生表”的周围出现浅色边界。在属性窗口中修改“grd 学生表”的列对象属性 ColumnCount 为 5（原值为 6，是“学生表. dbf”中的字段数），可以将“照片”字段从表格中删除掉。将鼠标指针指向表格的第一行，可以拖动鼠标修改列的宽度，尽可能地将所需要显示的字段长度都调整到可以看到，可将“grd 学生表”的 ScrollBars 属性在属性窗口中设置为 3。再将 Page2 的 Caption 属性设置为“浏览”。如图 7-94 所示。

③ 编写应用程序代码。

编写 CommandGroup1 的 Click 事件代码：

```
n = this. value
do case
  case n = 1
    go top
    thisform. butt (. f. )
  case n = 2
    skip  - 1
    if bof ( )
      go top
      thisform. butt (. f. )
    endif
    this. buttons (3) . enabled = . t.
    this. buttons (4) . enabled = . t.
```

```
  case n = 3
    skip
    if eof()
      go bottom
      thisform. butt(. t.)
    endif
    this. buttons(1). enabled = . t.
    this. buttons(2). enabled = . t.
  case n = 4
    go bottom
    thisform. butt(. t.)
endcase
if thisform. pageframe1. activepage = 2
  thisform. pageframe1. page2. grd 学生表. setfocus
endif
thisform. refresh
```

编写 PageFrame1 **中的第一页** Page1 **的** Activate **事件代码：**

```
this. txt 学号. setfocus
```

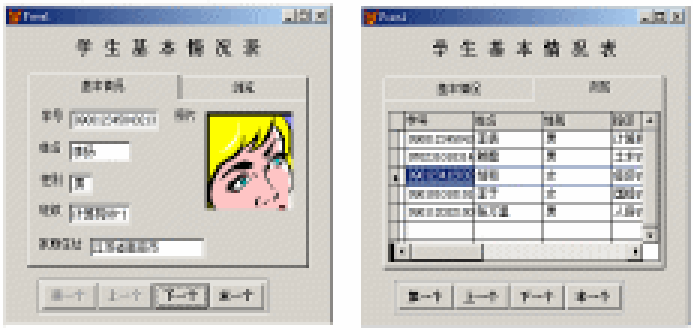
编写 PageFrame1 **中的第二页** Page2 **的** Activate **事件代码：**

```
this. grd 学生表. setfocus
```

编写第二页 Page2 **中表格“**grd **学生表”的** AfterRowColChange **事件代码：**

```
LPARAMETERS nColIndex
this. parent. parent. page1. refresh
```

④ **运行表单，运行界面如图** 7-95 **所示。**



(a) (b)

图 7-95 **运行界面**

第8章 数据库和表的高级应用

在前面第四章的学习中我们学习了数据库和表的基本应用。通过前面的学习我们了解到每个数据库都带有一个数据字典，数据字典使得对数据库的设计和修改更加灵活。使用数据字典，可以设置字段级和记录级的有效性检查，保证主关键字字段内容的惟一性，如果不用数据字典，这些功能就都必须由您自己编程实现。Visual FoxPro 数据字典可创建和指定：

主关键字和候选索引关键字。

数据库表间的永久关系。

表和字段的长名称。

各字段的标题，日后它们将作为标头显示在“浏览”窗口或表格列中。

字段的默认值。

表单中使用的默认控件类。

字段的输入掩码和显示格式。

字段级规则和记录级规则。

触发器。

存储过程。

与远程数据源的连接。

本地视图和远程视图。

对每个字段、表和数据库的注释。

一些数据词典的功能（例如长字段名、主关键字、候选关键字、默认值、字段级和记录级规则及触发器）存储在 .dbc 文件中，在创建表或视图的过程中创建。下面我们来逐一介绍。

§8.1 数据库的高级应用

8.1.1 使用其他数据库中的表

若要访问其他数据库中的表,我们可以使用 USE 命令和“!”符号访问该表。

使用“!”符号可以引用一个不在当前数据库中的表。例如,如果想浏览 testdata 数据库中的 orditem 表,可键入:

```
USE testdata!orditems
BROWSE
```

对于这个示例,使用 USE 命令时,自动打开 testdata 数据库,但 Visual FoxPro 并不把 testdata 设成当前数据库。当关闭表时,除非在表关闭之前已显示地打开该数据库,否则自动打开的数据库会自动关闭。

在操作中,若要访问其他数据库中的表,我们还可以在引用该表的数据库中创建视图。其使用方法将在后面章节中予以介绍。

8.1.2 创建存储过程

您可以为数据库中的表创建存储过程。存储过程是存储在 .dbc 文件中的 Visual FoxPro 代码,是专门操作数据库中数据的代码过程。存储过程可以提高数据库的性能,因为在打开一个数据库时,它们便加载到了内存中。

若要创建、修改或移去存储过程,可以在“项目管理器”中,选择并展开一个数据库,选定“存储过程”,然后选择“新建”、“修改”或“移去”按钮。

或者是在“数据库设计器”中,从“数据库”菜单中选择“编辑存储过程”按钮,也可在“命令”窗口中,使用 MODIFY PROCEDURE 命令。

这几种方法都将打开 Visual FoxPro 文本编辑器,帮助您创建或修改当前数据库的存储过程。

使用存储过程主要是为了创建用户自定义函数,字段级规则和记录级规则将引用这些函数。当把一个用户自定义函数作为存储过程保存在数据库中时,函数的代码保存在 .dbc 文件中,并且在移动数据库时,会自动随数据库移动。使用存储过程能使应用程序更容易管理,因为可以不必在数据库文件之外管理用户自定义函数。关于过程和函数请参阅程序设计部分。

8.1.3　引用多个数据库

为符合多用户环境的数据组织需要,您可以同时使用多个数据库。同时使用多个数据库具有以下优点:

⑴ 控制用户访问整个系统的子系列表。

⑵ 组织数据以有效地符合各组别使用系统时的信息需要。

⑶ 运行时允许排他地使用子系列表以创建本地视图或远程视图。

例如,您有一个包含销售信息的销售数据库,主要用于销售部门处理顾客方面的信息。另一个包含货物信息的数据库,主要用于采购部门处理供应商方面的信息。有时不同部门对信息的需求会重叠。这些数据库可同时打开,随意访问,但它们包含着不同类的信息。

多个数据库可增加系统灵活性。

可通过同时打开多个数据库,或引用关闭的数据库中的文件,使用多数据库。一旦打开多个数据库,可设置当前数据库,并选择其中的表。

1. 打开多个数据库

打开一个数据库后,表和表之间的关系就由存储在该数据库中的信息来控制。您可以同时打开多个数据库。例如,在运行多个应用程序时,可以使用多个打开的数据库,每个应用程序都以不同的数据库为基础。也可能您想打开多个数据库,从而能使用应用程序数据库之外的另一数据库中的存储信息,例如自定义控件。

若要打开多个数据库,在“项目管理器”中,选定一个数据库,然后选择“修改”按钮或“打开”按钮,或者使用 OPEN DATABASE 命令。

打开新的数据库并不关闭其他已经打开的数据库,这些已打开的数据库仍然保持打开状态,而新打开的数据库成为当前数据库。

2. 设置当前数据库

当打开多个数据库时,Visual FoxPro 将最后打开的数据库设置为当前数据库。所创建或添加到该数据库中的任何表或其他对象,均默认为当前数据库的一部分,处理打开数据库的命令和函数 (例如 ADD TABLE 命令和 DBC( ) 函数),也是对当前数据库进行操作。

可通过交互方式或使用 SET DATABASE 命令选择另外一个数据库作为当前数据库。

若要设置当前数据库,常用方法是在“常用工具栏”中,从“数据库”框中选择一个数据库。

或者使用 SET DATABASE 命令。

例如,下面的代码打开三个数据库,设置第一个数据库为当前数据库,然后使用 DBC ( ) 函数显示当前数据库的名称:

```
OPEN DATABASE testdata
OPEN DATABASE tastrade
OPEN DATABASE sample
SET DATABASE TO testdata
?DBC ( )
```

提示:当执行的查询或表单需要打开一个或多个数据库时,Visual FoxPro 可以自动打开这些数据库。为了保证您所处理的数据库是正确的数据库,最好在执行任何对当前数据库操作的命令前,明确设置当前数据库。

选择当前数据库中的表可以使用 USE 命令,在当前数据库的一系列表中选择要用到的表。

若要从当前数据库选择表请使用带“ ?”字符的 USE 命令。

显示“使用”对话框后,选定并打开一个表。

例如,下面的代码打开了 sales 数据库,并提示您在数据库的一系列表中选定一个表:

```
OPEN DATABASE SALES
USE ?
```

如果您想选定一个与当前数据库不关联的表,可选择“使用”对话框中的“其他”按钮。

3. 关闭数据库

若要关闭数据库,从“项目管理器”中,选定要关闭的数据库并选择“关闭”按钮。

或者使用 CLOSE DATABASE 命令关闭一个已打开的数据库。

例如,下面的代码关闭了 testdata 数据库:

```
SET DATABASE TO testdata
CLOSE DATABASE
```

这两种方法都自动关闭数据库,也可以使用 CLOSE 命令的 ALL 子句关闭数据库及所有其他已打开的对象。

如果用下面的方法打开一个数据库,那么在“命令”窗口执行 CLOSE DATABASE 命令时,不能关闭该数据库:

用“项目管理器”打开,并展开分层结构查看一个数据库的内容。

用正在自己的数据工作期内运行的表单打开。

在这两种情况下,数据库将一直保持打开状态,直到“项目管理器”关闭该数据库,或者使用该数据库的表单关闭。

4. 作用域

Visual FoxPro 把当前数据库作为命名对象 (例如表) 的主作用域。当打开一个数据库时,Visual FoxPro 首先在已打开的数据库中搜索所需的任何对象 (例如表、视图、连接等)。只有在当前数据库中没有找到所需对象时,Visual FoxPro 才在默认的搜索路径上查找。

例如,如果 customer 表和 sales 数据库关联,那么当您执行下面的命令时,Visual FoxPro 总会在此数据库中找到 customer 表:

```
OPEN DATABASE SALES
ADD TABLE F:\SOURCE\CUSTOMER.DBF
USE CUSTOMER
```

如果执行下面的命令,Visual FoxPro 首先在当前数据库中查找 products 表:

```
USE PRODUCTS
```

如果 products 表不在当前数据库中,Visual FoxPro 会使用默认的搜索路径在数据库外查找。

注意:如果您希望在数据库的内部和外部都能访问一个表 (例如对于表的位置会改变的情况),可以为该表指定完整路径。但是只使用表名,速度会更快,因为 Visual FoxPro 对数据库表名的访问速度比完整路径快。

5. 数据库错误处理

数据库错误,也称“引擎错误”,当记录级事件代码发生运行时刻错误时。例如当用户将 null 值存入一个不允许为 null 值的字段时,产生数据库错误。

当对数据库非法操作时,位于底层的数据库引擎会检测到错误并提交一条出错信息。出错信息的具体内容与检测到该错误的数据库管理系统有关,例如,远程数据服务器 (例如 Microsoft SQL Server) 产生的错误信息可能不同于本地的 Visual FoxPro 所产生的错误信息。

另外,引擎级的错误信息通常都很笼统,因为数据引擎并不知道记录更新的具体环境,所以 Visual FoxPro 应用程序的最终用户不用太关心引擎级错误信息。

若要产生对应用程序而言更具有针对性的错误信息,可用 CREATE TRIGGER 命令创建触发器。触发器每当记录更新 (删除、插入或更新) 时便被触发。您在对触发事件进行编程时,可以捕获针对当前应用程序的错误,并报

告出错信息。

若用触发器处理数据库错误，需打开缓冲器。这样，当更新记录时，触发器被触发，但记录不立即传送到底层的数据库引擎，因而可避免同时产生两个错误信息的可能：一个来自触发器，另一个来自底层的数据库引擎。

§8.2 设置表属性

一个 Visual FoxPro 表或 .dbf 文件，能够存在以下两种状态之一：数据库表（与数据库相关联的表），与数据库无关联的自由表。相比之下，数据库表的优点要多一些。当一个表是数据库的一部分时，它就可以具有：

① 长表名和表中的长字段名。
② 表中字段的标题和注释。
③ 默认值、输入掩码和表中字段格式化。
④ 表字段的默认控件类。
⑤ 字段级规则和记录级规则。
⑥ 支持参照完整性的主关键字索引和表间关系。
⑦ INSERT、UPDATE 或 DELETE 事件的触发器。

将表添加到数据库后，便可以立即获得许多在自由表中得不到的属性。这些属性被作为数据库的一部分保存起来，并且一直为表所拥有，直到表从这个数据库中移去为止。

通过设置数据库表的字段属性，您可以：

① 为字段设置标题。
② 为字段输入注释。
③ 为字段设置默认值。
④ 设置字段的输入掩码和显示格式。
⑤ 设置字段的控件类和库。
⑥ 设置有效性规则对输入字段的数据加以限制。

8.2.1 设置字段标题

我们在浏览表时，表的标题显示的是字段名，由于我们在定义字段的时候规定，字段名的最大宽度是十个字符，难以表达清楚该字段的数据属性。通常我们用了编程的方便，往往把字段名定义成英文代码，这更难以说明该字段的含义，通过在表中给字段建立标题，可以在“浏览”窗口中或表单上显示出字段的说明性标签。

若要给字段指定一个标题，通过项目管理器打开数据库gzk，出现如图 8-1 所示窗口，

图 8-1　数据库设计器

在“数据库设计器”中选定表 jbqk. dbf，然后在“数据库设计器”工具栏中选择“修改表”。选定需要指定标题的字段“编号”。在“标题”框中，键入为字段选定的标题，如图 8-2 所示。

图 8-2　表设计器

我们单击“确定”，屏幕又出现如图 8-3 所示对话框，我们选择“是”，字段的标题就定义好了。

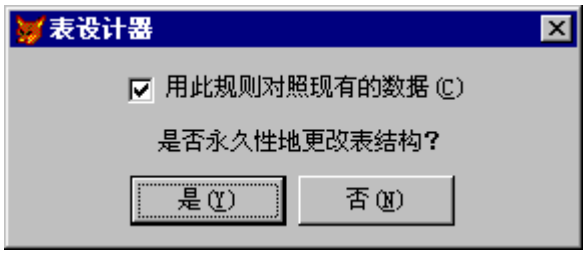


图 8-3 询问窗口

我们再次浏览该表时,就会发现表中的标题名发生了改变,其结果如图 8-4 所示。

登记顺序	姓名	性别	出生年月	部门代号	党员否	简历	照片
100001	赵一同	男	01/23/68	1	T	Memo	Gen
100002	王小小	女	08/12/75	4	F	Memo	Gen
100003	周玖月	女	09/19/60	1	T	Memo	Gen
100004	吴建国	男	07/09/50	2	T	Memo	Gen
100005	孙颖颖	女	11/23/70	5	F	Memo	Gen
100006	肖丽丽	女	06/01/65	2	F	Memo	Gen
100007	陈若无	男	02/12/66	1	T	Memo	Gen
100008	万維妙	女	06/24/81	3	T	Memo	Gen

图 8-4 加标题后的表内容

由于您可以用自己命名的标题取代原来的字段名,这为显示表单中的表提供了很大的灵活性。

8.2.2 为字段输入注释

在建立好表的结构以后,您可能还想输入一些注释,来提醒自己或他人表中的字段所代表的意思。在“表设计器”中的“字段注释”框内输入信息,即可对每一个字段进行注释。

若要为一个字段作注释,其操作步骤为:

在“表设计器”中,选定需要设置注释的字段。

在“字段注释”框中键入注释内容,如图 8-5 所示。

选择“确定”。

图 8-5　增加注释

8.2.3　控制字段数据输入

为了使表中的数据输入更容易、简单一些，方法是提供字段的默认值，定义输入到字段中值的有效性规则。

1. 设置默认字段值

若要在创建新记录时自动输入字段值，可以在“表设计器”中用字段属性为该字段设置默认值。例如，如果大部分职工的职称都是讲师，可把职称名称设为该字段的默认值。

若要设置字段的默认值，其操作步骤为：

① 在“数据库设计器”中选定表 jbqk. dbf。

② 从“数据库”菜单中选择“修改”。

③ 在“表设计器”中选定要赋予默认值的字段。

④ 在“默认值”框中键入要显示在所有新记录中的字段值（字符型字段应用引号括起来），如图 8-6 所示。

图 8-6　设置默认值

然后单击对话框的“确定”。默认值就设置好了。

2. 设置有效性规则和有效性说明

如果在定义表的结构时输入字段的有效性规则，那么可以控制输入该字段的数据类型，限制某字段可接受的输入项。

若要为字段设置有效性规则和有效性说明，其操作步骤为：

在“表设计器”中打开表 jbqk. dbf。

在“表设计器”中选定要建立规则的字段名“出生年月”。

在“规则”方框旁边选择对话按钮。出现表达式生成器。

在“表达式生成器”中设置有效性表达式，如图 8-7 所示，并选择“确定”。

图 8-7　表达式生成器

在“信息”框中，我们还可以键入用引号括起的错误信息，我们在这省略不写。

选择“确定”。

我们如果再向表中追加记录，如图 8-8 所示，把出生年月中的年输入为 98 年，按回车后，屏幕会出现如图 8-9 所示对话框，单击确定，回到编辑状态，重新输入。单击还原，取消已经定义的有效性规则。

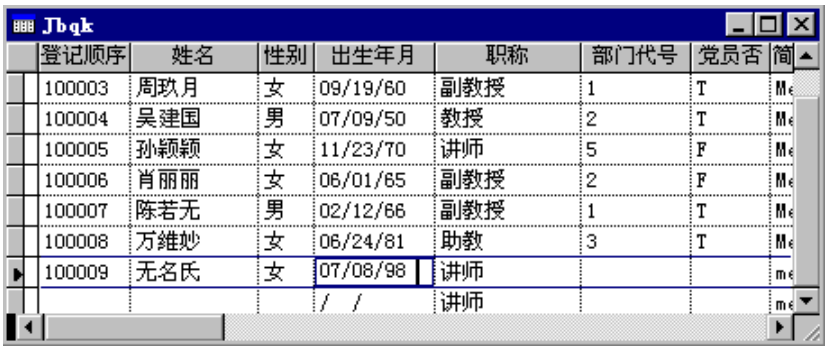


图 8-8　追加记录

图 8-9　提示信息

建立有效性规则时，必须创建一个有效的 Visual FoxPro 表达式，其中要考虑到这样一些问题：字段的长度、字段可能为空或者包含了已设置好的值等等。表达式也可以包含结果为真或假的函数。

例如，假设现在正使用 Visual Studio… hSamples hVfp98 hData 目录下的 Customer 表，并想确保新记录的 Customer ID 代码少于 6 个字符，则可在 Cust_ID 字段的“规则”框中，输入：

LEN (ALLTRIM (CUST_ID))　< 6

然后在“信息”框中输入下述错误提示：

"Customer ID must be less than 6 characters. Please reenter."

如果您以后输入的 Customer ID 太长,就会出现一个对话框,其中显示有效性说明。

当输入非法数值时,屏幕上会显示有效性说明。

8.2.4 控制记录的数据输入

不但可以给表中的字段赋予数据库的属性,而且可以为整个表或表中的记录赋予属性。在"表设计器"中,通过"表"选项卡可以访问这些属性。

1. 设置表的有效性规则

向表中输入记录时,若要比较两个以上的字段,或查看记录是否满足一定的条件时,可以为表设置有效性规则。

若要设置有效性规则,其操作步骤为:

选定表,然后选择"数据库"菜单中的"修改"。

在"表设计器"中选择"表"选项卡。

在"规则"框中,输入一个有效的 Visual FoxPro 表达式定义规则,或选择对话按钮来使用"表达式生成器"。

在"信息"框中输入提示信息。当有效性规则未被满足时,将会显示该信息。

选择"确定"。

在"表设计器"中选择"确定"。

例如,假如您正在使用 Visual Studio… hSamples hVfp98 hData 目录下的 Customer 表,并且不希望华盛顿州客户的定单超过 10 000 美元,则可以在"表"选项卡的"规则"框中键入下述表达式:

IIF(Customer. region = "WA" and Customer. maxordamt > 10000, .F., .T.)

"信息"框内的文字可以是:

"Orders from WA state customers cannot exceed $10,000"

按照有效性规则,某些输入将被拒绝。

2. 设置触发器

触发器是一个在输入、删除或更新表中的记录时被激活的表达式。通常,触发器一般需要输入一个程序或存储过程,在修改表时,它们被激活。

§8.3　建立表间的关系

relation（关系），是指表之间的一种链接，它允许您不仅能从当前选定表中访问数据，而且可以访问其他表中的数据。这种链接指的是连接条件。

通过链接不同表的索引，“数据库设计器”可以很方便地建立表之间的关系。因为这种在数据库中建立的关系被作为数据库的一部分保存了起来，所以称为永久关系。每当您在“查询设计器”或“视图设计器”中使用表，或者在创建表单时在“数据环境设计器”中使用表时，这些永久关系将作为表间的默认链接。

表之间创建关系之前，想要关联的表需要有一些公共的字段和索引。这样的字段称为主关键字字段和外部关键字字段。主关键字字段标识了表中的特定记录。外部关键字字段标识了存于数据库里其他表中的相关记录。还需要对主关键字字段做一个主索引，对外部关键字字段做普通索引。

若要决定哪些表需要这类字段，需要考虑数据与记录数目的关联方式。例如，一个客户可能有多个订单，因此，客户表应包含主记录，订单表包含相关记录。

若要在两个表之间提供公共字段，需要在带有关联记录的表中添加外部关键字字段。外部关键字字段必须以相同的数据类型与主关键字字段相匹配，而且一般用相同的名称，且以主关键字字段和外部关键字字段创建的索引必须带有相同的表达式。

8.3.1　创建关系

定义完关键字段和索引后，即可创建关系。如果表中还没有索引，需要在“表设计器”中打开表，并且向表中添加索引。有关添加表中索引的详细内容，请参阅前面所学的章节。

若要在表间建立关系，打开数据库文件 gzk，进入“数据库设计器”窗口。

确定父表和子表，并利用表设计器把父表和子表同名的字段“编号”定义为主索引字段，把子表中与父表同名字段定义为候选索引或主索引字段。

在“数据库设计器”窗口，激活父表中的主索引字段编号，按住鼠标左键拖动鼠标将一个表的索引拖到另一个表的匹配的索引上，松开鼠标左键，数据库中的两个表就有了一个“连线”，一对一关系就建好了。在这种关系中，主表中的每一个记录只与相关表中的一个记录相关联。

设置完关系之后，在数据库设计器中可看到一条连接了两表的线，如图

8-10 所示。

图 8-10 一对一关系

注意：只有在“数据库属性”对话框中的“关系”选项打开时，才能看到这些表示关系的连线。可以打开“数据库属性”对话框，方法是从“数据库设计器”的快捷菜单中选择“属性”。

关系线表示了两个表之间的关系。

在关联关系中，还存在一种 one - to - many relationship（一对多关系），表之间的这种关系，在这种关系中，主表中的每一个记录与相关表中的多个记录相关联（每一个主关键字值在相关表中可出现多次）。

其操作方法和建立一对一关系相似。

打开数据库文件，进入“数据库设计器”窗口。

确定父表和子表，并利用表设计器把父表和子表同名的字段定义为主索引字段，把子表中与父表同名字段定义为普通索引或候选索引。

在“数据库设计器”窗口，激活父表中的主索引字段编号，按住鼠标左键拖动鼠标将一个表的索引拖到另一个表的匹配的索引上，松开鼠标左键，数据库中的两个表就有了一个“连线”，一对多关系就建好了，在此不再举例说明。

8.3.2 编辑表间的关系

双击表间的关系线，出现如图 8-11 所示对话框，再选择“编辑关系”对话框中的适当设置。

图 8-11　“编辑关系”对话框

所建关系的类型是由子表中所用索引的类型决定的。例如，如果子表的索引是主索引或候选索引，则关系是一对一的；对于惟一索引和普通索引，将会是一个一对多的关系。

8.3.3　删除表间的关系

若要删除表间的永久关系，在“数据库设计器”中，单击两表间的关系线。关系线变粗，表明已选择了该关系。

按下 DELETE 键。

或者，在命令窗口，在 ALTER TABLE 命令中，使用 DROP FOREIGN KEY 子句。

例如，下面的命令删除了表 customer 和表 orders 之间的一个永久关系，这个关系基于 customer 表的主关键字 cust_id 和 orders 表的外部关键字 cust_id：

```
ALTER TABLE orders DROP FOREIGN KEY TAG cust
```

8.3.4　参照完整性

建立关系后，也可设置管理数据库关联记录的规则，这些规则控制参照完整性。例如，如果添加一个供应商记录，您可能想在产品表中自动添加关于供应商产品的信息。为了帮助设置规则，控制如何在关系表中插入、更新或删除记录，您可使用“参照完整性生成器”。

若要使用“参照完整性生成器”，其操作步骤如下：

在“数据库设计器”中建立两表之间的关系，或者双击关系线来编辑关系。

在“编辑关系”对话框中选择“参照完整性”按钮，出现如图 8-12 所示结

果。

图 8-12　参照完整性生成器

选择“确定”，然后选择“是”保存所做的修改，生成“参照完整性”代码，并退出参照完整性生成器。

如果表没有创建索引，则在使用“参照完整性生成器”之前，需要在“表设计器”中打开这个表，并创建索引。

建立参照完整性涉及生成一系列规则，以便在输入或删除记录时，能保持已定义的表间关系。

如果实施参照完整性规则，Visual FoxPro 可以确保：

当主表中没有关联记录时，记录不得添加到相关表中。

主表的值不能改变，若这个改变将导致相关表中出现孤立记录。

若某主表记录在相关表中有匹配记录，则该主表记录不能被删除。

您也可以编写自己的触发器和存储过程代码来实施参照完整性。Visual FoxPro 参照完整性生成器（RI）可以帮助您确定要实施的规则类型、要实施规则的表以及会导致 Visual FoxPro 检查参照完整性规则的系统事件。

RI 生成器是可处理级联删除和级联更新的一个工具，建议您用它建立参照完整性。

“参照完整性（RI）生成器”帮助设置触发器，用来控制如何在相关表中插入、更新或者删除记录，确保参照完整性。

“RI 生成器”在以下情况下显示：

在“数据库设计器”中双击两个表之间的关系线，并在“编辑关系”对话框

中选择“参照完整性”按钮。

从“数据库设计器”快捷菜单中选择“编辑参照完整性”选项。

选择“数据库”菜单中的“编辑参照完整性”选项。

参照完整性表格中各标题的含义如下：

父表：显示数据库关系中的表名，该数据库关系包含主索引或候选索引。

子表：显示数据库关系中的子表名。

更新：显示关系中参照完整性的更新规则。可能的取值为“级联”、“限制”或“忽略”。通过选择能显示其他选项的下拉列表的字段，可以更改此设置。

删除：显示关系中参照完整性的删除规则。可能值为“级联”“限制”或“忽略”。通过选择能显示其他选项的下拉列表的字段，可以更改此设置。

插入：显示关系中参照完整性的插入规则。可能的值为“限制”或“忽略”。通过选择能显示其他选项的下拉列表的字段，可以更改设置。

父标记：表格列显示父表中主关键字段或候选关键字段中索引的标识名。

子标记：表格列显示子表的索引标识名。

生成器选项卡的含义如下：

更新规则：指定修改父表中关键字 (key) 值时所用的规则。

删除规则：指定删除父表中的记录时所用的规则。

插入规则：指定在子表中插入新的记录或更新已存在的记录时所用的规则。

首先我们来看一看“更新规则”选项卡，“参照完整性生成器”中的“更新规则”选项卡指定修改父表中的关键字 (key) 值时所用的规则。选项卡选项包括下列三种情况：

级联：对父表中的主关键字段或候选关键字段的更改，会在相关的子表中反映出来。如果选择了该选项，不论何时更改父表中的某个字段，Microsoft Visual FoxPro 自动更改所有相关子表记录中的对应值。

限制：禁止更改父表中的主关键字段或候选关键字段中的值，这样在子表中就不会出现孤立的记录。

忽略：即使在子表中有相关的记录，仍允许更新父表中的记录。

根据以上定义，我们在更新规则中选择“级联”。

“参照完整性生成器”中的“删除规则”选项卡指定在删除父表中的记录时所用的规则。选项卡选项同样包括下列三种情况：

级联：指定在父表中进行的删除在相关的子表中反映出来。如果您为一个关系选择了“级联”，无论何时删除父表中的记录，相关子表中的记录自动删除。

限制：禁止更改父表中的记录，这些记录在子表中有相关的记录。如果

您为一个关系选择了“限制”,那么当在子表中有相关的记录时,则在父表中进行的删除记录的尝试就会产生一个错误。

忽略：允许删除父表中的记录,即使子表中有其相关的记录。

根据以上定义,我们在删除规则中选择“级联”。

“参照完整性生成器”中的“插入规则”选项卡指定在插入新的记录,或者在子表中更新已存在的记录时所用的规则。选项卡选项包括两种情况：

限制：禁止在子表中添加记录,这些记录在父表中没有相匹配的记录。如果您为一个关系选择了“限制”,那么当父表中没有相匹配的记录时,则在子表中添加记录的尝试就会产生一个错误。

忽略：允许向子表中插入记录,而不管父表中是否有匹配的记录。

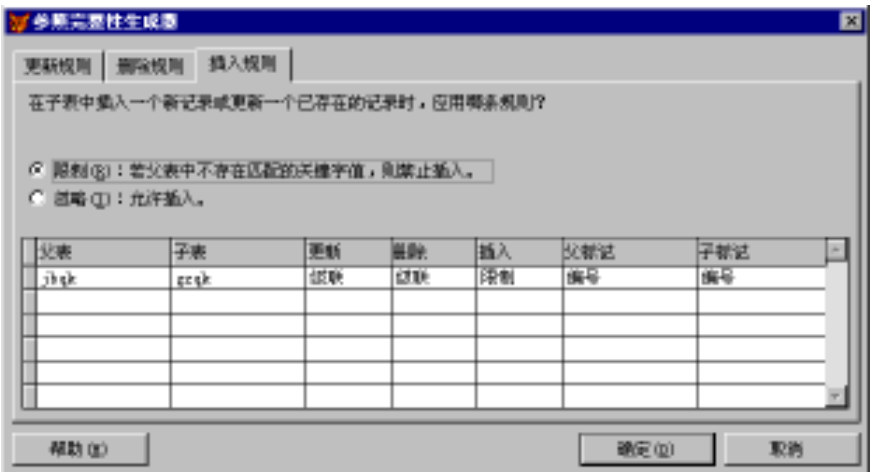


图 8-13　设置规则

根据以上说明我们在插入规则中选择限制。完成以上操作,人们单击确定按钮,出现如图 8-14 所示对话框,我们单击“是”键,至此,参照完整性就设计好了。

图 8-14　提示框

§8.4 使用多个表

若要使用多个表,就要使用多个工作区。一个工作区是一个编号区域,它标识一个已打开的表。可以在 32 767 个工作区中打开和操作 Visual FoxPro 表。

在应用程序中工作区通常通过使用该工作区的表的别名来标识。表别名是一个名称,它可以引用在工作区中打开的表。

除了在"数据工作期"窗口中的可见工作区之外,Visual FoxPro 还通过数据工作期自动为表单或表单集的每个实例提供一个独立的环境。一个数据工作期是指表单、表单集或报表使用的当前动态工作环境。每个数据工作期包含了它自己的一组工作区,这些工作区包括在工作区中打开的表、表索引以及表之间的关系。

8.4.1 查看工作区

打开"数据工作期"窗口,可查看在一个 Visual FoxPro 工作期中已打开表的列表。

若要打开"数据工作期"窗口,从"窗口"菜单选择"数据工作期"。

或者使用 SET 命令。

当在"命令"窗口中键入 SET 时,Visual FoxPro 打开"数据工作期"窗口如图 8-15 所示,并显示在当前数据工作期中打开表的工作区别名。

图 8-15 数据工作区

8.4.2　在工作区中打开表

若要在工作区中打开表，可以在“数据工作期”窗口中，选择“打开”。出现如图 8-16 所示对话框，在数据库 gzk 中我们选择表 jbqk. dbf，并以独占方式打开。

图 8-16　打开对话框

或者在“命令”窗口中，键入 USE 命令。

格式：

USE < 表文件名 > [ALIAS < 别名 >] [IN < 工作区号/工作区别名/表别名 >] [AGAIN]

上述操作就相当于在命令窗口输入了以下命令：USE d:\hvfbook\hjbqk.dbf IN 0 EXCLUSIVE

如果要在最低可用工作区中打开表，可以在 USE 命令 IN 子句后面加工作区 0。例如，如果工作区 1 到 10 中都有打开的表，可以使用以下命令，在工作区 11 中 打开 jbqk 表：

USE jbqk IN 0

在“文件”菜单中选择“打开”，也可以在工作区中打开表。

8.4.3　在工作区中关闭表

可通过“数据工作期”窗口或使用语言关闭工作区中的表。

若要在工作区中关闭表，可在“数据工作期”窗口中选定要关闭的表别名，然后选择“关闭”。

或者键入不带表名的 USE 命令。

或者使用 USE 命令的 IN 子句，指出想要关闭的表所在的工作区。

当执行不带表名的 USE 命令，并且在当前所选工作区中有打开的表文件时，则关闭该表。例如，可以使用以下代码打开 jbqk 表，显示“浏览”窗口，然后关闭此表：

```
USE jbqk
BROWSE
USE
```

当在同一工作区中打开其他表，或者发出有 IN 子句的 USE 命令并指定当前工作区时，可以自动关闭已打开的表。下面的代码通过发出 USE IN 命令和 jbqk 表别名，打开 jbqk 表，然后显示它，最后关闭：

```
use  jbqk
BROWSE
USE IN jbqk
```

在一个工作区中，不能同时打开多个表。

8.4.4 引用工作区

可以像下面显示的那样，在下一个可用工作区中打开表之前，指出该工作区的编号：

```
SELECT 0
```

1. 默认表别名

表别名是 Visual FoxPro 用来指定在一个工作区中打开的表的名称。打开一个表时，Visual FoxPro 自动使用文件名作为默认表别名。例如，如果用下面的命令在 0 号工作区打开文件 jbqk. dbf，则自动为表指定默认别名 jbqk：

```
SELECT 0
USE jbqk
```

然后，可以使用别名 jbqk 在命令或函数中标识该表。也可创建自己的别名。

2. 创建用户自定义别名

在打开表时，可以为它指定用户自定义的表别名。

若要以用户自定义的别名打开表，键入带表别名的 USE 命令。

格式：USE〈表文件名〉ALIAS〈别名〉

例如，可以使用以下命令，在 0 号工作区中打开文件 jbqk. dbf，并为它指定一个别名 people：

```
SELECT 0
USE jbqk ALIAS people
```

然后必须使用别名 people 引用打开的表。别名最多可以包括 254 个字母、数字或下划线，但首字符必须是字母或下划线。如果所提供的别名包含不支持的字符，则 Visual FoxPro 会自动创建一个别名。

3. 使用 Visual FoxPro 指定的别名

在以下情况中，Visual FoxPro 自动指定表的别名：

如果使用包含 AGAIN 子句的 USE 命令，同时在多个工作区中打开同一个表，并且在每个工作区中打开该表时没有指定别名。

在前 10 个工作区中指定的默认别名是工作区字母 A 到 J，在工作区 11 到 32 767 中指定的别名是 W11 到 W32 767。您可以像使用任何默认别名或用户自定义别名那样，使用这些 Visual FoxPro 指定的别名来引用在一个工作区中打开的表。

4. 使用别名选择工作区

可以使用 SELECT 命令从一个工作区移到另一个工作区。例如，如果在一个工作区中打开 jbqk. dbf 并指定默认别名 jbqk，可以使用下面的 SELECT 命令移到这个工作区中：

```
SELECT jbqk
```

5. 引用在其他工作区中打开的表

在别名后加上点号分隔符“. ”或“ - > ”操作符，然后再接字段名，可以引用其他工作区中的字段。例如，可以使用以下代码，在一个工作区中访问其他工作区中打开的 jbqk 表的姓名字段，其格式为 jbqk. 姓名。

如果要引用的表是用别名打开的，则也可以使用别名。例如，如果 jbqk 表是使用别名 people 打开的，那么，可以使用以下代码引用表中的姓名字段：people. 姓名。

可以在一个表所在的工作区之外，使用表名或表别名来明确标识该表。

8.4.5　设置表间的临时关系

索引用于在数据库中建立表间的永久关系。与 SET RELATION 命令设置的临时关系不同，永久关系在每次使用表时不需要重新创建。但是，由于永久

关系并不控制表中记录指针间的关系,因此在开发 Visual FoxPro 应用程序时,不仅需用永久关系,也需使用临时的 SET RELATION 关系。

在建立表间的临时关系后,就会使得一个表(子表)的记录指针自动随另一个表(父表)的记录指针移动。这样,便允许当在关系中"一"方(或父表)选择一个记录时,会自动去访问表关系中"多"方(或子表)的相关记录。

1. 临时关联表

可以通过"数据工作期"窗口或使用语言来创建表间的临时关系。

若要临时关联表,可以在"数据工作期"窗口中选定要关联的表,然后使用"关系"按钮创建关系。

或者使用 SET RELATION 命令。

格式:

SET RELATION TO [<关联表达式 1>] INTO <工作区> / <别名> [, <关联表达式 2> INTO <工作区> / <别名> …]] [IN <工作区> / <别名>] [ADDITIVE]

功能:在两个表之间建立关联。

参数描述:

<关联表达式 1>:指定用来在子表和父表之间建立关联的关联表达式。关联表达式经常是子表主控索引的索引表达式。

<关联表达式>可以是下列三种参数之一:

(1) <关联表达式>是记录号函数 RECNO()。此时,两个或多个关联表之间的联系是根据记录号来进行关联的,关联表与被关联表之间当前记录号保持相等。如果关联表记录的记录号大于被关联表的记录总数,则被关联表的当前记录指针指向最后一条记录的下一条记录,EOF()函数值为.T.。

(2) <关联表达式>是数值型表达式。此时,在表达式中通常含有 RECNO()函数,每当关联表的记录指针重新定位时,被关联表的记录指针将重新定位于<关联表达式>的值所对应的记录之上。如果<关联表达式>的值大于被关联表文件的记录总数,则被关联表文件的当前记录指针指向最后一条记录的下一条记录,EOF()函数值为.T.。

(3) <关联表达式>选择两个表的公共字段建立关联,使用这种方法时,要求被关联的表文件必须是按指定的公共关联字段建立并打开了索引文件。

INTO <工作区> / <别名>:指定被关联表的工作区或别名,也可以是被关联表的别名。

ADDITIVE:建立关联时,如果命令中不使用 ADDITIVE 子句,则父表以前建立的关联将自动解除;若使用了 ADDITIVE 子句,则父表以前建立的关联

仍然保留。

SET RELATION 命令可以建立两表之间的关系，其中一个是在当前选定工作区中打开的表，而另一个则是在其他工作区中打开的表。通常这两个表具有相同字段，而且用来建立关系的表达式常常就是子表主控索引的索引表达式。

例如，顾客可以有许多相关联的订单记录。如果在两个表共同拥有的字段之间创建关系，就能很容易地看到任何一个顾客的所有记录。下面程序中，在创建顾客 customer 表中的顾客编号 cust_id 字段和订单 orders 表中的 cust_id 索引标识之间的关系时，使用了两个表都有的字段 cust_id。

使用 SET RELATION 建立两个表间的关系：(表 8-1)

表 8-1

代　　码	注　　释
USE customer IN 1	在 1 号工作区中打开 customer 表 (父表)
USE orders IN 2	在 2 号工作区中打开 orders 表 (子表)
SELECTorders	选定子表工作区
SETORDER TO TAG cust_id	使用索引标识 cust_id 指定子表的顺序
SELECT customer	选定父表工作区
SET RELATION TO cust_id INTO orders	创建父表和子表的主控索引之间的关系
SELECTorders BROWSENOWAIT SELECT customer BROWSE NOWAIT	打开两个“浏览”窗口，请注意：移动父表的记录指针会改变在子表中显示的数据集合

“数据工作期”窗口显示两个打开的表：orders 和 customer，并通过 SET RELATION 命令建立关系。

2. 解除关联

用 SET RELATION 命令建立关联之后，当移动关联表的记录指针时，被关联表的记录指针也相应要移动，并且将要引起读/写磁盘操作，这样会降低系统的性能，因此，当某些关联不再使用，或暂时不再使用时，应及时解除关联，以提高系统的运行速度。

格式 1：SET RELATION TO

功能：删除当前工作区表与其他工作区表建立的关联。

格式 2：SET RELATION OFF INTO < 工作区号 > / < 别名 >

功能：删除当前工作区与由 < 工作区号 > / < 别名 > 指定的工作区中表

建立的关联。该命令必须在父表所在的工作区执行。例如：要关闭当前工作区与 C 工作区建立的关联，可以作下述命令进行：

SET RELATION OFF INTO C

当用 USE 关闭某些表时，系统将自动删除掉与它建立的关联。如果关闭的是父表文件，则它与子表的关联将全被删除。

当关闭父表时，将自动关闭与父表建立的所有关联。

3. 表的连接

格式：JOIN WITH <别名> / <工作区> TO <表文件名> FOR <条件> [FIELDS <字段名表>]

功能：连接当前工作区中打开的表和 <工作区> / <别名> 指定的表，生成 <表文件名> 规定的新的表文件。

参数描述：

<别名> / <工作区>：指定第二个表的别名或所在的工作号。

FOR <条件>：指定一个筛选条件。若 <条件> 的值为真，则向新表中写入一个新记录。

[FIELDS <字段名表>]：指定新表中包含的字段的列表。两个表中指定的字段都可以包含在 <字段名表> 中。

说明：

JOIN 命令也可以连接两个以上的表，这时可先连接其中的两个，生成一个新的表文件后再利用 JOIN 命令连接另外的表。

在应用程序中,若要创建自定义并且可更新的数据集合,可以使用视图。视图兼有表和查询的特点:与查询相类似的地方是,可以用来从一个或多个相关联的表中提取有用信息;与表相类似的地方是,可以用来更新其中的信息,并将更新结果永久保存在磁盘上。可以用视图使数据暂时从数据库中分离成为自由数据,以便在主系统之外收集和修改数据。

创建视图和创建查询的过程类似,主要的差别在于视图是可更新的,而查询则不行。查询是一种 SQL - SELECT 语句,作为文本文件以扩展名 . qpr 存储。若想从本地或远程表中提取一组可以更新的数据,就需要使用视图。

§9.1 创建视图

使用视图,可以从表中提取一组记录,改变这些记录的值,并把更新结果送回到源表中。您可以从本地表、其他视图、存储在服务器上的表或远程数据源中创建视图,如从 Microsoft SQL Server 和 ODBC 中创建视图。通过打开"发送更新"选项,在更新或更改视图中的一组记录时,由 Visual FoxPro 将这些更新发送到源表中。

视图是数据库中的一个特有功能,只有在包含视图的数据库打开时,才能使用视图。

9.1.1 利用"视图向导"创建视图

我们使用向导通创建一个单表 jbqk. dbf 的本地视图 jbqk,该视图中包括

字段编号、姓名、性别和职称，其操作步骤为：

在“项目管理器”中，选定一个数据库 gzk，并把它打开。

单击数据库菜单，选定“本地视图”，选择“新建”按钮，出现如图 9-1 所示对话框。

图 9-1　新建本地视图

选择“视图向导”按钮。进入“本地视图向导”步骤 1，见图 9-2 所示对话框，在此框图中我们选择 jbqk 为数据源，在可用字段中我们选择编号、姓名、性别和职称。

图 9-2　向导步骤 1

单击“下一步”，向导进入“本地向导步骤 3”，如图 9-3 所示对话框，询问

我们是否仅显示部分记录，我们可以选择我们需要的条件进行操作。在这里我们显示所有记录，不必设置条件。

图 9-3　向导步骤 3

单击“下一步”，进入“本地视图向导”步骤 4，如图 9-4 所示对话框，询问记录的排列顺序。

图 9-4　向导步骤 4

单击“下一步”,进入“本地视图向导”步骤 5,如图 9-5 所示对话框,询问视图保存方式,我们选择“保存本地视图”,然后单击“完成”。

图 9-5 向导步骤 5

单击“完成”后,屏幕弹出“视图名”窗口,如图 9-6 所示,输入视图名,然后按“确认”按钮,回到系统主菜单。

图 9-6 视图名窗口

在系统主菜单下,我们再来看数据库 gzk,你会发现 jbqk 视图已添加进数据库中,如图 9-7 所示。

在数据库设计器中,我们选中 jbqk 视图,单击鼠标右键,选择“浏览”,结果如图 9-8 所示。

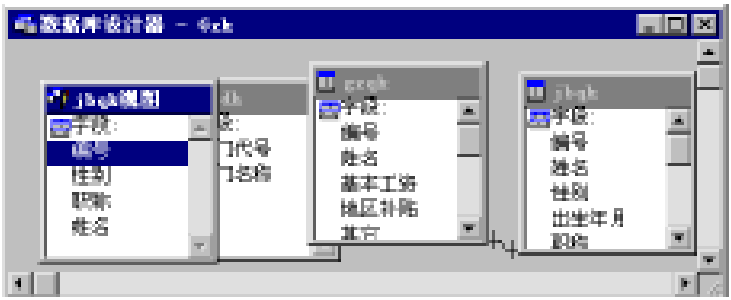


图 9-7　gzk 数据库

编号	性别	职称	姓名
1000001	男	讲师	赵一同
1000002	女	助教	王小小
1000003	女	副教授	周明月
1000004	男	教授	吴建国
1000005	女	讲师	孙晓静
1000006	女	副教授	肖强强
1000007	男	副教授	陈若无
1000008	女	助教	方淑珍

图 9-8　视图内容

9.1.2　利用视图设计器创建视图

若要创建本地表的视图，请使用“视图设计器”。本地表包括本地 Visual FoxPro 表、任何使用 .dbf 格式的表和存储在本地服务器上的表。

若要使用“视图设计器”，首先应创建或打开一个数据库，当展开“项目管理器”中数据库名称旁边的加号 + 时，“数据”选项卡上将显示出数据库中的所有组件。本地视图存储在数据库中。

我们来创建基于多表的本地视图，该视图包括编号、姓名、职称和部门名称。其操作步骤为：

从“项目管理器”中选定一个数据库 gzk。

单击“数据库”符号旁的加号 +。

在“数据库”下，选定“本地视图”并选择“新建”按钮。

选择“新建视图”按钮。出现如图 9-9 所示对话框。

在“添加表或视图”对话框中，选定想使用的表或视图，我们选择“jbqk”，再选择“添加”。用同样方式添加表 bmdh。

图 9-9 “添加表或视图”对话框

在添加完表后，进入“联接条件”窗口，如图 9-10 所示，我们选择好联接条件后，单击“确定”。

图 9-10 联接条件

在随后出现的“视图设计器”中，如图 9-11 所示，显示了选定的表或视图。

在“字段”选项卡上，选择要在视图结果中显示的字段编号、姓名、职称和部门名称。

图 9-11　“视图设计器”

关闭视图设计器窗口，进入系统窗口（见图 9-12），询问是否要将所做更改保存到视图 1，我们单击“是”。

图 9-12　系统窗口

在“视图”保存窗口（见图 9-13），我们输入视图名“综合视图”。单击“确定”。这样，整个操作完毕。

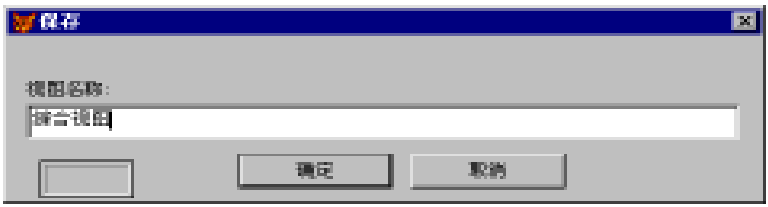


图 9-13　保存窗口

打开“数据库”菜单，选择“浏览”，进入“视图浏览”窗口，综合视图内容，如图 9-14 所示。

图 9-14　视图内容

§9.2　利用视图更新数据

可用本地或远程视图更新数据。

在“视图设计器”中（见图 9-15），“更新条件”选项卡可以控制把对远程数据的修改（更新、删除、插入）回送到远程数据源中的方式，也可以打开和关闭对表中指定字段的更新，并设置适合服务器的 SQL 更新方法。

图 9-15　更新数据

如果希望在表的本地版本上所作的修改能回送到源表中，需要设置“发送 SQL 更新”选项，必须至少设置一个关键字段来使用这个选项。如果选择的表中有一个主关键字段并且已在字段选项卡中，则“视图设计器”自动使用表中的该主关键字段作为视图的关键字段。

当在“视图设计器”中首次打开一个表时，“更新条件”选项卡会显示表中哪些字段被定义为关键字段。Visual FoxPro 用这些关键字段来惟一地标识那些已在本地修改过的远程表中的更新记录。

若要设置关键字段，在“更新条件”选项卡中，单击字段名旁边的“关键列”。我们在此设置编号为关键字段。

如果已经改变了关键字段，而又想把它们恢复到源表中的初始设置，请选择“重置关键字”。Visual FoxPro 会检查远程表并利用这些表中的关键字段。

可以指定任一给定表中仅有某些字段允许更新。若使表中的任何字段是可更新的，在表中必须有已定义的关键字段。如果字段未标注为可更新的，用户可以在表单或浏览窗口中修改这些字段，但修改的值不会返回到远程表中。

若要使字段为可更新的，其操作步骤为：

在“更新条件”选项卡中，单击字段名旁边的“可更新列”(笔形)。

在“更新条件”选项卡中使字段可更新，我们在这里选择“部门名称”为可更新字段。

若要使所有字段可更新，可在“更新条件”选项卡中，选择“全部更新”。

注意　若要使用“全部更新”，在表中必须有已定义的关键字段。“全部更新”不影响关键字段。

在“更新条件”选项卡中，“SQL WHERE 子句包括”框中的选项可以帮助管理遇到多用户访问同一数据时应如何更新记录。在允许更新之前，Visual FoxPro 先检查远程数据源表中的指定字段，看看它们在记录被提取到视图中后有没有改变，如果数据源中的这些记录被修改，就不允许更新操作。

在“更新条件”选项卡中设置 SQL WHERE 子句。这些选项决定哪些字段包含在 UPDATE 或 DELETE 语句的 WHERE 子句中，Visual FoxPro 正是利用这些语句将在视图中修改或删除的记录发送到远程数据源或源表中，WHERE 子句就是用来检查自从提取记录用于视图中后，服务器上的数据是否已改变。我们在这里选择“关键字和可更新字段”。

在“更新条件”选项卡中设置 SQL WHERE 子句，这些选项决定哪些字段包含在 UPDATE 或 DELETE 语句的 WHERE 子句中，Visual FoxPro 正是利用这些语句将在视图中修改或删除的记录发送到远程数据源或源表中，WHERE 子句就是用来检查自从提取记录用于视图中后，服务器上的数据是否已改变。

当源表中的关键字段被改变时，使更新失败，应选择的 SQL WHERE 选项为关键字段。

当远程表中任何标记为可更新的字段被改变时，使更新失败。应选择的 SQL WHERE 选项为关键字和可更新字段。

当在本地改变的任一字段在源表中已被改变时，使更新失败。应选择的 SQL WHERE 选项为关键字和已修改字段。

当远程表上记录的时间戳在首次检索之后被改变时，使更新失败（仅当远程表有时间戳列时有效），应选择的 SQL WHERE 选项为关键字和时间戳。

当我们把上述条件都设置好了之后，关闭“视图设计器”，进入系统提示窗口，如图 9-16 所示，单击“是”，结束操作。

图 9-16 系统提示

在系统菜单下，我们就可以利用“综合视图”来更新部门字段的内容了。

§9.3 创建查询

在很多情况下都需要建立查询，例如为报表组织信息、即时回答问题或者查看数据中的相关子集。无论目的是什么，建立查询的基本过程是相同的。

使用“查询设计器”可以搜索那些满足指定条件的记录，也可以根据需要对这些记录排序和分组，并根据查询结果创建报表、表及图形。

当确定了要查找的信息，以及这些信息存贮在哪些表和视图中后，可以通过以下几个步骤来建立查询：

使用“查询向导”或“查询设计器”开始建立查询。

选择出现在查询结果中的字段。

设置选择条件来查找可给出所需结果的记录。

设置排序或分组选项来组织查询结果。

选择查询结果的输入类型 表、报表、浏览等等。

运行查询。

9.3.1　利用“查询向导”建立查询

若要快速创建查询，可以利用 Visual FoxPro“查询向导”。“查询向导”将询问您从哪些表或视图中搜索信息，并根据您对一系列提问的回答来建立查询。

我们通过建立一查询，其包括 jbqk 和 gzqk 两个表中的编号、职称、党员否和基本工资四个字段，用向导建立此查询，其操作步骤为：

在“项目管理器”中选择“数据”选项卡，然后选择“查询”。

选择“新建”。

选择“查询向导”按钮。出现如图 9-17 所示对话框。

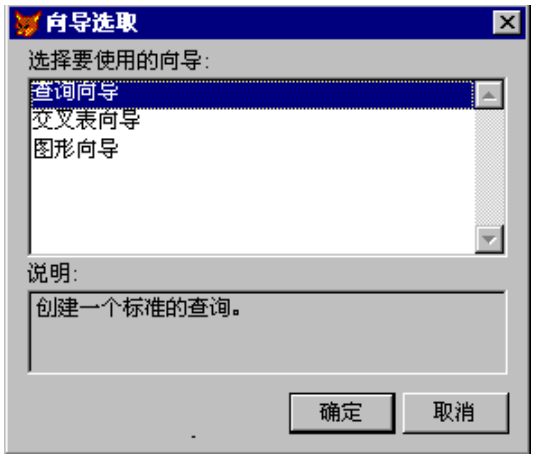


图 9-17　查询向导

您也可以从“工具”菜单中选择“向导”命令来启动向导。

选择所建查询的类型“查询向导”，然后单击“确定”。

按照向导屏幕上的指令进行以下操作：

步骤 1　字段选取

我们可以从几个表和视图中选择字段。首先，从一个表 jbqk 内选择字段编号、职称和党员否并且移到“选定字段”框中，然后从另外表 gzqk 选择基本工资并移动字段，如图 9-18 所示。

图 9-18　查询向导步骤 1

步骤 2　为表建立关系

可以从字段列表中选择匹配的字段用以决定表或视图之间的关系，如图 9-19 查询向导步骤 2 所示。在这里我们设置两表的编号要相同。

图 9-19　查询向导步骤 2

步骤 2a　包含记录

通过只从两个表中选择匹配的记录或者任意一个表中的所有记录，可以限制您的查询。默认情况下，只包含匹配的记录。如果想创建一个内部联接，

选择“仅包含匹配的行”。如果要创建一个外部联接，选择两个表之一的所有行，如图 9-20 所示。

图 9-20　查询向导步骤 2a

步骤 3　筛选记录

通过创建从所选的表或视图中筛选记录的表达式，可以减少记录的数目。可以创建两个表达式，然后用“与”连接，返回同时满足两个指定条件的记录，如果用“或”连接，则返回至少符合其中一个条件的记录，如图 9-21 所示。

图 9-21　查询向导步骤 3

选择“预览”,可以查看基于筛选条件返回的记录。

步骤 4　排序记录

最多选择三个字段或一个索引标识以确定查询结果的排序顺序。选择“升序”结果将按升序排序,选择“降序”结果按降序排序,如图 9-22 所示。

图 9-22　查询向导步骤 4

步骤 4a　限制记录

根据一定百分比的记录,或者选择一定数量的记录,可以进一步限制视图中的记录数目,例如,要查看前 10 个记录,可选择“数量”,然后在“部分值”框中输入 10。选择“预览”,可以查看强加的限制条件返回的记录,如图 9-23 所示。

图 9-23　查询向导步骤 4a

步骤 5　完成

保存查询后，可以像其它查询一样在“查询设计器”中打开或修改它，如图 9-24 所示。

图 9-24　查询向导步骤 4

9.3.2　利用查询设计器设计查询

使用“查询设计器”创建和修改查询。在“查询设计器”活动时，Visual FoxPro 显示“查询”菜单和“查询设计器”工具栏。

利用“查询设计器”，首先选择想从中获取信息的表或视图，指定从这些表或视图中提取记录的条件，然后按照想得到的输出类型定向查询结果，诸如浏览、报表、表、标签等等。若要保存创建的查询，可以给它指定一个名称，将查询文件保存为带 .qpr 扩展名的文件。

我们来建立一个查询 2，该查询包括“jbqk”中的编号、姓名、出生年月和部门代号。

若要启动“查询设计器”，在“项目管理器”中选择“数据”选项卡，选择“查询”。

选择“新建”。

选择“新建查询”，出现如图 9-25 所示对话框。

图 9-25　添加表或视图

在“添加表或视图”对话框中，选定想使用的表或视图，我们选择表“jbqk”，再选择“添加”。出现如图 9-26 所示查询设计器。

图 9-26　查询设计器

在“查询设计器”中选择“联接”选项卡，多表查询中，表间的联接有四种类型，分别是：

① Inner Join：内部联接，指定只有满足联接条件的记录包含在结果中，

此类型是默认的,也是最常用的;

② Right Outer Join 右联接,指定满足联接条件的记录,以及满足联接条件右侧的表中记录 (即使不匹配联接条件) 都包含在结果中;

③ Left Outer Join :左联接,指定满足联接条件的记录,以及满足联接条件左侧的表中记录 (即使不匹配联接条件) 都包含在结果中;

④ Full Join :完全联接,指定所有满足和不满足联接条件的记录都包含在结果中。

如果想修改各表间的联接,双击查询设计器上部窗口表之间的连线,系统将弹出"联接条件"对话框 或者通过打开查询设计器下部的"联接"选项卡进行。一般不应随便更改联接条件,不然会与实际数据间的关系不符。

"条件"列表中包含如下几项:

① = :指字段值与实例相等。

② LIKE:表示"字段名"栏中给出的字段值与"实例"栏中给出的文本值之间执行不完全匹配,它主要针对字符类型。例如,如设置查询条件为"jbqk.编号 LIKE 1104",那么诸如"编号"字段前四位为 1104 的记录都满足该条件。

③ = = :表示在"字段名"栏中给出的字段值与"实例"栏中给出的文本值之间执行完全匹配检查,它也主要是针对字符类型的。

④ > :即为"字段名"栏中给出的字段的值应大于"实例"栏中给出的值。

⑤ > = :即为"字段名"栏中给出的字段的值应大于或等于"实例"栏中给出的值。

⑥ < :即为"字段名"栏中给出的字段的值应小于"实例"栏中给出的值。

⑦ < = :即为"字段名"栏中给出的字段的值应小于或等于"实例"栏中给出的值。

⑧ Is Null:指定字段必须包含 Null 值。

⑨ Between:即为输出字段的值应大于或等于"实例"栏中的最小值,而小于或等于"实例"栏中的最大值。

⑩ IN (在…之中) :即为输出字段的值必须是"实例"栏中所给出值中的一个,在"实例"栏中给出的各值之间以逗号分隔。

此外,"联接"选项卡中的"否"列用于指定. NOT. 条件,"逻辑"列用于设置各联接条件和筛选条件之间的逻辑关系 (无、. AND. 和. OR.),"大小写"列用于指定是否区分大小写。下方的"插入"和"移去"按钮分别用于增加或移去查询条件。

最后,在设置筛选条件时,我们应注意如下几点:

① 备注字段和通用字段不能用于设置查询条件;

② 逻辑值的前后必须使用句点号,如.T.;

③ 只有当字符串与查询的表中字段名相同时,要用引号将字符串括起来,否则不需要用引号将字符串括起来;

④ 日期不必用花括号括起来。

"筛选"选项卡可确定用于选择记录的字段和比较准则,以及输入与该字段进行比较的示例值。

排序决定了查询输出结果中记录或行的先后顺序,我们可以通过"排序依据"选项卡设置查询的排序次序。

在查询设计器当中,在可用字段里我们选择编号、姓名、出生年月和部门代号。然后关闭该设计器。系统又弹出如图 9-27 所示对话框,询问我们是否保存修改,我们单击"是",在随后的保存对话框中输入名字"查询 2",单击"保存",这样一个查询就建好了。

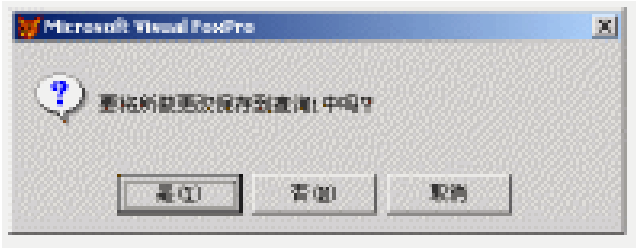


图 9-27 系统提示

§9.4 使用查询

9.4.1 运行查询

在完成了查询设计并指定了输出目的地后,可以用"运行"按钮启动该查询。

在"项目管理器"中选定查询的名称,然后选择"运行"。

Visual FoxPro 执行用"查询设计器"产生的 SQL Select 语句,并把输出结果送到指定的目的地。若尚未选定输出目的地,结果将显示在"浏览"窗口中。

9.4.2　定向输出查询结果

如果想使结果输出到不同的目的文件，可将结果定向输出到表单、表、报表或其他目的文件。如果想了解 SQL 语句，可查看生成的 SQL 语句。

若要选择查询结果的去向，在“项目管理器”中，选择“查询 1”，再选择“修改”，出现“查询设计器”。可从“查询”菜单中选择“查询去向”，或在“查询设计器”工具栏中选择“查询去向”按钮，此时将显示图 9-28“查询去向”对话框，从表 9-1 中您可以在其中选择将查询结果送往何处。

表 9-1　选择查询输出的去向

简　　要	选择的输出选项
在“浏览”窗口中显示查询结果	浏览
将查询结果存贮在一个命名的临时只读表中	临时表
使查询结果保存为一个命名的表	表
使查询结果可用于 Microsoft Graph（Graph 是包含在 Visual FoxPro 中的一个独立的应用程序）	图形
在 Visual FoxPro 主窗口或当前活动输出窗口中显示查询结果	屏幕
将输出送到一个报表文件（.frx）	报表
将输出送到一个标签文件（.lbx）	标签

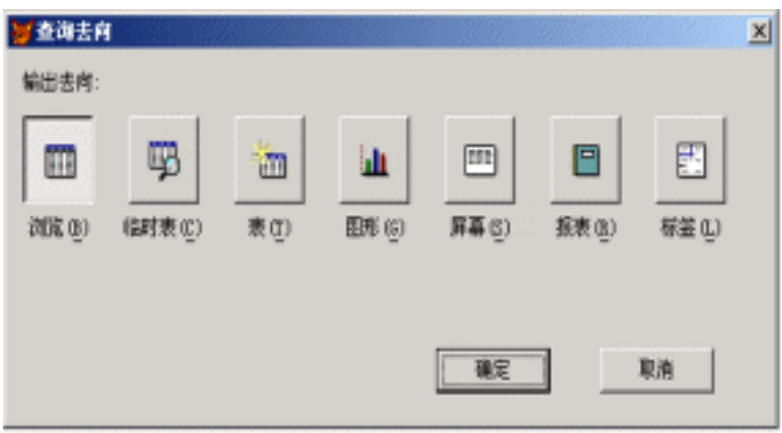


图 9-28　“查询去向”对话框

我们在输出去向中选择“表”，此时会在“查询去向”中出现询问表名的对话框，如图 9-29 所示，输入“查询 3”，单击“确定”，返回系统主菜单。

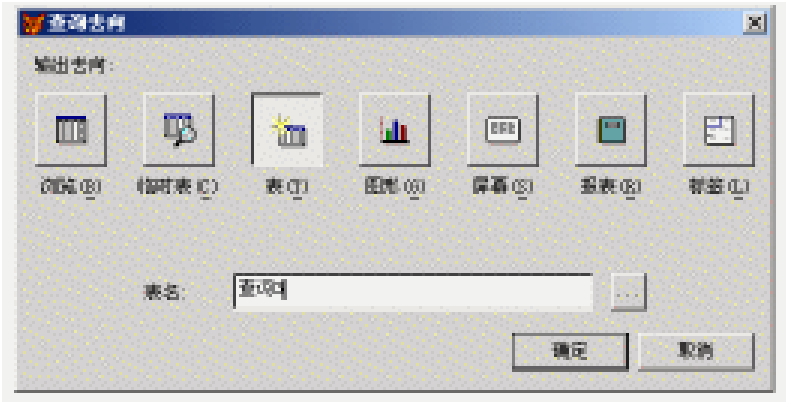


图 9-29　查询去向——表

在主菜单下,打开“显示”菜单,选择“浏览”,可以看到该表内容。

在输出去向中,如果我们选择“图形”,单击“确定”,返回系统主菜单。我们在主菜单下选择“运行查询”,进入“图形向导”步骤 2——定义布局,如图 9-30所示。

图 9-30　“图形向导”步骤 2—定义布局

在步骤 2 中我们将编号移至“坐标轴”,将“基本工资”移至“数据系列”,按“下一步”,出现如图 9-31 所示对话框。选择图 9-31 第一行第五个样式,单

击“下一步”,出现如图 9-32 所示对话框。

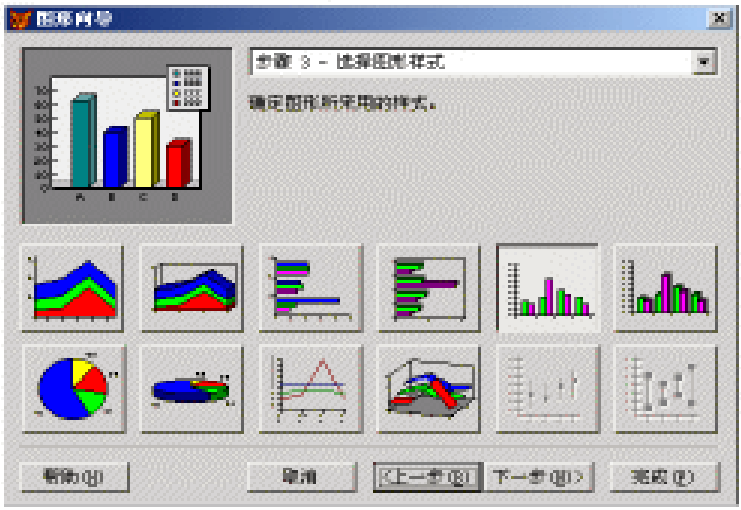


图 9-31 选择样式

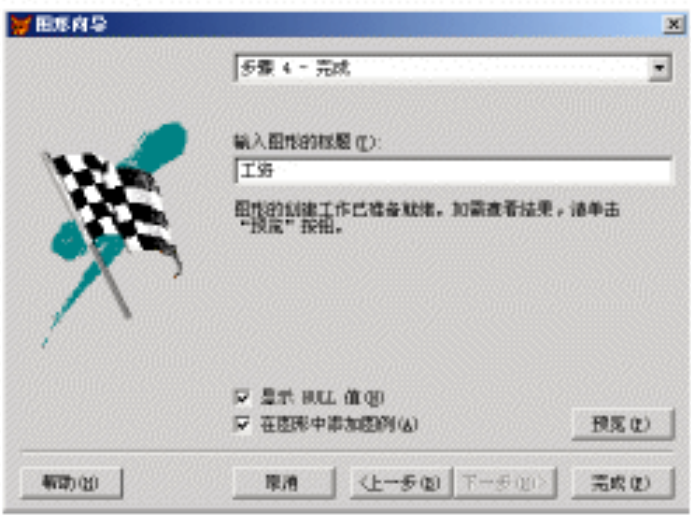


图 9-32 步骤 4——完成

在步骤 4 中,我们给图加上一个标题“工资”,单击“完成”,进入“保存”窗

口，输入该图形的名字，结果如图 9-33 所示。

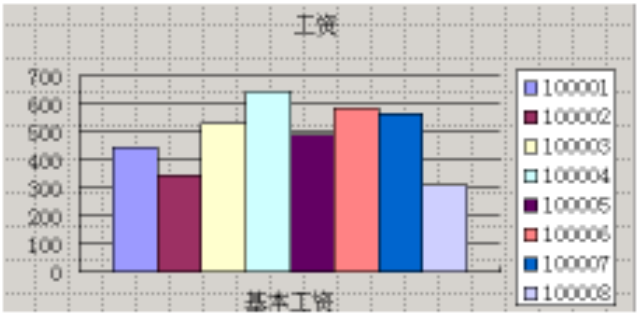


图 9-33　图形结果

许多选项都有一些可以影响输出结果的附加选择。例如，“报表”选项可以打开报表文件，并在打印之前定制报表，您也可以选用“报表向导”帮助自己创建报表。有关详细内容，请参阅第 10 章《设计报表和标签》。

第10章 设计报表和标签

报表和标签为在打印文档中显示并总结数据提供了灵活的途径。报表包括两个基本组成部分：数据源和布局。数据源通常是数据库中的表，但也可以是视图、查询或临时表。视图和查询将筛选、排序、分组数据库中的数据，而报表布局定义了报表的打印格式。在定义了一个表、一个视图或查询后，便可以创建报表或标签。

§10.1 创建报表

10.1.1 一对多报表向导

使用数据库中的一个自由表、一个表或视图来创建报表。当经过一系列简捷的步骤时，向导提示您回答简单的问题，从中可以指定用以创建报表中控制的表和字段。

我们以生成一报表名为“报表1”为例，内容包括基本情况表中的编号、姓名、职称，以及部门表中的部门名称，创建一个报表，它将父表和子表的记录分组。在一系列简捷的步骤中，向导提示您回答一些简单的问题，从中可以指定表和字段，用这些表和字段来创建报表。

若要运行一对多报表向导，打开“项目管理器 gzgl”，打开“文档”选项卡，选中“报表”，单击“新建”，在系统弹出的对话框中，我们选择“新建向导”，在随后弹出的对话框中在“向导选取”对话框中选择“一对多报表向导”，进入步骤1。

步骤 1　从父表选择字段

从图 10-1 中我们可以看到，只能从单个的表或视图中选取字段。我们首先把基本情况表中的三个字段选中。单击“下一步”，进入步骤 2。

图 10-1　从父表中选取字段

步骤 2　从子表选择字段

从图 10-2 中我们可以看到，子表也只能从单个的表或视图中选取字段。我们把部门表中的部门名称选中。单击“下一步”，进入步骤 3。

图 10-2　从子表中选取字段

步骤 3　为表建立关系

在图 10-3 中，我们可以从字段列表中接受或选择决定表之间关系的字段。我们在这里选择两表间部门代号相同。然后单击“下一步”，进入步骤 4。

图 10-3　建立两表间关系

步骤 4　排序记录

在图 10-4 中，按照结果排序的顺序选择字段或索引标识，我们选择按“编号”升序，然后单击“下一步”。

图 10-4　排列记录

步骤 5　选择报表样式

在图 10-5 中，选择报表样式，系统提供了一些常用样式供我们选择，当单击任何一种样式时，向导都在放大镜中更新该样式的示例图片。我们选择账务式，方向选择纵向，单击“下一步”。

图 10-5　选择报表样式

步骤 6　完成

在图 10-6 中，如果选择的字段数与报表的一行的宽度不一致，这些字段会放在下一行。如果不想自动换行，清除“对不能容纳的字段进行折行处理”选项。

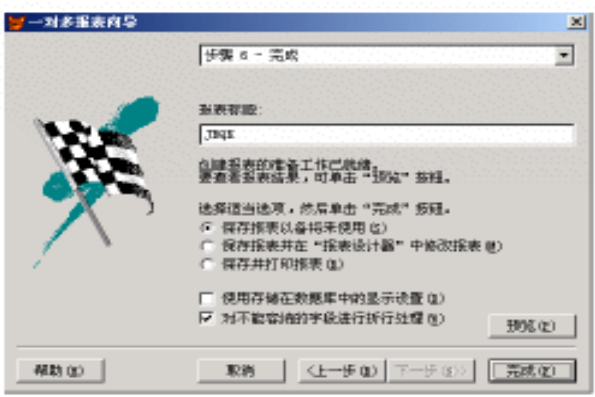


图 10-6　完成窗口

单击“预览”按钮,可以在退出向导之前显示报表 (如图 10-7 所示)。

单击“完成”,系统弹出一对话框,询问报表保存的位置,我们把文件保存在“d:\vfbook”中,取名为“报表 1”。

图 10-7 预览报表

向导保存报表之后,可以像其他报表一样,在“报表设计器”中打开并修改它。

10.1.2 利用报表生成器创建报表

通过设计报表,可以用各种方式在打印页面上显示数据。使用“报表设计器”可以设计复杂的列表、总结摘要或数据的特定子集,例如发票。设计报表有以下四个主要步骤:

① 决定要创建的报表类型。

② 创建报表布局文件。

③ 修改和定制布局文件。

④ 预览和打印报表。

创建报表之前,应该确定所需报表的常规格式。报表可能同基于单表的电话号码列表一样简单,也可能复杂得像基于多表的发票那样。另外您还可以创建特殊种类的报表,例如,邮件标签便是一种特殊的报表,其布局必须满足专用纸张的要求。

报表布局文件具有 .frx 文件扩展名,它存储报表的详细说明。每个报表文件还有带 .frt 文件扩展名的相关文件。报表文件指定了想要的域控件、要打印的文本以及信息在页面上的位置。若要在页面上打印数据库中的一些信息,可通过打印报表文件达到目的。报表文件不存储每个数据字段的值,只存

储一个特定报表的位置和格式信息。每次运行报表，值都可能不同，这取决于报表文件所用数据源的字段内容是否更改。

在“项目管理器”窗口中，选择“报表”，选择“新建”，在随后出现的对话框中，选择“新建报表”。此时显示“报表设计器”，如图 10-8 所示。可以使用“报表设计器”的任一功能来添加控件和定制报表。

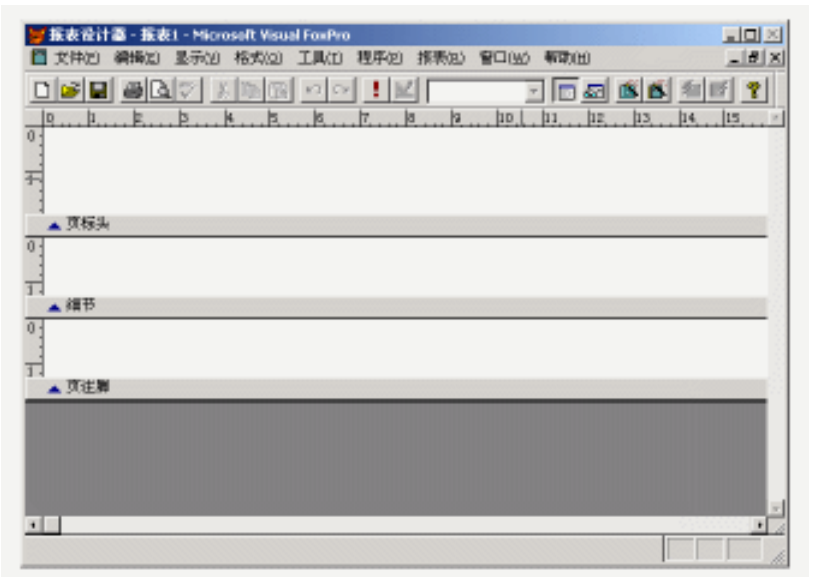


图 10-8　报表生成器

在报表生成器中，我们想增加标题和总结，单击菜单“报表”中的“标题和总结”，弹出如图 10-9 所示对话框，我们选中标题和总结，然后单击“确定”。

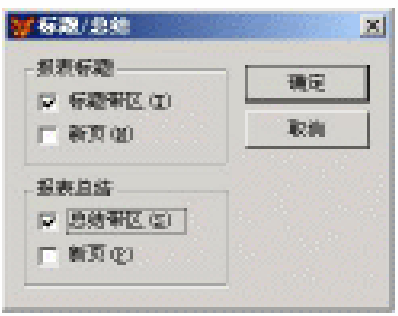


图 10-9　标题和总结

步骤 1　要向数据环境中添加表或视图

从“显示”菜单中，选择“数据环境”，从“数据环境”菜单中，选择“添加”。

在“添加表或视图”对话框中，从“数据库”框中选择一数据库 gzgl，在“选定”区域中选取“表”或“视图”，您的选择决定了出现在“数据库中的表/视图”框中字段列表的内容。

在“添加表或视图”对话框中从“数据库中的表”下选择 jbqk 表，再单击“添加”，然后单击“关闭”，即可将 jbqk 表添加到“数据环境设计器”中，如图 10-10 所示。若要添加视图，可在“选定”选项组中先选择“视图”，再进行上述操作。

图 10-10　数据环境设计器

添加表或视图后关闭“数据环境设计器”，进入步骤 2。

步骤 2　为报表添加标题

在设计报表时，可以在报表和标签布局中插入以下类型报表控件（表 10-1）。

表 10-1

简　要　显　示	请选用下列控件
表的字段、变量和其他表达式	域控件
原义文本	标签
直线	线条
框和边界	矩形
圆、椭圆、圆角矩形和边界	圆角矩形
位图或通用字段	图片/ActiveX 绑定控件

设置控件后，可以更改它们的格式、大小、颜色、位置和打印选项，也可仅出于参考目的而为每个控件添加注释，实际上在报表内并不打印。

欲在标题带区中加入标签“学生报表”的操作步骤：

先单击“报表控件”工具栏上的（标签）按钮，如图 10-11 所示。

图 10-11　报表控件

光标移至标题带区，在标题带区中单击，出现闪烁的光标。

在光标处开始输入标题名“职工基本情况表”。

然后进行格式设置，首先设置字号：先选中“职工基本情况表”标签，再在系统菜单中选择“格式”中的“字体”，在弹出的“字体”对话框中选择所需要的字体和字号以及字体样式。我们在这里选择“黑体”、“规则”、“二号”。

接着再进行版面设置，选择“格式”菜单中的“对齐”方式下的“水平居中”，其结果如图 10-12 所示。

图 10-12　标题设置

步骤 3　设置报表的页标头

设置报表的页标头与上面设置标题的方法相同。

首先在页标头中分别添加标签“编号”、“姓名”、“出生年月”、“职称”、“党员否”。

然后按住 Shift 键后分别单击每一个标签，将其全部选中。用与设置标题相同的方法将页标头的字体设为“黑体”、“规则”、“五号”，其结果如图10-13 所示。

标签布局调整，选中每个标签后用光标键可调整其位置，也可选中一批后

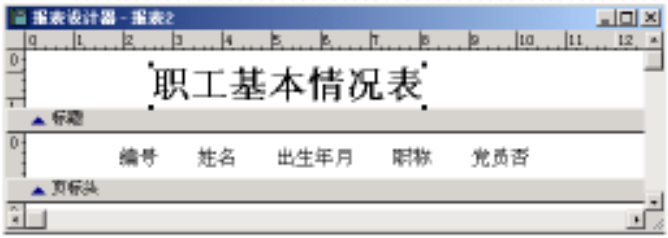


图 10-13 设计页标头

使用光标键调整位置。

步骤 4 设置细节

把“数据环境设计器”打开。

将要列在报表中的字段从“数据环境设计器”中逐个拖到细节带区，其结果如图 10-14 所示。

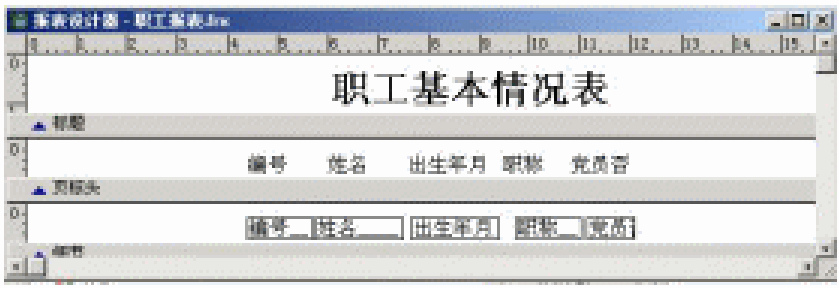


图 10-14 设计细节

设置字体字号及调整布局：方法同页标题。使用报表默认的字体和字号。

如果要控件对齐，选中细节中的全部域控件，然后在系统菜单中选择“格式”中“对齐”中的“顶边对齐”。

步骤 5 在页注脚中设置报表页码

单击工具栏上的“域控件”按钮，便可以向报表中添加域控件。然后在“页注脚”带区合适的位置单击添加一个域控件。在报表上单击时系统将打开“报表表达式”生成器。在表达式框边上，单击带有三点的按钮，进入表达式生成器（如图 10-15 所示）。在“表达式生成器”对话框中不是从“字段”列表框中选择，而是从“变量”列表框中选择“_pageno”。单击“确定”，返回报表表达式，再单击“确定”，出现如图 10-16 所示结果。

图 10-15　表达式生成器

图 10-16　设计页注脚

设置细节时也可以不从“数据环境设计器”中将字段拖到细节，而使用报表控件工具栏。

单击工具栏上的“域控件”按钮，再单击“按钮锁定”按钮，可以向报表中多次添加域控件，然后在“细节”带区合适的位置单击添加一个域控件。在报表上单击时系统将打开“报表表达式生成器”。

调整域控件大小：每个域控件都有 8 个控制柄，按住控制柄拖动鼠标可以调整域控件的大小。

步骤 6　设计数据分隔线

在此报表中，数据之间没有分隔线，若要求要有网格线，也可以用同上的基本方法，在“页标头”中的字段名上加上方框线，在“细节”中的域控件上加分隔线。

在工具栏上单击“矩形”按钮，然后在“页标题”带区划一个矩形框。

再单击工具栏上的“线条”按钮，然后在标签间划竖向分隔线。然后再在细节带区域控件的下部划一条水平线，再在域控件的中间及前后划竖向分隔线，如图 10-17 所示。

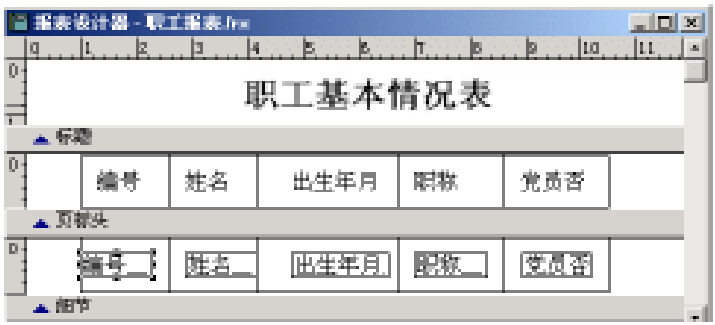


图 10-17　设计网格线

报表设计完成后，可以先单击系统菜单“常用”工具栏上的打印预览按钮，查看报表设计的效果，如果不满意还可以用上述方法进行修改，如果满意则可以结束“报表设计器”的设计工作，将报表保存到预定的目录“d：\vfbook\职工报表”中。预览结果如图 10-18 所示。

图 10-18　预览结果

§ 10.2　修改报表

要修改已生成的报表文件，应先将它打开。在项目管理器的“文档”选项卡中选择“报表”，并在展开的报表列表中选择要修改的报表，然后单击项目管理器上的“修改”按钮。

10.2.1　规划数据的位置

如果有报表布局，则可以修改数据在报表页面上的位置。使用“报表设计器”内的带区，可以控制数据在页面上的打印位置。报表布局可以有几个带区。下图为报表中可能包含的一些带区以及每个带区的典型内容。注意每个带区下的栏标识了该带区。

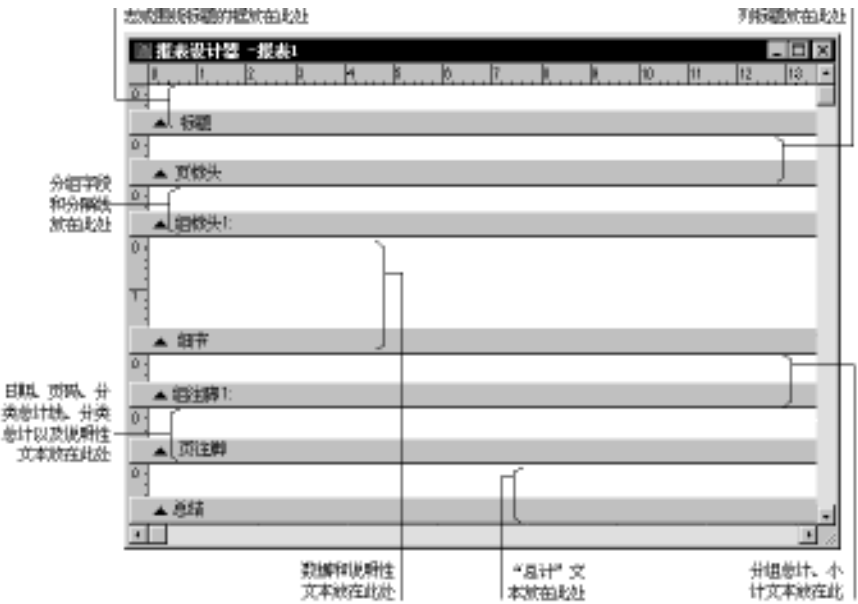


图 10-19　“报表设计器”中的报表带区

报表也可能有多个分组带区或者多个列标头和注脚带区。使用本章稍后的“按布局分组数据”部分中提供的过程，可以添加这些带区。可以使用表 10-2 决定所需的带区。

表 10-2

使用此带区	若要打印	使 用 的 命 令
标题	每报表一次	从“报表”菜单中选择“标题/总结”带区
页标头	每页一次	默认可用
列标头	每列一次	从“文件”菜单中选择“页面设置”,设置“列数”>1
组标头	每组一次	从“报表”菜单中选择“数据分组”
细节带区	每记录一次	默认可用
组注脚	每组一次	从“报表”菜单中选择“数据分组”
列注脚	每列一次	从“文件”菜单中选择“页面设置”,设置“列数”>1
页注脚	每页一次	默认可用
总结	每报表一次	从“报表”菜单中选择“标题/总结”带区

10.2.2 按布局分组数据

设计基本布局后,若要根据给定字段或其他条件对记录分组,会使报表更易于阅读。可以添加一个或多个组,更改组的顺序,重复组标头或者更改或删除组带区。分组允许明显地分隔每组记录和为组显示介绍和总结性数据。组的分隔基于分组表达式。这个表达式通常由一个以上的表字段生成,但也可以相当复杂。

分组之后,报表布局就有了组标头和组注脚带区,可以向其中添加控件。一般地,组标头带区中包含组所用字段的“域控件”,可以添加线条、矩形、圆角矩形或希望出现在组内第一条记录之前的任何标签。组注脚通常包含组总计和其他组总结性信息。

10.2.3 设计页面

可以设置报表的左边距并为多列报表设置列宽和列间距。在这种情况下,“列”一词指的是页面横向上打印的记录的数目,不是单条记录的字段数目。“报表设计器”没有显示这种设置,它仅显示了页边距内的区域,其中包含了页面中包含一条记录的一列,因此,如果报表中有多列,当更改左边距时,列宽将自动更改以显示出新的页边距。

如果更改了纸张的大小和方向设置,请确认该方向适用于所选的纸张大小。例如,如果纸张定为信封,则方向必须设置为横向。

若要设置左边距,从“文件”菜单中,选择“页面设置”。出现如图 10-20 所示对话框,在“左页边距”框中输入一个边距数值,页面布局将按新的页边距显示。

图 10-20　“页面设置”对话框

若要选择纸张大小,选择“打印设置”,在“打印设置”对话框中,从“大小”列表中选定纸张大小。

若要选择纸张方向,从“方向”区选择一种方向,再选择“确定”。

在“页面设置”对话框中,选择“确定”。

10.2.4　预览报表

通过预览报表,不用打印就能看到它的页面外观。例如,可以检查数据列的对齐和间隔,或者查看报表是否返回所需的数据。有两个选择:显示整个页面或者缩小到一部分页面。

“预览”窗口有它自己的工具栏,使用其中的按钮可以一页一页地进行预览。

注意:如果得到如下提示“是否将所做更改保存到文件?”,那么,您在选定关闭“预览”窗口时一定还选取了关闭布局文件。此时可以选定“取消”按钮回到“预览”,或者选定“保存”按钮保存所做更改并关闭文件。如果选定了“否”,将不保存对布局所做的任何更改。

10.2.5　打印报表

从“文件”菜单中,选择“打印”。

选择“确定”按钮。

注意：如果未设置数据环境，则显示“打开”对话框，并在其中列出一些表，从中可以选定要进行操作的一个表。

Visual FoxPro 把报表发送到打印机上。

§10.3 标　签

标签是多列报表布局，为匹配特定标签纸而具有相应的特殊设置。在 Visual FoxPro 里，可以使用“标签向导”或“标签设计器”迅速创建标签。

10.3.1 使用“标签向导”创建标签

利用“标签向导”是创建标签的简单方法。用向导创建标签文件后，可用“报表设计器”定制标签文件。

在“项目管理器”窗口中，选定“标签”，选择“新建”，选择“标签向导”。

按照向导屏幕上的指令操作。

步骤 1　选择表

选取一个表或视图，在图 10-21 中我们选取表 jbqk，单击“下一步”，进入步骤 2。

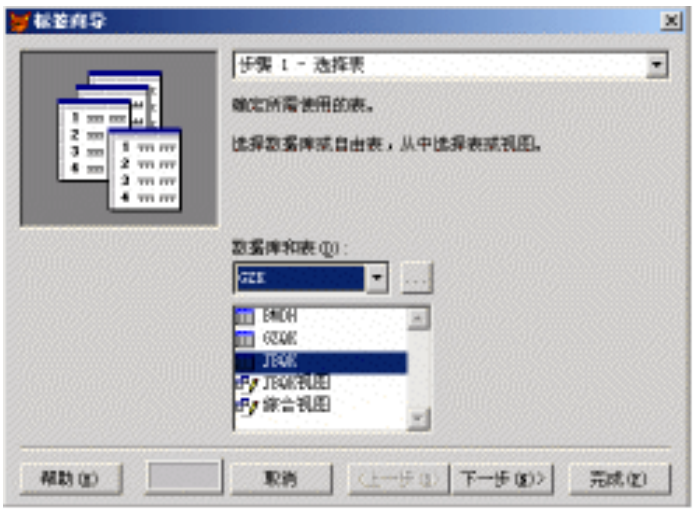


图 10-21　标签向导 1——选择表

步骤 2　选择标签类型

在图 10-22 中，向导列出 Visual FoxPro 安装的标准标签类型。向导同时列出使用 hVFP hTOOLS 目录下的 AddLabel 应用程序创建的任意标签，我们选择第一种类型，单击“下一步”，进入步骤 3。

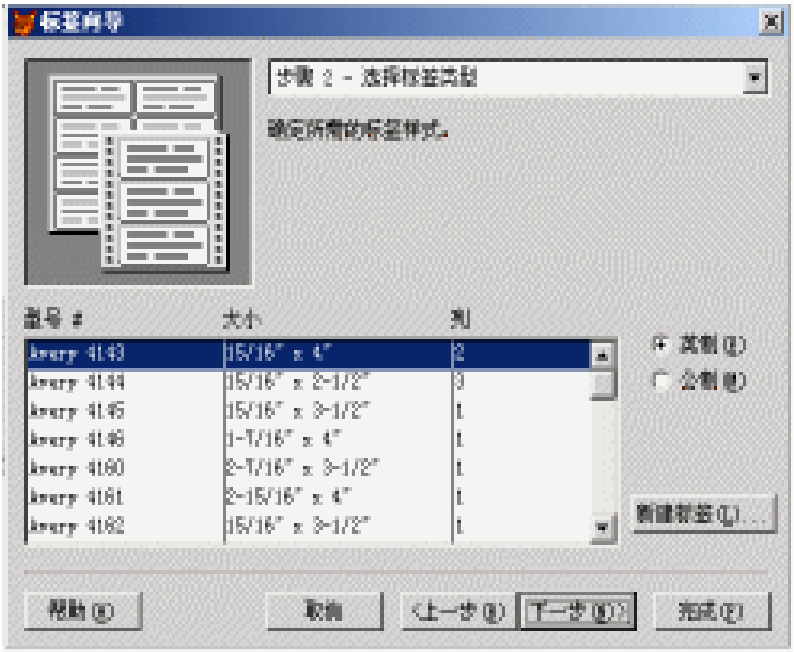


图 10-22　标签向导 2——选择标签类型

步骤 3　定义布局

按照在标签中出现的顺序添加字段。可以使用空格、标点符号、新行按钮格式化标签。我们要输入一些标题，使用在“文本”框输入文本，例如在显示编号的内容前想显示提示信息“编号”，就要在文本框中输入“编号”，结果如图 10-23 所示。单击“下一步”，进入步骤 4。

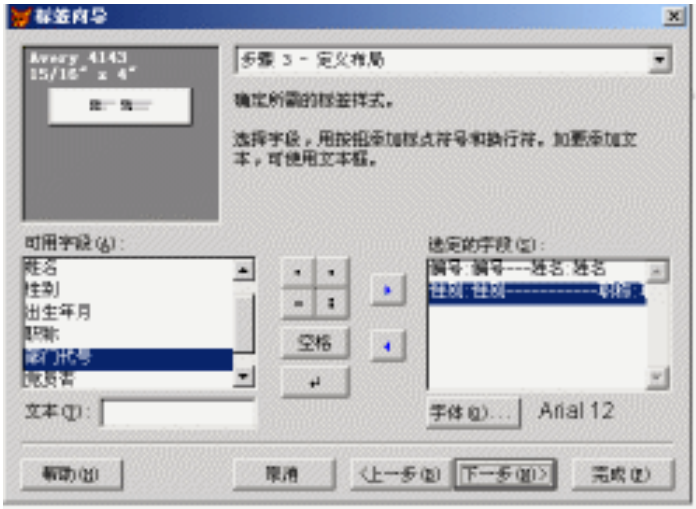


图 10-23　标签向导 3——定义布局

步骤 4　排序记录

在图 10-24 中，按照记录排序的顺序选择字段或索引标识。我们选择“编号”，单击“下一步”，进入步骤 5。

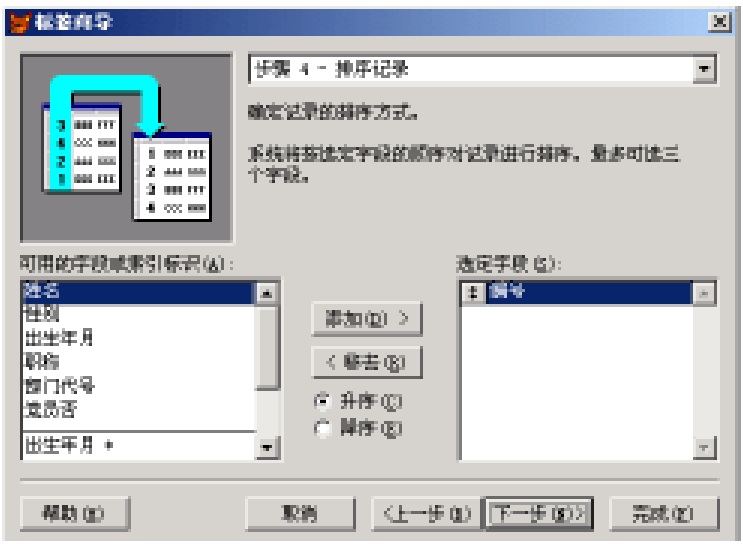


图 10-24　标签向导 4——排列记录

步骤 5　完成

完成之前，可以选择“预览”，确认您的选择。预览结果如图 10-25 所示。

图 10-25　标签向导中的预览

在图 10-26 中，单击“完成”，出现保存对话框，输入标签存放的位置“d：hvfbook h向导标签”，向导保存标签之后，可以像其他标签一样，在“标签设计器”中打开并修改它。

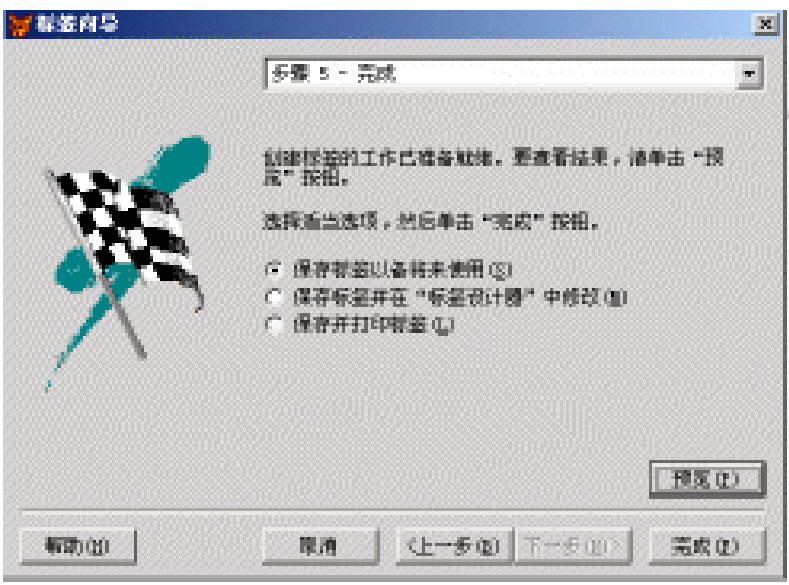


图 10-26　标签向导 6——完成

当向标签中添加各项时,向导窗口中的图片会更新来近似地显示标签的外观。查看这个图片,看选择的字段在您的标签上是否合适。如果文本行过多,则文本行会超出标签的底边。

10.3.2 利用“标签设计器”设计标签

如果不想使用向导来创建标签,您可以使用“标签设计器”来创建布局。“标签设计器”是“报表设计器”的一部分,它们使用相同的菜单和工具栏。两种设计器使用不同的默认页面和纸张。“报表设计器”使用整页标准纸张。“标签设计器”的默认页面和纸张与标准标签的纸张一致。

若要使用“标签设计器”创建标签,其操作步骤如下:

在“项目管理器”窗口中,选定“标签”。

选择“新建”。

选择“新建标签”,显示“新建标签”对话框。

标准标签纸张选项出现在“新建标签”对话框中。

从“新建标签”对话框中,选择标签布局,然后选定“确定”按钮。

“标签设计器”将出现刚选择的标签布局所定义的页面。

可以像处理报表一样给标签指定数据源并插入控件。有关详细内容,请参阅本章稍前所讲的报表的制作,在此不再详述。

菜单和工具栏为用户提供了一个结构化的、可访问的途径，便于使用应用程序中的命令和工具。恰当的计划并设计菜单和工具栏，将使应用程序的主要功能得以体现，用户也不会在使用应用程序时受挫。

用户在查找信息之前，首先看到的便是菜单。如果把菜单设计得很好，那么只要根据菜单的组织形式和内容，用户就可以很好地理解应用程序。为此，Visual FoxPro 提供了“菜单设计器”，可以用来创建菜单，提高应用程序的质量。

§11.1 菜单设计器的使用

11.1.1 创建菜单系统的步骤

创建一个菜单系统包括若干步骤。不管应用程序的规模多大，打算使用的菜单多么复杂，创建菜单系统都需以下步骤：

1. 规划与设计系统

确定需要哪些菜单、出现在界面的何处以及哪几个菜单要有子菜单等等。应用程序的实用性一定程度上取决于菜单系统的质量。花费一定时间规划菜单，有助于用户接受这些菜单，同时也有助于用户对这些菜单的学习。

2. 创建菜单和子菜单

使用菜单设计器可以定义菜单标题、菜单项和子菜单。

3. 按实际要求为菜单系统指定任务

指定菜单所要执行的任务，例如显示表单或对话框等。另外，如果需要，还可以包含初始化代码和清理代码。初始化代码在定义菜单系统之前执行，其中包含的代码用于打开文件，声明变量，或将菜单系统保存到堆栈中，以便以后可以进行恢复。清理代码中包含的代码在菜单定义代码之后执行，用于选择菜单和菜单项可用或不可用。

4. 生成菜单程序

运行生成的程序，以测试菜单系统。

11.1.2 Visual Basic 窗体的菜单界面元素

从图 11-1 中我们可以看到，菜单系统由菜单栏、菜单标题、菜单和菜单项组成。菜单栏出现在窗体的标题栏下面，并包含一个或多个菜单标题。当单击一个菜单标题（如“文件”），包含菜单项目的列表就被拉下来。菜单项可以包括命令（如“新建”和“退出”）、分隔条和子菜单标题。用户看到的每个菜单项和在“菜单编辑器”中定义的一个菜单控件对应（“菜单编辑器”将在本章后面讨论）。

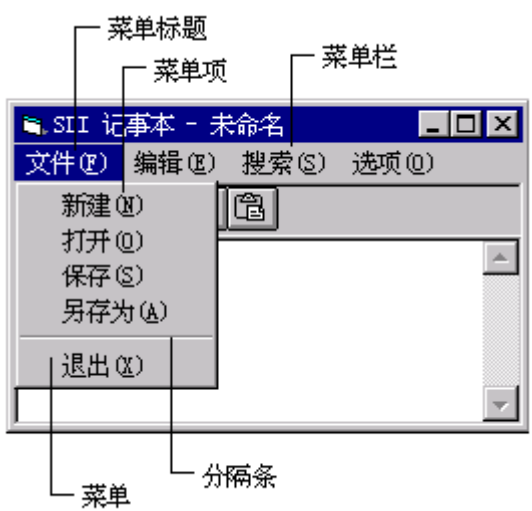


图 11-1 菜单界面元素

11.1.3　创建菜单

若要用“菜单设计器”创建菜单系统，从“项目管理器”中选择“其他”选项卡，再选择“菜单”，然后选择“新建”，出现如图 11-2 所示对话框，选择“菜单”。

图 11-2　新建菜单

此时出现“菜单设计器”，如图 11-3 所示。

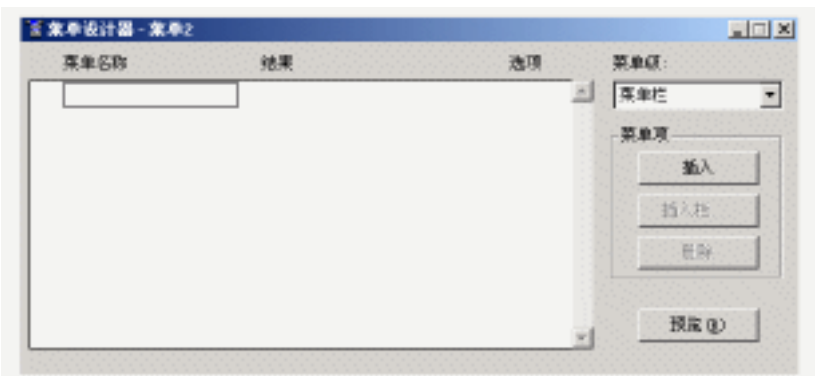


图 11-3　菜单设计器

在“菜单设计器”中有多个选项，各选项说明为：

“菜单名称”：在菜单系统中指定菜单标题和菜单项的名称。

“结果”：指定在选择菜单标题或菜单项时发生的动作，例如，可执行一个命令，打开一个子菜单或运行一个过程。

“创建”：在“结果”项中选择“子菜单”或“过程”时，用于指定菜单标题或菜单项的子菜单或过程。

“编辑”：在创建了“子菜单”或“过程”后，可以更改与菜单标题或菜单项相关的子菜单或过程。

“选项”：显示“提示选项”对话框，可以在其中定义键盘快捷键和其它菜单选择。

“菜单级”：弹出下拉菜单，让用户选择要处理的菜单或子菜单。

“预览”：显示正在创建的菜单的预览结果。

“插入”：在“菜单设计器”窗口中插入新的一行。

“插入栏”：显示“插入系统菜单条”对话框，使用户可以插入标准的 VFP 菜单项。

“删除”：从“菜单设计器”中删除当前菜单行。

利用菜单设计器窗口各个选项就可以设计完整的菜单。

步骤 1　要向菜单中添加主菜单项

在“菜单名称”栏中，选择要添加的菜单项的菜单标题。

在“结果”框中，选定“子菜单”命令，“创建”按钮出现在列表的右侧。

选定“创建”按钮。

出现一个空的设计窗口，在此窗口中，输入菜单项。

在“菜单名称”栏中，依次键入新建的各菜单项的名称，如图 11-4 所示，然后关闭菜单设计器，出现保存对话框，我们以“gzgl. mnx”把它保存在“d：hvfbook”下。

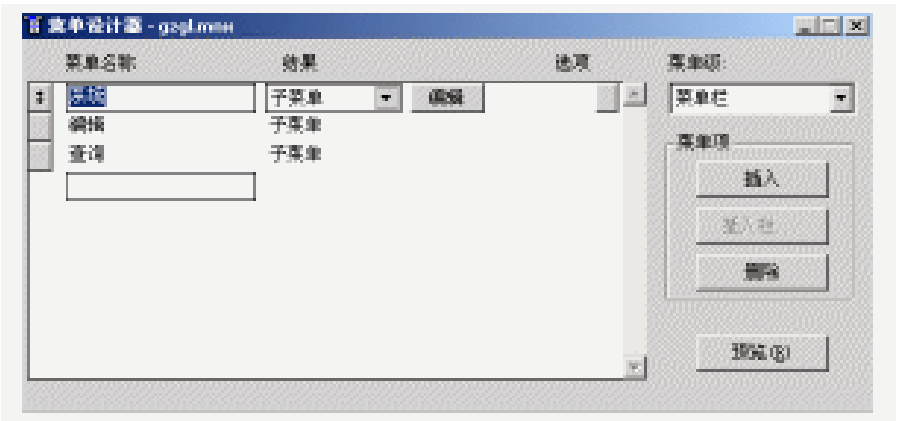


图 11-4　设计主菜单

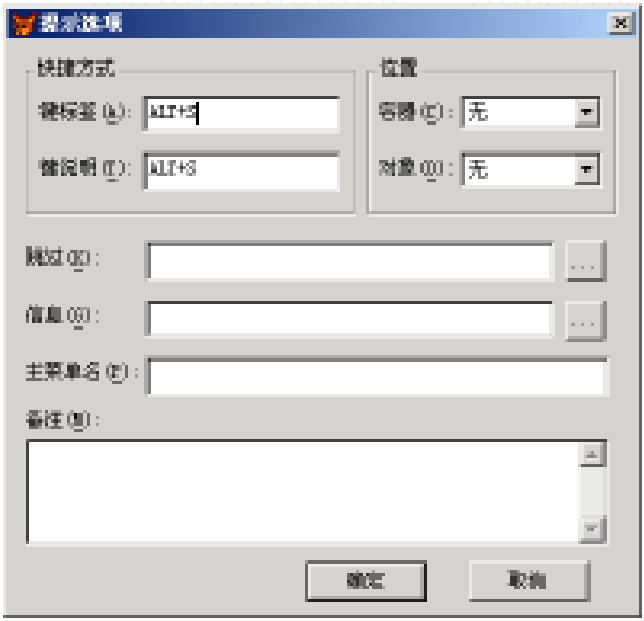


图 11-5　创建快捷方式

除了指定访问键以外，还可以为菜单或菜单项指定键盘快捷键。和使用访问键一样，使用键盘快捷键是让您在按下某个键的同时，再按另一个键而选择菜单或菜单项。访问键与键盘快捷键的区别是：使用快捷键可以在不显示菜单的情况下选择此菜单中的一个菜单项。

Visual FoxPro 菜单项的快捷键一般用 Ctrl 或 Alt 键与另一个键相组合，例如，按 Ctrl + N 可在 Visual FoxPro 中创建新文件（图 11-5）。

若要为菜单或菜单项指定快捷键，请在“菜单名称”栏中，选择相应的菜单标题或菜单项。

选择“选项”栏中的按钮，显示“提示选项”对话框。

在“键标签”框中，按下一组合键，可创建快捷键。

如果一个菜单项没有快捷键，Visual FoxPro 将在“键标签”框中显示“按下要定义的键”。

在“键说明”框中，添加您希望在菜单项的旁边出现的文本。

默认情况下，Visual FoxPro 在“键说明”框中重复“键标签”框的快捷键标记。不过，如果希望在应用程序中显示其他内容，也可以更改“键说明”框中的文字。例如，如果“键标签”和“键说明”均为“Ctrl + R”时，您可以将“键说

明”更改为“ R ”。

注意：Ctrl + J 是无效的快捷键，因为在 Visual FoxPro 中，经常将其作为关闭某些对话框的快捷键。

步骤 2　要向菜单中添加子菜单

对于每个菜单项，都可以创建包含其他菜单项的子菜单。

若要创建子菜单，请在“菜单名称”栏中，选择要添加子菜单的菜单项。

在“结果”框中，选择“子菜单”。

和创建主菜单的方法相同，在“菜单名称”栏中，键入新建的各子菜单的名称，如图 11-6 所示。

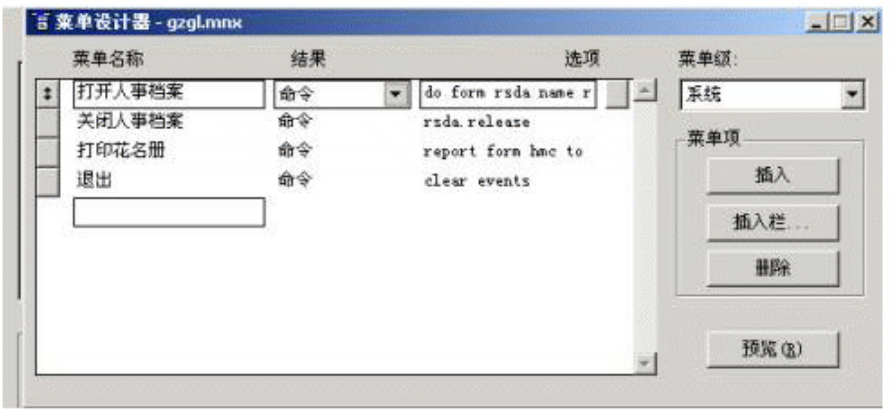


图 11-6　设计子菜单

步骤 3　为菜单或菜单项指定任务

选择一个菜单或菜单项，将执行相应的任务，如显示表单、工具栏或另一个菜单系统等等。要执行任务，菜单或菜单项就必须执行一个 Visual FoxPro 命令。此命令可以是一条语句，也可以是一个过程调用。

如果预计在若干个地方都会使用同样一组命令，则应编写一个过程。该过程必须在菜单清理代码或其他菜单、对象能引用的位置明确命名和编写。

若要为菜单或菜单项指定命令，在“菜单名称”栏中，选择相应的菜单标题或菜单项。

在“结果”框中，选择“命令”。

在“结果”框右侧的框中，键入正确的命令。如要“打印花名册”，我们输入 report form hmc to printer 在“退出”命令中，我们输入“clear events”，如图

11-6 所示。这样,一个菜单系统就设计好了。

在设计菜单系统时,我们应考虑下列准则:

按照用户所要执行的任务组织系统,而不要按应用程序的层次组织系统。

只要查看菜单和菜单项,用户就应该可以对应用程序的组织方法有一个感性认识,因此,要设计好这些菜单和菜单项,您必须清楚用户思考问题的方法和完成任务的方法。

给每个菜单一个有意义的菜单标题。

按照估计的菜单项使用频率、逻辑顺序或字母顺序组织菜单项。

如果不能预计频率,也无法确定逻辑顺序,则可以按字母顺序组织菜单项。当菜单中包含有八个以上的菜单项时,按字母顺序特别有效。太多的菜单项需要用户花费一定的时间才能浏览一遍,而按字母顺序则便于查看菜单项。

在菜单项的逻辑组之间放置分隔线。

将菜单上菜单项的数目限制在一个屏幕之内。

如果菜单项的数目超过了一屏,则应为其中的一些菜单项创建子菜单。

为菜单和菜单项设置访问键或键盘快捷键。

例如,Alt + F 可以作为"文件"菜单的访问键。

使用能够准确描述菜单项的文字。

描述菜单项时,请使用日常用语而不要使用计算机术语;同时,说明选择一个菜单项产生的效果时,应使用简单、生动的动词,而不要将名词当做动词使用;另外,请用相似语句结构说明菜单项。例如,如果对所有的菜单项的描述都使用了同一个词,则这些描述应使用相同的语言结构。

在菜单项中混合使用大小写字母。

步骤 4　运行菜单

菜单设计完成后,选择系统菜单上的"菜单"中的"生成",生成. MPR 文件。我们可以通过编程运行菜单。

我们设计一程序,名字为"cd. prg",程序清单如下:

```
__ screen. caption = "工资管理系统"
set sysmenu on
do gzgl. mpr
```

在命令窗口,我们执行

```
set defa to d :\vfbook
do cd
```

我们可以看到程序运行情况如图 11-7 所示。

图 11-7　菜单运行情况

如果创建的应用程序在网络环境中的多台计算机上运行，或者一个表单的多个实例对相同的数据进行访问，这时就要进行共享访问程序设计。共享访问意味着不仅要为多个用户使用和共享数据提供更有效的方法，并且在必要时要对访问进行限制。

Visual FoxPro 支持如下功能：对数据的共享或独占访问，锁定选项，数据工作期，记录缓冲和表缓冲以及事务处理。尽管这些功能主要用在共享环境里，但在单用户环境下也可以使用。

§12.1 多用户环境中数据访问技术

因为访问数据是在文件内进行，所以有效的数据管理首先从控制这些文件的环境开始，需要选择访问数据的方式，还要选择在什么时间、如何限制对数据的访问。

在共享环境中，可以用两种方式访问数据：从独占文件中访问或者从共享文件中访问。若打开一个共享访问的表，其他用户也能对该文件进行访问；若打开一个独占访问的表，那么其他用户就不能对该文件进行读写操作。独占访问不具备在网络环境中共享数据的许多优点，因而应该避免使用。

12.1.1 以独占访问的方式使用表

打开一个文件最严格的限制方式就是独占方式。通过界面打开一个表时，默认情况下是独占使用的。也可以利用 Visual FoxPro 命令明确地申明以

独占方式打开一个表。

例如要打开一个独占使用的表,在“命令”窗口中键入下列命令:

SET EXCLUSIVE ON

USE cMyTable

或者在“命令”窗口中键入下列命令:

USE cMyTable EXCLUSIVE

下列命令要求以独占方式打开一个表:

ALTER TABLE

INDEX (当创建、添加或删除一个复合索引标识时)

INSERT [BLANK]

MODIFY STRUCTURE (若要用此命令更改一个表结构,必须以独占方式打开该表。当以共享方式打开这个表时,可以,但只能在只读方式下使用此命令。)

PACK

REINDEX

ZAP

如果在一个共享表中执行上述命令,Visual FoxPro 将返回出错信息:文件必须以独占方式打开。”

可以使用 FLOCK () 函数限制对表的访问。如果使用 FLOCK () 锁定表,则其他用户不能对该表进行写操作,但可以读该表。

12.1.2 以共享访问的方式使用表

以共享方式打开一个表时,多个机器可同时访问该表。通过界面打开表时,可以不理会 SET EXCLUSIVE 默认的 ON 设置,而明确地用 Visual FoxPro 命令打开一个表供共享使用。

若要打开一个共享使用的表,在“命令”窗口中键入下列命令:

SET EXCLUSIVE OFF

USE cMyTable

或者在“命令”窗口中键入下列命令。

USE cMyTable SHARED

当在一个共享表中添加或更改数据时,必须首先锁定要更改的记录或整个表。对一个以共享方式打开的表或记录,可通过下列方式进行锁定:

使用命令来自动锁定记录或表。

用记录和表的锁定函数,人工锁定一个或多个记录。

用 CURSORSETPROP ( ) 函数初始化缓冲。

与共享表相关的备注和索引文件也是以相同的共享方式打开的。

如果应用程序所使用的表只用于查找数据,而且该应用程序的所有用户都要访问它,那么将该表标记为只读可以提高性能。

12.1.3　锁定数据

如果要共享访问文件,必须通过锁定表和记录来对这种访问进行管理。锁定不同于访问权限,它既可以对数据进行长期控制,也可以短期控制。Visual FoxPro 提供自动和人工两种锁定方式。

无论是自动记录锁定还是人工记录锁定,目的都是为了防止两个用户同时写一个记录。表锁定用来防止其他用户在表中进行写入操作,但允许读取整个表。由于表锁定阻止其他用户更新表中记录,因而很少使用。

除了选择记录锁定或者表锁定,还可以选择自动锁定或者人工锁定。

1. 自动锁定记录和表的命令

许多 Visual FoxPro 命令如表 12-1 在执行之前都会自动锁定一个记录或一个表,如果成功锁定了记录或表,则执行该命令,然后再解锁。

表 12-1

命　令	锁定范围	命　令	锁定范围
ALTER TABLE	整个表	APPEND	表头
APPEND BLANK	表头	CURSORSETPROP 0	取决于参数
APPEND FROM	表头	APPEND FROM ARRAY	表头
APPEND MEMO	当前记录	BLANK	当前记录
DELETE NEXT 1	当前记录	DELETE RECORD n	记录 n
删除多个记录的 DELETE	整个表	DELETE - SQL	当前记录
GATHER	当前记录	INSERT	整个表
INSERTSQL	表头	MODIFYMEMO	编辑开始时锁定当前记录
READ	当前记录和别名字段的所有记录	RECALL	当前记录
RECALL NEXT 1	当前记录	RECALLRECORD n	记录 n

续 表

命 令	锁定范围	命 令	锁定范围
恢复多个记录的 RECALL	整个表	REPLACE	当前记录和别名字段的所有记录
REPLACE NEXT 1	当前记录和别名字段的所有记录	REPLACE RECORD n	记录 n 和别名字段的所有记录
REPLACE 替换多个记录	整个表和别名字段的所有记录	SHOW GETS	当前记录和别名字段引用的所有记录
TABLEUPDATE ()	取决于缓冲	UPDATE	整个表
BROW、CHANGE 和 EDIT	一旦开始编辑字段,锁定对象为当前记录和相关表中别名字段的所有记录		

2. 人工锁定

可以用锁定函数人工锁定一个记录或一个表。

若要人工锁定一个记录或一个表,请使用下列命令:

RLOCK ()

LOCK ()

FLOCK ()

RLOCK () 函数等同于 LOCK () 函数,都可以锁定一个或多个记录;FLOCK () 锁定一个文件。LOCK () 和 RLOCK () 函数可以用于锁定表头。如果把 0 作为记录编号提供给 LOCK () 或 RLOCK (),而且测试表明表头未锁定,则该函数将锁定表头并返回“真”(.T.)。

如果锁定了一个记录或一个表,那么一定要尽快解锁,以方便其他用户访问。解锁可使用 UNLOCK 命令。

这些人工锁定函数可以完成如下操作:

测试记录或表的锁定状态。

如果测试表明该记录未锁定,则锁定该记录或表并返回“真”(.T.)。

如果该记录或表不能锁定,则根据 SET REPROCESS 的当前设置,再次进行锁定。

返回“真”(.T.)或“假”(.F.),表明锁定尝试是否成功。

提示:如果不想锁定记录,而测试该记录的锁定状态,请使用 ISRLOCKED () 或 ISFLOCKED () 函数。

如果锁定记录或表的操作失败,SET REPROCESS 命令和当前错误处理例程将确定是否再次尝试锁定。SET REPROCESS 会影响不成功的锁定结果,因

此可以使用 SET REPROCESS 来控制锁定尝试的次数以及时间。

3. 解锁数据

在共享环境下锁定记录或文件并完成了相应的数据操作之后，应及时解锁。有几种解锁的办法。有时，只需简单地移动到下一个记录就能解锁，其他情况则需要明确的命令。

要解锁被自动锁定的记录，只需移动记录指针，甚至在设置 MULTILOCKS ON 的情况下也是如此。而对于人工锁定的记录，则必须明确地对记录解锁，仅仅移动记录指针是不够的。

表 12-2 说明了对人工和自动的记录锁定和表锁定进行解锁的各命令：

表 12-2

命　　令	效　　果
UNLOCK	解锁当前工作区内的记录和文件
UNLOCK ALL	对当前工作期内所有的工作区解锁
SET MULTI LOCKS OFF	建立新锁定的同时自动解锁当前锁定
FLOCK()	在锁定文件之前解锁文件中的所有记录
CLEAR ALL CLOSE ALL USE QUIT	对所有记录和文件解锁
END TRANSACTION	对所有的自动锁定项解锁
TABLEUPDATE()	在更新表之前解锁所有锁定项

注意：如果记录在 UDF 中被自动锁定，那么当移开记录指针然后又移回该记录时，锁定将被解除，可以使用表缓冲避免这个问题。

§12.2　数据更新技术

可以使用缓冲、事务处理或视图更新数据。

12.2.1　使用缓冲进行更新

使用缓冲进行更新，在选择缓冲方法和锁定类型之后，就可启用记录或表缓冲。

若要启用缓冲，请选择下列选项之一：

在“表单设计器”中，设置表单数据环境中临时表的 BufferModeOverride 属性。

或者在代码中设置 Buffering 属性。

例如,将下列代码放进表单的 Init 过程中,可以启用保守式行缓冲:

CURSORSETPROP (Buffering ,2)

然后将进行更新操作的代码放在控件适当的方法程序代码中。

要把编辑结果写入原来的表,可以使用 TABLEUPDATE ( ) 。由于规则限制,造成对表的更新操作失败之后,若要取消编辑结果,可以使用 TABLEREVERT ( ) 命令。TABLEREVERT ( ) 即使在没有明确启用表缓冲的情况也是有效的。

12.2.2 使用事务处理管理更新

即便使用缓冲,事情有时也会出错。如果希望保护更新操作,并从作为一个单位的整段代码中还原回来,可以使用事务处理。

在应用程序中添加事务处理所提供的保护机制超过了记录缓冲和表缓冲提供的保护功能,它将整段代码作为一个受保护的、可恢复的单元;可以嵌套使用事务处理,并用它保护已操作的缓冲更新。Visual FoxPro 事务处理只能用于数据库中的表和视图。

1. 包装代码段

事务处理类似一层外包装,用于高速缓存对内存或硬盘的数据更新操作,而不直接对数据库进行更新。实际的数据库更新在事务处理结束之后进行。如果由于某种原因,系统不能执行对数据库的更新操作,就可以回滚整个事务处理,而不执行任何更新。

注意:在同一数据工作期的事务处理中,事务处理外面的缓冲更新操作将被忽略。

2. 控制事务处理的命令

Visual FoxPro 提供了三个命令一个函数来控制事务处理 (表 12-3):

表 12-3

操 作 项 目	指 令
初始化一个事务处理	BEGIN TRANSACTION
确定当前事务处理的等级	TXNLEVEL 0
取消最近一条 BEGIN TRANSACTION 语句以来所做的全部修改	ROLLBACK
锁定记录,提交最近一条 BEGIN TRANSACTION 语句以来对数据库中表所做的全部修改,然后解锁记录	ENDTRANSACTION

可以用事务处理缓冲对以下各项的修改：表、结构化的 .cdx 文件以及与数据库内表相关的备注文件。涉及内存变量和其他对象的操作不属于事务处理，因此不能回滚或提交这些操作。

注意：使用远程表中存储的数据时，事务处理命令控件只更新视图临时表的本地副本中的数据，对远程基表的更新不起作用。若要对远程表启用人工事务处理，请使用 SQLSETPROP ( )，然后用 SQLCOMMIT ( ) 和 SQLROLLBACK ( ) 控制事务处理。

一般来讲，除非缓冲 TABLEUPDATE ( ) 调用，事务处理最好和记录缓冲一起使用，而不与表缓冲一起使用。如果在事务处理中使用了 TABLEUPDATE ( ) 命令，那么可以回滚一个失败的更新操作，找到失败的原因，然后重试 TABLEUPDATE ( ) 命令而不丢失任何数据。这样，就确保了更新操作一定是所谓的“要么全都做，要么全不做”的操作。

尽管在正常情况下，简单的事务处理过程能够提供安全的数据更新操作，但是它并不能提供对系统失败的完全保护。如果在处理 END TRANSACTION 命令期间断电或产生其他系统中断，则数据更新仍将失败。

请使用下列事务处理代码模板：

```
BEGIN TRANSACTION
* 更新数据
IF lSuccess = .F. && 出错
    ROLLBACK
ELSE && 执行修改
    * 确认数据
    IF && 出错
        ROLLBACK
    ELSE
        END TRANSACTION
    ENDIF
ENDIF
```

3. 使用事务处理

下列规则适用于事务处理：

一个事务处理起始于 BEGIN TRANSACTION 命令，以 END TRANSACTION 或 ROLLBACK 命令终止。只有 END TRANSACTION 语句，而前面没有 BEGIN TRANSACTION 与之匹配则会出错。

ROLLBACK 语句前没有 BEGIN TRANSACTION 将出错。

除非应用程序中止 (这将导致回滚操作),事务处理一旦开始,直到遇到相应的 END TRANSACTION 语句 (或回滚语句) 这一期间,始终保持有效,在多个程序、函数之间切换的情况下也是如此。

对于涉及事务处理数据的查询,Visual FoxPro 在使用磁盘数据前,首先使用在事务处理缓冲区内的高速缓冲数据,确保使用的是最近的数据。

如果在事务处理过程中应用程序中止,则所有操作回滚。

事务处理只在数据库容器内进行。

如果 INDEX 命令改写了一个已有的索引文件,或者有任一索引文件已打开,则不能使用 INDEX 命令。

事务处理只在数据工作期中起作用。

事务处理完成了下列锁定动作:

当一个命令直接或间接调用事务处理时,Visual FoxPro 将强制执行锁定动作。任何系统或用户的直接或间接命令将放入被高速缓存,直至 ROLLBACK 或 END TRANSACTION 命令结束事务处理操作。

如果在事务处理中使用了锁定命令,如 FLOCK () 或 RLOCK (),则 END TRANSACTION 语句将不会解锁,因此必须明确地解除任何在事务处理中进行的明确锁定。应该使包含 FLOCK () 或 RLOCK () 命令的事务处理尽量简短 否则,用户会长时间不能访问被锁定的记录。

4. 嵌套事务处理

对于分隔在不同并发进程中的表,嵌套事务处理可以为表的更新操作提供逻辑组。事务处理命令 BEGIN TRANSACTION . . . END TRANSACTION 不需要在同一函数或过程中。下列规则适用于嵌套事务处理:

可以嵌套五层 BEGIN TRANSACTION. . . END TRANSACTION。

直到最外层的 END TRANSACTION 被调用时,才提交嵌套事务处理中的更新。

在嵌套事务处理中,一个 END TRANSACTION 只对最近的 BEGIN TRANSACTION 所引起的事务处理进行操作。

在嵌套事务处理中,ROLLBACK 语句只对最近 BEGIN TRANSACTION 所引起的事务处理进行操作。

5. 保护远程更新

在对远程表进行数据更新的过程中,事务处理可以避免系统造成的错误。

12.2.3 使用视图管理更新

可以使用 Visual FoxPro 视图具有的更新冲突管理技术来管理多用户对数

据的访问。通过使用 WhereType 属性,视图控制发送到基于视图的基表的内容。可以为本地和远程视图设置该属性。WhereType 属性提供了以下四种设置:

DB_KEY

DB_KEYANDUPDATABLE

DB_KEYANDMODIFIED (默认)

DB_KEYANDTIMESTAMP

通过选择四种设置之一,可控制 Visual FoxPro 如何生成发送到视图基表的 SQL Update 语句的 WHERE 子句。可以使用"视图设计器"的"更新条件"选项卡来选择需要的设置,或者使用 DBSETPROP ( ) 函数设置一个视图定义的 WhereType 属性。要想更改一个活动视图临时表的 WhereType 设置,请使用 CURSORSETPROP ( )。

在嵌套事务处理中对同一数据的更新操作,最内层的更新优先于同一嵌套事务处理块中的其他更新。

如果在事务处理中使用了锁定命令,如 FLOCK ( ) 或 RLOCK ( ),则 END TRANSACTION 语句将不会解锁。因此必须明确地解除任何在事务处理中进行的明确锁定。应该使包含 FLOCK ( ) 或 RLOCK ( ) 命令的事务处理尽量简短 否则,用户会长时间不能访问被锁定的记录。

§12.3　对访问冲突的处理

无论选择了缓冲、事务处理还是视图,都必须在更新过程中管理冲突。

12.3.1　管理缓冲冲突

在一个多用户环境中,通过精心选择打开、缓冲并锁定数据的时间和方式,可以更高效地进行数据更新操作。应该减少访问记录或表时发生冲突的时间,同时必须预测到不可避免的冲突将导致什么样的后果,并对这种冲突进行管理。冲突一般在一个用户试图锁定一个当前正被其他用户锁定的记录或表时发生,两个用户不能同时锁定同一记录或表。

应用程序中应包含管理冲突的例程。如果没有冲突例程,系统将死锁。死锁通常在这种情况下发生:第一个用户已经锁定一个记录或表,现在试图去锁定已被第二个用户锁定的记录,而第二个用户又反过来试图锁定第一个用户锁定的记录。尽管这种情况很少发生,但记录或表被锁定的时间越长,这种死锁的可能性就越大。

1. 出错处理例程

无论是设计一个多用户的应用程序还是为一个单用户系统添加网络支持服务,都要求找到错误并对其进行处理。使用 Visual FoxPro 记录缓冲和表缓冲可以部分简化这种工作。

如果试图锁定一个已被其他用户锁定的记录或表,Visual FoxPro 将返回出错信息。可使用 SET REPROCESS 自动处理不成功的锁定操作。此命令与 ON ERROR 例程和 RETRY 命令组合,可以继续或取消锁定的操作。

2. 检测并解决冲突

在数据更新操作过程中,特别是在共享环境下,您可能希望确定哪些字段已经更改,确定已更改字段的原有值或当前值是什么。Visual FoxPro 的缓冲和 GETFLDSTATE ()、GETNEXTMODIFIED ()、OLDVAL () 及 CURVAL () 函数可以提供这些功能。通过它们来确定哪些字段已经更改,查找已更改数据,比较当前值、原有值及已编辑的值,这样,可以决定如何处理错误或冲突。

⑴ 若要检测字段中所做的更改,在更新操作之后,使用 GETFLDSTATE () 函数。

GETFLDSTATE () 可以对非缓冲数据进行操作,但在启用了记录缓冲的情况下,此函数更有效。例如,将 GETFLDSTATE () 用于一个表单的 Skip 按钮代码中。移动记录指针时,Visual FoxPro 将检查记录中所有字段的状态,如下例所示:

```
lModified = .F.
FOR nFieldNum = 1 TO FCOUNT() && 检查所有字段
    if GETFLDSTATE(nFieldNum) = 2 && 已修改
        lModified = .T.
        EXIT && 可以在此处插入一个
    ENDIF && 请参阅下一个示例
ENDFOR
```

⑵ 若要检测并定位缓冲数据中已更改的记录,请使用 GETNEXTMODIFIED () 函数。

GETNEXTMODIFIED () 以 0 作为参数,查找第一个已修改的记录。如果其他用户对缓冲表进行了修改,则缓冲区中 TABLEUPDATE () 命令遇到任何修改都将导致冲突。可以用 CURVAL ()、OLDVAL () 和 MESSAGEBOX () 计算冲突值并解决冲突,CURVAL () 返回磁盘上记录的当前值,而 OLDVAL () 返回记录位于缓存内的值。

⑶ 若要确定缓冲字段的原有值,请使用 OLDVAL ( ) 函数。

OLDVAL ( ) 返回一个缓冲字段的值。

⑷ 若要确定磁盘上一个缓冲字段的当前值,请使用 CURVAL ( ) 函数。

CURVAL ( ) 返回一个在编辑操作完成前缓冲字段在磁盘中的当前值。

在共享环境中,可以创建一个出错处理过程,比较当前值与原有值,并决定是接受当前的修改还是更早一些的修改。

3. 使用备注字段检查冲突

使用 CompareMemo 属性来控制何时使用备注字段检查更新冲突。这个视图和临时表属性确定了在更新的 WHERE 子句中是否包含备注字段 (备注或通用型) 。默认设置是“真”(. T.) ,即在 WHERE 子句中包含备注字段。如果这个属性设置为“假”(. F) ,不论 UpdateType 如何设置,在更新的 WHERE 子句中都不会包含备注字段。

当 CompareMemo 属性设置为假时,备注字段上的开放式冲突检查是不可用的。如要使用备注字段的冲突检查,请将 CompareMemo 属性设置为真。

12.3.2　管理冲突的规则

管理在多用户环境中的冲突需要有扩展和可重用的代码。一个完整的冲突管理例程进行如下操作:

⑴ 检查冲突。

⑵ 确定冲突的性质和位置。

⑶ 提供足够的信息,用户可以有效地解决这些冲突。

有关冲突管理例程的示例,请参阅位于 Visual Studio… hSamples hVfp98 h Classes 目录下的 Samples. vcx 中的 datachecker 类。将该类添加到表单中,在将缓冲数据写入表的操作之前调用 CheckConflicts 方法程序,例如使用行缓冲移动记录指针,关闭表,或者发布 TABLEUPDATE ( ) 函数。