21世纪大学计算机专业教材

陕西省普通高等学校精品课程建设项目

Visual FoxPro 9.0 程序设计教程

主　编　谢膺白

副主编　桑国珍　奚建荣

西安交通大学出版社

XI'AN JIAOTONG UNIVERSITY PRESS

陕西省普通高等学校精品课程建设项目
“数据库原理及应用”计划教材

Visual FoxPro 9.0
程序设计教程

谢膺白 （主编）
桑国珍 奚建荣 （副主编）

西安交通大学出版社

内容简介

本书是2006年陕西省精品课程建设项目《数据库原理及应用》的计划教材。它根据教育部高等学校计算机科学与技术教学指导委员会2006年正式发布的《关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求(试行)》(白皮书)的精神、2004年全国计算机等级考试二级数据库考试大纲要求,结合目前我国高等院校计算机课程开设的实际情况,融汇作者多年从事数据库教学和数据库程序设计的实践经验而编写。全书以Visual FoxPro 9.0为平台,介绍了数据库的基本知识、基本操作、SQL语言、结构化程序设计、面向对象的程序设计方法。

全书内容丰富,讲述深入浅出,突出了系统性和实践性、加强了SQL语言和面向对象程序设计方面的内容。书中的所有例题和各章后习题中的程序设计习题全部在奔腾微机上调试通过,结果正确。它适合具有计算机文化基础的读者,是一本很好的高等院校各专业学习数据库程序设计用教材,也可用于具有中等以上文化程度及一定英语基础的读者自学。对于企业、事业单位的管理人员、广大从事数据库应用程序开发的技术人员亦有很好的参考价值。

图书在版编目(CIP)数据

Visual FoxPro 9.0程序设计教程/谢膺白主编;桑国珍等编著. —西安:西安交通大学出版社,2007.8
ISBN 978-7-5605-2478-8

Ⅰ.V… Ⅱ.①谢…②桑… Ⅲ.关系数据库-数据库管理系统,Visual FoxPro 9.0-高等学校-教材
Ⅳ.TP311.138

中国版本图书馆CIP数据核字(2007)第079322号

书　　名　Visual FoxPro 9.0程序设计教程
主　　编　谢膺白
出版发行　西安交通大学出版社
地　　址　西安市兴庆南路10号(邮编:710049)
电　　话　(029)82668357　82667874(发行部)
　　(029)82668315　82669096(总编办)
网　　址　http://press.xjtu.edu.cn
电子邮箱　eibooks@163.com
印　　刷　陕西宝石兰印务有限责任公司
字　　数　607千字
开　　本　787mm×1092mm　1/16
印　　张　25.125
版　　次　2007年8月第1版　2007年8月第1次印刷
书　　号　ISBN 978-7-5605-2478-8/TP·495
定　　价　32.00元

主编简介

谢膺白，男，汉族，中共党员，陕西省华县人，生于 1947 年 10 月，1982 年元月毕业于西安交通大学工程力学系力学师资 71 班。渭南师范学院计算机科学系教授，学院十大教学名师之一，陕西省计算机教育学会常务理事。原渭南师范学院计算机科学系主任，现任渭南师范学院陕西省精品课程建设项目《数据库原理及应用》负责人，课程网址：http://jpkc.wntc.edu.cn/ec/c15。

曾于 1987 年在美国科罗拉多州丹佛市的世界著名石油软件公司：SSI(Scientific Software Incorporation)公司学习稠油热力采油数值模拟软件的开发。先后参加过《“七五”国家重点科技攻关项目——克拉玛依油田九区蒸汽驱开采技术研究》、《“八五”国家重点科技攻关项目——克拉玛依油田化学驱提高原油采收率技术研究》。他治学严谨，平易近人，热爱学生，任教以来，先后讲授过计算机专业课程 10 余门，撰写发表科研论文 8 篇。参编、主编出版了计算机方面的教材 8 部，其中数据库方面的教材 4 部，他任主编的由西安交大出版社出版的《Visual FoxPro 及其程序设计教程》获 2004 年陕西省信息产业厅、陕西省计算机教育学会优秀教材二等奖。从事数据库方面的教学与研究，成果颇丰。多次获学院优秀教学成果奖，多次被评为学院优秀教师、优秀共产党员，2006 年获学院首批教学名师称号。

前　言

在数据库应用技术领域中，Visual FoxPro 9.0 是适用于微型计算机系统最优秀的小型关系型数据库管理系统之一。

多年来 Fox 系列的数据库管理系统一直是我国高校数据库程序设计的主流内容，也一直是我国计算机等级考试的必考内容，用它开发的应用程序在社会上最多最流行。而 Visual FoxPro 9.0 不但与 dBASE 及 Fox 系列的数据库管理系统具有良好的兼容性，而且它对关系型数据库标准 SQL 及互联网络的支持进一步增强。它因具有集成化的系统开发环境；既支持面向过程的编程技术又支持面向对象的编程技术；智能帮助易学习好操作而深受用户特别是高校师生和广大数据库工作者的青睐。

本书是 2006 年陕西省精品课程建设项目《数据库原理及应用》中的计划教材（课程网址：http://jpkc.wntc.edu.cn/ec/c15），它是根据教育部高等学校计算机科学与技术教学指导委员会发布的《关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求（试行）》（白皮书）的精神，结合目前我国高等院校计算机课程开设的实际情况，融作者多年从事数据库教学和数据库程序设计的实践经验而编写。其中许多章节及全部命令直接从 Visual FoxPro 9.0 自带的英文帮助文件中翻译而来。全书共 15 章，按照循序渐进、由浅入深的原则，以 Visual FoxPro 9.0 数据库管理系统为平台，介绍了数据库的基本知识和基本操作、SQL 语言、结构化和面向对象的程序设计方法。按照白皮书的要求，以加强实训培养应用型人才为出发点组织教材内容，本书的最大特点一是加大了 SQL 和面向对象程序设计部分的内容；二是每一章都给出了一定数量、难度和较高质量的例题习题，最后的第 15 章则是一个完整的实际例子供读者开发自己的应用系统参考。稍后还将出版与之配套的实验指导书及电子教案。

本书既可作为高等学校本专科数据库程序设计的教材，又可用作各类计算机应用培训班的教材或教学参考用书，也适用于具有中等以上文化程度的读者自学，对于企业、事业单位的管理人员、广大从事数据库应用程序开发的技术人员亦有很好的参考价值。

本书由桑国珍、奚建荣同志任副主编，马君、高升宇、谢稷光、阴国富、王玲、刘静、李云飞、王敏、索红军、花妮娜同志参加了教材编写工作。书中的所有例题及各章后的习题中的编程题，作者们全部在奔腾微机上调试通过，测试正确无误。

考虑到教材的完整性，本书的内容较以往的同类教材较多，在目录中我们对有关的章节用 * 号作了标记，授课时，教师可根据学时的多少情况，适当予以取舍。

本书在编写过程中得到了西安交通大学出版社的大力支持与帮助，在此，表示深深的谢意。

由于编著者水平有限，时间比较仓促，可能存在有错误和不妥之处，敬请读者谅解，批评指正。请读者将意见及时反馈给我们，以便我们进行修改。

主编　谢膺白 xieyingbai@126.com

2007.4

目　录

第 1 章　数据库基本概念

计算机技术的发展，特别是微型计算机在全球的普及，使得计算机应用领域得到了前所未有的扩展。逐渐地从单一的用于军事及科学目的的数值计算，发展到了当今的数值计算、自动控制、信息处理、测量和测试、教育和卫生、家用电器、人工智能等各个领域。它以不可抗拒的潮流渗透到了社会的政治、经济、文化生活的各个角落。很难想象现代社会离开了计算机，人们将如何生活、学习、工作和娱乐。

目前在计算机应用领域，信息处理占有越来越重要的地位。有人讲：从确切意义上说，计算机应当称为信息机，或者叫信息处理机，此话不无道理。信息处理主要包括两个方面：商务处理和管理应用。而管理应用则更加普遍，它能大幅度提高各级政府的工作效率，给商家带来勃勃商机，给企业带来巨大的经济效益，更能使事业单位产生不可估量的社会效益。计算机文化的概念已深入人心，本世纪凡不懂计算机、不会使用计算机者将被视为“文盲”。

管理是以数据库技术为基础的，如今它已成为了现代计算机科学一个新兴、重要的分支。

1.1　数据和数据库系统

在数据库系统中，人们首先遇到的基本概念是什么是数据？数据从何而来？它和人们常说的信息有何关系？本节将回答这些最基本的问题。

1.1.1　数据、信息、信息处理

数据(data)是存储在某种介质上能够被识别的物理符号。国际标准化组织(ISO)对数据给出了更为严格的定义“数据是对事实、概念或指令的一种特殊表达形式，这种特殊表达形式可以用人工的方式或用自动化的装置进行通信，翻译转换或者进行加工处理。”在计算机技术中，把能被输入计算机，并能被计算机所存储、处理、传输的符号统称为数据。它可以是通常情况下人们所熟悉的数值型数据，也可以是被数字化后的非数值型数据，例如：声音、图像等。

信息(information)是构成一定含义的一组数据。信息论的奠基人维纳曾经说过：“信息就是信息，不是物质，也不是能量。“信息”是人们在适应外部世界并且使之反作用于外部世界的过程中，同外部世界进行交换内容的名称。”可见，信息既是客观事物的特征、事物运动变化的反映，又是事物之间相互作用相互联系的反映。

信息处理(information process)也称为数据处理，它是指将数据转换成信息的过程。从数据处理的角度而言，信息是一种被加工成特定形式的数据，这种数据对于数据接收者来说是有重要意义的。

由此可见，信息和数据的关系是数据是信息的载体，信息是数据处理的结果。数据是重要

的,而将数据处理后得到的有用信息则更珍贵,对信息的筛选可以产生决策,从而为决策者的决策提供重要依据。

1.1.2 数据库系统

在数据库中,经常会遇到数据库、数据库管理系统、数据库应用系统、数据库系统等概念。本节将予以介绍。

1. 数据库

数据库(database)是存放数据的"仓库"。这个仓库"修建"在计算机的磁盘上。人们把数据按一定的结构,以文件的形式存放在磁盘上,这种特殊的磁盘文件称之为数据库文件,简称为数据库。它具有:数据和程序可相互独立、数据可以共享、数据冗余度小、便于管理和检索、可随时修改数据、存储结构等特点。

2. 数据库管理系统

数据库的创建、管理、使用、维护等,都需由一种称为数据库管理系统(DBMS——database management system)的软件来完成。DBMS 最基本的功能有三个。

(1)数据定义(data definition)

DBMS 所提供的数据定义语言(DDL:data definition language)用于方便地定义数据库中数据的逻辑结构。

(2)数据操纵(data manipulation)

DMBS 所提供的数据操纵语言(DML:data manipulation language)用于实现对数据库的各种操作。如数据的插入、查询、修改、删除等。

(3)数据控制(data control)

DMBS 提供数据控制语言(data control language),可实现对数据的各种控制。例如,访问控制(access control)、并发控制(concurrency control)等。

同时,DMBS 还要能够提供完整性约束检查(integrity constraint check)、数据库恢复(database recovery)等功能。

不同的数据库管理系统所形成的数据库,结构也不一定相同。常用的数据库管理系统主要有三种结构模型。即层次模型(hierarchical model)、网状模型(network model)、关系模型(relational model)。例如 Visual FoxPro 9.0 就是广泛被应用于微型计算机上的一个关系型的数据库管理系统。

3. 数据库应用系统

数据库应用系统(DBUS——database utility system)指数据库开发人员使用数据库管理系统,所开发的解决实际问题的应用软件。例如利用 Visual FoxPro 9.0 开发的一个"学生学籍管理系统",就是一个数据库应用系统。

4. 数据库系统

数据库系统(DBS——database system)是指引进了数据技术的计算机系统,实现有组织地、动态地存储大量相关数据,提供数据处理和信息资源共享的便利手段。由此可见,数据库系统应包括:计算机硬件、操作系统、数据库管理系统及其他软件、数据库、数据库管理员、用户等六大部分组成。

1.2　数据库理论中的三个世界

在数据库理论中，经常会提到“三个世界”，它们分别是：现实世界、信息世界、数据世界。

1.2.1　现实世界

现实世界是指客观存在的事物。每种事物都有自己的特性，事物与事物之间也存在着错综复杂的联系。计算机系统是不能直接处理现实世界的，现实世界只有数字化后，才能被计算机系统来处理。

1.2.2　信息世界

信息世界是现实世界在人脑中的反映。现实世界中的事物和事物特性在信息世界中分别被抽象为实体和实体的属性，而现实世界间的联系则被抽象为联系。这些抽象所产生的模型，称为概念模型，通常对于概念模型的描述是使用实体-关系图(E-R 图)来实现的。概念模型独立于具体的计算机系统和数据库管理系统。

1.2.3　数据世界

数据世界是信息世界数据化后的产物，即概念模型的数据化实现。在数据世界中，信息世界的实体被数据化为记录，信息世界的实体属性被数字化为数据项，而实体间的联系反映为记录间的联系。由于数据世界中数据模型与所选用的计算机系统及数据库管理系统密切相关，因此数据世界也被称为机器世界。

三个世界之间的关系如图 1-1 所示。从图中可以看出，信息世界的概念模型是不依赖于具体的计算机世界的。概念模型是从现实世界到机器世界的中间层次。现实世界只有先抽象为信息世界，才能进一步转化为数据世界。

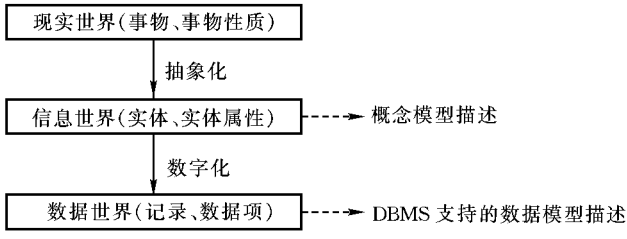


图 1-1　数据三个世界的层次关系

1.3　数据模型

1.3.1　实体的描述

由现实世界抽象到信息世界后，事物被抽象成了实体，事物的特性被抽象成了属性，事物间的联系被抽象为关系。

1. 实体(entity)

客观事物在信息世界中的反映。它既可以是实际存在的事物,也可以是某种概念。实体必须能够相互区别。例如:图书、读者是实际存在的事物,而看一次演出,听一场报告则是比较抽象的一个事件,但从数据库理论的角度看,它们都是实体。

2. 属性(attribute)

事物的特性在实体上的反映称为实体的属性。例如,学生实体可以用学生的(学号、姓名、性别、出生日期、班级)等属性来描述。属性有属性名和属性值之分,例如对于学生实体而言,姓名是它的一个属性名,而"张三"、"李四"等则是不同实体的属性值。不同的实体正是由它们不同的属性及属性值来区别的。

3. 域(domain)

任何实体的任何属性在取值上都是有一定的限制的,这种属性的取值范围就称为属性的域。例如,学号的域可选为长度为八个的字符串,性别的域为"男"、"女"等。

4. 实体集(entity set)

所有属性名完全相同的实体的集合。如全体学生就组成一个学生实体集。为了区分实体集,每个实体集都应命名一个名字,即实体名。例如,学生实体指的是名字为学生的实体集,而(10001001,张三,男,01/28/80,计算机 02)则是学生实体集中的一个实体。在一个实体集中不允许有两个完全相同的实体出现。

5. 实体型(entity type)

实体集的名称及其所有属性名的集合,称为实体型。例如,学生(学号,姓名,性别,出生日期,班级)就是学生实体集的实体型。如同大家所熟悉的整型、实型、双精度等数据类型一样,实体型也是一类数据类型,它抽象地描述了所有同集实体。在不引起混淆的情况下,实体型往往被简称为实体。

在 Visual FoxPro 中,表用来存放实体集。一个表包含若干个字段和若干个记录,每一个字段就是实体的一个属性,而每一个记录就表示一个实体。

6. 码(key)

在同一个实体集中不允许有两个或两个以上的实体,在各对应属性上的属性值都相同。在一个实体集中,根据一个或几个属性的值可惟一地确定每一个实体,则此属性或属性组被称为该实体集的码或关键字。例如学生实体集中的学号,就是他的码。

1.3.2 概念模型中实体间的联系(relationship)

现实世界中,事物之间存在着错综复杂的联系。反应在概念模型中,则有了实体集内部的联系和实体集之间的联系。实际上联系也是一种实体,但当概念模型确定之后,它只能作为联系而存在了。联系也有联系名,而联系的属性则大部分隐藏在发生联系的各实体之中。

两个实体之间的联系可归纳为三类。

1. 一对一(1:1)联系(one-to-one relationship)

设有两个实体集 A 和 B,若对于某个联系 K 而言,如果 A 中的每一个实体至多与 B 中的一个实体相联系,反之亦然,则称 A 与 B 对于联系 K 来说,是一对一的联系。例如学生实体集

和学生成绩实体集之间相对于学号的联系就是一个 1:1 的联系。

2. 一对多(1:m)联系和多对一(m:1)联系(one-to many relationship)

若对于某个联系 K 而言,如果 A 中的每一个实体,可以由 B 中的多个实体相联系,但 B 中的每一个实体,A 中至多有一个实体与之联系,则称 A 对 B 相对于联系 K 来说,是一对多的联系,而 B 对于 A 则是多对一的联系。例如学生实体集和学生图书借阅证实体集之间相对于学号的联系就是一个 1:m 的联系。

3. 多对多(m:n)联系(Many-to-many relationship)

若对于某个联系 K 而言,如果 A 中的每一个实体,可以由 B 中的多个实体相联系,反之亦然,则称 A 对 B 相对于联系 K 来说,是多对多的联系。例如足球比赛中裁判员集和队员集之间相对于执法关系来说,就是一个 m:n 的联系。

与现实世界不同,信息世界中实体集之间往往只有一种联系。此时在谈论两个实体集之间的联系时,就可以略去联系名,直接说两个实体集之间具有一对一的联系、一对多的联系或多对多的联系。

需要说明的是在同一实体集的各个实体之间也存在着联系。例如,班长和同班同学之间就存在着一对多的联系。

1.3.3　数据模型

由信息世界数字化而得到数据世界,将产生与 DBMS 及硬件密切相关的数据模型。数据模型是数据库系统的核心。

任何一种数据模型都规定了一种数据结构,它是用来表示信息世界实体和实体之间联系的方法。数据结构描述了系统的静态特性,这是数据模型最本质的内容。

数据模型还必须定义对其中的数据可执行的操作及其操作规则。数据操作描述了系统的动态特性。对数据库的操作主要是数据维护和数据检索两大类,这是任何数据模型都必须规定的操作,包括操作符、含义、规则等。

另外,数据模型还必须提定义完整性约束的手段,并在操作中自动予以检查。对那些不符合约束条件的操作,将自动拒绝执行;对符合约束条件的操作,才真正予以执行,从而最大限度地保证数据的正确、相容和有效。

实际的数据库管理系统所支持的数据模型目前主要有五类。

1. 层次模型(hierarchical model)

以树形结构表示实体与实体之间的联系的数据模型称为层次结构。这实际上是由若干个代表实体之间一对多联系的基本层次联系组成的一棵树。树的每一个节点代表一个实体型。层次模型中,最上层的节点称为根节点,下面的其他节点都称为子节点。根节点只能有一个,子节点数量不限。除根节点外,每一个节点都只能有一个父节点,但可以有多个子节点。例如,学校的组织机构数据库就可以用层次结构来描述。

2. 网状模型(network model)

用网状结构描述实体及其之间联系的模型称为网状模型。网中的每一个节点代表一个实体型,网状模型突破了层次模型的两大限制:它允许任何一个节点有多个父节点;可以有一个以上的节点无父节点。用来表示多个从属关系结构,它是一种交叉关系的网状结构。例如城市交通管理

系统,每一个道路都可能与另外几条道路相联系,很显然用层次结构是难以实现对其管理的。

3. 关系模型(relational model)

用二维表结构来描述实体及其实体之间联系的模型称为关系模型。关系模型以关系数学理论为基础,在关系模型中操作的对象和操作的结果都是二维表,这种表就是关系。例如上面提到的学生表就是一个关系。

4. 面向对象模型(object oriented model)

面向对象模型是上世纪 80 年代新兴的将面向对象技术与关系型数据库技术结合而产生的一种新的数据模型。这是一种新型的可扩充的数据模型,即可根据用户的需要,自己定义新的数据类型及其相应的约束和操作。

5. Web 模型(Web Model)

Web 模型是上世纪 90 年代将新型的 Web 技术与关系型数据库技术相结合而产生的新型数据库模型。

需要指出的是,层次模型和网状模型也称为非关系模型,它们在数据库系统初期发挥了巨大的作用。关系模型具有完备的理论基础、简单的模型、说明性的查询语言、简单易学、使用方便等优点而风靡全球,是目前使用最广泛的数据模型,Visual FoxPro、Oracle、SQL、Access 等都是关系型数据库。在关系模型发展后,非关系型数据库则迅速衰退,在我国现在已很难觅其踪。面向对象模型和 Web 模型都是近年才出现的数据模型,是目前数据库技术研究的方向。它们也无一例外地建筑在关系模型的基础上。

支持哪一类数据模型的数据库管理系统就称为哪一类数据库管理系统。例如 Visual FoxPro 9.0 支持关系模型,就称它为关系型数据库管理系统(RDBMS)。

1.4 关系模型

关系模型将与实际问题有关的数据,分别归纳成若干个简单的二元关系,每个二元关系都可以建立一个二维表,简称为表。这些表之间,还可用某种逻辑关系而建立相互关系。例如根据学生的学号、姓名、性别、出生日期、政治面貌、家庭住址、奖惩情况,就可以建立一个如表1-1所示的表。

表 1-1 学生基本情况表

学号	姓名	性别	出生日期	政治面貌	家庭住址	奖罚情况
010001	张凯华	男	02/13/1980	团	渭南市站南路 24 号	三年三等奖学金
010102	李会琴	女	07/25/1979	党	杨凌区西农路 59 号	两年二等奖学金
010111	李小茜	女	12/21/1979	团	宝鸡市红旗路 103 号	
011216	宋秀兰	女	10/31/1980	团	延安市枣园大街 5 号	
011217	郭正宏	男	11/25/1979	团	渭南市东风街 106 号	
011320	姜亚男	女	09/30/1980		咸阳市世纪大街 108 号	
012001	杨书敏	女	08/18/1980		延安市宝塔路 214 号	
012002	宋越辉	男	06/09/1979	团	宝鸡市红旗路 265 号	
012003	杜拥军	男	05/20/1980		杨凌区西农路 132 号	
012234	王向东	男	04/26/1979	党	咸阳市阳街 10 号	

1.4.1　关系术语

在关系模型中，有许多术语和前面介绍数据库的术语相类似，它们称为关系术语。

1. 关系(relation)

一个关系就是一张二维表，常称为表。每一个关系都有一个与其他关系不同的关系名。

2. 属性(attribute)

关系中的每一列称为一个属性。每个属性都有一个属性名，在每列的首行显示。一个关系中不能有两个同名属性。在 Visual FoxPro 9.0 中，将属性名称为字段名。

3. 域(domain)

一个属性的取值范围称为该属性的域。

4. 元组(tuple)

关系中的每一行称为一个元组。一个元组即为一个实体的所有属性值的总称。一个关系中不能有两个完全相同的元组。在 Visual FoxPro 9.0 中，将元组称为记录。

5. 分量(component)

一个元组在一个属性上的值称为该元组在该属性上的分量。在 Visual FoxPro 9.0 中，将分量称为字段值。

6. 主码(primary key)

一个关系中的某个属性(组)，根据它(们)的值可惟一标识关系中的各个元组，且又不含多余的属性，则该属性(组)称为该关系的一个候选码(candidate key)，也称为候选关键字。若一个关系中有多个候选码，则选其中一个为主码(primary key)。主码也称为主关键字。

7. 主属性(main attribute)

包含在任何一个候选码中的属性，都称为关系的主属性；不包含在候选码中的属性称为非主属性或非码属性。

8. 外部码(foreign key)

若 A 是基本关系 R1 的属性却不是它的码，但 A 却与基本关系 B 的码 K 相对应，则称 A 是 R1 的外部码。R1 和 R2 不一定是不同的关系。当然，A 和 K 一定在同一组域上。

9. 关系模式(relational schema)

一个关系的名字及其全部属性名的集合称为该关系的关系模式。它用来定义一个关系，其结构为：

关系名(属性 1，属性 2，……，属性 n)

例如，上面图 1-1 所对应的关系模式可写为：

学生情况(学号，姓名，性别，出生日期，政治面貌，家庭住址，奖惩情况)

必须强调指出，关系模式是对实体型的定义，描述的是一类关系的数据结构；关系则是该类关系型中的一个具体关系的值，是某个时刻关系模式的状态或内容。关系模式是稳定的、静态的，而关系则是变化的、动态的。不过，在不会引起混淆的情况下，人们一般把两者都称为关系。这如同数据类型和变量的概念一样，数据类型代表一类数据的性质及其允许进行的操作，

是稳定的,而变量则是具有某数据类型的一个值,时刻变化的。

1.4.2 关系的规范化

不是所有的二维表格都能称为关系。一个二维表要称为关系或合理的关系,还应满足一定的限制,即关系要规范化。

关系规范化是指关系模型中的每一个关系模式都必须满足一定的要求。这些要求可分为最基本要求和高级要求两大类。满足最基本要求的二维表才能称为关系,最基本的要求有三个。

1. 属性不可再分和多值

最基本的要求是,关系中的每个属性都必须是不可再分的数据单元且属性不得多值,即通常人们讲的表中不能再含表,属性值仅一个。这称为关系的一级范式:1NF(first normal form)。通常表示为:R∈1NF。

例如,表 1-2 所给出的表,由于在成绩数据项中,又包含了四个"子数据项",因此就不是一个关系。

表 1-2 具有组合数据项的非规范化表

学号	姓名	成绩			
		语文	数学	外语	计算机
001	张三	90	95	87	91
002	李四	78	76	90	65
…	…	…	…	…	…

又例如,表 1-3 所给出的表,由于在学历数据项中,张三的学历中包含了两栏数据,属于属性多值,因此也不是一个关系。

表 1-3 具有多值数据项的非规范化表

职工号	姓名	职称	学历	毕业年份
001	张三	教授	大学 研究生	1982 1998
002	李四	讲师	研究生	2002
…	…	…	…	…

2. 属性不得同名

同一关系中不能有相同的属性名出现,但属性的左右位置可任意。

3. 元组不得完全相同

同一关系中不允许有完全相同的两个元组,但元组的先后次序可任意。

必须指出,符合最基本要求的关系并不是好关系,它存在着冗余大、插入异常、删除异常、修改异常等危险。

(1)冗余

冗余指关系中的数据相互重复、彼此依赖。数据冗余大,不仅造成存储空间的浪费,而且更可怕的是对数据进行修改时会易造成数据的不一致。

例如,有关系:学生成绩(学号,姓名,语文,数学,外语,计算机,总分,平均)。从关系规范化的角度看存在大量数据冗余。

这里总分是各门功课成绩的和,这种属性间的对应关系称为函数依赖。起决定因素的属性或属性组称为决定因素,被决定因素限定的称为被决定因素。

平均则是总分除以课程门数的结果,这种属性间的关系称为传递依赖。

(2)插入异常

插入异常指对关系进行插入操作时,该插入的部分信息无法插入。这是由关系中码不得为空引起的。

例如,在关系:选修课程(学号,课程编号,课程名,成绩),加了下划线的属性表示该属性为码,此时想插入一个尚未有一个学生选修的新开课程的课程编号和课程名属性,则无法插入。因为码不能为空。

(3)删除异常

删除异常指对关系进行删除操作时,不该删除的信息被连带删除。这同样是由于码不得为空造成的。

例如,在关系:选修课程中,假设某门课只有一个学生报名选修因而无法开设,需要将该生删去,但应保留课程信息以便以后学生选修。但在进行删除操作时会发现,伴随着该生信息的删除,该课程的信息也被删除了。

(4)修改异常

修改异常指对关系进行修改操作时,修改了某个数据项,引起必须修改多个元组,从而造成修改的复杂化。

例如,当某个选修课的课程名发生变化时,对于上述关系就不得不对凡涉及到该课程的所有元组都进行修改。

可见有必要对符合一级范式的关系进一步进行规范化处理。处理的方法一般是将关系进行必要的分解,从而会逐步得到关系的二级、三级、改进的三级、四级、五级范式:2NF、3NF、BCNF、4NF、5NF。它们之间的关系是:

$$5NF \subseteq 4NF \subseteq BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$$

二级范式(2NF):所有非主属性都完全函数依赖于任一候选码的一级范式。

三级范式(3NF):所有非主属性都不传递依赖于任何候选码的二级范式。

改进的三级范式(BCNF):关系中的每一决定因素都包含码。

通常,进行到 3NF 已足够了。4NF、5NF 则要求更高,实际上又使用极少,此处不予介绍。

1.5　关系运算

如前所述,关系模型是建立在严格的关系代数理论基础上的。在对数据库进行查询时,人们总是希望能尽快找到所需要的数据,这就需要对关系进行一定的运算。关系的基本运算分为两类,一类是基于传统的集合运算的关系运算,另一类是专门的关系运算。

关系运算涉及到比较和逻辑运算符,它们分别是:

(1)比较运算符:>、<、≥、≤、=、≠;

(2) 逻辑运算符:∨(或)、∧(与)、¬(非)。

1.5.1 基于传统集合运算的关系运算

传统的集合运算包含集合并、差、交、广义的笛卡尔积,它们的运算符号分别为:∪、-、∩、×。它们都是双目运算,其中前三个运算要求参加运算的两个关系必须具有相同的关系模式,且对应的属性具有相同的域。设参加运算的两个关系分别是 R1 和 R2,元组为 t,运算的结果是 S,则几种运算可以如下定义。

1. 并(union)

并运算定义为:

$S=R1 \cup R2=\{t \mid t \in R1 \vee t \in R2\}$

其含义是:任取元组 t,当且仅当 t 属于 R1 或 R2 时,t 属于 S。S 的模式与 R1、R2 相同。通俗讲,并是属于两个关系的元组组成的集合。

例如:设有两个关系,一个是参加计算机等级考试的学生的报名册,一个是参加英语四六级考试的学生报名册,现要查看所有参加了计算机等级考试或英语四六级考试的学生信息,就是一个并运算。

2. 差(difference)

差运算定义为:

$S=R1-R2=\{t \mid t \in R1 \wedge t \subseteq R2\}$

其含义是:任取元组 t,当且仅当 t 属于 R1 但不属于 R2 时,t 属于 S。S 的模式与 R1、R2 相同。通俗讲,差是由仅属于 R1 但不属于 R2 的元组组成的集合,即从 R1 中减去也属于 R2 的元组。

例如:查看仅参加了计算机等级考试而未参加英语四六级考试的学生的信息,就是一个差运算。

3. 交(intersection)

交运算定义为:

$S=R1 \cap R2=\{t \mid t \in R1 \wedge t \in R2\}$

其含义是:任取元组 t,当且仅当 t 既属于 R1 又属于 R2 时,t 属于 S。S 的模式与 R1、R2 相同。通俗讲,交是由两个关系共有的元组组成的集合。

例如:查看既参加了计算机等级考试又参加英语四六级考试的学生的信息,就是一个交运算。

4. 广义笛卡尔积(extended cartesian project)

广义笛卡尔积并不要求参加运算的两个关系具有相同的关系模式。

设 R1 为 n 目(即具有 n 个字段)关系,R2 为 m 目关系,则 R1 与 R2 的笛卡尔积 S 为:

$S=R1 \times R2=\{t_{r1} t_{r2} \mid t_{r1} \in R1 \wedge t_{r2} \in R2\}$

其含义是:任取元组 t_{r1}、t_{r2},当且仅当 t_{r1} 属于 R1 且 t_{r2} 属于 R2 时,t_{r1} 和 t_{r2} 的连接即为 S 的一个元组。

S 是一个$(n+m)$目关系。其中任何一个元组的前 n 个属性为 R1 的各属性，后 m 个属性为 R2 的各属性。如 R1 与 R2 中有同名属性，则在该属性前分别加上关系名作为前缀，关系名和属性间用“.”隔开，以示区别。例如：R1.a1，R2.a1。

若 R1 有 K1 个元组，R2 有 K2 个元组，则 S 有 K1 * K2 个元组。

在实际操作时，可从 R1 的第 1 个元组开始，依次与 R2 的每一个元组相组合，然后对 R1 的下一个元组进行同样的操作，直到 R1 的最后一个元组也进行完相同的操作为止，即可得到 S 的所有元组。

例 1.1　设有两个关系 R1 和 R2，分别求出它们的并、差、交、笛卡尔积。

R1

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

R2

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

S＝R1∪R2

A	B	C
a1	b1	c1
a1	b2	c2
a1	b3	c2
a2	b2	c1

S＝R1－R2

A	B	C
a1	b1	c1

S＝R1∩R2

A	B	C
a1	b2	c2
a2	b2	c1

S＝R1×R2

R1.A	R1.B	R1.C	R2.A	R2.B	R2.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c1
a2	b2	c1	a1	b2	c2
a2	b2	c1	a1	b3	c2
a2	b2	c1	a2	b2	c1

1.5.2　专门的关系运算

专门的关系运算包括投影、选择、连接、自然连接、等值连接等。投影和选择是一元操作，其他都是二元操作。

1. 投影(projection)

设属性名表(AttributeNameList)中的所有属性都是关系 R 的属性,则 R 在属性名表上的投影为 R 中只保留在<属性名表>上的各分量后形成的新关系(但对于重复元组仅保留一个),记为:

$$\prod_{AttributeNameList}(R)$$

投影操作也可表示为:

PROJECT *RelationName* (*Attribute*1,*Attribute*2,……,*Attribute n*)

投影操作的实际操作方法为:从 R 中逐次取出一个元组,首先去掉不在<属性名表>上的各属性值,然后按<属性名表>中属性的次序重新排列剩下的各分量,将排列结果作为一个新的元组送入投影结果(但若遇见结果关系中已有的元组则舍弃之)。

例 1.2 对图 1-1 所示的学生情况表在性别和政治面貌两个属性上进行投影得到 R3:

$$R3 = \prod_{性别、政治面貌}(学生情况表)$$

或:PROJECT 学生情况表(性别,政治面貌)

结果如表 1-4 所示。

表 1-4 R3

性别	政治面貌
男	团
女	党
女	团
女	
男	
男	党

为了方便,也可用属性在原表中的编号来代替属性名表中的属性名,例如本例也可记为:

$$\prod_{3,5}(学生情况表)$$

2. 选择(selection)

在关系 R 中选择符合某给定条件的全部元组,生成新的关系的操作称为选择操作,记为:

$$\sigma_F(R) = \{t \mid t \in R \wedge F(t) = True\}$$

其中 F 为逻辑表达式。逻辑表达式的基本形式为:$X_1\theta Y_1$。其中,X_1、Y_1 为属性名、常量、变量、函数等,θ 为逻辑或关系运算符。

选择操作也可表示为:SELECT *RelationName* WHERE *Lexpression*

例 1.3 在学生情况表中选择所有政治面貌为"团员"的全体男同学,生成关系 R4。

$$R4 = \sigma_{性别='男' \wedge 政治面貌='团'}(学生情况表)$$

或:SELECT 学生情况表 WHERE 性别='男' ∧ 政治面貌='团'

结果如表 1-5 所示:

表 1-5 学生基本情况表选择操作的结果

学号	姓名	性别	出生日期	政治面貌	家庭住址	奖罚情况
010001	张凯华	男	02/13/1980	团	渭南市站南路 24 号	三年三等奖学金
011217	郭正宏	男	11/25/1979	团	渭南市东风街 106 号	
012002	宋越辉	男	06/09/1979	团	宝鸡市红旗路 265 号	

3. 连接(join)

从两个关系的笛卡尔积中选取属性值满足一定条件的元组组成一个新的关系。表示为：

$$R1 \bowtie R2 = \sigma_{A\theta B}(R1 \times R2)$$

其中：A **是** R1 **的属性组**(A1,A2,…,A_k)，B **是** R2 **的属性组**(B1,B2,…,B_k)。$A\theta B$ **的实际形式为：**

$$A_1\theta B_1 \wedge A_2\theta B_2 \wedge \cdots A_k\theta B_k$$

A_i **和** B_i **不一定同名，但必须可比较。**θ_i(i=1,2,…,K)**均为关系运算符。**

连接操作也可表示为：

JOIN *RepalionName*1 AND *RelationName*2 WHERE *Condition*

例 1.4　设有如下两个关系 R1 **和** R2，**对它们进行连接操作生成** R5，**连接条件：**B<D。

$$R5 = R1 \underset{B<D}{\bowtie} R2$$

或：JOIN R1　AND R2　WHERE B<D

结果如下：

R1

A	B	C
1	2	3
4	5	6
7	8	9

R2

D	E
3	1
6	2

R1

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

4. 等值连接(equivalence join)

当连接条件表达式中，所有的 θ_i **均为“=”时的连接，即按照属性值对应相等所进行的连接，称为等值连接。**

5. 自然连接(natural join)

去掉重复属性的等值连接。记为：

$$R1 \bowtie R2$$

6. 除(division)

除法是广义笛卡尔积的逆运算。此处不再作深入介绍，有兴趣者请参考有关关系运算方面的书籍。

1.6　思考与练习

一、选择题

1. **在下列关系代数的操作中，不属于专门的关系运算的是(　　)。**

A) **自然连接**　　B) **投影**　　C) **广义笛卡儿积**　　D)**选择**

2. **关系模式规范化的最起码的要求是达到第一范式，即满足(　　)。**

A)**每个非码属性都完全依赖于主码**　　B) **主码属性惟一标识关系中的元组**

C)关系中的元组不可重复　　D)每个属性都是不可分解的

3. 设关系 R 和 S 的元组个数分别为 100 和 300,关系 T 是 R 和 S 的笛卡儿积,则 T 的元组个数是(　　)。

A)400　　B) 10000　　C)30000　　D)90000

4. 在关系代数中,从两个关系的笛卡儿积中,选取它们属性间满足一定条件的元组的操作称为(　　)。

A)投影　　B) 选择　　C)自然连接　　D)连接

5. 在数据库中可以创建和删除表,这是因为数据库管理系统提供了(　　)。

A)数据定义功能　　B) 数据操纵功能　　C)数据维护功能　　D)数据控制功能

二、填空题

1. 数据库管理系统是位于用户和_____系统之间的一个数据管理软件。

2. 关系代数是一种关系操纵语言,它的操作对象和操作结果均为_____。

3. 用_____表示实体之间联系的模型称为层次模型,或者说数据的层次模型是以记录类型(实体)为结构的有向树。

4. _____是用二维表表示实体集属性间关系以及实体集之间联系的模型。

5. 若关系中的某一属性组的值能惟一地标识一个元组,则称该属性组为_____。

6. 设有关系 R、S 和 T,如下表示:

运用以上关系中的数据,完成下述运算:

(1)$R \cup S$、$R \cap S$、$R - S$、$R \times S$

(2)$\sigma_{A=3}(R)$、$\prod_{A=3}(R)$

(3)$R \underset{R.A=S.A}{\bowtie} S$

关系 R

A	B	C
2	4	3
5	2	7
1	9	6
3	4	2

关系 R

A	B	C
3	4	2
2	4	3
6	2	7
5	4	2

第 2 章　Visual FoxPro 9.0 使用基础

Visual FoxPro 9.0 是继目前社会上最流行的 Visual FoxPro 6.0 之后，美国微软公司于 2004 年推出的 Visual FoxPro 的最新版本。它不但继承了 Visual FoxPro 6.0、7.0、8.0 等以往版本功能强大、界面良好、容易学习、操作方便的特点，又增加了更多的功能，特别是对 SQL 的支持更全面，成为微机上最受推崇的数据库管理系统软件。

全国的大多数企事业单位的中、小型数据库大都是用 Visual FoxPro 开发研制的。大中专院校的数据库技术课程，也大多将 Visual FoxPro 作为首选课程。伴随着全国计算机等级考试的普及和发展，Visual FoxPro 已成为广大考生、特别是广大考生二级考试的首选语种。

2.1　微机关系型数据库发展史简介

如前所述，根据数据模型将数据库分为了三种模型，即层次型、网状型、关系型。

关系型数据库以完备的理论基础、简单的数据模型、说明性的查询语言、易学易用等优点使用最为广泛，目前几乎所有的数据库管理系统，都是关系型的。

自上个世纪 80 年代初微型计算机上的第一个关系型数据库管理系统 dBASE 问世至今，20 多年微机数据库管理系统就经过了 dBASE、FoxBASE 和 FoxBASE＋、FoxPro for DOS 和 FoxPro for Windows、Visual FoxPro 的四个发展阶段。目前正朝着 Web 数据库发展。

伴随着微型计算机的发展，作为 FoxPro 系列数据库，发展非常迅速。

1993 年推出了 FoxPro 2.5 版，这是一个跨平台的产品，既可以在 DOS 平台下运行，又可以在 Windows 平台下运行。

1994 年推出了 FoxPro 2.5 的更新版 FoxPro 2.5B 和 FoxPro 2.6，其中 FoxPro 2.6 for DOS 成为 DOS 平台下的 FoxPro 最终版本。

1995 年推出了 FoxPro 3.0 版，这是一个可运行于 Windows 95/98、Windows NT 操作系统下的 32 位数据库管理系统，引进了面向对象和可视化的概念，明确提出了客户/服务器体系结构。

1997 年推出了 FoxPro 5.0 版，该版本引进了 Internet 和 Intranet 的支持，首次在 FoxPro 中实现了 ActiveX 技术。

1998 年推出了可视化编程语言集成包 Visual Studio 6.0，Visual FoxPro 6.0 就是其中的一个成员。它全面支持 Internet 和 Intranet 应用，并增强了与其他产品之间的协作能力。

2001 年推出了 Visual FoxPro 7.0，它在 6.0 版的基础上增加了一些新功能。

2003 年推出了 Visual FoxPro 8.0，该版本增加了 IDE(interactive development environment)和语言。

2004 年推出了 Visual FoxPro 9.0，它比 8.0 版又添加了许多新的功能。

2.2 Visual FoxPro 9.0 的重要性能指标

Visual FoxPro 9.0 是一个关系型的数据库管理系统(DBMS),它的性能指标极多,表2-1列出了它的一些主要性能指标,供用户参考。

表 2-1 Visual FoxPro 9.0 的主要性能指标

分 类	功 能	指标
表文件及索引	每个表文件的最大记录条数	10 亿
	表文件大小的最大值	2G
	每个记录中字符的最大数目	65500
	每个记录中字段的最大数目(1)	255
	每个表字段中字符数的最大值	254
	非压缩索引中每个索引关键字的最大字节数	100
字段特征	字符型字段的最大字符数	254
	数值型和浮点型字段的最大位数	20
	数据库表的字段名的最大字符个数	128
	整型数的取值范围	$-(2^{31}-1)\sim 2^{31}-1$
	数值计算中,精确值的位数	16
内存变量与数组	默认的内存变量最大数目	16384
	可设置的内存变量最大数目	6500
	数组的最大数目	6500
	每个数组中元素的最大数目	6500
程序和过程文件	源程序文件的最大行数	无限制
	编译后的程序模块大小的最大值(4)	64K
	每个文件中过程的最大数目	无限制

2.3 Visual FoxPro 9.0 的安装

2.3.1 安装前的准备工作

1. 对计算机硬件的要求

目前,任何一台微机都是奔腾或其兼容机,操作系统也都是 Windows 2000 或更高,因此无一例外地都满足 Visual FoxPro 9.0 安装的需要。只是应注意,必须安装在一个本地硬盘而不是映射盘上,并且最好卸载原来安装的 Visual FoxPro 其他版本,尽管 Visual FoxPro 9.0 允许和老版本在同一台机器的不同路径下存在。

2. 对计算机软件的要求

由于 Visual FoxPro 9.0 必须在 Windows 2000 SP3 /Windows XP SP2/2003 Server 支持下，因此，如果操作系统为 Windows 2000，则必须安装 Service Pack 3 补丁。

2.3.2　安装步骤

(1)将 Visual FoxPro 9.0 光盘放入 CD-ROM 驱动器。如果是活动硬盘则更好。

(2)系统会自动启动安装向导，此时用户单击“Install Visual FoxPro”，将出现 Visual FoxPro 9.0 的安装界面。如图 2-1 所示。

图 2-1　Visual FoxPro 9.0 安装界面

(3)选择“1”：Prerequisites，对系统环境进行设置，以保证 Visual FoxPro 9.0 能正确运行。此时，安装程序会显示出如图 2-2 所示的对话框，供用户选择安装或更新 Prerequisites。

图 2-2　Prerequisites 安装信息

(4)单击[Update Now!]按钮安装 Prerequisites，待安装完成时，会出现图 2-3 所示的安装窗口。单击[Done]按钮，返回图 2-1 所示的安装界面。

图 2-3 Prerequisites 安装完成

(5)单击“2”:Visual FoxPro 选项,安装 Visual FoxPro 9.0。将出现如图 2-4 所示的 Visual FoxPro 9.0 的开始安装界面。要求用户正确输入产品序列号“Product Key:”和用户名称“Your Name:”。

图 2-4 Visual FoxPro 9.0 开始安装界面

(6)正确输入产品序列号和用户名称后,单击[Continue]按钮,继续进行安装,此时出现如图 2-5 所示的选项(Option)界面,用来完成对 Visual FoxPro 9.0 的安装选项进行配置。

图 2-5 Visual FoxPro 9.0 的安装选项界面

(7)单击[Install Now!]按钮,系统开始安装 Visual FoxPro 9.0 。此时蓝色的安装进度条会不断前进,并给出正在安装 Visual FoxPro 9.0,请等待(Installing Microsoft Visual FoxPro 9.0 professional Please wait…)的提示信息。当安装完成时会出现安装完成界面如图 2-6 所示。

图 2-6　Visual FoxPro 9.0 安装进程

(8)安装完成后,单击[Done],结束英文本的 Visual FoxPro 9.0 的安装。

注意　有些版本的 Visual FoxPro 9.0 带有汉化补丁程序,请在英文系统的 Visual FoxPro 9.0 安装结束后,再运行其汉化补丁程序。

2.4　Visual FoxPro 9.0 的启动和关闭

2.4.1　Visual FoxPro 9.0 的启动

Visual FoxPro 9.0 的执行程序文件名是 VFP9.EXE,它是在 Windows 2000 SP3/XP/2003 Server 支持下运行的,所以启动 Visual FoxPro 9.0 的方法和在 Windows 下启动其他的程序相同。用户可以用 Windows 中启动可执行程序的四种方法的任一种,来启动它。

1. 从“开始”菜单启动

从“开始”菜单启动 Visual FoxPro 9.0 的步骤如下:

(1)打开“开始”菜单,选择“程序”选项。

(2)在“程序”菜单下选择“Microsoft Visual FoxPro 9.0”并单击,即可启动。启动后将进入如图 2-7 所示的 Visual FoxPro 9.0 主窗口,这是一个类似于 Windows 的窗口。

Visual FoxPro 9.0 在命令窗口上,较前面的 VFP6.0 版有了较大的改进,它记录了历次用户操作 VFP 工作的一些命令(甚至将 Visual FoxPro9.0 重新安装后也继续存在),供用户本次使用,这样极大地方便了用户。

2. 使用“运行”对话框启动

打开“开始”菜单中的“运行”对话框,利用“浏览”按钮查找“VFP9.exe”所在的驱动器号和文件夹,单击“VFP9.EXE”的图标,将它的驱动器号、路径、文件名、扩展名由系统自动写入“打开”列表框,按回车键。

图 2-7 Visual FoxPro 9.0 主窗口

3. 使用桌面快捷方式启动

如果桌面上有 VFP9.EXE 的快捷方式图标，双击之即可启动。

2.4.2 Visual FoxPro 9.0 的关闭

与启动类似，用操作系统提供的关闭文件的方法，关闭 Visual FoxPro 9.0 的方法和在 Windows 下关闭其他的文件相同也有三种方法。而 Visual FoxPro 9.0 同时又有本身的关闭命令。用户可以任选用一种来关闭它。

1. 使用“控制菜单”关闭

在 VFP 主窗口，单击左上角标题栏的 Visual FoxPro 9.0 图标，弹出控制菜单，选最后一项“×关闭(C) ALT+F4”，可以关闭 Visual FoxPro 9.0。当然直接使用组合键 ALT+F4 或在打开控制菜单后，直接使用热键“C”。

2. 使用“关闭”按钮关闭

在 VFP 主窗口，单击右上角标题栏的“关闭”按钮×。

3. 使用“QUIT”命令

在 VFP 的命令窗口，输入命令“QUIT”，按回车键。

2.5 Visual FoxPro 9.0 的基本知识

2.5.1 Visual FoxPro 9.0 的窗口组成

图 2-7 给出了 Visual FoxPro 9.0 启动后的主窗口，它和所有的 Windows 窗口一样第一行是标题栏，第二行是菜单栏，第三行是工具栏，最底行是提示栏；中间的大片空间是工作区窗口。各栏的作用也和 Windows 操作系统中的各栏作用相同。

在工作区窗口当 Visual FoxPro 9.0 刚启动后会出现一个“任务面板管理器”窗口和“命令窗口”。

2.5.2　Visual FoxPro 9.0 的操作方式

Visual FoxPro 9.0 数据库管理系统，为用户提供了多种操作方式，以满足不同用户对象的需要。这些操作就大的方面而言，可分为交互式操作和程序操作两类。在交互式操作中又可分为：命令操作、菜单操作、工具操作三类。

1. 交互式操作方式

(1)命令操作方式

命令操作方式指用户在 Visual FoxPro 9.0 的命令窗口，输入有关的操作命令，按回车键后，系统则立即进行对该命令进行解释，若输入的命令正确就执行之，如果输入的命令错误，则系统立即给出出错信息，用户可以根据该出错信息修改命令，重新执行。例如，在命令窗口输入：

DELETE FOR 性别="男"

按回车键后，将当前打开的数据表中的所有性别为"男"的记录加上删除标记，这里必须注意，命令所用的标点符号应是英文的半角标点符号。

在命令操作方式下，系统具有"历史"记忆功能，将历次 Visual FoxPro 9.0 操作中所用过的命令记录并显示在命令窗口，用户可以方便地通过光标移动键向上或下翻动找到需要重新执行的命令。

(2)菜单操作方式

交互式方式的第二种操作是菜单方式操作。Visual FoxPro 9.0 的菜单系统是一个下拉式菜单系统。使用菜单方式，用户可以不必记住操作命令的具体格式，通过鼠标和键盘即可在各个界面上完成对数据库的操作。这种方法的优点是用户无须编写任何程序，就可实现对数据库的大部分操作与管理，并且将每次菜单操作所对应的命令显示在命令窗口，以便用户对照记忆。

例如：

第一步，单击"文件"菜单，选择"新建"选项，会弹出"新建"对话框。

第二步，在"新建"对话框中选"表"，单击"新建"，将弹出"创建"对话框。

第三步，在"输入表名："列表框中输入要创建的表名(如：MyTable)，单击"确定"，则打开了"表设计器"。

第四步，在表设计器中设计表(例如 MyTable.dbf)的结构。

其实这 4 步操作可用一条命令：CREATE MyTable 来实现。

以后为了简便，本书将：

第 n 步简记为："Sn："；

将单击、前进、选择等操作一律简记为："→"；

将退回、后退等操作一律简记为："←"；

对各种选项一律简记为："XX"，这里要加上引号；

将双击简记为："→→"；

将弹出简记为："↓"；

将各种按钮简记为：[XX]；

将对话框、窗口、设计器等简记为：XX。

例如上面操作的第一到第三步，可简记为：

“文件”→“新建”↓ 新建 →“表”→[新建]↓ 创建 →“输入表名：” → 表设计器

(3)工具操作方式

Visual FoxPro 9.0 提供了许多工具，启动这些工具可调出工具所对应的许多选择项和对话框，填入有关参数，即可方便地完成各种操作。和 Windows 操作系统相同，放在工具栏的命令按钮，只要单击它，就可以进行相应的操作。

2. 程序操作方式

通过编程将需要操作的命令写入程序文件中，然后执行该程序，只要程序正确，既无语法错误，又无逻辑错误，系统就会编译通过生成可执行文件，并自动、快速、准确地完成对数据库的各种操作。

2.5.3 在 Visual FoxPro 9.0 中使用帮助

Visual FoxPro 9.0 的帮助菜单如图 2-8 所示。它第一个选项可使用 Visual FoxPro 9.0 本身提供的帮助信息，我们称这种帮助为一般帮助；第二个选项则可以访问 MSDN 帮助文件。

图 2-8 VFP 9.0 帮助菜单

对于 MSDN 帮助需要单独安装。这里仅介绍一般帮助。当用户需要系统提供帮助时，无论是操作环境还是语句的语法帮助，都可以通过它得到联机帮助。这种帮助可通过下述几种方法来实现。

(1)列出帮助的各项主题

列出要帮助的各项主题，按 F1 键或单击工具栏的“?”按钮，等到帮助窗口出现后，单击[内容]按钮。也可直接在“命令”窗口执行 HELP 命令。

(2)取得对特定主题的帮助

要获得对某特定主题的帮助，执行 F1，在调出帮助窗口后，单击“搜索”，在“输入要查找的单词：”列表框输入主题，单击“列出主题”或“显示”。也可在命令窗口直接按格式：

HELP *主题*

的格式获取主题帮助。例如：HELP Create Table。

2.6 设置系统集成开发环境

Visual FoxPro 9.0 允许用户按照自己的习惯定制开发环境。包括：设置主窗口标题；设置默认选项(路径、项目、编辑器、调试器、工具选项等)；设置临时文件；设置拖放操作的域映射等。

用户可以采取交互式方式或编程来修改 Visual FoxPro 的设置，也可以创建自己的配置文件，使得 Visual FoxPro 9.0 在启动时自动加载。

最常用的更改 Visual FoxPro 9.0 配置的方法有三种：

- 利用“选项”对话框；
- 通过 SET 命令；

· 编辑 Visual FoxPro 9.0 配置文件。

2.6.1　通过“选项”对话框修改环境配置

Visual FoxPro 9.0 的“工具”菜单的最后一个选项是“选项”，这是一个对话框，利用它，可以修改 Visual FoxPro 9.0 的配置。通过它修改的配置，在 Visual FoxPro 9.0 本次运行中就会起到作用，并且这种修改是永久性的，直到再进行新的修改。修改的步骤是：

S1：“工具”→“选项…”↓ 选项；

S2：根据需要，选择单击有关的选项卡打开它们的对话框，设置有关参数。

注意　设置完毕后：

① 仅单击“确定”按钮，则设置仅在本次 Visual FoxPro 9.0 运行中有效。这类似于使用了 SET 命令；

② 如果先单击 “设置为默认值”，再单击“确定”，则设置对本次及后面的运行均有效。

例如，设置默认目录可以使用户每次在进入 Visual FoxPro 9.0 时，都能自动地到达自己所希望的文件夹下去工作。系统的缺省路径是 VFP 安装时提供的路径。用户常常需要对此缺省路径进行修改。使用“选项”对话框可以达到一劳永逸的结果。

例 2.1　将用户事先已建好的文件夹“d:\learnvfp”设置为默认目录。

S1：“工具”→“选项…”↓ 选项；

S2：→“文件位置”→“默认目录” →“修改”→“使用默认路径”↓ 更改文件位置，结果如图 2-9 所示；

图 2-9　系统缺省文件位置

图 2-10　用户要设置的文件位置

S3：→ … ，浏览选定要设置为缺省路径的文件夹的路径：“d:\learnvfp”，如图 2-10 所示；

S4：→[确定]← 更改文件位置，→[确定]← 更改文件位置；

S4：→[设置为默认值]→ [确定]，结果如图 2-11 所示；

S4：→ “确定”← VFP 窗口，设置完毕。

图 2-11 默认目录设置结果

2.6.2 通过"SET"命令设置环境配置

在"选项"对话框中的大多数选项,都可以通过"SET"命令来实现。"SET"命令主要用在程序方式下,一般写在程序的开头的初始化部分,用来完成对程序运行的环境进行初始化。当然在命令窗口也可以交互使用,它的格式是:

SET KeyWord [TO] *Value*

例如:

```
SET STATUS OFF              && 关闭工作状态条显示
SET CENTURY ON              && 设置 4 位年份显示
SET DEVICE TO PRINFTER      && 设置格式化输出到打印机
SET EXACT ON                && 打开字符串的精确比较
SET STRICDATE TO 0          && 设置日期型和日期时间型数据的输入格式为传
                               统格式
SET DATE YMD                && 设置年/月/日的日期型数据显示格式
```

等等。

2.6.3 编辑 Visual FoxPro 9.0 配置文件设置运行环境

Visual FoxPro 9.0 的配置文件是一个源程序文件,在它里面可以定制 SET 命令中的各种参数值,设置系统变量以及执行命令或函数。常用的配置文件名为"CONFIG.fpw"作为配置文件的主名。

Visual FoxPro 9.0 在启动时会自动检查是否有配置文件,若无则使用系统的缺省配置,若有则自动读取配置文件内容并覆盖掉缺省配置。

1. 创建配置文件的方法存放位置和启动

创建配置文件的方法是在任何一种文本文件编辑器下，输入有关的命令，然后将文件储存为一个程序文件(这里习惯上使用 .fpw 为文件的扩展名)即可。配置文件存放的位置一般应放在“选项”所设置的缺省路径或 Visual FoxPro 9.0 所安装的目录中，或用 DOS 的 Path 命令所设置的路径中，否则系统在加载 Visual FoxPro 9.0 时，将找不到配置文件，从而只好使用在“Options”对话框中所建立的默认设置。用户也可使用交互式方法来启动所希望的某个配置文件。其方法是：双击该文件名或使用命令行参数用该文件来启动 Visual FoxPro 9.0。

2. 配置文件内容的编辑

(1)在配置文件中编辑”SET”命令中环境配置的格式

如要将 2.6.2 节的通过“SET”命令对环境配置关键字的设置，在配置文件中来设置，其格式为：

KeyWord = *Value*

例如：

COLLATE=”Machine”　　&& 设置字符的排序次序为机器码

(2)在配置文件中调用函数或执行命令的格式

要在配置文件中写入当 Visual FoxPro 9.0 打开后就首先调用的函数或执行的命令，可在配置文件中以下面的格式写出：

COMMAND= *Command*|*FunctionName*

例如，要求 Visual FoxPro 9.0 一启动，就执行“OPEN DATABASE Mydatabase”打开数据库的命令，则可在配置文件中写入：

COMMAND=OPEN DATABASE mydatabase

要在配置文件中写入当 Visual FoxPro 9.0 打开后就首先启动的应用程序，可在配置文件的任何位置以下面的格式将程序文件名称指定给系统变量：_STARTUP，也可在配置文件的最后一行使用 COMMAND 命令。

_STARTUP= *ProgramName*

或：COMMAND=DO *ProgramName*

例如，要求当 Visual FoxPro 9.0 一旦启动，就立即去执行应用程序学生学籍管理系统：“xsxjglxt.prg”，则应在配置文件中写入配置语句：

_STARTUP=xsxjglxt.prg

或在配置文件的最后一行写上：

COMMAND= DO xsxjglxt.prg

(3)在配置文件中设置系统内存变量

在配置文件中设置系统内存变量的格式是：

SystemMemoryVariable>=<Value>

例如，指定一个在跟踪窗口打开的情况下，程序每行运行的延迟时间(单位为秒，允许的范围在 0 s～5.5 s 时间，缺省为 0 s)为 3 s，以便跟踪程序执行，可设置系统内存变量_THROTTLE 的值：

_THROTILE=3

2.7 项目管理器

开发一个数据库应用系统，从数据库开发的整个过程的角度讲就是做一个项目。项目是由文件、数据、文档以及对象等组成的集合，它也是一种名为“项目”的文件，项目文件的扩展名为.pjx。一个应用系统的开发一般都是从建立项目文件开始的。在 Visual FoxPro 9.0 中，项目是由项目管理器(project manager)来管理的。

项目管理器并不保存各种文件的具体内容，它只记录各种文件的文件名、路径、文件类型以及编辑、修改或执行这些文件的方法。通过项目管理器，用户可以很方便地完成各种文件的建立、修改、运行、浏览等操作；还可把存在项目管理器以外的文件添加到项目管理器中；可以完成应用程序的编译连接，使应用系统最后生成可脱离 VFP 系统运行的可执行文件。

2.7.1 创建项目文件

项目管理文件的创建，可以用“菜单”创建，也可以用“命令”创建。

1. 利用菜单创建项目文件

S1：“文件”→“新建”↓ 新建，如图 2-12 所示。

S2：→“项目”→ “新建文件”↓ 创建 →输入项目文件名：“Myproject” →[保存] ↓ 项目管理器。

创建完成，结果如图 2-13“项目管理器”窗口。

图 2-12 “新建”对话框

图 2-13 项目管理器

注意 用菜单法建立各种文件的方法基本相同，以后关于用菜单法创建其他文件将一掠而过。

2. 利用命令创建项目文件

使用 CREATE PROJECT 也可创建项目文件，在命令窗口中键入以下命令即可。

```
CREATE PROJECT Myproject
```

2.7.2　项目管理器的使用

从图 2－13 可以看出，项目管理器能分别对数据、文档、类、代码、其他等五类文件进行管理，它采用了页框的界面，在这界面中有六个选项："全部"、"数据"、"文档"、"类"、"代码"、"其他"。

(1)全部(all)：显示该项目中包含在其他五个选项的全部内容，选项前的"＋"号表示单击之可查看其中所包含的内容。

(2)数据(data)：显示该项目中的所有数据。包括：数据库、视图、表等。

(3)文档(documents)：显示该项目中处理数据时所用到的所有文档。包括：表单、报表和标签。

(4)类(class)：显示该项目中自定义的所有类。

(5)代码(code)：显示该项目中所用到的所有程序。如：API 函数、应用程序等。

(6)其他(other)：显示该项目中所用到的一些无法分类的文件。如图像文件、声音文件、菜单文件等。

2.7.3　项目管理器中按钮的意义

从图 2－13 看出，项目管理器是一个 Windows 的对话框，它有两个按钮和一个含有六个按钮的按钮组，各按钮的意义如下。

1. 关闭按钮

关闭按钮用于关闭这个项目管理器，常用于退出 VFP 时。但如果我们在退出 VFP 时没有对项目管理器进行关闭，则下次启动 VFP 时将自动打开前一次正在使用的项目管理器，减少每次对项目文件打开、关闭的麻烦。

2. 缩(放)按钮

屏幕的显示范围是有限的，有时为了避免项目管理器占用太多的屏幕，可按下此按钮，此时项目管理器将变成工具栏窗口，原本向上的箭头变为向下。如图 2－14。

图 2－14　被缩成工具栏的项目管理器

再次单击按钮，还可将项目管理器展开恢复原样。

3. 命令按钮组

这是一组含有六个按钮的动态变化的按钮组，当选中项目管理器中的不同页面时，按钮的个数、标题及可用与否将会有所不同。

(1)"新建"按钮：单击可创建一个在项目中所选中的文件或新对象。

(2)"添加"按钮：单击可向项目文件中加入已存在的文件。

(3)"修改"按钮：单击可打开并修改在项目中选中的文件。

(4)"运行"按钮：单击可运行项目中选中的查询、表单和程序等文件。

(5)“移去”按钮:单击可从项目中移去选定的文件。

(6)“编译”按钮:单击可将项目文件中包含所有应用程序文件进行编译。

2.7.4 打开一个项目

项目文件可在命令窗口下键入命令 MODIFY PROJECT 打开。格式:

MODIFY PROJECT *ProjectName*

也可以通过“文件”菜单中选择“打开”菜单项,打开的方法同于 Windows 打开文件的操作方法。若“文件类型”不是“项目”,通过下拉组合框选择文件类型为“项目”。不过,如前所述,若每次退出 VFP 时不关闭正在处理的项目管理器,则下次启动时该项目管理器自动保持打开。

2.8 在 Visual FoxPro 9.0 环境下创建用户文件夹

用户的文件夹,既可以在 Windows 环境下创建,又可以在 Visual FoxPro 9.0 命令窗口下创建。创建的方法是在命令窗口运行 DOS 的创建目录命令。格式如下:

1. 命令格式

RUN|! MD [*Driver*:\][*Path*\]*Directory*

2. 功能

在 Visual FoxPro 9.0 的命令窗口中利用 DOS 命令创建一个文件夹。

3. 参数说明

- RUN|!:在 Visual FoxPro 命令窗口执行 DOS 的有关命令。
- MD:DOS 中创建目录的命令。
- *Driver*:\ :指定要创建的用户文件夹所在的磁盘驱动器符。缺省时为当前盘。
- *Path*\:指定要创建的用户文件夹所在的路径。缺省时为当前目录。
- *Directory*:指定要创建的用户的文件夹名。

例如:

! MD e:*learnvfp9**mydir*

SET DEFAULT TO e:*learnvfp9**mydir*

表示,在 E:盘的根目录下创建一个 *learnvfp9**mydir* 的文件夹,并把它设置为本次 VFP9.0 运行时的缺省路径。

2.9 思考与练习

一、选择题

1. 要启动 Visual FoxPro 9.0 的向导可以(　　)。

A) 打开“新建”对话框　　B) 单击工具栏上的“向导”图标按钮

C) 从“工具”菜单中选择“向导”　　D) 以上方法都可以

2. 退出 Visual FoxPro 的操作方法是(　　)。

A）从“文件”菜单中选择“退出”选项　　B）用鼠标单击关闭窗口按钮

C）在命令窗口中输入 QUIT 命令，然后按回车　　D）以上方法都可以

3. 显示与隐藏命令窗口的操作是（　　）。

A）单击“常用”工具栏上的“命令窗口”

B）通过“窗口”菜单下的“命令窗口”选项来切换

C）直接按 Ctrl＋F2 或 Ctrl＋F2 组合键

D）以上方法都可以

4. 在“选项”对话框的“文件位置”选项卡可以设置（　　）。

A）表单的默认大小　　B）默认目录

C）日期和时间的显示　　D）程序代码的颜色

二、填空题

1. 项目文件的扩展名是_____。

2. 打开“选项”对话框之后，要设置日期和时间的显示格式，应当选择“选项”对话框的_____选项卡。

3. 扩展名为.prg 的程序文件在“项目管理器”的_____选项卡中显示和管理。

4. 项目管理器的“移去”按钮有两个功能：一是把文件_____，二是_____文件。

三、上机题

1. 用两种以上的方式启动 Visual FoxPro 9.0。

2. 用四种方法退出 Visual FoxPro 9.0。

3. 在机房供学生用的开放磁盘上，试创建一个个人文件夹，并将其设置默认目录。

4. 建立名称为“xxvfp9.pjx”的项目，熟悉项目管理器的各种操作。

第 3 章　数据元素与表达式

数据是信息的载体，是客观事物属性的记录。属性的不同形式决定了数据具有不同的类型。不同的数据类型在计算机中，具有不同的存储方式，也具有不同的运算。学习 Visual FoxPro 9.0 与学习任何一种程序设计语言一样，首先都要讲清楚该语言允许的数据类型。

3.1　最基本的显示命令

为了在下面对所讲解的常量、变量和表达式的结果能够显示，我们需要先讲解一个最基本的输出命令——“?”命令。在命令窗口中，我们可以使用? 或?? 来显示表达式的值。

1. 命令格式

? | ?? *Expression*1 [FONT *cFontName* [, *nFontSize*]
[STYLE *cFontStyle*]] [, *Expression*2] ...

2. 功能

输出表达式的值。单问号为换行从下行开头输出，双问号为不换行从光标所在位置开始输出。

3. 短语和参数说明

· FONT 短语：FONT 短语用来设置要显示的表达式的字体和字符大小，*cFontName* 为要输出内容的字体，缺省为宋体；*nFontSize* 为输出内容的大小，单位为磅，缺省为 10；

· STYLE 短语：指定输出的修饰类型，表 3-1 给出了各种允许的修饰字符和作用。这些修饰字符可以组合使用，但 “Q”不能和“T”组合。

表 3-1　修饰字符及作用

修饰字符	修饰效果说明
B	加粗(Bold)
I	右倾斜(Italic)
N	标准(Normal)，缺省值
Q	不透明(Opaque)
—	删除线(Strikeout)
T	透明(Transparent)
U	下划线(Underline)

当短语缺省时，按系统规定的缺省设置输出结果。请注意若缺省了 FONT 短语，则也不能有 STYLE 短语。

为了一目了然有所区别，以后在讲解任何命令、函数时，凡命令及命令中的关键字一律用大写正体给出，凡参数一律用小写斜体给出。

例 3.1　在命令窗口中输入如图 3－1 所示命令，结果在主窗口输出。

图 3－1　？输出命令示例

3.2　常量和数据类型

3.2.1　常量

常量(constant)是在程序或命令执行过程不允许其值发生变化的数据，是在命令或程序中被直接引用的实际值。按是直接写出还是先定义成一个符号再引用，可将常量分为直写常量和符号常量。

常用的直写常量在 Visual FoxPro 9.0 中有以下类型。

(1)数值常量，例如：123、－56.98

(2)浮点常量，例如：1.23E－6、－78.98E7

(3)货币常量，例如：$12345.6549、$876.6578

(4)字符常量，例如："Visual FoxPro 9.0"、[秦东大学]、'计算机科学系'

(5)逻辑常量，例如：.T.、.Y.、.F.、.N.

(6)日期常量，例如：{^2005/10/01}

(7)日期时间常量，例如：{^2005/10/01 10:00 am}

常用的符号常量也有上述的六个类型，但它必须用预编译指令 #DEFINE 进行预定义，方可使用。预定义的格式为：

#DEFINE *ConstantName Expression*

其中：*ConstantName* 为符号常量名称，*Expression* 为要对符号常量预定义的值。

在程序中当符号常量再无使用价值时，可通过 #UNDEF 语句予以释放。格式：

#UNDEF *ConstantName*

例 3.2

```
#DEFINE g 9.8
……
#UNDEF g
```

3.2.2 数据类型

数据类型确定了数据在计算机内部的存储形式、值域及其所允许的运算。表 3-2 给出了 Visual FoxPro 9.0 的常用数据的类型。在表 3-2 的第一列中，凡带下划线的字符，表示该类数据的类型缩写，在表的定义命令中将会用到。

表 3-2 Visual FoxPro 9.0 的常用数据的类型

数据类型	描述	长度	取值范围
字符型(Character)	任何文本	表中 1～254 个字符，内存中每个字符占 1 字节	任何字符串
数值型(Numeric)	整数或小数，最多 20 位数字，缺省为两位小数	8 字节存储，在表中占 1～20 列	-.9999999999E+19 ～ .9999999999E+20
浮点型(Float)	同于 N 型，可用科学计数法给出	8 字节在表中，占 1～20	
列货币型(Currency)	存储金融数据，固定为 4 位小数	8 字节	-$922337203685477.5807～$922337203685477.5807
整型(Integer)	整数，仅用于表字段，为二进制值	4 字节	-2147483647～2147483647
自动增量型整型(Integer AutoInc)	同于整型，但有一个自动增量，只读，仅用于表字段	4 字节	其值由自动增量 NEXT 和步长值控制
双精度型(Double)	采用压缩格式，最多 18 位数字，仅用于表字段	8 字节	+/-4.94065645841247E-324 ～ +/-8.9884656743115E307
日期型(Date)	表示日期	8 字节	{^0001/01/01}～{^9999/12/31}
日期时间型(DateTime)	表示日期和时间	8 字节	{^0001/01/01 0:0:0 am)}～{^9999/12/31 11:59:59 pm}
逻辑型(Logical)	真或假的布尔值	1 字节	.T. 或 .F. ;.Y. 或 .N.
备注型(Memo)	存备注型字段的指针，仅用于表字段	4 字节	受有效内存限制
通用型(General)	存放 OLE 对象指针，仅用于表字段	4 字节	受有效内存限制

表 3-3 给出了 Visual FoxPro 9.0 的特殊数据的类型。

表 3－3　Visual FoxPro 9.0 的特殊数据的类型

数据类型	描述	长度	取值范围
二进制型(Blob)	一个不确定长度的二进制数据,指向表的备注文件(.fpt)中,Blob 数据不进行代码页转换	4 字节	受有效内存和/或 2.0GB 的文件大小限制
二进制备注型(Memo Binary)	同于备注型,但值不随代码页的改变而改变	4 字节	受有效内存限制
变体型(Variant)	可表示任何 VFP 数据类型,其类型由所赋数值的类型而定	同于所赋的数据类型的长度	同于所赋数据的值域
任意字符型(VarChar)	与字符型相类似,但不包含后缀空格,仅用于表字段,字段值居中显示	表中占 1 ～254 个字符,内存中每个字符一个字节	任何字符
任意二进制型(VarBinary)	二进制值,类似于 VarChar 型数据,它不包含附加的填充 0 或截去后缀 0,数据与代码转换无关	每个十六进制数占 1 个字节,变量最长 255 字节,字段最长 254 个字节	任何十六进制值
二进制字符型(Character Binary)	不希望进行代码页转换的任何字符,仅用于表字段	1～254 字节	任何字符
二进制任意字符型(VarChar Binary)	不希望进行代码转换的任何变体字符型	1～254 字节,每个字符一个字节	任何字符

3.3　内存变量和表达式

变量是在程序运行中其值允许发生变化的量。在 Visual FoxPro 9.0 中的变量根据是否与表的结构有关又分为两大类,一类是与数据表的结构定义无关的变量,称为内存变量,另一类则是与表的结构定义密切相关的变量,存放在表的字段中,称为字段名变量。数组是一种特殊的内存变量。另外,Visual FoxPro 9.0 还提供了一系列系统内存变量,这些变量均是以下划线开头的。

上节的图 3－2 和图 3－3 已列出了变量可能具有的数据类型,本节将结合内存变量介绍各类数据类型所对应的表达式。关于字段名变量,将在后面的表定义节次着重讲解。

3.3.1　内存变量

1. 内存变量的命名

与常量不同,内存变量应有一个固定的名字,其作用是与内存中为它开辟某个存储单元相对应,以便通过对变量名字的操作实现对存储单元内容的访问。

Visual FoxPro 9.0 规定,内存变量的名字,必须是以英文字母或汉字或下划线开头,后跟

字母、汉字、数字、下划线的一串字符，总长不得超过 254 个字符。

例如，A_BC、姓名、M100、_456 都是合法的内存变量名，而：9A、A+N、C*G 则不是合法的变量名了。

应提醒注意的是，系统内存变量以下划线开头的，为了不使用户自定义内存变量与系统内存变量相混淆，用户在定义自己的内存变量时，最好不要以下划线开头。

2. 内存变量的创建、赋值及数据类型

在 Visual FoxPro 9.0 中，内存变量没有专门的创建语句，其数据类型属于变体型，它的具体类型由赋给它的表达式的类型来确定。对内存变量赋值的方法有两种。

(1)用 STORE 语句为内存变量赋值

① 格式

STORE *Expression* TO *MvarNamelist*

(为了简洁，以后无特殊说明，常用 *Exp* 代表 *Expression*，用 *Mvar* 表示内存变量。)

② 功能

将表达式 Exp 的值赋给内存变量列表 *MvarNamelist* 中的各个内存变量。若内存变量列表中有若干个内存变量时，各变量之间必须用逗号“,”隔开(以后凡提到列表均如此)。

例 3.3 为内存变量 a、b、c、d 分别赋值 0，然后显示之。

```
STORE 0 TO a,b,c,d
? a,b,c,d
```

在主窗口将显示：0 0 0 0

(2)使用赋值操作符“=”为内存变量赋值

① 格式

MvarName = *Exp*

② 功能

将“=”右边表达式的值赋给它左边的内存变量。

例 3.4 st='中华民族'

3.3.2 数组

数组(array)是一种特殊的内存变量，它是由同一名字，不同下标组织起来的内存变量的集合。它所包含的每一个内存变量，都称之为数组的元素或下标变量。Visual FoxPro 9.0 中的数组与其他高级语言中的数组不同，它的各个元素可以具有不同的数据类型，因此更像其他高级语言中的记录型数据、结构体型数据。在 Visual FoxPro 中，数组最多为二维。

1. 数组的声明

数组原则上必须先声明而后使用。

(1)格式

DIMENSION|DECLARE *ArrayName*1(*nRows*1[, nColumns1])

[, *ArrayName*2(*nRows*2 [, nColumns2])] ...

(2)功能

声明一个一维或二维数组，依次可以同时声明多个数组。

(3)参数说明

- *ArrayName*:要声明的数组的名字。
- *nRows*:要声明的数组的行数。
- *nColumns*:要声明的数组的列数。

例 3.5　DIMENSION a(10),b(10,20)

(4)说明

- 由于数组本身是一种特殊的内存变量,因此数组命名规则和内存变量相同;
- Visual FoxPro 9.0 中,最多可声明 65 000 个数组,每个数组最多 65000 个元素;
- 数组声明后各元素一律被初始化为:.F. ;
- 在计算机中数组是按行存储的,二维数组可以当一维数组来使用;
- 数组可重新声明而改变其维数与大小,此时原元素值将保持不变,除非数组的容量变小使后面的元素丢失。
- 数组在声明和使用时,也可在数组名或元素名后使用方括号而代替圆括号。

2. 数组的赋值

数组的赋值分为整体赋值和元素赋值。它们都可以像内存变量一样通过赋值语句或赋值符来实现。

(1)为数组整体赋值

数组的整体赋值只要将表达式的值赋给数组名即可。

例 3.6　声明一个 2 行 4 列的数组 a,并将各元素的初值一律设置为数值 15。

```
DIMENSION a(2,4)
A=15              && 或用: STORE 15 TO a
? a(1,1),a(1,2),a(1,3),a(1,4),a(2,1),a(2,2),a(2,3),a(2,4)
```

结果将显示 8 个 15。

(2)为数组元素赋值

对数组元素的赋值,只要使用下标即可。

例 3.7　对上例中的数组元素 a(1,3)、a(2,4)分别赋一个字符串、一个日期。

```
A(1,3)=秦东大学
A(2,4)={^2007/05/01}
? a(1,3),a(2,4)
```

结果显示:秦东大学 05/01/07

(3)将二维数组化为一维数组

数组的各个元素在内存中存储在一段连续的存储单元中。一维数组按下标序号递增的次序存放;二维数组按行序号递增次序存放,每行内则按列序号递增次序存放。所以完全可以把一个二维数组当成一个对应的一维数组来使用,只要二维数组中以行号和列号所确定的元素与一维数组的元素对应即可。

设有一个二维数组:a(M,N),一个一维数组:A(M * N),则 a(i,j)和 A(k)的下标对应关系为:

K=(i−1) * N+j (i=1,2,…,M　j=1,2…,N)

例 3.8　设有一个二维数组 b(5,7),它的元素 b(4,6)的值是一个字符串“将二维数组转

为一维数组使用”。请将 b(4,6) 用一维数组元素的形式显示出来。

```
? b(27)          && 结果:将二维数组转为一维数组使用
```

3.3.3 算术运算符和算术表达式

1. 算术运算符

算术运算的优先级分为 5 级,按照运算的优先级别由高到低排列如表 3-4 所示。

表 3-4 算术运算符及其功能

优先级	运算符	功能	VFP 表达式	数学表达式	结果值
1	+、-	数符	-5 +8	-5 +8	-5 8
2	* * 或^	幂	-5 * * 2	$(-5)^2$	25
3	*、/	乘、除	2 * 3/5	2×3÷5	1.2
4	%	取模(余)	10%3		1
5	+、-	加、减	-5+10	-5+10	5

2. 算术表达式

所谓表达式是指将常量、变量、函数用运算符连接起来组成的具有一定数据类型的式子,表达式的值由最终的运算结果而确定。

算术表达式和结果应具有下面的格式:

$$nExp \times nExp \rightarrow n$$

这里,$nExp$ 表示操作数是数值型的;× 表示操作符,对于算术表达式,则为上表中的运算符之一;→ 表示结果将是;n 表示运算的结果类型是数值型的。以后讲述格式与此意义雷同。

如果在表达式中要改变运算的优先级,应使用圆括号而绝不能用方括号或花括号,圆括号可以嵌套。表 3-5 给出了三个 Visual FoxPro 9.0 算术表达式和数学表达式关系的例子。

表 3-5 算术表达式和数学表达式关系

数学表达式	Visual FoxPro 9.0 算术表达式	说 明
$\frac{a+b}{c}$	(a+b)/c	分子的圆括号不得缺少
$\frac{-b+\sqrt{b^2-4ac}}{2a}$	(-b+SQRT(b_2-4×a×c))/2/a 或 (-b+SQRT(b×2-4×a×c))/(2×a)	函数的圆括号和分子的圆括号相嵌套
$e^x+\ln x+\lg 10$	Exp(x)+LOG(x)+LOG10(x)	注意指数函数和对数函数的正确使用

3. 关于取模(余)运算符%

对于取模(余)运算符,设有数值型数据 a 和 b(a、b 既可以是整数,也可以是小数),则取余表达式:a%b 的运算结果规定如下:

(1)a、b 同号

当除数和被除数的符号相同时,结果为余数。

例 3.9　? 10%3,−10%−3,7.8%2.5,−7.8%−2.5

结果显示:1 −1 0.3 −0.3

(2)a、b **异号**

当除数和被除数的符号不同时,余数的数符与 b **相同,余数的数值**:|b|−|a|%|b|。

例 3.10　? −10%3,10%−3

结果显示:2 −2

3.3.4　字符型运算符和字符型表达式

1. 字符运算符

字符型运算符仅有两个:+、−,**它们都称为字符串的并置符,系同级别运算符。其表达式和结果应具有下面的格式**:

$cExp \times cExp \rightarrow c$

表 3-6 **给出了字符型运算符和字符型表达式的功能。**

表 3-6　字符型运算符和字符型表达式

运算符	表达式	功　能	实　例	结　果
+	S1+S2	将 S2 追加到 S1 之后	[中国]+[北京]	'中国 北京'
−	S1−S2	如果 S1 有后缀空格,将 S2 追加到 S1 之后并将原 S1 的后缀空格移动到结果字符串之后	[中国]+[北京]	'中国北京 '

例 3.11

命令	结果
? [abc]+[defg]	abc defg
? [abc]−[defg]	abcdefg
? len([abc]+[defg])	8
? len([abc]−[defg])	8

3.3.5　日期时间型运算符和日期时间型表达式

1. 日期型和日期时间型运算符

日期型、日期时间型运算符有两个,分别是:+、−。**它们是同一优先级的。对应的表达式为日期型和日期时间型表达式。值得注意的是加运算仅可以发生在日期或日期时间型数据与数值型数据之间;而减运算不仅可以发生在日期型、日期时间型数据本身之间,也可以发生在日期型、日期时间型数据与数值型数据之间。**

2. 日期型和日期时间型表达式

日期型或日期时间型表达式和结果应具有下面的格式:

$dtExp-dtExp \rightarrow n$、$dtExp \pm nExp \rightarrow dt$

这里 dt **代表日期型或日期时间型数据,单独的** d **表示日期型数据,单独** t **表示日期时间型数据。**

表 3-7 **给出了日期型和日期时间型运算符、表达式的功能。**

表 3-7 日期、日期时间型运算符及表达式

运算符	表达式	功 能	数值数据 $nExp$ 的单位	实例	结果
+	$dExp+nExp$	得到新日期	天(day)	{^2005/09/25}+6	10/01/2005
	$tExp+nExp$	得到新日期时间	秒(sec)	{^2005/10/01 10:30:50}+3600	10/01/2005 11:30:50 am
−	$dExp-dExp$	相差的天数	天(day)	{^2005/10/01}-{^2004/10/01}	365
	$tExp-tExp$	相差的秒数	秒(sec)	{^2005/10/01 20:0}-{^2005/10/1 18:30}	5400
	$dExp-nExp$	得到新日期	天(day)	{^2005/10/10}-9	10/01/2005
	$tExp-nExp$	得到新日期时间	秒(Sec)	{^2005/10/01 20:0}-5400	10/01/2005 06:30 Pm

3.3.6 关系型运算符和关系表达式

1. 关系型运算符和关系型表达式

关系运算符用于进行两个类型相容的数据之间的大小比较,比较的结果是逻辑型数据:.T. 或.F. 。它的表达式和结果可用下面格式描述。

$Exp1 \times Exp2 \rightarrow l$

这里所谓类型相容,是指操作数的大类型应该相同,如数值型和整型尽管不相同但却是相容的,而数值型和字符型则既不相同也不相容了。可见相容比相同要条件宽松。

关系型运算符分为通用关系运算符和字符专用关系运算符。通用关系运算符有六个,字符专用关系运算符有两个。它们均系同级别运算符。

表 3-8 给出了各个关系运算符及表达式。

2. 数值型数据比较规则

数值型数据比较大小很简单,数值大者为大,数值小者为小。

例 3.12 ? 123.45<987.23 && 显示:.T.

表 3-8 关系运算符和关系表达式

分类	运算符	功能	实例	结果
通用关系运算符	<	小于	2<3	.T.
	<=、=<	小于等于	{^2005/11/25}<={^2005/12/20}	.T.
	=	等于	"abc"=[abd]	.F.
	>=、=>	大于等于	[中华]>=[中国]	.T.
	>	大于	.T. >.F.	.T.
	<>、#、!=	不等于	5!=6	.T.

续表 3-8

分类	运算符	功能	实例	结果
专用关系运算符	==	精确等于（仅用于字符串）	[中华]==[中国]	.F.
	$	包含于（仅用于字符串）	[大学]$[秦东大学]	.T.

3. 日期型及日期时间型数据比较规则

以日历和时钟为准，按年、月、日、时、分、秒的次序逐项比较，朝后者为大，朝前者为小。

例 3.13　? {^2005/10/2}>{^2006/10/2},{^2005/10/2 10:30}<{^2005/10/2 22:30}

结果显示：　.F. .T.

4. 逻辑型数据比较规则

逻辑型数据的大小很简单，逻辑真值 .T. 大于逻辑假值 .F. 。

5. 字符型数据比较规则

字符串比较总的原则是，从第一个字符开始逐字符比较，当碰到某个对应字符不同时，该字符的排列次序大者所在的字符串大。在 Visual FoxPro 中字符的排列次序有三种，选用的排列方法不同会有不同的结果。

(1)设置字符排序次序

字符的排序次序，既可通过“选项”对话框中的“数据”选项来设置，也可用命令设置。

① 命令格式

SET COLLATE TO *cOption*

② 参数说明

cOption：指定要选用的字符排列次序的方式，它必须以字符串形式给出，可以分别是：“Pinyin”、“Machine”、“Stroke”，它们分别表示字符串按：拼音、机器码、笔画进行排序。“Pinyin”是系统的缺省设置。

例 3.14　请通过命令将字符排序方式设置为机器码。

SET COLLATE TO “Machine”

(2)精确比较

设有两个字符串 S1 和 S2，精确比较指当两个字符串完全相同——长度相等、对应字符相同时，表达式：

S1==S2

结果为真，其余情况均为假。它与 SET EXACT OFF|ON 状态设置无关。如将 S1、S2 位置颠倒，结果相同。

例 3.15　精确比较“秦东大学”和“秦东”的大小。

? “秦东大学”==“秦东”

结果显示：　.F.

(3)非精确比较

非精确比较所用的比较运算符是“=”，它与 SET EXACT OFF|ON 状态的设置有密切的

关系。

设有两个字符串 S1 和 S2,有表达式:S1=S2

在 SET EXACT OFF 状态(这是系统的缺省设置)下,S1=S2 是当 S2 为 S1 从首字符开始的一个子字符串时为真,其余情况均为假。与精确比较不同,在非精确比较的状态下,如果将 S1、S2 位置颠倒,则可能产生不同结果。

在 SET EXACT ON 状态下,S1==S2 分两步进行。首先,在较短的字符串后面添加空格到和较长的串等长,然后再逐字符比较大小。

例 3.16 非精确比较"秦东大学"和"秦东"的大小。

```
SET EXACT OFF
?"秦东大学"="秦东" &&. .T.
SET EXACT ON
?"秦东大学"="秦东" &&. .F.
```

6. 包含运算符 $

设有两个字符串 S1 和 S2,有表达式:S1 $ S2

当 S1 属于 S2 的一个子字符串时表达式结果为真,否则为假。

例 3.17 包含运算符的使用实例

```
?"Y" $ "yY"
```

结果显示:.T. 。

3.3.7 逻辑表达式

逻辑表达式用于逻辑型数据之间的运算,共三个。按其优先级从高到低排列为逻辑非:NOT、逻辑与:AND、逻辑或:OR。表 3-9 给出了它们各自的意义。

表 3-9 逻辑运算符和逻辑表达式

运算符	说明	表达式	意 义	实例	结果
NOT(.NOT.) 或 !	逻辑非	NOT A	当 A 为真时,结果为假,否则结果为真	A=.F. NOT A	.T.
AND(.AND.)	逻辑与	A AND B	A、B 同时为真时,结果为真,否则为假	A=.f. B=.t. a and b	.F.
OR(.OR.)	逻辑或	A OR B	A、B 至少一个为真时,结果为真,否则为假	A=.f. B=.t. A OR B	.T.

3.3.8 表达式的优先级

要正确使用表达式,就必须掌握它们的优先级,特别是在 Visual FoxPro 9.0 混合表达式中的各种运算符的优先级。优先级的判别可按下述原则进行:

(1)同类型表达式,按本类型内各种运算符的优先级确定运算次序;

(2)不同类型的混合表达式,优先级从高到低依次为算术和字符及日期和日期时间运算符,关系运算符,逻辑运算符;

(3)同优先级时,表达式按自左向右的次序执行;

(4)表达式中可以使用圆括号改变运算次序,圆括号可嵌套。

例 3.18　在 SET EXACT OFF 状态下,有下列混合表达式:

.NOT. 10+5>=2^4 .AND. 5<8 .AND. (.t. .OR. .f.) .OR. .NOT. '计算机'$'计算机系'

求判断其运算结果。

S1:解括号:

.NOT. 10+5>=2^4 .AND. 5<8 .AND. .t. .OR. .NOT. '计算机'$'计算机系'

S2:做算术、字符运算:

.NOT. 15>=16 .AND. 5<8 .AND. .t. .OR. .NOT. '计算机'$'计算机系'

S3:做关系运算:

.NOT. .f. .AND. .t. .AND. .t. .OR. .NOT. .t.

S4:做逻辑运算,先做逻辑非,再做逻辑与,最后做逻辑或,最终结果为.t. 。

3.3.9　内存变量的操作

1. 内存变量的显示

(1)内存变量的显示命令格式

DISPLAY|LIST MEMORY [LIKE *FileSkeleton*]

[TO PRINTER [PROMPT] | TO FILE *FileName* [ADDITIVE]] [NOCONSOLE]

(2)功能

显示当前所用的内存变量、数组的有关信息。同时列出已定义的内存变量的数目、已使用的字节总数以及还可使用的内存变量的数目等。DISPLAY 为分页显示,LIST 为滚动显示。

(3)参数和短语说明

• LIKE *FileSkeleton*,该短语指明可使用由 FileSkeleton 所给出的通配符。"*"代表可通配从当前位置开始的任意字符;"?"代表可通配当前位置的任意一个字符。

例 3.19　在内存变量显示中使用通配符

LIST MOMERY LIKE a*　　&&滚动显示第一个字符为"aa"的所有内存变量

DISPLAY MOMERY LIKE a? b*　　&&分页显示第一个字符为"a"、第三个字符位"b"的所有内存变量

• TO PRINTER [PROMPT],将显示结果发送到打印机上打印。如带有关键字 PROMPT,则在打印开始前会调出打印机参数设置对话框,供用户进行参数调整。缺省本短语,结果不会打印出来。

• TO FILE *FileName* [ADDITIVE],将显示结果保存到一个指定的文本文件中。如带有 ADDITIVE 关键字且指定的文本文件已存在,则表示将结果追加到该文本文件中,否则将覆盖掉原有文本文件的内容。缺省时,结果并不写入文本文件。

注意　TO PRINTER 短语和 TO FILE 短语二者只能选一个。

• NOCONSOLE,禁止将结果输出到 Visual FoxPro 9.0 的主窗口或用户定义的活动窗

口。缺省时,允许。

上述这些短语的意义,在以后的命令中如无特殊说明,作用均相同。

例 3.20　显示内存变量。

```
Clear Memory
A=10
B=[欢迎]
C=[VFP]
D=Date()
DIMENSION arr(3,4)
LIST MEMORY TO FILE mymvar
TYPE mymvar.txt
```

结果如图 3-2 所示(图中删除了对部分系统内存变量的列表)。

图 3-2　内存变量的显示

2. 内存变量的保存

内存变量的保存是指将内存变量保存到某个内存变量文件或表的备注型字段中,以便需要使用时从中将它们恢复。内存变量文件的扩展名是:.mem。

保存内存变量到内存变量文件的命令是:SAVE TO。

(1)命令格式

SAVE TO *FileName* | MEMO *MemoFieldName*
[ALL LIKE *Skeleton* | ALL EXCEPT *Skeleton*]

(2)功能

将指定的内存变量、数组保存到指定的文件或当前表的备注型字段中。

(3)参数和短语说明

- *FileName*:指定要保存内存变量的文件名。
- MEMO *MemoFieldName*:要保存内存变量的表的指定的备注型字段的名字。

FileName 参数和 MEMO 短语仅选其中一个。

- ALL LIKE *Skeleton*：保存所有与通配符 *Skeleton* 匹配的内存变量。
- ALL EXCEPT *Skeleton*：保存所有除与通配符 *Skeleton* 匹配的内存变量。

例 3.21　请将所有以 A 开头的内存变量存入 mfile1.mem，将除以 A 开头的所有内存变量存入 mfile2.mem，将全部内存变量存入 mfile3.mem 文件中。

```
SAVE TO mfile1 ALL LIKE A *
SAVE TO mfile2 ALL EXCEPT A *
SAVE TO mfile3
```

3. 内存变量的恢复

内存变量的恢复是指将保存在内存变量文件或当前表备注字段中内存变量，重新调入内存。恢复内存变量的命令是：RESTORE FROM。

(1)命令格式

RESTORE FROM *FileName* | MEMO *MemoFieldName* [ADDITIVE]

(2)功能

从指定的内存变量文件或备注型字段中恢复内存变量、数组。

(3)参数和短语

- *FileName*|MEMO *MemoFieldName*：指定要恢复内存变量文件名或备注字段名。
- ADDITIVE：将恢复的内存变量追加到内存中，与原有的内存变量共同使用。如无此选项，则先将原有的内存变量删除，再从内存变量文件或备注型字段中恢复。

例 3.22　恢复内存变量文件，使得仅有以 A 开头的内存变量起作用。

```
RESTORE FROM mfile1
```

4. 内存变量的删除

内存变量的删除是指将内存变量从内存中删除，从而释放它所占有的内存空间。删除内存变量的命令有 4 种格式。

(1)命令格式

格式 1：CLEAR MEMORY

格式 2：RELEASE ALL [EXTENDED]

格式 3：RELEASE *MvarNamerList*

格式 4：RELEASE ALL [LIKE *Skeleton* | EXCEPT *Skeleton*]

(2)功能

- 格式 1：清除所有的内存变量。
- 格式 2：交互式方式下作用与格式 1 相同，如出现在程序中时，则若有关键字 EXTENDED，将同时删除所有的全局内存变量，否则将不删除全局内存变量。
- 格式 3：仅删除内存变量列表所列出的内存变量。
- 格式 4：删除所有和通配符匹配或除和通配符匹配的内存变量。

例 3.23　删除除数组 arr 以外的所有内存变量

```
RELEASE ALL EXCEPT arr *
```

3.4 Visual FoxPro 9.0 的常用标准函数

Visual FoxPro 9.0 提供有大量的标准函数供用户使用。按照函数的功能分类，可以划分为：数值函数、字符函数、日期和时间函数、数组函数、类型转换函数、测试函数、数据库操作函数、其他函数等类型。本节将介绍前六类中最常用的部分标准函数，其他各类函数将在后续章节中陆续介绍。

函数调用的格式是：

FunctionName(*ParameterList*)

一般即使没有一个参数，函数调用时圆括号也不得省略。圆括号内的参数称为实参。

3.4.1 常用数值函数

数值函数指自变量和结果一般均为数值型数据的函数。表 3-10 给出了常用的数值函数。其中的参数 *n* 代表数值型表达式。对于数值型数据系统缺省精度是保留两位小数。

表 3-10 Visual FoxPro 9.0 常用数值函数

函数名称	功能描述	举例	结果
ABS(*n*)	|*n*|	ABS(−5)	5
EXP(*n*)	e^n	EXP(2)	7.39
LOG(*n*)	自然对数	LOG(10)	2.30
LOG10(*n*)	常用对数	LOG10(20)	1.30
INT(*n*)	去掉小数部分，仅保留整数	INT(10.96)	10
CEILING(*n*)	返回最接近于 *n* 但大于等于 *n* 的一个整数	CEILING(10.1) CEILING(−10.8)	11 −10
FLOOR(*n*)	返回最接近于 *n* 但小于等于 *n* 的一个整数	CEILING(10.1) CEILING(−10.8)	10 −11
SQRT(*n*)	开平方根，*n*≥0	SQRT(4)	2.00
MOD(*n*1,*n*2)	同于 *n*1%*n*2 运算	MOD(10,3)	1
PI()	圆周率函数	PI()	3.14
ROUND(*n*1,*n*2)	四舍五入函数	ROUND(−19.546,2)	−19.55
RAND(*n*)	随机函数，生成(0,1)随机数	RAND()	0.85
SIN(*n*)	正弦函数，*n* 为弧度值	SIN(PI()/2)	1.00
COS(*n*)	余弦函数，*n* 为弧度值	COS(PI()/3)	0.50
TAN(*n*)	正切函数，*n* 为弧度值	TAN(PI()/4)	1.00
ASIN(*n*)	反正弦，返回值单位：弧度	ASIN(1)	1.57
ACOS(*n*)	反余弦，返回值单位：弧度	ACOS(1)	0.00
ATAN(*n*)	反正切，返回值单位：弧度	ATAN(1)	0.79

下面，对比较特殊但却广泛使用的数值函数作进一步的说明。

1. ROUND(*n*1,*n*2)函数

四舍五入函数 ROUND(*n*1,*n*2)中，参数 *n*1 是四舍五入的对象，*n*2 是精度。*n*2 可以大于 0、等于 0、小于 0。各种情况下函数的意义如下：

- *n*2＞0，精确到小数点后的 *n*2 位；
- *n*2＝0，精确到整数个位；
- *n*2＜0，精确到小数点前的|*n*2|＋1 位。

例 3.24　? ROUND(125463.4746,3), ROUND(125463.4746,0), ROUND(125463.4746,－3)

结果显示：125463.475 125463 125000

2. RAND(*n*)函数

随机数函数中的参数 *n* 是一个种子值，它确定随机函数 RAND()返回值的顺序。如果在第一次调用 RAND(*n*)函数时使用了同一个种子值，而在后面的调用中不使用种子值，则会得到相同的随机数序列。如果在第一次调用时，使用了小于 0 的种子值，则种子值将从系统时钟中取得。为了获得最大限度的随机数序列，最好在第一次调用 RAND(*n*)时使用小于 0 的种子值，而后面的调用不用种子值。如果缺省了种子值 *n*，RAND(*n*)函数将使用系统给出的缺省种子值：100001。

随机数函数生成的是(0,1)之间的纯小数。将它和 INT 函数结合就会生成两个给定的整数之间的随机整数。设有两个整数 *m* 和 *n*(*m*＜*n*)，则生成[m,n]之间随机整数的公式为：

INT(RAND() * (n－m＋1))＋m

例 3.25　编写一个程序，其功能是生成 100 个[10,99]之间的随机整数，要求每次运行结果不同。(关于程序设计在后面将讲到)

图 3－3、图 3－4 分别给出了源程序，及第一次、第二次运行的结果。对比可以看出，尽管程序相同，但由于种子值 s＝－1，因此两次运行得到的随机数序列次序不同。

图 3－3　随机数序列 1

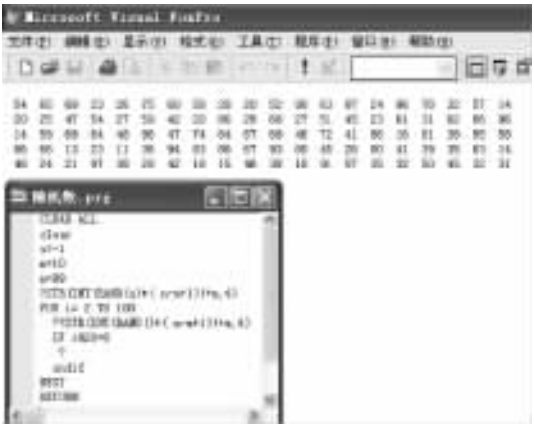


图 3-4 随机数序列 2

随机数函数的这个特性,在数据库技术中应用非常广泛,例如在考试试题生成系统中,利用它,可随机生成要抽取的试题号。

3.4.2 常用字符类函数

字符型函数是指自变量一般为字符型数据的函数。和数值型函数不同,字符型函数的返回值类型多样,有的函数结果是字符型,也有函数结果却是数值型或逻辑型等。表 3-11 给出了常用的字符型函数。表中的参数 s 表示字符型表达式。

下面对比较特殊但使用较广泛的字符型函数作进一步的说明。

表 3-11 常用字符型函数

函数名称	功能描述	举例	结果
LEN(*s*)	求字符串 *s* 的长度	LEN('中国')	4
LOWER(*s*)	将串 *s* 中的大写字母改为小写	LOWER('This')	this
UPPER(*s*)	将串 *s* 中的小写字母改为大写	UPPER('This')	THIS
SPACE(*n*)	生成 *n* 个空格的一个字符串	LEN(SPACE(5))	5
REPLICATE(*s*,*n*)	生成 *n* 个串 *s* 组成的新串	REPLICATE([A],4)	AAAA
LTRIM(*s*)	将字符串先导空格删除	LEN(LTRIM([我们]))	4
TRIM(*s*)、RTRIM(*s*)	将字符串后缀空格删除	LEN(TRIM(我们]))	4
ALLTRIM(*s*)	将字符串前后空格删除	LEN(ALLTRIM([我们]))	4
SUBSTR(*s*,*n*1[,*n*2])	从 *s* 的第 *n*1 个字符开始,取 *n*2 个字符,组成新串,*n*2 缺省指从 *n*1 一直取到最后	SUBSTR([秦东大学],5,4)	大学
LEFT(*s*,*n*)	从 *s* 的左边取 *n* 个字符	LEFT([中国共产党],4)	中国
RIGHT(*s*,*n*)	从 *s* 的右边取 *n* 个字符	RIGHT([中国共产党],6)	共产党

续表 3－11

函数名称	功能描述	举例	结果
OCCURS(*s*1,*s*2)	子串 *s*1 在串 *s*2 中出现的次数，若 *s*1 不是 *s*2 的子串，则返回 0	OCCURS ([ab], [abcdabefab-deadbg])	3
AT(*s*1,*s*2 [,*n*]) ATC(*s*1,*s*2 [,*n*])	子串 *s*1 在串 *s*2 中第 *n* 次出现的位置，*n* 缺省，指第 1 次；ATC()与 AT()功能类似，但不区分串中的大小写字符	AT([ab],[abcdabefAbdeaabg],3) ATC ([ab], [abcdabefAb-deaabg],3)	14 9
STUFF(*s*1,*n*1,*n*2[,*s*2])	子串替换函数：用 *s*2 替换 *s*1 中从 *n*1 开始的 *n*2 个字符，	STUFF([秦东大学物理学],3,6,[大])	秦大物理系
CHRTRAN(*s*1, *s*2, *s*3)	字符替换函数：当 *s*1 中的某一个或多个子串与 *s*2 中的某个子串相匹配时，用 *s*3 中的对应位置的等长子串替换 *s*1 中的子串	CHRTRAN([计算机 ABC],[计算机],[电脑])	电脑 ABC
LIKE(*s*1,*s*2)	字符串匹配函数：当两串对应字符都匹配时结果为.T. ；否则为.F. ；s1 中可用通配符：*、?	LIKE([ab*],[abcd])	.T.
&c*MvarName*	宏替换函数，将字符型内存变量的内容替换出来(从函数的调用格式看，称之为宏替换命令更确切)	C='FoxPro 9.0' ? "&C"	FoxPro 9.0

1. 宏替换函数 &c*MvarName*[.*cExpression*]

(1) 功能

在表达式中，宏替换函数出现的地方，将字符型内存变量的内容替换到该位置处。如果内存变量 *cMvarName* 后面还有其他表达式且又无运算符或其他分隔符将该内存变量与表达式分开，则在两者间加一个“. ”予以分隔。

(2)注意事项

- 在宏替换函数中，变量既不得递归定义也不得嵌套定义。
- 认真分析替换结果的属性是字符串还是变量名称，如果替换的结果是变量的名称，则该变量必须事先已有定义。

例 3.26　分析下面宏替换函数使用的结果。

```
S1='学习'
S2='s1'
S3='&s2'
S4=&s2
S5='&s4'
? s1,s2,s3,s4,s5          && 结果：学习 s1 s1 学习 学习
?'&s1&s1. 再 &s1'         && 结果：学习学习再学习
```

? &s1&s1. 再 &s1 && 出错信息:找不到变量"学习学习再学习"

2. 取子串函数 SUBSTR(s1, *n*1[, *n*2])

本函数的作用是:在字符串 S1 中从第 *n*1 个字符开始,取出 *n*2 个字符。

- *n*1≤LEN(*s*1);
- *n*2≤LEN(*s*1)－*n*1＋1。如果 *n*2＞ LEN(*s*1)－*n*1＋1 或 *n*2 缺省,则表示返回一个 *s*1 中从 *n*1 开始一直到最后一个字符所组成的子串。

SUBSTR 是 Visual FoxPro 9.0 中使用非常广泛的一个函数,它可以取代 LEFT 和 RIGHT 函数。

例 3.27 分析下面语句取子字符串的结果。

S='数据库技术及 Visual FoxPro 9.0 程序设计教程'

? SUBSTR(s,13,17) && 显示:Visual FoxPro 9.0

? SUBSTR(s,13) && 显示:Visual FoxPro 9.0 程序设计教程

? SUBSTR(s,1,10) && 代替 LEFT()函数,显示:数据库技术

? SUBSTR(s,LEN(s)－3,4) && 代替 RIGHT(),显示:教程

3.4.3 常用日期和时间类函数

日期和时间类函数用来返回日期和时间及其星期的。表 3－12 给出了常用的日期和时间函数。表中的自变量 *d* 代表日期型表达式、*t* 代表日期时间型表达式。

表 3－12 常用日期时间型函数

函数名称	功能描述	举例	结果
DATE()	返回系统当前日期	? DATE()	10/04/05
TIME()	返回系统当前时间(24 时制)	? TIME()	15:30:30
DATETIME()	返回系统当前日期时间(12 时制)	? DATETIME()	10/04/05 03：30：50 PM
YEAR(*d*|*t*)	返回自变量中的年份	? YEAR(DATE())	2005
MONTH(*d*|*t*) CMONTH(*d*|*t*)	MONTH()返回自变量中的月份(1～12),CMONTH()函数按中文或英文月份	? MONTH(DATE()) ? CMONTH(DATE())	10 十月 或 October
DAY(*d*|*t*)	返回自变量中月内的天数	? DAY(DATETIME())	4
DOW(*d*|*t*) CDOW(*d*|*t*)	返回自变量所对应的星期天数或中、英文星期几	? DOW(DATE()) CDOW(DATE())	4 星期三
WEEK(*d*|*t*)	返回自变量在当年内的周数	WEEK(DATE())	41
HOUR(*t*)	返回自变量中的小时部分(24 时制)	HOUR(DATETIME())	15
MIMUTE(*t*)	返回自变量中的分部分	MINUTE(DATETIME())	30
SEC(*t*)	返回自变量中的秒数部分	SEC(DATETIME())	50

3.4.4 类型转换类函数

类型转换函数的作用是实现自变量和函数返回值之间数据的转换。图 3－13 给出了常用

的数据类型转换函数。

下面将对有些函数做出更详细的说明。

1. 数字字符串转换成数值函数 VAL(s)

- 如果 s 为以非数字字符开头，转换结果为 0。如 VAL("A 123")的结果为：0。
- 如果 s 为以数字字符、.、+、－开头，转换结果为：截至到第一个非数字出现的位置为止。如 VAL("123A")的结果为 123。
- 如果字符串的前面部分如同科学计数常量的形式，则将其转换为对应的数值型数据。

2. 数值转换成字符串函数 STR(*n*1[, *n*2 [, *n*3]])

(1)参数说明

- *n*1：要转换的数值。
- *n*2：要转成的字符串的总长度。
- *n*3：要转成的字符串的精度。*n*3＞0，表示精确到小数点后的位数；*n*3＝0(缺省)，表示不保留小数位；系统转换时，对精度后面的紧跟一位数值进行四舍五入处理。

(2)注意事项

- 当 *n*2、*n*3 均缺省时，表示将 *n*1 转换为长度为 10 个字符的仅由 *n*1 的整数部分组成的一个字符串。
- 小数点也占一个字符位置。
- 转换时，首先保证整数部分能精确转换，然后剩余的长度才考虑小数部分。如果 *n*2 连整数部分都无法容纳，则结果将以 *n*2 个"＊"给出。

表 3－13　常用类型转换函数

函数名称	功能描述	举例	结果
Val(s)	将数字字符串转换为数值	? Val('123.456') ? Val("1.23e5")	123.46 123000.00
STR(*n*1,[*n*2[,n3]])	将数值 *n*1 转成长度为 *n*2、精度为 *n*3 的字符串	? STR(123.456,6,2)	123.46
CTOD(*s*)	将形式像日期型数据的字符串转换成对应的日期型数据	? CTOD([10/05/2005]) ? CTOD([-2005－10－6])	10/05/05 10/06/05
CTOT(*s*)	将形式像日期时间型数据的字符串转换为对应的日期时间型数据	? CTOT([10/05/2005 10:30]) ? CTOT[-2005－10－6 18:25])	10/05/05 10:30 AM 10/06/05 6:25 PM
DTOC(*d* [,1])	将日期型数据转换成对应的字符串，无参数 1 时结果为 MM/DD/YY[YY]格式，有参数 1 时结果为 YYYYMMDD 的格式	Set century on ? DTOC({-2005/10/6}) ? DTOC({-2005/10/6},1)	10/06/2005 20051006
TTOC(t [,1])	将日期时间型数据转换成对应的字符串，参数 1 与 DTOC()中的相似	SET CENTURY ON SET HOUR TO 24 ? TTOC({-2005/10/6 10：30：50}) ? TTOC({-2005/10/6 10：30：50},1)	10/06/2005 10：30：50 20051006103050

续表 3-13

函数名称	功能描述	举例	结果
CHR(*n*)	将十进制数值转换为对应的ASCII字符或汉字	? CHR(67) ? CHR(45217)	C 啊
ASC(*s*)	将字符串的第1个字符或汉字转换为对应的机内码值	? ASC("C") ? ASC("啊")	67 45217

例 3.28 设有内存变量 a,其值为:123456.4352764,给出下面的转换结果。

```
? STR(a)          &&. 123456
? STR(a,10,5)     &&.123456.435
? STR(a,8,5)      &&.123456.4
? STR(a,5,0)      &&.*****
```

3. 机器码转成对应的 ASCII 字符或汉字函数 CHR(*n*)

$n \leqslant 127$:转换成标准的 ASCII 字符;

$128 \leqslant n \leqslant 255$:转换成扩展的 ASCII 字符;

$41377 \leqslant n \leqslant 43518$:转换成标准图形符;

$45217 \leqslant n \leqslant 55289$:转换成一级汉字;

$55457 \leqslant n \leqslant 63485$:转换成二级汉字。

例 3.29 求出下面 CHR()的返回字符或汉字。

```
? CHR(122),CHR(148)          &&. z ”
? CHR(41378),CHR(43503)      &&.、十
? CHR(45217),CHR(55289)      &&. 啊 座
? CHR(55457),CHR(63486)      &&. 于 黯
```

ASC()与 CHR()函数功能相反。

3.4.5 测试函数

测试函数用来测试操作对象的状态,表 3-14 给出常用测试函数。函数中的自变量 *n* 表示工作区号,s 表示表的别名。同时假设有一个表 xsjkb.dbf 有 100 条记录,并已经打开。

表 3-14 常用测试函数

函数名称	功能描述	举例	结果
BETWEEN(vt,vl,vh)	测试 vt 是否在 vl～vh 之间,在其中返回.T.,否则返回.F.	? BETWEEN(5,2,10)	.t.
EOF([n|s])	测试记录指针是否指向表文件的结束标记,指向则返回.T.,否则返回.F.;空表时 EOF()为.T.	List ? EOF()	.t.

续表 3－14

函数名称	功能描述	举例	结果
BOF([n|s])	测试记录指针是否指向表文件的开始标记，指向则返回.T.，否则返回.F.；空表时 BOF()为.T.	GO TOP SKIP －1 ? BOF()	.t.
RECNO([n|s])	返回当前记录的物理记录号，注意当 BOF()为.T. 时，RECNO()为 1；EOF()为.T. 时，RECNO()为记录条数＋1	Go 10 ? RECNO()	10
RECCOUNT([n|s])	返回表的物理记录条数，注意空表时，该函数为 0	? RECCOUNT()	100
IIF(lExp,Exp1,Exp2)	条件测试函数；当 lExp 为真时，返回 Exp1 的值，否则返回 Exp2 的值	? IIF(2＞5,6,7)	7
DELETED([n|s])	删除标记测试函数：测试当前记录是否加有逻辑删除标记，加有则返回.T.，否则返回.F.	DELETE ? DELETED()	.T.
ISNULL(exp)	空值测试函数：当 exp 的值为 Null 时，返回.T.，否则返回.F.	x＝.null. ? ISNULL(x)	.T.
EMPTY(exp)	“空“值测试函数，当表达式为该数据类型规定的“空”值时，返回.T.，否则返回.F.	? EMPTY(0)	.T.
VARTYPE(Exp[,lExp])	数据类型测试函数	单独讲解	

下面给出关于测试函数的一些说明。

1. 各类数据的“空”值定义

函数 EMPTY()在自变量为“空”值时，将返回一个逻辑真。表 3－15 给出了各类数据的“空”值定义。

表 3－15　不同数据类型的“空”值定义

数据类型	“空”值	数据类型	“空”值
数值型(N)	0	字符型(C)	空串、空格、制表符、回车符、换行符、它们的组合
货币型(Y)	0	日期型(D)	空，如 CTOD(“”)
浮点型(F)	0	日期时间型(T)	空，如 CTOT(“”)
双精度型(B)	0	逻辑型(L)	.F.
整型(I)	0	备注型(M)	空(无内容)
二进制型	空(0h)或仅含 0 位	通用型(G)	空(无 OLE 对象)
变体型	空(0h)或仅含 0 位		

2. 数据类型测试函数

(1)**函数格式**

VARTYPE(*eExp* [, *lExp*])

(2)**功能**

返回表达式 *eExp* **的数据类型的一个大写字母,表** 3-16 **给出了这些字母的意义。如果表达式是一个数组,则返回数组第一个元素的类型。**

表 3-16 VARTYPE()函数返回的数据类型

返回字母	数据类型	返回字母	数据类型
C	字符型、备注型、随意字符型、二进制随意字符型	N	数值型、整型、浮点型、双精度型
D	日期型	Q	二进制型、变体型
G	通用型	T	日期时间型
L	逻辑型	X	.NULL.值
O	对象型	Y	货币型

(3)**参数** *lExp* **说明**

当表达式 *eExp* **的值为**.NULL. **时,将根据逻辑表达式的值决定是否返回的值:**

- **如果** *lExp* **的值为**.T. **,则返回** *eExp* **的原数据值;**
- **如果** *lExp* **的值为**.F. **或缺省,则返回** X **以表明** *eExp* **的运算结果为**.NULL. **。**

例 3.30

```
x="AAA"
y=10
x=.NULL.
z=$100.2543
u=0h
? VARTYPE(x), VARTYPE(x,.T.), VARTYPE(y), VARTYPE(z),
VARTYPE(u)
```

结果显示:X C N Y Q

3.5 Visual FoxPro 9.0 的命令结构

3.5.1 命令结构

Visual FoxPro 9.0 **的命令由命令动词开头,后随若干功能子句或关键字。前者用来说明"做什么",后者用来指明"操作对象"及其满足的条件等。这种命令格式规范,言简意赅,好记好用,深受用户喜爱。**

例 3.31 表记录显示命令

LIST [FIELDS *FieldList*] [*Scope*] [FOR *lExpression*1]
[WHILE *lExpression*2] [OFF] [NOCONSOLE] [NOOPTIMIZE]
[TO PRINTER [PROMPT] | TO FILE *FileName* [ADDITIVE]]

3.5.2　短语和关键字

Visual FoxPro 9.0 的命令中，功能短语和关键字很多，下面先介绍几种最常用的短语，这些短语的格式在上面的 LIST 命令中已给出。

1. 范围子句 scope

用于指定命令可处理的记录范围，是关系型数据库中完成表“筛选”功能的一种形式。共有四种形式可选择，其写法与含义分别是：

ALL	表示命令对表文件中的全部记录有效。
REST	表示命令对从当前记录开始到表的最后一条记录有效。
NEXT *n*	表示命令对从当前记录开始向后的 *n* 条记录有效。
RECORD *n*	表示命令只对第 *n* 条记录有效。

2. FIELDS 子句

用于指定命令允许处理的字段，完成关系型数据库中表的“投影”操作。

3. FOR 子句与 WHILE 子句

用于把命令的操作限制于符合条件的记录。既能完成对表“筛选”操作，又能完成对表之间的连接操作。但 FOR 子句与 WHILE 子句又有区别。

(1)FOR 子句将范围内的所有满足条件的记录都作为操作对象；WHILE 子句将范围内的满足条件的记录作为操作对象，但碰见第 1 条不满足条件的记录时则停止操作而不管后面是否还有满足条件的记录。

(2)当范围 scope 缺省时，FOR 子句的操作对象是全部记录；WHILE 子句的操作对象是 REST。

(3)当 FOR、WHILE 同时存在时，WHILE 子句优先于 FOR 子句。

4. FROM 子句

用于指定命令所需的信息来源。例如：RESTORE FROM myFile.mem

5. TO 子句

用于将命令操作的结果输出到文件或输出设备(如 PRINTER 等)。

3.5.3　命令的书写规则

(1) 所有命令必须以命令动词开始，回车键结束。命令动词和子句之间、子句与子句之间、子句与保留字、各保留字之间都应用至少一个空格隔开。

(2) 所有的功能子句在命令中出现的次序不影响命令的执行结果。

(3) 为了简化输入，Visual FoxPro 9.0 具有智能感知功能，用户在一般情况下正确输入命令动词或功能子句中前 4 个字符后按下空格键时，系统会自动将命令或保留字补全，同时显示出命令或短语的全部参数选项，供用户选择。

(4)在编写源程序时,一条命令可以分成若干行,只要在分行处加一个分号即可在下一行继续书写该命令的剩余部分。

3.6 思考与练习

一、选择题

1. 把日期 1999 年 5 月 1 日赋值给日期型变量 D 的方法(　　)。

A) D=05/01/99　　B) D="05/01/99"

C) D=CTOD ("05/01/99")　　D) D=DTOC("05/01/99")

2. 以下日期常量正确的是(　　)。

A){"2005-05-25"}　　B){^2001-05-25}

C){2001-05-25}　　D){[2001-05-25]}

3. 在下列表达式中,结果为日期型的是(　　)。

A) DATE ( )+TIME ( )　　B) DATE ( )+30

C) DATE ( ) - CTOD("01/01//98")　　D) 300-DATE ( )

4. 在下面的 Visual FoxPro 表达式中,不正确的是(　　)。

A) {^2001-05-01 10:10 10AM}-10　　B) {^2001-05-01} - DATE ( )

C) {^2001-05-01}+ DATE ( )　　D) [^2001-05-01] + [1000]

5. 在 Visual FoxPro 中求余的函数(　　)。

A) MOD ( )　　B) ROUND ( )

C) PI ( )　　D) SQRT ( )

6. 函数 LEN("计算机等级考试 Visual FoxPro")的计算结果是(　　)。

A) 计算机等级考试 Visual FoxPro　　B) 计算机等级考试

C) Visual FoxPro　　D) 27

7. 逻辑运算符的优先顺序是(　　)。

A) .NOT. 、.AND. 、.OR.　　B) .NOT. 、.AND. 、.OR.

C) .AND. 、.OR. 、.NOT.　　D) .AND. 、.NOT. 、.OR.

8. 字符型常量的定界符不包括(　　)。

A)单引号　　B)双引号　　C)花括号　　D)方括号

9. 执行? SUBSTR ("Welcome to the FoxPro System",12,12) 的结果是(　　)。

A) the FoxPro　　B) System　　C) to the　　D) Welcome

10. 在 Visual FoxPro 中,函数 ROUND (67.48759,2)的返回值是(　　)。

A) 67.48759　　B) 67.49　　C) 67.48　　D) 67.00000

11. 函数 MOD (73,-9)的值(　　)。

A) 1　　B) -1　　C) 8　　D) -8

12. 执行下面的命令后,屏幕上显示的结果是(　　)。

```
PP="ARE YOU SURE?"
MM="YOU"
? AT (MM, PP)
```

A) 5　　B) 7　　C) 4　　D) 0

13. 可以和 N 型数据一起计算的数据类型是(　　)。

A) C 型　　B) D 型　　C) L 型　　D) M 型

14. Visual FoxPro 中,有下面几个内存变量赋值语句(　　)。

X={^2001－07－28 10：15：20 PM}

Y=.T.

M=$123.45

N=123.45

Z="123.45"

执行上述赋值语句之后,内存变量 X, Y, M, N 和 Z 的数据类型分别是_____

A) D,L,Y,N,C　　B) D,L,M,N,C

C) T,L,M,N,C　　D) T,L,Y,N,C

15. 在下面 Visual FoxPro 表达式中,运算结果是逻辑值真的是(　　)。

A) EMPTY (.NULL.)　　B) 'AC'$'ACD'

C) AT ('a','123abc')　　D) 'AC'='ACD'

16. 设 D=5,命令 ? VARTYPE (D)的输出值是(　　)。

A) L　　B) C　　C) N　　D) D

17. 在下面函数中,函数值为数值的是(　　)。

A) BOF ( )　　B) CTOD ('01/01/96')

C) AT ('人民','中华人民共和国')　　D) SUBSTR (DTOC (DATE ( )), 7)

18. 设 N=886, M=345, K="M+N",表达式的值是(　　)。

A) 1232　　B) 数据类型不匹配　　C) "M+N"　　D) 346

19. 表达式 VAL(SUBSTR("1234386", 5, 1)) + LEN("Visual FoxPro")的结果是(　　)。

A) 13.00　　B) 14.00　　C) 15.00　　D) 16.00

20. 连续执行以下命令之后,最后一条命令的输出结果是(　　)。

SET EXACT OFF

X="A"

? IIF ("A"=X, X－"BCD", X+"BCD")

A) A　　B) BCD　　C) ABCD　　D) A BCD

21. 下列关于字段的命名规则,不正确的是(　　)。

A) 字段名必须以字母或汉字开头

B) 字段名可以由字母、汉字、下划线、数字组成

C) 字段名可以包含空格组成

D) 字段名可以由字母或合法的西文表示符组成

二、填空题

1. Visual FoxPro 9.0 中的数据类型有_____种,其中最常用的有_____种,仅用于表定义的有_____种,它们分别是:_____。

2. 符号常量的定义命令是_____。

3. 内存变量的命名规则是_____。

4. 字段名变量和内存变量的区别是_____、_____。

三、上机题

1. 在 Visual FoxPro 9.0 环境中，熟悉内存变量的赋值、显示命令。

2. 熟悉数值、字符、日期、关系、逻辑表达式及其混合表达式的运算规则。

3. 练习各种类型转换函数和测试函数。

1)指出以下左边各变量的值及类型(注意运算符的优先级)，并上机验证。

```
? A=10*2**3+100
? B=(10*2)**3+100
? C=A=B
? D=A<B
? E1=A=B.AND.A<B
? E2= A=B.OR.A<B
? E3=.NOT.A=B.AND.A<B
? E4=.NOT.(A=B.AND.A<B)
? E5=.NOT.(A=B.OR.A<B)
? E6="ABC"$"BDABCD".AND.A=B.OR.(10+5)<>10*5
? E7=E6.AND."C"+"A"$"A"
```

2)执行下述命令，熟悉函数功能。

(1)

```
B=DTOC(DATE(),1)
?"今天是:"+LEFT(B,4)+"年"
? ? IFF(SUBS(B,5,1)="0",SUBS(B,6,1),SUBS(B,5,2))
? ? IFF(SUBS(B,7,1)="0",SUBS(B,7,1),SUBS(B,7,2))+"日"
```

(2)

```
X=STR(12.4,4,1)
Y=RIGHT(X,3)
Z="&Y+&X"
? Z,&Z
```

(3)

```
DD=DATE()
? STR(YEAR(DD),4)+"年"+CMONTH(DD) + STR(DAY(DD),2)+"日"+
  CDOW(dd)
```

第 4 章　表的交互式操作

一个具体的关系模型，由若干个关系模式组成。反映在 Visual FoxPro 9.0 中，表现为一个数据库可以包含相互之间存在联系的多个表。同时，又有另一种情况，即有一种独立于数据库而存在的表。在数据库理论中，把数据库中包含的表称为数据库表，把独立于数据库的表称为自由表。

不管是自由表还是数据库表，对它们的操作都是关系模型最基础操作。

4.1　创建表

4.1.1　表的要素

在 Visual FoxPro 9.0 中，表有以下三个要素，即表文件名，表结构及表的记录。

1. 表文件名

在 Visual FoxPro 9.0 中，表文件名可使用不超过 255 个字符(可含空格，但不能包含以下字符：\、/、:、*、?、"、<、>、|)的文件名称，或使用中文命名。如要建立成一个关于学生情况的表，可以命名为“学生情况表”或“xsqkb”。

数据表的扩展名默认为.dbf，如非特殊需要，一般不再人为指定扩展名。

2. 表结构

Visual FoxPro 9.0 中表的结构用来定义关系模式，它是通过对表中各个字段的特性参数的定义来实现的。每个字段最少应包括：字段名称、数据类型、字段宽度(对于数值型字段还有小数位数)四个特性参数。

(1) **字段名**(Field_name)

字段又称字段变量，它的命名规则与内存变量命名规则一致。允许由字母、数字、汉字和下划线组成，但必须以字母或汉字开头，中间不能有空格。在自由表中，字段名长度不得超过 10 个字符。(在数据库表中，字段名的长度最长可达 128 个英文字符)。

(2) **字段类型**(Field_type)

在第 3 章中已介绍了 Visual FoxPro 9.0 中的字段数据类型。在实际应用中确定数据类型要与应用系统需求分析紧密结合。一般应按下述原则确定字段的类型：

- 如果字段值描述的信息与日期或时间有关的信息，采用日期类型或日期时间类型。
- 如果字段值描述的信息只有两种状态，为真或为假，采用逻辑型。
- 如果描述的信息要使用文字且所用字符不多(总长度不超过 254 个字符)，采用字符

型。

• 如果描述的信息要使用文字且超过 254 个字符，则只能采用备注型。

• 如果描述的信息用十进制数表示，但该项数据并不参与表的算术运算，则采用字符型比采用数值型更便于操作(如电话号码，邮政编码等)，否则则应采用数值型，具体采用整型、浮点类型还是双精度型，还应根据精度要求确定。

• 如果描述的信息涉及声音、图像、表格等，则必须采用通用型数据。

• 对于事先无法确定其长度的字符型字段，最好采用 VarChar 型。

(3) 字段宽度(Field_len)

在 Visual FoxPro 9.0 中有三种类型的字段宽度是可以改变的，它们是：字符型和二进制字符型字段，宽度在 1～254 个字符之间；数值型字段，宽度在 1～20 个字符；其他数据类型有由系统惟一确定的宽度值。在实际应用时，字段宽度定义要合适，定义太小将会使数据存储不下，太大将会浪费存储空间。

在定义数值型字段宽度时，应包括小数点与正负号，如某数值型字段宽度为 6，小数位为 2 位，则能存放的最大数值为 999.99，最小值为－99.99。

(4) 精度(Field_dec)

只有数值型数据才需要定义精度，即小数位数。小数位数至少要比字段总宽度小 2。

3. 表的记录

记录是表中字段值的集合，一条记录中最多可以有 255 个字段。Visual FoxPro 9.0 中表是以记录为单位来组织数据的，尽管可以对记录中的某个字段直接操作，但对表中数据进行定位仍然是以记录为单位进行搜索的。

4.1.2 创建表结构

创建表指建立表结构和向表录入记录的操作。在创建表之前，首先设我们的一切工作都限制的某个项目中进行。例如在学生学籍管理项目“xsxjgl.pjx”中进行。因此应首先创建并打开项目“xsxjgl.pjx”，然后再创建表。这样创建的表不言而喻将隶属于该项目。

常用的创建表结构的方法有三个，第一个是通过表设计器来创建；第二个是通过表向导来创建；第三个是通过 SQL 命令来创建。前两个方法，都是在交互方式下完成的，第三个实际上是 SQL 中的表定义功能。

1. 利用表设计器创建表结构

启动表设计器的方法有三种，方法一使用 Visual FoxPro 9.0 的创建表命令；方法二是使用“文件”菜单的“新建”选项卡；方法三是使用项目管理器。

(1)使用 Visual FoxPro 9.0 命令启动表设计器创建表结构

• 命令格式：CREATE *FileName*|?

• 功能：启动表设计器，在表设计器对话框中创建表结构

例 4.1 设有一个表文件xsqkb.dbf，其结构如表 4－1 所示。创建此表。

表 4－1　xsqkb.dbf各字段的属性

字段名称(Field_name)	字段类型(Field_type)	字段宽度(Field_len)	小数位数(Field_dec)
学号	C	8	
姓名	C	8	
性别	C	2	
出生日期	D	8	
政治面貌	C	2	
应往届生	L	1	
家庭住址	C	30	
奖惩情况	M	4	
照片	G	4	

S1：在命令窗口输入：

CREATE xsqkb

S2：弹出如图 4－1 所示的“表设计器”对话框，在“字段”选项卡中，输入相应的字段名、字段类型和宽度、小数位数等等。各字段输入完后(表结构已确定)，单击[确定]按钮将保存表结构(如果单击[取消]将不保存表结构)。

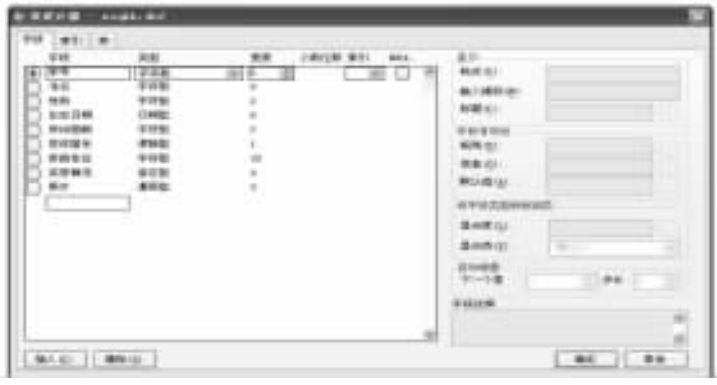


图 4－1　表设计器

(2)使用项目管理器启动表设计器创建表结构

S1：打开项目(如果项目尚未建立则先创建之)。

MODIFY PROJECT xsxjgl

S2：启动表设计器。→[数据]→[自由表]→[新建]；

S3：输入各字段属性，[确定]。

如图 4－2、图 4－3 所示。

2. 利用表向导创建表结构

利用表向导是菜单方式下创建表结构的另一种方法，下面举例说明创建的步骤。

图 4-2 新建对话框

图 4-3 项目管理器

例 4.2 利用表向导创建一个表文件 kcdmb.dbf,其结构如表 4-2 所示。

表 4-2 kcdmb.dbf 表结构

字段名	字段类型	字段宽度	小数位数
课程代码	C	8	
课程名称	C	30	
课程学时	I	4	
开课学期	C	4	

S1:打开表向导对话框。→[文件]→[新建]↓ 新建→"表"→[向导] ↓ 表向导第 1 步选择字段(Table Winzard Step1 Select Fields)。如图 4-4 所示。

S2:在"样表(Sample Tables)"栏中选择所需要的参考样表。如果无样表可供参考,则:→[添加(Add)]↓ 打开。选择与要创建的表结构相近的一个表,→[Add],从而将该表作为一个样表加入了"Sample Table"栏。(本题将xsqkb.dbf作为样表加入)。

S3: 在"样表"栏中选择 xsqkb 作为参考样表,此时在"可用字段(Available Fields)"栏中将显示出xsqkb.dbf中的各字段。在"可用字段"栏中,选取所需要的字段,把它写入"选定字段(Selected Fields)"中。这里有四个字段选取操作按钮。它们的功能分别是:

选取"可用字段"栏指定的一个字段到"选定的字段"栏。

选取"可用字段"栏的全部字段到"选定的字段"栏。

将"选定的字段"中指定的一个字段退回"可用字段"栏。

将"选定的字段"中全部字段退回"可用字段"栏。

本例选中四个字段:学号、姓名、性别、政治面貌,结果如图 4-5 所示。

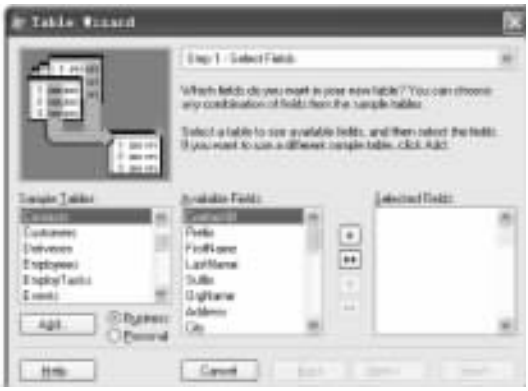


图 4－4　表向导窗口一

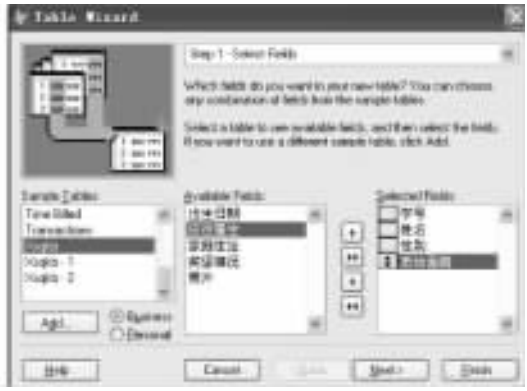


图 4－5　表向导窗口二

S4：→[**下一步**（Next）]↓ **步骤** 1a—**选择数据库**(Step1 a—Select a Database)。**如图** 4－6**所示**。

图 4－6　表向导窗口三

本窗口用来确定是否将要创建的表加入数据库文件中。共有两个选项。

选项 1：**“创建独立的自由表**(Create my table as a stand—alone free table)”。**选择该项，将在无数据库文件打开的情况下创建一个自由表，这是缺省选项。**

选项 2：**“将表添加到下列数据库**(add my table to the following database)”。**选择该项，允许用户在数据库栏中选择要将表加入的数据库文件名。**

本例中，由于要创建一个自由表，且无数据库文件打开，故选“创建独立的自由表”。

S5：→[Next]↓ **步骤** 2—**修改字段设置**(Step2—Modify Field Settings)，**根据情况对字段的属性做出修改，今将：**

“学号”字段改名为“课程编号”，长度改为 8；

“姓名”字段改名为“课程名称”，长度改为 30；

“性别”字段改名为“课程学时”，字段类型改为 I；

“党团关系”改名为“开课学期”，长度改为 4。

如图 4－7 所示。

图 4－7 表向导窗口四

图 4－8 表向导窗口五

S6：→[Next]↓ 步骤 3—为表建索引(Step3—Index the table)，如图 4－8 所示。

这里可以为表建立所需要的索引。关于索引的概念和创建复合索引的命令，将在后续节次中讲到。这里先不建索引。

S7：→[Next] ↓ 步骤 4—完成(Step4—Finish)，如图 4－9 所示。

图 4－9 表向导窗口六

在该对话框中有三个选项，用户可根据需要而选择合适的选项。本例中为了能及时地看到创建的表的结构，选第 2 项“保存表，然后浏览表(Save table and browse it)”。

S8：→[完成(Finish)] ↓ 另存为→[保存在(Enter Table)]，输入表文件名“kcdmb”。

S9：→[保存]，完成在向导指导下的表设计任务。

注意 以后用各种向导的操作均大同小异，只要打开向导对话框后，即可跟着提示走，所以如无特殊需要则不再讲述。

从上述的操作中可以看出，如果没有特殊要求，在菜单方式下，使用表向导创建表远没有

使用表设计器创建表快。而最快的方法则是使用 SQL(Structured Query Languige)命令来创建表结构。

3. 利用 SQL 命令创建表结构

(1)命令格式

CREATE TABLE | DBF *TableName*

(*FieldName*1 *FieldType* [(*nFieldWidth* [, *nPrecision*])] ,*FieldName*2…)

(2)参数和子句说明

- *TableName*:要创建的表的名称。
- *FieldName FieldType* [(*nFieldWidth* [, *nPrecision*])]:要创建的表的字段的名称、类型、长度、精确度。

例 4.3　创建一个学生成绩表,表名“xscjb.dbf”,表结构如表 4-3 所示。

表 4-3　xscjb.dbf 表结构

字段名	字段类型	字段宽度	小数位数
学号	C	8	
T8080101	N	4	1
T8080102	N	4	1
T8080203	N	4	1
Y3080101	N	4	1
W5080101	N	4	1
T8080204	N	4	1

在命令窗口输入:

CREATE TABLE xscjb(学号 C(8) NOT NULL,T8080101 N(4,1),T8080102 N(4,1),

T8080203 N(4,1),Y3080101 N(4,1),W5080101 N(4,1),T8080204 N(4,1))

按回车键,即得到了一个表 xscjb.dbf 的空表。所谓空表是指只有表结构但无一条表记录的表。

4.1.3　向表中录入数据

当表的结构一旦建好,下一步最重要的工作就是向表中录入数据。Visual FoxPro 9.0 中,向表中录入记录数据的方法有两种。一种是利用菜单交互式录入数据,第二种利用 SQL 命令添加记录。

1. 利用菜单交互式录入记录

利用菜单交互式录入记录的方法的步骤如下:

S1:打开表;

S2:打开表浏览窗口:[显示]→[浏览];

S3:进入追加模式:[显示]→[追加];

S4:逐条录入记录;

S5:存盘。

例 4.4　设学生情况表有如图 4-10 所示的记录,请将它们录入“xsqkb.dbf”中。

学号	姓名	性别	出生日期	政治面貌	应往届生	家庭住址	奖惩情况	照片
010001	张凯华	男	02/13/80	团	T	渭南市站南路24号	gen	gen
010102	李会琴	女	07/25/79	党	T	杨凌区西农路59号	gen	gen
010111	李小茜	男	12/21/79	团	F	宝鸡市红旗路103号	gen	gen
011216	宋秀兰	女	10/31/80	团	F	延安市杏园大街5号	gen	gen
011217	郭正宏	女	11/25/79	团	T	渭南市东风南路106号	gen	gen
011320	蒙亚男	女	09/30/80		F	咸阳市世纪大街108号	gen	gen
012001	杨书敏	女	08/18/80		T	延安市宝塔路214号	gen	gen
012002	宋延辉	男	06/09/79	团	T	宝鸡市红旗路265号	gen	gen
012003	杜拥军	男	05/20/80		F	杨凌区西农路132号	gen	gen
012234	王向东	男	04/26/79	党	T	咸阳市阳陵10号	gen	gen

图 4-10　学生情况表记录列表

按上述步骤进入追加模式后,光标开始停留在第一个记录的第一个字段上,录入相应的数据,移动光标到其他字段并录入数据,完成一条记录的数据录入。重复上述操作,依次录入全部记录。操作时应注意:

- **若录入数据充满整个字段,则光标自动移到下一字段,否则,须按回车键方可移到下一字段。**
- **逻辑型字段宽度为 1,它只能接受 T,t,Y,y,F,f,N,n 中任意一个字符。**
- **对备注型字段,可在该字段上双击鼠标或按下 Ctrl+PgDn 进入编辑器,录完数据后可用鼠标关闭窗口。**
- **对于通用型字段的录入,进入和退出通用字段编辑窗口的方法类似于备注型字段,但当打开通用字段编辑器后,要通过插入对象的操作来确定插入的对象。**
- **日期型字段应注意日期格式和日期的有效性,否则系统提示日期无效。**

2. 利用 SQL 命令录入记录

利用 SQL 命令向表中录入记录,可理解为向表中插入记录。命令如下。

(1)**命令格式** 1

INSERT INTO *dbf_name* [(*FieldName*1 [, *FieldName*2, ...])]
　　VALUES (*eExpression*1 [, *eExpression*2, ...])

(2)**参数和子句说明**

- *dbf_name*:**表名**。
- *FieldName*1 [, *FieldName*2, ...]:**要录入新记录数据的字段名,如果缺省指全部字段。**
- VALUES (*eExpression*1 [, *eExpression*2, ...]):**要录入新记录的各字段的值。如果命令中缺省了字段列表:***FieldName*1 [, *FieldName*2, ...]**,则** VALUES **子句中的表达式列表:***eExpression*1 [, *eExpression*2, ...]**必须依次和表结构中的各字段的类型相同,宽度相容,个数相等。**

例 4.5　设学生成绩表有如图 4-11 所示的记录,请将它们录入“xscjb.dbf”中。

INSERT INTO xscjb VALUES('010001',96,88,87,85,97,90)

学号	T3080101	T3080102	T3080203	T3080101	#5080101	T3080204
010001	96.0	88.0	87.0	95.0	97.0	90.0
010102	69.0	79.0	72.0	70.0	61.0	74.0
010111	73.0	69.0	77.0	76.0	69.0	66.0
011216	57.0	62.0	72.0	68.0	73.0	78.0
011217	67.0	69.0	80.0	53.0	76.0	68.0
011130	79.0	77.0	73.0	65.0	82.0	72.0
012001	83.0	85.0	74.0	89.0	88.0	73.0
012002	66.0	76.0	91.0	70.0	96.0	69.0
012003	72.0	87.0	86.0	72.0	73.0	65.0
012234	76.0	74.0	87.0	74.0	85.0	66.0

图 4-11　学生成绩表记录列表

……

INSERT INTO xscjb VALUES ('012234',76,74,87,74,85,66)

显然，这样的方法用来向表中录入数据是比较慢的，它适用于向表中仅插入个别记录的情况。

4.1.4　记录的显示

1. 菜单方式显示表记录

菜单方式下，显示表记录的步骤是：

S1：打开表；

S2：打开表浏览窗口。[显示]→[浏览]。

2. 命令方式显示表记录

在 Visual FoxPro 9.0 中，通过 LIST|DISPLAY 命令来显示当前表的记录列表。命令如下：

(1)命令格式

LIST|DISPLAY [[FIELDS] *FieldList*]
　　[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]
　　[OFF] [NOCONSOLE] [NOOPTIMIZE]
　　[TO PRINTER [PROMPT] | TO FILE *FileName* [ADDITIVE]]

(2)功能

显示记录数据，当显示内容超过一屏时，DISPLAY 具有分页显示功能，LIST 滚动显示。

(3)参数和子句说明

• FIELDS *FieldList*：指定要显示的字段。缺省时为全部字段，但备注型和通用型并不显示具体内容。备注字段仅以“Memo”或“memo”形式给出，通用字段仅以“Gen”或“gen”给出。

• *Scope*、[FOR *lExpression*1] [WHILE *lExpression*2]：均在 3.5 节中已讲过，意义相同。当它们全缺省时，LIST 命令的作用范围是全部记录，而 DISPLAY 命令的作用范围仅当前一条记录。

- OFF:指定是否显示记录号。有此关键字不显示记录号,否则显示记录号。
- NOCONSOLE:指定将命令执行的结果是否显示在屏幕上。有则不显示,无则显示。
- NOOPTIMIZE:指定是否采用 RUSHMORE 优化技术。选之不采用,不选采用。

以后命令中凡与本命令中的上述关键字同名且同功能者,则不再讲述。

例 4.6 显示表"xsqkb.dbf"的所有记录的学号、姓名、性别、出生日期四个字段。

```
USE xsqkb.dbf
LIST FIELDS 学号,姓名,性别,出生日期
```

显示结果:

记录号	学号	姓名	性别	出生日期
1	010001	张凯华	男	1980.02.13
2	010102	李会琴	女	1979.07.25
3	010111	李小茜	女	1979.12.21
4	011216	宋秀兰	女	1980.10.31
5	011217	郭正宏	男	1979.11.25
6	011320	姜亚男	女	1980.09.30
7	012001	杨书敏	女	1980.08.18
8	012002	宋越辉	男	1979.06.09
9	012003	杜拥军	男	1980.05.20
10	012234	王向东	男	1979.04.26

例 4.7 从第一条记录开始显示表"xsqkb.dbf"中男学生的学号、姓名、性别、出生日期四个字段,如果遇见第一个女学生的记录则停止显示。

```
USE xsqkb.dbf
LIST FIELDS 学号,姓名,性别,出生日期 WHILE 性别="男"
```

显示结果:

记录号	学号	姓名	性别	出生日期
1	010001	张凯华	男	1980.02.13

由于第二条记录不再满足指定条件,所以尽管后面还有满足条件的记录,也不再继续。

4.2 表结构的操作

表结构的操作,分为结构显示,结构修改,结构复制三方面的内容。

4.2.1 表的打开和关闭

一般情况下,对一个数据表使用之前必须先将其打开,操作结束后应及时关闭。和其他操作一样,表的打开和关闭既可以使用命令方式,又可以使用菜单方式。

1. 表的菜单方式打开与关闭

同于 Windows 的文件打开和关闭，无需介绍。

2. 命令方式打开表

(1)命令格式

USE [*TableName*|?] [EXCLUSIVE] [SHARED] [NOUPDATE]

(2)功能

打开指定表文件。

(3)参数和子句说明

· *TableName*|?，*TableName*：指定要打开的表的名称，? 指定调出文件对话框供选择要打开的表，缺省为?。表文件名可以带盘符和路径，扩展名.DBF 可省略。

· EXCLUSIVE，表示用户在网上操作时，表以独占方式打开。当表以独占方式打开时，用户可为表进行任何操作?

· SHARED，表示用户在网上操作时，表以共享方式打开。此时，用户对表的操作将受到网络的制约。建议对于初学者，一律使用独占方式打开表。

· NOUPDATE，禁止以修改方式打开表，防止修改表结构和表记录。

例 4.8　打开表文件“xsqkb.dbf”，然后显示它。

```
USE xsqkb EXCLUSIVE
LIST
```

3. 命令方式关闭表

用命令方式关闭表，有两种方法。

(1)方法 1——使用 USE 命令

- 命令格式：USE
- 功能：关闭本工作区内当前打开的表。

(2)方法 2——使用 CLOSE 命令

- 命令格式：CLOSE DATABASES|ALL
- 功能：关闭数据库或所有文件。

例 4.9　关闭当前打开的表文件“xsqkb.dbf”。

```
USE
```

4.2.2　表结构的显示

1. 利用表设计器显示表结构

S1：在共享方式下打开表；

S2：打开表设计器，浏览表结构。

为了防止显示表结构时，不小心将表结构进行某种修改，在打开表时最好使用共享方式。

例 4.10　利用表设计器及其显示表“kcdmb.dbf”的结构。

S1：→[打开]↓ 打开 →[文件类型] →“表”→[文件名] →“kcdmb” →“以只读方式”→[确定]

S2：→[显示] →[表设计器]，结果如图 4-12 所示。

图 4-12 利用表设计器显示表结构

用菜单法打开各种设计器,操作方法相同,以后如无特殊需要,则不再介绍。

2. 利用命令显示表结构

(1)命令格式

DISPLAY |LIST STRUCTURE

[TO PRINTER [PROMPT] | TO FILE *FileName* [ADDITIVE]] [NOCONSOLE]

(2)功能

显示当前表的结构。各子句和参数意义同前。

例 4.11 利用 LIST 命令显示 xsqkb 的结构,并将结果写到一个名为"sxqkb.txt"的文本文件中。

```
USE xsqkb
LIST STRUCTURE TO FILE xsqkb
```

4.2.3 表结构的修改

用户创建表结构后,因种种原因可能对所建的结构并不满意,需要进行修改。修改表结构的方法有两种,一种是通过表设计器,另一种是通过 SQL 命令。

1. 利用表设计器修改表结构

通过表设计器进行表结构修改,首先要打开表设计器。打开的方法分为:利用"显示"菜单、项目管理器、命令三种。

(1)利用"显示"菜单打开表设计器

利用[显示]菜单打开表设计器的步骤如下:

S1:以独占方式打开表。

S2:打开表设计器。

S3：修改字段属性。包括：字段名称、类型、宽度、精确度、插入字段、删除字段等。

S4：存盘。修改完后按[确定]；如果要作废本次修改，则按[取消]。

例 4.12　向表“xscjb.dbf”中的“学号”字段后插入一个新字段：总分 N(5,1)。

S1：打开表和表设计器。

S2：将光标移到字段“学号”所在的行，单击[插入]，则在字段“学号”之后插入了一个新字段，字段名为“新字段”。

S3：将“新字段”改为“总分”，并把类型改为“数值型”，将“宽度”改为“5”，将“小数位数”改为“1”，如图 4－13 所示。

图 4－13　利用表设计器修改表结构

S4：[确定]。

(2)利用项目管理器打开表设计器

S1：打开项目管理器；

S2：→[数据]→[数据库]→选表；

(如果是自由表，则：S2：→[数据]→[自由表]→选表；)

S3：→[修改]。

(3)利用 VFP 命令打开表设计器

① 命令格式

MODIFY STRUCTURE

② 功能

打开表设计器，修改当前表的结构。

2. 利用 SQL 命令修改表结构

对表结构的修改，对于初学者来说，最主要的修改是：插入或删除字段，修改已有的字段类型、宽度或小数位数，为字段更名。SQL 的表结构修改命令可以方便地完成这些工作。

(1) 向表中添加一个字段或修改某个字段的属性

① 命令格式

ALTER TABLE *TableName* ADD | ALTER [COLUMN] *FieldName*
FieldType [(*nFieldWidth* [, *nPrecision*])] [NULL | NOT NULL]

② 参数和子句说明

- *TableName*：指定要修改结构的表名。
- ADD | ALTER：指修改字段的操作方式。ADD：添加；ALTER：修改。
- COLUMN：列，可缺省，不影响操作结果。
- *FieldName FieldType* [(*nFieldWidth* [, *nPrecision*])]，要添加或修改的字段的名称、类型、宽度、精度。

例 4.13 向"xsqkb.dbf"中先添加"身高 N(3,2)"字段，然后再将它的宽度改为 3。

```
ALTER TABLE xsqkb ADD 身高 N(3,2)
ALTER TABLE xsqkb ALTER 身高 N(4,2)
```

(2) 从表中删除一个字段

命令格式：

```
ALTER TABLE TableName DROP FieldName
```

例 4.14 删除"xsqkb.dbf"中刚刚添加"身高"字段。

```
ALTER TABLE xsqkb DROP 身高
```

(3) 为表中某个字段更名

- 命令格式：

```
ALTER TABLE TableName RENAME OldFieldName TO NewFieldName
```

例 4.15 将"xscjb.dbf"中的"总分"字段改名为"成绩合计"。

```
ALTER TABLE xscjb RENAME 总分 TO 成绩合计
```

4.2.4 表结构的复制

表结构的复制有两种操作，第一种操作是将当前的表结构复制成另一个表的表结构；第二种操作是将表的结构复制成表的结构描述文件。

1. 复制表结构

表结构的复制是通过复制命令 COPY 来实现的。

(1)命令格式

```
COPY STRUCTURE TO TableName [FIELDS FieldList]
    [[WITH] CDX | [WITH] PRODUCTION]
    [DATABASE cDatabaseName [NAME cTableName]]
```

(2)功能

利用当前的表创建一个新的和当前表相似的空表。

(3)参数和子句说明

- *TableName*，要复制成的新空表的名字。
- FIELDS *FieldList*，该子句指定新表所包含的字段及其字段的顺序。缺省时将生成一个与当前表结构相同的表。
- [WITH] CDX | [WITH] PRODUCTION，为新表生成与当前表的结构复合索引相同的结构复合索引文件，CDX 与 PRODUCTION 功能相同。但对于当前表的主索引相对应的是将为新的空表生成一个候选索引。缺省本子句，将不生成复合索引文件。
- DATABASE *cDatabaseName* [NAME *cTableName*]，该子句指定把新表添加到一个

已存在的数据库中，使其成为数据库表。如果带有 NAME 子句，则可为表指定一个在数据库中出现的表名。当本子句缺省时，指生成一个自由表的表结构。

例 4.16　根据表“xscjb.dbf”复制一个新表“xscjb1.dbf”的结构。

```
USE xscjb
COPY STRUCTURE TO xscjb1
```

2. 将表结构复制成结构描述文件

结构描述文件是一个特殊的表文件，它的结构固定，有 18 个字段，各字段名字分别是：FIELD_NAME、FIELD_TYPE、FIELD_LEN、FIELD_DEC、FIELD_NULL、…、FIELD_STEP。它的记录由每个字段的特性组成。用户对表的结构描述文件可以像对表记录一样方便地进行修改。然后再利用它生成一个新表的结构。

将当前表的结构复制成结构描述文件的命令如下：

(1) 命令格式

COPY STRUCTURE EXTENDED TO *FileName*
　[DATABASE *DatabaseName*]
　[FIELDS *FieldList*]

(2)功能

利用当前表，创建一个表的结构描述文件。

(3)参数和子句说明

- *FileName*：要复制成的结构描述文件的名字，扩展名为“.DBF”；
- DATABASE *DatabaseName*：将结构描述文件添加到一个数据库中；
- FIELDS *FieldList*：要生成的结构描述文件各条记录所对应的当前表的字段名，缺省本子句指全部字段。

例 4.17　根据表“kcdmb.dbf”复制表结构描述文件“kcdmysb.dbf”，然后打开并显示它结构和记录。

```
USE kcdmb
COPY STRUCTURE TO kcdmysb EXTENDED;
USE kcdmysb
BROWSE
```

显示结果如图 4-14 所示。可以看出，表“kcdmysb.dbf”的记录刚好是表“kcdmb.dbf”的四个字段。

注意　图 4-14 中的字段因纸张宽度有限，并未全部显示出来。

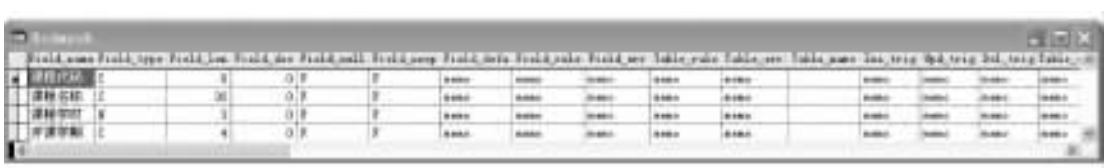


图 4-14　表的结构描述文件

3. 利用表结构描述文件创建表

利用表结构描述文件创建表结构的命令是：

(1) **命令格式**

CREATE *NewTableName*

[DATABASE *DatabaseName*] FROM *ext_TableName2*

(2)**功能**

由结构描述文件创建一个新表。

(3)**参数和子句说明**

- *NewTableName*：**要创建的新表名；**
- DATABASE *DatabaseName*：**指定要将生成的新表添加到一个数据库中；**
- ext_*TableName*：**结构描述文件名。**

例 4.18 根据表结构描述文件“kcdmysb.dbf**”，创建一个新表，表名“**kcdmb2.dbf**”，然后打开并显示该标的结构。**

```
CREATE kcdmb2 FROM kcdmysb
USE kcdmb2
DISPLAY STRUCTURE
```

结果显示和表“kcdmb2.dbf**”和“**kcdmb.dbf**”的结构完全相同。**

4.3 表记录的操作

4.3.1 记录指针定位

1. 记录指针

在操作表时，Visual FoxPro 9.0 **为表文件设置了一个记录指针，指针所指向的记录称为当前记录，**RECNO()**函数可返回当前记录的物理记录号。要对某条记录操作，需首先将记录指针指向该记录。**

表文件打开时，记录指针将指向第一条记录，即首记录将成为当前记录。记录指针可以移动指向任何一个记录。

指针可向下越过最后一条记录指向文件末尾。此时，RECNO()**函数的返回值为该文件的记录条数加** 1，EOF()**返回值为**.T. **。**

记录指针也可向上越过第一条记录而指向文件头。此时，RECNO()**函数的返回值仍为** 1，BOF()**返回值为**.T. **。**

按指针的移动是否以当前记录为准和是否与记录的索引有关，可以把指针定位分为绝对定位、相对定位、逻辑定位三种。按指针操作方式又可以把记录的定位分为命令定位和菜单定位两种方式。

2. 命令方式下的记录指针定位

(1) **记录指针的绝对定位命令**

① **命令格式**

[GO[TO]] [RECORD] *nRecordNumber*

② **功能**

将记录指针移到指定的物理记录号所给出的记录上。

③ **参数和子句说明**

• GO[TO] [RECORD]:**在记录指针绝对移动的操作方式下,均可缺省,但在逻辑方式下** GO **不得缺省。**

• *nRecordNumber*:**指定要指向的记录的物理记录号。**

例 4.19　请显示"xsqkb.dbf**"中的第** 3 **条记录。**

```
USE xsqkb
3                                    && 记录指针移到第 3 条记录上。
DISP FIELDS 学号,姓名,性别,出生日期
```

显示结果:

记录号	学号	姓名	性别	出生日期
3	010111	李小茜	女	1979.12.21

(2) **记录指针的相对定位**

① **命令格式**

SKIP [*nRecords*]

② **功能**

以当前记录指针位置为基准将记录指针向上或向下移动指定的记录条数。

③ **参数和子句说明**

• *nRecords*:**指定将指针向表文件开始或向结束的方向移动的记录条数。正值表示指针向表文件结束的方向移动,简称为指针下移,正号可缺省;负值表示指针向文件开始方向移动,简称为指针上移。缺省时表示指针下移到下一条记录位置,即缺省值为** 1。

例 4.20　分析下面记录指针的移动结果。

```
USE xsqkb
? RECNO( ),BOF( )                   && 显示结果为: 1        .F.
SKIP -1
? RECNO( ),BOF( )                   && 显示结果为: 1        .T.
SKIP 6
? RECNO( )                          && 显示结果为: 6
SKIP -2
? RECNO( )                          && 显示结果为: 4
```

(3) **记录指针的逻辑定位**

① **命令格式**

GO[TO] TOP|BOTTOM

② 功能

将记录指针移动到表的首记录或末记录上。这里,当表以索引方式打开时,指逻辑上的首、末;当表以非索引方式打开时,则指物理上的第一条、最后一条。

例 4.21 分析下面记录指针的移动结果。

```
USE xsqkb
GO BOTTOM
? RECNO(),EOF()                && 显示结果为: 10      .F.
SKIP
? RECNO(),EOF()                && 显示结果为: 11      .T.
GO TOP
? RECNO(),EOF()                && 显示结果为: 1       .F.
SKIP -1
? RECNO(),EOF()                && 显示结果为: 1       .T.
```

(4) 记录查找定位

所谓查找定位,指对表记录根据物理顺序或逻辑顺序从第一条记录开始向下查找,将记录指针定位在满足条件的记录中的第一条记录上,或确定无满足条件的记录。记录查找定位的命令如下:

① 命令格式

LOCATE [Scope] [FOR *lExpression*1] [NOOPTIMIZE]

② 功能

顺序查找当前表中满足给定的逻辑表达式条件的第 1 条记录。

③ 参数和子句说明

· FOR *lExpression*1:指定查询条件,当仅存在 FOR 短语时,指在整个表内进行查找定位。

(5) 定位测试函数

是否找到了满足条件的记录,可通过记录定位测试函数 FOUND()来确定,当函数值为 .T. 时,表示找到了;否则表示未找到。

(6) 继续查找

当表中有满足条件的若干条记录时,要找第 2 条、第 3 条、……、第 n 条满足条件的记录,可以重复使用:CONTINUE 命令。是否找到也可继续用测试函数 FOUND()予以测试,找完后 FOUND()返回.F. 。

例 4.22 在"xsqkb.dbf" 中逐条查找姓"李"的学生,先用测试函数测试是否找到,如果找到了再显示该记录。

```
USE xsqkb
LOCATE FOR 姓名='李'
? EOF(),FOUND(),RECNO()        && .F. .T. 2
DISPLAY                        && 显示关于李会琴的记录
CONTINUE
? EOF(),FOUND(),RECNO()        && .F. .T. 3
```

```
DISPLAY                                  && 显示关于李小茜的记录
CONTINUE
? EOF(),FOUND(),RECNO()                  && .T. .F. 11
```

可见对于 LOCATE……FOR……查找，当范围关键字缺省时 EOF()也可用来作为是否找到的测试函数。

3. 菜单方式下的记录指针定位

菜单方式下的记录指针定位通过“表”菜单中的“转移记录”选项来实现的。表 4－4 给出了命令操作和菜单操作记录指针的对应关系。

表 4－4　转移记录选项与记录移动指针间的对应关系

转移记录选项	对应命令	转移记录选项	对应命令
第一个	GO TOP	最后一个	GO BOTTOM
下一个	SKIP	上一个	SKIP －1
记录号	GO *nRecordNumber*	定位	LOCATE

4.3.2　记录的追加

记录的追加指在现有记录的基础上，再给表的最后一条记录之后添加上新的记录。根据一次操作所追加记录的条数可分为单条记录追加和成批记录追加。

1. 用命令方式追加记录

(1)单条记录追加命令

① 命令格式

```
APPEND [BLANK]
```

② 功能

为当前打开的表追加一条记录。

③ 参数和子句说明

• BLANK：指定向表中追加一条空白记录。如缺省，则表示追加一条空记录，同时将光标停留在本记录的第 1 个字段上等候输入字段值。

例 4.23　向表“xsqkb.dbf”中追加一条空白记录

```
USE xsqkb
APPEND BLANK
BROWSE
```

结果表明，在 xsqkb 的最后新添加了一条空白记录。请注意，添加空白记录的目的常常是为了通过记录替换命令来为表写入新的记录值。

(2)成批添加表文件的记录

在实际工作中，一个大型的表文件，可能包括成千上万条记录，如果由一个人来录入数据，那将需要较长时间。若由多个人分头向结构相同的不同表中录入数据，最后再将各个表汇聚到一个表文件中，就可在很短时间内完成数据录入。

将一个表文件的记录数据添加到另一个表文件的末尾，称为成批添加表文件的记录。

① 命令格式

APPEND FROM *TableName* [FIELDS *FieldList*] [FOR *lExpression*]

② 功能

将一个表文件中的记录按照指定的条件添加到当前表的末尾。

③ 参数说明

- *TableName*：要追加的表文件名。
- FIELDS *FieldList*：指定将字段列表中的字段值追加到当前表中。如果缺省，指将要追加的表中与当前表中字段名相同、类型相容的字段值追加过来。

④ 说明

APPEND FROM 命令可用于结构不完全相同的表文件间的记录添加，但只能处理同名字段同类型的数据追加。若两个表文件同名字段的宽度不同，则以当前表文件的字段宽度为基准。

例 4.24 将 xsqkb 中的结构(除“家庭住址”、“奖惩情况”、“照片”字段外)复制成 LS 表，再将 xsqkb 中的性别为“女”的记录追加到 LS 表文件之后。

```
USE xsqkb
COPY STRUCTURE TO LS FIELDS 学号,姓名,性别,出生日期,政治面貌,应届否
USE LS
APPEND FROM xsqkb FOR 性别=[女]
LIST
```

结果如下：

记录号	学号	姓名	性别	出生日期	政治面貌	应往届生
1	010102	李会琴	女	1979.07.25	党	.T.
2	010111	李小茜	女	1979.12.21	团	.F.
3	011216	宋秀兰	女	1980.10.31	团	.F.
4	011320	姜亚男	女	1980.09.30		.F.
5	012001	杨书敏	女	1980.08.18		.T.

2. 用菜单方式追加记录

菜单方式下也可以实现表记录的单条或成批追加。

(1)利用“表”菜单实现记录追加

利用“表”菜单既可以单条也可以成批向当前表中追加记录。

用“表”菜单向表中成批追加记录。这事实上是生成 APPEND FROM 命令，步骤如下：

S1：打开表。

S2：→[显示]→[浏览]。

S3：→[表]→[追加记录]↓ 追加来源，如图 4-15 所示。

图 4－15　“追加来源”对话框

图 4－16　“追加来源选项”对话框

S4：在追加来源对话框，分别在“类型”选项中选“表(DBF)”，在“来源于”选项中输入要添加的表名。

S5：如果有追加条件，→[选项] ↓ 追加选项，如图 4－16 所示。

S7：输入要追加的字段的字段名，追加所用的条件。

S8：→[确定]。

(2)利用“显示”菜单实现记录追加

对于当前表，也可以通过“显示”菜单操作实现记录的追加，不过这里的记录需要用户逐条输入。步骤如下：

S1：→[显示] →[浏览]。

S2：→[追加模式]。

4.3.3　记录的插入

当因种种原因需要向表的某两条记录之间插入一条新记录时，可使用此命令。

命令格式

INSERT [BEFORE] [BLANK]

注意　本插入语句是为了和过去的 FoxBase、dBase 兼容而保留的。从关系数据库的规范化要求看，表的记录应不分先后，因此在 Visual FoxPro 9.0 中，它已被 APPEND 或 INSERT_SQL 命令所取代。

4.3.4　修改记录数据

在数据录入过程中，难免会出现数据录入有误，或者当情况发生变化记录数据已经过时，这时就需要对记录数据进行修改。

1. 记录的命令修改法

(1)全屏幕编辑修改

① 命令格式

EDIT [FIELDS *FieldList*] [*Scope*]
[FOR *lExpression*1] [WHILE *lExpression*2]
[FONT *cFontName*[, *nFontSize*]] [STYLE *cFontStyle*]
[FREEZE *FieldName*]

② 功能

显示并修改记录。

③ 参数和子句说明

• FIELDS *FieldList*:要显示和修改的字段列表,缺省时为全部字段。

• *Scope*、FOR *lExpression*1、WHILE *lExpression*2 :全部缺省时,表示从当前记录开始进修显示并修改。

• FREEZE *FieldName*:指定在编辑窗口将 *FieldName* 所给出的一个字段设置为修改方式,而其他的字段此时处于只读状态。

例 4.25 对表"xsqkb.dbf"从第 5 条记录起进行编辑修改。要求以"楷体"、12 磅显示,"政治面目"字段可修改,其他字段只读。

```
USE xsqkb
GO 5
EDIT FONT "楷体",12 FREEZE 政治面貌
```

结果如图 4-17 所示。

图 4-17 EDIT 全屏幕编辑窗

(2) 浏览修改

BROWSE 命令具有浏览修改功能,其功能与用途非常广泛,在下一节详细讲解。

(3) 替换修改

EDIT 和 BROWSE 命令对于表中没有规律的随机修改是非常有用的。但在表的操作中,经常会遇到规律性很强的修改,例如职工住房公积金的扣除为职工工资的一定比例。此时再使用它们就显得非常慢且麻烦,而最有效的方法是使用替换命令 REPLACE。

① 命令格式

```
REPLACE FieldName1 WITH Expression1 [ADDITIVE]
    [, FieldName2 WITH Expression2 [ADDITIVE]] ...
    [Scope] [FOR lExpression1] [WHILE lExpression2]
```

② 功能

在指定"范围"内对指定字段用"表达式"的值成批地进行修改。

③ 参数和子句说明

• *Scope*、FOR *lExpression*1、WHILE *lExpression*2:当它们全部缺省时,表示仅修改当前

一条记录；

• $FieldName_n$ WITH $Expression_n$：用表达式 $Expression_n$ 的值修改字段 $FieldName_n$ 的值，(n=1,2,…,N)。

•[ADDITIVE]，仅对备注型字段有效，选之，则将表达式的内容追加到备注字段的原内容之后，如不选则替换掉原来的内容。

例 4.26　在表“xszhjfb.dbf”中，素质总分＝精神文明＊0.5＋社会活动＊0.2＋体育锻炼＊0.3，根据表中给出的数据，求每个学生的素质总分。命令序列如下：

```
USE xszhjfb
REPLACE ALL 素质总分 WITH 精神文明*0.5+社会活动*0.2+体育锻炼*0.3
LIST
```

记录号	学号	精神文明	社会活动	体育锻炼	素质总分	素质排名	综合积分	综合排名
1	010001	89.0	90.0	76.0	85.3	0	0.0	0
2	010102	90.0	91.0	88.0	89.6	0	0.0	0
3	010111	87.0	79.0	95.0	87.8	0	0.0	0
4	011216	76.0	85.0	79.0	78.7	0	0.0	0
5	011217	92.0	88.0	91.0	90.9	0	0.0	0
6	011320	96.0	85.0	79.0	88.7	0	0.0	0
7	012001	87.0	91.0	85.0	87.2	0	0.0	0
8	012002	85.0	79.0	84.0	83.5	0	0.0	0
9	012003	90.0	87.0	90.0	89.4	0	0.0	0
10	012234	96.0	88.0	92.0	93.2	0	0.0	0

可以看到，所有学生的“素质总分”字段全部按统一计算公式替换为现在的值。

(4) 用 SQL 修改命令修改记录

① 命令格式

```
UPDATE TableName
    SET FieldName1 = Expression1 [, Field_Name2 = Expression2 ...]
    [WHERE FilterCondition1 [AND | OR FilterCondition2 ...]
```

② 功能

对目标表指定的字段进行修改。

③ 参数和子句说明

• $TableName$：指定要修改的表名。注意这里的表并不一定是已经打开的表。

• SET $FieldName1$ =$Expression1$ [, $FieldName2$ = $Expression2$ ...]：用表达式 $Expression_n$ 的值取代字段 $FieldName_n$ 的原值(n=1,2,…,N)。

•[WHERE $FilterCondition1$ [AND | OR $FilterCondition2$ ...]]：记录替换条件。如果缺省该子句，则表中的所有记录的有关字段将被用 SET 子句中的表达式修改。

例 4.27　利用 UPDATE 语句重做例 4.26。

```
UPDATE xszhjfb SET 综合积分=(精神文明*0.5+社会活动*0.2+ 体育锻炼*0.3)
```

2. 记录的菜单修改法

对于表记录通过菜单操作也可方便地完成修改。但仅能完成对当前打开的表的修改。

(1) 菜单方式的编辑修改

菜单方式下实现表的编辑修改的步骤是:→[显示]→[编辑],此时将进入如图 4-17 所示的全屏幕编辑状态。

(2) 菜单方式的浏览修改

菜单方式下实现表的浏览修改的步骤是:→[显示]→[浏览]。

(3) 菜单方式的替换修改

菜单方式下实现表的浏览修改的步骤是:

S1:→[显示]→[浏览]→[表]→[替换字段]↓ 替换字段,如图 4-18 所示。

S2:在字段列表框选择要修改的字段;在替换为对话框输入替换表达式,或单击右侧的 … 调出表达式生成器生成替换表达式,如图 4-19 所示。

图 4-18 替换字段对话框

图 4-19 表达式生成器对话框

S3:在替换条件组合框中,分别选择使用范围、For、While 替换条件(也可通过表达式生成器生成替换条件)。

S4:→[替换]

4.3.5 浏览窗口的使用

浏览窗口命令 BROWSE 的功能非常齐全,它集显示、修改、查询、删除、增加记录等功能于一体。它是 Visual FoxPro 9.0 交互式命令中最常用的命令之一,而且利用系统菜单对数据表的显示也常采用的是浏览窗口。和 EDIT 命令一样,BROWSE 含有大量的参数和子句,下面的命令中仅给出了最常用的部分子句。

1. BROWSE 命令的常用格式

① 命令格式

BROWSE [FIELDS *FieldList*]
[FONT *cFontName* [, *nFontSize*]] [STYLE *cFontStyle*]
[FOR *lExpression*1] [FREEZE *FieldName*]
[LOCK *nNumberOfFields*]

[NOAPPEND] [NODELETE] [NOEDIT | NOMODIFY]
[TITLE *cTitleText*]
[VALID [:F] *lExpression2* [ERROR *cMessageText*]]
[WHEN *lExpression2*]

② 功能

对当前表进行浏览。

③ 子句和参数说明

• FIELDS *FieldList*、FONT *cFontName* [, *nFontSize*]、STYLE *cFontStyle*、FREEZE *FieldName*：意义同于 EDIT 中的子句；

• FOR *lExpression1*：仅浏览满足 *lExpression* 的记录。

• LOCK *nNumberOfFields*：指定一个在浏览窗口左边总能看到的字段的编号。

• NOAPPEND：表示不允许使用^Y 组合键向表文件中添加记录，无此可选项时则允许使用^Y 添加记录。

• NODELETE：表示不允许使用^T 组合键给表文件中记录打删除标记，无此可选项时则允许。

• NOEDIT|NOMODIFY：作用相同，表示不允许编辑浏览窗口中的记录，无此可选项时则允许。

• TITLE *cTitleText*：表示以 *cTitleText* 的内容作为标题在浏览窗口标题栏显示，无此可选项时，浏览窗口标题栏显示打开的表文件名。

• VALID [:F] *lExpression* [ERROR *cMessageText*]：表示对修改进行有效性验证。验证过程为：当表达式 *lExpression* 为真时，光标可以移向其他记录；否则只能停在原记录上。(此时若有[ERROR *cMessageText*]短语，则显示用户指定的出错信息；若无，则显示系统出错信息“Invalid Input”)。若 VALID 选择了[:F]可选项，则即使记录未被修改，当光标移向其他记录时，也要进行有效验证。

• WHEN *lExpression2*：当光标从一个记录移向其他记录后，系统自动判断 *lExpression2*]是否为真。若为真，才可修改当前记录；若为假，不能修改当前记录。

例 4.28　在浏览窗口中显示表文件 xsqkb.dbf 中所有男性党员的学号、姓名、家庭住址、奖惩情况，但只允许修改奖惩情况字段值，并将学号字段总显示在浏览窗口的左边。

```
USE xsqkb
BROWSE FIELDS 姓名,家庭住址,奖惩情况 ;
    FOR 性别='男' .AND. 政治面貌='党' FREEZE 奖惩情况 LOCK 1
```

2. 浏览窗口的外观

BROWSE 窗口如图 4-20 所示。当前打开的表文件 xsqkb.dbf 在浏览窗口中。可以看出，它是以表格形式显示表内容的。在 BROWSE 窗口，字段名被放在表的第一行，每一条记录占一行，记录与记录之间、字段与字段之间有网格隔开，这是常用模式。它类似于 Windows 窗口，由六部分组成。

• 表的标题(Title)：默认为数据表名，可以在命令中通过 TITLE 设置。

• 字段标题(Head)：默认为字段名，可以在命令中通过：H=""设置。

• 记录指针：指向当前记录的位置。

图 4-20 BROWSE 窗口

- 删除标记栏:当某条记录被逻辑删除后,该栏显示黑方框。
- 水平(垂直)滚动条:当表的内容在窗口中显示不下时,会出现相应的滚动条。
- 表记录。

3. 定制浏览窗口

所谓定制浏览窗口是指设置浏览窗口的有关显示参数。

(1)调整字段显示宽度

将鼠标指向两字段之间的调节线,当变成左右双向箭头时,按下鼠标左键进行左右拖动,即可调节字段的显示宽度。

(2)调整字段显示高度

将鼠标指向两记录之间的调节线,当变成上下双向箭头时,按下鼠标左键进行上下拖动,即可调节字段的显示高度。

(3)调整字段显示顺序

将鼠标悬停在要调整显示顺序的字段名上,当变成向下的粗黑箭头时,按下鼠标左键进行左右拖动,到合适位置时释放。

(4)拆分显示窗口

当表的字段较多,一个屏幕显示困难左右移动极不方便时,可将浏览窗口划分为两个称为"窗格"的两个小窗口,同时显示表的左右两端的部分字段。具体拆分步骤如下:

S1:将鼠标指向位于浏览窗口下边的水平滚动条左左的拆分条(矩形黑框)上,使光标变成带有三条竖线的双向箭头。

S2:按下鼠标左键,并向右拖动到适当位置,释放鼠标即可。拆分结果如图 4-21 所示。

当浏览窗口被分为两个窗格后,每个窗格的宽度可自由调节,并可在"浏览"和"编辑"窗格间相互切换。

在默认情况下,这两个窗格是相互关联的,当一个窗格的记录指针移动时,另一个窗格的记录指针也会同步移动到同一记录上。

图 4－21　浏览窗口的拆分

这种窗格间的关联可以通过取消“表”菜单中的“链接分区”命令的选中状态来中断，从而变成两个相对独立的窗格。

4.3.6　记录的删除

对表中无用的记录应予以及时删除。为了防止误删除有用记录，Visual FoxPro 9.0 把删除记录的操作分为两步进行。首先，在待删除的记录上加一个删除标记（称为逻辑删除）；然后，再将带有删除标记的记录从表文件中真正删除（称为物理删除）。对于带有删除标记的记录，可以去掉删除标记而恢复原样。

1. 记录的逻辑删除

（1）命令格式

DELETE [*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]

（2）功能

为表中指定范围内满足条件的记录加上删除标记。

（3）参数和子句说明

所有参数和子句意义均与前面其他命令中一致，当 *Scope*、FOR *lExpression*1、WHILE *lExpression*2]全缺省时，仅逻辑删除当前一条记录。

（4）关于删除标记

Visual FoxPro 9.0 在使用 LIST 或 DISPLAY 命令显示表记录时，以“＊”作为记录的删除标记；在 BROWSE 窗口以一个小黑方框作为删除标记。该标记位于记录号与第一字段之间，占用一个字符位置。

注意　加了删除标记的的记录能否继续参与数据库的操作，还将取决于 SET DELETED OFF|ON 命令的状态。

2. SET DELETE 命令

SET DELETE 命令用来指定 VFP 在进行操作时，是否对加了删除标记的记录，继续进行。

（1）命令格式

SET DELETE OFF|ON

(2) 参数说明

• ON:表示对于加了删除标记的记录已成为无效记录。对记录进行操作时,包括关联表中的记录,都不操作。

• OFF:表示对于加了删除标记的记录可同其他记录一样被操作。OFF 是系统给出的缺省状态。

例 4.29 分析 SET DELETE OFF|ON 的作用。

```
USE xsqkb
SET DELETE OFF
DELETE ALL FOR 性别='男'    && 删除所有男同学记录
LIST                        && 显示所有记录,有删除标记的记录也被显示出来
```

记录号	学号	姓名	性别	出生日期	政治面貌	应往届生	家庭住址	奖惩情况	照片
1	*010001	张凯华	男	1980.02.13	团	.T.	渭南市站南路 24 号	memo	Gen
2	010102	李会琴	女	1979.07.25	党	.T.	杨凌区西农路 59 号	memo	Gen
3	010111	李小茜	女	1979.12.21	团	.F.	宝鸡市红旗路 103 号	memo	Gen
4	011216	宋秀兰	女	1980.10.31	团	.F.	延安市枣园大街 5 号	memo	Gen
5	*011217	郭正宏	男	1979.11.25	团	.T.	渭南市东风街 106 号	memo	Gen
6	011320	姜亚男	女	1980.09.30		.F.	咸阳市世纪大街 108 号	memo	Gen
7	012001	杨书敏	女	1980.08.18		.T.	延安市宝塔路 214 号	memo	Gen
8	*012002	宋越辉	男	1979.06.09	团	.T.	宝鸡市红旗路 265 号	memo	Gen
9	*012003	杜拥军	男	1980.05.20		.F.	杨凌区西农路 132 号	memo	Gen
10	*012234	王向东	男	1979.04.26	党	.T.	咸阳市阳街 10 号	memo	Gen

```
SET DELETE ON
LIST      && 带有删除标记的记录被忽略
```

记录号	学号	姓名	性别	出生日期	政治面貌	应往届生	家庭住址	奖惩情况	照片
2	010102	李会琴	女	1979.07.25	党	.T.	杨凌区西农路 59 号	memo	Gen
3	010111	李小茜	女	1979.12.21	团	.F.	宝鸡市红旗路 103 号	memo	Gen
4	011216	宋秀兰	女	1980.10.31	团	.F.	延安市枣园大街 5 号	memo	Gen
6	011320	姜亚男	女	1980.09.30		.F.	咸阳市世纪大街 108 号	memo	Gen
7	012001	杨书敏	女	1980.08.18		.T.	延安市宝塔路 214 号	memo	Gen

注意 如果一条操作命令中范围的缺省值是当前记录或范围是以 RECORD *n* 形式指定的某一条记录,则 SET DELETE 将被忽略。另外,SET DELETE 命令对于 INDEX 和 REINDEX 命令及 RECCOUNT()函数不起作用。

3. 利用 SQL 命令删除表记录

(1)命令格式

```
DELETE FROM TableName
    [WHERE FilterCondition1 [AND | OR FilterCondition2 ...]]
```

(2)功能

对指定的表中满足条件的记录进行逻辑删除。

(3)参数和子句说明

• FROM *TableName*:指定要进行记录逻辑删除的表名。

• [WHERE *FilterCondition*1 [AND | OR *FilterCondition*2 ...]]:指定删除条件,缺省时为全部记录。

例 4.30　将表"xsqkb.dbf"中 1980 年前出生的男学生全部逻辑删除,浏览删除结果。

```
DELETE FROM xsqkb WHERE YEAR(出生日期)< 1980 AND 性别="男"
CLOSE ALL
USE xsqkb
BROWSE        && 结果如图 4-22 所示。
```

图 4-22　SQL 逻辑删除记录结果

4. 逻辑删除记录的恢复

逻辑删除记录的恢复是指将逻辑删除记录的删除标记取消,使它重新变为正常记录。

(1)命令格式

RECALL [*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]

(2)功能

用于表中指定范围内满足条件的记录的删除标志取消。

(3)参数和子句说明

所有参数和子句意义均与 DELETE 命令中一致,当 *Scope*、FOR *lExpression*1、WHILE *lExpression*2 全缺省时,仅恢复当前一条记录。

例 4.31　将表"xsqkb.dbf"中逻辑删除的记录恢复。

```
USE xsqkb
RECALL ALL
BROWSE        && 可以看到所有做了删除标记的记录都被恢复
```

5. 物理删除记录

物理删除记录指将加了删除标记的记录,在确定的确需要删除时,从表文件中剔除。物理删除又称为永久删除。

(1)命令格式

PACK [MEMO | DBF] [*Tablename*]

(2)功能

将所有带有删除标记的记录从指定的表文件中剔除,然后重新调整记录号。

(3)参数和子句说明

• MEMO:从备注文件中删除未使用的空间,但并不删除加了删除标记的记录。缺省时删除加了删除标记的记录。

• DBF:删除加了删除标记的记录但不影响备注文件。缺省时,对备注文件有影响,将在记录删除的同时删除与该记录有关的备注文件的内容。

• *Tablename*:指定要永久删除记录的表名。VFP 打开该表,然后永久删除加了删除标记的记录,再将表文件关闭。

例 4.32 物理删除记录示例。

```
USE xsqkb
COPY TO ls1
DELETE FROM ls1 WHERE 性别="男"
PACK
LIST
```

记录号	学号	姓名	性别	出生日期	政治面貌	应往届生	家庭住址	奖惩情况	照片
1	010102	李会琴	女	1979.07.25	党	.T.	杨凌区西农路 59 号	Memo	Gen
2	010111	李小茜	女	1979.12.21	团	.F.	宝鸡市红旗路 103 号	memo	Gen
3	011216	宋秀兰	女	1980.10.31	团	.F.	延安市枣园大街 5 号	memo	Gen
4	011320	姜亚男	女	1980.09.30		.F.	咸阳市世纪大街 108 号	memo	Gen
5	012001	杨书敏	女	1980.08.18		.T.	延安市宝塔路 214 号	memo	Gen

6. 清空表记录内容

(1)命令格式

ZAP [IN *nWorkArea* | *cTableAlias*]

(2)功能

物理删除指定的表文件的全部记录,从而使得它成为只有表结构而没有任何表记录的空表文件。

(3)说明

子句指明要进行清表操作的其他工作区或表的别名,缺省为当前表。用 ZAP 删除后的数据将无法恢复,因此操作时要谨慎。为了防止误清除表记录,Visual FoxPro 9.0 在安全保护 SET SAFETY ON(这是缺省状态)的环境下,将会给出一个提示信息框,询问用户是否真的要删除,缺省按钮为"否",从而给用户留出了一个挽救的机会。如果设 SET SAFETY OFF,则不给出任何提示就执行 ZAP 命令。

例 4.33 删除 xszhjf1 表文件的全部记录。

```
USE xszhjf1
ZAP
```

此时将弹出如图 4-23 所示的系统提示信息对话框。单击[是]。在继续执行下面的命

令。

图 4－23　系统提示信息

LIST

LIST STRUCTRUE

可以看到，只剩下一个空表结构了。

7. 菜单方式下的记录删除与恢复操作

菜单方式下对当前表记录的删除和恢复操作可以通过“浏览”窗口或“表”菜单的交互操作而实现。

(1) 利用“浏览”窗口删除或恢复表记录

S1：以独占方式打开表，打开浏览窗口。

S2：将光标移动到要删除的记录前边的删除标记列，单击鼠标，则该记录的删除标记列由空白框变为黑框，表示该记录已被加上了删除标记。

注意　恢复操作与删除类似，将光标移到要恢复的记录的删除标记列，单击鼠标，使该列由黑变白。

(2) 利用“表”菜单的“切换删除标记”选项删除或恢复表记录

此项操作与浏览窗口对记录的逻辑删除与恢复相类似。步骤如下：

S1：以独占方式打开表，然后打开浏览窗口，最后打开“表”菜单。

S2：将光标移动到要删除或恢复的记录上。

S3：交替单击“表”菜单的[切换删除标记]，即可实现记录的逻辑删除或恢复。

(3) 利用“表”菜单的“删除记录”选项删除表记录

本菜单操作可生成命令窗口 VFP 的 DELETE 命令，步骤如下：

S1：以独占方式打开表，然后打开浏览窗口，最后打开“表”菜单。

S2：→[删除记录]↓ 删除。

S3：后面的操作与记录追加的步骤相同。

(4) 利用“表”菜单的“恢复记录”选项恢复表记录

本菜单操作可生成命令窗口 VFP 的 RECALL 命令，步骤如下：

S1：以独占方式打开表，然后打开浏览窗口，最后打开“表”菜单。

S2：→[恢复记录]↓ 恢复。

S3：后面的操作与记录追加的步骤相同。

(5) 利用“表”菜单的“彻底删除”选项物理删除表记录

本操作将在命令窗口生成一个 VFP 的“PACK”命令，单击“表”菜单中的“彻底删除”，则

会将当前表中所有逻辑删除的记录永久删掉。

4.3.7 表与内存变量间的数据交换

表与内存之间的数据交换分为表的当前记录与一维数组间或内存变量组之间的数据交换和表的多条记录与二维数组之间的数据交换两种。不管是哪种交换,都是以表作为参考点的。把当前记录传递给内存变量(内存变量组、一维数组)称为发送,反之称为收集;把多条记录传递给二维数组称为复制,反之称为数组追加。

表和内存变量之间的数据交换是通过有关的 Visual FoxPro 9.0 的命令实现的。这种操作对于通用型字段无效。对于备注型字段的操作则有一定的限制。

1. 内存变量组、一维数组与表当前记录间的数据交换

(1)向内存变量组发送数据

① 命令格式

SCATTER [FIELDS *FieldNameList* | FIELDS LIKE *Skeleton*
| FIELDS EXCEPT *Skeleton*] [MEMO]
TO *ArrayName* | TO *ArrayName* BLANK| MEMVAR | NAME *Object-Name* [ADDITIVE]

② 功能

将当前表的当前记录发送到一个内存变量组、一维数组或一个 VFP 对象中。

③ 参数和子句说明

· FIELDS *FieldNameList*:要发送到数组或内存变量组的字段列表。

· FIELDS LIKE *Skeleton*| FIELDS EXCEPT *Skeleton*:把像或除带有用通配符表示的字段发送到内存变量组或数组。当 FIELDS 子句的三种形式都缺省时表示将记录的所有字段(除备注型)都发送。

· MEMO:指定字段列表中含有备注型字段,缺省时为不包含。

· TO *ArrayName* | TO *ArrayName* BLANK:表示发送到数组。若含有 BLANK 关键字,则表示生成一个元素个数与表的字段个数相等、类型对应相同,但值为空的数组。

· MEMVAR:表示将字段发送到内存变量组中。

注意 MEMVAR 前面不能带关键字 TO,否则将生成一个名字为 MEMVAR 的数组。

· NAME *ObjectName*,表示创建一个对象,对象的属性与表中的字段同名,属性的值即为字段的值。该子句对于表中备注型和通用型字段不能创建属性。

④ 注意事项

如果数组事先未声明,则系统会自动生成一个与发送的字段个数相等的一维数组。如果事先已声明,则当数组的元素个数等于要发送的字段个数时,发送的字段值刚好覆盖数组的对应各元素;当数组的元素个数小于要发送的字段个数时,多余的字段将不发送;当数组的元素个数大于要发送的字段个数时,多余的元素将保持原值不变。因此建议这里最好不要先声明数组而是直接由系统生成更好。

例 4.34 将"xsqkb.dbf"中的第三条记录传递给数组"jlsz"。

```
CLEAR MEMORY
CLEAR
```

```
USE xsqkb
GO 3
SCATTER TO jlsz
DISPLAY MEMORY
```

结果如图 4－24 **所示。**

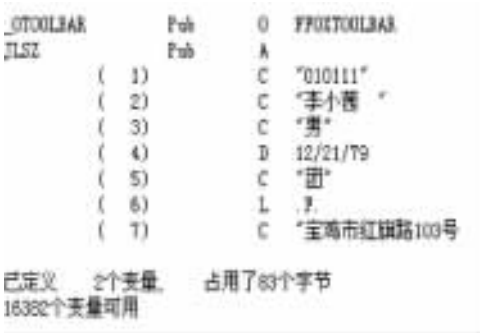


图 4－24　**将记录发送给数组**

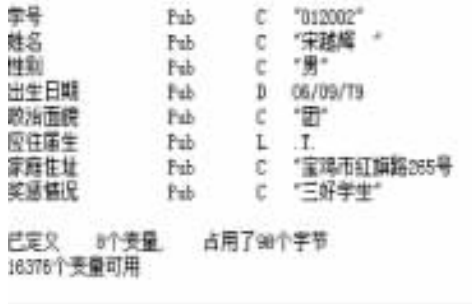


图 4－25　**将记录发送给内存变量组**

例 4.35　将“xsqkb.dbf**”中的第八条记录除照片字段外，都传递给内存变量组。**

```
CLEAR
USE xsqkb
GO 8
SCATTER MEMVAR MEMO
DISPLAY MEMORY
```

结果如图 4－25 **所示。**

(2)**从内存变量组、一维数组收集数据**

① **命令格式**

GATHER FROM *ArrayName* | MEMVAR | NAME *ObjectName*
[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]
[MEMO]

② **功能**

用选定的数组、内存变量组、VFP **对象来替换当前表的当前记录的字段。**

③ **参数和子句说明**

· FROM *ArrayName*，**指明数据的来源为数组。在无** FIELDS **短语的情况下，数组的第一个元素替换当前记录的第一个字段；第二个元素替换第二个字段，以此类推。如果数组的元素比字段少，则剩余的字段保持原值不变；如果数组的元素个数比字段多，则剩余的元素略去。**

· MEMVAR，**数据来源是内存变量组，在无** FIELDS **短语的情况下，系统将用内存变量组中与表字段同名的变量值替换字段值；如有** FIELDS **短语，则替换该短语给出的同名同类型字段。**MEMVAR **前面不能有** FROM **关键字，否则系统认为是从数组** MEMVAR **中收集数据。**

· NAME *ObjectName*，用 VFP 对象的属性值替换对应的字段值。

· FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*，同前。

· MEMO，选之替换备注字段，否则并不替换。

注意 数据的发送，实质上是向内存变量的赋值过程，数据类型由所赋的字段类型而确定。但收集数据时，其实质是将内存变量替换记录字段的过程，必须要求内存变量与对应的字段类型一致方可实现。

例 4.36 将“xsqkb.dbf”的结构复制成表“xsqk.dbf”的结构，然后将例 4.34、4.35 所得的内存变量组和数组的值追加成“xsqk.dbf”的记录并显示。结果如图 4-26 所示。

图 4-26 由数组、内存变量组收集数据为表记录

```
USE xsqkb
COPY STRU TO xsqk
USE xsqk
APPEND BLANK
GATHER FROM jlsz
APPEND BLANK
GATHER MEMVAR MEMO
BROWSE
```

2. 二维数组与表间的数据交换

表与二维数组之间的数据交换，任何情况下都不涉及备注型字段和通用型字段。

(1) 把成批记录复制给二维数组

① 命令格式

COPY TO ARRAY *ArrayName*
[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]
[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2] [NOOPTIMIZE]

② 功能

将当前表中，指定的范围内符合条件的记录，按字段子句所规定的字段，复制到二维数组中。

③ 参数和子句说明

· TO ARRAY *ArrayName*，指定将表的数据要复制的数组。

其他子句意义同前，当范围和条件子句都缺省时，指将全部记录复制成到数组中。

· 数组的每一行各存储一条记录，数组的每一列各存储记录的一个字段。对于每一条记录，数组的第一列存储第一个字段，数组的第二列存储第二个字段，以此类推。如果数组事先已经被声明，且声明的列数比表的字段多，则数组剩余的列元素保持原值不变；如果数组列数

比表的字段少，则剩余的字段不能保存到数组；如果数组的行数比表的记录多，则数组剩余的行元素保持原值不变；如果数组行数比表的记录少，则剩余的记录不能保存到数组。如果数组事先未声明而是直接使用，则生成一个和要复制的记录等行数、等字段数的二维数组。

例 4.37　将“xsqkb.dbf”的全部记录复制到数组 ewsz 中。

```
USE xsqkb
COPY TO ARRAY ewsz
```

(2) **由二维数组进行记录的成批追加**

① **命令格式**

APPEND FROM ARRAY *ArrayName* [FOR *lExpression*]
　　[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]

② **功能**

将数组的每一行追加成当前表的每一条记录。

③ **参数和子句说明**

- FROM ARRAY *ArrayName*：**要追加的数组。**
- FOR *lExpression*：**追加的条件，缺省时为数组的全部记录。**
- FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*：**意义同前，缺省时为全部字段。**

④ **注意事项**

从二维数组中，把符合条件的元素，按字段子句所规定的字段，依次追加表中，成为表的多条新记录。这里，数组的行数决定了要追加的记录条数，数组的列数则决定要追加的字段。如数组中有空行，则形成空记录；同行内有空元素，则形成空字段。如数组的列大于字段数，则多余列的元素不追加；列小于字段数，则多余的字段维持原值不变。当字段子句缺省时，指向所有字段(不含 M、G 型)都追加。条件、范围缺省时，指数组的全部行都追加成表的记录。和从一维数组收集数据为单条记录一样，这里也存在二维数组元素与对应的表字段类型一致的问题。

例 4.38　将表“xsqkb.dbf”的结构复制成表“xsqkb2.dbf”，然后将例 4.37 生成的数组追加到“xsqkb2.dbf”中。

```
COPY STRUCTURE TO xsqkb2
USE xsqkb2
APPEND FROM ARRAY ewsz
BROWSE
```

显示结果表明，表“xsqkb2.dbf”与表“xsqkb.dbf”记录完全相同。

4.4　表的排序与索引

一般情况下，表的记录是按物理次序显示的，这种方式对于记录的操作常常是不方便的。在许多时候，需要按记录的某种逻辑次序对表进行快速操作，这就要借助于表的排序或索引。

4.4.1 表的排序

排序是指根据表的某些字段值的大小,将表的记录次序重新排列从而生成一个新表。新表文件与原表文件相互独立。

(1)命令格式

SORT TO *TableName* ON *FieldName*1 [/A | /D] [/C]
[, *FieldName*2 [/A | /D] [/C] ...] [ASCENDING | DESCENDING]
[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]
[FIELDS *FieldNameList* | FIELDS LIKE *Skeleton*
| FIELDS EXCEPT *Skeleton*] [NOOPTIMIZE]

(2)功能

将当前表的记录排序生成有次序的新表文件。

(3)参数和子句说明

· *TableName*:指定要生成的新表的表名。

· ON *FieldName*1[/A | /D] [/C]:指定要排序的第一个关键字段的名字。后面的参数/A|/D分别表示升序和降序。无此可选项时为升序排序。/C表示不区分字段值的大小写,该参数只对包含有英文字母的字段起作用,且可以和/A或/D组合使用。

· *FieldName*2 [/A | /D] [/C]……:指定当第一个字段值相同时,要排序的第二个关键字段的名字。[/A|/D|/C]意义同上,……表示依此类推。

· ASCENDING:指定没有用/D参数标出的其他字段的排序方式是升序。

· DESCENDING:指定没有用/A参数标出的其他字段的排序方式是降序。当ASCENDING和DESCENDING都缺省时,指为升序。

· *Scope*、FOR *lExpression*1、WHILE *lExpression*2:意义同前,均缺省时指所有记录都排序。

· FIELDS *FieldNameList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*:意义同前,指定生成的新表中所包含的字段。缺省时,新表与原表结构相同。

例 4.39 对 xscjb.dbf 表按成绩合计降序进行排序。

```
USE xscjb
SORT ON 总分/D TO mcb FIELDS 学号,总分
USE mcb
LIST
```

显示结果:

记录号	学号	总分
1	010101	543.0
2	012001	492.0
3	012002	468.0
4	012234	462.0

5	012003	455.0
6	011130	448.0
7	010111	430.0
8	010102	425.0
9	011217	413.0
10	011216	410.0

4.4.2　索引概述

从排序过程可以看出，通过排序能产生用户所需要的新的有序的表文件。也正因为每一次排序都要产生新的表文件，所以会造成结果的数据冗余，占用大量的磁盘空间。而且由于生成的各数据表彼此独立，当修改原表文件数据时，必须对每个排序表文件重新排序，否则就会造成各表数据不一致。而数据表的索引正好可以克服这两个缺点。

1. 索引的概念

按照某个关键表达式的值，对表进行逻辑排序，称为索引。这里关键表达式可以是字段名、字段的组合、记录号函数等。

索引不改变记录的物理顺序，但要生成一个排序文件或排序标记。在排序文件或排序标记中，仅有两个字段：表达式值和一个指向表的对应记录的指针。这显然把数据冗余压缩到了最小程度。可是索引文件不能单独打开，它必须在表的打开后才起作用。

表以索引形式打开后，记录将按逻辑次序显示并操作；编辑原表的记录时，索引文件会自动更新；在数据库中，还可以建立库表之间的相互永久关联，实现表的一致性，所以索引的应用比排序要广泛。

2. 索引文件的类型

按索引文件所允许包含的索引个数，将索引文件分为两大类：单索引和复合索引。复合索引又分为结构复合索引和非结构复合索引。

(1)单索引文件

所谓单索引文件，指一个索引文件只允许包含一个索引，且扩展名是“.IDX”的索引文件。它的特点是：

① 可以与表同时打开，也可以在表先打开的情况下，独立打开它。

② 索引次序仅有升序。

③ 存储形式既可以是压缩形式，也可以是非压缩形式。非压缩形式是为了和 FoxBase＋兼容而设置的，而压缩形式的存储量小且访问速度快，然而一旦使用压缩形式，则在 FoxBase＋下不能使用此索引文件。

在有些书中也把单索引文件称为独立索引文件。

(2)复合索引文件

所谓复合索引文件指同一个索引文件可以包含多个索引。其中的每一个索引称为复合索引文件的一个标记，它等价于一个单索引。复合索引文件的扩展名是“.CDX”。它只以压缩形

式存放，FoxBase+ 无复合索引文件。

根据是否与表同名，将复合索引文件又分为结构复合索引文件和非结构复合索引文件。

结构复合索引文件指与表同名的复合索引文件。它和表的备注文件一样随着表打开和关闭而自动打开和关闭。凡是利用表设计器创建的索引一定是结构复合索引文件的标记。

非结构复合索引文件指与表不同名的复合索引文件。它不随着表打开和关闭而自动打开和关闭。有些书中也称非结构复合索引文件为独立复合索引文件。

3. 索引的类型

索引文件和索引的类型是两个不同的概念。索引的类型是指将索引按索引关键字的值是否允许重复及表允许索引个数而划分的。在 Visual FoxPro 9.0 中有五种类型的索引。但在库表中最常用的索引有三种：主索引、候选索引和普通索引；在自由表中最常用的索引有两种：候选索引、普通索引。如果建立索引的表达式是字段，也称之为索引关键字。

(1)**主索引**(primary index)

索引关键表达式可惟一标识每一个记录的索引称为主索引。即主索引不得有重复关键表达式的值。当用户没有指定其他索引为控主索引时，主索引被解释为主控索引。只有数据库表才可以建立主索引，且每一个表仅允许建立一个主索引。

注意　在数据库内部，主索引是表的重要部分。如果将库表移出为自由表，则主索引将被删除。如果要建立主索引的字段含有重复值，则 Visual FoxPro 9.0 给出一个错误信息。

如果建立结构复合索引中的主索引的表达式是一个字段，则该字段称为该表的主关键字(primary key)，也称为主码(primary code)。主索引用于建立一对多的表永久关联时，建立主表，它可以为表的引用创建参照完整性。

表 4-5　Visual FoxPro 9.0 的索引类型

索引类型	描　述	允许个数
主索引	指定的字段或表达式不得有重复值	每表一个
候选索引	指定的字段或表达式不得有重复值	每表多个
普通索引	指定的字段或表达式允许有重复值	每表多个
二进制索引	根据一个有效的非空逻辑表达式索引记录	每表多个
惟一索引	为了和老版本兼容而设置，根据指定关键表达式值第一次出现的记录选择一个子集	每表多个

(2)**候选索引**(candidate index)

和主索引类似，候选索引关键表达式也可惟一标识每一个记录。但不管是库表还是自由表都可以建立候选索引，且一个表允许建立多个主索引。在特定环境下(例如自由表)候选索引可视为主索引。

如果表中有相同的关键表达式值，则不能用该关键表达式创建主索引或候选索引。建立候选索引的表达式如果是一个字段，则也称之为候选关键字或候选码。

(3)普通索引(regular index)

普通索引的含义是把每个记录的关键表达式的值都存入索引文件中。如果记录的关键表达式值相同,则可以重复存储,并用独立的指针指向各条记录。

在普通索引中,索引关键表达式指按次序排列,对于关键表达式相同的记录,则按原记录录入的物理次序排列在一起。建立普通索引的表达式如果是一个字段,则也称之为普通关键字或普通码。

(4)二进制索引(binary index)

二进制或位图索引是根据一个有效且非空的逻辑表达式创建的索引。例如,标记删除的记录,它支持自由表和库表。但是,二进制索引不支持下列情况:

① 使用带有 Null 值的索引表达式。

② 过滤条件含有 For 子句。

③ 改变记录显示和处理次序的关键字,例如:ASCENDING、DESCENDING、UNIQUE、CANDIDATE。

④ 设置二进制索引为主控索引,例如 SET ORDER 命令。

⑤ 进行排序和索引查找操作。

二进制索引比非二进制索引要小并且可以改进索引维护的速度。

(5)惟一索引(unique index)

惟一索引是为了保持和早期的版本相兼容而设置的。它对于关键表达式的值相同的所有记录,仅将它们其中的第一条记录存入索引文件中,从而达到避免显示或访问关键表达式值相同的所有记录。对于添加到表中的记录,如果与表中原有记录的索引关键表达式值相同,则它不会被包含在索引文件中。

4.4.3　创建索引

索引的创建分为菜单方法创建和命令方法创建。

1. 菜单方法创建索引

使用菜单方法创建索引,是通过表设计器进行的,因此它创建的一定是结构复合索引文件的标记。根据创建索引的关键字是关键字还是表达式,创建的方法略有不同。

(1)创建单字段普通索引标记

创建单字段普通索引标记的步骤为:

S1:打开表(如果是数据库表,则先打开数据库,再打开表);

S2:打开表设计器,选要作为索引关键字的字段;

S3:→[字段]→[索引]↓ 索引;

S4:选索引次序;

S5:→[确定]。

例 4.40　对“xsqkb.dbf”根据“出生日期”字段,创建一个降序的普通索引。

```
USE xsqkb
MODIFY STRUCTURE
```

在表设计器窗口选索引关键字段“出生日期”,在[索引]列表框中选“↓”,结果如图 4-27

所示。最后单击[确定]。

图 4-27 创建单字段普通索引

(2)创建其他索引

创建其他索引的步骤为:

S1、S2:与创建单字段普通索引相同。

S3:→[索引] ↓ 索引。

S4:填写或选择有关选项卡:排序、索引、类型、表达式、筛选、排序的值或表达式。

S5:→[确定]。

关于各选项卡说明如下:

· 排序(次序):切换按钮,确定排序的次序为升序还是降序,箭头向下为降序,向上为升序。

· 索引:指定要生成的索引标记的名称。缺省为与字段同名。

· 类型:列表框,指定索引的类型。缺省为普通索引。

· 表达式:指定索引关键表达式,既可在其文本框直接写入,又可用表达式生成器创建。

· 筛选:指定筛选条件,既可直接在其文本框直接写入,又可用表达式生成器创建。缺省为所有的记录全部按关键表达式的值进行排序。

· 排序(方式):列表框,确定字符型表达式排序所使用的方式,该方式为:PinYin、Machine、Stroke 三种。

例 4.41 对"xsqkb.dbf"根据"学号"字段,对女学生创建一个降序的候选索引,索引标记"女生"。

打开表"xsqkb.dbf",打开表设计器,选[索引]选项卡,填写有关的选卡如图 4-28 所示。单击[确定]。

2. 命令方法创建索引

(1)命令格式

INDEX ON *Expression* TO *IDXFileName* | TAG *TagName* [BINARY]
[COLLATE *cCollateSequence*] [OF *CDXFileName*] [FOR *lExpression*]
[COMPACT] [ASCENDING | DESCENDING]
[UNIQUE | CANDIDATE] [ADDITIVE]

图 4-28　创建候选索引

(2)功能

建立单索引文件或增加索引标记。

(3)参数和子句说明

• *Expression*:指定索引关键表达式。

• TO *IDXFileName*:指定要建立单索引文件的文件名。

• TAG *TagName*:指定建立复合索引文件的索引标记,或增加索引标记。

• [BINARY],创建二进制索引。

• COLLATE *cCollateSequenc*:指定字符比较次序。缺省时为机器设置的缺省值。

• OF *CDXFileName*:指定索引标记所隶属的非结构复合索引文件的名称。若缺省则表示创建或在结构复合索引文件中添加新的索引标记。

• FOR *lExpression*:缺省时为全部记录。

• COMPACT:指定生成一个压缩的单索引文件。缺省表示不压缩。

• ASCENDING|DESCENDING:指定升序或降序索引,仅对复合索引有效。缺省为升序。

• UNIQUE|CANDIDATE:用于指定索引类型。前者表示惟一索引;后者表示候选索引,仅用于结构复合索引文件。缺省为普通索引。

• ADDITIVE:建立索引的同时不关闭前面打开的索引文件。缺省时表示在创建本索引的同时关闭除结构复合索引之外的其他所有索引。

(4)注意事项

当索引表达式为多字段共同索引时,必须将多个字段组成合理的有效表达式,一般组成字符串表达式。对于数值型字段,需用 STR()函数将数值型数据转换成字符串,对于日期型数据,需用 DTOC()函数将其转换成字符串,然后将它们用运算符"+"连接起来。

例 4.42　为表文件"xsqkb.dbf",根据"姓名"字段建立单索引文件"xm.idx",观察索引后的显示结果。

```
USE xsqkb
INDEX ON 姓名 TO xm
LIST FIELDS 学号,姓名,性别,出生日期 TO FILE xmsy
```

TYPE xmsy.txt

记录号	学号	姓名	性别	出生日期
9	012003	杜拥军	男	05/20/80
5	011217	郭正宏	女	11/25/79
6	011320	姜亚男	女	09/30/80
2	010102	李会琴	女	07/25/79
3	010111	李小茜	男	12/21/79
4	011216	宋秀兰	女	10/31/80
8	012002	宋越辉	男	06/09/79
10	012234	王向东	男	04/26/79
7	012001	杨书敏	女	08/18/80
1	010101	张凯华	男	02/13/80

例 4.43　为表文件xsqkb.dbf**建立结构复合索引文件。其中包含三个索引标记：**

- **以学号建立候选索引型索引标记：**xh；
- **对所有男生的出生日期以降序建立普通索引型索引标记：**sr；
- **以性别和姓名的机器码次序排列建一个升序的普通索引，索引标记：**xbxm。

```
USE xsqkb
INDEX ON 学号 TAG xh CANDIDATE
INDEX ON 出生日期 DESCENDING TAG sr FOR 性别='男'
INDEX ON 性别+姓名 TAG xbxm COLLATE "MACHINE"
```

4.4.4　打开与关闭索引

1. 主控索引

当用户为一个表创建了多个索引时，每个索引代表不同的记录集合和不同的显示及操作顺序。但在某一时刻，最多只能有一个索引对显示和操作次序起控制作用。这个索引叫主控索引。当打开多个索引时，通过设置主控索引，可以使用某个索引成为主控索引。有些书也把主控索引称为当前索引。

2. 索引文件的打开与关闭

索引文件的打开与关闭的方法因文件的类型不同而不尽相同。结构复合索引文件始终与表自动同时打开和关闭。单索引和非结构复合索引既可以与表同时打开和关闭，也可以在表文件打开之后单独打开和关闭。

在表文件打开之后打开索引文件的菜单方法与打开其他文件的方法相同，都是通过[打开]按钮进行的，无须赘述。下面介绍命令方式打开索引文件。

(1)**索引文件与表同时打开**

① **命令格式**

USE [[[*DatabaseName*!] *TableName* | ?][IN *nWorkArea* | *cTableAlias*]

[INDEX *IndexFileList* | ? [ORDER [*nIndexNumber* | *IDXFileName*
| [TAG] *TagName* [OF *CDXFileName*]
[ASCENDING | DESCENDING]] [ALIAS *cTableAlias*]
[EXCLUSIVE] [SHARED] [NOUPDATE]]

② 功能

打开表或关闭表的同时打开或关闭索引文件。

③ 参数和子句说明

• *DatabaseName*!:指定要打开表所隶属的数据库名。缺省时指当前库或自由表。注意,在数据库名的后面要跟一个"!",作为库名和后续表名的分隔符。

• *TableName*:指定由打开的表名。

• IN *nWorkArea* | *cTableAlias*:指定表打开的工作区号或工作区别名。

• INDEX *IndexFileList*:指明要和表同时打开的非结构复合索引和单索引文件的名称。缺省时如果事先已为表建立了结构复合索引文件则指仅打开它。如事先并未建立结构复合索引文件则指将表按普通方式打开。

• ORDER [*nIndexNumber* | *IDXFileName* | [TAG] *TagName* [OF *CDXFileName*]:指定主控索引。*nIndexNumber* 为索引的编号,*IDXFileName* 为单索引的名字,*TagName* 为结构复合索引的标记名称,*CDXFileName* 为非结构复合索引文件的名称。

• ASCENDING | DESCENDING:指明不管表的索引创建时是升序还是降序,这里都临时以升序或降序显示和操作。缺省时以创建索引时的次序为准。

• ALIAS *cTableAlias*:为表取一个别名。

④ 注意事项

关于索引的编号,Visual FoxPro 9.0 将所有打开的单索引和复合索引进行统一的编号,编号的次序是:表文件、单索引名、结构复合索引标记、非结构复合索引标记。单索引按在 INDEX *IndexFileList* 子句中的先后次序编号;索引标记按各索引在所属结构复合索引或非结构复合索引所建的先后次序编号。表 4-6 给出了编号的次序。但是许多时候,用户很难记清楚究竟有多少单索引被打开、复合索引中有多少个索引标记,各标记创建的次序,这就为使用索引编号确定主控索引带来麻烦。但是用户明白,希望把哪个单索引或复合索引的标记作为主控索引,为此使用 *nIndexFileList* 倒不如使用 *IDXFileName* 或[TAG] *TagName* [OF *CDXFileName*]子句来确定主控索引更方便。

表 4-6　索引的编号次序

文件类型	索引个数	索引编号
表文件		0
单索引文件	M	1,2,…,M
结构复合索引文件	N	M+1,M+2,…,M+N
非结构复合索引文件	K	(M+N)+1,(M+N)+2,…,(M+N)+K

例 4.44　将表"xsqkb.dbf"及其单索引文件"xm.idx"和结构复合索引文件"xsqkb.cdx"一并打开。

USE xsqkb INDEX xm

(2)索引文件的单独打开

在有些情况下,当表打开后需要打开一些尚未与表同时打开的非结构复合索引文件或单索引文件。此时,应使用的命令为 SET INDEX TO 。

① 命令格式

SET INDEX TO [*IndexFileList* | ?]

[ORDER [*nIndexNumber* | *IDXFileName* | [TAG] *TagName* [OF *CDXFileName*]

[ASCENDING | DESCENDING]][ADDITIVE]

② 功能

打开一个或多个与当前表有关的索引文件。

③ 参数和子句说明

本命令中的参数和子句,除 ADDITIVE 外均与 USE 命令中的意义相同。

ADDITIVE,指定在打开新的索引文件时,是否关闭除结构复合索引之外的原已打开的索引文件。有之,表示不关闭原来打开的索引文件,从而使新老索引同时起作用,否则表示关闭原来打开的索引。

例 4.45 为表"xszhjfb.dbf",首先以"学号"字段为索引关键字创建一个候选索引;分别以"精神文明"、"社会活动"、"体育锻炼"三个字段为索引关键字创建三个普通索引,它们分别为非结构复合索引文件"fjgfhsy.cdx"的三个标记;以"综合积分"为索引关键字创建一个单索引"zhjf.idx"。然后在重新打开表后,再分别打开非结构复合索引文件和单索引文件。

```
USE xszhjfb
INDEX ON 学号 TAG 学号 CANDIDATE
INDEX ON 精神文明 TAG jswm OF fjgfhsy
INDEX ON 社会活动 TAG shhd OF fjgfhsy
INDEX ON 体育锻炼 TAG tydl OF fjgfhsy
INDEX ON 综合积分 TO zhjf
CLOSE ALL
USE xszhjfb
SET INDEX TO zhjf,fjgfhsy
```

(3)索引的关闭

关闭索引文件,对于单索引文件和非结构复合索引文件,都要通过命令方式来进行。方法有:

① 方法 1:使用打开和关闭表文件命令:USE

由于索引文件是紧紧依赖表文件的,只有打开表文件它才能随表一起打开或单独打开。因此关闭表就可将与该表有关的所有复合索引文件和单索引文件关闭。其命令是:USE。

② 方法 2:使用打开和关闭索引命令:SET INDEX TO

当命令后带有 INDEX 子句时,表示打开表文件和结构复合索引、单索引文件,无 INDEX 子句时,表示关闭打开的所有与当前表有关的非结构复合索引文件和单索引文件。

③ 方法 3:使用专门的索引文件关闭命令:CLOSE INDEXES

此命令可关闭当前工作区打开的所有非结构复合索引文件和单索引文件。但并不关闭结

构复合索引文件。

3. 重新索引

如果索引文件与表没有打开，那么对表的修改将不会反映到索引中去，这样会造成表与索引文件的不一致。此时需要对索引进行重新索引，以取得索引与更新后的表文件数据的一致。重新索引非常简单，其步骤为：

S1：将所有与表有关的索引全部打开。

S2：使用命令：REINDEX

注意　第 2 步也可以通过菜单来实现，其方法是，在浏览方式打开表的前提下，打开[表]菜单，单击[重新建立索引]即可。

4.4.5　设置主控索引

在上一节中我们已介绍了主控索引的概念，本节介绍设置主控索引的方法。

1. 命令格式

SET ORDER TO [[*nIndexNumber* | IDXIndexFileName | [TAG] *TagName*
　[OF *CDXFileName*] [IN *nWorkArea* | *cTableAlias*]
　[ASCENDING | DESCENDING]]

2. 功能

将单索引或复合索引文件中指定索引标记设为主控索引或取消前面的主控索引。

3. 参数和子句说明

均与 USE 命令中的意义相同。如果在 SET ORDER TO 之后无参数和子句，表示取消先前的主控索引。

例 4.46　将例 4.45 中打开的结构复合索引文件中的索引标记“xm”设置为主控索引。

```
USE xszhjfb                    && 同时结构复合索引文件 xszhjfb.cdx 也被打开
SET ORDE TO TAG xm             && 将 xm 设为主控索引
```

4.4.6　索引查询

在表文件中查询满足条件的记录的过程叫查询。查询有两种方式：顺序查询和索引查询。

顺序查询又称记录定位，在前面讲述表记录指针的移动一节已介绍过。这种记录的定位操作，既适应于表以普通方式打开，又适应于表按索引方式打开。其最大特点是从表的第 1 条记录（物理的或逻辑的）开始，逐条向下查询，直到找到满足条件的第 1 条记录或确定表中在给出的范围内无此记录为止。这种查询最大缺点是，由于查询是逐记录进行的，所以定位速度较慢。设表有 N 条记录，则用顺序查询要找到满足条件的一条记录，平均要进行 INT((1+N)/2+0.5))次查询。例如对于一个有 1024 条记录的一个表，此法查询一条满足条件的记录的平均次数为 513 次。

索引查询在表以要查询的关键字为索引表达式创建了索引并打开了该索引，且该索引被设置为主控索引的前提下进行的。由于表此时的操作次序是按要查询的表达式的值升序或降序排列的，因此使用数学上的折半查询法来查询。折半查询的最多次数为 $\mathrm{Log}_2 N$ 次，对于

1024 条记录的表，最多查询 $Log_2 1024 = 10$ 次。可见，即使是最多查询次数，也比顺序查询要快出 50 余倍。数学上的这种折半查询法在 Visual FoxPro 9.0 中，称为索引查询，命令为 SEEK。

1. SEEK 命令

(1)命令格式

SEEK *Expression* [ORDER *nIndexNumber* | *IDXIndexFileName* | [TAG] *TagName*
[OF *CDXFileName*]
[ASCENDING | DESCENDING]]
[IN *nWorkArea* | *cTableAlias*]

(2)功能

查询关键字值与查询表达式值相匹配的记录，并将记录指针指向该记录。

(3)参数和子句说明

Expression：指定要查询的表达式，它必须是一个精确值。它可以是常量、变量、表达式、Null 值。除数值型外，其他类型必须要有各自的定界符。

其他子句和参数与前面各命令中的意义相同。记录是否找到，照样可以用 FOUND()、EOF()函数来判别。

例 4.47 在xsqkb.dbf中查询姓名为“张凯华”的学生，并显示其姓名、出生日期和奖惩情况。

```
USE xsqkb INDEX xm ORDER xm
SEEK "张凯华"
DISP 姓名,出生日期,奖惩情况
```

显示结果：

记录号	姓名	出生日期	奖惩情况
1	张凯华	02/13/80	三好学生

如果使用内存变量作为查询关键字，例 4.47 也可以写为：

```
USE xsqkb INDEX xm ORDER xm
Xm= "张凯华"
SEEK xm
DISP 姓名,出生日期,奖惩情况
```

由于表是以索引方式打开的，所以索引关键字值相同的记录已排列在了一起，因此当要查询满足条件的第二条记录时，只要使用 SKIP 命令即可。但此时不管下一条记录是否为满足条件的新记录，测试函数 FOUND()函数都保持着逻辑真不变，因此只能通过 DISPLAY 命令显示其记录来判断是否是满足条件的新记录了。

例 4.48 在xsqkb.dbf中查询姓名为“李会琴”的所有学生，并显示其姓名、出生日期和奖惩情况，分析显示结果。

```
USE xsqkb INDEX xm ORDER xm
Xm= "李会琴"
SEEK xm
```

```
? FOUND()                    && 显示为：.T.
DISP 姓名,出生日期,奖惩情况    && 2 李会琴 07/25/79 优秀学生干部
SKIP
? FOUND()                    && 仍显示为：.T. 仔细思索此处为何还显示.T.
DISP 姓名,出生日期,奖惩情况    && 3 李小茜 12/21/79 二等奖学金获得者
```

2. SEEK 函数

(1)**函数格式**

SEEK(*Expression* [, *nWorkArea* | *cTableAlias* [, *nIndexNumber* | *cIDXIndexFileName* | *cTagName*]])

(2)**功能**

在指定的工作区中查询索引关键字与表达式相匹配的第 1 条记录，如查到则返回.T. 并将记录指针指向该记录；否则返回.F. 并将记录指针指向表文件的尾部。

(3)**参数和子句说明**

- *Expression*：**要查询的表达式。**
- *nWorkArea* | *cTableAlias*：**指定查询的工作区或别名，缺省时为当前表。**
- *nIndexNumber* | *cIDXIndexFileName* | *cTagName*：**指定被查询的索引的标号、单索引文件名或结构复合索引的标记名称。**

可以看出 SEEK 函数的作用与 SEEK 命令和 FOUND 函数连用效果相同。

例 4.49　使用 SEEK 函数在xsqkb.dbf**中查询有关记录。**

```
USE xsqkb
? SEEK('李会琴','xsqkb','xm'),RECNO() && .T. 2
? SEEK('女'+DTOC({^1979/06/09},1),'xsqkb','xbsr'),RECNO(),EOF() && .F. 11 .T.
```

4.5　表的统计与汇总操作

统计和汇总是数据库应用的重要内容，下面介绍几个常用统计与汇总命令。

4.5.1　计数操作

所谓计数操作，指在表中统计满足条件的记录的条数。

1. 命令格式

```
COUNT [Scope] [FOR lExpression1] [WHILE lExpression2]
    [TO VarName][NOOPTIMIZE]
```

2. 功能

统计当前表文件中指定范围内符合条件的记录个数。

3. 参数和子句说明

- *Scope*、FOR *lExpression*1、WHILE *lExpression*2：**缺省指所有记录都在统计之列。**
- TO *VarName*：**指定要把统计结果写入的内存变量名；缺省时，仅在提示栏显示。**

例 4.50 统计xsqkb.dbf中男生人数。

```
USE xsqkb
COUN FOR 性别='男' TO nan
? nan                                    && 5
```

注意 COUNT 命令和 RECCOUNT()函数的功能不同。COUNT 命令统计的是满足某条件的记录,因此对于不满足条件的记录或者是 SET DELETE ON 状态下加了删除标记的记录(无效记录)是不统计的。而 RECCOUNT()函数则是返回表中的记录条数,与记录是否带有删除标记无关。

例 4.51 比较 COUNT 命令和 RECCOUNT()函数的不同。

```
USE xsqkb
COUN FOR 性别='男' TO nan
? nan,RECCOUNT()                         && 5 10
DELETE FOR 性别="男"
SET DELETE OFF
COUN FOR 性别='男' TO nan                 && 在 SET DELETE OFF 状态下,仍然统计
? nan,RECCOUNT()                         && 5 10
SET DELETE ON
COUN FOR 性别='男' TO nan                 && 在 SET DELETE OFF 状态下,不统计
? nan,RECCOUNT()                         && 0 10
```

4.5.2 求和操作

求和操作是指对表中的数值型字段或数值型表达式,进行纵向求和的操作。

1. 命令格式

SUM [*nExpression*List] [*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]
[TO *MemVarNameList* | TO ARRAY *ArrayName*] [NOOPTIMIZE]

2. 功能

在打开的表中,对数值表达式清单 *nExpression*List 中的各个表达式分别求和。

3. 参数和子句说明

- *nExpression*List:指定要进行求和的一个或多个数值型字段或数值型表式。缺省时为当前表的所有数值型字段。
- Scop FOR *lExpression*1、WHILE *lExpression*2:意义同于前面的 COUNT 命令。
- TO *MemVarNameList*,指定把求和的每一个结果写入到对应的一个内存变量中。
- TO ARRAY *ArrayName*:将求和结果存储到一个内存变量数组中。如果在 SUM 命令中指定的数组事先并未声明,则系统会自动生成它。如果数组事先已声明但数组的元素个数却比要存储的求和的表达式个数少,则数组的元素个数将自动增加,直到和表达式个数相等。当[TO *MemVarNameList* | TO ARRAY *ArrayName*]缺省时,求和结果仅在状态栏显示。

例 4.52 已知在课程代码表 kcdmb.dbf 中,对于凡多学期开设的课程每学期的课数都是

平均分配的，请统计出第二学期所开设课程的总课时数。

```
USE kcdmb
SUM 课程学时/LEN(TRIM(开课学期)) FOR '2' $ 开课学期 TO zxs
? zxs                                    && 显示结果 399.0000
```

4.5.3　求平均值

求平均操作是指对表中的数值型字段或数值表达式，进行纵向求算术平均的操作。

1. 命令格式

AVERAGE [*nExpression*List][*Scope*][FOR *lExpression*1] [WHILE *lExpression*2]
[TO *VarList* | TO ARRAY *ArrayName*] [NOOPTIMIZE]

2. 功能

对当前表的数值型字段或数值型表达式列表求算术平均。

3. 参数和子句说明

各子句与参数的意义均与 SUN 相同。

例 4.53　求 xscjb.sbf 中各科成绩的平均值，将结果写在数组 pjcj 中。

```
USE xscjb
BROWSE
AVERAGE TO ARRAY pjcj
DISPLAY MEMORY LIKE pjcj
```

结果如图 4-29 所示。图中，右边是 BROWSE 命令执行的结果，左边是求平均后数组 pjcj 中的各元素的值。

图 4-29　对学生成绩表的浏览和求平均

4.5.4　分类统计

实际应用中分类汇总经常要用到。像仓库的库存管理，经常要统计各类产品的库存总量，商店的销售管理经常要统计各类商品的售出总量等。它们共同的特点是首先要进行分类，将同类别数据放在一起，然后再进行数量求和之类的汇总运算。

1. 命令格式

TOTAL TO *TableName* ON *FieldName*
[FIELDS *FieldNameList*]

[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2] [NOOPTIMIZE]

2. 功能

对当前表文件按指定的字段进行汇总，并生成一个汇总的表文件，汇总文件除不能包含备注型，通用型字段外，其他结构都与被汇总文件的结构相同。

3. 参数和子句说明

• TO *TableName*：指定要生成的含有汇总结果的表的名字。

• ON *FieldName*：指定要用来进行分类汇总的字段。被汇总的表必须事先按命令中该字段进行排序或索引，如果是索引，则该索引必须设置为主控索引。

• FIELDS *FieldNameList*：指定要进行汇总的数值型字段名，若命令中缺省该可选项，则对当前表文件中所有的数值型字段进行汇总；

• *Scope*、FOR *lExpression*1、WHILE *lExpression*2：全部缺省时为所有记录。

注意　汇总后的记录中，只有关键字和各汇总字段的值才有实用价值。其他不进行汇总的字段的值尽管将保持关键字值相同的各组记录中第一条记录的值，但这些字段的值显然没有意义。

例 4.54　有如图 4－30 所示职工工资表，要求按部门对各项工资内容进行分类汇总。

图 4－30　职工工资情况表

```
USE zggzb
INDEX ON 部门 TO bm
TOTAL ON 部门 TO gzhz_bm
USE gzhz_bm && 打开汇总表
LIST && 显示汇总结果
```

记录号	部门	姓名	基本工资	奖金	补贴	应发	房改基金	水电费	应扣	实发
1	A	习贵生	2070.50	620.00	414.10	3104.60	248.36	37.00	285.36	2819.24
2	B	高长林	5072.00	1520.00	1014.40	7606.40	608.51	84.60	693.11	6913.29
3	C	卢延安	3265.50	977.00	653.10	4895.60	391.65	63.00	454.65	4440.95

本例汇总的结果是一个新的表文件(gzhz_bm)，其结构与原表文件相同，每个部门的工资数据汇总成一条记录。每一记录中，参与汇总的各个字段的值是同一单位所有记录对应字段汇总的结果，而其他非数值型字段如“姓名”等仍保留本单位的第一条记录中相应的字段值。

显然，这里“姓名”字段值已经没有意义，应予以舍弃，实际有意义的只有关键字“部门”字段和被汇总的字段。我们可以用下面的命令将这些字段复制出来生成一个汇总结果数据表。

COPY TO bmhzjc FIELDS 部门，基本工资，奖金，补贴，应发，房改基金，水电费，应扣，实发

4.6　表的投影与选择操作

关系型数据库有三种基本操作：投影、选择、连接。其中投影是对字段的操作，选择是对记录的操作，它们都是对一个表而言的，而连接操作则是对多个表而言的。

4.6.1　表的投影操作

表的投影操作可以通过设置对字段的筛选来实现，和表的其他操作类似，它也有菜单和命令有两种方式。

1. 菜单方式

菜单方式下设置字段的筛选是通过“字段选择器”来实现的。下面讲解其具体操作过程。

S1：打开数据表（如：xsqkb.dbf）。

S2：打开数据工作期窗口：→[菜单]→[数据工作期]↓ 数据工作期，结果如图 4-31 左所示。

S3：打开工作区属性窗口，如果已打开多个表，则先在“别名”列表框选表，再单击[属性]；如果是一个表，则直接单击[属性]。结果如图 4-31 右所示。

图 4-31　“数据工作期”和“工作区属性”窗口

S4：打开字段选择器窗口：→[由字段筛选器指定的字段]→[字段选择]↓ 字段选择器，结果如图 4-32 左所示。

S5：选择有关字段，如图 4-32 右所示，→[确定]→[确定]→[X]，完成字段筛选。

图 4-32　“字段选择器”窗口

此时用命令 BROWSE 浏览结果，将得到如图4－33的结果。

图 4－33 字段筛选结果

2. 命令方式

(1)命令格式

SET FIELDS ON | OFF | LOCAL | GLOBAL

SET FIELDS TO [[*FieldName*1 [, *FieldName*2 ...]] |

ALL [LIKE *Skeleton* | EXCEPT *Skeleton*]]

(2)功能

指定表中可以被访问的字段。

(3)参数及子句说明

• ON：指定仅 SET FIELDS TO 字段列表中给出的字段可以被访问。

• OFF(缺省值)：指定当前表中所有的字段可以被访问，即撤销对字段访问的限制。

• LOCAL：指定在字段列表中仅当前工作区列出的字段可以被访问。

• GLOBAL ：指定字段列表中列出的所有字段均可访问，包括在其他工作区中的字段。SET FIELDS GLOBAL 可以使用户在未使用 SET COMPATIBLE TO DB4 的情况下直接访问其他工作区中的字段。

• TO [*FieldName*1[, *FieldName*2...]] ：指定当前表中可以被访问的字段的名字。在下面的各种情况下，它必须包含表的别名：

情况 1：当包含该字段的表不在当前被选中的工作区打开时；

情况 2：当字段名在两个或多个表中相同时。

• ALL [LIKE *Skeleton* | EXCEPT *Skeleton*]]：含义和前面其他命令相同，可使用统配符，如无 LIKE|EXCEPT 子句，则表示可访问当前表的所有字段。

(4)注意事项

• SET FIELDS TO 是可添加的。使用带有字段列表的 SET FIELDS TO 命令，将给出的字段添加为可以被访问的字段。

• SET FIELDS TO 隐含执行了 SET FIELDS ON 命令。如果执行不带字段列表子句或 ALL 的 SET FIELDS TO 命令，则从当前表中移去字段列表中列出的所有字段，从而使每个字段都不能被访问。

例 4.55 只允许对数据表“xscjb.dbf”中的“学号”、“总分”两个字段进行操作。

```
USE xscjb exclusive
SET FIELDS TO 学号,总分
BROWSE
```

结果如图 4－34 所示。

图 4－34 使用字段筛选命令后的浏览结果

4.6.2 表的选择操作

表的选择操作也称为数据过滤。这可以通过设置对记录的筛选来实现。和表的其他操作类似，它也有菜单和命令有两种

方式。

1. 菜单方式

菜单方式下设置记录的筛选是通过“数据过滤器”来实现的。“数据过滤器”的打开步骤与上节讲的“字段选择器”的打开相类似。下面以对表“xsqkb.dbf”设置记录筛选为例讲解其具体操作过程。

S1～S3：与打开“字段选择器”相同。

S4：在图 4－31 右的“工作区属性”窗口的“数据筛选”文本框直接输入筛选表达式，或单击文本框右边的 … 打开表达式生成器来生成数据筛选的逻辑表达式，例如：“出生日期＜{^1980/01/01}”。

S5：如果需要使索引也同时起作用，可在“索引顺序”列表框选择输出的次序，例如选结构复合索引的“Xsqkb.学号”标记，结果如图 4－35 所示。

图 4－35　“数据筛选”和“索引顺序”

S6：→[确定]↓ 数据工作期，至此字段筛选完成。

S7：观察筛选结果如图 4－36。

图 4－36　“数据筛选”结果浏览

可以看出，经过上述筛选后，所有显示的都是 1980 年之前出生的学生，且按升序排列。

2. 命令方式

(1)命令格式

SET FILTER TO [*lExpression*] [IN *nWorkArea* | *cTableAlias*]

(2)功能

指定当前表中的记录可以被访问的条件。

(3)参数及子句说明

• *lExpression*:逻辑表达式,指定被访问的记录必须满足的条件。

• IN *nWorkArea* | *cTableAlias*:指定由 SET FILTER 命令所影响的工作区或表的别名。使用本子句,可以指定当前工作区之外的工作区或表。缺省时,为当前工作区。

(4)说明

当参数和子句全缺省时,指取消对记录的筛选。

例 4.56 用 SET FILTER 命令实现上面菜单操作的功能。

```
USE xsqkb ORDER TAG 学号
SET FILTER TO 出生日期<{^1980/01/01}
BROWSE
```

结果也如图 4-36 所示。

4.7 多表操作

前面我们学习到的命令,其操作对象都仅有一个表文件。在实际应用中,即使是一个数据库,也常常涉及到多个表文件。Visual FoxPro 9.0 提供了同时使用不同数据表文件或多个自由表的命令——多区操作命令。

4.7.1 Visual FoxPro 9.0 的内存工作区

所谓工作区,指将连续的内存空间,划分为若干个相对独立的区间,每个区间允许打开一个表文件。如果在一个工作区中打开第二个表文件,那么第一个表文件先自动关闭,然后才打开第二个文件。Visual FoxPro 9.0 共设置了 32 767 个工作区。

1. 多区操作的特点

① 每个工作区同时只能打开一个表文件,一个表文件也不能在一个以上的工作区同时打开,否则出现"文件正在使用"提示信息。

② 不论已经使用了多少个工作区,只有一个是当前工作区,在当前工作区中打开的表文件就是当前表文件。系统启动后,默认 1 号工作区为当前工作区。用户可以使用命令选择 1～32767个工作区号中的任意一个做为当前工作区。但同时最多能使用 255 个工作区。

③ 每个工作区为打开的表文件设置一个记录指针,在一般情况下它们各自独立移动,互不干扰。

2. 多区操作的种类

多区操作主要包括下列几种情况:

① 用同一条命令访问多个工作区中的数据。例如命令"LIST <*FieldList*>"中的字段名可以取自不同的工作区中的不同表文件。

② 使用专用的多区操作命令。如 SET RELATION 命令、UPDATE 命令、JOIN 命令等都能实现多区操作。它们的共同特点是:能使多个工作区中记录指针实现联动,从而提高数据处理效率。

③ 下列命令能作用于多个工作区,对多个工作区同时生效。例如:

CLOSE DATABASE 用于关闭所有的数据库、数据表文件、索引文件等。

CLOSE TABLES ALL 关闭所有的数据表文件。

CLEAR ALL 清除所有内存变量、窗口、菜单等，也关闭所有用户打开的文件。

4.7.2　工作区的选择

1. 工作区的标识

工作区可以有以下四种标识。

① VFP 系统工作区号：选择范围为 1～32 767。

② VFP 系统工作区别名：前 10 个工作区用 A～J 来表示各自的工作区别名，自 11 号工作区开始，在编号前加上“W”代表工作区的别名，例如：“W254”等价于“254”号工作区。

③ 表名：当有表文件打开时，表文件名还可作工作区标识。这种标识方法比较常用。

④ 表别名：工作区的标识还可以用表的别名来表示。

例如：如果当前工作区为 4 号，若执行了命令：

USE XSDA ALIAS ABC”

则当前工作区就有以下四种标识符：“4”、“D”、“XSDA”、“ABC”。

2. 工作区的选择

(1)命令格式

SELECT *nWorkArea* | *cTableAlias*|0

(2)功能

激活所选定的工作区。

(3)参数及子句说明

· *nWorkArea*：系统给出的工作区号，取值范围：1～32 767；也可使用 A～J、W11～W32767。

· *cTableAlias*：表的别名，如果表没有起别名，则可以用表名代替别名。

· 0：指定区号最小的一个未用工作区。

(4)说明

函数 SELECT()可返回当前工作区号；函数 ALIAS([*nWorkArea*])可返回当前工作区或指定的工作区别名。

例 4.57　工作区选择举例。

```
CLEAR ALL
USE xsqkb                    && 在 1 号工作区打开表“xsqkb”
SELECT 3                     && 选择 3 号工作区
USE xscjb ALIAS cj           && 在 3 号区以别名“cj”打开表“xscjb”
USE js in 0                  && 在 2 号工作区打开表“js”
```

4.7.3　工作区的联访

在当前工作区中，除对当前表文件中的数据访问外，还能访问其他工作区中表的字段，但此时应在被访问的字段加上它所在工作区的标识，并将工作区标识与字段名用联访符相连接。有两种格式：

· **格式 1**：AliasName －＞ *FieldName*

· **格式 2**：AliasName.*FieldName*

在第一种格式中，“－＞”由减号“－”和大于号“＞”组成，牢记两者之间不得有空格。常用第二种方法。

例 4.58　显示“李会琴”的综合积分情况。

```
CLEAR ALL
USE xsqkb
LOCATE FOR 姓名=[李会琴]          && 将指针定位于“李会琴”
DISPLAY                           && 显示“李会琴”有关数据
SELE 0
USE xszhjfb
LOCATE FOR 学号=xsqkb.学号        && 根据“李会琴”的学号定位本区记录指针
DISP 学号,xsqkb.姓名,精神文明,社会活动,体育锻炼
```

记录号	学号	Xsqkb－＞姓名	精神文明	社会活动	体育锻炼
2	010102	李会琴	90.0	91.0	88.0

```
LIST 学号,xsqkb.姓名,精神文明,社会活动,体育锻炼
```

记录号	学号	Xsqkb－＞姓名	精神文明	社会活动	体育锻炼
1	010001	李会琴	89.0	90.0	76.0
2	010102	李会琴	90.0	91.0	88.0
3	010111	李会琴	87.0	79.0	95.0
4	011216	李会琴	76.0	85.0	79.0
5	011217	李会琴	92.0	88.0	91.0
6	011320	李会琴	96.0	85.0	79.0
7	012001	李会琴	87.0	91.0	85.0
8	012002	李会琴	85.0	79.0	84.0
9	012003	李会琴	90.0	87.0	90.0
10	012234	李会琴	96.0	88.0	92.0

从上例可以看出，显示的结果来自两个数据表，其中的姓名来自于xsqkb.dbf**，这就是工作区的互访。同时还可以看到，在最后使用** LIST **命令显示时，随着学号及其他字段值的变化，姓名却没有发生变化，其原因是因为，两条显示命令均省略了范围和条件选项，此时，**DISPLAY **命令只显示当前一条记录，而两个表中的记录指针已经被我们提前设置好，所以显示出来的信息是同一个人李会琴的；当用** LIST **命令时显示的是当前表(**xszhjfb**)中所有记录内容，而当前表记录指针的变化不会引起另一个工作区(**xsqkb**)指针的变化，所以显示出来的姓名是不会变化的。**

要想使多个工作区指针发生相关变化，即“指针跟随”，则必须使用表的临时关联操作。

4.7.4　表的临时关联

数据表文件的关联操作是开发数据库应用系统工作中最常见的工作之一，表的关联分为数据库表之间的永久关联和表之间的临时关联两种。本节讲解表的临时关联，关于数据库内部表之间的永久关联，将在数据库的有关操作中讲解。

所谓表的关联是指利用表之间存在的被称为“关系表达式”的共有关键字将它们联系起来。永久关联将作为数据库结构的一部分被永久地保存下来；临时关联与是否属于同一数据库表还是自由表无关，这样建立的关联关系可随时予以解除。临时关联的最大特点是在建立了临时关联关系的表之间，将发生“指针跟随”现象，当父表的记录指针移动时，子表的记录指针将跟着移动到对应的记录上。

1. 命令方式创建表的临时关联

(1)命令格式

SET RELATION TO [*Expression*1 INTO *nWorkArea*1 | *cTableAlias*1
　　[,*Expression*2 INTO *nWorkArea*2 | *cTableAlias*2 ...]
　　[IN *nWorkArea* | *cTableAlias*] [ADDITIVE]]

(2)功能

在当前表文件(父表)与其他表文件(子表)之间建立临时关联。

(3)参数及子句说明

· *Expression*1：指定建立父表和子表之间建立临时关联所用的关系表达式。它通常是用来控制子表索引的一个索引表达式。如果 *Expression*1 是数值型的，则当父表的记录指针移动时它将被计算出来。子表的记录指针将移动到 *Expression*1 给出的记录上。如果缺省了所有参数和子句，SET RELATION TO 将解除当前工作区所有临时关联。*Expression*2 与 *Expression*1 意义相同。

· INTO *nWorkArea*1 | *cTableAlias*1：指定子表工作区号或别名。

· *Expression*2 INTO *nWorkArea*2 | *cTableAlias*2：与 *Expression*1 INTO *nWorkArea*1 | *cTableAlias*1 意义相同。

· IN *nWorkArea* | *cTableAlias*：指定父表的工作区或别名。IN 子句允许创建关联时，不必首先选择父表所在工作区。如确缺省本子句，则父表必须在当前工作区打开。

· ADDITIVE：保存当前工作区中已存在的所有关联关系，并创建新的关联关系。如缺省 ADDITIVE 关键字，则首先是取消原存在的关联关系，再创建新的关联关系。

2. 注意事项

(1)建立两个表临时关联的条件

① 两个表必须同时分别在不同的工作区打开。

② 两个表中都必须拥有“相同”的字段，所谓相同字段，并不是指字段名相同，而是指字段的宽度、类型一致，字段名可以不相同。

③ 子表必须按关联字段建立过索引且索引文件已打开。

(2)关联的方式

父表与子表之间可以有两种关联方式，若在命令中选择了＜索引关键字表达式＞，则表示

按关键字表达式值建立关联，为此，子表必须将有关的索引文件打开。若选择了＜数值表达式＞，则将按数值表达式的值建立关联，此时子表不必索引，该种方法建立关联的应用较少。

(3)记录指针的跟随

两表建立关联后，每当父表中的记录指针移动时，子表中的记录指针便按指定的关联条件而随之移动，移动方法也分两种情况。

① 表若按关键字表达式建立关联，此时在子表中便自动执行一次 SEEK 命令，若在子表中找到与关键字表达式值相匹配的记录，则子表的记录指针就定位于与之匹配的首记录上。若找不到，则子表指针移至文件末尾。

② 两表若按数值表达式建立关联，则在子表中便自动执行一次 GO 命令，将记录指针就定位于记录号等于此数值表达式值的那条记录上。

(4)临时关联的撤销

如果命令中所有的选项都缺省，就撤消当前表文件已经建立的一切关联。

例 4.59　显示所有学生的综合积分情况(包括姓名)。

```
CLEAR ALL
USE xsqkb
INDEX ON 学号 TO XS_XH
SELE 0
USE xszhjfb
INDEX ON 学号 TO ZH_XH          && 父表可以不用建立索引
SET RELA TO 学号 INTO XSQKB
LIST 学号,xsqkb.姓名,精神文明,社会活动,体育锻炼
```

记录号	学号	Xsqkb->姓名	精神文明	社会活动	体育锻炼
1	010001	张凯华	89.0	90.0	76.0
2	010102	李会琴	90.0	91.0	88.0
3	010111	李小茜	87.0	79.0	95.0
4	011216	宋秀兰	76.0	85.0	79.0
5	011217	郭正宏	92.0	88.0	91.0
6	011320	姜亚男	96.0	85.0	79.0
7	012001	杨书敏	87.0	91.0	85.0
8	012002	宋越辉	85.0	79.0	84.0
9	012003	杜拥军	90.0	87.0	90.0
10	012234	王向东	96.0	88.0	92.0

可以看出，此时的结果和前面显示的结果亦不相同，姓名字段显示的再也不是“李会琴”一个人，而是和学号相对应的各个学生的姓名。

关于表的永久关联的创建将在数据库一章中讲解。

4.8　文件操作

4.8.1　表文件的复制

1. 命令格式

COPY TO *FileName*

[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]

[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]

[[WITH] CDX | [[WITH] PRODUCTION]] [AS *nCodePage*]

2. 功能

把当前表文件复制成一个新的表文件。

3. 参数及子句说明

· *FileName*：**要生成的新文件的名字**。

· FIELDS *FieldList*：**要复制到新文件中的的字段。缺省为全部字段。**

· [WITH] CDX| [[WITH] PRODUCTION]：**生成一个新表的结构复合索引文件，该索引文件与原始表的结构复合索引文件相同。原结构复合索引文件的标识和索引表达式都被复制到新的结构复合索引文件中。子句中的**“WITH”**关键字有无效果相同；**CDX **和** PRODUCTION **子句相同。注意，当用户复制的文件如果不是一个新的** VFP **表，则不应不包含** CDX **或** PTODUCTION **子句。**

· AS *nCodePage*：**缺省** AS *nCodePage* **代码页子句，则新生成的表的代码页被转为当前的** Visual FoxPro **代码页。**

其他子句与过去相同。

例 4.60　将学生情况表“xsqkb.dbf”**连同它的结构复合索引文件一起复制成另一个表**“xsqkb1.dbf”。

```
USE xsqkb
COPY TO xsqkb1 CDX
USE xsqkb1 IN 0
COPY TO xsqkb2
USE xsqkb2 IN 0
SELECT xsqkb1
MODIFY STRUCTURE          && 结果如图 4-37 所示。
SELECT xsqkb2
MODIFY STRUCTURE          && 结果如图 4-38 所示。
```

比较图 4-37 **和图** 4-38，**可以看出** COPY TO **命令中是否含有** CDX **子句，结果完全不同。**

图 4-37 带有 CDX 子句的表复制结果

图 4-38 不带 CDX 子句的表复制结果

需要说明的是,表文件的复制命令不但复制表文件本身还将复制表的备注文件“.fpt”。

4.8.2 表结构的复制

1. 命令格式

COPY STRUCTURE TO *TableName*
[FIELDS *FieldList*] [[WITH] CDX | [WITH] PRODUCTION]
[DATABASE c*DatabaseName* [NAME c*TableName*]]

2. 功能

复制一个与当前表文件结构相同的空表。

3. 参数及子句说明

- *TableName*:指定要生成的新的空表文件的名字。
- FIELDS *FieldList*:缺省为复制全部字段。
- [WITH] CDX | [WITH] PRODUCTION:意义与上面的命令相同。值得注意的是,在 VFP 中,当前所选中的表的主索引会被自动转成新的空表的候选索引。
- DATABASE *cDatabaseName*:指定要将新的空表添加到已存在的某个数据库的库名。注意表和字段特性并不复制到数据库中。
- NAME *cTableName*:指定出现在数据库中的表的名字。

例 4.61 将学生情况表“xsqkb.dbf”的学号、姓名、性别、出生日期四个字段复制成另一个新的空表“bjg.dbf”,并复制结构复合索引。

```
USE xsqkb
COPY STRUCTURE TO bjg FIELDS 学号,姓名,性别,出生日期 PRODUCTION
USE bjg
LIST STRUCTURE TO bjg
TYPE bjg.TXT
4 表结构:                    D:\XYB\教材编写\VFP9.0\表\BJG.DBF
数据记录数:                  0
最近更新的时间:              02/08/06
```

代码页：　936

字段	字段名	类型	宽度	小数位	索引	排序	Nulls	下一个	步长
1	学号	字符型	6		升序	PINYIN	否		
2	姓名	字符型	8		否				
3	性别	字符型	2		否				
4	出生日期	日期型	8		降序	PINYIN	否		

** 总计 ** 25

4.8.3　表文件与其他格式文件的数据交换

Visual FoxPro 9.0 可以实现与其他格式的文件进行数据交换。这种交换既可通过命令方式进行，也可通过菜单方式进行操作。

1. 将表文件复制成其他格式的数据文件

(1)命令方式

① 命令格式

COPY TO *FileName* [DATABASE *DatabaseName* [NAME *LongTableName*]]
[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]
[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]
[[TYPE] [FOXPLUS | FOX2X | DIF | MOD | SDF | SYLK | WK1 | WKS | WR1 | WRK | CSV | XLS | XL5 | DELIMITED [WITH *Delimiter* | WITH BLANK | WITH TAB | WITH CHARACTER *Delimiter*]]] [AS *nCodePage*]

② 功能

将当前表复制成一个新的其他格式的数据文件。

③ 参数及子句说明

• *FileName*：要复制成的其他格式数据文件的名字。

• TYPE：如果要生成的文件不是 Visual FoxPro 的表文件，则需要用 TYPE 子句来说明文件类型。尽管此时必须给出文件的类型，但是“TYPE”关键字却并不需要一定要写出。命令中 TYPE 关键字之后的关键字给出的是要复制成其他格式的数据文件的文件格式标识。由于数据文件格式标识关键字极多，下面仅介绍几个常用的文件格式标识符。

• SDF：生成一个系统文件格式(system data format)文本文件。该文件的记录有固定的长度，以回车换行符作为结尾。字段间没有分隔符。缺省的扩展名为：“.TXT”。当用 COPY TO 命令生成一个 SDF 格式的文件时，SET CENTURY 设置将被忽略，Visual FoxPro 表中的日期型字段将以 YYYYMMDD 格式给出。如果日期信息以明确的格式存储，在执行 COPY TO 操作之前，用户应该确保日期是 YYYYMMDD 格式的。

• XL5：生成一个 Microsoft Excel 5.0 版的工作簿文件。

• DELIMITED：生成一个字段间带有分隔符的文本文件，其缺省的字段分隔符是一个逗号。由于字符型数据中可能包含逗号，因此字符型字段将自动用双引号来予以定界。

• DELIMITED WITH *Delimiter*:生成一个对于字符型字段用 *Delimiter* 字符做定界符的文本文件。

• DELIMITED WITH BLANK :生成一个用空格代替逗号做分隔符的文本文件。

• DELIMITED WITH TAB :生成一个用制表符代替逗号做分隔符的文本文件。

其他子句与前面命令介绍的意义相同。

例 4.62 将表"xsqkb.dbf"复制成一个 Excel 5.0 的工作簿文件"xsqkb.xls"。

```
CLOSE ALL
CLEAR
DELETE FILE *.XLS          && 删除当前文件夹中的所有 Excel 文件
USE xsqkb
COPY TO xsqkb XL5
DIR *.XLS
```

读者此时会发现,在当前文件夹中已存在一个刚生成的"xsqkb.xls"文件。

例 4.63 将学生情况表"xsqkb.dbf"的学号、姓名、性别、出生日期四个字段生成一个以空格为分隔符的文本文件"xs.txt",并显示该文本文件内容。

```
CLOSE ALL
CLEAR
DELETE FILE *.TXT      && 删除当前文件夹中的所有 Excel 文件
USE xsqkb
COPY TO xs FIELDS 学号,姓名,性别,出生日期 DELIMITED WITH BLANK
TYPE xs.txt
```

显示结果为:

```
"010101" "张凯华" "男" 02/13/1980
"010102" "李会琴" "女" 07/25/1979
"010111" "李小茜" "男" 12/21/1979
"011216" "宋秀兰" "女" 10/31/1980
"011217" "郭正宏" "女" 11/25/1979
"011320" "姜亚男" "女" 09/30/1980
"012001" "杨书敏" "女" 08/18/1980
"012002" "宋越辉" "男" 06/09/1979
"012003" "杜拥军" "男" 05/20/1980
"012234" "王向东" "男" 04/26/1979
```

(2)菜单方式

菜单方式下,也可以将 Visual FoxPro 的表文件复制成其他格式的数据文件。现以将表文件"xsqkb.dbf"复制成一个 Excel 文件为例讲解其复制步骤。

S1:打开表"xsqkb"。

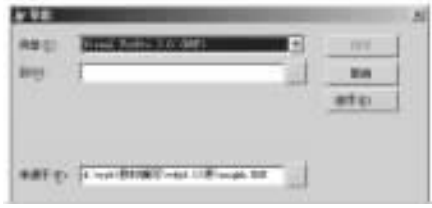


图 4-39 "导出"对话框之一

S2：→[文件]→[导出…]↓ 导出，如图 4－39 所示。

S3：在类型下拉列表框选：Microsoft Excel 5.0。

S4：→“到”文本框右侧的 … ↓ 另存为。

S5：在“导出”文本框输入要生成的 Excel 工作簿的名字：“xsqkb1”，如图 4－40 所示。

S5：→[保存]← 导出，结果如图 4－41 所示。

图 4－40　“另存为”对话框

图 4－41　“导出”对话框之二

S6：→[确定]，文件复制完成。同时在命令窗口生成了对应的操作命令：

COPY TO xsqkb1.xls TYPE XL5

1. 将其他格式的数据文件追加成表记录

这是上面 COPY 命令的逆操作。

(1)命令方式

① 命令格式

APPEND FROM *FileName* | ? [FIELDS *FieldList*]
　[FOR *lExpression*]
　[[TYPE] [DELIMITED [WITH *Delimiter* | WITH BLANK
　| WITH TAB | WITH CHARACTER *Delimiter*]
　| DIF | FW2 | MOD | PDOX | RPD | SDF | SYLK
　| WK1 | WK3 | WKS | WR1 | WRK | CSV
　| XLS | XL5 [SHEET *cSheetName*] | XL8 [SHEET *cSheetName*]]]
　[AS *nCodePage*]

② 功能

将其他文件中的数据追加为当前表的记录。

③ 参数及子句说明

- *FileName*：指定要追加的表文件的名字。
- XL5 [SHEET *cSheetName*]：从 Microsoft excel 5.0 版的工作簿中导入数据。如果缺

省 SHEET 子句，则工作簿中工作表 1 的数据被导入。如果要导入工作表中指定的工作表的数据，则应带有 SHEET *cSheetName* 子句，以便指明工作表。

例 4.64 先将例 4.61 生成的“bjg.dbf” 打开，然后将以空格为分隔符的文本文件“xs.txt”的各条记录追加成该表的新记录，最后显示表记录。

```
CLOSE ALL
CLEAR
USE bjg
APPEND FROM xs DELIMITED WITH BLANK
LIST
```

显示结果为：

记录号	学号	姓名	性别	出生日期
1	010101	张凯华	男	02/13/80
2	010102	李会琴	女	07/25/79
3	010111	李小茜	男	12/21/79
4	011216	宋秀兰	女	10/31/80
5	011217	郭正宏	女	11/25/79
6	011320	姜亚男	女	09/30/80
7	012001	杨书敏	女	08/18/80
8	012002	宋越辉	男	06/09/79
9	012003	杜拥军	男	05/20/80
10	012234	王向东	男	04/26/79

(2)菜单方式

用菜单方式将其他格式的文件数据导入 Visual FoxPro 的表文件，通过“表”菜单的“导入”选项实现。现以将 xs.txt 文件追加成与表“bjg.dbf”同结构的空表“bjg1.dbf”的记录为例，讲解使用“导入向导”将文本文件追加成表记录的操作步骤。需要强调的是，这时的表必须是关闭的，否则不能实现记录的追加。

图 4-42 “导入”对话框

S1：→[文件] → [导入…] ↓ 导入，如图 4-42 所示。

S2：→[导入向导] ↓ 输入导向(Import Wizard) 的第 1 步对话框“确定数据(Identify Data)”。

S3：在该话框的中：

S3-1：在“文件类型 (FileType)：”列表框选“文本文件(Text File)”。

S3-2：在“源文件(Source File)：”列表框选要追加的文本文件的名字，例如：xs.txt。

S3-3：在“目标文件(Destination File)：”单选框选择“已存在的表(Existing Table)”，并在其后的列表框选定目标表，例如：bjg1.dbf。结果如图 4-43 所示。

S4：→[下一步] ↓ 第 2 步确定数据格式 (Step 2 Detemine Data Format)。

S5：在该对话框中：

S5-1：在“数据格式(Data Format)”单选框，选“分隔(Delimited)”；

S5－2：**在“字段名所在行**(Field names in row)**”列表框，选**“0”；

S5－3：**在“开始导入所在行**(Begin import in row)**”列表框，选**“1”，**结果如图** 4－44 **所示**。

图 4－43　**“导入”向导步骤** 1

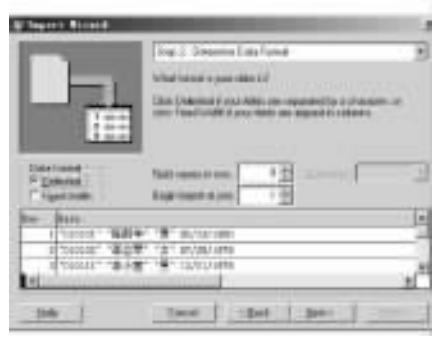


图 4－44　**“导入”向导步骤** 2

S6：→[**下一步**] ↓ **第** 2a **步　描述数据** (Describe Data)；

S7：**在“字段间分隔符**(Field Delimiters)**”单选框，选“空格**(Space)**”，如图** 4－45 **所示**。

S8：→[**下一步**] ↓ **第** 3 **步　确定导入字段** (Define Imported Fields)，**如图** 4－46 **所示**。

S9：→[**下一步**] ↓ **第** 3a **步　指定国际选项对话框“** (Specify international Option)”，**如图**4－47**所示**。

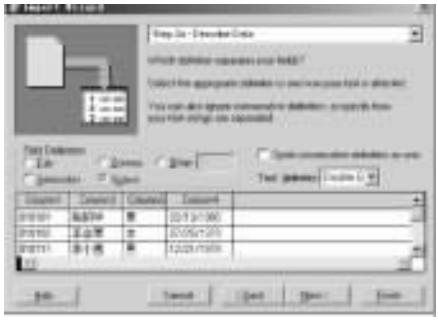


图 4－45　**“导入”向导步骤** 2a

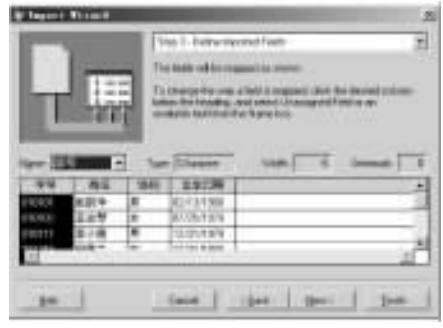


图 4－46　**“导入”向导步骤** 3

图 4－47　**“导入”向导步骤** 3a

图 4－48　**浏览** bjg1.dbf

S10:→[下一步] ↓ 第 4 步 完成。

S11:→[完成]。

S12:打开 bjg1.dbf。

S13: 浏览 bjg1.dbf。

结果如图 4-48 所示。

显然,使用"导入"向导从其他格式的数据文件中向表追加记录的步骤较慢,操作繁琐,这适用于初学者。对于熟练的用户则可直接"导入"对话框中选择有关的选项来完成数据的导入。

4.8.4 一般文件的复制

① 命令格式

COPY FILE *FileName1* TO *FileName2*

② 功能

将源文件 *FileName1* 复制成目标文件 *FileName2*,可用于任何类型的文件的复制。

③ 参数及子句说明

FileName1、*FileName2*:分别指定源文件、目标文件的名字。要复制的文件必须是关闭的。无论是源文件还是目标文件,都应包含扩展名,且都可以使用通配符 * 和?。但这样的复制不能改变文件的类型。

文件的复制也可在命令窗口使用 DOS 命令:

! |RUN COPY *FileName1* *FileName2*

例 4.65 对表文件"xsqkb.dbf"连同它的备注文件、结构复合索引文件等都做一个备份文件,备份文件的名字为:"xsqkbbf",各自的扩展名不变。

```
CLOSE ALL
COPY FILE xsqkb.* TO xsqkbbf.*
```

4.8.5 文件的更名

① 命令格式

RENAME *FileName1* TO *FileName2*

② 功能

将文件 *FileName1* 改名为文件 *FileName2*,包括扩展名。

③ 注意事项

- 如果文件缺省扩展名,系统将默认扩展名为".dbf"。
- 要对一个的确无扩展名的文件更名,应在文件名之后附加一个句点"."。
- 如果对一个附有备注文件".fpt"的自由表更名,一定要确保同时对该备注文件也更名。
- 无论是 *FileName1* 还是 *FileName2*,都可以包含通配符 * 和?。
- 如果文件不在缺省路径中,则文件名中应带有路径。
- 如果两个文件在不同的目录或文件夹中,则意味着将 *FileName1* 移动到 *FileName2* 所给出的路径或文件夹。
- 当进行文件更名时,必须确保 *FileName2* 不存在,*FileName1* 是关闭的。

文件的更名也可使用 DOS 命令：

! |RUN REN *FileName*1 *FileName*2

例 4.66　将当前目录中的文件的名字为："xsqkbbf"的各类文件全部移到 d 盘的 ys 文件夹下，且全部更名为"bf"，各自的扩展名不变。

```
CLOSE ALL
! REN xsqkbbf.* d:\ys\bf.*
```

4.8.6　文件删除

1. 命令方式删除文件

(1)命令格式

DELETE FILE [*FileName* | ?] [RECYCLE]

(2)功能

从磁盘上删除文件。

(3)参数及子句说明

· *FileName*：指定要删除的文件的名字，可以带通配符 * 和?；

· RECYCLE：指明文件并非立即从磁盘上删除而是被放入 Windows 回收站中。缺省则立即从磁盘删除。

(4)注意事项

· 如果文件名中包含有空格，应将文件名用引号括起来。

· 用该命令删除的任何文件都不能被找回。甚至在 SET SAFETY 命令被设置为 ON 的情况下，在文件被删除之前也不会给出警告信息。

· 要删除的文件必须是关闭的。如果要删除的文件的驱动器或卷标、目录与缺省的驱动器或卷标、目录不同，则文件名中必须包含路径。文件的扩展名也必须包含。

· 如果要删除的文件是一个数据库表，在删除该文件之前，应先将表从库中移出。移出的方法是在使用 DELETE FILE 命令之前先使用 REMOVE TABLE 命令。如果要删除的表附有 ".fpt"备注文件，则应保证该文件也被删除。

· 如果指定的文件并不存在，这个命令不生成错误信息。

· DELETE FILE 命令和 ERASE 命令作用相同。

文件的删除也可使用 DOS 命令：

! |RUN DEL|ERASE *FileName*

例 4.67　将上例中在 D 盘的 ys 文件夹下名字为"bf"的各类文件全部删除。

```
CLOSE ALL
! DEL      d:\ys\bf.*
```

或

```
ERASE     d:\ys\bf.*
```

2. 菜单方式删除文件

在菜单方式下删除文件，对于数据库表可以通过数据库操作来完成，对于其他文件可以通过项目管理器来实现。现以利用项目管理器删除表文件 xsqkb1.dbf 为例说明文件的删除步骤。

S1：在项目管理器中，选择要删除的表文件：xsqkb.dbf。

S2：→[移去]↓ 删除信息提示。

S3：→[删除]。

注意 通常信息提示对话框有三个按钮，[删除]表示将文件彻底从磁盘上删除；[移去]表示仅将表从数据库中移出，使之成为自由表；[取消]则表示取消此次操作。

4.9 思考与练习

一、选择题

1. 在 Visual FoxPro 中，逻辑删除表中性别为“女”的命令是(　　)。

A) DELLETE FOR 性别='女'　　B) DELLETE 性别='女'

C) PACK 性别='女'　　D) ZAP 性别='女'

2. 打开数据表文件后，设当前记录指针指向的记录号为 100，要使指针指向第 20 号记录，应使用命令(　　)。

A) LOCATE 20　　B) SKIP －80　　C) SKIP 20　　D) SKIP 80

3. 打开数据表文件，执行 LIST 命令后，记录指针指向(　　)。

A) 最后一条记录　　B) 文件末　　C) 指针未移动　　D) 第一条记录

4. 能显示当前 CJ.DBF 表中数学不低于 60 分又不超过 85 分的记录的命令是(　　)。

A) LIST FOR 数学>=60.AND. 数学<=85

B) LIST FOR 数学>=60.OR. 数学<85

C) LIST FOR 数学>=60.OR. 数学<=85

D) LIST FOR 数学>=60.AND. 数学>85

5. 在 DISPLAY 命令中缺省范围及条件短语，则显示(　　)。

A) ALL　　B) RECORD 1　　C) 8　　D) 当前记录

6. 将当前数据表文件 XSQK.DBF 中性别为"女"的记录复制为 XSQK2.DBF，应使用命令(　　)。

A) COY TO XSQK2 FOR 性别="女"

B) COY TO FOR 性别="女"

C) COY TO XSQK2.DBF WHILE 性别="女"

D) COY TO XSQK2 FIFL 性别="女"

7. 将两个结构相同的 SDADBF 和 SDBDBF 文件合并的命令是(　　)。

A) USE SAD
　APPEND

B) USE SAD
　APPEND FROM SDB SDF

C) USE SAD
　COPY TO SDB

D) USE SAD
　APPEND FORM SDB

8. 要为当前表所有职工增加 100 元工资，应该使用命令(　　)。

A) CHANGE 工资 WITH 工资+100

B) REPLACE 工资 WITH 工资+100

C) CHANGE ALL 工资 WITH 工资+100

D) REPLACE ALL 工资 WITH 工资+100

9. 以下关于自由表的叙述,正确的是(　　)。

A) 全部是用以前版本的 FoxPro (FoxBASE) 建立的表

B) 可以用 Visual FoxPro 建立,但是不能把它添加到数据库中

C) 自由表可以添加到数据库中,数据库表也可以从数据库中移出成为自由表

D) 自由表可以添加到数据库中,但数据库表不可以从数据库中移出成为自由表

10. 在已打开的表文件中有"姓名"字段,此外又定义了一个内存变量"姓名",要把内存变量姓名的值传送给当前记录的姓名字段,应使用命令(　　)。

A) 姓名=A->名字　　B) REPLACE 姓名 WITH M->名字

C) STORE M->姓名 TO 姓名　　D) GATHER FROM M->姓名 field 姓名

二、填空题

1. 在当前记录后插入一个空白记录,其操作命令是_____。在当前记录前插入一个空白记录的命令是_____。

2. 在 VFP 中,指定从当前记录开始直到表文件的最后一条记录进行操作的范围关键字是_____。操作结束后记录号是_____。

3. 设当前盘为 C 盘,打开 A 盘上的表文件 TEXT.DBF 使用的命令是_____。

4. 以学生简况表为例,完成下列问题:

1)若当前记录号是 5,当执行 APPEND BLANK 命令追加一条空白记录后,该空白记录的记录号是_____。

2)若当前记录号是 4,当执行 INSERT BEFORE BLANK 命令增加一条空白记录后,该空白记录的记录号是_____。

3)若当前记录号是 4,当执行 INSERT BLANK 命令增加一条空白记录后,该空白记录的记录号是_____。

5. 当使用 LIST 命令显示表文件记录时,如果不显示记录号,必须在命令中使用关键字_____。

6. 以书中xsqkb.dbf文件为例,按下列要求写出相应的命令:

1)显示所有姓"马"的学生的记录:_____。

2)显示年龄大于 27 岁学生的记录:_____。

3)显示年龄大于等于 27 岁的女学生的记录:_____。

4)显示所有未婚的记录:_____。

5)显示所有姓名中含有"锋"的学生的记录:_____。

三、上机题

1. 分别利用菜单操作、VFP 命令、SQL 命令创建三个表文件,各自的数据模式为:

xs (学号 c(7),姓名 c(8),性别 c(2),出生日期 D)

kc (课程号 c(6),课程名 c(20))

cj(学号 c(7),课程号 c(6),成绩 n(5,1))

在课程名中,各自录入 10 条记录,记录内容自定,但至少应有:高等数学、英语、VFP9 程

序设计等课程的记录,以备后面各章使用。

2. 为了不在后面的操作中破坏这三个表,先将 xs 表复制成 xs1,然后再对 xs1 表进行 3～5 步操作。

3. 分别利用菜单操作、VFP 命令、SQL 命令对记录进行逻辑删除。

4. 分别利用菜单操作、VFP 命令恢复记录。

5. 分别利用菜单操作、VFP 命令、SQL 命令对记录进行物理删除。

6. 利用菜单操作,为表 xs、kc 分别按学号、课程号创建结构复合索引的候选索引,为 cj 表按学号、课程号创建两个结构复合索引的普通索引。

7. 利用命令操作,为 sx.dbf 建立以姓名为索引关键字的一个单索引文件:xm.idx;以性别为索引关键字创建一个结构索引文件的索引标记:xb;以出生日期为索引关键字的非结构的复合索引文件:csrq.cdx,索引表记:dc。

8. 创建三个表的临时关联,并按:学号、姓名、课程号、课程名、成绩的排列次序显示内容。

第 5 章　数据库的基本交互式操作

在 Visual FoxPro 中，一个数据库常常包含若干个具有相互逻辑联系的表。这些表各自从不同的方面反映实体的某些特性，结合起来则可更全面客观地反映实体的各种属性。数据库(database)与表(table)是两个不同的数据实体，数据库可以管理表、视图等数据实体。数据库还提供了数据字典、各种数据保护和数据管理功能。

在 Visual FoxPro 中，表分为自由表和数据库表两种，它们之间有一定的差别。但是不管是自由表还是数据库表，它们都是真实存放数据的文件。而数据库所记载的仅仅是这些数据库表的访问路径、数据字典的信息以及数据库表之间的关联信息。图 5－1 给出了 Visual FoxPro 9.0 中数据库的信息结构。

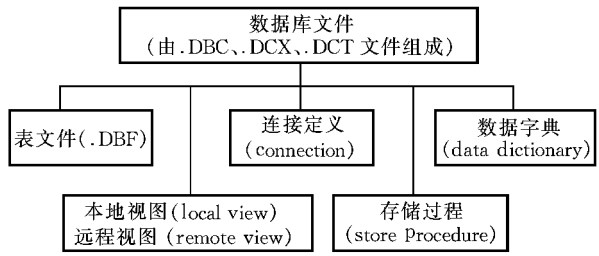


图 5－1　数据库的基本结构

上面的数据库的基本结构图中，数据库的各种对象，只有表可以是独立存储的文件，而其他数据库对象的信息，则完全存储在数据库中。

5.1　创建数据库

5.1.1　数据库设计的过程

创建一个比较完善的数据库，可以为以后访问所需要的信息奠定良好的基础。根据软件工程的思想和数据库的理论，创建数据库一般应遵照下面的步骤进行。

(1)认真调查研究，确定建库目的。此步用来确定需要 Visual FoxPro 保存哪些最基本的信息。

(2)统筹安排分解，明确独立主题。此步是在第一步的基础上，把创建数据库所需要的基础信息划分成独立的主题，每个主题都可以是数据库中的一个表。

(3)按照主题特点,划分各自字段。主题明确之后,所需要的表就确定了,此时需要进一步确定每个表中所要存储的信息,这些信息要以字段的形式在表中出现。

(4)根据工作需要,确定表间关联。根据用户需求,结合表的实际确定建立表之间的相互关联关系,这种关联关系是一种永久的关系。在必要的时候可为表增加某个字段,以便建立各种关系。

(5)反复调整设计,确保满足要求。设计有一个反复调整的过程,只有认真分析,不断改进才能达到用户的要求,创建出高效完善的数据库来。

5.1.2 创建数据库

创建数据库的方法有两种,既可以利用"新建"菜单创建数据库文件,也可以使用命令创建数据库文件。菜单方式创建数据库不再介绍。

1. 命令方式

(1)命令格式

CREATE [*DatabaseName*|?]

(2)功能

创建并打开一个数据库。

(3)参数和子句说明

· *DatabaseName*:指定要创建的数据库的名字。数据库文件按带有扩展名".dbc"的文件形式存储。

(4)注意事项

① 一个数据库将伴随生成一个数据库备注文件(.dct)和一个数据库索引文件(.dcx)。

② 不管 SET EXCLUSIVE 的设置如何,数据库都是以独占方式打开的。由于 CREATE DATABASE 的作用是在创建好数据库后立即打开它,因此用户不需要使用后面的 OPEN DATABASE 命令打开它。

③ 使用 CREATE DATABASE 命令不能自动地将数据库添加到一个项目中,即使项目管理器是打开的。用户必须明确地添加数据库到应用项目中。

例 5.1 用数据库创建命令创建一个数据库,库名"xsglsjk1.dbc"

```
CREATE DATABSE xsglsjk1
```

5.2 数据库的打开与关闭

5.2.1 数据库文件的打开

1. 菜单方式打开数据库

和利用菜单打开任何文件的方法相同,其实质是打开数据库设计器,无须再叙述。

2. 命令方式打开数据库

(1)命令格式

OPEN DATABASE [*FileName* | ?]

[EXCLUSIVE | SHARED] [NOUPDATE] [VALIDATE]

(2)功能

打开一个数据库。

(3)参数和子句说明

· EXCLUSIVE:以独占共享方式打开数据库。当数据库是以独占方式打开时,则不允许其他用户访问它。

· SHARED:数据库以共享的方式打开。此时,允许其他用户也可以访问它。如果用户并未使用 EXCLUSIVE 或 SHARED 关键字,则当前的 SET EXCLUSIVE 设置将决定数据库以哪种方式打开。

· NOUPDATE:以只读方式打开数据库。如果缺省 NOUPDATE,则数据库以读/写方式打开。

· VALIDATE:确保对数据库的引用是正确有效的。Visual FoxPro 将检查要引用的表及其索引在磁盘上的数据库中是否有效。同时检查要引用的字段和索引标识在表和索引文件中是否存在。

(4)注意

① 当数据库打开时,尽管所有包含在该库中的表都是可用的,但是表并未打开。

② 当本库内的表名和库外的表同名时,如执行 USE 命令,本库内表优先。

③ 当数据库被以独占方式打开时,不能再被另一用户打开。

④ 如果数据库在未被关闭的情况下连续打开两次,则数据库保留第一次打开时的各种设置。要改变设置,必须先关闭数据库,然后再使用带有新的设置的 OPEN DATABASE 命令打开它。

例 5.2　用命令打开数据库"xsglsjk1.dbc"。

```
OPEN DATABASE xsglsjk EXCLUSIVE
```

3. 命令方式打开数据库设计器

(1)命令格式

MODIFY DATABASE [*DatabaseName* | ?] [NOWAIT] [NOEDIT]

(2)功能

在数据库设计器中打开数据库,以便交互地修改。

(3)参数和子句说明

· NOWAIT:在打开数据库设计器后程序将继续执行。程序不等待数据库设计器被关闭而是继续执行 MODIFY COMMAND NOWAIT 语句下面的程序行。缺省 NOWAIT 子句,则程序将直到数据库设计器关闭后才能继续执行。NOWAIT 子句仅在程序中有效,以后均为此意。

例 5.3　用命令打开数据库数据库设计器及数据库"xsglsjk.dbc"。

```
MODIFY DATABASE xsglsjk
```

4. 设置当前数据库

(1)命令格式

```
SET DATABASE TO [DatabaseName]
```

(2)功能

设置一个打开的数据库为当前数据库或非当前数据库。

例 5.4 将数据库“xsglsjk1.dbc”设置为当前数据库。

```
SET DATABASE TO xsglsjk1
```

5.2.2 数据库文件的关闭

1. 菜单方式关闭数据库设计器

用菜单方式可以关闭当前打开的数据库设计器,它和 Windows 关闭文件的操作方法相同。

2. 命令方式关闭数据库

(1)命令格式

```
CLOSE [ALL | DATABASES [ALL] ]
```

(2)功能

关闭文件。

(3)参数和子句说明

· ALL:关闭所有打开的文件,并选择工作区 1 为当前工作区。

· DATABASE [ALL]:关闭当前数据工作区内的当前数据库和它的表文件。如果无当前数据库,则关闭所有工作区中的自由表、索引和格式文件,并且选择 1 号工作区。

例 5.5 用命令关闭数据库“xsglsjk1.dbc”。

```
SET DATABASE TO xsglsjk1      && 将 xsglsjk1 设置为当前打开的数据库
CLOSE DATABASES
```

5.3 数据库中表的添加与移去

在 5.1 节,创建了一个空白数据库,要使它真正成为有用的数据库,还必须向其中添加表。一个表只能隶属于一个数据库,一个数据库可以包含多个表。当表已经属于一个数据库时,要想让它成为另一个数据库的表,就必须先将该表从原数据库中移出使它成为自由表,然后再添加到希望的数据库中去。

5.3.1 向数据库中添加表

1. 菜单方式

设有一个空白数据库“xjsjk.dbc”,现以向该库添加表为例讲解向数据库添加表的步骤。

S1:打开数据库“xjsjk.dbc”。

S2:→[数据库]→[添加表]↓ 打开,选表,如xsqkb.dbf,→[确定]。

S3:重复 S2,把 kcdmb.dbf、xszhjfb.dbf、xscjb.dbf 都添加到数据库中,结果如图 5-2 所示。

以上功能也可以在“数据库设计器”中的空白处单击右键,在快捷菜单中选择[添加表]来实现。还可以单击“数据库设计器”工具栏中的“添加表”完成。

图 5－2　添加表之后的“数据库设计器”窗口

2. 命令方式

(1)**命令格式**

ADD TABLE *TableName* | ? [NAME *LongTableName*]

(2)**功能**

添加自由表到当前的数据库中。

(3) **参数和子句说明**

TableName:**指定要添加的表名。**

NAME *LongTableName*:**为表定义一个长名。长名最多** 128 **个字符,它可以出现在短表名出现的地方。**

例 5.6　将自由表“rkjsb.dbf” **添加到**“xjsjk.dbc”**中,使其成为数据表。**

```
OPEN DATABASE xjsjk
ADD TABLE rkjsb
```

5.3.2　从数据库中移去表

当数据库不再需要一个表,或在其他数据库中需要使用此表时,可以从该数据库中移去此表。被移去的表成为自由表,而在原数据库中存储的有关该表的信息将不再被保存。

1. 菜单方式

现以从当前数据库“xjsjk.dbc”**移去刚刚添加的表**“rkjsb.dbf”**为例,讲述从数据库中移去表的步骤。**

S1:**打开数据库**“xjsjk.dbc”**及数据库设计器。**

S2:**选定表**“rkjsb”。

S3:→[**数据库**] → [**移去**]↓ 信息框 1,**如图** 5－3。

图 5－3　“移去”信息框之一

S4:→[移去] ↓ 信息框 2,如图 5-4 所示。

图 5-4 “移去”信息框之二

S5:→[是],表被从当前库中移去,成为自由表。

2. 命令方式

从数据库中移去表的命令是 REMOVE TABLE。

(1)命令格式

REMOVE TABLE *TableName* | ? [DELETE] [RECYCLE]

(2)功能

将数据库表从数据库中移去或从磁盘上删除。

(3)参数和子句说明

- DELETE:指定从数据库中移去表并将表从磁盘上永久删除。
- RECYCLE :指定将表移动到 Windows 回收站中,而不是永久删除掉。

注意 REMOVE TABLE 移去所有的基础表、缺省值、与其他表的关联关系。它会影响到当前数据库的其他表,如果这些表与要删除的表之间有关联或一致性规则控制的话。当表从数据库中删除后,这些一致性规则和关联将不复存在。如果 SET SAFETY 设为 ON,Visual FoxPro 提示用户要从数据库中进行表删除。

例 5.7 在 xjsjk.dbc 中移去课程代码表“kcdmb.dbf”:

```
OPEN DATABASE xjsjk
REMOVE TABLE kcdmb
```

5.4 建立数据库表的永久关系

在第 4 章中曾讲过表的临时关联。本节将讲解另一种存在于数据库表之间、存放在数据库文件中、在数据库关闭之后其关系信息不会丢失的关系——永久性关系。

5.4.1 永久关系的特性

永久性关系和临时关系不同,它具有下面的特性:

① 在查询设计器和视图设计器中,将自动作为默认的条件。

② 在数据库设计器中,它显示为联系表索引的关系线。

③ 作为表单和报表的默认关系,它会出现在数据环境设计器中。

④ 可用来存储数据库表之间的参照完整性信息。

永久关系可分为一对一、一对多和多对多三种关系。

要建立数据库表之间的永久关系，必须事先对各个表根据它们的公共字段进行索引。建立关系的两表，一个称为父表，另一个称为子表。

若父表是用两个表的公共关键字建立了主索引，子表用该关键字建立了主索引或候选索引，根据主索引与候选索引都具有："索引关键表达式值惟一"的特点，它们间的关系就只能是一对一的关系。

如果子表建立的是普通索引或惟一索引，由于它们的索引关键表达式的值允许不惟一，此时所建立的关系就必然是一对多的关系。

如果两个表都用公共关键字建立了普通或惟一索引，则它们之间就只能创建多对多的关系了。

5.4.2　建立数据库中表之间的永久关系

1. 菜单方式

在菜单方式下，只能建立两个表之间的一对一或一对多关系而不能建立表间的多对多关系。下面以在表"xsqkb.dbf"和"xscjb.dbf"间建立一对多关系为例说明菜单方式下创建关系的步骤。

S1：打开数据库设计器及数据库"xjsjk.dbc"。

S2：对表"xsqkb.dbf"、"xscjb.dbf"用"学号"字段分别建立主索引、普通索引。

S3：自父表"xsqkb.dbf"的主索引"学号"处按下鼠标左键，拖动到子表"xscjb.dbf"的普通索引"学号"处，释放鼠标，则一对多的关系创建成功。结果如图 5-5 所示。

图 5-5　在数据库中定义永久关系

仔细观察关系连线，会发现关系线在父表一端不分叉，在子表一端却分为三个叉，不分叉代表"一"，分叉代表"多"。如果关系线两端都不分叉，表示一对一关系。

2. 命令方式

在 Visual FoxPro 中不提供专门的永久性关系的创建和编辑命令，但在 SQL 中有这方面的命令。这里我们先给出使用最简化的 SQL 命令创建永久关系的一个例子。

例 5.8　用 ALTER TABLE - SQL 命令创建"xsqkb"和"xscjb"之间的永久关系。

```
OPEN DATABASE xssjk
ALTER TABLE xscjb ADD FOREIGN KEY 学号 TAG 学号 REFERENCES xsqkb
```

5.4.3 编辑永久关系

下面以编辑表“xsqkb.dbf”和“xscjb.dbf”间已建立的一对多关系为例说明菜单方式下编辑关系的步骤。

S1:**打开数据库设计器及数据库**“xjsjk.dbc”。

S2:→**关系线**↓**编辑关系**(Edit Relationship),**如图** 5-6 **所示。**

图 5-6 **“编辑关系”对话框**

S3:**分别在父表和子表**(Related table)**下面的索引列表框选择所需要的索引,例如在**“xsqkb”**下选候选索引**“xh”,**在**“xscjb”**下选普通索引**“xh”,(这两个索引都是事先创建好的)结果如图** 5-7 **所示。**

S4:→[**确定**(OK)],**编辑完成。结果如图** 5-8 **所示。**

图 5-7 **编辑关系对话框**

图 5-8 **关系编辑结果**

关系的修改也可通过右击关系线,调出快捷菜单的方法来进行。

5.4.4 删除永久关系

要想删除数据库间的关系,最快捷的方法是:单击关系线,使关系线变成粗黑线。然后按 DEL **键。也可用右击关系线,弹出快捷菜单,单击[删除关系]的方法来删除关系。**

5.4.5 设置参照完整性

参照完整性用来定义主码和外码之间的引用规则,即一个数据库的不同表之间主关键字和外部关键字之间的引用规则。建立了参照完整性的数据库表,用来控制记录如何在相关表中被更新、删除、增加,以实现库中各表之间记录数据的数据库一致。

下面以对“xjsjk”数据库建立参照完整性为例，讲述建立参照完整性的步骤。

S1：打开 sjsjk.dbc 及数据库设计器。

S2：对表“xsqkb.dbf”以“学号”字段为关键字建立主索引，对“xscjb.dbf”、“xszhjfb.dbf”分别以“学号”为关键字建立候选索引。

S3：以“xsqkb.dbf”为父表，建立与其他两表之间一对一的永久关系。如图 5－9 所示。

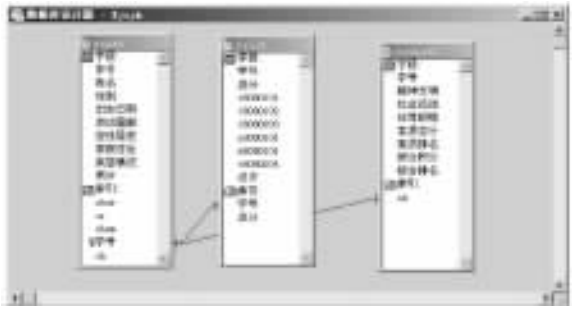


图 5－9　参照完整性生成器

S4：→[数据库]→[清理数据库]。

S5：→[数据库]→[编辑参照完整性]↓ 参照完整性生成器，如图 5－10 所示。

图 5－10　数据库表之间的永久关系

S6：设置参照完整性规则。

参照完整性包括有：更新规则、删除规则、插入规则。更新规则和删除规则各有三个选项，分别是：级联、限制、忽略，插入规则选项中无级联。它们各自意义如下。

(1)更新规则

规定当父表的关键字值被修改时，相关的子表应遵循的规则。

① 级联：当主表中的主关键字或候选关键字值被修改时，自动更新子表中所有的相关记录。

②限制：如子表中有相关的记录时，则禁止更新主表中主关键字段或候选关键字段的值。

③忽略：不管子表中是否有相关的记录，都允许更新父表的记录。

(2)删除规则

规定当父表的记录被删除时，相关的子表应遵循的规则。

① 级联：当主表中的记录被删除时，自动删除子表中所有的相关记录。

②限制：如子表中有相关的记录时，则禁止删除主表中的记录。

③忽略：不管子表中是否有相关的记录，都允许删除父表的记录。

(3)插入规则

规定当向子表中插入新的记录或修改其已存在的记录时,应遵循的规则。

① 限制:如果在父表中没有相匹配的记录,则禁止向子表插入记录。

②忽略:不管父表中是否有相匹配的记录,都允许向子表插入记录。

今对"xsqkb"和"xscjb"、"xscjb"和"xszhjfb",分别将更新、删除、插入规则分别选择为:级联、限制、限制,如图 5-11 所示。

图 5-11 一致性规则设置

S7:→[确定]↓ 参照完整性生成器,如图 5-12 所示。

S8:→[是]← 数据库设计器,参照完整性规则设置完成。

此时,所设置的参照完整性规则将在数据库表数据的更新、删除、插入中起约束作用。例如,试图将"xsqkb"中的"010101"号记录删除,由于子表"xscjb"中存在对应的记录,则系统会给出如图 5-13 所示的错误信息。

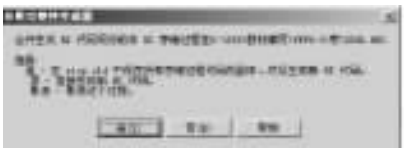


图 5-12 参照完整性生成器之二

图 5-13 父表记录删除失败

5.5 数据字典

在 Visual FoxPro 9.0 中,每一个数据库都有一个数据字典(data dictionary),用于存放数据库表的长名、字段的长名、有效性规则和触发器,以及数据库对象的定义等信息。存储在数据字典中的信息称为元素。由于数据字典的存在,数据库表在功能上大大强于自由表。下面介绍数据字典的功能及其应用。

5.5.1 设置数据库表的长名和表的注释

数据库表的长表名,指数据库中在表名出现的地方,可以用一个长名来代替原表的名字,以便更加能见其名而知其意。例如对于数据库表"xsqkb.dbf"可起一个长表名"学生情况表.dbf"。长表名最多 128 个字符。定义了长表名后,在数据库表出现的场合(例如数据库设计器、查询设计器、视图设计器、浏览窗口的标题栏等),表名都将以长名的形式给出。

为表设置长名可在菜单方式和命令方式下进行。

1. 菜单方式

现以为数据库表“xsqkb.dbf”起一个长表名“学生情况表.dbf”为例讲解菜单方式下设置表的长名和注释的步骤。

S1：打开数据库“sjsjk.dbc”和数据库设计器。

S2：双击“xsqkb.dbf”，打开该表。

S3：打开表设计器。→[表]↓ 表设计器，如图 5－14 所示。

图 5－14　表设计器对话框

S4：在“表名”文本框中输入长表名“学生情况表”。

S5：在“表注释”文本框输入注释文字，例“这是一个虚构的学生情况表”。

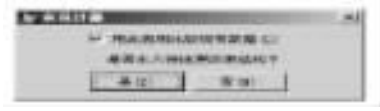


图 5－15　表设计器对话框之三

S6：→[确定] ↓ 表设计器，如图 5－15 所示。

S7：→[是]← 数据库设计器，此时表的长名和注释设置完成，用户会发现在数据库设计器中原来的短表名“xsqkb”已被长表名“学生情况表”所取代。而表注释会在下次打开表设计器的“表”选项卡时显示出来。

2. 命令方式

对于设置表的长表名，所有含有 NAME 子句的命令，均可为数据库表起一个长名。如 SQL 中的定义表命令 CREATE TABLE。

例如：

CREATE TABLE zggz NAME 职工工资表 (职工号 C(8)，基本工资 I，岗位工资 I)

就将在为当前打开的数据库创建一个表“zggz.dbf”的同时，为该表起一个长表名“职工工资表”。

注意　长表名是数据库表所特有的特性之一，自由表是没有长表名的。当表一旦从数据库中移出，则长表名将不复存在。

5.5.2　设置字段的标题和注释

和为数据库表设置长表名和注释相类似，可以为库表的字段设置标题和注释。字段标题

在表浏览时可见,字段注释在打开表设计器的“字段”选项卡时可见。菜单方式下设置的步骤也基本相同,只不过应在表设计器的字段选项卡中进行,用户可自己实践。

关于命令方式设置字段的标题将在 SQL 中讲解。

5.5.3 设置字段的有效性规则

数据库表字段的有效性规则,包括三个选项:规则、信息、默认值。

(1)规则:字段的值域。这是一个条件表达式,运算的结果是一个逻辑值.T. 或.F. 。在插入记录或修改字段值时被激活,对输入数据的正确性进行检查。如果不正确,则给出提示信息,并拒绝接受。

(2)信息:当输入错误时应给出的提示信息,这是一个字符串。

(3)默认值:当向表中插入或添加一条新记录时,字段的默认值。除非用户输入新值,否则默认值一直存于该字段中。

菜单方式下设置字段的有效性规则,也在表设计器的字段选项卡中进行。关于命令方式设置字段的有效性规则将在 SQL 中讲解。

5.5.4 设置字段值的格式码和掩码

字段值的格式码和掩码都可用来控制字段的输出和输入格式。但掩码仅能控制所在位置的一个字符的输入或输出格式,而格式码则控制整个字段的输入和输出格式。

例如电话号码格式可表示为:(9999)9999-9999,工资的格式可表示为:$999,999.99 等等。它们都要借助于掩码和格式码方可控制其正确的输入或显示。

1. 掩码

表 5-1 给出了“输入掩码”属性的掩码代码及其功能。

表 5-1 掩 码

掩码代码	功 能
A	只能输入英文字母或汉字
L	只能输入英文字母 T 或 F
N	只能输入英文字母、汉字或数字
X	允许输入任何字符
Y	只能输入英文字母 Y、y、N、n,并自动将小写转为大写字母
9	只允许输入字符型数据和数值型数据的数字
#	只能输入数字、空格、正负号和英文句号“.”
!	可输入任何字符,但所有输入的英文字母将全部转为大写
$	将数值型数据在其最前面的第一个位置上加一个货币符“$”显示
$ $	将数值型数据紧前面位置上加一个货币符“$”显示
*	自数值型数据之前,先是数个星号“*”,与“$”和用可保护数据
.	指定小数点的位置
,	分隔小数点左边的数字,例如以 3 位分节金额的表示方式

2. 格式码

表 5－2 给出的是“格式”码属性的格式代码及其属性。

表 5－2　格式码

掩码代码	功　能
A	只能输入英文字母或字母，且不允许空格与标点符号
D	依照目前的 SET DATE 命令所设置的格式来编辑与显示日期和日期时间型数据
E	依照 BRITISH 格式来编辑与显示日期和日期时间型数据
K	当鼠标移动到该字段时，便选取整个字段以便编辑
L	以 0 补满数值型数据前面的空格
M	将掩码中用“,”分隔的不同内容作为选项，按空格键可选择其中的一个选项为字段的值
R	掩码中出现的非格式化代码字符并不存于字段中，否则将存于字段中
T	删除字符串的前缀与后缀空格
YS	表示采用在“控制面板”的区域选项设置工具中所设置的简短日期格式来表示日期或日期时间值
YL	表示采用在“控制面板”的区域选项设置工具中所设置的完整日期格式来表示日期或日期时间值
Z	如果数值型字段的内容为 0，则将它显示为空格
!	可输入任何字符，但所有输入的英文字母将全部转为大写，仅适用于字符型数据
^	将数值型数据以科学计数的格式表示，仅用于数值型数据
$	将数值型数据以货币格式显示

3. 掩码和格式码应用举例

掩码和格式码最常用于对字符型和数值型数据的输入输出格式的控制，而且在许多时候，要通过掩码和格式码的联合设置，才能实现数据的正确输入输出。表 5－3 给出了常用的一些掩码和格式码的用法和功能。

从表 5－3 中可以看出，许多情况下，不但使用了掩码而且使用了格式码。

表 5－3　字段输入输出格式控制

数据类型	输入掩码	格式码	输入值	显示结果	说明
C(10)	! AAAXXX999		abcd－ef123	Abcd－ef123	
C(10)		!	abcd－ef123	ABCD－EF123	
C(10)		T!	abcdef	ABCDEF	
C(11)	(9999)999－9999	R	09132133053	(0913)213－3053	格式码 R 将非掩码的“(”、“)”、“－”，仅显示，但不占字符位置

续表 5-3

数据类型	输入掩码	格式码	输入值	显示结果	说明
C(11)	(9999)999-9999		09132133053	(0913)213-3	非掩码的“(”、“)”、“-”存于数据中占字符位置
C(6)	教授,副教授,讲师	M	按一次空格键,回车	副教授	按空格键,可在三个选项间切换,选中其中一项作为数据的输入值
N(10,2)	999999.99		-12345.12	-12345.12	
N(10,2)	$999999.99		1234.12	$ 1234.12	
N(10,2)	$$999999.99		1234.12	$1234.12	
N(10,2)	$********.99		1234.12	$***1234.12	
N(10,2)	$99,999,999.99	Z	1234.12	$ 1,234.12	数值非 0 时正常显示
N(10,2)	$99,999,999.99	L	1234.12	$000001,234.12	
N(10,2)		^	1234.12	1.23412E+3	
N(10,2)		Z	0.0		数值为 0 时显示为空格
N(10,2)		$	1234.12	$1234.12	

4. 菜单方式设置输入掩码和格式码

字段的输入掩码和格式码的设置步骤与字段的有效性规则设置方法相同,在打开数据库和表,选定表字段后,只要在分别在“格式”和“输入掩码”文本框输入有关的代码即可。

5. 命令方式设置输入掩码和格式码

在命令方式下设置输入掩码和格式码通过@…SAY…GET 语句来实现。不过在该命令中,将掩码称为标准符,将格式码称为功能符。下面给出它们各自的最简命令格式。

(1)格式输入命令

① 命令格式

@ *nrow*,*ncolumn* [SAY *cPrompt*] GET *variableName*
[PICTURE “[@*cFunctionSymbol*] [*cStandardSymbol*]”]

② 功能

按指定的格式为变量赋值。

③ 参数和子句说明

- *nrow*,*ncolumn*:SAY 子句开始显示的行和列位置。
- SAY *cPrompt*:提示字符串。缺省时无提示。
- GET *variableName*:要赋值的变量的名字,它可以是内存变量也可以是字段名变量。如果是内存变量,应事先有初值,该初值将决定内存变量的类型和长度;如果是字段名变量,则字段所在的表应事先打开。
- PICTURE “[@*cFunctionSymbol*] [*cStandardSymbol*]”:指定输入时的功能符和标准

符(即格式码和掩码)。其中在“@”之后紧跟的是功能符串,功能符串之后是标准符串。在功能符串和标准符串之间至少一个空格,以示区分。它们既可以同时出现,也可以仅有一个。

注意:在一系列的“@…SAY…GET”语句之后,应有一个“READ”语句,激活它们方可进行格式输入。如无 READ 语句则仅显示变量的原值而不能输入新值。

本语句常用于格式文件中。

例 5.9　下面的语句组可编辑表“xsqkb”中的“学号”、“姓名”两个字段。

```
USE xsqkb
@10, 5 SAY "学号:" GET 学号 PICTURE "999999"
@10,25 SAY "姓名:" GET 姓名 PICTURE "AAAAAAA"
READ
```

(2)格式输出命令

① 命令格式

@*nrow*,*ncolumn* SAY *Expression*

[PICTURE "[@*cFunctionSymbol*] [*cStandardSymbol*]"]

② 功能

按指定的格式显示变量。

③ 参数和子句说明

- *Expression*:要显示的表达式。

其他子句和参数与前面意义相同。

例 5.10　用格式语句实现表 5-3 中一些行的显示结果。

```
@ROW()+1,5 say "09132133053" picture "@R (9999)999-9999"   && 显示:(0913)213-3053
St=space(6)
@ROW()+1,5 get st picture "@m 教授,副教授,讲师"
READ    && 从下一行第五列起实现:教授,副教授,讲师的切换输入
@ROW()+1,5 say 1234.12 picture "$ * * * * * * * * .99"   && 从下一行第五列起显示:$ * * * 1234.12
```

5.5.5　设置记录的有效性规则

所谓记录的有效性检查是指对记录中各个字段进行某种运算的结果进行检查,判断其是否在合理的范围内。这种检查通过事先设置好的记录有效性规则来进行。当要追加的一条记录的所有字段录入完毕或已存在的一条记录的有关字段修改完毕而转向另一条记录时被激活。如合理则允许进入下一条记录,否则给出提示信息,要求重输。

记录有效性包含两个选项:规则、信息。

(1)规则:一般是算术表达式和关系表达式组成一个混合表达式。返回值为逻辑值:.T. 或.F. 。

(2)信息:一个字符串,当字段完整性检查出错时给出用户提示。

记录有效性规则的设置步骤与字段有效性规则设置相同,用户可自己予以实践设置。

5.6 多数据库操作

多数据库操作指同时打开多个数据库或引用关闭的数据库中的文件。必须指出的是即使同时打开了多个数据库,也只能有一个数据库被指定为当前数据库。

5.6.1 打开多个数据库

打开多个数据库的方法有菜单法和命令法两种。

在菜单方法是多次使用打开按钮。命令方法打开多个数据库,是多次使用 OPEN DATABASE 命令。

例 5.11 打开三个数据库。

```
OPEN DATABASE xjsjk
OPEN DATABASE xsglsjk
OPEN DATABASE 数据库 1
```

注意 最后一次打开的数据库为当前数据库。当前数据库的名字可以通过数据库函数 DBC()来得到。例如使用下面命令:

```
? DBC()
```

可得到:数据库 1

5.6.2 数据库中表的使用

要正确地引用其他数据库的表,可在表名之前加上数据库名和感叹号作为表的前缀。例如:

```
USE xjsjk! xsqkb
```

表示打开"xjsjk.dbc"数据库中的表"xsqkb.dbf"。

5.7 数据库的其他操作

5.7.1 浏览数据库结构

数据库文件为和数据库相关的每个表、视图、索引、标志、永久关系等保存了一个记录,同时保存了每个具有附加属性的表字段或视图字段的记录。它还有一个单独的记录,用来保存数据库的所有存储过程(将在程序设计一章中讲解)。

用户可以使用表设计器浏览数据库的结构,也可以使用 BROWSE 命令浏览数据库的记录。此时,数据库要使用 USE 命令打开,且必须带数据库的扩展名".dbc",否则系统会认为是打开表文件。

例 5.12 浏览数据库"sjsjk.dbc"的结构及内容。

```
CLAOSE ALL
USE xjsjk.dbc EXCLUSIVE
MODIFY STRUCTURE
```

BROWSE

上面命令组执行的结果会显示出数据库的结构和内容，图 5－16 和图 5－17 分别给出了数据库的结构及部分内容的显示结果。

图 5－16　用表设计器显示的数据库结构

图 5－17　部分数据库内容

注意　除非对数据库结构非常了解，否则请不要轻易使用 BROWSE 命令浏览数据库内容更不要修改数据库结构，否则会造成数据的永久性丢失，使数据库成为无效库。

5.7.2　删除数据库

通过项目管理器可以像删除表一样删除数据库。用数据库删除命令也可删除数据库。

(1)命令格式

DELETE DATABASE *DatabaseName* | ?
[DELETETABLES] [RECYCLE]

(2)功能

从磁盘上删除一个数据库。

(3)参数和子句说明

- *DatabaseName*：指定要从磁盘上删除的数据库的名字。该库必须是关闭的。
- DELETETABLES：从磁盘上删除数据库及其所包含的表。若缺省该关键字，则原数据库中所包含的表将变为自由表。
- RECYCLE：将要删除的数据库先放入 Windows 回收站中。

5.8　思考与练习

一、选择题

1. 扩展名为. DBC 的文件是(　　)。

A) 表单文件　　B) 数据库文件　　C) 表文件　　D) 项目文件

2. 对数据库表的结构进行操作，是在(　　)。

A) 表设计器环境下完成的　　B) 表向导环境下完成的

C) 表浏览器环境下完成的　　D) 表编辑环境下完成的

3. 在 Visual FoxPro 中，以只读方式打开数据库文件，命令中应包含子句(　　)。

A) EXCLUSIVE　　B) SHARED　　C) NOUPLATE　　D) VALDATE

4. 关于长表名和长字段名的表述正确的选项是(　　)。

A) 自由表可以使用长表名和长字段名,而数据库表不能

B) 数据库表可以使用长表名和长字段名,而自由表不能

C) 自由表可以使用长表名和短字段名

D) 自由表和数据库表都可以使用长表名和长字段名

5. 关于数据库表和自由表的区别,正确的选项是(　　)。

A) 可以为自由表添加标题和注释,而数据库表不能

B) 可以为自由表的字段设置默认值和输入掩码

C) 不能为数据库表的字段设置默认值和输入掩码

D) 可以为数据库表设置默认值、输入掩码、标题和注释

6. 关于字段级规则和记录级规则,表述正确的选项是(　　)。

A) 可以为自由表规定字段级规则和记录级规则,而数据库表不能

B) 可以为自由表规定字段级规则,但不能规定记录级规则

C) 可以为数据库表规定字段级规则和记录级规则,而自由表不能

D) 可以为数据库表规定字段规则,但不能规定记录级规则

7. 关于关键字、参照完整性和表之间的关系,表述正确的选项是(　　)。

A) 可以为数据库表设置主关键字、参照完整性和表之间的关系

B) 可以为自由表设置主关键字、参照完整性和表之间的关系

C) 可以为自由表设置主关键字,数据库表不能

D) 不可以为数据库表设置主关键字、参照完整性和表之间的关系

二、填空

1. 创建数据库的命令是_____。

2. 数据库由_____、_____、_____文件组成。

3. 在菜单方式下打开数据库,实际上执行了_____、_____命令。

4. 用命令 DELETE DATABASE 删除数据库时,数据库中的表将被_____。

5. 字段的有效性规则包括_____、_____、_____项内容。

6. 数据记录的有效性规则包括_____、_____项内容。

7. 数据库中表之间的永久关系是存储在_____中。

8. 建立数据库文件之间的表之间永久关系的先决条件是_____。

9. 参照完整性规则包括_____、_____、_____。

三、上机题

1. 创建一个名为“xscjgl”的数据库。

2. 将第 4 章习题中自建的三个表添加到数据库中。

3. 为 xs、kc 表分别按:学号、课程号创建主索引。

4. 建立表间的永久关联关系。

5. 设置表间的参照完整性。

第 6 章　关系数据库标准语言 SQL

尽管 Visual FoxPro 提供了查询语句，然而在许多时候这种查询并不太方便，甚至需要用户对数据库进行一系列处理才能达到目的。因此在 Visual FoxPro 中又引入了另一种非常方便的查询——结构化查询语言。

6.1　SQL 概述

SQL 是英语“structured query language”的缩写。翻译成中文，是“结构化查询语言”。因其具有综合的、通用的、功能强大而又简单易学的特点，早在 1987 年就被国际标准化组织 ISO (International Organization for Standardization)批准为关系型数据库国际标准。它既可以用于大型数据库系统，又可以用于微机数据库系统，是数据库的通用语言。

SQL 是一种非过程化语言。它的大多数语句都是可独立执行的，可用来完成一个独立的操作，与上下语句无关。它既不是数据库管理系统软件，也不是应用软件开发语言，仅用于对数据库中的数据进行操作。

在前面的章节中，我们有意识地向读者介绍了一些 SQL 对数据的定义和操纵命令，其目的是为本章的学习奠定一定的基础。

不能简单地从字面上将 SQL 理解为仅限于查询。实际上它由数据定义语言、数据操纵语言、数据控制语言三部分组成。

1. 数据定义语言

SQL 的数据定义语言由 CREATE(创建)、DROP(删除)、ALTER(修改)命令组成，用来完成数据库结构的创建、删除和修改。前面各章节的学习中我们已有所接触。

2. 数据操纵语言

数据操纵语言是完成对数据操作的命令。它有 INSERT(插入)、DELETE(删除)、UPDATE(修改)、SELECT(查询)等组成。前三个命令在前面的章节中已作过简单介绍。

3. 数据控制语言

数据控制语言用来控制用户对数据库的访问权限。由 GRANT(授权)、REVOKE(收回)命令组成。由于 Visual FoxPro 9.0 没有管理权限，因此没有数据库控制语言的命令。

在 Visual FoxPro 9.0 中，基本覆盖了 SQL 的数据定义、数据操纵语言的两大部分。由于 SQL 语言是用于所有的大、中、小型数据库管理系统，因此学会 Visual FoxPro 9.0 的数据定义和数据操纵语言，就可以在诸如 SQL Server、Oracle、DB2 等大型数据库中进行同样的操作。

6.2 SQL 的数据操纵功能之一——数据查询

如前所述 SQL 数据操纵，包含了对表记录的插入(INSERT)、删除(DELETE)、修改(UPDATE)和查询(SELECT)四条命令。

6.2.1 SQL—SELECT 命令解析

1. 命令结构

所谓查询，指从一个或多个表或视图中查询数据。当使用一个 SQL SELECT 生成一个查询的时候，Visual FoxPro 会解析查询并从表或视图中找出所指定的数据。用户可以在命令窗口、Visual FoxPro 程序、或者使用查询设计器来生成一个 SQL SELECT 查询。

SQL 的查询命令功能强大，命令很长，选项很多，但只要认真剖析就会找到学习的捷径。概括起来，SQL 的查询命令由以下几个部分组成：

(1)查什么？这是查询要输出的结果，在 SQL 查询命令中将以 SELECT 子句给出。

(2)到哪儿查？这是查询的对象，是数据的来源。在 SQL 查询命令中将以 FROM 子句的形式给出。

(3)查询对象间存在着什么连接关系？在 SQL 查询命令中将以 JOIN 子句的形式给出。

(4)查询的条件是什么？在 SQL 查询命令中将以 WHERE 子句的形式给出，WHERE 子句也可用于表的等值连接。

(5)按何种格式输出查询结果？在 SQL 查询命令中将以 GROUP、HAVING、ORDER 子句的形式给出。

(6)把查询结果的输出到何处？在 SQL 查询命令中将以 INTO、TO 子句的形式给出。

(7)附加的显示选项。

这七个子句中第一个和第二个是最基本的。没有它们就构不成查询。第三和第四个子句可以构成各种各样复杂的查询来。此外还有其他一些特殊功能的子句，如 UNION 等。

2. 命令格式

在对 SQL 查询语句结构分析的基础上，下面给出 SQL 查询命令：

```
SELECT [ALL | DISTINCT] [TOP nExpr [PERCENT]] Select_List_Item [, ...]
    FROM [FORCE] Table_List_Item [, ...]
    [[JoinType] JOIN [DatabaseName!]Table [[AS] Local_Alias] [ON JoinCondition
        [AND | OR [JoinCondition | FilterCondition] ...]
    [WITH (BUFFERING = lExpr)]
    [WHERE JoinCondition | FilterCondition
        [AND | OR JoinCondition | FilterCondition] ...]
    [GROUP BY Column_List_Item [, ...]] [HAVING FilterCondition
        [AND | OR ...]]
    [UNION [ALL] SELECTCommand]
    [ORDER BY Order_Item [ASC | DESC] [, ...]]
        [INTO StorageDestination | TO DisplayDestination]
```

[PREFERENCE *PreferenceName*] [NOCONSOLE] [PLAIN] [NOWAIT]

6.2.2　创建基本查询

所谓基本查询是指基本结构为 SELECT…FROM…[WHERE…]的查询，如果数据来源仅是一个表，将是最简单的查询。

1. SELECT 子句

该子句用来指定在查询结果中要显示的字段、常量和表达式。在简单查询中，它将以

SELECT [ALL | DISTINCT] *Select_List_Item* [, ...]

格式出现。

其中所含参数和关键字的意义如下：

- ALL：指定输出所有记录，包括重复的记录。这是缺省值。
- DISTINCT：指定输出无重复结果的记录。
- *Select_List_Item*：指定在查询结果中所包含的列表项。列表的每一项，在查询结果中生成一列。它可以被指定为：

① 常量，它将在查询结果的每一行中显示。

② 用户定义的函数或子查询表达式。

③ Alias. *Select_List_Item* 格式，其中 Alias 是表的别名，当多表查询时对于同名字段使用，以示区别。

④ FROM 子句中给出的表的字段名。

一个 SQL SELECT 的子查询，注意子查询的 SELECT 语句必须用圆括号括住。

2. FROM 子句

这里，由于是简单查询，FROM 子句将以 FROM *TableName* 的形式出现，给出要查询的表名。

3. WHERE 子句

这里 WHERE 子句将以 WHERE *FilterCondition* 的形式出现，给出查询的条件。

例 6.1　从“xsqkb.dbf”中查询全体学生的学号、姓名、出生日期。

SELECT 学号，姓名，出生日期 FROM xsqkb

结果如图 6-1 所示。

图 6-1　部分字段查询结果

如果要查询表中所有各列的信息,则可使用通配符来完成。

例 6.2 查询"xsqkb.dbf"中全体学生的全部信息。

SELECT * FROM xsqkb

结果如图 6-2 所示。

图 6-2 全部字段查询结果

例 6.3 查询"xsqkb.dbf"中性别为"女"的学生的全部信息。

SELECT * FROM xsqkb WHERE 性别='女'

结果如图 6-3 所示,修改"xsqkb.dbf"的结构,增加一个"系别"字段,并录入该字段的值,结果如图 6-4。

图 6-3 例 6.3 查询结果

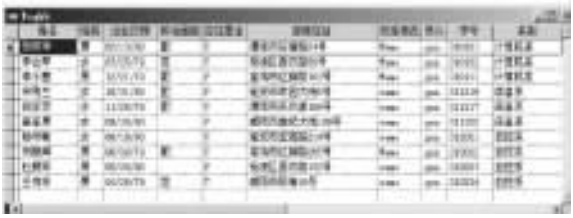


图 6-4 增加"系别"字段后的表

例 6.4 查询修改后的"xsqkb.dbf"中学号以"012"开头的学生所在的系。

SELECT DISTINCT 系别 FROM xsqkb WHERE 学号='012'

结果:自控系。

例 6.5　查询“xsqkb.dbf”中性别为“女”并且 1979 年后出生的学生的信息。

SELECT * FROM xsqkb WHERE 性别='女' AND YEAR(出生日期)>1979

结果如图 6-5 所示。

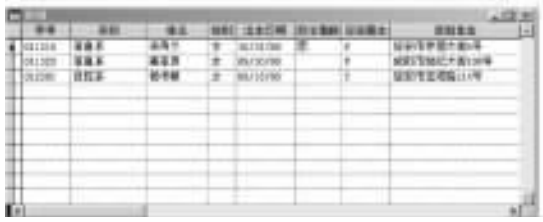


图 6-5　例 6.5 查询结果

6.2.3　创建内连接查询

前面的简单查询，涉及到了一个关系的投影、选择操作，而连接则是涉及到多个关系的另一类基本操作。

连接查询分为内连接和外连接两大类，内连接查询指连接以等值连接的形式给出的查询。所谓等值连接是指按照字段值对应相等所进行的连接。这是一种简单但却使用最普遍的连接查询。内连接的结果是只有满足连接条件的元组才出现在查询结果中。

连接操作将使用 JOIN 子句来实现。它隶属于 FROM 子句，其语法格式为：

[[*JoinType*] JOIN [*DatabaseName*!]Table [[AS] *Local_Alias*] [ON *JoinCondition*
　　[AND | OR [*JoinCondition* | *FilterCondition*] ...]

其中：

- [*DatabaseName*!]*Table*：**指定要建立连接关系的表名；**
- [AS] *Local_Alias*：**为表起一个局部别名；**
- ON *JoinCondition*：**指定表之间的连接条件；**
- *JoinType*：**表之间的连接类型。它可以是：**INNER | LEFT [OUTER] | RIGHT [OUTER] [FULL [OUTER]**四种格式中的任一种。内连接关键字是：**INNER，这是缺省值。

例 6.6　查询学生的：学号，姓名，总分。

SELECT xsqkb.学号，姓名，总分 FROM xsqkb INNER JOIN xscjb ON xsqkb.学号=xscjb.学号

结果如图 6-6 所示。

注意　在多表连接查询中，如果查询输出的字段为一个表所独有，则字段名前可不给出表名，否则应使用：“*TableName*.*fieldName*”的形式。

对于内连接也可通过在 FROM 子句中将所用表全部列出，将 JOIN 子句中的 ON 条件用 WHERE 子句给出的方法实现。例如上例也可改为：

图 6-6　例 6.6 查询结果

SELECT xsqkb.学号,姓名,总分 FROM xsqkb ,xscjb WHERE xsqkb.学号＝xscjb.学号

例 6.7　查询性别为"男"且 1980 年前出生的学生的:姓名,学号和各科成绩信息。

SELECT 姓名,xscjb.＊ FROM xsqkb INNER JOIN xscjb ON xsqkb.学号＝xscjb.学号 WHERE 性别＝'男' AND YEAR(出生日期)＜1980

结果如图 6－7 所示。本题也可用下面命令给出:

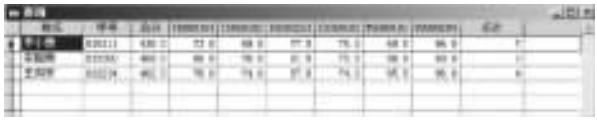


图 6－7　例 6.7 查询结果

SELECT 姓名,xscjb.＊ FROM xsqkb , xscjb;
WHERE xsqkb.学号＝xscjb.学号 AND 性别＝'男' AND YEAR(出生日期)＜1980

6.2.4　创建嵌套查询

嵌套查询指在一个 SQL－SELECT 查询的查询条件 WHERE 子句中包含由一对圆括号括起来的另一个 SQL－SELECT 查询语句,从而形成了如下的结构:

SELECT…FROM…WHERE…(SELECT…FROM… WHERE…)

这里称括号外的 SQL－SELECT 为外查询,括号内的 SQL－SELECT 为内查询,也称为子查询。根据它们的相互关系,嵌套查询被分为:普通嵌套、内外层相关嵌套查询两类。

1. 普通嵌套查询

在多表查询中常常会碰到一类情况:查询的结果出自于一个表,但相关的条件却涉及到另外的表。这样的查询称为普通嵌套查询。

例 6.8　查询总分高于 450 分的学生的姓名,性别,家庭住址。

SELECT 姓名,性别,家庭住址 FROM xsqkb WHERE 学号 IN(SELECT 学号 FROM xscjb WHERE 总分＞450)

结果如图 6－8 所示。当然,这样的查询也可用简单的连接查询来代替,例如本例也可写为:

SELECT 姓名,性别,家庭住址 FROM xsqkb, xscjb WHERE xsqkb.学号＝xscjb.学号 AND 总分＞450

图 6－8　例 6.8 嵌套查询

2. 内外层互相关嵌套查询

普通嵌套查询时内层的查询结果是外层的查询条件,而内层的查询与外层无关。然而还有另一种嵌套查询,内层的查询条件需要外层查询提供值,而外层的查询条件需要内层的查询结果。这样的查询称为内外层互相关嵌套查询。

例如有一个表:Kscjb(学号 C(8),科目 C(8),考试日期 D,成绩 I),记录如图 6－9(a)所示。

图 6－9　互相关查询

例 6.9　查询每个学生考试中成绩最高的科目的信息。

SELECT out.＊ FROM kscjb out WHERE **成绩**＝（SELECT MAX(**成绩**) FROM kscjb inn WHERE out.**学号**＝inn.**学号**)

结果如图 6－9(b)**所示。**

可以看出，在这个例子中，外层查询的输出结果中的“学号”字段的值供给了内层查询使用，而内层查询的结果——MAX(**成绩**)**则又为外层程序提供了查询条件。因此这是一个内外互相关嵌套查询。由于在这个查询中，内外层都用的是同一个表“**kscjb.dbf**”，因此为了区别，为内、外层表分别起了一个局部别名：**out、inn。

6.2.5　创建带特殊运算符、量词、谓词的查询

SQL－SELECT **中，在** WHERE **子句里，常用到一些特殊运算符，可以使问题变得非常简单，前面我们已经使用了** IN**，下面再介绍另外几个特殊运算符。**

1. BETWEEN 运算符和表达式

(1) **格式**

*Expression*1 BETWEEN *Expression*2 AND *Expression*3

(2) **功能**

当表达式 *Expression*1 **的值落在表达式** *Expression*2 **和** *Expression*3 **给出的值之间时返回一个逻辑真**.T. **，否则返回逻辑假**.F. **。其中，***Expression*1 **和** *Expression*2 **分别代表取值范围的初值和终值。**

该表达式等价于：(*Expression*1＞＝*Expression*2)AND(*Expression*1＜＝*Expression*3)**。**

例 6.10　查询总分在 450～600 **分之间的学生的姓名，性别，家庭住址。**

SELECT **姓名，性别，家庭住址，总分** FROM xsqkb，xscjb WHERE xsqkb.**学号**＝xscjb.**学号** AND **总分** BETWEEN 450 AND 600

查询结果如图 6－10 **所示。**

图 6－10　例 6.10 查询结果

例 6.11　查询总分不在 450～600 **分之间的学生的姓名，性别，家庭住址。**

SELECT **姓名，性别，家庭住址，总分** FROM xsqkb，xscjb WHERE xsqkb.**学号**＝xscjb.**学号** AND **总分** NOT BETWEEN 450 AND 600

查询结果如图 6-11 所示。

图 6-11 例 6.11 查询结果

2. IN 运算符和表达式

(1) 格式

Expression IN(*Set*)

(2) 功能

当表达式的值与 IN 中数据集合中的某个值相等时返回真.T.,否则返回假.F.。其中的 Set 是一个数据集合。它等价于:*Expression*=*Val*1 OR *Expression*=*Val*2 OR … OR *Expression*=*Valn*。

例 6.12 查询家庭住址在渭南、延安、宝鸡的学生信息。

SELECT * FROM xsqkb WHERE LEFT(ALLTRIM(家庭住址),4) IN('渭南','延安','宝鸡')

查询结果如图 6-12 所示。

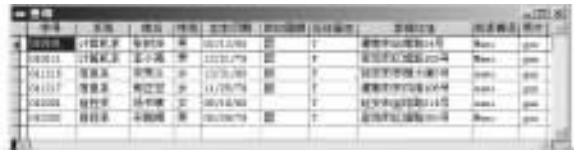


图 6-12 例 6.12 查询结果

如果取表达式的值不在数据集合中,则在 IN 前加 NOT,予以否定。

3. LIKE 运算符和表达式

(1)格式

c*Expression*1 LIKE *cExpression*2

(2)功能

当字符串表达式 1 的值与字符串表达式 2 的值相匹配时返回.T.,否则返回.F.。在字符串中可使用通配符:"%"、"_"。"%"可通配从所在位置开始的 0 个或多个长度的字符串;"_"仅可通配所在位置的一个字符。

例 6.13 查询学号的第三个字符为 2 的学生信息。

SELECT * FROM xsqkb WHERE 学号 LIKE "__2%"

查询结果如图 6-13 所示。

图 6-13 例 6.13 查询结果

注意 ① 如果要返回的相反功能,可在 LIKE 之前加 NOT。

② 在 Visual FoxPro 中，一个汉字也算一个字符，因此若通配一个汉字，则使用一个下划线。

③ 如果 LIKE 后不含通配符，则可用“=”代替 LIKE 子句，用：<>、!=、# 代替 NOT LIKE 子句。

④ WHERE 子句支持 ESCAPE 操作符，将 ESCAPE 放在“%”或“_”之前，则表示它们是一个普通的字符“%”或“_”。

例如在“xsqkb.dbf”中添加一个电话号码字段，电话号码的格式是：“XXX_XXXXXXXX”或“XXXX_XXXXXXX”，并按该格式录入学生的电话号码。

例 6.14　查询电话号码区位号为“029”的学生信息。

SELECT * FROM xsqkb WHERE 电话号码 LIKE "029_%" ESCAPE "_"

结果如图 6-14 所示。

图 6-14　例 6.14 查询结果

上面的 SQL 语句也可表示为：

SELECT * FROM xsqkb WHERE 电话号码 LIKE "029_%" ESCAPE "\"

4. EXISTS 谓词和表达式

(1) 格式

EXISTS (Subquery)

(2) 功能

检查是否至少有一行满足子查询中的条件。只要子查询不是空集，筛选的条件就为.T.。若要否定，可在其前面加 NOT。

例 6.15　查询课程号 “T8080101”的成绩小于 80 分学生的信息。

SELECT * FROM xsqkb WHERE NOT EXISTS(SELECT * FROM xscjb WHERE T8080101>=80 AND xsqkb.学号=xscjb.学号)

查询结果如图 6-15 所示。

图 6-15　例 6.15 查询结果

该句等价于：

SELECT * FROM xsqkb WHERE EXISTS(SELECT * FROM xscjb WHERE T8080101<80 AND xsqkb.学号=xscjb.学号)

5. 量词和表达式

(1) **格式**

Expression rOperator [ALL|ANY|SOME](*Subquery*)

(2) **功能**

表达式与关系运算符之后的[ALL|ANY|SOME](*Subquery*)**运算结果为**.T. **或**.F. **。其中** ANY **和** SOME **在此功能相同。**

表 6-2 **给出了** ANY(SOME)**和** ALL **谓词的功能。**

表 6-2 ANY(SOME)和 ALL 谓词的功能

表达式	功能	表达式	功能
>ANY	大于子查询结果中的某个值	<ANY	小于子查询结果中的某个值
>=ANY	大于等于子查询结果中的某个值	<=ANY	小于等于子查询结果中的某个值
>ALL	大于子查询结果中的所有值	<ALL	小于子查询结果中的所有值
>=ALL	大于等于子查询结果中的所有值	<=ALL	小于等于子查询结果中的所有值
=ANY	等于子查询结果中的某个值	=ALL	等于子查询结果中的所有值(无意义)
!=ANY	不等于子查询结果中的某个值	!=ALL	不等于子查询结果中的任何值

例 6.16 查询计算机系中成绩总分比信息系成绩总分最高的同学还要高的学生的信息。

SELECT xsqkb.*,总分 FROM xsqkb,xscjb WHERE xsqkb.学号=xscjb.学号 AND (总分>ALL (SELECT xscjb.总分 FROM xscjb WHERE LEFT(学号,3)='011')) AND left(xsqkb.学号,3)='010'

结果如图 6-16 **所示。**

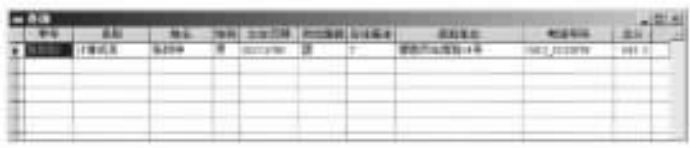


图 6-16 例 6.16 查询结果

6.2.6 创建带有计算的查询

在 Visual FoxPro 9.0 **的** SQL-SELECT **中，提供了五个集函数，使用它们可以使需要进行某种统计输出的功能实现起来变得方便快捷，它们是：**

① **计数函数：**COUNT()，**用来统计记录条数，类似于** Visual FoxPro **的** COUNT **命令。**

② **求和函数：**SUM()，**用于进行数值型字段的求和，类似于** Visual FoxPro **的** SUM **命令。**

③ **求平均函数：**AVG()，**用于进行数值型字段的求平均，类似于** Visual FoxPro **的** AV-

ERAGE 命令。

④ 求最大值函数：MAX()，用于选字段的最大值，类似于 Visual FoxPro 的 MAX()函数。

⑤ 求最小值函数：MIN()，用于选字段的最小值，类似于 Visual FoxPro 的 MIN()函数。

这五个函数除 COUNT()外，其他均仅用于数值表达式。同时，SQL－SELECT 还允许用户自定义一个表达式作为查询的输出项。

例 6.17　查询信息系的学生总人数。

SELECT COUNT(*) AS 信息系人数 FROM xsqkb WHERE LEFT(学号,3)='012'

查询结果如图 6－17 所示。

图 6－17　例 6.17 查询结果

6.2.7　创建分组与计算的查询

把分组子句 GROUP BY 与集函数或表达式相结合，会使查询的功能大幅度增强。分组子句的格式为：

[GROUP BY *Column_List_Item* [, ...]]

其中：

• *Column_List_Item* [, ...]：用来指定分组的列，它可以是一列或多列。其形式为下面的任一种：

① FROM 子句中给出的表的一个字段或一个子查询；

② SQL SELECT 列表中一个表的别名；

③ 指定查询结果表列位置的数值表达式。最左边的列是 1 号。

例 6.18　查询各系的学生总人数。

SELECT 系别,COUNT(*) AS 学生数 FROM xsqkb GROUP BY 系别

查询结果如图 6－18 所示。

图 6－18　例 6.18 查询结果

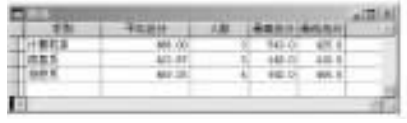


图 6－19　例 6.19 查询结果

例 6.19　查询各系的平均总分、学生人数、最高总分和最低总分。

SELECT 系别, AVG(总分) AS 平均总分,COUNT(xscjb.学号) AS 人数,MAX(总分) AS 最高总分, MIN(总分) AS 最低总分 FROM xscjb,xsqkb GROUP BY 系别 WHERE xsqkb.学号=xscjb.学号

查询结果如图 6－19 所示。

6.2.8　创建排序查询

通过 ORDER BY 子句可以实现查询结果的排序输出。该子句的格式为：

[ORDER BY *Order_Item* [ASC | DESC] [, ...]]

其中：

① *Order_Item*：指定将最终查询结果进行排序所用的项。它可以是下面的形式：

- FROM 子句中的表的一个字段或一个子查询。但不能是 Blob 或 General 型字段；
- 一个 SELECT 列表中的字段别名；
- 一个标识查询结果中列位置的数值表达式。最左边的列为第 1 列；
- 如 ORDER BY 子句出现在 UNION 子句中，则可以是最后一个 SELECT 主句而不是子查询的一个选择项。

② ASC：指定查询结果按升序排列。这是缺省次序。

③ DESC：指定查询结果按降序排列。

例 6.20 按总分从高到低的次序输出学生的信息。

SELECT xsqkb.*，总分 FROM xscjb，xsqkb WHERE xsqkb.学号＝xscjb.学号 ORDER BY 总分 DESC

查询结果如图 6－20 所示。

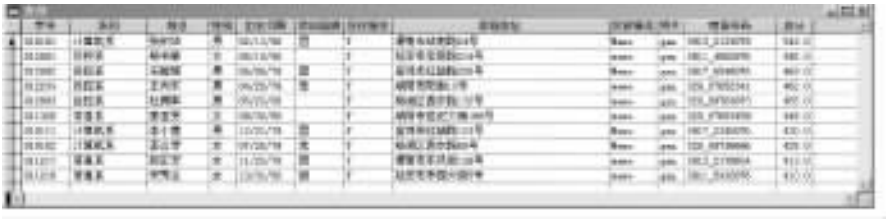


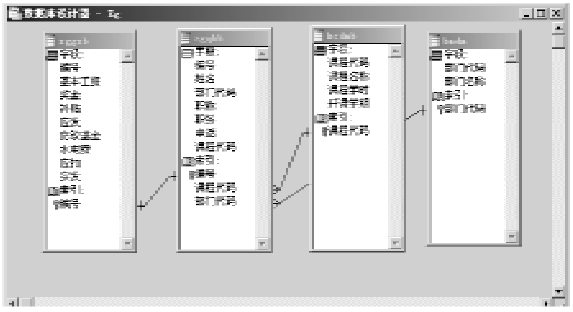
图 6－20 按总分从高到低排序查询结果

6.2.9 创建外连接和自连接查询

1. 外连接查询

在 SQL 中，它不仅有基于自然连接的内查询，还有另一种查询——外连接查询。外连接查询首先保证两个表的所有满足条件的元组都出现在查询结果中，然后再将不满足连接条件的元组进行连接，不满足连接条件的则应将来自另一个表的属性值置为空值(.Null.)。这样的连接分为左连接、右连接、完全连接。

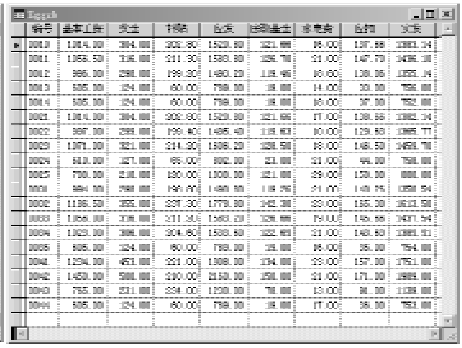
设有一职工数据库“zg.dbc”，其中包含五个表，如图 6－21 所示。下面以该数据库为例，完成各种连接。



(a) zg 数据库

(b) zg! kcdmb.dbf

(c) zg! zgqkb.dbf



(d) zg! zggzb.dbf

(e) zg! bmdm.dbf

图 6－21　zg 数据库

例 6.21　左连接。输出所有职工及所任课程情况。

SELECT a. 编号,a. 姓名,a. 职称,a. 职务,b. 课程名称,b. 课程学时;

FROM zgqkb a LEFT JOIN kcdmb b ON a. 课程代码＝b. 课程代码

查询结果如图 6－22 所示。右连接和全连接由读者自己完成。

图 6－22　例 6.21 左连接查询结果

2. 自连接查询

自连接查询指将同一关系与其自身进行的连接。在可以进行自连接查询的关系中,实际上存在着一种特殊的递归关系,即表中的某些元组,根据出自同一值域的两个不同的属性,可以与另外一些元组有一对多的关系。

下面用一个关系来说明自连接。该关系“zgqkb1.dbf”如图 6－23 所示。

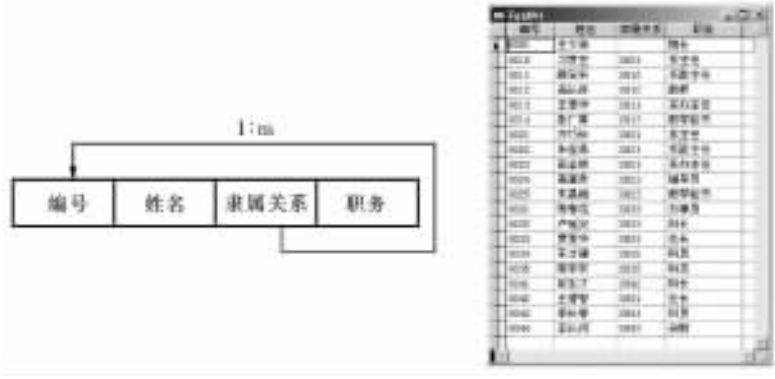


图 6－23　zgqkb1.dbf 中编号和隶属关系的一对多关系

例 6.22 自连接。根据“zgqkb1.dbf”中的隶属关系列出各级领导所领导的职工。

SELECT z.姓名 ,z.职务,"领导",f.姓名 ,f.职务 FROM zgqkb1 z,zgqkb1 f WHERE z.编号=f.隶属关系 ORDER BY z.编号

查询结果如图 6-24 所示。

图 6-24 自连接查询结果

6.2.10 集合的并运算——UNION

SQL 支持集合的并运算,对于两个具有相同字段个数、对应字段的值出自于同一值域的查询,将把第二个查询结果和第一个查询合并成一个查询结果。格式如下:

SQL_Select Query1 UNION *SQL_Select Query2*

例 6.23 在“xsqkb1.dbf”中查询男学生、女学生的信息。

SELECT * FROM xsqkb1 WHERE 性别='男'
UNION SELECT * FROM xsqkb1 WHERE 性别='女'

查询结果如图 6-25 所示。

图 6-25 查询的并运算结果

6.2.11 查询结果的输出

在前面几小节中,所有的查询都没有涉及希望输出几条查询结果,也没有涉及查询结果的去向问题。本节将学习这方面的知识。

1. 指定输出部分查询结果

对于指定输出部分查询结果，通过在 SELECT 子句中加入 TOP 子句的方法实现，它要和 ORDER BY 子句配合使用。TOP 子句的格式为：

TOP *nExpr* [PERCENT]

其作用是指定查询结果所要显示的记录条数，或者要显示的记录条数占查询结果的百分比。当指定要显示的记录条数时，*nExpr* 的取值范围是：1～32 767。而如果在子句中使用了 PERCENT 选择，则 *nExpr* 表示要显示记录所占的百分数，此时取值范围为了 0.01～99.99。Visual FoxPro 首先将记录进行排序，然后按该子句所指定值，显示最前面的 *nExpr* 条或百分之 *nExpr* 条记录。PECRENT 获得的要显示的记录条数是对查询结果总记录数按百分比计算后所得的最高整数。

例 6.24　查询实发工资最高的前五位职工的信息。

SELECT 姓名，zggzb.* TOP 5 FROM zgqkb，zggzb

ORDER BY 实发 DESC WHERE zgqkb.编号＝zggzb.编号

查询结果如图 6－26(a)所示。

(a) 部分查询结果输出一　　(b) 部分查询结果输出二

图 6－26　部分查询结果输出

例 6.25　查询实发工资最低的最前 15%的职工的信息。

SELECT 姓名，zggzb.* TOP 15 PERCENT FROM zgqkb，zggzb ORDER BY 实发 WHERE zgqkb.编号＝zggzb.编号

查询结果如图 6－26(b)所示。

2. 指定查询输出去向

查询结果的去向是通过 INTO 或 TO 子句实现的。

(1)INTO 子句

格式：INTO *StorageDestination*

指定将查询结果保存在一个数组、临时表或永久表的表中。如果没有 INTO 子句，查询结果仅在缺省的浏览窗口被显示。INTO 子句中的 *StorageDestination* 可以用下面的方式给出：

① ARRAY *ArrayName*：将结果保存在一个内存变量数组中。

② CURSOR *CursorName* [NOFILTER | READWRITE]：将查询结果保存在一个临时表中。如果用户指定了一个已经打开的表的名字，Visual FoxPro 将生成一个出错信息。SELECT 语句执行后，临时表保持打开、激活及只读状态，除非用户使用了 READWRITE 选择。当关闭临时表时，它将自动删除。临时表可以按临时文件的形式存在于由 SORTWORK 关键字指定的驱动器或卷标上。NOFILTER：生成一个可用在子查询中的一个临时表。

③ TABLE *TableName* [DATABASE *DatabaseName* [NAME Long*TableName*]：将查询结果存储在一个表内。SELECT 语句执行后，该表处于打开和激活状态。

(2)TO 子句

格式：TO *DisplayDestination*

将查询结果发送到一个文件、打印机、Visual FoxPro 主窗口或一个活动的用户自定义窗口。注意，在同一个查询中如果同时包含了 INTO 子句，则 TO 子句会被忽略。在 TO 子句中的 *DisplayDestination* 可以是下面的三个子句之一。

① FILE *FileName* [ADDITIVE]：指定将查询结果直接写入一个文本文件中。

② PRINTER [PROMPT]：直接将查询结果输出到打印机。

③ SCREEN：直接将结果输出到 Visual FoxPro 主窗口或一个活动的用户自定义的窗口。

例 6.26 查询实发工资最低的最前 15%的职工的信息，并将查询结果写入一个二维数组：zdgz 中，然后显示之。

```
CLEAR MEMORY
SELECT 姓名,zggzb.* TOP 15 PERCENT FROM zgqkb,zggzb ;
ORDER BY 实发 WHERE zgqkb.编号=zggzb.编号 ;
INTO ARRAY zdgz
DISPLAY MEMORY LIKE zdgz
```

结果如图 6-27。

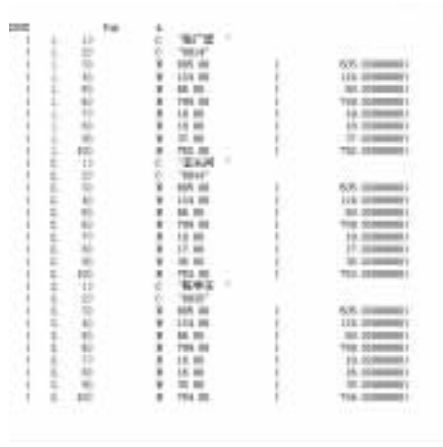


图 6-27 将查询结果输出到二维数组

例 6.27 查询实发工资最低的最前 15%的职工的信息，并将查询结果写入一个临时表：lsb.dbf 中，然后浏览该表。

```
SELECT 姓名,zggzb.* TOP 15 PERCENT FROM zgqkb,zggzb;
    ORDER BY 实发 WHERE zgqkb.编号=zggzb.编号;
    INTO CURSOR lsb
BROWSE
```

例 6.28 将职工的工资从高到低排序查询，将结果保存到一个文本文件 gzpx.txt 中，要求在查询时窗口不显示查询结果，查询结束后用 TYPE 命令显示生成的文本文件内容。

```
SELECT 姓名,zggzb.* FROM zgqkb,zggzb ORDER BY 实发 DESC ;
WHERE zgqkb.编号=zggzb.编号 TO FILE gzpx NOCONSOLE
TYPE gzpx.txt
```

6.3　SQL 的数据操纵功能之二

SQL 的数据操纵功能之二指除 SQL－SELECT 外的三个操纵功能，即数据的插入、更新和删除功能。

6.3.1　SQL 的数据插入功能

在 SQL 中元组的插入可利用 INSERT 命令实现。关系型数据库的特点之一是属性不论左右，记录不分前后，因此这里的 INSERT SQL 命令和 Visual FoxPro 中的 APPEND 命令有相似之处，该命令将在指定的表的末尾追加一条新记录。INSERT SQL 有三种语法格式。

1. 格式 1

```
INSERT INTO dbf_name [(FieldName1 [, FieldName2, ...])]
    VALUES (eExpression1 [, eExpression2, ...])
```

该命令在表的交互式操作一章中已作过介绍。

2. 格式 2

(1) 命令格式

```
INSERT INTO dbf_name FROM Source
```

(2) 功能

从数组元素、内存变量组，或者与表中字段名相匹配的对象的属性插入内容。该命令类似于 Visual FoxPro 中的：APPEND FROM *dbf_name*| ARRAY *ArrayName*

(3) 参数和子句说明

FROM *Source*：指定要插入的数据的来源。*Source* 可以是下面给出的有效项：

① ARRAY *ArrayName*：数据来自于数组，*ArrayName* 为数组名；

② MEMVAR：来自内存变量组中与字段同名的内存变量；

③ NAME *ObjectName*：指定一个有效的 Visual FoxPro 对象。

例 6.29　首先将 gzb.dbf 清空，然后将 zggzb.dbf 的记录复制到数组 arr 中，再利用 INSERT SQL 命令将数组的值追加成 gzb.dbf 的记录，最后浏览之。

```
CLOSE ALL
USE gzb
ZAP
USE zggzb IN 0
SELECT zggzb
COPY TO ARRAY arr
INSERT INTO gzb FROM ARRAY arr
SELECT gzb
```

BROWSE

显示结果表明,得到了一个和 zggzb.dbf 记录完全相同的表 gzb.dbf。

3. 格式 3

(1)命令格式

INSERT INTO *dbf_name* [(*FieldName1* [, *FieldName2*, ...])]

SELECT *SelectClauses* [UNION *UnionClause* SELECT *SelectClauses* ...]

(2)功能

将由 SQL SELECT 命令生成的结果插入表的指定字段中。

例 6.30 首先将 gzb.dbf 清空,然后再利用 INSERT SQL 命令将 zggzb.dbf 的记录中实发工资最高的前五条记录插入 gzb.dbf 中,最后浏览之。

```
CLOSE ALL
USE gzb
ZAP
INSERT INTO gzb SELECT * TOP 5 FROM zggzb ORDER BY 实发 DESC
BROWSE
```

执行结果如图 6-28 所示。

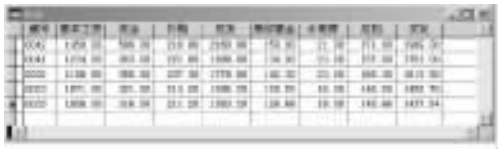


图 6-28 例 6.30 执行结果

6.3.2 SQL 的数据更新功能

SQL 中的更新命令是 UPDATE 命令,它既可用于一个表自身的更新,又可用于通过一个表来更新另一个表。

(1)命令格式

UPDATE Target

SET *Column_Name1* = *eExpression1* [, *Column_Name2* = *eExpression2* ...]

[FROM [FORCE] *Table_List_Item* [[, ...] | [JOIN [*Table_List_Item*]]]

WHERE *FilterCondition1* [AND | OR *FilterCondition2* ...]

(2)功能

用来更新一个单表的记录。

(3)参数和子句说明

· Target 指定一个要更新记录的永久表名、临时表名或它们的别名,或要被更新的文件名。

SET *Column_Name1* = *eExpression1* [,*Column_Name2* = *eExpression2* ...]:指定要更新的表的列和它们的值。

· 其他子句与 SELECT SQL 命令中的意义相同。当 WHERE 子句缺省时,指将全部记

录用同一个表达式的值来更新。

例 6.31　利用 UPDATE SQL 命令更新xszhjfb.dbf中的“素质总分”字段，使得每位学生的素质总分：素质总分 ＝精神文明 * 0.5＋社会活动 * 0.2＋体育锻炼 * 0.3，综合积分 ＝ xscjb.总分/6 * 0.8＋素质总分 * 0.2

```
UPDATE xszhjfb SET 素质总分 ＝精神文明 * 0.5＋社会活动 * 0.2＋体育锻炼 * 0.3
UPDATE xszhjfb SET 综合积分 ＝素质总分 * 0.2＋；
(SELECT xscjb.总分/6 * 0.8 FROM xscjb WHERE xszhjfb.学号＝xscjb.学号)
SELECT xszhjfb
BROWSE
```

执行结果如图 6－29 所示。

学号	精神文明	社会活动	体育锻炼	素质总分	素质排名	综合积分	综合排名
010101	89.0	90.0	76.0	85.3	0	89.5	0
010102	90.0	93.0	88.0	89.6	0	74.6	0
010111	87.0	79.0	95.0	87.9	0	74.9	0
011216	75.0	85.0	79.0	78.7	0	70.4	0
011217	82.0	88.0	91.0	90.9	0	73.2	0
011330	96.0	85.0	79.0	88.7	0	77.5	0
012001	87.0	93.0	85.0	87.2	0	83.0	0
012002	85.0	79.0	84.0	83.5	0	79.1	0
012003	90.0	87.0	90.0	89.4	0	79.5	0
012234	96.0	88.0	92.0	93.2	0	80.2	0

图 6－29　例 6.31 执行结果

6.3.3　SQL 的数据删除功能

SQL 数据删除功能指从表中逻辑删除满足条件的记录，类似于 Visual FoxPro 中的 DELETE 命令。

1. 命令格式

```
DELETE [Target]
    FROM [FORCE] Table_List [[, Table_List ...] | [JOIN [ Table_List]]]
    [WHERE FilterCondition1 [AND | OR FilterCondition2 ...]]
```

2. 功能

逻辑删除记录。

3. 参数和子句说明

• *Target*：指定一个要进行删除操作的目标表名、临时表名，或它们的别名，或文件名。如果 FROM 子句中列出了多个表，则必须给出此参数。

• FROM [FORCE] *Table_List* [[, *Table_List* ...] | [JOIN [*Table_List*]]]：指定要进行删除操作的一个或多个表的表名。语法要求与 SQL UPDATE 相同。

• WHERE 子句的语法要求与 SQL UPDATE 相同。缺省时删除全部记录。

例 6.32　利用 DELETE SQL 命令将 zg.dbc 中 zgqkb.dbf 表中编号的第 3 位为“4”的职工进行逻辑删除。

```
OPEN DATABASE zg
DELETE FROM zgqkb WHERE SUBSTR(编号,3,1)="4"
USE zgqkb
BROWSE
```

执行结果如图 6-30 **所示。**

图 6-30 **例** 6.32 **执行结果**

6.4 SQL 的数据定义功能

SQL **的数据定义功能包括表结构的创建、结构修改、表的删除三种操作。**

6.4.1 利用 SQL 定义表

在关系型数据库中,数据模式是通过:关系名(属性 1,**属性** 2,…,**属性** *n*)**的格式来描述的。关系模式就是表的结构,所谓定义表是指创建表的结构。**

对表而言,一个字段除过字段的名字、类型、宽度、精确度这四个最基本的特性外,还可以有辅助特性,诸如:是否允许有空值、是否设置字段完整性规则、是否为主关键字或候选关键字,是否设置为外部关键字,是否建立与其他表的永久关系等。 SQL **的表的定义命令应包括这些方面的内容。**

1. 命令格式

```
CREATE TABLE | DBF TableName1 [NAME LongTableName] [FREE]
    [CODEPAGE = nCodePage]
    ( FieldName1 FieldType [( nFieldWidth [, nPrecision] )] [NULL | NOT NULL]
    [CHECK lExpression1 [ERROR cMessageText1]]
    [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]]
    [DEFAULT eExpression1]
    [PRIMARY KEY | UNIQUE [COLLATE cCollateSequence]]
```

[REFERENCES *TableName*2 [TAG *TagName*1]] [NOCPTRANS]
[, *FieldName*2 ...]
[, PRIMARY KEY *eExpression*2 TAG *TagName*2
| , UNIQUE *eExpression*3 TAG *TagName*3 [COLLATE *cCollateSequence*]]
[, FOREIGN KEY *eExpression*4 TAG *TagName*4 [NODUP]
[COLLATE *cCollateSequence*] REFERENCES *TableName*3 [TAG *TagName*5]]
[, CHECK *lExpression*2 [ERROR *cMessageText*2]])
| FROM ARRAY *ArrayName*

2. 功能

使用指定的字段或由一个数组生成一个表。

3. 参数和子句说明

下面分小节进行参数和子句功能分析说明。

(1) CREATE TABLE | DBF *TableName*1 [NAME Long*TableName*] [FREE] [CODEPAGE = *nCodePage*]

· CREATE TABLE|DBF:SQL **的表创建命令。这里关键字** TABLE **与** DBF **等效。其他参数与过去讲过的意义相同。**

(2)*FieldName*1 *FieldType* [(*nFieldWidth* [, *nPrecision*])]

分别用来指定表第一个字段的字段名、字段类型、字段宽度,精度。

(3)NULL | NOT NULL

指明第一个字段是否允许有空值。

(4)CHECK *lExpression*1 [ERROR *cMessageText*1]

为第一个字段设置域完整性检查。其中:

· *lExpression*1:**指定字段的有效性规则。**

· ERROR *cMessageText*1:**字段完整性检查出错时要给出的出错信息。**

(5)AUTOINC [NEXTVALUE *NextValue* [STEP *StepValue*]] [DEFAULT *eExpression*1]

指定第一个字段为自动增量字段。其子句的作用为:

· NEXTVALUE *NextValue*:**指定字段的初始值,它的取值范围在** −2 147 483 647～2 147 483 647**之间,可正可负。缺省值是** 1。

· STEP *StepValue*:**指定步长值,可正可负,缺省为** 1。

· DEFAULT **子句:为字段设置一个缺省值。如果使用了** AUTOINC **子句来为字段打开自动增量并设置缺省值,**Visual FoxPro **会在表中保存缺省值但却并不使用它。如果使用** SQL ALTER TABLE **命令来改变字段的自动增量,**Visual FoxPro **会使用缺省值。**

(6)PRIMARY KEY | UNIQUE [COLLATE *cCollateSequence*]

指定字段的索引类型和比较方式。

· PRIMARY KEY | UNIQUE:**以该字段为关键字生成主索引或候选索引。索引标记与字段同名。**

· COLLATE *cCollateSequence*:**指定字符的比较次序。**

(8)REFERENCES *TableName2* [TAG *TagName1*] [NOCPTRANS]

该子句指定与父表创建一个永久关系。

· *TableName2*:父表名。它一定是一个数据库表。

· TAG *TagName1*:指定在父表 *TableName2* 中的索引标记名。如果缺省 TAG 子句,则使用父表的主关键字建立永久关系。如果主表没有主索引,Visual FoxPro 将给出一个错误。

· NOCPTRANS:防止对于字符型(C)、备注型(M)和任意字符型(Varchar)型字段被转换成不同的代码页。可以仅对字符型或备注型字段设置 NOCPTRANS。这样在表设计器中,会出现:字符(二进制)、备注(二进制)、Varchar(二进制)的数据类型。

(8)*FieldName2* ...

指定一个或多个其他字段及其属性。

(9)PRIMARY KEY *eExpression2* TAG *TagName2* |, UNIQUE *eExpression3* TAG *TagName3* [COLLATE *cCollateSequence*]

指定表中的任意一个字段或几个字段的组合用来生成主索引或候选索引。各子句作用如下:

· PRIMARY KEY *eExpression2* TAG *TagName2*:创建主索引,标记为 *TagName2*。如果前面已经用某个字段生成了表的主索引,则再不能使用本 PRIMARY KEY 子句。

· UNIQUE *eExpression3* TAG *TagName3*:创建候选索引,标记名为 *TagName3*。

(10) FOREIGN KEY *eExpression4* TAG *TagName4* [NODUP] [COLLATE *cCollateSequence*] REFERENCES *TableName3* [TAG *TagName5*]

生成一个外部索引,并建立和父表的永久关系。各子句作用如下:

· FOREIGN KEY *eExpression4* TAG *TagName4*:用来指定建立外部索引关键表达式和索引的标记名。

· [NODUP]:生成一个候选的外部索引。

· REFERENCES *TableName3* [TAG *TagName5*]:指定要建立永久关系的父表。参数 *TagName5* 指定父表 *TableName3* 中用来创建永久关系的索引标记名。如果缺省了 TAG 子句,将使用父表中主索引关键字来创建关系。

(11)CHECK *lExpression2* [ERROR *cMessageText2*]

指定表的有效性规则。子句作用如下:

· CHECK *lExpression2*:设置表的完整性规则。I*Expression2* 是一个关系表达式、用户自定义函数或一个存储过程。

· ERROR *cMessageText2*:设定表有效性规则的出错信息。

至此,对于一个表的定义已经完成。下面是用数组来定义一个表的子句。

(12)FROM ARRAY *ArrayName*

指定一个用来生成表的数组。该数组中应包含创建表所需要的各字段的名字、类型、长度、精度。用户可以用 FROM ARRAY 子句代替定义 CREATE TABLE 命令中的个别字段。

下面以一个实例来说明利用 SQL 创建数据库表的过程。

例 6.33 设有一个订货管理数据库 dhgl.dbc ,包括四个表文件:

ck(仓库号 C(4),城市 C(4),面积 I)

zg(仓库号 C(4),职工号 C(4),工资 I)

dgd(职工号 C(4),供应商号 C(4),订购单号 C(4),订购日期 D,总金额 N(10))

gys(供应商号 C(4),供应商名 C(16),地址 C(10))

数据模式中带有下划线的字段为主码。请用 SQL CREATE 命令来创建它们。

S1：创建数据库：dhgl.dbc。

CREATE DATABASE dhgl

S2：创建表：ck.dbf。以“仓库号”字段为主索引，为字段“面积”设置域完整性规则。

CREATE TABLE ck(仓库号 c(4) PRIMARY KEY，城市 C(4) ；

面积 I CHECK 面积>0 ERROR “仓库面积应大于 0!”)

S3：创建表：zg.dbf。以“职工号”字段为主关键字。为“工资”字段设置域完整性规则。以“仓库号”字段为外部关键字创建普通索引，并以表 ck.dbf 为父表创建一对多的永久关系。

CREATE TABLE zg(仓库号 C(4) ，职工号 C(4) PRIMARY KEY ；

工资 I CHECK 工资>500 ERROR “职工工资最低应不低于 500 元!” ；

DEFAULT 500 ；

FOREIGN KEY 仓库号 TAG 仓库号 REFERENCE ck)

S4：创建表：gys.dbf。以“供应商号”为主关键字。

CREATE TABLE gys(供应商号 C(4) PRIMARY KEY；

供应商名 C(16)，地址 C(10))

S5：创建表：dgd.dbf。以“订购单号”为主关键字，以“职工号”字段为外部关键字建立普通索引，并以 zg.dbf 表为父表建立永久一对多关系；以“供应商号”字段为外部关键字建立普通索引，并以 gys.dbf 表为父表建立永久一对多关系。

CREATE TABLE dgd(职工号 C(4)，供应商号 C(4)；

订购单号 C(4) PRIMARY KEY，订购日期 D，总金额 N(10)；

FOREIGN KEY 职工号 TAG 职工号 REFERENCE zg；

FOREIGN KEY 供应商号 TAG 供应商号 REFERENCE gys)

S6：浏览数据库，结果如图 6－31 所示。

MODIFI DATABASE

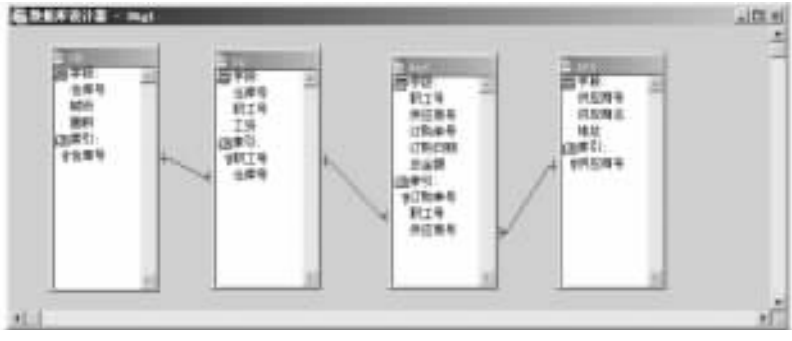


图 6－31　数据库 dhgl.dbc 中的表及其相互关系

6.4.2 利用 SQL 删除表

1. 命令格式

DROP TABLE *TableName* | *FileName* | ? [RECYCLE]

2. 功能

从当前的数据库中移去表并从磁盘上将它删除。

3. 参数和子句说明

参数均与过去各命令中的意义相同。

注意 当执行 DROP TABLE 命令后,与表相联系的所有的主索引、缺省值、有效性规则也被删除。如果该表与数据库中的其他表已经建立了关系,则应先解除关系,方可移去或删除。

例 6.34 利用 SQL DROP 命令将 ck.dbf 从数据库 dhgl.dbc 中移去,放入 Windows 回收站。

```
DROP TABLE ck RECYCLE
```

6.4.3 利用 SQL 修改表结构

在第 4 章表的交互式操作中,我们曾经对利用 SQL ALTER TABLE 命令修改表结构作过简单的介绍,本节将给出它的详细叙述。

SQL ALTER TABLE 有三种格式。

1. 格式 1

(1)命令格式

```
ALTER TABLE TableName1 ADD | ALTER [COLUMN] FieldName1
    FieldType [( nFieldWidth [, nPrecision])] [NULL | NOT NULL]
    [CHECK lExpression1 [ERROR cMessageText1]]
    [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]]
    [DEFAULT eExpression1]
    [PRIMARY KEY | UNIQUE [COLLATE cCollateSequence]]
    [REFERENCES TableName2 [TAG TagName1]]
    [NOCPTRANS] [NOVALIDATE]
```

(2)功能

向表中添加一个新字段(ADD)或修改(ALTER)一个已有的字段。

(3)参数和子句说明

- *TableName*1:指定要修改结构的表的名字。
- ADD:添加字段,ALTER:修改字段。
- 其他参数和子句与 CREATE TABLE 中相同。

例 6.35 利用 SQL ALTER 命令为 xjsjk.dbc 数据库中的xsqkb.dbf表添加一个新字段"身高 N(4,2)"。

```
ALTER TABLE xsqkb ADD 身高 N(4,2) CHECK 身高>1;
```

ERROR "身高至少在 1m 以上" DEFAULT 1.5

2. 格式 2

(1)**命令格式**

ALTER TABLE *TableName1* ALTER [COLUMN] *FieldName2* [NULL | NOT NULL]

[SET DEFAULT *eExpression2*]

[SET CHECK *lExpression2* [ERROR *cMessageText2*]]

[DROP DEFAULT] [DROP CHECK] [NOVALIDATE]

(2)**功能**

对表中已有的字段设置、修改或删除缺省值、有效性规则。

(3)**参数和子句说明**

· SET **子句:用来设置缺省值、有效性规则。**

· DROP **子句:用来删除缺省值、有效性规则。**

· NOVALIDATE:**指定** Visual FoxPro **在违反数据完整性的情况下改变表结构。如果缺省,**Visual FoxPro **禁止进行这样的表结构的改变。**

· **其他参数与前相同。**

例 6.36　利用 SQL ALTER **命令将** xjsjk.dbc **数据库中的**xsqkb.dbf**表新添加的字段"身高"的缺省值改为** 1m。

ALTER TABLE xsqkb ALTER **身高** SET DEFAULT 1;

SET CHECK **身高**>=1 ERROR "**身高至少在** 1m **以上(含** 1m)"

3. 格式 3

(1)**命令格式**

ALTER TABLE *TableName1* [DROP [COLUMN] *FieldName3*]

[SET CHECK *lExpression3* [ERROR*cMessageText3*]]

[DROP CHECK]

[ADD PRIMARY KEY *eExpression3* [FOR *lExpression4*] TAG *TagName2*

[COLLATE *cCollateSequence*]]

[DROP PRIMARY KEY]

[ADD UNIQUE *eExpression4* [[FOR *lExpression5*] TAG *TagName3*

[COLLATE *cCollateSequence*]]]

[DROP UNIQUE TAG *TagName4*]

[ADD FOREIGN KEY [*eExpression5*] [FOR *lExpression6*] TAG *TagName4*

REFERENCES *TableName4* [TAG *TagName4*][COLLATE *cCollateSequence*]

REFERENCES *TableName2* [TAG *TagName5*]]

[DROP FOREIGN KEY TAG *TagName6* [SAVE]]

[RENAME COLUMN *FieldName4* TO *FieldName5*] [NOVALIDATE]

(2)**功能**

删除字段,更改字段名,定义、修改、删除表一级的有效性规则。它实际上是对格式 1 **和格**

式 2 的功能补充。

(3) 参数和子句说明

· RENAME COLUMN *FieldName*4 TO *FieldName*5:为字段更名。

· ADD 子句:添加。

· DROP 子句:删除。ADD 和 DROP 都是对表一级而言的。

· SAVE:用于 DROP FOREIGN KEY TAG *TagName*6 子句中,表示在删除索引标记为 *TagName*6 的外部关键字时,用户可以使用 SAVE 在结构复合索引中继续保持这个索引标记。如果缺省 SAVE,则该标记将被从结构复合索引中删除。

· 其他参数和子句意义同前。

例 6.37 利用 SQL ALTER 命令将 xjsjk.dbc 数据库中的xsqkb.dbf表新添加的字段"身高"改为"身长"。

ALTER TABLE xsqkb RENAME COLUMN 身高 TO 身长

例 6.38 利用 SQL ALTER 命令将 xjsjk.dbc 数据库中的xsqkb.dbf表字段 "体重"删除。

ALTER TABLE xsqkb DROP 体重

例 6.39 利用 SQL ALTER 命令对 xjsjk.dbc 数据库中的xsqkb.dbf表以字段 "学号"和"出生日期"创建一个候选索引,索引名:xhsr 。

ALTER TABLE xsqkb ADD UNIQUE 学号+DTOC(出生日期) TAG xhsr

例 6.40 利用 SQL ALTER 命令将 xjsjk.dbc 数据库中的xsqkb.dbf表中的候选索引:xhsr 删除。

ALTER TABLE xsqkb DROP UNIQUE TAG xhsr

6.5 思考与练习

一、选择题

1. SQL 的核心是(　　)。

A) 数据查询　　B)数据修改　　C) 数据定义　　D)数据控制

2. SQL 语句中条件短语的关键字是(　　)。

A)WHERE　　B) FOR　　C)WHILE　　D) CONDITION

3. 从"xsqkb.dbf"表中查询所有的姓名,应输入命令(　　)。

A)SELECT xsqkb.dbf FROM 姓名

B)SELECT 姓名 FROM xsqkb

C) SELECT 姓名

D)SELECT xsqkb.dbf WHERE 姓名

4. 使用 SQL SELECT 可以将查询结果排序,排序的短语是(　　)。

A) ORDER BY　　B) ORDER　　C)GROUP BY　　D)COUNT

5. 嵌套查询命令中的 IN 相当于(　　)。

A)等号=　　B)集合运算符∈　　C) 加号+　　D) 减号-

6. 在"考生成绩"表中查找出成绩在 90～95 分的考生信息,应输入命令(　　)。

A)SELECT * FROM 考生成绩 WHERE 成绩 BETWEEN 90 AND 95

B) SELECT 信息 FROM 考生成绩 WHERE 成绩 BETWEEN 90 AND 95

C) SELECT * FROM 考生成绩 WHERE 成绩 BETWEEN 90～95

D)SELECT 成绩 WHERE 成绩 BETWEEN 90～95 FROM 考生成绩

7. SQL SELECT 语句中的 GROUP BY 和 HAVING 短语对应查询设计器上的选项卡是(　　)。

A) 字段　　B) 连接　　C)分组依据　　D) 排序依据

8. SQL 的数据操作语句不包括(　　)。

A) INSERT　　B)UPDATE　　C)DELETE　　D)CHANGE

9. SQL 的语句中修改表结构的命令是(　　)。

A)MODIFY TABLE　　B)MODIFY STRUCTURE

C) ALTER TABLE　　D) ALTER STRUCTURE

10. SQL 语句中删除表的命令是(　　)。

A)DROP TABLE　　B)DELETE TABLE

C)ERASE TABLE　　D)DELETE DBF

11. 向表中插入数据的 SQL 命令是(　　)。

A)INSERT　　B) INSERT INTO

C) INSERT IN　　D) INSERT BEFORE

12. 使用 SQL 语句进行分组检索时,为了去掉不满足条件的分组,应当(　　)。

A) 使用 WHERE 子句

B) 在 GROUP BY 后面使用 HAVING 子句

C) 先使用 WHERE 子句,再使用 HAVING 子句

D) 先使用 HAVING 子句,再使用 WHERE 子句

13. 下列选项中,不属于数据定义功能的 SQL 语句是(　　)。

A) CREATE　　B)ALTER　　C)SELECT　　D) DROP

二、填空题

1. VFP 9.0 中的 SQL SELECT 语句中为了将查询结果存放到临时表中应该使用_____短语。

2. VFP 9.0 中的 SQL SELECT 语句支持集合的并运算,运算符是_____。

3. 在 VFP 9.0 中的 SQL SELECT 中用于计算检索的集函数有 COUNT、_____、_____、MAX 和 MIN。

4. 在 VFP 9.0 中的 SQL 中,用于进行数据定义的语言 DDL 负责:_____、_____、_____表、索引和视图等对象,各自的命令动词为:_____、_____、_____。

5. 在 VFP 9.0 中的 SQL 中,用于进行数据操纵的语言 DML 负责数据库中数据的_____、_____、_____、_____,各自的命令动词为_____、_____、_____、_____。

三、上机题

1. 利用第 4 章创建三个表:xs.dbf、kc.dbf、cj.dbf,用 SQL 的命令完成以下查询:

(1)把 xs.DBF 显示出来。

(2)查询所有男同学的姓名及其高等数学的成绩。

(3)查询高等数学高于 75 分低于 90 分,且英语成绩及格的所有学生的信息。

(4)将学生信息按男女分组输出。

(5)将学生信息根据各门功课的成绩和排序,从高到低输出。

2. 用 SQL 语句建立一个与xs.dbf的结构类似的表 new.dbf,并用 SQL 语句对表new.dbf完成下列更新操作。

(1)向表中添加身高和照片字段。

(2)插入与xs.dbf的前两个记录相类似的记录,但不包含身高和照片字段。

(3)插入一个姓名为“张三”的记录,记录的其他内容自定。

(4)把姓名为“张三”的记录的身高更新为“1.82”。

(5)删除照片字段。

(7)逻辑删除性别为“男”的全部记录。

(8)删除表 new.dbf。

第 7 章　查询与视图

在 SQL 中，我们已详细介绍了它强大的查询功能 SQL SELECT 命令和强大功能的数据定义命令 CREATE 等。其实在 Visual FoxPro 中还具有创建查询文件和视图文件的功能。

视图与查询有许多相似之处，它们是为快速、方便地使用数据库而提供的两种不同方法，创建的步骤也非常类似。本章从讲解创建查询入手，将全面介绍查询和视图的创建、使用。

7.1　查询

7.1.1　查询的概念

本节所讲的查询就是将事先定义好的一个 SQL SELECT 语句，存放在一个指定的查询文件中，然后通过对该文件的直接或反复使用，提高对数据库的使用效率。

查询文件是一种程序文件，扩展名为 .qpr。它的主体是 SQL SELECT 语句，同时还有和输出定向有关的其他语句。

7.1.2　创建查询

1. 创建查询文件的方法

创建查询文件的方法有多种，常用的方法有通过查询设计器、查询向导、直接编程等。

(1)直接编程法创建查询文件

这种方法对于熟悉 SQL SELECT 命令的用户来说不失是一种方便快捷的方法。其步骤是：

S1：打开文本文件编辑器

MODIFY FILE *QueryName.qpr*

S2：在文本文件编辑窗口编写或修改查询命令。

S3：存盘。

S4：运行查询：DO *QueryName.qpr*

[S5：调试查询文件]

例 7.1　创建一个查询文件：gzpx.qpr，其作用是将 zg.dbc 中职工的工资从高到低排序保存到一个表文件 gzpx.dbf 中，然后运行该查询，最后浏览新生成的表：gzpx.dbf。

S1：MODIFY FILE gzpx.qpr

S2：在文件编辑窗口输入 SQL SELECT 命令：

```
SELECT 姓名,zggzb.* FROM zgqkb,zggzb;
```

ORDER BY 实发 DESC WHERE zgqkb.编号＝zggzb.编号；
INTO TABLE gzpx

S3：存盘：Ctrl＋W 或 Ctrl＋End 或 X

S4：DO gzpx.qpr

S5：USE gzpx

S6：BROWSE

(2)使用查询设计器创建查询

使用查询设计器创建查询的第 1 步就是打开查询设计器。打开查询设计器的方法很多，概括起来无非是两种。一种是使用命令打开；另一种则是通过菜单方式打开。

2. 使用命令打开查询设计器

(1)命令格式

CREATE QUERY [*QueryName*|?] [NOWAIT]

(2)功能

该命令可打开查询设计器。

(3)参数和子句说明

QueryName：要创建的查询文件的名字。

3. 使用菜单打开查询设计器

利用菜单打开查询设计器和打开 Visual FoxPro 的其他各种设计器的步骤相同。

4. 使用查询设计器创建查询

下面以一个具体的实例来介绍使用查询设计器创建查询的步骤。

例 7.2 使用查询设计器建立一个查询文件：cx72.qpr，其功能为求出各系男学生的平均总分、学生人数、最高总分和最低总分，将查询结果写到一个文本文件：fxtj.txt 中。最后运行该查询。

S1：打开项目文件 xsxjgl.pjx

MODIFY PROJECT xsxjgl.pjx

S2：向项目文件中添加数据库文件 xjsjk.dbc

S3：打开查询设计器创建查询：

CREATE QUERY cx72

图 7－1 查询设计器

结果如图 7－1 所示(也可使用菜单操作来打开)。

在查询设计器中有六个选项卡、一个“可用字段”列表框、一个“已选择字段”列表框、一个“函数和表达式”文本框、四个按钮。它们各自的作用是:

· 字段:对应于 SQL SELECT 中的 SELECT 子句,用来从“可用字段”列表框或“函数和表达式”文本框通过“添加”或“添加全部”按钮的操作,向“已选择字段”列表框添加要输出的字段或表达式,或从“已选择字段”列表框通过“移去”或“移去全部”按钮的操作,将不需要输出的字段或表达式移回“可用字段”列表框或“函数和表达式”文本框。

· 连接:对应于 SQL SELECT 中的 JOIN 子句。

· 过滤器:对应于 SQL SELECT 中的 WHERE 子句。

· 排序:对应于 SQL SELECT 中的 ORDER BY 子句。

· 分组:对应于 SQL SELECT 中的 GROUP BY 子句。

· 杂项:对应于 SQL SELECT 中的 INTO,TO 子句。

S4:添加表或视图。右击查询设计器任意空白处,↓ 快捷菜单 →[添加表] ↓ 添加表或视图。

S5:向查询设计器中添加表——确定 SQL－SELECT 中的 FROM 子句。

分别双击添加表或视图对话框中的表:xsqkb,xscjb,将它们添加到查询设计器中,然后单击[关闭],关闭添加表或视图对话框。结果如图 7－2 所示。

图 7－2　添加了表的查询设计器

S6:→“字段”↓ 字段,确定 SELECT 子句中的输出项。

S6－1:→→“Xsqkb.系别”,将它添加到“已选择字段”列表框。

S6－2:在“函数和表达式”文本框输入“AVG(总分) AS 平均总分”,→[添加],将该表达式添加到“已选择字段”列表框。

S6－3:重复 S6－2 的操作,分别将“COUNT(xscjb.学号) AS 人数”、“MAX(总分) AS 最高总分”、“MIN(总分) AS 最低总分”三个表达式添加到“已选择字段”列表框。如图 7－3 所示(也可以单击“函数和表达式”文本框右侧的 … 按钮,打开表达式生成器生成表达式)。

图 7-3 选好了输出项的查询设计器

S7:→“连接”↓ 连接,确定 JOIN 子句。

选择内连接,如图 7-4 所示。

图 7-4 生成了内连接的查询设计器

S8:→“过滤器”↓ 过滤器,确定 WHERE 子句。

选择“xsqkb.性别="男"”,如图 7-5 所示。

图 7-5 生成了过滤条件的查询设计器

S9:→“分组”↓ 分组,确定 GROUP BY 子句。

将“选择字段”列表框中的“xsqkb.系别”字段添加到“分组字段”列表框,结果如图 7-6 所

图 7-6 生成了分组条件的查询设计器

示。

S10：存盘，返回项目管理器，如图 7－7 所示。

S11：→［运行］，结果如图 7－8 所示。

图 7－7　项目管理器

图 7－8　查询运行结果

5. 查询的查看

如前所述，查询实际上是生成一个 SQL－SELECT 命令和其他的辅助命令。所生成的查询结果，在查询生成的过程中可以方便地进行浏览。方法是：右击查询设计器空白处，打开快捷菜单，然后选［查看 SQL］，就可打开写有查询文件的文本框，浏览文件内容。图 7－9 给出了例 7.2 最后生成的查询文件的结果。

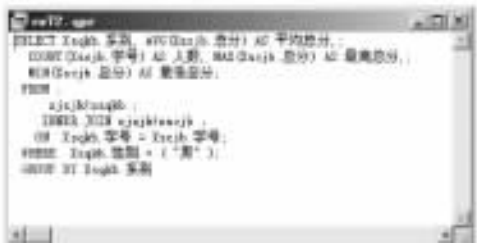


图 7－9　查询设计器生成的 cx72.qpr 文件

查询也可通过查询向导分步输入有关参数，由系统自动生成。如果对生成的查询不满意，还可以打开查询设计器，对查询进行修改。

7.1.3　查询结果的输出去向

查询结果的输出去向，类似于 SQL－SELECT 中的 TO 或 INTO 子句。在 Visual FoxPro 9.0 中，查询中查询去向有七种：浏览、临时表、永久表、屏幕、交叉列表、报表、标签。本节介绍查询去向的设置。

1. 查询去向的设置方法一

Visual FoxPro 9.0 查询去向设置方法一用来设置将查询结果输出到：浏览、临时表、表或屏幕。其中“浏览”指在浏览窗口显示查询结果，这是缺省方式；“屏幕”指在 Visual FoxPro 主窗口或当前活动的输出窗口显示查询结果。现以例 7.3 为例介绍设置步骤。

例 7.3　将例 7.2 的查询 cx72.vue 进行修改，将查询结果写到一个表文件：gfx.dbf 中，然

后打开该表浏览之。

S1：进入图 7－7 所示项目管理器，并选中“查询”下的“cx72”，→[修改]↓ 查询设计器 。

S2：在查询设计器中，调出快捷菜单，→[输出设置] ↓ 查询去向 (Query Destination) →[表(Table)]，在“表名”文本框，写入表名：gfx。如图 7－10 所示。

图 7－10　查询去向对话框

S3：→[确定]← 查询设计器 。

S4：→ ×← 项目管理器 。

S5：→ ! 。

S6：USE gfx

　　BROWSE

此时，将会获得和图 7－8 完全相似的结果，惟一不同的是将图 7－8 的标题“查询”换成了“gfx.dbf”。如果查看查询文件的内容，会发现与图 7－9 相比，多出了最后一个子句：

INTO TABLE gfx.dbf

注意　查询去向设置方法一，也可以通过单击[查询]，选[查询去向]来实现。

2. 查询去向设置方法二

查询去向设置方法二通过“杂项”选项卡(如图 7－11 所示)来实现。该选项卡可以实现查询的交叉列表、报表、标签输出，并可辅助输出到打印机或文本文件。因是菜单及向导操作，读者可自行上机实践。

图 7－11　杂项卡对话框

7.1.4　创建交叉查询

所谓交叉查询就是实现表的不同分类统计信息。它将查询结果以电子表格的形式显示。只有当“选定字段”仅为三项时，才可以使用“交叉查询”。方法与一般的查询向导类似，区别在于“向导选取”对话框中，选中“交叉表向导”项。

例 7.4　创建一个查询文件:cx75.qpr,**将数据库** zg.dbc **工资各项信息按部门进行分类建立交叉查询**。

S1:OPEN DATABASE zg.dbc

S2:SELECT **实发,姓名,部门名称** ;

FROM zggzb INNER JOIN zgqkb ON Zggzb.**编号** = Zgqkb.**编号**;

INNER JOIN zg! bmdm ON Bmdm.**部门代码** = Zgqkb.**部门代码**;

INTO TABLE lsb

S3:→[**新建**]↓ **新建**→"**查询**"→[**向导**] ↓ **向导选择**。

该对话框有三项选择,如图 7-12 **所示,它们分别是**:

- **交叉列表向导**(Cross-Tab Wizard)
- **图形向导**(Graph Wizard)
- **查询向导**(Query Wizard)

图 7-12　**向导选择对话框**

S4:→"**交叉列表向导**"→[**确定**] ↓ **第** 1 **步—选择字段**,**选表**:lsb,**将它的三个字段添加到"选定字段"框**。

S5:→[**下一步**]↓ **第** 2 **步—定义布局**,**定义布局如图** 7-13 **所示**。

S6:**将"性别"字段拖到列**(Column)**框,"姓名"字段拖到行**(Row)**框,"总分"字段拖到数据**(Data)**框,如图** 7-14 **所示**。

图 7-13　**交叉列表向导之一**

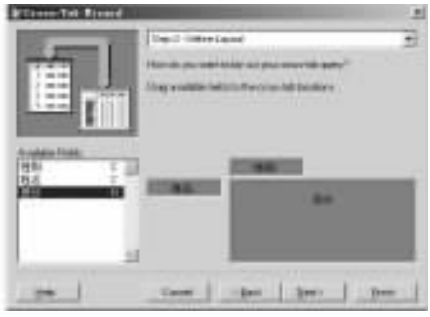


图 7-14　**交叉列表向导之二**

S7:→[**下一步**]↓ **第** 3 **步—添加综述信息**,**添加综述信息,如图** 7-15 **所示**。

该对话框包含两个单选按钮组,它们分别是:总结(Summary)、**分类汇总**(Subtotals)。**各自包含**:

- **总结单选组由五个集函数的单选按钮组成**。
- **分类汇总单选组有四个单选按钮:数据求和**(Sum of data)、**包含数据的单元格数目**(Number of cells containing data)、**占整张表的总计的百分比**(Percentage of the table total)、**无**(None)。

分别选总结为:Sum(),**选分类汇总为**:Sum of data。

S8:→[**下一步**]↓ **第** 4 **步—完成**,**如图** 7-16 **所示**。

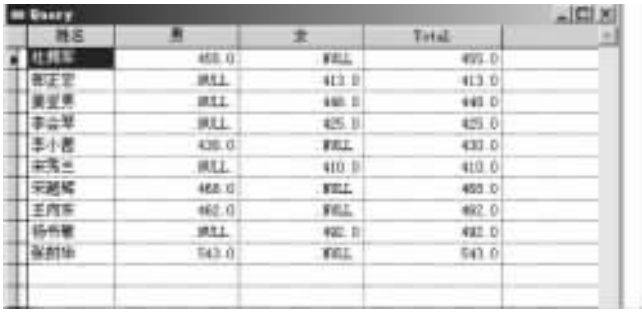
图 7-15 交叉列表向导之三

图 7-16 交叉列表向导之四

S9:→[完成] ↓ 另存为,为查询起名为:cx75.qpr,至此交叉查询创建完成。

S10:DO cx75.qpr

运行结果如图 7-17 所示。



姓名	男	女	Total
杜鹃军	455.0	.NULL.	455.0
陈正宏	.NULL.	413.0	413.0
黄亚男	.NULL.	448.0	448.0
李会琴	.NULL.	425.0	425.0
李小雪	430.0	.NULL.	430.0
刘飞兰	.NULL.	410.0	410.0
宋超辉	468.0	.NULL.	468.0
王西东	462.0	.NULL.	462.0
杨丹雪	.NULL.	482.0	482.0
张前锋	543.0	.NULL.	543.0

图 7-17 交叉查询结果

7.1.5 将查询结果以图形方式输出

在查询向导对话框,选"图形向导",即可在系统的指导下将结果按图形方式输出。下面以一个实例讲解具体步骤。

例 7.5 创建一个表单文件:cx76.scx,将数据库 zg.dbc 工资各项信息按部门进行分类合计查询后,用图形方式输出。

S1:OPEN DATABASE zg

S2: SELECT SUM(实发),部门名称;

FROM zgqkb JOIN zggzb ON zgqkb.编号=zggzb.编号;

JOIN bmdm ON zgqkb.部门代码=bmdm.部门代码;

GROUP BY 部门名称

S3:选择图形向导:→[新建]↓ 新建 →"查询"→[向导] ↓ 向导选择 →"图形向导"。

S4:选查询用字段:→[确定] ↓ 第 1 步—选择字段,选"查询",将它的两个字段添加到"选定字段"框。

S5:定义布局:→[下一步]↓ 第 2 步—定义布局,将"部门名称"字段拖入"坐标(Axis)"

框，将“实发合计”拖入“数据序列(Data Series)”框，结果如图 7－18。

图 7－18　图形向导之二

图 7－19　图形向导之三

S5：选择图形样式：→[下一步]↓ 第 3 步—选择图形样式。

Visual FoxPro 提供了九种图形供用户选择，今选择第 6 种图形——柱状图。如图 7－19 所示。

S6：完成：→[下一步]↓ 第 4 步—完成，在“Type a title for your graph”文本框输入图形标题“部门工资分析图”，如图 7－20 所示。

S7：预览→[Preview]，观察图形效果。

S8：返回向导：→[Return To Wizard] 向导。

S9：→[完成]↓ 表单，输入文件：cx76，→[确定]，系统即运行生成的表单文件，得到图 7－21所示的结果。

图 7－20　图形向导之四

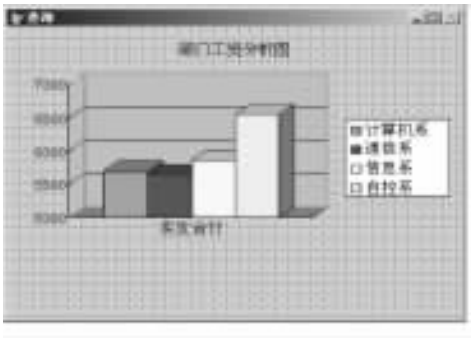


图 7－21　查询的图形输出

7.2　视图

7.2.1　视图的概念

所谓视图指以数据库表为基础导出的一个虚拟表，它可以像表一样的打开、操作、关闭，但

却不是一个真正的表,因为它并没有数据,它的数据来源于数据库表。视图本身的定义也仅存在于数据库的数据字典中。

建立视图的目的是为了查询表数据和更新表数据。利用视图,可以将与之有关的多个表中的数据提取出来组成视图的记录,对这些记录的更新,将被反馈回原所在表中,从而达到对多个表中记录的同时更新,保持数据库表数据的一致。

视图分为本地视图和远程视图。使用当前 Visual FoxPro 数据库中的表所建立的视图称为本地视图;如果数据源用本机中由其他数据库系统生成的数据库或使用网络服务器的数据库,所建立的视图称为远程视图。本书主要讲述本地视图的创建与使用,对远程视图将作一简单介绍。

视图与查询有许多相似之处,如它们都可以用来查询表数据。但它们又有不同之处,如视图可以用来更新表数据,而查询则做不到;查询有查询去向,而视图则没有等。

正因为它们有相似之处,因此,视图的创建和查询的创建也有相似的步骤。

7.2.2 创建视图

和其他文件的创建方法相同,创建视图也分为菜单方式和命令方式两种。

1. 使用视图设计器创建视图

使用视图设计器创建视图首先应该打开数据库,然后打开视图设计器进行设计。打开视图设计器方法类似于打开查询设计器。一般常用如下三种方法之一。

① 使用项目管理器:→[打开]→[项目]↓ 项目管理器 →"数据"→"本地视图"→[新建]。

② 使用菜单方法:→ 🗋 ↓ 新建 →"视图"→[新建](或:[向导])

③ 使用命令:CREATE VIEW *FileName*

例 7.6 用视图设计器创建一个视图文件:st77.vue,数据来自于数据库xjsjk.dbc 中的有关表。

S1: OPEN DATABASE xjsjk

S2: CREATE VIEW

S3:将表:xsqkb、xscjb、xszhjfb 添加到视图设计器中。

S4:用"字段"卡选择字段。

选择xsqkb.dbf中的"学号"、"系别"、"姓名"; xscjb.dbf 中的"总分"、"名次";xszhjfb.dbf 中的"综合积分"、"综合排序"字段。

S5:用"连接"卡建立三个表之间的根据学号的内连接关系。

S6:用"更新条件"卡创建更新条件。

→[更新条件]↓ 更新条件,如图 7-22 所示。

在"更新条件"卡中,包含多个选项,它们的功能分别是:

- "表"下拉式列表框:内有生成视图的各源表,可选择用于修改的表,缺省值为全部表。
- "重置关键字(Reset Key)"按钮:指定重新设置关键字与可修改字段。
- "全部更新(Update All)"按钮:指定全部字段均可被更新。

图 7－22　更新条件框

·“发送 SQL 更新(Send SQL Updates)”复选框：选之，可将在视图上对数据的修改反馈回源表去；反之，则不反馈。该选项要求每个表至少设置一个关键字段。

· ：用来标识关键字段确定视图与源表之间数据的对应关系。要更新源表中的数据，则应包括该表的候选码，且应将该候选码字段设置为关键字段。在 Visual FoxPro 9.0 中，只有设置了关键字段后才可以设置是否更新。如果没有一个字段设置为关键字段，则即使在视图中修改字段值，也不会更新源表中的数据。设置关键字段的方法很简单，只需在该字段前、按钮下方单击使之加上“√”即可。此时会在它的后边出现一个铅笔图标。

· ：指定要更新的字段，设置办法与设置关键字段类似，只要在字段名前、下方单击使之加上“√”即可。

·“SQL WHERE 子句包括 (SQL WHERE clause includes)”单选组：用来设置检测冲突规则。它可以帮助用户管理在对用户访问同一数据库的情况下更新记录的方法。它包含四个单选按钮，分别是：仅关键字段(Key Fields Ony)、关键字和可更新字段(Key and updatable fields)、关键字和已修改字段(Key and modified fields)、关键字和时间戳(Key and timestamp)。

· 使用更新(Update using)单选组：指定更新源表记录的方式，有两个单选按钮，分别是：先删除记录再插入记录(SQL DELETE then INSERT)、直接更新记录(SQL UPDATE)。

各参数选择后如图 7－22 所示。

S7：运行视图，会得到如图 7－23 所示结果。

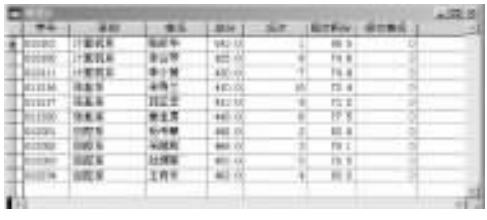


图 7－23　视图执行结果

S8:存盘。→[X]↓[Microsoft Visual FoxPro]→[是]↓[保存],在视图名称文本框输入"st77"→[确定]。

至此,视图设计完成,在数据库字典中会出现一个类似于表的视图文件。它可以像表一样用于被打开、浏览、编辑、统计、创建查询、创建新的视图等。

2. 使用 SQL 命令创建视图

(1)命令格式

CREATE [SQL] VIEW [*ViewName*] [REMOTE]
[CONNECTION *ConnectionName* [SHARE] | CONNECTION *DataSourceName*]
AS *SQLSELECTStatement*

(2)功能

利用 SQL 查询语句创建一个视图。

(3)参数和子句说明

· CREATE [SQL] VIEW:生成一个视图。

· *ViewName*:要生成的视图的名字。

· REMOTE:指定使用远程数据源的表或一个视图,或一个远程视图。缺省使用本地表生成一个视图。

· CONNECTION *ConnectionName*[SHARE] | CONNECTION *DataSourceName* :用来指定在打开视图时,建立一个先前已经定义了的连接的名字或需要连接的已经存在的数据源的名字。

· AS *SQLSELECTStatement*:指定一个用来创建视图的 SQL SELECT 语句。

注意 如果一个数据库已经打开,则所建的视图可以存在数据库中,否则视图不能保存。

例 7.7 利用 SQL 命令重做例 7.6。

```
CREATE VIEW st78 AS SELECT xsqkb.学号,xsqkb.系别,xsqkb.姓名,;
xscjb.总分,xscjb.名次,xszhjfb.综合积分,xszhjfb.综合排名;
FROM xsqkb,xscjb,xszhjfb where xsqkb.学号=xscjb.学号;
AND xscjb.学号=xszhjfb.学号
```

7.3 视图的操作

视图的操作指视图的修改、打开、删除、重新命名等。它们的操作同于表的操作,只是由于视图存在于数据库的数据字典中,因此必须先打开数据库。对于视图的各种菜单操作同于数据库表的操作,下面仅给出视图的操作命令。

(1)视图设计器的打开命令

MODIFY VIEW *ViewName* [REMOTE][NOWAIT]

(2)视图的打开命令

USE *ViewName*

(3)视图的删除命令

DELETE VIEW *ViewName*

(4)视图更名命令

RENAME VIEW Old*ViewName* TO New*ViewName*

7.4　利用视图修改表

如前所述,创建视图的根本目的在于使用视图来更新与之相关的表。现以视图 st77.vue 来讲解更新表数据的方法。

例 7.8　利用视图的编辑,将"xscjb.dbf"、"xszhjfb.dbf"中的"名次"、"综合积分"字段清空。

```
OPEN DATABASE xjsjk
UPDATE st77 SET 总分=0,综合积分=0     && 对 st76.vue 的两个字段清空
USE xscjb IN 0
USE xszhjfb IN 0
SELECT xscjb
BROWSE                  && 会发现 xscjb.dbf 中的"总分"字段已全部为 0
SELECT xszhjfb
BROWSE                  && 会发现xszhjfb.dbf中的"综合积分"字段已全部为 0
```

7.5　创建远程视图

迄今为止,本章介绍的视图的数据源还仅限基于 Visual FoxPro 数据库表。而 Visual FoxPro 9.0提供了访问其他类型的数据库的能力,例如 SQL Server 2000,Microsoft Access,Oracle 等。

Visual FoxPro 9.0 是通过远程视图来访问这些远程数据源的。远程数据源是指安装了 ODBC(Open database Connectivity)驱动程序和设置了 ODBC 数据源的数据服务器。

下面通过一个实例介绍创建远程视图的过程。

例 7.9　创建一个远程视图,用它来访问 Microsoft Access 数据库。

为了使用 Access 数据库,先建一个 Access 数据库:xsqkb.mdb,它里面包含一个表:xsxxb,如图 7-24 所示。

图 7-24　ACCESS 数据库

将该数据库复制到 Visual FoxPro 数据库所在的目录下，现在开始创建远程视图。

S1：打开数据库：OPEN DATABASE xjsjk。

S2：打开远程视图向导，选择数据源：→[新建]→[远程视图]→[向导]↓远程视图向导(Remote View Wizard)第 1 步—选择数据源，如图 7－25 所示。

S3：选择数据库：→[下一步]↓选择数据库。

在该对话框，选择要作为数据源的 Access 数据库：xsqkb.mdb，并选取“独占”复选框，如图 7－26 所示。

图 7－25　远程视图向导第 1 步

图 7－26　选择数据库对话框

S4：选择字段：→[下一步]↓第 2 步—选择字段，从“可用字段”栏向“选定字段”栏挑选有关字段，如图 7－27 所示。

S5：选择记录排序：→[下一步]↓第 4 步—记录排序，从“可用字段”栏向“选定字段”栏挑选“VFP 程序设计”字段，设置为降序(Descending)，如图 7－28 所示。

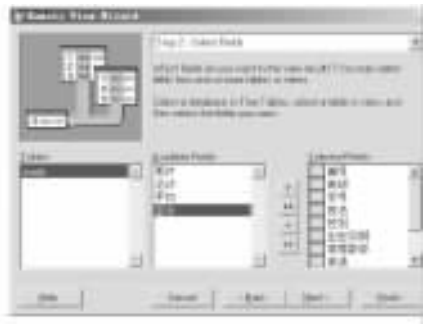


图 7－27　远程视图向导第 2 步

图 7－28　远程视图第 4 步

S6：选择记录筛选：→[下一步]↓第 5 步—记录筛选，设置记录筛选条件，如图 7－29 所示。

S7：完成：→[Next]↓Step6－Finish，如图 7－30 所示。

图 7－29　远程视图向导第 5 步

图 7－30　远程视图第 6 步

S8：为远程视图命名：→［Finish］↓视图名字，在文本框写入视图名：ycst77，结果如图7－31所示。

图 7－31　视图名对话框

S9：添加结束：→［OK］↓数据库设计器，可以看到在数据库中，已添加了远程视图：ycst77，结果如图 7－32 所示。

S10：运行该视图，结果如图 7－33 所示。

图 7－32　添加到数据库 xjsjk 中的远程视图：ycst77

图 7－33　远程视图：ycst77 运行结果

7.6 创建远程数据源连接

连接是远程视图和远程数据源之间的桥梁。通过已命名的连接的定义,用户可以在创建远程视图时引用它,并且可以通过设置连接的属性来优化 Visual FoxPro 9.0 与远程数据源之间的通信。

本节介绍创建远程数据源连接的方法。

7.6.1 设置连接设计器

要建立连接,应首先打开连接设计器,而连接是保存在数据库中的,因此要建立连接,还应首先打开数据库。现首先打开数据:xjsjk.dbc。

打开连接设计器(Connection Designer)的方法有三种。

① 方法 1——菜单法:→[新建] ↓ 新建 →[连接] →[新建]。

② 方法 2——使用项目管理器:打开项目管理器,→[数据] →[连接] →[新建]。

③ 方法 3——数据库打开后使用命令:

CREATE CONNECTION [*ConnectionName*]

使用上述 3 种方法的任一种方法打开连接设计器,如图 7-34 所示。

图 7-34 连接设计器

在“连接设计器”对话框中,可进行如下的设置。

1. “指定的数据源”选项组

本选项组各选项的功能如下:

(1)“数据源、用户标识、密码”单选按钮

如选中该按钮,将如图 7-34 所示,会显示四项内容,它们分别是:

- 数据源:从其中的数据源列表框列出的 ODBC 数据源中选择一个数据源。
- 用户标识:如果数据源有用户标识,则在此输入用户标识。
- 密码:如果数据源需要密码,则在此输入密码。
- 数据库:输入作为数据源连接目标的数据库的名字。

(2)“连接串”单选按钮

如选中该按钮,将显示如图 7－35 中左图所示的“连接设计器”。单击“连接串”文本框右侧的浏览按钮…,会弹出“连接数据源”对话框,如图 7－35 中右图所示。在此框中,用户可以在“文件数据源”和“机器数据源”之中选择。

图 7－35　连接设计器和选择数据源对话框

(3)“验证连接”按钮

对输入的内容进行检查,系统将根据是否连接成功给出相应的信息。

(4)“新建数据源”按钮

用来建立新的数据源。

2. “显示 ODBC 登录提示”选项组

本选项组包括三个单选框,它们的功能如下:

(1)未指定登录信息时显示

如果在命名连接定义中未存储用户标识和密码,在 Visual FoxPro 9.0 中将用“ODBC 数据源注册”对话框来提示用户。

(2)总显示

Visual FoxPro 9.0 中将用“ODBC 数据源注册”对话框来提示用户,在此对话框中,用户可以使用不同于存储在命名连接中的 ID 和密码登录数据源。

(3)从不显示

Visual FoxPro 9.0 不提示用户。

3. “数据处理”选项组

本选项组的内容与函数 DBSETPROP()的设置相对应,由四个复选框和一个列表框组成。它们分别是:

(1)“异步执行”单选按钮

与该函数中的 *Asynchronous* 连接属性相对应。

(2)“显示警告信息”单选按钮

显示不可捕捉的警告信息,与该函数中的 *DispWarning* 连接属性相对应。

(3)“批处理”单选按钮

以批处理的方式进行连接操作,与该函数的 *BatchMode* 连接属性相对应。

(4)“自动事务处理”单选按钮

自动执行事务处理,与该函数的 *Transations* 连接属性相对应。

(5)“包大小”列表框

指定与远程数据源之间传递信息时的信息网络包的大小,单位:字节,它与该函数的 *PacktSize* 连接属性相对应。

4.“超时间隔”选项组

本选项组有四个选项,它们分别是:

(1)连接(秒)

指定以秒为单位的连接超时时间间隔。与函数的 *ConnectTimeout* 连接属性相对应。

(2)空闲(分)

指定以分为单位的空闲超时时间间隔。当超过指定的空闲时间间隔后,活动的连接变为不活动的。与函数的 IdleTimeout 连接属性相对应。

(3)“查询(秒)”

指定以秒为单位的查询超时时间间隔。与函数的 *QueryTimeout* 连接属性相对应。

(4)“等待时间(ms)

以毫秒为单位指定在 Visual FoxPro 9.0 确定 SQL 语句是否执行完毕之前经过的时间。与函数的 *WaitTime* 连接属性相对应。

7.6.2 新建数据源

Visual FoxPro 9.0 中,新建数据源的步骤如下:

S1:打开“创建新数据源”对话框:在“连接设计器”中,→[新建数据源]↓ 创建新数据源之 1,如图 7-36 所示。

它提供了三种数据源类型供用户选择:文件数据源、用户数据源、系统数据源。现选择“用户数据源”。

S2:→[下一步]↓ 创建新数据源之 2,如图 7-37 所示。本对话框用来选择数据源所需要的驱动,现选择“Microsoft Excel Driver(*.xls)”。

图 7-36 创建新数据源之一

图 7-37 创建新数据源之二

S3：→[下一步]↓ 创建新数据源之 3，如图 7－38 所示。

S4：→[完成]↓ ODBC Microsoft Excel 安装，如图 7－39 所示。

图 7－38　创建新数据源之三

图 7－39　ODBC Microsoft Excel 安装

该对话框各个选项及按钮功能如下：

- “数据源名”文本框：用来输入数据源名。
- “说明”文本框：描述数据的内容或为数据库添加注释。
- “选择工作簿”按钮：用来选择 Excel 工作簿。例如选：D:\xyb\djksbmb.xls。
- “选项”按钮：打开驱动程序组，其中：“扫描行数”指定设置列和列数据类型时，设置程序或驱动程序将要扫描的行数，可以为 1 到 16 的任何数。现取最大值：16。“只读”用以禁止对 Excel 文件的更新。

S5：→[确定]← 连接设计器 →[数据源、用户标识码、密码]，在“数据源”列表框选“Myconnect” →[验证连接]

如果连接成功，则给出提示信息“连接成功”，如图 7－40 所示。

S6：→[确定]← 连接设计器 →💾↓ 保存，在“连接名称”文本框输入“MyConnect”，如图 7－41 所示。

图 7－40　连接成功

图 7－41　保存对话框

S7：→[确定]，则连接被保存到了数据库中。

7.7　思考与练习

一、选择题

1. 以下关于查询的描述正确的是(　　)。

A）不能根据自由表建立查询

B) 只能根据自由表建立查询

C) 只能根据数据库表建立查询

D) 可以根据数据库和自由表建立查询

2. 查询设计器中包括的选项卡有(　　)。

A) 字段、筛选、排序依据

B) 字段、条件、分组依据

C) 条件、排序依据、分组依据

D) 条件、筛选、杂项

3. 视图设计器比查询设计器多出的选项卡是(　　)。

A) 字段选取　　B) 排序依据

C) 连接条件　　D) 更新条件

4. 查询设计器和视图设计器的主要不同表现在(　　)。

A) 查询设计器有“更新条件”选项卡,没有“查询去向”选项

B) 查询设计器没有“更新条件”选项卡,有“查询去向”选项

C) 视图设计器没有“更新条件”选项卡,有“查询去向”选项

D) 视图设计器有“更新条件”选项卡,也有“查询去向”选项

5. 在 Visual FoxPro 中,关于视图的正确叙述是(　　)。

A) 视图与数据库表相同,用来存储数据

B) 视图不能同数据库表进行连接操作

C) 在视图上不能进行更新操作

D) 视图是从一个或多个数据库表导出的虚拟表

6. 下列关于查询的说法,不正确的一项是(　　)。

A) 查询是 Visual FoxPro 支持的一种数据库对象

B) 查询就是预先定义好的一个 SQL SELECT 语句

C) 查询是从指定的表或视图中提取满足条件的记录,然后按照想得到的输出类型定向输出查询结果

D) 查询就是查询,它与 SQL SELECT 语句无关

7. 查询设计器中的“筛选”选项卡用来(　　)。

A) 编辑连接条件　　B) 指定查询条件

C) 指定排序属性　　D) 指定是否要重复记录

二、填空题

1. 视图和表的区别在于视图是_____,而表是_____。

2. 视图和查询不同之处在于查询的结果是_____的,而视图却可以用来_____表数据。

3. 利用视图更新表,必须对视图访问的每一个表设置_____字段,否则表的更新将无法进行。

三、上机题

根据前面“xscjgl”数据库中的表：xs.dbf，kc.dbf，cj.dbf，完成以下题目。

1. **用向导设计、建立一个能查询**xs.dbf**的查询**q1.qpr。
2. **用查询设计器建立能查询学生学号、姓名、总分和平均分的查询**q2.qpr。
3. **用向导设计、建立一个能查询**xs.dbf**的视图**v1.vue。
4. **用视图设计器建立一个能更新数据库中三个表的信息的视图**v2.vue。

第 8 章　结构化程序设计基础

在前面讲的交互模式中，用户在命令窗口从键盘上输入一条命令后，按回车键方可执行。这种方式对于初学者显然是非常有利的，它既有助于初学者尽快掌握 Visual FoxPro 9.0 的有关命令，又可以使初学者享受到马上获得命令执行的结果而带来的喜悦。然而，正是因为这种逐条命令的交互执行模式，极大限制了计算机“快”的特长的正常发挥。当学习到一定程度的时候，人们就会产生一种莫明其妙的厌倦心情，而迫切希望能转到另一种模式下进行更高层次的学习和提高。

Visual FoxPro 9.0 所提供的程序模式圆满地解决了这一问题。它不但使计算机“快速、准确、精确、记忆”的特点表现得淋漓尽致，而且还提供了一批仅在程序模式下才有效的命令，从而使 Visual FoxPro 9.0 的功能更为强大。

Visual FoxPro 9.0 既支持面向过程的结构化程序设计，又支持面向对象的程序设计。

8.1　源程序文件的建立、修改与运行

Visual FoxPro 9.0 命令编制的程序，称为程序文件或命令文件，源程序文件的扩展名为“.prg”，经编译或连编以后生成的文件的扩展名为：“.fxp”，“.app”，“.exe”。

8.1.1　程序文件的建立与修改

由于 Visual FoxPro 9.0 自带了简易的文本编辑器，因此可以直接在集成环境中编写程序。打开 Visual FoxPro 9.0 内的文本编辑器的方法和打开其他设计器的方法一样，分为命令方式和菜单方式两种。

1. 命令方法打开程序编辑器

无论是创建还是修改程序源文件，都应在程序编辑器中进行，打开程序编辑器的命令是：

(1)命令格式

MODIFY COMMAND [*FileName* | ?] [NOEDIT] [NOMENU] [NOWAIT]
[RANGE *nStartCharacter*, *nEndCharacter*] [[WINDOW *WindowName*1]
[IN [WINDOW] *WindowName*2 | IN SCREEN]] [AS *nCodePage*] [SAVE]

(2)功能

打开一个文本编辑窗口，以便修改或创建一个程序文件。

(3)参数和子句说明

· *FileName*：指定要打开或创建的程序的文件名。源程序文件的缺省的扩展名是.prg。MODIFY COMMAND 命令允许使用通配符“*”、“?”，此时与通配符相匹配的每一个程序文

件都将在一个编辑窗口被打开。如果文件名缺省，则最初以“程序 1.prg”命名的程序文件将被打开。当用户关闭窗口时，可以将程序另存为其他的文件名。

• ? NOEDIT，NOWAIT，AS *nCodePage*：与前面各命令含义相同。

• NOMENU ：删除 Visual FoxPro 系统菜单条的格式菜单标题，从而避免改变字体等。

• RANGE *nStartCharacter*，*nEndCharacter*：当编辑窗口打开时，指定所选字符的范围。其中 *nStartCharacter* 为开始位置，*nEndCharacter* 结束位置。

• WINDOW *WindowName*1：指定编辑窗口的特性。例如，如果窗口是用 DEFINE WINDOW 中的 FLOAT 选项创建的，则它可以移动。窗口不需要被激活或可见，但必须被定义。

• IN [WINDOW] *WindowName*2：指定被打开窗口的父窗口。编辑窗口并不呈现父窗口的特性但却不能移到父窗口之外。如果父窗口被移动，则编辑窗口将随之而移动。为了访问编辑窗口，父窗口必须首先用 DEFINE WINDOW 命令定义，并且必须是可见的。

• IN SCREEN：当编辑窗口被以 IN WINDOW 子句放置在一个辅助窗口时，如果要在 Visual FoxPro 主窗口打开它，通过 IN SCREEN 字句指明。

• SAVE：允许编辑窗口打开后其他窗口被激活。如果缺省 SAVE，当其他窗口被激活时编辑窗口被关闭。SAVE 在命令窗口下无效。

例 8.1　编写一个求球表面积和球体积的程序 qiu.prg。

MODIFY COMMAND qiu

在程序编辑器中输入如下程序内容录，完成之后即用 Ctrl＋W，Ctrl＋End，X 存盘；若要放弃，可按下 Esc，Ctrl＋Q。

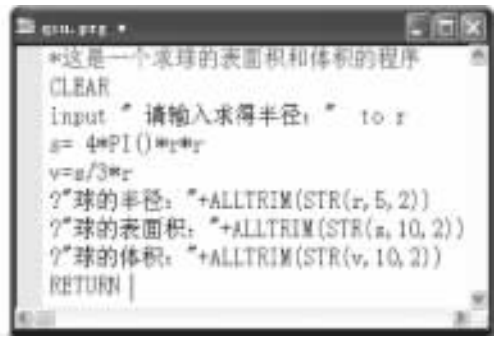


图 8－1　程序编辑窗口

2. 菜单方法打开程序编辑器

利用菜单法打开程序编辑器的步骤与打开其他设计器相同。

3. 程序的三大板块

一个程序大体都可以分为三大板块。

(1)程序初始化板块

在这个板块中，一般将完成对程序运行环境的设置和变量的初始化工作，例如例 8.1 中的 Clear 语句。最常用到的是一系列的 SET 语句。

(2)程序主体板块

该板块用来完成程序的任务。例如,例 8.1 中的从 INPUT 语句开始到"?"球的体积""语句之间的语句都是程序的主体语句。

(3)系统环境设置恢复板块

该板块的任务是当程序所预定的任务完成后,在结束程序运行之前,将系统环境恢复到系统的缺省设置状态,常用的也是 SET 语句。

当然,在程序中也应包含若干个注释语句,以增加程序的可读性,例如例 8.1 的第 1 句:

```
* 这是……
```

8.1.2 程序文件的运行

程序的执行分为两种情形。

1. 编辑状态下运行程序

单击工具栏 ! 按纽,即可运行正在编辑的程序。

2. 执行已存盘的程序

· 使用 DO 命令:

```
DO ProgramName
```

· 使用菜单:→[程序]→[运行]↓ 运行,选择或输入要运行程序的文件名,→[运行]。

例 8.2 运行求球表面积和球体积的程序:qiu.prg。

```
DO qiu
```

在工作区输入半径,例如"5",即会得到相应的结果。

8.2 一些常用命令

8.2.1 输入输出颜色的设置

输入输出颜色命令尽管是为了和较低版本的 Visual FoxPro 及 FoxBase 相兼容而设置的,但却很有用。

(1) 命令格式

```
SET COLOR TO [StandardColor [,EnhancedColor ][,BoundaryColor]]
```

(2) 功能

设置屏幕的标准色、增强色和边框颜色。

(3) 参数说明

· *StandardColor*:设置屏幕的标准颜色。所谓标准颜色,是指输出信息的颜色。它由用"/"分开的一对颜色代码组成,形式是:*ForeColor*/*BackColor*,分别代表系统的前景色和背景色。

· *EnhancedColor*:设置屏幕的增强颜色。所谓增强颜色,是指格式输入时变量值的颜色。设置方法同于标准色。

· *BoundaryColor*:设置边框色,由一个颜色代码组成。

当颜色参数均缺省时表示恢复系统的白底、黑字的缺省设置。

表 8－1 给出了 Visual FoxPro 9.0 中所允许的各种颜色的代码，其数字代码和字母代码可混合使用。

应指出的是：

· 如果在颜色代码的后面加一个“＋”，这表示以高亮度显示。

· 颜色设置语句所设置的颜色，将从下一个执行语句起有效，直到碰到一个新的颜色设置语句为止。

表 8－1　颜色代码

颜色	黑	蓝	绿	青	红	品红	黄	白
数字代码	0	1	2	3	4	5	6	7
字母代码	N	B	G	BG	R	RB	GR	W

例 8.3　定位输入、输出与颜色设置命令练习。

```
* L83.prg，定位输入与输出命令练习
SET COLOR TO 4＋/B＋,6＋/g＋
CLEAR
xm＝SPACE(10) && 内存变量赋值。
@10,10 SAY "请输入您的姓名：" FONT "宋体",24 GET xm FONT "宋体",24
READ
@10,10 SAY ALLTRIM(xm)＋"，您好！ 欢迎您学习 Visual FoxPro 9.0!";
FONT "楷体",30 COLOR 5＋/2＋
RETURN
```

运行 L83.prg：DO L83

设运行中输入了一个字符串“王向东”，则运行中及运行结果分别如图 8－2、图 8－3 所示。

图 8－2　输入窗口

图 8－3　运行结果

8.2.2 运行控制命令

1. 等待命令(WAIT)

本命令用于暂停程序的运行,等待用户按下键盘的任意一个键或鼠标后继续程序的运行。

(1)命令格式

WAIT [*cMessageText*] [TO *VarName*] [WINDOW [AT *nRow*, *nColumn*]]
[NOWAIT] [CLEAR | NOCLEAR] [TIMEOUT *nSeconds*]

(2)功能

显示信息文本并暂停 Viusal FoxPro 程序执行,直到用户按任意键或单击鼠标左键。

(3)参数和子句说明

· *cMessageText*:指定用于显示等待时的信息,最长 255 个字符。如果缺省,则将显示"按任意键继续……",Visual FoxPro 会将对象引用自动转换为字符串"对象"。如果函数的返回值不是一个非字符型数据,则系统自动使用 TRANSFORM()函数将它转换为字符型数据。

· TO *VarName*:指定将用户所按键的值存放在字符型内存变量中,若缺省此子句则不保存按键的值。

· WINDOW [AT *nRow*, *nColumn*]]:选中时,可以将所要显示的信息在一个窗口显示,该窗口默认的位置为屏幕的右上角,可使用 [AT *nRow*, *nColumn*]子句指定窗口显示位置。

· TIMEOUT *nSeconds*:用来设定 WAIT 命令等待的时间,单位是秒。如果限定时间内用户没有做任何操作,则程序继续运行。

· NOWAIT:意义同前。

· CLEAR :在程序中,将移去 Visual FoxPro 系统窗口,或从 Visual FoxPro 主窗口移去 WAIT 的信息窗口。例如:如果使用了 SET TALK WINDOW,则索引、排序等将被控制在 Visual FoxPro 窗口中。如果此时按了任一个键或移动了鼠标,则窗口将被交互式移去。该关键字仅在程序中有效。

· NOCLEAR :指定将 WAIT 信息窗口保留在 Visual FoxPro 主窗口中,直到 WAIT CLEAR 或其他的 WAIT WINDOW 命令执行,或者一个 Visual FoxPro 系统信息被显示为止。

例 8.4 Wait 命令练习。

```
* L84.prg
SET TALK OFF
CLEAR
WAIT "秦东大学热烈欢迎您的到来!" WINDOW;
AT SROWS()/2,SCOLS()/2 TIMEOUT 2
SET TALK ON
RETURN
```

2. 终止程序运行命令

终止程序运行的命令有三条,它们分别是:RETURN、CANCEL、QUIT。

· RETURN 命令用来终止正在执行的程序,并返回到上级调用程序或命令窗口下。

· CANCEL 命令中断正在执行的程序,返回到命令窗口下,并清除所有的私有和局部内存变量。

· QUIT 命令在终止程序运行后，清除所有用户内存变量，关闭所有文件，退出 Visual FoxPro 9.0 返回到 WINDOWS 状态。

8.2.3　常用状态设置命令

1. 设置会话状态命令(SET TALK ON | OFF | WINDOW [*WindowName*] | NOWINDOW)

Visual FoxPro 9.0 的一个特点是命令执行时会在屏幕上显示命令执行的有关信息，称为“会话(TALK)”。在交互方式下，这种人机对话方式可以帮助用户了解命令执行的情况。但在程序模式下，频繁显示执行信息与输出语句的输出结果相互夹杂，既使屏幕显得零乱不堪，又会严重降低程序的执行速度。用户通过 SET TALK 命令设置会话状态。

· ON：(缺省值)允许将会话设置到 Visual FoxPro 窗口、系统信息窗口、图形状态窗口、或者用户自定义的窗口。

· OFF：在上述窗口中关闭会话。

· WINDOW [*WindowName*] ：指定要打开或关闭会话的用户自定义窗口名。

· NOWINDOW：直接在 Visual FoxPro 窗口关闭或打开会话。

2. 设置系统提供保护状态命令(SET SAFETY ON|OFF)

在命令状态下，用户向 Visual FoxPro 9.0 发出修改、删除、清表等命令时，系统出现文件操作确认提示窗口，向用户提供下一步操作选择。在程序中，常常不需要这种信息，通过设置 SET SAFETY ON|OFF 命令的状态实现是否出现该类窗口，默认为 ON。

3. 设置默认路径命令(SET DEFAULT TO *Path*)

在完成一个项目(PROJECT)时，为了避免将本项目的文件与别的文件混为一谈，需要为之建立一个文件夹，然后将本项目所有文件存放在该文件夹中。以后每次上机，在启动 Visual FoxPro 9.0 后，只要将该文件夹设置成当前路径即可，这会给工作带来很大的方便。在带有还原卡或还原软件的机房公用计算机上，一般这个文件夹应设置在对普通用户开放的磁盘上。

例 8.5　设用户的文件夹为：d:\xxvfp9，现请将该文件夹设置为缺省路径。

```
SET DEFAULT TO d:\xxvfp9
```

此后，在本次 Visual FoxPro 运行期间，只要文件没有指明路径，则一定指 d:\xxvfp9 文件夹中的文件。

8.2.4　系统提示信息窗口 MESSAGEBOX()函数

MESSAGEBOX()函数是 Visual FoxPro 9.0 提供的重要函数之一，用户只要给出几个参数，便可以得到一个固定格式的系统窗口。

(1) 函数格式

MESSAGEBOX(*eMessageText* [, *nDialogBoxType*][, *cTitleBarText*][, *nTimeout*])

(2)功能

显示一个用户定义的对话窗口，并根据所按的键返回一个对应的数值。

(3)参数说明

· *eMessageText*：设定在对话窗口中要出现的文本。用户也可以设定一个有效的 Visual FoxPro 函数、对象或数据类型替代 *eMessageText*。文本最多 1 024 个字符。非字符型数据将

被使用 TRANSFORM()函数自动转换成字符型数据。

• *nDialogBoxType*：用来设定对话框的类型，即出现在对话框中的按钮和图标，缺省按钮及对话框特性。它由三部分组成。即：

nDialogBoxType＝*n*1＋*n*2＋*n*3

如果 *nDialogBoxType* 缺省，等价于将它设置为 0。

表 8－2 给出了对话框类型 *nDialogBoxType* 的三个分量的作用及取值。

nDialogBoxType 可三个值加起来一块使用，其中的每一个值都由表 8－2 中取得。例如：290 只能解释为(2＋32＋256)，代表所给出的对话框应具有如下特征：[终止]、[重试]、[忽略]三个按钮；信息框显示为问号图标；第 2 个按钮[重试]为缺省按钮。

表 8－2 MESSAGEBOX 函数中 *nDialogBoxType* 的三个分量取值及意义

*n*1		*n*2		*n*3	
取值	决定窗口中按钮的个数与内容	取值	决定窗口中图标类型	取值	决定默认按钮
0	仅[确定]	0	无	0	第 1 个按钮
1	[确定]、[取消]	16	停止符(×)	256	第 2 个按钮
2	[终止]、[重试]、[忽略]	32	提问符(?)	512	第 3 个按钮
3	[是]、[否]、[取消]	48	警告符(!)		
4	[是]、[否]	64	提示信息符(i)		
5	[重试]、[取消]				

• *cTitleBarText*：设定对话框的标题。如果缺省该参数，则“Microsoft Visual FoxPro ”将出现在标题栏。

函数的返回值代表用户按下的按钮的编号值，其返回值的含义如下：

1——[确定]；2——[取消]；3——[终止]；4——[重试]；5——[忽略]；6——[是]；7——[否]；－1——未击任何键。

• *nTimeout*：设置一个在清除信息文本前未从键盘或鼠标输入数据时，以毫秒为单位的 Visual FoxPro 显示时间。用户可设定一个有效的超时值。如果该值小于 1，则等价于缺省超时设置——不设置超时时限。

例 8.6 设置一个信息提示窗口，要求有三个按钮：[是]、[否]、[取消]，缺省按钮为[是]，信息图标为信息号。

```
? MESSAGEBOX("这是一个演示信息框的演示例子",3+64+0,"演示实例")
```

结果如图 8－4 所示。在单击[是]时会返回按钮[是]的标号 6。

图 8－4 信息对话框

注意 如增加一个超时参数值：2 000，但在该时限内并不按任何一个键，则超时后信息框

自动消失，且返回一个－1。

8.2.5　窗口操作命令

1. 窗口的定义

(1)**命令格式**

DEFINE WINDOW *WindowName*1
FROM *nRow*1, *nColumn*1 TO *nRow*2, *nColumn*2
| AT *nRow*3, *nColumn*3 SIZE *nRow*4, *nColumn*4
[IN [WINDOW] *WindowName*2 | IN SCREEN | IN DESKTOP [NAME *ObjectName*]
[FONT *cFontName* [, *nFontSize* [, *nFontCharSet*]]] [STYLE *cFontStyle*]
[DOUBLE | PANEL | NONE | SYSTEM | *cBorderString*]
[CLOSE | NOCLOSE] [FLOAT | NOFLOAT] [GROW | NOGROW] [MDI | NOMDI]
[MINIMIZE | NOMINIMIZE] [ZOOM | NOZOOM] [ICON FILE *FileName*1]
[FILL *cFillCharacter* | FILL FILE *FileName*2]
[COLOR SCHEME *nSchemeNumber* | COLOR *ColorPairList*]]

(2)**功能**

生成窗口并设置窗口属性。

(3)**参数和子句说明**

• *WindowName*1：**指定要创建的窗口的名字。**

• FROM *nRow*1, *nColumn*1 TO *nRow*2, *nColumn*2 ：**设定用户自定窗口在** Visual FoxPro **主窗口中的位置及大小。** FROM *nRow*1, *nColumn*1 **为用户自定义窗口在** Visual FoxPro **主窗口中的左上角的位置坐标；**TO *nRow*2, *nColumn*2 **为其右下角的位置坐标。**

• AT *nRow*3, *nColumn*3 SIZE *nRow*4, *nColumn*4：**设定自定义窗口的位置和大小。** AT *nRow*3, *nColumn*3 **为用户自定义窗口在** Visual FoxPro **主窗口中的左上角的位置坐标。** SIZE *nRow*4, *nColumn*4 **设定自定义窗口的行数和列数。**

• IN [WINDOW] *WindowName*2：**设定将用户自定义的窗口放置的父窗口。这样用户自定义的窗口将为该父窗口的子窗口，不允许由父窗口移出。**

• IN SCREEN ：**指明自定义的子窗口放置在** Visual FoxPro **主窗口中。这是缺省的。**

• IN DESKTOP ：**指明自定义的子窗口放置在** Visual FoxPro **主窗口之外的** Windows **桌面上。**

• NAME *ObjectName*：**生成一个可访问窗口对象。**

• FONT *cFontName*[, *nFontSize* [, *nFontCharSet*]]：**设置窗口中要放置的文本的字形。** *cFontName* **为字形名。** *nFontSize* **为字的大小，单位：点，缺省时为** 9 **点。用户可以使用** *nFontCharSet* **指定一个脚本语言。关于有效的脚本语言值请参看** GETFONT()**函数。如果缺省** FONT **子句，字的大小用缺省值：**10 **点。如果用户定义的字形无效，则与该字形特征相类似的字形将替代用户的设置。**

• STYLE *cFontStyle*：**设定窗口文本的字体。如果缺省或者设置的字体无效，将使用常规字体。这些字体符也可混合使用。表** 8－3 **给出字体和它们所对应的字体字符。**

表 8-3 字体字符与字体对照表

字符	字体风格	字符	字体风格
B	粗体(Bold)	—	删除线(Strikeout)
I	斜体(Italic)	T	透明(Transparent)
N	常规(Normal)	U	下划线(Underline)
Q	不透明(Opaque)		

• TITLE *cTitleText*:为窗口设置一个标题,该标题在窗口标题栏中居中。

• DOUBLE | PANEL | NONE | SYSTEM | *cBorderString* :设置用户自定义的窗口的边框式样。其缺省值是单线框。表 8-4 给出了它们各自的作用。DOUBLE 或用户边框字符串可生成一个宽边框的窗口。CLOSE,FLOAT,CROW,ZOOM,MINIMIZE 子句,甚至在无 SYSTEM 窗口定义子句时,也会将对应的控制放置到窗口。

表 8-4 边框关键字及其功能描述

关键字	描 述
DOUBLE	双线框
PANEL	宽边框
NONE	无边框
SYSTEM	用户自定义的窗口类似于系统窗口。当包含某些别的子句(GROW,ZOOM 等)时,对应的窗口控制将放到窗口边框上
cBorderString	设置用户自定义边框的字符串

• GROW|NOGROW、FLOAT|NOFLOAT、MDI|NOMDI:分别为是否允许使用键盘或鼠标改变窗口大小、是否允许使用键盘或鼠标移动窗口、是否生成一个用户自定义的 MDI 自适应窗口。

MDI(multiple document interface:多文档界面)是一个规范,它允许多有文档窗口并定义它们的结构和行为。如果缺省 MDI,则生成的窗口不是 MDI 自适应窗口。当 MDI 自适应窗口最大化时,有三大特点:一是窗口将采用 Visual FoxPro 主窗口的尺寸,窗口控制将不出现,而弹出式菜单的图标将出现在 Visual FoxPro 系统菜单栏中,窗口的存储按钮也被放置在 Visual FoxPro 系统菜单栏;二是窗口标题被放置在 Visual FxoPro 标题栏中,并且用一个连接号和 Visual FoxPro 标题分开;三是如果激活了别的 MDI 自适应窗口,它将自动最大化。

• MINIMIZE|NOMINIMIZE、ZOOM|NOZOOM、CLOSE|NOCLOSE:是否有最小化、最大化、关闭按钮。

• ICON FILE *FileName*:设定窗口被最小化时要显示的图标。此时,在 DEFINE WINDOW 中必须包含 MINIMIZE 关键字。注意,只能指定一个图标(.ico)文件,而不能是位图(.bmp)文件。

• FILL FILE *FileName2*:设定窗口的背景。它可以是一个位图(.bmp)文件。

• COLOR SCHEME *nSchemeNumber*：设定用户自定义窗口的颜色。调色板的编号是 1～24。在 Visual FoxPro 中，第 13，14，15 号调色板保留给内部使用，请用户不要使用它们。如果缺省，窗口的颜色将由 1 号调色板来控制。

• COLOR *ColorPairList*：设定用户自定义窗口的颜色。*ColorPairList* 既可以使用前面讲过的颜色代码设置，又可以使用 RGB()函数。RGB(*n*1，*n*2，*n*3，*n*4，*n*5，*n*6)中，*n*1，*n*2，n3 用于设置前景的红、绿、蓝三色强度；*n*4，*n*5，*n*6 用于设置背景的红、绿、蓝三色强度，它们各自的取值范围都是 0～255，0 为最弱，255 为最强。

2. 窗口的启动

窗口定义完后，只有激活才能显示。

(1)命令格式

ACTIVATE WINDOW *WindowName*1 [，*WindowName*2...]
| ALL [IN [WINDOW] *WindowName*3 | IN SCREEN
[BOTTOM | TOP | SAME] [NOSHOW]

(2)功能

显示并激活用户自定义窗口或 Visual FoxPro 系统窗口。

(3)参数和子句说明

• *WindowName*1 [，*WindowName*2...]：要激活并显示的窗口名字的列表。

• ALL：指定所有的窗口都将被激活。最后激活的窗口是当前窗口。

• IN [WINDOW] *WindowName*3：设定父窗口的名字。

• IN SCREEN：在 Visual FoxPro 主窗口放置并激活一个窗口。

• BOTTOM | TOP | SAME：设置被激活的窗口与先前激活的窗口的相关位置。缺省时，当窗口被激活时将出现在最顶层。BOTTOM 将窗口放置在最底层；TOP 将窗口放置在所有窗口之前；SAME 激活窗口但不影响它的前后次序。

• NOSHOW：它将激活并直接输出到窗口，但并不显示窗口。

3. 窗口的释放

窗口使用结束后要及时释放，否则会占用大量的内存。

(1)命令格式一

• 命令格式：RELEASE WINDOW [*WindowNameList*]

• 功能：从内存中删除用户自定义窗口或 Visual FoxPro 系统窗口。

• 参数说明：*WindowNameList* 为要删除的窗口名。如果缺省，则仅删除活动的用户自定义窗口。

(2)命令格式二

• 命令格式：CLEAR WINDOWS

• 功能：从内存中删除所有的用户自定义窗口。可以使用 SAVE WINDOW 命令把窗口定义保存到一个文件或备注型字段中，以便后面使用。

4. 窗口综合示例

例 8.7　这是一个窗口应用的综合示例，读者可在上机时修改各种参数，体会命令的用法。

```
* L87.prg
DEFINE WINDOW wn1 FROM 2,2 TO 20,80;
TITLE "我的窗口 1" SYSTEM;
COLOR RGB(255,255,0,255,0,0);
CLOSE MINI ZOOM FONT "隶书",24 STYLE "UB"
DEFINE WINDOW wn2 AT 2,85 SIZE 6,50;
TITLE "我的窗口 2" DOUBLE;
COLOR RGB(0,255,0,,,) FONT "楷体",14
ACTIVATE WINDOW wn1
@2,5 SAY '认真实践"八荣八耻",'
@row()+1,5 SAY '人人争当文明公民!'
ACTIVATE WINDOW wn2
@3,12 SAY "清正廉洁高效,构建和谐社会!"
WAIT""
CLEAR WINDOWS
RETURN
```

L87.prg 执行的结果如图 8-5 所示。

图 8-5 L87.prg 结果

8.2.6 其他辅助命令

1. 注释命令

用于在程序中插入注释行,提高程序的可读性。它可以采用以下两种格式:

(1)整行注释

* *cNoteText* 或 NOTE *cNoteText*

(2)命令的末尾加注释说明

&& *cNoteText*

2. CLEAR 命令

我们在前面的各章节中,都接触过 CLEAR 命令,现给出它的完整格式。

(1)命令格式

CLEAR [ALL | CLASS *ClassName* | CLASSLIB *ClassLibraryName*
| DEBUG | DLLS [*cAliasNameList*] | EVENTS | ERROR | FIELDS

| GETS | MACROS | MEMORY | MENUS | POPUPS | PROGRAM | PROMPT | READ [ALL] | RESOURCES [*FileName*] | TYPEAHEAD | WINDOWS]

(2)**功能**

从内存中释放指定的项目。

(3)**参数及子句说明**

· ALL:**从内存中释放所有的变量和数组以及用户自定义的所有菜单条、菜单和窗口;也可以关闭任何表、所有的索引、格式文件和备注文件,并选择** 1 **号工作区;还可以从内存中释放用** DECLARE－DLL **声明的所有外部函数。但它不能释放系统变量,不能释放编译程序缓冲区,也不能释放对象类型变量。**

· CLASS *ClassName*:**释放类定义**。

· CLASSLIB *ClassLibraryName*:**释放包含在可视化类中的所有类定义**。

· DEBUG:**释放调试器中的所有断点,并且将调试器窗口存储到它们默认的位置**。

· DLLS [*cAliasNameList*]:**释放用** DECLARE－DLL *cAliasNameList* **所声明的外部共享库**。

· EVENTS:**停止用** READ EVENTS **命令所开始的事件处理过程。当** CLEAR EVENTS**执行后,程序将继续执行** READ EVENTS **紧后面的程序**。

· ERROR:**重新设置** Visual FoxPro **错误结构,如同没有错误发生**。

· FIELDS:**释放用** SET FIELDS **生成的一个列表,并且执行** SET FIELDS OFF。

· GETS:**释放所有的**@…GET **控制**。

· MACROS :**从内存中释放所有的键盘宏,包括任何** SET FUNCTION **键分配**。

· MEMORY:**从内存中释放所有的公共内存变量、私有内存变量和数组,但不能释放系统内存变量**。

· MENUS:**从内存中释放所有的菜单定义**。

· POPUPS:**从内存中释放所有用** DEFINE POPUP **所生成的菜单定义**。

· PROGRAM:**释放编译程序缓冲区**。

· PROMPT:**释放用**@…PROMPT **命令生成的菜单**。

· READ [ALL]:**向后兼容。已被** CLEAR EVENTS **代替**。

· RESOURCES [*FileName*]:**指定要从内存中释放隐藏的位图、图形、字型、光标或图标文件。如果不给出文件名,则所有的位图、图形、字型、光标和图标文件将被从内存中释放**。

· TYPEAHEAD:**删除键盘击前缓冲区。当用户想避免输入一个字段或避免在字段或提示显示前就响应提示时,它非常有用**。

· WINDOWS:**从内存中释放用户自定义的窗口定义和从** Visual FoxPro **主窗口或活动的用户自定义窗口清除窗口。使用** SAVE WINDOW **命令可以把窗口定存储到一个文件或备注型字段中,以便后面使用。使用** CLEAR WINDOWS **也可以释放访问表单的任何系统变量**。

例 8.8　清除 L87.prg **所显示的窗口内容**。

```
CLEAR WINDOWS
```

3. 表达式输入命令

(1)**命令格式**

INPUT [*cMessageText*] TO *MVarName*

(2)**功能**

将由键盘上敲入的表达式运算的结果赋给内存变量:*MVarName*。

(3)**参数说明**

• *cMessageText*:**提示信息**。

• *MVarName*:**交互式获得表达式值的内存变量名**。

注意　表达式可以是由键盘上输入的常量、变量的名字、表达式。如果是常量则非数值型常量必须有各自的定界符;若是变量,则该变量必须事先已有了确定的值;如果是表达式,则表达式内的各个变量、函数都必须有定义。

例 8.9　观察 L89.prg **运行的结果**。

```
* L89.prg
A=6
B=7
Input"请输入表达式:A+B:" TO x
? x
RETURN
```

程序运行时,当用户从键盘上击入表达式 a+b,**就会显示出** 13。

4. 字符串输入命令

(1)**命令格式**

ACCEPT [*cMessageText*] TO *cMVarName*

(2)**功能**

将由键盘上敲入的字符串赋给内存变量:*cMVarName*

(3)**参数说明**

cMVarName:**交互式获得表达式值的字符型内存变量名**。

注意　由键盘上输入的字符串无需加定界符,如加了定界符,则定界字符也算作字符串的一部分。

例 8.10　观察 L810.prg **运行的结果**。

```
* L810.prg
A=6
B=7
ACCEPT "请输入一个字符串:" TO x
? A,B,x
RETURN
```

程序运行时,用户从键盘上击入"欢迎您进入 Visual FoxPro **程序设计的学习!",观察显示结果。**

8.3　程序的控制结构

与其他高级语言相似，Visual FoxPro 9.0 程序也有三种基本的控制结构，即顺序结构、选择结构与循环结构。本节分别介绍这三种结构及特点。

8.3.1　顺序结构

顺序结构是程序设计中最简单的基本结构。其特点是：在这种结构中，按照各条命令编程时书写的先后顺序依次逐条执行。本章前几节例题的程序结构均为顺序结构。

对于程序设计而言，在进行程序代码设计之前，应先设计出程序流程图。程序流程图有多种，目前使用最广泛的有三种。它们分别是：传统的程序流程图、N－S 图、PAD 图。本书各个程序的流程图为 N－S 图。

例 8.11　编程序求解鸡兔同笼问题。已知鸡兔总头数为 h，总脚数为 f，求鸡兔各有多少只？

根据题意，设鸡为 x 只，兔为 y 只，则有：

$$x = (4h - f)/2$$
$$y = h - x$$

据此写出流程图如图 8－6 所示，程序如下：

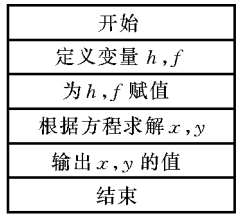


图 8－6　求鸡兔同笼流程图

```
* L811.prg
SET TALK OFF
CLEAR
h=0
f=0
s="输入的总脚数应大于总头数的 2 倍且应为偶数!"
WAIT s WINDOWS TIMEOUT 2
INPUT "请输入总头数:" TO h
INPUT "请输入总脚数:" TO f
x=(4 * h-f)/2
y=h-x
@6,20 SAY "有鸡"+ALLTRIM(STR(x))+"只"
```

```
@8,20 SAY "有兔"+ALLTRIM(STR(y))+"只"
SET TALK ON
RETURN
```

运行该程序，从键盘上输入：50，160，请观察运行结果。

8.3.2 选择结构

选择结构是另一种常见的程序结构，其特点是：根据所给定的选择条件为真（即分支条件成立）与否，而决定执行程序哪个分支中的语句（序列），并且在任何情况下都具有“无论分支多少，仅选其一”的特性。

1. IF…ELSE…ENDIF 结构

(1)语句结构与流程图

```
IF lExpression [THEN]
    Commands1
[ELSE
    Command2]
ENDIF
```

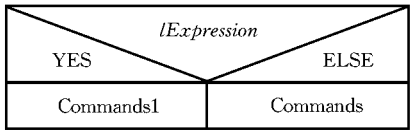


图 8-7 双分支选择结构 N-S 图

(2)语句功能

程序运行至 IF 语句时，首先判断逻辑表达式的值，当结果为.T. 时，执行程序段 Commands1，否则执行 Commands2，无论执行 Commands1 还是 Commands2，执行完后均执行 ENDIF 的后续语句。本结构也可以没有 ELSE 部分，此时当逻辑表达式结果为.T. 时，执行程序段 1，否则，相当于 IF 语句不存在，继续执行后续程序。

例 8.12 输入姓名之后输入性别，再根据所输入的性别输出不同的信息。

```
* L812.prg,选择结构语句练习
SET TALK OFF
CLEAR
xm=SPACE(10) && 给变量赋初值。
xbdm="1" && 性别初值为"男"
@2,23 SAY "请输入您的姓名：" GET xm
@4,6 SAY "请输入您的性别[1→男,2→女]：" GET xbdm VALID xbdm $"12"
READ
CLEAR
```

```
IF xbdm="1"
    xb="先生"
ELSE
    xb="女士"
ENDIF
@6,10 SAY ALLTRIM(xm)+xb+",您好！ 计算机世界 WELCOME YOU !!";
    FONT "楷体",16 COLOR w/2+
WAIT "按任意键返回!" WINDOW AT 25,20
SET TALK ON
RETURN
```

在本程序的 IF……ENDIF 也可用 IIF()函数来代替相应的程序段。IIF()函数的格式如下：

IIF(*lExpression*,*Expression*1,*Expression*2)

函数的功能是当 *lExpression* 的值成立(为.T.)时，该函数返回 *Expression*1 的值，否则返回，*Expression*2 的值。本例中的 IF 块可用：

```
xb=IIF(xbdm="1","先生","女士")
```

语句代替。

例 8.13　在数据表"xsqkb.dbf"中查询指定姓名的记录，并显示相关信息。

```
* L813.prg
SET TALK OFF
CLEAR ALL
ACCEPT "请输入您要想查询的姓名:" TO lxm
USE xsqkb
LOCATE FOR ALLTRIM(姓名)=ALLTRIM(lxm)
IF FOUND()
    IF MESSAGEBOX("记录已经查到,要显示吗?",4+32+0,"顺利找到!")=6
        DISPLAY 学号,姓名,性别,出生日期
    ENDIF
ELSE
    MESSAGEBOX("没有您要找的记录！",16)
ENDIF
USE
SET TALK ON
RETURN
```

2. DO CASE…ENDCASE 结构

在 Visual FoxPro 9.0 中，如果条件分支超过两个时，用 IF 语句虽然也能完成，但 IF 语句会比较长，程序的清晰度会降低。此时用 DO CASE…ENDCASE 语句就方便多了。

(1)语句结构与流程图

```
DO CASE
    CASE lExpression1
```

```
        Commands1
    CASE lExpression2
        Commands2
        ……
    CASE lExpression_n
        Commandsn
    [OTHERWISE
        Commands_{n+1}]
ENDCASE
```

条件					
1	2	3	…	n	其它
C_1	C_2	C_3	…	C_n	C_{n+1}

图 8-8 多路分支语句 N-S 图

(2)**语句功能**

程序在执行 DO CASE **命令时,依次判断各表达式是否满足,若条件成立,就执行相应的程序段,执行完后转向执行** ENDCASE **后续的语句程序。所以在一个** DO CASE **结构中,最多只能执行一个** CASE **语句对应的程序段。**

例 8.14 输入一个 1～7 **之间的数字,输出对应星期几,即** 1 **对应"星期日",**2 **对应"星期一"……**7 **对应"星期六"。**

```
* L814.prg
SET TALK OFF
CLEAR ALL
CLEAR
lxq=1
@2,13 SAY "请输入数字:" GET lxq
READ
DO CASE
    CASE lxq=1
        lweek="日"
    CASE lxq=2
        lweek="一"
    CASE lxq=3
        lweek="二"
    CASE lxq=4
        lweek="三"
```

```
    CASE lxq=5
        lweek="四"
    CASE lxq=6
        lweek="五"
    CASE lxq=7
        lweek="六"
ENDCASE
MESSAGEBOX("对应的星期为"+"星期"+lweek,0,"转换结果")
SET TALK ON
RETURN
```

8.3.3　循环结构

循环结构也是另外一种重要的程序结构，使用它可以避免重复同一程序段代码的书写，简化程序，提高效率。在 Visual FoxPro 9.0 **中，提供了三种循环结构。**

1. FOR 循环(FOR…NEXT/ENDFOR)

(1)语句结构

```
FOR VarName = nInitialValue TO nFinalValue;
[STEP nIncrement]
Commands
[EXIT]
[LOOP]
ENDFOR | NEXT
```

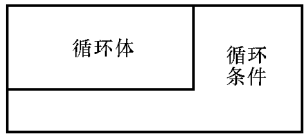


图 8-9　**循环结构流程图**

(2)功能

执行一个确定了重复次数的命令集。FOR **称为** FOR **循环开始语句，**ENDFOR|NEXT **称为** FOR **循环结束语句，夹在它们中间的命令序列称为循环体。**

(3)参数和子句说明

· *VarName* **：循环计数器变量名。**

· *nInitialValue*、*nFinalValue*：**计数器的初值、终值。**

· STEP *nIncrement*：**设置计数器值的递增或递减的步长值。***nIncrement* **为负数表示计数器值递减，为正数表示计数器值递增，缺省时为** 1。

· *Commands* **：要执行的** Visual FoxPro **命令序列。**

·EXIT:循环断路语句。当循环碰见该命令时,立即将控制转移到循环体外执行 ENDFOR 紧下面的命令。它可以出现在循环体内的任何位置。

·LOOP:循环短路语句。当循环碰见该命令时,立即停止执行 LOOP 和 ENDFOR 之间的语句而将控制转移到 FOR 子句,使计数器自动更新。

·ENDFOR|NEXT:循环的结束语句。

FOR 循环体内的 Visual FoxPro 命令序列一直执行到碰见 ENDFOR 或 NEXT 关键字。计数器变量的值然后按步长值递增或递减,再将更新后的计时器值与循环终值比较,当步长为正数时,如果计数器值小于或等于终值则继续执行循环体,否则退出循环转而去执行 ENDFOR 或 NEXT 后面的第 1 条命令。当步长为负数时,如果计数器值大于或等于终值则继续执行循环体,否则退出循环转而去执行 ENDFOR 或 NEXT 后面的第 1 条命令。

注意:循环计数器的初值:*nInitialValue*、终值:*nFinalValue*、步长值:*nIncrement* 仅在开始时读入。如果在循环体内改变计数器的值,将会影响到要执行的循环次数。但在 FOR 循环体内改变循环终值却并不影响循环次数。循环次数的计算公式为:

$$n=\mathrm{int}\left(\frac{nFinalValue-nInitialValue}{nIncrement}\right)+1$$

例 8.15 求 1 到 1 000 内的所有自然数的和。

```
* L815.prg
CLEAR
s=0
FOR i=1 TO 1000
s=s+i
ENDFOR
? 's='+STR(s,6)
RETURN
```

2. DO WHILE…ENDDO 结构

已知循环次数,用 FOR…NEXT 语句当然简单,但对于仅知循环结束条件而循环次数不能确定时,用 DO WHILE…ENDDO 语句就比较容易。

(1)语句结构

```
DO WHILE lExpression
    Commands
    [EXIT]
    [LOOP]
ENDDO
```

(2)功能

执行条件循环体内的命令集。

(3)结构说明

·程序执行到 DO WHILE 语句时,先计算 *lExpression* 的值,当值为真时(.T.),执行循环体,否则结束循环,转向执行 ENDDO 紧后面的程序。

·循环体中可插入 LOOP 及 EXIT 语句,其使用方法同 FOR…NEXT。

·循环体中一般存在一条改变循环变量值的语句，使得在某一时刻循环条件不成立（*lExpression* 的值为.F.）时，结束循环。否则应存在一个 EXIT 语句，保证循环能正常结束。不然循环将成为永远进行不完的死循环，这是程序设计的大忌。

例 8.16　使用牛顿迭代法求解一元三次方程：$3\times x^3-15\times x^2+2=0$ 在(0,5)区间内的解。

从高等数学知，牛顿迭代法的迭代公式为：$x_2=x_1-\frac{f(x_1)}{f(x'_1)}$，因此有以下程序：

```
* L816.prg 用牛顿迭代法求解一元三次方程：3 * x^3－15 * x^2＋2＝0
CLEAR
eps＝ 0.000001                    && 收敛精度
x1＝3                             && 估算的初解值
y1＝3 * x1^3－15 * x1^2＋2        && 求出函数在估算的初解 x1 点的值
y2＝9 * x1 * x1－30 * x1          && 求出函数在 x1 点的导数值
x2 ＝ x1－y1/y2                   && 求出函数在 x1 点的切线和 x 轴的新交点 x2
DO WHILE ABS(x2－x1)＞＝eps
    x1＝x2
    y1＝3 * x1^3－15 * x1^2＋2    && 求出函数在新的 x1 点的值
    y2＝9 * x1 * x1－30 * x1      && 求出函数在新的 x1 点的导数值
    x2 ＝ x1－y1/y2               && 求出函数在新的 x1 点的切线和 x 轴的新交点 x2
ENDDO
?"一元三次方程在(0,5)区间内的一个解是"＋STR((x2＋x1)/2,10,5)
RETURN
```

本程序运行结果为：一元三次方程在(0,5)区间内的一个解是－0.35291。

3. 表扫描循环(SCAN…ENDSCAN)

Visual FoxPro 9.0 专门提供了用于数据表记录逐条处理的循环结构“SCAN…ENDSCAN”，使用它不但可以缩短程序代码，节省存储器空间，还可以有效地加快循环执行速度。

(1)语句结构

```
SCAN [Scope] [FOR lExpression1] [WHILE lExpression2] [NOOPTIMIZE]
    [Commands]
    [LOOP]
    [EXIT]
ENDSCAN
```

(2)功能

在当前选定的表中移动记录指针，并且对满足给定条件的每一条记录都执行一个命令块。

(3)参数和子句说明

·*Scope*：要扫描的记录的范围，它可以是范围的四种形式之一。

·FOR *lExpression*1，WHILE *lExpression*2：对记录进行筛选，意义同前。范围和条件均缺省时为全部记录。

·*Commands*：要执行的 Visual FoxPro 命令集。

· LOOP、EXIT：意义同前。

· ENDSCAN：SCAN 循环过程结束语句。

使用 SCAN……ENDSCAN 编程，可以使对表的逐条操作显得非常简练。

例 8.17 用 SCAN 循环显示“xsqkb.dbf”表所有记录的学号、姓名、性别、出生日期等信息。

```
* l817.prg
USE xsqkb
CLEAR
SCAN
? 学号,姓名,性别,出生日期,政治面貌,IIF(应往届生,"应届","往届"),家庭住址
ENDSCAN
USE
RETURN
```

4. 三种循环结构的比较

在 Visual FoxPro 9.0 中提供的三种不同的循环控制结构各有特点。这里作以比较：

① 使用最广泛的是 DO WHILE…ENDDO，其他两种结构均可用 DO WHILE…ENDDO 结构代替。

② FOR…NEXT 结构适用于已知循环次数的循环。

③ SCAN…ENDSCAN 结构仅用作处理数据表记录，不能用于其他循环。

8.3.4 基本结构的嵌套

无论是选择结构还是循环结构，都可以进行嵌套从而组成更复杂的程序结构。所谓嵌套是指在一个大的循环或选择结构内完整地包含了一个或多个循环或选择结构。而且它们可以混合嵌套。单纯的循环嵌套称为多重循环，单纯的选择嵌套称为多重选择。选择与循环的混合嵌套则称为混合嵌套。

不管是哪一种嵌套，都应遵循下面的规则：

① 嵌套只能包含但不得交叉。

② 基本结构的开始与结束语句就近配对。

③ 控制只能由内层转出而不能外层转入。

例 8.18 使用混合嵌套，打印输出 3 到 1 000 内的所有素数，每行打印 20 个。

```
* L18.prg 打印 3 到 1 000 内的所有素数
CLEAR
c= 0                                  && 设置一个素数计数器,并赋初值为 0
?" 3～1000 之间的全部素数" FONT "隶书",24 STYLE "U" AT 24
?
FOR m=3 TO 1000 STEP 2                && 外循环对每一个奇数都判别是否为素数
    n=INT(SQRT(m))
    f=.t.                             && 先假设 m 是素数,将素数标记置为逻辑真
    FOR i=3 TO n                      && 内循环,判别 m 究竟是否为素数
```

```
        IF m%i=0                    && 该 m 不是素数
            f=.f.                   && 将素数标记置为逻辑假
            EXIT                    && 中断判别
        ENDIF
    ENDFOR
    IF f THEN                       && f 为.t.,该 m 是素数,打印输出
        c=c+1                       && 素数计数器加 1
        ?? STR(m,3)+" " FONT "宋体",14
        IF c%20=0 THEN
            ?                       && 每行输出 20 个素数,够 20 个时即换行
        ENDIF
    ENDIF
ENDFOR
RETURN
```

8.4　多模块程序设计技术

前面编写的程序都比较简单。而在实际应用中，一方面程序的结构往往比较复杂，如果将所有的语句都放在同一个程序中，会造成程序太长不便于阅读和维护。另一方面，在设计程序时还经常遇到同一段程序可能会在不同程序中使用，或在同一个程序的不同地方使用多次。如果多次重复书写，势必加大程序代码的编写工作量，浪费大量的人力和时间。

解决这类问题的有效方法，是将上述重复使用的代码段或功能独立的代码段，设计成能被其他程序调用的独立程序段，这种程序段称为功能模块或过程。过程按其定义的方式分为子程序过程和函数过程。

功能模块存在着调用和被调用的逻辑关系。在程序设计中，把被其他模块调用的模块称为被调模块，相对地把调用其他模块的模块称为主调模块。

实际上除主模块外，主调模块和被调模块的概念是相对的。对于一个功能模块，相对于上级主调模块它是被调模块，而对于被它所调用的下级模块而言它又是主调模块。Visual FoxPro 允许模块的嵌套调用，最深可以嵌套调用 128 层。模块间的调用关系可用图 8－10 示意

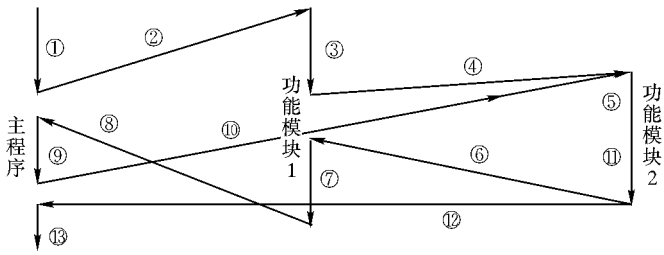


图 8－10　功能模块调用示意图

图来表示。

在程序中,存在着一个特殊的模块,它只调用其他模块而不被其他任何模块所调用,这个特殊的模块称为主程序。主程序在整个程序中处于主导地位,是整个程序的入口程序,程序的运行将从它开始。其他的模块均称为子模块。

8.4.1 过程的类型

根据过程的存储位置,把过程分为四种不同的类型:独立程序文件过程、程序文件过程、过程文件过程、存储过程。

1. 独立程序文件

此类过程和普通的程序文件完全相同,是一个个独立的.prg 文件。创建的方法、调用的方法也相同。它的缺点是当程序运行时,需要反复地访问硬盘,其运行速度较慢,现在一般很少使用。

2. 程序文件过程

此类过程与主程序同文件,写在主程序之后。每个过程都有过程开始语句:PROCEDURE|FUNCTION;过程结束语句:ENDPTROC|ENDFUNC。当程序文件打开时,这些过程就全部自动打开,当程序运行结束时,它们都将自动关闭,因而运行速度高。但过多的此类过程会使程序文件显得过分冗长。

3. 过程文件过程

此类过程写在一个独立的过程文件之中。过程文件是一种特殊的程序文件,它的扩展名也是.prg,但它的内容则是由一系列的带有过程开始与结束语句的过程组成。Visual FoxPro 允许一个过程文件最多包含 32 767 个过程。过程文件必须在使用前通过专门的打开语句"SET PROCEDURE TO"打开。由于打开过程文件时,其中所包含的所有过程一次全部被调入内存,不存在反复到磁盘上查找各过程的问题,因而运行速度高。

4. 存储过程

此类过程是一种特殊的程序文件,该文件并不是一个独立的程序文件,而是存放在数据库中的若干过程的集合。它有如下四个特点:

① 当编写完当前过程后保存并退出编辑窗口时,Visual FoxPro 就立即编译该过程,因此万一源代码有错误会被立即发现、更正,从而避免了在执行时才发现错误。

② 运行时无需编译因而速度快。

③ 只要数据库打开就自动打开。

④ 由于存储过程是存储在数据库中的过程,因而对于涉及到数据库表的各种操作,十分方便。用户不必担心将数据库移动到了另外的目录下而找不到应用程序。

在 Visual FoxPro 9.0 中大多使用存储过程。例如当创建了数据库的参照完整性后,系统就会自动向数据库中添加若干个存储过程。

8.4.2 过程的定义

这里讲的过程的定义是指对除独立程序文件以外的三种过程(程序文件过程、过程文件过程、存储过程)的定义方法,它们的定义方法是相同的。

1. 过程的定义

(1)**命令格式** 1

PROCEDURE|FUNCTION *ProcedureName*

　　[LPARAMETERS *parameter*1 [,*parameter*2] ,...]

　　　　Commands

　　[RETURN [*eExpression*]]

[ENDPROC|ENDFUNC]

(2)**命令格式** 2

PROCEDURE |FUNCTION *ProcedureName*([*parameter*1;

　　[AS *para*1*type*][,*parameter*2 [AS *para*2*type*]] ,...]) [AS *returntype*]

　　　　Commands

　　[RETURN [*eExpression*]]

[ENDPROC|ENDFUNC]

(3)**功能**

定义用户自定义过程。其中用 FUNCTION **定义的过程又称为用户自定义函数。**

(4)**参数和子句说明**

· PROCEDURE|FUNCTION *ProcedureName* :**指定用户自定义过程的开头处和过程的名字。**

· LPARAMETERS *parameter*1 [,*parameter*2] ,... :**从主调模块传递数据到子程序过程的局部变量或数组。也可用** PARAMETERS **代替** LPARAMETERS,**此时会从主调模块传递数据到定义模块的私有变量或数组。这些变量参数称为形参。最大允许的形参个数是** 26 **个。**

· ([*parameter*1[AS *para*1*type*][, *parameter*2[AS *para*2*type*]],...]):**从主调用模块传递数据到函数过程的局部变量或数组。可以使用** AS paraltype **子句说明变量的数据类型。**

· AS *returntype*:**定义返回值的数据类型。**

· *Commands* :**过程体内** Visual FoxPro **命令序列。**

· RETURN [*eExpression*]:**返回控制到主调模块或其他模块。***eExpression* **可以指定一个返回值。**

注意　RETURN **可以出现在过程体的任何地方,以便将控制返回到主调模块或别的模块以及返回一个值。如果过程中没有** RETURN **命令,**Visual FoxPro **当过程退出时会自动隐含执行一个** RETURN。**如果** RETURN **命令不包含返回值或者隐含执行** RETURN,Visual FoxPro **将把逻辑真.**T. **返回。**

· ENDPROC|ENDFUNC :**表示过程结构的结束。该关键字是可选择的,无此关键字时,当过程碰到其他的** PROCEDURE **或** FUNCTION **命令或程序的结尾时,该过程也会自动退出。**

例 8.19　定义一个求阶乘的功能模块。

PROCEDURE factor

LPARAMETERS m

```
f=1
FOR i=1 to m
f=f*i
ENDFOR
RETURN f
ENDPROC
```

或

```
FUNCTION factor(m)
f=1
FOR i=1 to m
f=f*i
ENDFOR
RETURN f
ENDFUNC
```

2. 存储过程的创建

对于过程的创建,如前所述,独立程序文件、包含在程序文件中的过程、过程文件过程,就其本质而言都是扩展名为.prg 的程序文件,因此他们都可以用 MODIFY COMMAND **命令来创建,无需赘述。然而存储过程是存储在数据库中的过程,因此创建的方法与前者不同。下面专门介绍存储过程的创建。**

(1)**命令格式**

MODIFY PROCEDURE [NOWAIT]

(2)**功能**

打开 Visual FoxPro **文本编辑器,为当前的数据库创建新的存储过程,或者编辑当前数据库中已经存在的存储过程。**

(3)**参数和子句说明**

NOWAIT:**意义同前。**

例 8.20　为数据库"xjsjk.dbc"**创建一个存储过程,用来逐条统计每个学生的成绩总分。**

S1:OPEN DATABASE xjsjk

S2:MODIFY PROCEDURE

S3:**在文本编辑器中写入如下内容:**

```
* 该存储过程的功能是逐条计算各条记录的总分字段的值
* 过程中使用了 FIELD()函数,其作用是返回字段的名称字符串
PROCEDURE zongfen
CLEAR
USE xscjb
UPDATE xscjb SET 总分 = 0
GO TOP
SCAN
    s=0                          && 对于每条记录,设成绩的总合初值为 0
```

```
    FOR i=2 TO 6                    && 五门功课,字段编号分别是 2,3,4,5,6
        zdm=FIELD(i)                && 提取各门课程的字段名
        s=s+&zdm
    NEXT
    REPLACE 总分 WITH s
ENDSCAN
USE
ENDPROC
```

S4:存盘,则存储过程"zongfen"创建完成。

3. 追加存储过程

有时,用户希望将已编写好的过程追加到数据库中使其成为存储过程。另一种情况是采用上面的 MODIFY PROCEDURE 打开文本编辑器书写存储过程时,如数据库中事先已存在了若干个存储过程,则它们将全部被打开显示在屏幕上,供人们浏览、修改,这样会容易引起原有的存储过程被破坏。因此人们常用的方法是先编写一个程序文件,待调试正确后,再为之加上过程的开始和结束语句,然后把它追加到当前数据库中,使之成为存储过程。追加存储过程的命令如下:

(1)命令格式

APPEND PROCEDURES FROM *FileName* [AS *nCodePage*] [OVERWRITE]

(2)功能

把文本文件中的一个过程追加成当前数据库的存储过程。

(3)参数和子句说明

· *FileName*:指明存储过程所在的文本文件的名字。

· AS *nCodePage*:指定存储过程所在的文本文件的代码页。如果代码页值为 0,则假定文本文件的代码页与当前数据库的代码页相同,不进行代码页的转换。

· OVERWRITE:指定数据库中的当前存储过程将被文本文件中同名存储过程所覆盖。如果缺省 OVERWRITE 关键字,数据库中的当前存储过程将不被覆盖,而文本文件中的存储过程将被追加到当前的存储过程中。

例 8.21　为数据库"xjsjk.dbc"创建一个文本文件存储过程 paimingci,用来对 xscjb.dbf 各条记录根据总分由高到低进行排名次,要求总分相同者名次并列。然后将它追加到数据库中,使其成为存储过程。

S1:OPEN DATABASE xjsjk EXCLUSIVE

S2:MODIFY COMMMAND paimingci

S3:在程序文本编辑器中输入下面的内容:

```
* 本过程逐条记录根据总分排列学生的名次,总分相同者名次并列
* PROCEDURE paimingci
CLEAR
USE xscjb EXCLUSIVE
INDEX ON 总分 TAG 总分 DESCENDING
SET ORDER TO TAG 总分
```

```
GO TOP
zf=总分                          && 将第 1 条记录的总分字段值赋给内存变量:zf
mc=1                             && 假设第 1 条记录的名次先为 1
SCAN                             && 逐条排名次
    IF zf>总分                   && 总分不同时,名次变量+1
        zf=总分
        mc=mc+1
    ENDIF
    REPLACE 名次 WITH mc
ENDSCAN
SELECT * FROM xscjb ORDER BY 名次
USE
RETURN
* ENDPROC
```

S4:运行程序,调试与测试其是否正确运行,待调试通过且测试正确后去掉 PROCEDURE 和 ENDPROC 两句的注释符,存盘。

S5:将程序文本文件追加为数据库的存储过程:

```
APPEND PROCEDURE paimingci.prg
```

至此,paimingci.prg 已由文本存储过程被追加为数据库"xjsjk.dbc"的一个存储过程。打开项目管理器的数据卡,会看到在存储过程中已有了存储过程"paimingci",如图 8-11 所示。

图 8-11 存储过程显示结果

4. 复制存储过程

复制存储过程是追加存储过程的逆操作。

(1)命令格式

```
COPY PROCEDURES TO FileName [AS nCodePage] [ADDITIVE]
```

(2)功能

将当前数据库中的存储过程复制成文本文件。

(3)参数和子句说明

• *FileName*:要复制存储过程的文本文件的名字。如果该文本文件不存在,Visual FoxPro 将自动生成它。如果文本文件缺省扩展名,则系统会自动添加上扩展名:.txt。

• AS *nCodePage*:意义同前。

• ADDITIVE:将存储过程追加到指定文本文件的尾部。如果缺省 ADDITIVE,存储过程将取代原文本文件的内容。

8.4.3　过程的打开

如前所述,存储过程是与数据库同时打开的,独立程序文件是在该文件被调用时直接调入的,程序文件过程是在主程序被执行时就一起调入内存中的,只有过程文件过程必须首先将过程文件打开,方可被调用。

(1)命令格式

SET PROCEDURE TO [*FileName*1 [, *FileName*2,...]] [ADDITIVE]

(2)功能

打开过程文件。

(3)参数说明

• *FileName*1 [, *FileName*2,...]:要打开的过程文件的名字列表;

• ADDITIVE:在不关闭当前打开的过程文件的情况下,另外再打开别的过程文件。缺省时将原打开的过程文件关闭,再打开指定的过程文件。

(4)注意

① 当参数全部缺省时,表示关闭所有的过程文件。要关闭个别过程文件,可使用:

RELEASE PROCEDURE *FileName*1 [, *FileName*2...]

来完成。

② 用 SET PROCEDURE TO 命令打开过程,该命令应出现在过程文件中的任何一个过程被第 1 次调用之前,一般将该命令写在主程序的开始的第 1 个板块——初始化板块中。

(5)过程的查找顺序

当程序调用一个过程时,Visual FoxPro 将按下面的顺序来查找该过程:

① 查找包含在本程序文件中的过程。

② 查找当前数据库中的存储过程。

③ 查找用 SET PROCEDURE TO 命令打开的过程文件中的过程。

④ 执行链中的程序文件。Visual FoxPro 将按照从最近执行的程序开始往回返到第 1 个执行的程序的次序,来查找程序文件。

⑤ 一个独立的程序文件。

如果找到了与调用命令相匹配的过程,Visual FoxPro 将执行该过程,否则将给出一个出错信息。

8.4.4　过程的调用

根据在过程调用中,是否有参数的传递,将过程调用分为带参数调用和不带参数调用;根据是否用 DO 命令,又可把过程调用分为 DO 调用和函数调用。

1. 不带参数的过程调用

不带参数的过程调用格式如下：

(1)**不带参数的** DO **过程调用**

① **命令格式**

DO *ProgramName*1 | *ProcedureName* [IN *ProgramName*2]

② **功能**

执行 Visual FoxPro **的程序或过程。**

③ **参数说明**

· *ProgramName*1：**要执行的独立程序名。如果不给出要执行的程序的扩展名，**Visual FoxPro **将按照：可执行程序**(.exe)、**应用程序**(.app)、**已编译程序**(.fxp)、**源程序**(.prg)**的次序查找并执行程序。**

注意　要执行一个指定的菜单程序、表单程序、查询，则必须给出它们各自的扩展名：.mpr,.scx,.qpr。

· *ProcedureName*：**要执行的过程名。**

· IN *ProgramName*2：**指定要执行的过程所在的程序文件名。当文件被找到时，过程将被执行。如果程序文件没有找到，信息“文件不存在”将出现。如果程序文件找到了但却并不包含要执行的过程，将显示“过程未找到”。缺省时为已打开的过程文件的过程、存储过程或程序文件过程。**

注意　不管过程定义时，使用的是 PROCEDURE **还是** FUNCTION **语句，都可以使用** DO **调用来完成。**

(2)**不带参数的函数过程调用**

① **命令格式**

*ProgramName*1 | *ProcedureName* [IN *ProgramName*2]( )

② **功能**

以函数调用的形式，实现对程序文件或过程的调用。

③ **参数说明**

其中的参数与 DO **调用意义相同。**

注意　以函数形式进行过程调用时，后面的一对圆括号不可缺少。

例 8.22　编写用无参过程调用的格式求两个数的排列与组合的程序，要求用程序文件过程的形式编写。

```
* L822.prg 求两个数的排列与组合的主程序
CLEAR
p=1                                    && 为排列的结果初值赋 1
zh=0                                   && 为组合的结果初值赋 0
j=1                                    && 为阶乘的结果初值赋 1
INPUT "请输入两个整数中较大的数：" TO m
INPUT "请输入两个整数中较小的数：" TO n
DO pailie                              && 调用求排列子程序
  @18,5 SAY STR(m,2)+"和"+STR(n,2)+"的排列是："+;
```

```
    ALLTRIM(STR(p,10)) FONT "宋体",20 COLOR r+/B+
DO zuhe                                    && 调用求组合子程序
  @ROW()+1,5 SAY STR(m,2)+"和"+STR(n,2)+"的组合是:"+;
    ALLTRIM(STR(zh,10)) FONT "宋体",20 COLOR r+/B+
RETURN                                     && 主程序结束,下面是过程
* 过程 1——求排列
PROCEDURE pailie                           && 求排列的子程序开始
  p=1
  FOR i=m TO n STEP -1
    p=p*i
  ENDFOR
ENDPROC                                    && 求排列子程序结束
*
* 过程 2——求组合
FUNCTION zuhe                              && 求两个数组合的子程序开始
  x=m
    jiecheng()
    jm=j
  x=n
    jiecheng()
    jn=j
  x=m-n
    jiecheng()
    jmn=j
  zh=jm/jn/jmn
ENDFUNC                                    && 求两个数组合的子程序结束
*
* 过程 3——求阶乘
PROCEDURE jiecheng                         && 求一个数阶乘的子程序
  j=1
  FOR i=1 TO x
    j=j*i
  ENDFOR
RETURN                                     && 求一个数阶乘的子程序结束
```

2. 带参数的过程调用

(1)参数传递的两种方式

带参数的过程调用中,主调模块中要传递给被调模块的参数——实际存在的参数(简称实参)与被调模块的接收参数(简称形参)之间存在着参数传递的问题。参数传递的原则是:形参与实参一一对应。

传递分为两种方式，分别是地址传递方式和值传递方式。

所谓地址传递是指在实参与形参的参数传递中，将实参自身的存储地址传递给形参。这样它们就使用了同一个存储地址，使得在被调模块中，形参值的任何改变都会反馈给主调模块的对应实参，从而实现参数的双向传递——可进可出。地址传递又简称为传址，也称为引用传递。

所谓值传递是指在实参与形参的参数传递中，系统首先为形参分配一个临时存储单元，然后将对应的实参的值复制到该临时单元中，供形参使用。这样的传递，形参和实参并不使用同一个存储地址，因而使得在被调模块中，形参值的任何改变都不会反馈给主调模块的对应实参，从而实现参数的单向传递——只进不出。

(2)带参数的过程调用格式

带参数调用也分为 DO 调用和函数调用两种格式，它们分别是：

① 格式 1

DO *ProgramName1* | *ProcedureName* [IN *ProgramName2*] WITH *ParameterList*

② 格式 2

ProgramName1 | *ProcedureName* [IN *ProgramName2*](*ParameterList*)

③ 参数说明

· *ParameterList*：实参列表，指定要传递到程序或过程中的参数。它可以是常量、表达式、内存变量、字段、数组名或用户自定的函数。参数的最大数目为 26 个。

· 在 DO 形式的过程调用中，如无特殊说明，除表达式和常量参数传递按传值方式进行外，其余参数传递均按传址方式传递。此时实参如果是数组则将数组的首地址传递给形参，形参变量将变成一个和实参数组相同的数组。用户可以通过设置圆括号的方法，将传址传递改为传值传递。

· 在函数形式的过程调用中，如无特殊说明，参数传递一律按传值方式进行。要改变变量参数的传递方式，可以通过 SET UDFPARMS 命令或变量前加“@”前缀的非法方式予以设置。

(3)函数调用格式中参数传递方式的设置

① 设置命令格式

SET UDFPARMS TO VALUE | REFERENCE

② 功能

在过程的函数方式中，指定参数传递按传值或传地址的方式进行。

③ 参数和子句说明

· TO VALUE：传值。这是缺省的。

· TO REFERENCE：传址。

注意　不管 SET UDFPARMS 怎么设置，用户都可以强制性规定变量按传值还是按传址方式进行参数传递。设置的方法是：在变量前加上“@”可以使其变为传址；将变量用一对圆括号括起来，此时就成了表达式，而表达式在任何情况下都仅传值而不传址。

例 8.23　对于一个有 10 个数值的数组，按冒泡排序法从小到大将各个元素排序。

```
* L823.prg 数组参数的传递，程序用来实现数据的冒泡排序。
CLEAR
```

```
n=10
DIMENSION s(n)
* 随机生成 10 个 10～100 之间的整数
FOR i=1 TO n
s(i)=INT(RAND() * 90+10)
NEXT i
? "排序前原始数组各元素的值"
szysdy(@s,n)
DO bubble WITH s , (n)
? "排序后数组各元素的值"
szysdy(@s,n)
RETURN
* bubble 过程
FUNCTION bubble
LPARAMETERS x,m
FOR i=1 TO m-1
    FOR j=m TO i+1 STEP -1
        IF x(j-1)>x(j) THEN
            t=x(j)
            x(j)=x(j-1)
            x(j-1)=t
            ENDIF
NEXT j
NEXT i
ENDFUNC
PROCEDURE szysdy(x,m)
  ?
  FOR i=1 TO m
    ? ? STR(x(i),4)+SPACE(4)
  NEXT i
  ?
ENDPROC
```

8.5　内存变量的作用域

在多模块程序中，每一个内存变量都有自己的有效范围，通常称之为作用域。内存变量按作用范围来分为公有变量、私有变量、局部变量三类。

8.5.1 公有变量

Visual FoxPro 9.0 规定，凡是在程序中用 PUBLIC 命令声明的变量或在命令窗口中定义的变量均为公有内存变量。它也被称为全局内存变量、全程内存变量等。

公有内存变量是内存变量中作用域最广的一种变量。它作用于本次 Visual FoxPro 运行期间。即使是程序运行结束，公有内存变量依然有效。

(1)命令格式

PUBLIC *MemVarList*

或

PUBLIC [ARRAY] *ArrayName1*(*nRows1* [, *nColumns1*]);

[, *ArrayName2*(*nRows2* [, *nColumns2*])]...[AS *Type*]

(2)功能

声明公有内存变量或数组，并对其进行初始化。

(3)参数和子句说明

· *MemVarList*：要声明为公有内存变量的变量名列表。注意单个字符 A～J，M 是保留字，不能用来作变量名。

· *ArrayName1*(*nRows1* [, *nColumns1*]) [, *ArrayName2*(*nRows2* [, *nColumns2*])]：定义数组并同时将其声明为公有内存变量数组。

· AS *Type*：定义变量和数组的数据类型。

(4)注意事项

① 在当前 Visual FoxPro 运行期间，用户在任何一个正在执行的过程中使用或修改公有变量或公有数组，都会影响到整个程序。

② 用 PUBLIC 语句声明的变量除变量 FOX，FOXPRO 被初始化为逻辑真(.T.)外，其余的一律被初始化为逻辑假(.F.)。

③ 公有内存变量必须先声明而后使用，否则将出错。

④ 在命令窗口生成的变量或数组，自动是公有的。

例 8.24 声明公有内存变量实例。

```
* L824.prg
CLEAR MEMORY
PUBLIC pc,pb(3,4),fox
DISPLAY MEMORY
RETURN
```

运行结果如图 8－12 所示。

图 8－12 公有内存变量的声明

8.5.2 私有变量

Visual FoxPro 9.0 规定，一个变量如果没有用 PUBLIC 或 LOCAL 命令声明，也没有在命令窗口下定义，而是在程序中直接被定义使用，则该变量为私有变量。

注意 对变量的声明和变量定义是两个不同的概念。变量声明指为变量在内存中申请一

个存储单元，如同旅客事先在酒店中登记了一个房间。变量定义指为变量赋值，该值是可以不断变化的，如同一个房间，所住的旅客却随时可改变一样。在其他的高级程序设计语言中，一般都要求变量必须先声明而后定义。在 Visual FoxPro 中，对有些变量可声明定义同时进行，例如在命令窗口中的为变量直接赋值，在程序中的私有变量。但大部分变量，如公有、局部、数组则必须先声明而后定义。

私有变量的作用域是定义它的模块及其被该模块所调用的各下级模块。当定义它的模块结束运行后，在这个模块中定义的所有私有变量会自动清除。

8.5.3　局部变量

局部变量在程序中由 LOCAL 命令声明的变量。局部变量的作用域仅限于本模块。其声明的语法格式与 PUBLIC 相类似。

(1)命令格式

LOCAL *Var*1 [AS *type*] | *ArrayName*1(*nRows*1 [, *nColumns*1]) [AS *type*]
[, *Var*2 [AS *type*] | [, *ArrayName*2(*nRows*2 [, *nColumns*2]) [AS *type*]]

或

LOCAL [ARRAY] *ArrayName*1(*nRows*1 [, *nColumns*1]) [AS *type*]
[, *ArrayName*2(*nRows*2 [, *nColumns*2]) [AS *type*]]

(2)功能

声明局部变量和变量数组，并对其进行初始化。

(3)参数和子句说明

· *Var*1，*Var*2，…：声明一个或多个局部变量。单字母 A～J，M 由于是保留字，不得用来作变量名。

· *ArrayName*1(*nRows*1 [, *nColumns*1])，*ArrayName*2(*nRows*2 [, *nColumns*2])：要声明为局部数组的数组名和它的界。

· AS *type*：指定变量或数组的数据类型。

8.5.4　变量的隐藏

公有变量的作用域是整个的 Viusal FoxPro 运行期间，私有变量的作用域是声明它的模块及其被调用的下级模块。有时会碰到这样一个问题，即在下级模块中有某些变量和上级模块中的变量尽管同名，但人们却不想让这些同名变量间发生数据传递，达到尽管上下模块中变量同名却并不是同一变量的目的，这就是变量隐藏。变量的隐藏也可以称为变量的屏蔽。它通过 PRIVATE 命令来实现。

(1)命令格式

PRIVATE *VarList*

PRIVATE ALL [LIKE *Skeleton* | EXCEPT *Skeleton*]

(2)功能

隐藏当前运行程序中在主调程序模块定义了的变量或数组。

(3)参数和子句说明

· *VarList*：要隐藏的变量或数组名列表。

• ALL:隐藏所有的变量和数组。

• LIKE *Skeleton*| EXCEPT *Skeleton* :意义同前,*Skeleton* 中可含通配符“*”,“?”。

(4)注意

① PRIVATE 语句仅实现对已有变量的隐藏,并不能用来声明变量为私有变量,尽管用它声明某个变量为私有变量时系统并未给出错误信息,因为私有变量并不需要声明而可直接定义。

例 8.25 演示 PRIVATE 无变量声明功能。

```
* L825.prg 演示 PRIVATE 无变量声明功能
s=0
? s
Yanshi()
? s
RETURN
FUNCTION yanshi
    PRIVATE c
    ? c
ENDFUNC
```

运行该程序,会给出如图 8-13 所示的出错信息。

图 8-13 PRIVATE 无变量声明功能的演示

而如果它有声明私有变量的功能,则根据对变量既声明又初始化的原则,程序中“? c”语句将显示.F. ,可见它无声明功能。

② 隐藏仅对上级主调模块隐藏,对下级被调模块则透明。

③ 当被调模块执行时,主调模块中的同名变量的值不能传给被调模块中那些隐藏了的变量。当被调模块执行结束将控制交给主调模块时,主调模块中的同名变量的值保持被调模块执行前的原值不变。

例 8.26 隐藏变量实例演示。

```
* L826.prg 演示 PRIVATE
SET TALK OFF
CLEAR MEMORY
CLEAR
```

```
val1 = 10
val2 = 15
DO down
? val1, val2                          && 10 100
RETURN
PROCEDURE down
PRIVATE val1
val1 = 50
val2 = 100
? ' Val1 Val2'
? val1, val2                          && 50 100
RETURN
```

例 8.27　变量作用域实例演示。

```
* L827 本程序用来演示各种类型的变量的作用域及变量的屏蔽功能
* 主程序
CLEAR
CLEAR MEMORY
PUBLIC aa,bb,fox
STORE 5 TO aa,bb
? "调用过程前 aa,bb,fox 的值:"
? aa,bb,fox                           && 5 5 .t.
DO sub1
? "调用过程后,aa,bb,fox,gg 的值:"
? aa,bb,fox,gg                        && 15 5 .F. 15
RETURN
* 主程序结束,下面是过程 sub1
PROCEDURE sub1
    PRIVATE bb                        && 变量 bb 被对上屏蔽,但对下开放
    STORE 10 TO aa,bb,cc
    ? "在过程 sub1 中,aa,bb,cc,fox,gg 的值:"
? aa,bb,cc,fox                        && 10 10 10 .t.
    sub2()
    ? "调用过程 sub2 后,aa,bb,cc,fox,gg 的值:"
    ? aa,bb,cc,fox,gg                 && 15 15 15 .F. 15
RETURN
ENDPROC
* 下面是过程 sub2
FUNCTION sub2
    PUBLIC g
```

```
    ?"进入 sub2 时,aa,bb,cc,fox,gg 的值:"
    ? aa,bb,cc,fox,gg                    && 10 10 10 .T. .F.
    FOX=.F.
    STORE 15 TO aa,bb,cc,gg
ENDFUNC
```

8.6 预处理语句

在 Visual FoxPro 9.0 中,对于源程序可加入一些预处理命令,用来改进程序的设计环境,提高编程效率。

由于预处理命令并非 Visual FoxPro 本身的命令,因此编译程序并不识别它。故而在对源程序进行通常的编译之前,必须先对源程序中的这些特殊的预处理命令进行预先的处理。即根据预处理命令首先对源程序进行相应的转换,使其转为合法的 Visual FoxPro 9.0 的源程序而再也不含预处理命令。然后再由编译程序对合法的源程序进行编译,生成目标代码。

Visual FoxPro 9.0 中预处理命令前面都带有一个字符"#"。它提供的预处理命令主要有以下三种:

① 常量的定义和释放命令:#DEFINE,#UNDEF

② 文件包含命令:#INCLUDE

③ 条件编译命令:#IF…#ENDIF

8.6.1 头文件

Visual FoxPro 9.0 的头文件一般都具有扩展名.h,但也可以带其他扩展名。头文件中包含了许多预编译命令供用户使用,其中#INCLUDE 命令使用最广泛。

(1)命令格式

#INCLUDE *FileName*

(2)功能

指定 Visual FoxPro 预处理器要处理指定的头文件的内容。

(3)参数说明

FileName:指定在编译期间要插入程序的头文件的名字,它可带有头文件的路径。当在头文件中含有路径时,Visual FoxPro 仅在规定的区域内查找头文件。如果在头文件中没有包含路径,Visual FoxPro 将首先在缺省的目录内查找头文件,然后再按用 SET PATH 目录所设置的路径进行查找。

(4)注意事项

① 用户可以自己生成一个包含预处理命令的头文件,然后当编译程序时,利用#INCLUDE 将头文件的内容插入程序中。编译时,头文件的内容将被插到#INCLUDE 命令出现的程序位置。

② 仅#DEFINE…#UNDEF,#IF…#ENDIF 和#INCLUDE 预处理命令可以出现在头文件中。注释语句和 Visual FoxPro 命令如包含在头文件中,将被忽略。

③ 一个程序可以包含任意数目的#INCLUDE 命令。这些命令可以出现在程序中的任

何地方。在头文件中写上 #INCLUDE 命令，可以实现 #INCLUDE 命令的嵌套。

④ 尽管头文件可以拥有任意的扩展名，但是最典型的扩展名是 .h。系统提供了有一个 Visual FoxPro 的头文件 FoxPro.h，它含有整个本文档所描述的许多常量。

例 8.28　#INCLUDE 命令示例。

```
* L828.prg 头文件的使用演示
* * * 头文件 CONS T.H * * *
#DEFINE ERROR_NODISK 1
#DEFINE ERROR_DISKFULL 2
#DEFINE ERROR_UNKNOWN 3
* * * 程序文件 myprog.prg * * *
#INCLUDE CONS T.H
FUNCTION chkerror
PARAMETER errcode
  DO CASE
  CASE errcode = ERROR_NODISK
    ? "Error - No Disk"
  CASE errcode = ERROR_DISKFULL
    ? "Error - Disk Full"
  CASE errcode = ERROR_UNKNOWN
    ? "Unknown Error"
  ENDCASE
RETURN
```

8.6.2　条件编译

(1)命令格式

```
#IF nExpression1 | lExpression1
    Commands
[#ELIF nExpression2 | #ELIF lExpression2
    Commands...
#ELIF nExpressionN | #ELIF lExpressionN
    Commands]
[#ELSE
    Commands]
#ENDIF
```

(2)功能

在编译时有条件地包含源代码。

(3)参数和子句说明

nExpressionl | *lExpressionl*：*nExpression*1(l=1,2,3,…*n*)是一个数值表达式，*lExpressionl* (l=1,2,3,…,*n*)是一个逻辑表达式。数值表达式不为 0 或逻辑表达式为.T. ，均表示条

件编译条件为真,否则表示条件编译条件为假。其作用和分支语句类似。

例 8.29 使用条件编译从下面的程序中编译一个求 1 到 10 的自然数的和与 10 的阶乘的程序代码。

```
*l829.prg 条件编译
#DEFINE COND 1
#IF COND
  c="1 到 10 间自然数的和:"
  s=0
  FOR i=1 TO 10
    s=s+i
  ENDFOR
#ELSE
  c="10 的阶乘:"
  s=1
  FOR i= 1 TO 10
    s=s*i
  ENDFOR
#ENDIF
? c+LTRIM(STR(s))
RETURN
```

本程序编译后,将得到一个求 1 到 10 的自然数和的程序,运行后将得到结果 55。如果将 #DEFINE COND 中的 1 改为 0,则编译后得到求 10 的阶乘的程序,运行后结果为:3 628 800。

8.7 思考与练习

一、选择题

1. 在 Visual FoxPro 中,用于建立或修改过程文件的命令是(　　)。

A) MODIFY *FileName*

B) MODIFY COMMAND *FileName*

C) MODIFY PROCEDURE *FileName*

D) 上面 B)和 C)都对

2. 在 DO WHILE…ENDDO 循环结构中,LOOP 命令的作用是(　　)。

A) 退出过程,返回程序开始处

B) 转移到 DO WHILE 语句行,开始下一个判断和循环

C) 终止循环,将控制转移到本循环结构 ENDDO 后面的第一条语句继续执行

D) 终止程序执行

3. 过程的入口语句是(　　)。

A) DO 过程名　　B) DO CASE

C) PROCEDURE 过程名　　D) RETURN

4. 下面关于过程调用的陈述中,哪个是正确的?

A) 实参与形参的数量必须相等

B) 当实参的数量多于形参的数量时,多余的实参被忽略

C) 当形参的数量多于实参的数量时,多余的形参取逻辑假

D) 上面的 B)和 C)都对

5. 若一个过程不包含 RETURN 语句,或 RETURN 语句中没有指定表达式,那么该过程(　　)。

A) 没有返回值　　B) 返回 0

C) 返回.T.　　D) 返回.F.

6. 在 Visual FoxPro 中,如果希望一个内存变量只限于在本过程及下级过程中使用,说明这种内存变量的命令是(　　)。

A) PRIVATE

B) LOCAL

C) PUBLIC

D) 在程序中直接使用的内存变量(不通过 A),B),C)说明)

7. 有如下程序:

```
INPUT TO A
IF A=10
S=0
ENDIF
S=1
? S
```

问:假定从键盘输入的 A 的值一定是数值型,那么上面程序的执行结果是(　　)。

A) 0　　B) 1

C) 由 A 的值决定　　D) 程序出错

二、上机编程题

1. 编制一个通用的交换记录的程序,即要求对换表中任意两个记录的内容进行交换。

2. 输入一个字符串,要求分别统计出其中英文字母、空格、数字和其他字符的个数。

3. 已知:表成绩.DBF 含有学号、平时、考试、等级,共四个字段,前三个字段已存有某班学生的数据,平时成绩、考试成绩均填入了百分制数。请以平时成绩 20%、考试成绩 80%的比例确定等级并填入等级字段。等级评定办法是:90 分以上为优,75～89 为良,60～74 为及格,60 分以下不及格。要求用条件、步长、扫描循环三种循环语句分别来编写程序。

4. 已知汉字在计算机中的机内码以是两个字节存储一个汉字,机内码和外码中的区位码有着简单得对应关系:

高字节＝区码＋160

低字节＝位码＋160

在区位码中,汉字按一个 94 行 94 列的矩阵进行编码,行代表区码,列代表位码。区位码

分为两个大区。

第一大区:1 到 15 区,用来存放图形字符。其中 1 到 9 区存放通用的标准图形字符,例如英、俄、日、希腊等文字的字母、各种数学符号、运算符、制表符等;10 到 15 区存放自定义图形字符。

第二大区:16 区到 94 区,用来存放汉字。其中 16 到 55 区共 40 个区存放一级汉字,此处用汉语拼音编码,存 3 755 个常用汉字(其中第 55 区的最后 5 位没有汉字);56 区到 87 区存放二级汉字,此处用偏旁部首笔画编码,存 3 008 个非常用汉字和汉字的偏旁部首笔画等;88 区到 94 区为自定义汉字区。

编程输出 16 到 87 取得全部汉字。要求:

(1)每个区必须另换一行输出。

(2)每行输出 32 个汉字,每个汉字间有一个空格。

5. 求 1～100 之间能被 7 整除的偶数个数及它们之和。

6. 编写主程序通过参数调用子程序的程序。

第 9 章　面向对象程序设计基础

C,PASCAL,FORTRAN 等高级语言是结构化的程序设计语言,它们是面向过程的。当初提出结构化程序设计方法的目的是为了更好地解决软件危机,但是这个目标并未完全实现。而实际的生产生活中,要解决的问题愈来愈复杂、规模愈来愈大,结构化程序设计方法需要书写的代码愈来愈庞杂,面向过程的语言显得力不从心。为了更好地解决软件危机,在 20 世纪 80 年代提出了面向对象的程序设计(Object-Oriented Programming,简称 OOP)。

面向对象是一种解决问题的思维方式,它将观察焦点放在构成客观世界的成分——对象上,将对象作为需求分析和系统设计的主体,将对象间有意义的相互作用来通信,即把整个问题集合抽象为相互通信着的一组对象集合。将相似或相近的一组对象聚合为类,采用各种手段将相似的类组织起来,实现问题空间到解空间的映射。这种方法描述的现实世界模型贴切、合理,更符合人们认识世界的思维方法。本章围绕面向对象技术的基本概念和运行机制,结合实例,介绍面向对象程序设计的基本知识。

9.1　类和对象的基本概念

Visual FoxPro 9.0 既支持面向过程的程序设计,同时还支持面向对象的程序设计方法。在面向对象的程序设计中,有一系列的新概念。如类(Class)和对象(Object)等,它们既是本章的重点,同时也是一个全新的概念。

9.1.1　对象的属性与特征

对象是对客观存在的一个实体(Entity)属性及行为特征的描述。每个对象都有描述其特征的属性及其附属于它的行为。在面向对象程序设计中,对象是基本运行的实体,每一个对象都具有如下的一些基本特征。

1. 对象的名称(name)

与每个内存变量都应有一个名字一样,每一个实体对象,也都有一个名字,用以区别其他的对象,并在引用该对象时使用。

2. 对象的属性(property)

对象的属性标识了对象的物理性质,是区别不同对象的重要标志。例如:每个人都有自己的姓名、性别、出生年月、身高、体重、视力、听力、指纹、脚码、血型、脸型、体型、五官四肢、年龄、DNA 等,从而使我们能够在芸芸众生的大千世界中准确地区分开每一个人。

Visual FoxPro 9.0 有一个属性窗口,供用户方便快捷地进行对象属性的设置。属性窗口

如图 9－1 所示。

属性有的仅可以在设计阶段设置，有的则仅可以在程序运行阶段得到修改，而有的则既可以在设计阶段设置又可以在程序运行阶段予以修改。这如同人的性别、血型一生下来就是固定的，而人的身高、体重却随着年龄的增大而在发生着变化一样。表 9－1 给出了 Visual FoxPro 9.0 中对象的常用属性及其应用范围。

图 9－1 属性窗口的组成

表 9－1 对象的常用属性

属性	说明	应用范围	缺省值
Caption	指定对象的标题(显示时标识对象的文本)	表单、标签、命令按钮	与 Name 相同
Name	指定对象的名字	任何对象	对象类型＋序号
Value	指定控件的当前状态(取值)	文本框、列表框	无
ForeColor	指定对象中的前景色	表单、标签、文本框、命令按钮等	黑色
FontName	指定对象的字体	表单、标签、文本框、命令按钮等	宋体
FontSize	指定对象的字号	表单、标签、文本框、命令按钮等	9 像素
BackColor	指定对象内部的背景色	表单、标签、文本框、列表框等	各自不同
BackStyle	指定对象的背景是否透明	表单、标签、图像等	1:不透明
BorderStyle	指定对象的边框样式	表单、标签、文本框等	各自不同
AlwaysOnTop	指定表单是否总处于其他窗口之上	表单	.F.
ScaleMode	指定表单的坐标单位	表单	3:像素
Closable	指定表单标题栏的关闭按钮是否有效	表单	.T.
ControlBox	是否显示标题栏的所有按钮和图标	表单、工具栏	.T.
MaxButton	指定表单是否含有最大化按钮	表单	.T.
MinButton	指定表单是否含有最小化按钮	表单	.T.
Moveale	指定表单在运行时能否移动	表单	.T.
WindowsState	指定表单在运行时最大化、最小化或普通	表单	0:普通
AutoCloseTables	指定表单在释放时是否关闭表或视图	数据环境	.T.
AutoOpenTables	指定表单在加载时是否打开表或视图	数据环境	.T.

3. 对象的行为特征——方法(Method)程序

对象的行为特征指附属于对象的所能执行的行为动作，也称为对象的事件和方法。例如

人类具有制造复杂工具和使用复杂工具的行为。

同类对象具有相同的行为，当某个行为作用于对象时，就称对象执行了一个方法程序。Visual FoxPro 中的方法程序是指它的系统内部定义的一系列通用的过程程序，使用它就能使对象执行某个指定的操作。方法程序的代码是由系统事先定义好的，用户无法看见。下面仅举出两个方法程序。

(1)Cls 方法程序

格式：Object.Cls

功能：清除表单中的文本和图形

(2)AddClass 方法

格式：AddClass()

功能：把一个类从类库(.VCX)中复制到另一个类库

注意　尽管方法程序代码是不可见的，但用户仍然可以对它作一定的修改。用户在代码编辑窗口所写入的代码相当于为该方法程序添加了新的功能，这些新添加的功能将和原有功能同时起作用。

Visual FoxPro 支持 Access 和 Assign 方法程序。在查询属性值或更改属性值时，可使用这些用户自定义的方法程序来执行代码。

在查询属性值时，将执行 Access 方法程序中的代码。通常是通过在一个对象的引用中使用属性，从而将属性保存到一个变量中，也可通过使用“?”命令来显示属性的值。

在进行属性值的更改时，则执行 Assign 方法程序中的代码。通常通过 STORE 命令或赋值操作符“=”来为属性赋新值。

Access 和 Assign 方法程序都是在运行阶段，查询或更新属性值时才能执行，在设计阶段将不起作用。

4. 事件(Event)

在现实客观世界中，各个对象之间存在着千丝万缕的联系和相互作用。正是由于对象之间的相互作用、相互联系和相互连接，才构成了世界上各种不同的系统。

(1)对象之间的交互

面向对象的程序设计中，把对象之间所进行的各种联系，称之为对象间的交互。例如，人们通过键盘、鼠标、屏幕可以和计算机进行联系，我们就称这是两个对象：人和计算机之间的交互。

(2)交互通过对象之间的消息传递来实现

例如人通过键盘输入了一条显示命令，计算机便将执行的结果显示在屏幕上。传过去的是指令消息，传回来的是显示结果消息。

(3)事件

当一个对象收到一个要它进行的某种操作时，称发生了某个事件。例如计算机收到了要它进行显示结果的命令，就将触发显示事件。

(4)Visual FoxPro 中的事件

在 Visual FoxPro 9.0 中，事件是预先定义好的特定的动作。由用户或系统来激活它。由用户激活的事件的动作有：单击、双击、移动鼠标、按键。如用户单击了鼠标，将触发 Click 事件，而双击了鼠标则将触发 DblClick 事件。事件被触发后，将执行该事件所对应的程序代码。

一个对象可以有许多隶属于它的事件，这些事件的名字事先都是由系统规定好的。一个

事件将对应一个程序，称之为事件程序。和方法程序不同，事件程序尽管名字事先已由系统规定好，但它们的代码一般却都是由用户自己编制。表 9－2 给出了 Visual FoxPro 9.0 中一些常见的事件。

表 9－2 Visual FoxPro 9.0 中的常见事件

事件名称	激活条件
Load	在对象创建前触发，首先是表单集，然后是所包含表单。它发生在 Activate，GotFocus，Init 事件之前。为了防止表单的生成，Load 事件将返回一个逻辑假.F.，Destroy 事件将不被执行
Init	在对象创建时触发。在容器对象的 Init 事件触发前，先触发它所包含的所有对象的 Init 事件，所以在容器对象的 Init 事件代码中可以访问它所包含的所有对象
Activate	当激活表单集、表单或页对象时，或者显示工具栏对象时触发
GotFocus	对象得到焦点时触发
Click	单击鼠标左键时触发
DblClick	双击鼠标左键时触发
MouseUp	释放鼠标键时触发
MouseDown	按下鼠标键时触发
KeyPress	当用户按下并释放某个键时触发
Valid	在控制失去焦点之前触发
LostFocus	当对象失去焦点时触发
Destroy	在对象被释放时触发。容器对象的 Destroy 事件发生在它所包含的所有对象的 Destroy 事件之前，所以在容器对象包含的所有对象被释放之前，容器对象的 Destroy 事件可以访问它们
UnLoad	释放对象时触发。它发生在 Destroy 事件和所有包含的对象被释放之后。另外，UnLoad 事件的触发还取决于对象的类型。代码中表单对象的释放，发生在当表单所涉及的对象变量被释放或当它的表单集被释放时

事件的触发方式有三种：

① 由用户触发。例如单击鼠标左键，将触发 Click 事件。

② 由系统触发。例如计时器事件 Timer，将会按设置的计时器时间间隔属性 Interval 的值自动触发。

③ 由代码触发。程序中由代码触发事件的格式为：

Object.*Event*

例如，若在某个程序代码中有：

Command1.Click

语句，则当程序执行到该句时，命令按钮 Command1 的 Click 事件将被激发。

9.1.2　类

面向对象程序设计另一个最重要的概念是类(Class)。类是对具有相同的属性结构和操作行为的一组对象的抽象。

1. 类的概念及类与对象的关系和区别

对象是类的实例,是客观存在的事实。而类则是对客观事实的抽象,是一个概念的东西。例如作为每一个人,在世界上是实实在在存在的,因此每个人都是一个实例,即一个对象。而由于每一个人都具有如前所述的:姓名、性别、出生年月、身高、体重、视力、听力、指纹、脚码、血型、脸型、五官四肢、体型、年龄、DNA 等属性结构,以及能制造和使用复杂工具的行为和动作,我们便从中抽象出了人类的概念。而人类绝不是指某一个具体的人。同样每一个具体的人,如张三、李四、王五等,他们本身并不是类,也不能代表类。但正是由于这么一种抽象,才使我们能清楚地将人类与动物和其他的类区分开来。

类是创建对象的模板,它包含了创建对象的属性描述和事件方法。在现实生活中,人们认识世界的规律是从特殊到一般,从具体到抽象,即先有对象,后有类的概念。例如,人们见了一只狗是四条腿且会跑,会吃会叫;见了一只羊也是 4 条腿会跑,会吃会叫……抽象出了凡四条腿且会运动,会吃会叫……的东西都属于动物这个类的概念,从而与四条腿的家具区分了开来。然而在程序设计时,人们却常常是反其道而行之,先定义类,包括定义类的属性、事件和方法,然后再去根据创建好的类来创建对象。这符合人们在认识世界的基础上改造世界的规律,人们总是先设计出图纸,才建起一座座工厂的。这里图纸是类,工厂是对象实体。类要做的任何事情,都必须通过建立具体的对象和在对象上进行操作才能实现。如同一张图纸若不建成工厂进行生产,则将只是一张纸。

2. 子类与继承

继承性表达了从一般到特殊的进化过程。在面向对象的方法里,继承是指基于现有的类来创建新类时,新类将继承现有类的方法和属性。此时,称现有的类为父类,称根据现有类而创建的新类为该现有类的子类。

父类和子类之间的继承关系,有以下三个特征:

① 父类和子类之间具有共享的特征,包括数据和程序代码。

② 父类和子类之间存在着差异和新增的部分,包括非共享数据和程序代码。

③ 两个类之间存在着层次结构。

对于一个子类而言,它的成员自然而然地应该包括:

① 从其父类继承而来的成员,包括属性、方法。

② 由子类自己所定义的成员,包括属性、方法。

继承可以使在父类中所作的添加、修复、改动自动反映到它的所有子类上。这种自动更新可节约用户的大量时间和精力。

3. 封装性

封装性是指在继承的过程中可以隐藏继承的信息。简单地说就是将许多功能包裹在一起,并且在适当的时间提供给程序设计者。

在 Visual FoxPro 9.0 中,可以对一个对象的属性和方法进行抽象处理,将它们封闭在对

象的内部。当用户在引用一个对象或创建一个新的对象时,该对象就有了一定的属性和方法,这就是对象的封装。

封装作用使得用户在使用 Visual FoxPro 9.0 提供的基类时,不必关心这些基类的内部使用什么代码和方法实现的,只要将其声明一下,就可以调用其固有的功能。因此代码的修改和维护将变得更加方便,因为对类的某个方法或属性进行修改时,只会在该类内部起作用,并不会影响到其他类的正常操作。

9.2 系统类

为了提高系统的开发速度,Visual FoxPro 9.0 提供了大量的类。这些类可以分为三类:基类(Base Class)、基础类(Foundation Class)、向导类(Wizard Class)。

9.2.1 基类(Base Class)

基类是 Visual FoxPro 9.0 所提供的基础类,又可以分为容器类和控件类。在 Visual FoxPro 中共提供了 29 个基类。

在 Visual FoxPro 9.0 的基类中,常用的容器类如表 9-3 所示。

表 9-3 Visual FoxPro 9.0 中的容器类

容器类名称	中文含义	可以包含的对象
CommandGroup	命令按钮组	命令按钮 Command
Form	表单(也称其为窗体)	任意控件、页框、容器、自定义对象
FormSet	表单集	表单、工具栏
Grid 中的 Column	表格中的列	标头和除表单、表单集、工具栏、计时器、其他表格列外的任意对象
Grid	表格	表格列
Page	页	任意控件、容器、自定义对象
OptionGroup	选项按钮组	选项按钮
PageFrame	页框	页
ToolBars	工具栏	任意控件、页框、容器
Container	容器	任意控件、页框、命令按钮组、选项按钮组、表格

1. 容器类(Container Class)

容器类指可以容纳别的对象的类。容器类可以包含其他的对象,并且也允许访问所包含的对象。它提供了将多个对象进行组合的功能。例如:表单是一个使用最广泛的容器类,它可以将命令按钮、文本框、标签、编辑框、图形框、复选框、表格等对象囊括其中,从而使功能变得十分强大。

2. 控件类(Control Object Class)

控件类指不能包含其他对象的类。它比容器类封装得更加完整,因此使用起来更加方便灵活。对于由控件类而创建的对象,无论是在设计还是在运行时都是按一个单元来对待的,构成控件对象的各个部分不能单独地操作和修改。所有的控件类都没有 AddObject 方法。在

Visual FoxPro 中，控件对象只能包含在容器中。常见的 Visual FoxPro 9.0 控件类（包括容器类）都列在表单控件工具栏中，如图 9－2 所示。

图 9－2　表单控件工具栏

9.2.2　基础类（Foundation Class）

Visual FoxPro 提供了 98 个基础类，它们全部以类库的形式，分布在 14 个类库中。系统的可视类库保存在系统安装目录的\Ffc\文件夹中。用户可以利用类设计器或类浏览器打开这些可视类中所包含的类，观察类的结构、属性和代码。图 9－3 给出了可视类库_Table2.vcx 中的类_DialogButton 的结构、属性和它的 dodialog 方法的代码。

图 9－3　_DialogButton 的结构、属性、dodialog 方法的代码

9.2.3　向导类（Wizard Class）

Visual FoxPro 9.0 中还提供了许多向导功能。系统的向导类保存在系统安装目录的 wizard 目录中。用户可以在 Visual FoxPro 9.0 的主菜单的[工具]菜单中选取[向导]选项，列出系统提供的所有向导如图 9－4 所示。这些向导的内部支持部分以向导类的形式存在。

图 9－4　向导类

9.3　类的创建与编辑

如前所述，类是创建对象的模板，创建类就是创建对象的模板，如同要铸造机器的外壳应先制造出铸造用的沙模一样。用户可以利用 Visual FoxPro 9.0 提供的基类、基础类和向导类，来创建新类，从而为对象的创建

奠定基础。创建类的任务就是要完成对类的属性和方法的定义。

和面向过程的程序设计中,过程的存储相类似,类程序的设计按程序的存放位置,可以分为如下三类。

① 内存形式:在主程序的后面直接写入类程序,当主程序执行时,它将与主程序一块同时调入内存执行。

② 程序文件方式:类以一个独立的文件形式存在于磁盘上,它可以通过"SET PROCEDURE TO [*ClassName*]"命令来打开类文件或关闭类文件。

③ 类库的方式:将设计好的类放在指定的类库中,供编程使用。

这三种方式中,最常用的方式是第 3 种:类库方式。因为类库提供了专门的管理工具,使类的创建和使用变得非常方便。

本节以示例的方式,介绍如何利用类库的方式先创建一个类库 myclslib,然后再创建一个 mycommand 类。

9.3.1 启动类设计器(Class Designer)

类设计器是创建类时惟一可使用的工具。用它可以设计可视类。可视类库的扩展名是 .VCX,它的备注文件的扩展名是 .VCT。打开类设计器的方法有菜单法和命令法。

1. 创建一个可视类库

使用下面的命令可以创建一个新的可视类库。

(1)命令格式

CREATE CLASSLIB *ClassLibraryName*

(2)功能

创建一个新的空可视类库(.VCX)文件。

(3)参数及子句说明

· *ClassLibraryName*:指定要创建的可视类库的名字。

(4)注意

可视类库的扩展名为 .VCX,系统将自动在文件名:*ClassLibraryName* 之后添加。使用 ADD CLASS 和 CREATE CLASS 命令,类的定义可以添加到可视类库中。

例 9.1 创建新的空可视类库:myclslib.vcx。

```
CREATE CLASSLIB myclslib          && 生成一个新的可视类库
```

2. 创建可视类

创建新类可以在类设计器中进行,打开类设计器的方法有菜单法、命令法和使用项目管理器三种。

(1)使用菜单打开类设计器

例 9.2 在例 9.1 创建的空类库中添加一个命令按钮类:mycommand。添加的结果如图 9-6 所示。

使用菜单打开类设计器创建类的步骤如下:

S1:打开"新建类"对话框:→[新建]→[类]↓ 新建类,如图 9-5 所示。

图 9－5　新建类对话框

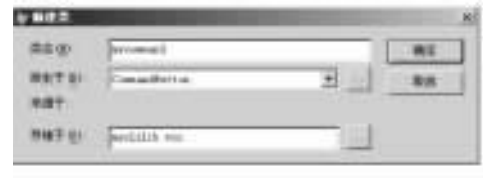


图 9－6　新建类对话框控件

新建类对话框中，各控件的意义如下：

• “类名”文本框：用来输入新创建的类的名字，现输入 mycommand。

• “派生于”组合框：提供了 Visual FoxPro 9.0 所有的基类的名字供选择，输入 CommandButton。

• “存储于”文本框：指定要存储到类库的名字，输入 myclslib.vcx。

S2：打开类设计器：→[确定]↓ 类设计器，如图 9－7 所示。

图 9－7　类设计器

(2)使用命令方式打开类设计器

如果命令打开类设计器，则可以使用下面的命令：

① 命令格式

CREATE CLASS *ClassName* [OF *ClassLibraryName*1]

　　[AS cBase*ClassName* [FROM *ClassLibraryName*2]] [NOWAIT]

② 作用

打开类设计器，允许用户创建一个新类的定义

③ 参数及子句说明

• *ClassName*：指定要创建的新类定义的名字。

• OF *ClassLibraryName*1：指定一个要生成的可视类库.VCX 的名字。如果该类库已经存在，则将类定义添加到它当中。

• AS cBase*ClassName*：指定要创建类的父类，它可以是 Visual FoxPro 9.0 中除 Column 和 Header 外的基类，也可以是用户自定义的类，如果缺省本子句，类定义将根据 Visual FoxPro 9.0 FormSet 基类而创建。

• FROM *ClassLibraryName*2：指定由 cBase*ClassName* 给出的用户自定义类所在的可视类库名。

例 9.3　在例 9.1 创建的类库中再添加一个命令按钮类：mycommand1。

CREATE CLASS myCommand1 OF myclslib AS "CommandButton"

执行该命令后将打开与图 9－7 相类似的类设计器，只不过此时类的名字已变为：mycommand1。

(3)使用项目管理器打开类设计器

使用项目管理器也可以方便打开类设计器。其步骤如下：

S1：打开[新建类]对话框：[项目管理器]→[类]→[新建]。

S2：填入有关参数，[确定]↓ 类设计器。

9.3.2 新建类的属性

对于刚创建的类，尽管已继承了父类的所有属性，但用户还可以根据需要，为其添加一些新的属性。步骤为：

S1：打开“类设计器”对话框。

S1－1：打开“打开”对话框：→[打开]↓ 打开。

S1－2：选择文件类型为可视类库(*.VCX)；选择类库名 myclslib，如图 9－8 所示。

S1－3：→[确定]↓ 打开，此时“打开”对话框中，以包含了本类库中的类列表。如图 9－9 所示。

图 9－8 “打开”对话框一

图 9－9 “打开”对话框二

S1－4：→[打开]↓ 类设计器，如图 9－7 所示。

S2：打开“新建属性”对话框：→菜单栏[类]→[新建属性]↓ 新建属性，如图 9－10 所示。

在“新建属性”对话框中有以下几项选项：

① 名称：指定要新建的属性的名称，例如输入 test。

② 可视性：该列表框下有三个选项：

· 公共(Public)：类似于面向过程程序设计中的全局内存变量，表示该属性可以被其它类或过程所引用。

· 保护(Protected)：类似于面向过程程序设计中的私有内存变量，表示该属性只能被本类或子类的方法程序引用。

图 9－10 “新建属性”对话框

• 隐藏(Hidden):类似于面向过程程序设计中的局部内存变量,表示该属性仅能被本类的方法程序引用。

③ Access 方法程序:指定是否为新属性创建 Access 方法程序。如果为该属性创建了 Access 方法程序,则只要查询该属性,就可以执行 Access 方法程序中的代码。

④ Assign 方法程序:指定是否为新属性创建 Assign 方法程序。如果为该属性创建了 Assign 方法程序,则只要查询该属性,就可以执行 Assign 方法程序中的代码。

⑤ 描述:可在该文本框中输入对该属性的说明。例如,输入"该属性用于学生学习新建属性"。

S3:→[添加],则新建的属性被添加到"属性"框中。查看"属性"框,会有如图 9-11 所示的结果。

用户还可继续添加新属性。

S4:添加结束。→[关闭]。

图 9-11　"属性"对话框

9.3.3　新建类的方法程序

新建类的方法程序的操作步骤与新添属性的步骤相类似。

S1:打开"类设计器"对话框。

S2:打开"新建方法程序"对话框:→[类]→[新建方法程序]↓新建方法程序,如图9-12所示。

在"新建方法程序"对话框中有以下几项选项:

① 名称:指定要新建的方法程序的名称。例如输入:dayin。

图 9-12　"新建方法程序"对话框

② 可视性:该列表框下有三个选项:

• 公共(Public):表示该方法程序可以在应用程序内的任何位置访问。

• 保护(Protected):表示不能为对象实例所访问,但可为子类所访问。

• 隐藏(Hidden):表示既不能为对象实例所访问,有不能为子类所访问。

③ 描述:为新添加的方法程序加以说明。

本例中在"名称"文本框输入"dayin",选"可视性"为"公共",在"描述"文本框输入"本方法程序用来打印文件",结果如图 9-12 所示。

S3:→[添加],则新建的方法程序被添加到类的方法程序中。

S4:添加结束,单击[关闭],退出"新建方法程序"对话框。

9.3.4 编辑类的属性和方法

创建类后,用户还可以编辑它的属性和方法程序。用户可以增加、修改、删除属性或方法程序,也可以增加类的功能或修改类的错误。其操作步骤如下:

S1:→[类]→[编辑属性/方法程序]↓编辑属性/方法程序,如图 9-13 所示。

图 9-13 "编辑属性/方法程序"对话框

可以看到,前面添加的"test"属性和"dayin"方法程序,已出现在对话框中了。

S2:在该对话框中,可以修改用户自定义的类的名称、说明和可视性,也可创建新建属性、方法程序,或移去它们。

注意 在类型栏中,"M"代表方法(Method),"P"代表属性(Property)。

用户也可通过命令来打开类设计器,创建或修改类的定义。命令如下:

(1)命令格式

MODIFY CLASS *ClassName* [OF *ClassLibraryName*1]

[AS *cBaseClassName* [FROM *ClassLibraryName*2]]

[NOWAIT] [METHOD *MethodName*] [SAVE]

(2)功能

打开类设计器,修改已存在的类定义或创建一个新的类定义。

(3)参数及子句说明

• *ClassName*:指定要修改或生成的类定义的名称。

• OF *ClassLibraryName*1:指定包含类定义的可视类库的名字,如果用户正在创建一个新的类定义并且可视类库已经存在,类定义将添加到该类库中。

• AS *cBaseClassName*:指定派生新类的父类。

• FROM *ClassLibraryName*2:指定含有由 *cBaseClassName* 给出的用户自定义类的可视类库名。

· METHOD *MethodName*：为类设计器中打开的代码窗口指定一个事件或方法。例如，如果要对 MyClassLibrary 可视类库中的 MyClass 类下的 txtFirstName 文本框内的 Click 事件立即编写代码，则可用下面的命令：

MODIFY CLASS MyClass OF MyClassLibrary METHOD txtFirstName.Click

· NOWAIT：作用同于前面的各命令。

· SAVE：有该关键字，当别的窗口激活时类设计器仍然保持打开状态；缺省 SAVE 关键字，则当其窗口打开激活时类设计器被关闭；如果在命令窗口执行时，SAVE 无影响。

9.4　对象的创建与使用

在类与对象的概念中，我们曾反复强调，类是对象的抽象，是创建对象的模板；对象则是类的实例。类不能被直接引用，只有通过类创建的相应对象，方可被引用。本节介绍对象的创建与使用。

9.4.1　对象的创建

在 Visual FoxPro 中，对象的创建既可以通过命令方式进行，又可以通过表单设计器创建。

1. 可视类库的打开

创建对象必须根据某个类来进行，因此在在创建对象前，应先打开包含该类的可视类库。打开类库的命令是：

(1)命令格式

SET CLASSLIB TO *ClassLibraryName*

[IN *APPFileName* | *EXEFileName*] [ADDITIVE] [ALIAS *AliasName*]

(2)命令功能

打开包含有类定义的可视类库。

(3)参数和子句说明

· *ClassLibraryName*：指定要打开的可视类库.VCX 的名字。当执行不带 *ClassLibraryName* 的 SET CLASSLIB 命令时，将关闭所有打开的可视类库。使用 RELEASE CLASSLIB 也可以关闭一个可视类库。

· IN APP*FileName* | EXE*FileName*：指定类库所在的应用程序文件名(.APP)或可执行文件名(.EXE)。

· ADDITIVE：在打开.VCX 可视类库时，不关闭任何当前已打开的可视类库。如果缺省该子句，则所有打开的可视类将被关闭。

· ALIAS *AliasName*：指定可视类库的别名，通过别名可以引用可视类库。例如，下面的命令将打开名字为 MyClass 的可视类库，为它分配一个别名：MyCntrls。

```
SET CLASSLIB TO MyClass ALIAS MyCntrls
```

2. 命令方式生成对象实例

在命令方式下，对象变量的创建是利用对象创建函数 CREATEOBJECT 来进行的。

(1)函数格式

CREATEOBJECT(c*ClassName* [, *eParameter*1, *eParameter*2,...])

(2)函数功能

从一个类定义或一个自动激活的应用软件创建一个对象。

(3)参数说明

• c*ClassName*:指定一个要创建新对象的类或 OLE 对象。

• *eParameter*1, *eParameter*2,…:这些可选择参数用来将值传递到类的初始事件过程。当用户执行 CREATEOBJECT()时,执行初始化事件使对象初始化。

注意　通常将该函数的返回的对象引用赋给某个变量,然后通过这个变量来标识对象、访问对象属性及调用对象方法。

例 9.4　为例 9.3 创建的按钮类: myCommand1 创建一个按钮控件对象 myButton。

myButton=CREATEOBJECT("myCommand1")

3. 使用表单设计器方式创建对象实例

在实际工作中,只要不是在程序中创建对象,一般都使用表单设计器来交互式创建对象。表单类本身是容器类,由它创建的表单是一种容器类对象。在表单中添加的对象既可以是容器类对象,又可以是控件类对象。关于表单我们将在表单一章详细讲解。

9.4.2　对象属性的种类

属性是对象的特征。在 Visual FoxPro 中,对象的属性可以在设计或运行时设置。在 Visual FoxPro 9.0 中,根据对象属性的设置方法,将属性分为三类。

图 9-14　"属性"对话框

1. 可擦写类属性

将在设计和运行期间均可设置的属性称为可擦写类属性。其类型是 Public 型的,在属性窗口中表现为正常字形的属性均为可擦写属性。例如图 9-14 中的 BackColor 属性就是可擦写属性。

2. 保护类属性

所谓保护类属性,指在设计与运行期间均不能重新定义的属性。这类属性只能引用它在设计期间所默认的值,即只能在运行期间读取而不能重新设置。这类属性一般在属性窗口中以斜体字显示出来。如图 9-14 中的 BaseClass 属性,就是保护类属性。

3. 隐藏性属性

隐藏性属性指这样一类属性。若用户只在定义的类中添加若干新的属性,但它们却并未被立即使用,就可以将这些属性设置为隐藏,从而使在设计和运行期间均无法获取或设置这些属性。

设置隐藏类属性的步骤如下:

S1:打开"编辑属性/方法程序"对话框:↓ 类设计器 →[编辑属性/方法程序],如图 9-15

所示。

图 9－15　“编辑属性/方法程序”对话框

S2：在“属性/方法信息”列表框中选取某个属性或方法，在对话框右侧的可视性列表框中选“隐藏”，则该属性被设置为隐藏。

9.4.3　对象属性值的设置

对象属性的值设置既可通过属性窗口直接赋值设置，又可利用命令来设置。前者最常用于表单的设计。

1. 使用属性窗口设置属性值

在表单设计期间，最常用的方式是使用属性窗口来设置对象的属性。属性窗口如图9－14所示。此部分内容将在表单一章予以详细介绍。

2. 使用命令设置属性值

对象属性值的设置命令，按其设置的对象属性值的个数，可分为设置单个属性和设置多个属性。

(1)设置对象的单个属性

为对象的某个属性赋值的格式为：

ParentObjectName.ObjectName.Property=Value

例 9.5　将例 9.4 所创建的按钮对象的标题属性 Caption 设置为“命令按钮”。

myButton.Caption=″命令按钮″

例 9.6　下面的语句是为表单“欢迎”中的标签控件 Label1 设置宽、高、左边界、顶边界、字体、颜色、背景样式属性的。

欢迎.label1.Width=600

欢迎.label1.height=40

欢迎.label1.top=60

欢迎.label1.left=欢迎.label1.Width/2

欢迎.label1.FontName=″华文新魏″

欢迎.label1.ForeColor=RGB(255,0,0)

欢迎.label1.BackStyle=0

显然,用设置单个属性值的方法对来设置多个属性值是不方便的。因此对于多个属性值的设置一般都采用 WITH 命令来进行。

(2)设置对象的多个属性

设置对象的多个属性,采用 WITH……ENDWITH 语句甚为方便,它的语法格式为:

```
WITH ParentObjectName.ObjectName
    .Property1=Valie1
    .Property2=Valie2
    .……
ENDWITH
```

例 9.7 将例 9.6 用 WITH 命令实现。

```
WITH 欢迎.label1
    .Width=600
    .height=40
    .top=60
    .left=欢迎.label1.Width/2
    .FontName="华文新魏"
    .ForeColor=RGB(255,0,0)
    .BackStyle=0
ENDWITH
```

注意 在 WITH 体中,各属性名称前面的“.”不得省略。

9.4.4 对象的引用

在面向对象的程序设计中,常常需要引用对象,或引用对象的属性、事件与调用方法程序。引用时,根据路径的书写格式,将引用分为相对引用和绝对引用。

1. 对象的相对引用规则

对象的相对引用常通过以下的关键字来进行。这些关键字及其所代表的对象是:

- ThisFormSet:当前的表单集。
- ThisForm:当前的表单。
- This:当前的对象。

引用的格式是:

```
KeyWord.Object.Property|.Event|.Method
```

例如,如果例 6 的“欢迎”表单为当前表单对象,则下面的语句表示:设置该表单的标签控件 Label1 的标题 Caption 属性。

```
ThisForm.Label1.Caption="欢迎教育部专家莅秦东大学进行本科教学水平评估!"
```

如果标签控件 Label1 是当前控件,则上句也可简写为:

```
This.Caption="欢迎教育部专家莅临秦东大学进行本科教学水平评估!"
```

2. 对象的绝对引用规则

对象的绝对引用，是指引用路径不采用关键字，直接给出各层次的对象名的引用方式。其格式为：

ParentName.ObjectName.Property|.Event|.Method

例如：

欢迎.Label1.Caption ="欢迎各位专家莅临秦东大学进行本科教学水平评估!"

3. 对象方法程序的调用

前面已经给出了无返回值的对象的方法程序的调用格式。对于有返回值的对象方法程序调用，只要在对象的方法程序名字之后带上一对圆括号即可。如果方法程序系带参数调用，则参数应写在圆括号内。

例如，下面的语句将把用户自定义的 GetNewCaption 方法程序的返回值设置成表单的标题。

ThisForm.Caption=ThisForm.GetNewCaption()

又如，下面的语句将把实参 n 传递给方法程序 Show 的代码。

ThisForm.Show(n)

例 9.8　显示表单"欢迎"，并将焦点设置在文本框 text1 上。

```
欢迎.show
ThisForm.Text1.SetFocus
```

9.5　思考与练习

一、选择题

1. 下面关于面向对象的叙述中，错误的是(　　)。

A) 每个对象在系统中都有惟一的对象标识

B) 事件作用于对象，对象识别事件并做出相应反应

C) 一个子类能够继承其所有父类的属性和方法

D) 一个父类包括其所有子类的属性和方法

2. 对于创建新类，Visual FoxPro 提供的工具有(　　)。

A) 类设计器和报表设计器

B) 类设计器和表单设计器

C) 类设计器和查询设计器

D) 类设计器

3. 在 Visual FoxPro 中，当对象方法或事件代码在运行过程中产生错误时将引发的事件是(　　)。

A) Load　　B) Init　　C) Destroy　　D) Error

4. "类"是面向对象程序设计的关键部分，创建新类不正确的方法是(　　)。

A) 在.PRG 文件中以编程方式定义类

B) 从菜单方式进入"类设计器"

C) 从命令窗口键入 CREATE CLASS 命令,进入“类设计器”

D) 在命令窗口输入 ADD CLASS 命令

5. 下列关于属性、方法和事件的叙述中,错误的是(　　)。

A) 属性用于描述对象的状态,方法用于表示对象的行为

B) 基于同一个类产生的两个对象可以分别设置自己的属性值

C) 事件代码也可以像方法一样被调用

D) 在创建一个表单时,可以添加新的属性、方法和事件

6. 下列说法错误的是(　　)。

A) 事件既可以由系统引发,也可以由用户激发

B) 事件代码既能在事件触发时执行,也能够像方法一样被调用

C) 在容器对象的嵌套层次里,事件的处理遵循独立性原则,即每个对象的识别并处理属于自己的事件

D) 事件代码不能由用户创建,是惟一的

7. 下面关于面向对象数据库的说法,不正确的选项是(　　)。

A) 任何对象,都有自己的属性和方法

B) 每个对象在系统中都有惟一的对象标识

C) 用<父类名>-<方法>的命令继承父类的事件和方法

D) 一个子类能够继承其所有父类的属性和方法

二、填空题

1. _____是一种由系统预先定义而由用户或系统发出的动作。

2. 在 Visual FoxPro 中的类一般分为两种类型:_____和_____。

3. _____是对一组对象的抽象,这组对象具有相同的属性结构和操作行为。

三、上机题

1. 上机熟悉类的创建、注册和使用方法。

2. 创建一个命令按钮类,设置其属性、事件代码,使其具有移动表的指针的功能。

第 10 章　表单和表单集

表单和表单集都是一种容器类对象，一个表单集可以包含若干个表单，一个表单可以包含若干个容器类或控件对象。在 Visual FoxPro 9.0 的应用系统中，用户界面设计是通过对表单设计来实现的。在外观上，表单与窗口无异，有标题栏、最大化、最小化、关闭按钮等。表单是一个容器对象，设计者可以在表单中添加各种对象，定义它们的属性并为其编写事件代码，通过这些对象的各种事件，完成特定功能。如显示与编辑数据、运行其他表单、报表和其他应用程序等。

表单文件的扩展名为：.scx，表单备注文件的扩展名是：.sct。

表单设计一般应按下面的步骤进行：

S1：创建一个新表单，设置表单的各种属性值。

S2：设置表单的数据来源。

S3：添加表单控件并设置其属性。

S4：为表单或它所包含的各个对象编写有关的事件代码。

表单制作方法有：使用表单向导、表单生成器、表单设计器。

10.1　表单情况下的菜单与工具栏

10.1.1　菜单与工具栏说明

当建立表单时，Visual FoxPro 9.0 会自动增加“表单”菜单，并改变了“显示”、“格式”两个菜单中的选项内容。

“格式”菜单中的选项多用于控制对象的外观或位置，与“布局工具栏”中的按钮基本对应。

“表单”菜单和“显示”菜单分别如图 10－1、图 10－2 所示。

图 10－1　“表单”菜单

图 10－2　“显示”菜单

当要创建一张表单时，除 Visual FoxPro 本身的工具栏外，还将涉及到：表单设计器、布局、表单控件、调色板等工具栏。它们各自的图标如图 10－3 所示。和所用的 Windows 窗口界面一样，当鼠标移动到任何一个图标时，系统将自动显示出该图标的功能，供用户选择。

图 10－3　表单创建界面的四个工具栏

10.1.2　控件的基本操作方法

在表单上对控件的操作，指在表单设计阶段对控件的操作，包括控件的选定、取消、移动、复制、删除、改变大小等。它们基本类似于 Windows 下的图标操作，对控件的各种基本操作要遵循“先选定后操作”的原则。

单个控件或多个控件的选定与取消选定、复制与删除、控件的移动方法均同于 Windows 下对文件的操作，既可使用鼠标，也可使用键盘，还可使用鼠标和键盘的联合操作。

控件大小的改变，则同在 Windows 下对图形的操作，对于选定的控件在四周会出现八个控制点（当控件比较小时只在四个角上），用户只要将鼠标移到任意一个控制点，然后按住左键拖曳，可以改变控件的大小。

如果要在多个被选对象中取消一个控件，则按住 Shift 键后再单击该控件即可。

值得一提的是，当控件被复制时，它的属性、方法程序、事件及事件代码均被复制。

10.2　利用表单向导创建表单

用表单向导既可以创建单表表单，又可以创建一对多数据表表单。本节以sjsjk.dbc为数据源，详细讲解使用表单向导建立和运行表单的过程。

10.2.1　创建表单

1. 使用“表单向导”创建单表表单

例 10.1　根据数据表 xsqkb.dbf，用表单向导创建一个单表表单“xsqkb.scx”。

步骤如下：

S1：打开项目管理器及数据库“sjsjk.dbc”对话框。

S2：打开“新建表单”对话框。在项目管理器窗口下，→[文档]→[表单]→[新建]↓

新建表单，如图 10－4 所示。

图 10－4　在项目管理器中创建表单

S3：打开“向导选择”对话框。→[表单向导]↓ 向导选择，如图 10－5 所示。

在“向导选择”对话框中，有两个选项供用户选择：

① “表单向导(Form Wizard)”：创建单表或视图的表单向导。

② “一对多表单向导(One-To-Many Wizard)”：建立一对多关联数据表的表单向导。

→“表单向导” →[确定] ↓ 表单向导第 1 步—选择字段，如图 10－6 所示，选择字段。“表单向导”的后面各步骤与“查询向导”、“视图向导”类似，用户将在它的指引下，逐步完成表单设计。

图 10－5　“向导选择”窗口

图 10－6　“表单向导”第 1 步

S4：当执行到表单向导“第 4 步完成”时，得到如图 10－7 所示对话框。

S5：→[预览]，将得到表单的预览结果如图 10－8 所示。

S6：将生成的表单存入名字为“xsqkb.scx”的表单文件中。

图 10－7　“表单向导”第 4 步——完成

2. 使用“表单向导”创建一对多表单

在表单向导方式下建立一对多表单，方法和建立单表表单雷同，只要跟着向导指引的步骤走，就一定能创建出一个表单来。此处无需再加介绍，留给读者自己上机完成。

图 10－8　表单“xsqkb.scx”的预览结果

10.2.2　运行与修改表单

单击项目管理器的文档卡中，选定表单，例如选择xsqkb.scx，然后单击“运行”按钮，可以运行表单，运行结果也如图 10－8 所示。从图中可以看出，表中的各个字段名都被做成了标签，各字段根据不同的数据类型以不同的控件方式显示出来：字符型、日期型字段用文本框显示，逻辑型字段用复选框显示，备注型字段用编辑框显示，通用型字段用 ActiveX 绑定控件显示。同时有十个命令按钮，包含在一个命令按钮组中。但有些控件的属性却并非完全随人心愿，这就需要对表单进行修改。

单击图 10－8 中的退出按钮(Exit 或表单右上角的关闭按钮)，可退出运行状态，单击项目管理器中的[修改]按钮，可进入表单设计器如图 10－9 所示。将各个控件的位置、大小等属性进行修改。关于控件属性的设置，将在后面的表单控件一节中详细讲解。在修改过程中，随时可以使用工具栏上的 ! 按钮运行表单。稍加修改后的表单即 xsqkb.scx 运行的结果，如图 10－10 所示。

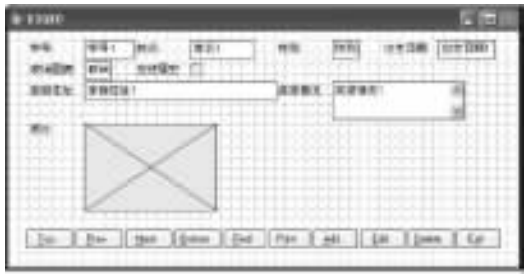


图 10－9　表单控件分布图

图 10－10　稍加修改后的表单 xsqkb.scx 运行结果

在命令窗口下运行表单命令是：

DO FORM form

例如，要通过命令窗口运行表单：xsqkb.scx，则应在命令窗口输入：

DO FORM xsqkb

10.3　利用表单设计器创建表单

表单向导只能创建表单，但却不能修改表单。在 Visual FoxPro 应用系统开发过程中，常常采用“表单设计器”创建表单，这种方法的步骤为：

S1：启动“表单设计器”。

S2：设置数据环境。

S3：添加控件并设置控件属性。

S4：为控件编写过程代码。

本节以创建一个与表单 xsqkb.scx 相类似的表单 xsqkb1.scx 为例，介绍用表单设计器创建表单的方法和步骤。

10.3.1　启动表单设计器

启动表单设计器的方法同于启动其他设计器，有菜单法和命令法。

1. 使用系统菜单

在 Visual FoxPro 9.0 系统菜单中：→[文件] →[新建] →“表单” →[文件] →[新建] → 表单设计器。一般伴随着表单设计器的打开还常常显示出表单控件工具栏和属性对话框，如图 10-11 所示。

图 10-11　表单设计器及表单控件工具栏和属性框

2. 使用项目管理器

在项目管理器中：→[文档] →[表单] →[新建] →[新建表单]。

3. 使用命令

命令格式：CREATE FORM [*form*|?]

例如：CREATE FORM xsqkb1

如果要修改原有的表单则可以通过：

MODIFY FORM *form*

打开表单设计器。例如，要通过命令窗口修改表单：xsqkcjb.scx，则应在命令窗口输入：

MODIFY FORM xsqkcjb

10.3.2 为表单设置数据源

在项目管理器中，设置数据环境的步骤是：

S1：打开“添加表或视图”对话框。右击“表单窗口”空白处↓ 快捷菜单 →“数据环境”↓ 添加表或视图，与此同时“数据环境设计器”窗口亦同时显示出来。如图 10－12 所示。

图 10－12 添加表或视图对话框和数据环境设计器窗口

S2：将表或视图添加到数据环境设计器。在“添加表或视图”对话框，选择数据库为：Xssjk，数据表为：xsqkb，→[添加] →[关关闭]，则添加表或视图对话框被关闭，选中的数据表就进入数据环境窗口，如图10－13所示。

图 10－13 添加了表的数据环境设计器

注意 在数据环境设计器中单击鼠标右键，也可以打开添加表或视图对话框，添加其他的表。还可以对表进行各种操作、设置数据表之间的关联、浏览数据表内容。

10.3.3 使用快速表单

用户创建的新表单，常常是一个不含任何控件的空表单。而如果要在表单中显示表的字段，则应将表的字段添加到表单中。为了方便添加，Visual FoxPro 提供了一种快速表单，以完成此项工作。

快速表单的使用方法如下。

例 10.2 利用快速表单创建表单：xsqkb1.scx。

创建步骤如下：

S1：创建一个新表单。

CREATE FORM xsqkb1

S2：打开“表单生成器”对话框。在系统菜单中：→[表单] →[快速表单]↓ 表单生成器，

如图 10－14 所示。

S3：选择表单字段和样式。将“可用字段”列表中的所有字段全部选入“选中字段”列表；选“风格”卡为：轮廓。

S4：→[确定]，得到如图 10－15 所示的快速表单设计结果。

图 10－14　表单生成器

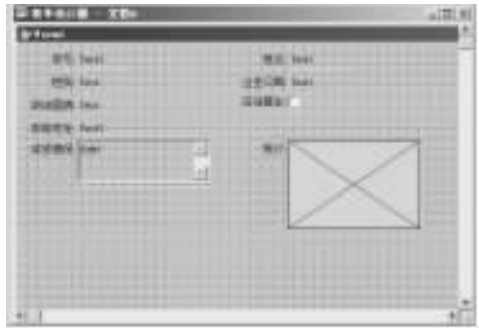


图 10－15　快速表单创建结果

S5：运行该表单，会得到如图 10－8 相类似的结果。

只要稍加留神就会发现，快速表单并无按钮组，因此运行时也只能显示一条记录。可见快速表单仅是将与数据环境中的表有关的字段，快速地添加到表单中的一种方法。要在表单中加入其他控件，则只能用另外的方法来实现。这些方法将在后面介绍。

10.4　表单的常用属性方法及事件

表单本身具有自己的属性、方法程序和事件，上节是在表单向导的指导下设计表单的，对于表单的属性、方法程序和事件的设置理解并不深刻，本节将较详细地介绍一些最常用的属性、方法程序和事件，并且介绍设置它们的方法和步骤。

10.4.1　表单的常用属性

表单的属性很多，常用属性有：外观属性、标题栏属性、其他常用属性三类。

1. 表单的外观属性

表单的外观属性指表单的大小、颜色位置、边框等。其常用的外观属性有：

(1)高度属性——Height

设置表单在屏幕上的高度，它并不包含边框和标题栏的高。设置表单对象的高度时，其单位由表单的 ScaleMode 属性决定，缺省情况下以像素为单位。

(2)宽度属性——Width

设置表单在屏幕上的宽度，不包含边框的宽。

(3)边框样式属性——BorderStyle

设置表单边框的显示样式。有四种选择值：

0——无边框；1——单线边框；2——固定对话框边框；3——可调型边框(缺省型)。

(4)窗口型属性——WindowState

设置表单在运行时是否可以最大化或最小化。该属性有三种选择值：

0——普通，即按表单设计大小激活；1——最小化（只用于 Windows）；2——最大化。

(5)滚动条属性——Scrollbars

设置表单的滚动条类型。该属性有四种选择值：

0——无（缺省）；1——水平滚动条；2——竖直滚动条；3——既水平又竖直滚动条。

设置了滚动条后，当表单显示不下它所包含的内容时会自动出现设置的滚动条。

(6)背景颜色属性——BackColor

用来设置表单内文本和图形的背景颜色，可采用 RGB 三色设置，也可通过单击属性框中属性值文本框右侧的属性设置按钮调出颜色对话框，自己选择需要的颜色。如图 10－16 所示。

图 10－16 “颜色”设置

(7)自动居中属性——AutoCenter

设置表单第一次显示于 Visual FoxPro 主窗口时，是否自动居中。这是一个逻辑属性：.F. ——假（默认值）；.T. ——真。

表单一些属性的设置可以在表单设计阶段，通过属性对话框进行。打开表单设计器后，如果属性对话框未出现，则可以用下面的任一方法打开它。

① 方法 1：使用系统菜单，→[窗口] →[属性窗口] ↓ 属性窗口。

② 方法 2：使用快捷菜单，右击表单 ↓ 快捷菜单 →[属性] ↓ 属性窗口。

③ 方法 3：使用系统工具栏，→ [属性窗口]。

例 10.3 创建一个表单：bd1.scx，并设置如表 10－1 所示的外观属性。

表 10－1 表单bd1.scx的外观属性

属性名称	设置值	属性名称	设置值
height	384（像素）	ScrollBars	3（既水平又竖直滚动条）
width	512（像素）	BackColor	0,0,255（高亮度蓝色）
BorderStyle	3（可调性）	AutoCenter	.T.
WindowState	0（普通）		

S1：CREATE FORM bd1。

S2：打开“属性”对话框，设置有关属性，例如设置 height 属性，如图 10－17 所示。

S3：继续设置其他属性，直到表 10－1 给出的属性全部设置完毕。

S4：存盘并运行：bd1.scx，即会得到如图 10－18 的显示结果。

图 10－17　属性对话框

图 10－18　表单外观属性设置结果

2. 标题栏属性

表单的标题栏属性包括表单的图标、标题、最小化按钮、最大化按钮、关闭按钮、可移动性、标题栏可见等。这些属性的名字、功能、取值由表 10－2 给出。

表 10－2　表单的标题栏常用属性

属性名称	属性功能	取值	缺省值
Icon	设置表单图标	图标文件	小狐狸
Caption	设置表单标题	字符串	
MinButton	指定表单是否带有最小化按钮	.t.（带），.f.（不带）	.t.
MaxButton	指定表单是否带有最大化按钮	.t.（带），.f.（不带）	.t.
Closable	指定表单是否带有关闭按钮	.t.（带），.f.（不带）	.t.
Movable	设置是否允许通过拖曳表单标题栏来拖动表单	.t.（允许），.f.（不允许）	.t.
TitleBar	设置表单标题栏是否可见	1（显示），0（不显示）	1

例 10.4　修改表单：bd1.scx，将它的图标用另一个图形代替，将表单标题设置为“学生学籍管理系统”，其余标题栏属性一律采用缺省值。

S1：打开表单：bd1.scx。

S2：利用属性框设置：Icon 为：Cab.ico；Caption 为：“学生学籍管理系统”。

S3：存盘并运行表单，结果如图 10－19 所示。

图 10－19　添加了标题属性的表单：bd1.scx

3. 其他常用属性

除上述两大类属性外，表 10－3 给出了表单最常用的其他属性。

表 10-3 表单的其他常用属性

属性名称	属性功能	取值	缺省值
Enabled	指定表单是否活动	.T.（活动）,.F.（不活动）	.T.
Visible	指定表单是否可见	.T.（可见）,.F.（不可见）	.T.
AlwaysOnTop	指定是否将表单总设置在最顶层	.T.（是）,.F.（不）	.F.
AlwaysOnBottom	指定是否将表单总设置在最底层	.T.（是）,.F.（不）	.F.
ShowTips	指定当鼠标移动到表单对象和工具栏的对象上的控件时，是否给出该对象功能的提示	.T.（是）,.F.（不）	.F.
WindowType	指定表单类型	0（无模式）、1（模式）	

注：表单类型 WindowType，指当表单显示或用 DO 语句运行时如何动作。它分为无模式表单和模式表单两种。无模式表单又称为一般表单，它可进行表单之间的工作交换；模式表单是一种独占表单，当这种表单被激活后，处理程序就完全交给了本表单，只有当这个表单运行结束而关闭后，原来的程序才会继续运行。对于表单，它有两个值可供选择：0——无模式；1——模式。

10.4.2 表单的常用事件

Visual FoxPro 9.0 支持 68 种事件和方法程序。本节按表单事件触发的事件顺序介绍常用的表单事件。

1. 表单加载时触发的事件 Load

表单加载事件，是指在执行：DO FORM 命令开始运行表单或表单集时，首先将表单装入内存而要触发的事件。本事件只有一个，即：Load。在 Load 事件触发时，系统还没有创建包括表单和表单集在内的任何对象。可见 Load 事件是先于任何其他事件而触发的事件，所以在 Load 事件中是不能引用表单内任何控件对象的。

Load 事件的触发是按先外后里的次序进行的，即先触发表单集的 Load 事件，再触发表单的 Load 事件。在 Visual FoxPro 9.0 中，除表单集和表单外，其他的对象一律没有 Load 事件。

2. 表单建立时触发的事件 Init

表单建立时触发的事件指表单加载后根据表单的设计所触发的表单的初始化事件：Init，发生在 Load 事件之后。

和 Load 事件不同，Init 事件触发的次序是由里层到外层，即先触发表单内的各个控件，然后才触发表单，最后触发表单集。在一个表单内部，各个控件的触发次序依照设计时它们的添加次序而确定。所以，可以在表单的 Init 事件代码中处理表单内的任何一个控件对象，在添加较迟的控件对象的 Init 事件代码中也可访问比它添加早的控件对象，但反之则不可。

3. 表单交互操作期间触发的常用事件

在表单、表单集等容器类对象被加载、创建之后，所进行的常常是表单的交互式操作。在表单的交互式操作阶段，常常会触发下面几个事件。表 10-4 给出了这些事件的名字和触发

条件。

表 10－4　表单交互操作期间触发的常用事件

实践名称	事件触发条件说明
Activate	当表单作为容器类对象被激活时所触发
Deactivate	与 Activate 事件相反，当表单失去控制权时触发
KeyPress	当用户按下并释放键盘中的某个键时触发。当表单为空表单或所包含的控件对象不可见时，才可触发该事件。用户应该将表单的 KeyPreview 属性设置为：.T. ，才能激活 KeyPress 事件。同时注意任何与 Alt 键的组合键都不能触发 KeyPress 事件
Click 事件	鼠标单击对象时触发
DblClick	鼠标双击对象时触发
RightClick	鼠标右击对象时触发

4. 表单释放阶段触发的事件 Destroy

在表单释放阶段，将触发 Destroy 事件。和表单创建时触发的 Init 事件相反，Destroy 事件触发的次序是由外层到里层，即先触发表单的 Destroy 事件，然后再触发表单容器内所有各个控件对象的 Destroy 事件。在同一个表单内，控件对象 Destroy 的触发次序也是按添加时的逆序进行的。因此在表单的 Destroy 事件代码中能够访问它所包含的所有控件对象。

5. 表单卸载阶段触发的事件 Unload

表单卸载时将触发 UnLoad 事件。它发生在表单、表单集的 Destroy 事件之后。

表单的 Destroy 和 UnLoad 事件一般在以下两种情况下触发：

① 当表单执行：Release 方法时。

② 单击表单的关闭按钮时。

例 10.5　修改表单：bd1.scx，将表单的高度和宽度适当调整，在表单中添加一个标签控件 Label1，设置其标题属性 Caption＝""，背景样式属性 BackStyle＝0（透明），字体名称属性 FontName＝"华文新魏"，字大小属性 FontSize＝14，自动大小属性 AutoSize＝.t. ，左边界 Left＝30，顶边界 Top＝110，在它的有关事件中写入如表 10－5 所示的各种代码，然后运行表单观察其结果。

S1：打开表单：bd1.scx。

S2：添加一个标签框到 bd1，并在属性对话框中设置有关属性为题目给出的数据。

S3：双击表单，调出代码编写窗口。

在代码编写窗口中，对象列表框给出了表单中所有的对象的名字（含表单本身）供选择；过程列表框列出了所有的事件及方法程序的名字供用户选择。

选对象为：Form1，选过程为：Load，在代码编辑框中写入：

Wait "引发 Load 事件，稍等后按任意键继续！" Window

如图 10－20 所示。

图 10－20　表单bd1.scx的 Load 事件代码

S4：重复第 3 步，分别为表单bd1.scx的 Init，Click，Destroy，UnLoad 事件写入表 10－5 所给出的代码。

表 10－5　在表单bd1.scx中要设置的事件代码

事件	代码
Load	Wait "Load 事件已引发，稍等后按任意键引发 Init 事件！" Window
Init	ThisForm.Label1.Caption ="Init 事件已引发，稍等后按任意键引发 Click 事件！"
Click	ThisForm.Label1.Caption="Click 事件已触发，请稍候按关闭按钮引发 Destroy 事件！"
Destroy	ThisForm.Label1.Caption=="Destroy 事件已引发，请稍等后按任意键引发 UnLoad 事件！"
Unload	Wait "UnLoad 事件已引发，稍等后按任意键继续！" Window

S5：关闭表单设计器，存盘。

S6：DO Form bd1 ，运行表单bd1.scx，屏幕上会得到如图 10－21 所示的图形序列。当屏幕显示到图 10－21(5)时，再按任意键，将关闭本表单，返回 Visual FoxPro 9.0 环境。

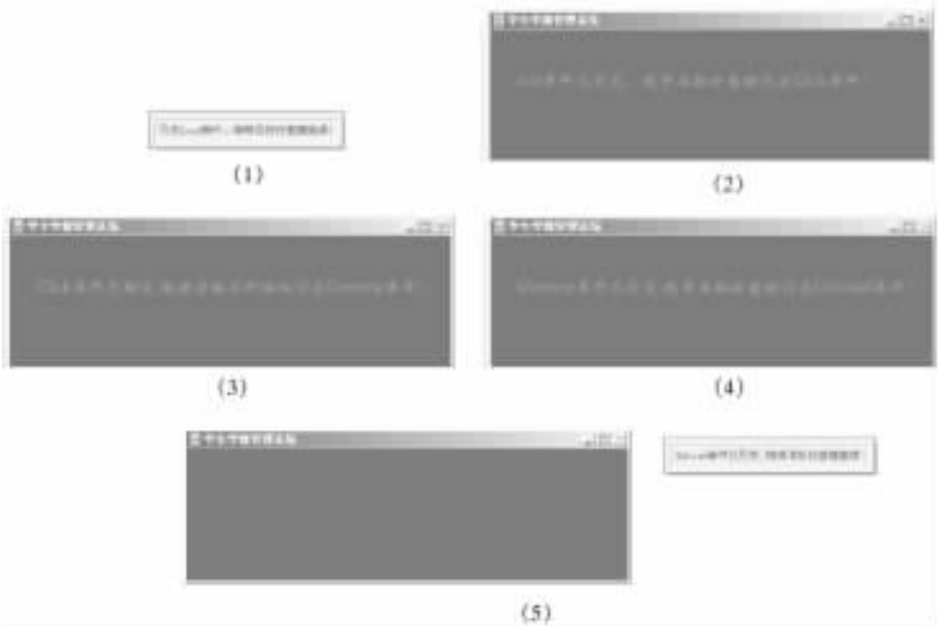


图 10－21　表单bd1.scx各步执行结果

10.4.3　表单的常用方法程序

方法程序的引用格式在上一章已作过介绍，采取：

Object.*Method*[(*ParameterList*)]

的格式进行，表 10－6 给出了在表单中最常用的方法程序。

表 10－6　表单中常用的方法程序

方法程序名	功能说明
AddObject	在表单运行中向表单添加控件
Release	从内存中被释放表单
Refresh	重新绘制表单或控件，并刷新它们的所有值。当表单被刷新时，表单内的所有控件也全部被刷新，但页框被刷新时仅刷新活动的页
Show	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
Hide	隐藏表单。该方法将表单的 Visible 属性设置为.F.

10.5　表单的数据环境

在表单中，数据环境起着将表与表单相联系的桥梁作用。因此凡涉及到表的表单，都要事先定义数据环境。数据环境的定义包括：数据环境对象(Data Environment)、指针对象(Cursor)和关系对象(Relation)，根据系统的要求，将这些对象联系在一起将构成数据环境的定义。

数据环境是一个容器类对象，它也具有自己的属性、事件和方法程序。

10.5.1　数据环境的属性

数据环境是一个非可视化对象，它具有自己的属性。常用的数据环境属性有五个。表 10－7给出了这些属性的名字、功能、取值。

表 10－7　数据环境的属性

属性名称	属性功能	取值	缺省值
Name	供在代码中引用数据环境对象用		
AutoOpenTables	指定当表单被激活时，是否自动打开数据环境中的表或视图。如果要在编码环境中设置，必须在 Init 事件中编写代码，并通过调用 OpenTable 方法实现	.T.（是），.F.（否）	.T.
AutoCloseTables	指定当释放表单或表单集时，是否自动关闭数据环境中的表或视图。如果要在编码环境中设置，必须在 Destroy 事件中编写代码，并通过调用 CloseTable 方法实现	.T.（是），.F.（否）	.T.

续表 10-7

属性名称	属性功能	取值	缺省值
InitialSelectedAlias	当加载数据环境时,指定一个与临时对象有关的别名为当前的别名。这仅在表单的设计阶段进行,而在表单的运行阶段为只读		
OpenView	确定与表单、表单集,或报表自动打开的数据环境相关的视图类型。仅在设计阶段有效,运行阶段为只读	0:本地和远程视图 1:仅本地视图 2:仅远程视图 3:无	0

10.5.2 数据环境的方法程序和事件

数据环境的方法程序和事件都有多个,最常用的方法程序有两个,事件有四个。表 10-8 给出了这些常用的方法程序和事件。

表 10-8 数据环境常用的事件和方法程序

事件	触发条件	方法程序	功能说明
Init	创建数据环境对象时	CloseTable	关闭数据环境中的所有表和视图
BeforeOpenTables	在数据环境中未打开表之前	OpenTable	打开数据环境中的所有表和视图
AfterCloseTables	在数据环境中关闭表之后		
Destroy	数据环境被释放时		

10.5.3 数据环境设计器的打开

数据环境的激活可以通过下面三种方法之一来实现。

- 使用系统菜单:打开表单设计器→[显示]→[数据环境]。
- 使用工具栏:打开表单设计器→[数据环境]。
- 快捷菜单:打开表单设计器,右击表单→[数据环境]。

10.5.5 指针对象 Cursor 和关系对象 Relation

数据环境包含有 Cursor 和 Relation 对象。当在数据环境设计器中添加了表或视图后,就生成了 Cursor 对象。向数据环境中添加表或视图的方法前面已作了介绍。有时为了在仅打开数据环境设计器而未显示添加表或视图对话框的情况下,继续添加表或视图,可右击数据环境设计器,打开快捷菜单,选[Add]后,打开添加表或视图对话框,继续添加,从而生成相应的指针对象 Cursor。

当表单中同时要操作两个相关联的表时,就要创建两个表之间的关系对象。创建的方法前面已作过讲述。

10.6　表单的设计

表单分为表单的设计和表单的运行两个阶段，要获得一个在运行阶段显得美观、灵活、方便、快捷的表单，必须先在设计阶段进行卓有成效的表单设计工作。

在设计阶段要做的工作是向表单中添加控件和设置表单及控件的初始属性。

10.6.1　向表单中添加控件

向表单中添加控件的方法很多，前面介绍的快速表单就是一种添加方法。现在再介绍两种添加方法。这两种方法都是在表单设计器环境中进行的，即在表单设计阶段进行。

1. 从数据环境的表中直接拖动字段到表单

这种添加法与快速表单类似，只能添加与数据环境中表或视图有关的控件。下面通过一个实例介绍添加的过程。

例 10.6　创建一个 xsqkcjb1.scx 表单，用从数据环境的表中直接拖动字段的方法向表单中添加有关控件。

S1：CREATE FORM xsqkcjb1.scx。

S2：打开数据环境设计器，将 xsqkb.dbf、xscjb.dbf 添加到其中，如两表无关系，则应根据学号创建两表的关联关系。

S3：在数据环境设计器窗口，在指针对象 Cursor 中选取需要的字段，将其拖动到表单的适当位置。

S4：重复执行第 3 步，直到要显示的字段全部拖入表单，如图 10－22 中左图所示。

注意　如要将表的全部字段拖入表单，最方便的方法是将光标移动到数据环境中表的“字段”图标上，将该图标拖入表单即可。图 10－22 中右图给出了这种拖动的结果。也可使用快速表单。

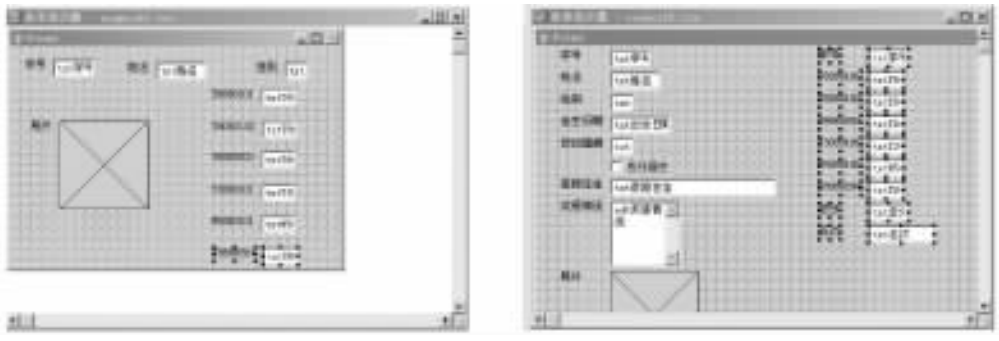


图 10－22　直接拖入法添加控件

以上的两个表单都可以运行，但和快速表单一样，由于没有按钮组，因此只能显示一条记录的结果。

2. 标准添加法

最常用的控件添加方法，是使用表单控件工具栏。表单控件工具栏列出了所有的表单控

件，供用户选定。图 10－23 给出了 Visual FoxPro 9.0 表单控件工具栏的全部控件图标。按图标自左向右的次序，表 10－9 给出了表单工具栏所有控件对象的名称和功能。

图 10－23　表单控件工具栏控件图标

在表单设计器中还可以使用调色板对各种表单中控件进行着色，用布局工具栏调整控件在表单中的位置。

利用表单控件工具栏，可以方便地向表单中添加控件。使用标准添加法，分为常用控件的添加法和特殊控件的添加法。

(1)常用控件的添加法

常用控件指除 ActiveX 控件和自定义可视类之外的控件。向表单中添加常用控件的步骤如下：

S1：打开表单设计器，并打开表单控件工具栏。

S2：选定控件。在控件工具栏单击需要添加的控件。

S3：添加控件。将鼠标移到表单的适当位置，单击，则选定的控件就被添加到了表单中。

例 10.7　首先创建一个空表单“bd2.scx”，然后向其中添加一个标签控件。

步骤如下：

S1：CREATE FORM bd2

S2：打开控件工具栏。

S3：选定标签控件。单击表单控件工具栏中的标签控件图标。

S4：添加标签控件。将光标移动到表单中，此时光标变为“＋”字，在适当位置单击表单即可。此时标签控件已添加到了表单中，而鼠标也恢复了原来的形状。

表 10－9　表单控件工具栏控件对象功能一览表

按钮	作用
选定对象	移动和改变控件的大小，取消对其他控制对象的选取状态
查看类	用来选择显示一个已注册的类库，当选择一个类库后，工具栏只显示选定类库中类的图标
标签	主要用来显示文本，在运行时可以通过对方法的调用来动态地更改文本
文本框	用于显示并编辑字段或内存变量及其他部分所需的信息
编辑框	用来编辑较长数据，在备注型字段中放置说明等
命令按钮	用来创建单个命令按钮，通常用于触发一个事件，如移动记录指针等
命令组	创建一组命令按钮，可以作为单个或者一组操作
选项组	创建一组单选按钮，让用户从中选择一个按钮，圆中有点则表示选中
复选框	设置复选框，让用户根据需要自行指定，框中显示“√”则选中，为真；框中空白则未选中，为假

续表 10－9

按钮	作用
组合框	设置下拉式组合框和下拉列表框
列表框	显示一系列数据项，比组合框多了一个多项选择功能
微调控件	用来接受用户输入，单击微调控件的上箭头或下箭头及在微调框内输入一个数值，使用微调控件可以在一定的数值范围内进行选择
表格	按行和列显示数据容器对象。表格包含列作为其子对象，而列又可以包含标头对象及控件，表格中还可以再嵌入表格控件
图像	用来创建一个可以显示.bmp 图片的图像控件，响应事件，并且可以在运行时改变
页框	包含一个或多个页面的容器对象，各页面还可以包含控件
计时器	在程序中加入计时器进行后台处理，例如截取系统时钟或者在一定时间间隔内执行某种操作等
ActiveX 控件	用于向表单中添加 OLE 对象。OLE 容器对象包括 OLE 控件(.ocx 文件)、Windows 应用程序，如 Word 和 Excel 创建的可插入 OLE 对象
ActiveX 绑定控件	OLE 对象与表的通用型字段相连接
线条	用于在表单上画各种类型的线条
形状	用于在表单上画矩形、圆角矩形、正方形、圆角正方形、椭圆或圆等各种类型的形状
分隔符	在创建定制工具栏时，在工具栏的控件之间加上空格
超链接	链接到一个指定的对象上
生成器锁定	为添加到表单上的控件自动打开一个生成器
按钮锁定	按下该按钮后，可以多次添加同种类型的多个控件，而不需多次按压

注意　如果想一次选定某个控件后，向表单中添加多个该控件，可在控件工具栏双击该控件图标(或单击控件图标后再单击按钮锁定图标)，使该控件被锁定，然后到表单中随意添加。当添加完成后，再单击按钮锁定图标进行解锁，使光标恢复正常。

(2)ActiveX 控件的添加方法

在表 10－9 中已说明 ActiveX 控件的作用是向表单中添加 OLE 对象。OLE 容器对象包括 OLE 控件(.ocx 文件)、Windows 应用程序，如 Word 和 Excel 创建的可插入 OLE 对象。所以，用户必须首先将 ActiveX 控件添加到表单控件工具栏，然后才能像添加标准控件一样，将它们添加到表单中。

将 ActiveX 控件添加到表单控件工具栏的步骤如下：

S1：打开表单设计器。

S2：打开“选项”对话框。→[工具]↓ 选项，如图 10－24 所示。

S3：打开“选定”复选框。→[控件]→“ActiveX 控件”↓ 选定。

S4：选定 ActiveX 控件。在“选定”复选框中，选择所需要的 ActiveX 控件，例如选定“Microsoft Graph 图表”，如图 10－25 所示。

图 10-24 选项对话框

图 10-25 选项对话框中的控件选定复选框

S5:返回表单设计器。→[确定]。

S6:将 ActiveX 控件添加到表单控件栏。在表单的控件工具栏:→[查看类]→[ActiveX 控件],则选定的 ActiveX 控件被添加进了表单控件工具栏。如图 10-26 所示。

图 10-26 添加了 ActiveX 控件的控件工具栏

(3)自定义可视类的添加方法

和 ActiveX 控件的添加步骤相似,要将自定义可视类添加到表单中,也必须先将它们添加到表单控件工具栏,然后才能继续添加到表单中。将自定义可视类添加到表单控件工具栏,其步骤如下:

S1:打开表单设计器和表单控件工具栏。

S2:打开表单控件工具栏的"打开"对话框。在表单控件工具栏中:→[查看类]→[添加]↓打开。

S3:选择自定义可视类,添加到表单控件工具栏。在"打开"对话框,选定需要添加的自定义可视类,例如选定了自定义类"Myclslib.vcx",单击[打开],则自定义可视类被添加进了表单控件工具栏。

(4)常用控件工具栏的再显示

当添加了特殊控件:ActiveX 控件或自定义可视类之后,表单控件工具栏将按图 10-26 的形式显示,而将常用控件隐藏,要显示常用控件,则应进行控件工具栏的转换,转换的方法是:在工具栏中,→[查看类]→[常用]。

10.6.2 设置控件的属性

设置控件的属性,和设置表单属性的方法相同,一般既可在表单设计阶段进行,也可在代码中编写。但也有些属性仅能在设计阶段或在代码编写中进行。本节讲解在表单设计阶段设置表单和控件的属性。

在设计阶段,无论是表单还是控件的属性设置,都是利用属性对话框进行的,无需赘述其

添加步骤，此处主要讲解属性对话框的组成。打开属性对话框，如图 10－27 所示。

图 10－27　属性对话框

从图 10－27 看出属性对话框由标题栏、组合框、页框三大部分组成。

· 标题栏：给出当前所设计的表单的名称。

· 组合框：以层次的形式给出当前所设计的表单中的所有控件。

· 页框：属性对话框的页框，由“全部”、“属性”、“方法程序”、“布局”、“其他”、“常用”六个选项卡组成。

如要设置某个控件的属性，可先在组合框中选定此控件，然后在页框中选“全部”或“属性”卡，再在下面的属性列表框中选中要设置的属性。若该属性可以在设计阶段设置，则在页框栏的下面会出现一个文本框或组合框，供输入属性值。

例 10.8　创建表单：db3.scx，使其具有三个标签控件，表单和标签控件的属性分别如表 10－10，表 10－11 所示。

S1：MODIFY FORM bd3。

S2：设置表单各属性如表 10－10 所示。

表 10－10　db3.scx 表单属性

属性名称	属性值	属性名称	属性值
Height——高	384	Caption——标题	学生成绩管理系统
Width——宽	512	Icon——图标	Cab.ico
BackColor——背景色	0,0,255	Picture——图片	xy.jpg
AutoCenter——自动居中	.T.	其他属性	使用缺省值

S3：添加三个标签控件，并设置各标签控件属性如表 10－11 所示。

表 10－11　db3.scx 表单控件的属性

属　性	控　件		
	Lable1	Lable2	Lable3
Caption——控件标题	学生成绩管理系统	2006 年 9 月	教务处
AutuSize——自动大小	.T.	.T.	.T.
ForeColor——前景色	255,0,0（红色）	0,255,0（绿色）	255,0,255（洋红）
FontName——字体名	华文新魏	楷体	
BackColor——背景色			255,255,0（黄色）
BackStyle——背景样式	0－透明	0－透明	1－不透明
FontSize——字符大小	50	20	16
Top——顶边界	220	490	560
Left——左边界	264	460	480
其他属性	使用缺省值	使用缺省值	使用缺省值

S4：存盘，运行表单：bd3，结果如图 10－28 所示。

图 10－28　添加了三个标签控件的表单 bd2.scx 运行结果

10.6.3　添加表单及控件事件的代码

事件和方法程序不同，用户必须编写事件代码，该事件被激活后方可根据代码而运行。前面在介绍表单的事件和方法程序时，已简单介绍过代码的添加过程。对于表单和控件，其事件的代码添加步骤是相同的。本节主要介绍代码编辑窗口的打开方法。

代码编辑窗口有如下四种打开方法：

·方法 1：使用系统菜单“显示”。其步骤为：打开表单设计器，→［显示］→［代码］↓ 代码编辑窗口。

·方法 2：使用快捷菜单。其步骤为：右击表单→［代码］↓ 代码编辑窗口。

·方法 3：使用系统工具栏“代码窗口”按钮。

·方法 4：双击表单或控件。

例 10.9　修改表单：db3.scx，为其添加一个按钮控件 Command1，其 Caption 属性设置为“退出”，位置、大小属性适当即可。添加按钮控件的代码：Thisform.Release。

S1：MODIFY FORM db3。

S2：添加命令按钮 Command1 到表单的合适位置。

S3：在属性对话框中，将 Caption 属性由“Command1”改为“退出”；设置它的其他属性如下：

AutuSize：.T.

FontName：楷体

FontSize：18

ForeClolor：255，128，0

BackColor：128，0，64

S4：双击按钮图标，打开代码编辑窗口，在：Command1.Click 代码窗口写入：

ThisForm.Release

S4：存盘。

S5：DO FORM bd3。

结果如图 10－29 所示，如果用户单击表单中的[退出]按钮，将结束表单的运行而返回 Visual FoxPro 9.0 界面。

图 10－29　添加了命令按钮及代码的 bd3 运行结果

另外，对于表单，用户也可以添加自己的自定义属性及方法程序，过程与添加类的属性和方法程序相同，它们都通过系统菜单“表单”进行。

10.6.4　单文档与多文档界面

Visual FoxPro 9.0 允许创建单文档界面(SDI)和多文档界面(MDI)两种类型的应用程序。

1. 单文档界面应用程序

单文档界面 SDI(single document interface)应用程序，它由一个或多个独立的窗口组成，这些窗口均在 Windwos 桌面上单独显示。

由单个窗口组成的应用程序通常是一个 SDI 应用程序。

2. 多文档界面应用程序

多文档界面 MDI(multi-document interface)应用程序，由单一的主窗口组成，而且应用程序窗口包含在主窗口中或浮动在主窗口顶端。

Visual FoxPro 9.0 基本上是一个 MDI 应用程序，它带有包含于 Visual FoxPro 主窗口中的命令窗口、编辑窗口、设计器窗口等。

3. Visual FoxPro 9.0 的表单类型

为了支持 SDI，MDI 两种应用程序，Visual FoxPro 9.0 允许创建三种类型的表单。

(1)子表单

包含在另一个表单中的表单称为子表单。子表单的特点是不允许移动到父表单的边界之

外;当其被最小化时,将显示在父表单的底部;如父表单被最小化,则子表单也跟着最小化。子表单来创建 MDI 应用程序。

(2)浮动表单

浮动表单属于父表单的一部分,但却并不包含在父表单中。浮动表单的特点是:它可以被移动到屏幕的任何位置,但不能在父窗口后台移动;当其被最小化时,将显示在桌面的底部;如父表单被最小化,则浮动表单也跟着最小化。浮动表单也用来创建 MDI 应用程序。

(3)顶层表单

没有父表单的独立表单称为顶层表单,与其他的 Windows 应用程序同级,可出现在其前台或后台,并且显示在 Windows 任务栏中。顶层表单用来创建 SDI 应用程序,或用作 MDI 应用程序中其他子表单的父表单。

4. 表单的创建

三种类型的表单的创建方法与其他表单的创建方法基本相同,但需要设置表单的一些特殊属性,以确定它将如何进行工作。

如果创建的表单是子表单,则不但要指定它要在哪个表单中显示,而且要指定它是否为 MDI 类应用程序的子表单,即指定表单最大化时的工作方式。如果子表单是 MDI 类的,则它会包含在父表单中,并与父表单共享标题栏、标题、菜单和工具栏。非 MDI 类的子表单最大化时,将占据父表单的全部用户区域,但仍保存本表单的表和标题栏。

(1)父表单和顶层表单的创建步骤

S1:打开表单设计器及属性对话框。

S2:将表单的显示窗口属性:ShowWindow 设置为:2 -作为顶层表单。

例 10.10 修改表单:db3.scx,将其设置为顶层表单。

S1: MODIFY FORM bd3

S2: 将 ShowWindow 属性设置为 2 -作为顶层表单,如图 10 - 30 所示。

图 10 - 30 ShowWindow 属性的设置

S3:存盘,运行修改后的 db3,结果如图 10 - 31 所示。

图 10－31　顶层表单的运行结果

比较图 10－29 和图 10－31 可以清楚地看出在修改前，由于表单 bd2.scx 的属性 ShowWindow 设置为“0－在屏幕中(默认)”，而修改后将它设置为了“1－作为顶层表单”，因此修改前后的运行结果中，前者表单位于 Visual FoxPro 窗口中，而后者则占据了整个屏幕。

(2)子表单的创建步骤

S1：打开表单设计器及属性窗口。

S2：将该表单的显示窗口属性 ShowWindow 设置为“0－在屏幕中(默认)”或“1－在顶层表单中”。

·0－在屏幕中(默认)：其作用是指定子表单的父表单就是 Visual FoxPro 9.0 的主窗口，因此会有例 10.9 的运行结果，如图 10－29 所示。

·1－作为顶层表单：其作用是指定当子表单显示时，子表单的父表单是活动的顶层表单，如果希望子表单出现在顶层表单而不是出现在 Visual FoxPro 的主窗口中，则应选择该属性值。

注意　如果希望子表单在最大化时与父表单组合成一体，应将表单的 MDIForm 属性设为：.T.；如果希望子表单在最大化时，仍保留为一个独立的窗口，则应将 MDIForm 设为：.F.。

(3)创建浮动表单的步骤

S1：执行创建子表单的前两步。

S2：将表单的 Desktop 设置为：.T.。

例 10.11　修改表单：db3.scx，将其设置为顶层表单，若后面有子表单，则 db2.scx 作为活动的顶层表单。

S1：MODIFY FORM bd3。

S2：将 ShowWindow 属性设置为：1－作为顶层表单。

S3：设置属性 Desktop 为：.T.。

S4：存盘，运行。

10.6.5 创建参数表单

表单的调用和过程的调用相类似，有不带参数的表单调用和带参数的表单调用之分。前面介绍的都是不带参数的表单调用，它只能使表单完成一种功能。当需要一个表单具有多种功能时，则需要通过参数传递的方法来调用表单。

要使表单能够在运行时接收参数，可参考下面的步骤实现。

S1：创建一个新表单。

S2：在表单的 Init 事件中添加 Parameters 语句，其格式和过程中的 Parameters 相同，作用是用来接收表单运行命令传递来的参数。该语句格式为：

Parameters *dummyParameterList*

ThisForm.*Object*.*Property*=*dummyParameter*1

……

S3：使用如下格式调用表单：

DO FORM *form* WITH *realParameterList*

例 10.12 保持表单：db3.scx 的结构，但是其在运行中能根据表 10－12 给出参数值变换其显示内容。

表 10－12 db3.scx 表单运行中传递的参数表

运行	THISFORM.Picture	控件属性		
		Lable1.Caption	Lable2.Caption	Lable3.Caption
第一次	jxzl.jpg	学生成绩管理系统	2006 年 9 月	教务处
第二次	tsdl.jpg	学生就业管理系统	2006 年 10 月	学生处
第三次	ysl.jpg	学生餐饮管理系统	2006 年 11 月	后勤处

S1：MODIFY FORM bd3

S2：将 THISFORM.Caption，THISFORM.Picture，THISROM.Lable1.Caption，THISFORM.Lable2.Caption，THISFORM.Lable3.Caption 全部置为空，但表单和控件的其他属性不改。

S3：编写表单 bd3.scx 的初始化事件 Init 代码如下：

```
PARAMETERS vb1,vb2,vb3,vb4
THISFORM.Picture=vb1
THISFORM.Caption=vb2
THISFORM.Label1.Caption=vb2
THISFORM.Lable2.Caption=vb3
THISFORM.Lable3.Caption=vb4
```

S4：存盘，并运行表单，运行时按如下格式逐次进行。

```
DO FORM bd3 WITH "jxzl.jpg","学生成绩管理系统","2006 年 9 月","教务处"
DO FORM bd3 WITH "tsdl,jpg","学生就业管理系统","2006 年 10 月"," 学生处"
DO FORM bd3 WITH "ysl.jpg","学生餐饮管理系统","2006 年 11 月","后勤处"
```

此时会看到，三次运行会有三个内容不同的表单，如图 10－32(1)，10－32(2)，10－32(3)所示。

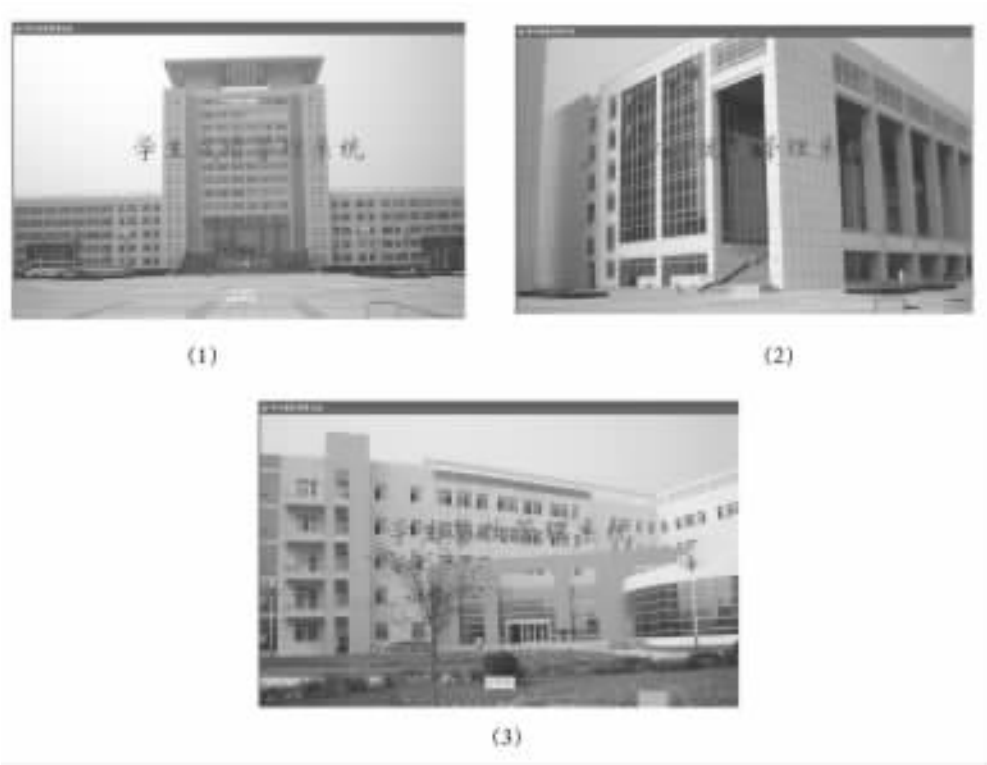


图 10－32　带有参数的表单调用结果

10.6.6　从表单中返回值

从被调用表单中返回值和从被调用的过程中返回值相类似，应在被调表单的卸载事件 UnLoad 代码中添加 RETURN 语句。其参考步骤如下：

S1：打开表单设计器和属性对话框，将 WindowsType 属性设置为 1－模式表单。

S2：在表单的 UnLoad 事件中添加：RETURN *expression* 语句，以便返回信息。

S3：用下面的格式调用表单：

DO FORM form TO variable

例 10.13　创建一个新表单：bd4.scx，它与表单：db2.scx 同结构，运行的结果是分别显示例 10.12 基本相同的三个画面。不同处是在第 1，2 张画面中，按钮标题显示为“下张画面”、在第 3 张画面中，按纽标题显示为“退出系统”。要求只能使用参数传递和参数返回方式实现。

S1：COPY FILE bd2.* TO bd4.*。

S2：MODIFY FORM db4。

S3：设置 WindowsType 属性为 1－模式表单。

S4：编写 bd4.scx 的事件代码。

S4－1：Load 事件代码：

```
    PUBLIC c && 表单运行次数标识变量
    PUBLIC xy
S4-2:改写 Init 事件代码:
    PARAMETERS vb1,vb2,vb3,vb4,v5 && 接收由 DO 调用传递给表单的参数
    thisform.Picture=vb1
    thisform.Caption=vb2
    thisform.Label1.Caption=vb2
    thisform.Label2.Caption=vb3
    thisform.Label3.Caption=vb4
    c=v5
    IF c<=2
        thisform.command1.Caption="下张画面"
    ELSE
        thisform.command1.Caption="退出系统"
    endif
S4-3:UnLoad 事件代码:
    Do case
      Case c= 1
        St=["tsdl.jpg","学生就业管理系统","2006 年 10 月","学生处"]
      Case c= 2
        St=["ysdl.jpg","学生餐饮管理系统","2006 年 11 月","后勤处"]
      Endcase
      Return st
S5:Modify Command yxbd
* 程序内容如下:
    cs= ["jxzl.jpg","学生成绩管理系统","2006 年 9 月","教务处"]
    DO FORM bd4 WITH &cs, 1 TO cs1
    DO FORM bd4 WITH &cs1, 2 TO cs2
    DO FORM bd4 WITH &cs2, 3
    RETURN
S6:DO yxbd
```

结果将与例 10.12 相似,惟一不同处就是按钮的标题。

10.7 表单集

表单集是表单的集合。在实际应用程序开发中,经常会碰到一个界面中使用多个表单的情况,此时就需要创建一个表单集来统一控制和管理这些表单的位置和功能。

10.7.1　表单集的创建

创建表单集的步骤如下：

S1：创建一个表单，例如 Form1，此时 Visual FoxPro 的系统菜单中将自动添加“表单”菜单。

S2：→[表单] →[创建表单集]，则表单集创建完成。

10.7.2　向表单集中添加表单

向表单集中添加新表单的方法是：→[表单] →[添加新表单]。图 10－33 就是一个添加了五个表单的表单集。

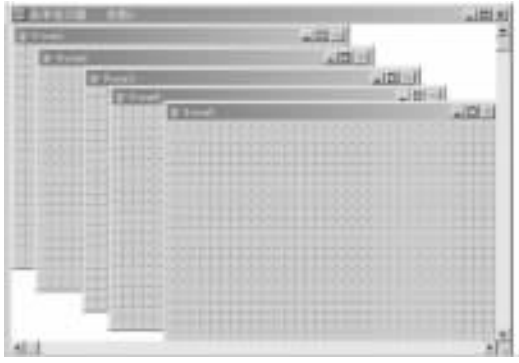


图 10－33　表单集

10.7.3　从表单集中移去表单

从表单集中移去表单的方法是：选中表单→[表单] →[移去表单]。

10.7.4　表单集中对象的引用结构

在 Visual FoxPro 中，表单集也是以表单文件而存储的，它的扩展名也是 .scx。在引用表单集中不同表单的某个对象时，必须在表单名前加上 ThisFormSet 的关键字。例如假设在表单 Form5 中有一个命令按钮控件 Command1，今要将其标题属性设置为“按钮 1”，则正确的引用方法为：

ThisFormset.Form5.Command1.Caption＝“按钮 1”

10.7.5　表单集的数据环境

不管同一表单集中包含多少个表单，它们都使用同一数据环境对象，用户在进行程序开发时务必高度注意。

10.8 思考与练习

一、选择题

1. 下列关于属性、方法和事件的叙述中，哪个是错误的(　　)。

A) 属性用于描述对象的状态，方法用于描述对象的行为

B) 基于同一个类产生的两个对象可以分别设置自己的属性值

C) 事件代码也可以像方法一样被显式调用

D) 在新建一个表单时，可以添加新的属性、方法和事件

2. 下面关于表单控件基本操作的陈述中，不正确的是(　　)。

A) 要在“表单控件”工具栏中显示某个类库中自定义类，可以单击工具栏中的“查看类”按钮，然后在弹出的菜单中选择“添加”按钮

B) 要在表单中复制某个控件，可以按住 Ctrl 键并拖放该控件

C) 要使表单中所有被选控件具有相同的大小，可单击“布局”工具栏中的“相同大小”按钮

D) 要将某个控件的 Tab 序号设置为 1，可在进入 Tab 键次序交互式设置状态后，双击控件的 Tab 键次序框

3. 下面关于数据环境和数据环境中两个表之间关系的陈述中，正确的是(　　)。

A) 数据环境是对象，关系不是对象

B) 数据环境不是对象，关系是对象

C) 数据环境是对象，关系是数据环境中的对象

D) 数据环境和关系都不是对象

4. 决定表单为主表单还是子表单的属性是(　　)。

A) SindowState　　B) WindowType　　C) ShowWindow　　D) ShowTips

5. 要从表单 Myform 中返回值，以下操作正确的是(　　)。

A) 该表单必须是模式表单，且必须为其 INIT 事件编程

B) 该表单必须是无模式表单，且必须为其 UNLOAD 事件编程

C) 该表单必须是无模式表单，且必须为其 INIT 事件编程

B) 该表单必须是模式表单，且必须为其 UNLOAD 事件编程

6. 设表单内有一个文本控件 Text1 和一个命令按钮组 Cmgroup，Cogroup 中含有 Cm1、Cm2 两个命令按钮。今要在 Cm2 的某个事件中访问 Text1 的 Value 属性值，正确的语句应为(　　)。

A) THIS.PARENT.PARENT.Text1.value

B) THIS.THISFORM.Text1.value

C) PARENT.PARENT.Text1.value

B) THIS.PARENT.Text1.value

二、填空题

1. 表单的数据环境包括可与表单交互的_____、_____ 和_____。

2. 可使用_____和_____来设置对象的 Tab 次序。

三、上机题

1. 使用表单向导创建一个单表表单，该表单可用来管理“学生成绩表”中的记录。

2. 使用表单向导创建一个一对多表单，该表单可用来管理“学生情况表”和“学生成绩表”中对应的记录。

3. 使用表单设计器创建顶层表单、浮动表单和子表单。

4. 创建一个可以接受运行参数并且能够返回值的表单。

第 11 章　表单控件的使用

在 Visual FoxPro 9.0 中，表单是应用程序界面的载体，而表单内所包含的控件，则是用来实现操作、美化、数据显示等功能的基本元件。

Visual FoxPro 9.0 提供了 21 种控件供设计表单时使用，这些表单控件根据其功能大致可以分为：输出显示类、输入类、控制类、容器类、连接类五大类控件。

11.1　输出显示类控件

输出显示类控件包括标签 Label、图像 Image、线条 Line、形状 Shape 共四种控件。

11.1.1　标签(Label)控件

标签控件是一种能在表单上显示文本信息的输出控件，常用来作为其他控件的提示或说明，也用来显示在程序运行中的文本输出信息。两者不同之处在于：作为其他控件的提示或说明时，其显示在表单运行期间信息不变；作为程序运行中的输出文本信息，其显示在表单运行期间却是可变的。在上一章的快速表单中，对每一个字段所生成的一个名称说明，就都是显示信息不变的标签控件，而带有参数返回的表单调用中的标签控件就是显示文本可变的标签控件。

标签控件常用属性如表 11－1 所示。

表 11－1　标签控件的常用属性

属性名称	功能说明	取值范围
AlignMent	指定与对象相关联的文本的对齐方式	0－左对齐(默认)，1－右对齐，2－居中
AutoSize	指定是否自动调整控件大小以容纳控件内容	.F.（默认），.T.
BackStyle	指定对象的背景是否透明	0－透明，1－不透明(默认)
Caption	指定对象的标题文本	字符串
FontBold	指定文字是否为粗体	.F.（默认），.T.
FontItalic	指定文字是否为斜体	.F.（默认），.T.
FontSize	指定对象文本的字符大小	自然数，缺省为 9
FontName	指定显示文本的字体名称	字库中的所有字体，缺省为宋体
ForeColor	指定显示对象中文本和图形的前景颜色	RGB(0,0,0)～RGB(255,255,255)

续表 11－1

属性名称	功能说明	取值范围
Height	指定屏幕上一个对象的高度	数值 AutoSize 为.T. 时无效
Left	指定对象的左上角相对于父对象的横坐标	数值
Top	指定对象的左上角相对于父对象的纵坐标	数值
Name	指定在代码中用以引用对象的名称	数值
Width	指定对象的宽度	数值 AutoSize 为.T. 时无效
WordWrap	指定控件中的对象是否沿纵向或横向扩展	AutoSize 为真时有效:.F.（默认）,.T.

需要指出的是,这里列出的许多属性,在其他控件中也常用到,以后无须再赘述。

标签控件在表单运行中是不能获得焦点的,因此它显示的内容,即 Caption 属性的值在表单运行期间无法进行交互式编辑。关于标签控件属性的设置可参考上一章中的几个例子学习和理解。

11.1.2　图像(Image)控件

图像控件是一个可显示图片的控件。Visual FoxPro9.0 支持多种类型的图像文件。诸如:.bmp,.ico,.gif,.jpg,.cur,.ani,.tiff,.png,.wmf,.emf 等类型的文件均可。除高、宽、左、顶、背景样式等通用属性外,图像控件还有:Picture,Stretch,BorderStyle 三个常用属性。它们如表 11－2 所示。

表 11－2　图像控件的主要常用属性

属性名称	功能说明	取　值
Picture	指定显示在控件上的图形文件或通用字段	对于图形文件,直接写入其路径和文件,也可浏览图形文件,选中后单击[打开]
BorderStyle	指定对象的边框样式	0－无(默认)、1－固定单线
Stretch	指定如何对图像进行尺寸调整以便放入控件中	0－裁剪(默认)。裁掉图像大于控件边框的部分 1－等比填充。调整图像大小以适应控件边框大小,且保持图像原比例不变 2－变比填充。调整图像大小以适应控件边框大小,但不保持图像原比例不变

例 11.1　设计一个表单:bd5.scx,含有一个图像控件。在设计阶段,图像控件的 Picture 属性置为空,BorderStyle 属性置为 1,Stretch 属性置为 2,其他属性适当。表单的作用是当程序运行而循环调用表单时,分别显示有关的照片文件 n.jpg,(n=1,2,3,…,10)。

S1:CREATE FORM bd5

S2:添加一个图像控件 Image1 到表单 bd5 中,并用属性对话框设置其:Picture 为空,BorderStyle 为 1,Stretch 为 2,位置尺寸适当。

S3:添加一个按钮控件,标题为下张照片。

S4:编写事件代码。

S4-1:表单的 Init 事件代码:

```
Parameters zp
THISFORM.Caption="显示照片"+ ALLTRIM(STR(VAL(zp)))
THISFORM.Image1.Picture=zp
THISFORM.Activate
```

S4-2:表单的 ACTIVATE 事件代码:

```
Inkey(3)
THISFORM.Release
```

S5:编写程序 xszp.prg

```
n=10
FOR i=1 TO n
zpbh=ALLTRIM(STR(i))
zpmc= zpbh+".jpg"
DO FORM bd5 WITH zpmc
ENDFOR
RETURN
```

S6:DO xszp

会看到在当前路径下,文件名为:"1.jpg"~"10.jpg"照片会依次显示在屏幕上。只要将程序 xszp.prg 中的 N 值修改,就会显示出 N 张照片,只要这些照片的文件名是:"n.jpg"(n=1,2,3…,N)的形式给出即可。其中的第 8 张照片如图 11-1 所示。

图 11-1 xszp.prg 运行的显示结果之一

11.1.3 线条(Line)控件

线条 Line 控件是一个图形控件。这里一定要区别开图像控件和图形控件,图像控件操作的对象是"像",它并不是用户现场画出来的,而是事先已作好且存在允许类型的文件中,例如

.bmp,.jpg 等文件中。而图形控件操作的对象则是“形”,在表单运行中通过该类控件现场可制作需要的图形来,例如画线、画几何图形等。

使用线条控件 Line,可在表单的任意两点,画出一条直线。

线条控件的常用属性如表 11－3 所示。

表 11－3　线条控件的主要常用属性

属性名称	功能说明	取　值
LineSlant	指定直线的倾斜方向	“\”左上角到右下角,“/”右上角到左下角
BorderWidth	指定线条的宽度(单位像素)	数值
BorderStyle	指定线条的样式	0－透明线、1－实线(默认)、2－虚线、3－点线、4－点划线、5－双点划线、6－内实线
Height	指定以线条为对角线的矩形的高度	数值,取 0 表示画一条水平线
Width	指定以线条为对角线的矩形的宽度	数值,取 0 表示画一条竖直线

11.1.4　形状(Shape)控件

形状 Shape 控件也是一个图形控件。使用它,可在表单上画矩形、正方形、椭圆、圆等几何图形。其主要作用是美化表单界面,也可以用来作为其他控件的背景。

形状控件的常用属性如表 11－4 所示。

表 11－4　形状控件的主要常用属性

属性名称	功能说明	取　值
BackColor	指定形状控件的背景颜色	RGB(0,0,0)～RGB(255,255,255)
Curvature	指定形状控件角的曲率	0～99,0 为直角(Height＝Width 时为正方形)、99 为椭圆(Height＝Width 时为圆形)
FillStyle	指定形状控件的填充类型	0－实线、1－透明(默认)、2－水平线、3－竖直线、4－向上对角线点划线、5－向下对角线点划线、6－交叉线、7－对角交叉线
Height	指定对象的高度	数值
Width	指定对象的宽度	数值

例 11.2　创建一个表单:bd6.scx,在它上面添加三个标签框、一个图像控件、一条横线和一个圆。图像为一簇鲜花、圆为一轮皓月。

S1:CREATE FORM bd6

S2:向表单添加三个标签、一个图像、一个线条、一个形状控件。各控件位置、大小通过鼠标调整,标签控件的 AutoSize 均设为.T.,BackStyle 均设为 0－透明,其他主要属性如表11－5 所示。

表 11-5 bd6.scx 的控件主要属性

控件名称	控件属性	属性值	说明
Lable1	Caption	计算机系学生成绩管理系统	
	FontName、FontSize、ForeColor	"隶书"、48、RGB(255,0,0)	红色
Lable2	Caption	版本：1.0	
	FontName、FontSize、ForeColor	"楷体"、18、RGB(255,0,255)	洋红
Lable3	Caption	设计：陈小虎	
	FontName、FontSize、ForeColor	"仿宋"、24、RGB(255,0,255)	洋红
Line1	Height	0	横线
	BorderColor	RGB(0,255,0)	绿色
Shape1	BackStyle	1-不透明	
	BackColor	255,0,255	黄色
	BorderStyle	0-透明	边框式样
	Curvature	99	要画圆，应使 Height 与 Width 属性相等
Image1	Picture	10.jpg	自拍的一簇鲜花

设计结果如图 11-2 所示。

图 11-2 表单 bd6.scx 设计结果

S3：编写事件 THISFORM.Click 代码 ThisForm.Release。

S4：存盘，运行 bd6.scx。

11.2 输入类控件

输入类控件的作用是接受数据的输入。在 Visual FoxPro 9.0 中，输入类控件包括：文本框控件、编辑框控件、组合框控件、列表框控件、微调控件、复选控件等。

11.2.1 文本框(TextBox)控件

文本框(TextBox)控件是表单中最常用的控件之一。其作用是用来显示或供用户输入文本信息。使用文本框控件，可以编辑内存变量、数组或表的字段数据。它具有所有 Visual FoxPro 的标准编辑功能，例如文本的剪切、复制、粘贴等。

1. 文本框的属性

除诸如边界位置、大小尺寸通用属性外，文本框的常用属性如表 11－6 所示。

表 11－6　文本框控件的主要常用属性

属性名称	功能说明	取　值
Alignment	指定文本框中文本的对齐方式	0－左对齐、1－右对齐、2－居中、3－自动(默认)
ControlSource	指定与文本框建立联系的数据源(数据绑定)	表的字段、内存变量
Enabled	指定表单或控件能否响应由用户引发的事件	.T.（默认）、.F.
Format	指定一个控件的 Value 属性的输入输出格式(格式码)	同于表字段的输入格式码
InputMark	指定在一个控件中如何输入和显示数据(掩码)	同于表字段的输入掩码
IMEMode	判断缺省的 IME 状态是关闭/打开或自动	0－无控件(默认)、1－打开 IME、2－关闭 IME
ReadOnly	指定文本框中的数据是否只读	.F.（默认）、.T.
SelectOnEntry	指定当单击列表中的单元格或用 Tab 键移到该单元时，该单元是否被选定	.F.（默认）、.T.
Value	指定控件的当前的具体值	0－N 型、{}－D 型、无－C 和 L 型
Visible	指定运行中对象是否可见	.F.（默认）、.T.
PasswordChar	指定文本框输入时的占位字符	常用“＊”

2. 文本框生成器

利用文本框生成器可以在 Visual FoxPro 9.0 的引导下方便地设置文本框的常用属性。文本框生成器的使用步骤为：打开文本框生成器。右击文本框，在快捷菜单中选“生成器”，则文本框生成器打开，如图 11－3 所示。生成器有三个选项：格式、风格、值。现显示的是格式选项的各选项。

图 11－3　文本框生成器“格式”选项卡

在“格式”选项卡中，各种控件一目了然，用户也对它们的功能大部分比较熟悉。

(1)“数据类型”列表框

用于设置 Value 属性允许的数据类型：N，C，D，L。不同的数据类型，对于格式选项卡中的复选框选项不同。

(2)“复选框”选项功能

·运行时启用：设定表单运行时，该文本框是否可用。用于设置 Enabled 属性。

·只读：设定文本框中的数据是否为只读。用于设置 ReadOnly 属性。

·隐藏选择：指定当文本框失去焦点时，框中所选定的内容是否被取消。

·仅字母字符：指定只对字符型数据可用，用于设置 Format 属性为“A”。

·进入时选定：指定当非空文本框获得焦点时，文本框中的数据被选定。用于设置 Value 属性的值。

·显示前导零：只对数值项数据可用，以 0 字符补满数值型数据的前置空格。用于设置 Format 属性为“L”。

(3)“输入掩码”列表框

“输入掩码”列表框列出了各种输入掩码及其组合供用户选择，用于设置 InputMark 属性。

图 11-4、图 11-5 分别给出了“风格”、“值”选项卡的各控件项。

图 11-4 文本框生成的“风格”选项卡

图 11-5 文本框生成的“值”选项卡

在“风格”选项卡中可以设置特殊效果、边框、字符对齐方式。其中：

·“特殊效果”选项按钮：用于设置 SpecialEffect 属性，选“三维”相当于将该属性设置为：3D；选“平面”相当于将该属性设置为 Plain。

·“边框”选项按钮：用来设置 BorderStyle 属性。

·“字符对齐”列表框：用来设置 Alignment 属性，选择文本对齐方式。

在“值”选项卡中，“字段名”列表框用来设置文本框的 Control*Source* 属性。

注意 其他控件的设计器均与文本控件的设计器相似，其作用都是指导用户设置其属性，以后无需再讲解，用户完全可以自己学习。

3. 文本控件的常用方法程序

文本控件最常用的方法是设置焦点方法。当一个对象被选定时，则称该对象获得了焦点。文本框获得焦点的标志是文本框内出现了闪烁的光标；而命令控件获得焦点的标志则是命令

按钮上出现了虚线的方框。

对于控件对象，设置焦点的方法有三种。一是使用 Tab 键，将焦点逐步移动到需要的文本框上；二是鼠标单击控件对象；三是使用在程序代码中调用设置焦点方法程序 SetFocus。调用 SetFocus 方法程序的格式为：

Object. SetFocus

说明：

① 如果控件对象的 Enabled 或 Visible 属性为.F.，或者控件对象的 When 事件返回值为.F.，则对象不能设置焦点。因此要为它设置焦点，则事先必须设置它们为.T.。

② 对象一旦获得焦点，则任何输入都是针对该控件进行的。

例如：

THISFORM.Text1.SetFocus

其功能是为当前表单中的文本框 Text1 设置焦点。

4. 文本框的常用事件

文本框的常用事件有 GotFocus，LostFocus，InteractiveChange，Valid，When 等。

· GotFocus 事件：当对象获得焦点后触发。

· LostFocus 事件：当对象失去焦点后触发。

· InteractiveChange 事件：文本框中的值发生变化后触发。用于检测文本框中输入的数据的合法性。

· Valid 事件：当对象失去焦点前触发。在该事件中加入代码，可以用来判别文本框中输入数据的合法性，也可用来根据文本框的数据读取其他信息。

· When 事件：控件获得焦点前触发。如它返回的值为.T.，则该控件对象可获得焦点，否则不可。对于列表框，当用户单击某项，或通过箭头键使焦点在列表框中移动时触发。对于其他控件，当光标移动到控件上时触发。对于文本框，When 事件发生在 GotFocus 事件之前。

例 11.3　以表单 bd6.scx 为基础，创建一个表单 bd7.scx。要求：

(1)调整原有的控件位置。

(2)添加三个新的标签框、两个文本框、一个命令按钮。

(3)编写必要的事件代码，使得操作员姓名和密码最多输入三次，如正确，则显示“您是一个合法用户，欢迎使用本系统!”，如连输三次姓名或密码不对，则显示“您是非法用户，谢绝使用本系统”，三次之内，也应给出出错的相应信息。

(4)运行之。

S1：COPY FILE bd6.* TO bd7.*

S2：MODIFY FORM bd7

S3：调整原有控件的位置到新的合适位置。

S4：添加三个标签框 Label4，Label5，Label6；两个文本框 Text1，Text2；一个命令按钮 Command1，它们各自的主要属性如表 11-7 所示。

表 11－7　bd7.scx 的控件主要属性

控件名称	控件属性	属性值	说明
Lable4	Caption	操作员姓名	
	FontSize、ForeColor	18、RGB(0,255,0)	
Lable5	Caption	操作员密码	
	FontSize、ForeColor	18、RGB(0,255,0)	
Label6	Caption		
	FontName、FontSize、ForeColor	新魏、24、RGB(0,255,255)	显示信息用
Text1	Value		字符型数据
Text2	PasswordChar	*	字符占位符
	Value		
Command1	Caption	确定	

设计结果如图 11－6 所示。

图 11－6　bd7.scx 设计结果

S5：编写事件代码。

S5－1：Thisform.Load 事件：

```
PUBLIC c,czyxm,czymm,xm,mm
* c:输入次数计数器
* czyxm:操作员姓名
* czymm:操作员密码
* xm: 由文本框要输入的用户姓名
* mm:由文本框 2 要输入的用户密码
c=0
czyxm="student"
czymm="xxwbkj"
xm=""
mm=""
```

S5－2：ThisForm.Init 事件：

```
    thisform.text1.setfocus
    thisform.text1.Value=xm
    thisform.text2.Value=mm
```

S5-3：ThisForm.Text1.lostFocus **事件**：

```
    xm=This.value
```

S5-4：ThisForm.Text2.lostFocus **事件**：

```
    mm=This.value
```

S5-4：ThisForm.Command1.Click **事件**：

```
    c=c+1
    IF xm=czyxm AND mm=czymm
      ThisForm.label6.Caption="您是一个合法用户,欢迎使用本系统!"
      INKEY(3)
      ThisForm.Release
    ELSE
      IF c=3
        ThisForm.label6.Caption="您是一个非法用户,谢绝使用本系统!"
        INKEY(3)
        ThisForm.Release
      ELSE
        DO CASE
          CASE xm#czyxm AND mm=czymm
            ThisForm.text1.Value=""
            ThisForm.label6.Caption="操作员姓名有错误,请另输!"
            INKEY(2)
            ThisForm.label6.Caption=""
            ThisForm.text1.SetFocus
          CASE mm#czymm AND xm=czyxm
            ThisForm.text2.Value=""
            ThisForm.label6.Caption="操作员密码有错误,请另输!"
            INKEY(2)
            ThisForm.label6.Caption=""
            ThisForm.text2.SetFocus
          CASE mm#czymm AND xm#czyxm
            ThisForm.text1.Value=""
            ThisForm.text2.Value=""
            ThisForm.label6.Caption="操作员姓名和密码都有错误,请另输!"
            INKEY(2)
            ThisForm.label6.Caption=""
            ThisForm.text1.SetFocus
```

```
            ENDCASE
        ENDIF
    ENDIF
```

S6:存盘、运行。

在程序中,给出的 czyxm 为"student",czymm 为"xxbdkj"。如果操作员姓名输入正确,操作员密码输入错误,则运行结果如图 11-7 所示。

图 11-7 bd7.scx 运行结果之一

11.2.2 编辑框(EditBox)控件

编辑框可以说是一个功能较齐全的文本编辑器。它不但可以像文本框一样对数据进行剪切、复制、粘贴,而且可带有竖直滚动条。它可以用来编辑字符型的内存变量、数据元素、字段或备注型字段。它和文本框的主要区别在于:

(1)编辑框只能用于输入和编辑字符型数据,而文本框却可用于字符型、数值型、日期型、逻辑型四种数据。

表 11-8 编辑框控件的主要常用属性

属性名称	功 能 说 明	取 值
Text	包含的值是未设置格式的,与用户输入的文本相一致	设计时不用,运行时只读
AllowTabs	指定能否在编辑框中插入 Tab 键,而不是将光标移动到下一个控件。如允许,则应提示用户可用 Ctrl+Tab 将光标移到下一个控件	.F.(默认),.T.
ControlSource	指定与编辑框绑定的数据源	字符型内存变量、数组元素、字段;备注型字段
HideSelection	指定当控件失去焦点时,控件中选定的文本是否仍处于选中状态	.T.(默认),.F.
ScrollBars	指定控件是否具有竖直滚动条	0-无,1-垂直
SelStart	返回在文本输入区域内所选定的文本的起始点或指定插入点的位置	数值,0 为文本的首位置,设计时不用,运行时可读/写。
SelLength	返回已选定的或指定要选定的字符的数目	数值,设计时不用,运行时读/写
SelText	返回在文本区中选定的文本	设计时不用,运行时读/写

(2)编辑框可输入和编辑多段字符型数据,回车不能终止输入和编辑的进行;而在文本框中只能输入和编辑一段字符,回车将终止对文本框的操作。

(3)编辑框允许使用 PageUp、PageDown 键及竖直滚动条来浏览文本,文本框则没有此功能。

(4)在编辑框中,文本可自动换行。

编辑框的常用主要属性如表 11－8 所示。

图 11－8　bd8.scx 运行结果之一

例 11.4　设计表单:bd8.scx,为其添加两个编辑框和三个命令按钮。命令按钮 1 的作用是打开编辑框 1,并在其中显示一个文本文件的内容,然后在其中选中某一部分。命令按钮 2 的作用是在将编辑框 1 中选定的文本内容复制到编辑框 2 中并显示。命令按钮 3 的作用是退出表单。

分析:编辑框不能直接编辑文本文件。要编辑一个文本文件,应借助于一个临时表文件来进行。该临时表文件应包含有备注型字段,首先将文本文件的内容复制到备注字段中,然后将备注字段与编辑框相绑定。这样方可进行文本内容的编辑。编辑结束后将被指字段的内容存储到文本文件中,则可以实现利用编辑框对文本文件的编辑;如果将编辑的内容存入另一个备注字段,并将另一个编辑框与该字段绑定,则将在另一个编辑框显示复制的内容。今设一个临时表 lsb,有三个字段:C 型字段 wjm,用来存放文本文件的文件名; M 型字段 wjnr,用来存放文本文件的内容;M 型字段 fznr,用来存放复制的文本内容。

表单设计:

S1:CREATE FORM bd8

并将 bd8.scx 的标题 Caption 取名为“编辑框功能演示”。

S2:添加两个编辑框,三个命令按钮,标题分别设为打开、复制、退出。

S3:编写代码。

S3－1:Thisform.Load

```
PUBLIC xdzf
```

S3－2:ThisForm.Init

```
ThisForm.Command2.enabled=.F.
ThisForm.Command3.enabled=.T.
ThisForm.Command1.enabled=.T.
```

S3－3:ThisForm.Command1.Click

```
    CREATE CURSOR lsb(wjm c(60), wjnr m,fznr m)
    APPEND BLANK
    REPLACE lsb.wjm WITH GETFILE("txt") && 打开"打开"对话框
    IF EMPTY(lsb.wjm)                   && 如果在"打开"对话框选择了[取
                                        && 消],而未选中任何文本文件
      RETURN                            && wjm 字段为空,返回
    ENDIF
    APPEND MEMO lsb.wjnr FROM ALLTRIM(lsb.wjm ) ;
        OVERWRITE                       && 将文件内容写入备注字段
    ThisForm.Edit1.ControlSource="lsb.wjnr"
    * 将 Text1 与备注字段绑定, 用来显示选定的文件内容
    ThisForm.Edit1.setfocus
    ThisForm.Refresh
    ThisForm.command2.Enabled=.t.
S3-4:ThisForm.Text1.LostFocus
    xdzf=thisform.text1.SelText
S3-5:ThisForm.Command2.Click
    ThisForm.edit1.lostfocus
    REPLACE lsb.fznr WITH xdzf          && 将选中的内容写入备注字段
    ThisForm.edit2.setfocus
    ThisForm.Edit2.ControlSource= "lsb.fznr"
    * 将 Text2 与备注字段绑定,用来显示复制到 Edit2 的内容
    ThisForm.Refresh
S3-6:ThisForm.Command3.Click
    ThisForm.Release
```

S4:存盘,运行。

单击[打开],选中一个文本文件(如 zf.txt),并选中要复制的内容后结果如图 11-8 所示;单击[复制]按钮后结果如图 11-9 所示。

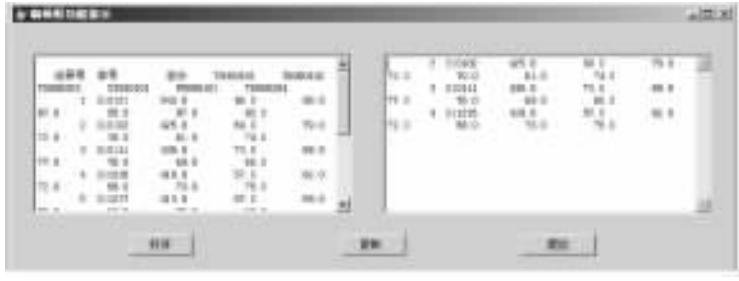


图 11-9 bd8.scx 运行结果之二

11.2.3　列表框(ListBox)控件

列表框提供了一个列表,让用户在该列表中选择数据。一般情况下,列表框会显示出若干条目,用户可以通过滚动条来浏览其他条目。

列表框的主要属性如表 11－9 所示。

表 11－9　列表框控件的主要常用属性

属性名称	功 能 说 明	取　值
RowSourceType	指定列表框或组合中条目的数据源的类型	0－无、1－值、2－别名、3－SQL 语句、4－查询(.qpr)、5－数组、6－字段、7－文件、8－结构、9－弹出式菜单(详见表 11－10)
RowSource	指定列表框或组合框条目的数据源	由 RowSourceType 决定
List	用以存取列表框或组合框中数据项的字符串数组	读列表框的第 3 条目：Var＝Thisform.myList.List(3) 将第 3 条目赋值为“OK”：Thisform.myList.List(3)＝″OK″
ListCount	获得列表框或组合中数据条目的数目	设计时不用,运行时只读
ColumnCount	指明列表框、组合框或表格的列数	数值
Value	返回列表框中被选中的条目	数值型:返回被选题条目在列表框中的次序号 字符型:被选条目的本身内容,如列表框不止一列,则返回由 BoundColumn 属性指明的列上的数据项
ControlSource	指定一个字段和变量,以保存从列表框中选择的结果	在列表框中与在其他控件中用法不同
Selected	指定列表框或组合框中的某个条目是否处于选中状态	设计时不用,运行时读/写,.F.(默认),.T.
MultiSelect	指定在列表框中能否进行多重选择	0 或.F.(默认),1 或.T.

在列表框中,最重要的属性是选项条目的数据源类型 RowSuorceType,它决定了条目数据源的属性选择。关于 RowSourceType 的取值的详细说明,由表 11－10 给出。

表 11－10 RowSourceType 属性的设置与数据源

属性值	列表项的数据源
0－无	缺省值。运行时可由 AddItem 或 AddListItem 方法程序将数据添加到列表中，通过 ReMoveItem 方法移去列表条目
1－值	通过 RowSource 属性手动指定列表条目。例如：RowSource＝"张学军，王国胜，李富强，赵爱国"
2－别名	将表的字段作为列表框的条目。ColumnCount 指定要取得字段的数目，即列表框的列数。指定的字段总是表中最前面的若干字段。如 ColumnCount 取为 0 或 1，则列表框将显示第一个字段的值
3－SQL 语句	在 RowSource 属性中包含一个 SELECT—SQL 语句。它执行的结果将作为列表框的条目的数据源 例如：RowSource＝"SELECT ＊ FROM czy INTO CURSOR myList"
4－查询(.qpr)	将查询文件(.qpr)执行的结果作为列表框的数据源。例：RowSource＝"myQuery.qpr"
5－数组	将数组中的内容作为列表条目的数据来源
6－字段	将表中的一个或几个字段最为列表框条目的数据源。例如：RowSource＝"myList.czyxm，czymm"
7－文件	将某文件作为列表框的条目。运行时用户可选择不同的驱动器和目录
8－结构	将表中的字段名作为列表框的条目，由 RowSource 实际指定表。如果 RowSource 属性值为空，则列表框显示当前表中字段名的清单
9－弹出式菜单	用一个先前已定义好的弹出式菜单作为列表框的数据源

例 11.5 以表单 bd7.scx 为基础创建 bd9.scx，将其文本框 Text1 改为列表框 List1，用于列表显示操作员的姓名。该姓名来自于表文件 czyqk.dbf。该表的结构为 czyqk (czyxm c(8)，czymm c(6))，它有四条记录，内容用户自定。表单的功能不变。

S1：创建 czyqk.dbf。

```
CREATE TABLE czyqk(czyxm c(8), czymm c(6))
INSERT INTO czyqk VALUES("张学军","123456")
INSERT INTO czyqk VALUES("王国胜","246135")
INSERT INTO czyqk VALUES("李富强","135246")
INSERT INTO czyqk VALUES("赵爱国","654321")
```

S2：复制 bd7.scx 为 bd9.scx。

```
COPY FILE bd7.* TO bd9.*
```

S3：打开 bd9.scx，并修改。

```
MODIFY FORM bd9
```

在 bd9.scx 中，删去文本框 Text1，添加列表框 List1。

S4：设置数据环境为：czyqk.dbf。

S5：设置 List1 的属性如下：

```
RowSourceType＝2          && 数据源类型为表
RowSource＝"czyqk.dbf"    && 数据源为表 czyqk.dbf
```

```
    ColumnCount=0                    && 选第 1 个字段作为数据源
```

S6：**编写、修改代码**。

S6－1：ThisForm.Load **代码**：

```
    PUBLIC c,xm,mm &&C:输入次数计数器
    C=0
    mm=""
```

S6－2：ThisForm.Text2.LostFocus **代码**：

```
    mm=This.Value
```

S6－3：ThisForm.Command1.Click **代码**：

```
    c=c+1
    IF mm=czyqk.czymm
      thisform.label6.Caption="您是合法用户,欢迎使用本系统!"
      INKEY(3)
      thisform.Release
ELSE
      IF c=3
      thisform.label6.Caption="您是一个非法用户,谢绝使用本系统!"
      INKEY(3)
      thisform.Release
      ELSE
      thisform.text2.Value=""
      thisform.label6.Caption="操作员密码有错误,请另输!"
      INKEY(2)
      thisform.label6.Caption=""
      thisform.Text2.SetFocus
      ENDIF
    ENDIF
```

S7：**存盘，运行。如在列表框中选“赵爱国”，在文本框中输入其密码，会有如图** 11－10 **所示的画面出现**。

图 11－10　**运行结果**

11.2.4 组合框(ComboBox)控件

列表框有一个致命的弱点,就是只能选择已有的条目而不能修改或输入新的条目。组合框就是为克服列表框的这一弱点而设计的另一种数据输入控件。

组合框(ComboBox)顾名思义是两种控件:列表框和文本框组合而成的一种新控件。它既具有列表框的属性、功能,又兼备文本框的属性、功能。

除无多条目选择属性 MultiSelect 外,上节介绍的列表框的其他各项属性,组合框均具有。

但组合框和列表框又有明显的区别。主要表现为:

① 列表框任何时候都可以显示多个条目,而组合框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。

② 组合框不提供多重选择功能。

③ 组合框具有两种形式:下拉式组合框和下拉式列表框。它们可以通过设置 Style 属性来选定。Style 属性由表 11-11 给出。

表 11-11 组合框控件的 Style 属性值与组合框类型

属性值	列表项的数据源
0	下拉组合框。用户既可以从列表中选择,也可在编辑区内输入。输入的内容从 Text 属性获得
1	下拉列表框。仅能从列表框中选择

例 11.6 以表单 bd9.scx 为基础创建 bd10.scx,将其列表框 List1 改为组合框 Combo1,用于列表显示操作员的姓名。表单的功能不变。

S1:复制 bd9.scx 为 bd10.scx。

```
COPY FILE bd9.* TO bd10.*
```

S2:打开 bd10.scx,并修改。

```
MODIFY FORM bd10
```

在 bd10.scx 中,删去列表框:List1,添加组合框:Combo1。

S3:设置数据环境为 czyqk.dbf。

S4:设置 Combo1 的属性如下:

```
Style=0                    && 下拉式组合框
RowSourceType=2            && 数据源类型为表
RowSource="czyqk.dbf"      && 数据源为表 czyqk.dbf
ColumnCount=0              && 选第 1 个字段最为数据源
```

S5:存盘、运行。

11.2.5 微调(Spinner)控件

微调控件主要用来接收一定范围的数据。它既允许由键盘输入数据,也允许通过使用微调控件所具有的向上/向下箭头对微调控件中的当前值进行增减操作。

微调控件的主要属性如表 11-12 给出。

表 11－12　微调框控件的主要常用属性

属性名称	功 能 说 明	取 值
KeyBoardHighValue	设定能输入到微调文本框的最高值	数值，应大于 KeyBoardLowValue
KeyBoardLowValue	设定能输入到微调文本框的最低值	数值，应小于 KeyBoardHighValue
SpinnerHighValue	设定单击向下按钮时，微调文本框微调框能显示的最高值	数值
SpinnerLowValue	设定单击向下按钮时，微调文本框微调框能显示的最低值	数值
Value	微调文本框的当前值	数值
Increment	单击向上/向下箭头时，微调文本框的增量	数值，缺省值为 1；设置为正时，单击向下按钮 Value 增加，单击向下按钮 Value 减少；设置为负时反之

微调控件的常用事件有两个：

① DownClick：单击向下箭头按钮时触发。

② UpClick：单击向上箭头按钮时触发。

例 11.7　创建 bd11.scx，用来显示和调整日期。

分析：微调框是不能用来直接显示日期型数据的，但将微调框与文本框相结合却可以用来调整多种类型的数据。本题的解决步骤如下：

S1：CREATE FORM bd11

S2：向 bd11.scx 中添加一个文本框、一个微调框。微调框的位置位于文本框的右侧，高度和宽度以仅能显示出上下两个按钮为宜，文本框和微调框高度一致。

设计结果如图 11－11(1)。

图 11－11　微调框与文本框应用实例

S3：编写代码：

S3－1：Thsiform.Load 代码：

```
SET DATE TO LONG                    && 为显示符合中国人习惯的日期数据而设
```

S3－2：Thisform.Text1.Init

```
This.Enabled= .F.
```

S3－4：Thisform.Spinner1.UpClick

```
Thisform.Text1.enabled=.t.
Thisform.Text1.Value= Thisform.Text1.Value+1
```

S3－3：Thisform.Spinner1.DownClick

```
Thisform.Text1.enabled=.t.
Thisform.Text1.Value= Thisform.Text1.Value-1
```

S4:存盘、运行。刚运行表单时,如图 11-11(2)所示,连续两次单击向上箭头按钮,会得到图 11-11(3)显示。如果又连续单击四次向下箭头,则显示的日期又会减少,结果如图 11-11(4)所示。

11.2.6 复选框(CheckBox)控件

一个复选框控件用于标识一个值的两种状态。为.T. 时表示选中,为.F. 时表示未选中。当处于选中状态时,复选框内显示√,未选中则复选框为空白。

复选框控件属于绑定型控件,常用来显示或保存表中的逻辑型字段。将复选框与表的字段绑定后,表的字段将有三种状态。

① 复选框未选中,表示记录的该逻辑型字段的值为假(.F.),复选框的值为 0,系默认值。

② 复选框选中,表示记录的该逻辑型字段的值为真(.T.),复选框的值为 1。

③ 复选框为灰色,表示记录的该逻辑型字段的值为空(.NULL.),复选框的值为 2。仅在代码中使用。

常用的复选框属性有 Caption,AutoSize,BackStyle,ControlSource,Enabled。

例 11.8 创建 bd12.scx,用来改变标签框标题的显示。

S1:CREATE FORM bd12

将其 Caption 设为:复选框的使用。

S2:添加一个标签框,标题:努力学习,奋勇前进!

S3:添加四个复选框,标题分别为华文新魏、斜体、粗体、下划线。

表单的设计结果如图 11-12 所示。

图 11-12 bd12 设计结果

S4:编写代码。

```
S4-1:Thisform.Check1.Click
    If This.Value=1
    Thisform.Label1.FontName="华文新魏"
    Else
    Thisform.Label1.FontName="宋体"
    Endif
S4-2:Thisform.Check2.Click
    If This.Value=1
    Thisform.Label1.FontItalic=.t.
    Else
    Thisform.Label1.FontItalic=.f.
    Endif
S4-3:Thisform.Check3.Click
    If This.Value=1
    Thisform.Label1.FontBold=.t.
    Else
```

```
    Thisform.Label1.FontBold=.f.
    Endif
S4-4:Thisform.Check4.Click
    If This.Value=1
    Thisform.Label1.FontUnderline=.t.
    Else
    Thisform.Label1.FontUnderline=.f.
    Endif
```

S5:**存盘、运行。四项全选,结果如图** 11-13 **所示。**

图 11-13　bd12 **运行结果**

11.3　控制类控件

在 Visual FoxPro 9.0 **中,控制类控件主要有命令按钮控件、命令组、单选按钮控件、计时器控件等。下面将分小节详细介绍这些控件。**

表 11-13　命令按钮控件的主要常用属性

属性名称	功 能 说 明	取 值
Caption	指定命令按钮的标题	字符串。可在标题字符串中通过加入“\<”,后跟一个字符的方法设置快捷键
Cancel	指定该命令按钮是否为“取消”按钮	.F.(默认) .T.,按 Esc 键将触发该按钮的 Click 事件
Default	指定该命令按钮为缺省命令按钮	.F.(默认) .T.,按 Enter 键触发该按钮的 Click 事件
Picture	指定命令按钮的图形文件	.ico,.bmp,.gif,.jpg 等
ToolTipText	指定命令按钮的提示文本	字符串
WordWrap	指定按钮标题折行显示(标题过长时使用)	.F.(默认),.T.
DisabledPicture	当按钮失效时显示指定的图形文件	.Bmp 文件
DownPicture	当按钮按下时显示指定的图形文件	.Bmp 文件
Style	指定按钮的样式	0-标准;1-不可见

11.3.1 命令按钮(CommandButton)控件

命令按钮控件是 Visual FoxPro 中最常用的一种控件。其功能是当用户单击、双击或右击命令按钮时触发对应的事件来完成特定的任务,前面我们已接触了许多使用它的例子。

1. 命令按钮控件的常用属性

命令按钮控件的属性除基本属性:AutoSize,Enabled,FontSize,FontColor,Visible 外,常用的其他属性如表 11-13 所示。

2. 命令按钮控件的常用事件

在 Visual FoxPro 9.0 中,命令按钮控件的常用事件如表 11-4 所示。

表 11-14 命令按钮控件的常用事件

事件名称	触 发 条 件
Click	鼠标单击命令按钮时触发
DblClick	鼠标双击命令按钮时触发
RightClick	鼠标右击命令按钮时触发
MiddleClick	鼠标中击命令按钮时触发
MouseDown	按下鼠标键时触发
MouseUp	释放鼠标键时触发
MouseMove	鼠标指针指在命令按钮上方时触发

例 11.9 创建 bd13.scx,用来逐条显示 xsqkb.dbf 的记录。

S1:CREATE FORM bd13

将其 Caption 设为命令按钮的使用。

S2:设置数据环境:xsqkb.dbf。

S3:按住“字段”图标,将其拖入表单后释放。

S4:修改表单,并添加五个按钮,标题分别为:第一条、下一条、上一条、最后一条、退出。

表单的设计结果如图 11-14 所示。

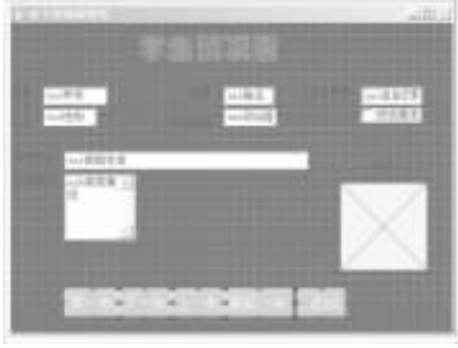


图 11-14 bd13 设计结果

S5：**编写代码**。

```
S5－1：Thisform.Command1.Click
        Go top
        This.Enabled=.F.
        Thisform.Command2.Enabled=.t.
        Thisform.Command3.Enabled=.f.
        Thisform.Command4.Enabled=.t.
        Thisform.Refresh
S5－2：Thisform.Command2.Click
        If Recno()<Reccount()
          Skip
        Thisform.Refresh
        Else
          This.Enabled=.f.
          Thisform.Command4.Enabled=.f.
        endif
          Thisform.Command1.Enabled=.t.
          Thisform.Command3.Enabled=.t.
S5－3：Thisform.Command3.Click
        If Recno()>1
          Skip －1
        Thisform.Refresh
        Else
          This.Enabled=.f.
          Thisform.Command1.Enabled=.f.
        endif
          Thisform.Command2.Enabled=.t.
          Thisform.Command4.Enabled=.t.
S5－4：Thisform.Command4.Click
        Go Bottom
        Thisform.Refresh
        This.Enabled=.F.
        Thisform.Command1.Enabled=.t.
        Thisform.Command2.Enabled=.f.
        Thisform.Command3.Enabled=.t.
S5－5：Thisform.Command5.Click
        Thisform.Release
S5－6：Thisform.Init
        Thisform.Command1.Enabled=.t.
```

```
Thisform.Command2.Enabled=.f.
Thisform.Command3.Enabled=.f.
Thisform.Command4.Enabled=.f.
```

S6:存盘并运行。当运行到最后一条记录时,显示结果如图 11-15 所示。

图 11-15 bd13 运行结果

11.3.2 命令按钮组(CommandGroup)

命令按钮组对象是一种容器,它可以包含多个按钮,并对这些按钮进行有效管理。命令按钮组和其中的按钮都有各自的属性、方法程序、事件,因而既可以单独操作各命令按钮,又可以作为一个组来统一操作。

例如:

```
Thisform.CommandGroup1.Visible=.F.
```

表示该命令按钮组不可见。

命令按钮组对象的常用属性有 AutoSize,ButtonCount,BackStyle,Height,Width 等,其中最重要的是 ButtonCount,用它可以设定命令按钮组中所包含的按钮的个数,其缺省值为 2。

例 11.10 以 bd13.scx 为基础创建 bd14.scx,在 bd14.scx 中用命令按钮组取代其中的五个按钮,bd14.scx 的作用同于 bd13.scx。

```
S1:COPY FILE bd13.* TO bd14.*
S2:MODIFY FORM bd14
```

S2-1:将其 Caption 设为:按钮组的使用。

S2-2:删除五个按钮。添加一个命令按钮组控件,将其 ButtonCount 属性设为 5,其他属性视其情况选取。

S2-3:为按钮命名,标题分别为:第一条、下一条、上一条、最后一条、退出。

S3:编写个按钮事件代码,代码与例 11.9 相同,但应在所有的涉及到按钮命令的前缀关键字:ThisForm 应改为:ThsiForm.CommandGroup1。例如,将 ThisForm.Command1.Enabled=.t.,改为:ThisForm.CommandGroup1.Command1.Enabled=.t.。

11.3.3 选项组(OptionGroup)

和复选框控件不同,复选框控件独立存在于表单中,可一次选择多项;而选项组对象是一个容器类对象,它里面可以包含多个选项,但一次却只允许选择一项,被选中的选项将以圆心

为黑点的圆显示，其他各未选的项均以白色的圆显示。

选项组对象最重要的两个属性是 ButtonCount 和 Value。

- ButtonCount 属性，用来设置选项组中按钮的数目，其缺省值为 2。
- Value 属性，用来标识用户选中的选项组中的某个选项按钮。如果是数值，则表示选中的选项按钮的索引号。例如：选中了第 3 个按钮选项，则 Value 属性为 3。如果选项组的 ControlSource 属性是表的一个字符型字段，或者在运行表单前将 Value 设置成了一个字符值，则运行中 Value 的值就是被选择的选项按钮的标题。

在表单运行中还可使用 Buttons 属性，并指定选项按钮的索引号来设置选项按钮的属性。例如：

```
ThsiForm.OptionGroup1.Value=0          && 选项按钮组中的所有按钮全部
                                       && 未选中
ThsiForm.OptionGroup1.Option1.Value=1  && 选项按钮组的第 1 按钮被选中
ThsiForm.OptionGroup1.Buttons(2).Value=1  && 选项按钮组的第 2 按钮被选中
```

例 11.11　以 bd14.scx 为基础创建 bd15.scx，在 bd15.scx 中添加一个含有五个选项按钮选项组控件，五个选项按钮各自的标题是：汉族、回族、藏族、满族、维吾尔族。表单的作用是为表 xsqkb3.dbf 录入学生的民族(xsqkb3.dbf 与 xsqkb.dbf 结构相似，仅多一个字符型字段：民族)。

S1：COPY FILE xsqkb.* TO xsqkb3.*

S2：ALLTER TABLE xsqkb3 ADD 民族 C(10)

S3：COPY FILE bd14.* TO bd15.*

S4：MODIFY FORM bd15

S4－1：将其 Caption 设为：选项组控件的使用。

S4－2：将表单的数据环境设为 xsqkb3。

S4－3：添加一个选项按钮组，将其 ButtonCount 属性设为 5，将它的 ControlSource 属性设置为 xsqkb3.民族，其他属性视其情况选取。

S4－4：为选项按钮命名，标题分别为汉族、回族、藏族、满族、维吾尔族。

S4－5：将表单中所有的文本框的 Controlsource 属性一律由 xsqkb.fieldname 改为 xsqkb3.fieldname。

设计结果如图 11－16 所示。

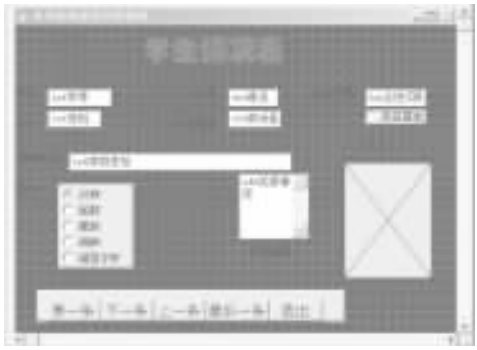


图 11－16　bd15 设计结果

S5:存盘、运行,通过选选项按钮组控件为各条记录录入民族字段。最后用 Browse 命令浏览 xsqkb3,结果如图 11-17 所示。

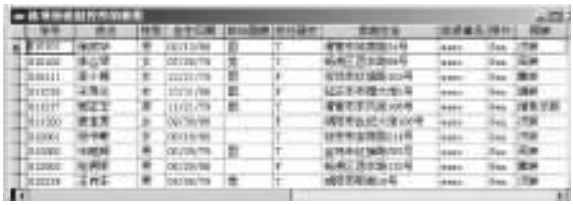


图 11-17 运行 bd17 或浏览表 xsqkb3 的结果

11.3.4 计时器(Timer)控件

计时器控件用来以用户给定的时间间隔触发自己的 Timer 事件来反复执行某种操作。它与用户的操作是相互独立的。计时器控件在表单运行期间本身并不可见。

计时器控件的最主要的属性是:Interval。它用来设置计时器的时间间隔,单位为毫秒(ms),取值范围:0～2 147 483 647(约合 596.6 小时,24.8 天)。由于系统时钟每秒钟中产生 18 次时钟跳动,因此 Interval 的真正精确度不超过 1/18 秒。

计时器控件最重要的事件有两个,它们是:

- Timmer:周期性被激发。
- Rest:重新设置计时器控制,使其从 0 开始。

例 11.12 设计一个运行时能在百花丛中翩翩起舞的一只蝴蝶的表单 bd16.scx。

S1:CREATE FORM bd16

S2: 设置各种对象的属性如表 11-15 所示。

表 11-15 bd16.scx 的控件主要属性

控件名称	控件属性	属性值	说明
Form1	Caption	计时器控件的使用	
	Picture	选一张百花盛开的照片	
Image1～Image2	Picture	bfly1,bfly2	均为.bmp 文件,由 Windows 提供,1 为开翼蝴蝶,2 为闭翼蝴蝶
Timer1	Interval	60	每秒可实现 16 次图像切换

设计结果如图 11-18 所示。

图 11-18 bd16 设计结果

S3：**编写事件代码**。

S3－1：Form1.Init

```
    Thisform.bfly1.Visible=.t.          && 开翅膀蝴蝶可见
    Thisform.bfly2.Visible=.f.          && 闭翅膀蝴蝶不可见
```

S3－2：Timer1.timer

```
if thisform.image1.visible=.t.          && 开翅膀蝴蝶可见闭翅膀蝴蝶不可见
                                        && 情况，计算闭翅膀蝴蝶的下一个位置
thisform.image2.top=thisform.image1.top+int(rand()*20)-10
                                        && 随机计算 bfly2 的顶边界位置
thisform.image2.left=thisform.image1.left+int(rand()*30)-15
                                        && 随机计算 bfly2 的左边界位置
if thisform.image2.top<150              && 预防蝴蝶飞出画面顶部
thisform.image2.top=150
endif
if thisform.image2.left<150             && 预防蝴蝶飞出画面左边界
thisform.image2.left=150
endif
thisform.image1.Visible=.f.             && 开翼蝴蝶变为不可见
thisform.image2.Visible=.T.             && 闭翼蝴蝶变为可见，实现了一次翅
                                        && 膀煽动
else                                    && 闭翅膀蝴蝶可见开翅膀蝴蝶不可见情况
                                        && 下，计算开翅膀蝴蝶的下一个位置
thisform.image1.top=thisform.image2.top+int(rand()*20)-10
thisform.image1.left=thisform.image2.left+int(rand()*30)-15
if thisform.image1.top<150
thisform.image1.top=150
endif
if thisform.image1.left<150
thisform.image1.Left=150
endif
thisform.image2.Visible=.f.             && 闭翼蝴蝶变为不可见
thisform.image1.Visible=.T.             && 开翼蝴蝶变为可见，实现又一次翅
                                        && 膀煽动
endif
thisform.refresh
```

S4：**存盘、运行，会发现一只彩蝶在百花丛中翩翩起舞，十分好看**。

11.4 容器类对象

Visual FoxPro 的容器类对象主要包括命令按钮组、选项按钮组、表格、页框和容器控件等。其中命令按钮组控件、选项按钮组又属于控制类对象的范畴，因而在前一节中已经介绍过，本节仅介绍表格控件、页框控件、容器控件。

11.4.1 表格(Grid)

表格和表是完全不同的两个概念。表是存储关系数据的文件，而表格是一个容器类对象。一个表格对象由若干列对象组成，每一个列对象又含有一个表头对象和一些其他控件。表格对象、列对象和表头对象都有各自的属性、事件、方法程序。

1. 表格对象的常用属性

在 Visual FoxPro 9.0 中，表格对象最常用的属性如表 11-16 所示。

表 11-16 表格对象的主要常用属性

属性名称	功 能 说 明	取 值
AllowAddNew	指定是否可以将表格中的新记录添加到表中	.F.（默认）,.T.
ColumnCount	指定表格对象中列对象的数目	数值，缺省值：-1，表格列出数据源中的所有字段
GridLine	指定是否显示表格中的网格线	0-无、1-水平、2-竖直、3-既水平又竖直（默认）
RecordSourceType	指定填充表格的数据源的类型	0-表：自动打开 RecordSource 指定的表 1-别名（默认）：按指定的方式处理数据源 2-提示：在运行时，向用户提示数据源 3-查询（Qpr）：将 RecordeSource 指定为一个查询文件 4-Sql 说明：将 RecordeSource 指定为一个 SQL 语句
RecordSource	指定于表格建立联系的数据源	表或临时表
ReadOnly	指定表格是否只读	.F.（默认）,.T.
ScrollBars	指定是否显示滚动条	0-无、1-水平、2-竖直、3-既有水平又有竖直（默认）
SplitBar	指定是否表格控件中显示拆分线	.T.（默认）,.F.
LinkMaster	如当前表为子表，则要指定其父表的关键字段	
ChildOrder	如当前表为父表，则指定其子表的索引名。如已设置了 ChildOrder 属性，则忽略子表、临时表的 Order 属性	

注意　LinkMaster 和 ChildOrder 属性用来连接两个有一对多关系的表。

2. 表格控件中列对象的常用属性

表格控件中，列对象最常用的属性有 ControlSource，CurrentControl，Sparse 属性。它们各自的功能和取值表 11－17 所示。

表 11－17　表格控件中列对象的主要常用属性

属性名称	功 能 说 明	取 值
ControlSource	指定表格中列对象的数据源	一般为表的一个字段
CurrentControl	指定列对象中的活动控件	Text1(默认)文本框，如在列对象中添加了另一个控件，则可指定它为 CurrentControl
Sparse	指定单元的数据是否全部按 CurrentControl 显示	.T.：仅列中的活动单元使用 CurrentControl 属性显示，其他单元仍以文本方式显示 .F.：列中的所有单元使用 CurrentControl 属性显示数据，活动单元以 CurrentControl 属性接收数据

2. 表格生成器

相对于命令按钮组和选项按钮组而言，表格对象的属性设置起来较复杂。较好的方法是首先使用表格生成器来设计表格对象的大体布局，然后再使用属性对话框详细设置细节方面的属性。

例 11.13　设计表单 bd17.scx，作用为使用表格来显示学生的基本情况和学生的综合素质成绩。

S1：CREATE FORM bd17

S2：添加两个表格控件，如图 11－19 所示。

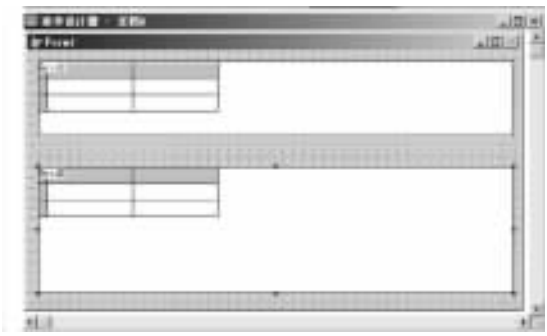


图 11－19　创建两个表格

S3：打开父表“表格生成器”。右击上面表格对象↓ 快捷菜单 →[生成器] ↓ 表格生成器。

S4：打开“网格项”页框，选择需要的字段。选数据表：xsqkb.dbf 及需要的字段，伴随着字段的选定，表格 Grid1 将自动添加上表头对象和列对象。结果如图 11－20 所示。

S5：打开“风格”页框，设置页框的显示式样。在五种式样中选择所需要的式样。如图 11－21所示。

图 11－20　表格生成器的“网格项”页框

图11－21　表格生成器的“风格”页框

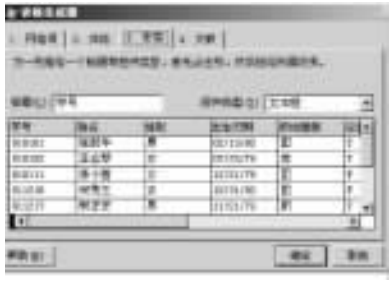


图 11－22　表格生成器的“布局”页框

S6：打开“布局”页框，设置列的显示标题、列控件类型、调整列宽。如图 11－22 所示。

S7：打开“关联”页框，设置子表中的相关索引。在“子表中的相关索引”列表框选相关的索引字段，如“学号”。

S8：→[确定]，表格对象 Grid1 设置完成。

S9：右击 Grid2，打开子表“表格生成器”，重复 S4，S5，S6，选子表 xszhszb.dbf，设置 Grid2 的表头对象、列对象、风格、布局。

S10：打开“关联”页框，设置父表中的关联字段。在“父表中的关联字段”列表框选关键字段，如“xsqkb.学号”，“子表中的相关索引”列表框选相关的索引字段“学号”。

S11：单击[确定]，结束设计。存盘、运行，结果如图 11－23 所示。

图 11－23　bd17 运行结果

注意　在运行时，父表格仅显示当前一条记录，子表格却显示全部记录。当指针在子表格中移动时，父表格也将指向与子表格对应的记录。

表格控件还可以设置其他许多属性，例如可设置表格控件中当前记录与其他记录背景色不同，奇偶记录背景色不同等。表格也可用表单设计器快速设计。

例 11.14　设计表单 bd18.scx，用来显示学生的基本情况，要求当前记录与其他记录的背景色不同。

S1：CREATE FORM bd18

S2：设置表单标题为“表格控件的使用”。设置数据环境 xsqkb.dbf。

S3：添加一个表格对象：Grid1，设置它的 RecordSourceType 属性为 1 -别名，RecordSource 属性为 xsqkb，其他属性使用缺省值。

S4：为表单添加一个自定义属性：Currec，设置其初始值为 1。

S5：编写事件代码。

S5 - 1：ThisForm.Grid1.Init

```
SELECT xsqkb
This.SetAll("DynamicBackColor",;
"IIF(RECNO()=ThisForm.Currec, RGB(255,0,0), RGB(0,0,255))", Column)
ThisForm.Refresh
```

S5 - 2：ThisForm.Grid1.AfterRowColChange

```
LPARAMETERS nColIndex
ThisForm.Currec=RECNO()
ThisForm.Refresh
```

S6：存盘、运行，结果如图 11 - 24 所示，当前记录的背景色为红色，其他记录的背景色为蓝色。

图 11 - 24　bd18 运行结果

11.4.2　页框(PageFrame)

页框是包含页面的容器类对象。用户可以在一个页框中定义多个页面，每一个页面可以包含各种控件，以便生成带有选项卡的对话框。运行中，可单击页面标题来选择页面，被选中的页面为活动页面。

页框的常用属性如表 11 - 18 所示。

表 11 - 18　页框的主要常用属性

属性名称	功 能 说 明	取 值
ActivePage	设置或返回页框中活动页面的页码	数值，缺省为 1
PageCount	指定页框中页面的数目	数值，缺省为 2
Pages	用以存取页框中各页的数组	数值，缺省为 0
Tabs	指定页框是否有选项卡	.T.（默认），.F.
TabStretch	指定页面标题能否分行显示	1 -单行(默认)、0 -多行

页框中的页面,也具有自己的属性,最常用的是页面的标题 Caption,它用来指定选项卡的标题。

例 11.15 设计表单 bd19.scx,含有页框,页框中有两个页面,分别用来显示 xsqkb 和 xscjb。

S1:CREATE FORM bd19

S2:设置表单标题为"页框的使用"。设置数据环境 xsqkb.dbf、xscjb.dbf。

S3:添加一个页框对象 PageFrame1,设置它的 PageCount 属性为 2,其他属性使用缺省值。

S4:将页面 Page1,Page2 的 Caption 属性分别设为学生情况表、学生成绩表。

S5:分别在 Page1,Page2 中各添加一个表格 Grid1,设置:

PageFram1.Page1.Grid1.RecordSource=xsqkb

PageFrame1.Page2.Grid1.RecordSource=xscjb

其他属性均使用缺省值。

S6:存盘,运行,单击页面标题,会在两页面间实现切换显示,效果如图 11-25 所示。

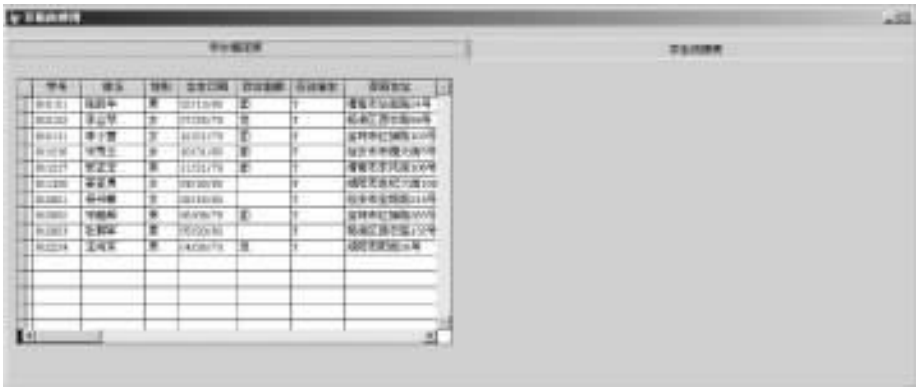


图 11-25 Bd19 运行结果之一

11.4.3 容器(Container)

命令按钮组、选项按钮组、表格、页框这些容器类对象都具有特定的格式或包含一些特定的控件对象。在 Visual FoxPro 中,还有一种可以包含不同类型控件对象的容器类对象——容器。容器所包含的对象既可以是普通类型的控件,又可以是容器类对象。

容器的常用属性有 BackStyle 和 SpecialEffect。

向容器控件添加其他控件的步骤如下:

S1:在表单中添加一个容器控件。

S2:右击容器控件,弹出快捷菜单,选"编辑"。

S3:向容器控件中添加其他控件。

11.5 嵌入与连接类控件

Visual FoxPro 的一个重要特点是它不仅能使用本系统的数据,还能使用其他系统提供的

数据。它可以通过对象链接与嵌入技术(OLE)能将文本、声音、图像或视频等应用程序的文件嵌入或链接到表或表单中。例如可以把 Word 文档、声音文件、视频文件、Excel 电子表格嵌入或链接到表的通用型字段或表单中,可以把文本文件嵌入或链接到备注型字段中。

嵌入与连接是两个不同的概念。嵌入是将数据存储到表或表单中,物理上包含了被嵌入对象的副本;连接是指对被链接对象的一种引用,使逻辑上指明了被链接对象的位置。

嵌入与连接类控件有 ActiveX 控件、ActiveX 绑定控件、超链接(HyperLink)控件。

11.5.1　ActiveX 控件

ActiveX 控件是符合微软公司 ActiveX 技术标准的控件动态链接库。它由 OLE 发展而来,与开发平台无关,在一种编程语言中开发的 ActiveX 控件,无须做任何修改就可在另一种编程语言中使用,效果如同使用 Windows 的通用控件一样。

ActiveX 控件提供了广泛的功能,其数量已超过 1 000 余种。在 Windows 的 System 文件夹中含有大量的扩展名为.ocx 的文件,它们都属于 ActiveX 控件。

Visual FoxPro 中的 ActiveX 控件可划分为标准的 ActiveX 控件和可插入表单的 OLE 对象。

标准 ActiveX 控件,分为两种:

① Windows 控件,如: TreeView Control、Microsoft Statusbar Control、Microsoft Rich-Text Control 等。

② 系统控件,如:Communications 和 MAPI 等。

在 Visual FoxPro 中,不能用来创建新的 ActiveX 控件,但却可以方便地使用 ActiveX 控件,而大量使用的则是标准 AcitiveX 控件。在上一章的向表单中添加控件一节中,我们已经介绍了将 ActiveX 控件添加到控件工具栏的方法和步骤。现具体讲解如何将 ActiveX 控件或对象添加到表单中。

例 11.16　设计表单 bd20.scx,运行时显示一个文件图标。

S1:CREATE FORM bd20

S2:向表单添加一个 ActiveX 控件,将弹出"插入对象"对话框。如图 11-26 所示。

图 11-26　添加 ActiveX 控件及"插入对象"对话框

插入对象对话框含有一个"选择"选项按钮组、一个控件类型列表框、四个命令按钮、一个"结果"标签框。它们的功能如下:

① "选择"选项按钮组,含有三个选项按钮,分别是新建、由文件创建、插入控件。各自的

功能是：

·新建：插入的对象由 Windows 的应用软件来建立。选定该项后，可在“控件类型”列表框中选择需要的控件类型，从而调用对应的应用程序来创建 ActiveX 控件。

例如：在“选择”选项按钮组如果选中了“新建”，在“控件类型”列表框选中了“画笔画片”，单击[确定]，则将调用画笔画片软件，如图 11－27 所示，供用户在表单上的 ActiveX 控件中画图。要结束画图，只要单击控件外表单的任何地方即可。

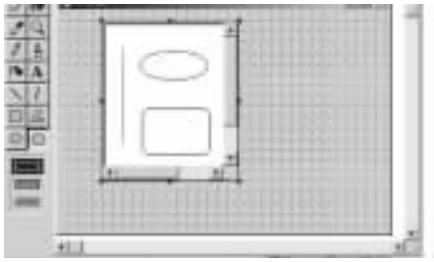


图 11－27 调用画笔图片软件插入图形

·由文件创建：将指定的文件内容插入控件对象。用户选定该选项，将弹出如图 11－28 所示的“插入对象”对话框，可在其后的文本框输入文件的路径和文件名或者单击下方的[浏览]按钮，在弹出的“打开”对话框中选择文件，然后按[确定]，即将所选中的文件插入表单的 ActiveX 控件。注意对话框中的“链接”复选框，其作用是实现表单的 ActiveX 控件与指定的文件的链接。而对话框中的“显示为图标”，将把选中的文件，以图标的形式插入表单，运行时只有双击此图标，才能打开文件。选好文件及“显示为图标”项后，单击[确定]，即将选中的文件插入。

·插入控件：用户可在“对象类型”列表框中选择一个 ActiveX 控件，然后单击[确定]，将该控件放置在表单上。例如在“对象类型”选中“日历控件”，单击[确定]，则将该控件放置在了表单上，运行表单会得到如图 11－29 所示的结果。

图 11－28 由文件创建对象

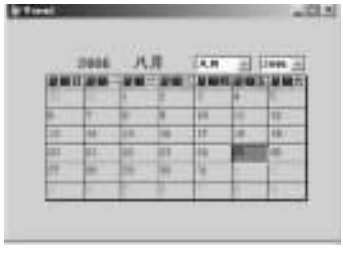


图 11－29 由插入控件创建对象

② “控件类型”列表框：列出了各种控件，供用户选定。

S3：存盘、运行。

11.5.2 ActiveX 绑定控件

ActiveX 绑定控件也称为 OLE 绑定控件，主要用来显示表中的通用型字段。当 ActiveX 绑定控件与表的通用型字段绑定后，就可以显示通用字段中的 OLE 对象，甚至可以调用创建这些 OLE 对象的应用源程序，并以可视的方式来查看或操作这些数据。

要显示表的通用字段的 OLE 对象，应注意下面两点：

① 必须将 ActiveX 绑定控件与表的通用字段相绑定。绑定的方法是将 ActiveX 绑定控

件的 ControlSource 属性设置为要显示的通用字段。

② 表单运行时，只有当记录指针指向含有通用字段数据的记录时，才会显示出绑定的内容。

ActiveX 绑定控件的创建步骤如下：

S1：在表单中添加一个大小、位置合适的 ActiveX 绑定控件。

S2：将 ActiveX 绑定控件的 ControlSorce 属性设置为表的通用字段名。

前面各表单例子中，凡是涉及到 xsqkb.dbf 中的照片显示的，一律都采用 ActiveX 绑定控件来完成，只不过当时是通过快速表单、拖曳字段图标、表单向导等方法创建的，用户不知而已。

例 11.17　设计表单：bd21.scx。运行中，当指针指向表的第 1 条记录时，播放一段视频剪辑，指向第 2 条记录时播放一个歌曲。

S1：创建一个表文件 sp.dbf。

```
CREATE TABLE sp(序号 c(2), 视音频 g)
```

S2：向表中添加记录记录：1 号记录为“1,sp1.avi”；2 号记录为“2,g117.mp3”。

S3：创建一个表单 bd21.scx。

```
CREATE FORM bd21
```

S4：设置表单数据源为 sp.dbf。

S5：向表单中添加两个命令按钮，按钮标题：轮换、退出；添加一个 ActiveX 绑定控件，将它的 ControlSource 属性设置为 sp.视音频。

S6：编写事件代码。

S6－1：thisform.command1.click

```
If recno()<2
   Skip
Else
   Skip －1
Endif
Thisform.refresh
```

S6－2：thisform.command2.click

```
Thisform.release
```

S7：存盘，运行。图 11－30 给出了在播放一段视频时，抓获的一个画面。

图 11－30　bd21 运行时被抓获的一个视频画面

11.5.3 超链接(HyperLink)控件

超链接控件可以使用户使用 IE 浏览器打开某个网页。它含有的方法程序 NavigateTo 允许用户指定一个网址,当执行该方法程序时,Visual FoxPro 就会自动启动 IE 浏览器,并按照指定的网址进入某个站点来显示网页。

由于超链接控件在程序运行时并不显示出来,因此不需要设置其属性,只要对其方法程序 NavigateTo 讲解清楚,用户便可方便地使用它。

1. NavigateTo 方法程序

(1)格式

Object.NavigateTo(*cTaget*[,*cLocation*[,*cFrame*]])

(2)功能

启动注册的 Active Document 容器,漫游到指定的地址。

(3)参数说明

- *cTaget*:指定要访问的 URL(Universal Resource Locator,通用资源定位),通常是网址或 HTML 文档。
- *cLocation*:指定在 *cTaget* 所给出的 URL 或 HTML 文档中要定位的位置。
- *cFrame*:指定在 *cTaget* 所给出的 URL 或 HTML 文档中要定位的页框。

2. Active Document 方法程序

Active Document 方法程序,是基于 Windows 的嵌入在浏览器中的非 HTML 应用程序,它可提供直接从浏览器界面访问应用程序的方法。

例 11.18 设计表单 bd22.scx。其中包含一个超链接和组合框,在组合框中存放了一些网站的网址。运行时,双击下拉列表框或按 Enter 键,将链接到在下拉列表框选定的网站。

S1:创建一个表 wz.dbf。

```
CREATE TABLE wz(网站名 C(40),网址 C(40))
INSERT INTO wz values("陕西省精品课程—数据库原理及应用","http://jpkc.
                      wntc.edu.cn/ec/c15")
INSERT INTO wz values("主持人电子信箱","xieyinbai@126.com")
……
```

S2:创建一个标题为"超链接控件的使用"的表单 bd22.scx。

```
CREATE FORM bd22
```

S3:将表单的数据环境设置为 wz。

S4:向表单中添加一个超链接控件 Hyperlink1、一个组合框 Combo1。

S5:设置组合框的有关属性:

Style:0;RowSourceType:2;RowSource:wz;

ColumnCount:2;BoundColumn:2。

S6:编写组合框 Combo1 的 DblClick 事件的代码:

```
Thisform.HyperLink1.NavigateTo(Thisform.Combo1.Value)
```

表单设计结果如图 11-31 所示。

图 11－31　表单 bd22 的设计结果

S7：存盘、运行。在运行中，当用户在组合框中选中某个网站时，在双击网站名，就可以启动 IE 浏览器而进入该网站。图 11－32 是选中“陕西省精品课程——数据库原理及应用”网站双击后所进入对应网站后的首页显示结果。

图 11－32　表单 bd22 运行结果之一

11.6　思考与练习

一、选择题

1. 在 Visual FoxPro 中，为了将表单从内存中释放（清除），可将表单退出命令按钮的 Click 事件代码设置为（　　）。

A）ThisForm.Refresh　　B）ThisForm.Delete

C）ThisForm.Hide　　D）ThisForm.Release

2. 在表单控件工具栏中，创建一个用于显示一段固定的文本信息字符串的控件是

(　　)。

A) 文本框　　B) 命令组　　C) 标签　　D) 复选框

3. 以下属于非容器类控件的是(　　)。

A) Form　　B) Lable　　C) Page　　D) Container

4. 表单有自己的属性、事件和(　　)。

A) 图形　　B) 方法　　C) 容器　　D) 形状

5. DbClick 事件在(　　)时触发。

A) 当创建对象时　　B) 当从内存中释放对象时

C) 当表单或表单集装入内存时　　D) 当用户双击该对象时

6. 下面关于列表框和组合框的陈述中,正确的是(　　)。

A) 列表框和组合框可以设置成多重选择

B) 列表框可以设置成多重选择,而组合框不能

C) 组合框可以设置成多重选择,而列表框不能

D) 列表框和组合框都不能设置成多重选择

7. 在用命令调用表单时,可用来传递参数的子句是(　　)。

A) WITH　　B) WHERE　　C) FOR　　D) AND

8. 在表单运行中,当结果发生变化时应刷新表单,刷新表单所用的命令是(　　)。

A) RELEASE　　B) DELETE　　C) REFRESH　　D) PACK

9. 可以在表单上对齐和调整控件的位置的工具栏是(　　)。

A) 调色板　　B) 布局　　C) 表单控件　　D) 表单设计器

10. 在表单中,"Caption"是对象的(　　)。

A) 标题属性　　B) 名称属性　　C) 背景透明属性　　D) 字体尺寸属性

11. 在下列对象中,属于容器类对象的是(　　)。

A) 文本框　　B) 组合框　　C) 表格　　D) 命令按钮

12. 决定微调控件最大值的属性是(　　)。

A) KeyboardHighValue　　B) KeyboardLow Value

C) Value　　D) Interval

13. 在表单控件中,既可作为接收输入数据用,又可作为编辑现有数据用的控件是(　　)。

A) 文本框　　B) 标签　　C) 复选框　　D) 列表框

二、填空题

1. ThisformSet,ThisForm,This 分别指_____、_____、_____对象。

2. 属性 RowSourceType 和 RowSource 分别指明列表框的_____、_____属性

2. 图形框属性 Stretch 的三个选择的功能分别是:_____、_____、_____。

三、上机题

1. 使用表单向导设计一个能管理 xscjgl.dbc 的表单,表单文件名为"向导.scx"。

2. 设计一个系统介绍表单,表单文件名为"系统介绍.scx"。

3. 设计一个口令检查表,表单文件名为"登录.scx"。

第 12 章　报表和标签

数据库一切操作之后，人们总是要将最后的结果进行输出。计算机屏幕输出因受屏幕大小的限制，且不能永久保存而只能用于计算机操作者观察、调试程序用。要将结果数据供其他人员使用，就需将它打印为纸质文档。此时，应将数据库操作的结果，设计成报表的形式输出。

报表包括数据源和报表布局两部分。数据源的作用是定义报表中数据的来源。它可以是表（包括数据库表和自由表）、视图、查询等；报表布局用来定义报表的打印格式。

用户设计的报表保存在扩展名为 .frx 的报表文件中，扩展名为 .frt 的文件是报表的备注文件。需要指出的是报表文件并不保存数据，每次运行报表文件，系统都将从数据源文件中取出最新数据，形成输出报表。

Visual FoxPro 9.0 中分为三种：

① 简单报表：数据源是一张表的报表。

② 分组/总计报表：对表中的数据根据某一标准进行分组后而得到的一种汇总报表。此表可为用户提供每组数据的总计值。

③ 一对多报表：根据两张表创建的报表，而这两张表本身是利用一对多关系而创建的。

12.1　创建报表

12.1.1　报表简介

Visual FoxPro 报表布局大致可分为图 12－1 所示的几种情形。一般而言，在创建报表之前，用户应该先确定自己要创建的报表属于这几种报表中的哪一种，才好往下进行。创建报表

图 12－1　报表的各种布局

可利用“快速报表”、“报表向导”和“报表设计器”等多种方式来完成。在创建报表的过程中，用得最多的工具是“报表控件”和“布局”两种。下面结合实例引导读者熟悉这些方法和工具。并掌握创建报表的整个过程。

12.1.2 用“快速报表”法建立列报表

列报表是几种报表中最为简单，但在实际工作中应用得较多的一种报表。

例 12.1 使用“快速报表”创建图 12－2 样式的报表，文件名：bb1.frx。其步骤如下：

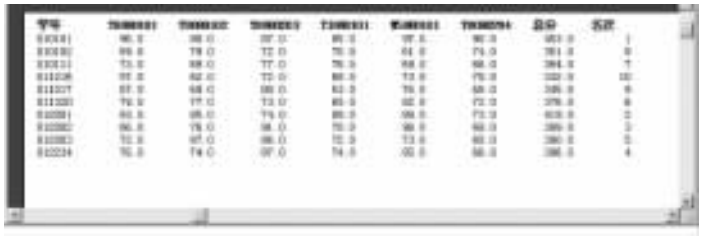


图 12－2 学生成绩表

S1：打开项目管理器：xsxjgl.pjx，选数据库：xssjk.dbc。

S2：打开“报表设计器”。在项目管理器中：→[文档]→[报表]→[新建]↓ 新建报表 →[新新建文件]↓ 报表设计器。如图 12－3 所示。

图 12－3 报表设计器

S3：打开“快速报表”，设计报表。在主菜单上：→[报表]→[快速报表]↓ 打开 →“xscjb”，如图 12－4 所示。→[确定]↓ 快速报表，如图 12－5 所示。

图 12－4 “打开”对话框

图 12－5 “快速报表”对话框图

S3－1：选字段布局。在“快速表表”对话框中有：“字段布局”按钮左边为“列报表”按钮（默认）、右边为“行报表”，今选默认值。

S3－2：对于“标题”、“添加别名”、“将表添加到数据环境中”复选框全部选定。

S3－3：选定字段。→[字段] ↓ 字段选择器，选择有关字段，如图 12－6 所示。→[确定] ←快速报表。

图 12－6　“字段选择”器对话框

S4：→[确定] →报表设计器，如图 12－7 所示。

图 12－7　快速报表设计结果

S5：预览报表。在主菜单中：→ [显示]→ [打印预览]，或者是点击工具栏中的[打印预览]按钮，则报表的结果一览无遗，如图 12－2 所示。

S6：保存报表文件。

由上面操作过程，大家能体会到“快速报表”的确简便快捷，然而只要稍加留神，就会发现这样的报表未必符合我们的习惯，如它并不含有表格线。若想效果更好一些，就应对报表进行改进。

12.1.3　用“报表向导”创建一对多报表

和其他向导一样，“报表向导”可以引导用户方便快捷地创建出自己所需要的报表。

例 12.2　以表 xsqkb.dbf、xscjb.dbf 为数据源，使用“报表向导”建立一个一对多报表，文件名 bb2.frx。

S1：打开项目管理器 xsxjgl.pjx，选数据库 xssjk.dbc。

S2：打开“报表向导”。在项目管理器中：→[文档]→[报表]”→[新建] →[报表向导] ↓ 向导选择→“一对多报表向导(One-Many Roport Wizard)”。如图 12－8 所示。

图 12－8　“向导选择”对话框

S3：**选择字段**。

S3－1：**选择父表字段**。→[**确定**] ↓ **第 1 步—选择文表字段**，**从表** xsqkb.dbf **选中有关字段，如图** 12－9 **所示**。

图 12－9 **“一对多报表向导”第** 1 **步选父表字段**

图 12－10 **“一对多报表向导”第** 2 **步选子表字段**

S3－2：**选择子表字段**。→[**下一步**] ↓ **第 2 步—选择子表字段**，**从表** xscjb.dbf **选中有关字段，如图** 12－10 **所示**。

S3－3：**建立表关系**。→[**下一步**] ↓ **第 3 步—建立表关联**，**如图** 12－11 **所示**。

图 12－11 **“一对多报表向导”第** 3 **步选表间关联**

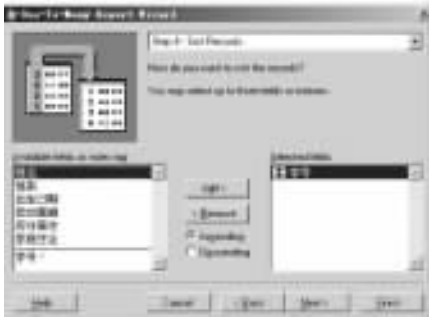


图 12－12 **“一对多报表向导”第** 4 **步选记录排序**

S3－4：**选择记录排序字段及次序**。→[**下一步**] ↓ **第 4 步—记录排序**，**选**：**学号**、**升序**(Ascending)，**如图** 12－12 **所示**。

S3－5：**选择报表样式**。→[**下一步**] ↓ **第 5 步—选择报表样式**，**选**：**账务式**(Ledger)，**如图** 12－13 **所示**。

注意　报表式样共有五种，它们分别是：经营式(Executive)、**帐务式**(Ledger)、**简报式**(Presentation)、**带区式**(Banded)、**随意式**(Casual)，**均可由左上方的放大镜给出样式显示**。

S3－6：**完成**。→[**下一步**] ↓ **第 6 步—完成**，**输入报表标题，如图** 12－14 **所示**。

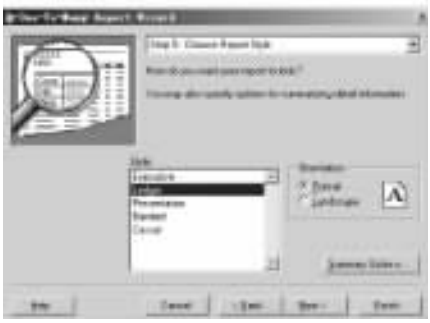


图 12－13　“一对多报表向导”第 5 步选表报式样

图 12－14　“一对多报表向导”第 6 步“完成”

S3－7：预览效果。→[预览]，会显示出报表的效果如图 12－15。

图 12－15　一对多报表预览效果

S4：完成设计。如认为合适，则→[关闭]←一对多报表向导→[完成]↓保存，为报表起名为：“bb2”，→[保存]。

至此，报表设计工作基本完成。

注意　如果不满意设计效果，可在第 6 步中点击[上一步]，返回重新设计。

12.1.4　用“报表设计器”创建分组报表

1. 报表属性对话框

Visual FoxPro 9.0 所提供的“报表属性”页框，可以使用户利用它的七个选项卡，方便地设计出满意的报表来。

打开报表属性对话框的方法是：在报表设计器窗口任意空白处，右击鼠标↓快捷菜单→

[属性]↓ 报表属性。图 12－16 显示的是进入了“报表属性”对话框后，选取了“页面”选项卡后所得到的页面。

图 12－16 “报表属性”页框的“页面”页面

2. 报表设计器带区介绍

“报表设计器”是创建报表的基本工具，在前面的例子中，我们已经初步看到了它由若干个各带区组成。在 Visual FoxPro 9.0 中，除了报表的三个基本带区外，用户还可以根据需要，添加别的带区，以便报表显得更加丰富多彩。带区一共有以下八种类型。

(1) 报表标题带区(Title)

在本区定义的任何对象只出现在报表第一页的最顶端，或者以单独一页给出。常用来设计报表的封面、标题等。

(2) 页标头带区(Page Header)

又称页标题，是系统所设置的基本报表格式中三大带区的第 1 个带区。本区定义的对象仅在每一页的开头处显示一次，通常用来显示数据的提示说明，如报表页上方的固定文字、表头等。

(3) 细节带区(Detail)

又称数据显示区，是系统所设置的基本报表格式中三大带区的第 2 个带区。该区是报表文件最重要的部分，用来显示数据或记录。本区的对象常为字段变量名和修饰字段的线条等。

(4) 页注脚带区(Page Footer)

是系统所设置的基本报表格式中三大带区的第 3 个带区。本区任何数据仅在每一页报表的最底端显示一次，对象常为文字、日期、表达式，如制表人、页码等，与页标头带区对应。

(5) 总结带区(Summary)

本区的数据只会在报表的最后一页的底端出现一次，如数值数据的总计或平均等。与标题带区相对应。

(6) 组标头带区(Group Header)

和页标头带区相类似，当设计分组报表时，本带区的数据会在每一个分组的开始处出现。它一般是分组的标题、组的说明或组标识符等，每组显示一次。

(7) 组注脚带区(Group Footer)

和组标头对应，在每个分组的结束处，会显示组脚注带区的数据，包括分组数据的数值总计、分类汇总、对该组有意义的文字等。如果没有分组，则没有此项。也可以有多重分组。

(8) **列标头带区**(Column Header)

此带区的数据会出现在每页报表列的最顶端,打印报表时每列一个,一般是报表的列标题。

并不是每一份报表都会完整地使用所有的带区,而是根据需要设定。例如,如果报表不需要分组,则没有组标头和组脚注带区;普通报表可以没有标题带区和总结带区等。

用户可以根据需要,在有关栏目中设置报表的参数。

3. 报表设计器带区设置

(1)基本带区设置

打开报表设计器,即获得了三个基本带区:页标头、细节、页脚注。如图 12 - 17 所示。

图 12 - 17　"报表设计器"的基本带区

(2)其他带区设置

使用"报表属性"页框。例如要设置标题和总结带区,可利用"可选带区"页面进行,图 12 - 18给出了在"可选带区"页面中所选的参数及其效果。另外,列表头的设置,可通过"页面"页面中的"栏数"文本框予以设置,组标头和组注脚可通过"数据分组"页面"分组条件"文本框来设置,如果是多重分组,还可通过该页面的"分组嵌套次序"列表框添加。

图 12 - 18　利用"报表属性"页框的"可选带区"页面设置的报表带区效果

4. "报表控件"工具栏

"报表控件"工具栏如图 12 - 19 所示。这里的各个控件(从左往右)的功能如下:

图 12 - 19　报表控件工具栏

· "选定对象":移动或更改控件的大小。在创

建一个控件后,系统将自动选定此按钮,除非选中“按钮锁定”。

· 标签:创建一个标签控件,用来填写必要的说明性的文字。

· 域控件:创建一个字段控件,用于显示表字段、内存变量、系统函数以及其他表达式的内容。

· 线条:用于绘制线条。

· 矩形:用于绘制矩形。

· 圆角矩形:用于绘制椭圆和圆角矩形。

· 图片/ActiveX 绑定控件:显示图片或通用型字段的内容。

· 按钮锁定:允许连续添加多个同种类型的控件,而不需要每次都重复选取相同的控件按钮。

例 12.3 用“报表设计器”建立一个以“班级”为分组条件的分组报表,报表名称:bb3.dbf。设数据源为:xscjb.dbf,字段“学号”的前三个字符代表班级。

S1:打开报表设计器:

```
CREATE REPORT bb3
```

S2:设置数据环境为:xscjb.dbf,生成快速报表。

S3:打开“报表属性”页框,设置有关参数。

S3-1:设置“数据分组”:在“报表属性”框,→[数据分组]↓ 数据分组 →[添加]↓ 表达式生成器,在此页面的表达式文本框输入“Left(xscjb.学号,3)” →[确定]← 报表属性,结果如图 12-20 所示。

图 12-20 设置的“数据分组”条件

S3-2:设置标题和总结带区:在“报表属性”框,→[可选带区]↓ 可选带区,选择如图 12-18右图所示的各复选框。

S3-3:完成属性设置: →[确定]← 报表生成器。

此时报表生成器已添加了:标题、组标头、组脚注、总结四个带区,如图 12-21 所示。

S4:输入并修饰标题。

S4-1:调整各带区高度,向标题带区的合适位置添加一个标签框,输入报表标题“按班级分组成绩表”。

S4-2:在系统工具栏,→[字体属性],设置字体为:隶书、粗体,字号为 3 号。

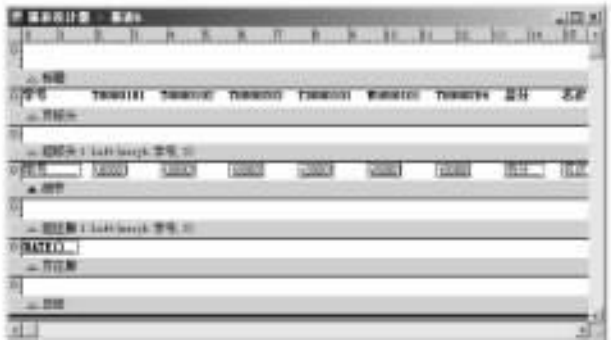


图 12－21　添加了新带区的报表设计器

S4－3：使用“线条”控件，在该标题的上、下方各画出一条直线。

S4－3：用光标将三个控件框定，单击“布局”工具栏中的“水平居中”按钮，将它们放置在报表标题的水平居中位置，标题的设置与修饰完成。

S5：设置页标头属性。

S5－1：鼠标框定页标头带区上的所有控件，在“布局”工具栏单击[水平居中]，为开始腾出一定空间。

S5－2：选定在页标头开头添加标签“班级代码”，为了与其他标签统一，将字体设为宋体、粗体，字号 9。

S5－3：重复 S5－1，使页标头带区内容水平居中。

S6：设置组标头带区属性，本例中忽略组标头内容。

S7：设置细节带区属性。

S7－1：使带区内各域控件实现水平居中，方法同于 S5－1。

S7－2：添加新的域控件：Left(学号，3)。选“报表控件”工具栏，→[域控件]→[细节]↓ 字段属性，如图 12－22所示，在表达式栏输入“LEFT(xscjb. 学号，3)”，→[确定]。

图 12－22　标签属性对话框

S7－3：调整各域控件位置，使之和页标头各标签控件对齐。

S8：设置组注脚带区属性。

S8－1：添加“班级各科最高分：”标签；将细节带区各门功课成绩域控件选定，复制到组脚注带区的“班级各科最高分：”标签控件之后，调整位置使其和细节带区的域控件对应对齐。

S8－2：设置求最大值计算。双击域控件“j8080101”↓ 字段属性→[计算]↓ 计算页面，在“计算类型”列表框选“最大值”，如图 12－23 所示，→[确定]，设置完成。同样做法，设置其他成绩域控件的最大值计算。

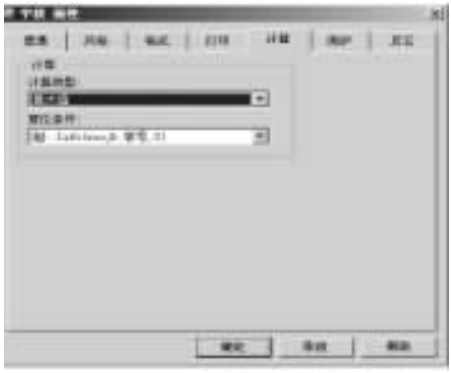


图 12-23 “字段属性”框的“计算”页面

图 12-24 “字段属性”框的“格式”页面

S8-3:添加“班级各科平均分:”标签及有关的成绩域控件,设置计算平均值,方法同于S8-1和S8-2,只不过在“计算”页面的“计算类型”列表框选“平均值”而已。

另外,若想使得数值型表达式的格式整齐,可在“字段属性”框中选“格式”,页面如图12-24所示,在“格式”文本框输入格式,如“999.9”——表示数型数据的总宽度是5位,整数占3位,小数点占1位,另外保留一位小数。

S9:设置页注脚带区属性。

S9-1:设置日期和页码。在“页注脚”带区的左端,填域控件,然后输入系统函数“DATE()”,得到报表当前的日期;在其右端,添加域控件,输入系统属性“_PAGENO”得到报表当前的页码。事实上,这个工作基本已完成,用户只需适当调整它们的位置即可。

S9-2:添加制表人。在页脚注带区中间位置,添加一标签控件“制表人:VFP学习者”,字形、字体、字号自定,令其居中。

S10:设置总结带区属性。

S10-1:将组注脚带区的所有控件复制到本带区。

S10-2:修改各控件属性:将两个标签控件分别改为:各科最高分、各科平均分。对每个域控件,打开“字段属性”框的“计算”页面,将复位条件列表框选为:表。

S11:画线。为表添加上横竖方向的网格线。至此,设计完成,报表设计结果如图12-25所示。注意,画垂直线时一定画到页注脚带区的上限线上,方可使报表输出时,会有一个完整

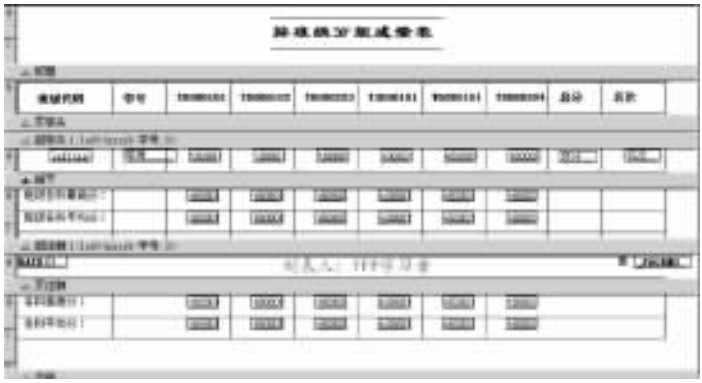


图 12-25 报表 bb3.frx 设计结果

的表格线。

S12：打印浏览报表，得到如图 12－26 的结果，存盘。

图 12－26　报表 bb3.frx 运行的部分结果

由此看来用报表设计器设计报表，步骤非常麻烦，特别是画报表线，没有经验的用户是很难一次画成功的。最好的办法是首先使用报表设计器设计出一个带有报表线的初报表，然后再对该报表进行修改，添加有关的带区等，可以得到事半功倍的结果。

和其他的设计器一样，报表设计器也可使用命令打开，打开报表设计器创建报表的命令是：

CREATE REPORT [*ReportNmae*|?]

打开报表设计器修改报表文件的命令是：

MODIFY REPORT *ReportNmae*

12.2　报表的打印和预览

报表输出有两种方法：

一种方法是用 Visual FoxPro 系统菜单。在这种方式下，若要在屏幕上观看报表，则用在系统菜单的“报表”菜单或在报表的快捷菜单中选“打印预览”；若要打印输出，则再选中“运行报表”选项。

另一种方法是用命令方式。在命令方式下，打印报表的命令是：REPORT。

(1)命令格式

REPORT FORM *Reportname* | ?
[*Scope*] [FOR *lExpression*1] [WHILE *lExpression*2]
[HEADING *cEexpression*] [NOCONSOLE] [PLAIN]
[PREVIEW] [IN WINDOW *FormName* | IN SCREEN]]
[TO PRINTER [PROMPT] | TO FILE *FileName*] [SUMMARY]

(2)参数及子句说明

- *Scope*，FOR *lExpression*1，WHILE *lExpression*2 均缺省：指全部记录。
- HEADING *cExpression*：指定页标头。

• PLAIN:指定只在报表开始位置出现的页标题。当 HEADING 和 PLAIN 同时选定时,应把 PLAIN 子句放在前面。

• NOCONSOLE:指定输出报表时,不在 Visual FoxPro 主窗口或当前活动窗口显示有关信息。

• PREVIEW:预览报表。

• IN WINDOW *FormName* | IN SCREEN:将报表输出到表单还是屏幕。

• TO PRINTER [PROMPT] | TO FILE *File_name*]:将报表打印输出还是写入一个文件。打印输出时,若有 PROMPT 关键字,则在开始打印前显示打印机设置对话框。

• SUMMARY:指定只打印总计和分类总计信息。

例 12.4 用命令方式浏览报表 bb3.frx。

```
REPORT FORM bb3 PREVIEW
```

12.3 标签

12.3.1 用"标签向导"创建标签

标签是一种多列报表,为了与特定的标签纸相匹配而具有相应的特殊设置。其文件扩展名默认为.lbx。用"标签向导"来设计标签是非常方便的。下面举例讲述制作标签的过程。

例 12.4 创建如图 12-27 所示的标签。

图 12-27 标签输出结果

操作步骤如下:

S1:打开标签向导。

S1-1:打开项目:xsxjgl.pjx;打开"数据"页面,选择数据库表:xsqkb.dbf,并浏览之。

S1-2:打开标签向导:→"文档"→[标签]→[新建] ↓ 新建标签 →[标签向导] ↓ 标签向导。

S2:选择数据源表。

在标签向导的第一个对话框 **第 1 步—选择表**，选“数据库和表”框中分别选“xssjk”、“xsqkb”，如图 12－28 所示。

图 12－28　标签向导第 1 步——选表

S3：确定标签类型。

→[下一步] ↓ **第 2 步—确定标签类型**，在其中选：公制(Metric)、2 列(Column)、尺寸(Dimension)23.4 mm * 89 mm，如图 12－29 所示。

S4：定义标签布局。

→[下一步] ↓ **第 3 步—定义标签布局**，在其中输入有关的内容，添加结果如图12－30。

图 12－29　标签向导第 2 步——选标签类型

图 12－30　标签向导第 3 步——定义布局

例如：“学号：学号”，这里第 1 个“学号”是文本，应在左下角的文本框输入“学号”后按添加按钮，添加到“选定的字段”编辑框中；第 2 个“学号”是字段，可双击“可用字段”列表框的“学号”添加。中间的冒号“：”，则通过单击窗口中的标点符号按钮来添加。

S5：定义标签排序方式。

→[下一步] ↓ **第 4 步—记录排序**，选学号的升序。

S6：完成标签定义。

→[下一步] ↓ **第 5 步—完成**，此时可单击[预览]浏览设计结果，得到如图 12－27 所示的结果。

S7：存盘，为该标签起名为：bq1.lbx，至此标签设计结束。

12.3.2 用“标签设计器”创建标签

创建标签的另一种办法是使用“标签设计器”。“标签设计器”和“报表设计器”的操作十分相似,这里就不再赘述。

创建标签的命令是:

CREATE LABEL [*LabelName*|?]

修改标签文件的命令是:

MODIFY LABEL *LabelName*

12.3.3 标签的打印和预览

和报表输出相同,标签的打印和预览也有两种方法:

一种方法是用 Visual FoxPro 系统菜单。在这种方式下,和报表的打印与预览完全相同。若要在屏幕上观看标签,则用在系统菜单的“报表”菜单或在报表的快捷菜单中选“打印预览”;若要打印输出,则选中 “运行报表”选项。

另一种方法是用命令方式。在命令方式下,打印报表的命令是:LABEL。

(1)命令格式

LABEL FORM *Reportname* | ?
[*Scope*] [FOR *lExpression*1] [WHILE lExpression2]
[NOCONSOLE] [PREVIEW] [IN WINDOW *FormName* | IN SCREEN]]
[TO PRINTER [PROMPT] | TO FILE *FileName*]

(2)参数及子句说明

基本同于 REPORT FORM 命令,不再介绍。

例如,要预览标签:bq1.lbx,可直接使用命令:

LABEL FORM bq1 PREVIEW

要打印标签,可使用命令:

LABEL FORM bq1

12.4 思考与练习

一、选择题

1. 报表数据源可以是(　　)。

A) 自由表和其他报表　　B) 自由表和库表

C) 自由表、库表和视图　　D) 自由表、库表、查询和视图

2. 下列选项中属于报表文件的扩展名的是(　　)。

A) .FRX　　B) .MNX　　C) .FPT　　D) .FRT

3. 报表的数据源可以是数据库表、视图、查询或(　　)。

A) 表单　　B) 临时表

C) 记录　　D) 以上都不是

4. 报表中加入图片(　　)。

A）允许　　B）不允许

C）也可、也不可　　D）以上答案都不对

5. 报表的标题打印方式为(　　)。

A）每页打印一次　　B）每列打印一次

C）每个报表打印一次　　D）每组打印一次

6. 表头的设计包括(　　)。

A）表单　　B）临时表

C）记录　　D）以上都不是

7. 在“报表设计器”中，可以使用的控件是(　　)。

A）标签、域控件和线条　　B）标签、域控件和列表框

C）标签、文本框和列表框　　D）布局和数据源

8. 用于打印报表或视图中的字段、变量和表达式的计算结果的控件是(　　)。

A）报表控件　　B）域控件

C）标签控件　　D）图片/绑定控件

9. 在创建快速报表时，基本带区包括(　　)。

A）标题、细节和总结　　B）页标头、细节和页注脚

C）组标头、细节和组注脚　　D）报表标题、细节和页注脚

二、填空题

1. 设计报表通常包括两部分内容：_____和_____；报表标题要通过_____控件定义。

2. 如果已对报表进行了数据分组，报表会自动包含_____和_____带区。

3. 多栏报表的栏目数可以通过_____来设置。

三、上机题

1. 用向导设计一个能输出 xs.dbf 的报表“xsbb.frx”。

2. 分别设计能输出 xs.dbf，kc.dbf，cj.dbf 的三个表，但不得有重复字段的报表。

第 13 章　菜单与工具栏

菜单和工具栏是计算机应用中常见的一种技术。尤其是在 Windows 操作系统环境下，稍具规模的系统软件和应用软件，都有一套完备的菜单和工具栏。菜单层次分明、条理清晰、快捷方便、界面美观、严谨规范，工具栏以按钮为对象，形象易辨，单击即可调用对应程序完成有关操作，它们备受用户的喜爱。Visual FoxPro 除提供系统菜单和工具栏外，还提供一整套灵巧轻便的工具，使得用户很容易地设计出适合自己任务特点的菜单体系和工具栏。

本章将介绍如何在 Visual FoxPro 9.0 中使用自定义的菜单和工具栏。

13.1　Visual FoxPro 的系统菜单

此前，我们利用 Visual FoxPro 所提供的系统菜单，方便而快捷地完成了各项工作，然而要设计自己的菜单，就有必要进一步认识系统菜单的结构，只有了解了系统菜单的结构、特点和行为，方可设计出自己满意的自定义用户菜单。

13.1.1　Visual FoxPro 的菜单结构

Visual FoxPro 支持的两类菜单：条菜单和弹出式菜单。

1. 条菜单

Visual FoxPro 的系统主菜单是一个条菜单。条菜单有一组菜单选项，它们以横向排列在 Visual FoxPro 主窗口顶部的标题栏之下。每一个菜单选项都有一个选项名称和一个内部名字。菜单选项的名称将显示在屏幕上供用户识别，而菜单的内部名字则在程序代码中使用。这类似于表单中的 Caption 与 Name 属性的关系。而条菜单本身也有一个内部名字：_MSYSMENU，它和表单的 Name 属性类似，被看成是整个系统菜单的名字。

例如，Visual FoxPro 主菜单的第 1 个选项，它的选项名称为：文件，它的内部名字为：_MSM_FILE。

2. 弹出式菜单

弹出式菜单是在用户单击了条菜单的某个菜单选项，或右击了某个对象而打开的一种菜单。每一个弹出式菜单，都有一个内部名字和一组菜单选项，每一个菜单选项都有一个选项名称和选项序号。这里选项的名称将显示在屏幕上供用户识别，选项的序号将在代码中被引用。

例如，当用户单击了 Visual FoxPro 主菜单的“文件”选项后，就会弹出与之对应的弹出式菜单，列出文件菜单的 15 个选项供用户选择，如图 13-1 所示。

图 13-1　文件菜单

3. 热键和快捷键

在 Visual FoxPro 子菜单中，可为每一个菜单选项有选择地设置一个热键和快捷键。

(1)热键

所谓热键，是一个字符键，当菜单被激活后，只要按此字符键，就可以快速地选择该菜单选项。热键常用在弹出式菜单中，在菜单中，热键常用一个带有下划线的字母给出。例如，Visual FoxPro 文件菜单中的“新建(N)”、“打开(O)”，就表示 N，O 分别是“新建”、“打开”选项的热键。

(2)快捷键

所谓快捷键是指由 Ctrl 键和一个字母键组合成的组合键。快捷键与热键不同，它不管菜单是否激活，只要按下它的组合键，就可以快速地选中该菜单项。例如文件菜单中“打印(P)…Ctrl+P”，说明 P 键是打印选项的热键，Ctrl+P 则是它的快捷键。

4. 访问键

对于主菜单的各选项，可以设置一个访问键。所谓访问键是指由 Alt 键和一个字母键组成的组合键，当用户同时按下这两个键时，就可以快速地打开它们的弹出式子菜单。Visual FoxPro 主菜单各选项名称后的括号内带有下划线的字母，就是该选项的访问键。

例如，主菜单选项“文件(F)”，用户既可以鼠标单击该选项而弹出它的子菜单，又可以直接按下 Alt+F 而弹出子菜单。

5. 菜单的分隔线

在子菜单中，常常把同一类操作功能的菜单放在一起，为了明确区分两类功能不同的菜单选项，可在它们之间加一条横线，这条线称之为菜单的分隔线。

另外，和 Windows 菜单一样，如果在某菜单选项之后，有省略号“…”，则表示该项后还有下一级菜单或窗口、对话框；如果某菜单选项以灰色给出，表示本菜单目前尚未激活。

6. 菜单的动作

无论是哪个菜单项，一旦被选中就一定要执行一定的动作。这些动作可以是下面三种情况中的一种：执行一条命令，执行一个过程，激活另一个菜单。

13.1.2　Visual FoxPro 的系统菜单

Visual FoxPro 的系统菜单和 Windows 的系统菜单一样是一种下拉式菜单。这种下拉式菜单实际上是条菜单和弹出式菜单的组合。条菜单被称之为主菜单；弹出式菜单则被称为子菜单。

1. 系统主菜单

Visual FoxPro 的系统主菜单，是一个可变化的动态菜单，它随着操作的不同而会有选项的增减，但系统的基本主菜单选项有七个。

主菜单选项的选项名称，打开 Visual FoxPro 9.0 后将显示在屏幕顶部的标题栏之下，供用户选择，它们各自的内部名字，除“帮助”外，都是在该选项名称的英文单词或它们的缩写前加一表示系统主菜单的“_MSM_”词头组成。各自的选项名称及内部名如表 13-1 所示。

表 13－1 主菜单(_MSYSMENU)常用选项名称及内部名字

选项名称	内部名字	选项名称	内部名字
文件	_MSM_FILE	程序	_MSM_PROG
编辑	_MSM_EDIT	窗口	_MSM_WINDO
显示	_MSM_VIEW	帮助	_MSM_SYSTM
工具	_MSM_TOOLS		

2. 常用弹出式菜单的名称和内部名字

弹出式菜单的名称，指选定系统主菜单的选项后弹出的子菜单名称，它与系统主菜单选项同名称，但它具有自己内部名字。弹出式菜单的内部名字除“帮助”外，都是由名称对应的英文单词在前面加上“_M”前缀组成。常用的弹出式菜单的名称及内部名字由表 13－2 给出。

表 13－2 弹出式菜单的名称及内部名字

选项名称	内部名字	选项名称	内部名字
文件	_MFILE	程序	_MPROG
编辑	_MEDIT	窗口	_MWINDOW
显示	_MVIEW	帮助	_MSYSTEM
工具	_MTOOLS		

3. 系统菜单的访问与设置

在应用程序运行期间，是否允许访问 Visual FoxPro 9.0 的系统主菜单，或者重新配置系统主菜单，可以使用：SET SYSMENU 命令来设置。

(1)命令格式

SET SYSMENU ON | OFF | AUTOMATIC
　　| TO [*MenuList*] | TO [*MenuTitleList*] | SAVE | NOSAVE

(2)功能

在程序执行期间，打开或关闭 Visual FoxPro 系统主菜单，并且允许用户重新配置它。

(3)参数及子句说明

· ON：在程序运行期间，当 Visual FoxPro 正在等待输入一个键盘命令(例如 BROWSE，READ，MODIFY COMMAND 等)时，允许使用 Visual FoxPro 主菜单条。

· OFF：在程序运行期间不允许使用 Visual FoxPro 主菜单条。

注意　OFF 参数必须在程序(.prg)代码中被执行，方可起作用。例如，当在程序中运行下面的代码时，将会使 Visual FoxPro 的主菜单条变为非激活状态。

```
SET SYSMENU OFF
WAIT
```

· AUTOMATIC：在程序运行期间，使得 Visual FoxPro 主菜单条可见。主菜单条可访问而菜单项是否激活则与当前的命令有关。这是缺省设置。

· TO [*MenuList*] | TO [*MenuTitleList*]：指定菜单项或 Visual FoxPro 9.0 主菜单条的

标题。菜单或菜单标题列表可以包含任何由逗号分隔开来的菜单或菜单标题的组合。菜单的内部名和菜单标题在系统菜单名字表中列出。例如，下面的命令可以把除文件和窗口菜单外的其他各 Visual FoxPro 主菜单项移去。

```
SET SYSMENU TO _MFILE, _MWINDOW
```

· TO DEFAULT：恢复主菜单条为缺省配置。如果用户已经改变了主菜单条或它的菜单项，使用 SET SYSYMENU TO DEFAULT 可以重新恢复。

· SAVE：将当前的菜单配置设置为缺省配置。如果在 SET SYSMENU SAVE 命令之后用户再次修改了菜单系统，则可以利用 SET SYSMENU TO DEFAULT 来恢复先前的配置。

· NOSAVE：重置菜单系统为缺省的 Visual FoxPro 系统菜单。但是，缺省的 Visual FoxPro 菜单并不显示，直到使用 SET SYSMENU TO DEFAULT 命令之前。

注意　不带辅助参数的 SET SYSMENU TO 命令将使得 Visual FoxPro 主菜单条被屏蔽。

13.2　下拉式菜单的设计

13.2.1　菜单设计器的打开

菜单设计器的打开方法同于其他设计器的打开方法，分为使用菜单交互式打开和命令打开两种。菜单设计器设计的菜单在没有进行菜单生成之前，一律为 .mnx 文件。

1. 菜单交互式打开菜单设计器

使用菜单交互式打开菜单设计器有三种方法，与其他设计器的打开步骤均相同。

① 方法 1：通过文件菜单的“新建”选项进行。

② 方法 2：通过工具栏的“新建”按钮进行。

③ 方法 3：通过项目管理器中的“其他”页面中的“菜单”选项进行。

因与其他设计器打开的方法相同，此处无需赘述。

2. 使用命令打开菜单设计器

命令打开菜单设计器，有创建菜单命令和修改菜单命令之分。

(1)创建菜单命令

① 命令格式：

```
CREATE MENU [FileName | ?] [NOWAIT] [SAVE]
    [WINDOW WindowName1] [IN [WINDOW] WindowName2 | IN SCREEN
```

② 功能：打开菜单设计器，创建新菜单。

③ 参数及子句说明：

· *FileName*：要创建的菜单的文件名，缺省的扩展名 .mnx。

· SAVE：当其他窗口被激活后，允许菜单设计器仍然打开。如果无 SAVE，当其他窗口被激活后菜单设计器被关闭。SAVE 在命令窗口不起作用。

· WINDOW *WindowName1*：指定一个窗口特性，该窗口的特性菜单设计器可继承。例如，如果用一个带有 FLOAT 选项的 DEFINE WINDOW 命令生成一个窗口，则菜单设计器可

以移动。该窗口不需要激活或可见,但必须事先定义好。菜单设计器的缺省尺寸可以比它所继承特性的窗口大。此时,菜单设计器一直呈现该窗口的特性。菜单设计器的左上角和窗口的左上角坐标相同,而窗口的边界则可扩展。

· IN [WINDOW] *WindowName2*:指定一个菜单设计器打开的父窗口。菜单设计器不能呈现父窗口的特性,也不能移到父窗口的外边。如果父窗口移动,菜单设计器则跟着移动。为了访问菜单设计器,父窗口必须首先使用 DEFINE WINDW 命令进行定义,并且必须是可见的。

· IN SCREEN:指定在菜单设计器已经放置到了一个父窗口后,把它在 Visual FoxPro 主窗口中打开。菜单设计器通过子句 IN WINDOW 被放置到父窗口。

· 不含任何参数的 CREATE MENU 命令,可以打开一个菜单设计器,在它里面用户能够定义一个菜单系统。

例如,打开菜单设计器,设计一个菜单 cd1.mnx,可使用命令:

```
CREATE MENU cd1
```

(2)修改菜单命令

通过修改已有的菜单命令,也可以打开菜单设计器,其命令是:

```
MODIFY MENU MenuName
```

13.2.2 菜单设计器的结构

一个空的菜单设计器,如图 13-2 所示。

图 13-2 菜单设计器窗口

菜单设计器由“菜单名称”列、“结果”列、“选项”列、“菜单级”列表框,“菜单项”按钮组等组成。

1. “菜单名称”列

“菜单名称”用来输入菜单项的名称和访问键,对于子菜单还允许加入分隔线。

2. “结果”列

“结果”列是一个列表框,当用户输入了菜单名称后,该列表框会变得可见。它有“命令”、“填充名称”、“子菜单”、“过程”四个选项。

(1)命令

选择该选项,允许用户在“结果”列后面弹出的文本框中,输入一条要执行的 Visual Fxo-Pro 命令。

(2)填充名称

供用户定义第一级菜单的菜单名字或子菜单的菜单序号, Visual FoxPro 9.0 将通过它来引用这个菜单项。实际上系统会自动设定菜单名字或菜单序号,但系统所起的名字并不好

记，不便于在菜单程序和在程序中引用。用户选择此项，可在“结果”后面弹出的文本框中输入一个自己的菜单选项名字。但本选择在菜单设计中并不需要选用。

(3)子菜单

表示该菜单选项包含有子菜单，该选项为缺省选项。选择该项，将在“结果”的右侧弹出“创建”按钮，单击之，可以调出新的菜单设计器供设计下一级菜单。当设计结束返回上级菜单设计器时，“创建”按钮将变为“编辑”按钮。

(4)过程

表示这里是一个 Visual FoxPro 过程程序。选择该项后，在“结果”之后会弹出一个过程编辑窗口，供用户输入过程程序。但在这里编写过程程序时，并不需要以 Procedure 或 Function 语句开头，也不需要以 EndProc 或 EndFunc 结尾。

3. “选项”列

单击“选项”列的命令按钮，会弹出一个如图 13－3 所示的“提示选项”对话框，该对话框用来设置菜单选项的各种高级属性，例如快捷键、启动和废止菜单项、显示状态栏信息等。

图 13－3　“提示选项”对话框

(1)定义快捷菜单

由“键标签”、“键说明”两个文本框组成。要定义快捷键，只要在“键标签”文本框，按下 Ctrl＋快捷字母即可，例如按下 Ctrl＋Q。此时，在两个文本框同时出现“Ctrl＋Q”的快捷键标识。如果希望在应用程序中显示其他的内容，则可在“键说明”文本框更改其中的内容。例如将其改为“^Q”。若果要取消所定义的快捷键，则应在“键标签”文本框按空格键。

(2)启用和废止菜单项

启用菜单项指该菜单项被激活，变为深颜色，使之可用。废止菜单项则正好相反。

设置和废止菜单或菜单项的方法是：

S1：→[跳过 …] ↓ 表达式生成器。

S2：在“表达式生成器”对话框的文本框中输入跳过表达式。其含义是当菜单运行时，如果该表达式条件满足(为.T.)，则该菜单项被废止。

例如：姓名＝“王” or 姓名＝“李”，就表示凡遇见姓“王”或姓“李”的人则此菜单选项变为灰色而不可用。

(3)信息

该文本框的作用是：设置菜单或菜单项的较详细的说明信息，当鼠标移动到该菜单选项时，该信息将在状态栏出现。

4. “菜单级”列

这是一个列表框，用于在主菜单(菜单栏)、子菜单间的相互选择。

5. 插入按钮

用于在当前菜单行前插入一个新的菜单行。

6. 插入菜单栏按钮

用于在当前菜单行前插入一个系统已经定义好的菜单项。单击之,会弹出如图 13-4 所示的"插入系统菜单栏"对话框,用户可在其中选择一个 Visual FoxPro 菜单项,插入。

7. 删除按钮

用于删除当前菜单行。

图 13-4 "插入系统菜单栏"对话框

8. 移菜单项按钮

用于将当前菜单行移动到一个新的位置。单击之,会弹出一个如图 13-5 所示的"移菜单项"对话框,供用户在"移动位置"列表框中为当前菜单选项,选择一个合适的位置。

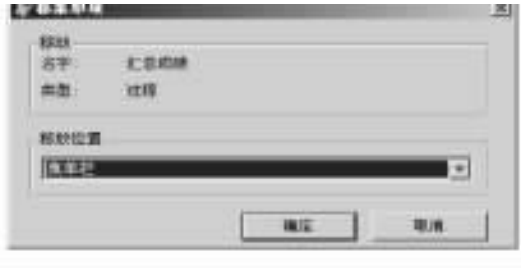


图 13-5 "移菜单项"对话框

9. 预览按钮

供用户观察设计效果,但并不生成菜单程序文件.mpr。

13.2.3 使用菜单设计器创建下拉式菜单

使用命令创建菜单,非常复杂,相对使用菜单设计器问题简单得多,用户只要通过菜单设计器将菜单设计好,然后运行"生成菜单"就可以得到复杂的菜单程序。菜单程序的扩展名为.mpr。

无论用户的应用程序有多大,使用的菜单有多复杂,创建菜单一般需按如下步骤进行:

S1:规划与设计菜单系统。这一步要确定需要哪些菜单出现在菜单界面的何处,哪些菜单都需要子菜单等。

S2:创建主菜单和子菜单。本步使用菜单设计器定义菜单标题、菜单选项和子菜单。

S3:为菜单系统指定任务。本步根据实际要求,确定三大动作之一为菜单选项的任务。如果是过程,应编写过程代码。

S4:生成菜单程序。本步将用户用菜单设计器设计的扩展名为.mnx 的菜单(菜单的备注文件的扩展名为:.mnt)通过系统菜单中"菜单"选项中的"生成…"选项,生成扩展名为.mpr 的菜单程序文件。

S5:运行并调试、测试菜单功能。

现通过一个实例讲述使用菜单设计器创建下拉式菜单的过程。

例 13.1　设有项目：xsxjgl.pjx 中有一个表：kscj.dbf(学号 C(4)，性别 C(2)，姓名 C(8)，语文 N(4,1)，数学 N(4,1)，英语 N(4,1)，总分 N(4,1)。建立如图 13－6 所示的一个菜单，要求菜单运行时，仅保留系统菜单的“文件”、“窗口”、“帮助”选项。菜单文件名 cd2.mnx。

其操作步骤如下：

S1：规划与设计菜单。

规化与设计的菜单如图 13－6 所示，第 1 行是主菜单，下面的是二级、三级菜单。主菜单选项有访问键；子菜单选项可以有热键、快捷键。

成绩处理(p)		成绩查询(S)	系统管理(m)
编辑表(e)	Ctrl＋e	按性别查询(s)	退出(g)
汇总成绩(t)	Ctrl＋t	按学号查询(n)	备份数据(c)
建立索引(i)	Ctrl＋i	按姓名查询(m)	
├ 按姓名索引		查总分最高前 3 名(h)	

图 13－6　菜单样式

S2：打开项目管理器 xsxjgl.pjx。

```
MODIFY PROJECT xsxjgl
```

S3：打开菜单设计器。

```
CREATE MENU cd2
```

S4：定义主菜单选项。

S4－1：在菜单名称列，分别输入成绩处理(\＜p)、成绩查询(\＜s)、系统管理(\＜m)，其中的(\＜P)、(\＜s)、(\＜m)，用来定义访问键。

S4－2：为主菜单指定任务，在“结果”列，分别选择“子菜单”。

主菜单栏创建完成，结果如图 13－7 所示。

图 13－7　定义主菜单各选项

S5：定义各子菜单。

主菜单定义后应开始子菜单的定义。现以定义子菜单“成绩处理”为例讲解。

S5－1：打开新的“菜单设计器”(以定义子菜单“成绩处理”为例)。→“成绩处理(\＜p)”

→[创建]↓ 菜单设计器。

S5－2：定义子菜单选项。

和定义主菜项方法相同，分别在“菜单名称”列输入：编辑表(\<e)、\－、汇总成绩(\<t)、建立索引(<i)。“(\<e)”、“(\<t)”、“(\<i)”分别用来设置热键和分割线。为它们在“结果”列分别指定任务为子菜单、子菜单、过程、子菜单，结果如图 13－8 所示。

图 13－8 定义子菜单“成绩处理”各选项

注意 “(\<字母)”在菜单栏用来设置访问键，在子菜单中用来设置热键，“\－”用来设计分隔线。

S5－3：定义快捷键(以定义“编辑表”选项的快捷键：Ctrl＋E 为例)。

· 打开“提示选项”对话框。→“编辑表”→[选项]↓ 提示选项。

· 定义快捷键。在“键标签”文本框，按下组合键：Ctrl＋E。

· 返回“菜单设计器”。→[确定]← 菜单设计器，此时“选项”按钮上将出现“√”，表示“提示选项”框的各项选项设置完成，本例中则表示“编辑表”选项的快捷键设置完成。

S5－4：重复 S5－3，完成其他两个选项的快捷键设置。

S5－5：“重复 S5－1 到 S5－4，完成其他两个子菜单的设置。

S5－6：为“成绩汇总”选项创建过程程序。

· 打开过程编辑框。→“成绩汇总”→[结果]→[过程]→[创建]↓ 过程编辑框。

· 写入过程程序：

```
UPDATE kscj SET 总分＝语文＋数学＋英语
SELECT * FROM kscj
```

· →X← 菜单设计器，此时原[创建]按钮已变为[编辑]按钮，一般用于修改过程程序。

S5－7：为“建立索引”选项，定义下级子菜单选项为按姓名索引、按总分索引，分别设置它们的“结果”列为“命令”，创建命令分别为：

```
INDEX ON 姓名 TO xm
INDEX ON 总分 TO zf
```

S5－8：重复 S5－1 到 S5－7，定义“成绩查询”、“系统管理”子菜单。

S6：为“成绩查询”各选项指定任务。

S6－1：“按性别查询”过程代码：

```
CLEAR
```

```
ACCEPT "请输入要查询的学生的性别:" TO xb
SELECT * FROM kscj WHERE 性别=xb
```

S6－2:**“按学号查询”过程代码:**

```
CLEAR
ACCEPT "请输入要查询的学生的学号:" TO xh
SELECT * FROM kscj WHERE 学号=xh
```

S6－3:**“按姓名查询”过程代码:**

```
CLEAR
ACCEPT "请输入要查询的学生的姓名:" TO xm
SELECT * FROM kscj WHERE 姓名=xm
```

S6－4:**“查总分最高前 3 名”过程代码:**

```
CLEAR
SELECT * TOP 3 FROM kscj ORDER BY 总分 DESC
```

S6－5:**“查单科不及格者”过程代码:**

```
CLEAR
ACCEPT "请输入课程名:" TO kcm
SELECT 学号,姓名,&kcm FROM kscj WHERE &kcm<60
```

最终设计的“成绩查询”子菜单如图 13－9 所示。

图 13－9　子菜单“成绩查询”及选项“按性别查询”的过程程序

S7:**为“系统管理”各选项指定任务。**

S7－1:**“退出”选项过程代码:**

```
CLEAR
SET SYSMENU TO DEFAULT
RETURN
```

S7－2:**“备份数据”选项过程代码:**

```
CLEAR
CLOSE ALL
COPY FILE kscj.* TO d:\Kscj.*
```

S8:**预览设计结果。**

在设计菜单过程中,用户随时可点击菜单设计器上的“预览”按钮,查看菜单的层次关系是否清楚,子菜单弹出是否正确等。

S9：**生成菜单程序**.mpr **文件**。

在系统菜单栏：→[**菜单**]→[**生成菜单**]。

S10：**修改菜单程序文件**.mpr。

由于题目要求要保留系统菜单栏的“文件”、“窗口”、“帮助”选项，而通过“生成”选项生成的菜单文件运行时并无此三项，因此需要对生成的.mpr **文件作适当修改。修改的步骤是**：

S10－1：**打开菜单程序，如图** 13－10 **所示**。

图 13－10　**由菜单设计器生成的菜单文件** cd2.mpr

MODIFY COMMAND cd2.mpr

S10－2：**修改菜单程序开头的菜单设置命令** SET SYSMENU **的有关参数及其他语句，将图** 13－10 **中用黑颜色框出来的两句**：

SET SYSMENU TO

SET SYSMENU AUTOMATIC

改为：

SET SYSMENU TO _MFILE, _MSYSTEM, _MWINDOW

为了清理屏幕，添加一句：CLEAR，**结果如图** 13－11 **所示**。

图 13－11　**修改后的菜单文件** cd2.mpr

S11：**运行菜单，结果如图** 11－12 **所示。在主菜单有六项菜单选项，单击“成绩处理”(或** Alt＋P)**会弹出“成绩处理”子菜单，单击建立索引(或** Ctrl＋I，**或单击“I”键)会弹出“建立索**

引”的二级菜单选项。

图 13－12　菜单 cd.mpr 运行结果

13.2.4　运行菜单

运行菜单的方法有两种，菜单法和命令法。

1. 菜单法运行菜单程序(.mpr)

菜单法运行菜单程序，有三种方法。

- 在项目管理器中运行：→“其他”→“菜单”→菜单文件名→[运行]。
- 通过主菜单：→“程序”→“运行”↓ 运行 →菜单文件名→[运行]。
- 通过工具栏的 ! 运行当前菜单。

2. 命令法运行菜单程序(.mpr)

DO *MenuFileName*.mpr

注意　这里菜单文件的扩展名.mpr 不得缺省，否则系统将执行.prg 文件。

另外，本例中“编辑表”选项的子菜单，由读者自己完成。

13.3　快捷菜单的设计

快捷菜单一般从属于某个对象，当用鼠标右击该对象时，就会弹出快捷菜单。它的设计方法与下拉菜单的设计方法相同。

S1：打开快捷菜单设计器。

在系统工具栏：→[新建]→[菜单]→[新建文件]→[快捷菜单]↓ 快捷菜单设计器。

S2：添加菜单项，并指定任务。

S3：保存菜单，生成.mpr 菜单程序文件。

S4：在命令窗口运行：

ON Key Label RightMouse DO Menu*FileName*.mpr

例 13.2　创建如图 13－13 所示的快捷菜单。

操作步骤如下：

S1：制作快捷菜单。

打开快捷菜单设计器，在“菜单名称”列输入两个菜单项：“现在是：”、“离 08 年北京奥运会开幕还有：”。

图 13－13　快捷菜单：kjcd1.mpr

S2：为各菜单项指派任务。

指定“结果”列为“过程”，给每项过程按表 13－3 所给出的过程代码编写过程程序。

S3：**生成菜单程序文件**：kjcd1.mpr。

S4：**设计表单** bd24.scx，**它有两个标签控件** Lable1、Label2，**属性根据情况设置**。

表 13-3 各项菜单及结果设计

菜单名称	过程代码
今天是：	SET DATE LONG St="**现在是**：" xx=DATETIME() DO FORM bd24 WITH st, xx
离 08 **年北京奥运会开幕还有**：	St="**离** 08 **年北京奥运会开幕还有**：" xx="{^2008/08/08}-DATE()" DO FORM bd24 WITH st, xx

S5：**编写表单** bd24.scx **的** init **事件代码如下**：

```
PARAMETERS c,t
thisform.label1.Caption=c
thisform.label2.left=thisform.width/2
IF ALLTRIM(c)="现在"
   thisform.label2.Caption=TTOC(t)
else
   thisform.label2.Caption=ALLTRIM(str(&t))+"天"
endif
```

S6：**对表单存盘**。

S7：**在命令窗口执行命令**：

```
ON KEY Label RightMouse DO kjcd1.mpr
```

至此，快捷菜单设计完成，只要在屏幕的任何地方右击鼠标，就可弹出如图 13-13 **所示的快捷菜单。单击"现在是："，得到如图** 13-14 **的显示；单击"离** 08 **年北京奥运会还有："，又会得到图** 13-15 **的显示**。

图 13-14 **快捷菜单显示的当前时间**

图 13-15 **快捷菜单显示的离奥运会开幕的天数**

13.4　创建与使用自定义工具栏

在 Windows 操作系统中，工具栏是应有非常广泛方便的对象。在 Visual FoxPro 9.0 中创建工具栏的方法有两种：

方法一：使用 Microsoft 公司提供的 ActiveX 控件：Microsoft Toolbar Control。

方法二：使用工具栏的方法程序创建一个自定义工具栏类，然后再将自定义类实例化。本节主要介绍第 2 种方法。

13.4.1　创建自定义工具栏类

下面以创建一个具体的自定义工具栏类来讲述创建自定义工具栏类的步骤。

例 13.3　创建一个含有七个按钮的自定义工具栏类 zdyl，将它存放在 MyClsLib 类库中。

S1：打开“新建类”对话框。

S2：输入有关自定义类的参数。

- 在“Class Name：”（类名）文本框输入要创建的类的名称：“zdyl”。
- 在“Based On：”（派生于）列表框选“ToolBar”。
- 在“Store in：”（存储于）文本框选存储位置，结果如图 13-16 所示。

图 13-16　“新建类”对话框

图 13-17　类设计器

S3：打开“类设计器”窗口。

→[确定]↓类设计器，如图 13-17 所示。

S4：利用“表单控件”工具栏向类设计器中添加所需要的控件。

此处添加了八个按钮，如图 13-18 所示，存盘。

图 13-18　创建好的自定义工具栏类

需要说明的是，关于各个按钮的 Click 事件的代码，需要用户自己编写，此处不再讲述。至此，自定义工具栏类创建完成。

13.4.2 使用自定义工具栏

自定义的工具栏类创建完成后，必须首先将它添加到表单控件工具栏中，然后将工具栏类添加到表单集中，方可使用。下面以实例说明具体步骤。

例 13.4 用自定义工具栏类：zdyl，创建一个工具栏实例。

S1：新建一个表单。

S2：打开“打开”对话框。

在表单工具控件栏中：→[查看类]→[添加]↓打开。

S3：添加自定义工具栏类。

在“打开”对话框中，找到存放自定义工具栏的可视化类库 MyClsLib.vcx，如图 13-19 所示。单击[打开]，将该类库添加到表单控件中。添加了自定义工具栏类的表单控件工具栏如图 13-20 所示。

图 13-19 “打开”对话框

图 13-20 添加了自定义工具栏类的表单控件工具栏

现在，就可以像使用标准控件一样使用了。可是，自定义工具栏实际上是一个表单，所以无法将它添加到表单中，只有将它添加到一种比表单更大的容器类对象即表单集中方可使用。

S4：创建一个表单集。

新建一个表单，在系统菜单中：→[表单]→[创建表单集]。

S5：将自定义工具栏类拖放到表单集，设置 Caption 为“自定义工具栏”，Name 为“zdygjl”，添加完成。

S6：添加表单集的 Init 事件代码：

```
This.zdygjl.Dock(0)
```

S7：运行，此时添加了自定义工具栏的表单，运行结果如图 13-21 所示。

图 13-21 添加了自定义工具栏的表单集

在对表单集进行初始化时，使用了工具栏的 Dock 方法，Dock 方法的作用是设置工具栏停放的初始位置。关于 Dock 方法的值，如表 13－4 所示。

表 13－4　Dock 方法的却只及作用

取值	常量	作用说明
－1	TOOL－NOTDOCKED	不停放工具栏
0	TOOL_TOP	在 Visual FoxPro 主窗口顶部停放工具栏
1	TOOL_LEFT	在 Visual FoxPro 主窗口左侧停放工具栏
2	TOOL_RIGHT	在 Visual FoxPro 主窗口右侧停放工具栏
3	TOOL_BOTTOM	在 Visual FoxPro 主窗口底部停放工具栏
X，Y		指定停放工具栏的水平和竖直坐标

当停放一个工具栏时，工具栏的标题将被隐藏，边框也变为单线边框。工具栏的大小被调整成一个单行的按钮条，窗口的大小也将自动调整，以保证工具栏不会遮盖住屏幕上的任何信息。

除使用 Dock 方法外，工具栏还可以手动调整其停放位置，只要用鼠标按住工具栏的头部，拖动到合适的位置即可。图 13－21 中的自定义工具栏的停放位置，实际上开始在表单的顶部，现在显示的则是拖动后的结果。

13.4.3　工具栏与菜单的协调

自定义工具栏后，必须使其和菜单的命令具有相互对应的功能，即必须协调工具栏和菜单。例如，点击了某个按钮，就必须启动相应的菜单选项。因此在创建应用程序的工具栏和菜单时注意：一是菜单选项与工具栏按钮功能必须一致；二是两者应具有相同的可用和不可用属性。

协调菜单和工具栏按钮，一般按如下步骤进行。

S1：创建工具栏类，添加按钮，编写 Click 代码。

S2：创建一个与之相协调的菜单。

S3：添加协调的工具栏和菜单到一个表单集中。

创建与工具栏相协调的菜单，应按下面的步骤进行。

S1：创建一个新菜单，“菜单名称”列中，各选项的名称应与工具栏中有关的按钮提示信息相一致。

S2：在菜单的“结果”列中，选“命令”，调用对应工具栏按钮的 Click 事件代码。

例如：假设在菜单的“菜单名称”列添加了一个“笔记本”，从图 13－21 中可以看出，对应的“笔记本”按钮位于自定义工具栏的第 3 个按钮，即 zdygjl.Command3，那么在菜单的“结果”列选“命令”后，在其后弹出的文本框中应输入命令：

```
This FormSet.zdygjl.Command3.Click
```

S3：→[选项]↓ 提示选项。

S4：在“提示选项”对话框的“跳过”文本框中，输入表达式，当表达式为真时，使得相应的

菜单项不可用,为假时可用。

例如:对于"笔记本"选项,可设置"跳过"表达式为

NOT This FormSet.zdygjl.Command3.Enabled

S5:生成菜单,把菜单添加到具有工具栏的表单集中,运行该表单集。

此时,当打开菜单时,Visual FoxPro 首先计算"跳过"条件值,如果相关的菜单选项不可用,则工具栏对应的按钮亦不可用。当用户选定菜单的某项时,相应按钮的 Click 代码则被执行。这就达到了菜单与工具栏的协调。

13.4.4 在顶层表单中添加工具栏

所谓顶层表单,是指 Visual FoxPro 主窗口被隐藏后的用户应用程序主窗口。向顶层表单中添加工具栏的方法与相前面介绍的将工具栏停放在 Visual FoxPro 主窗口的方法不同。下面以实例讲解添加的步骤。

例 13.5 将自定义工具栏 xdyl,添加到顶层表单中。

S1:创建一个表单集,将表单 1 设置为顶层表单。

打开表单设计器,在属性框中,将 Caption 属性设置为:自定义工具栏的使用、表单的"ShowWindow"属性为 2—作为顶层表单,并在系统菜单的"表单"中选"创建表单集"。

S2:将自定义工具栏类设为最上层表单。

打开类设计器,将自定义工具栏类的"ShowWindow"属性设置为 1—最上层表单。

S3:添加表单集的 Init 事件代码:

```
PUBLIC f &&表示是否已创建了工具栏
f=0
```

S4:添加表单集的 Activate 事件代码:

```
If f=0
  f=1
  SET ClassLib to myclslib
  This.AddObject("zdygjl","xdyl")
  This.zdygjl.Show
  This.zdygjl.Dock(0)
Endif
```

设置完成,运行表单集,会得到如图 13-22 所示的结果。

图 13-22 在顶层表单中使用工具栏

13.5 为顶层表单添加菜单

下面以一个实例介绍向顶层表单添加菜单的步骤。

例 13.6 设计一个菜单 cd3.mnx,将它添加到顶层表单中。

S1:设计一个下拉式菜单。

S1-1:为了方便,将原有菜单:cd2.mnx 及其所有与它有关的其他文件全部拷贝成:cd3.*。

```
COPY FILE cd2.* TO cd3.*
MODIFY MENU cd3
```

S1－2：将菜单重新生成一遍。

S2：打开“常规选项”对话框，设置表单为顶层表单。

在系统菜单中：→[显示]→[常规选项]↓常规选项→[顶层表单]→[确定]。

S3：创建一个表单 bd27.scx，将它的 ShowWindow 属性设为 2—作为顶层表单。

S4：在表单 bd27 的 init 事件中，添加调用菜单程序的命令。

```
DO MenuName.mpr WITH This [,cMenuName]
```

其中，This 表示本表单，*cMenuName* 可为被添加的下拉菜单的条菜单指定一个内部名字。本例中使用的具体命令如下：

```
DO cd3.mpr WITH This ,"Mycd"
```

S5：在表单 bd27 的 Destroy 事件中，添加清除菜单的命令；

```
RELEASE MENU cMenuName [EXTENDED]
```

其中，EXTENDED 表示连同子菜单一起清除掉。本例中，具体使用：

```
RELEASE MENU Mycd EXTENDED
```

至此，为顶层表单添加菜单的工作已全部完成，只要运行表单 bd27，就会得到添加了菜单的顶层表单，如图 13－23所示。

图 13－23　添加了菜单的顶层表单

从图中可以看到，向顶层表单添加菜单后，不见了工具栏，因此一般在用户的应用程序设计中，常常采用的是向顶层表单添加工具栏的方法，这样可以获得比向顶层表单添加菜单更美观、一致的界面。

13.6　思考与练习

1. 设计下面的一个下拉式菜单。

其中各菜单项的功能：

信息处理——含两个子菜单项，分别为“各单位运动员信息录入”和“编排比赛分组”。

竞赛成绩处理——含两个子菜单，分别为“田赛项目处理”和“径赛项目处理”。

公布成绩——含四个子菜单，分别为“教工团体成绩预览”、“学生团体成绩预览”、“教工团体成绩打印”以及“学生团体成绩打印”。

退出——恢复 VFP 标准的系统菜单。

2. 在设计主菜单中的“结果”下拉列表中的四项内容与设计子菜单中的“结果”下拉列表中的四项内容有什么不同？

3. 如何使程序与菜单连接起来？

4. 请设计用于工资管理系统的菜单，主菜单要求如下：

第 1 项：初始化（子菜单有：建立新表、增减部门）。

第 2 项：数据管理（子菜单有：人员变动、部门修改、数据修改）。

第 3 项:查询(子菜单有:姓名查询、部门查询、编号查询)。

第 4 项:计算汇总(子菜单有:个人工资、汇总工资、分类汇总)。

第 5 项:打印报表(过程)。

第 6 项:退出系统(过程)。

第 14 章　应用程序的调试编译和发布

要开发一个 Visual FoxPro 9.0 的应用程序，首先应进行合理的程序规划，以确保程序开发的顺利进行。在应用程序初步开发完成后，应对程序进行调试和各种严格的测试，以纠正程序中可能存在的各种错误。调试测试完成后，还应将代码连编成可执行文件发布给用户使用。此后应用系统进入使用、维护、改进期，这将贯穿于软件使用的整个生命历程。

14.1　Visual FoxPro 9.0 应用程序的规划

根据软件工程的思想和数据库原理理论，在应用 Visual FoxPro 9.0 进行应用程序开发之前，应该对应用程序的开发进行合理的规划，以确保应用程序能够顺利地开发。

14.1.1　应用程序开发的一般步骤

使用 Visual FoxPro 9.0 进行应用程序开发，数据库的设计一般有两种策略：

第一种策略是：自顶向下(Top－Down)—— 从一般到特殊的开发策略。这种开发策略首先从高层入手，分析目标、对象、策略，构建高层的抽象数据模型；然后再不断细化直到能够识别特定的数据库及其应用。

第二种策略是：自下向顶(Bottom－Up)——从特殊到一般的开发策略。这种策略首先从各种基本数据与业务入手，设计出各个子系统；然后将各子系统集中形成整个信息系统。

不管是哪一种设计策略，对于数据库的设计，都应按照下面三个方面进行。

1. 数据库的设计

1978 年 10 月召开的新奥尔良(New Orleans)会议提出了数据库设计的四个步骤：

(1)用户需求分析

根据用户对数据库的：信息要求、处理要求、系统要求(安全性要求、使用方式要求、可扩充要求等)，进行功能分析和利用数据分析图(DFD)进行数据流分析。

(2)信息分析和定义(概念设计)

根据需求分析所获得的信息，使用 E－R 图设计出反映系统信息所需要的数据库概念结构，即概念模式。概念模式独立于 DMBS。

(3)设计实现(逻辑设计)

把概念设计所得到的数据库模型转换为具体的 DBMS 所能接受支持的数据库的逻辑结构，包括数据库模式和外模式，即将 E－R 图转换为关系模式。

(4)物理设计

根据具体的计算机系统(DBMS、硬件系统等)特点，为给定的数据模型确定合理的存储结

构和存储方法。

2. 应用程序设计

在进行数据库设计的同时,应根据需求分析所得到的信息进行应用程序的设计,一般需要进行四种应用程序的设计:

(1)设计用户界面程序

如:开始和结束屏、输入输出表单、工具栏和菜单等。

(2)设计事务处理程序

如:查询、统计和计算等。

(3)设计输出形式和界面

如:浏览、排序、报表、标签、图形等。

(4)确定主程序

主程序是应用程序的入口点,必须确定好。

3. 调试测试和发布

做好应用系统的整体调试、测试、编译、连编和应用系统的发布。

值得说明的是根据全面质量管理(TQM)和软件工程的思想,应用系统的发布不是应用系统开发的结束。恰恰相反,这只是"万里长征走完了第一步",后面还需要投入大量的人力、物力和财力,进行应用系统的维护、完善、扩充、升级、换代。所以上面给出的三个方面的工作,都具有周而复始的特点。

14.1.2 使用项目管理器开发应用程序

Visual FoxPro 9.0 的数据库应用程序,会涉及到各种各样的文件,对这些文件进行人工管理会对应用程序的开发者带来非常大的麻烦,甚至会引起误操作,造成难以弥补的损失。所幸的是为了规范开发过程和提高开发效率,Visual FoxPro 9.0 提供了项目管理器来管理应用程序的开发过程。

关于项目管理器,在第 2 章已做过介绍。利用项目管理器开发数据库应用系统,可按创建项目、添加组件、设置主文件、设置项目信息、连编项目、发布应用程序六个步骤进行。

14.2 编译应用程序

创建应用程序的最后一步就是将程序编译连接成应用程序或可执行程序,然后将其发布。

14.2.1 创建主程序

主程序是整个应用程序的入口。它是将项目编译成为可执行文件所调用的第一个程序。

在 Visual FoxPro 9.0 中,主程序可以是.prg 的程序文件,也可以是菜单、表单等。但一般情况下,经常使用.prg 的程序文件。在该文件中,命令一般为应用程序运行环境的设置、声明系统所必须的全局变量、显示系统启动时的用户界面、控制事件循环、退出应用程序时关闭打开的文件及恢复系统环境的设置等。

1. 设置应用程序运行的应用环境

Visual FoxPro 9.0 的应用程序的运行环境的设置，由一系列 SET 命令组成。例如，关闭对话的命令为：

```
SET TALK OFF
```

2. 声明应用程序所需的全局变量

在主程序中使用 PUBLIC 语句声明应用程序运行所需的全局内存变量。不过应注意，从软件工程的思想出发，程序内各模块内聚度越高越好，而模块间的耦合度越低越好，因此尽量少用全局内存变量而多用局部内存变量。这样做便于程序调试，也不会因一不小心使全局变量出错而影响整个程序的执行。

3. 显示用户界面

主程序执行后紧接着就是用户界面的显示，表示此应用程序系统已经正式启动。用户界面一般由一个表单来完成。表单显示有欢迎使用该应用程序系统、程序的版本、开发商等信息，同时提供对用户合法性检验的操作。例如，在设置了应用程序的环境和全局变量之后，使用：

```
DO mainmenu.mpr
DO FORM start.scx
```

分别启动主菜单和开始表单。

4. 控制事件循环

当用户初始用户界面显示后，需要创建一个事件的循环，等待用户交互地使用该应用程序。控制事件循环的命令是：

```
READ EVENTS
```

该命令可以使 Visual FoxPro 9.0 开始处理诸如鼠标单击等用户事件。从执行 READ EVENTS 命令开始，直到执行 CLEAR EVENTS 命令，这期间主程序的所有操作被全部挂起。由此可见，将 READ EVENTS 命令正确地放置在主程序的合理位置，十分重要。例如，将它作为初始化运行环境的最后一条命令，在初始化环境并显示了用户界面后执行。如果在初始化运行环境设置中无此命令，那么应用程序将返回到操作系统中，使用户无法交互地使用应用程序。

在启动了事件循环之后，应用程序将处在最后显示的用户界面元素的控制之下。例如，如果在主程序中执行了下面两条命令：

```
DO FORM start.scx
READ EVENTS
```

则系统将显示 start 表单。

如果在主程序中没有包含 READ EVENTS 命令，在原开发环境中可以正确地运行应用程序，但如果在菜单或主屏幕中运行应用程序，持续可能显示片刻，然后会自动退出。

与 READ EVENTS 命令相对应，在应用程序中也需要一个结束事件循环的命令。否则，程序将陷入死循环而无法退出。结束事件循环的命令是：

```
CLEAR EVENTS
```

例如，将该命令写在一个“退出”按钮或菜单中，当系统执行了此命令后，将挂起 Visual

FoxPro 9.0 的循环处理过程，同时将控制权返回给主程序，开始执行主程序中 READ EVENTS之后的命令。

5. 主程序实例

下面给出一个主程序 main.prg 文件的实例。这里假设主菜单：mainmenu.mpr 中除其他菜单项外，还有一个“退出”菜单选项。该选项是一个过程选项，该过程由下面三条命令组成：

```
SET SYSMENU NOSAVE          && 将菜单的缺省配置恢复成 VFP9.0 的标准配置
SET SYSMENU TO DEFAULT      && 与上条命令配合使用恢复系统的缺省菜单
CLEAR EVENTS                && 事件结束循环
```

同时，假设开始菜单已设计好，菜单名为：start.scx。主程序代码为：

```
* 这是主程序：main.prg
* 声明全局变量
PUBLIC cUserName
DO setup                    && 调用环境设置子程序
DO FORM start               && 调用开始表单
READ EVENTS                 && 开始事件循环
DO mainmenu.mpr
DO clearup
RETURN
PROCEDURE setup
    * 系统环境设置
    CLEAR ALL
    CLEAR WINDOW
    CLEAR
    SET TALK OFF
    SET SAFETY OFF
    SET STATUS BAR OFF
    SET SYSMENU OFF
    SET SYSMENU TO
    SET CENTURY ON
    SET DATE MYD
ENDPROC
PROCEDURE clearup
    * 恢复系统环境设置
    SET SAFETY ON
    SET STATUS BAR ON
    SET CENTURY OFF
    SET DATE YDM
    * 关闭所有打开的文件
```

```
    CLOSE ALL
    SET SYSMENU TO DEFAULT
    SET TALK ON
ENDPROC
```

14.2.2　隐藏 Visual FoxPro 9.0 主窗口

使用 Visual FoxPro 9.0 开发应用程序，它默认显示的是 Visual FoxPro 9.0 的主窗口。该窗口可用来方便地装载菜单、用户自定义工具栏等。但是，大多数应用程序都会创建一个非常人性化的漂亮的表单作为自己的主窗口，并利用该窗口来装载用户自定义的菜单和工具栏。这样就会在屏幕上出现两个主窗口，从而影响界面的美观。解决这一问题的方法是将 Visual FoxPro 9.0 的主窗口隐藏起来。

隐藏 Visual FoxPro 9.0 主窗口最常用的方法是修改 VFP 的配置文件 Config.fpw。步骤如下：

S1：修改 VFP 的配置文件，在其中添加上命令：

```
SCREEN＝OFF
```

S2：将修改后的 Config.fpw 文件，添加到项目管理器的“其他”选项卡的“其他文件”目录中。

Visual FoxPro 9.0 的主菜单被隐藏后，又一个问题一定要引起注意。这就是当应用程序中只显示一个表单时，一定要保证该表单的 ShowWindow 属性值为“2 -作为顶层表单”。否则，如果将表单的该属性值设为：“0 -在屏幕中”或设为：“1 -在顶层表单中”，这实质上是将 Visual FoxPro 9.0 的主窗口作为载体，那么当 Visual FoxPro 9.0 的主窗口被隐藏时，表单也将被隐藏。

14.2.3　设置文件的排除与包含

1. 排除与包含的概念

在一个项目中会包含各种类型的文件，这些文件在项目中的位置和存取方式是不同的。例如，数据库表是项目中的数据部分，它是动态的，随着对于数据的存取会发生变化。表单和菜单在程序运行中，一般不需要改写，它是项目中的命令代码部分，是静态的。因此在编译之前需要将动态数据排除在项目外，而将静态数据包含于项目内。即对于动态数据不进行编译，而对于静态数据则进行编译。

在项目中，哪些文件应包含在项目之内，哪些文件又应排除在项目之外，一般选取的原则是：所有包含可执行代码的文件应包含在项目之内，例如表单、报表、菜单、查询、程序文件、过程文件等；所有数据文件应排除在项目之外。但是，也可以根据应用程序的需要包含或排除文件。例如，如果一个文件含有敏感的系统信息或含有仅用作查询的信息，则这样的文件状态应被设置为“包含”，以免稍不留神而被修改。反之，如果应用程允许用户修改一个报表，则该报表文件的状态则又应设置为“排除”。通常，将不需要用户更新的文件设置为“包含”，应用程序文件(.app)不能设为“包含”，对于类库文件(.ocx，.fll，.dll)可以有选择地设置为“排除”。

在项目管理器中，对于包含可执行代码的文件，缺省状态为包含；而对于所有的数据库文件和表文件，缺省状态则为排除。

在项目管理器中，文件的排除标记为“Φ”，当一个文件之前有一个前缀“Φ”时，则表示该文件已被排除。

在“项目信息”对话框中，文件若用“×”标注，表示包含，空表示为排除。

2. 将排除文件设置为包含文件

在项目管理器中，将排除文件设置为包含文件的方法很简便，只要在选定该文件后，再右击鼠标，打开快捷菜单，选择“包含”即可。如图 14－1 所示。

图 14－1　设置文件为包含

利用项目菜单中的“包含”选项卡也可将排除文件设置为包含文件。

3. 将包含文件设置为排除文件

要将包含文件设置为排除文件，方法同上，只不过选择文件时应选择未加排除标记的文件。这时，快捷菜单或“项目”菜单中将显示出“排除”选项供选择。如图 14－2 所示。

图 14－2　设置文件为排除

14.2.4　设置项目信息

一个项目中通常会包含多种信息，这些信息的设置一般用“项目信息”对话框来设置。设置步骤如下。

S1：打开“项目信息”对话框。在项目管理器的目录树列表中单击右键，打开快捷菜单，在

快捷菜单中选“项目信息”，或在 Visual FoxPro 9.0 主菜单中打开“项目”菜单，选“项目信息”选项卡。打开后的“项目信息”对话框，如图14－3所示。

图 14－3　项目信息对话框的项目选项卡

S2：信息填写。“项目信息”对话框包含三个选项卡，用户应分别填写其中的各选项的内容。

S2－1：填写“项目”选项卡。图 14－3 显示的是“项目”选项卡的各种选项，用来设置与创建人有关的信息，用户可根据情况予以填写。

S2－2：填写“文件”选项卡。“文件”选项卡如图 14－4 显示，用来设置文件的包含于排除。无需赘述。

图 14－4　项目信息对话框的文件选项卡

S2－3：填写“服务程序“选项卡。

14.2.5　编译应用程序

在项目管理器中，单击[连编(Build)]，打开“连编选项(build options)”对话框，如图14－5

所示。连编对话框包含一个“建立操作”单选框和一个“选项”复选框。

图 14-5 连编选项对话框

1.“建立操作”单选框

在该单选框中可用来设置 Visual FoxPro 9.0 所能进行五种类型的编译的任一种,其意义分别如下:

① 重新连编项目:可用来发现项目中的错误。例如在程序调用了一个并不存在的过程等。

② 应用程序(app):将项目编译成应用程序。应用程序必须在 Visual FoxPro 9.0 环境中才能运行。

③ Win32 可执行程序/COM 服务程序(exe):将项目编译成可执行文件,这是最常用的编译类型。

④ 单线程 COM 服务程序(dll):将项目编译成单线程的 DLL 程序。这将使用项目文件中的类信息,创建一个动态链接库。

⑤ 多线程 COM 服务程序(dll):将项目编译成多线程的 DLL 程序。

2.“选项”复选框

在该复选框中,提供了如图 14-5 所示的四个选项可供选择。它们的意义分别为:

① 重新编译全部文件(C):重新编译所有的文件。

② 显示错误(D):编译时如发现错误,将在编译完成后显示出来。

③ 连编后运行:编译结束后立即运行程序。

④ 重新生成组件的 ID(G):重新生成组件 ID。只有当选定了“Win32 可执行程序/COM 服务程序(exe)”并已连编包含 OLEPublic 关键字的程序时,才能选择此项。

3. 将项目编译成可执行文件

将项目编译成可执行文件是最常用的编译方式。而在可执行程序中,可以包含关于程序的许多信息。这些信息将用户在“建立操作”单选框中选择了“Win32 可执行程序/COM 服务程序(exe)”项后,通过单击[版本]后打开的“版本”对话框来设置。版本对话框如图 14-6 所示。

版本对话框可用来设置可执行程序的版本号、文件名称、语言 ID、注册商标、注册版权等信息。

图 14-6　“版本”对话框

当设置完成后，在连编选项对话框，单击[确定]，即开始编译。

连编应用程序也可以使用连编命令 BUILD 来进行，连编命令的格式可以是下面三种之一，分别用来连编生成.exe，.app，.dll 文件：

BUILD APP APP*FileName* FROM *ProjectName* [RECOMPILE]

BUILD EXE EXE*FileName* FROM *ProjectName* [RECOMPILE]

BUILD DLL DLL*FileName* FROM *ProjectName* [RECOMPILE]

例如，将项目 xsxjgl.pjx 连编成可执行文件“学生学籍管理.exe”，则应使用命令：

BUILD EXE 学生学籍管理 FROM xsxjgl

14.3　发布应用程序

将项目文件连编生成可执行程序后，就可以将应用程序予以发布。在发布应用程序之前，应做好下面几方面的预备工作。

1. 选择合适的发布类型

发布类型一般和编译时的选择类型基本相同。主要包括应用程序(.app)、可执行程序(.exe)和动态链接库(.dll)。

2. 充分考虑程序的运行环境

应考虑和测试应用程序的最小运行环境，如磁盘空间、内存需求等。测试的结果有助于正确选择合适的编译类型、应用程序中应包含的文件级及建立发布结构的方法。

3. 确保程序能够正确运行

在将应用程序发布给用户之前，必须保证应用程序能够正确地运行，否则交给用户的程序必将是一个失败的短命的垃圾程序，而没有应用价值及生命力。

4. 妥善保护好源代码

源代码从某种意义上说是程序开发人员的命根子，必须妥善予以保护。只有做好了对源代码的保护，才既可以防止对源代码的非法修改，又可以防止盗版有效保护程序开发者的合法权益，同时为应用程序的升级和改进提供可能。

要防止对源代码的非法修改，在认真做好源代码的备份(通常不止一份备份)后，再加密源

程序并删除调试信息。

加密源程序的步骤如下：

S1：在项目管理器中打开“项目信息”对话框。

S2：在“项目信息”对话框的“项目”选项卡中，选择“调试信息”和“加密”两项，如图 14－7 所示。

图 14－7 “项目信息”对话框

S3：在主程序中加入：“SET DEBUGOFF”命令。

S4：在“项目管理器”中单击[连编]，打开“连编选项”对话框，在“选项”复选框中选“重新编译全部文件”复选框，然后单击[确定]，进行重新编译。

5. 必要时可更改程序图标

在项目文件进行连编生成可执行程序后，程序的默认图标是 Visual FoxPro 9.0 的程序图标。用户也可以根据需要自行设计图标。步骤如下：

S1：创建包含两幅图像的图标文件，其中一个图像的大小为 16×16，另一个为 32×32，将这两个图标设置为 16 色。

S2：在“项目管理器”中选“附加图标”复选框，在打开的“打开”对话框中选择一个图标文件。

S3：将项目连编成可执行程序。

这样，当运行可执行程序时，就使用的是用户自定义的图标。

6. 定制要发布的文件

项目文件被连编成可执行程序后，所有生成的文件都被直接保存在项目文件所在的目录下。然而，实际上应该提供给用户的仅是可执行程序和数据库等相关的文件。因此，在连编完成之后，应将可执行程序和数据库等相关文件按照原来的目录结构保存到另一个文件夹下，以备发布时使用。

需要发布的文件除了程序文件、数据库文件外，还应包括如下文件：

① Visual FoxPro 9.0 运行库。这是最重要的一个文件，因为使用 Visual FoxPro 9.0 所开发的可执行应用程序的运行，是需要运行 Visual FoxPro 9.0 动态链接库的支持的。通常该动态链接库位于 C:\Program Files\Common Files\Microsoft Shared\VFP 目录下。使用这些动态链接库最简单的方法是将它们复制到要发布的程序所在的目录下，与程序一块发布。

② FOXUSER 资源文件。它包括两个文件：FOXUSER.DBF，FOXUSER.FPT。它们存放着一些对应用程序有用的信息，包括窗口位置、浏览窗口以及标签等。如果应用程序需要使用这些资源项，则必须在项目中加入这两个文件。

③ 外部库。

④ COM 组件。

⑤ 配置文件 CONFIG.FPW。

⑥ 用于特定地区的资源文件。

⑦ 自定义的文件。

当上述工作完成后，就可以将应用程序交付给用户使用，这个过程称为发布。发布的方法很多，最简单、最直接的方法是将需要发布的文件拷贝给用户。也可使用 Visual FoxPro 9.0 的“安装向导”来完成。

14.4　思考与练习

一、选择题

1. 下列文件中，不能用来作为项目的主文件是(　　)。

A).prg 程序　　B)表单　　C)报表　　D)菜单

2. 在 Visual FoxPro 中，不能将程序编译连接成(　　)文件。

A).exe 文件　　B).dll 文件　　C).app 文件　　D).txt 文件

3. 把一个项目编译成一个应用程序时，正确的叙述是(　　)。

A) 所有的项目文件将组合成一个单一的应用程序

B) 所有项目的包含文件将组合成一个单一的应用程序

C) 所有项目的排除文件将组合成一个单一的应用程序

D) 由用户选定的项目文件将组合成一个单一的应用程序.txt 文件

4. 在 Visual FoxPro 应用程序的入口点的主程序中，至少应具有(　　)功能。

A) 初始化环境

B) 初始化环境、显示用户初始界面

C) 初始化环境、显示用户初始界面、控制事件循环

D) 初始化环境、显示用户初始界面、控制事件循环、退出时恢复环境

二、填空题

1. 在 Visual FoxPro 中隐藏了 Visual FoxPro 主窗口后可以作为主窗口的表单的 ShowWindow 属性值为_____。

2. 在 Visual FoxPro 中开始事件循环要使用_____语句，结束事件循环要使用_____语句。

3. 在应用程序生成器的“常规”选项卡中，选择程序类型时若选中“顶层”，将生成一个_____。

4. 进行应用程序发布时，除了数据库文件、程序文件外，一般还应包括七种文件。这七种文件中的 FOXUSER 资源文件包括有_____、_____文件。

三、上机题

1. 创建一个项目，使用一个.prg 程序作为主文件，创建一个主表单，将其编译成.exe 文件。

2. 为上面的项目设定一个用户喜爱的图标，然后将其编译成一个.exe 文件。

3. 隐藏 Visual FoxPro 主窗口，然后再将其编译成.exe 文件。

第 15 章　应用系统设计举例

学习的目的在于应用，前面的每一章节，给出了一些关于 Visual FoxPro 9.0 的基本操作。但仅有这样的讲授，学生获得的知识将是支离破碎的，只有通过对实际应用系统的开发研制，哪怕是一个很小的实际应用系统的开发，才能使所学的知识达到融会贯通。本章将以“学生学籍管理系统”为例，介绍 VFP 应用系统的研制过程。

15.1　系统需求分析

需求分析是系统开发的第一步。通常的情况是：程序设计者不了解用户业务，用户可能对计算机能干什么知之甚少。因此在系统开发前，程序设计者和用户相互学习和交流显得特别重要，通过学习和交流，开发者能熟悉用户的业务范围和工作进程，用户也能从计算机的角度提出对应用系统的要求。解决系统开发“做什么”、“怎样做”的问题，为抽象系统功能作好了准备。下面是“学生学籍管理系统”的需求分析结果。对于读者来说，可以先按顺序完成系统，再了解需求分析。

15.1.1　用户能提供的信息

1. 学生基本情况表(学号、姓名、性别、出生日期、政治面貌、应往届生、家庭住址、奖惩情况、照片)。

2. 课程代码表(课程代码、课程名字、课程学时、开课学期)。

3. 学生成绩表(学号、所选各门课程……)。

4. 学生综合积分表(学号、精神文明、社会活动、体育锻炼)。

15.1.2　用户需求信息

1. 学籍档案卡(包括学生基本情况表中所有内容及学生所有课程成绩)。

2. 学生成绩表(包括学生学号、n 门课成绩、成绩合计和智育排名)，其中：

$$\text{成绩合计} = \frac{\sum_{i=1}^{n}(\text{成绩 } i \times \text{课程学时 } i)}{\sum_{i=1}^{n} \text{课程学时 } i}$$

(1) n 为该学期所开课程门数。

(2) 智育排名按成绩合计由高到低降序排序，当成绩合计相等者，排名次也相同。

3. 学生综合积分表(学号、精神文明、社会活动、体育锻炼、素质总分、素质排名、综合积分、

综合排名),其中:

(1) 素质总分=精神文明×0.5+社会活动×0.2+体育锻炼×0.3。

(2) 综合积分=(成绩合计×0.8)+(素质总分×0.2)。

(3) 综合排名根据综合积分由高到低排为 1,2,…,n 名。对综合积分相同者,排名相同。

15.2 系统功能

应用系统功能模块体现了系统的菜单组成。"学生学籍管理系统"系统功能模块组成如图 15-1 所示。由于只是一个教学演示系统,所以功能模块划分比较简单。

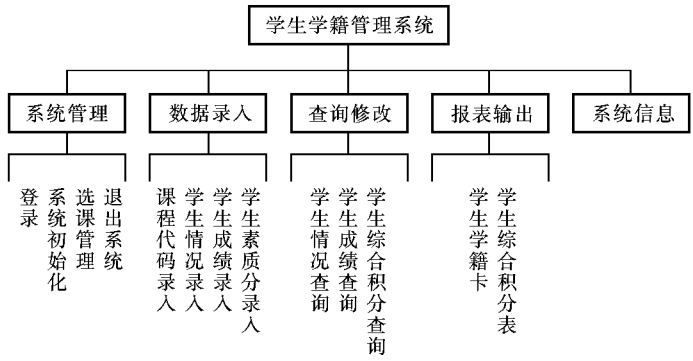


图 15-1 系统功能结构图

15.3 数据库与数据表设计

本系统所用到的数据表并不复杂,可用自由表完成,但为了充分展示数据库表(Database Table)的功能,所涉及到的数据表均属于 xjsjk 数据库,并将数据库及除学生成绩表以外的所有数据表均保存于 C:\xsxjglxt\dbf 文件夹中。学生成绩表有可能因选课不同而结构有所不同,所以该表以自由表的形式出现在项目文件中。为了便于让读者了解整个系统的开发情况,本例将各数据表的数据与表结构一起显示出来,见表 15-1 至 15-11。

表 15-1 学生基本情况表结构(表文件名:xsqkb.dbf)

序号	字段名	类型	宽度	小数位	备注
1	学号	字符型	6		主索引(索引名为 xh)
2	姓名	字符型	8		
3	性别	字符型	2		
4	出生日期	日期型	8		
5	政治面貌	字符型	2		
6	应届否	逻辑型	1		
7	家庭住址	字符型	30		
8	奖惩情况	备注型	4		
9	照片	字符型	15		

表 15－2　学生基本情况表记录

学号	姓名	性别	出生日期	政治面貌	应往届生	家庭住址	照片
010001	张凯华	男	02/13/1980	团	T	渭南市站南路 24 号	Zp01.bmp
010102	李会琴	女	07/25/1979	党	T	杨凌区西农路 59 号	
010111	李小茜	女	15/21/1979	团	F	宝鸡市红旗路 103 号	
011516	宋秀兰	女	10/31/1980	团	F	延安市枣园大街 5 号	
011517	郭正宏	男	11/25/1979	团	T	渭南市东风街 106 号	
011320	姜亚男	女	09/30/1980		F	咸阳市世纪大街 108 号	
015001	杨书敏	女	08/18/1980		T	延安市宝塔路 214 号	
015002	宋越辉	男	06/09/1979	团	T	宝鸡市红旗路 265 号	
015003	杜拥军	男	05/20/1980		F	杨凌区西农路 132 号	
015234	王向东	男	04/26/1979	党	T	咸阳市阳街 10 号	

注：照片字段中存放的不是图片，而是照片图片的文件名及扩展名。具体图片文件存入在特定文件夹下。

表 15－3　课程代码表结构(kcdmb.dbf)

序号	字段名	类型	宽度	小数位	备注
1	课程代码	字符型	8		主索引(索引名为 kcdm)
2	课程名称	字符型	18		
3	课程学时	字符型	2		
4	开课学期	日期型	8		

表 15－4　课程代码表记录

课程代码	课程名称	课程学时	开课学期
J8080101	C＋＋程序设计	90	1
J8080102	VFP 程序设计	72	2
J8080103	操作系统	72	2
J8080104	数据结构 C＋＋描述	72	3
J8080105	自动控制原理	54	3
J8080106	计算机网络	54	3
J8080107	微机原理	72	4
J8080108	计算方法	54	4
J8080201	多媒体技术	54	4
J8080202	网页制作	54	5
J8080109	计算机通信	54	5
J4080101	高等数学	180	15
Y3080101	大学英语	300	1534
Y3080102	专业外语	40	6
J8080110	计算机文化基础	72	1
J8080111	编译原理	72	6
J8080115	接口技术	54	6
J8080113	计算机图形学	72	7

续表 15－4

课程代码	课程名称	课程学时	开课学期
J8080114	图像处理技术	72	7
J8080115	软件工程	54	7
J8080203	计算机组装与维护	54	7
J8080204	网络数据库	54	8
J8080205	JAVA 程序设计	54	8
W5080101	大学物理	180	15

表 15－5　学生成绩表(xscjb.dbf)

字段	字段名	类型	宽度	小数位	备注
1	学号	字符型	6		普通索引(索引名为 xh)
N	*N* 门课代码	数值型	6	1	
N＋1	成绩合计	数值型	6	1	
N＋2	智育排名	整型	4		

表 15－6　学生成绩表记录 (课程任选,成绩随机产生)

学号	J8080101	J8080102	J8080203	Y3080101	W5080101	J8080204	成绩合计	智育排名
010101	83.0	85.0	76.0	62.0	70.0	67.0		
010102	76.0	88.0	67.0	65.0	77.0	70.0		
010111	69.0	79.0	52.0	70.0	61.0	74.0		
011516	73.0	69.0	77.0	76.0	69.0	66.0		
011517	57.0	62.0	72.0	68.0	73.0	58.0		
011320	67.0	69.0	80.0	53.0	76.0	68.0		
015001	90.0	77.0	73.0	65.0	82.0	72.0		
015002	83.0	55.0	74.0	69.0	88.0	73.0		
015003	66.0	83.0	91.0	70.0	56.0	69.0		
015234	72.0	82.0	86.0	72.0	73.0	65.0		

表 15－7　学生综合积分表(xszhjfb.dbf)

字段	字段名	类型	宽度	小数位	备注
1	学号	字符型	6		普通索引(索引名为 xh)
2	精神文明	数值型	6	1	
3	社会活动	数值型	6	1	
4	体育锻炼	数值型	6	1	
5	素质总分	数值型	6	1	
6	素质排名	整型	4		
7	综合积分	数值型	6	1	
8	综合排名	整型	4		

表 15－8　学生综合积分表记录

学号	精神文明	社会活动	体育锻炼	素质总分	素质排名	综合积分	综合排名
010001	89.0	90.0	76.0	0.0	0	0.0	0
010102	90.0	91.0	88.0	0.0	0	0.0	0
010111	87.0	79.0	95.0	0.0	0	0.0	0
011516	76.0	85.0	79.0	0.0	0	0.0	0
011517	92.0	88.0	91.0	0.0	0	0.0	0
011320	96.0	85.0	79.0	0.0	0	0.0	0
015001	87.0	91.0	85.0	0.0	0	0.0	0
015002	85.0	79.0	84.0	0.0	0	0.0	0
015003	90.0	87.0	90.0	0.0	0	0.0	0
015234	96.0	88.0	92.0	0.0	0	0.0	0

表 15－9　操作员姓名及口令表(czy.dbf)

字段	字段名	类型	宽度	小数位
1	姓名	字符型	6	
2	密码	字符型	6	

表 15－10　操作员姓名及口令表记录

姓名	密码
张三	111111
李四	222222
王五	333333

表 15－11　数据表结构延伸文件(dbfstr.dbf)

字段	字段名	类型	宽度	小数位	索引	排序	Nulls
1	FIELD_NAME	字符型	158	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			
15	TABLE_NAME	字符型	158	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			

15.4 应用系统框架的建立

一个应用系统的设计，往往采用“自顶向下，逐步细化”的结构化程序设计方法，本系统也采用这种设计思想，先设计出系统框架，后完成各模块的设计任务。在 VFP 中，建立应用系统的框架一般包括以下几个问题：

① 建立项目文件。

② 建立菜单文件。

③ 建立设置主文件。

④ 编译项目文件，生成.exe 文件。

15.4.1 建立应用系统文件夹

开发一个应用系统，应该首先建立一个应用系统文件夹，以免与其他文件发生混淆。一个数据库应用系统一般由报表文件(.frx)、程序文件(.prg)、表单文件(.scx)、数据表文件(.dbf)、说明性文档(.doc)、图片文件(.bmp,.jpg)等组成，所以还应建立应用系统文件夹的下级文件夹，用来分别存放相应文件。

本例在 C 盘根目录下建立 xsxjglxt 文件夹作为应用系统文件夹，在 xsxjglxt 下建立 frx,prg,dbf,scx,doc,pic 六个二级文件夹，用于分别存放各种类型的文件。建成后的目录结构如图 15-2 所示。这些文件夹中分别安排如下：

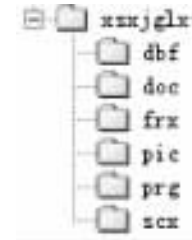


图 15-2 系统目录结构

① xsxjglxt 文件夹下存放项目文件及后期编译好的.exe 文件。

② dbf 文件夹下存放数据库文件及所有数据表文件。

③ doc 文件夹下存放菜单文件及系统各种说明、帮助文件。

④ frx 文件夹下存放所有报表格式文件。

⑤ prg 文件夹下存放所有的程序文件。

⑥ scx 文件夹下存放所有的表单文件。

⑦ pic 文件夹下存放所有的图片文件。

当然，每一个人在设计应用系统时都会有自己的风格。将文件分类存放管理起来会更加方便。

15.4.2 设置默认工作目录及搜索路径

启动 VFP，在命令窗口分别输入：

```
SET DEFAULTO TO
SET PATH TO c:\xsxjglxt\prg,dbf,doc,frx,scx,pic
```

15.4.3 建立项目文件

项目文件能有效地组织管理一个应用系统中所包含的表、数据库、查询、表单、报表及应用程序。设项目文件名为：xsxjglxt.pjx，则应在命令窗口输入下面命令：

```
CREATE PROJECT xsxjglxt
```

建好项目文件后，系统中包含的所有文件一般均能通过项目管理器建立或编辑。如果项目文件提前已经建好，则可使用项目文件打开命令：

MODIFY PROJECT xsxjglxt

打开项目。

15.4.4　建立数据库和数据表

① 建立数据库：

CREATE DATABASE c:\xsxjglxt\dbf\xjsjk

② 建立数据表。根据表 15－1 和表 15－2 创建学生情况表：xsqkb.dbf。

CREATE TABLE xsqkb(学号 C(6) PRIMARY KEY，姓名 C(8)，性别 C(2)，出生日期 D，政治面貌 C(2)，应届否 L，家庭住址 c(30)，奖惩情况 M，照片 C(10))

INSERT INTO xsqkb (学号，姓名，性别，出生日期，政治面貌，应届否，家庭住址，照片) VALUES ('010001'，'张凯华'，'男'，{^1980/02/13}，'团'，.T. ，'渭南市站南路 24 号'，'zp01.bmp')

反复使用 INSERT 命令将所有记录都录入。当然也可以使用 BROWSE 命令等方法录入记录。此时，用户还可以录入备注型字段和通用型字段的值。

③ 重复第 2 步，继续完成 kcdmb、xszhjfb 两个数据表的建立，将 xscjb 放入自由表中，再用命令

COPY STRUCTURE EXTENDED TO dbf\dbfstr

将其结构复制生成数据表结构文件 dbfstr 放入 dbf 文件夹中，最后，将表 dbfstr.dbf 添加到自由表中。

15.4.5　建立菜单文件

应用系统的菜单项目是根据系统功能结构图来建立的步骤如下。

S1：CREATE MENU cd。

S2：创建一级菜单：按系统功能构成，输入一级菜单及快捷键。完成后结果如图 15－3 所示。

图 15－3　用菜单设计器设计的一级菜单

S3：创建二级菜单(以系统管理为例)：选中“系统管理”，单击“创建”按钮，出现二级菜单编辑画面，按系统功能构成中的内容输入第二级菜单，完成后结果如图 15－4 所示。在各个菜

单项中分别写入了命令，其中有的执行表单（如“do form”登录），有的执行程序（如“do 系统初始”），有的直接写入了命令（如“退出系统”之后的“QUIT”命令）。

图 15－4　二级菜单及其命令

S4：重复第 3 步，完成其他二级子菜单建立。

S5：单击“预览”按钮，预览菜单的设计效果。不满意时可对菜单进行修改。

S6：为了防止非法用户使用该系统，系统设置了“登录”菜单，其作用是只有先执行登录表单，在登录表单中输入合法的操作员姓名及密码后，才可以使用系统的其他菜单项。处理办法为：

① 在一级菜单项中除“系统管理”外，对所有菜单项的选项进行修改，如图 15－5 所示，在“跳过”的文本框中录入表达式“mm＝.f.”，即当变量 mm 的值为假时，菜单项失效。

图 15－5　为系统菜单项设置条件

② 在“系统管理”的二级菜单中，除“登录”和“退出”两项外，对其他菜单项的选项也进行修改，同样，在其“跳过”的文本框中录入表达式“mm＝.f.”，即当变量 mm 的值为假时，菜单项失效。

S7：在系统菜单中选择“菜单”→“生成”，在“生成菜单”对话框中输入文件名，单击“产生”后，将菜单文件编译 MPR 文件，供主程序调用。

15.4.6　建立并设置主文件

主文件是一个应用系统程序运行的入口，在 VFP 中，一个应用系统的入口可以是一个程序文件，也可以是一个表单等。一般以程序文件为多，文件名习惯上为 main.prg，内容基本相似。

本系统的主文件名为 main.prg，内容为：

```
* 这是主程序：main.prg
* 声明全局变量
PUBLIC mm                    && 设置全局变量用于控制菜单条有效与否
ON SHUTDOWN QUIT
DO setup                     && 调用环境设置子程序
READ EVENTS                  && 开始事件循环
DO FORM strat                && 调用开始表单
DO clearup
RETURN
PROCEDURE setup
    * 系统环境设置
    CLEAR ALL
    CLEAR WINDOW
    CLEAR
    SET TALK OFF
    SET SAFETY OFF
    SET STATUS BAR OFF
    SET SYSMENU OFF
    SET SYSMENU TO
    SET CENTURY ON
    SET DATE MYD
    _SCREEN.WindowState =2
    _SCREEN.Caption="学生学籍管理系统"
ENDPROC
PROCEDURE clearup
    * 恢复系统环境设置
    SET SAFETY ON
    SET STATUS BAR ON
    SET CENTURY OFF
    SET DATE YDM
    * 关闭所有打开的文件
    CLOSE ALL
    SET SYSMENU TO DEFAULT
```

```
    SET TALK ON
ENDPROC
```

在项目管理器的“代码”中新建程序 main.prg，并输入上述内容，存盘后，在这个文件名上单击鼠标右键，将这个文件设置成主文件。当一个文件被设置成主文件后，这个文件名字符将变成粗体字。

15.4.7 编译项目文件

在 VFP 中，一个应用系统有了菜单文件，有了主文件，这个应用系统的框架就构成了，也就可以编译成 EXE 文件并在 WINDOWS 下直接运行了。

在项目管理器中单击“连编”，出现图 15－6 所示对话框。用户可以根据自己的需要进行选择，这里的选择如图所示。再单击“确定”按钮后，由于在菜单中写入的各种表单、程序等文件还没有建立，所以编译过程中会出现如图 15－7 所示对话框，选择“全忽略”后，可生成 xsxjglxt.exe 文件。

图 15－6 连编项目选项

图 15－7 定位文件消息框

退出 VFP，从资源管理器中双击 xsxjglxt.exe 文件，即可看到程序运行结果，如图 15－8 所示。从图中可以看到，除了“登录”和“退出系统”两个菜单项能使用外，其他的菜单项均不能使用。当合法登录后，将全局变量 mm 的值改为.t.，就可以使用其他的菜单项了。如果要对操作员的操作权限进行更多层次的限制，可通过设置不同的变量来完成。

图 15－8 系统运行界面

一个应用系统需要多次调试运行，所以建议读者在练习时先在桌面上建立一个快捷方式。

至此，应用系统的框架已建立，剩下的任务就是为各菜单项建立相应的程序或表单了。

15.5　表单设计

本节以本章 15.2 中系统功能组成内容为序，逐一介绍每一个菜单项对应的表单或程序代码设计。为了便于介绍，每一个表单均先给出结果，再介绍其制作过程和其中每一个对象被修改的属性。各表单的布局读者在实习过程中可以根据自己的爱好重新安排。

15.5.1　系统管理

1. 登录表单

登录表单界面如图 15－9 所示，主要由一个组合框、两个标签、一个文本框和一个命令按钮组成。其中组合框通过生成器生成，数据源来自 czy.dbf 表中姓名字段。事件代码包括：

图 15－9　登录表单布局

(1) 表单的 Init 事件代码

```
with this
    .alwaysontop=.t.
    .autocenter=.t.
    .borderstyle=2
    .caption="登录"
    .maxbutton=.f.
    .minbutton=.f.
endwith
```

(2) 命令按钮的 Click 事件代码

```
IF ALLTRIM(thisform.text1.value)=czy.密码
    mm=.t.                    && 修改全局变量的值使菜单全部有效
ELSE
    =MESSAGEBOX("您无权使用本系统!",0+16,"警告")
    QUIT
ENDIF
Thisform.Release
```

实际应用中，各操作员的密码是可以修改的，这些密码可以存放在数据表中，也可以存放在一个文本文件或其他任意类型的文件中，且密码都经过加密后存放，使用时再解密。

2. 系统初始化

系统初始化主要用于对各个数据表进行清空处理，对各数据表进行初始化是因为在开发应用系统过程中有许多实验数据，在输入真实数据前需要清空。

初始化数据表主要是清空系统涉及到的各数据表，其中包括对学生基本情况表、课程代码表、学生成绩表和学生综合积分表。由于不涉及到界面，所以直接在程序代码中完成。程序文件名为“系统初始.prg”，程序代码如下：

```
SET SAFETY OFF
```

```
CLOSE ALL
msg="本模块将清空学生基本情况表、课程代码表、"+ ;
    CHR(10)+"学生成绩表和学生综合积分表中所有内容！确认吗?"
IF MESSAGEBOX(msg,4+32+256,"注意")=7
    RETURN
ELSE
    usedbf("xsqkb")                    && 调用自定义函数 USEDBF
    ZAP
    usedbf("kcdmb")
    ZAP
    usedbf("xscjb")
    ZAP
    usedbf("xszhjfb")
    ZAP
ENDIF
=messagebox("初始化结束!",0+64,"提示")
RETURN
FUNCTION usedbf
    LPARAMETERS dbf_name
    IF NOT USED (dbf_name)    && USED 为系统函数,确定表是否被打开,若打
                              && 开返回.T.,并将表的工作区置为当前工作区
        SELECT 0
        use (dbf_name) EXCLUSIVE
    ENDIF
ENDFUNC
```

3. 选课管理

一般学校每学期对学生要求既有必修课,也有选修课。本实例从课程代码表中将开课学期中有"2"的课列为必修课,其他课程均列为选修课供学生选择。

表单设计思路为:

① 在表单初始化事件中清空 dbfstr.dbf **表中内容,执行以下程序代码:**

```
usedbf("dbfstr")
ZAP
APPEND BLANK
REPLACE FIELD_NAME with "学号",FIELD_TYPE with "c",FIELD_LEN with 6
usedbf("kcdmb")
SCAN FOR "2"$开课学期
    SELECT dbfstr
    APPEND blank
```

```
    REPLACE FIELD_NAME WITH kcdmb.课程代码,FIELD_TYPE WITH "n",;
FIELD_LEN WITH 6,FIELD_DEC WITH 1
    SELECT kcdmb
ENDSCAN
SET FILTER TO NOT "2"$开课学期        &&将开课学期中有2的课程屏蔽
```

本段程序代码运行后，表 dbfstr 中的记录内容如下：

记录号	FIELD_NAME	FIELD_TYPE	FIELD_LEN	FIELD_DEC
1	学号	c	6	0
2	J8080102	n	6	1
3	J8080103	n	6	1
4	J4080101	n	6	1
5	Y3080101	n	6	1
6	W5080101	n	6	1

② 在表单中用列表框列出所有可选课程代码及名称，供学生选课，将所选课程代码加入表 dfbstr.dbf 中，表单界面如图 15－10 所示，运行结果如图 15－11 所示。程序代码放入命令按钮的单击事件中，代码如下：

图 15－10　表单界面布局

图 15－11　表单运行效果

```
n=0
x=""                &&用于存放被选中的课程代码
m=thisform.list1.listcount
FOR i=1 TO m
    IF thisform.list1.selected(i)=.t.
        n=n+1
        IF n<=3
            x=x+thisform.list1.list(i)+chr(10)+chr(13)
        ELSE
```

```
                MESSAGEBOX("注意！ 您的选择超过了 3 门课,请重新选择!")
                EXIT
            ENDIF
        ENDIF
    NEXT i
    IF n=0
        =MESSAGEBOX("您没有进行任何选择!")
    ENDIF
    ans=MESSAGEBOX("确认您选的课吗?"+chr(10)+chr(13)+x,4+32,"提示")
    IF ans=6
        usedbf("dbfstr")
        FOR i=1 TO m
            IF thisform.list1.selected(i)=.t.
                APPEND BLANK
                    REPLACE FIELD_NAME with thisform.list1.list(i),;
                        FIELD_TYPE with "n",FIELD_LEN with 6,FIELD_DEC with 1
            ENDIF
        NEXT i
        APPEND BLANK
        REPLACE FIELD_NAME with "成绩合计",FIELD_TYPE with "n",;
                FIELD_LEN with 6,FIELD_DEC with 1
        APPEND BLANK
        REPLACE FIELD_NAME with "智育排名",FIELD_TYPE with "n",FIELD_LEN with 4
        USE
        CREATE.\dbf\xscjb FROM dbfstr
        MESSAGEBOX("选课结束!",0+64,"信息")
        USE
        SELECT kcdmb
        SET FILTER TO
        USE
        RELEASE thisform
    ENDIF
```

以后代码执行结束后，表 dbfstr 的记录内容如表 15－12 所示。这些记录，最后通过 create 命令生成学生成绩表。

表 15－12　结构描述文件 dbfstr.dbf 文件记录内容

记录号	FIELD_NAME	FIELD_TYPE	FIELD_LEN	FIELD_DEC
1	**学号**	c	6	0
2	J8080102	n	6	1
3	J8080103	n	6	1
4	J4080101	n	6	1
5	Y3080101	n	6	1
6	W5080101	n	6	1
7	J8080105	n	6	1
8	J8080108	n	6	1
9	Y3080102	n	6	1
10	**成绩合计**	n	6	1
11	**智育排名**	n	4	0

读者在上机实习时，各学生成绩即可以随便输入，也可以通过程序用随机函数生成。为了方使用读者理解后续的程序，下面给出各学生成绩见表 15－13，**限于篇幅成绩合计和智育排名没有列出。**

表 15－13　学生成绩表记录内容

学号	J8080102	J8080103	J4080101	Y3080101	W5080101	J8080105	J8080108	Y3080102
010001	93.0	78.0	96.0	92.0	62.0	89.0	78.0	56.0
010102	69.0	79.0	67.0	84.0	70.0	91.0	60.0	64.0
010111	95.0	94.0	82.0	81.0	76.0	70.0	69.0	55.0
011516	86.0	69.0	65.0	50.0	95.0	62.0	54.0	90.0
011517	97.0	71.0	85.0	52.0	70.0	91.0	91.0	87.0
011320	63.0	98.0	82.0	81.0	59.0	97.0	98.0	52.0
015001	97.0	62.0	57.0	64.0	53.0	56.0	79.0	70.0
015002	64.0	60.0	84.0	62.0	94.0	69.0	73.0	91.0
015003	60.0	62.0	74.0	72.0	72.0	80.0	68.0	68.0
015234	69.0	86.0	69.0	82.0	81.0	75.0	90.0	65.0

代码如下：

```
use xscjb
zap
append from xsqkb field 学号
for i=1 to 10                          && 共 10 条记录
```

```
        go i
        for j=2 to 9                    && 需要添加数据的字段在第 2～9 字段
            x=int(rand() * 51+50)       && 生成 50～100 之间的随机整数
            fld=field(j)                && 通过函数取出字段名
            repl &fld with x            && 填入产生的随机数
        next j
    next i
```

4. 退出系统

退出系统只需用 VFP **命令** QUIT,**所以可在菜单项中直接输入命令。**

15.5.2 数据录入

一个数据库应用系统中的数据录入是非常重要的工作,因为它关系到后续工作的准确性,如果数据录入错误,那数据处理就没有任何意义了。根据录入数据的特点,一般可用两种方式录入数据。

1. 用 Grid 控件录入

这种方式是录入的数据要求数据表字段不能太多,字段宽度不是很大。如本系统中的课程代码录入、学生成绩录入、学生素质分录入等。

下面以课程代码录入为例,说明在表单上用 Grid **控件录入数据的特点。表单运行效果如图** 15-12 **所示。**

图 15-12 **课程代码表录入界面**

本表单由一个表格(Grid1,**通过生成器生成**),**两个命令按钮**(Command1,Command2)**组成,其中各控件对象主要属性设置如下**(**其他属性采用默认值**):

```
This Form.Caption = "课程代码录入"
This Form.MaxButton = .F.
This Form.MinButton = .F.
This Form.Grid1.AllowAddNew = .T.
```

```
This Form.Grid1.DeleteMark = .T.
This Form.Grid1.RecordSource = "kcdmb"
This Form.Command2.Caption = "返回"
```

表单的 Unload **事件代码**：

```
dele for empty(课程代码)
pack
use
```

表单的 Load **事件代码**：

```
use kcdmb
```

表格的 AfterRowColChange **事件代码**：

```
LPARAMETERS nColIndex
thisform.command1.caption=iif(deleted(),"恢复","删除")
```

命令按钮 1 **的** Click **事件代码**：

```
if dele()
    recall
else
    dele
endif
thisform.command1.caption=iif(deleted(),"恢复","删除")
```

命令按钮 2 **的** Click **事件代码**：

```
release thisform
```

在完成学生成绩录入和学生素质分录入时，还会涉及到计算工作，计算一般可在数据录入完成并退出表单时完成，所以可将计算代码放入“返回”按钮的 Click **事件中**。

学生成绩录入后需要计算成绩合计和智育排名，代码为：

```
usedbf("kcdmb")
usedbf("xscjb")
n=fcount()-3                    && 获得本学期所选课程数
dime kc(n,3)                    && 分别用于存放课程代码及学时数
x=""                            && 用于生成计算成绩合计表达式
y=0
for i=2 to fcount()-2
    * 本循环用于从学生成绩表中取出课程代码，
    * 再到课程代码表中对应取出其学时数和课程名称
    kc(i-1,1)=fields(i)
    * 第一个字段是学号，从第二个字段开始才是课程代码
    sele kcdmb
    locate for 课程代码=kc(i-1,1) && 根据课程代码定位代码表中的指针位置
    kc(i-1,2)=课程名称
    kc(i-1,3)=课程学时/len(allt(开课学期))
```

```
        y=y+kc(i-1,3)
        * 多个学期都开的课程学时数为开课学期数的平均值
        * 如大学物理为第 1～2 学期开的课,共 180 学时,则每学期 90 学时
        sele xscjb
        x=x+fields(i)+"*"+allt(str(kc(i-1,3)))+"+"
    next i
    x=left(x,len(x)-1)
    * 最后变量 x 的值可能是
    * J8080102*72+J8080103*72+J4080101*90+Y3080101*75
    * +W5080101*90+J8080105*54+J8080108*54+Y3080102*40
    * 为什么要这样做呢? 因为课程是由程序开始选出来的,所以
    * 学生成绩表的字段名就有可能是变化的
    repl all 成绩合计 with (&x)/y      && 这里注意宏替换函数的应用
    * 以下为智育排名
    inde on -成绩合计 to xscjb_cj
    x=0
    p=0
    scan
        * 以下选择结构可实现成绩合计相同者名次相同
        if 成绩合计<>x
            x=成绩合计
            p=p+1
        endif
        repl 智育排名 with p
    endscan
    use
    sele kcdmb
    use
    return
```

学生素质分录入后需要计算素质总分,代码如下:

```
usedbf("xscjb")
inde on 学号 to xscjb_xh
usedbf("xszhjfb")
set order to xh
set rela to 学号 into xscjb
repl all 素质总分 with 精神文明*0.5+社会活动*0.2+体育锻炼*0.3,;
         综合积分 with xscjb.成绩合计*0.8+素质总分*0.2
* 以下为素质排名
inde on -素质总分 to zhjf_sz
```

```
x=0
p=0
scan
    if 素质总分<>x
        x=素质总分
        p=p+1
    endif
    repl 素质排名 with p
endscan
*以下为综合排名
inde on -综合积分 to zhjf_zh
x=0
p=0
scan
    if 综合积分<>x
        x=综合积分
        p=p+1
    endif
    repl 综合排名 with p
endscan
use
sele xscjb
use
return
```

2. 用文本框等控件录入

用这种方式一屏只能录入一条记录，录入的数据表各字段长度差异较大，字段类型较多。如学生基本情况表就可以采用这种方法。

与各字段有关的标签、文本框、复选框、编辑框等通过数据环境得到。其他控件属性如下：

```
This Form.BorderStyle = 2
This Form.Caption = "学生情况录入"
    This Form.MaxButton = .F.
    This Form.MinButton = .F.
    This Form.Command1.Caption = "第一个"
    This Form.Command2.Caption = "上一个"
    This Form.Command3.Caption = "下一个"
    This Form.Command4.Caption = "末一个"
    This Form.Command5.Caption = "添 加"
    This Form.Command6.Caption = "删 除"
    This Form.Command7.Caption = "退 出"
```

图 15-13 用文本框等控件录入数据

```
This Form.Image1.Stretch = 1
This Form.Image1.BorderStyle = 0
This Form.Label1.AutoSize = .T.
This Form.Label1.Caption = "当前记录/总记录数"
This Form.Label2.AutoSize = .T.
This Form.Caption = (allt(str(recn()))+"/"+allt(str(reccou())))
```

有关事件及代码如下：

```
(1) Form1.Refresh
        fn=".\pic\"+allt(thisform.txt照片.value)
        thisform.image1.picture=fn
        thisform.label2.caption=allt(str(recn()))+"/"+allt(str(reccou()))
        thisform.command1.enabled=recno()<>1
        thisform.command2.enabled=recno()<>1
        thisform.command3.enabled=recno()<>reccount()
        thisform.command4.enabled=recno()<>reccount()
(2)Form1.Load
        use xsqkb
(3)command1.Click
        go top
        thisform.refresh
(4) command2.Click
        skip -1
        if bof()
            go top
        endif
        thisform.refresh
(5) command3.Click
```

```
        skip
        if eof()
            go bottom
        endif
        thisform.refresh
(6)     command4.Click
        go bottom
        thisform.refresh
(7)     command5.Click
        append blank
        thisform.refresh
        thisform.txt 学号.setfocus
(8) txt 照片.Valid
        fn="".\pic\"+allt(this.value)
        if not file(fn) and not empty(this.value)
            messagebox("指定照片文件不存在!",0+64,"提示")
            this.value=""
        endif
        thisform.image1.picture=fn
        return .t.
(9) command6.Click
        dele
        pack
        go 1
        thisform.refresh
(10) command7.Click
        release thisform
```

15.5.3　查询修改

查询修改在 VFP 中可以通过过滤器(set filter to)命令实现，也可以通过 Browse 命令加条件实现，还可以通过视图、SQL 语句等方法。由于本系统涉及到的数据量较少，以下以学生情况查询为例说明查询中最复杂的组合条件查询方法。

组合条件查询是指根据用户需要设计各种条件的组合查询，如在学生情况表中有学号、姓名、性别、出生日期等字段，用户可能查姓名去查找(如查询姓名＝“李会琴”)，也可能多种条件组合模糊查询(姓“李”的、家在“杨凌区”的、性别为“女”的等等)。为了完成这些要求，可设计表单界面如图 15－14 所示。

图 15－14　组合条件查询表单

在图 15-14 的表单中,包含以下控件:

(1) 三个复选框,从上到下分别为 check1,check2,check3。

(2) 两个文本框,分别为 text1,text2。

(3) 一个单选按钮组 optiongroup1,其中包含 option*n*1 和 option*n*2 两个选项。

(4) 一个命令按钮 command1。

1. 各控件属性设置

```
ThisForm.BorderStyle = 2
ThisForm.Caption = "学生情况查询"
ThisForm.MaxButton = .F.
ThisForm.MinButton = .F.
ThisForm.Check1.AutoSize = .T. , ;
ThisForm.Check1.Caption = "按姓名查询", ;
ThisForm.Check1.Value = .F. , ;
ThisForm.Check2.AutoSize = .T. , ;
ThisForm.Check2.Caption = "按性别查询", ;
ThisForm.Check2.Value = .F. , ;
ThisForm.Check3.AutoSize = .T. , ;
ThisForm.Check3.Caption = "按住址查询", ;
ThisForm.Check3.Value = .F. , ;
ThisForm.Optiongroup1.AutoSize = .T. , ;
ThisForm.Optiongroup1.ButtonCount = 2, ;
ThisForm.Optiongroup1.BorderStyle = 0, ;
ThisForm.Optiongroup1.Value = 1, ;
ThisForm.Optiongroup1.Option1.Caption = "男", ;
ThisForm.Optiongroup1.Option1.Enabled = .F. , ;
ThisForm.Optiongroup1.Option2.Caption = "女", ;
ThisForm.Optiongroup1.Option2.Enabled = .F. , ;
```

2. 相关事件

(1) Form1 的 Unload 事件

```
use
```

(2) Form1 的 Load 事件

```
private xm,xb,zz,ti
use xsqkb
```

(3) Check1 的 InteractiveChange 事件

```
this.parent.text1.enabled=this.value
```

(4) check2 的 InteractiveChange 事件

```
this.parent.optiongroup1.option1.enabled=this.value
this.parent.optiongroup1.option2.enabled=this.value
```

(5) Check3 的 InteractiveChange 事件

```
this.parent.text2.enabled=this.value
```

(6) Text1 的 LostFocus 事件

```
if empty(this.value)
    this.parent.check1.value=.f.
    this.enabled=.f.
endif
```

(7) Text2 的 LostFocus 事件

```
if empty(this.value)
    this.parent.check1.value=.f.
    this.enabled=.f.
endif
```

(8) Command1 的 Click 事件

```
ti=".t."
if this.parent.check1.value=.t.
    xm=allt(this.parent.text1.value)
    ti=ti+".and.xm$姓名"
endif
if this.parent.check2.value=.t.
    xb=iif(this.parent.optiongroup1.value=1,"男","女")
    ti=ti+".and.xb=性别"
endif
if this.parent.check3.value=.t.
    zz=allt(this.parent.text2.value)
    ti=ti+".and.zz$家庭住址"
endif
set filt to &ti
brow
```

表单运行界面如图 15－15 所示，若查询在“杨凌区”姓“李”的学生，结果如图 15－16 所示。

图 15－15　表单运行效果

图 15－16　查询结果显示

15.5.4 报表输出

报表要根据用户需求来设计，对于学籍卡输出，应该具有查询功能，以方便用户有选择地输出任意一个学生的学籍卡。对于学生综合积分表，由于是简单的二维表格，这里就不再赘述。

15.5.5 系统信息

系统信息主要用于说明应用系统的版权信息，该表单的制作非常简单，由若干个标签组成，有时为了美观，还可以为表单设置一个漂亮的背景图片等，读者可参考 Windows 下各种软件的“关于”模块界面。

参考文献

[1] 教育部高等学校计算机科学与技术教学指导委员会. 关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求(试行)[S]. 北京:高等教育出版社,2006

[2] 萨师煊,王珊. 数据库系统概论(3 版)[M]. 北京:高等教育出版社,2000

[3] 杨长兴,周春艳,彭卫国. 数据库应用基础[M]. 长沙:中南大学出版社,2001

[4] 郭盈发,张红娟. 数据库原理(2 版)[M]. 西安:西安电子科技大学出版社,2002

[5] 赵忠孝. 数据库原理及 Visual FoxPro 应用[M]. 北京:高等教育出版社,2004

[6] 邓洪涛. 数据库系统及应用(Visual FoxPro)[M]. 北京:清华大学出版社,2004

[7] 范荣,屈昕,赵云海. Visual FoxPro 8.0 数据库开发教程[M]. 北京:清华大学出版社,2004

[8] 高伟,陈林等. Visual FoxPro 9.0 基础教程[M]. 北京:清华大学出版社,2005

[9] 谢膺白,高升宇,于晰. Visual FoxPro 6.0 程序设计教程[M]. 北京:人民邮电出版社,2002

[10] 谢膺白,陈勇,白学清. Visual FoxPro 及其程序设计教程[M]. 西安:西安交通大学出版社,2003

[11] 王利,崔巍等. 全国计算机等级考试耳机教程——Visual FoxPro 程序设计[M]. 北京:高等教育出版社,2001

[12] 郑阿奇,刘启芬,顾韵华. SQL Server 实用教程[M]. 北京:电子工业出版社,2002

[13] [美]Peter Rob,Carlos Coronel 著. 陈立军等,译. 数据库系统设计、实现与管理(5 版)[M]. 北京:电子工业出版社,2004