

Visual FoxPro 6.0

应用与开发教程

主 编 王 勇

副主编 丁亚明 崔跃林

编 者 (按编写章节为序)

崔跃林 周昌权 李如平

丁亚明 王雪峰 潘洁珠

秦晓彬 桑冬青 王 勇

安徽大学出版社

图书在版编目(C I P)数据

Visual FoxPro 6.0 应用与开发教程/王勇主编. —合肥:
安徽大学出版社,2007. 8
ISBN 978-7-81110-336-6

I. V... II. 王... III. 关系数据库—数据库管理系统,
Visual FoxPro 6.0—程序设计—教材 IV. TP311. 138

中国版本图书馆 CIP 数据核字(2007)第 126193 号

Visual FoxPro 6.0 应用与开发教程		王 勇 主编	
出版发行	安徽大学出版社 (合肥市肥西路 3 号 邮编 230039)	印 开	刷 本 787×1092 1/16
联系电话	编辑部 0551-5108498 发行部 0551-5108397	印 字	张 数 24.25 560 千
责任编辑	李 梅	版 次	2007 年 8 月第 1 版
封面设计	张 鑫	印 次	2007 年 8 月第 1 次印刷
经 销	新华书店		
ISBN 978-7-81110-336-6		定价 31.50 元	

如有影响阅读的印装质量问题,请与出版社发行部联系调换

面向 21 世纪高等院校课程教材

编写委员会

主 任 王 勇

副主任 崔景茂 郭玉堂

委 员 (按姓氏笔画为序)

丁亚明 肖 军 张兴元

周秋平 崔跃林

前言

Visual FoxPro 6.0 是一个面向对象的数据库管理系统,它具有强大的功能和丰富的工具。本书针对学生的特点,突出素质教育,以培养学生的动手能力为主要目的,力图通过大量的实例和练习,深入浅出、较全面系统地引导学生掌握 Visual FoxPro 6.0 的基本操作,并利用 Visual FoxPro 6.0 进行简单的数据库应用程序的开发。

全书共分十章,第一章绪论介绍了数据库的基本概念、数据库管理系统与数据库应用系统、Visual FoxPro 的界面组成和工作方式、项目管理和表达式,让读者对 Visual FoxPro 6.0 有一个初步的了解;第二章表的基本操作介绍了表的建立与修改、表的维护等有关数据表的基本操作,这是学习 Visual FoxPro 6.0 非常重要的入门基础;第三章查询与统计介绍了排序与索引、查询、数据工作期、统计、Select-SQL 查询、数据库与视图等内容,这些内容是本书的重点内容,建立数据库就是要实现对已有数据进行快速便捷的查询;第四章结构化程序设计初步介绍了程序设计的基本概念、程序文件的建立、修改与运行方式、基本输入输出命令、程序的控制结构等内容,以结构化程序设计思想方法来介绍程序设计的过程和调试方法,本章内容是后续面向对象程序设计章节的基础;第五章表单设计基础介绍了表单设计器,并引入了面向对象程序设计思想,为以后各章做好了铺垫;第六章表单控件设计和第七章表单高级设计分别介绍了在表单中涉及到的常用控件的设计方法、多表单应用程序、用户定义属性与方法程序和类的创建与使用方法,让读者以面向对象程序设计方法轻松地设计出数据浏览和查询表单界面;第八章菜单设计介绍了下拉式菜单和弹出式菜单的设计方法,让读者设计出与 Windows 窗口界面类似的菜单系统,以方便用户的操作;第九章报表与标签设计介绍了报表和标签的设计和输出方法,实现数据的打印输出;第十章开发应用程序介绍了系统开发基本步骤和生成应用程序的方法,让读者在掌握前面知识内容的基础上将自己的

设计成果转换成应用程序,并产生一种成就感。

本书由王勇同志担任主编,丁亚明、崔跃林同志担任副主编,第一章由崔跃林同志编写、第二章由周昌权同志编写、第三章由李如平同志编写、第四章由丁亚明同志编写、第五章由王雪峰同志编写、第六章由潘洁珠同志编写、第七章由秦晓彬同志编写、第八章由桑冬青同志编写、第九章和第十章由王勇同志编写;统稿工作由主编王勇同志完成。

由于时间仓促、水平有限,书中难免出现疏漏或错误,恳请广大读者、专家批评指正,也欢迎与作者联系,共同探讨。

编者

2007 年 6 月于合肥

目 录

第1章 绪论	1
1.1 数据库的基本概念	1
1.1.1 信息、数据和数据处理	1
1.1.2 数据模型	2
1.1.3 数据库	4
1.1.4 关系型数据库	5
1.2 数据库管理系统与数据库应用系统	7
1.2.1 数据库管理系统	7
1.2.2 数据库应用系统	8
1.2.3 数据库系统	8
1.3 Visual FoxPro 6.0 的界面组成	8
1.3.1 标题栏	9
1.3.2 菜单栏	9
1.3.3 工具栏	9
1.3.4 主窗口	10
1.3.5 命令窗口	10
1.3.6 状态栏	11
1.4 Visual FoxPro 6.0 的工作方式	11
1.4.1 交互操作方式	11
1.4.2 程序运行方式	11
1.5 项目管理器	12
1.5.1 项目文件的创建	12
1.5.2 项目管理器简介	14
1.5.3 项目管理器的窗口操作	15
1.6 表达式	16
1.6.1 常量	16
1.6.2 变量	18
1.6.3 表达式	21

1.6.4 函数	24
1.6.5 VFP 中命令的常用子句	31
习题	33
第2章 表的基本操作	35
2.1 表的建立与修改	35
2.1.1 表结构的概念	35
2.1.2 建立表的结构	37
2.1.3 修改表的结构	41
2.1.4 向表中输入数据	45
2.1.5 表中数据的修改	46
2.2 表的维护	48
2.2.1 表结构与表的复制	48
2.2.2 记录指针的移动	50
2.2.3 插入与追加记录	53
2.2.4 删除与恢复记录	56
2.2.5 数据的替换	57
2.2.6 逻辑表的设置	61
习题	62
第3章 查询与统计	67
3.1 排序与索引	67
3.1.1 排序	67
3.1.2 索引	69
3.2 数据的查询	77
3.2.1 顺序查询	77
3.2.2 索引查询	79
3.3 数据工作期	81
3.3.1 多工作区查询	81
3.3.2 数据工作期	84
3.3.3 视图文件	85
3.3.4 表的关联	87
3.4 数据的统计	94
3.4.1 求记录个数的命令	94
3.4.2 求和命令	94
3.4.3 求平均值命令	95
3.4.4 计算命令	95
3.4.5 汇总命令	95

3.5	Select-SQL 查询	96
3.5.1	Select-SQL 命令查询	96
3.5.2	利用查询设计器查询	102
3.5.3	查询结果的图形处理	110
3.6	数据库与视图	113
3.6.1	数据库	113
3.6.2	数据词典	118
3.6.3	视图	131
	习题	141
第4章	结构化程序设计初步	143
4.1	程序设计概述	143
4.1.1	程序	143
4.1.2	结构化和可视化程序设计	144
4.2	程序文件的建立、修改与运行	144
4.2.1	程序文件的建立	144
4.2.2	程序文件的修改	146
4.2.3	程序文件的运行	147
4.3	基本输入输出	148
4.3.1	基本输入	148
4.3.2	基本输出	153
4.4	程序的控制结构	154
4.4.1	顺序结构	154
4.4.2	分支结构	155
4.4.3	循环结构	159
4.5	结构化程序设计	172
4.5.1	子程序	173
4.5.2	过程	177
4.5.3	自定义函数	181
4.5.4	参数传递	182
4.5.5	变量的作用域	186
4.5.6	程序调试	189
	习题	198
第5章	表单设计基础	207
5.1	表单设计器	207
5.1.1	表单向导	207
5.1.2	表单设计器的基本操作	213

5.2 面向对象的程序设计方法	227
5.2.1 基本概念	227
5.2.2 对象引用	230
习题	232
第6章 表单控件设计	233
6.1 输出、输入类控件	233
6.1.1 标签、图像、文本框、编辑框	233
6.1.2 列表框、组合框、微调控件	245
6.2 控制类控件	254
6.2.1 命令按钮与命令按钮组	254
6.2.2 复选框与选项按钮组	264
6.2.3 计时器	268
6.3 容器类控件	269
6.3.1 表格	270
6.3.2 页框	273
6.3.3 容器	275
6.4 连接类控件	275
6.4.1 ActiveX 控件	276
6.4.2 ActiveX 绑定控件	278
6.4.3 超级链接	279
6.4.4 设计实例	279
习题	283
第7章 表单高级设计	288
7.1 多表单应用程序	288
7.1.1 应用程序界面	288
7.1.2 表单集	293
7.2 用户定义属性与方法程序	297
7.2.1 用户定义属性	297
7.2.2 用户定义方法程序	304
7.3 类	305
7.3.1 编辑用户定义的子类	305
7.3.2 子类的应用	305
7.3.3 用户定义的子类注册	306
7.3.4 将表单保存为类	307
7.3.5 为字段设置类	309
7.3.6 用户定义工具栏	310

习题	312
第8章 菜单设计	313
8.1 下拉式菜单的设计	313
8.1.1 菜单生成的基本步骤	313
8.1.2 快速菜单生成	315
8.1.3 菜单设计器窗口	316
8.1.4 “显示”菜单的命令	319
8.2 弹出式菜单设计	324
8.2.1 利用菜单设计器设计快捷菜单	325
8.2.2 用菜单命令设计弹出式菜单	326
习题	332
第9章 报表与标签设计	335
9.1 报表设计与应用	335
9.1.1 报表向导	335
9.1.2 打开报表设计器	339
9.1.3 报表设计器窗口介绍	340
9.1.4 快速报表	341
9.1.5 修改用快速报表产生的报表	343
9.1.6 设计报表	343
9.1.7 报表输出	349
9.2 标签的设计与使用	354
9.2.1 标签向导	354
9.2.2 标签设计器	356
9.2.3 标签的创建与输出命令	357
习题	357
第10章 开发应用程序	359
10.1 应用项目综合实践	359
10.1.1 系统开发基本步骤	359
10.1.2 连编应用程序	362
10.1.3 主程序设计	365
10.2 应用程序生成器	366
10.2.1 使用应用程序向导	367
10.2.2 应用程序生成器	368
10.2.3 使用应用程序生成器	371
习题	376



第 1 章

绪 论

从计算机产生以来,人类就开始使用机器来存储和管理数据,随着计算机信息处理技术的日益发展,计算机管理数据的方式也在不断改进,并于 20 世纪 50 年代末产生了文件管理系统,该系统将数据组织在一个个独立的数据文件中,实现了“按文件名来访问,按记录进行存取”的管理技术。目前,文件管理仍是一般高级语言普遍采用的数据管理方式,但在需要处理的数据量较大的系统中,数据间不免会存在这样或那样的联系,而文件管理系统所采用的则是在数据文件之间缺乏联系的结构,以及一次最多存取一个记录的访问方式,以这种数据存储结构与访问方式来处理数据,已不能适应较大信息量处理的需要。于是数据库管理系统便应运而生,到了 20 世纪 60 年代末期,在美国诞生了第一个商品化的数据库系统——IMS 系统(Information Management System)。

计算机数据管理方式从文件管理系统发展到数据库管理系统,标志着计算机数据管理技术的一次质的飞跃。Visual FoxPro 6.0 是 Microsoft 公司近年推出的关系数据库管理系统,其特点是方便易学、有效灵活、功能强大,特别适用于中小型数据库的管理和开发,成为开发、设计各种管理信息系统的流行工具。通过本书的学习能够帮助学习者迅速地掌握数据库管理的相关应用技术。

1.1 数据库的基本概念

1.1.1 信息、数据和数据处理

1.1.1.1 信息

信息是指通过各种方式传播的、可被感受的声音、文字、图像、符号等所表现的某一特定事物的消息、情报或知识,它是对客观事物的反映,并为某一特定目的提供决策依据。

1.1.1.2 数据

数据通常指存储在某一媒体上,并用符号记录下来的可加以鉴别的符号集合。

数据的概念包括两个方面:其一,数据内容是事物特性的反映或描述;其二,数据是存储在某一媒体上符号的集合。

数据的概念在数据处理领域中的内涵比在科学计算领域中的内涵要广泛地多。“符号”不仅指数字、字母、文字和其他特殊字符,而且还包括图形、图像、声音等多媒体数据;

所谓“记录下来”也不仅是指印在纸上,而且包括记录在磁介质、光介质半导体存储器等存储介质中。数据在空间上的传递称为通信,在时间上的传递称为存储。

1.1.1.3 数据处理

数据处理是指将数据转换成信息的过程。广义地讲,它包括对数据的收集、存储、加工、分类、检索、传播等一系列活动。狭义地讲,它是指对所输入的数据进行加工整理。其基本目的是从大量的、已知的数据出发,根据事物之间的固有联系和规律,通过分析归纳、演绎推导等手段,萃取出对人们有价值、有意义的信息,作为人们决策的依据。由此可见,信息是一种被加工成特定形式的数据,这种数据形式对于数据接收者来说是有意义的。对数据的加工可以相对比较简单也可以相当复杂。简单加工包括组织、编码、分类、排序等;也可以通过使用统计学方法、数学模型等对数据进行深层次的复杂加工。

在数据处理中,数据管理技术是重要的组成部分,其发展大致经历了三个阶段:

1. 人工管理:这种数据管理的特点是数据与程序不具有独立性,一组数据对应一组应用程序。程序运行结束后退出计算机系统,一个程序中的数据不能被其他程序利用。程序与程序间存在着大量的数据冗余。

2. 文件管理:将有关数据组织成数据文件,这种数据文件脱离程序而具有相对的独立性,并允许对数据文件命名,而应用程序是通过数据文件名来存取数据文件中的数据。

文件管理方式具有以下弱点:一是尽管数据以数据文件方式独立存放,但程序与数据紧密相关,一旦数据文件离开了使用它的应用程序,便失去了存在的价值;二是由于不同的应用程序需要各自建立相应的数据文件,从而造成了数据冗余,而占用大量的存储空间;三是由于同一数据存放在不同的数据文件中,很容易造成数据的不一致性;四是文件管理方式不能反映信息之间的相互联系。

3. 数据库系统:这种方式与文件管理方式不同,其数据组织是面向整个系统,即用整体观点规划数据,构成一个数据的仓库,库中的数据能满足所有用户的不同要求,供不同用户共享。这时,应用程序不再与一个孤立的数据文件相对应,而是将整体数据集合中的某个子集作为逻辑文件与应用程序对应,通过一个系统管理软件——数据库管理系统 DBMS(DataBase Management System)实现逻辑文件与物理数据之间的映射。

1.1.2 数据模型

数据模型是数据库管理系统用来反映实体及其实体间联系的数据组织的结构形式。实体是指客观存在的并可互相区别的任何事物。比如:人、工厂、设备、一个规划等。在现实世界中,实体之间的联系可分为三种类型:“一对一”的联系:如现实生活中的夫妻关系;“一对多”的联系:如老师和学生的关系;“多对多”的联系:如商店和顾客之间的关系。

数据模型可分为三种基本的类型:

1. 层次模型:层次模型的基本结构是树型结构,也叫树状模型。它是以实体(记录型)为结点构成的树,结点间树枝表示实体间的某种关系。

层次模型的特点:

- (1)有且仅有一个根结点无双亲(上级结点);
- (2)其他结点有且仅有一个双亲(上级结点)。

层次模型具有层次分明、结构清晰的优点,它适用于描述客观存在的事物中有主、次之分的结构关系。缺点是只能反映实体间的“一对多”的联系。

具有这种层次模型的结构有国家的行政单位间的行政隶属关系、单位内部各部门间的隶属关系、军队建制等,如图 1.1 表示了一所学院的行政组织间的层次关系。

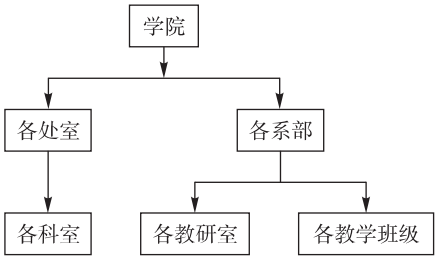


图 1.1 层次关系示意图

2. 网状模型:网状模型是以实体(记录型)为结点的网络,它反映现实世界中较为复杂的事物间的联系。

网状模型的特点:

- (1)可以有一个以上的结点无双亲;
- (2)至少有一个结点有两个以上的双亲。

网状模型表达能力强,它能反映实体间的“多对多”的联系,它既能表达实体间的纵向联系,又能表达实体间的横向联系。但其在概念上、结构上和使用上都比较复杂,而且对计算机的硬件环境要求高。

具有这种网状模型的结构有一个学校的教师、班级、课程间的关系,产品与生产单位间的关系等。如图 1.2 表示了一个简单的教师、班级、课程间的网状关系。

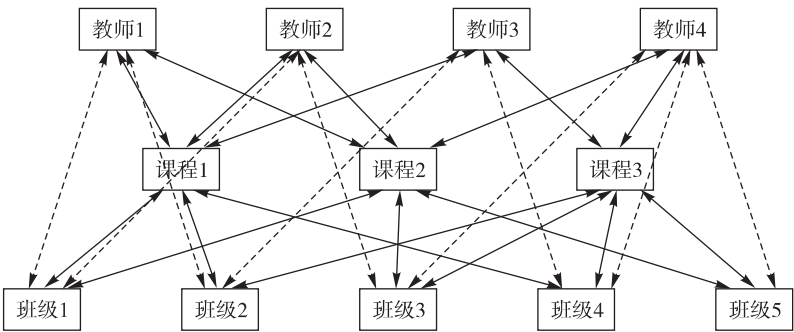


图 1.2 网状关系示意图

3. 关系模型:关系模型是一张二维表格,它是用表格来描述实体间的联系。表格中的栏目称为关系框架,也称关系模式。表中垂直方向的列称为表的属性,每一列的列标题称为表的属性名或字段名,标题栏下方输入的内容为表的数据,每一行数据称为表的一个记录。如表 1.1 所示。

关系模型既能反映属性间一对一的联系,也能反映属性间一对多的联系,还能反映属性间多对多的联系,它具有以下性质:

- (1)同一张表中不允许有重复的字段名;

(2)同一列的数据类型、宽度必须相同;数据的长度不足自动以空格代替,数据的长度过宽自动截去超出部分;

表 1.1 职员表

Employee											
身份号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
20050001	张丽平	女	30	汉	07/12/95	01	T	1356.25	14858359778	Memo	Gen
20050002	陈国庆	男	40	汉	03/22/85	11	T	1850.60	14978589657	memo	Gen
20050003	方世玉	男	25	汉	08/05/00	12	F	1042.38	14654896589	memo	gen
20050004	王国其	女	22	汉	09/23/02	11	F	988.57	14898589875	memo	gen
20050005	关鹏	男	27	满	07/12/96	13	T	1125.12	14588796225	memo	gen
20050006	孙宏伟	男	50	回	08/06/75	12	T	2056.87	14658582328	memo	gen
20050007	李莉	女	40	汉	05/12/88	13	T	1788.95	14583258772	memo	gen
20050008	杨剑雄	男	21	汉	08/23/03	01	F	985.12	14587685685	memo	gen

- (3)表中不允许有重复的行,各行相异;
- (4)行的次序和列的次序可任意排列;
- (5)关系中的任何一个属性都必须是不可再分的元素。

1.1.3 数据库

数据库是存储在计算机存储设备上的、结构化的相关数据集合。它不仅包括描述事物的数据本身(如表),而且还包括相关事物之间的联系(如表与表之间的联系等)。

这种数据集合具有如下特点:

- (1)面向多个应用,可以被多个用户、多个应用程序共享;
- (2)其数据结构独立于使用它的应用程序,对数据进行的增加、删除、修改和检索都由系统软件统一进行管理和控制。

数据库的基本结构分三个层次,反映了观察数据库的三种不同角度:

- (1)物理数据层:它是数据库的最内层,是物理存储设备上实际存储的数据集合。这些数据是原始数据,是用户加工的对象,由内部模式描述的指令操作处理的位串、字符和字组成;
- (2)概念数据层:它是数据库的中间一层,是数据库的整体逻辑表示。指出了每个数据的逻辑定义及数据间的逻辑联系,是存储记录的集合。它所涉及的是数据库所有对象的逻辑关系,而不是它们的物理情况,是数据库管理员概念下的数据库;
- (3)逻辑数据层:它是用户所看到和使用的数据库,表示了一个或一些特定用户使用的数据集合,即逻辑记录的集合。

数据库不同层次之间的联系是通过映射进行转换的。数据库具有以下主要特点:

- (1)实现数据共享。数据共享既可以让所有用户同时存取数据库中的数据,也可以让用户用各种方式通过接口使用数据库,并提供数据共享。数据访问的最小单位是字段;
- (2)减少数据的冗余度。同文件系统相比,由于数据库实现了数据共享,从而避免了用户各自建立自己的数据文件,在很大程度上减少了数据冗余,维护了数据的一致性;
- (3)数据的独立性。数据的独立性既包括数据库中数据库的逻辑结构与应用程序相互独立,也包括数据物理结构的变化不影响数据的逻辑结构;

简历,照片 For 性别="女" && 执行该命令前必须先在当前工作区中打开相关的表
还可以用以下命令实现同一结果:

Select 身份号,姓名,性别,年龄,民族,工作时间,部门,婚否,工资,电话,简历,照片;

From Employee Where 性别="女" && 执行该命令时相关表可以打开,也可以不打开
由于在本例中已经显示了所有字段,所以上述两条主要命令还可以用以下格式:

Browse For 性别="女"

Select * From Employee Where 性别="女"

注意:一条命令需要在多于两行书写时,除最后一行外,每行的最后必须用分号结束。

例 1.2 在表 1.1 中查询所有记录的字段名为身份号、姓名、工资、部门的信息,用投影运算查询的结果如表 1.3 所示。

表 1.3 投影运算查询结果表

身份号	姓名	工资	部门
20050001	张丽平	1356.25	01
20050002	陈国庆	1850.60	11
20050003	方世玉	1042.38	12
20050004	王国真	988.57	11
20050005	关鹏	1125.12	13
20050006	孙宏伟	2056.87	12
20050007	李莉	1768.95	13
20050008	杨剑雄	985.12	01

表 1.4 部门代码表

部门代码	部门名称
01	办公室
11	一车间
12	二车间
13	三车间

可实现本例结果的命令举例:

Use Employee

Browse Fields 身份号,姓名,工资,部门

还可以用以下命令实现同一结果:

Select 身份号,姓名,工资,部门 From Employee

例 1.3 表 1.4 为部门代码表,要求通过表 1.1 和表 1.4 查询其中所有职员所对应的部门名称等所有信息,用连接运算查询的结果如表 1.5 所示。

表 1.5 连接运算查询结果

身份号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片	部门名称
20050001	张丽平	女	30	汉	07/12/95	01	T	1356.25	14856359778	Memo	Gen	办公室
20050002	陈国庆	男	40	汉	03/22/85	11	T	1850.60	14978589657	memo	Gen	一车间
20050003	方世玉	男	25	汉	08/05/00	12	F	1042.38	14854896589	memo	gen	二车间
20050004	王国真	女	22	汉	09/23/02	11	F	988.57	14896589875	memo	gen	一车间
20050005	关鹏	男	27	满	07/12/96	13	T	1125.12	14568796225	memo	gen	三车间
20050006	孙宏伟	男	50	回	08/06/75	12	T	2056.87	14656562328	memo	gen	二车间
20050007	李莉	女	40	汉	05/12/86	13	T	1768.95	14563256772	memo	gen	三车间
20050008	杨剑雄	男	21	汉	08/23/03	01	F	985.12	14567665685	memo	gen	办公室

可实现本例结果的命令举例:

Select * From Employee,Department;

Where Employee.部门=Department.部门代码

上述命令用 * 号表示要求输出连接的两个表中的所有字段,如表 1.6 所示,若要对

可运行的目标程序。

3. 数据操作功能:提供“数据操作语言(Data Manipulation Language)”,对数据库中的数据进行查询、更新(修改、插入和删除等等)。如 SQL(Structured Query Language)语言。

4. 数据的组织和存取:提供数据在外围存储设备上的物理组织和存取的方法。

5. 事务的运行管理:提供事务的运行管理及运行日志,事务运行的安全性监控和数据完整性检查,事务的并发控制及系统恢复等功能。

6. 数据库的维护:为数据库管理员提供软件支持,包括数据安全控制、完整性保障、数据库备份、数据库重组以及性能监控等维护工具。

1.2.2 数据库应用系统

数据库应用系统 DBAS(DataBase Application System)是指系统开发人员利用数据库系统资源开发出来的,面向某一类实际应用的应用软件系统。一个数据库应用系统(DBAS)通常由数据库和应用程序两部分组成。如以数据库为基础的财务管理系统、人事管理系统、图书管理系统、生产管理系统、户籍管理系统、计算机售票系统等均为数据库应用系统。

1.2.3 数据库系统

数据库系统 DBS(DataBase System)是引进了数据库技术后的整个计算机系统,实现了有组织、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手段。它主要由以下几个部分组成:硬件系统及计算机操作系统、数据库管理系统及数据库应用系统、数据库集合、数据库管理员和用户等。在数据库系统中各组成部分间的相互关系如图 1.3 所示。

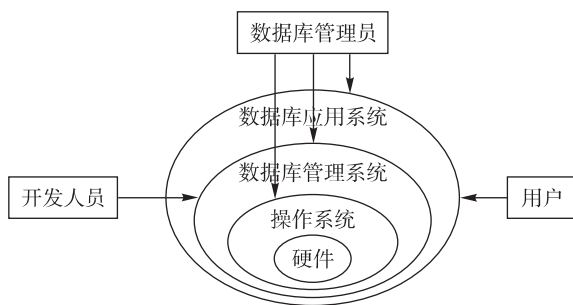


图 1.3 数据库系统组成层次关系示意图

1.3 Visual FoxPro 6.0 的界面组成

Visual FoxPro 6.0 的界面如图 1.4 所示,其组成与其他 Windows 应用程序的窗口类似,所不同的是在主窗口(工作区)中有一命令窗口。Visual FoxPro 6.0 窗口中包括:标题栏、菜单栏、工具栏、主窗口、命令窗口和状态栏 6 个部分。

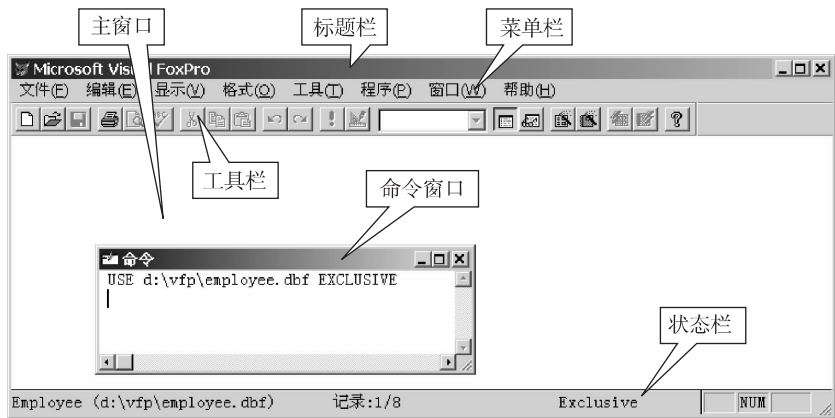


图 1.4 Visual FoxPro 6.0 的界面组成

1.3.1 标题栏

标题栏位于 Visual FoxPro 6.0 窗口的顶部，由一个狐狸图标、Microsoft Visual FoxPro、控制按钮等几部分组成。

1.3.2 菜单栏

菜单栏位于标题栏的下方，通常菜单栏包括：文件、编辑、显示、格式、工具、程序、窗口和帮助八个菜单项目。但菜单栏会随着用户的某些操作而有所变化。例如：刚启动 Visual FoxPro 6.0 时，菜单栏如图 1.5 所示，但当表的浏览窗口打开时，其菜单栏如图 1.6 所示，出现了菜单“表”。



图 1.5 初始启动时的菜单



图 1.6 表的浏览窗口打开时的菜单

1.3.3 工具栏

在菜单栏的下面是“常用”工具栏。Visual FoxPro 6.0 共提供了 11 种不同的工具栏，用户可通过单击菜单“显示/工具栏”，出现如图 1.7 所示的工具栏选择窗口，可在该窗口中选择或定制工具栏。每种工具栏都是由一些按钮组成，工具栏上的按钮对应着相应的任务，只有在执行特定的任务时相应的按钮才起作用，否则按钮呈灰色，表示不可用。如图 1.8 为打开的表单设计器、查询设计器和表单控件三个工具栏。



图 1.7 工具栏

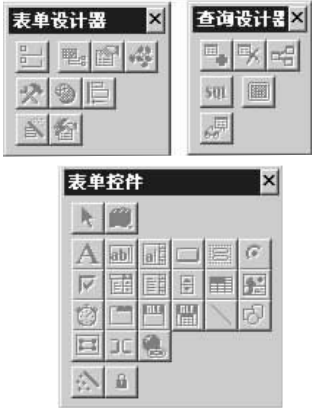


图 1.8 表单设计器、查询设计器、表单控件

1.3.4 主窗口

菜单栏和工具栏下面最大的窗口区域为主窗口，主要用于显示命令和程序执行的结果。例如可在命令窗口输入：“中华人民共和国”，并按回车键，则会在主窗口中显示出“中华人民共和国”，如图 1.9 所示。



图 1.9 主窗口

1.3.5 命令窗口

在 Visual FoxPro 6.0 启动成功后，主窗口中会出现一个标题为“命令”的小窗口，这就是命令窗口，如图 1.9 所示。其主要的功能是输入单个的命令，在用户按回车键后计算机将执行该光标所在行的命令。

命令窗口除了有可直接在窗口中输入并执行命令的功能外，还有两个辅助功能：

- 1. 当用户使用界面操作方式，完成某种操作后，系统会自动在命令窗口中会显示出相应的、完整的执行命令。
- 2. 执行过的命令会依次保留在命令窗口中，用户可利用上下光标的移动等方法，让光标停留在曾经使用的命令行上，用户可对该命令进行修改、剪贴或重新执行。

1.3.6 状态栏

状态栏位于 Visual FoxPro 6.0 窗口的最下方,主要用于显示 Visual FoxPro 6.0 的当前工作状态,包括打开的数据库名、表文件名和记录状态,以及按钮和菜单的功能说明等,如图 1.4 所示。

1.4 Visual FoxPro 6.0 的工作方式

Visual FoxPro 6.0 支持两种工作方式,即交互操作方式和程序运行方式。

1.4.1 交互操作方式

交互操作方式又分为两种:命令方式和界面操作方式。

命令方式是指用户在命令窗口中输入一条命令并按回车键,系统立即执行该命令并在主窗口中显示执行结果,如图 1.9 所示。采用命令方式时,要熟悉命令格式及其使用方法,通常适用于比较简单的操作。

用户在使用命令方式进行操作时,需要注意一些事项和技巧:

1. 键入命令语句后,在还没有按【Enter】键执行前,可以按【Esc】键来删除键入的命令语句。
2. 执行过的每一条命令都会保留在命令窗口中,因此需要再次输入同一命令时,只要在命令窗口中选择该命令行,再次按【Enter】键执行即可。
3. 在命令窗口中的命令文本可以随意编辑如修改、拖动、复制。

随着 Windows 的推广,基于 Windows 的可视化界面操作方式已成为主要的交互操作方式。界面操作方式是指通过对系统菜单和工具按钮的点击操作来实现交互操作的一种操作方式。Visual FoxPro 6.0 还进一步完善了图形界面操作,它提供的向导、设计器等辅助设计工具,其直接的可视化界面正被越来越多的用户所熟悉和使用。界面操作方式最突出的优点是操作简单、直观,其不足之处是步骤较为繁琐。

1.4.2 程序运行方式

交互操作方式虽然方便,但用户操作与机器执行互相交叉,却降低了执行的速度。因此,在 Visual FoxPro 6.0 系统中,为了解决实际问题需要,可以将按 Visual FoxPro 6.0 系统约定编写的命令序列存储为程序文件,用户需要时,通过特定的命令自动执行程序文件,让用户对程序的介入程度减小到最小限度,从而提高系统的执行效率。

例 1.4 若在默认目录下有一简单程序 abc.prg 其程序内容为:

Clear	&& 对主窗口清屏
Use Employee	&& 打开 Employee 表
List	&& 在主窗口中显示所有记录内容
Wait	&& 命令执行到此时暂停,按任意键继续
Browse For 性别="男"	&& 在浏览窗口中显示所有性别为“男”的记录内容



图 1.11 新建

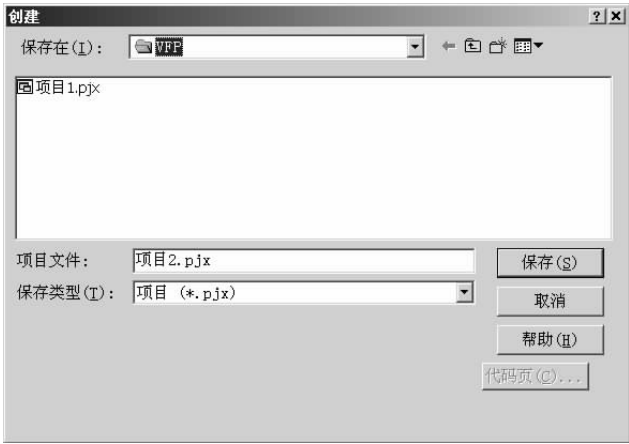


图 1.12 创建

- (3) 在“创建”对话框中输入项目文件的名字和选择项目文件要保存的位置,然后,单击“保存”按钮。若不作选择和输入,则系统会用“项目 1. pjx”文件名保存在默认路径下;
- (4) 建立的项目文件会自动在项目管理器中打开,如图 1.13 所示。

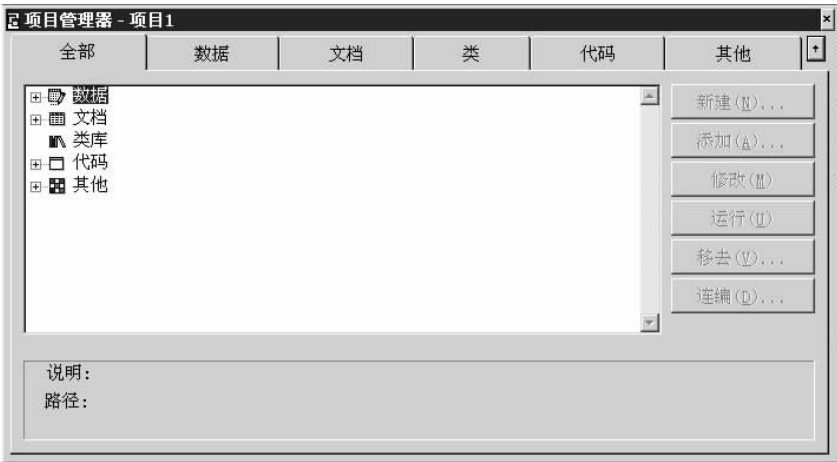


图 1.13 项目管理器

1.5.1.2 命令方式

格式一:

Create Project [<文件名>]

功能:创建一个项目文件。

例 1.6

Create Project d:\vfp\aaa &&. 在 d 盘 vfp 目录下创建一个 aaa 项目文件,并打开该文件

格式二:

Modify Project [<文件名>]

功能:当指定的项目文件不存在时,创建它。否则,打开指定的项目文件。

例 1.7

Modify Project d:\vfp\bbb && 若 d 盘 vfp 目录下不存在项目文件 bbb.pjx 则创建并打开该文件,若该项目文件存在则打开该文件。

1.5.2 项目管理器简介

项目管理器界面主要由选项卡和命令按钮组成。

1.5.2.1 选项卡

项目管理器窗口由 6 个选项卡构成,它们分别是“全部”、“数据”、“文档”、“类”、“代码”和“其他”。

“全部”选项卡:集中了其他 5 种选项卡的全部内容,用户几乎可以在此选项卡下完成所有的操作,它是另外 5 个选项卡的全部功能的集合。从图 1.13 所示的界面也可以看出,它的 5 个项目“数据”、“文档”、“类库”、“代码”和“其他”正好与另外 5 个选项卡一一对应。

“数据”选项卡:用于一个项目的所有数据管理。按大类划分它可以管理数据库、自由表和查询;在一个数据库中又可以管理数据库表、本地视图、远程视图、连接和存储过程。

“文档”选项卡:包含了处理数据时所用的三类文件,如输入和查看数据所用的表单及打印表与查询结果所用的报表和标签等。

“类”选项卡:包含了进行面向对象程序设计时用户所创建的类。

“代码”选项卡:列出了应用程序的主程序及 API 等相关代码。

“其他”选项卡:所列出的对象包括应用程序的菜单文件以及一些用于程序说明的文本文件等。

1.5.2.2 命令按钮

在项目管理器窗口中的右边有众多的命令按钮,这些按钮会随着所选选项卡中内容的不同而发生变化。以下介绍的所有命令按钮都可以在“项目”菜单中找到相应的命令。

“新建”按钮:创建一个由用户在项目管理器内某一选项卡中选定类型的新文件(如数据库、表、表单、报表、查询、程序等)并将其添加到项目文件中。当在项目管理器窗口的某一选项卡中选择一个文件类型(如数据库)时,该按钮被激活。单击该按钮时,用户可通过出现的对话框新建选定类型的文件。

“添加”按钮:在项目文件中加入一个已有的文件。当在项目管理器窗口选择一个文件类型(如数据库)时,该按钮被激活。单击该按钮用户可通过出现的打开对话框,选择要添加的文件。

“修改”按钮:当在项目管理器窗口中选中一个数据库、自由表、查询、表单、报表、标签、类库、程序、菜单或文本文件时,该按钮被激活。单击该按钮,用户可通过打开的设计器或编辑器来修改选定的文件。

“关闭”与“打开”按钮:用于关闭或打开一个数据库。只有在项目管理器窗口选中了一个数据库后,该按钮才会出现。如果选中的数据库已经打开,则这个按钮为“关闭”,否则为“打开”。

“浏览”按钮:用于打开选定表的浏览窗口,用户可在打开的浏览窗口中查看、修改表

中的数据。该按钮只有在选中表的时候才出现。

“预览”按钮:用于预览选定的报表或标签文件的打印情况。只有在项目管理器窗口中选中一个报表或标签文件后,该按钮才可用。

“运行”按钮:运行选定的查询、表单、菜单或程序文件。只有在项目管理器窗口中选中一个查询、表单、菜单或程序文件时,该按钮才能被激活。

“移去”按钮:用于从项目文件中移去或删除当前选定的文件。只有在项目管理器窗口中选中一个文件后该按钮才被激活。

“连编”按钮:用于将所有在项目文件中引用的文件(除了那些标记为排除的文件)合成为一个应用程序文件或重新连编一个已存在的项目文件。

1.5.3 项目管理器窗口操作

“项目管理器”通常显示为一个独立的窗口,但具有工具栏窗口的特性,可以对其进行移动、折叠和调整窗口大小等操作。项目管理器窗口关闭后,数据库、数据库表、数据库视图和自由表等均不自动关闭,此时可用 Close Database 和 Use 命令来关闭它们。

1.5.3.1 改变窗口的显示外观

1. 位置的移动:将鼠标指针指向标题栏,按住鼠标左键不放,即可将“项目管理器”窗口拖到屏幕上的其他位置。

2. 改变窗口大小:将鼠标指针指向窗口的顶端、底端、两边或角上,拖动鼠标就可扩大或缩小它的尺寸。

3. 窗口的折叠与展开:单击“项目管理器”选项卡的最右边的“箭头”按钮,就可折叠或展开“项目管理器”的窗口。窗口在折叠情况下只显示6个选项卡的选项栏,这时只要单击某选项按钮就能显示该选项页的内容,再单击一次该选项按钮则还原。单击右上角的“下箭头”,就可还原“项目管理器”的窗口。当“项目管理器”放置在常用工具栏上时,将鼠标指向“项目管理器”选项卡的空白处双击或拖离工具栏,则还原“项目管理器”窗口。

1.5.3.2 选项卡的拆分

折叠“项目管理器”窗口后,可以拖开选项卡,该选项卡成为浮动状态,可根据需要重新安排它们的位置。拖下某一选项卡后,它可以在 Visual FoxPro 6.0 的主窗口中独立移动。若要拖开某一选项卡,其操作步骤为:

- (1) 折叠“项目管理器”;
- (2) 选定一个选项卡,将它拖离“项目管理器”。

当选项卡处于浮动状态时,在选项卡中单击鼠标右键,在弹出的快捷菜单中增加了“项目”菜单中的选项。单击选项卡右上方的窗口“关闭”按钮,则还原到“项目管理器”窗口。

1.5.3.3 钉住浮动选项卡

当拖开多个选项卡后,选项卡之间会产生重叠。为使其中一个选项卡始终显示在最上面,可以使用该选项卡上端的“图钉”将其钉住。

1.5.3.4 项目内容的折叠和展开

“全部”选项卡中包括5个项目,而一个项目包含了许多的内容。“项目管理器”是采

用层次结构方式管理和组织这些内容的。在显示时,各部分用图标连接表示,各层次关系可以通过[+]按钮展开,通过[-]按钮折叠。

1.5.3.5 项目间文件共享

通过与其他项目共享文件,你可以调用在其他项目开发上的工作成果。此文件并未复制,项目只储存了对该文件的引用。文件可同时与不同的项目连接。若要在项目之间共享文件:

- (1) 在 Visual FoxPro 6.0 中,打开要共享文件的两个项目;
- (2) 在包含该文件的“项目管理器中”,选择该文件;
- (3) 拖动该文件到另一个的项目管理器中。

1.6 表达式

在 Visual FoxPro 6.0 中对于任何数据,我们都不可避免的要涉及到对其数据类型的判断问题,只有在正确了解和掌握了数据类型的基础上,才能对同一类型的数据进行计算。然而,有时我们必须对不同类型的数据进行计算,但不同的数据类型是不可能直接进行计算的,如果强行对不同类型的数据进行计算,则系统会提示“数据类型不匹配”的错误提示。要解决这个问题,就必须对不同类型的数据进行一些转换,将他们转换为合适的同一类型数据后,再进行计算,所以在进一步学习前,我们必须首先熟练掌握本节涉及的内容。

1.6.1 常量

常量(Constants)是一个具体的、保持不变的值,又称为常数。在 Visual FoxPro 6.0 中定义了 6 种类型的常量。

1.6.1.1 数值型常量(N)

数值型常量可以是整数和实数。有两种表示形式:常用表示形式、科学计数法表示形式。

例如:常用表示形式:128,75.34, -38.2, -87 等。

指数表示形式:1.6E10 表示:1.6×10¹⁰。

1.6.1.2 字符型常量(C)

字符型常量是用定界符括起来的由字符、数字、汉字以及空格符所组成的字符串。定界符是指一对半角的双引号、单引号或方括号。例如:"128","12/03/45",'Visual FoxPro 6.0',[姓名]。定界符不是字符型常量的内容,它只是字符型常量的起始和终止标志。

字符型常量的使用说明:

(1) 当字符型常量的内容中含有某种定界符时,必须用另一种定界符作为该字符型常量的定界符;

例如:<关系型数据库"Visual FoxPro 6.0">作为字符型常量的内容应表示为:

[关系型数据库"Visual FoxPro 6.0"]

或 '关系型数据库"Visual FoxPro 6.0"'。

(2) Visual FoxPro 6.0 中字符型常量的字符串最大长度为 254 个字符;

(3) 不包含任何字符的字符串叫空串,例:“ ”。空串与包含空格的字符串不同,例如包含空串的字符串:“ ”;

(4) 定界符一定要在半角状态下输入。

1.6.1.3 逻辑型常量(L)

逻辑型常量只有两个值,“真”和“假”。分别用 .T. (或 .t.)与 .Y. (或 .y.)表示“真”;用 .F. (或 .f.)与 .N. (或 .n.)表示“假”。注意:逻辑值在表示时两边的半角小圆点符号不能省略。

1.6.1.4 日期型常量(D)

日期型常量用于表示日期,表达时必须用一对大括号括起来。大括号内包括年、月、日 3 个部分,各部分之间用分隔符分隔,系统默认的分隔符为斜杠“/”。常用的分隔符还有“-”、“.”等。

日期型常量的表示格式有两种:严格日期格式和一般日期格式。

严格日期格式的表示:{^YYYY/MM/DD}

例如:{^1998/01/24} 或 {^1998-01-24} 或 {^1998.01.24}。

一般日期格式的表示:{MM/DD/YYYY}

或 {DD/MM/YYYY}

或 {YYYY/MM/DD}

例如:{01/24/1998} 或 {24/01/1998} 或 {1998/01/24}

在任何日期格式环境下,用严格日期格式表示日期型常量都是有效的,但是否可以使用一般日期格式,则由系统当前的日期格式环境决定。改变系统日期格式环境命令的格式为:

Set Strictdate To [0|1]

功能:设置系统日期格式环境。

参数说明:参数 0 和 1 两者只能选择其中之一,缺省时,默认值为 1。选择参数 1 执行该命令后,日期型常量只能用严格日期格式表示;选择参数 0 执行该命令后,日期型常量既可用严格日期格式表示,也可以用一般日期格式表示。

使用一般日期格式表示日期型常量时,要注意系统日期的表示形式,一般默认情况下为美国日期格式,但通过命令可以改变,一般日期表示形式的改变命令格式为:

Set Date To Ansi | America | Britain

参数说明:

当选择参数 Ansi 时,一般日期格式为 {YYYY/MM/DD};

当选择参数 America 时,一般日期格式为 {MM/DD/YYYY};

当选择参数 Britain 时,一般日期格式为 {DD/MM/YYYY}。

1.6.1.5 日期时间型常量(T)

日期时间型常量用于表示日期和时间,在表达时也必须用一对大括号括起来。大括号内包括年、月、日、时间 4 个部分。

日期时间型常量的表示格式有两种:严格日期时间型格式和一般日期时间型格式。

严格日期时间型格式的表示:{^YYYY/MM/DD[,][HH[:MM[:SS]]][A|P]}

例如: {1998/01/26 10:15:25a}, {1998-01-26,05:22:37p}。

一般日期时间型常量的表示: {MM/DD/YYYY[,][HH[:MM[:SS]]][A|P]]}
或 {DD/MM/YYYY[,][HH[:MM[:SS]]][A|P]]}
或 {YYYY/MM/DD[,][HH[:MM[:SS]]][A|P]]}

日期时间型常量应用时应注意的格式问题同日期型常量。

1.6.1.6 货币型常量(Y)

货币型常量用于表示货币值,其书写格式为在数字前加一个前置符号“\$”。货币型常量在存储和运算时采用保留4位小数,多于4位小数时采用四舍五入。

例如: \$48.34, \$123.89

1.6.2 变量

变量(Variables)的使用在程序中最基本的,同时也是不可缺少的数据元素。变量的值在程序运行过程中可以变化。Visual FoxPro 6.0中有三种形式的变量:字段变量、内存变量和系统变量。字段变量和内存变量的名称必须是以汉字或字母开头,且仅由英文字母、汉字、数字和下划线组成。系统变量是 Visual FoxPro 6.0 自动生成和维护的内存变量,它以下划线“_”开头,用于控制输出和显示信息的格式,其名称由系统规定,用户不能再对它重新进行定义,而只能使用它。

例 1.8 系统变量“_DiaryDate”用于设置或返回系统的当前日期。

_DiaryDate={2006/2/21} &&. 将系统时期设置为 2006 年 2 月 21 日

? _DiaryDate &&. 返回并在主窗口显示系统当前日期

1.6.2.1 字段变量

表的每一个字段就是一个字段变量。Visual FoxPro 6.0 中字段变量常用的数据类型有:字符型(C)、数值型(N)、日期型(D)、逻辑型(L)、备注型(M)、通用型(G)等。每个字段变量都是用户在建立表时定义的,其字段变量名称由 1~10(或 1~128)个字符组成,它是个多值变量,其值会随记录指针的移动而发生变化,它的当前值是记录指针所指向记录(当前记录)的相应字段值。

1.6.2.2 内存变量

内存变量是一种独立于表文件而存在的变量,内存变量建立后存储于内存中,它主要用于存储数据处理过程中所需要的数据;也可以作为控制变量来控制应用程序的运行。

内存变量名是以字母、汉字开头的,由字母、数字、汉字、下划线组成,并且其组成的字符数最多为 128。

1.6.2.2.1 用内存变量时要注意两个原则

第一:内存变量名不能与 Visual FoxPro 6.0 的保留字同名,最好也不要和当前数据表中的字段名同名。若内存变量与字段变量同名,在访问内存变量时,则必须在内存变量名前加一个前缀 m. 或 m->(一个减号加一个大于号);

第二:程序在访问内存变量前,一定要为其赋初值,否则将显示“找不到变量”的出错信息。

1.6.2.2.2 内存变量的数据类型

内存变量的数据类型包括字符型(C)、数值型(N)、逻辑型(L)、日期型(D)、日期时间型(T)和货币型(Y)等。内存变量的数据类型由其本身值的类型决定。

1.6.2.2.3 内存变量的赋值

内存变量的建立不需要事先定义,可直接通过赋值来建立内存变量,内存变量的赋值有两种格式:

格式一:

<内存变量名> = <表达式>

格式二:

STORE <表达式> TO <内存变量名表>

功能:将<表达式>的值赋给一个或多个内存变量。格式一每次仅能给一个变量赋值,而格式二每次能给多个变量赋相同的值,变量名表中的各变量间用逗号“,”分隔。

例 1.9 请将数值 345 赋给变量 Mynumber

Mynumber=345 或

Store 345 To Mynumber

例 1.10 请将日期时间值{^2006/03/12 02:59:01 PM}赋给变量 Dt1、Dt2、Dt3

Store {^2006/03/12 02:59:01 PM} To Dt1, Dt2, Dt3

1.6.2.2.4 内存变量的输出

格式一:

?[<表达式表>]

格式二:

??[<表达式表>]

功能:依次输出<表达式表>中各表达式的值,表达式间用逗号分隔。格式一表示从下一行的起始位置输出表达式的值,若省略<表达式表>,则仅仅是输出一个换行符(即换行);格式二表示从当前行的光标处输出表达式的值。

例 1.11

Clear &&. 清屏

Mynumber=345

Store {^2006/03/12 02:59:01 PM} To Dt1, Dt2, Dt3

?Mynumber &&. 从下一行的起始位置输出 345

?Dt1 &&. 从下一行的起始位置输出 03/12/2006 02:59:01 PM

??Mynumber &&. 从当前行的光标处输出 345

执行以上命令后,主窗口的显示情形:

345

03/12/2006 02:59:01 PM

345

1.6.2.2.5 内存变量的显示

格式:

List|Display Memory [Like <通配符>] [To Printer [Prompt] | To File <文件名>]

功能:显示当前已经定义的内存变量的名称、作用范围、类型和值。

参数说明:

- (1)Like 子句表示显示与通配符相匹配的内存变量;缺省时则显示全部的内存变量(包括系统变量),并同时显示当前内存变量总的个数、字节数等;
- (2)To Printer 将屏幕内容输出到打印机,使用 Prompt 则出现是否打印的提示;
- (3)To File <文件名>将显示内容存入到指定文本文件。

例 1.12

Disp Memory Like *

其执行结果:

MYNUMBER	Pub	N	345	(345.00000000)
DT1	Pub	T	03/12/06 02:59:01 PM		
DT2	Pub	T	03/12/06 02:59:01 PM		
DT3	Pub	T	03/12/06 02:59:01 PM		

1.6.2.2.6 内存变量的释放

为了节省内存空间,我们可以将不用的内存变量从内存中清除,这就是内存变量的释放,也称为内存变量的清除。

格式一:

Clear All

格式二:

Release [<内存变量名表>] [All [Like | Except <通配符>]]

功能:格式一是从内存中清除所有用户定义的内存变量;格式二是从内存中清除指定的内存变量。

例 1.13

Release Dt1,Dt2	&&. 清除内存变量 Dt1,Dt2
Release All Like M*	&&. 清除第一个字母是 M 的所有内存变量
Release All Except ?M*	&&. 清除除第二个字母是 M 以外的所有内存变量
Clear All	&&. 关闭所有的表,并选择第一个工作区为当前工作区,从内存中释放所有内存变量及用户定义的菜单和窗口

1.6.2.3 数组

数组也是内存变量的一种特殊形式,是按一定顺序排列的一组内存变量,数组中的各个变量被称为数组元素。用户在使用数组的过程中,必须遵循先定义后使用的原则。

1.6.2.3.1 数组的定义

格式:

Dimension|Declare <数组名 1>(<下标 1>[,<下标 2>]);
[,<数组名 2> (<下标 1>[,<下标 2>])...]

功能:

- (1)定义一维或二维数组,以及数组下标的上界;
- (2)系统规定各下标的下界为 1;

(3)理论上在 Visual FoxPro 6.0 中最多可定义 65000 个数组,每个数组最多可含有 65000 个元素。

例 1.14

Dimension A(5), B(3,4)

上述命令在内存中定义了一个一维数组和一个二维数组:

一维数组 A 有五个元素,其下标的下界为 1,上界为 5:

A(1)、A(2)、A(3)、A(4)、A(5)

二维数组 B 有 12 个元素,第一维的下界为 1,上界为 3,第二维的下界为 1,上界为 4:

B(1,1)、B(1,2)、B(1,3)、B(1,4)

B(2,1)、B(2,2)、B(2,3)、B(2,4)

B(3,1)、B(3,2)、B(3,3)、B(3,4)

1.6.2.3.2 数组的赋值

(1)在定义数组时,系统将数组中的各元素的初值设置为逻辑假值. F. ;

(2)在 Visual FoxPro 6.0 中允许数组中各元素取不同类型的值;

(3)用赋值命令既可以为单个数组元素赋值,也可为某个数组的全体元素赋同一值;

例 1.15

A(1) = [中华人民共和国] &&. 将常量“中华人民共和国”赋给数组 A 的第一个元素

A(2) = 123.56 &&. 将数值常量 123.56 赋给数组 A 的第二个元素

B = {^2006/10/1} &&. 将日期型常量 {^2006/10/1} 赋给数组 B 的每一个元素

B(1,1) = {^2006/5/1 14:30} &&. 将日期时间型常量赋给数组 B 的第一个元素

B(2,3) = .T. &&. 将逻辑型真值. T. 赋给数组 B 的第二行第三列元素

(4)二维数组各元素在内存中是先按行再按列的顺序存储的,所以二维数组可按一维数组来表示。

例如 b(2,3)和 b(7)是指同一个元素,b 数组在内存中的第 7 个元素的位置在本例中就是第二行第三列元素。

1.6.2.3.3 数组的访问

在访问数组时,若指明了数组的下标,则返回指明下标元素的值;若仅仅只有数组名,则返回数组的第一个元素值。

例 1.16

X = A(2) &&. 将数组 A 的第二个元素的数值型值 123.56 赋给变量 X

Y = B(2,3) &&. 将数组 B 的第二行第三列元素的逻辑型值. T. 赋给变量 Y

Y = B(7) &&. 将数组 B 的第七个元素的值赋给变量 Y,该命令与上一命令等效

Z = A &&. 将数组 A 的第一个元素的字符型值[中华人民共和国]赋给变量 Z

T = B &&. 将数组 B 的第一个元素的日期时间型值赋给变量 T

1.6.3 表达式

表达式(Expressions)是 Visual FoxPro 6.0 命令的重要组成部分,它是通过运算符将常量、变量、函数等运算对象连接起来的式子,该式子必须符合 Visual FoxPro 6.0 命

令语法规则。表达式的运算顺序:先计算优先级别高的运算;对同级别的运算符,按自左向右的顺序进行计算。单个的常量、变量、函数是表达式的特例。

根据组成表达式结果类型的不同,Visual FoxPro 6.0 将表达式分为:数值表达式、字符表达式、日期表达式和逻辑表达式。

在同一表达式中运算符有不同的优先级,按从高到低的次序排列,依次是算术运算、关系运算、逻辑运算,如表 1.7 所示。

表 1.7 算术运算符、关系运算符、逻辑运算符

运算	优先级	运算符	意义	示例(备注为该示例的显示结果)
算术	8	()	圆括号	
	7	* * 或 ^	乘方	
	6	*	乘	
		/	除	
		%	取模	? 15%—4 &&. —1 ? 15%4 &&. 3
	5	+	加	
		—	减	
关系	4	<	小于	? 100<60 &&. .F.
		>	大于	? 100>60 &&. .T.
		<=	小于等于	? 100<=99 , 3<=3 &&. .F. .T.
		>=	大于等于	? 100>=50 , 9>=9 &&. .T. .T.
		=	相等	? 100=120 &&. .F. ? [安徽大学]=[安徽],[安徽]=[安徽大学] &&. .T. .F.
		= =	完全相等	? [安徽大学]= =[安徽] &&. .F.
		<>、 #、!=	不相等	? 100 < > 11 , 2 # 3 , 8 != 8 &&. .T. .T. .F.
		\$	包含	? [安徽大学]\$[安徽],[安徽] \$ [安徽大学] &&. .F. .T.
逻辑	3	NOT、!	逻辑非	? not .T. , !.F. &&. .F. .T.
	2	AND	逻辑与	? T. and .T. , .T. and .F. &&. .T. .F.
	1	OR	逻辑或	? T. or .T. , .T. or .F. &&. .T. .T.

1.6.3.1 数值表达式

数值表达式是用算术运算符将数值型数据(常量、变量和函数)连接起来的式子,其运算结果的数据类型也是数值型。

例 1.17

? 51/3 * 2 — 35/5 * 4 &&. 显示 6

? 3 * 9^2%5+3 &&. 显示 6

? 15% - 4

?.?. 显示 -1

? 15% 4

?.?. 显示 3

这里需要说明的是取模运算是取前后两数相除的余数,余数的符号与除数相同。

1.6.3.2 字符表达式

字符表达式是使用算术运算符“+”或“-”以及逻辑运算符“\$”将字符型数据(常量、变量、函数)连接起来的式子,其运算结果的数据类型仍是字符数据。

格式一:

<字符串 1> + <字符串 2>

功能:将<字符串 1>和<字符串 2>完整的依次连接。

例 1.18

?[计算机] + [技术]

?.?. 显示长度为 11 的字符串“计算机 技术”

格式二:

<字符串 1> - <字符串 2>

功能:将<字符串 1>和<字符串 2>依次连接,但在连接的过程中将<字符串 1>的尾部空格移到结果字符串尾部。

例 1.19

?[计算机] - [技术]

?.?. 显示长度为 11 的字符串“计算机技术 ”

格式三:

<字符串 1> \$ <字符串 2>

功能:当<字符串 1>被<字符串 2>包含时,其值为真. T. ,否则为假. F. 。

例 1.20

?[计算机] \$ [计算机技术]

?.?. 为逻辑真值. T.

?[中国] \$ [中华人民共和国]

?.?. 为逻辑假值. F.

Store "我喜欢" To String1

?.?. 将字符串常量"我喜欢"赋给变量 String1

Store "计算机" To String2

?.?. 将字符串常量"计算机"赋给变量 String2

?String1 + String2

?.?. 显示长度为 13 的字符串[我喜欢 计算机]

?String1 - String2

?.?. 显示长度为 13 的字符串[我喜欢计算机]

? "喜欢" \$ String1

?.?. 显示逻辑真值. T.

? "喜欢" \$ String2

?.?. 显示逻辑真值. F.

1.6.3.3 日期表达式

日期表达式是用算术运算符“+”或“-”将日期型数据(常量、变量、函数)、数值型数据连接起来的式子。

格式一:

<日期型数据> + <数值型数据>

格式二:

<日期型数据> - <数值型数据>

功能:在<日期型数据>的基础上增加或减少<数值型数据>所表示的天数,所得结果为日期型值。

格式三:

<日期型数据 1> - <日期型数据 2>

功能:结果为<日期型数据 1>与<日期型数据 2>相差的天数,为数值型值。

例 1.21

? {^2006/03/23} + 3 &.&. 显示 03/26/2006,为日期型值

? {^2006/03/23} - 3 &.&. 显示 03/20/2006,为日期型值

? {^2006/03/23} - {^2006/02/23} &.&. 显示 28,为数值型值

1.6.3.4 逻辑表达式

逻辑表达式是指用关系运算符或逻辑运算符连接起来的运算式子,其运算结果的数据类型是逻辑型。

1. 关系运算符:可以实现同类数据间的比较运算。如字符串与字符串的比较、日期值与日期值的比较以及数字与数字的比较等。表 1.7 表示了关系运算符的种类和意义。

2. 逻辑运算符:可以实现逻辑数据间的逻辑运算。逻辑运算有 AND(与)、OR(或)以及 NOT(非)三种。表 1.7 表示了逻辑运算的种类和意义。

逻辑运算符也可以用圆点括住,如:. AND. ., OR. ., NOT. .。

例 1.22

Store 10>6 To X

Store . T. To Y

? X . And. Y &.&. 显示 . T.

1.6.4 函数

在 Visual FoxPro 6.0 中,函数(Function)可用来执行一些较特殊的复杂运算,使程序更加简洁易懂。

例 1.23 将三个不同的数据类型的变量用一个表达式输出形成一句完整的话。

姓名="张丽平" &.&. 将字符串常量赋给变量姓名

工作时间={^1995/07/12} &.&. 将日期型常量赋给变量工作时间

工资=1356.25 &.&. 将数值型常量赋给变量工资

? 姓名+"参加工作的时间是:" + Dtoc(工作时间) + ", 其月工资是" +
Str(工资,7,2) + "元。"

显示结果:张丽平 参加工作的时间是:12/07/95,其月工资是 1356.25 元。

上例是一个比较复杂的表达式,其中涉及到三种数据类型(字符型、日期型和数值型),由于在一个表达式中必须做到将要运算的运算符两边的数据类型相同,所以,如果在表达式中不使用日期转换函数 DTOC()和数值转换函数 STR(),则上述表达式在运行时,将会因出错而无法运行。当然在上例的结果中张丽平之后还有两个空格不符合中国语言表达形式,这个问题在学习完函数后,就可以轻松的解决了。

每一个函数都有一个名称,而紧接在名称之后有一对小括号,例如:EOF()。大部分的函数在小括号中需要一个或一个以上的参数(自变量),每一个函数都会有一个返回值。

函数的一般格式：

<函数名>(<参数表>)

例 1.24

X=Substr([Visual FoxPro 6.0 命令与函数], 23, 4) && 将函数的值赋给 X
? X && 显示“函数”

在使用函数的过程中应注意：

(1) 分清函数值的数据类型和函数参数的数据类型，且参数的数据类型要符合函数的要求；

(2) 函数的参数只能出现在圆括号内。调用无参数的函数时，函数名后的圆括号不能省略；

(3) 当在表达式中调用函数时，函数值的数据类型应与表达式要求的数据类型相同。

1.6.4.1 数值型常用函数

1. 平方根函数。

格式：Sqrt(<数值表达式>)

功能：求数值表达式的平方根

例 1.25

?Sqrt(16) && 显示 4

2. 取整函数。

格式：Int(<数值表达式>)

功能：求数值表达式的整数部分，舍去小数部分

例 1.26

Stor 5 To X
?Int(3.5 * X) && 显示 17

3. 四舍五入函数。

格式：Round(<数值表达式 1>,<数值表达式 2>)

功能：<数值表达式 2>的整数值规定<数值表达式 1>应保留的小数位，并对其进行四舍五入。

例 1.27

Stor 3.25 To X
?Round(2.34 * X,X-1) && 显示 7.61

4. 取模函数。

格式：Mod(<数值表达式 1>,<数值表达式 2>)

功能：用<数值表达式 1>的值除以<数值表达式 2>的值，取余数

例 1.28

?Mod(16, 5), 16%5 && 显示 1,1
?Mod(-16, 5), -16%5 && 显示 4,4
?Mod(16, -5), 16%-5 && 显示 -4,-4
?Mod(-16, -5), -16%-5 && 显示 -1,-1

5. 求最大值函数。

功能: 计算(<数值表达式 1>和<数值表达式 2>)值, 返回最大值

?Max (55 ,66)

&& 显示 66

格式: Min (<数值表达式 1>, <数值表达式 2>)

功能: 计算(<数值表达式 1>和<数值表达式 2>)值, 返回最小值

?Min (55 ,66)

&& 显示 55

1. 取子串函数。

功能：

(1) <数值表达式 1> 决定从<字符表达式>中截取子串的起始位置,<数值表达式 2> 决定截取的子串长度,若省略<数值表达式 2>或<数值表达式 2>的值大于剩下的字符长度时,则表示从起始位置截取剩余的字符串;

(2) <数值表达式 1>和<数值表达式 2>在参与函数运算时,自动舍去小数部分。

?Substr ([Visual FoxPro 6.0 命令与函数], 23, 4)

&& 显示“函数”

?Substr (「Visual FoxPro 6.0 命令与函数」, 17.9, 4.9)

&& 显示“命令”

2. 左取子串函数。

格式:Left (<字符表达式>,<数值表达式>)

功能:从<字符表达式>左边第一个字符起截取子串,<数值表达式>决定截取的子串长度。当<数值表达式>大于<字符表达式>结果的长度时,则取全部结果。

?Left (「Visual FoxPro 6.0 命令与函数」,5)

&& 显示“Visua”

3. 右取子串函数。

格式: Right (<字符表达式>, <数值表达式>)

功能:从<字符表达式>右边第一个字符起截取<数值表达式>个字符。<数值表达式>决定截取的子串长度。当<数值表达式>大于<字符表达式>结果的长度时,则取全部结果。

?Right (『Visual FoxPro 6.0 命令与函数』,10)

&& 显示“命令与函数”

4. 寻找子串函数。

格式: At (<字符表达式 1>, <字符表达式 2>)

功能:找出<字符表达式 1>在<字符表达式 2>中的起始位置,值为数值型;若<字

符表达式 2> 中不包含<字符表达式 1>时,则函数返回 0 值

例 1.34

?At ("book", "This is a book.") &&. 显示 11

5. 字符串长度函数。

格式:Len (<字符表达式>)

功能:返回<字符表达式>的字符串的长度值,值为数值型。

例 1.35

?Len ("This is a book.") &&. 显示 15

6. 空格函数。

格式:Space (<数值表达式>)

功能:产生空格字符串,<数值表达式>决定产生空格的个数。

例 1.36

?Space (10/3) &&. 显示 3 个空格

?Len (Space (10/3)) &&. 显示 3

7. 消除字符串尾部空格函数。

格式:Trim (<字符表达式>) 或 Rtrim (<字符表达式>)

说明:删去<字符表达式>结果的尾部空格

例 1.37

?[计算机]-[技术], Len ([计算机]-[技术]) &&. 显示“计算机 技术 ” 11

?trim([计算机]-[技术]), Len (trim([计算机]-[技术])) &&. 显示“计算机技术” 10

例 1.23 中遗留的问题就可以用这个函数解决,其表达式更改为:

? Trim(姓名)+"参加工作的时间是:"+Dtoc(工作时间)+" ,其月工资是"+
Str(工资,7,2)+"元。”

8. 消除字符串前导空格函数。

格式:Ltrim (<字符表达式>)

功能:删去(<字符表达式>)前导空格。

例 1.38

?[计算机], Len([计算机]) &&. 显示显示“ 计算机 ” 7

?Ltrim ([计算机]), Len(Ltrim ([计算机])) &&. 显示显示“计算机” 6

9. 清除前导和尾部空格函数。

格式:Alltrim (<字符表达式>)

功能:删去(<字符表达式>)前导空格和尾部空格。

例 1.39

?[计算机], Len([计算机]) &&. 显示显示“ 计算机 ” 8

?Alltrim ([计算机]), Len(Alltrim ([计算机])) &&. 显示显示“计算机” 6

10. 小写字符转为大写字符函数。

格式:Upper (<字符表达式>)

功能:将<字符表达式>中的小写字符全部转为大写字符。

例 1.40

?Upper ("this is a BOOK.")

&& 显示 "THIS IS A BOOK."

11. 大写字符转为小写字符函数。

格式: Lower (<字符表达式>)

功能: 将字符串<字符表达式>中的大写字符全部转为小写字符。

例 1.41

例: ?Lower ("THIS is A BOOK.")

&& 显示 "this is a book."

12. 宏替换函数。

格式: &<字符型内存变量>[.]

功能:

(1) 将<字符型内存变量>所代表的字符串去掉字符串定界符后, 重新放到该函数所在位置上;

(2) 若宏替换函数中字符型变量后跟有非空格字符时, 则宏替换函数“.”不能省略, 以表示内存变量名结束;

(3) 宏替换函数符号与内存变量名之间不能有空格。

例 1.42

X = "this is a book."

Y = "X"

?Y && 显示 "X"

?&Y && 显示 "this is a book."

A = " * " && 将字符常量 " * " 赋给变量 A

?5&A.6 && 经宏替换后表达式变为 5 * 6, 结果为 30。宏替换函数中“.”不可缺省。

宏替换函数的作用:

(1) 可把数字字符组成的字符串转变为数值;

例 1.43

C1 = "34.98"

?Int(&C1) + Round(&C1, 1) && 显示 69.0

(2) 替换命令中需要重复书写的较长部分。

例 1.44

Use Employee

A = "List For 性别 = " && 将字符串常量 "List For 性别 = " 赋给变量 A

&A "男" && 该命令相当于命令 List For 性别 = "男", 显示性别为 "男" 的所有记录

&A "女" && 该命令相当于命令 List For 性别 = "女", 显示性别为 "女" 的所有记录

例 1.45

TableName = "Employee"

Use &TableName. && 在当前工作区打开 Employee 表

TableName = "Department"

Use &TableName. && 在当前工作区打开 Department 表

1.6.4.3 日期时间型常用函数

1. 系统日期函数。

格式:Date ()

功能:返回当前系统日期(默认格式为 MM/DD/YY),为日期型值。

2. 系统日期和时间函数。

格式:Datetime ()

功能:返回当前系统的时期和时间,为日期时间型值。

3. 系统时间函数。

格式:Time ()

功能:返回系统当前的时间(默认格式为 HH:MM:SS),为字符型值。

4. 求年份函数。

格式:Year(<日期表达式>)

功能:返回<日期表达式>的年份,为数值型值。

例 1.46

?Year({^2005/05/23}) &.& 显示 2005

5. 求月份函数。

格式:Month(<日期表达式>)

功能:返回<日期表达式>的月份,为数值型值。

6. 求日期函数。

格式:Day(<日期表达式>)

功能:返回日期<日期表达式>中指定的是那一天,为数值型值。

1.6.4.4 类型转换常用函数

1. 字符与 ASCII 码之间的转换函数。

(1)由字符串转换为 ASCII 码函数;

格式:Asc(<字符表达式>)

功能:返回<字符表达式>中第一字符对应的十进制 ASCII 码值,为数值型值。

例 1.47

?Asc("A") &.& 显示 65

(2)由 ASCII 码求字符的函数。

格式:Chr(<数值表达式>)

功能:将<数值表达式>按 ASCII 码值转换成对应的字符或控制命令

例 1.48

?Chr(65) &.& 显示“A”

2. 字符串与数值之间的转换函数。

(1)将数值转换成字符串的转换函数;

格式:Str(<数值表达式 1>[,<数值表达式 2>[,<数值表达式 3>]])

功能:将<数值表达式 1>的值转换成字符串,<数值表达式 2>为转换结果的长度,
<数值表达式 3>为保留的小数位

例 1.49

?Str(234,567,6,2) &&. 显示“234.57”

(2) 将字符串转换成数值的转换函数。

格式:Val(<字符表达式>)

功能:将<字符表达式>中的字符串按从左到右的顺序依次转换成保留两位小数的数值。若<字符表达式>中含有非数字字符,则直到碰到第一个非数值字符时停止转换,并返回已经转换的结果。若<字符表达式>的第一个字符为非数字字符,则转换结果为0.00。

例 1.50

?Val("234.567"), Val("12a.56"), Val("M12a.56") &&. 显示 234.57 12.00 0.00

3. 字符串与日期之间的转换函数。

(1) 字符串转换成日期函数;

格式:Ctod(<字符表达式>)

功能:将日期格式形式表示的<字符表达式>转换成日期型数据,返回值为日期型值

例 1.51

?Ctod("2006/04/14") &&. 显示 04/14/2006

(2) 日期转换成字符串函数。

格式:Dtoc(<日期表达式>)

功能:将<日期表达式>转换成字符型数据,返回值为字符型值。

例 1.52

?Dtoc({2006/04/14}) &&. 显示“04/14/2006”

1.6.4.5 常用测试函数

1. 记录指针指向记录首部测试函数。

格式:Bof([<数值表达式> | <字符表达式>])

功能:

(1) 测试指定工作区中已打开表的记录指针是否指向记录首部,若是返回值为.T., 否则为.F.;

(2) 若指定工作区中未打开任何表,则返回值为.F.;

(3) <数值表达式> | <字符表达式>用来表示所检查的工作区,<数值表达式>为工作区数字编号(1~32767),<字符表达式>为工作区别名或工作区字母代号(A~J);

(4) 省略参数则是测试当前工作区。

2. 记录指针指向记录尾部测试函数。

格式:Eof([<数值表达式> | <字符表达式>])

功能:

(1) 测试指定工作区中已打开表的记录指针是否指向记录尾部,若是返回值为.T., 否则为.F.;

(2) 若指定工作区中未打开任何表,则返回值为.F.;

(3) <数值表达式> | <字符表达式>用来表示所检查的工作区,<数值表达式>为

工作区数字编号(1~32767), <字符表达式> 为工作区别名或工作区字母代号(A~J);

(4) 省略参数则是测试当前工作区。

3. 记录个数测试函数。

格式: Recount ([<数值表达式> | <字符表达式>])

功能:

(1) 测试指定工作区中已打开表的所有记录个数, 返回值为数据型值;

(2) 若指定工作区中未打开任何表, 则返回值为 0。

4. 记录号测试函数。

格式: Recno([<数值表达式> | <字符表达式>])

功能:

(1) 测试指定工作区中已打开表的当前记录指针指向的记录号, 返回值为数据型值;

(2) 若指定工作区中未打开任何表, 则返回值为 0。

5. 数据类型测试函数。

格式: Type ([<表达式>]) 或 Type ("<表达式>")

功能: 返回 C, N, D, L, U 等字符表示<表达式> 的值类型; 返回值为字符型值。

6. 查找测试函数。

格式: Found ([<数值表达式> | <字符表达式>])

功能: 当用户用 locate/Continue, Seek, Find 等命令进行查找操作后, 用该函数测试是否找到满足条件和记录, 返回值为逻辑值。

1.6.5 VFP 中命令的常用子句

1.6.5.1 命令和子句的书写规则

前面介绍的一些命令中, 已经涉及到一些符号, 为了方便阅读, 在介绍命令和子句的书写规则前, 我们首先介绍命令格式中一些符号的意义:

| : 表示该符号两边的命令或参数只能选择其中一个;

[] : 表示该选项为可选项;

< > : 表示该选项为用户定义的内容, 且为必选项, 若该符号外有 [] 符号, 则表示当由 [] 括起来的参数被使用时, 该选项为必选;

; : 命令若在一行无法书写完成, 或用户要将命令书写成多行时, 换行前在本行的末尾以“;”结束。表示符号“;”所在的下一行与该行为同一条命令; 在 VFP 命令窗口中, 允许一条命令的长度最多为 8192 个字符;

* : 这是一条程序的注释命令, 表示在程序中若命令行以符号“*”开头, 表示该行为注释行, 是一条非执行命令, 即: 计算机会自动跳过该行, 去执行该行的下一行命令;

&& : 这也是一条注释命令, 与上一条注释命令不同的时, 该注释可以在命令后添加, 该注释命令的用法已经在前面的例子中列出。

命令和子句的书写规则:

(1) 命令动词与子句、子句与子句、子句内各部分之间必须以空格符隔开;

(2) 命令动词必须在命令语句的第一个位置上, 各子句间次序排列允许任意调整, 但

子句内各部分次序不允许改变；

(3)命令动词与子句中的保留字,包括函数名,当字母数超过四个时,命令格式中至少要求有前四个字母；

(4)命令格式中字母大小写等效；

(5)在命令书写过程中除用户定义的内容外,其他所有字符都必须为英文半角字符,包括用户输入的空格分隔符,否则将会出现错误提示。

1.6.5.2 四种常用命令子句

在 VFP 命令中,四种常用命令子句,如[Fields] <表达式表>、[范围]、[For <条件>]、[While <条件>],经常会出现,这里我们仅以 List | Display 为例来介绍这些子句,为了保持该命令的完整性,这里也介绍了该命令的其他参数。

命令格式：

List | Display [[Fields] <表达式表>] [范围] [For <条件>] [While <条件>]；
[Off] [To Print [Prompt] | To File <文件>]

功能：按指定的参数要求,显示当前工作区中表的记录,或将显示结果送到指定的位置。

参数说明：

(1)命令动词：用来表示命令的操作,如“List”、“Display”、“Create”等；

(2)范围：用来确定该命令涉及的记录,有 4 种限定方法：

All:所有记录 Next <N>:从当前记录开始的 n 个记录

Record <N>:第 n 个记录 Rest:从当前记录起剩下的所有记录

(3)For <条件>子句:<条件>为逻辑表达式,在指定<范围>内选择符合<条件>的所有记录；

(4)While <条件>子句:在指定<范围>内开始选择符合<条件>的记录,直到碰到第一个不满足<条件>的记录为止；

(5)[Fields] <表达式表>子句:<表达式表>指定需要操作的字段等。在该命令中,使用该子句时,可以省略单词 Fields。若省略<Fields>子句,则显示除备注、通用型字段以外的所有字段；

(6)Off 子句:省略该子句时,在显示过程中不显示记录号；

(7)List 命令是滚动显示方式,Display 命令是每显示满一屏停顿,按任意键继续。缺省<范围>时,List 是对所有记录操作,Display 是对当前记录操作；

(8)To Print [Prompt]:表示在主窗口显示的同时,将结果输出到打印机打印;不带 Prompt 参数时直接打印,否则弹出打印对话框；

(9)To File <文件>:表示在主窗口显示的同时,将结果输出到文本文件<文件>中并以文本格式保存。

习 题

一、选择题

1. 用二维表来表示实体与实体之间联系的数据模型称为()。
A. 实体__联系模型 B. 层次模型 C. 网状模型 D. 关系模型
2. 数据库 DB、数据库系统 DBS 和数据库管理系统 DBMS 之间的关系是()。
A. DBS 包含 DB 和 DBMS B. DBMS 包含 DB 和 DBS
C. DB 包含 DBS 和 DBMS D. DBS 就是 DB,也就是 DBMS
3. 下列描述正确的是()。
A. 数据库中只存在数据项之间的联系
B. 数据库的数据项之间和记录之间都存在联系
C. 数据库的数据项之间无联系,记录之间存在联系
D. 数据库的数据项之间和记录之间都不存在联系
4. 数据库系统与文件系统的主要区别是()。
A. 数据库系统复杂,而文件系统简单
B. 文件系统不能解决数据冗余和数据独立问题,而数据库系统可以解决
C. 文件系统只能管理程序文件,而数据库系统能够管理各种类型的文件
D. 文件系统管理的数据量较小,而数据库系统可以管理庞大的数据量
5. VFP 是一种关系型数据库管理系统,所谓关系是指()。
A. 各记录中的数据彼此有一定的关系
B. 一个数据库文件与另一个数据库文件之间有一定的关系
C. 数据模型符合满足一定条件的二维表格形式
D. 数据库中各个字段之间彼此有一定的关系
6. 关系数据库的核心是()。
A. 数据库 B. 操作系统 C. 数据库管理系统 D. 文件
7. 项目管理器的“数据”选项卡用于显示和管理()。
A. 数据库、自由表、查询 B. 数据库、视图、查询
C. 数据库、自由表、查询、视图 D. 数据库、表单、查询
8. 项目管理器的“文档”选项卡用于显示和管理()。
A. 表单、报表、查询 B. 数据库、表单、报表
C. 查询、报表、视图 D. 表单、报单、标签
9. 在 VFP 中依次执行下列几个赋值命令:
X={^2001-01-22 20:23:20 PM}
Y=. T.
M=\$ 1234.56
N=36.89
P="12.45"
当执行完成上述命令后,X,Y,M,N,P 变量的数据类型是()。

- ## 二、简答题

- ### 三、应用题

2. 练习将各种不同类型的变量(如数值型、日期型、逻辑型、字符型等),置于同一表达式中,并且要求表达式合法。



第 2 章

表的基本操作

表是处理数据和建立关系型数据库及应用程序的基本单元。表的基本操作包括表结构的建立与修改、表记录的添加和表的维护等方面的内容。

2.1 表的建立与修改

2.1.1 表结构的概念

在 Visual FoxPro 中,表是用来存放数据的,如表 2.1 职员表所示。但是,只有事先定义好表结构后,才能将相关数据存放于表中,并且存放的数据类型必须与表结构相匹配。因此,在学习如何建立表前,必须先了解表及表结构相关的概念。

表的存在形式有两种,存在于一个数据库中的表,称为数据库表;独立于数据库的表,称为自由表。数据库表和自由表之间可以相互转换,将一个自由表添加到某一数据库中,则该自由表则转变成为这个数据库的库表;反之,若将某一数据库表从数据库中移出,则该数据库表则转变成为一个自由表。关于数据库表的有关问题我们在第三章的数据库与视图一节中详细介绍。

表(Table)是一组相关的数据按行(Row)和列(Column)排列的二维表格,每个表都有一个名称,称为表名;表中的每一列称为一个字段(Field),同一列中只能存放与字段数据类型相同的数据,不同的列的字段类型可以不同;表中每一行的各个字段的数据构成一条数据记录(Record),如表 2.1 所示。

表 2.1 职员表

字段名(列名)	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
字段数据	20050001	张丽平	女	30	汉	1995-7-12	01	T	1356.25	14856359778	Memo	Gen
	20050002	陈国庆	男	40	汉	1985-3-22	11	T	1850.6	14978589657	Memo	Gen
	20050003	方世玉	男	25	汉	2000-8-5	12	F	1042.38	14654896589	Memo	Gen
	20050004	王国真	女	22	汉	2002-9-23	11	F	988.57	14896589875	Memo	Gen
	20050005	关鹏	男	27	满	1996-7-12	13	T	1125.12	14568796225	Memo	Gen
	20050006	孙宏伟	男	50	回	1975-8-6	12	T	2056.87	1.46566E+11	Memo	Gen
	20050007	李莉	女	40	汉	1986-5-12	13	T	1768.95	14563256772	Memo	Gen
	20050008	杨剑雄	男	21	汉	2003-8-23	01	F	985.12	14567665685	Memo	Gen

由表 2.1 可以看出,表是由结构(Structure)和数据记录两大部分组成。在向表中存放数据记录前,必须首先建立表的结构,即确定表由哪些字段组成,并定义各个字段的属

性,字段的基本属性包括字段名、字段类型、字段宽度、小数位数、为字段添加索引以及能否允许为空值(NULL),如表 2.2 所示。

表 2.2 职员表的结构

字段名	类型	宽度	小数位数	索引	NULL
身份号	字符型	8			
姓名	字符型	8			
性别	字符型	2			
年龄	数值	3	0		
民族	字符型	2			
工作时间	日期	8			
部门	字符型	2			
婚否	逻辑	1			
工资	数值	8	2		
电话	字符型	12			
简历	备注型	4			
照片	通用型	4			

字段名:用来标识字段(列),是以字母或汉字开头,可由字母、汉字、数字或下划线组成的字符序列。如:身份号、XH、NAME_1 等。自由表的字段名长度为 1~10 个字符,数据库表的字段名长度为 1~128 个字符。在同一表中不允许字段名重复。

字段类型:决定存储在字段中的数据特性。常用的字段类型有字符型(C)、数值型(N)、日期型(D)、逻辑型(L)、备注型(M)和通用型(G)。对于备注型和通用型字段,其数据都保存在与表文件主名相同的备注文件中,备注文件的扩展名是 .FPT,该文件会随表的打开而自动打开。如果表文件的备注文件损坏或丢失,则该表文件将不能打开。

字段宽度:字段宽度决定允许存储数据的最大字节数。如:字符型字段所占宽度为 1—254,数值型字段所占宽度不超过 20,日期型字段所占宽度固定为 8,逻辑型字段所占宽度固定为 1,备注型字段所占宽度固定为 4,通用型字段所占宽度也固定为 4。

小数位数:只有数值型和浮点型字段可以规定小数位,请注意在设定字段宽度时,小数点应占一位。如:为将数值 12888.85 存放于字段中,其字段宽度至少应设定为 8,因为该数值的整数位为 5,小数位为 2,再加上小数点所占的位,所以其宽度至少为 8。

索引:指定是否按该字段的升序或降序排序。主要用于记录排序,该内容将在第三章的排序与索引一节中介绍。

NULL:指定该字段可以接收空值。等价于没有任何值。如在户籍系统中,区分每一个人最基本的依据便是身份证号,因此身份证号绝对不能是未知的,必须设定身份证号码字段不能接收 NULL 值。对于一些不是非常重要的字段,如身高、体重等,可以将其设定成可以接收 NULL 值。

表 2.3 字段类型与宽度

数据类型	代码	数据宽度	说明
字符型 (Character)	C	1~254 个字节	最多可储存 254 个字符,常用来存储姓名、住址、编号、电话号码等不会被用来进行数值计算的数据。
数值型 (Numeric)	N	最多 20 位	范围从 -0.9999999999E+19 到 0.9999999999E+20 之间,常用来存储用来计算的数据,包含数字、小数点、正负号的数字。
浮动数型 (Float)	F	最多 20 位	与数值类型完全相同。
货币值型 (Currency)	Y	8 个字节	常用来存储一些金融数值。
整数型 (Integer)	I	4 个字节	占内存空间少,速度快
双精度型 (Double)	B	8 个字节	存储的数值很大,精确度高。
日期型 (Date)	D	8 个字节	日期的默认格式为 MM/DD/YY(即月/日/年)。例如 04/22/06 即 2006 年 04 月 22 日
日期时间型 (DateTime)	T	8 个字节	存储的数据可包含日期和时间。例如:03/28/05 11:30:55PM
逻辑型 (Logical)	L	1 个字节	.T. (真) 或 .F. (假)
备注型 (Memo)	M	4 个字节	用来存放文本字符,存储量仅受限于可使用的存储空间,数据保存于与表文件主名相同的备注文件中,其扩展名是 .FPT。
通用型 (General)	G	4 个字节	用来存放图片、电子表格、声音、影片、统计分析图等数据,存储量仅受限于可使用的存储空间,数据保存于与表文件主名相同的备注文件中,其扩展名是 .FPT。
字符类型 (二进制)		同字符型	与字符型(Character)相同,但是其数据不会随字码页的改变而改变。
备注类型 (二进制)		同备注型	与备注型(Memo)相同,但是其数据不会随字码页的改变而改变。

2.1.2 建立表的结构

2.1.2.1 存储文件默认目录的设置

在开始介绍如何建立表结构之前,必须了解你将建立的文件是如何存放的,假设你的计算机是将 Visual FoxPro 安装在 C 盘的 VFP6 目录中即 C:\VFP6,在你第一次启动 Visual FoxPro 时,C:\VFP6 为系统原始默认目录,当建立一个文件时,若不指定或更改文件的存放位置,则该文件将会储存在默认目录 C:\VFP6 中。

能否改变默认目录?答案是肯定的,改变默认目录的方式可采用界面操作或命令两种方式。例如:先在磁盘 D 上建立 VFP 目录,再将 D:\VFP 设定为默认目录。

1. 采用界面方式设置。

例 2.1 用界面方式将 D:\VFP 设定为默认目录。

操作步骤：

(1) 击“工具/选项”菜单，弹出“选项”对话框，如图 2.1 所示；

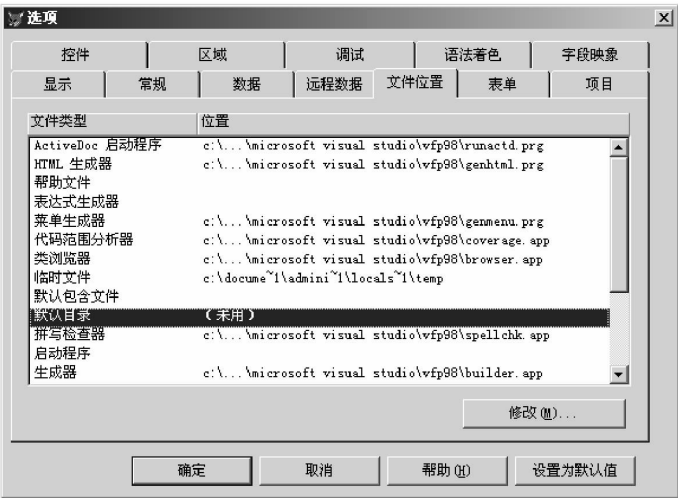


图 2.1 选项对话框

(2) 在“选项”对话框中选定“文件位置”选项卡，在列表中选定“默认目录”选项后，单击“修改”按钮或双击“默认目录”行，弹出“更改文件位置”对话框，如图 2.2 所示；

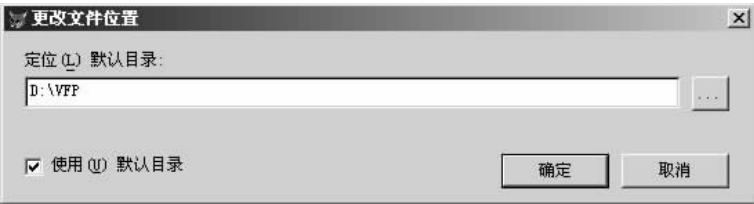


图 2.2 更改文件位置对话框

(3) 在“更改文件位置”对话框中，选定“使用默认目录”复选框，在“定位默认目录”文本框内输入文件存放的路径，单击“确定”；

(4) 路径的输入方式有两种，一是如上所述直接在“更改文件位置”对话框中输入要指定为默认目录的盘符与路径；二是通过单击“更改文件位置”对话框中的浏览按钮，弹出“选择目录”对话框，在该对话框中浏览并选定路径，如图 2.3 所示；

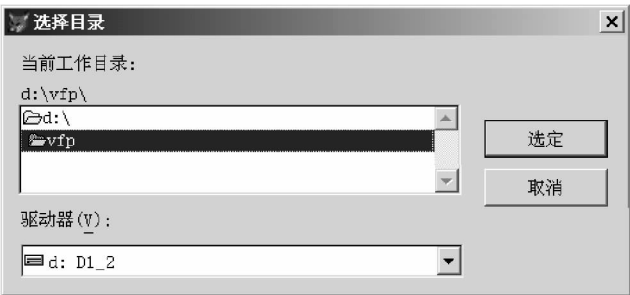


图 2.3 选择目录对话框

(5) 在“选项”对话框中,如图 2.1 所示,若单击“设置为默认值”按钮,再单击确定“按钮”,则可将该目录设置为默认目录,这样即使关闭 Visual FoxPro 后重新启动 Visual FoxPro 该默认目录也仍然有效。若只单击了“确定”按钮,没有单击“设置为默认值”按钮,则设置的默认目录仅仅只在 Visual FoxPro 没有关闭前有效。

2. 命令方式设置。

格式:

Set Default To[<磁盘目录名称>]

功能:设置存储文件默认目录,该方式的设置效果与界面方式中只单击“确定”按钮的设置效果相同。

例 2.2 将 D:\VFP 设定为默认目录

Set Default To D:\Vfp

2.1.2.2 建立表结构

1. 通过界面方式建立。

例 2.3 要求建立如表 1.1 所示的职员表(表名:Employee)的结构,其结构为:身份证号(C(8)),姓名(C(8)),性别(C(2)),年龄(N(3)),民族(C(2)),工作时间(D),部门(C(2)),婚否(L),工资(N(8,2)),电话(C(12)),简历(M),照片(G)。

建立过程如下:

(1)单击“文件/新建”菜单,弹出“新建”对话框,如图 2.4 所示;

(2)在“新建”对话框中选定文件类型“表”后,单击“新建文件”按钮,弹出“创建”对话框,如图 2.5 所示;



图 2.4 新建

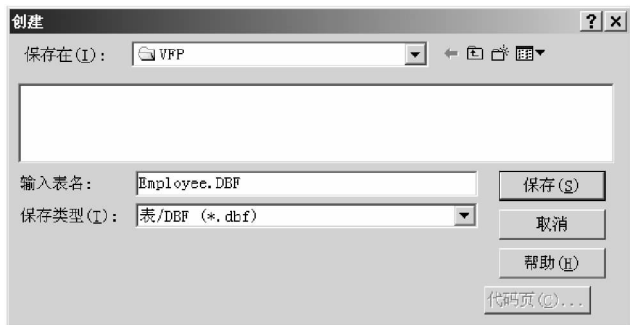


图 2.5 创建

(3)在“创建”对话框中的“保存在”文本框中选择文件保存的位置(如 D:\VFP)(在设置好默认目录的前提下,D:\VFP 会自动出现在“保存在”文本框中),并在“输入表名”对话框中输入表名“Employee”后,单击“保存”按钮,弹出“表设计器”对话框;

(4)在“表设计器”对话框中,依次输入表中每个字段的字段名、类型、宽度、小数位数,

如图 2.6 所示：



图 2.6 表设计器

(5)在“表设计器”对话框中还可对字段进行修改、删除、插入等操作，通过拖动某一字段前的方块(当某一字段为当前编辑行时，该方块上有一个上下的双向箭头)，可以调整该字段的上下位置；

(6)在“表设计器”对话框中设置好表中所有字段后，单击“确定”按钮，将弹出询问对话框，询问“现在输入数据记录吗？”，如图 2.7 所示；



图 2.7 询问对话框

(7)在询问对话框中，单击“否”按钮，完成表结构的建立；

(8)若单击“是”按钮，则可进入到数据添加的界面；

(9)无论你选择什么按钮，刚刚创建的表都将处于打开状态，通过 Visual FoxPro 窗口最下面的状态栏可以观察到表的状态。

2. 通过命令方式建立。

格式一：

Create <表名>

功能：打开表设计器，其后创建表结构的过程同界面方式。

例 2.4

Create Employee &&. 弹出如图 2.6 所示的表设计器，随后的过程同上

命令格式二：

Create Table <表名>(<字段名 1><字段类型>[(字段宽度>,
[,<小数位数>)]],<字段名 2>……)]

功能：属于 SQL 命令语句，直接通过命令构造表名为<表名>的表结构，执行该命令后，将直接创建一个在命令行中描述的完整表结构。

例 2.5 直接用命令方式建立例 2.3 中职员表 Employee 的结构。

Create Table Employee(身份证号 C(8),姓名 C(8),性别 C(2),年龄 N(3),民族 C(2),;

工作时间 D,部门 C(2),婚否 L,工资 N(8,2),电话 C(12),简历 M,照片 G)

2.1.3 修改表的结构

表的结构建立完成后,还可以修改表的结构和属性。如删除字段,或添加、更改字段的名称、宽度、数据类型,改变默认值或规则等。但要注意的是无论对表进行何种操作,都必须首先打开表,而对表操作完后,还必须关闭表,以防止表中数据丢失。

2.1.3.1 通过界面方式修改表的结构

例 2.6 通过界面方式修改 Employee 表的结构。

操作步骤:

1. 打开表。

打开方式一:

(1) 单击菜单“文件/打开”或单击工具栏“打开”按钮,弹出“打开”对话框;

(2) 在“打开”对话框中选定文件类型“表”,再选择已建立表的文件名 Employee. dbf 后,单击“确定”按钮,即可打开表,如图 2.8 所示。



图 2.8 “打开”对话框

打开方式二:

单击菜单“窗口/数据工作期”,弹出“数据工作期”窗口,如图 2.9 所示,单击该窗口中的打开按钮,弹出“打开”对话框。打开对话框的操作同方式一。



图 2.9 数据工作期

2. 通过表设计器修改表的结构。

(1) 打开表设计器的方式:

方式一: 修改当前工作区的表结构, 可直接单击菜单“显示/表设计器”菜单项, 弹出“表设计器”对话框, 如图 2.10 所示。



图 2.10 工作区属性

方式二: 在数据“工作期窗口”的“别名”框中, 选中要修改结构的表名; 单击窗口中的“属性”按钮, 弹出“工作区属性”对话框, 如图 2.11 所示; 在该对话框中单击“修改”按钮, 则弹出“表设计器”对话框。



图 2.11 “表设计器”对话框

(2) 在表设计器中修改表的结构:

① 修改某字段: 将光标移到要修改的字段上直接编辑即可;

例如: 把光标移到“身份证号”字段上, 将其字段名修改为“身份证号”, 宽度修改为 10 个字节宽度。

② 插入一个字段: 单击“插入”按钮, 会在当前字段的前一个位置插入一个新字段, 并在该新字段行上输入字段名, 设定数据类型、字段宽度等;

例如: 把光标移到“工作时间”字段上, 单击“插入”按钮, 就在“工作时间”字段前插入了一个新字段, 并在该新字段上输入字段名“籍贯”, 设定数据类型为字符型、字段宽度为 10。

③ 删除某字段: 只要将光标移到要删除的字段上, 单击“删除”按钮即可;

④完成对表结构的修改后,单击“表设计器”中的“确定”按钮或按“Ctrl+W”键保存表结构,系统弹出如图 2.12 所示的确认对话框,单击“是”按钮,即可完成表结构的修改;按“Ctrl+Q”或“取消”按钮放弃表结构修改,系统将出现提示对话框,用户按“是”或“否”确定。



图 2.12 确认对话框

3. 关闭表。

方式一:在数据工作期窗口中选中要关闭的表,单击“关闭”按钮。

方式二:关闭 VFP 主窗口时,表自动关闭。

2.1.3.2 通过命令方式修改表的结构

1. 打开表。

格式:

Use [<数据库名!>] <表名>

功能:在当前工作区打开表。

2. 修改表结构。

格式:

Modify Structure

功能:打开当前表的“表设计器”对话框,并在“表设计器”对话框中对表的结构进行修改。

3. 关闭表。

格式:

Use

功能:关闭当前工作区中的表。

例 2.7 用命令方式在表设计器中打开表 Employee.dbf,并进行修改,修改完后,关闭表。

Use Employee &&. 打开表 Employee

Modify Structure &&. 出现表设计器对话框,用户界面操作修改表结构

Use &&. 关闭表

其他与关闭表有关命令:

格式一:

Clear All

功能:关闭所有的表,并选择工作区 1;从内存中释放所有内存变量及用户定义的工作菜单和窗口,但不释放系统变量。

格式二:

Close All

功能:关闭所有打开的数据库与表,并选择工作区 1;关闭表单设计器、查询设计器、

报表设计器、项目管理器。

格式三：

Close Database [All]

功能：关闭当前数据库及其的表；若无打开的数据库，则关闭所有自由表，并选择工作区 1，带有 All 则关闭所有打开的数据库及其中的表和所有打开的自由表。

格式四：

Close Tables [All]

功能：关闭当前数据库中所有的表，但不关闭数据库；若无打开的数据库，则关闭所有的自由表。带有 ALL 则关闭所有数据库中所有的表和所有的自由表，但不关闭数据库。

2.1.3.3 SQL 命令方式修改表的结构

通过 SQL 中表结构的修改命令，可直接通过命令语句对表的结构进行直接修改，而不需要打开表设计器。

格式一：

Alter Table <表名> Add [Column] <字段名><类型>;

[(<宽度>[, <小数位数>)] [Null]

功能：向指定的表添加字段。

例 2.8 在表 Employee 中添加字符型字段“家庭住址”，宽度是 20。

Alter Table Employee Add 家庭住址 C (20)

格式二：

Alter Table <表名> Alter [Column] <字段名><类型>;

[(<宽度>[, <小数位数>)] [Null]

功能：修改指定表中的某个指定字段的类型、宽度、小数位数等，本命令不能修改字段名。

例 2.9 将表 Employee 中的部门字段的更改为数值型的，宽度为 10，两位小数。

Alter Table Employee Alter 部门 N(10,2)

格式三：

Alter Table <表名> Drop [Column] <字段名>

功能：删除指定表中的指定字段。

例 2.10 从表 Employee 中删除“家庭住址”字段。

Alter Table Employee Drop 家庭住址

格式四：

Alter Table <表名> Rename [Column] <源字段名> To <目标字段名>

功能：将指定表的<源字段名>更改为一个新的字段名。

例 2.11 将表 Employee 中“身份号”更改为“身份证号”

Alter Table Employee Rename 身份号 To 身份证号

格式五：

Drop Table <表名>

功能：删除指定的表

例 2.12

Create Table 学生名单(学号 C(8), 姓名 C(10))

&&. 创建学生名单表

Drop Table 学生名单

&&. 删除学生名单表

2.1.4 向表中输入数据**例 2.13** 向 Employee 表中添加记录,记录内容如图 2.15 所示。

前面所创建的表文件到目前为止,仅仅只有一个结构(空表),只有向表中输入数据,才能建立一个完整的有意义的表文件。

当刚刚创建好表结构,并存盘退出表设计器时,系统将会出现如图 2.7 所示的询问对话框,提示信息“现在输入数据记录吗?”,单击“是”按钮,出现如图 2.13 所示的数据输入窗口,其中显示一项空白的数据记录,以便将数据一一填入各个字段中。

图 2.13 数据输入窗口

数据输入窗口中各字段的排列次序以及光带所能容纳的字符个数都与已经定义的表结构相符。其中数值型字段的小数点位置及日期型字段的两个间隔符号“/”也已在相应的光带中标出。特别是备注型和通用型字段中已分别填入 memo 和 gen 字样,它提示有关数据将另作处理而存入数据表备注文件中。具体操作时需注意:

(1) 输入数据若超出当前字段宽度时,光标将自动移到下一个字段等待输入,同时计算机发出一个响声“嘟”。反之,若未超出当前字段宽度,则需要在输入完数据后按下 [Enter] 键,让光标移至下一个字段;

(2) 通常日期型字段数据输入的默认格式为月/日/年格式:MM/DD/YY,若需要更改日期的输入格式,请参阅第一章第六节表达式中日期型常量相关的内容,在此与日期常量不同的是在表中输入日期时,不需要使用 {} 括号,另外也不能使用严格日期格式;

例:1986 年 05 月 20 日,计算机通常的默认输入为:05/20/86

(3) 逻辑型字段宽度为 1,它只能接受 T,t,Y,y,F,f,N,n 等单个字母的输入;

(4) 备注型字段数据的输入:通过双击“memo”打开备注型字段编辑窗口,也可将光标定位于“memo”处,按 Ctrl-Home、Ctrl-PgUp 或 Ctrl-PgDn 键,打开备注型字段编辑窗口,备注型字段的编辑与记事本的文字编辑相似。当需要退出备注型字段的编辑窗口返回到表的浏览或编辑窗口时,可直接通过“关闭”按钮或按 Ctrl-W 键,系统将保存输入的

内容,并关闭备注型字段编辑窗口,返回继续输入后面的数据。若按 Esc 或 Ctrl-Q 键则弹出询问对话框,询问“放弃修改?”,单击“是”按钮,则放弃修改,并关闭备注型字段编辑窗口,返回到表的浏览或编辑窗口;单击“否”按钮,则返回到备注型字段编辑窗口;

(5)通用型字段数据的输入:通用型字段编辑窗口的进入、保存和放弃保存的操作同备注型字段。系统进入通用型字段编辑窗口后,是通过插入对象的方式来输入通用型字段内容的。其方法是:单击菜单“编辑/插入对象……”,打开“插入对象”对话框,如图 2.14 所示。插入对象的方式有“新建”和“由文件创建”两种:



图 2.14 插入对象

若选择“新建”方式,则选择“对象类型”列表框中的某一类型后,按“确定”按钮,即可进入相应插入对象类型的编辑界面创建对象,新建完成后,关闭新建窗口,即完成向通用型字段的输入数据;

若选择“由文件创建”方式,则用户可通过已经存在的文件向通用型字段输入数据,在这种方式中若选择了“链接”复选框,则表示向通用型字段输入的数据与原文件间是链接关系,即当原文件中的数据内容发生改变时,通用型字段中的数据会随之改变;若没有选择“链接”复选框,则表示向通用型字段输入的数据与原文件间是非链接关系。

完成数据输入后,则可关闭通用型字段编辑窗口,返回到表的浏览或编辑窗口;

(6)当备注型字段或通用型字段没有输入内容时,其字母“memo”或“gne”全部为小写字母;当存有数据时,其字母“Memo”或“Gne”的第一个字母为大写。

2.1.5 表中数据的修改

当表中已经存有数据后,对表中数据的修改与更新将是必然的,为了便于学习,本节只介绍简单的界面方式:在编辑或浏览窗口中对数据进行直观的修改。表中数据的修改既可在浏览窗口中进行,也可在编辑窗口中进行,无论进入上述何种窗口,只需将光标移至需要编辑的记录字段中,就可以对数据进行编辑。关于备注字段或通用字段内容的修改方式类似于这两种数据的输入方式。

2.1.5.1 打开表的浏览或编辑窗口

1. 通过界面方式打开表的浏览或编辑窗口。

通过界面操作方式在浏览或编辑窗口打开表时,要求表已经在当前工作区被打开。否则,菜单上不会出现“显示/浏览”菜单,即使单击数据工作期窗口的浏览按钮,也是首先

弹出“打开”对话框。

方式一：单击菜单“显示/浏览”，弹出表的浏览或编辑窗口，浏览方式与编辑方式的切换是在已经打开了浏览或编辑窗口的前提下，通过单击菜单“显示/浏览”或“显示/编辑”进行的。编辑窗口的形式如图 1.13 所示，浏览窗口的形式如图 2.15 所示。

Employee											
身份号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
20050001	张丽平	女	30	汉	07/12/95	01	T	1356.25	14856359778	Memo	Gen
20050002	陈国庆	男	40	汉	03/22/85	11	T	1850.60	14978589657	memo	Gen
20050003	方世玉	男	25	汉	08/05/00	12	F	1042.38	14654896589	memo	gen
20050004	王国真	女	22	汉	09/23/02	11	F	988.57	14896589875	memo	gen
20050005	关鹏	男	27	满	07/12/96	13	T	1125.12	14568796225	memo	gen
20050006	孙宏伟	男	50	回	08/06/75	12	T	2056.87	14658562328	memo	gen
20050007	李莉	女	40	汉	05/12/86	13	T	1768.95	14563256772	memo	gen
20050008	杨剑雄	男	21	汉	08/23/03	01	F	985.12	14567665685	memo	gen

图 2.15 职员表

方式二：在“数据工作期”窗口的别名中，选择要浏览的表，单击“浏览”按钮，弹出表的浏览或编辑窗口。

2. 通过命令方式打开表的浏览或编辑窗口。

格式一：

Browse

功能：打开浏览窗口，以浏览方式显示当前表记录，若当前工作区中没有表打开，则弹出“打开”对话框。

格式二：

Change | Edit

功能：打开编辑窗口，以编辑方式显示当前表记录，若当前工作区中没有表打开，则弹出“打开”对话框。

2.1.5.2 在浏览或编辑窗口中追加与删除记录

在表的浏览或编辑窗口中，用户除可以对已有的数据进行浏览（查看）和编辑、修改外，还可以通过该窗口追加或删除记录。

1. 追加记录。

在追加的过程中新的记录都将添加到表的末尾，追加记录的界面操作方式有两种。

方式一：单击菜单“显示/追加方式”，系统允许用户以连续添加的方式向表添加多个记录，直到关闭浏览或编辑窗口。

方式二：当打开浏览或编辑窗口时，系统菜单中将自动增加一个“表”菜单，单击菜单“表/追加新记录”，系统允许用户每次向表中添加一条新记录。

2. 记录的删除。

在 Visual FoxPro 6.0 中，删除表中的记录有逻辑删除和物理删除两种。对记录的删除需分两步进行：首先是在要删除的记录打上删除标记（称逻辑删除），然后再将带删除标记的记录一次性从表中清除（物理删除）。经逻辑删除的记录，用户还可以将其恢复，即去除删除标记。但要注意一旦记录被物理删除，将无法恢复，数据将永远丢失。这里仅仅

只简单介绍一下单个记录的逻辑删除与恢复方法。

方法一：在浏览或编辑窗口中，用鼠标直接在要作删除标记的记录左侧的矩形区域中单击，使该区域变黑，当这个区域变黑后，表示该记录已被打上删除标记；当记录已经打上删除标记时，若用鼠标单击该区域则是清除删除标记；

方法二：在浏览或编辑窗口中，让光标停留在要作删除标记的记录上，单击菜单“表/切换删除标记”，该记录将被打上删除标记；该菜单命令是一个重复命令，当再次单击该菜单命令时，将会清除删除标记。

关于记录追加和删除的其他界面方式以及命令方式，将在下一节中作详细介绍。

2.2 表的维护

在表中存储了有价值的数据信息。若由于某种原因使数据库遭受破坏，可能丢失部分或全部数据。为了减少不必要的损失，就需要对表进行维护，采取措施来确保表的安全。

2.2.1 表结构与表的复制

2.2.1.1 复制当前表的结构

格式：

Copy Structure To <表文件名> [Fields <字段名表>]

功能：将当前打开的表文件结构的部分或全部复制，产生<表文件名>所指定的表的结构。仅复制当前表的结构，不复制其中的数据。

参数说明：

(1) <表文件名> 指定生成新表结构的表文件名；

(2) Fields<字段名表> 指定在新表中包含的字段及顺序。若省略，则按字段原来的顺序复制全部字段。

例 2.14 利用 Employee 表的结构复制建立表 Zgda.dbf 的结构，要求表 Zgda.dbf 的结构中仅仅包括身份号、姓名、性别、工作时间字段。

Use Employee

Copy Structure To Zgda Fields 身份号, 姓名, 性别, 工作时间

2.2.1.2 复制当前表的结构和数据，形成新表或其他类型的文件

格式：

Copy To <文件名> [<范围>] [For <条件>][While <条件>]；

[Fields <字段名表> | Fields Like <通配字段名> | Fields Except <通配字段名>]；

[[Type] [Sdf] | [Xls] | [Delimited [With <定界符> | With Blank |

With Tab]]]

功能：将当前表中选定的部分记录和部分字段复制成一个新表或其他类型的文件。

说明：

(1) 含有备注或通用型字段时，在复制扩展名为.dbf 文件的同时，自动复制扩展名

为.fpt 的备注文件;

(2)复制产生的新表,处于关闭状态,只有打开新表,才能对新表进行操作;

(3)字段名可用通配符,Like 表示取与<通配字段名>匹配的字段;Except 表示取除<通配字段名>匹配的字段以外的字段;

(4)不含[Type]子句,默认新文件的类型是表。

含[Type]子句时,Type 可以省略;

Xls:新文件为 Microsoft Excel 文件;

Sdf:数据无定界符、分隔符的文本文件

Delimited:逗号为分隔符,双引号为定界符;

Delimited With <定界符>:用户指定新的字符为定界符,分隔符为逗号;

Delimited With Blank:用空格作为分隔符,双引号为定界符;

Delimited With Tab:用制表符作为分隔符,定界符为双引号。

例 2.15 将 Employee 表中男职工的姓名,性别,工作时间复制成一个新表 Zgda,并在主窗口中显示新表的记录。

Use Employee

Copy To Zgda Fields 姓名,性别,工作时间 For 性别 = [男]

Use Zgda && 关闭当前工作区中已经打开表,并同时打开 zgda 表

List && 在主窗口中显示表 zgda 中的记录

例 2.16 将 Employee 表中男职工的姓名,性别,年龄,民族,工作时间记录转换为 sdf 标准数据文件 Zgda.Txt。

Use Employee

Copy To Zgda Sdf Fields 姓名,性别,年龄,民族,工作时间 For 性别 = [男]

Type Zgda.txt && 显示文本文件 Zgda.txt 的文本内容,结果如下

陈国庆 男 40 汉 19850322

方世玉 男 25 汉 20000805

关鹏 男 27 满 19960712

孙宏伟 男 50 回 19750806

杨剑雄 男 21 汉 20030823

2.2.1.3 磁盘文件的复制

格式:

COPY FILE <源文件名> TO <目标文件名>

功能:复制<源文件名>文件生成<目标文件名>文件,文件名可以使用通配符 * 和 ?。被复制的文件必须处于关闭状态。

例 2.17 复制 Employee 表文件生成新的表文件 Zgda。

Clear

Close All

Copy File Employee.dbf To Zgda.dbf

Copy File Employee.Fpt To Zgda.Fpt

注:用该命令复制表文件时,若表文件中存在备注型字段或通用型字段,则应该同时与表文件主名同名的备注文件。

格式：

功能:显示磁盘文件目录信息,文件名可以用通配符 * 和 ?。

Dir	&.&. 显示系统默认当前盘或路径的表文件目录
Dir D:	&.&. 显示 D 盘上的表文件目录。
Dir D:*.*.fpt	&.&. 显示 D 盘根上目录下扩展名是 fpt 的所有文件。

格式：

功能：将磁盘文件改名。

例 2.19 将表名 Zgda 改名为 abc

Rename Zgda. fpt To Abc. fpt

格式一：

格式二：

功能:删除磁盘上的文件,可以使用通配符 * 和 ?。

或 delete File D:*.Bak

格式：

功能:显示或同时打印文本文件。

示例请参阅例 2.16。

我们在使用表设计工具建立一个表并输入数据记录时,按照这些记录被输入的次序,

数据库系统会将表的每一项数据记录赋予一个唯一的标记“记录号”。在对表中的数据进行修改、存取时,首先必须打开相应的表,并在打开的表中进行记录定位,然后再进行相应的操作,如修改、存取等。

实际上每一个打开的表中都存在唯一一个与当前操作的记录相关的记录指针,记录指针所指向的那一个记录,就是当前操作的记录,即当前记录。当刚打开表时,表中的记录指针就自动指向第一个记录。表文件一旦打开,表中的记录指针是可以移动的,而要移动记录指针,可以借助于界面方式和 Go, Goto 及 Skip 等命令方式。我们可以通过当前记录测试函数 Recno(⟨数值表达式>|⟨字符表达式>)来测试当前记录号。

2.2.2.1 界面方式

在界面方式下移动记录指针,必须将相应的表(如 Employee)在浏览或编辑窗口中打开(打开浏览编辑窗口分两步:首先是打开表,其次是打开浏览或编辑窗口),并单击菜单“表/转到记录”,出现如图 2.16 所示的级联菜单。然后在下一级菜单中选择相应的移动动作,菜单功能如下:



图 2.16 “表/转到记录”菜单

- (1) 第一个(T):将记录指针移动到第一个记录;
- (2) 最后一个(B):将记录指针移动到最后一个记录;
- (3) 下一个(N):将记录指针移动到当前记录所在位置的下一个记录;
- (4) 上一个(P):将记录指针移动到当前记录所在位置的上一个记录;
- (5) 记录号(R)...:将记录指针移动到指定记录号的记录处;
- (6) 定位(L)...:将记录指针移动到满足某一个条件表达式的第一个记录上。

2.2.2.2 命令方式

命令方式下移动记录指针,要求表必须处于打开状态,但表的浏览或编辑窗口可以不处于打开状态。观察当前记录的位置可以通过状态栏(如图 2.17 所示),也可以通过记录号测试函数 Recno()。



图 2.17 记录指针移动

1. 记录指针绝对移动命令。

格式一：

Go|Goto <数值表达式> | Top | Bottom

格式二：

<数值表达式>

功能:将记录指针直接移动到指定的记录上。

参数说明：

(1) <数值表达式>:指定一个物理记录号,将记录指针移至该记录上；

(2) Top:将记录指针移动到表的第一个记录；

(3) Bottom:将记录指针移动到表的最后一个记录；

(4) <数值表达式>的值必须大于 0,当<数值表达式>的值大于当前表文件的记录个数时,记录指针指向表文件的尾部。

例 2.21

Use Employee	&.&. 在当前工作区打开表,记录指针同时自动指向第一个记录
Display	&.&. 显示第一个记录内容
Go 5	&.&. 将记录指针移到第 5 个记录
Display	&.&. 显示第 5 个记录内容
3	&.&. 将记录指针移到第 3 个记录
Display	&.&. 显示第 3 个记录内容
Go Bottom	&.&. 将记录指针移到最后一个记录
Display	&.&. 显示最后一个记录内容

2. 记录指针相对移动 SKIP。

格式：

Skip [<数值表达式>]

功能:以当前记录位置为基准,移动记录指针。

参数说明: <数值表达式> 表示移动记录的个数, 若 <数值表达式> 的值为负数, 则记录指针向前移动; 若 <数值表达式> 的值为正数, 则记录指针向后移动。当指针移动超过第一个记录或最后一个记录时, 记录指针则指向文件表的首部或尾部。记录指针是否指向表文件的首部或尾部可用测试函数 Bof () 和 Eof () 测试。

例 2.22

Use Employee	&&. 打开表, 记录指针同时指向第一个记录
Skip 2	&&. 将记录指针后移 2 个记录, 当前记录为 3
? Recno ()	&&. 函数返回当前记录号, 并显示当前记录号 3
Display	&&. 显示第 3 个记录内容, 当前记录为 3
Go 5	&&. 将记录指针移到第 5 个记录, 当前记录为 5
? Recno ()	&&. 函数返回当前记录号, 并显示当前记录号 5
Skip -3	&&. 将记录指针前移 3 个记录, 当前记录为 2(5-3)
Display	&&. 显示第 2 个记录内容, 当前记录为 2
Skip -3	&&. 将当前记录前移 3 个记录, 但超范围, 记录指针指向文件首部
? Recno ()	&&. 函数返回当前记录号, 并显示当前记录号 1
Display	&&. 显示第 1 个记录内容, 当前记录为 1
? Bof ()	&&. 返回. T. , 表示记录指针指向文件首部
? Eof ()	&&. 返回. F. , 表示记录指针没有指向文件尾部
Go Bottom	&&. 将记录指针移到最后一个记录
Skip	&&. 将记录指针向后移动一个记录, 记录指针指向文件尾部
? Recno ()	&&. 函数返回当前记录号, 并显示当前记录号为表记录个数加 1
Display	&&. 没有任何信息显示
? Bof ()	&&. 返回. F. , 表示记录指针没有指向文件首部
? Eof ()	&&. 返回. T. , 表示记录指针指向文件尾部

注意: 不仅仅记录指针的移动命令可以移动记录指针, VFP 的一些其他命令, 如 List, Copy 命令等也会对当前操作的表的当前记录产生影响, 大家可以在练习的过程中用 Recno ()、Bof () 和 Eof () 等函数进行测试。

2.2.3 插入与追加记录

追加记录除了前面提到的在建立好的表的结构时立即输入数据方式, 以及通过浏览编辑窗口, 用菜单“显示/追加方式”或“表/追加新记录”追加记录外, 还可以采用其他界面和命令方式添加数据。

2.2.3.1 界面方式

在打开表的浏览或编辑窗口的前提下, 单击“表/追加记录...”菜单, 将出现“追加来源”对话框, 可将另一个表文件的一批记录追加到当前表中, 如图 2.18 所示。

(1) 选择“类型”为 Table(DBF);

(2) 在“来源于”文本框中输入表名, 或者单击“...”按钮, 弹出“打开”对话框, 从中选择一个表文件;

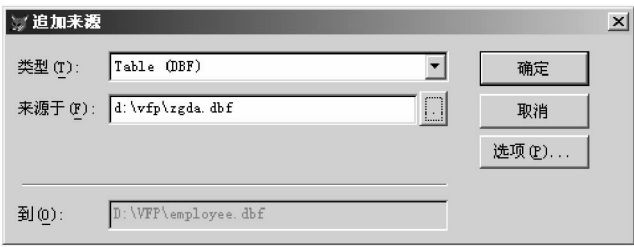


图 2.18 追加来源对话框

(3) 单击“追加来源”对话框中的“确定”按钮,则将刚才选定表中的一批记录追加到当前表中;

(4) 在这种追加方式中还可能通过单击“选项”按钮对追加记录的字段和条件进行筛选。

2.2.3.2 命令方式

1. 追加记录。

格式一:

Append [Blank]

功能:在当前使用的表文件尾部追加记录。

参数说明:

(1) Blank 是可选项,当有参数 Blank 时,系统自动在打开的表的末尾追加一条空白记录,用户可以使用 Browse,Change,Edit 或 Replace 命令编辑新记录;

(2) 当没有参数 Blank 时,该命令相当于界面操作方式的“显示/追加方式”操作,但与该方式不同的是,若当前工作区没有表打开时会自动弹出“打开”对话框。

例 2.23

Use Employee &&. 打开 Employee 表
Append &&. 以浏览或编辑窗口的形式追加记录

例 2.24

Use Employee
Append Blank &&. 在表的尾部追加一条空白记录,与浏览编辑窗口是否打开无
 &&. 关,若浏览编辑窗口没有打开,执行完该命令后,也不会弹
 &&. 出浏览编辑窗口。

格式二:

Append From <文件名> [Fields <字段名表>] [For <条件>];
[[Type][Sdf][Xls][Delimited[With <定界符>|With Blank|With Tab]]]
功能:将<文件名>指定的文件中满足<条件>的所有记录,追加到当前表的尾部。
参数说明:[Type]的用法同表与表的复制命令中的用法;源文件可以不打开。

例 2.25 将表 Employee 中的男职工的姓名、年龄、工作时间记录追加到当前表 Zgda 中。

Use Zgda
Append From Employee Fields 姓名,年龄,工作时间 For 性别=[男]

格式三：

Insert Into <表名> From Array <数组名> | Form Memvar

功能：属于 SQL 插入命令，将数组或同名内存变量中的数据追加到表的尾部。

参数说明：

(1) 表可以不打开；

(2) 将数组中的数据依次追加到记录尾部，若数组为一维数组，则追加一条记录，若为二维数组，则可追加多条记录；

(3) 将与字段同名的内存变量的值，追加到记录尾部。

示例请参阅例 2.32、例 2.34

2. 插入记录。

格式一：

Insert [Before] [Blank]

功能：在当前表的当前记录前或后插入一条新记录。

参数说明：

(1) 无任何选项时，打开浏览编辑窗口，并在表的当前记录之后插入一个空白；

(2) 含“Before”选项时，打开浏览编辑窗口，并在表的当前记录之前插入一个空白；

(3) 含“Blank”选项时，在当前记录的指定位置，自动插入一条空白记录，并不会弹出浏览编辑窗口。

例 2.26 在员工情况表的第 6 条记录之前插入一条空记录。

Use Employee &&. 打开表文件 Employee

Go 5 &&. 将记录指针移动到第 5 条记录

Insert Blank &&. 在第 5 条记录之后插入空记录(即在第 6 条记录之前插入空记录)

Use &&. 关闭表文件 Employee

上例还可以用以下命令实现

Use Employee

Go 6 &&. 将记录指针移动到第 6 条记

Insert Before Blank &&. 在第 6 条记录之前插入一条空白记录

格式二：

Insert Into <表名>[(<字段名 1> [, <字段名 2> ...]];

Values (<表达式 1> [, <表达式 2> ...]]

功能：属于 SQL 插入命令，在表尾追加一个新记录，并直接写入记录数据。

参数说明：

(1) 表可以不打开，<字段名>与所对应的<表达式>类型必须一致；

(2) <字段>缺省时，表达式的顺序与表结构中字段顺序一致。

例 2.27 在 Employee 表中插入一条完整记录。

Insert Into Employee (身份号, 姓名, 性别, 年龄, 民族, 工作时间, 部门, 婚否;

工资, 电话) Values("2005009", "欧阳红", "女", 28, "汉", {"1998/6/20"}, "02", ;
. T., 1300, "13856359886")

2.2.4 删除与恢复记录

前面介绍了在浏览或编辑窗口中,用鼠标操作和菜单“表/切换删除标记”操作,直接对记录作删除标记或清除删除标记的两种方法,这里我们将介绍其他几种删除与恢复记录的方法。

2.2.4.1 界面方式

在打开表的浏览或编辑窗口的前提下,单击“表/删除记录(D)...”,打开“删除记录”对话框,如图 2.19 所示。可逻辑删除满足一定条件和范围的记录。



图 2.19 “删除”对话框



图 2.20 “恢复记录”对话框

单击“表/恢复记录(E)...”打开“恢复记录”对话框,如图 2.20 所示。可恢复满足一定条件和范围的记录。

单击“表/彻底删除(M)”,打开“确认删除”对话框,如图 2.21 所示。若单击“是”按钮,将彻底删除所有带有删除标记的记录,删除的记录将无法恢复。



图 2.21 确认删除

2.2.4.2 命令方式

1. 逻辑删除记录。

格式:

Delete [<范围>] [For <条件>] [While <条件>]

功能:对当前表文件中指定范围内满足条件的记录作删除标记。若省略<范围>和<条件>,则只给当前记录加删除标记。

说明:执行该命令只是在这些记录前面加了一个删除标记,并没有真正删除。

2. 恢复逻辑删除的记录。

格式:

Recall [<范围>] [For<条件>] [While<条件>]

功能:取消满足一定条件和范围的记录的删除标记。若省略<范围>和<条件>,则只取消当前记录删除标记。

3. 物理删除带有删除标记的记录。

格式:Pack

功能:直接将作过删除标记的记录从表文件中永久删除,并将记录号重新排列。

说明:使用该命令时,一定要慎重,确认无误后,再使用 Pack 命令永久删除。一旦删除记录则不能恢复。

4. 物理删除表中全部记录。

格式:Zap

功能:将当前表文件中的所有记录删除掉,仅保留表结构。

说明:这是不可恢复的物理删除,要慎用,它相当于对表执行了 Dele All 和 Pack 两条命令。

例 2.28 删除记录。

Use Employee

- Delete All For 性别 = "男" &&. 给男职员作删除标记
- List &&. 显示逻辑删除情况,有删除标记的记录前有 * 号
- Recall All &&. 取消所有逻辑删除标记
- List &&. 显示
- Go 6 &&. 定位到 6 号记录
- Dele &&. 给 6 号记录作删除标记
- Pack &&. 物理删除 6 号记录

2.2.5 数据的替换

2.2.5.1 界面方式

- (1)单击“文件/打开...”菜单,打开数据表;
- (2)单击“显示/浏览”菜单,打开数据表浏览窗口;
- (3)单击“表/替换字段...”菜单,打开“替换字段”对话框,如图 2.22 所示;



图 2.22 “表/替换字段...”菜单

(4)在“替换字段”对话框里,选择替换字段名,输入替换值,选择作用范围,输入替换条件,单击替换按钮。如图 2.23 所示:将表中所有身份号是“20050006”的电话字段值替换为“12345678”。



图 2.23 替换字段对话框

2.2.5.2 命令方式

2.2.5.2.1 成批修改数据

格式一:

```
Replace <字段名 1> With <表达式 1> [Additive];
    [, <字段名 2> With <表达式 2> [Additive]...];
    [<范围>] [For<条件>] [While<条件>]
```

功能:在当前表的指定记录中,将有关字段的值用相应的表达式值来替换,若<范围>和<条件>选项都缺省,只对当前记录进行替换。

参数说明:

(1)该命令对<范围>内符合<条件>的记录用<表达式 i>的值来替换<字段 i>的值;

(2)Additive 用于备注型字段,表示将值添加到字段原来内容之后,如果缺省,则表示以<表达式>的内容替换原有内容。

例 2.29 将表 Employee 中的身份号为 20050006 的员工电话替换成 12345678,工资加 20 元。

```
Replace All 电话 With "12345678",工资 with 工资+20 For 身份证号="20050006"
```

格式二:

```
Update <表名> Set 字段名 1=表达式 1,字段名 2=表达式 2...[Where <条件>]
```

功能:表可以不打开,属于 SQL 更新命令,根据条件更新表中指定字段的数据。

参数说明:省略[Where <条件>]可更新所有数据行的值。

例 2.30 将员工表 Employee 中 1985 年以前工作的人员的工资增加 60 元。

```
Update Employee Set 工资=工资+60 Where 工作时间<{^1985/01/01}
```

利用函数可以使表达式更加简洁易懂,以上示例的等效命令:

```
Update Employee Set 工资=工资+60 Where Year(工作时间)<1985
```


2.2.5.2.2 单个记录与数组间的数据传送

1. 将当前记录传送到数组或内存变量。

格式：

Scatter [Fields <字段名表> | Fields Like <通配字段名> | Fields Except;
<通配字段名>] [Memo] Memvar [Blank] | To <数组名> [Blank]

功能：将当前记录的字段值按<字段名表>的顺序传递给数组或内存变量。

参数说明：

- (1)使用 Memvar 将数据复制到一组内存变量中。变量由系统自动产生,与相应的字段名同名,与字段同名的内存变量用 m.<内存变量名>或 m-><内存变量名>表示;
- (2)To <数组名>:将数据复制到数组中。若数组未定义,系统自动产生,若数组元不够,系统自动扩大;
- (3)[Fields]缺省只传递除备注字段外的所有字段,若要传递备注字段,需加[Memo]选项;

- (4)有[Blank]时,产生空的内存变量,或数组中元素值为空。

例 2.31

Use Employee.dbf

Scatter Fields 身份号,姓名,简历 Memo To M

&&. 数组 M 的第一、二、三元素分别获取当前记录的身份号、姓名与简历值

? M(1) &&. 显示存储在 M(1)中的当前记录的身份号

? M(2) &&. 显示存储在 M(2)中的当前记录的姓名

? M(3) &&. 显示存储在 M(3)中的当前记录的简历

2. 将数组或内存变量的数据传送到当前记录。

格式：

Gather Memvar | From <数组名> [Fields <字段名表> | Fields Like;
<通配字段名> | Fields Except <通配字段名>] [Memo]

功能：将数组或内存变量的数据依次传送到当前记录,以替换相应字段的值。

参数说明：

- (1)Gather 是将数据传送到当前记录,所以在执行该命令前,需对记录进行定位;
- (2)若数组元素多于字段数,则多出的数据不传送,若少于字段数,则多出的字段其值不变;
- (3)内存变量与字段间是同名传送,不同名则不传送;
- (4)使用 Fields 子句,仅<字段名表>中的字段被替换;
- (5)缺省 Memo 时忽略备注字段。

例 2.32

Use Employee

Dime A(2)

&&. 定义数组

A(1) = "20060006"

&&. 给数组的第一个元素赋值

Go 6

&&. 将记录指针指向 6 号记录

Gather From A Field 身份号 &&. 将 A 的第一个元素值传递给当前记录的身份号字段

Insert Into Employee From Array A &&. 将数组追加到表尾,形成一条新记录

2.2.5.2.3 成批记录与数组间的数据传送

1. 将表的一批记录复制到数组。

格式:

Copy To Array <数组名> [Fields <字段名表>][<范围>];

[For<条件>][While<条件>]

功能:将当前表选定的数据复制到<数组>中,但不复制备注字段。

参数说明:

(1)若数组不存在,系统自动创建,但若数组已经存在,则不再自动扩大,因此,可能会有数据传递的遗失;

(2)可将单个记录的数据复制到一维数组中;

(3)将当前表中的多个记录复制到二维数组中。

例 2.33 将 Employee 表中的所有“男”性记录传送到数组中。

Use Employee

Copy To Array B Field 身份号,姓名,性别,工作时间,工资 For 性别="男"

2. 从数组向表追加记录。

格式:

Append From Array <数组名> [For<条件>][Fields <字段名表>]

功能:将满足条件的数组行数据按记录依此追加到当前表的尾部,忽略备注型字段。

说明:

(1)一维数组追加一个记录,二维数组一行追加一个记录;

(2)若数组列数多于字段数,多余列数据忽略;若数组列数少于字段数,则多出的字段留空。

例 2.34

Dime D(7)

D(1) = "20060008"

D(2) = "张丰"

D(3) = "男"

D(4) = 40

D(5) = "汉"

D(6) = {^2006/03/21}

D(7) = "11"

Use Employee

Append From Array D

&&. 在表中追加一条记录

Insert Into Employee From Array D

&&. 与上一条命令等价

在上例中给数组元素赋值时,要注意与字段所对应的元素值的类型,应与字段类型一致,否则将会出现数据传送错误。

例 2.35 利用例 2.33 所产生的数组 B,将数组 B 中的 2000 年前工作的“男”性记录追加到 Employee 表尾。

Append From Array B Field 身份号,姓名,性别,工作时间,工资 For Year(工作时间)<2000

2.2.6 逻辑表的设置

1. 过滤器

格式:Set Filter To [<条件>]

功能:屏蔽不满足条件的记录。

说明:缺省<条件>表示取消前面所设置的过滤器。

例 2.36

Use Employee

- | | |
|--------------------------|-------------------------|
| Set Filter To 工资>1200.00 | &&. 设置过滤器 |
| List | &&. 仅仅只显示工资大于 1200 元的记录 |
| Set Filter To | &&. 取消过滤器的设置 |
| List | &&. 显示全部记录 |

2. 字段表

格式一:

Set Fields To [[<字段名 1>[,<字段名 2>…]]];
All[Like <通配字段名>| Except <通配字段名>]]

格式二:

Set Fields On | Off

注:

- 1. Set Fields To 命令用来为当前表设置字段表,All 表示所有字段都在字段表中;
- 2. Set Fields On | Off 决定字段表是否有效;
- 3. 执行 Set Fields To 后,Set Fields 自动置为 On。

例 2.37

Use Employee

- | | |
|---------------------|------------------------|
| Set Fields To 姓名,工资 | &&. 设置字段表 |
| List | &&. 显示所有记录,但只显示姓名与工资字段 |
| Set Fields Off | &&. 设置字段表无效 |
| List | &&. 显示所有记录的所有字段 |

习 题

一、选择题

1. 在表尾追加记录的命令是()。
A. Append B. Change C. Replace D. Insert
2. 在修改表结构时,先打开表,再选择()菜单的“表设计器”项。
A. 编辑 B. 显示 C. 表 D. 工具
3. 要想在 Student 表中存储学生照片,则应在表结构中增加()型字段。
A. 备注 B. 字符 C. 通用 D. 逻辑
4. 使用 Replace 命令时,如果范围是 All 或 Rest,则执行该命令后记录指针指向()。
A. 未记录 B. 首记录 C. 未记录的后面 D. 首记录的前面
5. 设 Student 有 20 条记录,执行以下命令后,当前记录号是()。
Use Student
Go 8
Skip 4
A. 12 B. 10 C. 13 D. n
6. 在 Visual FoxPro 6.0 的表结构中,日期型、逻辑型和备注型字段的宽度分别为()。
A. 1,8,10 B. 8,1,4 C. 6,8,10 D. 5,8,任意
7. 逻辑删除和物理删除的命令分别是()。
A. Delete, Clear B. Zap, Delete
C. Delete, Pack D. Zap, Pack
8. 当前表中有 4 个数值型字段:数学、英语、网页设计和总分。其中数学、英语和网页设计的成绩已录入,总分字段为空。要将所有学生的总分自动计算出来并填入总分字段中,使用的命令是()。
A. Replace 总分 With 数学+英语+网页设计
B. Replace 总分 With 数学、英语、网页设计
C. Replace 总分 With 数学+英语+网页设计 All
D. Replace 总分 With 数学+英语+网页设计 For All
9. 数据表中,“是否团员”为逻辑型字段,显示所有非团员的记录的命令是()。
A. List For 是否团员="F" B. List For 是否团员="N"
C. List For 是否团员 D. List For . Not. 是否团员
10. 在 Visual FoxPro 6.0 中,如果有一数据表目前处于浏览状态,若要在浏览窗口中直接从记录尾添加记录,则首先应执行“显示”菜单中的()命令。
A. 编辑 B. 追加方式 C. 表设计器 D. 追加新记录
11. 在 Visual FoxPro 6.0 中,ZAP 命令的功能是()。
A. 删除当前数据表文件中的全部记录
B. 删除当前数据表文件

- C. 恢复当前数据表文件做过删除标记的记录
D. 删除所有数据表文件
12. 在 Visual FoxPro 6.0 中要打开一个数据表,应选择主菜单中的()菜单项。
A. 编辑 B. 文件 C. 数据库 D. 显示
13. 不能对记录进行编辑修改的命令是()。
A. Edit B. Browse C. Change D. Modify Structure
14. 数据表文件中备注型字段的宽度为 4,它用来存储()。
A. 指向 .ftp 文件的指针 B. .dbf 文件的文件名
C. 指向 .dbf 文件的指针 D. 备注的具体内容
15. 要求一个表的数值型字段具有 5 位小数,该字段的宽度最少应是()。
A. 5 位 B. 6 位 C. 7 位 D. 8 位
16. 当前数据表中有 N 条记录,若想在 3 号记录后面增加一条新记录,在执行命令 "Go 3"后,应执行()命令。
A. Insert B. Browse C. Replace D. Insert Before
17. 用 Use 命令打开一个表文件后,其记录指针指向()。
A. 第一条记录 B. 任意一条记录
C. 最后一条记录 D. 文件尾

二、判断题

1. 独立于数据库存在的表,称为数据库表。 ()
2. "表"菜单是在浏览表时才出现在菜单栏上的,是动态菜单。 ()
3. 用 Display 命令显示数据时,若未指定<范围>,则显示当前一条记录。 ()
4. 记录的删除分为逻辑删除和物理删除。 ()
5. List 命令和 Display 命令在使用上完全相同。 ()
6. <范围>子句 All,使用它可以指出命令对表文件的作用范围是从当前记录开始至最后一个记录进行操作。 ()

三、填空题

1. 要给数据表中备注型字段输入内容,可以通过_____备注型字段的 memo 区。
2. 表由_____和_____两部分组成。
3. 要将当前表中"总分"字段值全部清零,可使用_____命令。
4. 若要恢复用 Delete 删除的记录,应使用_____命令。

四、应用题

1. 建立如下所示的三个数据表 Collegedepartment. dbf, Student. dbf, Reportcard. dbf:

Collegedepartment. dbf 的结构:

系部代码(C,2)、系部名称(C,16)

Collegedepartment. dbf 的记录:

系部代码	系部名称
01	文学

02	数学
03	计算机科学
04	工商管理
05	机械工程
06	哲学
07	建筑工程

Student. dbf 的结构:

学号(C,10)、系别(C,2)、姓名(C,10)、性别(C,2)、出生日期(D)、籍贯(C, 8)、是否团员(L)、备注(M)、照片(G)

Student. dbf 的记录:

学号	系别	姓名	性别	出生日期	籍贯	是否团员	备注	照片
2003010001	01	王晓玲	女	22-Jan-85	湖北武汉	. T.		
2003020001	02	丁 华	男	21-Jun-85	浙江杭州	. F.		
2003030001	03	潘亚江	男	11-Sep-84	江苏南京	. T.		
2003040001	04	程 东	男	12-Oct-85	山东济南	. F.		
2003050001	05	李 卫	男	22-Nov-86	安徽合肥	. T.		
2003060001	06	高天翔	男	23-Dec-85	江西南昌	. F.		
2003070001	07	汪 佳	女	15-Feb-84	上海	. F.		
2003010002	01	朱世玉	男	24-Mar-85	北京	. T.		
2003070002	07	陈 娟	女	11-Aug-85	重庆	. F.		
2003050002	05	欧阳瑞	男	01-Sep-85	四川成都	. F.		
2003040002	04	李 芳	女	16-Jul-86	云南昆明	. T.		
2003060002	06	汪 波	男	26-Apr-85	广东广州	. F.		
2003020002	02	陈琳琳	男	24-Oct-84	福建福州	. F.		
2003030002	03	王 琴	女	01-Dec-84	海南三亚	. F.		
2003050003	05	郭 芳	女	30-Nov-86	河南郑州	. F.		
2003070003	07	赵俊臣	男	12-May-85	天津	. T.		
2003040003	04	孙镇西	男	13-Jul-84	陕西西安	. F.		
2003030003	03	李 杰	男	06-Jun-85	山西太原	. F.		
2003010003	01	周世芬	女	28-Jul-84	贵州贵阳	. F.		

Reportcard. dbf 的结构:

学号(C,10)、应用文(N,3)、高等数学(N,3)、计算机基础(N,3)、VB 程序设计(N,3)、平均分(6,2)、总分(N,3) 名次(N,2)

Reportcard. dbf 的记录:

学号	应用文	高等数学	计算机基础	vb 程序设计	平均分	总分	名次
2003010001	82	85	78	89	0	0	0
2003020001	89	87	85	88	0	0	0
2003030001	78	92	94	89	0	0	0
2003040001	90	77	84	87	0	0	0
2003050001	85	83	90	89	0	0	0
2003060001	81	70	67	64	0	0	0
2003070001	70	81	89	93	0	0	0
2003010002	88	82	88	86	0	0	0
2003070002	66	66	60	75	0	0	0
2003050002	85	66	70	77	0	0	0
2003040002	80	84	88	89	0	0	0
2003060002	80	67	73	79	0	0	0
2003030002	55	63	45	27	0	0	0
2003020002	79	63	66	87	0	0	0
2003050003	73	64	34	80	0	0	0
2003070003	85	84	88	88	0	0	0
2003040003	85	66	76	80	0	0	0
2003030003	80	64	71	84	0	0	0
2003010003	62	62	62	61	0	0	0

操作要求:

(1) 建立上述三个数据表的结构;

(2) 建立好表结构后,立即在三个表中输入相应的记录;Student. dbf, Reportcard. dbf 只要求先输入前 10 条记录,表 Student. dbf 的备注和照片字段也请至少输入两条记录的数据,照片可用图片替代;完成后请关闭表;

(3) 分别重新打开表 Student. dbf, Reportcard. dbf, 记录追加最后几条记录。

2. 打开表 Student. dbf, 以命令方式完成下列要求:

(1) 显示表的结构;

(2) 显示表的全部记录;

(3) 显示所有男同学的姓名、出生日期、籍贯和是否团员四个字段的内容;

(4) 显示 1985 年出生的所有女生的记录;

(5) 显示所有“王”姓或“陈”姓同学的记录;

(6) 显示姓名第二个字是“世”的所有记录;

(7) 显示籍贯中含有“州”的所有同学的记录;

(8) 将记录指针移到第 5 条记录;

(9) 在第 5 条记录前后各插入一条空记录,要求原第 5 条记录前后均为一条空记录;

(10) 逻辑删除学号为 2003030001 的记录;

- (11) 将学号为 2003030001 的记录恢复删除;
- (12) 将刚才插入的两条空记录从表中彻底删除;
- (13) 利用命令方式在表中直接添加一条记录;
- (14) 利用命令方式在表中第 5 条记录之后添加有字段内容的一条记录;
- (15) 复制表的结构,产生一个新表;
- (16) 将系别为“01”和“05”的所有记录,复制到一个 Excel 表 Student.xls 中;
- (17) 将表 Student.xls 中的记录追加到表的尾部。

3. 打开表 Reportcard.dbf,以命令方式完成下列要求:

- (1) 计算每个同学的平均分和总分,并填入到表平均分和总分字段中;
- (2) 给高等数学成绩为 60 至 70 之间的同学的高等数学成绩每人增加 5 分;
- (3) 显示高等数学成绩为 70 至 90 之间,并且计算机基础成绩为 80 至 95 之间的所有记录;
- (4) 关闭 Reportcard.dbf 表,利用命令方式直接向该表中追加一条记录,完成追加后观察该表是否打开,以及记录的追加情况;
- (5) 复制表 Reportcard.dbf 产生一个新表 Reportcard1.dbf;
- (6) 利用命令方式向新表 Reportcard1.dbf 添加两个字段:姓名(C,10)、性别(C,2);
- (7) 利用命令方式将新表 Reportcard1.dbf 性别字段更改为奖学金(N,7,2);
- (8) 利用命令方式删除新表 Reportcard1.dbf 的姓名字段。



第 3 章

查询与统计

数据查询与统计是数据库系统中经常使用的两种基本应用。在本章中我们将学习排序与索引,数据工作期、查询设计器和视图设计器 3 种界面操作查询工具,以及统计命令与函数,最后我们还将学习数据库及数据库表等内容。

3.1 排序与索引

建立表的结构,向表中输入记录,并不是数据库系统的目的,数据库系统的一个重要应用,就是查询。在我们建立好表的结构,并向表输入一些数据后,表中的数据是以记录号为序排列。如果直接在以记录号为序的表中查询数据,将大大降低查询的效率。如果将表中的记录按照需要查询数据的特征进行排列,则将大大提高记录的查询速度。排序与索引就是将表中的记录,按照用户需要的特征进行重新排列。

排序将在原始表的基础上,依据一定的排序规则,产生一个新的独立的排序表。新产生的排序表与原始表之间相互独立,其中任何一个表中的数据若发生变化,将不会影响另一个表的数据。因此,这将产生一个新的问题:即当这种情况出现时,将会发生数据的不一致现象。正是由于这一问题存在,所以索引在数据库中的应用比排序要广泛地多。

索引是由指针构成的文件,这些指针逻辑上按照索引关键字值进行排序,当索引为主索引时,表中的记录将按照主索引关键字值进行排列,此时对数据进行的任何操作,都是直接对表中数据的操作,从而避免了数据的不一致现象的产生。索引文件和表文件是分别存储的,索引文件并不改变表中记录的物理顺序。

3.1.1 排序

格式:

Sort To <表文件名> On <字段名 1> [/A | /D] [/C];

[,<字段名 2> [/A | /D] [/C] ...];

[Ascending|Descending] [<范围>] [For <条件>] [While <条件>];

[Fields <字段名列表> | Fields Like <通配字段名> | Fields Except <通配字段名>]

功能:对当前选定的表排序,并依据排序后的记录创建名为<表文件名>的新表。

参数说明:

- 例 3.1** 对 Employee 分别按以下要求排序:

(2) 将已婚职工按部门降序排序。

①Use Employee;

Use Gzsj && 打开排序文件

List && 显示排序文件的结果,结果如下

②Use Employee.

Use Bmpx && 打开排序文件

List && 显示排序文件的结果,结果如下

记录号	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
1	20050005	关鹏	男	27	满	07/12/96	13	.T.	1125.12	14568796225	memo	gen
2	20050007	李莉	女	40	汉	05/12/86	13	.T.	1768.95	14563256772	memo	gen
3	20050006	孙宏伟	男	50	回	08/06/75	12	.T.	2056.87	14656562328	memo	gen
4	20050002	陈国庆	男	40	汉	03/22/85	11	.T.	1850.60	14978569657	memo	Gen
5	20050001	张丽平	女	30	汉	07/12/95	01	.T.	1356.25	14856359778	Memo	Gen

3.1.2 索引

3.1.2.1 索引的概念

1. 索引的概念。

索引是对表文件中相关记录按照索引关键字值进行的逻辑排序。索引文件仅仅只包含关键字和记录号两个字段,所以索引文件要比表文件小得多;索引文件中的所有记录按照索引关键字值的升序或降序排列,每个值对应原表文件的一个记录号,这样便确定了记录的逻辑顺序。如果要按某一逻辑顺序输出所有记录,系统将按索引文件中索引关键字值的顺序所对应的记录号依次取出表中的物理记录,从而达到以关键字值顺序来列出记录的效果。

增删或修改表的记录时,在打开索引文件的前提下,索引会自动更新;若没有打开相应的索引文件,也可使用索引的更新命令,对索引文件进行更新,以保证索引文件与被索引的表文件之间数据的一致性。

2. 索引的种类。

(1)按扩展名来分类:

在 VFP 中有两类索引文件:单索引文件和复合索引文件,其索引文件的扩展名分别为 IDX 和 CDX。单索引文件只包括一个索引,而复合索引文件则可以包含多个索引,并且复合索引文件中的每个索引都有一个相对应的索引标记,表示一种逻辑顺序;复合索引文件总是以压缩方式存储,而单索引文件在产生时,只有在建立索引的命令中选用压缩参数 Compact 时,才能以压缩方式建立单索引文件,以压缩方式存储可以减少文件所占的存储空间。

对于复合索引文件而言,又可将其分为结构复合索引文件和非结构复合索引文件两种。在定义复合索引文件的过程中,用户所取的复合索引文件名与表文件的主名不不同时,则这种复合索引文件为非结构复合索引文件,否则为结构复合索引文件。

在打开非结构复合索引和单索引文件时,必须要使用专门的打开索引文件命令;非结构复合索引文件和单索引文件都可以在表没有关闭的前提下,单独关闭,也可随同表文件一起关闭。然而,结构复合索引文件却只能随同表文件一起打开和关闭,当向表文件中添加、更改或删除记录时,由于结构复合索引文件处于打开状态,所以它还会自动维护,以保持索引与数据的一致性。

(2)按功能来分类:

索引不仅能建立记录的逻辑顺序,还能控制是否允许相同的索引关键字值在不同的记录中重复出现,对于候选索引和主索引还可以用于在永久关系中建立参照完整性。

VFP 对复合索引文件中的索引提供了四种类型选择:主索引、候选索引、唯一索引和普通索引。其中主索引类型只能适用于数据库表,并且每个数据库表只能有一个主索引;

其他三类索引,既适用于数据库表,也适用于自由表,并且每个表中都可以建立多个同类型的索引。表 3.1 列出 4 种索引功能类型。

表 3.1 索引功能分类表

索引类型	关键表达式重复值	说明	创建修改命令	索引个数
普通索引	允许	可作为一对多永久关系中的多方	Index	允许多个
唯一索引	允许,但输出无重复值	为与以前版本兼容而设置		
候选索引	不允许,输入重复值将禁止存盘	可用做主关键字,可用于在永久关系中建立参照完整性	Index Create Table Alter Table	允许多个
主索引		仅适用数据库表,可用于在永久关系中建立参照完整性	Create Table Alter Table	只能 1 个

主索引不允许索引关键字的值在记录中重复出现,从而确保输入的记录中索引关键字值的唯一性。主要用于主表或被引用的表,用来在数据库的永久关系中建立参照完整性。一个表只能创建一个主索引。

候选索引与主索引具有相同的特性。在数据库表和自由表中均可以为每个表建立多个候选索引。

唯一索引允许索引关键字的值在记录中出现重复,但对于关键字值相同的记录,索引中只列入其中的第一个记录。

普通索引不仅允许索引关键字的值在记录中出现重复,而且也允许索引项中出现重复值。在一个表中可以有多个普通索引。

为了概念的完整性,前面涉及到数据库表、永久关系、参照完整性等概念,这些概念将在本章数据库与视图一节中作详细介绍。

3.1.2.2 建立索引

无论是建立索引,还是建立排序,都必须首先打开要建立索引或排序的表文件。

一、界面方式

在界面方式下创建的索引标记均为结构复合索引标记。

方式一:

例 3.2 Employee 表已经在当前工作区中打开,请建立一个索引标记为“身份号”,索引关键字为“身份号”的普通索引。

(1) 单击“显示/表设计器”菜单,弹出“表设计器”对话框,如图 3.1;

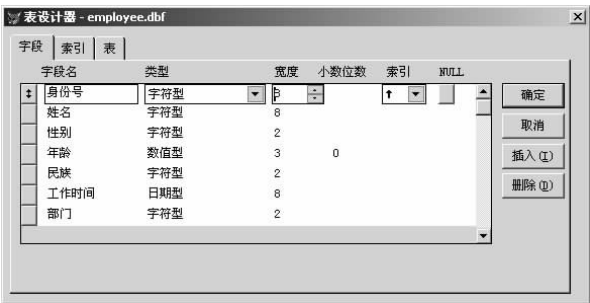


图 3.1 “表设计器”对话框

- (2) 单击对话框中“字段”选项卡,选中要建立索引的字段,如“身份号”,在“索引”列表框中选择升序或降序,如升序,如图 3.1;
- (3) 单击对话框中“索引”选项卡,在“类型”列表框中选择索引类型,如“普通索引”,如图 3.2 所示:



图 3.2 表设计器索引选项卡对话框

- (4) 重复 2,3 步骤,可以继续建立其他的索引,但用这种方式建立的索引关键字一般为单个字段。
- 方式二:

例 3.3 再建立一个记录以性别排序,当性别相同时,则以工资排序的普通索引。

- (1) 在表设计器中单击“索引”选项卡,直接在索引名栏中输入新的索引名,如“xbgz”,选择索引类型为“普通索引”,如图 3.2 所示;
- (2) 单击表达式列的按钮,弹出“表达式生成器”对话框,在该对话框中的“表达式”栏中输入:性别+Str(工资,8,2),如图 3.3 所示;



图 3.3 表达式生成器对话框

说明：

- ① 由于身份证号字段是字符型，而工资字段为数值型，所以在上述表达式中的工资字段使用了数字字符转换函数 Str()，将数值型的工资转换为字符型；
 - ② 用户既可以直接向“表达式”栏输入表达式，也可以借助“表达式生成器”中的函数、数学、逻辑、日期、字段等列表框选择所需的函数、运算符或字段，形成表达式；
 - ③ 完成表达式的输入后，单击“确定”按钮，返回“表设计器索引选项卡”对话框。
- (3) 单击“确定”按钮，则保存已创建的索引。

二、命令方式

在用命令方式建立的索引文件和索引标记时，最新建立的索引文件和索引标记，分别为主控索引文件和主控索引标记。

1. 单索引文件的建立。

格式：

Index On <索引关键字> To <索引文件名> [Unique] For <条件>[Additive]

功能：对当前表中满足条件的记录，按<索引表达式>的值建立一个单索引文件，并打开此索引文件，其缺省的文件扩展名为.idx。

参数说明：

- (1) <索引关键字>：用以指定记录重新排序的字段或表达式。<索引关键字>可以是字段名，也可以是含有当前表中字段的合法表达式。表达式值的数据类型可以是字符型、数值型、日期型、逻辑型。若在表达式中包含有几种类型的字段名时，必须使用类型转换函数将其转换为相同类型的数据；
- (2) [Unique]：指定 unique 子句时，若存在多条记录的<索引关键字>值相同时，则仅仅只将第一次遇到的记录加入到索引文件中进行排序；省略该子句时，则把所有遇到的记录值都加入到索引文件中；
- (3) [Additive]：若省略 additive 子句，当建立新的索引时，除结构复合索引文件外，所有其他打开的索引文件都将会被关闭；若选择此选择项，则已打开的索引文件仍然保持打开状态；
- (4) For <条件>：指定一个条件，只显示和访问满足这个条件的表达式<条件>的记录，索引文件只为那些满足条件表达式的记录创建索引关键字。

例 3.4 建立部门是升序排列的普通索引型单索引文件，并显示索引后的结果。

Use Employee

Index On 部门 To Bmsy

List

记录号	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
1	20050001	张丽平	女	30	汉	07/12/95	01	.T.	1356.25	14856359778	Memo	Gen
8	20050008	杨剑雄	男	21	汉	08/23/03	01	.F.	985.12	14567665685	memo	gen
2	20050002	陈国庆	男	40	汉	03/22/85	11	.T.	1850.60	14978589657	memo	Gen
4	20050004	王国真	女	22	汉	09/23/02	11	.F.	988.57	14896589875	memo	gen
3	20050003	方世玉	男	25	汉	08/05/00	12	.F.	1042.38	14654896589	memo	gen
6	20050006	孙宏伟	男	50	回	08/06/75	12	.T.	2056.87	14656562328	memo	gen
5	20050005	关鹏	男	27	满	07/12/96	13	.T.	1125.12	14568796225	memo	gen
7	20050007	李莉	女	40	汉	05/12/86	13	.T.	1768.95	14563256772	memo	gen

2. 复合索引文件的建立。

复合索引文件中可以包含多个索引标记,每个索引标记都有标记名,一个索引标记相当于一个单索引文件。

格式:

Index On <索引关键字> Tag <标记名> [Of <复合索引文件名>][For <条件>];
[Ascending | Descending] [Unique | Candidate] [Additive]

功能:建立复合索引文件标记,并打开此索引文件,其缺省的文件扩展名为.cdx。

参数说明:

(1) <索引关键字>、[For <条件>]、[Additive]:与上相同;

(2) Tag <标记名> [Of <复合索引文件名>]:创建一个复合索引标记。若不包含可选的[Of <复合索引文件名>]子句或复合索引文件名与表文件主名相同时,则创建结构复合索引文件的索引标识。如果在 Tag <标记名>参数后包含可选项[Of <复合索引文件名>]子句,且复合索引文件名与表文件主名不相同,则打开或创建非结构复合索引文件,并创建非结构索引文件的索引标识;

(3) [Ascending | Descending]:Ascending 指定复合索引文件为升序,这是默认值。Descending 指定复合索引文件为降序;

(4) [Unique | Candidate]:缺省该项时,默认建立普通索引文件;选择 Unique 参数表示建立唯一索引型索引;选择 Candidate 参数表示建立候选型索引;

(5) 索引标记名的命名规则与变量名的命名规则相同,但同一复合索引文件中,不允许有相同的索引标记名;

(6) 单索引文件只能按<索引关键字>的值的升序排列,而复合索引文件既可以按升序排列也可以按降序排列;

(7) 表的显示和访问顺序只由一个索引文件(主控索引文件)和标识(主控标记)控制。但是在修改表时,所有已打开的索引文件都将被自动更新。

例 3.5 为 Employee 建立一个结构复合索引文件,其中包括 3 个索引:

(1) 记录以工资降序排列,索引标记为普通索引;

(2) 记录以部门升序排列,部门相同时则按工资升序排列,索引标记为普通索引;

(3) 记录以部门升序排列,部门相同时则按工资降序排列,索引标记为候选索引。

Use Employee

List && 以记录号为序显示记录

Index On 工资 Tag Gz Desc && 建立以工资降序排列的结构索引标记 Gz

Clear

List && 以工资的降序排列显示记录

Index On 部门+Str(工资) Tag Bmgz Of BmInd

&& 在非结构复合索引 BmInd 中建立部门升

&& 序、工资升序排列的索引标记 Bmgz

Clear

List && 以部门升序、工资升序排列显示记录

Index On 部门+Str(10000-工资) Tag Bmgz1 Candidate Additive

&& 请大家想想该命令的功能,检查一下以前建立

&& 的索引文件有哪些关闭了,哪些没关闭

List

&& 以部门升序、工资降序排列显示记录

3.1.2.3 使用索引

一、打开索引文件

1. 在打开表的同时打开索引文件。

打开或关闭表时,结构复合索引文件自动随表被打开或关闭,而不能使用索引文件打开命令打开或关闭。

格式:

Use <表文件名> Index <索引文件名表> [Order <数值表达式> | <单索引文件名> |
[Tag] <索引标记名> [Of <复合索引文件名>] [Ascending| Descending]]

功能:打开指定的表,并且打开由<索引文件名表>指定的所有索引文件,同时指定主控索引文件和主控索引标记。

参数说明:

(1) <索引文件名表>:指定要打开的一个或多个索引文件。索引文件列表中可以包含多个单索引和复合索引文件。索引文件列表中,第一个索引文件成为主控索引文件;

(2) 当第一个索引文件为单索引文件,并且又省略 Order 选项时,单索引文件也同时为主控索引标记;

(3) [Order <数值表达式>]:指定一个索引文件或标识为主控索引,数值表达式指定在索引文件列表中出现的索引标记的顺序位置编号。系统给打开的单索引文件和复合索引文件的索引标记的顺序位置编号是:首先按照打开索引文件时的单索引文件名的排列顺序编号,再按照结构复合索引文件中索引标记建立的顺序编号,最后按照非结构复合索引文件中的索引标记建立的顺序编号。如果数值表达式为 0,表中记录以物理顺序显示和访问,而索引文件保持打开状态;

(4) [Order <单索引文件名>]:指定一个单索引文件为主控索引;

(5) [Order [Tag] <索引标记名> [Of <复合索引文件名>]]:指定复合索引文件中的一个标记作为主控标记。标记名来自结构化复合索引文件或非结构化复合索引文件。在打开的非结构化复合索引文件中,如果存在与其他索引文件的索引标记相同名称的索引标记,则要使用[Of <复合索引文件名>]指定标记所在的复合索引文件;

(6) [Ascending/ Descending]:指定显示和访问表记录时,是以升序还是以降序进行;该参数只有在选用 Order 时才有效。

例 3.6

Use Employee

Index On 身份号 To Sfh

&& 按身份号建立一个单索引文件

List

&& 以身份号为升序显示记录

Index On 姓名 Tag Xm Of Xx

&& 建立或打开名为 xx 的复合索引文件,并在该复合索引文件中,按姓名建立一个索引的标记 Xm,关闭以前打

	&&. 开的除结构索引文件以外的其他索引文件。
List	&&. 以姓名为升序显示记录
Index On 性别 Tag Xb Of Xx	&&. 在复合索引文件 xx 中按性别建立一个索引标记 xb
List	&&. 以性别为升序显示记录
Use	&&. 关闭当前工作区的表文件,所有与该表文件有关的、 &&. 已经打开的索引文件随之关闭
Use Employee Index Xx,Sfh Order Sfh	&&. 打开表,并同时打开非结构复合索引文件 xx &&. 和单索引文件 Sfh,设置单索引文件为主索引标记
List	&&. 以主索引标记 Sfh 的索引顺序显示所有记录
Use	
Use Employee Index Xx,Sfh Order Xb	&&. 打开表,并同时打开非结构复合索引文件 xx &&. 和单索引文件 Sfh,Xx 的索引标记 xb 为主索引标记
List	&&. 以主索引标记 xb 的索引顺序显示所有记录
Use	
Use Aa Index Xx,Sfh Order 1	&&. 打开表,同时打开非结构复合索引文件 xx 和单索引文 &&. 件,并将表设计器中“索引”标签中排在第一行的索 &&. 引标记作为主索引标记,本例为 Sfh
List	&&. 以主索引标记 Sfh 的索引顺序显示所有记录
Use	
Use Aa Index Xx,Sfh Order 2	&&. 打开表,并同时打开非结构复合索引文件 xx &&. 和单索引文件,并将表设计器中“索引”标签中排在第 &&. 二行的索引标记作为主索引标记,本例为 xm
List	&&. 以主索引标记 xm 的索引顺序显示所有记录

2. 在打开表后打开索引文件。

格式:

```
Set Index To [<索引文件名表>| ?][Order <索引号>|;  
    <单索引文件名>|[Tag] <索引标记> [Of <复合索引文件名>]];  
    [Ascending | Descending] [Additive]
```

功能:打开指定的索引文件或关闭索引文件。省略所有选项为关闭当前工作区中除结构复合索引文件外的所有索引文件。

参数说明:

- (1) ?:弹出打开对话框,从中可以选择并打开一个索引文件;
- (2) 其他参数的用法同 USE 和 Index 命令中参数的用法;
- (3) 在有索引文件的表中,记录的显示顺序和访问顺序可以由主控索引标记决定;
- (4) 使用 Set Index 命令可以打开单索引文件和复合索引文件;
- (5) 如果一个表有结构复合索引文件,打开表时该文件随表自动打开;
- (6) 执行不带参数的 Set Index To 命令,会关闭当前工作区中所有打开的索引文件(结构复合索引文件除外)。

例 3.7

Use Employee Index Bmsy, BmInd &&. 打开表的同时分别打开在例 3.4 和 3.5 中建立的单
 &&. 索引文件 Bmsy 和非结构复合索引文件 BmInd, 单索引文件为主索引标记
Set Index To Xx, Sfh Order Xb &&. 打开复合索引文件 Xx 和单索引文件 Sfh, 并同时关闭以
 &&. 前打开的除结构索引文件以外的其他索引文件, 设置索引标记 Xb 为主索引标记
List

二、设置主控索引

复合索引文件的索引标记会在建立时自动成为主控索引, 但如果在打开索引文件时未指定主控索引, 打开索引文件之后需要指定主控索引, 或者希望改变主控索引, 可使用下面的命令:

格式:

Set Order To [<数值表达式>|<单索引文件名>|[Tag] <索引标记> [Of <复合索引;
 文件名>] [Ascending|Descending]

功能: 在打开的索引文件中指定主控索引标记。

例 3.8 根据例 3.4、3.5 建立的索引改变主控索引。

Use Employee Index Bmsy, BmInd

List

Set Order To Tag Bmgz

List

Set Order To 1

List

Set Order To

List

Set Order To 3

List

Set Index To

List

说明:

(1) 如果命令中不使用任选项, 只写 Set Order To 或当数值表达式的值为 0 时, 那么不会指定主索引, 记录仍按照记录号的顺序显示, 但并未关闭索引文件;

(2) Vfp 允许在 Set Order 命令中使用 Ascending 和 Descending 暂时转换主控索引的顺序, 但它不同于 Index On 命令中的 Ascending 和 Descending;

(3) 指定的主控索引标记必须存在于已经打开的索引文件中。

三、删除索引

1. 删除索引文件。

若用删除文件命令来删除索引文件, 必须遵循先关闭后删除的原则, 这与删除表类似。

2. 删除索引标记。

命令格式:

Delete Tag All|<索引标记 1>[,<索引标记 2>]...

功能:删除打开的复合索引文件的索引标记。

说明:All 子句用于删除复合索引文件的所有索引标记。若某索引文件的所有索引标记都被删除,则该索引文件会自动被删除。

例 3.9 删除例 3.5 建立的结构复合索引文件中的索引标记 Gz

Use employee

Delete tag Gz

四、索引文件的更新

1. 自动更新。

当表中的数据发生变化时(例如对它进行插入、删除、添加或更新操作之后),所有当时已打开的索引文件都会随数据的改变自动改变记录的逻辑顺序,实现索引文件的自动更新。

2. 索引文件更新命令。

修改表的记录时,若索引文件没有打开,那么没有打开的索引文件就不会自动更新。如果要维持没有打开的索引文件中记录的逻辑顺序,可将索引文件打开,并用 Reindex 命令重新索引:

格式:Reindex[Compact]

功能:重建当前打开的所有索引文件。

说明:

使用 Reindex 命令之前,必须首先打开需要重建索引的全部索引文件。

3.2 数据的查询

数据查询是 VFP 中一个非常重要的应用,用户经常会使用查询来查找需要的数据,即按照用户指定的条件在表中查找所需的记录。通常的查询方法有顺序查询和索引查询两种。在 VFP 中还可直接使用 SQL 的查询命令,即 Select-SQL 命令进行查询。

3.2.1 顺序查询

顺序查询也叫直接查询,是按照记录顺序逐个比较,逐个查询,它包括 Locate 和 Continue 两条命令。

1. Locate 命令。

格式:Locate [<范围>] [For <条件>] [While <条件>]

功能:按表中记录的顺序查找满足指定逻辑表达式的第一个记录,并将记录指针指向该记录。

参数说明:

(1) [<范围>]:指定要查找的记录范围。只有在范围内的记录才被找到定位。缺省范围时,默认是 All;

(2) 其他参数的用法同前。

2. Continue 命令。

格式:Continue

功能:它与 Locate 命令配合使用,按 Locate 命令格式要求,从当前记录位置开始继续查找下一条满足条件的记录。

例 3.10 在 Employee 中查询工资>1600 元的人的信息。

Use Employee

Locate For 工资>1600 &&. 在状态栏显示查到的记录号
? Found() &&. 查找测试函数 Found()返回真值
Display &&. 显示结果如下

记录号	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
6	20050006	陈国庆	男	40	汉	03/22/85	11	.T.	1850.60	14978589657	memo	gen

Continue &&. 在状态栏显示查到的记录号
? Found() &&. 查找测试函数 Found()返回真值
Disp

记录号	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
6	20050006	孙宏伟	男	50	回	08/06/75	12	.T.	2056.87	14658582328	memo	gen

Continue &&. 在状态栏显示查到的记录号
? Found() &&. 查找测试函数 Found()返回真值
Disp

记录号	身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资	电话	简历	照片
7	20050007	李莉	女	40	汉	05/12/86	13	.T.	1768.95	14563258772	memo	gen

Continue &&. 在状态栏中显示“已到定位范围尾部”
? Found() &&. 查找测试函数 Found()返回假值

由于该例的查找范围是 All,所以还可以用测试函数 Eof()测试,如:

? Eof() &&. 该函数结果为真值,表示记录指针指向表的尾部,同时
 &&. 状态栏显示:记录 EOF/8,表示已到定位范围末尾,查找结束

如果该例的使用范围不是 All,当状态栏中显示“已到定位范围尾部”时,当前记录指针则指向定位命令查找范围的最后一条记录上。

说明:

(1) 被搜索的表可以没有索引,在没有索引时,将按记录号的顺序查找;当有索引时,则按照索引顺序查找;

(2) 若 Locate 发现一个满足条件的记录,就将记录指针定位在该记录上。此时的 Recno()函数的值为该记录的记录号,Found()函数的值为“真”,Eof()函数的值为“假”。如果没有找到,Found()函数的值为“假”,主窗口状态栏中显示“已到定位范围末尾”,Recno()和 Eof()函数值与 Locate 命令的范围设置有关;

(3) 如果执行 Set Exact Off 命令后,再用 locate 命令查找字符型数据时,不要求字符型数据精确匹配;如果执行 Set Exact On 命令后,则要求精确匹配;

(4) Locate 命令只能查找第一条满足条件的记录。若表中有多条满足条件的记录

中间的或是后面的字符；

(5) 执行 Set Exact Off 命令后,再用 Find 命令查找字符串时,字符串可以是索引表达式值的全部或是从首字符开始的一个子串。如果执行了 Set Exact On 命令后再用 Find 命令来查找字符串,则字符串只能和索引表达式的值精确匹配,即只能是索引关键字值的全部；

(6) 建立索引文件时,索引表达式可以是多个字段组成的表达式,若字段之间用“+”连接,那么用 Find 命令查询时,查询内容应当是包含空格符在内的索引表达式值的全部或是从首字符开始的一个子串,究竟是用哪一种,这取决于 Set Exact 命令的设置；若字段之间用“-”号连接,用 Find 查找时,查询内容应当不包含空格符在内的索引表达式值的全部或是从首字符开始的一个子串；

(7) Find 命令只能使记录指针定位于第一条符合条件的记录,而 Continue 命令又不能和 Find 命令一起配合使用,当符合条件的记录不止一条时,可用 Find 先查到满足条件的第一条记录,因为记录是根据索引关键字排序了的,因此符合条件的记录是在一起的,因此可用 Skip 命令配合 Disp 命令查找,直到发现某条记录不满足条件时,则该记录以后的记录就一定不符合查找的条件。

2. Seek 命令。

格式：

Seek <表达式> [Order <索引号>|<单索引文件名>| [Tag] <索引标记>;
[Of <复合索引文件名>] [Ascending| Descending]]

功能:在打开的索引文件中快速查找与<表达式>相匹配的第 1 条记录。

参数描述：

(1) <表达式>:指定 Seek 搜索的关键字。<表达式>可以是空字符串；

(2) 当省略 Order 子句时,按表的当前主控索引关键字值查找；

(3) Order <索引号>|<单索引文件名>|[Tag] <索引标记> [Of <复合索引文件名>] [Ascending| Descending]:指定<表达式>是在哪一个索引或索引标记的关键字表达式值中进行查找。其使用方法同设置主控索引。

例 3.12 Seek 命令用法示例。

Use Employee

Index On 部门 Tag Bm

Seek "12"

? Found()

&&. 显示:.. T.

? Recno()

&&. 显示:3

Index On 工作时间 Tag Gzsj

Index On 年龄 Tag Nl

Seek 40

? Found()

&&. 显示:.. T.

Seek {^1996-07-12} Order Gzsj

? Recno()

&&. 显示:5

说明：

当省略所有的可选项时,Seek 命令与 Find 命令的功能基本相同,但 Seek 命令的功能更强,Seek 命令不仅可以查找字符串和常数,它还可以查找字符型、数值型、日期型或逻辑型表达式的值。用 Seek 命令查找字符串常量时,字符串常量必须放在定界符中。

Seek 命令中的表达式必须和索引表达式的类型相同。通常我们可以借助索引关键字表达式来描述 Seek 命令表达式,即将索引关键字表达式的变量转换为我们需要查找的常量,即为 Seek 命令表达式。

3. 顺序查询和快速查询比较。

表 3.2 顺序查询和快速查询比较

	Locate	Find	Seek
查询内容	字符型、数值型、日期型、逻辑型表达式,还可以查找备注型字段	字符串常量或常数	字符型、数值型、日期型或逻辑型表达式
对表要求	无论是否建立了索引文件均可方便地查询	必须建立并打开索引文件,只能在主控索引中查询	必须建立并打开索引文件
命令特点	可使用<范围>子句限定查询范围,可与 Continue 命令配合使用,找出表中全部符合条件的记录	在整个表中查询,只能找出满足条件的第一条记录	同左
查询速度	慢	快	快

3.3 数据工作期

在数据工作期窗口中我们可以利用交互式操作方式,来设置数据工作环境,如打开的表及其索引,多个表之间的关联等。

利用数据工作期来建立环境还有以下优点：

- 1. 利用数据工作期窗口操作,显得比较直观方便;而直接用命令来建立数据环境,则显得抽象不容易理解,用命令建立数据环境时,也可借数据工作期窗口,来观察命令执行的情况。
- 2. 数据工作期的数据环境可以作为视图文件保存起来,需要时将相应的视图文件打开就能恢复保存的环境。

3.3.1 多工作区查询

在实际应用过程中,我们常常需要同时查询多个相关表的数据,本书以员工管理为例设计了 3 个表,分别是 Employee. dbf,Contribution. dbf,Department. dbf。Employee 表我们在前面已经使用过,Contribution 记录了人员每次捐款的日期和金额,Department 表达了表中部门字段的意义,例如代码 11 表示的部门名称是一车间。用代码来表示汉字的名称可使输入或修改变得简便,而且能避免称呼不一致引起的麻烦,例如“一车间”和“第一车间”在 VFP 中就不是一个等价的字符串,但其含义却都是指同一个车间。

一、工作区号与别名

VFP6 提供了多达 32767 个工作区,每个工作区都有一个工作区号,分别用 1~32767 表示,其中 1~10 号工作区还分别对应有别名 A~J。所以访问 1~10 号工作区既可用工作区号直接访问,也可用工作区的别名访问。

除了前 10 个工作区有别名外,在不同工作区打开的表也都有一个别名,通常是打开表时,若没有指定表的别名,系统则默认表的别名就是该表的主文件名。如果在打开表时,使用 Alias 参数指定了表的别名,则表就有了一个与表文件主名不同的别名。

命令:

Use <表文件名> [Alias <别名>] [In <工作区号 | 工作区别名 | 表别名>] [Again]

功能:在指定的工作区打开指定的表文件,并为该表文件起一个别名。若省略可选项时,系统将表的主文件名作为该表文件的别名,并且是在当前工作区中打开表文件。

参数描述:

1. [In <工作区号> | <工作区别名> | <表别名>]:指定要打开表的工作区。其中:工作区号、工作区别名都是直接指定工作区。但<表别名>不是直接指定工作区,而是将别名为<表别名>的表文件所在的工作区作为指定的工作区,并先将该工作区的表文件关闭,然后再打开指定的表文件。如果省略该选择项,则在当前工作区打开表。

2. [Alias <别名>]:为要打开的表指定一个别名。

3. [Again]:若要在多个工作区中打开同一个表,可以按以下方法操作:

- (1) 选择另一个工作区,并执行带有表名和 Again 子句的 Use 命令;
- (2) 执行带有表名和 Again 子句的 Use 命令,并且用 In 子句指定一个不同的工作区。

例 3.13

Use Employee

Use Employee In B again

二、工作区的选择与多表文件的打开与关闭

1. 工作区的选择。

若想改变当前工作区,则可使用 Select 命令来转换当前工作区。

格式:

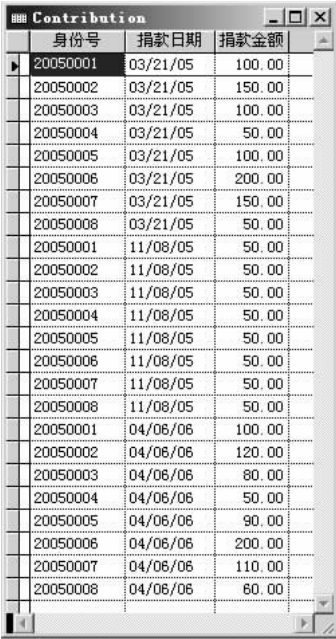
Select <工作区号> | <别名> | 0

功能:选择一个工作区作为当前工作区。



部门代码	部门名称
01	办公室
11	一车间
12	二车间
13	三车间

图 3.4 Department 表



身份号	捐款日期	捐款金额
20050001	03/21/05	100.00
20050002	03/21/05	150.00
20050003	03/21/05	100.00
20050004	03/21/05	50.00
20050005	03/21/05	100.00
20050006	03/21/05	200.00
20050007	03/21/05	150.00
20050008	03/21/05	50.00
20050001	11/08/05	50.00
20050002	11/08/05	50.00
20050003	11/08/05	50.00
20050004	11/08/05	50.00
20050005	11/08/05	50.00
20050006	11/08/05	50.00
20050007	11/08/05	50.00
20050008	11/08/05	50.00
20050001	04/06/06	100.00
20050002	04/06/06	120.00
20050003	04/06/06	80.00
20050004	04/06/06	50.00
20050005	04/06/06	90.00
20050006	04/06/06	200.00
20050007	04/06/06	110.00
20050008	04/06/06	60.00

图 3.5 Contribution 表

参数说明:

- (1) 选择一个工作区作为当前工作区,该工作区若有表打开,则该表就成为当前表;
- (2) 执行该命令时,既可以使用工作区号作为标识符,也可以用工作区的别名来作为工作区的标识符;
- (3) 若某一工作区中已经有表打开,如果要选择该工作区为当前工作区,也可用该工作区打开的表的别名;
- (4) 若选择 0,则系统自动选取当前未使用的最小工作区号作为当前的工作区。

例 3.14

Select 3	&&. 选择第三个工作区为当前工作区
Use Employee Alias Yg	&&. 在第三工作区打开 Employee 表,其别名为 Yg
Select 1	&&. 选择第一个工作区为当前工作区
Use Department	&&. 在第一工作区打开 Department 表
Use Department Alias Bm In D Again	&&. 在别名为 D 的工作区中再次打开 Department
Select Bm	&&. 选择 Bm 所在工作区为当前工作区

2. 使用 Use 命令关闭打开的表文件。

格式:Use In <工作区号>|<别名>

功能:关闭指定工作区中的表文件。

参数说明:别名可以是工作区的别名,也可以是表的别名。

在单工作区操作时,用户只能操作当前工作区中的表,而在多工作区操作时,用户除了可以操作当前工作区中的记录外,还可以操作其他工作区中的表,实现多个表文件之间的数据处理。在当前工作区调用其他工作区的数据时,非当前工作区中的表文件的字段名前要加上该表文件的<别名>和“->”符号,或者是<别名>和符号“.”。格式如下:

<别名> -><字段名>

或<别名>.<字段名>

例 3.15 通过多工作区操作由部门代码查出部门名称

Select 2	
Use Department Alias Bm	&&. 在第二工作区中打开 Department 表,并定义其别名为 Bm
Select 1	
Use Employee	
Display 身份号,姓名,部门,Bm. 部门名称	&&. 主窗口的显示结果如下

记录号	身份号	姓名	部门	Bm->部门名称
1	20050001	张丽平	01	办公室

List 身份号,姓名,部门,Bm. 部门名称	&&. 主窗口的显示结果如下
-------------------------	----------------

Use	&&. 关闭当前工作区的表
-----	---------------

Use In Bm	&&. 关闭别名为 Bm 的表所在工作区中的表
-----------	-------------------------

问题:通过以上示例可以发现,一般情况下部门代码与部门名称不对应,为什么?这个问题将在下面介绍。

记录号	身份号	姓名	部门	Bm->部门名称
1	20050001	张丽平	01	办公室
2	20050002	陈国庆	11	办公室
3	20050003	方世玉	12	办公室
4	20050004	王国真	11	办公室
5	20050005	关鹏	13	办公室
6	20050006	孙宏伟	12	办公室
7	20050007	李莉	13	办公室
8	20050008	杨剑雄	01	办公室

3.3.2 数据工作期

一、数据工作期窗口的打开与关闭

1. 界面方式。

单击菜单“窗口/数据工作期”或单击工具栏上“数据工作期”按钮,弹出数据工作期窗口,参见图 3.7 所示。

2. 命令方式。

格式一:

Set

或 Set View On

功能:打开数据工作期窗口。

格式二:

Set View Off

功能:关闭数据工作期窗口。

二、数据工作期窗口的操作

数据工作期窗口主要由三个部分组成。左边的别名列表框用于显示已打开的表,并可从多个表中选定一个表为当前表。右边的关系列表框用于显示表之间的关联状况。中间一列的 6 个功能按钮的功能如下:

1. 属性按钮:用于打开工作区的属性对话框,其作用与菜单“表/属性”相同,如图3.6 所示。

在工作区属性对话框中,选定“修改”按钮会弹出表设计器,可用于修改当前表的结构、建立或修改索引;在“索引顺序”组合框中可指定和更改主控索引;通过“字段筛选”按钮与“数据过滤器”文本框还可设置字段表与过滤器。当选定允许缓冲复选框后,还可设置多用户操作时记录的锁定方式和缓冲方式。

在多用户环境中,修改记录前进行记录锁定,能防止因其他用户同时访问而发生冲突。但编辑时锁定与写入时锁定比较,显然前者降低了系统运行速度。“当前记录”表示仅缓冲当前记录,而“所有编辑过的记录”指缓冲所有编辑过的记录,显然前者能使系统运行较快。

2. 浏览按钮:为当前表打开浏览窗口,供浏览或编辑数据。

3. 打开按钮:弹出“打开”对话框来打开表,若某数据库已打开,还可打开数据库表。

4. 关闭按钮:关闭当前表。

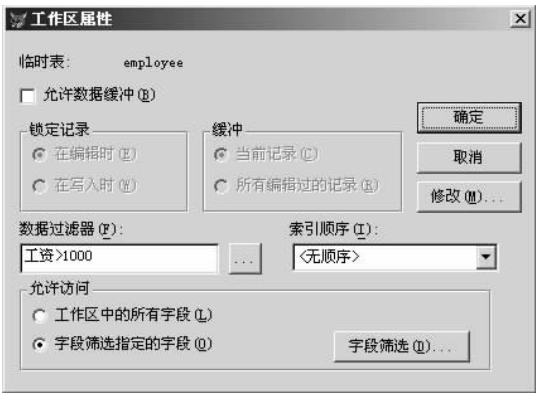


图 3.6 工作区属性对话框

5. 关系按钮:以当前工作区的表为父表,与其他工作区中的表建立关联。

6. 一对多按钮:系统默认表之间以多一关系关联,通过单击这个按钮,可以建立一对多关系,与 Set Skip To 命令等效。

例 3.16 数据工作期窗口操作示例,要求:

(1) 同时打开 Employee 和 Department;

(2) 为 Employee 设置包括身份号,姓名,部门,工资的字段表,并以工资大于 1000 为条件设置过滤器,并打开浏览窗口。

操作步骤:

(1) 打开数据工作期:选定菜单“窗口/数据工作期”命令,屏幕出现数据工作期窗口,如图 3.7 所示;



图 3.7 数据工作期窗口与浏览窗口

(2) 打开表:单击“打开”按钮,弹出“打开”对话框,在该对话框中选定表 Employee,单击“确定”按钮返回数据工作期窗口。以同样的方法打开 Department;

(3) 设置字段表:在别名列表框中选定表 Employee 使之成为当前表,单击“属性”按钮,弹出“工作区属性”对话框,如图 3.6 所示;在“工作区属性”对话框中单击“字段筛选”按钮,弹出“字段筛选器”对话框,如图 3.8 所示;在“字段筛选器”对话框中将身份号,姓名,部门,工资字段移到选定字段列表框中,单击“确定”按钮,返回工作区属性对话框,单击“字段筛选指定的字段”选项按钮,完成字段表的设置;

(4) 设置过滤器:在“数据过滤器”文本框中键入条件“工资>1000”,单击“确定”按钮返回数据工作期;

(5) 为 Employee 打开浏览窗口:单击“浏览”按钮。结果如图 3.7 所示。

3.3.3 视图文件

对于数据工作期设置的环境我们将其作为视图文件保存,并通过打开指定的视图文件来恢复其所保存的数据环境。

一、视图文件的建立



图 3.8 字段选择器对话框

1. 界面方式。

在数据工作期窗口为当前窗口时,可选定菜单“文件/另存为”来建立视图文件,系统默认视图文件的扩展名为.vue。

当数据工作期窗口不是当前窗口,则菜单“文件/另存为”以淡灰色显示而无法选用。

2. 命令操作。

格式:

Create View <视图文件名>

功能:为 VFP 的当前数据工作环境建立一个视图文件,该方式与数据工作期窗口是否打开无关。

二、视图文件的打开

打开视图文件意味着恢复环境,相当于重新执行一系列先前的设置命令。

1. 界面方式。

单击菜单“文件/打开”或“打开”按钮,弹出“打开”对话框,在对话框的文件类型中选定“视图(*.VUE)”,选定要恢复的视图文件,单击“确定”按钮。

2. 命令方式。

格式:

Set View To <视图文件名>

功能:打开指定名为<视图文件名>的视图文件。

例 3.17

Clear All

Use Employee In 1

Index On 工资 Tag Gz

Use Department In 2

Index On 部门名称 To Bmmc

Create View Employeevi

Clear All

Set View To Employeevi

&& 建立视图文件 Employeevi.vue

&& 关闭刚才在各工作区打开的表

&& 将刚才各工作区的状态恢复

3.3.4 表的关联

3.3.4.1 关联的概念

在例 3.15 中我们可以发现不同的部门代码,总是对应一个部门名称,这显然不合理,那么这是由什么原因引起的呢?我们知道:每个打开的表都有一个记录指针,用以指示当前记录。如果在多个工作区同时打开多个表文件,在当前工作区中移动表的记录指针时,通常情况下,其他表的记录指针是不会随之移动。如果想让其他表的记录指针也随之移动,则要建立表之间的关联。

所谓关联就是让两个或两个以上不同工作区的表之间建立某种联接,使其他表的记录指针随主表的记录指针同步移动。用来建立关联的表称为父表,被关联的表称为子表。

1. 关联条件。

建立关联的两个表之间,总有一个是主动的,我们称为父表,一个表是从动的,我们称为子表。执行涉及这两个表数据的命令时,子表记录指针会随父表记录指针的移动自动移到满足关联条件的记录上。

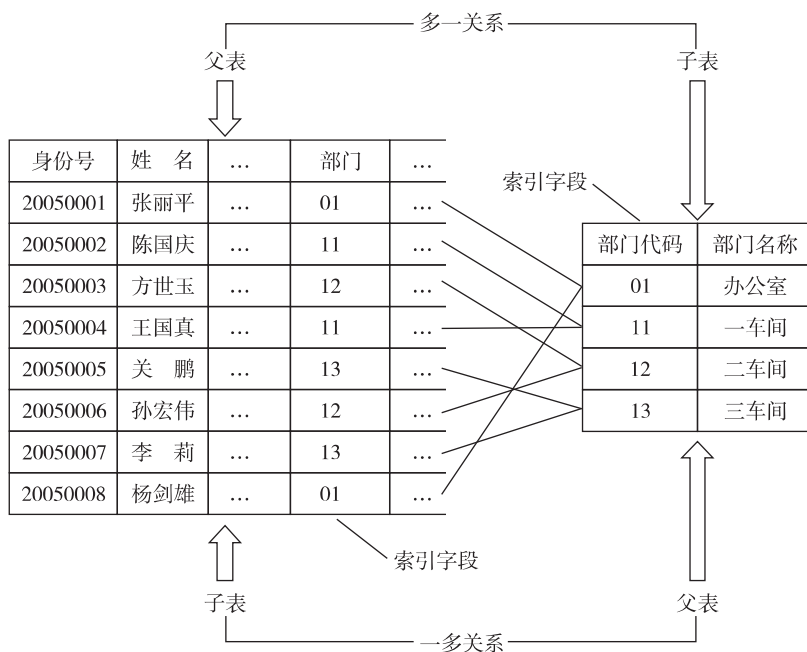


图 3.9 关联及其多一关系与一多关系

关联条件是比较父表的一个字段表达式值是否与子表的一个字段表达式值相等,但在实际建立关联时,通常是将父表的一个字段表达式的值与子表的相应的索引关键字进行比较,所以对于子表而言,必须先建立相应的索引。图 3.9 表示了为表 Employee 和 Department 建立关联,条件是“Employee. 部门=Department. 部门代码”两个字段的值相等,图中符合条件的记录画出了连线,表示子表记录指针会随父表记录指针的移动而移动。

2. 多一关系。

在对表进行关联时,若出现父表有多条记录对应子表中一条记录的情况,便称这种关

联为多一关系。在图 3.9 所示,若将 Employee.dbf 作为父表来建立关联,父表的部门字段值有多于两个的 01 与 Department.dbf 部门代码字段值仅有的一个 01 对应,这表示两个表是按照多一关系来关联的。

3. 一多关系。

在对表进行关联时,若出现父表的一条记录对应子表中多条记录的情况,我们将这种关联称为一多关系。在图 3.9 中若将 Department.dbf 作为父表进行关联,父表代码字段中仅有一个 13,而 Employee.dbf 部门字段值有多于一个的 13 与之对应,这表示两个表是按照一多关系来关联的。

VFP 关联不处理“多多关系”,若出现“多多关系”则需将其中的一个表进行分解,然后以多一关系或一多关系处理。

3.3.4.2 建立关联

一、界面方式建立关联

利用数据工作期窗口建立关联的一般步骤为:

1. 在不同的工作区分别打开需建立关联的表。
2. 按关联的关键字为子表建立索引,若该索引已经存在,则将其确定为主控索引。
3. 选定父表工作区作为当前工作区,并与一个或多个子表建立关联。
4. 说明建立的关联为一多关系。缺省本步骤,默认为多一关系。

例 3.18 查询 1995 年以后开始工作的人员,要求显示人员的身份号、姓名,工作时间,部门名称。

分析:由于身份号、姓名,工作时间,部门名称分别属于表 Employee 和表 Department 的字段,要求记录信息显示正确,则必须将表 Employee 和 Department 进行关联。关联时将 Employee 的“部门”字段值与 Department 的“部门代码”字段值进行比较。因此,以哪个表作为主表,其关联形式将也不同。

解一:以 Employee.dbf 为父表,Department 为子表建立多一关联(如图 3.12 所示):

1. 打开表:打开数据工作期窗口,然后打开 Employee.dbf 和 Department.dbf。
2. 为子表 Department 的部门代码字段建立索引。

(1) 在别名列表框中选定子表 Department,并单击“属性”按钮,弹出如图 3.6 所示的“工作区属性”对话框;

(2) 在弹出的“工作区属性”对话框中单击“修改”按钮,弹出“表设计器”窗口;

(3) 在“表设计器”窗口中,为部门代码字段建立索引标记后,单击“确定”按钮返回数据工作期窗口。

3. 以表 Employee 为父表建立关联。

(1) 在数据工作期窗口的别名列表框中选定父表 Employee 后,单击“关系”按钮;

(2) 在别名列表框中选定子表 Department,系统弹出“设置索引顺序”对话框,如图 3.10 所示;

(3) 在“设置索引顺序”对话框中选定子表 Department 的部门代码的索引后,单击“确定”按钮,弹出“表达式生成器”对话框,如图 3.11 所示;



图 3.10 设置索引顺序对话框



图 3.11 表达式生成器对话框

(4) 在“表达式生成器”对话框中的字段列表框中双击父表 Employee 的“部门”字段，单击“确定”按钮，完成多一关系的建立。完成后的数据工作期显示如图 3.12 所示。



图 3.12 建立多一关系后的数据工作期窗口

身份号	姓名	工作时间	部门名称
20050001	张丽平	07/12/95	办公室
20050003	方世玉	08/05/00	二车间
20050004	王国真	09/23/02	一车间
20050005	关鹏	07/12/96	三车间
20050008	杨剑雄	08/23/03	办公室

图 3.13 多一关系显示部门名称

其中“设置索引顺序”对话框，其作用是选定子表的一个相关索引作为主控索引，如果在建立关联前已经设置了相关的主控索引，该对话框就不会弹出了。

4. 显示结果。

要在浏览窗口显示所指定的 4 个字段，也可用如下命令显示结果，执行下列命令的显示结果如图 3.13 所示：

Browse Fields Employee. 身份号, Employee. 姓名, Employee. 工作时间, ;

Department. 部门名称 For 工作时间 >= {^1995/01/01}

5. 建立视图文件。

选定文件菜单的“另存为”命令→在另存为对话框的文本框中输入视图文件名 Emde →选定“保存”按钮，即产生视图文件 Emde. vue，建立视图文件的目的是为了保存已经建立的数据环境，如两个表的关联等。

解二：以 Department. dbf 父表, Employee. dbf 为子表建立一多关系：

(1) 打开表：在数据工作期窗口分别打开 Employee. dbf 和 Department. dbf；

(2)为子表 Employee. dbf 的部门字段建立索引,方法与前例类似;

(3)以 Department. dbf 为父表建立关联。

①在数据工作期窗口的别名列表框中选定父表 Department. dbf 后,单击“关系”按钮;

②在别名列表框中选定子表 Employee,系统弹出设置“索引顺序”对话框;

③在“设置索引顺序”对话框中选定对子表 Employee 的部门的索引后,单击“确定”按钮,弹出“表达式生成器”对话框;

④在“表达式生成器”对话框中的字段列表框中双击父表 Department 的“部门代码”字段,选定“确定”按钮,完成多一关系的建立,由于父表 Department 与子表 Employee 之前的关系应该是一多关系,因此还必须设置一多关系。

(4)设置一多关系:在数据工作期窗口的别名栏中选定父表 Department 的前提下,单击“一对多”按钮,系统弹出如图 3. 14 所示的“创建一对多关系”对话框,在该对话框中将子表从“子表别名”列表框移入“选定别名”列表框中,单击“确定”按钮,完成一多关系的设置,完成结果如图 3. 15 所示,该图在关系列表框中显示形式与图 3. 12 有所不同,父表与子表的连线,在子表一端用双线表示;



图 3. 14 创建一对多关系对话框



图 3. 15 建立一多关系后的数据工作期窗口

(5)显示结果:往命令窗口输入如下命令。

Browse Fields Employee. 身份号,Employee. 姓名,Employee. 工作时间,;

Department. 部门名称 For 工作时间>={^1995/01/01}

命令执行结果如图 3. 16 所示,该图与图 3. 13 略有不同,部门名称为办公室的有两个记录,仅第一个记录显示部门名称,体现了一多关系浏览窗口的特点。

Department			
身份号	姓名	工作时间	部门名称
20050001	张丽平	07/12/95	办公室
20050008	杨剑雄	08/23/03	*****
20050004	王国真	09/23/02	一车间
20050003	方世玉	08/05/00	二车间
20050005	关鹏	07/12/96	三车间

图 3. 16 一多关系显示部门名称

二、命令方式建立关联

1. 关联命令。

格式：

```
Set Relation To [<关联表达式 1> Into <工作区>|<别名> ;
    [, <关联表达式 2> Into <工作区>|<别名>... ;
    <关联表达式 n> Into <工作区>|<别名 n>]] [Additive]
```

功能：将当前表作为父表，建立父表与一个或多个表之间的关联。

参数说明：

(1) Into <工作区>|<别名>：指定子表的工作区或别名，也可以是子表的别名；

(2) Additive：建立关联时，如果命令中不使用 Additive 子句，则父表以前建立的关联将自动解除；若使用了 Additive 子句，则父表以前建立的关联仍然保留；

(3) <关联表达式>：指定父表中用来与子表建立关联的关联表达式。该关联表达式通常应该借鉴子表主控索引的索引关键字来建立。<关联表达式>可以是下列三种参数之一：

①<关联表达式>是返回当前记录号函数 Recno()：

在这种情况下，子表可以不设置索引，两个或多个关联表之间的联系是根据记录号来进行关联的，让父与子表之间当前记录号保持相等。如果父表记录的记录号大于子表的记录总数，则子表的当前记录指针指向记录尾部，EOF()函数对子表的返回值为 .T.。

例 3.19

```
Select 2
```

```
Use Department Alias Bm
```

```
Select 1
```

```
Use Employee
```

```
Set Relation To Recno() Into Bm
```

```
Browse Fields Employee. 身份号, Employee. 姓名, Employee. 工作时间, ;
```

```
Employee. 部门, Department. 部门名称 &&. 显示结果如图 3.17 所示
```



身份号	姓名	工作时间	部门	部门名称	
20050001	张丽平	07/12/95	01	办公室	
20050002	陈国庆	03/22/85	11	一车间	
20050003	方世玉	08/05/00	12	二车间	
20050004	王国真	09/23/02	11	三车间	
20050005	关鹏	07/12/96	13		
20050006	孙宏伟	08/06/75	12		
20050007	李莉	05/12/88	13		
20050008	杨剑雄	08/23/03	01		

图 3.17 以记录号关联的结果

②<关联表达式>是数值型表达式：

在这种情况下，子表可以不设置索引，表达式中通常含有 RECNO()函数，每当父表的记录指针重新定位时，子表的记录指针将重新定位于<关联表达式>的值所对应的记

录之上。如果<关联表达式>的值大于子表文件的记录总数,则子表文件的当前记录指针指向记录尾部,Eof()函数对子表的返回值为.T.。

```
例 3.20
Select 2
Use Department Alias Bm
Select 1
Use Employee
Set Relation To 3 * Recno() - 2 Into Bm
Browse Fields Employee. 身份号, Employee. 姓名, Employee. 工作时间, ;
Employee. 部门, Department. 部门名称 && 显示结果如图 3.18 所示
```

Employee					
	身份号	姓名	工作时间	部门	部门名称
	20050001	张丽平	07/12/95	01	办公室
	20050002	陈国庆	03/22/85	11	三车间
	20050003	方世玉	08/05/00	12	
	20050004	王国真	09/23/02	11	
	20050005	关鹏	07/12/96	13	
	20050006	孙宏伟	08/06/75	12	
	20050007	李莉	05/12/86	13	
	20050008	杨剑雄	08/23/03	01	

图 3.18 以数据表达式关联的结果

请比较图 3.17 与图 3.18 中的结果不同之处,并解释。

③<关联表达式>选择父表中与子表相关的字段建立关联:
要求子表文件必须按需要关联的字段建立索引,并设置为主控索引。

例 3.21 例 3.18 还可以用命令方式建立关联,并显示参加工作的职员的身份号、姓名、工作时间、部门名称。

```
Use Department Alias Bm In A
Index On 部门代码 Tag 部门代码 Additive
Use Employee In B && 在第二工作区打开表
Select B && 选择第二工作区为当前工作区
Set Relation To 部门 Into Bm Additive
Browse Fields Employee. 身份号, Employee. 姓名, Employee. 工作时间, ;
Department. 部门名称 For 工作时间 >= {^1995-01-01}
&& 显示结果如图 3.13 所示
```

通过以上例子,我们可以看出:在建立关联之前,必须打开一个表(父表),而且还必须在另一个工作区内打开其他表(子表)。相关的各表通常有一个相同的字段(关联字段)。父表可以同时与多个子表建立关系,称为“一父多子”的关系。<关联表达式>可以是字符型、数值型、日期型表达式。建立父子关联之后,每当当前工作区父表的记录指针重新定位时,就检索子表,将子表的记录指针定位于<关联表达式>值与<索引表达式>值相同的第一条记录之上。

在多个工作区中建立多个表之间的关联时,表之间的关联不能构成循环。

2. 对建立的关联设置一多关系。

用 Set Relation To 命令建立的关联,为多一关系,若需将父表与子表的关系设置为一多关系,则使用 Set Skip To 命令。

格式:

Set Skip To [<工作区号>|<别名 1> [,<工作区号>|<别名 2>...]]

功能:在父表的每条记录与子表中所有对应记录之间建立或取消“一对多”的关联。

参数说明:

(1) 执行该命令时,父表所在工作区必须为当前工作区;

(2) 如果已经建立了一父多子的关联,并且父表与每个子表都要建立“一对多”的关联,那么只要在 Set Skip To 命令中分别写出子表所在的<工作区号>或者<别名>即可;

(3) 当省略所有参数时,则表示取消一多关联,恢复为多一关联。

3. 解除关联。

用 Set Relation 命令建立关联之后,当移动父表的记录指针时,子表的记录指针也相应会移动,并且会引起读/写磁盘操作,这样将降低系统的性能。因此,当某些关联不再使用,或暂时不再使用时,应及时解除关联,以提高系统的运行速度。

格式一: Set Relation To

功能:删除当前工作区的表与其他工作区的表建立的所有关联。

格式二: Set Relation Off Into <工作区号>|<别名>

功能:删除当前工作区与由<工作区号>/<别名>指定的工作区中表建立的关联。

以上两条命令必须在父表所在的工作区执行。

例 3.22 关闭当前工作区与 B 工作区建立的关联。

Select A

Use Department Alias Bm

Index On 部门代码 Tag 部门代码 Additive

Use Contribution Alias Jk in B &&. 在第二工作区打开表

Select B &&. 选择第二工作区为当前工作区

Index On 身份号 tag Sfh

Use Employee In 0 &&. 在第三工作区打开表(选择最小工作区)

Select Employee &&. 选择表别名为 Employee 的工作区为当前工作区

Set Relation To 部门 Into Bm,身份号 Into Jk Additive &&. 建立与 Bm,Zk 的关联

Set Relation Off Into B &&. 取消与 B 工作区表的关联

当关闭某些表时,系统将自动删除与它建立的关联。如果关闭的是父表文件,则删除它与子表的全部关联。若要保存已经建立的关联,可创建视图文件来保存数据环境。

3.4 数据的统计

3.4.1 求记录个数的命令

格式:

Count [<范围>] [For <条件 1>] [While <条件 2>] [To <内存变量>]

功能:统计当前表中指定范围内满足条件的记录个数,并存在<内存变量>中。

参数说明:

(1)<范围>、<条件>子句的用法和前面所有命令的用法相同;

(2)若使用<范围>、<条件>子句,可统计出指定<范围>内满足<条件>的记录个数,否则统计表中记录的总数。对 Count 命令,默认的范围是 All;

(3)若使用选项 To <内存变量>可选项,可将统计的结果送到内存变量中保存。

例 3.23 统计 Employee 表中女性员工的人数。

Set Exact Off

Use Employee

Count For 性别 = "女" To Xb

? Xb

3.4.2 求和命令

格式:

Sum [<数值型表达式表>] [<范围>] [For <条件 1>] [While <条件 2>];
[To <内存变量名表> | To Array <数组名>]

功能:对当前表中指定范围内满足条件的数值型字段或是由字段组成的数值型表达式累加求和,并把结果存放在对应的变量中。

参数说明:

(1)若使用选项<数值型表达式表>,则只对<数值型表达式表>中的各表达式累加求和,否则将对当前表中的所有数值型字段累加求和,数值型表达式之间用逗号分隔;

(2)若使用 To <内存变量名表>,可将求出的各表达式的值依次赋给各内存变量,但要注意:表达式表中的表达式的个数应该与内存变量表中的变量个数相等,若省略<数值型表达式表>,内存变量的个数应该与表的数值型字段个数相等。也可将求出的各表达式的值存放于指定的数组中,一个数组元素等同于一个内存变量;

(3)若<范围>缺省,默认为 All。

例 3.24 求出所有员工的捐款金额和,试写出命令序列。

Use Contribution

Sum 捐款金额 To Jkze

? "捐款总额:", Jkze

3.4.3 求平均值命令

格式:

Average [<数值型表达式表>] [<范围>] [For <条件 1>] [While <条件 2 式 2>];
[To <内存变量列表> | To Array <数组名>]

功能:对当前表中指定范围内满足条件的记录的数值型字段求算术平均值,并把结果存入<内存变量名表>或<数组名>指定的变量中。

参数说明: Average 和 Sum 命令的不同之处,仅仅在于前者是求数值型字段或数值型表达式的平均值,而后者是求和。

3.4.4 计算命令

Calculate 命令用于对表中的字段进行财经统计,其计算工作主要由函数来完成。

格式:

Calculate<表达式表> [<范围>] [For<条件 1>] [While<条件 2>];
[To<内存变量> | Array<数组>]

功能:分别计算<表达式表>中表达式的值。

注意:表达式表中至少须包含系统规定的 8 个函数之一。如 Avg(),Cnt(),Max(),Min(),Sum(),Npv(),Std(),Var()函数等。

例 3.25 求所有人的捐款总和与人均捐款额。

Use Contribution

Calculate Sum(捐款金额), Avg(捐款金额) To Jkzh,Rjjk
?"捐款总数:", Jkzh, "人均捐款额:", Rjjk

3.4.5 汇总命令

汇总命令可对表的数据进行分类合计。

格式:

Total To <表文件名> On <关键字> [Fields <数值型字段表>] [<范围>];
[FOR<条件 1>] [WHILE<条件 2>]

功能:在当前表中,分别对<关键字>值相同的记录的数据值型字段值求和,并将结果存入一个新表。一组关键字值相同的记录在新表中产生一个记录;对于不参加汇总的字段,只将关键字值相同的第一个记录的字段值放入该记录。

参数说明:

(1) 执行该命令前必须对表按汇总项进行排序或索引,对于无序的表执行该命令无意义,<关键字>是指已经排序的字段名或设置主控索引的关键字;

(2) 在 Fields 子句中指定要汇总的字段。缺省时,对表中所有数值型字段汇总。



身份号	捐款金额
20050001	250.00
20050002	320.00
20050003	230.00
20050004	150.00
20050005	240.00
20050006	450.00
20050007	310.00
20050008	160.00

图 3.19 个人捐款汇总表

例 3.26 在表 Contribution 中按人员身份号汇总捐款金额。

Use Contribution

Index On 身份号 Tag Sfh

Total On 身份号 To Jkhz Fields 捐款金额

Use Jkhz

Brow Fields 身份号,捐款金额 &&. 结果如图 3.19 所示

3.5 Select-SQL 查询

SQL 语言的核心是查询,Select 是 SQL 的查询命令,其基本形式由 Select-From-Where 查询块组成,多个查询可以嵌套执行。VFP 支持在命令窗口直接使用 Select-SQL 命令进行查询,也可以通过“查询设计器”窗口来设计查询步骤、生成查询文件,然后运行定制的查询。

Select 语句具有强大的查询功能,有的用户甚至只需要熟练掌握 Select 语句的一部分,就可以轻松地利用数据库来完成自己的工作。

3.5.1 Select-SQL 命令查询

在命令窗口发送一条 Select-SQL 命令,系统就可以按命令的要求执行一次查询,但由于该命令中的参数较多,理解起来也比较复杂,所以为了便于对 Select-SQL 命令的理解,我们先介绍比较简单的 Join 命令,然后再介绍 Select-SQL 命令。

在 VFP 中,Join 的连接功能完全包含在 Select-SQL 命令中,但为了与早期的 Xbase 类数据库语言兼容,VFP 仍保留了传统的 Join 命令。Join 命令可实现两个表的联接。

3.5.1.1 表的联接

1. 四种专门的关系运算。

选择、投影、联接和除法,是关系数据库的 4 种关系运算。VFP 直接支持其中的前 3 种运算。

选择:命令中的 For<条件>子句,过滤器命令 Set Filter To <条件>等对表的记录的筛选,执行的就是选择运算。

投影:命令中的 Fields<字段名表>,Set Fields To <字段名表>等对表的字段的筛选,执行的就是投影运算。

联接:Join 执行的就是联接运算。

除法:该运算在 VFP 中没有相应的专用命令。

2. JOIN 命令。

格式:

Join With <别名> | <工作区> To <表文件名>;

For <联接条件> [Fields <字段名表>]

功能:按照 For 子句规定的联接条件,将当前工作区中的表与另一个以<工作区>或<别名>表示的工作区中的表进行联接,从而产生一个新表。

参数说明:

(1) <别名> | <工作区>:指定第二个表所在的工作区;

(2) For <条件>:指定一个联接条件。若<条件>的值为真,则向新表中写入一个新记录;

(3) [Fields <字段名表>]:指定新表中包含的字段的列表。在<字段名表>中只能包含两个表中已有的字段,缺省时,默认为两个被联接表的全部字段(自动删除重名字段)。

(4) Join 命令也可以联接两个以上的表,但只能先连接其中的两个,生成一个新的表文件后,再利用新表与另外的表建立连接。

例 3.27 将 Employee 和 Contribution 联接为一个新表 Emco,要求包含身份号,姓名,部门,捐款金额。

Close All

Select 1

Use Contribution

Select 2

Use Employee

Join With Contribution To Emco Fields 身份号,姓名,部门,Contribution. 捐款金额;

For Employee. 身份号=Contribution. 身份号

Use Emco

Brow

&&. 结果如图 3.20 所示

身份号	姓名	部门	捐款金额
20050001	张丽平	01	100.00
20050001	张丽平	01	50.00
20050001	张丽平	01	100.00
20050002	陈国庆	11	150.00
20050002	陈国庆	11	50.00
20050002	陈国庆	11	120.00
20050003	方世玉	12	100.00
20050003	方世玉	12	50.00
20050003	方世玉	12	80.00
20050004	王国真	11	50.00
20050004	王国真	11	50.00
20050004	王国真	11	50.00
20050005	关鹏	13	100.00
20050005	关鹏	13	50.00
20050005	关鹏	13	90.00
20050006	孙宏伟	12	200.00
20050006	孙宏伟	12	50.00
20050006	孙宏伟	12	200.00
20050007	李莉	13	150.00
20050007	李莉	13	50.00
20050007	李莉	13	110.00
20050008	杨剑雄	01	50.00
20050008	杨剑雄	01	50.00
20050008	杨剑雄	01	60.00

图 3.20 Emco 表的浏览结果

3.5.1.2 Select-SQL 命令

格式:

Select [All | Distinct];

[<别名>.]<Select 表达式>[As<列名>];

[, [<别名>.]<Select 表达式>[As<列名>]...];

From [Force][<数据库名> !]<表名>[<本地名>];

[[Inner|Left[Outer]|Right [Outer]|Full[Outer];

Join <数据库名> !]<表名> [<本地名>]On<联接>...];

[Into<Array<数组>|Cursor<临时表>|Dbf<表名>>];

[To File<文件名>][Additive]|To Printer [Prompt]|To Screen]]

[Preference <名字>][Noconsole][Plain][Nowait]

[Where<联接条件>[And<联接条件>...][And|Or<筛选条件>;

[And|Or<筛选条件>...]]]

[Group By<组表达式>[,<组表达式>...]]

[Having<筛选条件>]

```
[Union [All]<Select 命令>]
[Order By<关键字表达式> [Asc|Desc]][,<关键字表达式>[Asc|Desc]...]
[Top<数值表达式>[Percent]]
```

参数说明:

(1)Select 子句:对要查询的数据进行说明。

①All:查询结果中包括重复记录,这是缺省值;

②Distinct:选出的记录中不包括重复记录;

③[<别名>.]<Select 表达式>[As<列名>]:<Select 表达式>既可以是字段名,也可以包含用户自定义函数和如表 3.3 所示的系统函数。<别名>是字段所在的表名,<列名>用于指定输出时使用的列标题,可以不同于字段名;

④SELECT 表达式可用一个“*”号来表示,表示对所有的字段进行查询。

表 3.3 <Select 表达式>中可用的系统函数

函 数	功 能
Avg(<Select 表达式>)	求<Select 表达式>值的平均值
Count(<Select 表达式>)	统计记录个数
Min(<Select 表达式>)	求<Select 表达式>值中的最小值
Max(<Select 表达式>)	求<Select 表达式>值中的最大值
Sum(<Select 表达式>)	求<Select 表达式>值的和

当 Select 表达式中包含上述函数时,查询结果的输出行数可能与表的记录数不相同。

例 3.28

```
Select 身份号,工资*0.05 As 捐款 From Employee &&. 查询窗口每个记录显示一行数据
Select Avg(工资) As 平均工资 From Employee &&. 查询窗口只显示一行平均值数据
```

(2)From 子句及其选项:说明要查询的数据来源于哪个表或哪些表,并可指定多表查询的联接类型。

①该命令会自动选择工作区,并在所选的工作区中打开<表名>所指定的表。对于非当前数据库的表,用“<数据库名>!<表名>”来指定该数据库中的表。<本地名>是表的暂用名,取了本地名后,本命令中对该表的访问,只可使用这个名字访问;

②Join 关键字:用于联接其左右两个<表名>所指的表。可参阅 Join 命令;

③[[Inner|Left[Outer]|Right[Outer]|Full[Outer]]选项:指定两表联接时的联接类型,联接类型有 4 种;

[Inner]:内部联接,只有当左表和右表记录满足 On 子句指定的联接条件时,才产生一条新记录;

[Left]:左外部联接,用左表中的每一个记录与右表记录中的所有记录逐个进行比较,满足 On 子句指定的联接条件时,产生一条新记录;不满足 On 子句指定的联接条件时,也产生一条新记录,但与右表相关的字段为 Null;

[Right]:右外部联接,用右表中的每一个记录与左表记录中的所有记录逐个进行比

较,满足 On 子句指定的联接条件时,产生一条新记录;不满足 On 子句指定的联接条件时,也产生一条新记录,但与左表相关的字段为 Null;

[Full]:先用右外部联接方式产生新记录,再用左外部联接方式产生新记录,但不包括新产生的重复记录。

[Outer]:表示外部联接,既允许满足联接条件的记录,又允许不满足联接条件的记录。若省略 Outer 选项,效果不变。

④On 子句:用于指定两个表的联接条件,On 子句只能与 Join 搭配使用,当选用 Join 方式进行两个表或两个表以上的联合查询时,必须在该子句中指定表间的联接条件;

⑤Force 子句:严格按指定的联接条件来联接表,避免 VFP 因进行联接优化而降低查询速度。

(3)Into 与 To 子句:用于指定查询结果的输出方式,默认查询结果显示在浏览窗口。

①使用 INTO 子句中时,可将查询结果以三个选项中的一个选项方式输出:

Array<数组>:将查询结果输出到指定的数组中;

Cursor<临时表>:将查询结果输出到临时表,临时表关闭时自动从内存中清除;

Dbf<表名>:产生一个新表,将查询结果输出到该表中。

②To File <文件名>子句表示输出到指定的文本文件。省略 Additive 选项时,将输出结果输出到指定的文件中,当该文件存在时,复盖它;当选择 Additive 选项时,将输出结果输出到指定的文件中,当该文件存在时,在原文件内容后添加新数据;

③To Printer 表示将查询结果输出到打印机,Prompt 表示打印前先显示打印确认框;

④To Screen 表示将查询结果输出到屏幕。

(4)Preference 子句:用于记载浏览窗口的配置参数,再次使用该子句时可用 .<名字>引用此配置。

(5)Noconsole 子句:禁止将输出送往屏幕。若指定过 Into 子句则忽略它的设置。

(6)Plain 子句:输出时省略字段名。

(7)Nowait 子句:显示浏览窗口后程序继续往下执行。

(8)Where 子句:

当使用 Join 方式进行表的联合查询时,由于其联接条件必须在 On 子句中说明,所以在 Where 子句中就只能指定筛选条件,表示在已按联接条件产生的记录中筛选记录。若省去 Join 子句,则联合查询的表之间以“,”分隔,Where 子句应同时指定联接条件和筛选条件。

例 3.29 在表 Employee 中查询所有男职工的身份号,姓名和部门名称:

解一:

```
Select Employee. 身份号,Employee. 姓名,Department. 部门名称 From Employee,;  
Department Where Employee. 部门=Department. 部门代码 and Employee. 性别="男"
```

解二:

```
Select Employee. 身份号,Employee. 姓名,Department. 部门名称 From Employee Join;  
Department On Employee. 部门=Department. 部门代码 Where Employee. 性别="男"
```

以上两个解法中的命令完全等效,请大家仔细理解其中书写格式的不同之处。

(9)Group By 子句:对输出结果按<组表达式>值分组,常用于分组统计。

(10)Having 子句:当含有 Group By 子句时,Having 子句可用做记录查询的限制条件;无 Group By 子句时,Having 子句的作用如同 Where 子句。

(11)Union 子句:在 Select-SQL 命令中可以用 Union 子句嵌入另一个 Select-SQL 命令,选择该子句时,两个命令的查询结果合并一起输出,但输出字段的类型和宽度必须一致。

例 3.30 同时显示表 Employee 的部门记录和表 Department 的部门代码记录,可用下列命令实现,但其执行结果将首先显示表 Employee 的部门记录,接着显示表 Department 的部门代码记录:

```
Select 部门 From Employee Union All Select 部门代码 From Department
```

(12) Order By 子句:指定查询结果中记录按<表达式>排序,默认升序。<表达式>只可以是字段,或表示查询结果中列的位置的数字。选项 Asc 表示升序,Desc 表示降序。

(13) Top 子句:Top 子句必须与 Order By 子句同时使用。<数值表达式>表示在符合条件的查询结果中选取的记录数,排序后并列的若干记录只计一个。含 Percent 选项时,<数值表达式>表示百分比,记录数为小数地自动取整,范围 0.01—99.99。例如在上例命令中若增加子句“Top 50 Percent”,表示在符合条件的记录中选取 50%个记录。

3.5.1.3 Select-SQL 命令的举例

例 3.31 查找所有员工姓名和身份号

```
Select 姓名,身份号 From Employee
```

例 3.32 查找所有员工的所有信息,提示:用“*”表示指定表中所有的列

```
Select * From Employee
```

 &&. 在查询结果中显示表的所有列

例 3.33 显示查找结果中的前 5 个员工的所有信息

```
Select Top 5 * From Employee Order By 身份号
```

例 3.34 从 Employee 中查找出所有部门,请比较下列两命令执行后其结果的差异。

```
Select Distinct 部门 From Employee
```

```
Select All 部门 From Employee
```

例 3.35 将查询结果中员工的工资增加 20%。

```
Select 身份号,姓名,工资,工资 * 1.2 From Employee
```

例 3.36 在查询结果中将员工的工资增加 20%,并指定新的列名为“目前工资”。

```
Select 身份号,姓名,工资,工资 * 1.2 As "目前工资" From Employee
```

例 3.37 查询 1990 年 1 月 1 日后参加工作的员工信息

```
Select * From Employee Where 工作时间 >= {^1990-01-01}
```

例 3.38 求出每一个职工的捐款金额的总和。

```
Select 身份号,Sum(捐款金额) From Contribution Group By 身份号
```

例 3.39 找出捐款金额总和大于 300 的人及捐款金额。

```
Select 身份号,Sum(捐款金额) From Contribution Group By 身份号;
```

Having Sum(捐款金额) \geq 300

例 3.40 查找捐款人员的身份号、姓名以及每次捐款金额,并将查询结果以身份号为序排列。

```
Select Employee. 身份号, Employee. 姓名, Contribution. 捐款金额;  
From Employee Inner Join Contribution On Employee. 身份号 = Contribution. 身份号;  
Order By Employee. 身份号
```

上述命令还可写成如下格式:

```
Select Employee. 身份号, Employee. 姓名, Contribution. 捐款金额;  
From Employee, Join Contribution;  
Where Employee. 身份号 = Contribution. 身份号;  
Order By Employee. 身份号
```

例 3.41 试汇总每个人的捐款金额,要求显示身份号、姓名及捐款金额(分组);显示结果按捐款金额降序排列(排序)。

```
Select Contribution. 身份号, Employee. 姓名, Sum(Contribution. 捐款金额);  
As 捐款金额  
From Employee, Contribution;  
Where Employee. 身份号 = Contribution. 身份号;  
Group By Contribution. 身份号;  
Order By 捐款金额 Desc
```

例 3.42 在前例的基础上,要求同时显示人员所在的工作部门。

分析:该例中涉及到三个表 Employee, Department, Contribution 中的字段,所以应该是三个表的联合查询,在多表的联合查询中,查询联接条件个数是涉及的表的个数减一,查询条件间以逻辑运算符 And 连接。

```
Select Contribution. 身份号, Employee. 姓名, Department. 部门名称, ;  
Sum(Contribution. 捐款金额) As 捐款金额;  
From Employee, Contribution, Department;  
Where Employee. 身份号 = Contribution. 身份号 And Employee. 部门 = ;  
Department. 部门代码;  
Group By Contribution. 身份号;  
Order By 捐款金额 desc
```

上述命令还可写成如下形式,从下列命令形式看,每两个表之间的联接关系非常清晰容易,但上述命令的书写形式却比较简单:

```
Select Contribution. 身份号, Employee. 姓名, Department. 部门名称, ;  
Sum(Contribution. 捐款金额) As 捐款金额;  
From Employee Join Contribution On Employee. 身份号 = Contribution. 身份号, ;  
Employee Join Department On Employee. 部门 = Department. 部门代码;  
Group By Contribution. 身份号;  
Order By 捐款金额 Desc
```

3.5.2 利用查询设计器查询

在 VFP 用户不仅可以使⤵用 Select-SQL 命令进行查询操作,同时还提供了一种更直观的利用查询设计器来进行数据查询的查询方式。

3.5.2.1 查询设计器的操作步骤

- (1) 打开查询设计器;
- (2) 打开第一个需要查询的表;
- (3) 在打开第一个表后,再次打开表时,并同时设置表间的联接条件;
- (4) 设置需要输出的字段等,输出要求和查询结果的去向;
- (5) 执行查询;
- (6) 保存查询设置。

例 3.43 试用查询设计器来显示人员的身份号、姓名、部门名称。

操作步骤:

- (1) 打开查询设计器窗口。

方式一:单击菜单“文件/新建”或工具上的“新建”按钮,在“新建”对话框中选择查询选项,单击“新建文件”按钮,即出现如图 3.21 所示的“查询设计器”及“打开”对话框。

方式二:单击菜单“文件/打开”命令,在“打开”对话框的“文件类型”组合框中选定查询(*.qpr)选项,在“文件名”文本框中键入新查询文件名 RY(若是老文件则在列表框中选定文件),选定“确定”按钮,即出现如图 3.21 所示的“查询设计器”及“打开”对话框。



图 3.21 查询设计器窗口的初始状态

方式三:

打开查询设计器的方式,除有菜单方式外,还有命令方式:

格式一:

Create Query [<查询文件名>]

功能:创建一个查询文件。

格式二:

Modify Query [<查询文件名>]

功能:打开一个查询文件,当查询文件不存在时创建。

所以对于本例来说,打开查询设计器还可使用以下命令:

Modify Query RY &-&- 或 Create Query RY

(2)打开第一个需要查询的表 Employee. dbf。

在如图 3. 21 所示“打开”对话框的列表框中选定 Employee. dbf,单击“确定”按钮,该表就被添加到查询设计器的上部窗格中。

(3)打开第二个需要查询的表 Department. dbf。

单击“添加表或视图”对话框的“其他”按钮,通过弹出的“打开”对话框将 Department 添加到查询设计器中。

说明:向查询设计器添加表,第一个表在打开对话框的列表框中选定,若还需要其他表,可在 VFP 自动提供的“添加表或视图”对话框中添加,用户可在其中直接选用数据库表或利用“其他”按钮来选定所需的表。

只要“查询设计器”窗口成为当前窗口,就可用查询菜单的添加表命令来增入表,也可用该菜单的移去表命令将选定的表从窗格中移去。

(4)设置联接条件。

Department 加入查询设计器后即出现如图 3. 22 所示的“联接条件”对话框,若已添加的表中,与正在添加的表有同名字段,VFP 会根据字段名自动配对联接条件并显示,如联接条件为 Employee. 身份号=Department. 身份号,联接类型为内部联接。否则用户需要选择条件,单击“确定”按钮。



图 3.22 联接条件对话框

如果还需要打开其他表,则可重复(3)、(4)两个步骤,当完成表的添加后,关闭“添加表或视图”对话框,返回到“查询设计器”窗口。

说明:若 VFP 自动配对的联接条件不合用户需要,可在联接条件对话框中修改。该对话框中的取消按钮表示不要联接条件。

(5) 选取输出字段。

在“查询设计器”的字段选项卡中将 Employee. 身份号, Employee. 姓名和 Department. 部门名称 3 个字段从“可用字段”列表框移入“选定字段”列表框中, 如图 3. 23 所示。



图 3. 23 查询设计器的字段选项卡

说明: 创建查询允许不设联接条件, 但若未选取输出字段则查询不能运行。

(6) 执行查询。

单击菜单“查询/运行查询”即出现如图 3. 24 所示浏览窗口, 由图中可见, 当 Employee. 部门与 Department. 部门代码相同时产生一行查询记录。

身份号	姓名	部门名称
20050001	张丽平	办公室
20050002	陈国庆	一车间
20050003	方世玉	二车间
20050004	王国真	一车间
20050005	关鹏	三车间
20050006	孙宏伟	二车间
20050007	李莉	三车间
20050008	杨剑雄	办公室

图 3. 24 查询的浏览窗口

执行查询除在查询菜单选定运行查询命令外, 还有以下 3 种方法:

方法一: 在查询设计器窗口单击右键, 选定快捷菜单的“运行查询”命令。

方法二: 选定菜单“程序/运行”命令后, 在“运行”对话框中, 选定需要运行的查询后, 单击“运行”按钮。

方法三: 执行命令“Do <查询文件名全名>”。应注意此时不可缺省扩展名。例如: Do Ry. qpr。

(7) 查询的保存。

将查询设计器切换为当前窗口, 按组合键 CTRL+W, 保存查询文件 Ry. qpr。

说明: 查询经过修改后, 在查询设计器窗口关闭前可用以下 3 种方法之一来保存查询设置。

方法一: 按组合键 CTRL+W 后, 存盘并关闭查询设计窗口。

方法二: 单击窗口右上角的关闭按钮, 双击窗口左上角的控制菜单按钮, 或打开控制

菜单后选定关闭命令,都会出现“确认”对话框要求用户回答是否保存查询。单击“是”按钮用于保存查询,“否”按钮不保存本次修改且关闭窗口,“取消”按钮返回到查询设计器窗口。

方法三:选定文件菜单的保存命令存盘,但窗口不关闭。

3.5.2.2 查询设计器的界面组成

如图 3.23 所示,查询设计器分为上部窗格和下部窗格两部分,上部窗格用来显示查询的表,下部窗格则包含字段等 6 个选项卡。查询设计器打开后,VFP 还能在查询菜单、快捷菜单和查询设计器工具栏中提供有关的功能。

1. 上部窗格。

上部窗格显示已打开的表,每一个表用内含字段和索引,且大小可调整的窗口表示。

将表添入上部窗格的方法为:单击查询菜单(或快捷菜单)中的“添加表”命令,或选定查询设计器工具栏“添加表”按钮,出现“添加表或视图”对话框,用户可在该对话框中选取要添加的表。完成表的添加后,会弹出一个“联接条件”对话框,若两个表具有同名字段,则会自动在该框中列出字段相等的式子作为默认的联接条件,但允许用户修改;选定“确定”按钮返回查询窗口,该联接条件即自动出现在“联接”选项卡中。

若在分属于两个表的字段之间出现连线,表示它们之间设置了联接条件,联接条件除可在添加表时设置外,也可在表间拖放相关字段来创建。若要显示联接条件对话框只要双击某条连线,单击“查询设计器”工具栏的“添加联接”按钮可添加新的联接条件。

2. 下部窗格。

下部窗格包含以下 6 个选项卡。

(1) 字段选项卡。

指定要在查询结果中显示的字段、函数或其他表达式,如图 3.23 所示:

①“可用字段”列表框:列出查询设计器中表的所有字段,供用户选用;

②“函数和表达式”文本框:用来指定一个表达式,该表达式既可直接在文本框中键入,也可通过文本框右侧的对话按钮在表达式生成器中生成;

③“添加”按钮:将可用字段列表框、函数和表达式文本框中的选定项添入“选定字段”列表框。“移去”按钮则用于反向操作;

④“选定字段”列表框:用来列出输出表达式,上下拖动选项左边的双箭头按钮可调整输出的顺序。

(2) 联接选项卡。

该选项卡用于指定联接条件,联接条件可用来为一个或多个表或视图匹配和选择记录。

图 3.25 显示了 Employee. dbf 与 Department. dbf 的联接条件,即“Inner Join Employee. 部门=Department. 部门名称”。

通过菜单“查询/查看 SQL”命令,弹出一个只读窗口,在该窗口中显示与查询操作等效的 Select-SQL 命令,联接条件即被列在该命令的 On 子句中。

该选项卡还提供了几个列,其功能如下:

①类型列:指定联接的类型。在该列的组合框其中有五个选项,其中一个选项为“无”,表示不进行联接,其余 4 项联接类型的意义同 Select 命令;



图 3.25 联接选项卡

② 字段名列: 指定联接条件的第一个字段。在创建条件时, 单击字段会显示一个包含所有可用字段的下拉列表;

③ 条件列: 指定比较类型。除常用的关系运算符外, 还有 Between, In, Is Null 三种运算符, 但这三种运算一般用于指定筛选条件:

Between: 表示在低值 (含低值) 与高值 (含高值) 之间, 两值之间用逗号隔开。例如在“字段名”选择“工作日期”, 条件列选择 Between, “值”列是通过选择表达式, 弹出“表达式生成器”, 在表达式文本框中输入: {^1995/01/01}, {^2003/01/01}, 表示工作日期为 1985 年 1 月 1 日到 2003 年 01 月 01 日之间的记录均满足条件。

函数 Between(<表达式>, <低值表达式>, <高值表达式>) 有类似的功能, 即 <表达式> 的值在低值与高值之间返回 .T. 。

In: 表示取值范围是以逗号分隔的几个值。例如字段名选择“部门”, 条件列选择 In, 值列是通过选择表达式, 弹出“表达式生成器”, 在表达式文本框中输入: "11", "12", 表示部门是 11, 12 的记录满足条件。

函数 Inlist(<表达式>, <表达式 1>[, <表达式 2>...]) 有类似的功能, 即 <表达式> 与其他各表达式之一的值相等就返回 .T. 。

Is Null: 表示可包含 Null 值。

④ 否列: 选定否列按钮并使其打上“√”, 表示取上述条件之反;

⑤ 值列: 指定联接条件中另一表的字段;

⑥ 逻辑列: 用于在联接条件列表中添加 And 或 Or 运算, 仅当本行条件与下一行联接条件组成复合条件时使用;

⑦ “插入”按钮: 在所选定条件上方插入一个空白条件行;

⑧ “移去”按钮: 从查询中删除选定的联接条件;

⑨ 如果有多行条件, 则联接条件必须在前面排列, 并且作为联接条件的行数为所需联接表的个数减一, 联接条件间必须用 And 联接; 当在该标签中的行数多于需联接表的个

数减一时,随后的行为筛选条件。

(3) 筛选选项卡。

指定记录的筛选条件除可在联接条件后指定外,还可以在筛选条件选项卡中指定。筛选条件通常是在联接条件选出记录的基础上筛选记录,这种条件被列在 Select-SQL 命令的 Where 子句中。

由图 3.26 可见,筛选条件与联接条件的格式相似,但该选项卡没有联接条件的“值”列,却有“实例”列。在“实例”列中,只能指定筛选的具体值,不可包含变量,如字段等,也就是说,筛选条件只能将字段值与筛选值进行比较,而联接条件必须比较两个表的字段值。



图 3.26 筛选选项卡

(4) 排序依据选项卡。

如图 3.27 所示,该选项卡可用来指定多个排序字段或排序表达式,并可选定排序种类为升序或降序。



图 3.27 排序依据选项卡

在该选项卡中可以直接选择排序字段,而排序表达式,则必须先通过在字段选项卡中设定,然后才可以在排序依据选项卡中选择。

(5) 分组依据选项卡。

分组是指将表中具有相同字段值(或表达式值)的记录合并为一组,使整个表的所有

记录分成了若干组。分组依据选项卡用来指定分组字段或分组表达式,如图 3.28 所显示,选项卡中的满足条件按钮可用于为分好的记录组设置筛选记录的条件。

分组表达式必须先在字段选项卡中设定,然后才能在分组依据选项卡中选定。



图 3.28 分组依据选项卡

(6) 杂项选项卡。

指定是否要对重复记录进行查询,并且是否对记录做限制,包括返回记录的最多个数或最大百分比等。

例 3.44 试通过查询设计器汇总 1995 年 1 月 1 日前工作员工的捐款金额,要求:显示身份号、姓名及捐款金额总数;显示结果按捐款金额总数降序排列。

操作步骤:

(1) 打开查询设计器:在命令窗口键入命令 `Modify Query Ryjk`,即出现 `ryjk` 查询设计器(可参阅图 3.21);

(2) 向查询设计器添加要查询的表 `Employee` 和 `Contribution`:在打开对话框的列表框中双击表 `Employee`,完成表 `Employee` 的添加;然后单击“添加表或视图”对话框的“其他”按钮,并在打开对话框的列表框中双击表 `Contribution`,完成表 `Department` 的添加;

(3) 设置联接条件“`Employee. 身份号 = Contribution. 身份号`”:完成表 `Contribution` 的添加时,系统自动弹出“联接条件”对话框,单击“确定”按钮确认 `Employee. 身份号` 与 `Contribution. 身份号` 进行内部联接,然后关闭“添加表或视图”对话框。选定“联接”选项卡,其列表框中已显示联接条件“`Inner Join Employee. 身份号 = Contribution. 身份号`”,如果联接条件需要更改,可直接在联接选项卡中修改;

(4) 设置筛选条件:“`Employee. 工作时间 <= {^1995-01-01}`”,如图 3.26 所示;

(5) 设置输出表达式 `Employee. 身份号`,`Employee. 姓名`和 `Sum(Contribution. 捐款金额)`:选定字段选项卡,参阅图 3.23,将 `Employee. 身份号`和 `Employee. 姓名`从可用字段列表框移入选定字段列表框,然后在函数和表达式文本框中键入:`Sum(Contribution. 捐款金额)`,选定“添加”按钮将该表达式移入“选定字段”列表框;

(6) 设置按表达式 `Sum(Contribution. 捐款金额)`降序输出:选定如图 3.27 所示的“排序依据”选项卡,并在“选定字段”列表框选定表达式 `Sum(Contribution. 捐款金额)`,

选择“排序选项”的“降序”选项按钮,然后,通过单击“添加”按钮将该表达式从选定字段列表框移入排序条件列表框;

(7) 按字段 Contribution. 身份号分组:选定如图 3.28 所示的分组依据选项卡,双击可用字段列表框的字段 Contribution. 身份号,将其移入分组列表框,单击“确定”按钮;

(8) 执行查询:在查询设计器窗口单击右键,选定快捷菜单的运行查询命令,即出现如图 3.29 所示的浏览窗口;



身份号	姓名	Sum_捐款金额
20050006	孙宏伟	450.00
20050002	陈国庆	320.00
20050007	李莉	310.00

图 3.29 查询浏览窗口

(9) 若选定快捷菜单的“查看 SQL”,就会显示一个窗口,可将窗口内显示的 Select-SQL 命令复制到命令窗口上运行,其运行结果同上;

(10) 保存查询:将查询设计器切换为当前窗口,按组合键 Ctrl+W,以上(1)至(7)步的查询设置即存入查询文件 Ryjk. qpr。

3. 查询菜单。

查询设计器打开后,且查询设计器为当前窗口时,系统菜单会自动增加一个查询菜单。该菜单包含查询设计器下部所有选项卡的弹出命令,也包含快捷菜单和查询设计器工具栏的大部分功能,但不含查询设计器工具栏中的“添加联接”按钮,以及快捷菜单中的“帮助”按钮。下面仅介绍运行查询、查看 SQL 和查询去向这三条命令:

(1) 运行查询。

单击菜单“查询/运行查询”,执行当前查询设计器设计的查询,并输出查询执行结果。

(2) 查看 SQL。

单击菜单“查询/查看 SQL”,系统弹出一个新窗口,并在该窗口中显示在查询设计器中操作产生的相应的 Select-SQL 命令。但显示出来的命令只能阅读,不能编辑,通过剪贴板可以复制。大家通过对该窗口的查阅,能够了解在查询设计器中设计的查询与 Select-SQL 命令中的参数的对应关系,帮助大家更进一步理解 Select-SQL 命令的功能。

(3) 查询去向。

选定“查询去向”命令即出现如图 3.30 所示的查询去向对话框,其中共包括 7 个按钮,表示查询结果不同的输出类型。



图 3.30 查询去向对话框

浏览按钮:在浏览窗口中显示查询结果,当不指定查询去向时,浏览为默认值。

临时表按钮:将查询结果暂存于临时表中,当临时表关闭时,将被系统从内存中清除。

表按钮:将查询结果作为一个新的表文件保存起来。

图形按钮:使查询结果利用 Microsoft 的图形功能,该图形功能其实是由包含在 VFP 中的一个独立的 OLE 应用程序提供的。

屏幕按钮:在当前输出窗口中显示查询结果,在“次级输出”选项中也可指定是否输出到打印机或文件等。

报表按钮:向报表文件发送查询结果与屏幕输出一样,也可以在“次级输出”中指定其他输出方式。

标签按钮:向标签文件发送查询结果,也可以在“次级输出”中指定其他输出方式。

3.5.3 查询结果的图形处理

关于以报表或标签形式输出查询结果的方法,我们将在以后报表一章中介绍,现在我们将通过一个例子,简介如何利用 VFP 的图形向导工具输出图形。

例 3.45 将上例的查询结果以柱形图形式输出。

1. 运行支持图形输出的应用程序。

在“查询去向”对话框中,“图形”按钮通常是灰色,不能使用,让该按钮成为可用形式,则必须在命令窗口执行下一行命令:

```
_Gengraph="D:\Program Files\Microsoft Visual Studio\Vfp98\Wizards\;  
Wzgraph.app"
```

命令中的 _Gengraph 为系统变量,Wzgraph.app 应用程序由 VFP 提供,该文件的位置由安装目录确定,也可用搜索的方法来查找该文件。这里我们假设 VFP 的安装目录为:"D:\Program Files\Microsoft Visual Studio"。

2. 在查询设计器中打开 ryjk。

用界面方式打开:单击菜单“文件/打开”或工具栏上的“打开”按钮,弹出“打开”对话框,在该对话框“文件类型”列表框中选择“查询”后,选中 Ryjk.qpr 文件,单击“确定”按钮。

用命令方式打开:Modify Query RYJK

3. 确定输出类型为图形方式输出。

单击菜单“查询/查询去向”命令,使出现如图 3.30 所示的“查询去向”对话框,单击“图形”按钮后,按“确定”按钮返回查询设计器。

4. 对图形输出的形式进行设置。

单击菜单“查询/运行查询”,弹出“图形向导”对话框,在该对话框中的可用字段列表框中将“Sum_捐款金额”拖放到数据系列列表框中,将“姓名”拖放到显示“坐标轴”字样的矩形框中,如图 3.31 所示;



图 3.31 图形向导对话框步骤 2: 定义布局

选定“下一步”按钮,在如图 3.32 所示的图形向导对话框(步骤 3)中选择三维柱形图图形按钮,单击“下一步”按钮;



图 3.32 图形向导对话框步骤 3:选择图形样式

在弹出的图形向导对话框(步骤4)中的“输入图形的标题”文本框中键入“职员捐款汇总图”,单击“完成”按钮,在“另存为”对话框的保存表单文本框中键入 ryjk,单击“保存”按钮将图形存入表单文件 Ryjk. scx,并弹出表单设计器,在表单设计器中可对图形进行调整,如何调整我们将在表单设计一章中介绍。

5. 运行表单文件输出图形:执行命令 Do Form Ryjk,即显示如图 3.33 所示人员捐款汇总三维柱形图。

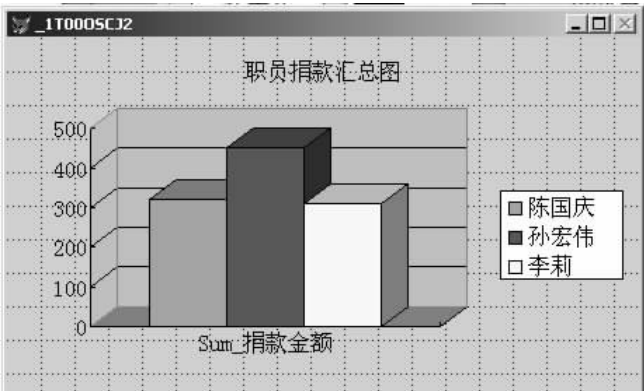


图 3.33 捐款金额汇总三维柱形图

从本例可以看到，VFP 能通过图形向导引导用户方便地输出图形。下面说明几个与本例有关的问题：

- (1) 图形向导对话框步骤 2：布局定义中通常需选取两个以上表达式，并且至少要有 一个数值表达式。这是出于构图的需要，即用数值表达式表示幅度，另一个表达式则表示坐标轴分点；
- (2) 图形向导对话框步骤 3：选择图形样式时，用户可从柱形图、折线图、圆饼图等多种统计图形中挑选；
- (3) 输出的图形将存入扩展名为 .scx 的表单文件，文件主名可由用户定义。表单文件用 Do Form 命令来执行，执行后即显示图形；
- (4) 除利用上例的方法打开图形向导外，还可以用以下两种方法：

方法一：在命令窗口中直接运行 Wzgraph. app 应用程序

直接调用 Wzgraph. app 应用程序来打开图形向导对话框，其步骤是：首先将与图形有关的数据存入一个表，然后打开它，再在命令窗口中利用 do 命令运行应用程序 Wzgraph. app，屏幕上即出现与图 3.21 相似的图形向导对话框，以后的操作类似于上例。

例 3.46

```
Use Employee
Copy To Employee1 Fields 身份号,捐款金额
Use Employee1
Do "D:\Program Files\Microsoft Visual Studio\Vfp98\Wizards\Wzgraph. app"
&&. 出现图形向导对话框
```

方法二：直接利用图形向导：

单击菜单“工具/向导/全部”，在弹出的“向导选取”对话框中选取“图形向导”后单击“确定”，又弹出“图形向导”对话框，根据向导对话框的提示，可以完成图形的输出。

使用图形向导方法对查询进行图形输出操作时，需先运行查询，然后关闭查询浏览窗口，查询结果并不关闭，而是临时保存在一个工作区中，然后再启动“图形向导”，在图形向导的字段选取对话框中选择要输出的查询或表，并选取字段，以后的操作同上。

使用该方法对表或数据库的视图进行图形输出操作时，表或视图可以事先打开，也可

以通过“图形向导”对话框打开,然后对表或视图进行选取。

3.6 数据库与视图

前面我们已经介绍了对自由表的一系列操作,本节将引入表的另一种形式数据库表概念,并介绍数据库表的建立与基本操作方法。另外,视图(View)是在数据库表(不是自由表)的基础上创建的一种虚拟表,在查询中也有着广泛的应用。

3.6.1 数据库

3.6.1.1 数据库的基本概念

一个自由表可以保存一些比较简单的信息,如学生成绩,这些信息一般是单独使用,与其他数据没有多少联系,如果在数据查询中涉及到多个表的信息,如开发一个企业或事业单位的人事管理系统,虽然可以利用查询和关联进行一些相关的操作,但无论是关联还是查询其关系都是一个临时关系,所以处理这种多表的数据使用上述方式,将极为不便。如果将这些有一定联系的表,都集中到一个数据库中,并且在各表之间建立若干固定的关系(关联),管理与使用这些表的数据,就会更加方便。

VFP 提供了这类功能,并将这类从属于某一数据库的表统称为数据库表。一个较大的应用项目可以创建若干个数据库,每个数据库可定义一组数据库表,然后用一定的关系将它们相互联接。数据库文件的扩展名为 .dbc。

3.6.1.2 利用数据库设计器设计数据库

3.6.1.2.1 数据库设计器

VFP 为设计数据库提供了一种简单的辅助设计窗口:数据库设计器,在数据库设计器中能显示当前数据库全部的表,视图和关系,并可以让用户比较直观的操作这些对象。

数据库表的建立有两种方式:利用数据库设计器直接建立新的数据库表;将已经存在的自由表添加到数据库中,使其成为数据库表。

在数据库设计器窗口中,每个表为一个可调整大小的窗口,并在该窗口中列出该表的字段和索引,在表之间用连接索引的线段表示永久关系。表或视图可以在数据库设计器中任意拖动,也可以使用数据库菜单的“重排”来重排位置,使用数据库菜单中的“属性”可以显示或隐藏数据库设计器窗口的某类对象,如表、关系、本地视图和远程视图等。

“数据库”菜单和“数据库设计器”工具栏,只有在“数据库设计器”窗口打开,并为当前窗口的前提下才会出现。

数据库表可以像自由表一样直接打开,而不用关心数据库设计器或数据库是否打开,但打开数据库表时,数据库表所在的数据库也自动打开。

例 3.47 在完成下例 3.48 后如执行命令 Use Employee 后,主屏幕和数据工作期窗口状态栏都会显示出它所在的数据库名:ZY。

3.6.1.2.2 数据库的打开与创建

例 3.48 创建数据库 ZY,要求包含 Employee,Department,Contribution 三个表。

本题要求创建的数据库也可简写为 zy {Employee,Department,Contribution},具体

创建过程如下：

1. 打开数据库设计器窗口。

(1) 界面方式。

单击菜单“文件/新建”，并在弹出的“新建”对话框中选择“数据库”，单击“新建文件”按钮，弹出“创建”对话框，在该对话框中键入数据库名：Zy(可省略扩展名，系统默认扩展名为 .dbc)，单击“保存”按钮创建数据库文件 Zy，系统同时自动弹出“数据库设计器”窗口(参阅图 3.38)。

(2) 命令方式。

格式一：

Create Database <数据库名>

功能：仅仅创建一个新的数据库，并使该数据库成为当前数据库，但不打开该数据库的数据库设计器。

格式二：

Modify Database <数据库名>

功能：当由<数据库名>指定的数据库存在时，打开该数据库；当由<数据库名>指定的数据库不存在时，创建该数据库；执行该项命令后，由<数据库名>指定的数据库将成为当前数据库，同时也打开该数据库的数据库设计器。

格式三：

Open Database <数据库名>

功能：打开由<数据库名>指定的数据库，并使该数据库成为当前数据库，但不打开该数据库的数据库设计器。

由上面三条命令的功能可知：本例中适用的命令只有格式一和格式二，其中使用格式二最为简单，但为了不与例题中的数据库发生重名，这里我们将新建的数据库名命名为 Zyy，具体命令如下：

Modify Database Zyy &&. 创建数据库 Zy，并打开该数据库的数据库设计器

上述功能还可以用以下命令实现：

Create Database Zyy &&. 仅仅创建数据库 Zy，并使该库为当前库

Modify Database Zyy &&. 打开该数据库的数据库设计器

2. 设置当前数据库。

当在 VFP 中同时打开多个数据库时，最后打开的数据库为当前数据库，此时向数据库添加或创建的所有表，均成为当前数据库的数据库表，如果想向某个特定的数据库添加表，则首先要将该数据库指定为当前数据库，其指定方式有两种：

(1) 界面方式。

在工具栏“数据库”下拉式列表框中，直接选择已经打开的、并要指定为当前数据库的数据库名。

(2) 命令方式。

格式：

Set Database To [<数据库名>]

功能:指定已经打开且名为<数据库名>的数据库为当前数据库;当省略参数时,为不设置任何数据库为当前数据库。

结合本例的要求,我们将 Zy 数据库设置为当前数据库,其命令如下:

Set Database To Zy && 将 Zy 数据库设置为当前数据库

3. 向当前数据库设计器窗口添加表。

(1)界面方式。

单击快捷菜单“添加表”(或单击“数据库设计器”工具栏的“添加表”按钮),弹出“打开”对话框,在该对话框中选择表 Employee,单击“确定”按钮,将表 Employee 添加到数据库设计器窗口中。以同样的方法添加 Department,Contribution。

(2)命令方式。

格式:

Add Table <自由表表名>

功能:向当前数据库中添加表名为<自由表表名>的自由表,并使其成为当前数据库的数据库表。当添加的表已经是其他数据库的数据库表时,系统将会出现出错提示。

当 Employee、Department、Contribution 三个表为自由表时,本例也可直接用命令方式向当前数据库添加,命令如下:

Add Table Employee && 向当前数据库添加自由表,并使其成为数据库表

Add Table Department

Add Table Contribution

Modify Database Ry && 打开 Ry 数据库的数据库设计器,并将 Ry 设置为当前数据库

Add Table Employee && 该命令在这里执行时,会出现什么问题呢?

至此例 3.48 已经完成,在完成该例的过程中为了说明其他问题,我们在其中增加了一些内容,现在请大家自己总结一下,并练习用比较简洁的方式完成本例。

(3)向数据库增加表的其他方式。

除了向当前数据库添加自由表外,还可以直接在当前数据库中创建新的表,其创建方式同自由表的创建方式,所不同的是,在当前数据库不存在时,无论是用界面方式,还是用命令方式创建的表均为自由表,若有某一数据库为当前数据库时,这时无论用何种方式创建表,该表均会自动成为该当前数据库的数据库表。

例 3.49

Set Database to && 不设置任何数据库为当前数据库

Create Table Salary(身份证号 C(18),姓名 C(8),性别 C(2),年龄 N(3),民族 C(2),;
 工作时间 D,部门 C(2),工资 N(8,2)) && 创建一个自由表

Set Database to Ry && 设置 Ry 数据库为当前数据库

Create Table Wage(身份证号 C(18),姓名 C(8),性别 C(2),年龄 N(3),民族 C(2),;
 工作时间 D,部门 C(2),工资 N(8,2)) && 在当前数据库中创建一个数据库表

由上例我们可以看出:创建表 Salary 时,由于没有设置任何数据库为当前数据库,所以该表创建后为自由表;而创建表 Wage 时,由于设置 Ry 数据库为当前数据库,所以该表创建后自动成为当前数据库 Ry 的数据库表。

3.6.1.2.3 数据库的关闭与删除

1. 数据库的关闭。

在第二章第一节的 2.1.3.2 通过命令方式修改表的结构中,已经介绍了数据库的关闭命令,通过前面数据库的初步学习,会对这些命令有一个新的认识,为了方便大家学习,将这些命令重新列举如下:

格式一:

Close All

功能:关闭所有打开的数据库与表,并选择工作区 1。关闭表单设计器、查询设计器、报表设计器、项目管理器。

格式二:

Close Database [All]

功能:关闭当前数据库及其中表;若无打开的数据库,则关闭所有自由表,并选择工作区 1,带有 All 则关闭所有打开的数据库及其中的表和所有打开的自由表。

格式三:

Close Tables [All]

功能:关闭当前数据库中所有的表,但不关闭数据库。若无打开的数据库,则关闭所有的自由表。带有 ALL 则关闭所有数据库中所有的表和所有的自由表,但不关闭数据库。

2. 数据库的删除。

命令格式:

Delete Database <数据库名>[Delete Tables]

功能:删除指定的数据库。

参数说明:

(1)当包含 Delete Tables 子句时,删除<数据库名>表示的数据库及其中的表;否则仅删除数据库,并将它的表变为自由表;

(2)删除数据库时,被删除的数据库必须已经处于关闭状态,否则不能删除。

3. 从数据库中移去数据库表。

(1)界面方式。

方法一:在数据库设计器中选要移去的表后,按 Del 键,并在弹出的信息对话框中单击“移去”或“删除”按钮;

方法二:在数据库设计器中选要移去的表后,单击快捷菜单“删除”,并在弹出的信息对话框中单击“移去”或“删除”按钮;

方法三:在数据库设计器中选要移去的表后,单击菜单“数据库/移去”,并在弹出的信息对话框中单击“移去”或“删除”按钮。

在以上三种方法中,若在弹出的对话框中单击“移去”按钮,则仅仅是将选定的表从数据库中移去,成为自由表;若在弹出的对话框中单击“删除”按钮,则是将选定的表从磁盘中删除。

(2)命令方式。

格式:

Remove Tables <数据库表名> [Delete]

功能:从数据库中移去或删除一个表。

参数说明:有[Delete]选项时,表示从数据库中移去表,并删除该表;没有[Delete]选项时,表示从数据库中移去表,使之成为自由表。

例 3.50

Modify Database RRYT

Create Table Salary(身份证号 C(18),姓名 C(8),性别 C(2),年龄 N(3),民族 C(2),;
工作时间 D,部门 C(2),工资 N(8,2)) && 创建一个数据库表

Create Table Wage(身份证号 C(18),姓名 C(8),性别 C(2),年龄 N(3),民族 C(2),;
工作时间 D,部门 C(2),工资 N(8,2)) && 创建一个数据库表

Remove Table Salary && 从数据库中移去表 Salary,使其成为自由表

Remove Table Wage Delete && 从数据库中移去表 Wage,并从磁盘中删除该表

3.6.1.2.4 数据库文件

数据库文件本身也是一个表,其中记载了它的所有表的参数,及索引、关联等有关参数。数据库文件的结构和记录是在创建和修改数据库时建立的,也可以直接编辑修改,但若修改后出现错误,数据库将会失效,并可能丢失数据。所以通常对数据库的操作我们应在数据库设计器中进行,而不对数据库文件本身进行直接的修改。

数据库文件可以像普通表一样打开并浏览,但在以表的形式打开数据库文件时,要求带扩展名且数据库文件必须处于关闭状态。

例 3.51

Close Database All && 关闭所有已经打开的数据库

Use Zy. dbc && 以表的形式打开数据库文件 Zy. dbc

Browse && 浏览数据库文件内容

Modify Structure && 打开表设计器,在表设计器中千万不要修改结构

3.6.1.3 用项目管理器管理数据

通过项目管理器,可以对项目中的数据库,数据库表、自由表进行新建、添加、修改浏览和移去等各种操作。

例 3.52 创建一个项目 Zzgl,要求添入数据库 Zy,并查看该数据库。

(1)打开项目管理器窗口。

在第一章第五节的 1.5.1 项目文件的创建中,我们已经介绍了创建项目管理器的方法,这里我们仅仅只用命令方式来创建并打开项目管理器,命令如下:

Modify Project Zzgl && 创建项目 Zzgl,并在项目管理器中打开

(2)将数据库 Zy 添入项目 Zzgl 中。

选择项目管理器的“数据”选项卡中的“数据库”图标,如图 3.34 所示,单击“添加”按钮,在弹出的“打开”对话框中选择数据库名为 Zy 的数据库,单击“确定”按钮,完成添加,并返回项目管理器。完成添加后,数据库图标左侧显示一个“+”号,表示数据库 Zy 已添入。



图 3.34 项目管理器的数据选项卡

(3)查看数据库 Zy。

单击数据库图标左侧“+”号,打开该分支,并在该分支下显示所包含数据库的名字 Zy,在出现的内容中,有的对象图标左侧有“+”号,单击这些“+”号可进一步打开他们的下级分支,如图 3.35 所示。选定其中一个表,就可以利用项目管理器窗口右侧的按钮对表进行新建、添加、修改、浏览、移去等操作。



图 3.35 项目管理器中的数据库与数据表

若要修改数据库 ZY,只要在项目管理器数据选项卡中选定 ZY 项,单击“修改”按钮后,系统就打开数据库设计器。

3.6.2 数据词典

数据词典用于保存对数据库中各种数据的定义或设置信息,包括表的属性、字段属性、记录规则、表间关系,以及参照完整性等。

在设计自由表时,表设计器也用来设置字段、索引和表的各种属性,但若所设计的是数据库表,设置的内容比起自由表来将丰富得多,参见图 3.36。数据词典的某些内容(如

长表名、长字段名、字段标题等)设置分为界面操作方式和命令操作方式两种,在界面操作方式中,数据词典的属性设置均是在表设计器中的选项卡中完成的,所以对数据词典的属性用界面操作方式进行设置时,必须先打开数据库表的表设计器;而用命令方式操作时,则必须打开要操作的表所在的数据库,并设置其为当前数据库。需要说明的是,命令方式只要求大家了解一下,知道数据词典的操作也可以用命令方式设置,在借助命令格式的前提下,能够完成相应的操作。

经过前面的学习,我们已经知道许多打开表设计器的方法,现在我们归纳一下,打开表设计器常用的几种方法:

方法一:若表属于某个项目中的对象,通过该项目的项目管理器打开;

方法二:通过数据工作期窗口打开;

方法三:在当前工作区打开该表后,单击菜单“显示/表设计器”打开;

方法四:在当前工作区打开该表后,直接在命令窗口中执行修改表结构命令打开。

由于打开数据库表的表设计器的方法与打开自由表的表设计器的方法相似,所以,我们在这里选择以通过项目管管理器的打开方式为例进行说明,具体过程如下:首先打开表所在项目的项目管理器,在已经打开的项目管理器中选择要设置字段属性的表后,单击项目管理器中的“修改”按钮,弹出表设计器,如图 3.36 所示。

3.6.2.1 长名(表名/字段名)与注释

自由表及其字段通常使用较短的名称(不超过 10 个字符),但 VFP 允许在数据词典中为数据库表和字段设置不超过 128 个字符的长表名,以及对表和字段增加适当的注释。长表名和长字段名一般能在浏览窗口或各种设计器(如数据库设计器、查询设计器和视图设计器等)的标题栏内显示;而对字段和表的注释则通常只会在项目管理器的底部显示。

3.6.2.1.1 长表名的设置

1. 界面方式。

在数据库表的表设计器中,选择“表”选项卡,在该选项卡中的“表名”文本框中输入长表名,退出表设计器时保存设置,如图 3.37 所示。

2. 命令方式。

格式:

```
Create Table <数据库表名> Name <数据库表的长表名>(<字段名1><字段类型>;  
[(字段宽度>[,<小数位数>])][,<字段名2>……])
```

功能:在当前数据库中创建一个表,该表具有一个长表名。

在创建数据库表时,没有特别指定长表名时,该数据库表的文件主名与长表名同名,否则,该数据库的文件主名为<数据库表名>,而其长表名为用户指定的<数据库表的长表名>。

例 3.53 在数据库 RRYT 中创建了一个数据库表文件名为 Salaryy.dbf 的表,其长表名为“ThisIsALongNameTable”。

Modify Database RRYT

```
Create Table Salaryy Name ThisIsALongNameTable(身份证号 C(18),姓名 C(8);  
性别 C(2),年龄 N(3),民族 C(2),工作时间 D,部门 C(2),工资 N(8,2))
```

3.6.2.1.2 长字段名的设置

用表设计器或命令方式创建或修改数据库表的结构时,直接对字段名进行修改,创建或修改的字段名长度可以超过 10 个字符,但最多为 128 个字符。

3.6.2.1.3 字段注释

1. 界面方式。

在表设计器“字段”选项卡的“字段注释”文本框中,可以对当前字段进行一些详细解释说明。这些解释说明只有在项目管理器中选择了有“字段注释”的字段后,才会在项目管理器的底部显示这些注释。

2. 命令方式。

格式:

Dbsetprop(<数据库表名>.<字段名>,"Field","Comment",<字段注释内容>")

功能:为当前数据库中数据库表的字段添加注释。

参数说明:所有参数都必须用定界符界定。

例 3.54 为表 Employee 的身份号字段添加注释“存放职员的身份代码”。

Modify DataBase Zy

&& 数据库也可以用命令 Open DataBase Zy 打开

Dbsetprop("Employee. 身份号","Field","Comment","存放职员的身份代码")

3.6.2.1.4 表注释

1. 界面方式。

在表设计器“表”选项卡的“表注释”文本框中,可以对当前表进行一些详细解释说明。这些解释说明只有在项目管理器中选择了有“表注释”的表或有“表注释”的表的某个字段后,才会在项目管理器的底部显示这些注释。

2. 命令方式。

格式:

Dbsetprop(<数据库表名>,"Table","Comment",<表注释内容>")

功能:为当前数据库中数据库表添加注释。

参数说明:所有参数都必须用定界符界定。

例 3.55 为表 Employee 添加注释“这是一个职员表”。

Modify DataBase Zy

Dbsetprop("Employee","Table","Comment","这是一个职员表")

3.6.2.2 设置字段属性

3.6.2.2.1 格式

1. 界面方式。

表设计器“字段”选项卡的“格式”文本框用于键入当前字段的格式表达式,确定该字段在浏览窗口、表单或报表中显示时采用的大小写和样式。在“格式”文本框中可以输入“!”或字母“A”符号,这两个符号可以单独输入,也可以同时输入,他们分别的含义如下:

!:在浏览窗口输入输出时将字母转为大写;

A:表示仅仅只允许输入字母。

2. 命令方式。

格式：

```
Dbsetprop("<数据库表名>.<字段名>", "Field", "Format", "<格式码>")
```

功能：设置指定字段的格式码。

例 3.56 将表 Employee 的姓名字段设置为只允许接受和输出大写字母。

Modify DataBase Zy

```
Dbsetprop("Salaryy. 姓名", "Field", "format", "A !")
```

3.6.2.2.2 输入掩码

1. 界面方式。

表设计器“字段”选项卡的“输入掩码”文本框用于键入当前字段的输入掩码，指定当前字段的输入格式，限制其输入数据的范围，控制输入的正确性。

输入掩码字符：

@：当需要在输入掩码中输入格式控制符时使用，并且只有在输入掩码最前面才有效

X：允许输入字符

9：允许输入数字

#：允许输入数字，空格，+，-

\$：显示 Set Currency 命令指定的货币号

*：在指定宽度中，值左显示星号

.：指出小数点位置

,：用逗号分隔小数点的数字（千分符）

一个输入掩码字符仅仅只能指定其所在位置的输入格式，所以，输入掩码是以位的形式来指定格式的，但除了使用以上输入掩码字符外，还可以使用其他符号，这些非输入掩码符号在数据输入过程中，仅仅只是以原样显示在其相应的插入位置，光标碰到这些非输入掩码符号时，将自动跳过非掩码符号指定的位。

例如设置电话字段的输入掩码为：9999—9999999，则在电话字段中每一个输入掩码“9”所对应的位仅仅只允许输入数字，其中符号“—”不是输入掩码，而是一个以原样显示的插入性字符，在输入数据时光标会自动跳过它。这样，既限制了输入数据的范围，也加快了输入速度。当设置好电话字段的输入掩码后，在数据输入过程中，用户只要输入：05518600879，该字段就会自动接受字符串：“0551—8600879”。本例设置身份证号字段的输入掩码为 xxxxxxxx，表示可输入 8 位字符，如图 3.36 所示。



图 3.36 Employee 数据库表的字段选项卡

2. 命令方式。

格式：

Dbsetprop("<数据库表名>.<字段名>", "Field", "Inputmask", "<掩码格式>")

功能：设置指定字段的输入掩码。

例 3.57 将上述电话字段的输入掩码设置,改用命令方式设置。

Open Database Zy

Dbsetprop("Employee. 电话", "Field", "Inputmask", "9999-9999999")

3.6.2.2.3 标题

1. 界面方式。

表设计器“字段”选项卡的“标题”文本框用于键入当前字段的标题,该标题作为当前字段的标题可在浏览窗口、表单或报表中显示。

例如设置身份号字段的“标题”文本框的内容为:职员身份号,那么在以后浏览该表时,身份号字段的列标题,将显示为:职员身份号,如图 3.36 所示。

2. 命令方式。

格式：

Dbsetprop("<数据库表名>.<字段名>", "Field", "Caption", "<注释内容>")

功能：为当前数据库中数据库表的字段添加标题。

例 3.58 将上述在表设计器中对身份号的标题设置,改用命令方式设置。

Modify DataBase Zy &&. 数据库也可以用命令 Open DataBase Zy 打开

Dbsetprop([Employee. 身份号],[Field],[Caption],[职员身份号])

3.6.2.2.4 字段验证

1. 界面方式。

表设计器“字段”选项卡的字段验证包含 3 个文本框,各文本框均可直接键入数据,也可通过其右边的对话按钮显示表达式生成器对话框,在表达式对话框中进行设置。

“规则”文本框用于输入对字段数据有效性进行检查的规则,它实际上是一个条件。例如对年龄字段设置字段验证“规则”文本框的内容为:年龄 ≥ 18 And 年龄 ≤ 130 。保存这个规则后,若对该字段输入数据,VFP 会自动检查它是否符合条件。若不符合,当光标离开该字段时,系统会弹出出错提示信息,提示用户需要修改输入的数据,直到与条件符合才允许光标离开该字段。

“信息”文本框用于指定系统弹出的出错提示信息内容。当该字段输入的数据违反字段验证的“规则”条件时,系统弹出的出错信息将照此内容显示,例如:“输入的年龄应在 18 至 130 之间!”

“默认值”文本框用于指定当前字段的默认值。当增加记录时,字段默认值会在新记录中显示出来,从而提高输入速度。例如将年龄的默认值设置为:20。

字段有效性验证除了可以在表设计器中直接对数据库表字段进行设置外,还可以直接用命令 Create table 和 Alter Table 对数据库表字段进行设置,具体格式如下:

2. 命令方式。

格式一：

Create Table <数据库表名>;

(<字段名 1><字段类型>[(<字段宽度>[,<小数位数>)]];

[Default <默认值>][Check <字段规则表达式> [Error <出错信息>]];

[,<字段名 2>……])

功能:在创建数据库表的同时,指定字段设置默认值以及字段有效性验证规则。

格式二：

Alter Table <数据库表名>;

Add [Column] <字段名><类型>[(<宽度>[,<小数位数>)]];

[Default <默认值>][Check <字段规则表达式> [Error <出错信息>]]

功能:在向数据库表添加字段的同时,为添加的字段设置默认值以及字段有效性验证规则。

格式三：

Alter Table <数据库表名>;

Alter [Column] <字段名><类型>[(<宽度>[,<小数位数>)]];

[Default <默认值>][Check <字段规则表达式> [Error <出错信息>]]

功能:在修改数据库表的字段属性时,可为修改的字段设置默认值和字段有效性验证规则。

格式四：

Alter Table <数据库表名> Alter [Column] <字段名> Set Default <默认值>;

[Check <字段规则表达式> [Error <出错信息>]]

功能:不对数据库表的字段进行修改,仅仅只是为指定字段设置默认值和字段有效性验证规则。

参数说明:以上四个命令格式的参数说明。

(1)有选项 Default 时,为与该选项对应的字段设置默认值,省略时默认值为 Null;

(2)有选项 Check 时,为与该选项对应的字段,设置字段有效性验证规则;

(3)有选项 Error 时,为与该选项对应的字段,设置违犯字段有效性验证规则时的出错提示信息;

(4)上述三条命令中的其他参数已经在第二章创建和修改表的结构中介绍过,所以不再作说明。

例 3.59 在数据库 Zy 中创建一个表 Salaryy(姓名 C(8),工资 N(7,2)),将工资的默认值设置为 1000.00,设置字段有效性规则:工资≥100,向该数据库表添加一个字段身份证号 C(18),并将其默认值设置为“34010”,设置字段有效性规则为:身份证号=“34010”,将姓名字段的默认值设置为“张三”。

Modify Database Zy

Create Table Salaryy(姓名 C(8),工资 N(7,2) Default 1000.00 Check 工资≥100 Error “工资值必须大于 100”)

Alter Table Salaryy Add 身份证号 C(18) Default "34010" Check 身份证号;
="34010"Error "身份证号要求必须以 34010 开头"

Alter Table Salaryy Alter 姓名 Set Default "张三"

在执行上述命令时,每执行完成一条命令后,要求观察表结构及相应字段的有效性验证规则发生了哪些变化,并通过向表中添加一条记录来观察字段默认值的变化。

3.6.2.3 设置记录规则

如图 3.37 所示,在数据库表设计器的“表”选项卡中,含有设置记录有效规则和触发器规则,设置这些规则一方面可以直接在该选项卡中设置,另一方面也可以通过命令方式设置。



图 3.37 Employee 数据库表的表选项卡

3.6.2.3.1 记录验证

1. 界面方式。

记录有效性规则是用来检查同一记录中不同字段之间逻辑关系,当光标离开记录时,对该记录验证该规则,当记录违犯该规则时,系统会弹出出错提示信息,指导用户修改出错内容,直到符合该规则,系统才允许光标离开该记录。

“规则”文本框:用于指定记录级有效性检查规则,光标离开当前记录时进行校验。

信息文本框:用于指定出错提示信息。在校验记录级有效性规则时,发现输入与规则不符时该信息将会显示。

例如在“规则”文本框输入的记录级有效性规则为:年龄-16=>((Date()-工作时间)/365),在“信息”文本框中输入的出错提示信息为:“参加工作的年龄必须满 16 周岁!”,如图 3.37 所示,当保存该设置后,在年龄和工作时期字段输入的数据,必须满足该规则,否则,系统会弹出出错提示信息:“参加工作的年龄必须满 16 周岁!”,指导用户修改出错内容。

2. 命令方式。

格式一：

Alter Table <表名>;

Add [Column] <字段名><类型>[(<宽度>[,<小数位数>]);

[Default <默认值>][Set Check <记录规则表达式> [Error <出错信息>]]

格式二：

Alter Table <表名>;

Alter [Column] <字段名><类型>[(<宽度>[,<小数位数>]);

[Default <默认值>][Set Check <记录规则表达式> [Error <出错信息>]]

格式三：

Alter Table <表名> Alter [Column] <字段名> Set Default <默认值>;

[Set Check <记录规则表达式> [Error <出错信息>]]

参数说明:以上命令格式与设置字段有效性验证命令的格式基本相同,但在选项 Check 的前面增加了一个保留字 Set,它表示的是设置记录有效性规则设置,而不是字段有效性验证规则设置。

例 3.60 将数据库 Zy 中的数据库表 Employee 的记录级有效性规则设置为:年龄-16=>((Date();-工作时间)/365),出错提示信息为:"参加工作的年龄必须满 16 周岁!"。

Modify Database Zy

Alter Table Employee Alter 工作时间 D Set Check 年龄-16=>((Date();

-工作时间)/365)Error "参加工作的年龄必须满 16 周岁!"

3.6.2.3.2 触发器

1. 界面方式。

在“表”选项卡中的 3 个触发器,分别用来指定记录插入、更新、删除时所发生的规则。

(1)插入触发器:用于指定一个规则,每次向表中插入或追加记录时,该规则被触发,并检查插入的记录是否满足规则。

如图 3.37 所示,在“插入触发器”文本框输入的条件为:Between(Dow(Date()),2,6),表示只有在计算机的系统日期为每周的周一到周五期间时,才可以对表进行记录的插入操作。否则,当光标离开该记录时,系统将弹出触发器失败信息框。

这里需要说明的是函数 Dow(),该函数返回指定日期在一周中的排列位置,西方国家一周的排列是从星期日、星期一、星期二、星期三、星期四、星期五、星期六的排列顺序,在该函数中对应的数值是 1,2,3,4,5,6,7。

(2)更新触发器:用于指定一个规则,每次更新表中记录时触发该规则。

在“更新触发器”文本框输入条件为:Inlist(Cdow(Date()),"Monday","Wednesday"),如图 3.37 所示,表示只有在每周的星期一和星期三才能对表的记录进行更新操作。函数 Cdow()与函数 Dow()不同之处是它返回当天星期的英文字符串,如:"Sunday","Monday"等。

(3)删除触发器:用于指定一个规则,每次在表中删除记录(打上删除标记)时,触发该规则。

如图 3.37 所示,在“删除触发器”文本框输入的条件为:Day(Date())=15,表示只有在每月的 15 日这一天,才允许对表中的记录作删除标记。

在所有的三种触发器中,都不显示具体的触发器失败的原因,因此,我们可以根据实际应用,针对这一特性来设置对表操作的一些限制条件。

2. 命令方式。

(1) 创建和修改触发器的命令格式。

Create Trigger On <数据库表名> For Delete | Insert | Update;

As <触发器规则表达式>

功能:当数据库表中已经存在相应的删除、插入和更新触发器,则修改它,否则创建数据库表的删除、插入和更新触发器。

(2) 删除触发器的命令格式。

Delete Trigger On <数据库表名> For Delete | Insert | Update

功能:删除数据库表的删除、插入和更新触发器

参数说明:

以上二个命令格式的 Delete | Insert | Update 参数为三选一,即每次只能创建、修改或删除一种触发器。

例 3.61 将以上界面操作方式设置的触发器规则用命令方式创建。

Modify Database Zy

Delete Trigger On Employee For Insert && 删除界面方式设置的插入触发器

Delete Trigger On Employee For Update && 删除界面方式设置的更新触发器

Delete Trigger On Employee For Delete && 删除界面方式设置的删除触发器

Create Trigger On Employee For Insert As Between(Dow(Date()),2,6)

Create Trigger On Employee For Update ;

As Inlist(Cdow(Date()),"Monday","Wednesday")

Create Trigger On Employee For Delete As Day(Date())=15

Create Trigger On Employee For Delete As Day(Date())=16 && 修改删除触发器规则

3.6.2.4 主索引与永久关系

3.6.2.4.1 主索引

在本章 3.1.2 索引一节按功能我们将索引分成了 4 种索引类型,在本节中,我们发现数据库表与自由表的表设计器有很大的不同,除以上已经介绍的内容外,索引选项卡中的类型组合框中列出的索引类型也是不同的,数据库表除了有普通索引、唯一索引、候选索引 3 种索引类型外,还增加了一个主索引类型,前面三种类型的索引大家已经掌握了,这里主要介绍一下主索引。

主索引与候选索引一样,其主要作用有:

1. 主索引不允许出现重复值,发现重复值会禁止存盘,故可用做主关键字。
2. 主索引可用于建立永久关系,从而建立参照完整性。

主索引和其他索引一样也可以在表设计器中通过界面方式直接创建和修改,但若使用命令方式创建或删除主索引,则需用以下命令格式:

格式一：

Alter Table <数据库表名> Add Primary Key <索引关键字>[Tag<索引标记名>]

功能：创建主索引

格式二：

Alter Table <表名> Drop Primary Key

功能：删除主索引

命令中的 Add 用于添加主索引，Drop 用于删除主索引。

例 3.62 为数据库表 Employee 添加以身份证号字段为主索引。

Alter Table Employee Add Primary Key 身份证号 Tag Sfh

例 3.63 删除主索引。

Alter Table Employee Drop Primary Key

3.6.2.4.2 永久关系与临时关系

永久关系是存储在数据库中数据库表之间的关系，在数据库设计器中表现为表索引之间的连线，连线的一端为一根，另一端为三根，分别代表一多关系的一端和多端。永久关系建立后存储在数据库文件中，只要不作删除或变更就一直保留。

永久关系在查询和视图中能自动成为联接条件；能作为表单和报表的默认关系，并显示在数据环境设计器中；允许建立参照完整性。

与永久关系相区别，以前建立的关系可称为临时关系。临时关系仅用于关联，即控制关联表的记录指针移动，永久关系在查询和视图中则起联接作用。

1. 用界面方式建立永久关系。

利用数据库设计器来建立永久关系，是在数据库表间建立连线，而删除永久关系也只需去掉连线。

建立永久关系的连线规则：在数据库设计器中，从一个表的主索引或候选索引拖到另一个表的任一索引（即出现表间连线）。

例 3.64 为数据库 Zy 建立如图 3.38 所示的永久关系。

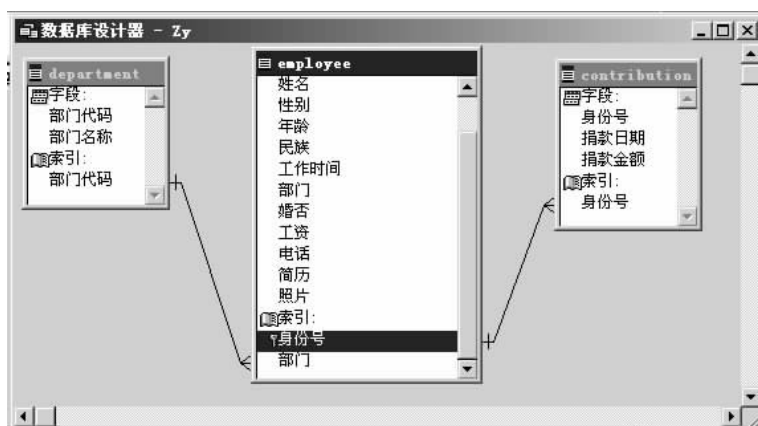


图 3.38 数据库 Zy 的永久关系

(1)打开数据库设计器窗口。

Modif Database Zy

(2)按表 3.4 建立相关索引。

Use Department

Index On 部门代码 Tag 部门代码 Candidate &&. 建立候选索引

Use Employee

Alter Table Employee Add Primary Key 身份号 &&. 建立主索引

Index On 部门 Tag 部门

Use Contribution

Index On 身份号 Tag 身份号

表 3.4 ZY 永久关系索引状况

数据库表	索引关键字	索引类型
Department	部门代码	候选索引
Employee	身份号	主索引
Employee	部门	普通索引
Contribution	身份号	普通索引

(3)画出连线。

将表 Employee 窗口的主索引“身份号”拖到表 Contribution 窗口的索引“身份号”上,释放鼠标后,在表 Employee 窗口的主索引“身份号”与表 Contribution 窗口的索引“身份号”间产生一条连线;以同样方法创建其他连线。结果如图 3.38 所示。

2. 用命令方式建议永久关系。

格式:

Alter Table <数据库表名(从表)> Add Foreign Key <从表关键字表达式>;

Tag <索引标识名> References <数据库表名(主表)> [Tag <主表索引标识>]

功能:从表创建索引标识的同时,在父线与从表间创建永久关系。

参数说明:

(1)<数据库表名(从表)>:指定从表;

(2)Foreign Key <从表关键字表达式> Tag <索引标识名>:为从表建立多方关系的索引标识;

(3)References <数据库表名(主表)> [Tag <主表索引标识>]:指定主表名,若是用主表的主索引与从表建立永久,则可省略 Tag <主表索引标识>;若用主表的一个候选索引与从表建立永久关系,则必须在 Tag 的<主表索引标识>中指明候选索引名。

例 3.65 例 3.64 中为数据库 Zy 建立永久关系还可命令方式实现。

Modify Database Zy

Use Department

Index On 部门代码 Tag 部门代码 Candidate

Use Employee

Alter Table Employee Add Primary Key 身份证号

Alter Table Employee Add Foreign Key 部门 Tag 部门;

References Department Tag 部门代码

Alter Table Contribution Add Foreign Key 身份证号 Tag 身份证号 References Employee

从本例可以看出,用命令方式建立永久关系比较简洁,但要求大家对命令格式非常熟悉;用命令方式创建永久关系前,仅仅只要求单独对主表建立主索引或候选索引,从表的索引可在建立永久关系的同时建立。无论是用界面方式还是用命令方式建立的永久关系,在查询设计器中就会反映出来。

例 3.66 利用查询设计器新建一个查询,并添加数据库表 Employee、Department 和 Contribution。

在向查询窗口添加以上三个数据库表时,弹出任何联接条件都选择取消,当将三个数据库表添加完成后,在查询设计器上部窗格将显示用线相连的三个窗口(Employee 窗口、Department 窗口和 Contribution 窗口),联接选项卡中也已列入相应的联接条件。

关于利用查询设计器创建查询的过程请大家根据前面学习过的查询,自己创建,这里就不详细介绍。

说明:添加三个表时,顺序不同将会有不同的操作过程,请大家体会一下,并根据已经学习过的知识解释为什么?

3.6.2.5 参照完整性

前面介绍的数据库表的字段和记录级验证规则是属于数据库表内部的规则,而参照完整性则属于数据库表之间的规则。

对于永久关系的两个数据库表,在更新、插入或删除记录时如果只改其中一个表的内容,而不修改另一个表的相应内容,则可能会影响到数据的完整性。如修改父表中关键字值后,子表关键字值未作相应改变;删除父表的某记录后,子表的相应记录未删除,致使这些记录成为孤立记录;在子表中插入的记录,父表中没有相应关键字值的记录等等。对这些涉及表间数据的完整性,统称为参照完整性(RI)

VFP 提供了参照完整性规则,可让用户利用“参照完整性生成器”来选择是否要保持数据库表间的参照完整性;并控制是否在相关表中,对相应的记录进行相应的更新、插入或删除操作。

3.6.2.5.1 参照完整性(RI)生成器窗口的打开

在数据库设计器窗口中,进行参照完整性设置之前,必须首先单击菜单“数据库/清理数据库”命令,对数据库进行清理,在系统执行完成数据库清理后,系统才允许打开参照完整性生成器窗口,打开参照完整性生成器窗口的方法有三种:

(1)从数据库设计器快捷菜单选择编辑参照完整性命令;

(2)选择数据库菜单中的编辑参照完整性命令;

(3)在数据库设计器中双击两个表之间的连线,并在“编辑关系”对话框中单击“参照完整性”按钮。

例 3.67 打开数据库 Zy.dbc 的参照完整性生成器窗口。

1. 打开数据库 Zy.dbc 的数据库设计器。

打开数据库设计器的方法有很多,有命令方式和界面方式,这里利用项目管理器来打开数据库设计器其方法是:在项目管理器中选定数据库 Zy.dbc,单击“修改”按钮打开数据库设计器。

2. 对数据库 Zy.dbc 进行清理。

单击菜单“数据库/清理数据库”命令,对当前数据库进行清理。

3. 打开数据库 Zy 的参照完整性生成器窗口。

单击快捷菜单的“编辑参照完整性”或系统菜单“数据库/编辑参照完整性”命令,系统弹出参照完整性生成器窗口,如图 3.39 所示。

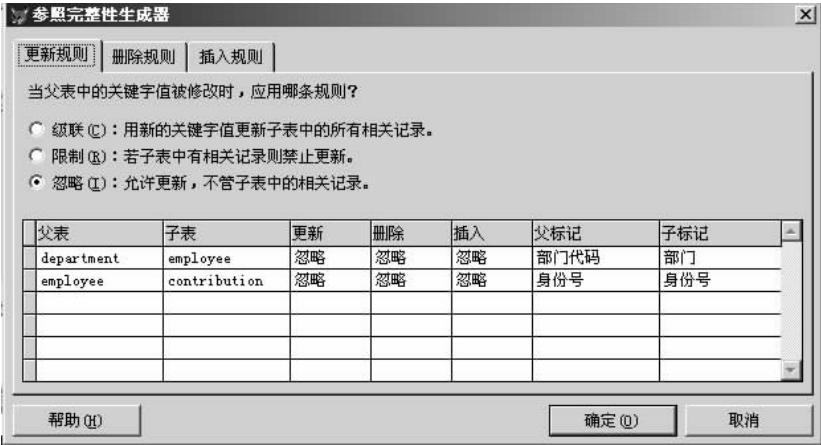


图 3.39 Zy.dbc 永久关系的参照完整性

3.6.2.5.2 参照完整性生成器窗口的组件

参照完整性生成器窗口具有更新规则、删除规则和插入规则 3 个选项卡,每个选项卡又含有级联、限制和忽略等选项按钮和一个表格。

在不同的选项卡中,各选项按钮的具体功能解释如表 3.5 所示。

表 3.5 参照完整性生成器窗口中选项按钮的功能

	更新规则选项卡	删除规则选项卡	插入规则选项卡
级联	更改父表关键字段值时,VFP 会自动更改所有子表相关记录的对应值,使之与父表一致	删除父表中的记录时,相关子表中的记录将自动删除	
限制	若子表有相关记录,则更改父表关键字段值就会产生“触发器失败”的提示信息	若子表有相关记录,则在父表中删除记录时,产生“触发器失败”的提示信息	若父表没有相匹配的记录,则在子表添加记录就会产生“触发器失败的信息”
忽略	允许父表更新,删除或插入记录,与子表记录无关		

参照完整性生成器窗口中的表格表达了表的联接情况及是否要保持参照完整性的规则。其中父表列、子表列分别显示联接表的父表名、子表名;父标记列显示父表的索引标识,该标识可以是主索引或候选索引;子标记列显示子表的索引标记;更新列、删除列和插

入列则显示所选定规则的名称。

3.6.2.5.3 参照完整性设置步骤

设置参照完整性规则有两种方式：一种是直接在参照完整性窗口的表格中，单击相应列(更新、删除、插入)的组合框按钮，选择级联、限制和忽略等功能；另一种是在相应的选项卡中，单击相应的规则单选按钮。

例 3.68 更新 Department 表的代码字段时，若要求 Employee 表自动按参照完整性调整。

(1)在上例的基础上，选择更新规则选项卡；

(2)单击表格第一行行首的按钮，即选择 Department 与 Employee 两表的联接；

(3)选定级联选项按钮，单击“确定”按钮保存参照完整性设置；

(4)分别打开表 Department 和 Employee 的浏览窗口，在表 Department 浏览窗口中，将代码字段值由 11 改为 35；单击表 Employee 的浏览窗口，使其成为当前窗口，可发现该表中部门字段值为 11 的均自动变成了 35。

如果上例中选定的是限制选项按钮，并在浏览 Department 表时将代码字段值由 11 改为 35，当光标离开该记录时，即出现触发器失败的提示信息，原因是 Employee 表中也具有部门字段值为 11 的记录。

3.6.3 视图

VFP 视图又称为 SQL 视图，视图兼有“表”和查询的特点，与查询相似之处是可以从一个或多个相关联的表中提取信息；与表相类似之处是可以用来更新其中的信息，并将更新结果保存在源表中。视图中的数据来源于已有的数据库表或其他视图。视图中的数据在数据库中并不实际存储，仅仅只在其数据词典中存储视图的定义。但视图一经定义，就成为数据库的组成，可以像数据库表一样接受用户的查询。

使用 VFP 当前数据库中的数据库表建立的视图是本地视图，使用当前数据库之外的数据源(如 SQL Server)中的表建立的视图是远程视图。例如从 SQL Server 或其他 ODBC 数据源中创建视图，并选择“发送更新”选项，在更新或更改视图中的一组记录时，由 VFP 将这些更新发送到源表中。

3.6.3.1 视图的创建

1. 界面方式。

方法一：借助项目管理器创建本地视图：

(1)在项目管理器中选择要创建视图的数据库；

(2)展开该数据库分支，并在分支列表中选定本地视图；

(3)单击项目管理器的“新建”按钮，弹出“新建本地视图”对话框；

(4)在“新建本地视图”对话框中单击“新建视图”按钮，弹出视图设计器；

(5)在视图设计器中对新建的视图进行设置。

方法二：在数据库设计器中创建本地视图。

(1)打开数据库设计器；

(2)单击菜单“数据库/新建本地视图”命令，弹出“新建本地视图”对话框；

(3)以下操作同方法一中的 4、5 步。

在“新建本地视图”对话框中,还包含一个“视图向导”按钮,用于引导用户快速地创建视图。

关于远程视图的创建方法,我们通过后面的一个例题,进行说明。

2. 用命令来创建。

格式:

Create Sql View <视图名>[Remote][Connection<新建连接名>[Share]];

<已连接数据源名>[As Select-SQL]

功能:按照 As 子句中的 Select-SQL 命令提出的查询要求,创建一个本地或远程的 SQL 视图。视图的名称由命令中的<视图名>指定。

参数说明:

(1)As 子句中的 Select-SQL 命令,指定视图从哪些数据库表提取数据,提配哪些字段,以及提配的条件;

(2)若要创建远程视图,需使用 Remote 可选项,并用 Connection 子句创建一个新的联接或指定一个已联接的源。有没有 Connection 子句,是区分本地视图和远程视图的标志。

例 3.69 从 Zy 库所属的 Employee 和 Contribution 两个表中抽取身份号,姓名和捐款金额 3 个字段,组成名称为“我的视图”的本地视图。

Open Database Zy

Create Sql View 我的视图;

As Select Employee. 身份号,Employee. 姓名,Contribution. 捐款金额;

From Employee, Contribution Where Employee. 身份号=Contribution. 身份号

创建的视图可以在项目管理器或数据库设计器中浏览或修改。这里介绍一下在项目管理器中浏览修改的方法:在项目管理器中先将数据库图标、数据库 Zy 图标和本地视图图标逐级展开,选择要进行浏览或修改的视图后,单击“浏览”或“修改”按钮。若单击“浏览”按钮,则将在视图浏览窗口显示查询结果;若单击“修改”按钮,则打开视图设计器,供用户修改视图。

例 3.70 在 D 盘的 VFP 目录下,利用常用办公软件 Office 的 Access 创建一个名为 Acce 的 Access 数据库,本例中该数据库存放于:D:\VFPACC 目录中,ACCE 数据库包括两个字段:身份号(文本 8)和家庭住址(文本 50),内容如图 3.40 所示。在 VFP 的数据库中利用该表创建一个名为“我的远程视图”的远程视图。

解法一:通过界面方式完成。

(1)在 D:\VFPACC 目录中创建 Access 表:Acce,并输入相应的数据,如图 3.40 所示;

(2)创建数据连接:

① 打开项目管理器中的项目 Zygl,并将数据库分支展开,选择“远程视图”图标;



身份号	家庭住址
20050001	合肥市美菱大道128号
20050002	合肥市阜阳路129号
20050003	合肥市长江路68号
20050004	合肥市金寨路39号
20050005	合肥市东流路66号
20050006	合肥市望江路77号
20050007	合肥市寿春路99号
20050008	合肥市蒙城路156号

记录: 1 共有记录数: 8

图 3.40 Access 数据库中的 Table1 表

② 单击“新建”按钮,弹出“选择连接或数据源”对话框;

③ 单击“选择连接或数据源”对话框中的“新建”按钮,弹出“连接设计器”;在第二步中直接选择“连接”图标后,单击“新建”按钮,将直接弹出“连接设计器”对话框,如图3.41所示;



图 3.41 “连接设计器”对话框

④ 在“连接设计器”对话框中的“指定的数据源”中选择“数据源、用户标识、密码”后,并在“数据源”列表框中,选择“MS Access Database”选项,然后单击“新建数据源”按钮,弹出“DOBC 数据源管理器”对话框;如图 3.42 所示;



图 3.42 “数据源管理器”对话框

⑤ 在“DOBC 数据管理器”对话框中,若“用户数据源”列表中不存在“MS Access Database”选项,则可单击“添加”按钮,弹出“创建新数据源”对话框,如图 3.43 所示;在“创建新数据源”对话框中选择“Microsoft Access Driver”,单击“完成”按钮,弹出“DOBC Access Database 的安装”对话框,如图 3.44 所示;若“用户数据源”列表中有“MS Access

Database”选项,则单击“配置”按钮,则直接弹出“DOBC Access Database 的安装”对话框。



图 3.43 “创建新数据源”对话框



图 3.44 “DOBC Access Database 的安装”对话框

⑥ 在“DOBC Access Database 的安装”对话框中的“数据源名”文本框中输入一个名称如:MS Access Database,单击“选择”按钮,在弹出的“选择数据库”对话框中选择 D:\VFPACC\Acce.mdb,如图 3.45 所示,分别单击“选择数据库”对话框和“数据源管理器”对话框的“确定”按钮,返回到“连接设计器”对话框;



图 3.45 “选择数据库”对话框

⑦ 关闭“连接设计器”对话框,在弹出的保存对话框中输入连接名:Remot_1,单击“确定”,完成连接的创建;

3. 创建远程视图。

(1) 在项目管理器的项目 Zygl 中,选择“远程视图”图标,单击“新建”按钮,弹出“选择连接或数据源”对话框,如图 3.46 所示;



图 3.46 “选择连接或数据源”对话框

(2) 在“选择连接或数据源”对话框中选择刚才已经建立的连接:Remote_1,单击“确定”按钮,弹出视图设计器及其“打开”对话框,如图 3.47 所示;



图 3.47 视图设计器及其“打开”对话框

(3) 在“打开”对话框中将选择的表添加到视图中;在视图设计器的“字段”选项卡中,将选定的字段添加到“选定字段”列表框中;保存视图名为:我的远程视图。关于视图的其他操作,将在下面的视图设计器中介绍。

4. 浏览“我的远程视图”。

在项目管理器中选择刚刚创建的远程视图“我的远程视图”图标,单击“浏览”或“修改”按钮,对视图进行浏览或修改。

解法二:通过命令方式完成。

(1) 在 D:\VFPAACC 目录中创建 Access 表:Acce,并输入相应的数据,如图 3.40 所示;

(2) 创建数据连接:

用下面命令创建一个连接：

Create Connection Remote_1 Datasource "MS Access Database"

在项目管理器中选择新建的连接:Remote_1,单击修改,在接下来的操作中,配置数据源,其配置方法同上。

(3) 创建远程视图：

Create Sql View 我的远程视图 Remote Connection Remote_1 As Select * ;
From Table1

(4) 浏览“我的远程视图”：浏览方法同上。

3.6.3.2 视图设计器

视图设计器同查询设计器界面非常相似,视图设计器窗口只是比查询设计器窗口多出一个“更新条件”选项卡,在该选项卡中可以对视图设置更新条件。因此,视图设计器除了具有与查询设计器相似的功能外,还多了一个数据更新功能。用户如果想实现从本地或远程表中提取一组数据,并可以对其进行直接更新,使用视图是一个非常好的解决方案。

3.6.3.2.1 利用界面方式设置视图的更新

在前面我们学习的查询,其结果只能阅读,而不能更新。但是我们往往有时需要对查询的数据进行更新,由于视图不仅具有查询的功能,还可更新数据,并且可以让源数据随其一同更新。

例 3.71 根据例 3.69 的查询要求,用视图设计器建立我的视图 1,要求可以通过修改视图中的捐款金额来更新表 Contribution 中原有的捐款金额。

1. 建立我的视图 1。

- (1)打开 ZYGL 项目管理器,并选择 Zy 数据库中的“本地视图”图标,单击“新建”按钮;
- (2)在弹出的“新建本地视图”对话框中单击“新建视图”按钮;
- (3)通过“添加表或视图”对话框向刚刚打开的视图设计器中添加表 Employee、Contribution,完成的添加后,在视图设计器窗口两表间已经以身份号联接进行了联接,如图 3.48 所示;



图 3.48 向视图设计器中添加表

(4)在“字段”选项卡中将可用字段列表中的字段:Employee. 身份号、Employee. 身份号、Contribution. 捐款金额等,移到“选定字段”列表框中。其操作过程同查询设计器;

2. 设置更新条件:设置结果如图 3.49 所示。



图 3.49 视图设计器“更新条件”选项卡

(1)在视图设计器窗口选择“更新条件”选项卡;

(2)设置关键字段:在更新选项卡中若某行的“钥匙”符号列为“√”号时,表示该行的字段为关键字段,选取关键字段可使视图中修改的记录与表中原始记录按关键字进行匹配;

(3)设置可更新字段:在更新选项卡中若某行的“铅笔”符号列为“√”号时,表示该字段的源数据可随视图同时更新,但只有在选择“发送 SQL 更新”复选框后,才能将视图记录中的修改传送给源数据。本例选择的可更新字段为 Contribution. 捐款金额;

注:在设置可更新字段时,可更新字段所在的源数据至少应该有一个字段应被设置为关键字段。如本例中将 Contribution. 捐款金额 设置为可更新字段,但由于表 Contribution 在本视图中只仅仅有这一个字段,若不将其首先设置为关键字段,则不可能将其设置为可更新字段。

(4)保存视图为:我的视图 1。

3. 更新捐款金额。

(1)打开表 Contribution 的浏览窗口;

(2)打开我的视图 1 的浏览窗口:打开视图浏览窗口的方法与打开表的浏览窗口方法相似,这里介绍通过视图设计器浏览的方式:让视图设计器成为当前窗口,单击菜单“查询/运行查询”弹出我的视图 1 的浏览窗口;

(3)在我的视图浏览窗口中,修改捐款金额,完成数据输入后,单击另一记录使光标离开当前记录,完成对该记录的修改,单击表 Contribution 的浏览窗口使之成为当前窗口,我们可以看到该窗口中的数据也随视图一起完成了数据的修改。

3.6.3.2.2 利用命令方式设置视图的更新

视图使用五个属性控制更新,这些属性及其默认值设置如下表 3.6 所示:

表 3.6 视图更新属性及其默认值

视图属性	默认设置
Tables	具有可更新字段和至少一个主关键字段的全部表
KeyField	数据库关键字段和表的远程主关键字
UpdateName	所有字段,形式为:<表名>.<字段名>
SendUpdates	除主关键字段的所有字段,默认设置只在工作期内有效,初始值为假,若将其设置为真,则成为在工作期内所创建的全部视图的默认值
CompareMemo	默认值为真,批备注字段包含在 Where 子句中,用来检测更新冲突

下面我们主要介绍三个视图属性的设置:

格式一:

```
Dbsetprop ("<视图名>.<字段名>","Field","KeyField",. T. | . F.)
```

功能:设置或取消视图的字段为关键字段。

说明:

(1)可以通过设置一个或多个 KeyField 属性,作为可更新表的唯一关键字;

(2)需要设计更新的表,在设置其字段的 UpdateName 属性前,必须先设置该表的 KeyField 属性。

格式二:

```
Dbsetprop ("<视图名>.<字段名>","Field","Updatename","<数据库表名>.<字段名>")
```

功能:在<视图名>.<字段名>与<数据库表名>.<字段名>之间建立映射关系,系统已经默认视图与数据源间以同名建立了字段映射关系,所以一般情况下不用特别说明视图字段与数据源间的映射关系。

格式三:

```
Dbsetprop ("<视图名>.<字段名>","Field","Updatable",. T. | . F.)
```

功能:将已经具有 Updatename 属性的<视图名>.<字段名>设置为可更新或不能更新,其默认值为. T. 。

格式四:

```
Dbsetprop ("<视图名>","View","SendUpdates",. T. | . F.)
```

功能:允许将视图的更新信息发送到可更新表和可更新字段。

例 3.72 建立一个名为:“我的视图更新”的视图,在视图中要有部门名称、身份号、姓名、和工资字段,要求可以通过修改视图中数据,仅仅只有源表的工资字段可以随视图更新。

(1)打开数据库 Zy。

```
Open Database Zy
```

(2)创建一个视图,该视图名为:我的视图更新。

执行完下面一条的命令后,打开该视图的视图设计器的“更新条件”选项卡,大家应该注意到:在建好视图中,视图字段所属的数据源表已经设置有主索引,并且该主索引字段也是视图中的一个字段时,那么,该主索引字段自动被设置为该数据源在视图中的关键字段,与该数据源相关的字段也自动被设置为可更新字段,如图 3.50 所示。



图 3.50 视图设计器“更新条件”选项卡

Create Sql View 我的视图更新;

As Select Department. 部门名称,Employee. 身份号,Employee. 姓名,;
Employee. 工资 From Employee,Department;
Where Department. 部门代码=Employee. 部门

(3)设置表 Department 的部门名称为关键字段。

执行完下面一条命令后,重新打开视图设计器,从其“更新条件”选项卡中,我们可以看到:与部门名称字段同源的字段,都会自动被设置为可更新字段,这里仅仅只有部门名称字段,如图 3.51 所示。

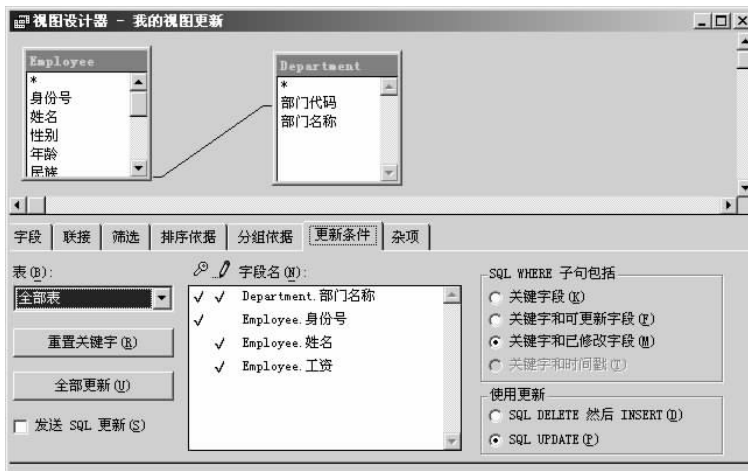


图 3.51 视图设计器“更新条件”选项卡

Dbsetprop ("我的视图更新. 部门名称", "Field", "KeyField", . T.) && 设置关键字
(4)设置其他不需要更新的字段为不可更新字段。

执行完下面一条命令后,重新打开视图设计器,观察一下“更新条件”选项卡有什么变化,其结果如图 3.52 所示。



图 3.52 视图设计器“更新条件”选项卡

Dbsetprop ("我的视图更新. 姓名", "Field", "Updatable", . F.)
Dbsetprop ("我的视图更新. 部门名称", "Field", "Updatable", . F.)

(5)设置可以向源数据发送更新信息。

Dbsetprop ("我的视图更新", "View", "SendUpdates", . T.)

(6)浏览“我的视图更新”,并在浏览窗口中修改视图中的数据,观察与之相关的源表中的数据变化。

Select 1

Use Employee

Browse

Select 2

Use Department

Browse

Select 3

Use 我的视图更新 && 打开视图

Browse && 打开当前工作区中视图的浏览窗口

Index on 身份号 tag 身份号 && 为视图建立索引

Close Database All

对视图进行浏览,修改其中数据的操作同表的操作,观察数据变化的方法同上例。

3.6.3.3 对视图的一般命令操作

格式一：

Modify View <视图名>

功能:打开当前数据库视图名称为<视图名>的视图设计器

格式二:

Delete | Drop View <视图名>

功能:删除当前数据库名称为<视图名>的视图

对视图的其他操作与表相似。

例 3.73

Open Database Zy

Use 我的视图 1

&& 在当前工作区打开视图

Browse

&& 打开当前工作区中视图的浏览窗口

Drop View 我的视图 1

&& 从数据库中删除视图

习 题

1. 试对 Employee.dbf 分别排序。

(1) 将工资低于 1500 的职工按部门降序排序,新文件包含所有字段。

(2) 将职员按部门排序,当部门相同时则按工作时间升序排序。

2. 使用命令为 Student.dbf 建立一个结构复合索引文件,其中包括三个索引:

(1) 记录以学号升序排列,并且索引标记为普通索引型;

(2) 记录以性别降序排列,性别相同时按出生日期降序排列,并且索引标记为唯一索引型;

(3) 记录以性别降序排列,性别相同时按出生日期升序排列,并且索引标记为候选索引型。

3. 分别用以下方法查询学生的高等数学成绩。

(1) 顺序查询。

(2) 索引查询。

(3) 在数据工作期窗口建立关联后查询。

4. 用以下两种方法从 Student,Collegedepartment,Reportcard 三个表中列出每个学生的学号,姓名,系部名称,VB 程序设计成绩。

(1) 写出利用数据工作期窗口的操作步骤。

(2) 写出命令序列。

5. 试写出性别为男,并且不是团员的学生人数。

6. 根据 Reportcard 表的内容,算出每个学生的各门功课平均分、总分。

7. 根据 Student 和 Reportcard 两表进行汇总:

(1) 按系别汇总平均分。

(2) 按系别汇总各系人数。

8. 对于下列查询要求,分别写出查询的操作步骤及 SELECT-SQL 命令。

(1) 查询学生的学号,姓名,性别,是否团员。

(2) 查询学生的学号,姓名,系部名称。

(3) 查询 1986 年前出生并且是团员的学生学号,姓名,系部名称。

(4) 试算出 1986 年前出生的学生人数。

9. 写出 SELECT-SQL 命令序列。

(1) 查询高等数学成绩最高和最低的学生。

(2) 查询每个人每门功课成绩与平均成绩之差。

10. 试比较视图与 SELECT-SQL 查询的异同,从功能,存储,打开设计器的方法等方面来说明。

11. 建立一个数据库 Student,要求如下:

(1) 将表 Student,Collegedepartment,Reportcard 添加到 Stud 数据库中,并在三个表间建立合理的永久关系;

(2) 验证数据字典的使用方法;

(3) 建立一个视图该视图能完整的反映记录的全部信息。



第 4 章

结构化程序设计初步

前面各章节我们所进行的操作都是以交互式方式的,即在命令窗口逐条输入命令或通过选择菜单来执行 Visual FoxPro 6.0 命令。除此之外,我们还可以采用程序方式来调用 Visual FoxPro 6.0 的系统功能,以完成更为复杂的任务。

4.1 程序设计概述

尽管通过 VFP 用户界面环境可以交互式地完成许多数据库管理任务,但要充分发挥 VFP 的强大功能,便于用户使用利用 VFP 开发出来的应用软件,软件设计人员还需要掌握 Visual FoxPro 程序设计语言。

4.1.1 程序

程序是为了达到某一目标,由一系列相应语句或命令有序排列而成的语句序列。程序一般存储在磁盘上,以程序文件名(如 lx.prg)的形式体现出来。Visual FoxPro 程序是包含一系列命令的文本文件,也可简单地认为程序就是命令的集合。如下例中命令的集合就是一个程序:

例 4.1 下面是一个简单的程序文件内容,其功能是打开数据库中的表,并打开浏览窗口浏览,当关闭浏览窗口后,关闭数据库。该程序文件名为 EmpBrow.prg。

```
* EmpBrow.prg
Open Database Zy          &&. 打开数据库
Use Employee              &&. 打开表
Browse                    &&. 浏览表
Close Database            &&. 关闭数据库和表
Return
```

如果在命令行中执行:Do EmpBrow 命令时,计算机将自动完成程序文件 EmpBrow.prg 内指定的命令。因此,与在命令窗口中直接逐条的执行命令的方式相比,程序文件被执行后有如下优点:

(1)自动执行程序文件中已经事先编制好的命令序列,省去在命令窗口中逐行输入执行命令的麻烦,并且也节约了宝贵的时间;

- (2)可以在不同的文本编辑器中,建立和修改程序文件;
- (3)程序文件,可重复运行;
- (4)可以在一个程序文件中调用其他程序文件;
- (5)一些结构化程序设计命令只能在程序中使用,不能在命令窗口中使用。

4.1.2 结构化和可视化程序设计

所谓结构化程序设计,就是采用自顶向下逐步求精的设计方法,通过顺序、分支和循环三种单入口单出口的基本程序结构进行的程序设计。结构化程序设计是面向过程、按顺序进行的机制。但是随着软件规模的扩大、功能提高和需求变化,结构化程序设计的开发效率和维护问题也比较突出。

随着窗口化、多任务的 Windows 操作系统的普及,Visual FoxPro,Visual Basic 等也迅速发展起来。Visual FoxPro 是在 Dbase,Foxbase,Foxpro 的基础上建立的,Visual 意为“可视化的”,它是指一种开发图形用户界面的方法。所以 Visual FoxPro 是基于 FoxPro 面向对象的可视化数据库程序设计语言。在 Visual FoxPro 中,一方面继承了结构化程序设计的特点,另一方面在其编程系统中采用了面向对象、事件驱动的编程机制。

4.2 程序文件的建立、修改与运行

要进行 VFP 程序设计,首先必须掌握 VFP 中程序文件的编辑和运行。建立和运行 VFP 程序文件的方式有:命令方式、菜单方式和项目管理器方式等。

4.2.1 程序文件的建立

4.2.1.1 命令方式

格式:Modify Command [<程序文件名>]

功能:打开一个程序编辑窗口,以使用户录入或修改程序文件。

参数说明:

1. <程序文件名>缺省,自动以“程序 1”或“程序 2”等作为新建程序文件名称;
2. 有<程序文件名>选项时,若该文件不存在,则打开以该文件名为标题的文本编辑窗口,新建该文件;若该文件已经存在,则打开以该文件名为标题的文本编辑窗口,调入源程序内容,以供用户修改该文件,如图 4.1 所示;



图 4.1 程序编辑窗口

3. <程序文件名>可以包含路径和扩展名 prg。如果缺省路径,则默认当前路径,如果缺省扩展名,则在存盘时,系统自动加上 prg 扩展名;

4. 保存程序。程序录入或修改完毕,按“Ctrl+W”存盘退出,或选择“文件”菜单中的“保存(S)”,或单击工具栏中的“保存”按钮,然后关闭文本编辑窗口退出。若用户要关闭一个没有保存的程序,则会弹出如图 4.2 所示的提示框,询问用户是否存盘,单击“是”存盘,“否”不存盘,“取消”则放弃本次操作。若用户保存一个尚未命名的程序,则会打开“另存为(A)...”对话框,提示用户为程序指定程序名。



图 4.2 “是否存盘”提示框

例 4.2 建立上例中的程序文件。

Modify Command D:\VFP\EmpBrow

在命令窗口执行上述命令后,将打开标题为 Empbrow.prg 的文本编辑窗口,用户录入该程序的具体内容后,存盘退出。

4.2.1.2 菜单方式

使用菜单方式创建程序文件的步骤为:

1. 单击菜单“文件/新建”命令,或单击工具栏中的“新建”按钮,或者直接按热键“Ctrl+N”,弹出“新建”对话框;
2. 在“新建”对话框中,选择“程序”后,单击“新建文件”按钮,弹出文本编辑窗口;
3. 在文本编辑窗口中录入程序内容;
4. 完成程序录入后,存盘。

4.2.1.3 项目管理器方式

使用项目管理器方式创建的程序文件会自动成为项目的组成部分,其创建的步骤如下:

1. 打开“项目管理器”;
2. 展开“代码”分支,选择“程序”图标,如图 4.3 所示;



图 4.3 项目管理器

3. 单击“新建”按钮,弹出文本编辑窗口;
4. 在文本编辑窗口中完成程序录入后,存盘。

4.2.2 程序文件的修改

4.2.2.1 命令方式

其命令格式同程序文件创建的命令格式,但用该命令格式时,<程序文件名>中指定的程序文件必须存在,否则,该命令为创建一个新的程序文件命令。

例 4.3

Modify Command D:\VFP\EmpBrow && 在文本编辑窗口中打开程序文件

Modify Command D:\VFP\myprog1.prg && 在文本编辑窗口中创建一个新的程序文件

4.2.2.2 菜单方式

使用菜单方式修改程序文件的步骤为:

1. 单击菜单“文件/打开”命令,或单击工具栏中的“打开”按钮,或者直接按热键“Ctrl+O”,弹出“打开”对话框;
2. 在“文件类型”下拉列表框中选择“程序”,然后在“查找范围”列表框中选择文件的路径和程序名,也可直接在“文件名”文本框中直接输入程序文件的全称,单击“确定”按钮,弹出文本编辑窗口;
3. 若输入的程序文件名在当前路径下不存在,则打开以该文件名为标题的文本编辑窗口,新建该文件。在此的“打开”菜单相当于执行命令 Modify Command;
4. 在文本编辑窗口可任意修改程序内容,然后存盘退出。

4.2.2.3 项目管理器方式

使用项目管理器方式修改程序文件的方法与使用项目管理器创建程序文件的方法相似,不同的是在展开的程序图标下,选择要修改的程序文件后,单击“修改”按钮,如图 4.4 所示。该方法只适用于修改项目中的程序文件。



图 4.4 在项目管理器中选择程序文件

4.2.3 程序文件的运行

VFP 程序文件在被执行的过程中,会被自动编译成目标程序文件(目标程序文件的主名与程序文件相同,其扩展名为 fxp),而 VFP 实际运行的是目标程序文件,该文件是一个紧凑的非文本文件,其运行速度快,并可起到对源程序加密的作用。

当用户编辑好程序文件,并且至少运行一次后,再删除程序文件,若此时再次运行该程序,系统运行的是目标程序文件,其执行效果同直接运行程序文件的效果,然而,由于直接运行目标程序文件时少了一个编译过程,所以其运行速度要比直接运行程序文件快。因此,人们往往在完成一个项目后,删除程序文件,而仅仅只保留编译后的目标程序文件。

这里需要特别指出两个问题:一是当同名的程序文件存在时,在执行运行程序命令时,计算机总是优先运行程序文件而不是目标程序文件,若要运行指定的目标程序文件,则必须在运行命令中指明目标程序文件的扩展名 fxp。另外当多次重复运行程序文件时,计算机总是用最后一次运行程序文件时编译的目标程序文件去自动覆盖以前编译的目标程序文件。

例 4.4 运行 EmpBrow. prg 程序。

DO D:\VFP\EmpBrow. prg

&.&. 执行命令前,将程序文件编译成目标程序文件 EmpBrow. fxp,然后再执行

Rename D:\VFP\EmpBrow. fxp to D:\VFP\EmpBrow1. fxp

Do D:\VFP\EmpBrow1

&.&.EmpBrow1. prg 文件并不存在,但该命令的执行结果同:DO D:\VFP\EmpBrow. prg

4.2.3.1 命令方式

格式:

Do <程序文件名>

功能:运行指定的程序文件。

说明:在命令窗口输入 DO 命令,单击“回车”键后执行该命令。执行该命令时,可省略程序文件的扩展名。

例 4.5 运行 EmpBrow. prg 程序另一种形式。

由于本书约定目录 D:\VFP 为 VFP 的默认目录,所以该命令可以写为:

DO EmpBrow

4.2.3.2 菜单方式

单击菜单“程序/运行”命令,或按热键“Ctrl+D”,弹出如图 4.5 所示的“运行”对话框,从中选择要执行的程序文件,然后双击程序文件名或单击“运行”按钮即可。

4.2.3.3 项目管理器方式

在“项目管理器”中选择要运行的程序文件名,然后单击“运行”按钮即可。



图 4.5 “运行”对话框

&.&. 执行默认位置的程序文件

4.2.3.4 单击工具栏“运行”命令按钮方式

在程序文件的文本编辑窗口处于打开状态,并且是当前窗口时,单击工具栏的“运行”按钮也可以直接运行程序文件。

4.3 基本输入输出

程序设计中最基本的操作就是输入输出。输入让程序获取数据,输出则将程序的执行结果显示出来。输入语句的本质就是给变量赋值,它可以接收表达式的值,也可以接收用户从键盘输入的数据。而输出语句就是将数据输出到屏幕或打印机。

4.3.1 基本输入

程序中变量的赋值方式有两大类:一类是利用前面介绍的赋值命令语句给变量直接赋值,另一类是通过交互式键盘输入,获取由用户输入的新值。所谓交互式键盘输入,就是用户通过键盘输入一个常量,并让变量接收这个常量的值。Visual FoxPro 提供的交互式键盘输入命令主要是:INPUT、ACCEPT、WAIT 和 @...GET。

4.3.1.1 INPUT 命令

格式:Input [<字符表达式>] To <内存变量>

功能:从键盘接收输入的数据,并存入指定的内存变量中。

参数说明:

1. 通过键盘给内存变量输入数据,如果内存变量没有定义,则 Input 命令自动建立;
2. <字符串表达式>的内容将在输入数据时,作为提示信息显示在屏幕上;
3. Input 命令执行时,输入的数据值的类型决定了内存变量的类型,即如果输入的是数值型,则建立一个数值型内存变量,如果输入的是字符串,则建立一个字符型内存变量,依此类推。

例 4.6

```
Input To Name           &&. 执行该命令后,计算机等待用户输入数据,如输入:"陈国庆"
?Name                   &&. 输出"陈国庆"
Input "Enter Age: " To Age &&. 如输入:40
?Age                     &&. 输出:40
```

交互式键盘输入命令通常不在命令行中直接应用,而是在程序中应用,所以,我们可以将上述四条命令写入一个简单的程序文件中,如 ExempInput.prg,其文件内容如下:

```
* ExempInput.prg
Input To Name
?Name
Input "请输入年龄:" To Age
?Age
Return
```

在建立好上述程序文件后,在命令行中执行命令:Do ExempInput,计算机就会自动

的运行上述四条命令,无论是命令方式还是程序方式运行上述命令时,我们可以发现运行第一个输入命令时,计算机没有任何提示信息,而运行第二个输入命令时计算机会在主程序窗口显示提示信息“请输入年龄:”,并且光标紧随其后,等待用户输入数据。

4.3.1.2 Accept 命令

格式:Accept [<字符表达式>] To <内存变量>

功能:从键盘接收输入的字符串,并存入指定的内存变量中。

说明:

1. 只能通过键盘接受没有定界符的字符型数据,当输入的数据有定界符时,定界符也将成为字符串本身的一部分;
2. <字符串表达式>的内容将在输入数据时,作为提示信息显示在屏幕上;
3. 如果未键入任何数据就直接单击“回车”键,则内存变量接收一个空串;
4. 如果未键入任何数据就直接单击 ESC 键,则内存变量接收一个空串。但是,当设置 Set Escape On 时,按下 ESC 键,则会出现中止程序运行的提示对话框;如果设置 Set Escape Off 时,按下 ESC 键,则不会出现中止程序运行的提示对话框;
5. Accept 与 Input 是不同的,因为使用 Accept 时,所有数据都被当作字符处理,而且在字符数据的两端不需要使用定界符,它可以接收空串。而 Input 可接收任何类型的表达式,并且这个表达式的值赋给内存变量同时也决定了内存变量的类型,而且它必须接受一个有意义的值即不能只有回车字符。

例 4.7 ExempAccept. prg 文件内容如下:

```
* ExempAccept. prg
```

```
Clear
```

```
Clear All
```

```
Set Escape Off           &&. Set Escape On
```

```
Accept "A=" To A         &&. Input "A=" To A
```

```
Display Memory Like A *  &&. 显示所有以 a 开头的内存变量
```

```
Return
```

运行上述程序文件,在运行的过程中练习输入数据、不输入数据直接单击“回车”键和不输入数据直接单击“Esc”键三种情况下程序的运行情况;分别用注释中的命令来替换形成四种程序组合,重复上述三种情况,看看有什么不同。

4.3.1.3 WAIT 命令

格式:

Wait [<字符表达式>] [To <内存变量>][Windows [At <行>,<列>]]; [Timeout <等待时限>]

功能:显示一条信息并暂停系统的执行,直到按下任意键或单击鼠标。

参数说明:

1. 如果含有“To <内存变量>”子句,则等待用户键入一个字符(不需回车),键入后机器立即继续运行。内存变量的类型是字符型,其值是一个字符;
2. <字符表达式>用来指定等待键盘输入时的提示信息,如果省略该参数,则默认

的提示信息为“按任意键继续...”；

3. 当命令中指定关键字 Window[At <行>,<列>]时,则指定提示信息在屏幕上出现的位置,若省略参数,等待提示信息将显示在屏幕右上角的一个窗口中;

4. Timeout 子句用来设定等待输入的时间(秒)限制,如果在规定的时间内没有任何输入,也没有按下鼠标键,等待终止,Wait 返回一空串,程序继续执行。Timeout 子句必须是 Wait 命令中的最后一个子句,放在任何其他位置都将导致语法错误。

例 4.8 Exem Wait.prg 文件内容如下:

```
* ExempWait.prg
Clear
Wait
Wait "继续吗?"
Wait "请选择 1、2、3" To C
Wait "请选择 A、B、C" To CC Window At 10,10 Timeout 10
Display Memory Like C*
Return
```

在上述命令中有两条命令是让用户选择按键,但用户不使用提示的按键,而单击其他按键又会是什么情况呢?有什么方法可以让用户只能选择规定的键呢?具体方法我们将在以后介绍 Do While 循环语句时再加以介绍。

4.3.1.4 格式输入输出命令@... Say/Get

格式:

```
@<行>,<列> [Say <表达式 1> [Picture <输入掩码>]];
    [Get <变量> [Picture <输入掩码>] [Default <表达式 2>]]
...
READ
```

功能:在指定的行列位置编辑变量。

参数说明:

1. 屏幕坐标由行、列组成,屏幕左上角的坐标为(0,0),然后依次向下、向左递增行列。<行>,<列>;表示将当前光标定位在指定<行>和<列>位置;

2. Say <表达式 1>在指定位置输出<表达式 1>的内容,但同 Get 子句配合使用时,通常是对输入进行提示的说明;

3. Get 子句中的变量数据在<表达式 1>显示完成后的位置显示,若缺少 Say 子句,则在指定位置显示;

4. 有 Get 子句时,变量必须有初始值,初始值的赋值可以在该语句之前,也可以在该语句之中利用选项 Default <表达式 2>实现;

5. Get 命令在指定位置显示文本框(按变量的宽度),若不执行 Read 命令,则文本框不接受从键盘输入的内容,变量的值仍为默认值;

6. 执行 Read 命令后,将激活 Read 命令行前 Get 命令中的变量,此时光标会自动依次停留在 Get 命令指定的变量文本框中,等待接收用户从键盘编辑、输入数据,单击“回

车”键完成一个变量的数据输入；

7. Say 和 Get 子句中的选项 Picture 为输入输出控制符,其用法参阅前面数据库中数据字典介绍的输入掩码的用法；

8. 在此命令中若只省略 Get 子句,则该命令只有输出功能;若只省略 Say 子句,则该命令只有输入功能,当两个子句同时省略时,该命令则仅仅只具有重新设置光标位置的功能；

9. 当该命令中的 Get 子句中使用的变量为当前表的字段变量时,该命令还具有直接修改字段内容的功能；

10. 在输出定位过程中,还可以利用下列函数确定输出的位置：

Row() : 返回当前屏幕光标所处的行位置

Col() : 返回当前屏幕光标所处的列位置

Prow() : 返回打印机当前打印所处的行位置

Pcol() : 返回打印机当前打印所处的列位置

例 4.9 ExempGet1. prg 文件内容如下：

* ExempGet1. prg

* 建立四个变量,进行编辑输入,了解变量赋值的方式,及 Say 和 Get 子句的使用

Clear

C="This is a exemple !"

@ 2,20 Say "Input Logic" Picture "@ !" Get L Default . T.

@ Row()+2,20 Say "Input Charstring" Get C Picture "@ XXXXXXXXXXXXXXXXXXXXXXXX"

@ Row()+2,20 Say "Input Number" Picture "@ !";

Get N Picture "9,999.99" Default 2222.22

@ Row()+2,20 Say "Input Date" Get D Default Date()

Read

@8+4,20

@ Row()+2,25 Say "逻辑值:"

@ Row(),Col() Say L

@ Row()+2,25 Say "字符值:"

@ Row(),Col() Say C

@ Row()+2,25 Say "数值:"

@ Row(),Col() Say N

@ Row()+2,25 Say "日期:"

@ Row(),Col() Say D

Return

程序的运行结果如图 4.6 所示：



图 4.6 ExempGet1.prg 的运行结果

例 4.10 利用格式输入输出命令@... Say/Get 直接向数据库表 Employee 添加一条新记录。

```
* ExempGet2.prg

Clear

Use Employee

Append Blank

@ 2,20 Say "身份号:" Get 身份号 Picture "99999999"

@ Row()+2,20 Say "姓名" Get 姓名 Picture "xxxxxxx"

@ Row()+2,20 Say "性别" Get 性别 Picture "xx"

@ Row()+2,20 Say "年龄" Get 年龄

@ Row()+2,20 Say "民族" Get 民族

@ Row()+2,20 Say "工作时间" Get 工作时间

@ Row()+2,20 Say "婚否" Get 婚否

@ Row()+2,20 Say "工资" Get 工资

@ Row()+2,20 Say "电话" Get 电话

@ Row()+2,20 Say "简历" Get 简历

@ Row()+2,20 Say "照片" Get 照片

Read

List

Use

Return
```

上述程序每运行一次,用户就可以直接向表 Employee 添加一次记录,备注型和通用型字段的编辑方式与在表中的编辑方式相同。利用格式输入输出命令@... Say/Get 可以直接向字段变量输入新值,所以结合指针定位,可以完成修改指定记录的数据。而 Input 和 Accept 命令只能将输入的值赋给内存变量,若要利用这两条命令完成记录的修改和追加,则必须结合替换命令 Replace 完成。

在程序的运行过程中,除了可以利用上述命令方式进行交互式的数据输入外,我们还可以利用表单中的文本框、组合列表框等方式进行交互式的数据输入,这些方法将在第七章表单控件设计中介绍。

4.3.2 基本输出

程序命令中的基本的输出命令有 ? 和 ?? 命令和在格式输入输出命令 @... Say/Get, ? 和 ?? 命令已经在第一章的表达式一节中作了介绍,格式输入输出命令 @... Say/Get 若省略 Get 子句,则该命令的功能也仅仅只有输出功能。所以关于这两条命令就不再作说明,这里我们主要介绍一下信息对话框函数 MessageBox 函数

函数格式:

MessageBox(<信息内容表达式>[,<对话框类型表达式>[,<对话框标题表达式>]])

功能:显示一个自定义的信息对话框,单击一次对话框的按钮后,函数返回一个值。

参数说明:

- 1. <信息内容表达式>:必选项,用于指定在信息对话框中显示的信息文本(由字符串或字符串表达式组成),其最大长度为 1024 个字符,如果超过该长度,则多余字符将自动被截掉。如果信息框的内容超过一行,可在每一行之间用回车符(chr(13))、换行符(chr(10))、或回车符与换行符的组合(chr(13))+(chr(13))将各行分开;
- 2. <对话框类型表达式>:该参数项为可选项,其值通常是由 3 个部分相加而得到的一个整型值,默认值为 0。用于指定信息框中命令按钮的数目、形式和使用的图标样式与缺省按钮等。<对话框类型表达式>各组成部分的值及其含义如表 4.1 所示;

表 4.1 信息对话框类型表达式各组成部分

组成部分	值	功能说明
命令按钮数目及形式	0	只有“确定”按钮
	1	“确定”和“取消”按钮
	2	“终止”、“重试”和“忽略”按钮
	3	“是”、“否”和“取消”按钮
	4	“是”和“否”按钮
	5	“重试”和“取消”按钮
图标样式	16	红色叉号错误图标(叉号)
	32	蓝色问号图标(问号)
	48	黄色惊叹号图标(惊叹号)
	64	蓝色 i 图标(信息图标)
默认按钮	0	第 1 个按钮是缺省值
	256	第 2 个按钮是缺省值
	512	第 3 个按钮是缺省值

表 4.2 函数的返回值表

返回数值	按下按钮
1	确定
2	取消
3	终止
4	重试
5	忽略
6	是
7	否

3. <对话框标题表达式>:指定对话框标题栏的标题,若省略此项,系统给出默认的标题:Microsoft Visual FoxPro

4. 当单击信息对话框中的某个按钮,并关闭对话框时,函数将返回一个数值,其返回值由该按钮的类型决定,如表 4.2 所示。

例 4.11 在命令窗口中键入下列命令: ? MessageBox("请选择按钮")
将弹出如图 4.7 所示的对话框,该对话框只有“确定”按钮,其函数返回值为 1。



图 4.7 MessageBox("请选择按钮")
产生的信息框



图 4.8 MessageBox("请选择按钮",2+16+256,
"MessageBox 练习")产生的信息框

例 4.12 在命令窗口中键入下列命令,将产生如图 4.8 所示的信息对话框,单击“终止”、“重试”或“忽略”按钮时,函数将返回 3,4,5 三个值中的一个。

? MessageBox("请选择按钮",2+16+256,"MessageBox 练习")

4.4 程序的控制结构

在结构化程序设计的编程语言中都包含着三种基本的程序结构,即顺序结构、分支结构和循环结构。顺序结构是最简单的程序结构,它是按命令在程序中出现的先后位置依次执行命令,这种顺序结构一般只能用来解决最简单的问题,绝大多数问题还要用到选择结构和循环结构来加以解决,本节将分别这三种基本结构语句。

4.4.1 顺序结构

顺序结构的程序在运行时是按照语句在程序中排列的先后次序,一条接一条的依次顺序执行的,顺序结构在程序设计中是一种最简单的基本结构。前面介绍的例 4.6,例 4.7,例 4.8,例 4.9,例 4.10 中涉及的程序都是一种简单的顺序结构,他们的共同特点都是依次去执行程序中的命令语句,直到程序中所有命令语句执行完毕为止。

例 4.13 下列程序文件 ExempSerial. prg 的作用就是打开一个数据库,并浏览其中的一个表的内容,关闭浏览窗口后,关闭数据库,这个小程序也是一个典型的顺序结构:

```
* ExempSerial. prg
Note 顺序结构示例      &&Note 与 * 的作用一样在语句的最前面时,表示该语句为注释
Open Database Zy
Use Employee
Browse
Close Database          && 关闭当前数据库和其表
Return
```


4.4.2 分支结构

顺序结构的程序运行时,总是按照命令语句在程序中的先后次序,一条一条的依次顺序执行,即程序的执行流程总是单一直线式的。而分支结构语句在程序中是根据一定的条件选择执行或跳过程序语句中的一部分命令语句。根据一次条件判断后分支多少,分支结构又分为双分支(IF 语句)和多分支(CASE 语句)两类。分支结构亦称选择结构。

4.4.2.1 双分支语句(IF 条件结构)

格式

If <条件>

[<语句序列 1>]

[Else

[<语句序列 2>]]

Endif

功能:首先计算 IF 后“条件”表达式的逻辑值,若为“真”(即满足条件),执行“语句序列 1”,然后执行 Endif 后的语句;若为“假”(不满足条件),则跳过“语句序列 1”,执行 Else 后面的“语句序列 2”,然后执行 Endif 后的语句。流程如图 4.9 所示。

参数说明:

1. If 条件结构是一个整体,必须以 If<条件>开始,以 Endif 结束。Endif 表示程序跳出 If 条件结构,转而继续执行 Endif 后面的语句;

2. 语句序列可以是一条或多条语句;

例 4.14 无论用户输入的数是正数还是负数,均以正数形式显示。

* ExempIf1. Prg

Clear

@10,10 Say "请输入一个数:" Get X Default 100 Picture "999"

Read

If X>=0

@12,10 Say "您输入的数是:"

@Row(),Col() Say X

Else

@12,10 Say "您输入的数是:"

@Row(),Col() Say -X

Endif

Return

3. Else 子句可以省略,当省略该子句,并且当<条件>为逻辑假值时,表示程序直接跳出 If 条件结构,而不执行其中的语句序列;

4. 为阅读和理解程序的层次关系,块结构中的“语句序列”应向后缩进,如上例所示;

5. 当 If 条件结构的语句序列中包含有另一个 If 条件结构时,我们称这种情况为 If

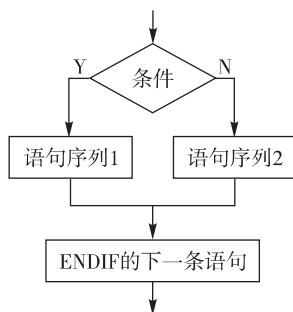


图 4.9 IF 语句流程

的嵌套。If 语句在嵌套使用时要注意其结构的完整性,即每个 If 仅仅只和自己的 Endif 相配对,相互之间不允许有交叉。其嵌套形式如图 4.10 所示,从该图中我们可以发现 If 的嵌套既可以在 If 语句后的语句序列中,也可以在 Else 子句的语句序列中。利用 IF 语句的嵌套形式,我们可以实现多分支的选择,但为了避免不必要的麻烦,VFP 还提供了更加清晰的多分支选择 Case 语句。

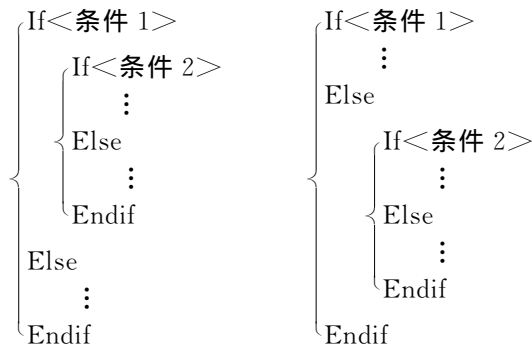


图 4.10 If 条件语句的嵌套

6. 当 If 语句的语句序列比较简单时,VFP 还提供了 If 函数,利用该函数也可以实现 If 语句的简单功能:

函数格式:

If(<条件>,<结果表达式 1>,<结果表达式 2>)

功能:该函数返回一个值,如果条件为真,则返回<结果表达式 1>的值,否则返回<结果表达式 2>的值。

例 4.15 输入一个数,并提示该数是正数还是负数。

* ExempIf1.Prg

Clear

@10,10 Say "请输入一个数:" Get X Default 100 Picture "99999"

Read

@12,10 Say If(X>=0,"您输入的数是一个正数!", "你输入的数是一个负数!") +
"其值为:" + Alltrim(Str(X))

Return

如果使用 If 语句来实现上例的功能,其程序中的语句要比上例要多一些,其形式也要复杂一些,请大家用 If 语句来写一段程序来实现上例的功能。

例 4.16 输出学生的姓名和性别。年龄大于等于 20 输出大,年龄小于 20 且大于等于 18 的输出为中,年龄小于 18 输出小。

* ExempIf2.Prg

Clear

Sname=""

Sold=0

@10,10 Say "输入学生姓名:" Get Sname Picture "xxxxxx"

@Row()+2,10 Say "输入学生性别:" Get Ssex Default "男" Picture "xx"

```
@ Row()+2,10 Say "输入学生年龄:" Get Sold Picture "999"
Read
If Sold>=20
    @16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"大"
Else
    If Sold>=18      &&. 该条件与 Sold<20 And Sold>=18 等价
        @16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"中"
    Else
        @16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"小"
    Endif
Endif
Return
```

4.4.2.2 多分支语句

格式:

```
Do Case
    Case <条件 1>
        [<语句序列 1>]
    [Case <条件 2>
        [<语句序列 2>]]
    ⋮
    [Case <条件 n>
        [<语句序列 n>]]
    [Otherwise
        [<语句序列 n+1>]]
Endcase
```

功能:依次判断是否符合其中的某一种情况,在执行完符合条件的语句序列后,跳出 Do Case 语句,执行 Endcase 后的语句,语句的执行流程如图 4.11 所示。

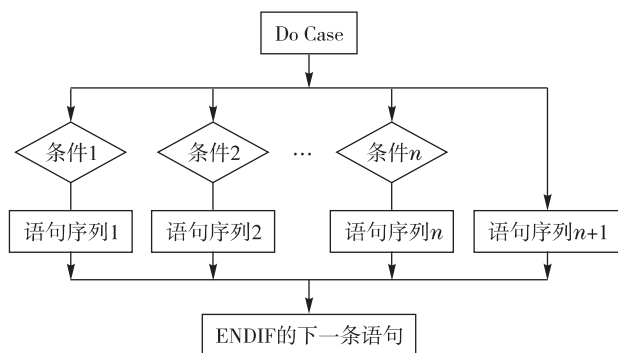


图 4.11 Do Case 语句流程

说明:

1. 执行该语句时,依次判断 Case 后的条件,当某个条件为“真”值时,则执行完该 Case 相应的语句序列后跳出 Do Case 语句,执行 Endcase 后的语句;若所有条件都不满足,并且有 Otherwise 子句,则执行完其后的语句序列 n+1 后跳出 Do Case 语句,执行 Endcase 后的语句;若所有条件都不满足,并且没有 Otherwise 子句,则直接跳出 Do Case 语句,执行 Endcase 后的语句;

2. 执行该语句时,只能执行 Case 子句条件为真值时语句序列。如果有多个 Case 子句的条件为真值,则仅执行第一个满足条件的 Case 语句序列,后面满足条件的 Case 语句序列将被忽略;

3. Do Case 语句主要用于多个选择(3 个以上)的情况。对于并列的条件,应将最常用的选择条件放在前面,把不常用的条件放在后面,这可以提高程序的执行效率。

例 4.17 采用 Do Case 语句,改写上例程序。

* ExempCase1. Prg

Clear

Sname=""

Sold=0

@10,10 Say "输入学生姓名:" Get Sname Picture "xxxxxx"

@Row()+2,10 Say "输入学生性别:" Get Ssex Default "男" Picture "xx"

@ Row()+2,10 Say "输入学生年龄:" Get Sold Picture "999"

Read

Do Case

Case Sold>=20

@16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"大"

Case Sold>=18 && 该条件与 Sold<20 And Sold>=18 等价

@16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"中"

Otherwise

@16,10 Say "姓名:"+Sname+Space(2)+"性别:"+Ssex+Space(2)+"小"

Endcase

Return

从上例我们可以看出,使用 Do CASE 语句可以让多分支选择程序的逻辑更加清楚、简洁。

例 4.18 输入一个成绩,判断该成绩属于哪个分数段。

* ExempCase2. Prg

Clear

@ 10,10 Say "输入成绩:" Get Attainment Picture "999" Default 0

Read

Do Case

Case Attainment<60

X="您输入的成绩属于小于 60 分的分数段!"

```

Case Attainment<70
    X="您输入的成绩属于大于等于 60 分、小于 70 分的分数段 !"
Case Attainment<80
    X="您输入的成绩属于大于等于 70 分、小于 80 分的分数段 !"
Case Attainment<90
    X="您输入的成绩属于大于等于 80 分、小于 90 分的分数段 !"
Otherwise
    X="您输入的成绩属于大于等于 90 分的分数段 !"
Endcase
@ 12,10 Say X
Return

```

4.4.3 循环结构

如果要求输出 $Y=\sin(X)$ 在 $0^\circ \leq X \leq 180^\circ$ 范围内,每隔 1° 的所有值。若用顺序结构来完成求值的过程,则要写出 300 多条语句,如:

```

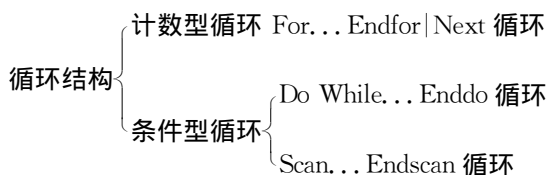
X=0
? Sin(X)
X=3.14159/180
? Sin(X)
:
X=3.14159
? Sin(X)

```

从上面一段程序代码中,我们不难发现:计算和显示所使用的函数和形式都是相同的,不同的只是 X 的值一直在变化,但 X 值的变化具有一定的规律性(每次增加 1°)。显然用这种顺序结构来解决重复处理的问题是极其不便的。VFP 提供了解决这个问题的方法:利用循环结构来解决这个问题。

循环结构也称为重复结构,是指程序在被执行的过程中,其中的某一段代码被重复的执行若干次。被重复执行的代码段,通常称之为循环体。

VFP 提供了实现循环结构的三种循环语句,循环结构及相应语句表示如下:



在以上三个循环体中均可以插入 Exit 语句来强制程序跳出循环,也可以插入 Loop 语句来强制要求程序进入下一循环。Exit 和 Loop 为循环辅助语句只能在循环体中使用。

4.4.3.1 FOR 循环

格式

```
For <内存变量> = <初值> To <终值> [Step <步长>]
```

```
[<循环体语句序列 1>]  
[Exit]  
[<循环体语句序列 2>]  
[Loop]  
[<循环体语句序列 3>]
```

Endfor | Next

功能:重复执行 For 和 Endfor 或 Next 之间的循环体,重复执行的次数由循环内存变量来控制。该语句主要用于已知循环次数的循环控制。

参数说明:

- 1. 初值、终值和步长(也称增量)为数值表达式(也可以是数值型常量或变量),当缺省 Step 时,默认的步长为 1;
- 2. For 语句是循环的起始位置,循环结束语句可以是 Endfor,也可以是 Next,两者任选其一;
- 3. 循环体是指 For 和 Endfor(或 Next)之间的语句体,可以是一条或多条 VFP 语句;
- 4. 当循环体中存在 Exit 语句,且执行到该语句时,循环将会被强制中断,从 Exit 语句处直接跳出循环,并执行 Endfor(或 Next)语句后的语句;
- 5. 当循环体中存在 Loop 语句,且执行到该语句时,循环将会被强制从 Loop 语句处直接跳到 Endfor(或 Next)语句处,进行下一次循环;
- 6. Exit 和 Loop 可以出现循环体中的任何位置,但通常是放在判断语句中,当满足条件时,则执行对应的 Loop 或 Exit 语句;
- 7. For 循环也称计数型循环,它属于前测试的循环类型。这里我们利用下列常用的循环结构的一般格式,来说明循环的运行机理:

```
For I=A To B Step C  
  <循环体>  
Endfor | Next
```

其中 I 代表循环变量,A、B、C 分别代表“初值”、“终值”和“步长”。该循环的执行过程为:

- 第一步:执行 For 语句时,首先记下 A、B、C 的值(如为表达式则先计算),并将初值 A 赋给循环变量 I;
- 第二步:将循环变量的值与终值 B 比较,如果 I 的值未超过终值 B,则转第三步;如果 I 之值超过终值 B,则转第六步;
- 第三步:依次执行循环体内各语句;
- 第四步:执行 Endfor(或 Next)语句,循环变量 I 按步长 C 增值,即 $I+C \rightarrow I$;
- 第五步:返回第二步继续执行;
- 第六步:跳出循环结构,继续执行 Endfor(或 Next)语句的下一条语句。

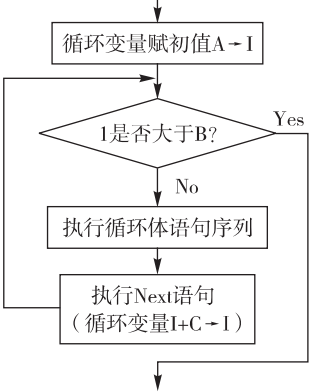


图 4.12 For...Next 循环的执行过程

For...Next 循环的执行过程如图 4.12 所示。

8. 当循环变量的值等于终值时仍执行循环体,循环的终止条件是“循环变量的值超过终值。”所谓“超过”终值是指沿变化的方向超过终值,即“超过”有两层含义:一是当步长为正数(即递增)时,“超过”就是“大于”的意思;二是当步长为负数(即递减)时,“超过”就是“小于”的意思。如果步长为“零”,则循环变量永远不起变化,要么循环 0 次;要么循环永不终止,也称为“死循环”或“永久循环”,此时可按 Esc 键终止执行。

例 4.19 For 循环执行过程举例。

* ExempFor1. Prg

For I=1 To 3

?? I

Next

Return

运行结果:

1 2 3

程序运行过程分析:

1. 将 1 赋给变量 I,比较变量 I 的值是否小于等于终值,由于 I 的值小于终值 3,故应执行一次循环体(?? I 语句),输出 I 的当前值 1;
2. 执行 Next 语句,I 的值在原有值的基础上加步长,变为 2;
3. 比较变量 I 的值是否小于等于终值,由于 I 的值小于终值 3,因此,又执行一次循环体,输出 I 的当前值 2;
4. 执行 Next 语句,I 的值在原有值的基础上加步长,变为 3;
5. 比较变量 I 的值是否小于等于终值,由于 I 的值等于终值 3,因此,还应执行一次循环体,输出 I 的当前值 3;
6. 执行 Next 语句,I 的值在原有值的基础上加步长,变为 4;
7. 比较变量 I 的值是否小于等于终值,由于 I 的值超过(在此大于)终值 3,故不再执行循环体了,而跳出循环。本循环一共执行了 3 次(而不是 2 次)循环体,如图 4.13 所示。

例 4.20 计算 $\text{Sum}=1\cdot 1+2\cdot 2+3\cdot 3+\dots+N\cdot N$ 的值,但要求当 Sum 的值第一次大于 100 时退出循环。

* ExempFor2. Prg

Clear

Sum=0

For I=1 To 100

Sum=Sum+I*I

If Sum>100

Exit

Endif

Endfor

?Sum=" ,Sum

Return

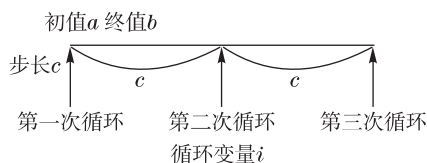


图 4.13 循环变量 I 的变化过程

运行结果：

Sum= 140

例 4.21 计算 $\text{Sum}=2^2+4^4+6^6+\dots+N^N$ 的值,但要求显示每次求和的值,当 Sum 的值第一次大于 100 时退出循环。

* ExempFor3. Prg

Clear

?"程序将被执行一遍！其执行结果如下:"

Sum=0

For I=2 To 100 && 若将 For 循环的步长设置为 2,这段程序是否可以改的更加简洁？

 If Mod(I,2)=0 && 判断 I 是否为偶数

 Sum=Sum+I*I

 * ?Sum=",Sum

 If Sum>100

 Exit

 Endif

Else

 * ?Sum=",Sum

Loop

 * ?Sum=",Sum

Endif

 * ?Sum=",Sum

Endfor

?"Sum=",Sum

Return

运行结果：

Sum= 120

上例中我们可以省略 Else Loop 子句其运行效果相同,但其运行的机理却不相同,当外层 If 条件结构的结束语句 Endif 与 For 循环的结束语句 Endfor 之间还存在有语句序列时,Else Loop 子句省略和保留的运行效果将大不相同。因为,若省略 Else Loop 子句时,外层 If 条件结构的结束语句 Endif 与 For 循环的结束语句 Endfor 之间的语句序列在循环过程中,将始终被执行;而不省略 Else Loop 子句时,外层 If 条件结构的结束语句 Endif 与 For 循环的结束语句 Endfor 之间的语句序列在循环过程中,只有外层 If 条件为真值(偶数)时,才会被执行。

在上例中共有五处输出语句 ?Sum=",Sum,其中有四处用注释 * 标明,请大家试着运行五次,每次只让其中的一个输出语句被执行,看看运行结果有什么不同。为便于比较请在第一次运行完该程序后,将命令行 Clear 改为注释行。

例 4.22 求 $N!$,求 $1*2*3*4*\dots*N$ 的乘积。

* ExempFor4. Prg


```

* P=1 * 2 * 3 * 4 * ... * N
Clear
Input "N=" To N
P=1                      &&. 为什么将 1 赋给 P ?
For I=1 To N
    P=P * I              &&. 阶乘的通项式为 P=P * I
Next
? Alltrim(Str(N))+"! =",P
Return

```

例 4.23 给定一个整数 n , 判断它是否为素数。

分析: 所谓素数, 是指除了 1 和该数本身之外, 不能被其间任何整数整除的数。如 2, 3, 5, 7, 11 等均为素数, 对于 11 来说, 除了 1 和 11 以外, 不能被其间的 2~10 任何一个整数整除。

```

* ExempFor5. Prg
Clear
Input "N=" To N
Flag=0
For I=2 To N-1
    If N % I=0
        Flag=1
        Exit
    Endif
Endfor
If Flag=0
    ?N,"Is A Prime Number."
Else
    ?N,"Is Not A Prime Number."
Endif
Return

```

例 4.24 现在我们可以解决介绍循环时提出的问题了, 即如果要求输出 $Y = \sin(X)$ 在 $0^\circ \leq X \leq 180^\circ$ 范围内, 每隔 1° 的所有值。

经过前面的学习这个问题其实是一个非常简单的问题了, 大家可以试着完成一下吗? 提示: 函数 $\sin()$ 中的 X 变量是以弧度来表示的, 所以存在着将角度转化为弧度的问题, 循环时步长为 1° 。

4.4.3.2 Do While 循环

我们已经学习了计数型循环, 但很多情况下, 事先不可能预知程序所需的循环次数, 为此 VFP 提供了条件控制的 Do While 循环结构。

格式：

Do While <条件>

[<循环体语句序列 1>]

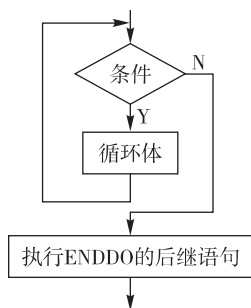
[Exit]

[<循环体语句序列 2>]

[Loop]

[<循环体语句序列 3>]

Enddo



功能：当满足条件时，重复执行循环体；不满足条件时，跳出循环，执行 Enddo 语句的下一条语句，其执行流程如图 4.14 所示。

参数说明：

1. Do While 和 Enddo 必须配对使用；
2. Do While... Enddo 循环首先判断条件表达式是否为真值，若为真值，则执行循环体，否则跳出循环；
3. Exit 和 Loop 语句的用法与 For 循环语句中的 Exit 和 Loop 语句用法相似；
4. 在通常情况下，必须先给 While 条件中的循环控制变量赋值；同时，在循环体中，也必须要有能改变循环条件值的语句，让循环条件在适当的时候变为假值，从而结束循环。否则有可能造成死循环；
5. 当 Do While 循环为永真循环时，只能使用 Exit 语句跳出循环。

Do While 循环结构的用法与 For 循环相似，大家可以尝试一下将前面介绍的 For 循环示例，用 Do While 循环结构来实现。

例 4.25 还记得我们在例 4.8 中提到的问题吗？即用户不使用提示的按键，而选择其他键时，就会出现问题，那么怎样限制用户的输入行为呢？下面仅仅只对 Wait “请选择 1、2、3” To C 的输入方式进行举例。

* ExempWhile1. Prg

Clear

Do While . T.

Wait “请选择 1、2、3:” To C

If C="1" Or C="2" Or C="3"

Exit

Else

Messagebox("选择错误，请重新选择！", 0+64)

Endif

Enddo

Display Memory Like C *

Return

例 4.26 显示 Employee 表中工资超过 1 000 元的所有记录。

* ExempWhile2. Prg

```

Clear
Set Talk Off
Use D:\Vfp\Employee           &.&. 打开 Employee 表
Locate For 工资>=1000           &.&. 定位工资超过 1000 元的记录
Do While !Eof()                 &.&. 当未到达文件尾时,执行循环体
    Display
    Continue
Enddo
Use
Return

```

例 4.27 根据输入的记录号来显示 Employee 中的记录,如果输入负数或 0,则循环结束;如果输入记录号太大,超过了最大记录号,则要求重新输入。

```

* ExempWhile3. Prg
Clear
Set Talk Off
Use Employee                   &.&. 打开 employee 表
Count To N                     &.&. N 为记录总数
Do While . T.
    Input "请输入记录号:" To I
    Do Case
        Case I<=0
            Exit
        Case I>N
            ?"记录号太大!"
            Loop
        Otherwise
            Go I
            Disp
    Endcase
Enddo
Use
Return

```

例 4.28 试用辗转相除法(又称欧几里得算法)求两个正整数的最大公约数。如求 27 和 15 的最大公约数,方法是:

$$\frac{27}{15} = 1 \cdots \cdots 12$$

$$\frac{15}{12} = 1 \cdots \cdots 3$$

$$\frac{12}{3} = 4 \dots\dots\dots 0$$

一直除到余数为 0, 最大公约数就是最后一个除式中的除数 3。

1. 算法:

- (1) 以大的数 M 作被除数, 小的数 N 作除数, 相除后余数设为 R;
- (2) 如果 $R \neq 0$, 则 $N \rightarrow M, R \rightarrow N$, 再转第一步进行相除, 得到新的 R;
- (3) 重复第二步, 直到 $R=0$ 为止, 此时的 N 就是最大公约数。

2. ExempWhile4. Prg 源程序:

```
* ExempWhile4. Prg
* 功能: 求最大公约数
Input "M=" To M
Input "N=" To N
If M<N Then           &&. 如果 M<N, 则交换两者之值
    C=M
    M=N
    N=C
Endif
R=M % N               &&. 求 M 和 N 的余数
Do While R !=0        &&. 当余数不为 0 时, 交换再求余
    M=N
    N=R
    R=M % N
Enddo
?"最大公约数是:", N   &&. 输出最大公约数
Return
```

思考: 在上例中 M 和 N 两个变量, 获得一个值后, 在求最大公约数的过程中, 这两个变量获得的初值, 将会丢失, 若要求在得到最大公约数的同时, 也要求保留原有数据应如何修改上述程序?

4.4.3.3 Scan 循环

格式:

```
Scan [<范围>][For <条件> | While <条件>][Nooptimize]
    [<循环体语句序列 1>]
    [Exit]
    [<循环体语句序列 2>]
    [Loop]
    [<循环体语句序列 3>]
Endscan
```

功能: 在指定范围内, 扫描满足给定条件的记录, 执行循环体。

参数说明:

1. <范围>用来指定 Scan 命令作用的范围,如果<范围>缺省,则默认的范围是当前表中的所有记录;如果<范围>不缺省,Scan 只扫描该范围之内的记录;
2. Exit 和 Loop 语句的用法与前两个循环结构中 Exit 和 Loop 语句的用法相似;
3. 这是一条针对表记录操作的循环语句。该语句执行时,首先把记录指针定位到指定范围的第 1 个记录,然后判断它是否满足过滤条件,如满足条件,执行循环体语句序列。当执行到达 Endscan 时,记录指针自动向后移动指向下一条记录,进行下一次循环,直到处理完指定范围内的所有记录后退出循环,执行 Endscan 后面的语句;
4. 由于 Scan 命令是对当前表进行扫描操作,所以在使用该命令前,必须先在当前工作区中打开表文件。

例 4.29 显示表 Employee.dbf 的前 10 个记录中性别为“女”的记录的有关字段。

* ExempScan1. Prg

Clear

Use D:\Vfp\Employee

Scan Next 10 For 性别="女"

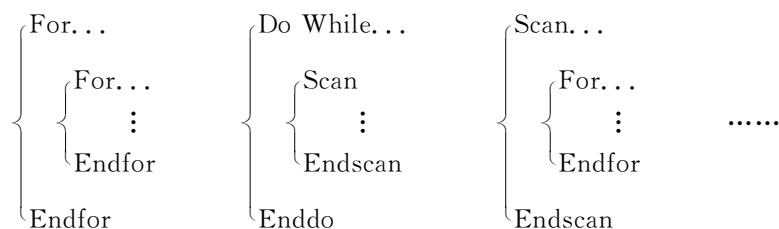
Display 身份号,姓名,性别,年龄,民族,工作时间

Endscan

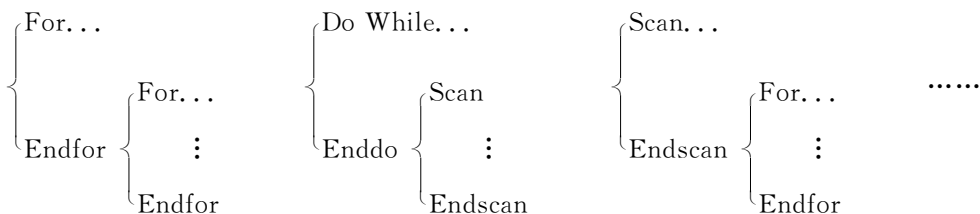
Return

4.4.3.4 循环的嵌套

在一个循环体内包含另一个完整地循环,我们把这种结构称为循环的嵌套,也称为多重循环。一般将外层的循环称为外循环,内层的循环称为内循环。循环的嵌套形式如下:



在循环嵌套中,各循环结构之间不得相互交叉,如下列循环嵌套的结构形式就是错误的:



不要从循环体外,在不经循环入口的情况下,强制转入循环体内。VFP 规定,不论是哪一种循环,循环入口和循环出口是配套的,即 Enddo、Endfor、Endscan 总是和最接近他的 Do While、For、Scan 配套。每当执行一次 Exit 语句时,表示程序只能退出当前循环,不可能一次退出多层循环体。每当执行一次 Loop 子句时,表示程序将在其所在层次的循环内进行下一次循环;

不论哪种情况,当程序进行循环入口时,由于条件不满足便会立即退出循环体,在这种情况下,程序将不执行循环体部分的内容。对于 Scan 循环,若带有 While 子句,涉及其条件的字段,最好是排过序或建立了索引,这样才不致漏掉必须处理的记录。

例 4.30 双重循环。

* ExempDouble1. Prg

Clear

Set Talk Off

? "I", "J"

For I=1 To 2

For J=3 To 5

? I,J

Endfor

?

Endfor

Return

运行结果:

I J

1 3

1 4

1 5

2 3

2 4

2 5

循环体“?I,J”共执行了 6 次,外循环变量每取值一次,内循环变量的值将取一遍(一个循环)。

例 4.31 找出 100~200 间的全部素数。

* ExempDouble2. Prg

Set Talk Off

Clear

For N=101 To 200 Step 2

K=Int(Sqrt(N))

I=2

Flag=0

Do While (I<=K) . And. (Flag=0)

If N % I=0

Flag=1

Else

```

        I=I+1
    Endif
Enddo
If Flag=0
    ?? N
Endif
Endfor
Set Talk On
Return

```

例 4.32 试编程输出乘法口诀表。

程序一：用 For 循环实现

```

* ExempDouble3. Prg
Set Talk Off
Clear
For I=1 To 9
    For J=1 To I
        K=I * J
        ??Str(J,1)+" * "+Str(I,1)+"=" +Str(K,2)+" "
    Endfor
    ?
Endfor
Set Talk On
Return

```

程序二：用 Do While 循环实现

```

* ExempDouble4. Prg
Set Talk Off
Clear
Store 1 To I,J,K
Do While I<=9
    J=1
    Do While J<=I
        K=I * J
        ??Str(J,1)+" * "+Str(I,1)+"=" +Str(K,2)+" "
        J=J+1
    Enddo
    I=I+1
    ?
Enddo

```

Set Talk On

Return

程序执行结果：

```
1*1= 1
1*2= 2 2*2= 4
1*3= 3 2*3= 6 3*3= 9
1*4= 4 2*4= 8 3*4=12 4*4=16
1*5= 5 2*5=10 3*5=15 4*5=20 5*5=25
1*6= 6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7= 7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8= 8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9= 9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

若要求将乘法口诀表分别以下列两种形式输出，以上程序又将如何修改呢？

```
1*1= 1 2*1= 2 3*1= 3 4*1= 4 5*1= 5 6*1= 6 7*1= 7 8*1= 8 9*1= 9
2*2= 4 3*2= 6 4*2= 8 5*2=10 6*2=12 7*2=14 8*2=16 9*2=18
3*3= 9 4*3=12 5*3=15 6*3=18 7*3=21 8*3=24 9*3=27
4*4=16 5*4=20 6*4=24 7*4=28 8*4=32 9*4=36
5*5=25 6*5=30 7*5=35 8*5=40 9*5=45
6*6=36 7*6=42 8*6=48 9*6=54
7*7=49 8*7=56 9*7=63
8*8=64 9*8=72
9*9=81

1*1= 1 2*1= 2 3*1= 3 4*1= 4 5*1= 5 6*1= 6 7*1= 7 8*1= 8 9*1= 9
      2*2= 4 3*2= 6 4*2= 8 5*2=10 6*2=12 7*2=14 8*2=16 9*2=18
            3*3= 9 4*3=12 5*3=15 6*3=18 7*3=21 8*3=24 9*3=27
                  4*4=16 5*4=20 6*4=24 7*4=28 8*4=32 9*4=36
                        5*5=25 6*5=30 7*5=35 8*5=40 9*5=45
                              6*6=36 7*6=42 8*6=48 9*6=54
                                    7*7=49 8*7=56 9*7=63
                                          8*8=64 9*8=72
                                                9*9=81
```

例 4.33 求 $\sum n!$ ，即求 $1! + 2! + \dots + n!$ 的值。

分析：这个问题可以用循环的嵌套来解决，嵌套的要求与 If 条件结构语句相似，不允许交叉嵌套。外循环用于解决求和问题，其通项式为： $S = S + P$ 而内循环用来解决求阶乘，其通项式为： $P = P * I$ ，具体程序代码如下：

```
* ExempDouble5. Prg
* S=1! + 2! + ... + n!
Clear
Input "N=" To N
S=0
For I=1 To N
    P=1
    For J=1 To I
        P=P * J
    Next
```



```

S=S+P
Next
?"1!+2!+...+"+Alltrim(Str(N))+"!=",S
Return

```

例 4.34 前例其实可以写成更简单的形式,即在一次循环中先求出阶乘,再对阶乘求和。其源程序如 ExempDouble6. Prg,在此增加了对输入的控制!(上例中若输入非数值型数据,会出现什么情况呢?)

```

* ExempDouble6. Prg
* S=1!+2!+...+n!
Clear
Input "N=" To N
If Type("N")="N"
    If N>=1
        S=0
        P=1
        For I=1 To N
            P=P*I           &&. 求阶乘
            S=S+P           &&. 对阶乘求和
        Next
        ?"1!+2!+...+"+Alltrim(Str(N))+"!=",S
    Else
        MessageBox("您输入了一个小于 1 的数!",0+48)
    Endif
Else
    MessageBox("您输入了一个非数值型数据!",0+16)
Endif
Return

```

从上例中我们可以看到,在计算机接收键盘输入的数据后,首先对数据进行识别,当数据不符合要求时,给出提示。在今后的编程过程中,特别是在编制一些应用程序时,要求用户输入合法的数据问题是一个非常重要的问题,解决这个问题的唯一方法就是对输入的数据进行识别,当输入的数据合法时,允许程序继续执行,当输入的数据不合法时,则弹出出错提示。

例 4.35 在上例中,每次运行程序只能输入一次数据,如果要求每次运行程序,可以计算多个数的阶乘,并由用户决定是否继续。另外,每次用户输入的数据发生错误时,提示用户数据输入错误,并允许用户重新输入数据直到数据正确为止。实现上述功能的程序代码如下:

```

* ExempDouble7. Prg
* S=1!+2!+...+n!

```

```

Clear
Do While .T.
    Input "N=" To N
    If Type("N")="N"
        If N >= 1
            S=0
            P=1
            I=1                                &&. 给循环控制变量赋初值
            Do While I <= N
                P=P * I                        &&. 求阶乘
                S=S+P                          &&. 对阶乘求和
                I=I+1
            Enddo
            ?1!+2!+...+"+Alltrim(Str(N))+"!="+Alltrim(str(S))
            * 下一行命令弹出只有“是”和“否”两个按钮的对话框,当用户选择“否”时,函数
            * 返回值为 7,其他有关问题可查阅前面介绍的 MessageBox()函数
            X=MessageBox("继续计算阶乘吗?",4+48)
            If X=7
                Exit
            Endif
        Else
            MessageBox("您输入了一个小于 1 的数!",0+48)
        Endif
    Else
        MessageBox("您输入了一个非数值型数据!",0+16)
    Endif
Enddo
Return

```

上例中使用了双重 Do While 循环,其中内循环的功能是计算阶乘之和,循环执行的次数由循环控制变量 I 控制,在循环体语句中,有改变循环变量的语句,让循环在执行的过程中自动改变循环控制变量的值,而达到控制循环的目的。而外层循环是一个永真循环,即循环条件永远为真值,其作用是让用户输入一个数据,直到数据合法后,再去执行内循环,执行完内循环后,由用户选择是否退出程序。

4.5 结构化程序设计

在应用程序设计过程中,通常我们是将复杂的程序分解成一个个功能相对简单的功能模块,这样不仅便于程序的开发,也有利于程序的阅读和维护。模块可以被其他模块所

调用,也可以去调用其他模块。一般,我们将被其他模块调用的模块称为子模块,将调用其他模块而没有被其他模块调用的模块称为主程序。所以模块可以是主程序、子程序、过程和自定义函数等。

4.5.1 子程序

两个程序文件之间如果存在着调用关系,则称调用程序为主程序,被调用程序为子程序,所以,子程序也是一个独立的程序文件。

4.5.1.1 不带参数子程序的调用与返回

调用格式:

Do <子程序文件名>

功能:在主程序中直接用该命令调用指定的子程序文件,当子程序执行完毕后(或运行到子程序的 Return 语句处),返回到主程序的调用处继续执行 Do <子程序文件名> 后面的语句。

例 4.36 我们将例 4.34 的源程序 ExempDouble6. Prg,进行修改,主程序只处理数据的接收和子程序的调用,子程序进行数据的判断和求阶乘之和。

主程序 ExempMaster1. Prg 源代码:

```
* ExempMaster1. Prg
* S=1!+2!+...+n!
Clear
Input "N=" To N           && 变量 N 的作用范围为主程序及下级子模块
Do ExempSub1
Return
```

子程序 ExempSub1. Prg 源代码:

```
* ExempSub1. Prg
If Type("N")="N"           && 变量 N 已经在主程序中赋值,这里仅仅是判断其类型
    If N >= 1
        S=0
        P=1
        For I=1 To N
            P=P*I           && 求阶乘
            S=S+P           && 对阶乘求和
        Next
        ?"1!+2!+...+"+Alltrim(Str(N))+ "!=",S
    Else
        MessageBox("您输入了一个小于 1 的数!",0+48)
    Endif
Else
    MessageBox("您输入了一个非数值型数据!",0+16)
```

```
Endif  
Reutrn
```

在完成了上述两个命令文件的建后,在命令行中执行命令:Do ExempMaster1,其运行结果与 ExempDouble6. Prg 完全相同。

4.5.1.2 带参数子程序的调用与返回

调用格式一:

Do <子程序文件名> [With <实际参数表>]

调用格式二:

<子程序文件名>(<实际参数表>)

功能:在调用子程序的同时完成主程序与子程序之间的参数传递。

参数说明:

1. <实际参数表>中的参数可以是表达式,但若为内存变量,则必须具有初值;
2. 当实际参数有两个以上时,参数间以半角逗号分隔;
3. 进行带参调用时,被调用的子程序或过程中也必须有相应的接收参数和回送参数

语句:

Parameters | Lparameters <形式参数表>

功能:在子程序中指定形式参数,用以接收 Do <子程序文件名> [With <实际参数表>](或<子程序文件名>(<实际参数表>))命令发送的实际参数值,返回时把形式参数的值返回给调用程序中相应的实际参数。

参数说明:

- (1) Parameters 必须是被调用子程序的第一个语句;
- (2) Parameters <形式参数表>中定义的内存变量,我们称之为形式参数,当有多个形式参数时,形式参数之间以半角逗号分隔;
- (3) Parameters 中的形式参数,VFP 默认为子程序的私有变量;
- (4) Parameters 中的形式参数依次与 Do <子程序文件名> [With <实际参数表>](或<子程序文件名>(<实际参数表>))命令中参数表的参数一一对应;
- (5) Lparameters 的用法与 Parameters 相同,二者不同的是用 Parameters 定义的形式参数为私有变量,用 Lparameters 定义的形式参数为局部变量。私有变量除在本模块中有效外,对其下级模块也有效,而局部变量只在本模块中有效。

4. 带参调用的参数传递过程:首先将实际参数传递给被调用的子模块中的形式参数,并运行子模块,执行到子模块的 Return 语句或子模块运行结束时,子程序将形式参数的值返回给对应的实际参数,程序退出被调用模块的同时,从内存中清除被调用模块的所有私有变量;

5. 调用格式一只有一种调用形式,即以命令形式调用。调用格式二有两种调用形式,若调用格式二独立使用时,则是以命令形式调用;若调用格式二是某表达式的组成部分,则将以表达式形式调用。

例 4.37 我们将例 4.36 修改为参数调用形式。

主程序 ExempMaster2. Prg 源代码:

```

* ExempMaster2. Prg
* S=1!+2!+...+n!
Clear
Input "N=" To NN
Do ExempSub1 With NN    &&.NN 为实际参数,其作用范围为主程序及调用它的子程序
Return

```

子程序 ExempSub2. Prg 源代码:

```

* ExempSub2. Prg
Parameters N            &&.N 为形式参数,作用范围为子程序及程序调用的下级模块,
                        &&. 返回到调用程序时,将值传递给实际参数后,并从内存中清除
If Type("N")="N"
    If N >= 1
        S=0
        P=1
        For I=1 To N
            P=P*I &&. 求阶乘
            S=S+P &&. 对阶乘求和
        Next
        ?"1!+2!+...+"+Alltrim(Str(N))+"!=",S
    Else
        Messagebox("您输入了一个小于 1 的数!",0+48)
    Endif
Else
    MessageBox("您输入了一个非数值型数据!",0+16)
Endif
Return

```

例 4.38 用子程序调用的形式,求长方形的面积。

主程序 ExempMaster3. Prg 源代码:

```

* ExempMaster3. Prg
Set Talk Off
Clear
@10,10 Say "请输入长方形的第一条边的边长:" Get L default 0 Picture "999999"
@Row()+2,10 Say "请输入长方形的第二条边的边长:;"
    " Get W default 0 Picture "999999"
X=Row()
Read
@X+2,10 Exempsub4(L,W)    &&. 以调用格式二的形式调用子程序
Return

```

子程序 ExempSub3. Prg 源代码:

```
* ExempSub3. Prg
Parameters LL,WW
S=LL * WW
Return "长方形的面积为:"+Alltrim(Str(S))
```

4.5.1.3 子程序的嵌套调用

当一个子程序被上级程序调用时,这个子程序本身还可调用其他子程序,即子程序调用子程序,我们将这种形式的调用称为子程序的嵌套调用。VFP 的返回命令包含了因子程序嵌套而引起的多种返回形式,其格式如下:

```
Return [<表达式> | To Master | To <程序文件名>]
```

参数说明:

- 1. 当子程序以格式:<子程序文件名>(<实际参数表>)的形式被调用时,<表达式>为子程序的返回值,若缺省<表达式>,默认的返回值为逻辑真值;
- 2. To Master :直接返回到最外层主程序的调用处;
- 3. To <程序文件名>:强制返回到指定的程序的调用处;
- 4. 缺省选项时,返回到上级模块的调用处。

子程序的嵌套调用的形式如图 4.15 所示。

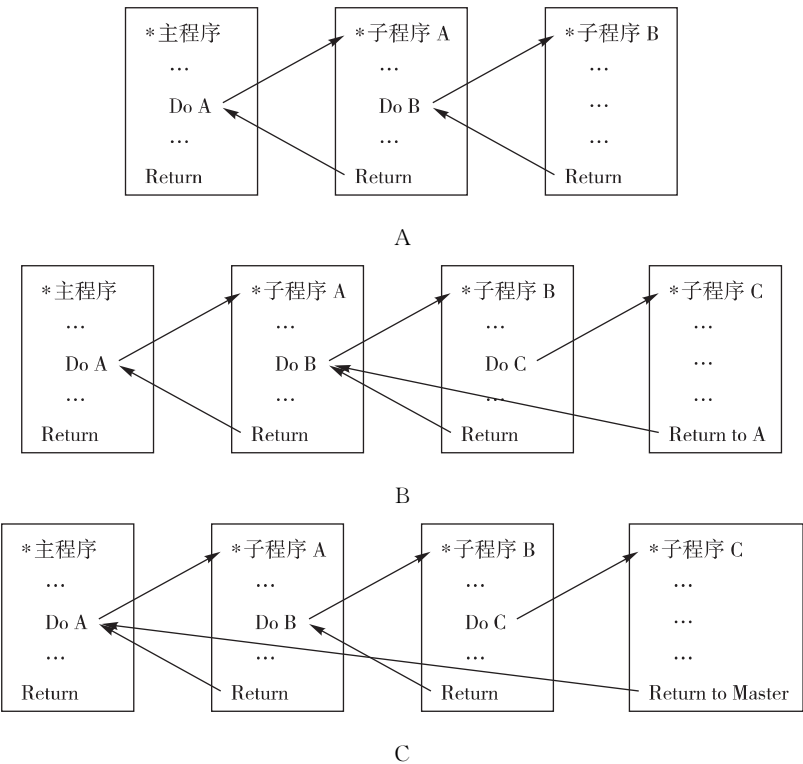


图 4.15 子程序的嵌套调用形式

例 4.39 我们将例 4.37 修改子程序的嵌套调用形式。ExempMaster4. Prg 为主程

序文件、ExempSub41. prg 和 ExempSub42. prg 为子程序,他们之间的调用关系是 ExempMaster4 调用 ExempSub41, ExempSub41 再调用 ExempSub42,本例中的调用都采用了带参数调用形式。

主程序 ExempMaster4. Prg 接收输入的数据,调用子程序,其源代码为:

```
* ExempMaster4. Prg
* S=1!+2!+...+n!
Clear
Input "N=" To NNN
Do ExempSub41 With NNN  &&.NNN 为实际参数,其作用范围为主程序及调用它的子程序
Return
```

子程序 ExempSub41. Prg 判断接受数据的合法性,并调用子程序,其源代码为:

```
* ExempSub41. Prg
Parameters NN          &&.NN 为形式参数,作用范围为子程序及程序调用的下级模块,
                        &&. 返回到调用程序时,将值传递给实际参数后,从内存中被清除
If Type("NN")="N"
  If NN>=1
    Do ExempSub42 With NN
  Else
    MessageBox("您输入了一个小于 1 的数!",0+48)
  Endif
Else
  MessageBox("您输入了一个非数值型数据!",0+16)
Endif
Return
```

子程序 ExempSub42. Prg 仅仅只求阶乘之和,其源代码为:

```
* ExempSub42. Prg
Parameters N          &&.N 为形式参数,作用范围为子程序及程序调用的下级模块,
S=0
P=1
For I=1 To N
  P=P*I          &&. 求阶乘
  S=S+P          &&. 对阶乘求和
Next
?"1!+2!+...+"+Alltrim(Str(N))+"!=",S
Return
```

4.5.2 过程

4.5.2.1 建立过程

若将多模块程序中的每个模块分别都保存为程序文件,则每执行一个模块,都要打开

一个程序文件,这样将会降低程序的运行效率。

在 VFP 中允许将多个程序模块存放在一个文件中,在这个文件中各个模块都被定义为过程,这种包含过程的文件称为过程文件,其扩展名与程序文件的扩展名相同。过程也可以放置在程序文件代码的后面。

过程的格式:

Procedure <过程名>

[Parameters | Lparameters <形式参数表>]

<语句序列>

[Return [<表达式> | To Master | To <程序文件名(或过程名)>]]

参数说明:

1. 过程由 Procedure 开始,以 Return 结束;
2. <过程名>是标识该过程的标识符号,每个过程都是通过其名称实现调用的;
3. 同子程序的调用一样,过程若需要与调用模块进行参数传递,也需用 Parameters (或 Lparameters) 语句;
4. Return 的用法与子程序调用的用法相似。

4.5.2.2 过程的调用

过程可放在一般程序文件代码的后面,也可将所有过程保存在单独的过程文件中。若需要调用的过程不是存放在当前程序文件代码的后面,而是保存在一个过程文件中,那么调用之前就需要将过程文件打开,过程文件使用完毕,就需要将其关闭。

4.5.2.2.1 打开过程文件

格式:

Set Procedure To <过程文件名表> Additive

功能:打开指定的过程文件

参数说明:

1. 有 Additive 选项时,表示打开过程文件时,不关闭以前打开的过程文件,否则关闭;
2. 当过程文件处于打开状态,其包含的过程就可以被直接调用。

格式:

Release Procedure <过程文件名表>

功能:关闭指定的过程文件

4.5.2.2.2 过程的调用

过程调用格式一:

Do <过程名> [With <实际参数表>][In <过程文件>]

过程调用格式二:

<过程名>(<实际参数表>)

参数说明:

过程的调用方法与子程序的调用方法相似,但当打开的两个程序文件中存在同名的过程时,则用 In 选项,指定被调用过程的过程文件名。

例 4.40 编写一个一行输出 40 个“*”的过程,主程序调用它两次。

主程序文件 ExempMaster5. Prg 源代码：

```
* Main Program
Do Stars                && 调用无参过程 Stars
Do Stars                && 再次调用无参过程 Stars
Return
Procedure Stars
    For I=1 To 40
        ?? " * "
    Endfor
    ?
```

Return

运行结果(共两行,每行 40 个“ * ”号)

```
* * * * *
* * * * *
```

例 4.41 我们将例 4.39 修改成过程调用的形式,其中主程序部分为主程序文件 ExempMaster6. Prg,两个过程都存放在一个过程文件中。

主程序文件 ExempMaster6. Prg 源代码：

```
* ExempMaster6. Prg
* S=1!+2!+...+n!
Clear
Input "N=" To NNN
Set Procedure To ExempPruce6    && 打开过程文件
Do ExempPruce61 With NNN        && 调用过程 ExempPruce61
Release Procedure ExempPruce6    && 关闭过程文件 ExempPruce6
Return
```

过程文件 ExempPruce6. Prg 源代码：

```
* ExempPruce61 判断接受数据的合法性,并调用另一个过程
Procedure ExempPruce61
    Parameters NN
    If Type("NN")="N"
        If NN>=1
            ? ExempPruce62(NN)    && 以表达式的形式调用过程 ExempPruce62
        Else
            MessageBox("您输入了一个小于 1 的数!",0+48)
        Endif
    Else
        MsgBox("您输入了一个非数值型数据!",0+16)
    Endif
```

```

Return
Endproc
* ExempPruce62 仅仅只求阶乘之和
Procedure ExempPruce62
Parameters N
S=0
P=1
For I=1 To N
    P=P*I
    S=S+P
Next
Return "1! + 2! + ... + " + Alltrim(Str(N)) + "! =" + Alltrim(Str(S))
Endproc

```

例 4.42 我们将上例的过程调用存放在主程序文件 ExempMaster7. Prg 代码后, 这样在调用这两个过程时, 就不存在打开和关闭过程文件的问题。

主程序文件 ExempMaster7. Prg 源代码:

```

* ExempMaster7. Prg
* S=1! + 2! + ... + n!
Clear
Input "N=" To NNN
Do ExempPruce71 With NNN           &&. 调用过程 ExempPruce71
Return
* ExempPruce71 判断接受数据的合法性, 并调用另一个过程
Procedure ExempPruce71
Parameters NN
If Type("NN")="N"
    If NN>=1
        ? ExempPruce72(NN)       &&. 以表达式的形式调用过程 ExempPruce72
    Else
        MessageBox("您输入了一个小于 1 的数!", 0+47)
    Endif
Else
    MessageBox("您输入了一个非数值型数据!", 0+16)
Endif
Return
Endproc
* ExempPruce72 仅仅只求阶乘之和
Procedure ExempPruce72

```

```

Parameters N
S=0
P=1
For I=1 To N
    P=P*I
    S=S+P
Next
Return "1!+2!+...+"+Alltrim(Str(N))+"!="+Alltrim(Str(S))
Endproc

```

4.5.3 自定义函数

函数分为两类:标准函数和自定义函数。标准函数是 VFP 已经定义好的子程序,可供用户直接调用,如前面章节介绍的 Len(),Left(),Date()等各种函数。

4.5.3.1 建立自定义函数

格式:

```

Function <函数名>
    [Parameters | Lparameters <形式参数表>]
    <语句序列>
Return [<表达式>]

```

参数说明:

1. 函数以 Function 开始,以 Return 结束;
2. 函数名是标识自定义函数的标识符号,每个函数都是通过其名称实现调用的,函数名不能与其他子程序同名;
3. Parameters | Lparameters 的用法与子程序和过程的用法相似;
4. Return 为返回调用程序的语句,即返回到函数调用语句的下一条语句。函数的返回值为<表达式>的值,如果缺省<表达式>,则函数默认的返回值为逻辑真值;
5. 函数的存放如同过程的存放一样,既可以在一般程序文件代码的后面,也可将所有过程、函数等保存在单独的过程文件中。但调用过程文件中的过程或函数时,必须先打开过程文件。

4.5.3.2 调用函数

自定义函数同标准函数一样,可以出现在任何表达式中,并与 VFP 标准函数一样使用。

调用格式:

<函数名>([实际参数表])

例 4.43 我们将上例修改成函数调用的形式。

主程序文件 ExempMaster8. Prg 源代码:

```

* ExempMaster8. Prg
* S=1!+2!+...+n!
Clear

```

```

Input "N=" To NNN
Do ExempPruce81 With NNN          &&. 调用过程 ExempPruce81
Return
* ExempPruce81 判断接受数据的合法性,并调用另一个过程
Procedure ExempPruce81
Parameters NN
If Type("NN")="N"
    If NN>=1
        ?1!+2!+...+"+Alltrim(Str(NN))+"!="+Alltrim(Str
        (ExempFunc81(NN)))          &&. 函数调用
    Else
        MessageBox("您输入了一个小于1的数!",0+48)
    Endif
Else
    MessageBox("您输入了一个非数值型数据!",0+16)
Endif
Return
Endproc
* ExempFunc81 仅仅只求阶乘之和
Function ExempFunc81
Parameters N
S=0
P=1
For I=1 To N
    P=P*I
    S=S+P
Next
Return S

```

4.5.4 参数传递

在介绍子程序、过程和函数的调用时,为了说明问题的方便,对调用模块与被调用模块间参数传递的问题,是这样介绍的:在多模块调用中,调用模块的实际参数,通过调用下层模块,将实际参数值传递给下层模块的形式参数,当下层模块运行结束时,又将运行后的形式参数值返回给上层模块的实际参数。实际情况是这样的吗?其实在 VFP 中,多模块调用时,实际参数与形式参数的传递方式有传址和传值两种方式。

4.5.4.1 传址方式

当一个模块调用另一个下层模块时,系统将实际参数的地址传递给形式参数,即形参变量与实参变量共用存储单元,我们将这种形式的参数传递方式称为传址方式。因此,在

传址方式下,在被调下层模块中对形式参数的任何操作,实际上都是对相应实际参数的操作,所以实际参数的值会随着形式参数的值而改变。也就相当于在调用时将实际参数的值传递给被调用模块的形式参数,当被调用模块运行结束后,又将形式参数的值返回给实际参数。前面所有示例中介绍的参数传递方式都是传址方式。

4.5.4.2 传值方式(又称值传递方式)

在有些程序的设计过程中,要求实际参数完成传递后,其值不随形式参数而改变,以传值方式进行的参数传递就能实现这一功能。传值方式是指调用一个下层模块时,系统仅将实际参数的值复制给形式参数;在被调用的模块中形式参数有自己独立的存储地址,当被调用模块运行结束时,这些形式参数所占用的存储单元也同时被释放,而并不向实际参数回传形式参数的值,因此在被调用模块中对形式参数的任何操作都不会影响到实际参数。所以,以传值方式进行的参数传递是“单向”的,即只能将实际参数传递给形式参数,而不能将形式参数值返回给实际参数。

4.5.4.3 传值方式与传址方式的实现

1. 在 Do <文件名 | 过程名> [With <实际参数表>]调用方式下,VFP 默认采用传址方式,但当实际参数是常量时,为传值方式;如果实际参数是变量,但又要求采用单向的传值方式进行参数传递,其调用格式是将需要以传值方式进行参数传递的实际参数变量用括号括起来,如 Do P With A,(B),该调用命令中 A 是传址方式,B 是传值方式;

2. 在<文件名 | 过程名 | 函数名>(<实际参数表>)调用方式下,VFP 默认采用传值方式;若要求实际参数中的某些变量采用传址方式进行参数传递,其调用格式是将需要以传址方式进行的实际参数前加“@”符号;

3. 当使用数组进行参数传递时,如果采用传值方式,则只能传递数组的第一个元素的值,其他元素的值将不会传递。所以,数组的传递一般采用传址方式,即将作为实际参数数组的地址传给了被调用子模块的形式参数。其调用格式是直接在实参数组名前应加“@”符号,在被调用模块中的参数接收语句中直接使用数组名作为形式参数;

4. 在<文件名 | 过程名 | 函数名>(<实际参数表>)调用方式下,使用下列设置命令,也可以改变参数传递方式:

格式:

Set Udfparms To Value | Reference

功能:在<文件名 | 过程名 | 函数名>(<实际参数表>)调用方式下,通过上述命令改变没有指明传递方式的参数的传递方式。

参数说明:

(1) To Value:按传值方式传递,形参变量值的改变不影响实参变量的取值;

(2) To Reference:按传址方式传递,形参变量值改变时,实参变量的值也随之改变。

例 4.44 传址方式的数据传递。

主程序文件 ExempRefer.Prg 源代码:

```
* ExempRefer.Prg
```

```
Clear
```

```
A=8
```

```

B=10
?未调用子过程前:"
?"A=",A,"B=",B
Do P With A,B
?调用子过程后:"
?"A=",A,"B=",B
Retu
Procedure P
    Parameters M,N
    M=M+N
    N=M+N
    ?"M=",M,"N=",N
    Return
Endproc

```

运行结果：

未调用子过程前：

```

A=      8      B=      10
M=     18     N=     28

```

未调用子过程后：

```

A=     18     B=     28

```

例 4.45 传值方式的数据传递。

主程序文件 ExempValu. Prg 源代码：

```

* Exempvalu. Prg
Clear
A=8
B=10
?未调用子过程前:"
?"A=",A,"B=",B
Do P With (A),(B)
?调用子过程后:"
?"A=",A,"B=",B
Retu
Procedure P
    Parameters M,N
    M=M+N
    N=M+N
    ?"M=",M,"N=",N
    Return

```

Endproc

运行结果

未调用子过程前:

A= 8 B= 10

M= 18 N= 28

调用子过程后：

A= 8 B= 10

例 4.46 传递数组。

主程序文件 ExempDime. Prg 源代码:

* ExempDime. Prg

Clear

Dimension A(10)

For I=1 To 10 && 给数组 A 的各元素赋值的同时输出数组 A 的各元素值

$$A(I) = I * I$$

If Mod(I,5)-1=0 && 每行输出 5 列数据后换行

?

Endif

?? $A(I)$

Endfor

Do Dx With A

For I=1 To 10 &&. 输出数组 A 的各元素值

If $\text{Mod}(I, 5) - 1 = 0$

?

Endif

?? $A(I)$

Endfor

Return

Procedur Dx

Parameters B	&& 将整个数组作为一个整体接收
--------------	------------------

For I=1 To 5 && 将整个数组元素按值的大小倒个次序存放

$$X=B(I)$$
$$B(I) = B(11 - I)$$
$$B(11-I)=X$$

Endfor

Return

Endproc

程序的运行结果为：

1 4 6 9 25

36	49	64	81	100
100	81	64	49	36
25	16	9	4	1

例 4.47 计算长方体的体积。

主程序文件 ExempMaster9. Prg 源代码：

```
* ExempMaster9. Prg
Clear
Dimension A(3)
A=0
@8,10
For I=1 to 3
    @Row()+2,10 Say "请输入长方形的第"+Str(I,1)+"条边的值:" ;
    Get A(I) Picture "999999"
Endfor
X=Row()+2
Read
@X,10 Say "长方体的体积为:"
@Row(),Col() Say Volumes(@A)
Return
Function Volumes
    Parameters A
    Return A(1) * A(2) * A(3)
```

在上例中,如果输入的数值小于 0,则会出现错误的输出值,为了避免这种错误,大家可以尝试完善以上代码,增加对输入的数据的处理模块,其处理过程可参阅前面一些例子介绍的处理方法,但需要指出的是本例中数组 A 只能接受数值型数据,所以在调用函数前,只需要判断每个数据是否大于等于 0。

另外,这里仅仅只介绍了函数参数采用数组传递的用法,关于在子程序和过程的调用中采用数组传递参数的用法与上例相似。

4.5.5 变量的作用域

内存变量的作用域,简称为变量的作用域。在多模块程序设计过程中,某子模块中的变量是否在其他模块中也有效呢?答案是不肯定的,因为用户定义的每一个变量有各自的作用域(作用范围)。按变量的作用域来划分内存变量,可分为公共变量、私有变量和本地变量三类。

4.5.5.1 公共变量

在任何程序中都可使用的内存变量称为公共变量。在程序设计过程中,公共变量要先建立再使用,建立公共变量的命令格式:

```
Public <内存变量表>
```


功能:将<内存变量表>中的内存变量设为公共变量,并为它们赋初值:. F.。

参数说明:

1. 在建立公共变量以后,可以再给它们赋任何类型的值;
2. 在任何位置说明的公共变量均可在所有程序中使用;
3. 在 Command 窗口中,直接用命令方式建立的内存变量均为公共变量,但以这种方式建立的公共变量不能在程序中使用;
4. 程序运行结束时,公共变量不会自动清除,而只能用专用的清除内存变量命令来清除,如:Release、Clear Memory 或 Clear All 等命令。

4.5.5.2 私有变量

凡在程序中没有通过 Public 或 Local 命令事先建立或声明,而直接使用的内存变量均为私有变量。私有变量的作用域仅在定义他的模块及其调用的下层模块中有效,当定义该私有变量的模块运行结束时,该模块中的私有变量将同时自动从内存中释放。

私有变量还可以通过 Private 命令来声明,其命令格式为:

Private [<内存变量表>] [All [Like | Except <通配符>]]

功能:声明私有变量并隐藏上级程序中的同名变量,直到声明它的模块结束运行后,才恢复使用被隐藏的上级同名变量。

参数说明:

1. 当要使用的私有变量与上层模块中的变量名发生重名时,为了区分这两个变量,必须使用 Private 命令声明;
2. Private 命令只是声明私有变量,并不是建立变量,因而不对其赋值;
3. 在对下层模块的调用时,用 Parameters 语句说明的形式参数也是私有变量。

4.5.5.3 本地变量

本地变量又称为局部变量,本地变量只能在建立它的模块中使用,而不能在其上层模块或下层模块中使用,与私有变量一样,当定义本地变量的模块运行结束时,该模块中的本地变量将同时从内存中自动释放。建立本地变量的格式:

Local <内存变量表>

功能:将<内存变量表>中的内存变量设为本地变量,同时对它们赋初值. F.。

说明:

1. 在对下层模块的调用时,用 Lparameters 语句说明的形式参数也是本地变量;
2. Local 与 Locate 前 4 个字母相同,因而不能缩写。

学习过变量的作用域这部分知识后,大家再将前面介绍的一些例题阅读一下,了解掌握各模块中变量的作用域。下面以一个简单的示例,来说明三类变量的差异。

例 4.48 变量作用域示例。

主程序文件 ExempMaster10. Prg 源代码:

```
* ExempMaster10. Prg
```

```
clear
```

```
clear all
```

```
Public X1
```

```
&.& 建立公共变量 X1
```

```
Store 23 To X2           &&. 建立私有变量 X2
Local X3                 &&. 建立本地变量 X2
X1="职业技术"
X3=.F.
?主程序中原值
?X1=',X1,', X2=',X2,', X3=',X3
List Memor Like X *
Do P1
?
?返回主程序后的值
?X1=',X1,', X2=',X2,', X3=',X3
List Memor Like X *
Return
```

```
* 过程 P1
Procedure P1
X1="安徽合肥"
X2=88
X3=.T.
?
?子程序中的值
?X1=',X1,', X2=',X2,', X3=',X3
List Memor Like X *
Retu
Endproc
```

运行结果如图 4.16 所示。

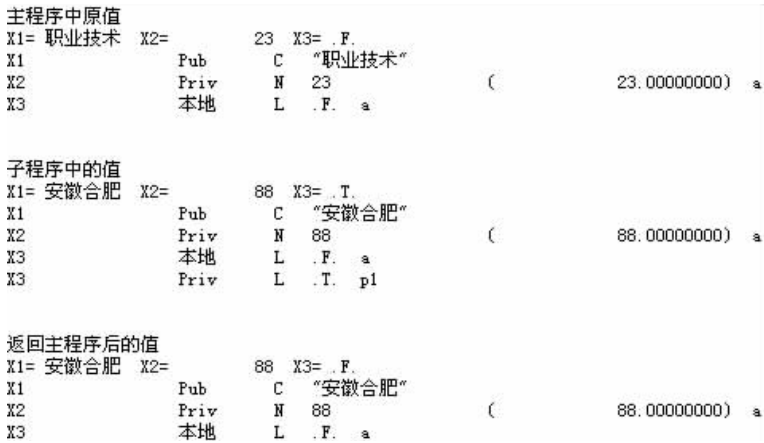


图 4.16 变量作用域示例结果

4.5.6 程序调试

大家已经学习和掌握了 VFP 的基本编程方法,也可以编写一些简单的 VFP 代码,但是往往会出现错误;另外,在开发了系统应用程序后,总是希望他能正常工作,但在编译运行时,会发现它有许多错误,或不能像预期设想那样工作,或者两者都有。程序调试的目的就是检查并纠正程序中的错误,调试程序如同预期一样工作,以保证程序运行的可靠性。

4.5.6.1 错误类型

通常程序中可能出现三种类型的错误,即语法错误、运行错误和逻辑错误。

1. 语法错误。

语法错误通常是在语句结构(即语法)不正确时出现的错误。例如,丢失或写错了符号,关键字拼写不正确,以及括号不匹配等。当程序运行到具有语法错误的语句时就会停下来,并弹出如图 4.17 所示的“程序错误”提示框。



图 4.17 “语法错误”提示对话框

取消——中止程序运行,回到命令窗口。相当于执行了 Cancel 命令,在程序中创建的所有变量被释放(除全局变量),但数据库及数据库表一般保持当时的状态。

挂起——暂停程序。相当于执行了 Suspend 命令,这时程序中的所有变量都保持原值,我们可以用 ? 和 ?? 命令查看变量的值,也可以查看数据表的情况。

忽略——忽略所出现的错误,即跳过出错的语句继续执行后面的语句。

帮助——显示有关出错的帮助信息。

2. 运行错误。

语法正确,但运行时无法执行的错误叫运行错误。该错误在编码时很难发现,往往是由于试图执行一个不可进行的操作(即非法操作)而引起的。常见的运行错误如除数为 0、数值溢出、数组下标越界,或使用一个不存在的对象等。

3. 逻辑错误。

与语法错误和运行错误不同,逻辑错误一般不报告错误信息,它是由于设计错误或其他原因导致运行流程及运行结果与原先设计不符的错误。程序能正常运行,但不能得到预期的结果。逻辑错误是最难查找和排除的错误,必须有调试工具的帮助。可通过跟踪运行流程,显示变量值等方法查找错误所在。

4.5.6.2 程序调试

程序调试是应用程序开发过程中非常重要的工作,通过程序调试发现和解决程序中的错误。VFP 为用户提供了调试器环境。

4.5.6.2.1 调试器环境

打开调试器环境的方法一般有两种:

方法一:单击菜单“工具/调试器”

方法二:在命令窗口中执行命令:Debug。

在“调试器”窗口中含有 5 个子窗口:跟踪、监视、局部、调用堆栈和调试输出,如图 4.18 所示,用户可以有选择地打开或关闭其中的子窗口。要打开子窗口,可以单击“调试器”窗口的“窗口”菜单中的相应命令;要关闭子窗口,只需单击窗口右上方的“关闭”按钮。

1. “跟踪”子窗口。

“跟踪”子窗口用于显示并跟踪正在调试执行的程序。当通过“调试器”窗口的菜单“文件/打开”命令,打开要调试的程序文件后,被打开的程序文件代码就会显示在这个子窗口中。

跟踪窗口左端的灰色区域会显示某些符号,最常用的就是断点和当前行。

(◇表示指向的代码行为调试中正在执行的代码行。)

(•表示断点,当程序执行到设置了断点的代码行时,暂停程序的执行。)

2. “监视”子窗口。

“监视”子窗口用于监视指定表达式在程序执行过程中的取值变化情况。设置监视表达式的方法是:在“监视”文本框中输入表达式,然后单击回车键,表达式便添加到下方的列表框中。也可以将其他窗口中的文本拖到监视窗口,来创建监视表达式。列表框中表达式的删除,可在选中要删除的表达式的基础上,通过单击快捷菜单中的“删除监视”命令删除。当调试执行时,列表框内将显示输入的所有监视表达式的名称、当前值及类型,如图 4.18 中“监视”窗口所示。

3. “调用堆栈”子窗口。

“调用堆栈”子窗口用于显示当前处于执行状态的模块。如果当前正在执行的模块是一个被调用模块,那么在该子窗口中将显示调用该模块的主程序及所有上层模块和当前被调用模块的名称,如图 4.18 中“调用堆栈”窗口所示。

4. “调试输出”子窗口。

当需要观察中间结果的时候,可以在程序中安置一些 Debugout 命令,具体格式为:

Debugout <表达式>

当程序执行到该命令时,将计算表达式的值并将其结果输出到“调试输出”子窗口,如图 4.18 中的“调试输出”窗口所示。

另外,选择调试器窗口菜单“文件/另存输出”命令,可以将“调试输出”子窗口的内容保存到指定的文本文件中。选择调试器窗口菜单“文件/清除输出窗口”命令,可清除该窗口内容。

5. “局部”子窗口

“局部”子窗口用于显示某个模块中的内存变量(简单变量、数组、对象)的名称、类型和值,如图 4.18 中的“局部”窗口所示。

从“位置”下拉列表框中选择一个模块,在下方的列表框中将显示该模块内有效(可视)的内存变量的当前情况。要控制在列表框中显示的变量种类,可右击列表框窗口,从快捷菜单中选择公共、局部、常用、对象等。

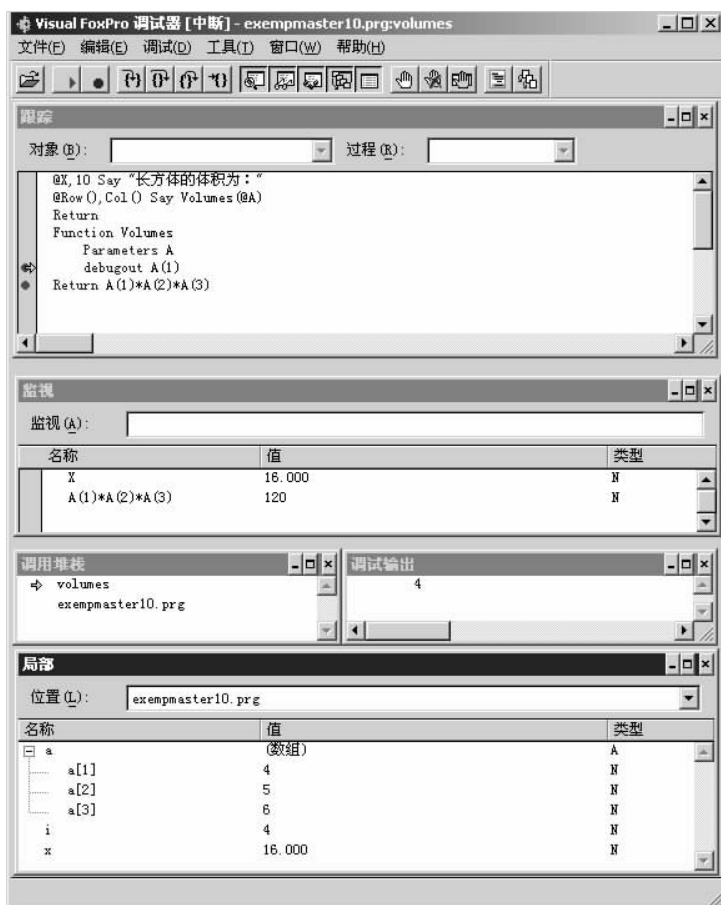


图 4.18 VFP 调试器窗口

4.5.6.2.2 调试器菜单

程序调试一般可以采用中断调试和单步调试两方法。其中,运行错误多采用中断调试;逻辑错误则多采用单步调试。为便于程序调试,VFP 提供了许多有力的调试工具,在调试程序时往往需要以各种方式执行程序。可以从调试器的工具栏或“调试”菜单选择调试程序的方法。“调试”菜单如图 4.19 所示,“调试”菜单项的功能如表 4.3 所示。



图 4.19 “调试”菜单

这里需要说明的是:当不涉及到子程序或过程时,“跳出”、“单步”和“单步跟踪”的效果都是一样的,即单步执行一条语句。

表 4.3 “调试”菜单项的功能

菜单项	意 义
运行或继续执行	开始执行在跟踪窗口打开的程序,或从当前代码行开始执行跟踪窗口中的程序,遇到断点时暂停执行。
取消	终止当前程序的调试,并将被调试的程序从调试器窗口中清除。
定位修改	终止当前程序的调试,并将被调试的程序从调试器窗口中清除,然后自动切换到命令文件编辑窗口,文件编辑器中光标自动停留的位置与程序调试器程序关闭时光标停留的程序位置相对应。
跳出	在跟踪被调用模块时不再进行单步跟踪,而以连续方式继续执行完被调用模块的剩余代码后,在调用程序的调用语句的下一行处中断。
单步	逐行执行代码,如果执行的语句正好是过程或函数等调用时,调用将其作为一条独立的命令单步执行,不进入过程进行跟踪。
单步跟踪	逐行执行代码,如果执行的语句正好是过程或函数等调用时,进入被调用模块,并单步跟踪被调用模块中的语句。
运行到光标处	可以将光标设置在任意命令行,然后选择该命令,程序将从上一次的断点执行到光标所在的语句行。
调速	为了能够动态清楚地观察程序的执行过程,可以选择“调速”命令打开“调整运行速度”对话框,设置两条语句之间执行延时的秒数。
设置下一条语句	程序中选择该命令可以使光标当前所在行成为恢复执行时马上执行的语句。

4.5.6.2.3 断点的设置

VFP 中可以设置 4 种类型的断点,下面分别予以简单介绍。

1. 普通断点。

设置普通断点是调试程序中最常用的手段,程序调试执行到这类断点处将无条件中断或暂停。设置或取消普通断点的方法有二种:

方法一:用鼠标双击要设置或取消断点的程序行左侧的灰色区域;

方法二:按 F9 键可对当前代码行设置或取消断点。

说明:断点只能设置在可执行语句上,当在非可执行语句上设置断点时,断点将自动设置在该语句之后的第一条可执行语句上。

2. 条件定位断点。

条件定位断点也是在固定位置设置,但只有执行到断点处条件为真值时,程序才会中断。设置条件定位断点的步骤是:

- (1) 在指定位置处设置普通断点,如将主程序的第七行设置为普通断点;
- (2) 单击调试器的菜单“工具/断点”,打开“断点”对话框;
- (3) 在“类型”下拉列表框中选择“如果表达式值为真则在定位处中断”;
- (4) 在“断点”列表框中选择断点,“定位”和“文件”编辑框中将自动显示相关内容;
- (5) 在“表达式”编辑框中输入断点生效条件表达式,如 $I=2$,如图 4.20 所示;
- (6) 单击“添加”按钮,将新设置的条件定位断点添加到“断点”列表框中;

(7) 在“断点”列表框中,将该行的其他断点删除或设置为无效(除刚才添加的断点外的断点,即将图 4.20 中“断点”列表框中的第一行断点删除或设置为无效);



图 4.20 设置条件断点

(8) 单击“确定”按钮,完成条件定位断点的设置。

说明:条件断点设置过程中,单击添加按钮是将设置好的条件断点添加到“断点”列表框中,列表框中还保留有该断点原设置的普通断点,所以若要仅仅只让刚刚设置的条件断点生效,则必须删除原普通断点或使其无效。另,暂时无效的断点将显示为红色空心圆圈。

3. 条件断点。

条件断点不固定位置,只要设置的条件(逻辑表达式)为真,程序可以在任何位置中断。设置条件断点的步骤是:

- (1) 单击调试器的菜单“工具/断点”,打开“断点”对话框;
- (2) 在“类型”下拉列表框中选择“当表达式值为真时中断”;
- (3) 在“表达式”编辑框中输入断点生效条件表达式;
- (4) 单击“添加”按钮,将设置的条件断点添加到“断点”列表框中;
- (5) 单击“确定”按钮,完成条件断点的设置。

4. 表达式断点。

表达式断点是只要表达式的值改变了就发生中断,可以是任意表达式,程序可以在任何位置中断。设置表达式断点的步骤是:

- (1) 选择调试器的“工具”→“断点”菜单,打开“断点”对话框。
- (2) 在“类型”下拉列表框中选择“当表达式值改变时中断”;
- (3) 在“表达式”编辑框中输入断点生效条件表达式;
- (4) 单击“添加”按钮,将设置的表达式断点添加到“断点”列表框中;
- (5) 单击“确定”按钮,完成表达式断点的设置。

说明:由于条件断点和表达式断点没有固定位置,所以也不会出现与普通断点类似的圆点。

本章在介绍一些程序命令时,为了方便大家对命令的理解,列举了一些简单的示例,

因此涉及到表或数据库的示例很少,下面以一个比较复杂的示例,来对本章的内容作一个总结。

例 4.49 设计一段程序,要求可以实现三个方面的功能:

1. 通过提示,让用户可以创建新表;
2. 通过提示,让用户选择打开表;
3. 向表中添加记录。

源文件 ExempTable. prg 的源程序代码如下:

* ExempTable. prg &&. 创建用户选择菜单,通过判断用户的选择来调用不同的过程

* 来完成相应的任务

Set Talk Off

Do While . T.

 Clear

 @8,10 Say "1> 创建一个新表"

 @Row()+2,10 Say "2> 打开表"

 @Row()+2,10 Say "3> 向表中添加记录"

 @Row()+2,12 Say "创建一个新表,请按数字键 1;打开表,;

 请按数字键 2;向表中添加记录,请按数字键 3;按其他键则退出。"

 @Row()+2,12 Say "请选择:" Get Y Default " " Picture "9"

 Read

 Do Case

 Case Y="1"

 Do Createtable

 Case Y="2"

 Do Opentable

 Case Y="3"

 Do Apprecord

 Otherwise

 Exit

 Endcase

Enddo

Use

Return

Procedure Opentable &&. 打开指定的表

 Clear

 @6,10 Say "请输入要打开的表名:" Get Tablename Default Space(20)

 Read

 Tablename=Alltrim(Tablename)


```

Use &.Tablename          &.&. 利用宏替换函数打开不同的表
Endproc

Procedure Apprecord        &.&. 向打开的表中添加记录
Clear
Do While . T.
    Append Blank
    @8,10 Say "请输入新的记录:"
    For I=1 To Fcount()    &.&. 通过循环打开表中所有字段,
                           &.&.Fcount()函数返回当前表的字段数
        Fieldname=Field(I) &.&. Field(I)函数返回第 I 个字段的字段名称
        @Row()+2,10 Say Fieldname+" ":" Get &.Fieldname
    Next
Read
Yn=Messagebox("是否继续添加新的记录?",4+32+0,"添加记录")
If Yn#6
    Exit
Endif
Enddo
Return
Endproc

Procedure Createtable      &.&. 创建一个新表
Clear
@8,10 Say "请输入新表名称:" Get Tablename Default Space(20)
X=Row()
Y=Col()
Read
If Len(Alltrim(Tablename))=0
    Messagebox("新表名不能为空!")
    Return
Endif
@X+2,0 Say "请输入新表的字段个数:" Get Fieldnumber Default 0 Picture "99"
X=Row()
Y=Col()
Read
@X,Y
Dimension Fieldname(Fieldnumber),Fieldtype(Fieldnumber),;

```

```

Fieldwidth(Fieldnumber),Fielddecimal(Fieldnumber)      &&. 定义数组
Fieldname=Space(10)                                     &&. 给数组赋初值
Fieldtype=Space(1)                                       &&. 给数组赋初值
Store 0 To Fieldwidth,Fielddecimal                      &&. 给数组赋初值
For I=1 To Fieldnumber
    @Row()+2,10 Say "请输入新表的第"+Alltrim(Str(I))+ "个字段名:" ;
    Get Fieldname(I)
    @Row(),Col()+4 Say "请输入第"+Alltrim(Str(I))+ "个字段的类型:" ;
    Get Fieldtype(I)
    X=Row()
    Y=Col()
    Read
    @X,Y
    If Len(Alltrim(Fieldname)) # 0 And ;
        Inlist(Upper(Fieldtype(I)),"C","N","D","L","M","G")
        Do Case
        Case Upper(Fieldtype(I))="C"                      &&. 处理字符型字段
            @Row(),Col()+4 Say "请输入第"+Alltrim(Str(I))+ ;
            "个字段的宽度:" Get Fieldwidth(I)
            X=Row()
            Y=Col()
            Read
            @X,Y
            If Fieldwidth(I) >= 1 And Fieldwidth(I) <= 254
                If I=1 Then                                &&. 当输入的是表的第一个字段时,创建表
                    Create Table &.Tablename (&.Fieldname(I);
                    &.Fieldtype(I);(Fieldwidth(I)))
                Else                                       &&. 当输入的不是表的第一个字段时,向表添加字段
                    Alter Table &.Tablename Add &.Fieldname(I);
                    &.Fieldtype(I). (Fieldwidth(I))
                Endif
            Else
                MessageBox("字段长度不能为 0 !")
            Return
        Endif
    Else
        Case Upper(Fieldtype(I))="N"                      &&. 处理字数值型字段
            @Row(),Col()+4 Say "请输入第"+Alltrim(Str(I))+ ;
            "个字段的宽度:"Get Fieldwidth(I)

```

```

@Row(),Col()+4 Say "请输入第"+Alltrim(Str(I))+
    "个字段的小数位:"Get Fielddecimal(I)
X=Row()
Y=Col()
Read
@X,Y
If Fieldwidth(I)>=1
    If I=1 Then
        Create Table &.Tablename (&.Fieldname(I) ;
            &.Fieldtype(I)(Fieldwidth(I),Fielddecimal(I)))
    Else
        Alter Table &.Tablename Add &.Fieldname(I) ;
            &.Fieldtype(I). (Fieldwidth(I),Fielddecimal(I))
    Endif
Else
    Messagebox("字段长度不能为 0 !")
    Return
Endif
Otherwise                                &&. 处理除字符型和数值型字段以外的字段
    If I=1 Then
        Create Table &.Tablename (&.Fieldname(I) &.Fieldtype(I))
    Else
        Alter Table &.Tablename Add &.Fieldname(I) &.Fieldtype(I)
    Endif
Endcase
Else
    I=I-1
Loop
Endif
Next
Return
Endproc

```

上例是一段针对普通表而设计的程序,其特点是不局限于某一个表,如可以打开任何 VFP 表,向任何已经打开的表添加涉及该表所有字段的新记录,也可以创建任何结构的 VFP 表。但上例是以实现简单的通用功能为目的,同时为了让程序显得更简单一些,并便于阅读,也仅仅只使用了三个小的功能模块,实现比较简单的任务,另外还省略了大量的数据验证。所以在运行上述程序时,请输入正确的数据,否则程序会出现错误。

另外,目前 VFP 的此类程序并不采用这种方式完成,而是利用表单就可以方便的实

现,在这儿列举上例,其目的是为了让大家对结构化程序设计有一个比较全面的了解,上例中各模块的功能划分非常明确,每个模块只完成一个简单的任务,将几个简单的模块按照一定规律结合在一起就形成了一个比较复杂的程序。关于程序设计中出现的一些特殊函数,在程序的命令行中已经进行了注释说明,若要了解这些特殊函数更详细的功能,请查阅其他相关参考资料。

习 题

一、选择题

- Visual FoxPro 6.0 的程序文件可以由()命令建立或修改。
A. Modify Command B. Edit Command
C. Change Command D. Do Command
- 在项目管理器中,程序存放在()页中。
A. 数据 B. 文档 C. 类 D. 代码
- 下列不正确的赋值语句是()。
A. $X=25$ B. Store 35 X C. $'X'=24$ D. Input 'X' To X
- 假定 $X=4$ 执行命令 $?X=X+1$ 后,结果是()。
A. 4 B. 5 C. . T. D. . F.
- 在下列语句中,不是输入命令的是()。
A. ? B. Input C. Accept D. @... Say/Get
- 在当前位置处输出结果不回车换行的命令是()。
A. ? B. ?? C. Output D. Print
- 下列语句中()是分支语句。
A. If... Endif B. For... Endfor
C. Do While... Enddo D. Scan... Endscan
- 对 Do Case Endcase 语句,下列说法中正确的是()。
A. 执行第一个语句组 B. 执行满足条件的第一个语句组
C. 执行满足条件的第一个语句 D. 执行 Otherwise 后面的语句组
- 对于 For I=1 To 8 Step2 循环,正常退出循环后的 I 值为()。
A. 7 B. 8 C. 9 D. 10
- 在 Do While... Enddo 循环中,若循环条件设置为 . T. ,则下列说法中正确的是()。
A. 程序无法跳出循环 B. 程序逻辑不会出现死循环
C. 用 Exit 可跳出循环 D. 用 Loop 可跳出循环
- 针对表的循环语句是()。
A. For... Endfor B. Dowhile... Enddo
C. Scan... Endscan D. Do Case... Endcase
- 将数组的内容传送到表的命令是()。

- A. Dimension B. Append From
C. Copy To Array D. Release
13. 在 Visual FoxPro 6.0 中,用 Dimension A(3,4)定义数组后,数组 A 包含的数组元素〔下标变量〕的个数是()。
A. 3 个 B. 4 个 C. 7 个 D. 12 个
14. 过程的引导语句是()。
A. Procedure B. Program C. Parameters D. Function
15. 函数的返回值是跟在()之后。
A. Return B. Lparameters C. Parameters D. Function
16. 定义全局变量的命令是()。
A. Public B. Local C. Dimension D. Private
17. 下列关于参数传递不正确的说法是()。
A. 传值的方式是“单向”的,即只能由实参传递给虚参,而虚参不能返回给实参。
B. 传址的方式是“双向”的,即实参变量和虚参变量被分配为同一存储单元。
C. 将实参变量用括号括起来为传值方式,在实参变量前加“@”符号则为传址方式。
D. 在默认的情况下,VFP 在调用子程序时采用传址方式。
18. 关于方法不正确的说法是()。
A. 方法名必须为 Method。
B. VFP 中方法分为内部方法和自定义方法。
C. 可以在事件过程或其他的方法程序中调用自定义方法。
D. 方法代码的格式同过程和函数,其值由 Return 返回。
19. 关于变量不正确的说法是()。
A. 全局变量在所有程序中都有效,都可以使用。
B. 局部变量只能在创建它们的过程内部使用和修改。
C. 私有变量在定义它的程序及它所调用的子程序范围内有效,即本层及下层有效。
D. 系统默认定义的变量都属于全局变量。
20. 在用户自定义函数或过程中设置参数,应使用()命令。
A. Procedure B. With C. Parameters D. Function

二、判断题

1. Input 命令只能接收字符串。 ()
2. Wait 命令只能接收一个字符。 ()
3. ? 换行输出, ?? 不换行输出。 ()
4. If 后的条件表达式可以不是逻辑表达式。 ()
5. 在 Do Case 语句中,若所有条件不满足,则执行 Otherwise 后面的语句组。 ()
6. For 循环的结尾可以使用 Next 语句。 ()
7. 循环可以嵌套,各循环之间也可以交叉。 ()
8. 数组的下标用圆括号或方括号括起来,每个数组元素必须具有相同的数据类型。 ()

9. 过程可以返回值也可以不返回值,函数可以返回值也可以不返回值。 ()

10. 自定义方法可以在事件过程或其他的方法程序中调用。 ()

三、阅读下列程序,写出运行结果。

程序 1

```
Set Talk Off
S=0
P=5
Do While P<=10
    P=P+1
    S=S+P*2
Enddo
?S
Return
```

运行结果:

程序 2

运行下列程序,若依次输入数据 2.5, 8, 2.5, 3, 2, 2, 10 请写出输出结果。

```
Set Talk Off
I=1
Do While I<=2
    Clear
    Input "A=" To A
    If A>Int(A). Or. A>=10
        Loop
    Else
        Input "B=" To B
        If B=Int(B). And. B<10
            Loop
        Else
            ?A,"+",B,"=",A+B
        Endif
    Endif
    I=I+1
Enddo
Return
```

运行结果:

程序 3

```
* * 主程序 PROG. PRG * *
```

```
Clear
```

```
Set Talk Off
```

```
X="同学们"
```

```
Y="你们好 !"
```

```
?"主程序中 X=",X
```

```
?"主程序中 Y=",Y
```

```
DO SUBPRO
```

```
?"返回主程序后 X=",X
```

```
?"返回主程序后 Y=",Y
```

```
?"返回主程序后 Z=",Z
```

```
Set Talk On
```

```
Z=300
```

```
?"子程序中 X=",X
```

```
?"子程序中 Y=",Y
```

```
?"子程序中 Z=",Z
```

```
Return
```

```
* * 过程 SUBPRO * *
```

```
Procedure Subpro
```

```
Private X
```

```
Public Z
```

```
X=100
```

```
Y=2
```

```
Return
```

运行结果：

程序 4

```
* MAIN. PRG
```

```
Set Talk Off
```

```
A=3
```

```
B=5
```

```
Do Pp With 2 * A,B
```

```
Set Talk On
```

```
Return
```

```
Procedure Pp
```

```
Parameters X,Y
```

```
Clear
```

```
S=X * Y
```

```
?"X="+Str(S,3)
```

```
Return
```

运行结果：

程序 5

```
* Main Program
```

```
Set Talk Off
```

```
Store 3 To L,H
```

```
Area=0
```

```
Do Sub With L,H,Area
```

```
  ?L,H,Area
```

```
Do Sub With L,H,Area
```

```
  ?L,H,Area
```

```
Return
```

```
* Sub
```

```
Procedure Sub
```

```
Parameters A,B,C
```

```
C=A * B
```

```
A=A * 2
```

```
B=B * 2
```

```
Return
```

运行结果：

程序 6

```
Set Talk Off
```

```
Clear
```

```
Dimension B(5)
```

```
B=5
```

```
I=1
```

```
S=0
```

```
Do While I<5
```

```
  S=S+B(I)
```

```
  I=I+2
```

```
Enddo
```

```
?"S,I=",S,I
```

```
Return
```

运行结果：

程序 7

Set Talk Off

Clear

Store 0 To X,Y,S1,S2,S3

Do While X<10

X=X+1

Do Case

Case Int(X/2)=X/2

S1=S1+X/2

Case X % 3 = 0

S2=S2+X/3

Case Int(X/2)<>X/2

S3=S3+1

Endcase

Enddo

?S1,S2,S3

Set Talk On

Return

运行结果：

四、阅读分析下列程序,在_____线处填上适当的内容,使程序完整。

1. 下列程序是用来求长方形面积,请将它写完整。

X=3

Y=5

S=Area(X,Y)

?"S=",S

Function Area

_____1_____

S1=X * Y

Return _____2_____

2. 设有图书表 TSH,包括字段(总编号、分类号、书名、作者、出版单位、单价);读者表 DZH(借书证号、姓名、性别、单位、职称、地址);借阅表 JY(借书证号、总编号、借阅日期、备注)。下面程序的功能是打印已借书读者的借书证号、姓名、单位,以及借阅图书的书名、单价、借阅日期,请阅读程序并填空。

Set Talk Off

Select 1

Use Dzh

```

Index On 借书证号 To Dshh
Select 2
Use Tsh
Index On 总编号 To Shh
Select 3
Use Jy
Set Relation To 借书证号 Into A
_____ 1
List _____ 2 To Print
Close All
Return

```

3. 已给定程序 MOD11.PRG 的功能是:按每屏十条记录,顺序显示数据表 G01.DBF 职工的记录。若显示完所有记录后,则显示提示信息“显示完毕!”,否则按任意键继续显示下一屏。

填空要求:请删除 * * * found * * * 下面一行中的下划线,并在原下划线处填入正确的内容,并调试运行。

注意:要求在指定位置修改,不得增加或删减程序行。

```

Set Talk Off
Clear
P=0 && 变量 p 记录翻页次数
Use G01
Do While . Not. Eof()
    P=P+1
    L=5
    Clear
    * * * * * Found * * * * *
    @ 2,30 Say "第"+Str(_____,2)+" 屏"
    * * * * * Found * * * * *
Do While _____
    @ L,10 Say 工号
    @ L,20 Say 姓名
    @ L,30 Say 性别
    @ L,40 Say 基本工资
    @ L,50 Say 附加工资
    @ L,60 Say 交通费
    @ L,70 Say 职称
    * * * * * Found * * * * *

```

```

If Eof()
    ?"显示完毕 !"
Use
Return
Endif
L=L+1
Enddo
?
Wait"按任意键继续下一屏"
Enddo
Use
Set Talk On
Return

```

五、编程题

1. 键盘输入两个数 a、b, 如果两数相等, 则显示 $a=b$ (如 $5=5$), 如果两数不等, 如 3 和 5, 则显示 $3<5$ 或 $5>3$ 。

2. 表 Reportcard.dbf 的结构为学号(C,10)、应用文(N,3)、高等数学(N,3)、计算机基础(N,3)、VB 程序设计(N,3)、平均分(6,2)、总分(N,3)、名次(N,2)。试设计一个程序完成以下功能:

- (1) 根据每条记录的成绩, 计算每个同学的平均分、总分和名次;
- (2) 分别用 DO 循环、FOR 循环和 SCAN 循环求出平均不及格的人数;
- (3) 建立索引, 将表中的所有记录按总分从高到低的顺序显示出来。

3. 编程计算 0 到 100 之间能被 3 整除的奇数个数及其累加和。

4. 数列的前两个数是 1、2, 从第三个数开始以后的每个数都是其前两个数之和。请编写一段程序, 要求输出此数列从第一个数开始的前 30 个数。

5. 编制一个通用的交换记录的程序, 可以对换任意表中的任意两个记录的内容。

6. 编制一个程序可以向表中指定位置的记录位置插入一个新记录, 位置和新记录内容由用户输入。

7. 根据键入的 X 值, 计算下面分段函数的值, 并输出结果。

$$Y = \begin{cases} 5x^2 + 6x - 1 & (X \leq 0) \\ x^2 - 4xc + 1 & (0 < X \leq 20) \\ 3x^2 + 1 & (X > 20) \end{cases}$$

8. 用下列级数的前 21 项之和计算自然对数之底 e 的近似值。

$$E = 1 + \frac{1}{1!} + \frac{2}{1!} + \frac{3}{1!} + \cdots + \frac{1}{20!}$$

9. 运用带参调用的方式分别采用子程序、过程、自定义函数, 编程计算:

$$C_m^n = \frac{m!}{n!(m-n)!}$$

10. 现有空数据表 B01.DBF 有数据 1(C,2)和数据 2(N,2)两个字段。请编写程序 Prog1.prg,其功能是:向表中添加 26 条记录,其中:“数据 1”字段各记录的数据从字母 A——Z 依次取值,即 A,B,C,...,Y,Z;在“数据 2”字段中填入相应的 ASCII 码值,再根据 B01.DBF 复制出表 B02.DBF。请调试并运行该程序。

提示:字母 A 的 ASCII 码为 65,取字符的函数为 CHR()。

11. 试设计一个学生成绩管理系统,使其具有查询、统计、显示和修改表文件内容的功能,具体要求如下:

(1) 程序由一个主控模块和 4 个子模块:查询、统计、显示和修改组成;

(2) 能够按照给定的单一条件或复合条件查询、显示一个或多个记录内容及进行修改与统计。



第 5 章

表单设计基础

经过第四章程序设计概述的学习,我们已经掌握了 Visual FoxPro 6.0 的基本编程知识,甚至可以编写一个较复杂的程序;但一个应用程序的好坏,给用户的第一印象并不是程序代码的好坏、性能运行效率的高低,而是用户界面是否友好。运用前面学习的知识进行界面设计,设计过程非常复杂,而且容易出错,为此,Visual FoxPro 6.0 提供了强大的表单设计功能,表单是 VFP 提供的用于建立应用程序界面的最主要的工具之一,最常见的界面,各种对话框和窗口等都属于表单的不同表现形式。

从本章起将逐步系统的介绍一些面向对象的程序设计方法,表单设计就属于面向对象程序设计的一部分内容。用表单设计的方法来实现数据库信息的显示、输入、编辑等功能,其设计过程非常简便,设计过程中的大部分代码,都由 VFP 系统自动生成。表单的设计过程就是程序界面的设计过程。

5.1 表单设计器

表单的设计是主要通过 VFP 提供的表单设计器来实现的,表单设计器中的表单实际上是一个容器,在上面放置了一些需要的控件(所谓控件,就是设计置在一个表单上用以显示数据、执行操作或使表单更易阅读的一种图形对象),并对这些控件的属性进行一些设置,让这些控件的显示形式以及在程序中运行时他们的一些行为等符合程序设计的要求。

表单的创建方法,除了可以利用表单设计器来实现外,VFP 还提供了另外两种更为简单的创建表单的方法,如利用表单向导和快速表单来创建表单。通常利用表单向导和快速表单设计出的表单其功能比较简单,有时往往不能满足需要,所以在表单的设计过程中,通常也有这样一种设计方法,即先利用表单向导或快速表单创建一个表单,然后再在表单设计器中对创建的表单进行修改的设计方法。下面我们将分别介绍这几种创建表单的方法。

5.1.1 表单向导

表单向导以交互方式引导用户选定表来产生实用的表维护窗口,窗口中含有用户所选择的字段,还包含供用户操作的各种按钮,这些按钮分别具有翻页、编辑、查找、打印

等功能。

在 VFP 中,有两种类型的表单向导:表单向导和一对多表单向导,表单向导适用于创建单个表的表单,一对多表单向导适用于创建具有一对多关系的两个表的表单。

5.1.1.1 打开表单设计向导选取对话框的方法

方法一:单击系统菜单“文件/新建...”命令或单击工具栏“新建”按钮,在弹出的如图 5.1 所示的对话框中选定“表单”项,然后单击“向导”按钮,弹出如图 5.2 所示的“向导选取”对话框;



图 5.1 新建对话框

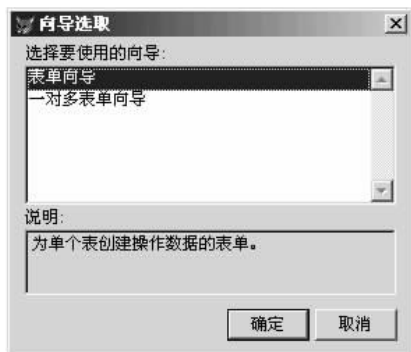


图 5.2 向导选取对话框

方法二:单击系统菜单“工具/向导/表单”命令,或单击工具栏中的“表单”按钮,弹出“向导选取”对话框;

方法三:选择项目管理器“文档”选项卡中的“表单”项,单击“新建...”,在弹出的“新建表单”对话框中,单击“表单向导”,如图 5.3 所示,弹出“向导选取”对话框。



图 5.3 在项目管理器中打开表单向导

在实际的软件开发中,通常都是在项目管理器中添加表单,这样便于管理。

5.1.1.2 利用表单向导创建表单

这里我们通过一个例子来介绍利用表单向导来创建表单的方法。

例 5.1 使用表单向导创建一个维护职工基本信息表(文件名:Employee.dbf)的表单。

操作步骤:

1. 按前面介绍的三种方法之一,打开如图 5.2 所示“向导选取”对话框,然后单击“确定”按钮,这时屏幕出现如图 5.4 所示的“表单向导”对话框;



图 5.4 数据表选择

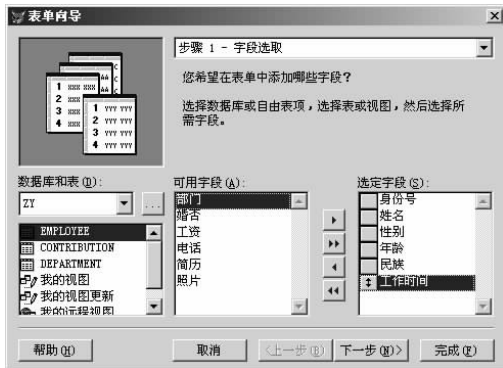


图 5.5 字段选择

2. 单击“数据库和表”列表框旁的按钮,在弹出的打开文件对话框中选择 D:\vfp 文件夹下的“Employee.dbf”数据表。单击“确定”按钮,打开数据表“Employee”;如图 5.5 所示;

3. 在选定的数据表中,选取表单所需的字段,如图 5.5 所示。在本例中选择全部字段,字段选定后,单击“下一步”按钮,即进入如图 5.6 所示的“选择表单样式”对话框;



图 5.6 选择表单样式对话框

4. 在“选择表单样式”对话框中,用户可以在“样式(S)”列表框中选定表单的样式;在“按钮类型”单选按钮中选定按钮形状,在这里“样式”选定为“浮雕式”,按钮类型选定为“文本样式”,如图 5.6 所示。完成后,单击“下一步”按钮,进入“排序次序”对话框,如图 5.7 所示;



图 5.7 选择排序字段

5. 在“排序次序”对话框中用户可以选定记录的排序方法，其中包括关键字段、升序和降序的选择，在本例中选择按“姓名”字段升序排序记录。然后单击“下一步”，进入如图 5.8 所示的“完成”对话框；



图 5.8 完成

6. 在“完成”对话框中用户可以选择表单的保存方式、输入表单标题。这里保存方式选择“保存表单并用表单设计器修改表单”，在请键入表单标题文本框里键入“职工基本情况”；用户可以单击“预览”命令按钮来预览一下表单；如果不满意，可以单击“上一步”命令按钮返回前面的屏幕重新选择，否则单击“返回向导”按钮返回“完成”对话框，单击“完成”按钮，进入“另存为”对话框；

7. 在弹出的“另存为”对话框中，选择保存的文件夹，并将表单命名为“职工情况”（扩展名为 scx），单击“保存”后，新建的表单在如图 5.9 所示的表单设计器显示，在这里可以进一步对表单进行修改。例如调整各个控件的大小、布局等。在这里我们重新调整了控件的布局。

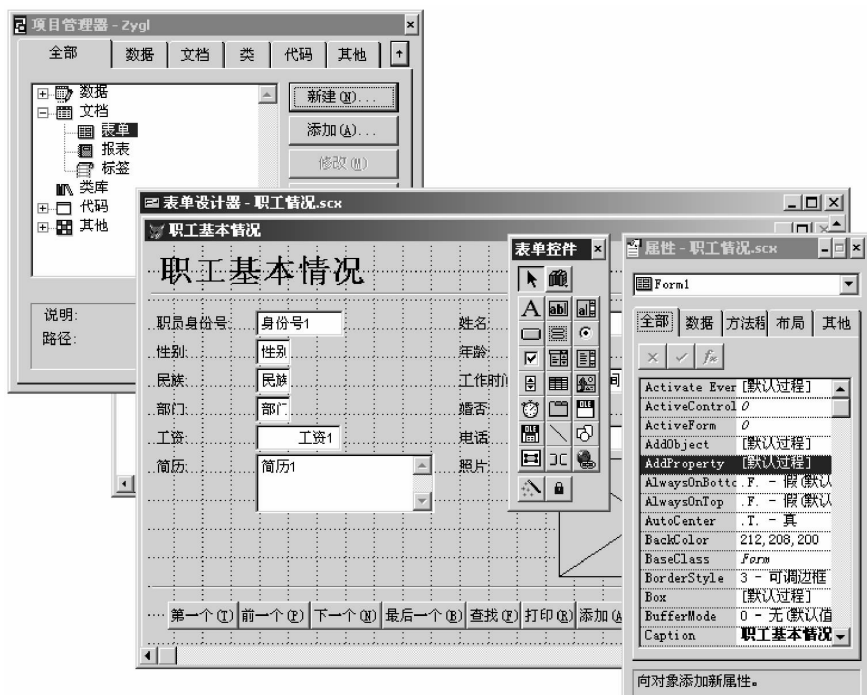


图 5.9 在表单设计器中打开的表单

8. 运行表单。要在表单设计器中运行表单,请选择“表单”菜单,点击“执行表单”菜单项,或点击常用工具栏上的运行按钮,也可以按快捷键 Ctrl+E,运行界面如图 5.10 所示。



图 5.10 运行表单后的界面

5.1.1.3 利用表单向导创建一对多表单

这里我们通过一个例子来介绍利用一对多表单向导来创建表单的方法。

例 5.2 使用一对多的表单向导,创建一个维护职工信息表(文件名:Employee.dbf)和工作部门信息表(文件名:Department.dbf)的表单。

操作步骤：

1. 要建立一对多表单，在图 5.2 所示的表单向导对话框中，选择“一对多表单向导”，单击“确定”按钮，进入向导的步骤一，从父表 (Department. dbf) 中选择要显示的字段，如图 5.11 所示；



图 5.11 从父表中选择字段

2. 单击“下一步”，进入步骤二，从子表 (Employee. dbf) 中选择要显示的字段，如图 5.12所示；



图 5.12 从子表中选择字段

3. 确定了父表和子表后，单击“下一步”，进入步骤三，建立表之间的关系。对于自由表而言，要在两个表中分别选择两个字段，建立两个表之间的关联。两个表建立关联后，父表中的记录和子表中的记录对应，子表中显示的记录是和当前父表中显示的记录相关联的记录；即“父表 (Department. dbf)”里的一条记录对应“子表 (Employee. dbf)”里的多

条记录,如图 5.13 所示:



图 5.13 建立表之间的关系

4. 确定了表的关联后,单击“下一步”,弹出与前例相同的设计过程,如确定表单样式、字段排序等,最后完成,并将表单名设置为“各部门职员情况表”保存。可以运行表单,看看运行结果,如图 5.14 所示;



图 5.14 一对多表单运行结果

一对多表单运行时,将在表单的上部每次显示父表的一条记录,在下部以表格的形式显示与父表相关的记录。底部的定位按钮仅对父表起作用。

使用表单向导可以快速地创建表单的初始模型,但用表单向导所创建的表单无论从布局还是功能上来说,还是比较粗糙的;为了改进表单布局和功能,可以用表单设计器来修改初始模型或修改用户已有的表单。

5.1.2 表单设计器的基本操作

5.1.2.1 表单设计器的启动

1. 界面方式。

方法一:单击系统菜单“文件/新建...”命令或单击工具栏“新建”按钮,在“新建”对话

框中选择“表单”后,单击“新建文件”按钮,如图 5.15 所示,弹出表单设计器窗口;



图 5.15 文件菜单启动表单设计器

方法二:选择项目管理器“文档”选项卡中的“表单”项,单击“新建...”,在弹出的“新建表单”对话框中,单击“新建表单”,如图 5.3 所示,弹出表单设计器窗口。

2. 命令方式。

格式一:

命令格式:Create Form [<表单文件名>]

功能:创建一个新表单。

参数说明:

- (1)指定的<表单文件名>存在时,在表单设计器中直接创建并复盖它;
- (2)省略<表单文件名>选项时,在表单设计器中创建名为文档 1 的表单。

格式二:

如果要对一个创建好的表单进行修改,则可以使用命令:

Modify Form [<表单文件名>]

功能:创建或修改表单。

参数说明:

- (1) 当指定的<表单文件名>存在时,在表单设计器中打开它;
- (2) 若指定的<表单文件名>不存在时,则在表单设计器中创建它;
- (3) 若省略<表单文件名>选项,则弹出打开对话框,当用户在打开对话框中指定了表单后,在表单设计器中打开。

例 5.3 要对例 5.1 中生成的名为“职工情况”的表单进行修改,则可以执行命令:

Modify Form 职工情况

若是以修改表单的形式打开表单设计器窗口,其表单设计器窗口形式可参阅图 5.9;若是以新建表单的形式打开表单设计器窗口,其表单设计器窗口形式可参阅图 5.16。

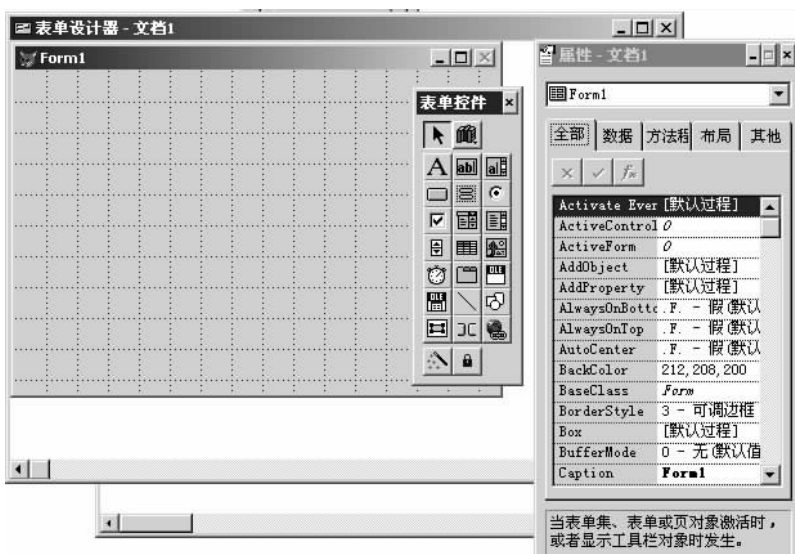


图 5.16 创建一个新表单

打开表单设计器后,系统菜单自动添加了“表单”项,同时激活表单设计器工具栏。当新建一个表单,并启动表单设计器后,系统会自动新建一个空白表单 Form1,如图 5.16 所示。用户可以在该表单上添加所需的表单控件,并通过设置所添加控件的相关属性,来完成表单设计。通过表单设计器工具栏中的一些相应的按钮,可以显示或隐藏如表单控件、属性等工具栏,这些工具栏可以帮助用户完成表单的设计。

表单设计器工具栏上常用按钮所对应的工具栏或窗口:

- (1) 表单控件工具栏:弹出“表单控件”工具栏,可向表单上添加创建各种控件;
- (2) 布局工具栏:可以设置表单的中控件的布局格式,如对齐方式、叠放次序等;
- (3) 调色板工具栏:弹出“设色板”工具栏,可设置指定控件的前景色和背景色;
- (4) 设置 Tab 键次序:显示表单中各对象 Tab 键的次序,并设置调整表单中对象的 Tab 键次序;
- (5) 数据环境:弹出“数据环境设计器”,可设置表单或表单集中使用的数据源,如:表、视图和关系等;
- (6) 属性窗口:弹出“属性窗口”,可设置各个控件的相关属性;
- (7) 代码窗口:弹出代码窗口,并显示和设置对象的事件代码;
- (8) 表单生成器:弹出“表单生成器”,选取字段和设置表单的样式;
- (9) 自动格式:弹出“自动格式生成器”,设置选定控件的样式。

以上工具除了可以利用“表单设计器”工具栏上的相应按钮来显示外,前七种工具还可利用单击系统菜单“显示”的相应命令,来显示相应的工具栏或窗口。

5.1.2.2 数据环境设计器

5.1.2.2.1 数据环境的概念

数据环境是指定义表单或表单集时使用的数据源,如表、视图和关系。数据环境一旦建立,在数据环境中已经设置的表或视图会随表单的打开或运行而打开,并随表单的关闭或释放而关闭。

5.1.2.2.2 数据环境设计器的作用及其打开

数据环境设计器可用来可视化地创建或修改数据环境。在打开表单设计器的前提下,利用三种方法可以打开数据环境设计器:

- (1)单击菜单“显示/数据环境”命令;
- (2)单击快捷菜单“数据环境”命令;
- (3)单击表单设计器工具栏的“数据环境”按钮。

在打开数据环境设计器时,会弹出“数据环境设计器”窗口,并在系统菜单中增加一个数据环境菜单。

5.1.2.2.3 数据环境设计器的快捷菜单与数据环境菜单

在数据环境菜单中提供了添加、移动、浏览等命令,他们的作用如下:

(1)添加:弹出“添加表或视图”对话框,供用户将相关的表或视图添加到数据环境设计器窗口中;

该窗口中的表或视图的显示形式与数据库设计器中表或视图的显示形式类似,但表与表之间仅仅只显示已经存在的永久关系连线。

用户也可以添加或删除两个表之间的关系连线,在数据环境设计器中添加关系的连线规则与数据库设计器中添加永久关系的规则有所不同,在数据环境设计器中添加连线的规则是:在数据环境设计器中,从父表的字段拖到子表的索引。删除关系的方式是:在选定关系连线后,按 Del 键删除连线。

(2)移去:在数据环境设计器窗口中移去一个选定的表或视图,移去的功能还可利用 Del 键实现,其方法是:在选定表或视图时,直接按 Del 键。被移去的表或视图仅仅是从数据环境中移出,而不会被删除;

(3)浏览:在浏览窗口中浏览选中的表或视图。

例 5.4 新建一个名为 Form1 的表单,并设置数据环境,要求在数据环境中包含两个表 Department,Employee。

操作步骤:

1. 新建一个表单。

在命令窗口中执行命令:Create Form Form1

2. 打开数据环境设计器。

在表单中右击鼠标,在弹出快捷菜单中,单击“数据环境”命令,或单击系统菜单“显示/数据环境”命令,打开“数据环境设计器”窗口;

3. 打开“添加表或视图”对话框。

在弹出的“数据环境设计器”窗口中,单击鼠标右键,在弹出的快捷菜单中选择“添加”菜单,则弹出“添加表或视图”对话框,如图 5.17 所示;



图 5.17 设置数据环境

4. 通过“添加表或视图”对话框向数据环境设计器添加表或视图。

在“添加表或视图”对话框中双击某个数据表,将选中的数据表加入到“数据环境设计器”中。若继续添加表或视图,可重复本步骤。单击“关闭”按钮,关闭“添加表或视图”对话框;

5. 删除数据环境设计器中的表或视图。

如果要删除“数据环境设计器”中的表或视图,首先选定要选中数据表或视图,然后单击鼠标右键,在弹出的快捷菜单中,单击“移出”菜单项,将该表从“数据环境设计器”中移去。

5.1.2.3 表单控件工具栏

5.1.2.3.1 表单控件工具中的辅助按钮

“表单控件”工具栏如图 5.18 所示,在该工具栏的第一行和最后一行共有四个辅助按钮,这些按钮的具体功能如下:

1. 选定对象。

当该按钮处于按下状态时,表示不可以在表单中创建控件,只能在表单中选定、编辑控件,如改变控件大小,位置等。当该按钮处于未按下状态时,表示可以在表单中创建控件。

2. 按钮锁定。

当该按钮处于按下状态时,并在“表单控件”工具栏中选定某个控件按钮后,每次可以在表单窗口中连续添加多个该类型的控件。当该按钮处于未按下状态时,在选定控件按钮后,每次只能向表单中添加一次相关控件,若需要再次添加时,则需要再次选定相关控件按钮。

3. 生成器锁定。

当该按钮处于按下状态时,每次向表单中添加控件时,系统会自动打开相应的生成器对话框,用户可以通过该对话框来设置相应控件的常用属性。



图 5.18 表单控件工具栏

对于表单中已经存在的控件,可以通过单击快捷菜单命令“生成器”,来打开相应的生成器对话框。

4. 查看类。

在可视化设计表单时,除了可以利用 VFP 提供的一些基类,还可以使用保存在类库中用户自定义的类,但使用前应将它们添加到“表单控件”工具栏中。关于用户自定义类的定义方法将在 5.2 面向对象的程序设计方法一节中介绍。

将一个类库文件中的类添加到“表单控件”工具栏的方法:单击工具栏上的“查看类”按钮,在弹出的菜单中单击“添加”命令,弹出“打开”对话框,在对话框中选定所需的类库文件(类库文件的扩展名为 vcx),单击“确定”按钮。

在向“表单控件”工具栏添加了类后,工具栏中将只显示添加的类。要让“表单控件”重新显示 VFP 的基类,可单击“查看类”按钮,在弹出的菜单中单击“常用”命令。

5.1.2.3.2 表单控件工具中的控件按钮

“表单控件”工具栏中除第一行和最后一行的按钮外,还有 21 个控件按钮,关于这些控件按钮产生的控件的具体设置,将在以后介绍各个控件时说明,这里只介绍向表单添加控件的一般方法:

方法一:不使用“表单控件”工具栏中的辅助按钮,向表单添加控件的步骤:

- (1)在“表单控件”工具栏上单击一个要添加的控件;
- (2)将鼠标移到表单窗口的合适位置单击鼠标或拖运鼠标,拖运鼠标可以在添加控件的同时完成控件大小的设置;
- (3)完成添加后,“表单控件”工具栏自动选择“选定对象”按钮。

方法二:在按下“生成器锁定”按钮的前提下,向表单添加控件的步骤:

- (1)在“表单控件”工具栏上点击“生成器锁定”按钮,使其处于凹陷状态;
- (2)在“表单控件”工具栏上单击一个要添加的控件;
- (3)将鼠标移到表单窗口的合适位置单击鼠标或拖运鼠标,当松开鼠标时,弹出与控件相对应的“生成器”对话框,根据对话框中的内容填写有关信息;
- (4)完成添加后,“表单控件”工具栏自动选择“选定对象”按钮。

方法三:在按下“控件锁定”按钮的前提下,向表单同时添加多个同类型控件的步骤:

- (1)在“表单控件”工具栏上单击“按钮锁定”,使其处于凹陷状态;
- (2)可根据方法一或方法二来创建控件;
- (3)用该方法与前两种方法不同之处是:当一个控件创建完成后,可直接在表单中连续创建同类型的其他控件。

5.1.2.4 布局工具栏

当表单上具有多个控件时,可使用“布局”工具栏对控件进行布局调整,如图 5.19 所示。

在使用“布局”工具栏时,要先选中多个控件,然后点击“布局”工具栏上所需的布局功能,完成控件的布局排版;同时选中多个控件的方法有两种:一是按住 Shift 不放,然后用鼠标点击需要布局的控件;二是拖动鼠标选中所需的控件。

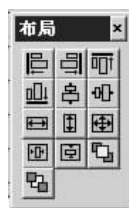


图 5.19 布局

布局中的按钮从上到下,从左到右依次的名称和功能如表 5.1 所示:

表 5.1 “布局”工具栏各按钮的功能

按钮	功能
左边对齐	让选定的所有控件沿其中最左边那个控件的左侧对齐
右边对齐	让选定的所有控件沿其中最右边那个控件的右侧对齐
顶边对齐	让选定的所有控件沿其中最顶端那个控件的顶边对齐
底边对齐	让选定的所有控件沿其中最下端那个控件的底边对齐
垂直居中对齐	使所有被选定的控件的中心处在一条垂直轴线上,该轴线位于所选控件的水平中心位置
水平居中对齐	使所有被选定的控件的中心处在水平一条轴线上,该轴线位于所选控件的垂直中心位置
相同宽度	调整所有被选定控件的宽度,使其宽度与其中宽度最宽控件的宽度相同
相同高度	调整所有被选定控件的高度,使其高度与其中高度最高控件的高度相同
相同大小	调整所有被选定控件的宽度与高度,使其宽度与其中宽度最宽控件的宽度相同,使其高度与其中高度最高控件的高度相同
水平居中	让被选定控件在表单内水平居中
垂直居中	让被选定控件在表单内垂直居中
置前	将被选定控件移到最前面,可能会覆盖其他控件
置后	将被选定控件移到最后面,可能会被其他控件覆盖

5.1.2.5 设置 Tab 键次序

当表单运行时,用户可以利用 Tab 键选择表单中的控件,让焦点在控件之间移去,但焦点的移动先后次序由控件的 Tab 次序决定,在没有设置 Tab 次序时,其次序通常默认为控件设置的先后顺序。但在设计过程中,默认的 Tab 键次序有时并不符合设计要求,这就需要调整设置 Tab 键次序。

VFP 提供了两种方式来设置 Tab 键次序:交互式方式和列表方式,用户究竟要选择那一种方式可以通过下面的设置来实现:

- (1)单击系统菜单“工具/选项”命令,弹出“选项”对话框;
- (2)在弹出的“选项”对话框中,选择“表单”选项卡;
- (3)在“表单”选项卡中的“Tab 键次序下拉式列表框中选择“交互”或“按列表”。

在交互式方式下,设置 Tab 键次序的步骤:

1. 单击系统菜单“显示/Tab 键次序”命令或单击“表单设计器”工具栏上的“设置 Tab 键次序”按钮,进入 Tab 键设置状态;在此状态控件的上方出现深蓝色的小方块,我们称之为 Tab 键次序盒,里面显示了相应控件的 Tab 键次序,如图 5.20 所示;
2. 按希望的顺序依次单击控件的 Tab 键次序盒;
3. 确认 Tab 键次序设置时,可单击表单空白处,或单击系统菜单“显示/Tab 键次序”命令、单击“表单设计器”工具栏上的“设置 Tab 键次序”按钮,退出 Tab 键设置状态;若放弃并退出 Tab 键设置状态,则直接按 Esc 键。

在列表方式下,设置 Tab 键次序的步骤:

1. 单击系统菜单“显示/Tab 键次序”命令或单击“表单设计器”工具栏上的“设置 Tab 键次序”按钮,打开“Tab 键次序”对话框,在该的对话框中以 Tab 键次序显示各控件,如图 5.21 所示;

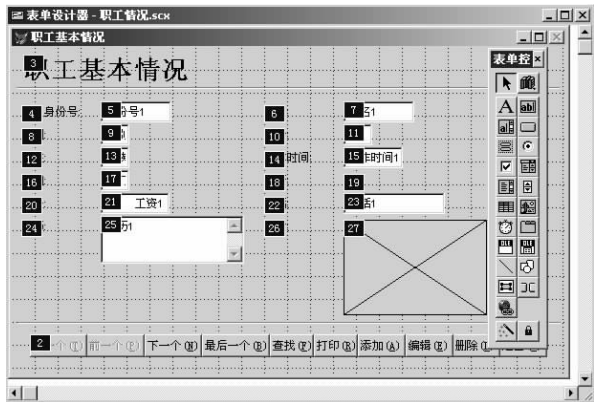


图 5.20 以交互式方式设置 Tab 键次序



图 5.21 按列表式方式设置 Tab 键次序

- 2. 通过拖运控件左侧的移动按钮来移动控件,改变控件的 Tab 键次序;
- 3. 单击“按行”按钮,将按各控件在表单上的位置从左到右,从上到下自动设置各控件的 Tab 键的次序;单击“按列”按钮,将按各控件在表单上的位置从上到下,从左到右自动设置各控件的 Tab 键的次序。

5.1.2.6 属性窗口

5.1.2.6.1 属性窗口

属性窗口如图 5.22 所示,一般包括对象组合框、属性设置框和列表框。其中对象组合框用来显示当前被选定的对象的名称。单击该组合框右侧的下拉箭头,将弹出当前表单及表单中所有对象的名称列表,用户可以选择需要编辑修改的对象或表单。

“属性”窗口的列表框显示当前被选定的对象的所有属性、方法和事件,用户可以选择需要编辑修改的属性、方法或事件。当选择的是属性项时,窗口内将出现属性设置框,用户可以在此对选择的属性进行设置。

属性窗口还包括五个选项卡,分别显示对象的属性、方法程序、布局等,各选项卡中的选项在列表框中以字母顺序排列。

“全部”选项卡:列出选定的表单或某个控件对象所有的属性、事件和方法程序。

“数据”选项卡:只列出选定的表单或某个控

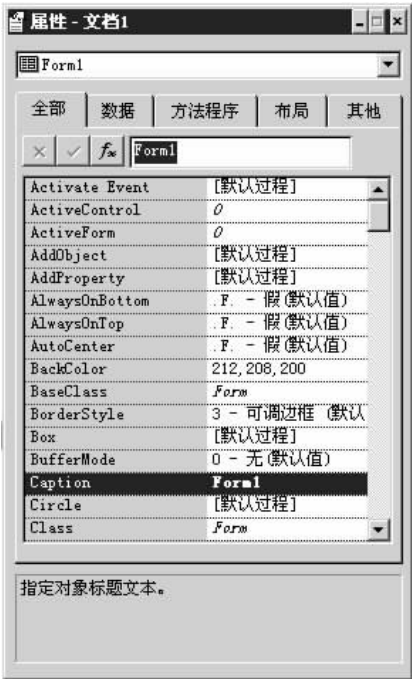


图 5.22 属性窗口

件对象显示或操纵数据的属性,如对象数据的源、输入格式、长度,字段是否只读等。

“方法程序”选项卡:只列出选定的表单或某个控件对象的方法程序和事件。其中有“Event”的项是事件,其余的是方法。

“布局”选项卡:只列出选定的表单或某个控件对象的位置、大小、颜色、可视性、字体,以及其他一些与对象外观相关的一些属性。

“其他”:只列出类信息、拖动方式、状态的允许与禁止,对象名以及其他的一些内容。

注意:在“属性”窗口中共有 60 多个属性,如果用户选定了多个控件,则这些控件的公共属性将显示在“属性”窗口中。要编辑某个对象的“属性”,可在“对象”框中选择该控件对象,或用鼠标在表单中选中该对象,这时“属性”窗口中将显示该控件对象的属性。在“属性”窗口中,我们会注意到有些属性的值是以斜体显示的,这表明该属性是只读属性,用户不能进行修改,只可以在程序中读取。除此之外,其他属性都是可以修改的。

5.1.2.6.2 对象的常用属性

属性(Properties):是指对象所具有的特征。每个对象都有一定的属性,不同类型的对象其属性也存在着差异。“属性”值可以在设计时设置,也可以在运行时更改,但有的“属性”是只读属性,不可改变。对象的常用属性见下表 5.2:

表 5.2 对象的常用属性

属 性	说 明	应用于
Caption	设置对象的标题(显示时对象显示的文本内容)	表单、标签、命令按钮等
Name	设置对象的名称(即在代码中引用该对象的名称)	任何对象
Value	设置当前控件当前取值	文本框、列表框、组合框等
ForeColor	设置对象的前景色(如文本和图形的颜色)	表单、标签、文本框、列表框、组合框等
BackColor	设置对象的背景颜色	
FontName	设置对象显示文本的字体	
FontSize	设置对象显示文本的字体的大小	
FontBold	设置对象的显示文本是否为粗体	
BackStyle	设置对象背景是否透明(透明时背景色无效)	标签、文本框、图像等
BorderStyle	设置对象的边框样式,如无边框、单线框等	表单、标签、文本框等
Top	设置控件上边界与包含它的容器上边界的距离	表单、标签、文本框、列表框、组合框等
Lift	设置控件左边界与包含它的容器左边界的距离	
Hight	设置对象的高度	
Width	设置对象的宽度	
AlwaysOnTop	设置表单是否总位于其他打开窗口之上	表单
AutoCenter	设置表单运行时是否自动位于 VFP 主窗口中央	
ScaleMode	设置坐标单位	
Closeable	设置标题栏中关闭按钮是否有效	
Controlbox	设置是否取消标题栏中所有的按钮	表单、工具栏

续表

MaxButton	设置是否具有最大化按钮	表单
MinButton	设置是否具有最小化按钮	
Movable	设置运行时表单是否可以任意移动	
WindowState	设置运行时是以最大化、最小化还是正常状态运行	
Picture	指定显示的图形文件名	表单、容器
AutoCloseTables	表单释放时是否关闭表或视图,默认为.T.	数据环境
AutoOpenTables	表单打开时是否打开表或视图,默认为.T.	

关于上述属性的查看和修改,大家可以打开表单名为“职工情况”的表单,来查看熟悉以上相关的属性,也可以试着修改一下其中的一些属性。

例 5.5 要将图 5.20 所示的表单的标题“职工基本情况”改为“职工情况”,将显示“职工基本情况”的标签的前景色和背景色进行修改。

一、交互式操作方式

1. 在“表单设计器”中打开“职工情况”。

2. 设置表单的 Caption 属性。

(1) 单击表单,在属性窗口中选定“Caption”属性,修改属性设置框的内容为:职工情况;

(2) 完成属性修改后,单击属性窗口中的“√”或按回车键或直接选择其他属性,就完成了对表单“Caption”属性的设置;

(3) 若要恢复原属性,则在没有执行第(2)步前,单击属性窗口中的“×”或按 Esc 键,取消刚才的修改。

3. 设置标签名为 Label1 的前景色。

(1) 在表单设计器中选择显示内容为“职工基本情况”的标签,由属性窗口的对象组合框中可以发现,该标签的名称为 Label1,即该标签的 Name 属性为 Label1;

(2) 在属性列表框中选择 ForeColor 属性;

(3) 直接在属性列表框中双击该属性或单击属性设置框旁的按钮,弹出“颜色”对话框;

(4) 在该“颜色”对话框,中选定所需要的颜色,如红色;完成该属性设置;

(5) 也可直接在属性设置框中输入颜色,如 255,0,0,这三个数字分别表示了红绿蓝三种颜色的混合比。

4. 设置标签名为 Label1 的背景色。

(1) 选择对象 Label1 的 BackColor 属性,将其颜色设置为蓝色:0,0,255,其设置过程同 ForeColor 属性的设置;

(2) 由于用向导创建该表单时,对象 Label1 的 BackStyle 属性为 0,即透明,所以背景色不显示;

(3) 将对象 Label1 的 BackStyle 属性为 1,即不透明,其背景色出现。

二、程序命令方式

ThisForm.Caption="职工情况"

&& 设置表单的 Caption 属性值

ThisForm.Label1.ForeColor=RGB(255,0,0) && 设置对象 Label1 的 ForeColor 属性
ThisForm.Label1.BackColor=RGB(0,0,255) && 设置对象 Label1 的 BackColor 属性
ThisForm.Label1.BackStyle=1 && 设置对象 Label1 的 BackStyle 属性

通过以上四行命令就可以完成交互式方式操作的内容,但这四行命令不能直接在命令窗口中执行,而应该将其放一个事件中去执行,如将其作为表单的单击事件执行的代码,为了让程序执行过程有恢复和设置功能,我们在表单的双击事件中设置执行代码,还恢复原来的设置。事件的设置过程如下:

首先打开代码窗口,然后选择对象,如表单 Form1,在选择好对象的基础上选择对象的过程(事件),如 Click 或 DblClick,并输入相应的代码,如图 5.23 所示。

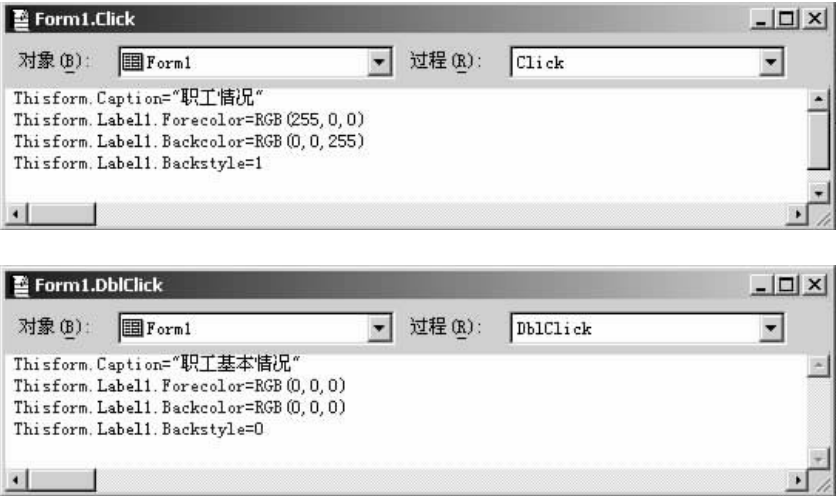


图 5.23 Click 和 DblClick 事件代码

在上例的程序命令方式中,涉及到许多新的内容,如对象的引用、Rgb()函数、事件代码的书写等等,我们将在以后的内容中逐步介绍。

5.1.2.6.3 事件

事件(Event)是泛指由用户或系统触发的一个特定的操作。一个事件对应着一个程序,这个程序称为这个事件的过程。一个对象可以有多个事件,每个事件都是由系统预先定义的。在上例中为了说明属性的设置,介绍了表单的两个事件,如单击事件 Click 和双击事件 DblClick,并在这两个事件中设置了相应的事件代码,在运行这个表单时,单击(或双击)表单的空白处就会激发这些事件,并运行该事件对应的代码过程,即事件过程,如改变表单的标题名称、改变标签的前景色和背景色等等。在属性窗口的“方法程序”列表中,凡名称中含有 Event 的项目,表示其为某一对象的事件,不含 Event 的项目,表示其为某一对象的方法程序。当选定某一事件名称或方法名称时,在属性窗口最下面的注释栏中,会出现对该事件或方法程序的说明。常用的一些事件见表 5.3。

表 5.3 VFP 常用事件表

事件	触发时机	事件	触发时机
Load	创建对象前	MouseUp	释放鼠标时
Init	创建对象时	MouseDown	按下鼠标时
Activate	对象被激活时	MouseMove	移动鼠标指针到一个对象上时
GotFocus	对象得到焦点时	KeyPress	按下并释放某个键时
Click	单击鼠标左键时	Valid	对象失去焦点前
DblClick	双击鼠标左键时	LostFocus	对象失去焦点时
InterActiveChange	更改控件的值时	Unload	释放对象时

1. 事件驱动工作方式。

事件被触发时,系统会立即执行与该事件对应的事件过程,在执行完事件过程后,系统又处于一种等待某个事件发生的状态,这种程序执行方式我们称之为应用程序的事件驱动工作方式。使用面向对象方式设计的应用程序,基本上都是利用事件驱动工作方式运行的。

事件的触发形式有三种:

- (1) 由用户触发:如前例介绍的单击和双击事件等;
- (2) 由系统触发:如计时器事件;
- (3) 由代码触发:用代码来调用事件过程。

2. 为事件(或方法程序)编写代码。

代码窗口的打开方式五种:

- (1) 双击对象;
- (2) 选择对象后,单击快捷菜单的“代码”命令;
- (3) 单击系统菜单“显示/代码命令”;
- (4) 单击表单设计器工具栏的“代码窗口”按钮;
- (5) 双击属性窗口中的事件(或方法程序)。

代码窗口打开后的形式如图 5.24 所示,该窗口中包含一个对象组合框、一个过程组合框和一个代码编辑框。利用对象组合框可以重新确定对象;利用过程组合框可以确定一个需要的事件(或方法程序);利用代码编辑框可以书写或编辑确定对象事件(或方法程序)的代码。

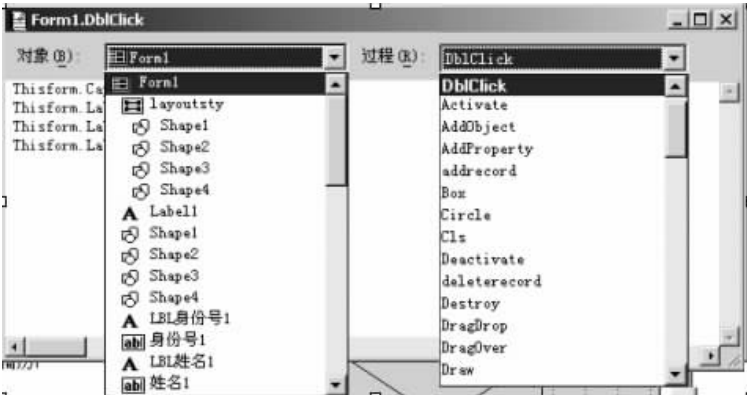


图 5.24 代码窗口

5.1.2.6.4 方法程序

方法程序是 VFP 为对象事先在系统内确定的通用过程,能让对象执行一个相应的操作,其代码用户不可见。方法程序只能在程序运行时被调用,其调用格式如下:

Parent. Object. Method

功能:调用方法程序。

方法程序的具体调用可参阅本章第二节的 5.2.2.2 方法程序的调用。

常用的方法程序见表 5.4。

表 5.4 常用方法程序

方法程序	用途
AddColumn	在表格控件中添加一个列对象
AddObject	在表单对象中添加一个对象
Box	在表单对象中画一个矩形
Circle	在表单对象中画一个圆或椭圆
Clear	清除控件中的内容
Cls	清除表单上的图形和文本
Draw	重画表单对象
Hide	隐藏表单、表单集或工具
Line	在表单对象中画一条线
Move	移动对象
Point	返回表单上指定点的颜色值
Pset	在表单上描点
Refresh	重绘表单或控件,并刷新所有数据
Release	从内存释放表单或表单集
SetFocus	使指定控件获得焦点
Show	显示表单并且决定表单是模态或非模态

例 5.6 在一个表单中画一个同心圆和同心矩形,可以擦去,并重新画,双击表单进释放表单。

设计步骤:

- 1. 创建一个名为 Method 的表单,修改其 Caption 属性值为 Method;
- 2. 各事件代码如下:

(1) Form1 的 Load 事件代码:

ThisForm. Scalemode=3

&& 表单坐标以像素为单位

ThisForm. Drawwidth=2

&& 设置线条的宽度

ThisForm. Forecolor=Rgb(255,0,255)

&& 设置前景色

(2) Form1 的 click 事件代码:

X=ThisForm. Width/2

&& 获取表单水平位置的中心点

Y=ThisForm. Height/2

&& 获取表单垂直位置的中心点

Min=Iif(X<Y,X,Y)

&& 取表单宽和高的最小值

For R=0 To Min/2 Step 5

ThisForm.Circle(R,X,Y)

ThisForm.Box(R,R,X*2-R,Y*2-R)

Next

&& 在循环过程中画同心圆和同心矩形

&& 指定圆的半径和圆心坐标

&& 指定矩形两个顶点的坐标

(3) Form1 的 Dblclick 事件代码：

ThisForm.Release

&& 从内存中释放表单

(4) Form1 的 Dblclick 事件代码：

ThisForm.Cls

&& 清除表单上的图形

5.1.2.7 表单的保存和运行

1. 表单的保存。

单击系统菜单“文件/保存”或“文件/另存为”菜单项,对于第一次保存来讲,都会打开“另存为”对话框,如图 5.25 所示。

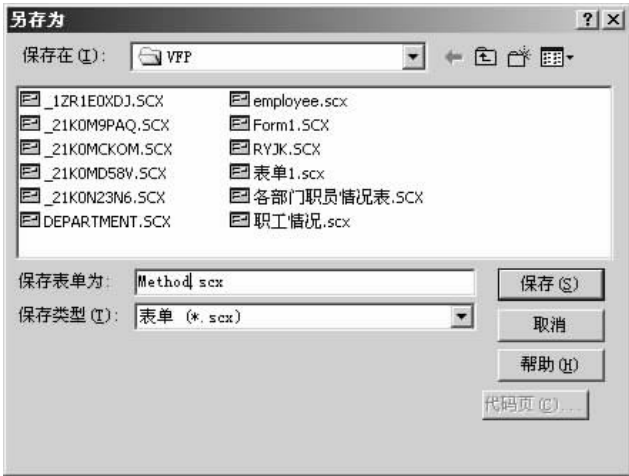


图 5.25 “属性”窗口

2. 表单的运行方式。


(1)在表单设计器中,单击常用工具栏上的按钮,如图 5.26 所示。或者单击系统菜单“表单/执行表单”菜单项；



图 5.26 运行表单

(2)在项目管理器中,选择表单后,单击“运行”按钮;如图 5.27 所示；

(3)在命令窗口或程序中运行表单。

命令格式：

Do Form <表单文件名>

例如:在完成上述操作后,在命令窗口中输入:Do Form Method 命令,则执行该表

单。在表单的运行过程中,大家可以单鼠标左键,右键以及双击鼠标左键来观察表单的变化。



图 5.27 运行表单

5.2 面向对象的程序设计方法

5.2.1 基本概念

在面向对象的程序设计中,最重要的概念是类和对象,它们是关系密切但又完全不同的两个概念。

5.2.1.1 类

类是一组具有公共的方法和一般特性的对象的描述,他是对象的原型描述,它是定义对象的特征、外观和行为的模板。类是对一类对象的归纳和抽象。

例如,在 VFP 中系统定义了一个表单类,它有许多的属性(如长度、宽度、背景色、表单标题等),还定义了事件和操作方法,但它是一个抽象的框架,属性和方法都没有具体的值。

类的两个主要特性:

1. 封装性:指将类中的属性、方法程序和事件的实际实现过程隐藏起来,通过一些引用可以对类的属性进行修改、对方法程序进行直接调用、设置相应的事件过程。在编程过程中,对类的属性、方法程序和事件的操作,只能通过类的实例的操作来实现,即对该类创建的对象的操作来实现。在对对象的操作过程中,用户不用具体去了解这些对象的属性、方法和事件的处理过程,而仅仅只通过设置该对象的属性、调用该对象的方法程序和设置事件的代码,来设置和规定对象的形状和行为等。例如在例 5.6 中涉及的两个方法 Circle 方法和 Box 方法,我们只需要将参数如圆的半径 R 及圆心坐标 (X, Y) 传递给 Circle 方法的,将矩形的两个坐标顶点的参数传递给 Box 方法,而不需要去关心 Circle 方法和 Box 方法是如何去画圆和画矩形的。又如若要将表单的背景色设置为红色,我们只需要将该对象的 BackColor 设置为红色,而不用关心程序是如何将背景色改变为红色的过程。

2. 继承性:通过现有的类可以派生出新的类。派生类继承并具有父类的所有特性,包括父类的所有属性、方法程序和事件。例如,我们以基类 CommandButton 来创建一个新的命令按钮子类,并对该子类的属性等进行一些增加或修改,但该子类仍然会继承父类的功能(基本属性、事件和方法)。

5.2.1.2 基类和子类

5.2.1.2.1 基类和子类的概念

类可划分为基类(Base Class)和子类(Subclass),也称为根类和派生类。基类是 VFP 内部定义的类,可作为其他用户自定义类的基础。如 VFP 表单和所有控件就是基类,用户可以在此基础上创建新类,增添自己需要的功能。由用户创建的类称为子类或自定义类。子类是以其他类定义为起点而建立的新类,该新类将继承父类的特性。

5.2.1.2.2 子类的创建步骤

1. 单击系统菜单“文件/新建”命令或单击工具栏上的“新建”命令按钮,弹出“新建”对话框;

2. 在“新建”对话框中,选择“类”后,单击“新建文件”命令按钮,弹出“新建类”对话框,如图 5.28 所示;

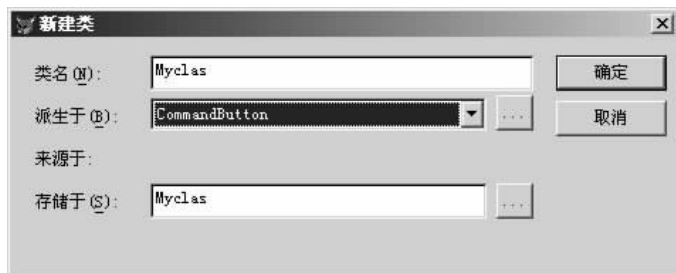


图 5.28 “新建类”对话框

3. 在“新建类”对话框中的“类名”文本框中,输入新建类的类名;
4. 在“派生于”选择列表框中选择父类;
5. 在“存储于”文本框中输入类文件名后,单击“确定”按钮,弹出类设计器窗口;
6. 类设计器窗口的操作与表单设计器的操作类似,用户可在该设计窗口中对类进行修改;
7. 完成后关闭即可保存。

说明:在一个类文件中可以创建多个不同的子类。

5.2.1.2.3 子类的应用

当创建好一个子类后,其应用如同控件的使用方法,但首先要将用户创建的类添加到表单控件工具栏中,其添加方法可参阅本章第一节 5.1.2.3 表单控件工具栏中的查看类。

例 5.7 创建一个新的命令按钮类,该命令按钮上有一个狐狸图标,并在新建的表单中添加该控件。

操作步骤:

1. 创建一个新的命令按钮类。

(1)单击工具栏“新建”按钮,弹出“新建”对话框,如图 5.28 所示;

- (2)在“类名”文本框中输入:Myclas;
- (3)在“派生于”列表框中选择基类:CommandButton;
- (4)在“存储于”文本框中输入:Myclas;
- (5)单击“确定”,弹出“类设计器”窗口;
- (6)修改 Myclas 类的 Picture 属性,将其属性值修改为:D:\Program Files\Microsoft Visual Studio\Vfp98\Fox.bmp,该步骤的实质是指定命令按钮上的图片;
- (7)如果还要修改其他属性可继续,修改完成后,保存退出“类设计器”窗口;

2. 在新建的表单中添加该控件。

- (1)新建一个空白表单;
- (2)单击表单控件工具栏中的“查看类”按钮,并单击弹出菜单中的“添加”命令;
- (3)在打开对话框中,选择刚才保存的类库文件:Myclas.vcx;
- (4)在表单控件工具栏中单击“Myclas”按钮,向表单中添加该控件;

3. 运行表单。

说明:与其他基类控件一样,该命令按钮控件也可通过设置其事件代码等,对该命令按钮进行相应的操作。

5.2.1.3 对象

对象是类的运行实例。例如:表单及表单上的所有控件都是对象,如标签、文本框、命令按钮等等。

从可视化编程的角度来看,对象是一个具有属性(数据)和方法(行为方式)的实体。一个对象建立以后,其操作只能借助与该对象相关的属性、事件和方法程序进行。

任何对象都具有自己的外观和行为。对象的外观由它的各种属性来描述,对象的行为则由它的各种事件和方法程序来表达。不同的对象有不同的属性和行为。

关于对象的属性、方法程序和事件的使用方法可参阅本章第一节 5.1.2.6 属性窗口中的内容。

5.2.1.4 类的层次

VFP 提供的基类可分为两种类型:容器类和控件类。

1. 容器类。

容器类可以包含其他对象,并且允许访问这些对象;容器类控件可以通过手工或编程的方式(Addobject 方法)向容器内部添加控件。表 5.5 列出了 VFP 中的容器类和各容器类所能包含的对象。

表 5.5 容器类及其所包含的对象

容器	包含的对象
表单集	表单、工具栏
表单	任何控件以及页框、容器对象、命令按钮组、选项按钮组、表格等
页框	页面
页面	任何控件以及容器对象、命令按钮组、选项按钮组、表格等
表格	表格列

续表

表格列	标头和除表单、表单集、工具栏、计数器和列对象以外的对象
选项按钮组	选项按钮
命令按钮组	命令按钮
容器	任何控件以及页框、容器对象、命令按钮组、选项按钮组、表格等

2. 控件类。

控件类不能包含其他对象,只能被加入到其他对象中;控件类的封装比容器类更为严密。VFP 中常用的控件类有:复选框(CheckBox)、组合框(ComboBox)、命令按钮(CommandButton)、编辑框(EditBox)、图像(Image)、标签(Labe)、线条(Line)、列表框(ListBox)、选项按钮(OptionButton)、形状(Shape)、微调(Spinner)、文本框(TextBox)、计时器(Timer)、控件(Contro)、列标题(Header)、OE 绑定型控件(OE Bound Contro)、OE 容器控件(OE Container Contro)、自定义类(Custom),其中计时器和自定义类是不可见类。

5.2.2 对象引用

在前面的一些例子中,已经涉及到一些对象的引用问题,如属性的引用,方法程序的调用等,这里我们将系统的介绍一下对象的引用方法。

在面向对象的程序设计中经常需要引用对象,或引用对象的属性、事件或调用方法程序。但在处理一个具体的对象时,往往需要知道该对象相对于容器的层次关系。这种层次关系的引用就好比指出该对象的具体位置。表 5.6 列出了一些在编程中经常用容器对象关键字,这些关键字允许用户方便地引用对象。

表 5.6 容器对象关键字

关键字	引用
Parent	表示当前对象的直接容器(运行时引用)
This	表示当前对象
ThisForm	表示包含该对象的表单
ThisFormSet	表示包含该对象的表单集
ActiveControl	表示当前运行表单中得到焦点的对象(运行时引用)
ActiveForm	表示当前运行的表单(运行时引用)
ActivePage	表示当前表单中的活动页(运行时引用)

5.2.2.1 属性设置

格式一:

Parent. Object. Property=Value

功能:设置属性值。

参数说明:

- (1)Parent:父对象名称,可以是表 6.5 中的任意关键字;
- (2)Object:对象名称,是对象的 Name 属性指定的名称;

(3)Property:该对象的某个属性名称。

例 5.8 在 Form1 表单中通过编程方式设置命令按钮 Command1 的 Caption 属性,可以在表单的 Init 事件中写入代码:

```
ThisForm.Command1.Caption="确定"
```

上面的语句形式可以出现在对象 Command1 所在表单上的任何控件的事件或方法代码中,因为该控件的引用使用的是绝对引用方式,表示:设置当前表单的 Command1 对象的 Caption 属性值为“确定”。

例 5.9 如果要求用户在单击该命令按钮时改变命令按钮的 Caption 属性,可以在对象 Command1 的 Click 事件中编写代码:

```
This.Caption="确定"
```

这里的 This 表示当前对象 Command1,所以该语句形式只能在该对象的事件和方法程序中使用,表示:设置当前对象的 Caption 属性值为“确定”。

例 5.10 如果要求用户在单击命令按钮 Command1 时,改变表单的背景颜色为红色,则可以在 Command1 的 Click 事件中编写代码:

```
This.Parent.BackColor=RGB(255,0,0)
```

这里的 Parent 表示当前对象的父对象,表示设置当前对象 Command1 的父对象(即当前表单)的 BackColor 属性值为 RGB(255,0,0)。

格式二:

```
With Parent.Object
```

```
Property1=Value1
```

```
[...]
```

```
Propertyn=ValueN]
```

```
Endwith
```

功能:一次设置一个对象的多个属性值。

参数说明:

(1)在 With 语句之后指明对象的位置;

(2)在 With 和 EndWith 语句之间使用该对象的属性或方法程序引用时,在属性和方法程序前不能加对象名称等。

例 5.11 在一个表单中,要设置命令按钮 Command1 的多个属性,可以使用如下代码:

```
With ThisForm.Command1
```

```
Forecolor=Rgb(255,0,0)
```

```
FontSize=12
```

&& 设置字号为 12 号字

```
Caption="确定"
```

```
Endwith
```

5.2.2.2 方法程序的调用

对象被建立后,就可以在应用程序的任何一个地方调用此对象的方法程序。

格式:

```
Parent.Object.Method
```

功能:调用方法程序。

参数说明:

Method:该对象的某个方法名称。

例 5.12 在表单上放置一个命令按钮 Command1 和两个文本框,当用户单击 Command1 时,文本框 Text1 得到焦点,则可以在 Command1 的 Click 事件中编写代码:

```
ThisForm.Text1.SetFocus
```

5.2.2.3 事件过程的调用

在同一对象或不同对象间,如果一个事件要执行的程序代码的中有一部分与另一个事件的程序代码完全相同,我们就可以使用事件过程的调用。

格式:

Parent. Object. Event

功能:调用另一个事件过程。

参数说明:

Event:该对象的某个事件名称。

事件过程调用的举例,可参阅第六章第一节的例 6.6 中的设置 List1 列表框的 DblClick 事件代码和设置 List2 列表框的 DblClick 事件代码。

习 题

1. 操作练习,向表单添加下面控件:

(1) 图像、标签、文本框、编辑框、形状,利用布局工具调整各控件间的布局。

(2) 用生成器锁定控件、按钮锁定控件给表单添加控件。

2. 利用“表单设计器”创建一个标题为“Hello Welcome”的表单,当单击表单时,表单标题变成“欢迎”;当双击表单,表单关闭。

3. 设计一个标题栏为“文本框练习”的表单,有 2 个文本框和一个命令按钮 Command1;在第一个文本框中输入 23,当单击命令按钮时,就会将第一个文本框中的内容送到第二个文本框,同时清除第一个文本框的内容。

4. 类与对象的区别是什么?类有哪些特点?

5. 创建一个标签类,该类创建后,若添加到表单中,则显示当前时间。

6. 以表 Student 为数据源,利用表单向导创建表一个名为 Student 的表单。

7. 以表 Collegedepartment, Student 为数据源,利用表单向导创建一个名为 StudColl 的一对多表单。



第 6 章

表单控件设计

表单中常常会包含许多控件, Visual FoxPro 的表单控件工具栏的控件按其功能可划分为四大类:

1. 输出、输入类: 标签、图像、线条、形状、文本框、编辑框、列表框、组合框、微调控件;
2. 控制类: 命令按钮、命令按钮组、复选框、选项按钮组、计时器;
3. 容器类: 表格、页框、Container 容器;
4. 连接类: ActiveX 控件、ActiveX 绑定控件、超级链接。

为简洁起见, 本书有时将表单窗口简称为表单; 在提到对象时也会省略控件两字, 例如标签控件就简称为标签。

6.1 输出、输入类控件

本节讨论标签、图像、文本框、编辑框、列表框、组合框和微调控件 7 个控件。标签和图像控件为输出类控件, 用于在表单上设置文本和图形。编辑框、列表框、组合框和微调控件为输入类控件, 除列表框只能以选项的方式选用数据以外, 其他的控件都可以用键盘直接输入数据。

6.1.1 标签、图像、文本框、编辑框

6.1.1.1 标签和图像

标签和图像是常用的输出控件, 标签用于文本控件的输出, 图像控件用于图像信息的输出。

例 6.1 设计如图 6.1 所示的应用程序封面表单 Welcome.scx, 在右击表单时退出。

设计步骤:

1. 创建表单 Welcome.scx;
2. 在表单上创建两个标签和一个图像控件, 并通过布局工具栏中的工具将图像控件置后;
3. 设置相应控件的属性:



图 6.1 运行效果图

各属性值的设置,可通过表单的属性窗口分别设置,具体设置值见表 6.1。

表 6.1 表单 Welcome 属性设置

对 象	属 性	属性值	说 明
Form1	AutoCenter	. T.	自动在主窗口内居中
	BackColor	255,255,255	背景色为白色
	TitleBar	0	取消表单标题栏
	Height	242	
	Width	385	
	BorderStyle	1	单线边框
Label1	Caption	员工信息管理系统	封面文字
	AutoSize	. T.	Label1 区域自动适应标题大小
	BackStyle	0	背景透明,不显示 Label1 区域
	BorderStyle	0	无边框
	FontName	隶书	字体
	FontSize	24	文体大小
	ForeColor	0,0,255	字体颜色为红色
	Left	96	
	Top	36	
Label2	Caption	Ver 1.0	版本信息
	AutoSize	. T.	Label1 区域自动适应标题大小
	BackStyle	0	背景透明,不显示 Label1 区域
	BorderStyle	0	无边框
	FontName	Arail Black	字体
	FontSize	18	文体大小
	ForeColor	255,0,0	字体颜色为红色
	Left	258	
	Top	96	
Image1	Picture	c:\windows\web\wallpaper\autumn.jpg	也可选择其他图片
	Stretch	2	变比填充
	Height	217	
	Left	12	
	Top	12	
	Width	168	

4. 表单 Welcome 的 RightClick 事件代码如下：

ThisForm . Release &&. 右击表单时,结束表单运行

6.1.1.1.1 标签

标签是一种在表单上显示文本的输出控件,经常用作提示和说明。标签具有自己的一套属性、方法及事件,能响应绝大多数的鼠标事件。这里仅仅介绍其主要属性：

1. Caption 属性：

用来指定标签显示的文本内容,最多包含 256 个字符；

2. Name 属性：

Name 属性是对象的引用名称，每个对象都有自己的 Name 属性值，在引用对象时，必须使用对象的 Name 属性值。需要注意的是，在同一个容器层次上的两个对象的 Name 属性值不能相同。

3. Alignment 属性：

指定 Caption 属性的文本在标签中显示时的对水平齐方式，见表 6.2。

4. AutoSize 属性：

是否自动调整控件大小以容纳其内容，默认值为 F.。

5. WordWrap 属性：

设置控件是沿纵向还是沿横向扩展，默认值为 F.，沿横向扩展。

6. FontName 属性：

设置用于对象文本显示的字体名，默认值为“宋体”。

7. FontSize 属性：

设置用于对象文本显示的字体大小，默认值为 9。

8. FontBold 属性：

设置用于对象文本显示的字体是否为粗体，默认值为 F.。

9. ForeColor 属性：

设置对象显示内容的前景色。

10. BackColor 属性：

设置对象的背景色。

11. BorderStyle 属性：

设置对象的边框样式，默认值为 0，无边框；设置为 1 时为固定单线。

12. BackStyle 属性：

设置对象背景是否透明，默认值为 1，不透明；设置为 0 时为透明。当该属性设置为透明时，背景色的设置将不可见。

13. Visible 属性：

设置对象是可见还是隐藏，默认值为 T.，可见。

14. Enabled 属性：

设置表单或控件能否响应由用户引发的事件，默认值为 T.；当该属性设置为 F. 时，该对象将不响应由用户引发的事件。

关于标签的 Top, Lift, Width, Height 属性可参见表 5.2 对象的常用属性的介绍，也可以直接在属性窗口中，选择这些属性在属性窗口下方察看注释说明。大家在上机操作时，要养成一个注意阅读注释说明的习惯，这样可以帮助大家理解属性、方法程序和事件的使用方法。

6.1.1.1.2 表单及控件常见的事件、方法

1. Click 事件、Dblick 事件和 RightClick 事件：

表 6.2 Alignment 属性及其设置值

设置值	说 明
0	水平左对齐(默认值)
1	水平右对齐
2	水平居中

表单及本章介绍的控件都具有这三个事件。Click 事件一般发生在鼠标单击控件时，DblClick 事件发生在鼠标双击控件时，RightClick 事件发生在鼠标右击控件时。可以在这些事件中编写代码，完成指定功能。

2. 表单的 Release 方法：

将表单从内存中释放(清除)，实现关闭表单的功能。

6.1.1.1.3 图像

利用图像控件可以在表单上显示图像，图像文件类型可以是. BMP, . ICO, . GIF 和. JPG 等。图像控件的主要属性有：

1. Picture 属性：

设置显示在控件上的图形文件。

2. Stretch 属性：

设置如何对图像进行尺寸调整以放入图像控件，默认值为 0，裁剪；1 为等比填充，2 为变比填充。

3. BorderColor 属性：

设置对象的边框颜色。

6.1.1.1.4 线条和形状

除了图像控件，还可以使用线条控件和形状控件来美化表单。线条控件用于在表单上画各种类型的线条，包括水平线、垂直线和斜线。形状控件用于在表单上画出各种类型的形状，包括矩形、圆角矩形、正方形、圆、椭圆等形状。

1. 线条控件的主要属性：

(1) LineSlant 属性：

设置线条的倾斜方向，默认值为“\”，向右倾斜；“/”为向左倾斜。

(2) BorderWidth 属性：

设置直线的宽度，默认值为 1。

另外通过 Height 和 Width 属性可以控件斜线的倾斜度，Height 值越大或 Width 值越小，斜线的倾斜度越大，当 Width 值为 0 时，为一条垂直直线；Height 值越小或 Width 值越大，斜线的倾斜度越小，当 Height 值为 0 时，为一条水平直线

2. 形状控件的主要属性：

形状对象的形状类型可由 Curvature, Width 和 Height 属性值来指定，见表 6.3。

表 6.3 形状对象的形状设置

Curvature	Width 和 Height 值相等	Width 和 Height 值不等
0	正方形	矩形
1~99	圆角正方形→圆	圆角矩形→椭圆

例 6.2 例 6.1 还可以通过代码的形式实现，但在本例中我们增加了一个功能，即运行表单时，在 Label1 对象的后面显示一个黄色的椭圆形图形，双击标签时图形消失，单击鼠标时重新出现。

设计步骤：

1. 创建表单 Welcome.scx；
2. 在表单上任意位置创建两个标签、一个图像控件和一个形状控件；
3. 在 Form1 的 Init 事件中写入以下代码，设置如表 6.1 的属性，其中新增加的属性将加注释：

```

ThisForm.Autocenter=.T.
ThisForm.BackColor=RGB(255,255,255)
ThisForm.Titlebar=0
ThisForm.Height=242
ThisForm.Width=385
ThisForm.Borderstyle=1
ThisForm.Shape1.Visible=.F.
ThisForm.Label1.Caption="员工信息管理系统"
ThisForm.Label1.Autosize=.T.
ThisForm.Label1.Backstyle=0
ThisForm.Label1.Borderstyle=0
ThisForm.Label1.Fontname="隶书"
ThisForm.Label1.Fontsize=24
ThisForm.Label1.Forecolor=RGB(0,0,255)
ThisForm.Label1.Left=96
ThisForm.Label1.Top=36
ThisForm.Label2.Caption="Ver 1.0"
ThisForm.Label2.Autosize=.T.
ThisForm.Label2.Backstyle=0
ThisForm.Label2.Borderstyle=0
ThisForm.Label2.Fontname="Arial Black"
ThisForm.Label2.Fontsize=18
ThisForm.Label2.Forecolor=RGB(255,0,0)
ThisForm.Label2.Left=258
ThisForm.Label2.Top=96
ThisForm.Image1.Picture="C:\Windows\Web\Wallpaper\Autumn.Jpg"
ThisForm.Image1.Stretch=2
ThisForm.Image1.Height=217
ThisForm.Image1.Left=12
ThisForm.Image1.Top=12
ThisForm.Image1.Width=168

```



图 6.2 运行效果图

&& 设置图形为不可见

4. 在 Form1 的 RightClick 事件中写入以下代码：

```
ThisForm.Release
```

5. 在 Form1 的 Active 事件中写入以下代码：

```
ThisForm.Shape1.Curvature=99                                && 设置图形为椭圆形
ThisForm.Shape1.Backcolor=Rgb(255,255,0)                  && 设置图形的背景色为黄色
ThisForm.Shape1.Width=ThisForm.Label1.Width*1.2           && 设置图形的宽度
ThisForm.Shape1.Left=ThisForm.Label1.Left+(ThisForm.Label1.Width-;
    ThisForm.Shape1.Width)/2                                && 设置图形的左边界位置
ThisForm.Shape1.Height=ThisForm.Label1.Height*1.8          && 设置图形的高度
ThisForm.Shape1.Top=ThisForm.Label1.Top-(ThisForm.Shape1.Height-;
    ThisForm.Label1.Height)/2                                && 设置图形的上边界位置
ThisForm.Shape1.Visible=.T.                                && 设置图形为可见
```

6. 在 Label1 的 Click 事件中写入以下代码：

```
ThisForm.Shape1.Visible=.T.                                && 设置图形为可见
```

7. 在 Label1 的 DblClick 事件中写入以下代码：

```
ThisForm.Shape1.Visible=.F.                                && 设置图形为不可见
```

6.1.1.2 文本框和编辑框

文本框和编辑框控件是常用的输入和输出控件,文本框和编辑框的用法基本相似,但文本框可用于如字符型、数值型、日期型和逻辑型等数据的编辑输入和输出,而编辑框只能用于文本类数据的编辑。

例 6.3 新建一个名为 Text1 的表单,在表单上添加一个文本框、一个编辑框和四个标签,其设计界面如图 6.3 所示,运行表单时要求有如下功能：

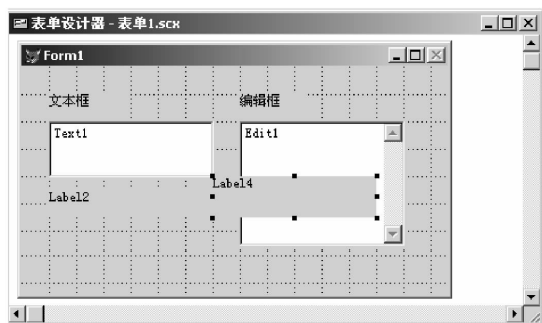


图 6.3 设计界面

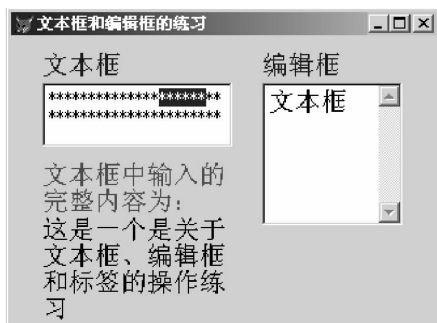


图 6.4 运行界面

1. 完成向 Text1 文本框输入数据,并选中输入的部分文本后,单击表单空白处时,在 Label4 标签中显示文本框中的全部内容,在 Edit1 编辑框中显示文本框中选中的文本,并让 Edit1 编辑框获取焦点,如图 6.4 所示；

2. 双击表单空白处时,Text1 文本框重新获得焦点；

3. 在向文本框中输入数据时,该文本框中只能显示 * 号；

4. 请先完成前两个功能,并运行表单,然后,再修改表单,实现第三项功能。

设计步骤：

- 1. 创建如图 6.3 所示的表单,并添加相应的控件；
- 2. 设置已添加控件的属性,相应控件的属性设置见表 6.4 所示；

表 6.4 控件的属性设置

对 象	属 性	属性值	说 明
Label1	Caption	文本框	对 Text1 文本框的说明
Label3	Caption	编辑框	对 Edit1 编辑框的说明

3. 设置 Form1 的 Init 事件代码：

```
This. Caption="文本框和编辑框的练习"           && 设置表单标题栏内容
This. AutoCenter=. T.
* 设置 Label1 标签的字体大小、前景色等
ThisForm. Label1. Fontsize=15
ThisForm. Label1. Autosize=. T.
ThisForm. Label1. Forecolor=Rgb(0,0,255)
* 设置 Label2 标签的显示内容、字体大小、前景色、显示排列方式及不可见等
ThisForm. Label2. Visible=. F.
ThisForm. Label2. Caption="文本框中输入的完整内容为："
ThisForm. Label2. Autosize=. T.
ThisForm. Label2. Fontsize=15
ThisForm. Label2. Forecolor=Rgb(255,0,0)
ThisForm. Label2. Wordwrap=. T.
* 设置 Label3 标签的字体大小、前景色等
ThisForm. Label3. Fontsize=15
ThisForm. Label3. Autosize=. T.
ThisForm. Label3. Forecolor=Rgb(0,0,255)
* 设置 Label3 标签的字体大小、前景色、显示排列方式及不可见等
ThisForm. Label4. Visible=. F.
ThisForm. Label4. Fontsize=15
ThisForm. Label4. Autosize=. T.
ThisForm. Label4. Wordwrap=. T.
ThisForm. Edit1. Fontsize=15           && 设置 Edit1 标签的字体大小
4. 设置 Form1 的 Click 事件代码：
* 设置 Edit1 编辑框的内容为 Text1 文本框中选定文本,并让 Edit1 编辑框获得焦点
ThisForm. Edit1. Value=This. Text1. Seltext
ThisForm. Edit1. Setfocus
* 设置 Label4 标签的 Caption 属性值为 Text1 文本框的全部内容
ThisForm. Label4. Caption=ThisForm. Text1. Text
```

* 当 Text1 文本框为空时, Label2、Label4 标签为不可见, 否则为可见

```
If Len(ThisForm.Label4.Caption) # 0
```

```
    ThisForm.Label2.Visible = .T.
```

```
    ThisForm.Label4.Visible = .T.
```

```
Else
```

```
    ThisForm.Label2.Visible = .F.
```

```
    ThisForm.Label4.Visible = .F.
```

```
Endif
```

* 设置 Label4 标签与 Label2 标签的相对位置

```
ThisForm.Label4.Left = ThisForm.Label2.Left
```

```
ThisForm.Label4.Top = ThisForm.Label2.Top + ThisForm.Label2.Height
```

5. 设置 Form1 的 DblClick 事件代码:

```
This.Text1.Setfocus
```

&& 设置 Text1 文本框获得焦点

6. 运行表单: 运行表单时, 在弹出的保存对话框中保存表单名为 Text1。当表单运行时, 在文本框中输入文本, 并选中部分文本后单击表单空白处, 观察表单的变化, 双击表单空白处, 比较文本框的选中文本与编辑框的内容;

7. 在表单设计器的属性窗口中, 设置 Text1 的 Passwordchar 属性为“*”, 或在 Form1 的 Init 事件代码中加入以下一行代码: This.Text1.Passwordchar = “*”

8. 再次运行表单。

通过例 6.2 和例 6.3 我们可以看出, 控件的属性也可以通过代码的形式进行设置。

6.1.1.2.1 文本框

文本框是一种常用控件, 供用户输入或编辑数据。文本框可以用来编辑多种类型的数据, 如字符型、数值型、日期型和逻辑型等。一般只包含一行数据, 即按回车键表示输入结束。标准的 Visual Foxpro 的编辑功能, 如剪切、复制和粘贴, 在文本框中都可以使用。

1. 文本框的主要属性。

在前面介绍控件中也如 Alignment, BackColor, BackStyle, FontName, FontSize, ForeColor, Height, Left, Name, Top, Visible, Width 等属性, 文本框和编辑框关于这类属性的用法与前面控件的用法类似, 所在就不再在这里作一一介绍。这里主要介绍前面没有涉及到的属性:

(1) ControlSource 属性:

指定与文本框建立联系的数据源, 即将文本框与某个字段或内存变量联系起来, 实现数据绑定。运行时, 文本框先显示该字段或变量的内容, 用户对文本框的编辑结果, 最终保存到该字段或变量中。该属性在设计和运行时可用。

(2) Value 属性:

设置或返回文本框的当前内容。默认值为空。如果指定了 ControlSource 属性, 则 Value 属性值与 ControlSource 属性指定的字段或变量具有相同的数据及类型。

(3) PasswordChar 属性:

指定文本框内是显示用户输入的字符还是显示占位符; 该属性设置为某一个字符(通

常为 *)后,不会显示用户输入的实际内容,只显示占位符,但不影响 Value 属性值,Value 属性值为用户输入的实际内容。

(4) InputMask 属性:

输入掩码属性,可限制用户输入数据和显示数据的格式。该属性值通常是一个字符串,输入掩码的字符的用法请参阅第三章第六节的 3. 6. 2. 2. 2 输入掩码。

(5) ReadOnly 属性:

只读属性,指定用户能否编辑控件的值,或指定与 Cursor 对象相关联的表或视图是否允许更新,属性值设置情况如表 6. 7 所示。

(6) Enable 属性:

指定文本框是否响应由用户引发的事件,即是否可以编辑文本框的数据。默认值为 . T. ,能编辑。

(7) HideSelection 属性:

指定当文本框失去焦点时,文本框中选定的文本是否仍显示为选定状态。其属性值设置情况如表 6. 5 所示。

表 6. 5 HideSelection 属性的设置值

设置值	说 明
. T. (默认值)	失去焦点时,编辑框选定文本不显示为选定状态;当编辑框再次获得焦点时,选定文本重新显示为选定状态。
. F.	失去焦点时,编辑框中选定文本仍显示为选定状态。

(8) SelStart 属性:

返回用户在文本输入区中所选定文本的起始点位置,或指出插入点位置。取值范围为 0 至编辑区中的字符总数。该属性设计时不可用,运行时可读写。

(9) SelLength 属性:

返回用户在文本输入区所选定字符的数目,或指定要选定的字符数目。取值范围为 0 至编辑区中的字符总数。

(10) SelText 属性:

返回用户在文本输入区内选定的文本,若没选定任何文本,返回空串。

SelStart 属性、SelLength 属性和 SelText 属性常常配合使用。

2. 焦点(Focus)。

一个表单中往往包含了很多对象,但在某一时刻只允许一个选定对象被操作。对象被选定,我们称它获得了焦点。对象可以通过两种情况获得焦点:一是通过用户操作获得,如单击对象或使用 Tab 键切换到该对象上;二是在代码中使用 SetFocus 方法。与焦点相关的还有两个事件:GotFocus 事件和 LostFocus 事件。

(1) SetFocus 方法:

让控件获得焦点,使其成为活动对象。如让表单中的文本框 Text1 获得焦点可用如下代码:ThisForm. Text1. SetFocus 。

注意:如果一个控件的 Enabled 属性或 Visible 属性值为 . F. 时,将不能获得焦点。

(2) GotFocus 事件：
当对象接收到焦点时发生本事件。

(3) LostFocus 事件：
当对象失去焦点时发生本事件。

例 6.4 设计如图 6.5 所示的员工基本信息表单 Employee。运行表单时,显示 Employee 表中第一条记录的身份号、姓名、性别、年龄和简历内容,右击表单退出。



图 6.5 运行效果图

设计步骤：

- 1. 创建表单 Employee. scx, 设置其 Caption 属性值为:Employee;
- 2. 打开表单数据环境设计器,添加表 Employee;
- 3. 在表单上创建五个标签,分别设置其 Caption 属性为“身份号”、“姓名”、“性别”、“年龄”和“简历”,根据需要设置标签控件其他属性;
- 4. 在表单上创建四个文本框和一个编辑框,分别设置属性,见表 6.6,这些属性的设置既可以直接在属性窗口中设置,也可以在 Form1 的 Init 事件中设置,设置的代码形式如:ThisForm. Text1. ControlSource="Employee. 身份号";

表 6.6 控件主要属性设置

对 象	属 性	属 性 值
Text1	ControlSource	Employee. 身份号(字符型)
Text2	ControlSource	Employee. 姓名(字符型)
Text3	ControlSource	Employee. 性别(字符型)
Text4	ControlSource	Employee. 年龄(数值型)
Edit1	ControlSource	Employee. 简历(备注型)

5. 表单 Employee 的 RightClick 事件代码如下：

ThisForm. Release

前例要求大家主要掌握的是文本框和编辑框的数据绑定问题,其实在表单中将文本框或编辑框与表中字段绑定的一个最简单的方法就是将创建好的数据环境中的表的相关字段直接拖放到表中的相应位置即可。每当完成一个字段的拖放,新产生的文本框就会自动与拖放的字段建立联系,即实现了文本框或编辑框与字段的数据绑定。

6.1.1.2.2 编辑框

编辑框也是一种供用户输入、编辑数据的常用控件。与文本框相比较,编辑框只能处理字符型数据,但可包含多段文本,能实现自动换行,有自己的垂直滚动条。编辑框的许多属性与文本框相似,这里主要介绍与文本框不同的 ScrollBars 属性。

ScrollBars 属性：

指定编辑框具有滚动条的类型,默认值为 2,表示编辑框包含垂直滚动条。取 0 时表示编辑框无滚动条。

例 6.5 设计如图 6.6 所示的表单 Form3, 表单中包含一个编辑框 Edit1、两个文本框 Text1 和四个标签。运行表单时, 要求有以下功能:

1. 当单击 Label3 标签(查找)时, 让 Edit1 编辑框中与 Text1 文本框匹配的字符串处于选中状态;

2. 单击 Label4 标签(替换)时, 用 Text1 文本框中被选择的内容去替换 Edit1 编辑框中被选择的内容。

设计步骤:

1. 创建表单, 并将表单的 Caption 属性设置为: 查找与替换;

2. 在表单上创建控件, 并设置相关属性, 设置效果如图 6.6 所示;

3. “查找”标签(Label3)的 Click 事件代码如下:

* 查找 Edit1 编辑框中与 Text1 文本框的内容相同的字符串,

* 当相同时, 让编辑框中的内容以选中状态显示

```
If At(Alltrim(ThisForm.Text1.Value), ThisForm.Edit1.Value) # 0;
```

```
    ThisForm.Edit1.Selstart = At(Alltrim(ThisForm.Text1.Value);
```

```
    ThisForm.Edit1.Value) - 1
```

```
    ThisForm.Edit1.Sellength = Len(Alltrim(ThisForm.Text1.Value))
```

```
    ThisForm.Edit1.Hideselection = .F.
```

```
Endif
```

4. “替换”标签(Label4)的 Click 事件代码如下:

* 将 Edit1 编辑框的被选中文本, 用 Text2 文本框内容替换,

* 并让替换后的内容也处于选中状态。

```
ThisForm.Edit1.Value = Left(ThisForm.Edit1.Value, ThisForm.Edit1.Selstart) +;
```

```
ThisForm.Text2.Value + Right(ThisForm.Edit1.Value, Len(ThisForm.Edit1.Value);
```

```
    - ThisForm.Edit1.Selstart - ThisForm.Edit1.Sellength)
```

```
ThisForm.Edit1.Sellength = Len(Alltrim(ThisForm.Text2.Value))
```

在上例中, 为了突出重点, 省略了许多常用属性的设置, 如果需要对表单进行更细致的设计, 大家可以参照前面学习过的一些属性, 对表单及其控件进行色彩及字体等等设置。

6.1.1.2.3 文本框与编辑框生成器的使用

对于文本框和编辑框, 除了用上述方式进行设计外, 我们可以利用其生成器进行快速设置。生成器的启动方式有两种:

1. 选中控件, 单击鼠标右键, 在弹出的快捷菜单中选择“生成器……”菜单项;

2. 在创建控件之前, 单击表单控件工具栏的“生成器锁定”按钮, 让这个按钮处于凹陷状态, 然后选择表单控件工具栏的“文本框”(或“编辑框”)按钮, 创建文本框(或编辑框)。



图 6.6 运行效果图



图 6.7(a) 文本框生成器的“格式”选项卡

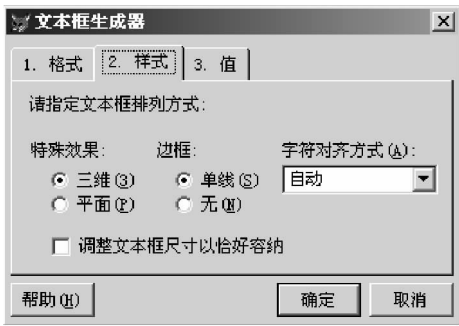


图 6.7(b) 文本框生成器的“样式”选项卡

文本框生成器包含 3 个选项卡,如图 6.7 所示。在格式选项卡中对文本框的数据类型和格式进行设置;在样式选项卡中对文本框的排列方式进行设置;在值选项卡中,对文本框进行数据绑定。在这三个选项卡中可选择设置的内容都可以直接使用属性设置来实现,其中一个最好最直观的学习方式就是,设置生成器中的某一选项,并关闭生成器后,再去观察属性窗口中该控件有哪些相应的属性值发生了什么样的变化?通常情况下,当对象的属性值为非默认值时,该属性值会以加粗的形式显示,所以修改过属性值的属性项在属性窗口中可以非常容易的查找。

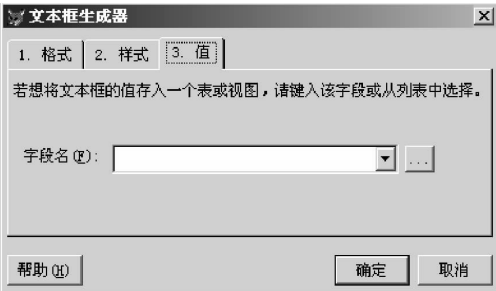


图 6.7(c) 文本框生成器的“值”选项卡

下面我们介绍三个选项卡中各选项所对应的属性项,见表 6.7 所示,其中属性值中的第一项为默认值。

表 6.7 文本框选项卡中各选项与对应属性的关系

选项名称	对应属性	属性值	意 义
数据类型	Value	(空)/{/./F./0	字符型/日期型/逻辑型/数值型
运行时启用	Enabled	. T. /. F.	响应或不响应由用户引发的事件
使用只读	ReadOnly	. F. /. T.	是否是只读状态
隐藏选定内容	HideSelection	. F. /. T.	失去焦点时,选定文本是否显示为选定状态,默认不显示
仅字符表中的字符	Format	A	指定输入和输出格式,取“A”值时,表示仅仅只允许输入字母
进入时选定		K	取“K”值表示,进入控件时文本内容处于选中状态
使用当前的 Set Date		D	默认为美国日期格式,取“D”值表示,表示日期格式为当前系统日期格式
英国日期		E	取“E”值表示,表示日期格式为英国日期格式
显示前导零		L	取“L”值表示,在数值的前面空白处显示 0

输入掩码	InputMask		参阅第三章第六节 3.6.2.2.2 输入掩码
特殊效果	SpecialEffect	0/1	指定边框特殊效果,0 为三维,1 为平面
边框	BorderStyle	1/0	指定边框样式,1 为固定单线,0 为无
字符对齐方式	Alignment	3/0/1/2	指定文本对齐方式,自动/左/右/中间
调整文本框尺寸以恰好容纳			自动调整文本框的大小,以恰好容纳数据,数据长度是其输入掩码的长度或 ControlSource 指定的字段长度
字段名	ControlSource		指定与文本框绑定的数据源即表名与字段名

编辑框生成器与文本框生成器一样也有三个选项卡,如图 6.8 其中在格式选项卡中,可以设置编辑框的格式属性,除第一个格式选项卡外,另外两个选项卡的形式与文本框一样,这里只介绍格式选项卡中各选项与对应属性,见表 6.8 所示,但与文本框中重复的选项请参阅前表。

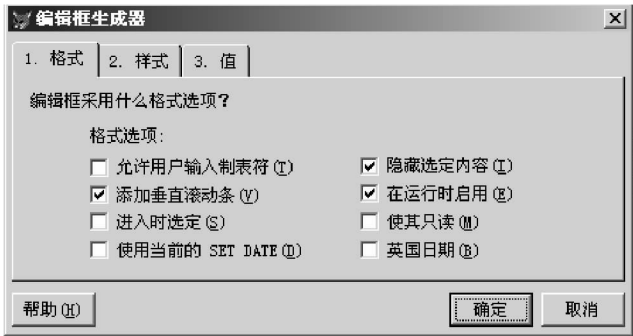


图 6.8 编辑框生成器

表 6.8 编辑框格式选项卡中各选项与对应属性的关系

选项名称	对应属性	属性值	意义
允许用户输入制表符	AllowTabs	. F. /. T.	指定编辑框中能否使用 Tab 键
添加垂直滚动条	ScrollBars	2/0	指定是否有垂直滚动条,2 为有,0 为无

控件生成器其实是一个小型向导,但生成器仅能设置常用属性,不能包括所有属性;另外,并非所有控件都有生成器。控件生成器的使用,请读者通过上机练习掌握。

6.1.2 列表框、组合框、微调控件

在处理数据的输入输出时,常会出现提供一组条目(数据项)以供用户从中选择一个或多个条目的情况,此时,我们可以考虑用列表框或组合框来实现这个任务。

6.1.2.1 列表框

例 6.6 设计如图 6.9 所示的表单,表单中包含两个列表框 List1、List2 和四个标签。表单运行时,要求有如下功能:

1. 在列表框 List1 中显示表 Employee 中所有字段名称,选中某字段名称后,双击字

段名称或单击 Label3(选择),将选中的字段从 List1 中移出,添加到 List2 中;

2. 在列表框 List2 中选择某字段后,双击字段名称或单击 Label4(移去),将选中的字段从 List2 中移回到 List2 中的原位置。

设计步骤:

1. 创建表单,打开表单数据环境设计器,添加表 Employee;

2. 在表单上只添加创建相关控件,不设置控件的属性;

3. 设置 Form1 的 Init 事件代码:

This. Autocenter=. T.

* 设置 Label1 标签的字体、前景色和显示内容

This. Label1. Fontsize=15

This. Label1. Fontbold=. T.

This. Label1. Alignment=2

This. Label1. Forecolor=Rgb(0,0,255)

This. Label1. Caption="可用字段"

* 设置 Label2 标签的字体、前景色和显示内容

This. Label2. Fontsize=15

This. Label2. Fontbold=. T.

This. Label2. Alignment=2

This. Label2. Forecolor=Rgb(0,0,255)

This. Label2. Caption="选择字段"

* 设置 Label3 标签的字体、前景色和显示内容

This. Label3. Fontsize=15

This. Label3. Fontbold=. T.

This. Label3. Alignment=2

This. Label3. Forecolor=Rgb(0,0,255)

This. Label3. Caption="选择"

* 设置 Label4 标签的字体、前景色和显示内容

This. Label4. Fontsize=15

This. Label4. Fontbold=. T.

This. Label4. Alignment=2

This. Label4. Forecolor=Rgb(0,0,255)

This. Label4. Caption="移去"

This. List1. Rowsource="Employee"

This. . Rowsourcetype=8

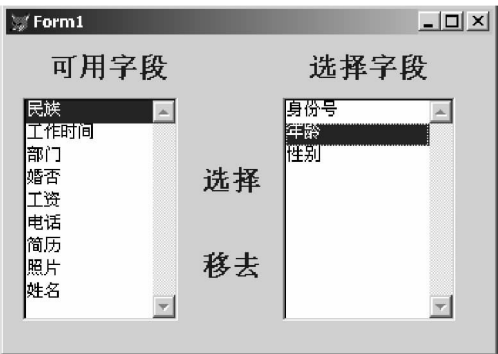


图 6.9 运行效果图

&& 设置 List1 列表框数据源

&& 设置 List1 列表框数据源的类型为结构

4. 设置 Label3 标签(选择)的 Click 事件代码如下:

* 依次判断 List1 中被选条目位置,并将选中的条目添加到 List2 中

```
For I=1 To ThisForm.List1.Listcount
```

* 当 List1 中第 I 个条目被选中时为 .T.,I 即为条目在 List1 中的顺序位置

```
If ThisForm.List1.Selected(I)
```

* 向 List2 添加 List1 中被选中的条目

```
ThisForm.List2.Additem(ThisForm.List1.List(I))
```

* 使 List2 中刚添加的条目为选中状态

```
ThisForm.List2.Listindex=ThisForm.List2.Listcount
```

* 移去 List1 中被选中条目

```
ThisForm.List1.Removeitem(I)
```

```
Endif
```

```
Endfor
```

5. 设置 List1 列表框的 DblClick 事件代码:

```
ThisForm.Label3.Click
```

&& 调用 Label3 标签(选择)的 Click 事件

说明:由于双击 List1 列表框与单击 Label3 标签(选择)的功能相同,因此,List1 列表框的 DblClick 事件代码与 Label3 标签(选择)的 Click 事件代码内容可以完全相同。所以在 List1 列表框的 DblClick 事件代码中,利用事件的调用来执行 Label3 标签(选择)的 Click 事件。

6. 设置 Label4 标签(移去)的 Click 事件代码如下:

* 将 List2 中所有条目依次与 List2 中被选中条目比较,确定被选条目位置

```
For I=1 To ThisForm.List2.Listcount
```

```
If ThisForm.List2.Selected(I)
```

&& 当 List2 中第 I 个条目被选中时为 .T.

* 判断被选中的 List2 中第 I 个条目所对应的字段在数据表中字段的顺序位置 J

```
For J =1 To Fcount()
```

```
If Field(J)=ThisForm.List2.List(I)
```

```
Exit
```

```
Endif
```

```
Next
```

* 依次判断 List1 中各个条目所对应的字段正好在数据表中字段的顺序位置 Q,

* 并且只有当 $Q>J$ 时,表示 List2 中第 I 个条目的插入位置应在 List1 中第 P 项之前

```
For P=1 To ThisForm.List1.Listcount
```

```
For Q=1 To Fcount()
```

```
If Field(Q)=ThisForm.List1.List(P) Then
```

```
Exit
```

```
Endif
```

```
Next
```

```
If Q>J Then
```

```
Exit
```

```
        Endif
    Next
    ThisForm.List1.Listindex=P
    * 将 List2 中被选中条目添加 List1 的第 P 个位置
    ThisForm.List1.Additem(ThisForm.List2.List(I), P)
    * 移去 List2 中被选中条目
    ThisForm.List2.Removeitem(I)
Endif
Next
```

7. 设置 List2 列表框的 DblClick 事件代码:

```
ThisForm.Label4.Click
```

 && 调用 Label4 标签(移去)的 Click 事件

6.1.2.1.1 列表框的属性

列表框提供一组条目(数据项),用户可以从其中选取一个或多个条目。

下面介绍该控件的一些常见属性和方法。

1. ColumnCount 属性:

设置列表框或组合框中条目的列数。

2. BoundColumn 属性:

当列表框或组合框中有多列时,指定将哪一列绑定到该控件的 Value 属性上。默认值为 1。

3. ColumnWidths 属性:

当列表框或组合框中有多列时,指定每列的宽度。

例如: ThisForm.List1.ColumnWidths="28,28,28"。

4. ControlSource 属性:

指定与对象建立联系的数据源,即指定将列表框或组合框的值存储的位置。

5. ListCount 属性:

返回列表框中或组合框条目的个数,设计时不可用,运行时只读。

6. ListIndex 属性:

设置或返回列表框或组合框中选定项的序号。

7. Value 属性:

该属性在列表框为只读,在组合框中可输入新值,用于设置或返回列表框或组合框中的当前内容。

8. DisplayValue 属性:

设置指定列表框或组合框中当前条目的第一列内容。

9. List 属性:

List 是一个字符串数组,设计时不可用,运行时可用以读写列表框条目中的数据项。

格式:

List(nRow,[nCol])

功能:返回列表框或组合框第 nRow 行,第 nCol]列的内容

例如,将列表框 List1 中第 2 个条目第 1 列上的数据项赋给变量 X,可用如下语句:
X=ThisForm.List1.List(2,1) 或 X=ThisForm.List1.List(2)

将列表框 List1 中第 1 个条目第 2 列上的数据项设置为“ABC”,可用如下语句:
ThisForm.List1.List(1,2)="ABC"

10. Selected 属性:

格式:

Selected(nIndex)

功能:用于确定列表框或组合框中的某个条目是否被选中。选中时,属性值为. T. ;
没选中时,属性值为. F. 。

例如,下面代码在列表框 List1 中第 5 个条目选中时,用消息框输出相关信息。

```
If ThisForm.List1.Selected(5)
    MessageBox(选中,0+64,提示信息)
Endif
```

11. MultiSelect 属性:

设置列表框中的条目是否允许多重选定。属性值设为 0 或. F. 时,不允许多重选定;
设置为 1 或. T. 时,允许多重选定。

12. RowSourceType 属性和 RowSource 属性:

前者设置列表框中条目的数据源的类型,后者设置列表框中条目的数据源。两个属
性常常一起设置。RowSourceType 属性及设置值说明见表 6. 9。

表 6. 9 RowSourceType 属性及其设置值

设置值	说 明
0	无(默认值)。程序运行时,可以通过 AddItem 或 AddItemlist 方法添加条目;通过 RemoveItem 或 RemoveListItem 移除条目。
1	值。在 RowSource 属性中指定列表框的条目。例如:RowSource ="电视机,电话机,洗衣机"
2	别名。表由数据环境提供,用 RowSource 属性中指定列表框要显示的表的别名,用 ColumnCount 确定列表框中显示的表的字段数。
3	SQL 语句。将 SQL SELECT 语句查询结果作为列表框条目的数据源。例如:RowSource ="Select * From Employee Into Cursor E1"
4	查询。将扩展名为. qpr 的查询文件执行的结果作为列表框条目的数据源。
5	数组。在 RowSource 中设置数组名,将数组的值作为列表框条目的数据源。
6	字段。将表中记录的一个或多个字段值作为列表框条目的数据源。例如:RowSource ="Employee. 身份号,姓名,性别",同时将 ColumnCount 设置为 3,则在列表框中每个条目显示 3 列,将 BoundColumn 的值设置为某一列号时,则列表框将该列的当前值作为列表框的值。
7	文件。在 RowSource 中设置路径和文件类型,可使用通配符。将指定路径下指定类型的文件名作为列表框条目的数据源。
8	结构。在 RowSource 中设置表名,将该表的字段名称作为列表框条目的数据源。若 RowSource 属性值为空,则列表框显示当前表中的字段名称。
9	弹出式菜单,为与以前版本兼容而设置。

6.1.2.1.2 列表框的方法程序和事件：

1. AddItem 方法和 AddListItem 方法：

格式：

AddItem(cItem[,nIndex1][,nIndex1])

或 AddListItem(cItem[,nIndex1][,nIndex2])

功能：当组合框或列表框的 RowSourceType 属性为 0 时，使用这两个方法可用来向列表框中添加一个新条目。

参数说明：

(1) cItem 新添加条目的字符表达式；

(2) nIndex1 指定新添加条目在组合框或列表框中的位置；当 Sorted 属性为 .T. 时，新条目按字母顺序插入，否则添加到列表的末尾；

(3) nIndex2 用来指定放置新条目的列位置，默认值为 1。

2. RemoveItem 方法和 RemoveListItem 方法：

格式：

RemoveItem (nIndex)

或 RemoveListItem (nIndex)

功能：从组合框或列表框中移除第 nIndex 个条目。

3. InterActiveChange 事件：

当使用键盘或鼠标改变控件的值时，发生的事件。该事件的应用对象包括：文本框、编辑框、列表框及以后要学习的微调控件、复选框、选项按钮组、命令按钮组。

6.1.2.1.3 列表框生成器

列表框是一种常用的控件，其属性既可以在属性窗口中直接设置，也可以在通过编写代码来设置这些属性，还可以利用列表框生成器快速设置相关属性。在列表框的快捷菜单中选中“生成器...”，即可打开对话框。该对话框包含以下 4 个选项卡。

我们通过几个例子来说明列表框生成器的用法：

例 6.7 在表单中建立一个列表框和一个文本框，列表框中显示 Employee 表的字段值，如身份号，姓名，工作时间，要求：用鼠标或键盘选择列表条目时，返回第二列的值，并在文本框中显示。本例的运行结果如图 6.10 所示。



图 6.10 运行效果

设计步骤：

1. 创建一个表单，并向表单中添加一个 List1 列表框和一个 Text1 文本框；

2. 打开表单数据环境设计器，添加表 Employee；

3. 打开列表框生成器，对列表框进行设置：

(1) 选择 List1 列表框，并打开其列表框生成器，如图 6.11 所示；

(2) 选择列表项选项卡，并在用此填充列表的选项中选择：表或视图中的字段；选择可用字段身份号、姓名、工作时间，其设置如图 6.11(a) 所示；

(3) 选择样式选项卡，对列表框的显示样式和显示的行数等进行设置，其设置结果如图 6.11(b) 所示；

(4) 选择布局选项卡，在该选项卡中可以对列宽进行调整，并可隐藏列等，其设置如图 6.11(c) 所示；

(5) 选择值选项卡，在第一个组合框中选择姓名为该控件的返回值，还可以设置字段名，以便将控件的返回值送给指定的字段保存，本例没有设置字段名，其设置如图 6.11(d) 所示。

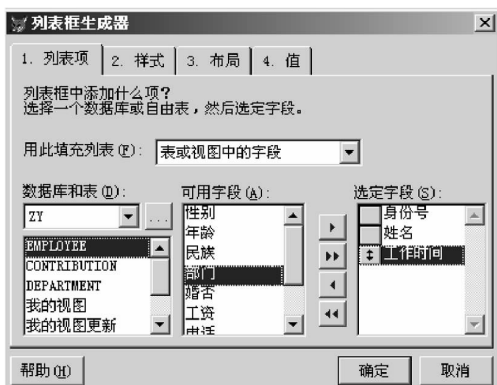


图 6.11(a) “列表项”选项卡的设置

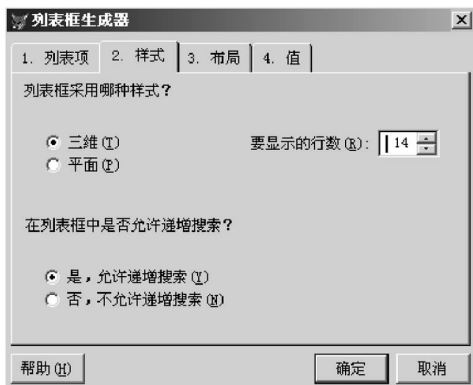


图 6.11(b) “样式”选项卡的设置

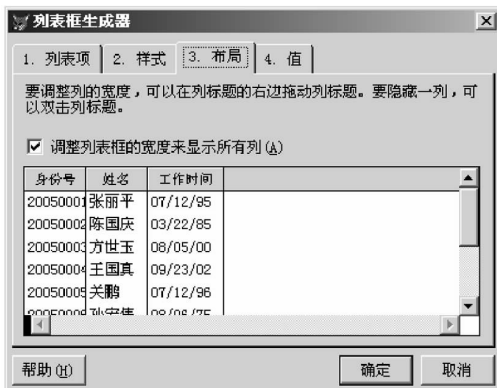


图 6.11(c) “布局”选项卡的设置

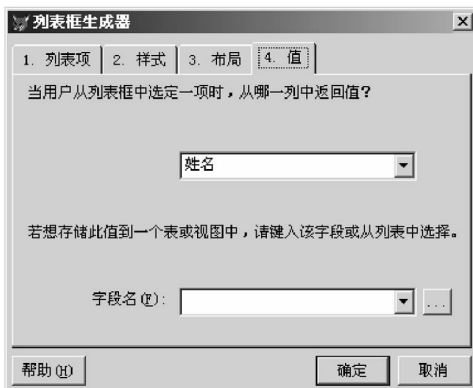


图 6.11(d) “值”选项卡的设置

图 6.11 列表框与字段绑定时的生成器设置

4. 设置 List1 列表框的 Interactivechange 事件代码：

```
ThisForm.Text1.Value=This.Value
```

通过上例我们可以掌握当在列表框生成器的“列表项”选项卡的“用此填充列表”中选择:表或视图中的字段时生成器的设置方式,关于另外两个选项:手工输入数据、数组中的值的设置方式可以通过生成器的引导来进行设置完成,这里就不一一介绍了。

6.1.2.2 组合框

组合框和列表框类似,也可提供一组条目供用户选择,与列表框之间主要区别如下:

- 1. 组合框平时只显示一个条目,用户单击组合框的向下按钮后才显示可滚动的下拉列表,以节省显示空间;
- 2. 组合框没有 MultiSelect 属性,不提供多重选择;
- 3. 组合框又分为下拉组合框和下拉列表框两种形式,前者允许输入条目,后者仅有选项功能。

下面,我们通过一个例子来学习组合框的使用。

例 6.8 在例 6.4 中,表单 Employee 运行时,只是显示第一条记录的信息。现在修改该表单,将身份号由文本框改为组合框,当选中某位员工的身份号时,表单显示该员工的信息,如图 6.12 所示。

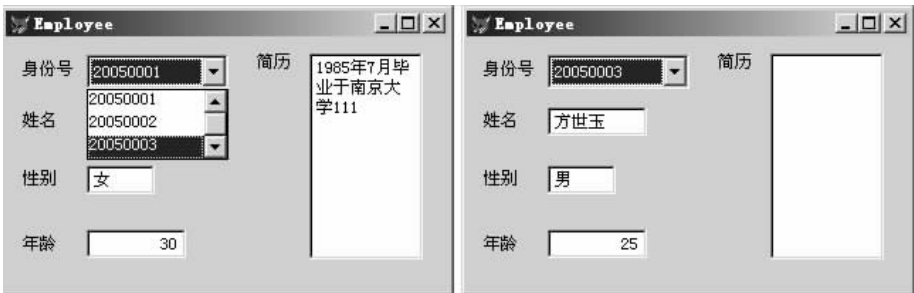


图 6.12 运行效果

设计步骤：

- 1. 在表单设计器中打开表单 Employee,删去身份号对应的文本框 Text1,添加组合框控件 Combo1,并设置相关属性,见表 6.10;

表 6.10 控件主要属性设置

对 象	属 性	属 性 值	说 明
Combo1	ControlSource	Employee. 身份号	
	DisplayCount	3	指定显示在下拉列表框中条目个数为 3
	RowSource	Employee. 身份号	
	RowSourceType	6—字段	设置值及含义与列表框相同
	Style	2—下拉列表框	仅可在列表中选择

- 2. Combo1 的 InteractiveChange 事件代码如下：

```
ThisForm.Refresh                      && 刷新表单
```

6.1.2.2.1 组合框属性

组合框与列表框的大多数属性、方法及事件都相同,比如:RowSourceType 属性、RowSource 属性、ColumnCount 属性、ListCount 属性、List 属性、Selected 属性及 Value 属性和 AddItem 方法、AddListItem 方法、RemoveItem 方法和 RemoveListItem 方法等。这些属性、方法的使用,请参考前面列表框的说明。这里仅介绍 Style 属性,通过设置该属性可选择组合框的形式,如表 6.11 所示。

表 6.11 Style 属性的设置值

设置值	说 明
0	下拉组合框。用户既可以从下拉列表中选择,又可以输入。
2	下拉列表框。用户只能从下拉列表中选择,不可以输入。

6.1.2.2.2 组合框的事件与方法程序

1. KeyPress 事件:

本事件在用户按下并释放某个键时发生。该事件的应用对象包括:文本框、编辑框、列表框、微调控件、复选框、选项按钮、命令按钮。

2. 表单的 Refresh 方法:

该方法用以重新绘制表单,刷新所有值。当表单被刷新时,表单上所有控件也都被刷新。

6.1.2.3 微调控件

使用微调控件可以让用户通过单击微调控件的上箭头或下箭头,或者直接在微调框中键入一个数值,来实现将微调控件的值在一个数值范围内进行选择的功能。

例 6.9 在前例的基础上,继续修改表单 Employee。将年龄由文本框改为微调控件,并将员工年龄的值限定在 18 至 65 之间,如图 6.13 所示。



图 6.13 运行效果

设计步骤:

在表单设计器中打开表单 Employee,删去年龄对应的文本框 Text4,添加微调控件 Spinner1,并设置相关属性,见表 6.12。

表 6.12 控件主要属性设置

对 象	属 性	属 性 值	说 明
Spinner1	ControlSource	Employee. 年龄	
	SpinnerHighValue	65	设定通过单击微调按钮输入的最大值
	SpinnerLowValue	18	设定通过单击微调按钮输入的最小值
	KeyBoardHighValue	65	设定从键盘输入微调框的最大值
	KeyBoardLowValue	18	设定从键盘输入微调框的最小值

由上例可知,微调控件的值为数值型。除了在上例中使用到的 5 个属性之外,微调控

件还有如下几个常用属性和事件。

1. Value 属性：

本属性返回微调控件的当前值。

2. Increment 属性：

本属性设定按一次箭头按钮的增减数，默认为 1.00。

3. DownClick 事件和 UpClick 事件：

DownClick 事件是按微调控件的向下箭头时发生；UpClick 事件是按微调控件的向上箭头时发生。在上例中，添加微调控件 Spinner1 的 UpClick 事件代码如下：

```
Wait Str(This.Value)Windows Timeout 1
```

运行表单，单击微调控件向上箭头，请观察屏幕上的变化。

6.2 控制类控件

本节讨论命令按钮、命令按钮组、复选框、选项按钮组与计时器 5 个控件，这 5 个控件都属于控制类控件。

6.2.1 命令按钮与命令按钮组

命令按钮常用于启动某个事件代码、完成特定功能，在程序中起控制作用。

6.2.1.1 命令按钮

例 6.10 设计如图 6.14 所示的登录界面，当用户输入用户名和密码并单击“确定”按钮后，检查是否为合法用户，若是合法用户，则显示“欢迎您！”消息框，并关闭表单；若不正确，则显示“用户名或密码错误，请重试...”，如果连续 3 次输入错误，显示“登录失败”并关闭表单。单击“退出”按钮，直接关闭表单。合法用户的用户名为 Employee 表中的姓名，密码为该员工的身份号。



图 6.14 运行效果

设计步骤：

1. 创建表单 Login.scx；
2. 打开数据环境设计器，添加 Employee 表；
3. 在表单上创建 2 个标签、2 个文本框和 2 个命令按钮控件，形式如图 6.14 所示；
4. 设置属性：主要属性设置见表 6.13；

表 6.13 控件主要属性设置

对 象	属 性	属 性 值	说 明
Form1	BorderStyle	2—固定对话框	不允许改变表单大小
	Caption	登录	
	Maxbutton	. F.	表单的最大化按钮不可用
	Autocenter	. T.	表单启动时自动居中

续表

Label1	Caption	用户名：	
Label2	Caption	密 码：	
Text2	Passwordchar	*	用户输入密码时,回显“*”
Command1	Caption	确定	
	Default	. T.	按下 Enter 键,该命令按钮进行响应
Command2	Caption	退出	
	Cancel	. T.	按下 Esc 键,该命令按钮进行响应

5. 表单 login 的 load 事件代码：

* 定义一个公共变量 Ncount 来统计用户错误输入的次数,初值为 0

Public Ncount

Ncount=0

6. 表单 login 的 Unload 事件代码:(表单被释放时发生)

* 释放公共变量 Ncount

Release Ncount

7. 命令按钮 Command1 的 Click 事件代码如下：

* 定义两个变量 Username 和 Userpass 来保存用户输入的用户名与密码

* Alltrim()函数去掉字符串前后空格

Username= Alltrim(ThisForm. Text1. Value)

Userpass =Alltrim(ThisForm. Text2. Value)

* 用== 进行字符串的精确比较。请读者思考如果此处改用“=”并能；

使程序正确运行,该如何修改代码？

Locate For Alltrim(姓名)==Username And Alltrim(身份证号)== Userpass

* Found()函数返回. T. ,表示找到匹配记录;否则返回. F.

If Found()

 MessageBox("欢迎您！",0+64,"登录")

 ThisForm. Release

Else

 * 公共变量 Ncount 起到计数器的作用

 Ncount=Ncount+1

 If Ncount =3

 MessageBox("登录失败！",0+16,"登录")

 ThisForm. Release

 Else

 MessageBox("用户名或密码错误,请重试...",0+32,"登录")

 * 下面三行代码将文本框内容清空,并为文本框 Text1 设置焦点

 ThisForm. Text1. Value=""

```
        ThisForm.Text2.Value=""  
        ThisForm.Text1.Setfocus  
    Endif  
Endif
```

8. 命令按钮 Command2 的 Click 事件代码如下:

```
ThisForm.Release
```

6.2.1.1.1 命令按钮的属性

命令按钮是最常见的控制类控件之一。在上例中,我们设置了命令按钮的 Caption (标题)属性、Default 属性、Cancel 属性和 Click(单击)事件的代码。下面为大家介绍命令按钮的其他常用属性和事件。

1. Cancel 属性:

Cancel 属性为 .T. 的命令按钮也称为“取消”按钮。在表单运行时,当用户按 Esc 键时,将引发“取消”按钮的 Click 事件。

2. Default 属性:

Default 属性为 .T. 的命令按钮也称为“确认”按钮。在表单运行时,当用户按 Enter 键时,将引发“确认”按钮的 Click 事件。一个表单中,只能有一个“确认”按钮。若一个表单内有多个命令按钮时,若此时设置某个按钮为“确认”按钮,则以前设置过“确认”按钮的 Default 属性自动变为 .F.。

3. Enabled 属性:

Enabled 属性用来确定控件是否可用,即能否响应用户引发的事件。当 Enabled 属性取值为 .T. 时,控件可用;属性取值为 .F. 时,控件不可用。

4. Visible 属性:

Visible 属性用来确定控件是否可见,其值为 .T. 时,控件可见;其值为 .F. 时,控件不可见。需注意的是,若一个控件的 Enabled 属性值设为 .T.,而 Visible 属性值设为 .F.,该控件实际上不可用。

5. Picture 属性:

本属性可用来设置命令按钮上显示的图形。

6. ToolTipText 属性:

在本属性中可以设置提示文本,最多 127 个字符。表单运行时,当鼠标指针移到设置了此属性的控件时,会出现提示文本。特别注意,该属性只有当包含控件的表单或工具栏的 ShowTips 属性取值为 .T. 时方可用。

6.2.1.1.2 表单对象的 Load 事件和 Unload 事件

Load 事件在表单对象创建之前引发;Unload 事件在表单对象释放时引发,是表单对象释放时最后一个发生的事件。

例 6.11 继续修改例 6.9 中的表单 Employee,为表单添加 4 个命令按钮,通过命令按钮,实现记录的浏览。如图 6.15 所示。






图 6.15 运行效果图

设计步骤：

1. 用表单设计器打开表单 Employee，在表单上创建 4 个命令按钮，设置相关属性，见表 6.14；

表 6.14 控件主要属性设置

对 象	属 性	属 性 值	说 明
<div>cmdTop</div> <div>第一条</div>	Caption	第一条	
	Picture	d:\program files\microsoft visual studio\vfp98\wizards\wizbmps\wztop. bmp	设置命令按钮上的图形,选定图形文件
	Name	cmdTop	设置“第一条”命令按钮的名称
<div>cmdPrev</div> <div>上一条</div>	Caption	上一条	
	Picture	d:\program files\microsoft visual studio\vfp98\wizards\wizbmps\wzback. bmp	设置命令按钮上的图形,选定图形文件
	Name	cmdPrev	设置“一条”命令按钮的名称
<div>cmdNext</div> <div>下一条</div>	Caption	下一条	
	Picture	d:\program files\microsoft visual studio\vfp98\wizards\wizbmps\wznext. bmp	设置命令按钮上的图形,选定图形文件
	Name	cmdNext	设置“下一条”命令按钮的名称
<div>cmdEnd</div> <div>最后一条</div>	Caption	最后一条	
	Picture	d:\program files\microsoft visual studio\vfp98\wizards\wizbmps\wzend. bmp	设置命令按钮上的图形,选定图形文件
	Name	cmdEnd	设置“最后一条”命令按钮的名称

2. 添加表单 Employee 的 Init 事件代码：

＊ 打开表单时，默认显示第一条记录，此时“第一条”和“上一条”按钮设置为不可用。

ThisForm. cmdTop. Enabled=. F.

ThisForm. cmdPrev. Enabled=. F.

3. “第一条”命令按钮 CmdTop 的 Click 事件代码：

Go Top

```

ThisForm.CmdTop.Enabled=.F.           && 禁用“第一条”按钮
ThisForm.CmdPrev.Enabled=.F.          && 禁用“上一条”按钮
ThisForm.CmdNext.Enabled=.T.          && 启用“下一条”按钮
ThisForm.CmdEnd.Enabled=.T.           && 启用“最后一条”按钮
ThisForm.Refresh                       && 刷新表单

```

4. “上一条”命令按钮 CmdPrev 的 Click 事件代码：

```

Skip -1
If Bof ( )
    MessageBox ("已到第一条记录！",48,"信息窗口")
    This.Enabled=.F.                   && 引用自己
    ThisForm.CmdTop.Enabled=.F.
Endif
ThisForm.CmdNext.Enabled=.T.
ThisForm.CmdEnd.Enabled=.T.
ThisForm.Refresh

```

5. “下一条”命令按钮 CmdNext 的 Click 事件代码：

```

Skip 1
If Eof ( )
    MessageBox ("已到最后一条记录！",48,"信息窗口")
    This.Enabled=.F.                   && 引用自己
    ThisForm.CmdEnd.Enabled=.F.
Endif
ThisForm.CmdTop.Enabled=.T.
ThisForm.CmdPrev.Enabled=.T.
ThisForm.Refresh

```

6. “最后一条”命令按钮 CmdEnd 的 Click 事件代码：

```

Go Bottom
ThisForm.CmdTop.Enabled=.T.           && 启用“第一条”按钮
ThisForm.CmdPrev.Enabled=.T.          && 启用“上一条”按钮
ThisForm.CmdNext.Enabled=.F.          && 禁用“下一条”按钮
ThisForm.CmdEnd.Enabled=.F.           && 禁用“最后一条”按钮
ThisForm.Refresh

```

说明：

若本例中 4 个命令按钮的 Caption 属性设置为空,将得到如图 6.16 所示的图形按钮。



图 6.16 图形按钮

1. 对象的 Init 事件：

Init 事件在对象建立时发生。在引发表单对象的 Init 事件前,先引发它所包含的控件对象的 Init 事件,所以在表单对象的 Init 事件代码中可以访问它所包含的所有控件对象。运行表单时,先引发表单的 Load 事件,再引发表单的 Init 事件。

2. Destroy 事件:

Destroy 事件在对象释放时发生。表单对象的 Destroy 事件在它所包含的控件对象的 Destroy 事件前引发,所以在表单对象的 Destroy 事件代码中可以访问它所包含的所有控件对象。关闭表单时,先引发表单的 Destroy 事件,再引发表单的 Unload 事件。

3. 对象的 Name 属性:

该属性值是一个字符表达式,用来指定在程序代码中引用对象时所用的名称。当新建一个对象时,默认名称是对象类型加上一个正整型的序数。例如,在表单上添加第 4 个命令按钮控件时,其 Name 属性值默认为 Command4,在程序中可以通过 ThisForm.Command4 来使用该控件。

为了代码的清晰性,往往需要修改控件的 Name 属性值,做到见名知义。常见命名规则是对象类型的缩写加上控件功能的描述字符。如上例中,“最后一条”按钮其 Name 属性值由默认的 Command4 改为 CmdEnd,其中,“Cmd”是“Command”的缩写,“End”表示此按钮功能是转到最后一条记录,在程序中通过 ThisForm.CmdEnd 来使用它。

另外,控件创建时,Caption(控件标题)的属性值和 Name(控件名称)的属性值相同。初学者往往容易混淆这两个属性,出现错误,请读者注意。

6.2.1.2 命令按钮组

当一个表单上需要多个命令按钮时,还可以选择另一种控件——命令按钮组。命令按钮组是包含一组命令按钮的容器控件,用户可以单独操作或作为一组来统一操作命令按钮。

例 6.12 设计如图 6.17 所示的表单 Form5。表单上有一命令按钮组和一个标签。当单击“时间”按钮时,标签显示当前系统时间;当单击“日期”按钮时,标签显示当前系统日期;当单击“退出”按钮时,关闭表单。

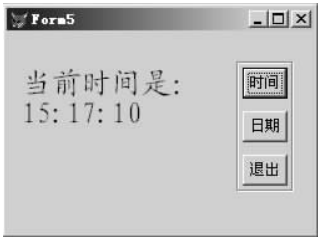


图 6.17 运行效果

设计步骤:

1. 创建表单 Time.scx;
2. 在表单上创建 1 个标签和 1 个命令按钮组;
3. 设置属性:主要属性设置见表 6.15;

表 6.15 控件主要属性设置

对 象	属 性	属 性 值	说 明
Label1	Caption	空	
	FontName	楷体_GB2312	
	FontSize	18	
	WordWrap	. T.	允许标签内文本换行显示

续表

CommandGroup1		ButtonCount	3	按钮组中有 3 个按钮
		Value	1	
	Command1	Caption	时间	
	Command2	Caption	日期	
	Command3	Caption	退出	

4. 添加 CommandGroup1 的 Click 事件代码如下：

```
Do Case
    Case This. Value=1                                &&. 单击"时间"按钮
        ThisForm. Label1. Caption='当前时间是:' + Time ( )
    Case This. Value=2                                &&. 单击"日期"按钮
        * Date()返回当前日期,Dtoc()将日期型数据转换为字符型数据
        ThisForm. Label1. Caption='当前日期是:' + Dtoc(Date ( ))
    Case This. Value=3                                &&. 单击"退出"按钮
        ThisForm. Release
Endcase
```

6.2.1.2.1 命令按钮组内命令按钮控件的选定

在对某个控件设置属性、方法程序和事件时,通常需要在选定该控件的基础上进行。前面介绍的几个控件都可以在表单上直接用鼠标单击选定;但对于容器中的控件,则不能直接用鼠标单击选定,而应该首先激活容器,使其处于编辑状态,然后再用鼠标单击选定的容器中的控件。在容器被激活的编辑状态下,若用鼠标单击容器以外的其他地方时,则容器失去活性,容器内的控件将又不能用鼠标直接选定。这里介绍一下容器的激活方法:

- 1. 直接用鼠标右击容器,在弹出的快捷菜单中选择“编辑”菜单项后,容器控件处于激活的编辑状态;
- 2. 同其他控件一样,我们也可以利用属性窗口的对象下拉组合框来选择相应的容器中的控件,此时,该控件所在的容器自动处于激活的编辑状态。

由于命令按钮组控件属于容器类控件,命令按钮组内的命令按钮控件属于命令按钮组容器内的控件,所以,命令按钮组内的某个命令按钮控件的选定也可以使用以上两种方法,自首先激活容器,然后用鼠标直接选定容器内的控件。

6.2.1.2.2 命令按钮组常用属性

- 1. ButtonCount 属性:
指定命令按钮组中按钮的个数。
- 2. Buttons 属性:

用于存取命令按钮组中每个按钮的数组。例如,将命令按钮组 CmdGroup1 中第 3 个按钮设置为不可用,代码如下:

```
ThisForm. CmdGroup1. Buttons(3). Enabled=. F.
```

3. Value 属性:

当单击某个命令按钮时,命令按钮组的 Value 属性将获得一个数值或一个字符串。当 Value 属性值为 1(默认值)时,将获得命令按钮的序号,它是一个数值;当 Value 属性值为空时,将获得命令按钮的 Caption 属性值,它是一个字符串。在命令按钮组的 Click 事件代码中,通常我们用多分支判断 Do Case 语句对命令按钮组的 Value 属性值进行判断,来确定用户单击的是哪个按钮,并执行相应动作,如例 6.12 中的 CommandGroup1 的 Click 事件代码。

6.2.1.2.3 命令按钮组的 Click 事件

发生在用户单击命令按钮组时。请读者注意,如果命令按钮组内的某个按钮有自己的 Click 事件代码,那么当单击该按钮时,会优先执行按钮本身的代码,而不会执行命令按钮组的 Click 事件代码。

6.2.1.2.4 命令按钮组生成器

设置命令按钮组的常用属性,使用命令按钮组生成器更为方便。在命令按钮组的快捷菜单中选中“生成器...”,即可打开对话框。该对话框包含以下 2 个选项卡。

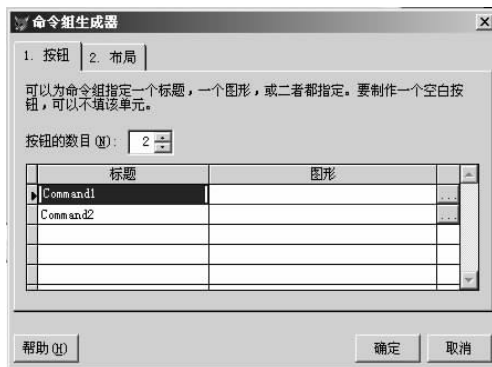


图 6.18(a) “按钮”选项卡

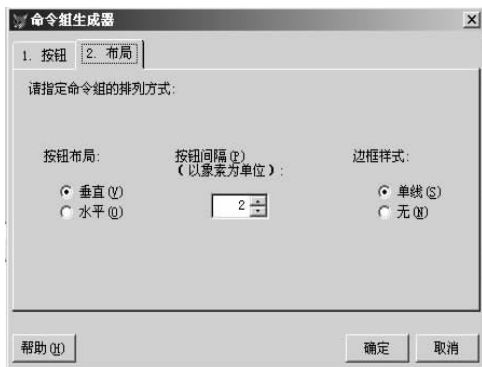


图 6.18(b) “布局”选项卡

图 6.18 命令按钮组生成器的布局选项卡

1. 按钮的数目:可通过微调控件来指定命令按钮组中按钮数,与命令按钮组的 ButtonCount 属性对应;

2. 表格:

(1) 标题列:指定各按钮的标题,与命令按钮组中各按钮的 Caption 属性对应;

(2) 图形列:指定各按钮的图形,与命令按钮组中各按钮的 Picture 属性对应。

3. 按钮布局:通过单选按钮来指定命令按钮组内各按钮的排列方向,如是按垂直方向排列,还是按水平方向排列;

4. 按钮间隔:通过微调控件来调整各按钮间的间隔;按钮布局和按钮间隔的调整会影响命令按钮组的 Height 属性和 Width 属性;

5. 边框样式:指定命令按钮组的边框为单线,还是无边框,与命令按钮组的 BorderStyle 属性对应。

例 6.13 试用命令按钮组来完成例 6.11 中命令按钮的功能。



图 6.19 运行效果

设计步骤：

- 1. 用表单设计器打开表单 Employee,并删除表单中已经设计的命令按钮,及有关命令按钮引用的一些属性等；
- 2. 向表单中添加一个命令按钮组 CommandGroup1,该命令按钮组的各命令按钮控件的属性设置,如表 6.16 所示；

表 6.16 命令按钮组各命令按钮控件的属性设置

对 象	属 性	属 性 值	说 明
Command1	Caption	第一条	
	Picture	D:\Program Files\Microsoft Visual Studio\ Vf p98\ Wizards\ Wizbmps\ Wztop. Bmp	设置命令按钮上的图形,选定图形文件
Command2	Caption	上一条	
	Picture	D:\Program Files\Microsoft Visual Studio\ Vf p98\ Wizards\ Wizbmps\ Wzback. Bmp	设置命令按钮上的图形,选定图形文件
Command3	Caption	下一条	
	Picture	D:\Program Files\Microsoft Visual Studio\ Vf p98\ Wizards\ Wizbmps\ Wznex t. Bmp	设置命令按钮上的图形,选定图形文件
Command4	Caption	最后一条	
	Picture	D:\Program Files\Microsoft Visual Studio\ Vf p98\ Wizards\ Wizbmps\ Wzend. Bmp	设置命令按钮上的图形,选定图形文件

3. 修改表单 Employee 的 Init 事件代码：

```
* 设置命令按钮组中的第一、二个按钮的 Enabled 属性为 . F. ,即禁用第一、二个按钮
ThisForm. Commandgroup1. Command1. Enabled=. F.
ThisForm. Commandgroup1. Command2. Enabled=. F.
```

4. 设置 CommandGroup1 的 Click 事件代码：

Do Case

Case This. Value="第一条"

Go Top

* 利用命令按钮组 Buttons 属性,禁用"第一条"按钮

This. Buttons(1). Enabled=. F.

利用控件名的引用,禁用"上一条"按钮

This. Command2. Enabled=. F.

This. Buttons(3). Enabled=. T.

&&. 启用"下一条"按钮

This. Command4. Enabled=. T.

&&. 启用"最后一条"按钮

ThisForm. Refresh

&&. 刷新表单

Case This. Value="前一条"

Skip -1

If Bof ()

Messagebox ("已到第一条记录!",48,"信息窗口")

This. Command2. Enabled=. F.

This. Command1. Enabled=. F.

Endif

This. Command3. Enabled=. T.

This. Command4. Enabled=. T.

ThisForm. Refresh

Case This. Value="后一条"

Skip 1

If Eof ()

Messagebox ("已到最后一条记录!",48,"信息窗口")

This. Command3. Enabled=. F.

This. Command4. Enabled=. F.

Endif

This. Command1. Enabled=. T.

This. Command2. Enabled=. T.

ThisForm. Refresh

Case This. Value="最后一条"

Go Bottom

This. Command1. Enabled=. T.

&&. 启用"第一条"按钮

This. Command2. Enabled=. T.

&&. 启用"上一条"按钮

This. Command3. Enabled=. F.

&&. 禁用"下一条"按钮

This. Command4. Enabled=. F.

&&. 禁用"最后一条"按钮

ThisForm. Refresh

Endcase

6.2.2 复选框与选项按钮组

复选框与选项按钮组(也称单选按钮)是对话框中的常见对象。复选框用于标记一个两值状态,如“真”或“假”。当状态为“真”时,复选框内显示一个对勾(√);状态为“假”时,复选框内为空白。选项按钮组也称选项组,是包含选项按钮的一种容器控件。一个选项组中往往有多个选项按钮,但用户只能选定其中一个,被选定的按钮显示一个圆点,其他按钮变为未选定状态。

例 6.14 修改表单 Employee,将性别由文本框改为选项按钮组;添加婚否字段,用复选框表示,如图 6.20 所示。

设计步骤:

- 1. 在表单设计器中打开 Employee.scx;
- 2. 在表单上创建 1 个复选框,将性别对应的文本框去掉,添加一个选项组并修改外观;
- 3. 设置属性:主要属性设置见表 6.17。



图 6.20 运行效果

表 6.17 控件主要属性设置

对 象		属 性	属 性 值	说 明
Check1		Caption	婚否	复选框旁边的文字显示为“婚否”
		ControlSource	Employee. 婚否	
		Value	0	默认值,未选
Optiongroup1		ControlSource	Employee. 性别	
	Option1	Caption	男	
		Value	1	
		Style	0—标准	
	Option2	Caption	女	
		Value	0	
		Style	0—标准	

6.2.2.1 复选框的属性

1. Caption 属性:

指定显示在复选框旁边的文字内容。

2. Value 属性:

指明复选框的当前状态,取值及含义见表 6.18。

表 6.18 Value 属性的设置值

设置值	说 明
0 或. F.	未选中,(默认值)
1 或. T.	选中
2 或. Null.	不确定,只在代码中有效

3. Style 属性:

指定复选框的外观,取值及含义见表 6.19。

表 6.19 复选框 Style 属性的设置值

设置值	外 观	选定状态
0—标准	(默认值),显示方框,旁边显示 Caption 文本	出现复选标记(✓),如:  婚否
1—图形	显示为按钮,按钮上居中显示 Caption 文本	按钮呈按下状态,如  婚否

4. ControlSource 属性:

指定与复选框建立联系的数据源。作数据源的字段或内存变量,类型可以是逻辑型或数值型。。F. 或 0 对应未选中状态;. T. 或 1 对应选中状态;. Null. 或 2 对应不确定状态。用户对复选框的操作结果会自动保存到数据源和 Value 属性中。

请读者注意,复选框的不确定状态并非指其不可用(Enabled 属性值为. F.),而是指复选框当前状态既不是“选中”,也不是“未选中”的确定状态,通常显示为灰色。用户可以对其进行选择操作,使其变为确定状态。

6.2.2.2 选项按钮组

选项按钮组是一种容器控件,设置时请注意区分选项组的属性和选项按钮的属性。激活该容器,选定选项按钮的方法参见本章 6.2.1.2.1 命令按钮组中有关容器中控件的选定方法的说明。

6.2.2.2.1 选项按钮组的常用属性

1. ButtonCount 属性:

指定选项组中按钮的数目。

2. 选项按钮组中各选项按钮的 Caption 属性:

指定各选项按钮的标题文本。

3. Value 属性:

选项按钮组的 Value 属性:用于指定选项组中哪个选项按钮被选中。该属性值类型是数值型或字符型的。若为数值型 N,则表示选项组中第 N 个选项按钮被选中;若为字符型值 C,则表示选项组中 Caption 属性值为 C 的选项按钮被选中。

4. 选项按钮组中各选项按钮的 Value 属性:

指明各选项按钮的状态,1 表示选定,0 表示未选定。

5. 选项按钮组中各选项按钮的 Style 属性:

指定各选项按钮的外观,取值及含义见表 6.20。

表 6.20 选项按钮组中各选项按钮的 Style 属性的设置值

设置值	外 观	选定状态
0—标准	(默认值),显示方框,旁边显示 Caption 文本	出现复选标记(—)  女
1—图形	显示为按钮,按钮上居中显示 Caption 文本	按钮呈按下状态 

6. ControlSource 属性:

指定与选项按钮组建立联系的数据源。数据源可以是字段或内存变量,类型可以是数值型或字符型数据。数据源的值将决定选项组的 Value 属性的值。用户对选项组的操作结果亦会自动保存到数据源和 Value 属性中。

6.2.2.2.2 选项按钮组生成器

设置选项按钮组的常用属性,还可以使用选项按钮组生成器。在选项按钮组的快捷菜单中选中“生成器...”,即可打开“选项组生成器”对话框,如图 6.21 所示。该对话框包含以下 3 个选项卡。

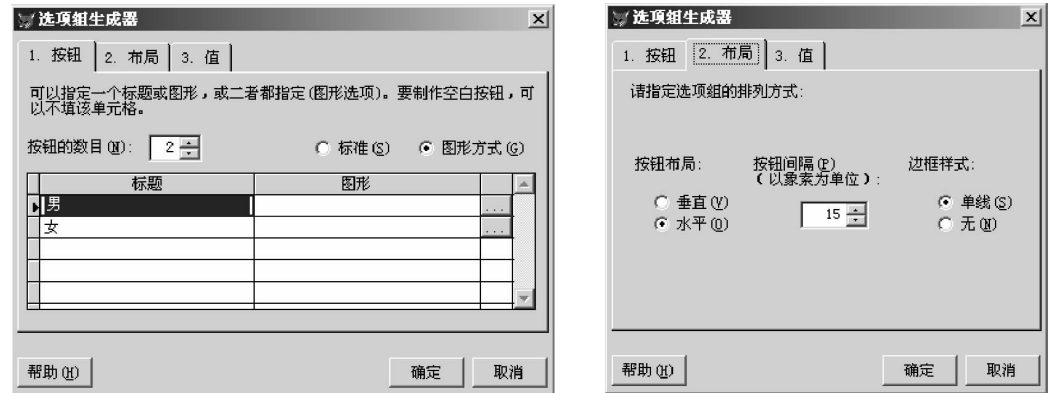


图 6.21 (a)“按钮”选项卡

图 6.21 (b)“布局”选项卡

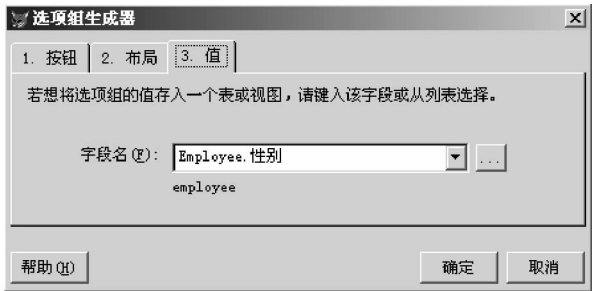


图 6.21 (c)“值”选项卡

图 6.21 选项按钮组生成器的 3 个选项卡

选项按钮组生成器中的按钮选项卡和布局卡的设置与命令按钮组的设置相似,不同的是在按钮选项卡中,多出了一个选项按钮组,由用户选择选项按钮的外观图形方式,当选择为标准或图形方式时,相当于将选择按钮组中的所有按钮的 Style 属性设置为 0 或

1. 第三个“值”选项卡,是用于绑定一个数据源,相当于设置选项按钮组的 ControlSource 属性,如:ThisForm. Optiongroup1. ControlSource= "Employee. 性别"

例 6.15 例 6.12 中是使用命令按钮组来控制时间、日期和退出,将其改变为利用选项按钮组来控制。

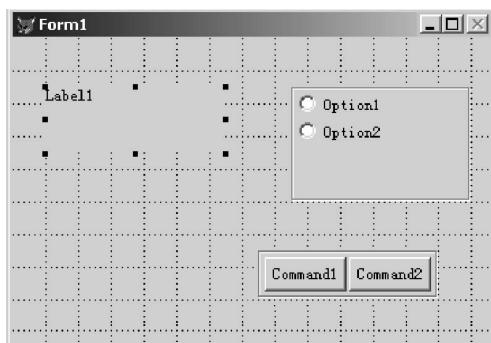


图 6.22 设计界面

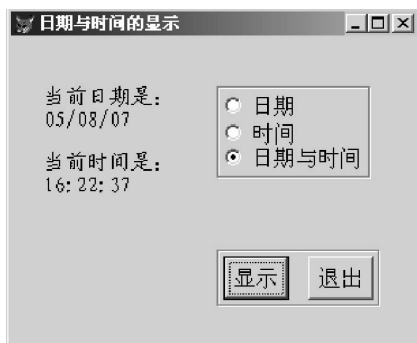


图 6.23 运行效果

设计步骤:

1. 创建表单 TimeDate. scx;
2. 在表单上创建一个标签、一个选项按钮组和一个命令按钮组,如图 6.22 所示;
3. 设置 Form1 的 Init 事件代码:

```
ThisForm. AutoCenter = . T.
```

```
This. Caption = "日期与时间的显示"
```

* 设置 Label1 标签的属性

```
ThisForm. Label1. Caption = "当前日期是:" + Chr(13) + Dtoc(Date())
```

```
This. Label1. FontName = "楷体_ GB2312"
```

```
This. Label1. FontSize = 12
```

```
This. Label1. AutoSize = . T.
```

```
This. Label1. Wordwrap = . T.
```

* 设置 OptionGroup1 选项按钮组的属性

```
This. OptionGroup1. ButtonCount = 3
```

```
This. OptionGroup1. Option1. Caption = "日期"
```

```
This. OptionGroup1. Option2. Caption = "时间"
```

```
This. OptionGroup1. Option3. Caption = "日期与时间"
```

```
This. OptionGroup1. Option1. FontSize = 12
```

```
This. OptionGroup1. Option2. FontSize = 12
```

```
This. OptionGroup1. Option3. FontSize = 12
```

```
This. OptionGroup1. Option1. AutoSize = . T.
```

```
This. OptionGroup1. Option2. AutoSize = . T.
```

```
This. OptionGroup1. Option3. AutoSize = . T.
```

```
This. OptionGroup1. AutoSize = . T.
```

```
This. OptionGroup1. Value="日期"
```

* 设置 CommandGroup1 命令按钮组的属性

```
This. CommandGroup1. Command1. Caption="显示"
```

```
This. CommandGroup1. Command2. Caption="退出"
```

```
This. CommandGroup1. Command1. FontSize=12
```

```
This. CommandGroup1. Command2. FontSize=12
```

```
This. CommandGroup1. Command1. AutoSize=. T.
```

```
This. CommandGroup1. Command2. AutoSize=. T.
```

```
This. CommandGroup1. AutoSize=. T.
```

```
This. CommandGroup1. Command1. Default=. T.
```

```
This. CommandGroup1. Command2. Cancel=. T.
```

```
This. CommandGroup1. Value="显示"
```

4. 设置 CommandGroup1 的 Click 事件代码:

```
Do Case
```

```
Case This. Value="显示"
```

```
Do Case
```

```
Case ThisForm. OptionGroup1. Value=="日期"
```

```
ThisForm. Label1. Caption="当前日期是:"+Chr(13)+Dtoc(Date())
```

```
ThisForm. Label1. AutoSize=. T.
```

```
Case ThisForm. OptionGroup1. Value=="时间"
```

```
ThisForm. Label1. Caption="当前时间是:"+Chr(13)+Time()
```

```
ThisForm. Label1. AutoSize=. T.
```

```
Case ThisForm. OptionGroup1. Value=="日期与时间"
```

```
ThisForm. Label1. Caption="当前日期是:"+Chr(13)+Dtoc(Date());  
+Chr(13)+Chr(13)+"当前时间是:"+Chr(13)+Time()
```

```
ThisForm. Label1. AutoSize=. T.
```

```
EndCase
```

```
Case This. Value="退出"
```

```
ThisForm. Release
```

```
EndCase
```

该例题的所有控件的属性均用代码完成,在 CommandGroup1 的 Click 事件代码中,运用两个 Do Case 语句分别来判断命令按钮组和选项按钮组的动作,并去执行相应的命令语句,其中有些语句中的 Chr(13)函数的作用是产生一个回车换行符。

6.2.3 计时器

计时器控件能周期性地按指定时间间隔执行它的 Timer 事件代码。在程序中,可以利用它来处理需周期性执行的任务或一定时间后执行的任务。

例 6.16 修改应用程序封面表单 Welcome.scx,使其具备如下功能:

- 1. 将用户右击表单退出改为 4 秒钟后自动释放表单；
- 2. 随机改变 Label1 的标题文本“员工信息管理系统”的颜色,使之呈现闪烁状态。

设计步骤：

- 1. 在表单设计器中打开 Welcome.scx。
- 2. 在表单上创建 2 个计时器控件。
- 3. 设置计数器控件属性:主要属性设置见表 6.21。

表 6.21 控件主要属性设置

对 象	属 性	属 性 值	说 明
TimerExit	Name	TimerExit	
	Interval	4000	4 秒后执行 Timer 事件代码
TimerColor	Name	TimerColor	
	Interval	200	每隔 200 毫秒执行该控件的 Timer 事件代码

- 4. 设置 TimerExit 的 Timer 事件代码：

ThisForm. Release

- 5. 设置 TimerColor 的 Timer 事件代码：

Local I,J,K

* Rand()函数返回 0~1 间的一个随机数

* Int()函数返回参数的整数部分；

I=Int(Rand() * 255)

J=Int(Rand() * 255)

K=Int(Rand() * 255)

* Rgb()函数根据给定的红、绿、蓝颜色参数返回一个单一的颜色值。

ThisForm. Label1. Forecolor=Rgb(I,J,K)

ThisForm. Refresh

计时器的主要属性

计时器控件为不可见控件,即运行表单时看不到该控件。它通常用来检查系统时钟,确定是否到了执行某一任务的时间,适用于后台处理。常见属性和事件有如下 2 个：

- 1. Interval 属性：

属性值为数值型,用以指定 Timer 事件的触发时间间隔,单位为毫秒。除非必要,请尽量不要设置太小的时间间隔,否则会降低整个程序的性能。

- 2. Timer 事件：

该事件在经过 Interval 属性中指定时间的毫秒数时发生。

6.3 容器类控件

Visual FoxPro 的类有两大主要类型,因此 Visual FoxPro 对象也分为两大类型,它们便是容器类和控件类。容器类可以包含其他对象,并且允许访问这些对象。

常见的容器类有表单、命令按钮组、选项按钮组,还将在第七章第一节的 7.1.2 中介绍的表单集,及本节要重点介绍的表格、页框和 Container 容器控件等。

常见容器类及其所能包含的对象可参阅第五章第二节表 5.5 容器类及其所包含的对象。

6.3.1 表格

表格控件外观与 Browse 窗口类似,按行列方式处理数据。

例 6.17 创建表单 Department.scx,以表格方式显示表 Department 中的数据,如图 6.24。

设计步骤:

- 1. 创建表单 Department.scx,设置表单布局属性如图 6.23 的效果;
- 2. 在数据环境中添加表 Department.dbf ,用鼠标将数据环境中的 Department 表窗口的标题栏拖到表单上释放,即得表格控件 grdDepartment,内容为 Department 表中的数据。



图 6.24 运行效果

6.3.1.1 表格的属性

表格包含列(Column),而列又可以包含标头(Header)和其他控件。列中包含的默认控件为一个文本框,所以表格的默认功能很像一个“浏览”窗口。然而表格内部结构使它具有无限可扩展性。表格以及表格容器内的列、标头和控件等都可以设置属性、事件和方法,操作非常灵活。下面分别介绍它们的常用属性。

6.3.1.1.1 表格常用属性

- 1. RecordSourceType 属性与 RecordSource 属性:

前者指明表格数据源类型,后者指定表格数据源。RecordSourceType 属性及设置值见表 6.22。

表 6.22 RecordSourceType 属性及其设置值

设置值	说 明
0	表。数据来源于 RecordSource 属性指定的表,该表自动打开。
1(默认值)	别名。数据源为一已打开的表,RecordSource 属性说明该表的别名。
2	提示。运行时向用户提示数据源,若已打开某个数据库,则用户可以选择其中的一张表作为数据源。
3	查询。RecordSource 属性指定一个查询文件(.qpr 文件)。
4	SQL 语句。RecordSource 属性中指定一条 SQL 语句。

- 2. ColumnCount 属性:

该属性指定表格中的列对象的数目。默认值为-1,表示自动创建足够的列数来容纳数据源中的所有字段。最大值为 255。

- 3. AllowAddNew 属性:

指定是否可以通过表格向表添加新记录,默认值为 F.,不能添加。将 AllowAddNew

属性为 T. 值后,在运行过程中,当光标在最后一记录时,可通过按向下箭头键,来增加一条新的空白记录,由用户添加内容。否则只能用 Append Blank 或 Insert 等命令来添加新记录。

6.3.1.1.2 列对象常用属性

1. ControlSource 属性:

指定与列对象绑定的数据源,通常是表中的某个字段。若未设置该属性,则列中将显示表格数据源中下一个未显示的字段。

2. CurrentControl 属性:

指定列对象中显示和接受活动单元数据的控件名,默认控件是 Name 属性值为 Text1 的文本框。

3. Sparse 属性:

设置 CurrentControl 属性是影响列对象中的所有单元,还是只影响列对象中的活动单元。若属性值为 T. (默认值),表示只有列对象的活动单元使用 CurrentControl 属性指定控件来接收和显示数据,而其他的单元使用默认的文本框;若属性值为 F.,表示列对象的所有单元使用 CurrentControl 属性指定控件显示数据,活动单元也使用 CurrentControl 属性指定控件接收数据。

6.3.1.1.3 标头对象常用属性

1. Caption 属性:

指定标头对象的标题文本,显示于列顶部。

2. Alignment 属性:

指定标题文本的对齐方式,共 10 种对齐方式,请读者通过上机练习了解。

6.3.1.2 一对多表单

在实际应用中,常需要设计一对多表单,即在一个表单中显示两张表中的数据,并且两张表的关系是一对多的。在这种关系中,主表(也称为“父表”)中的每一个记录与相关表(也称为“子表”)中的多个记录相关联(每一个主关键字值在相关表中可出现多次)。利用表单向导我们可以轻松的生成一对多表单,下面通过实例来学习利用表单设计器创建一对多表单。

例 6.18 在例 6.14 的基础上,设计如图 6.25 所示的表单:Empotm,要求能查询员工的信息及个人捐款的详细信息。



图 6.25 运行效果

设计步骤：

1. 打开例 6.14 中已经设计好的表单 Employee,调整表单大小,并添加一个表格控件,各控件设置布局属性如图 6.25 所示；

2. 将表 Contribution.dbf 添加表单设计器的数据环境中,此时数据环境中有两个表,如表 Employee.dbf和表 Contribution.dbf。由于这两个表属于数据库表,在前面建立数据库时,这两个表之间已经建立了永久关系,所以向数据环境中添加这两个表后,在数据环境设计器中自动的出现如图 6.26 的一对多关系；

3. 将设置表格属性语句添加到 Form1 的 Init 事件代码中：

```
ThisForm. Caption="一对多表单"
ThisForm. Grid1. RecordSourceType=1
ThisForm. Grid1. RecordSource="Contribution"
```

4. 将表单另存为:Empotm.scx,并运行表单。

前例说明：

1. 建立数据环境中的表的一对多的关系：

前例中的重点在数据环境中设置表之间的一对多关系。如果在数据库中没有事先设置好两个表的一对多关系,在数据环境设计器中也可进行设置：

首先,分别在表 Employee.dbf 和 Contribution.dbf 的字段“身份号”上建立索引；其次,在数据环境中,用鼠标选中 Employee 表中的“身份号”拖到 Contribution 表的“身份号”索引上放开,此时出现一条连线,说明联系建立好了。读者可选中该联系,查看属性窗口,常用属性设置值见表 6.23,这些属性值在联系建立时自动生成,无需读者设置。

表 6.23 一对多联系的常用属性设置

对 象	属 性	属 性 值	说 明
Relation1	Name	Relation1	该联系的名称
	ChildAlias	Contribution	指定子表的别名
	ChildOrder	身份号	指定一对多关系中子表用到的索引。
	ParentAlias	Employee	指定父表的别名
	RelationalExpr	身份号	指定基于父表中的字段而又与子表中的索引相关的表达式。

2. 前例还可利用表格生成器快速完成：

在表单上添加一个表格控件,在其快捷菜单中选择“生成器...”,分别设置“表格项”、“样式”、“布局”、“关系”选项卡,如图 6.27 所示。

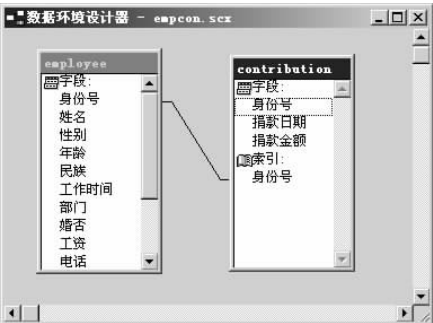


图 6.26 一对多关系数据环境

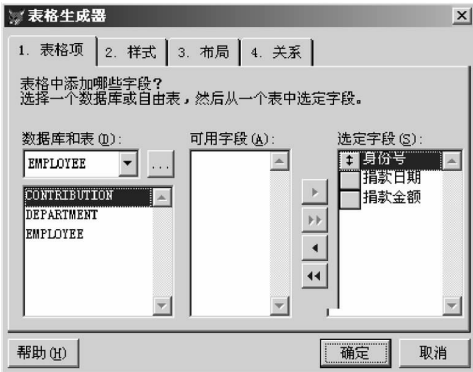


图 6.27 (a)“表格项”选项卡图

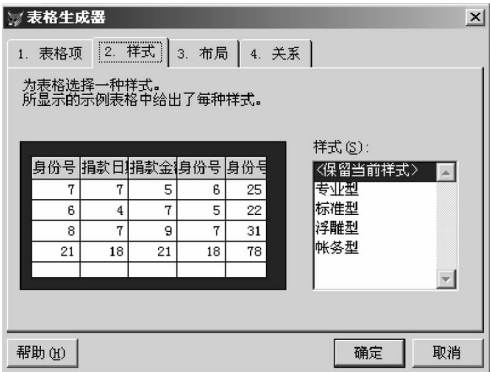


图 6.27 (b)“样式”选项卡



图 6.27 (c)“布局”选项卡

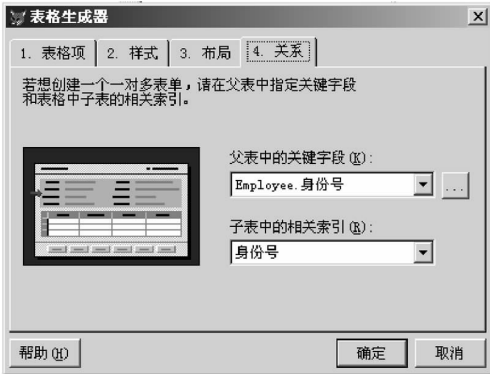


图 6.27 (d)“关系”选项卡

图 6.27 表格生成器

- (1) “表格项”选项卡用于指定表格控件中要显示的字段；
- (2) “样式”选项卡用于指定表格显示的样式；
- (3) “布局”选项卡用于指定列标题和显示字段值的控件类型；
- (4) “关系”选项卡用于指定一对多关系中父表和子表的相关索引。

3. 例 6.18 还可利用两个表格控件来完成,请大家试着来完成一下,提示,设置好数据环境后,对两个表格控件的 RecordSourceType 属性和 RecordSource 属性进行设置,即可。

6.3.2 页框

页框是包含页面(Page)的容器,可用以构建大家熟悉的选项卡对话框。同时,含有多页的页框可起到扩展表单面积的作用。

例 6.19 创建如图 6.28 的表单 Page,显示员工基本信息、捐献总额及部门代码等信息。

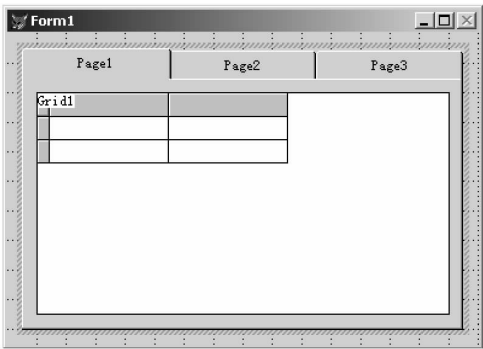


图 6.28 (a) 表单页框设计



6.28 (b) 职工基本信息



图 6.28 (c) 职工捐款信息



图 6.28 (d) 部门代码信息

图 6.28 运行效果

设计步骤：

1. 创建表单 Page.scx, 创建页框控件 PageFrame1, 并设置 PageCount 属性值为 3, 即建立三个页面；

2. 在每个页面上创建一个表格控件, 设置如图 6.28(a) 所示；

3. 设置 Form1 的 Init 事件代码：

This. AutoCenter = . T.

* 设置 PageFrame1 页框的 Page1 页面及该页面上的表格控件属性

This. PageFrame1. Page1. Caption = "职工基本信息"

This. PageFrame1. Page1. Grid1. RecordSourceType = 1

This. PageFrame1. Page1. Grid1. RecordSource = "Employee"

* 设置 PageFrame1 页框的 Page2 页面及该页面上的表格控件属性

This. PageFrame1. Page2. Caption = "职工捐款信息"

This. PageFrame1. Page2. Grid1. RecordSourceType = 4

This. PageFrame1. Page2. Grid1. RecordSource = "Select Employee. 身份号, 姓名, ;

Sum(捐款金额) As 捐款总额 From Employee Join Contribution;

On Employee. 身份号 = Contribution. 身份号;

Group By Employee. 身份号 Order By 1 Into Cursor Mycur2"

* 设置 PageFrame1 页框的 Page3 页面及该页面上的表格控件属性

```
This. PageFrame1. Page3. Caption="部门代码信息"
```

```
This. PageFrame1. Page3. Grid1. RecordSourceType=1
```

```
This. PageFrame1. Page3. Grid1. RecordSource="Department"
```

页框的属性:

一个页框中可包含多个页面,页面中可以包含多种控件。若要设置某个页面的属性或添加控件、设置控件属性,须先激活页框,然后再用鼠标选定此页面。激活页框的方法参见本章 6.2.1.2.1 命令按钮组中有关容器中控件的选定方法的说明。

页框常用属性如下:

1. PageCount 属性:

该属性说明页框包含页面的个数。取值范围从 0 到 99。

2. ActivePage 属性:

设定页框中活动页的页码。

3. Tabs 属性:

指定页框有无选项卡。属性值为 .T. (默认值),表示页框有选项卡;为 .F.,表示页框无选项卡。

4. TabStretch 属性:

该属性用来指定当页框控件的选项卡的页面标题过长时,选项卡时的行为。属性值为 1(默认值),选项卡页面标题在一行内显示,页面标题文本太长时会自动截取;属性值为 0,选项卡页面标题分多行显示。

6.3.3 容器

容器是一种可以包含其他控件的控件。请读者注意:和其他容器类控件使用相同,若想添加控件或对容器中已有的控件进行操作,需先激活该容器控件,即在容器控件快捷菜单中选择“编辑”菜单项。可以在容器内新建控件,也可以通过“剪切”、“粘贴”将表单上已有控件添加至容器中。当移动或删除一个容器控件时,它所包含的控件会一并移动或删除。

该控件的 SpecialEffect 属性可用来指定控件不同的格式选项,SpecialEffect 属性值为 0 时,容器为凸起状;SpecialEffect 属性值为 1 时,容器为凹下状;SpecialEffect 属性值为 2(默认值)时,容器为平面状。

6.4 连接类控件

VFP 除了可使用本身的数据外,还可以使用外部数据,从而扩展了 VFP 的功能。本节简介的 ActiveX 控件、ActiveX 绑定控件和超级链接控件,可轻松实现 VFP 与外界的连接。

6.4.1 ActiveX 控件

根据微软 MSDN 的定义,ActiveX 控件是可编程元素的新名称,以前称为 OLE 控件、OCX 或 OLE 自定义控件。ActiveX 控件与其他控件相同,可以把它放在表单上,使用户能够或加强同一个应用程序的交互能力。ActiveX 控件具有事件,并且可以集成到其他控件中。这些 ActiveX 控件具有 .ocx 的扩展名。

ActiveX 控件是封装了功能、属性、事件和方法程序的对象。ActiveX 控件提供了范围广泛的功能,使用这些功能可以容易地进行软件开发。Visual FoxPro 中的 ActiveX 控件必须包含在一个 OLE 容器控件(基类是 OLEControl)中,当向表单添加一个 ActiveX (OLEControl)控件时,将同时弹出的“插入对象”对话框,并由用户选择想添加到表单中的 ActiveX 控件种类。

6.4.1.1 “插入对象”对话框

在弹出的“插入对象”对话框中,可以选择三种方式向表单添加控件或对象,现分别介绍如下:



图 6.29 (a)新建对象

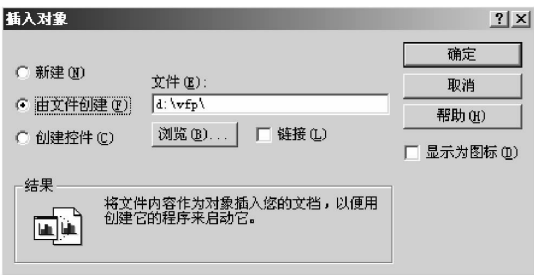


图 6.29 (b)由文件创建



图 6.29 (c)新建对象

图 6.29 插入对象对话框

1. “新建”选项按钮(如图 6.29(a))。

选择此选项,将在表单上新建一个对象,该对象为某种特定文件类型的文档。从“对象类型”列表中选中一项并按确定按钮后,VFP 将自动打开对应的应用程序,供用户输入。对话框中还有一个“显示为图标”复选框,选中时该文档在表单上显示为一个图标,表

单运行时双击该图标,VFP 调用对应的应用程序打开文档;不选中时,则在表单上直接显示文档内容。

2. “由文件创建”选项按钮(如图 6.29(b))。

选择此选项,用户先单击“浏览(B...)”按钮,在“浏览”窗口中选择某一个已存在的文件;或直接在“文件”文本框中输入路径及文件名。按“确定”按钮后表单窗口内即生成一文档对象。该文档是以图标显示,或直接显示内容,可由“显示为图标”复选框指定。

3. “创建控件”选项按钮(如图 6.29(c))。

选择此选项,用户可指定一个 ActiveX 控件并放置在表单上。ActiveX 控件种类繁多,用法各不相同。

例 6.20 创建如图 6.30 的表单:日历控件.scx。



图 6.30 运行效果

设计步骤:

1. 新建表单:日历控件.scx,创建 ActiveX 控件;

2. 在弹出的“插入对象”对话框中,首先选择“创建控件”选项,然后在“对象类型”列表框中选择“Calendar 控件 9.0”,单击“确定”按钮完成该控件的添加后,再调整控件的大小和位置等,如图 6.29(c)所示;

3. 在“对象类型”列表框中如果“Calendar 控件 9.0”不存在,其添加方式是:单击“添加控件”按钮,在弹出的“浏览”对话框中选择 Comctl32.ocx 文件后,单击打开,则“对象类型”列表框中就会出现“Calendar 控件 9.0”选项。Comctl32.ocx 文件的存放位置一般是在 VFP 的安装目录下,如本书设 VFP 安装在 D 盘的默认安装目录下,所以 Comctl32.ocx 文件在 D:\Program Files\Microsoft Visual Studio\Vfp98\Distrib.src\System 目录中。

6.4.1.2 向表单控件工具栏添加 ActiveX 控件按钮

为方便使用,还可以将 ActiveX 控件添加到表单控件工具栏中,操作如下:

1. 选择需要添加的 ActiveX 控件。

VFP 系统菜单中选择“工具”→“选项”菜单项→“控件”选项卡→“ActiveX 控件”选项按钮→选定需要添加的控件→单击“设置为默认值”按钮→单击“确定”按钮退出,如图 6.31所示。

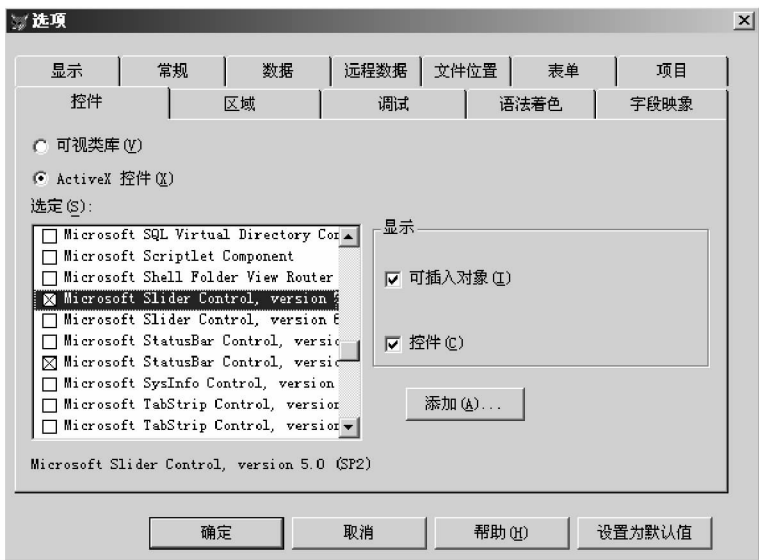


图 6.31 添加 ActiveX 控件

2. 在表单工具栏中添加 ActiveX 控件。

单击表单控件工具栏“查看类”按钮→“ActiveX 控件”。表单控件工具栏替换为 ActiveX 控件工具栏,上面显示选中的控件按钮,如图 6.32 所示。



图 6.32 表单控件工具栏

若要删除某控件按钮,可以打开“工具|选项”的对话框,清除“控件”选项卡列表中的复选框即可。

6.4.2 ActiveX 绑定控件

前面介绍过,通用字段可以保存其他应用程序的数据,如文本、图片、声音和视频等。如果希望在表单上显示通用字段的数据,可以使用 ActiveX 绑定控件。并通过 ActiveX 绑定控件的主要属性 ControlSource 属性,指定 ActiveX 绑定控件与对象建立联系的数据源。

例 6.21 修改表单 Employee.scx,使之显示 Employee 表中照片字段的数据,如图 6.33 所示。

设计步骤:

- 1. 打开表单 Employee.scx,创建 ActiveX 绑定控件,调整控件大小;
- 2. 设置 ActiveX 绑定控件的 ControlSource 属性值为 Employee.照片。



图 6.33 运行效果

6.4.3 超级链接

超级链接控件在运行时同计时器控件一样是不可见控件,可用于在表单上创建超级链接对象。超级链接对象含有一个 NavigateTo 方法程序,允许用户指定 URL,执行该方法程序时 VFP 会启动因特网浏览器,并根据指定的网址显示网页。

例 6.22 创建表单 Form7. scx(如图 6. 34),通过单击命令按钮打开新浪网页。



图 6.34 运行效果

设计步骤:

- 1. 新建表单 Form7. scx,创建一个 Command1 命令按钮和一个 HyperLink1 超级链接按钮,并调整布局如图;
- 2. 设置命令按钮 Caption 属性值为“新浪”;
- 3. 设置 Command1 命令按钮的 Click 事件代码:
THIS.Parent. HyperLink1. NavigateTo("www. sina. com. cn")

6.4.4 设计实例

例 6.23 创建表单 EmpConInfor. scx(如图 6. 35(a)),表单上有 2 个命令按钮,单击分别打开如图 6. 35(b),(c)的表单 EmpInfor. scx 和 ConInfor. scx。

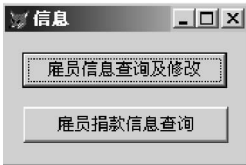


图 6.35 运行效果 (a)



图 6.35 运行效果 (b)



图 6.35 运行效果 (c)

功能简介如下：

1. “雇员信息查询及修改”表单运行时，自动在列表框中显示所有记录的身份号，当用户选中某身份号时，右边将显示该雇员的相关信息；同时可以直接修改雇员信息。单击“还原”按钮，撤销对雇员信息修改；单击“保存”按钮，将修改内容保存至数据库表中；单击“关闭”按钮，释放表单；

2. “雇员捐款信息查询”表单运行时，首先在表格中显示 Contribution 表中所有记录内容。用户可以构造查询条件完成特定查询，在“字段”组合框内可以选择 Contribution 表中的字段；在“操作符”组合框内可以选择“大于”、“大于等于”等六种关系运算。

设计步骤：

1. 新建表单 EmpConInfor. scx，创建 2 个命令按钮，调整布局如图 6. 35(a)，属性设置如表 6. 24：

表 6. 24 “信息”表单中控件主要属性设置

对 象	属 性	属 性 值	说 明
Form1	Caption	信息	设置标题栏显示文本
Form1	AutoCenter	. T.	设置表单启动时自动居中
Command1	Caption	雇员信息查询及修改	设置文本第一个命令按钮的显示
Command2	Caption	雇员捐款信息查询	设置文本第二个命令按钮的显示

Command1 的 Click 代码：

```
Do Form EmpInfor. scx
```

Command2 的 Click 代码：

```
Do Form ConInfor. scx
```

2. 新建表单 EmpInfor. scx，在数据环境中添加“Employee. dbf”，创建各种控件、设置相关属性、调整布局如图 6. 35(b)；

分析：由于在这个表单修改记录后，是通过“保存”或“还原”按钮来保存或还原 Employee 表中的当前记录，所以该表单中的各控件不能直接与 Employee 表中的相应字段进行绑定，所以各控件在显示相应字段的值是通过赋值语句进行的，当要保存修改结果时是通过 Replace 替换命令来实现的。但为了简化本例，让大家更容易理解，本例中的照片字段还是与控件进行了直接绑定。

(1) 表单的 Init 代码如下：

```
This. AutoCenter=. T.
```

```
This. Caption="雇员信息查询及修改"
```

* 将 Employee 表中的所有记录的身份号，依次添加到 List1 列表框中

```
Scan
```

```
    This. List1. AddItem(Employee. 身份号)
```

```
Endscan
```

* 将 Combo1 组合框与数组 Bmtmp 绑定

* 数组 Bmtmp 保存 Employee 表中各记录的部门代码

```
Dime Bmtmp(RecCount())
```

Select Distinct 部门 From Employee Into Array Bmtmp Order By 部门

ThisForm.Combo1.RowSource="Bmtmp"

ThisForm.Combo1.RowSourcetype=5

ThisForm.OleBoundControl1.ControlSource="Employee. 照片"

(2) 用以显示“身份号”的值的列表框 List1 的 InterActiveChange 代码如下:

* 当改列表框的值时,首先将 Employee 表的记录指针定位到相应的记录

* 然后将 Employee 表的各字段值分赋值给表单中各相应的控件,并刷新表单

Locate For 身份号 == This.Value

ThisForm.Text1.Value = 姓名

ThisForm.Text2.Value = 民族

ThisForm.Combo1.Value = 部门

ThisForm.Text3.Value = 工资

ThisForm.Text4.Value = 工作时间

ThisForm.Text5.Value = 电话

ThisForm.Spinner1.Value = 年龄

ThisForm.Optiongroup1.Value = 性别

ThisForm.Check1.Value = 婚否

ThisForm.Edit1.Value = 简历

ThisForm.Refresh

(3) “保存”按钮的 Click 代码如下:

* 用表单中控件的当前值去替换 Employee 表的当前记录值,保存当前修改内容

Replace 姓名 With ThisForm.Text1.Value

Replace 民族 With ThisForm.Text2.Value

Replace 部门 With ThisForm.Combo1.Value

Replace 工资 With ThisForm.Text3.Value

Replace 工作时间 With ThisForm.Text4.Value

Replace 电话 With ThisForm.Text5.Value

Replace 年龄 With ThisForm.Spinner1.Value

Replace 性别 With ThisForm.Optiongroup1.Value

Replace 婚否 With ThisForm.Check1.Value

Replace 简历 With ThisForm.Edit1.Value

(4) “还原”按钮的 Click 代码如下:

* 调用 List1 列表框的 InterActiveChange 事件

* 在表单中恢复 Employee 表当前记录值

ThisForm.List1.InterActiveChange

(5) “删除”按钮的 Click 代码如下:

* 这里仅仅只执行逻辑删除

* 如果要完成物理删除,可以在关闭表单时执行 PACK 命令

```

Delete For 身份号 = ThisForm. List1. Value
ThisForm. List1. RemoveItem(ThisForm. List1. Listindex)
If ThisForm. List1. Listcount = 0
    This. Enabled = . F.
Endif
ThisForm. List1. InterActiveChange

```

(6) “关闭”按钮的 Click 代码如下：

```
ThisForm. Release
```

3. 新建表单 ConInfor. scx, 在数据环境中添加“Contribution. dbf”, 创建各种控件、设置相关属性、调整布局如图 6.35(c)；

(1) 表单的 Init 代码如下：

```

This. AutoCenter = . T.
This. Caption = "雇员捐款信息查询"
This. BorderStyle = 2           &&. 设置表单运行时, 不允许调整大小
* 设置 Combo1 组合框的数据源类型为 Contribution 表的结构
This. Combo1. RowSource = "Contribution"
This. Combo1. RowSourceType = 8
* 设置 Combo2 组合框的数据源类型为值类型
This. Combo2. RowSource = "大于, 大于等于, 小于, 小于等于, 等于, 不等于"
This. Combo2. RowSourceType = 1
* 设置 Grid1 的数据源类型为 SQL 查询
This. Grid1. RecordSourceType = 4

```

(2) Combo1 的 InteractiveChange 代码如下：

```

If ThisForm. Combo1. Value = "捐款日期"
    ThisForm. Text1. Inputmask = "9999/99/99"
Else
    ThisForm. Text1. Inputmask = ""
Endif

```

(3) “查询”按钮的 Click 代码如下：

```

Set Exact On
F = ThisForm. Combo1. Value
V = Alltrim(ThisForm. Text1. Value)
* 考虑不同类型数据的处理
If ThisForm. Combo1. Value = "身份号"
    V = "[" + V + "]"
Else
    If ThisForm. Combo1. Value = "捐款日期"
        V = "{" + V + "}"
    Endif
Endif

```



```
        Endif
    Endif
Do Case
    Case ThisForm.Combo2.Value="大于"
        Op=">"
    Case ThisForm.Combo2.Value="大于等于"
        Op=">="
    Case ThisForm.Combo2.Value="小于"
        Op="<"
    Case ThisForm.Combo2.Value="小于等于"
        Op="<="
    Case ThisForm.Combo2.Value="等于"
        Op="="
    Case ThisForm.Combo2.Value="不等于"
        Op="<>"
EndCase
S="Select * From Contribution Where "+F+Op+V+" Into Cursor Mycur"
ThisForm.Grid1.Recordsource=S
Set Exact Off
```

(4) “关闭”按钮的 Click 代码如下：

```
ThisForm.Release
```

本例中制作了三张表单,用到了本章介绍的大多数控件。有时候,同样的功能可以有不同的实现方法,只要多思考、多练习,就能掌握 VFP 表单控件的使用。应用程序往往由多个窗口组成,即多表单的界面,这部分内容将在第七章中详细介绍。

习 题

一、选择题

1. 新创建的命令按钮默认标题为“Command1”,将其改为“关闭”,该设置它的()。
A. Name 属性 B. Caption 属性 C. Closable 属性 D. AlwaysTop 属性
2. 在表单上创建命令按钮 CmdClose,为实现当用户单击此按钮时能关闭表单的功能,应把语句 ThisForm.Release 写入 CmdClose 对象的()。
A. Caption 属性 B. Name 属性 C. Click 事件 D. Refresh 方法
3. 关于表单数据环境中的表与表单之间的关系正确的描述是()。
A. 当表单运行时,自动打开表单数据环境中的表
B. 当表单关闭时,不能自动关闭表单数据环境中的表
C. 当表单运行时,表单数据环境中的表处于只读状态,只能显示不能修改
D. 以上说法都不对
4. 在默认状态下,把表单数据环境中的表的一个字符型字段拖动到表单上,会自动

- 在表单上产生()。
- A. 一个表格容器对象 B. 一个标签控件对象和一个文本框控件对象
C. 一个列表框对象 D. 一个命令按钮对象
5. 将正在运行的 VFP 表单从内存中释放的正确语句是()。
- A. ThisForm. Close B. ThisForm. Clear
C. ThisForm. Release D. ThisForm. Refresh
6. 在表单设计阶段,下面说法不正确的是()。
- A. 拖动表单上的对象,可以改变该对象的位置
B. 拖动表单上对象的边框,可以改变该对象的大小
C. 通过设置表单上对象的属性,可以改变对象的大小和位置
D. 表单上的对象一旦建立,其位置和大小均不能改变
7. 下述描述中不正确的是()。
- A. 表单是容器类控件 B. 表格是容器类控件
C. 选项组是容器类控件 D. 命令按钮是容器类控件
8. 在 Visual FoxPro 中,为了将表单从内存中释放(清除),可将表单中退出命令按钮的 Click 事件代码设置为()。
- A. ThisForm. Refresh B. ThisForm. Delete
C. ThisForm. Hide D. ThisForm. Release
9. 在当前表单的 Label1 控件中显示系统时间的语句是()。
- A. ThisForm. Label1. Caption=Time()
B. ThisForm. Label1. Value=Time()
C. ThisForm. Label1. Text=Time()
D. ThisForm. Label1. Control=Time()
10. 如果文本框的 InputMask 属性值是 #999999,允许在文本框中输入的是()。
- A. +11111 B. Abc11 C. \$ 11111 D. ABCDEE

二、填空题

1. 为使表单运行时在 VFP 的主窗口居中显示,应设置表单的 AutoCenter 属性值为_____。
2. 在表单中确定控件是否可见的属性是_____。
3. 让控件获得焦点,使其成为活动对象的方法是_____。

第 4, 5, 6 题使用如下图所示的表单。表单名为 Form1, 表单中有两个命令按钮(Command1 和 Command2)、两个标签、两个文本框(Text1 和 Text2)。



4. 运行表单时,使表单的标题显示“登录窗口”,则可以在 Form1 的 Load 事件中加入语句_____。

5. 运行表单时,向 Text2 中输入字符,回显字符显示的是“*”号,则可以在 Form1 的 Init 事件中加入语句_____。

6. 假设用户名和口令存储在自由表“口令表”中,当用户输入用户名和口令并单击“登录”按钮时,若用户名输入错误,则提示“用户名错误”;若用户名输入正确,而口令输入错误,则提示“口令错误”。若命令按钮“登录”的 Click 事件中的代码如下,将程序补充完整。

```
USE 口令表
GO TOP
flag = 0
DO WHILE .not. EOF()
IF Alltrim(用户名) = Alltrim(ThisForm.Text1.Value)
    If Alltrim(口令) = Alltrim(ThisForm.Text2.Value)
        WAIT "欢迎使用" WINDOW TIMEOUT 2
    ELSE
        WAIT "口令错误" WINDOW TIMEOUT 2
    ENDIF
    flag = 1
ENDIF
SKIP
ENDDO
IF _____
    WAIT "用户名错误" WINDOW TIMEOUT 2
ENDIF
```

7. 要改变表单上表格对象中当前显示的列数,应该设置表格的_____属性。

三、应用题

以下第 1 题到第 5 题使用 Student 数据库。其中包含三张表: Student. dbf, Collegedepartment. dbf 和 Reportcard. dbf。

1. 制作如下图所示的“学生信息管理系统”主界面表单,上有两个标签、一个容器、一个图像和一个命令按钮组。当单击命令按钮时,将打开对应的表单。



2. 制作如下图所示的“系部信息”表单,上有一个标签、一个表格和一个命令按钮。



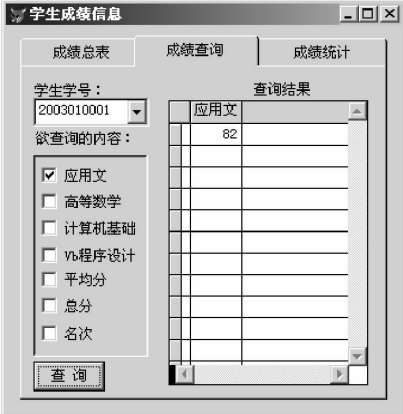
3. 制作如下图所示的“学生基本信息”表单,上有一个编辑框、一个复选框、一个命令按钮组和若干标签、文本框。能实现学生记录的浏览、增加、修改和删除。单击“修改”按钮后,可编辑记录;原“修改”和“删除”按钮变成“保存”和“还原”,其他按钮不可用。

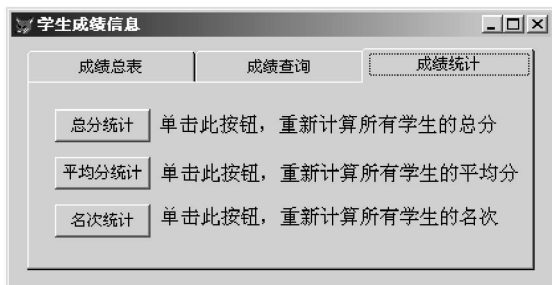


4. 制作如下图所示的“学生成绩信息”表单,能完成学生成绩信息的查询和统计功能。

要求:

- (1) 在“成绩总表”选项卡上有一表格,运行时显示 Reportcard 表中所有记录;
- (2) 在“成绩查询”选项卡上,有一个组合框、一个命令按钮、一个容器、一个表格和若干标签和复选框。当选择“学生学号”和“欲查询内容”后单击查询,在“查询结果”表格中显示指定信息;
- (3) 在“成绩统计”上有三个命令按钮,单击时完成相应统计功能。





5. 制作一个“关于”表单,功能:显示程序版本信息、制作者信息,并能自下而上循环滚动显示。

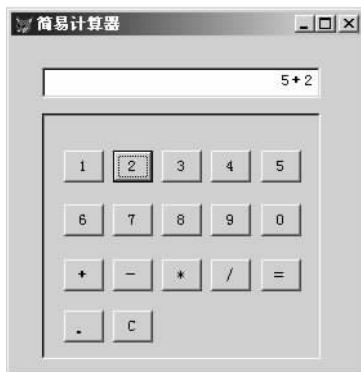
6. 设计一个简易计算器,如右图所示。

要求:

(1) 单击 C 按钮:能清除文本框中的内容;

(2) 单击数字按钮、小数点按钮和运算符按钮:能将相应字符添加到文本框字符串末尾。

(3) 单击 = 按钮:若文本框内容为合法表达式,则计算该表达式的值,显示于文本框内;否则,显示一个消息框提示“表达式不合法”,并将文本框内容清除,等待重新输入。



7. 设计表单如下图所示。要求:用户输入阿拉伯数字,单击“转换”按钮,将其转换为对应的中文大写格式;单击“退出”按钮,释放表单。



8. 设计表单如下图所示。表单上有一图像控件,每 3 秒钟随机显示一个图像(共有 5 个图像文件)。单击“退出”按钮释放表单。





第 7 章

表单高级设计

7.1 多表单应用程序

应用程序通常包含多个窗口,而一个表单只显示为一个窗口。前一章的例 6.23 实际上已经涉及到多表单的操作问题,这里,将就多表单应用程序的界面、相关属性及方法进行探讨。

7.1.1 应用程序界面

7.1.1.1 单文档界面与多文档界面

在 VFP 创建的应用程序中,用户界面分为两类:单文档界面 (Single-Document Interface,简称为 SDI)和多文档界面 (Multiple-Document Interface,简称为 MDI)。两者的区别如下表:

比较项目	SDI	MDI
最多能显示的文档窗口数	1	1 个以上
能同时打开文档数	1	1 个以上
举例	记事本	Word

至于采用何种界面,就根据具体情况而定。比如说音乐播放器就应该是 SDI,因为一般不可能让两个歌曲同时播放。

7.1.1.2 SDI 与 MDI 的实现

VFP 允许创建顶层表单和子表单,以支持 SDI 与 MDI 两类界面。

7.1.1.2.1 顶层表单和子表单

1. 设置顶层表单或子表单。

在属性窗口中对表单的 ShowWindow 属性直接进行设置,来指定表单为顶层表单或子表单,其属性值可分别取 0,1,2 三个值,分别表示:本表单作为 VFP 主窗口的子表单、本表单作为顶层表单的子表单、本表单作为顶层表单显示在桌面上。

表单的 ShowWindow 属性运行时为只读。

2. 顶层表单。

顶层表单没有父表单,它如同其他 Windows 应用程序一样直接显示在 Windows 桌面上,并在 Windows 任务栏中显示图标。顶层表单适用于创建一个 SDI 应用程序,或用作 MDI 应用程序中的父表单。

3. 子表单。

子表单用于创建 MDI 应用程序的文档窗口,它又分为非浮动表单和浮动表单两种。程序运行时,不能被移动到父表单边界之外的表单,称为非浮动表单。非浮动表单最小化时将显示在父表单底部。程序运行时,能被移动到桌面的任何位置的表单,称为浮动表单。浮动表单不能置于父窗口之后,最小化时将显示在桌面底部。无论是浮动表单还是非浮动表单,当父表单最小化时它们也会同时最小化。

(1) 设置表单的 Desktop 属性:

Desktop 属性为 . T. 时,表单为浮动表单;Desktop 属性为 . F. (默认值)时,表单为非浮动表单。

(2) 设置表单的 MDIform 属性:

MDIform 属性设置为 . T. 时,在表单运行过程中,当让表单最大化后,表单与父表单组合成一体,即包含在父表单中,并共享父表单的标题栏、标题、菜单以及工具栏。

MDIform 属性设置为 . F. 时,在表单运行过程中,当让子表单最大化后,表单为一独立窗口,即保留它本身的标题和标题栏,并占据父表单的全部用户区域。

(3) 子表单的调用:

子表单的调用命令格式同表调用命令格式,可直接在顶层表单的某个事件代码中写入子表单的调用命令:Do Form <子表单名>,来调用子表单。但需要注意的是:不要在顶层表单的 Init 事件中调用子表单,因为此时顶层表单本身尚未激活,所以被调用打开的子表单不是在顶层表单中打开的。

7.1.1.2.2 表单的显示与隐藏

通过设置表单的 Visible 属性,可以使表单隐藏或显示。表单的 Visible 属性为 . F. 时,表单为隐藏不可视;Visible 属性为 . T. (默认值)时,表单为可视。

使用 Hide 和 Show 方法程序,也可使表单隐藏或显示。如 ThisForm.Hide 与 ThisForm.Visible = . F. 效果相同。但 Show 方法程序通常用于表单集中,如 ThisFormSet.Form2.Show 与 ThisFormSet.Form2.Visible = . T. 效果相同。

Do Form 命令也能显示表单,该命令的功能是将命令指定的表单装入内存运行。Visible 属性、Show 和 Hide 方法程序则是在表单已经装入内存的前提下而使用的。

7.1.1.2.3 MDI 应用程序的运行

由以上介绍可知,VFP 的 MDI 应用程序由顶层表单和子表单组成,顶层表单作为 MDI 应用程序中的父表单,子表单则为 MDI 应用程序的文档窗口。所以,MDI 应用程序的运行是从父表单(顶层表单)开始的。

7.1.1.2.4 多表单程序的调试

1. 用工具菜单的调试器命令打开调试器来调试程序。
2. 程序运行时若发现不正常情况,可用程序菜单的取消命令撤销程序的运行。
3. 若程序运行已中断,但程序中开启的窗口尚未关闭,可在命令窗口键入 Clear All

命令,从内存中释放所有由用户定义的窗口。

7.1.1.3 在顶层表单中添加菜单

在表单中添加菜单必要条件:

1. 在菜单设计时,通过系统菜单“显示/常规选项”命令,打开“常规选项”对话框,并在该对话框中,选中“顶层表单”选项,将菜单设定为用于顶层表单;
2. 要添加菜单的表单必须是顶层表单,而且应在该表单的 Init 事件中设置一条调用菜单程序的命令,格式如下:

Do <菜单程序> With <参数>

参数说明:

- (1) <菜单程序>是指 MPR 文件。
- (2) <参数>用来引用本表单对象,通常用关键字 This 来表示。为使菜单程序能感知表单,此参数不可省略。
- (3) 菜单程序能自行接收和使用参数。

例 7.1 利用例 6.23 创建的三个表单 EmpConInfor. scx, EmpInfor. scx 和 ConInfor. scx,来完成进行顶层表单和子表单的练习,要求如下:

1. 设置表单 EmpConInfor. scx 为顶层表单,并在该顶层表单中增加一个系统菜单;
2. 设置表单 EmpInfor. scx 和 ConInfor. scx 为子表单,其中表单 EmpInfor. scx 为浮动表单,表单 ConInfor. scx 为非浮动表单;并设置两个表单的 MDIform 属性分别为. T. 和. F. 。

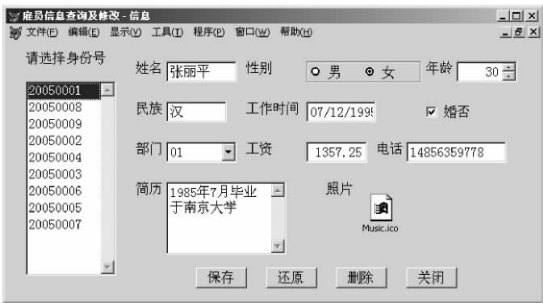


图 7.1 表单 EmpInfor 运行时最大化效果



图 7.2 表单 ConInfor 运行时最大化效果

设计步骤:

1. 设计系统菜单程序 SysMenu1. Mpr:
 - (1) 分析:由于本例以前,我们没有学习过菜单设计,菜单设计将在下章介绍,所以只要求增加一个设计过程比较简单的系统菜单,然后将系统菜单添加到顶层表单 EmpConInfor. scx 中;
 - (2) 设计系统菜单程序 SysMenu1. Mpr:
 - ①单击系统菜单“文件/新建”,并在弹出的“新建”对话框中选择“菜单”并单击“新建”文件按钮;

②在弹出的“新建菜单”对话框中单击菜单“按钮”，弹出如图 7.3 所示的空白菜单设计器窗口；



图 7.3 菜单设计器窗口

③单击系统菜单“菜单/快速菜单”，VFP 系统将自动产生一个与系统菜单类似的菜单，如图 7.3 所示；

④单击系统菜单“显示/常规选项”，在弹出的“常规选项”对话框中选中“顶层表单”后，单击“确定”按钮，如图 7.4 所示；



图 7.4 常规选项对话框

⑤单击系统菜单“菜单/生成”，将弹出一个保存对话框，将菜单保存为 SysMenu1.mnx，单击确定后，弹出“生成菜单”对话框，如图 7.5 所示，单击该对话框的“生成”按钮，生成 SysMenu1.mpr 菜单程序文件，完成系统菜单的设计。

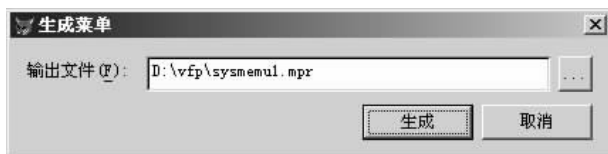


图 7.5 生成菜单对话框

2. 设置表单 EmpConInfor. scx 为顶层表单,并在该顶层表单中增加一个系统菜单:

(1) 在表单设计器中打开表单 EmpConInfor. scx,直接在属性窗口中修改以下属性,调整大小,设置本表单为顶层表单,其中 ShowWindow 属性只能在设计中可以修改;

表 7.1 表单 EmpConInfor. scx 修改的属性

对 象	属 性	属 性 值	说 明
Form1	ShowWindow	2	作为顶层表单
	Width	700	作为顶层表单,要容纳其他表单,所以窗体设计要大一些。
	Height	500	

(2) 在表单 EmpConInfor. scx 的 Init 事件代码中添加如下代码:

* 让两个命令按钮自动水平居中

```
ThisForm.Command1.Left=(ThisForm.Width-ThisForm.Command1.Width)/2  
ThisForm.Command2.Left=(ThisForm.Width-ThisForm.Command2.Width)/2
```

* 在顶层表单 EmpConInfor. scx 中添加菜单

```
Do SysMenu1.mpr With This
```

3. 设置表单 EmpInfor. scx 为浮动表单,并设置表单的 MDIform 属性分别为. T. :

在表单设计器中打开表单 EmpInfor. scx,直接在属性窗口中修改以下属性,其中 ShowWindow,DeskTop 属性只能在设计中可以修改。

表 7.2 表单 EmpInfor. scx 修改的属性

对 象	属 性	属 性 值	说 明
Form1	ShowWindow	1	在顶层表单中
	DeskTop	. T.	设置为浮动表单
	MIDForm	. T.	设置为多文档窗口

4. 设置表单 ConInfor. scx 为非浮动表单;并设置表单的 MDIform 属性分别为. F. :

在表单设计器中打开表单 ConInfor. scx,直接在属性窗口中修改以下属性,其中 ShowWindow,DeskTop 属性只能在设计中可以修改。

表 7.3 表单 ConInfor. scx 修改的属性

对 象	属 性	属 性 值	说 明
Form1	ShowWindow	1	在顶层表单中
	DeskTop	. F.	设置为非浮动表单
	MIDForm	. F.	设置为多文档窗口

如果要练习表单的隐藏属性,当表单被隐藏后,将无法使用表单的属性或方法程序让表单重新显示,所以需借助计时器来实现一定时间后让表单自动弹出。具体设计步骤如下:

1. 在表单上添加一个 Timer1 计时器;

选定方法只有一种,即通过属性窗口的对象列表来选定。

7.1.2.2.2 添加表单

在表单设计器中打开表单集的所在的表单文件,通过系统菜单中“表单/添加新表”命令添加表单。但此时添加到表单集中的表单是新表单,并不能将已经存在的表单添加到表单集中。

7.1.2.2.3 移去表单

若要从表单集中移去表单可先选定需要移去表单窗口,或在属性窗口的对象列表中选定要移去的表草,然后通过系统菜单中“表单/移除表单”命令移去选定的表单。

对于表单集中表单的具体设计方法同前。

例 7.2 利用表单集,设计运行结果如图 7.6 表所示的单集,在表单集的三个表单中分别显示三个表 Department,Employee,Contribution 内容,其中,三个表的关系为数据库已经定义过的永久关系。

操作步骤:

Figure 7.6 displays three overlapping windows from a Visual FoxPro form set, each showing a table of data.

职员基本信息 (Employee Basic Information)

职员身份号	姓名	性别	年龄	民族	工作
20050001	张丽平	女	30	汉	07/
20050008	杨剑雄	男	20	汉	08/

捐款情况 (Donation Situation)

身份号	捐款日期	捐款金额
20050001	03/21/05	200.00
20050001	11/08/05	50.00
20050001	04/06/06	400.00

部门代码 (Department Code)

部门代码	部门名称
01	办公室
11	一车间
12	二车间
13	三车间

At the bottom of the '部门代码' window, there are three buttons: '隐藏职员信息情况表' (Hide Employee Information Table), '隐藏职员捐款情况' (Hide Employee Donation Situation), and '退出' (Exit).

图 7.6 运行结果

1. 在命令窗口中输入命令:Modif Form Formset,执行该命令,创建名为 Formset 的表单。
2. 单击系统菜单“表单/创建表单集”命令,将表单 Formset 转化为表单集,此时该表单集中原有的表单的 Name 属性为 Form1。
3. 通过系统菜单“表单/添加新表单”命令,向表单集中添加两个表单 Form2 和 Form3。
4. 设计表单集中的表单 Form1。
 - (1) 向该表单添加一个 Label1 标签、一个 Grid1 表单和三个命令按钮控件,并设置该表单及相应控件的属性,如表 7.4 所示,在该表中省略了关于字体等属性;

表 7.4 表单集 Formset 中表单 Form1 及其相关控件的属性

对 象	属 性	属 性 值	说 明
ThisForm	Form1	Caption	部门代码
ThisForm, Form1	Label1	Caption	部门代码
	Command1	Caption	显示职员信息情况表
	Command2	Caption	显示职员捐款情况
	Command3	Caption	退出

(2) Form1 的 Init 事件代码：

```
ThisForm. AutoCenter=. T.           &&. 运行时表单自动居中
* 指定与 Grid1 表格控件建立联系的数据源以别名方式打开
ThisForm. Grid1. RecordSourcetype=1.
ThisForm. Grid1. ColumnCount=-1      &&. 在表格控件中显示数据源的所有字段
* 指定与 Grid1 表格控件建立联系的数据源的别名
ThisForm. Grid1. RecordSource="Department"
* 运行时先隐藏表单集 Formset 中的 Form2 和 Form3 表单
ThisFormSet. Form2. Hide
ThisFormSet. Form3. Hide
```

(3) Form1 的 Command1 控件的 Click 事件代码：

```
* 让该命令按钮具有显示或隐藏 Form2 的功能，
* 在 Form2 显示时为隐藏功能，在 Form2 隐藏时为显示功能
If ThisForm. Command1. Caption="显示职员信息情况表"
    ThisForm. Command1. Caption="隐藏职员信息情况表"
    ThisFormSet. Form2. Show           &&. 显示 Form2
    * 设置表单显示的位置
    ThisFormSet. Form2. Top=0
    ThisFormSet. Form2. Left=0
Else
    IF ThisForm. Command1. Caption="隐藏职员信息情况表"
        ThisForm. Command1. Caption="显示职员信息情况表"
        ThisFormSet. Form2. Hide       &&. 显示 Form2
    EndIf
EndIf
```

(4) Form1 的 Command2 控件的 Click 事件代码：

```
If ThisForm. Command2. Caption="显示职员捐款情况"
    ThisForm. Command2. Caption="隐藏职员捐款情况"
    ThisFormSet. Form3. Visible=. T.   &&. 显示 Form3
    ThisFormSet. Form3. Top=0
```

```
ThisFormSet. Form3. Left = ThisFormSet. Form2. Width + 20
Else
    IF ThisForm. Command2. Caption = "隐藏职员捐款情况"
        ThisForm. Command2. Caption = "显示职员捐款情况"
        ThisFormSet. Form3. Visible = .F.    && 隐藏 Form3
    EndIf
EndIf
```

(5) Form1 的 Command3 控件的 Click 事件代码：

```
ThisFormSet. Release    && 从内存中释放表单集
```

(6) Form1 的 UnLoad 事件代码：

* Form1 表单为主表单, 当该表单关闭时, 应同时释放表单集

```
ThisFormSet. Release
```

5. 设计表单集中的表单 Form2。

(1) 向 Form2 表单中添加一个 Label1 标签和一个 Grid1 表格; 调整该表单中控件的相对位置如运行效果图 7.6 所示;

(2) Form2 的 Init 事件代码：

* 在标题栏不显示关闭按钮, 由于 Form1 表单对 Form2 表单的操作是显示和隐藏操作,

* 若保留标题栏的关闭按钮, 用户单击该按钮关闭, 并关闭该表单后,

* 在 Form1 表单中利用单击显示该表单的命令按钮时将会出错

```
ThisForm. ControlBox = .F.    && 在标题栏中不显示控件按钮
```

```
ThisForm. Caption = "职员基本信息"
```

```
ThisForm. Label1. Caption = "职员基本信息"
```

```
ThisForm. Grid1. RecordSourceType = 1
```

```
ThisForm. Grid1. ColumnCount = -1
```

```
ThisForm. Grid1. RecordSource = "Employee"
```

6. 设计表单集中的表单 Form3：

(1) 向 Form2 表单中添加一个 Label1 标签和一个 Grid1 表格; 调整该表单中控件的相对位置如运行效果图 7.6 所示;

(2) Form2 的 Init 事件代码：

```
ThisForm. Closable = .F.    && 在标题栏中不显示关闭按钮
```

```
ThisForm. Caption = "捐款"
```

```
ThisForm. Label1. Caption = "捐款情况"
```

```
ThisForm. Grid1. RecordSourceType = 1
```

```
ThisForm. Grid1. ColumnCount = -1
```

```
ThisForm. Grid1. RecordSource = "Contribution"
```

7.2 用户定义属性与方法程序

在 VFP 系统中,表单集或表单已经由系统定义了一些各有自己的属性、事件和方法程序,一般情况下通过设置这些属性值、编写事件代,以及调用相应的方法程序,就可以比较方便的完成一些设计工作。但是,VFP 还允许用户为表单或表单集定义用户自己的属性和方法程序。

用户定义的属性作法类似于变量,用户定义的方法程序用法相当于过程。用户定义属性或方法程序的作用范围是整个表单文件,其用法与系统定义的属性、方法程序完全相同。

由前面的学习我们已经了解到:创建并设计好一个单表单时,当保存该表单时,该表单文件(扩展名为:*.SCX)存储了该单表单的所有相关内容;而对于表单集而言,它同样也存储在一个表单文件(扩展名为:*.SCX)中,该表单文件存储了该表单集及该表单集中各个表单的所有相关内容。所以,在表单集中用户定义的属性与方法程序的作用范围覆盖该表单集中所有的表单;而对于在单表单中用户定义的属性与方法程序的作用范围而言,则仅仅只覆盖该单表单。

7.2.1 用户定义属性

用户定义的属性包括定义的属性名与属性值,其用法与变量名与变量值类似,所不同的是用户定义的属性的作用范围是整个表单文件,即在表单文件所包含的单表单或表单集中有效。用户定义的属性可分为变量属性和数组属性两种。

7.2.1.1 变量属性与数组属性

7.2.1.1.1 变量属性

例 7.3 创建一个 Calculation 表单集,在第一个表单 Form1 中输入两个数,将计算结果在第二个表单 Form2 中显示,运行结果如图 7.7 所示。

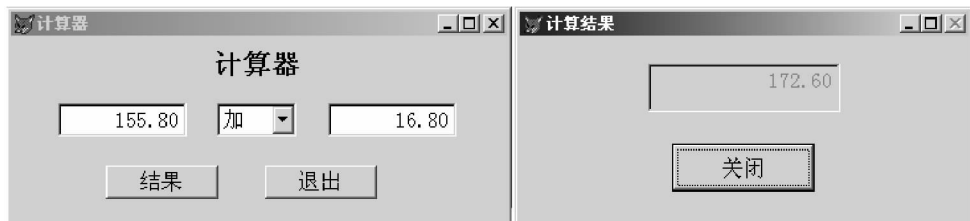


图 7.7 运行结果

设计步骤:

1. 在命令窗口中输入命令: Modif Form Calculation, 执行该命令, 创建名为 Calculation 的表单。
2. 单击系统菜单“表单/创建表单集”命令, 将表单 Calculation 转化为表单集, 此时该表单集中原有的表单的 Name 属性仍为 Form1。
3. 通过系统菜单“表单/添加新表单”命令, 向表单集中添加表单 Form2。

4. 创建表单集的变量属性 Numbe。

(1) 单击系统菜单“表单/添加属性”命令,弹出如图 7.8 所示的新建属性对话框;



图 7.8 新建属性对话框

(2) 在新建属性对话框中输入如图 7.6 所示的内容后,单击“添加”按钮后关闭些对话框;

(3) 查看添加的属性:在属性窗口中的下拉式组合框中选择 FormSet1 对象,在全部或其他选项卡中的最后一行,可以看到刚才添加的用户自定义的属性 Numbe;

5. 设计表单集中的表单 Form1。

(1) 按照图 7.7 在 Form1 表单中,添加一个 Label1 标签、两个文本框、一个 Combo1 组合框、两个命令按钮,并设置如表 7.5 所示的属性,关于各控件字体大小等属性设置请大家自行设置。

表 7.5 表单集 Calculation 中表单 Form1 及其相关控件的属性

对 象		属 性	属性值
ThisFormSet	Form1	Caption	计算器
ThisFormSet, Form1	Label1	Caption	计算器
	Command1	Caption	结果
	Command2	Caption	退出

(2) Calculation 表单集 Form1 表单的 Init 事件代码:

* 设置 Form1 表单的显示定位

ThisForm. AutoCenter=. T.

ThisForm. Top= ThisForm. Top+ ThisForm. Height/3

ThisForm. Caption="计算器"

* 设置两个文本框的初值为数个型

ThisForm. Text1. Value=0. 00

ThisForm. Text2. Value=0. 00

* 设置组合框的值类型,并设置值源和初值

ThisForm. Combo1. RowSourceType=1 &&. 指定数据值的类型为值类型

ThisForm. Combo1. RowSource="加,减,乘,除"

ThisForm. Combo1. Value="加"

ThisFormSet. Form2. Hide &&. 初始化运行时不显示 Form2 表单

(3) Calculation 表单集 Form1 表单 Command1 按钮的 Click 事件代码：

* 判断是运算何种运行，计算出运算结果，并将该结果赋值给用户定义的属性 Numbe

Do Case

Case Thisform. Combo1. Value="加"

ThisFormSet. Numbe= THisForm. Text1. Value+ THisForm. Text2. Value

Case Thisform. Combo1. Value="减"

ThisFormSet. Numbe= THisForm. Text1. Value- THisForm. Text2. Value

Case Thisform. Combo1. Value="乘"

ThisFormSet. Numbe= THisForm. Text1. Value * THisForm. Text2. Value

Case Thisform. Combo1. Value="除"

ThisFormSet. Numbe= THisForm. Text1. Value/THisForm. Text2. Value

EndCase

ThisFormSet. Form2. Show &&. 显示表单集的 Form2 表单

(4) Calculation 表单集 Form1 表单 Command2 按钮的 Click 事件代码：

ThisFormSet. Release &&. 释放表单集

(5) Calculation 表单集 Form1 表单的 UnLoad 事件代码：

Release ThisFormSet &&. 释放表单集，注意该释放方式与上面释放方式的不同

6. 设计表单集中的表单 Form2。

(1) 按照图 7.7 在 Form2 表单中，添加一个 Text1 文本和一个 Command1 命令按钮；并设置如表 7.6 所示的属性。

表 7.6 表单集 Calculation 中表单 Form2 及其相关控件的属性

对 象		属 性	说 明
ThisFormSet	Form2	Caption	计算结果
ThisFormSet. Form2	Command1	Caption	关闭

(2) Calculation 表单集 Form2 表单的 Init 事件代码：

* 设置 Form2 表单的显示定位

ThisForm. AutoCenter=. T.

ThisForm. Top= ThisFormSet. Form1. Top- ThisForm. Height * 8/9

ThisForm. Closable=. F. &&. 设置该表单的标题栏不含关闭按钮

ThisForm. Text1. ReadOnly=. T. &&. 设置文本框为只读

ThisForm. Text1. Value=0.00 &&. 设置文本框的初值为数值型

(3) Calculation 表单集 Form2 表单 Text1 文本框的 GotFocus 事件代码：

ThisForm. Text1. Value= ThisFormSet. Numbe &&. 访问用户定义属性 Numbe

说明：由于该表单的 Text1 文本框已经被设置为默认获得焦点的控件，所以这段代码的作用是当表单显示时让表单的文本框获得计算结果。当然我们也可以不设置这段事件代码，而在 Calculation 表单集 Form1 表单 Command1 按钮的 Click 事件代码中的 ThisFormSet. Form2. Show 代码行前添加这样一条代码：ThisFormSet. Form2. Text1.

Value=ThisFormSet. Numbe,也能获得同样的运行效果。

(4) Calculation 表单集 Form2 表单 Command2 按钮的 Click 事件代码:

```
ThisForm. Hide                                && 隐藏表单
```

需要说明的几个问题:

1. 创建变量属性。

(1) 在打开表单设计器的前提下,单击系统菜单“表单/新建属性”命令,弹出新建属性对话框;

(2) 在弹出的新建属性对话框中的名称文本框中输入用户定义属性的名称,在“说明”编辑框中输入需要显示在属性窗口底部的属性说明,该说明可以省略;

(3) 单击“添加”按钮,添加用户刚才定义的属性,此时,用户还可以接着定义下一个属性,属性定义完成后,单击“关闭”按钮关闭该对话框;

(4) 用户定义的属性可在属性窗口的“全部”或“其他”选项卡中查到。

2. 变量属性的编辑。

(1) 单击系统菜单“表单的/编辑属性方法程序”命令,弹出“编辑属性/方法程序”对话框,如图 7.9 所示;



图 7.9 编辑属性/方法程序对话框

(2) 通过该对话框的“新建属性”或“新建方法程序”按钮也可以添加用户定义的属性或方法程序;

(3) 在该对话框中还可对用户定义的属性和方法程序进行修改或删除:

①用户定义的属性和方法程序的修改。

在该对话框的“属性/方法程序”文本框中可修改选定的用户定义的属性或方法程序的名称,在“说明”编辑框中可以修改用户定义的属性或方法程序的说明;

②用户定义的属性和方法程序的删除。

在左边的列表框中,选定要删除的属性或方法程序,单击“移去”按钮,并在随后弹出的确认框中单击“是”按钮,完成用户定义的属性或方法程序的删除。

3. 变量属性的引用格式。

在表单集中创建的变量属性,其引用基本格式为:ThisFormSet. 变量属性名。

对于单表单所创建的变量属性,其引用基本格式为:ThisForm. 变量属性名。

7.2.1.1.2 数组属性

数组属性的创建、编辑、引用格式以及其作用范围与变量属性相同,所不同的是,数组属性只有在运行时才能对其赋值等操作,在属性窗口中为只读。

例 7.4 在例 6.23 中将 EmpInfor. scx 表单的 Combo1 组合框与数组 Bmtmp 的绑定,修改为与数组属性的绑定。

设计步骤:

1. 在命令窗口输入命令:Modif Form Empinfor,在表单设计器中打开 EmpInfor. scx 表单;

2. 单击系统菜单“表单/添加属性”命令,弹出如图 7.6 所示的新建属性对话框;

3. 在新建属性对话框中的文本框中输入:Bmtmp(1),单击“添加”按钮后关闭些对话框,完成向表单添加数组属性;

4. 修改 mpInfor. scx 表单的 Init 原代码,修改结果如下:

* 说明:以下代码中的斜体加粗部分为修改和添加的内容。

This. AutoCenter=. T.

This. Caption="雇员信息查询及修改"

* 将 Employee 表中的所有记录的身份号,依次添加到 List1 列表框中

Scan

This. List1. AddItem(Employee. 身份号)

Endscan

* 将 Combo1 组合框与数组 Bmtmp 绑定

* 数组 Bmtmp 保存 Employee 表中各记录的部门代码

* Dime Bmtmp(RecCount()) 删除此行代码

Select Distinct 部门 From Employee Into Array ThisForm. Bmtmp Order By 部门

ThisForm. Combo1. RowSource=" ThisForm. Bmtmp"

ThisForm. Combo1. RowSourceType=5

ThisForm. OleBoundControll. ControlSource="Employee. 照片"

由上例可以看出,修改后的 Combo1 组合框是与用户定义属性数组属性绑定在一起,而原定义数组代码,所定义的数组 Bmtmp 作用范围为 Init 事件内,而修改后的数组属性 Bmtmp 的作用范围为表单。

7.2.1.2 多表单应用程序的有效参数

变量和用户定义的属性都有一定的作用范围,在多表单应用程序设计过程中,有时往往需要在表单间传递一定的参数,表单间的参数传递,我们可以使用以下三种方式实现:

1. 用 Public 来定义公共变量来进行参数传递。

用这种方式定义的公共变量对所有的表单文件都有效,但这种变量在程序运行结束或不再使用该变量时,需要用专用的内存变量释放命令释放。

2. 用户在表单或表单集中通过自定义属性来进行参数传递。

由于用户定义的属性的作用范围是表单文件内,而表单集中的所有表单都存放在一个共同的表单文件内,所以,表单集中的每一个表单都可以直接访问用户定义的属性;但对于单表单而言,用户的属性的作用范围仅仅只在该表单内有效。而且用户定义的属性会在表单关闭后,自动随表单从内存中清除。

3. 用 Do Form...With...To 命令进行表单间的参数传递:

命令格式:

Do Form <表单名> [With <参数表>] [To <变量名>]

功能:运行表单,并将参数传递给表单,或接受其返回值。

参数说明:

(1) With <参数表>

在父表中设置该命令代码,并通过 With <参数表>向子表单发送参数;子表单接收参数的语句只能在该表单的 Init 事件中设置,其设计语句同子程序调用的参数接收语句 Parameters。

(2) To <变量名>

①在父表中设置该命令代码,并通过 To <变量名>接收子表单的返回值,该子句中的<变量名>可以不用先定义。

②在子表单的 UnLoad 事件中设置一条命令:Return <表达式>,当子表单释放时将值返回给父表单的接收变量;

③将子表单的 WindowType 属性设置为 1,即模式。子表单在模式方式下运行时,只有在关闭子表单的前提下,用户才能返回到父表单。

例 7.5 设计一个表单,通过另一个表单输入记录号对该表单的内容进行更新。运行效果如图 7.10 所示。

图 7.10 运行效果

设计分析:

父表单 EmployeeRec 与子表单 EmployeeRecord 之间必然存在参数的调用问题,使用 Do Form... With... To 命令进行表单间的参数传递时,要考虑到以下几个问题的处理:

1. 子表单 EmployeeRecord 运行时,文本框中应显示当前记录的记录号,这样就必然

要将父表单的记录号传递给子表单。

2. 在子表单返回用户输入的记录号给父表单时,要判断用户输入的记录号是否是有效记录号,所以还要传递一个表的记录个数给子表单,用以比较判断。

3. 在子表单中用 Parameters 语句中的参数为局部变量,其作用范围仅仅为定义其的 Init 事件中,而子表单接收的参数,要在该表单的其他事件中使用,因此在子表单中定义了两个用户定义的变量属性,用以接收用 Parameters 定义的形参的值,并在表单内传递。

设计步骤:

1. 创建表单 EmployeeRec。

(1) 利用命令:Modif Form EmployeeRec,创建一个新表单;

(2) 单击系统菜单“表单/快速命令”,弹出表单生成器对话框;在该对话框中的“字段选取”选项卡中,选择 Employee 表单,并选取其全部字段,在“样式”选项卡中选择“浮雕式”,单击“确定”按钮,创建好如图 7.10 所示的职工信表的记录部分;

(3) 在表单中添加一个 Label1 标签和一个 Command1 命令按钮;

(4) 设置表单 EmployeeRec 的 Init 事件代码:

```
This. AutoCenter=. T.
```

```
This. Caption="职工信息表"
```

```
This. Label1. Caption="记录号:"+Alltrim(Str(Recno())) && 显示记录号提示
```

```
This. Label1. AutoSize=. T.
```

(5) 设置表单 EmployeeRec 控件 Command1 命令按钮的 Click 事件代码:

* 调用子表单,并将当前记录指针和当前表的记录数传递给子表单,

* 同时当子表单运行结束后,接收其返回的记录号值

```
Do Form EmployeeRecord With Recno(),RecCount() To N
```

```
Goto N && 记录指针定位到用户指定位置
```

```
ThisForm. Label1. Caption="记录号:"+Alltrim(Str(Recno()))
```

```
ThisForm. Label1. AutoSize=. T.
```

```
ThisForm. Refresh
```

2. 创建表单 EmployeeRecord。

(1) 利用命令:Modif Form EmployeeRecord,创建一个新表单;

(2) 在表单中添加一个 Text1 文本框和一个 Command1 命令按钮;创建好如图 7.10 所示的记录号表单;

(3) 单击系统菜单“表单/新属性”命令,给 EmployeeRecord 表单建立两个变量属性:M,Rec,用以在表单内传递参数;

(4) 设置表单 EmployeeRecord 的 Init 事件代码:

```
Parameters M,Rec && 接收父表单传递来的参数值 Recno(),RecCount()
```

* 将接收的值赋值给用户定义的变量属性

```
ThisForm. Rec=Rec
```

```
ThisForm. M=M
```

* 将当前记录指针的值赋值给 Text1 文本框,表单运行时,首先显示当前记录号

```
ThisForm.Text1.Value = ThisForm.M
```

```
ThisForm.AutoCenter = .T.
```

(5) 设置表单 EmployeeRecord 控件 Command1 命令按钮的 Click 事件代码:

* 用户输入的记录号,不能小于1,也不能大于记录总数

```
IF Thisform.Text1.Value > 0
```

```
    IF Thisform.Text1.Value > ThisForm.rec
```

```
        ThisForm.m = ThisForm.rec
```

```
    Else
```

```
        ThisForm.m = Thisform.Text1.Value
```

```
    EndIf
```

```
EndIf
```

```
ThisForm.release
```

(6) 设置表单 EmployeeRecord 的 UnLoad 事件代码:

```
Return ThisForm.m
```

7.2.2 用户定义方法程序

在设计应用程序时,某些代码可能需要经常重复使用,利用用户定义的方法程序保存这些代码,然后在其他方法或事件中去调用这些用户定义的方法程序。利用这种方法可以简化程序设计,提高代码的可重复性。所谓用户定义方法程序,就是用户为表单或表单集定义的过程。

7.2.2.1 方法程序创建

1. 方法程序的创建。

在表单或表单集中创建一个新方法程序,与创建一个用户定义属性的方法相似:单击系统菜单“表单/新建方法程序”命令,在弹出的“新建方法程序”对话框中键入方法程序的名称和说明(说明可以省略),单击“添加”按钮完成方法程序的添加,单击“关闭”按钮,退出该对话框。

用户定义的方法程序可在属性窗口的“全部”或“方法程序”选项卡中查到。

2. 方法程序名和说明的编辑。

参阅变量属性的编辑器。

7.2.2.2 方法程序过程代码的编辑

用户定义的方法程序的编辑方法与系统提供的事件代码的编辑方法相同。

7.2.2.3 用户定义方法程序的调用

调用方法程序即是运行该方法程序的过程代码。与用户定义属性相似,当用户定义的方法程序是在表单集中定义时,该用户定义方法程序在表单集中有效,其调用基本格式:

```
ThisFormSet.<方法程序名>
```

当用户定义的方法程序是在单表单时定义时,该方法程序仅对该单表单有效,其调用基本格式:

```
ThisForm.<方法程序名>
```

7.3 类

在第五章第二节面向对象的程序设计方法中,介绍了类的基本概念、子类的创建与应用等内容,本节将继续介绍子类的编辑、删除、为字段设置类和用户定义工具栏等内容。

7.3.1 编辑用户定义的子类

如果要对已经创建的子类设置或修改属性、编写事件代码或方法程序代码或创建新的用户定义属性和方法程序时,都需要将该子类在类设计器中打开,具体步骤如下:

7.3.1.1 在类设计器中打开要修改的子类的方法

1. 界面方式:

- (1) 通过系统工菜单“文件/打开”或单击“打开”按钮,弹出打开对话框;
- (2) 在文件类型列表框中选择可视类库(*.vcx)后,选择要打开的类库;
- (3) 在接着弹出的对话框中选择要打开的类;

2. 命令方式:

命令格式:

Modify Class <类名> OF <类库名>

功能:在类设计器中打开指定<类库名>中的<类名>指定的类。

7.3.1.2 在类设计器中修改的子类

在类设计器中修改子类的方法与在表单设计器中对表单进行设计和修改的方法相同。

7.3.1.3 删除类库中的一个类

1. 界面方式。

将该类库添加到项目管理器中,然后选定类库中的一个类,单击“移去”按钮,从类库中移去选定的类。

2. 命令方式。

命令格式:

Remove Class <类名> OF <类库名>

功能:删除指定<类库名>中的<类名>指定的类。

7.3.1.4 删除类库

直接在磁盘上删除类库文件。

7.3.2 子类的应用

在第五章第二节中已经介绍了通过表单控件工具的“查看类”按钮,将可视类库中的用户定义类添加到表单控件工具中,然后如基类控件一样使用用户定义类。除上述方法外,还可通过将项目管理器中的类用鼠标直接将选定的用户定义类拖入表单的方法,将用户定义类添加到表单中。

例 7.6 创建一个时间显示类控件,完成该控件类的创建后,将该控件添加到表单

中,不用修改运行表单,该控件显示当前时间。

1. 设计用户定义类——时间显示控件。

(1) 进入项目 managers 的类页面,单击“新建”按钮,弹出的“新建类”对话框,在“类名”文本框中输入类名:TimeShow,在“存储于”文本框中输入类文件名:TimeShow,在“派生于”列表框中,选择“Container”(容器),单击“确定”按钮,弹出“类设计器”窗口;

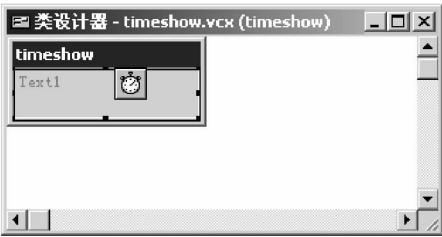


图 7.11 用户定义的子类 TimerShow 设计界面

(2) 在类 TimeShow 窗口中添加一个 Text1 文本框和一个 Timer1 计时器,如图 7.11 所示,各控件属性设置如表 7.7 所示;

表 7.7 类 TimeShow 相关控件的属性

对象	属性	属性值	说明
TimeShow	BorderWidth	0	指定边框的宽度为 0,
TimeShow. Text1	Top	0	
	Left	0	
	Height	25	
	Width	97	
	Specialeffect	1—平面	指定控件以平面方式显示
	FontSize	15	
	FontBold	. T. —真	
	DisabledForecolor	0,0,0	指定该控件失效时的前景色为黑色
	Alignment	2—中间	
	Enable	. F.	指定该控件不响应用户引发的事件
TimeShow. Timer1	Interval	1000	指定调用计时器事件的时间间隔

(3) 设置 TimeShow 的 Timer1 计时器的 Timer 事件代码:

This. Parent. Text1. Value=Time()

(4) 将容器调到与文本框大小一样;

(5) 保存设计好的类。

2. 在表单上应用这个用户定义的类。

方法一:建一个表单,将这个类直接拖放到表单上;

方法二:将这个类添加到表单控件工具栏中,再通过表单控件工具栏,向表单添加这个控件;

3. 再运行表单观察用户定义类的运行效果。

7.3.3 用户定义的子类注册

也许您觉得将用户定义的子类拖到表单上或添加到工具栏上不太方便,那么可将该

子类注册。注册后的子类,可以像 VFP 标准控件那样在表单设计器的工具栏上通过“查看类”按钮直接选择用户定义的类,方法如下:

1. 单击系统菜单“工具/选项”,打开“选项”对话框;
2. 在“选项”对话框中,选择“控件”选项卡,如图 7.12 所示;

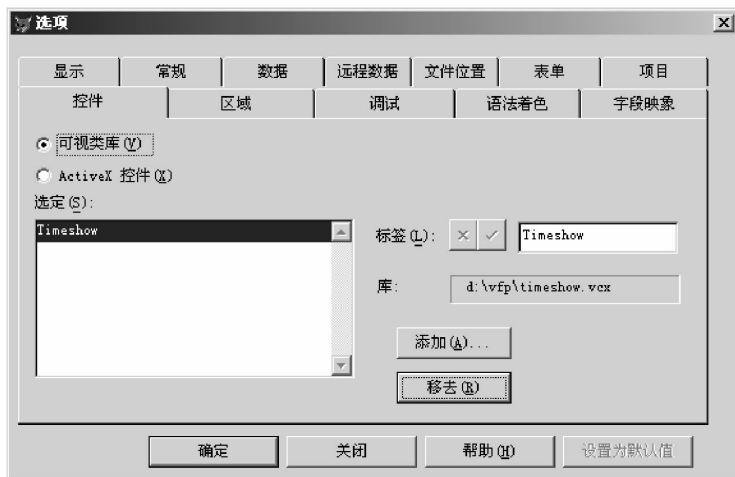


图 7.12 选项对话框

3. 选择“可视类库并选择添加”按钮;
4. 在弹出的“打开”对话框中,选择要注册的类库文件后,单击“打开”按钮,完成添加;
5. 在单击“选项”对话框的“确定”按钮前,若单击“设置为默认值”按钮,则 VFP 在以后启动时,以上的设置同样有效;否则只在本次 VFP 窗口未关闭前有效;
6. 在设计表单时,在表单控件工具栏上选择“查看类”按钮,就可以看到注册的子类。

7.3.4 将表单保存为类

在前面我们学习了表单集,通过表单集可以很方便的进行多个表单间的操作,但向已经存在的表单集中添加的表单,只能是新表单,对于已经设计好的表单不能直接向表单集中添加,这样将介绍如果将已经设计好的表单添加到表单集中。向表单集中添加已经设计好的表单的步骤:

首先,将已经设计好的表单另保存为一个类,类名可由用户自定义;

然后向表单或表单集添加刚才保存的类,添加的方式有:

1. 如果该类存在于其一个项目,可直接通过该项目的项目管理器,向表单或表单集添加;
2. 通过将该类注册或直接通过表单控件工具栏的“查看类”按钮的“添加”按钮向表单控件工具栏添加该用户定义的类控件,然后通过表单控件工具栏向表单或表单集添加。

例 7.7 将例 6.23 由多个单表单运行方式,修改设计为表单集运行方式。

设计步骤:

1. 将表单 Empinfor 和表单 Coninfor 另存为类。

- (1) 在表单设计器中打开表单 Empinfor;
- (2) 单击系统菜单“文件/另存为类”,弹出“另存为类”对话框,如图 7.13 所示;



图 7.13 “另存为类”对话框

(3) 在该对话框中的“保存”选项中,选择当前表单;在“类名”文本框中输入用户定义的类名如:Empl;在“文件”文本框中输入该类要保存的文件名如:Empl;在说明编辑框中输入:Employee;

- (4) 单击确定,将该表单保存为类;

(5) 将表单 Coninfor 保存为类的方法同上,只是将类名改为 Cont;而且这个类与刚才保存的类 Empl 保存在同一个类库文件 Empl 中。

说明:由于类设计器中没不存在数据环境的设置,所以当将表单保存为类后,该类并不保存原有的数据环境;

2. 将刚才保存的类库文件进行注册。

- (1) 单击系统菜单“工具/选项”,在弹出的“选项”对话框中单击“控件”标签;

(2) 单击“添加”按钮,将 Empl.vcx 类库文件添加到“选定”列表框中,单击“确定”按钮,完成类库文件的注册。

3. 向表单 Empconinfor,添加刚才定义的两个类。

- (1) 在表单设计器中,打开表单 Empconinfor;

(2) 单击表单控件工具栏“查看类”按钮,在弹出的选项中选择“Empl”,在表单控件工具栏中显示刚才定义的两个表单类图标;

(3) 单击 Empl 图标,再在表单 Empconinfor 上单击一次,由于该表单不是表单集,所以弹出一个询问对话框,询问“要添加一个表单,需要一个表单集对象。创建吗?”,单击“是”按钮,完成 Empl 表单类的添加,系统设置的表单名为:Empl12。实际上我们可以将其看作是表单 Empinfor 添加到表单集中,只是通过类来完成,并且系统自动将表单名改名为 Empl12;

(4) 单击 Cont 图标,再在表单 Empconinfor 上单击一次,完成 Cont 表单类的添加,由于这次添加是向表单集添加,所以不会弹出刚才的对话框,系统设置的表单名为 Cont12。

4. 设置数据环境:在数据环境中添加表 Employee.dbf 和表 Contribution.dbf。

5. 通过属性窗口将表单 Empl12 和表单 Cont12 的 Closable 属性设置为. F. 值,即不允许通过关闭按钮关闭表单。

6. 设置表单集中各表单的代码。

(1) 在表单 Empconinfor 的 Init 事件代码中添加如下代码:

ThisFormSet. Empl12. Hide

ThisFormSet. Cont12. Hide

(2) 设置表单 Empconinfor 控件 Command1 的 Click 代码:

ThisFormSet. Empl12. Show

(3) 设置表单 Empconinfor 控件 Command2 的 Click 代码:

ThisFormSet. Cont12. Show

(4) 设置表单 Empconinfor 的 UnLoad 代码:

ThisFormSet. Release

(5) 设置表单 Empl12 和表单 Cont12 关闭按钮的 Click 事件代码为:

ThisForm. Hide

对上例的补充说明:上例,在第三步就已经完成了将表单类添加到表单或表单集的过程;第四步是为了设置表单运行数据环境,由于表单 Empconinfor 在原设计中没有设置数据环境,创建为表单集后,同样也没有设置数据环境,所以,必须为表单集的运行设置数据环境;第五步是为了让表单 Empl12 和表单 Cont12 不能通过单击“关闭”按钮关闭,由于在主表单中调用这两个表单是通过 Show 方法来调用的,所以,当这两个表单在不能运行过程中,用户直接单击“关闭”按钮关闭该表单后,若再用 Show 方法调用将会出错;第六步是将表单间的调用,由原来的 Do Form 方式调用,修改为 Show 或 Hide 方式调用,并且在父表单关闭时,要求关闭表单集。

读者也许注意到,当打开表单 Empl12 和表单 Cont12 的一些事件,可以发现除前面我们修改的内容外,许多原来有代码的事件都为空,但表单集却运行正常,这是为什么呢?

在第五章第二节类的基本概念中我们已经介绍了类的继承性问题,即,派生类继承并具有父类的所有特性,包括父类的所有属性、方法程序和事件。这里的表单类是由表单派生而产生的,所以,该表单类也继承了原表单及原表单中所有控件的所有属性、方法程序和事件。但这些属性、方法程序和事件又都具有封装性,所以大家在一些事件代码中看不到原有代码。

由用户定义的类创建的表单或控件,当用户不在这些表单或控件的某些事件中设置新的代码时,该事件将执行其父类原有的代码,而当用户在这些在该事件中设置了新代码时,该事件将执行用户设置的代码。但有时,既希望执行用户设置的新代码,又执行其父类原有的代码时,这就需要在用户设置的新代码中添加一条语句:DoDefault()。如果在事件代码中仅仅添加这样一条语句:NoDefault(),则表示不执行其父类原有的代码。

7.3.5 为字段设置类

在表单上创建控件我们既可以通过表单控件工具栏向表单中添加控件,也可以直接将项目管理器中的类拖入到表单中创建用户定义的类,还可以将数据环境中的表或视图

中的字段直接拖入表单中,来创建该字段的相应控件,这类控件一般情况下由系统默认设置,我们也可以来改变系统的默认设置,其方法是通过表设计器来改变。具体步骤如下:

- 1. 将要修改的表在表设计器中打开,如图 7.14 所示;



图 7.14 表设计器

- 2. 选择要重新设置类的字段;
- 3. 在显示类中选择要更改的类名;或通过显示库选择用户定义的更改类名;
- 4. 依次完成各个要修改字段的类设置;
- 5. 单击“确定”按钮,保存结果。

大家可以试着将表 Employee 的年龄和工资字段的类设置为 Spinner 类,然后创建一个表单,在该表单的数据环境中添加表 Employee,然后将该表的各字段拖入到表单中,观察一下年龄和工资字段的显示控件的变化。

7.3.6 用户定义工具栏

创建用户自定义工具栏的步骤:

- 1. 从 Toolbar 基类创建一个自定义工具栏类;
- 2. 通过表单控件工具向自定义工具栏添加按钮,并设置按钮的相关属性、方法程序和事件代码等;
- 3. 在表单集中创建该自定义工具栏。

例 7.8 将例 6.11 中添加的四个命令按钮更改为用户定义工具栏形式,然后,向设计的表单中添加该工具栏。运行效果如图 7.15 所示。

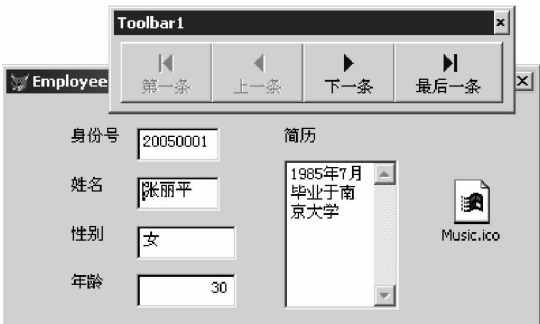


图 7.15 运行效果图

设计步骤：

1. 创建自定义工具栏 Toolbar1。

通过 Toolbar 基类,创建一个 Name 属性为 Toolbar1 的用户自定义类:

(1)在命令行中输入命令:Modif Class Toolbar1 Of Toolbar1,创建一个类库名为 Toolbar1,类名为 Toolbar1 的用户自定义类;执行完该命令后,弹出一个类设计器对话框,如图 7.16 所示;

(2)向 Toolbar1 对象中添加四个命令按钮,并按例 6.11 中的“表 6.14”设置四个命令按钮的相关属性;设计好后效果如图 7.16 所示;



图 7.16 “Toolbar1 工具栏”类设计器

(3)设置 Toolbar1 表单及其控件的事件代码:

①设置表单 Toolbar1 表单的 Init 事件代码:

该事件代码与例 6.11 中的 Employee Init 事件代码相同。

②分别设置“第一条”、“第二条”、“第三条”、“第四条”命令按钮的 Click 事件代码:

这四个命令按钮的 Click 事件代码与例 6.11 中四个命令按钮的事件代码基本相同,在本例中只需要将例 6.11 中四个命令按钮的事件代码中的表单刷新语句:Thisform.Refresh,更改为表单集的刷新语句:ThisformSet.Refresh 即可。

(4)关闭类设计器,并保存用户自定义类。

2. 在表单控件工具栏中添加用户自定义的类 Toolbar1。

将用户定义的类添加到表单控件工具栏中的方法有很多,如通过系统菜单“工具/选项”,在弹出的“选项”对话框中的“控件”选项卡中,将用户刚才定义的类进行注册;又如直接通过表单控件工具栏的“查看类”按钮,也可将用户刚才定义的类添加到表单控件工具栏中。

3. 打开一个设计好的表单,向该表单添加用户定义的工具栏。

打开例 6.11 中的 Employee 表单,删除该表单中命令按钮及与命令按钮相关的语句;通过表单控件工具栏向该表单添加工具栏,由于工具栏也是一个表单,所以添加时会出现一个询问对话框,询问“要添加一个表单,需要一个表单集对象。创建吗?”,单击“是”按钮,完成工具栏的添加。

4. 保存表单集,运行该表单集,运行效果如图 7.15 所示。

5. 在本例中我们还可以设置表单集的 Form1 的 RightClick 事件代码,来实现表单集运行时,右击 Form1 表单空白处时,让工具栏显示或隐藏,具体代码设置如下:

```
IF ThisFormSet.Toolbar11.Visible=.T.  
    ThisFormSet.Toolbar11.Visible=.F.  
Else
```

```
ThisFormSet.Toolbar11.Visible=.T.  
Endif
```

习 题

1. 在什么情况下,含有两个表单的应用程序的两个表单是父表单与子表单关系?

2. 建立一个多表单应用程序,要求如下:

(1) 利用 Student 数据库的 Student,Collegedepartment,Reportcard 三个表,分别建立三个显示数据库表数据的表单;

(2) 建立的三个表单,要求至少有一个表单不是以表格形式显示记录;

(3) 建立一个新表单,并将其作为父表单,将刚才建立的三个表单作为子表单,通过父表单显示子表单。

3. 将上例设计成表单集的形式,通过将上例的表单分别设计成表单类后,再依次添加到表单集中。

4. 创建一个用户定义工具栏,将其分别添加到第 2 题的三个数据表单,和第 3 题中的表单集中,体会在添加的过程中有什么不同,同时验证工具栏中的按钮是否可用(用户设计的工具栏可以是编辑工具栏,也可以是一个记录定位工具栏)。



第 8 章

菜单设计

菜单是 VFP 应用程序中非常重要的一部分内容,通过菜单可以让用户很方便地使用应用程序,本章主要介绍利用 VFP 菜单设计器来设计下拉式菜单与快捷菜单,最后还将介绍利用菜单命令来设计弹出式菜单。

8.1 下拉式菜单的设计

8.1.1 菜单生成的基本步骤

利用菜单生成器设计并生成下拉式菜单或快捷菜单的基本步骤包括:打开菜单设计器窗口,菜单设计,保存菜单定义,生成菜单程序,运行菜单程序。特别需要指出的是当在菜单设计器窗口中设计好菜单后,必须通过系统菜单“菜单/生成”命令,将菜单文件(扩展名为 .mnx)转化为菜单程序文件(扩展名为 .mpr),运行菜单时系统运行的是菜单程序文件。

8.1.1.1 打开菜单设计器窗口

菜单设计器窗口的打开方式通常有三种:

1. 通过系统菜单新建菜单或打开菜单时,打开菜单设计器窗口。

(1) 新建菜单时,打开菜单设计器窗口:

①单击系统菜单“文件/新建”命令或“新建”命令按钮,弹出“新建”对话框窗口;

②在“新建”对话框中选定“菜单”选项,并单击“新建文件”按钮,弹出“新建菜单”对话框,如图 8.1 所示;

③在“新建菜单”对话框中,单击“菜单”按钮进入菜单设计器,如图 8.2 所示。

(2) 打开菜单文件时,打开菜单设计器窗口:

①单击系统菜单“文件/打开”命令或“打开”命令按钮,弹出“打开”对话框;

②在“打开”对话框中的文件类型组合框中选择“菜单”选项,在文件列表中选择某菜单文件后,单击该对话框的“确定”按钮,打开菜单设计器窗口。



图 8.1 “新建菜单”对话框

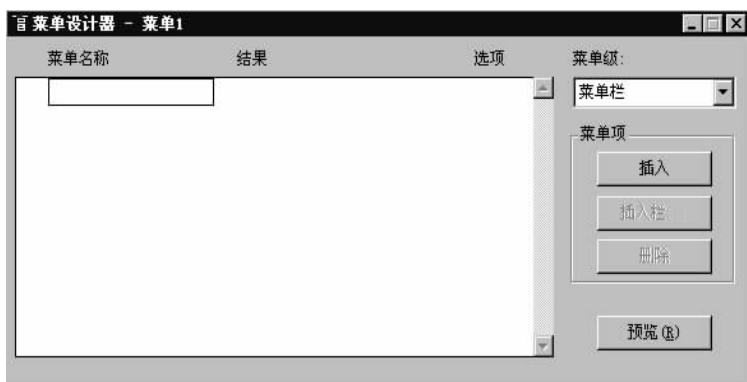


图 8.2 菜单设计器窗口

2. 用命令建立或打开菜单时,打开菜单设计器窗口。

格式一:Create Menu [<文件名>]

格式二:Modify Menu [<文件名>]

功能:用来在菜单设计器窗口建立或打开菜单。

参数说明:

(1) 命令中的<文件名>指菜单文件,其扩展名为.mnx,可以缺省;

(2) 格式一表示创建名为<文件名>的菜单,如果指定位置存在与该文件名同名的菜单文件,则系统会提示是否覆盖;

(3) 格式二表示建立或打开<文件名>指定的菜单文件,如果指定位置不存在与该文件名同名的菜单文件,则创建该菜单文件;否则在菜单设计器窗口中打开<文件名>指定的菜单文件。

3. 使用项目管理器来建立或打开菜单时,打开菜单设计器窗口。

(1) 新建或打开项目管理器,在项目管理器窗口的“全部”或“其他”选项卡中,选择“菜单”选项;

(2) 如果需要新建菜单,则单击“新建”按钮,打开如图 8.1 所示新建菜单对话框,单击“菜单”按钮打开菜单设计器窗口建立菜单;

(3) 如果是修改菜单,则需要先在项目管理器中选择需要修改的菜单文件,然后,单击“修改”按钮,打开菜单设计器窗口;

(4) 如果需要修改的菜单文件不在项目管理器中,可通过单击“添加”按钮,将菜单文件添加到项目管理器中,然后再单击“修改”按钮,打开菜单设计器窗口。

8.1.1.2 菜单设计

当打开菜单设计器窗口后,在 VFP 系统菜单中会自动增加一个“菜单”菜单,同时在系统的“显示”菜单中也会增加了“常规选项”、“菜单选项”两个命令。利用菜单设计器窗口和这些系统新增的菜单命令,可以很方便地进行比较复杂的菜单设计,具体设计过程将在本节的稍后内容中介绍。

8.1.1.3 保存菜单定义

当在菜单设计器窗口中设计好菜单后,菜单文件将保存在扩展名.mnx 的菜单文件

和扩展名为.mnt的菜单备注文件中。保存菜单定义的方法：

1. 单击菜单设计器窗口的关闭按钮；
2. 按组合键 Ctrl+W,此时菜单定义存盘且菜单设计器窗口被关闭；
3. 单击系统菜单“文件/保存”命令或单击“保存”按钮,系统即可保存当前的菜单定义,但菜单设计器窗口并不关闭；
4. 直接单击系统菜单“菜单/生成”命令,会在生成菜单程序文件(扩展名为.mpr)前,自动保存菜单定义。

需要指出的是:如果是新建的菜单,并且没有保存过菜单定义,无论使用以上何种方式保存,系统都会弹出对话框,询问“要将所做更改保存到菜单设计器中吗?”,在窗口中单击“是”,并按系统提示完成菜单定义的存盘。

8.1.1.4 生成菜单程序

利用“菜单设计器”设计菜单选项及每个菜单选项任务后,菜单设计工作仍未结束,用户必须通过系统提供的生成器,将菜单文件转换成菜单程序文件(扩展名为.mpr)。菜单文件只有在“菜单设计器”中打开的前提下,才可以通过系统菜单“菜单/生成”命令,将菜单文件转化生成菜单程序文件(扩展名为.mpr)。其转化生成步骤如下:

1. 在“菜单设计器”窗口中打开要转化生成的菜单文件(扩展名为.mnx)；
2. 单击系统菜单“菜单/生成”命令,弹出“生成菜单”对话框,如图8.4所示见下页；
3. 在“生成菜单”对话框中的输出文件文本框中,系统默认的菜单程序文件的主名与菜单文件的主名相同,但用户可输入一个新菜单文件名；
4. 单击“生成”按钮,系统将自动创建一个扩展名为.mpr的菜单程序文件。

8.1.1.5 运行菜单

菜单程序文件的运行方式:

1. 菜单方式:单击系统菜单的“程序/运行”命令后,在弹出的“运行”对话框中选择需运行的菜单程序文件名后,单击“运行”按钮,运行菜单。

2. 命令方式。

命令格式:

Do <菜单程序文件名.mpr>

功能:执行名为<菜单程序文件名.mpr>的菜单程序文件。

(1) 菜单程序的扩展名.mpr不能省略,若省略,系统则默认执行扩展名为.prg的程序文件;

(2) 运行菜单程序时,VFP自动对.mpr文件进行编译并产生目标程序.mpx,而且对于主文件名相同的.mpr和.mpx系统总是运行后者。

3. 项目管理器方式。

在“项目管理器”中选择相应菜单文件,并单击“运行”按钮。

8.1.2 快速菜单生成

在第七章的例7.1中,我们已经介绍过了快速菜单生成的操作方式,现在我们将详细地介绍快速菜单的生成方式,以及需要注意的问题。

当打开菜单设计器窗口后,在系统菜单中就会增加一个“菜单”的菜单,其中有一个“快速菜单”选项,用户可以使用“快速菜单”命令创建和系统菜单一样的菜单,并且可以在菜单设计器窗口中修改这个菜单,使它符合自己的需要。但需要注意两点:

1. “快速菜单”命令只有在菜单设计器窗口中没有任何菜单选项时才有效,否则它为无效的灰色。

2. 快速菜单命令只能用于产生下拉式菜单,快捷菜单不通过快速菜单产生。

使用“快速菜单”命令创建用户菜单,可按照以下步骤进行:

1. 单击系统菜单“文件/新建”,并在弹出的“新建”对话框中选择“菜单”并单击“新建”文件按钮。

2. 在弹出的“新建菜单”对话框中单击菜单“按钮”,弹出如图 8.1 所示的菜单设计器窗口,此时在菜单设计器窗口中,没有任何菜单项。

3. 单击系统菜单“菜单/快速菜单”,则在菜单设计器窗口中就会出现一个与 VFP 系统菜单一样的菜单,如图 8.3 所示。

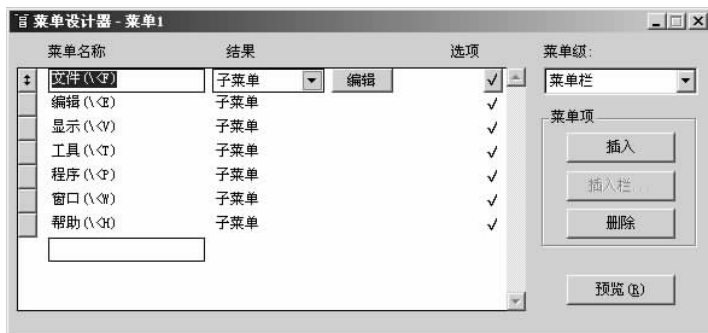


图 8.3 快速菜单设计器窗口

4. 在“菜单设计器”中可以选择添加或者修改菜单项,以便定制菜单系统。用户可以通过“插入”按钮插入新菜单,也可以通过“删除”按钮删除不需要的菜单。具体操作方法将在下面一节介绍。

5. 单击系统菜单“菜单/生成”,将弹出一个保存对话框,将菜单保存为 SysMenu1.mnx,单击确定后,弹出“生成菜单”对话框,图 8.4 所示,单击该对话框的“生成”按钮,生成 SysMenu1.mpr 菜单程序文件,完成系统菜单的设计。

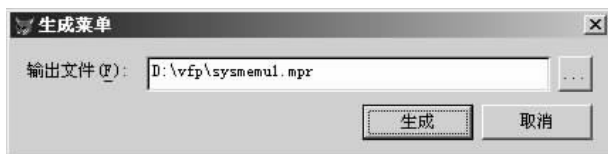


图 8.4 生成菜单对话框

8.1.3 菜单设计器窗口

菜单设计器窗口由左右两大部分组成,在窗口的左边是一个列表框,在该列表框中每行都可定义一个菜单项,列表中的“菜单名称”、“结果”、“选项”列表示菜单项属性。菜单

设计器窗口的右边由一个菜单级组合框和四个按钮组成,通过菜单级组合框,可以从下级菜单页切换到上级菜单页;“插入”、“插入栏”、“删除”、“预览”按钮,分别可进行插入菜单项、删除菜单项和菜单模拟显示等操作。

8.1.3.1 “菜单名称”列

该列用于输入显示在菜单系统中的菜单标题或菜单项名称。

如果用户想为某菜单项定义访问键,即允许用户利用键盘访问方式调用该菜单项,则可以在该菜单项“菜单名称”列中,在欲设定为访问键的字母前面加上一反斜杠和小于号,格式为:

\<字母。

例如,在“文件(F)”菜单中设计访问键为“F”,则是在菜单名称“F”的前面加上“\<”,即:“文件(\<F)”。

用户在创建子菜单时还可以在“菜单名”列中输入“\—”以创建一条分隔线将内容相关的菜单项分组。

8.1.3.2 “结果”列

此栏选定菜单项的功能类别,该栏的下拉列表框中有命令、子菜单、过程和填充名称或菜单项等四种选项。

1. 命令。

为菜单项定义一条动作命令,命令输入到其右边的文本框中。当该用户定义菜单被激活后,该菜单项对应的响应事件就是执行该条命令。

2. 过程。

为菜单项定义一个过程,当该用户定义菜单被激活后,该菜单项对应的响应事件就是执行该过程。

在设计时,选择“过程”,在其右侧会出现“创建”按钮,单击“创建”按钮,系统自动弹出一个过程编辑窗口,供用户输入和编辑过程代码。

一旦过程建立完毕后,原来“创建”按钮自动变成“编辑”按钮,通过“编辑”按钮可修改过程代码。

3. 子菜单。

供用户为当前菜单项定义下一级菜单,选择“子菜单”,其右侧会出现“创建”按钮。单击“创建”按钮,菜单设计器切换并进入到当前菜单项的子菜单页,用户可以在该子菜单页中建立和修改子菜单。

子菜单创建完成后,“创建”按钮自动变成“编辑”按钮,通过“编辑”按钮可修改子菜单。

从子菜单页返回到上级菜单页的方式是:选择右上角菜单级组合框“菜单栏”选项,并选择要返回的上一级菜单,返回到选定的菜单页。

4. 填充名称或菜单项#。

该选项可用来定义条形菜单(下拉式菜单的一级菜单)的菜单名或者是弹出式菜单的(下拉式菜单的子菜单)菜单项序号。当用户选择该项,其右侧出现文本框,如果定义的是条形菜单则此选项是“填充名称”,可输入菜单项名称,如果定义的是弹出式菜单则此选项

是“菜单项#”，可输入菜单项序号。

8.1.3.3 “选项”列

“选项”列含有一个无符号按钮，选定该按钮就会出现如图 8.5 所示“提示选项”对话框，在该对话框中可以定义菜单项的附加属性。当菜单项定义过附加属性后，按钮面板上则显示符号“√”。“提示选项”对话框的主要功能说明如下：

1. 快捷方式。

该属性用于定义当前菜单项的快捷键，用户将光标定位于“键标签”文本框内，然后可在键盘上按下快捷键，如 Ctrl+X，这时在文本框内就会出现相应的按键组合 Ctrl+X（如果要取消快捷键只要按空格键即可）。

同时“键说明”文本框也会出现与“键标签”文本框内相同的内容（此内容可修改），当该菜单被激活后，在菜单项的右侧显示“键说明”文本框中的内容，用于对快捷键的说明。

但是需要注意，在输入快捷键的时候一般用 Ctrl+字母键或者 Alt+字母键，也可以是 Ctrl+功能键或者 Alt+功能键。另外，Ctrl+J 是无效的快捷键，因为在 VFP 中，经常将其作为关闭某些对话框的快捷键。

2. 跳过。

该属性用于确定菜单项是否可选，可以在“跳过”文本框中输入一个逻辑表达式或者单击右侧按钮构造一个逻辑表达式。当该菜单被激活后，如果该表达式为假（.F.），则该菜单项可选用，如果该表达式为真（.T.）则该菜单项不可选用，并以淡色显示。

3. 信息。

该属性用于定义菜单项的提示信息，在“信息”文本框中可输入一个字符串或者字符表达式作为提示信息。当该菜单被激活后，并选中该菜单项时，该信息将会再状态栏中显示。

4. 主菜单名。

该属性用于定义水平选单的选单项名或者是弹出式选单的选项序号，作用同上面的“结果”列中的“填充名称或菜单项#”选项功能相似。

8.1.3.4 插入按钮

单击该按钮后，在当前菜单行之前插入一个新菜单行，等待用户输入新的菜单项。该按钮与系统菜单“菜单/插入菜单项”命令的功能相同。

8.1.3.5 插入栏按钮

单击该按钮后，系统会弹出一个“插入系统菜单栏”对话框，要求用户选择一个与系统菜单一样的菜单项，并将用户选择的菜单项插入在当前菜单行之前。

但该按钮只有在建立或编辑子菜单时，才有效，否则为浅色无效。该按钮与系统菜单“菜单/插入栏”命令的功能相同。



图 8.5 “提示选项”对话框

8.1.3.6 删除按钮

单击该按钮后,删除当前的菜单行。该按钮与系统菜单“菜单/删除菜单项”命令的功能相同。

8.1.3.7 预览按钮

该按钮供菜单模拟显示,用户可用此按钮显示当前设计的菜单。该按钮与系统菜单“菜单/预览”命令的功能相同。

8.1.4 “显示”菜单的命令

当菜单设计器窗口为当前窗口时,在 VFP 的系统菜单“显示”菜单中,会自动增加“常规选项”和“菜单选项”两个命令,如图 8.6 所示。在菜单设计窗口的基础上,利用这两个命令,能够更方便地设计出比较完善的菜单。



图 8.6 “显示”菜单图

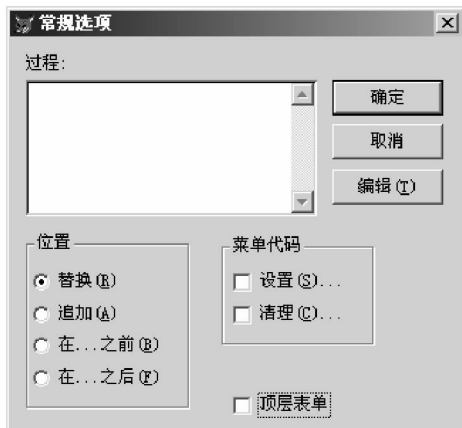


图 8.7 “常规选项”对话框

8.1.4.1 常规选项

单击系统菜单“显示/常规选项”命令,系统弹出“常规选项”对话框,如图 8.7 所示。

1. “过程”编辑框。

可直接在该编辑框中输入公共过程,也可通过单击“编辑”按钮,在弹出的过程编辑窗口中输入公共过程。

如果在设计菜单时,第一级菜单中某些菜单项既没有设置过任何命令或过程语句,也没有下级菜单,则该用户定义菜单被激活后,当选择这些菜单项时,系统会自动去执行这个公共过程。

2. 位置区。

位置区有 4 个单选按钮选项,用来描述用户定义菜单被激活后,用户定义菜单与当前系统激活的菜单之间的关系。

替换:用用户定义菜单替换已经激活的菜单,该选项为默认选项。

追加:将用户定义菜单添加到已经激活的菜单右边。

在...之前:将用户定义菜单插在选定的某菜单之前。选中此项时,其右侧会出现一个下拉式列表框,在该下拉式列表框中显示当前已经被激活的菜单的第一级菜单项,从中

选择一个菜单项,在用户定义的菜单被激活时,该菜单将插在选中的菜单项前。

在...之后:将用户定义菜单插在选定的某菜单之后。其使用方法同“在...之前”选项的用法。

3. 菜单代码区。

菜单代码区有两个复选框即“设置”和“清理”。无论选定“设置”复选框,还是选定“清理”复选框,都会弹出一个代码编辑窗口,在该窗口可以键入一些相应的处理代码。

选择“设置”复选框弹出的代码编辑窗口,用来设置菜单程序的初始化代码,在将菜单文件转化生成菜单程序文件后,该代码段位于菜单程序文件的最前面。所以,该代码编辑窗口,可用于全局性设置,如设置全局变量、数组、运行环境等。

选择“清理”复选框弹出的代码编辑窗口,用来设置菜单程序的清理代码,在将菜单文件转化生成菜单程序文件后,该代码段位于菜单程序文件的最后面。所以,该代码编辑窗口,可用于菜单显示后的清理代码的设置。

4. 顶层表单。

选中该项表示可将当前定义的菜单添加到顶层表单,在第七章的“7.1.1.3 在顶层表单中添加菜单”中,通过例 7.1 介绍了,向顶层表单添加菜单的操作过程,这里就不再具体说明。

需要说明的是当该可选项被选中时,该用户定义菜单只能作为顶层表单的菜单运行;否则作为一个独立的菜单系统。

8.1.4.2 菜单选项

单击系统菜单“显示/菜单选项”命令,系统弹出“菜单选项”对话框,如图 8.8 所示。

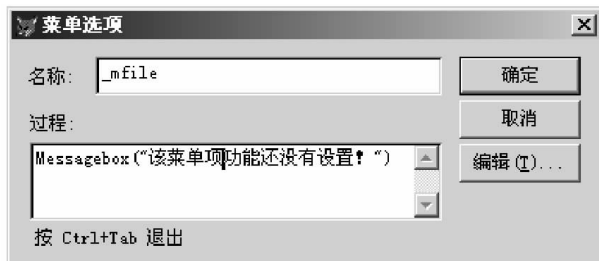


图 8.8 菜单选项对话框

1. 名称。

显示当前菜单名,默认情况下和“菜单设计器”窗口中的“菜单级”列表中的文本一致,用户可手工输入新的菜单名称。

2. 过程编辑框。

用于为某个子菜单中的某些菜单项编写公共过程代码。

如果在设计菜单时,该子菜单中某些菜单项既没有设置过任何命令或过程句,也没有下级菜单,则该用户定义菜单被激活后,当选择该子菜单下的这些菜单项时,系统会自动去执行这个公共过程。

3. 编辑。

单击该按钮,打开一个过程编辑窗口,可在其中输入公共过程代码。

例 8.1 利用前面设计的表单,设计运行效果如图 8.9 的菜单系统。

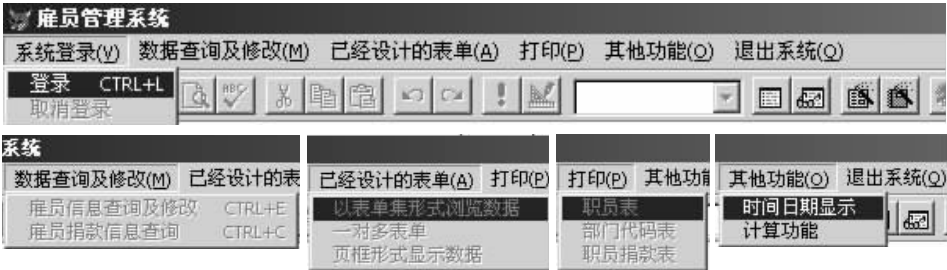


图 8.9 菜单运行效果

设计步骤：

1. 在 Command 窗口输入并执行命令:Modify Menu Emplmenu,打开菜单设计器。

2. 一级菜单的设置:在菜单设计器窗口中填入如图 8.10 所示的一级菜单项,而且所有的一级菜单均设置了访问键。



图 8.10 一级菜单项设置

3. 设置菜单系统的初始化代码。

单击系统菜单“显示/常规选项”,弹出“常规选项”对话框,选择“设置”复选框,弹出“设置”编辑窗口,在该窗口输入菜单的初始化代码:

Clear

Clear all

Close all

* 关闭当前窗口,因菜单程序是命令方式执行的,所以是关闭 Command 窗口

Keyboard "{Ctrl+F4}"

Modify Window Screen Title "雇员管理系统" &&. 设置菜单窗口标题

Public IDVerify

IDVerify=. F. &&. 通过该变量的设置,来控制一些菜单项是否为灰色不可用

4. 设置“系统登录”子菜单选项。

在一级菜单设置窗口中单击“系统登录”菜单项的“编辑”按钮,弹出“系统登录”菜单设计器窗口,在该窗口中输入如图 8.11 所示的内容。

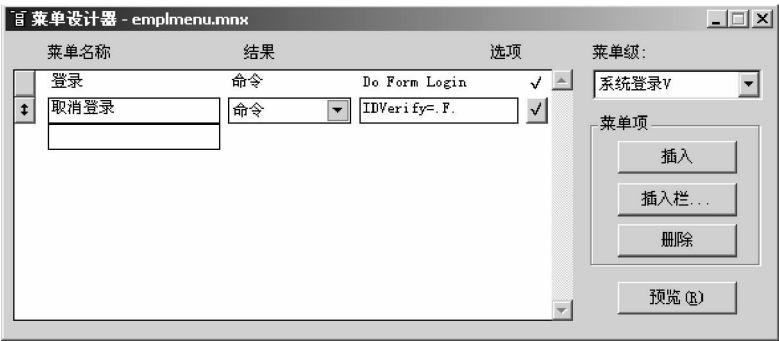


图 8.11 “系统登录”子菜单中各菜单项设置

“登录”菜单项中输入的命令:Do Form Login,是调用例 6.10 设计的表单 Login.scx。

“取消登录”菜单项中输入的命令:IDVerify=. F.,其作用是当用户单击些菜单时,将一些菜单设置为不可用的灰色显示形式;其中 IDVerify 是在菜单初始化代码中设置的一个公共变量。

通过菜单项的“选项”按钮,设置“登录”菜单项的快捷键:CTRL+V,设置“取消登录”菜单项的跳过规则:IDVerify=. F.。如图 8.12 和 8.13 所示。在该菜单被激活后,用户可以通过按快捷键“CTRL+V”直接执行“登录”菜单项,并且在用户没有成功登录前,“取消登录”菜单项为灰色不可用。



图 8.12 “登录”菜单项选项设置



图 8.13 “取消登录”菜单项选项设置

5. 设置“数据查询及修改”子菜单选项。

单击菜单级下拉式列表框,选择“菜单栏”返回到一级菜单,并单击“数据查询及修改”菜单项的“编辑”按钮,弹出“数据查询及修改”菜单设计器窗口,在该窗口中输入如图8.14 所示的内容。

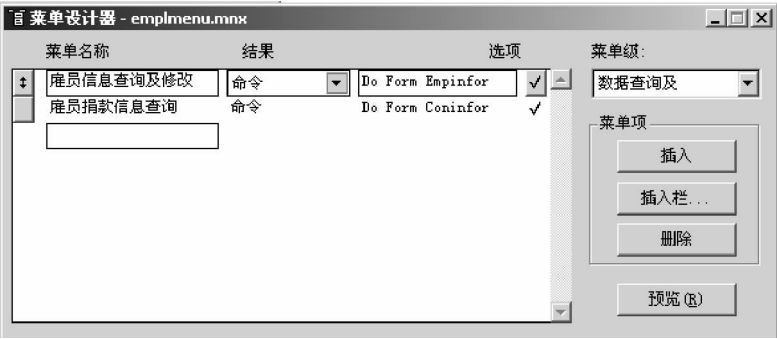


图 8.14 “数据查询及修改”子菜单中各菜单项设置

“雇员信息查询及修改”菜单中输入的命令:Do Form Empinfor,是调用例 6.23 中设计的表单 EmpInfor.scx;同时设置该菜单项的快捷键为:CTRL+E。

“雇员捐款信息查询”菜单中输入的命令:Do Form Coninfor,是调用例 6.23 中设计的表单 ConInfor.scx;同时设置该菜单项的快捷键为:CTRL+C。

并设置两个子菜单项的跳过规则为:IDVerify=.F.,其设置过程同“系统登录”菜单下的“取消登录”菜单项的设置过程。

6. 设置“已经设计的表单”子菜单选项。

单击菜单级下拉式列表框,选择“菜单栏”返回到一级菜单,并单击“已经设计的表单”菜单项的“编辑”按钮,弹出“已经设计的表单”菜单设计器窗口,在该窗口中输入如图8.15所示的内容。

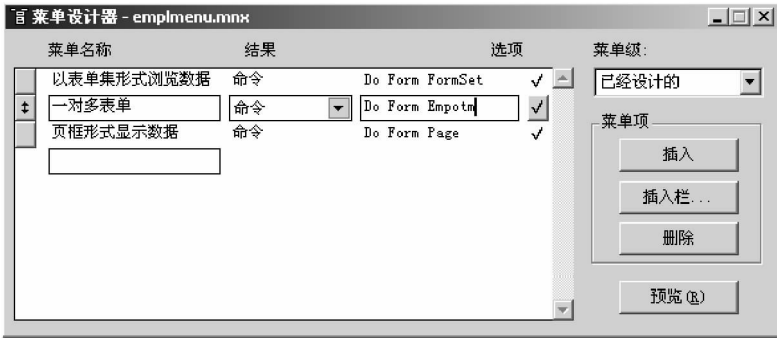


图 8.15 “已经设计的表单”子菜单中各菜单项设置

“以表单集形式浏览数据”菜单中输入的命令:Do Form FormSet,是调用例 7.2 中设计的表单 FormSet.scx。

“一对多表单”菜单中输入的命令:Do Form Empotm,是调用例 6.18 中设计的表单 Empotm.scx。

“页框形式显示数据”菜单中输入的命令:Do Form Page,是调用例 6.19 中设计的表单 Page.scx。

并设置以上三个子菜单项的跳过规则为:IDVerify=.F.,其设置过程同前。

7. 设置“打印”子菜单选项。

单击菜单级下拉式列表框,选择“菜单栏”返回到一级菜单,并单击“打印”菜单项的

“编辑”按钮,弹出“已经设计的表单”菜单设计器窗口,在该窗口中输入如图 8.16 所示的内容,三个菜单项的命令均为空,路过规则也均为:IDVerify=. F. 。

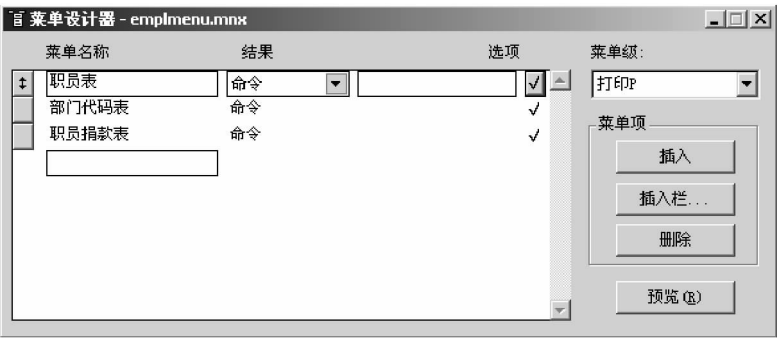


图 8.16 “已经设计的表单”子菜单中各菜单项设置

单击系统菜单“显示/菜单选项”,弹出图 8.17 所示的“菜单选项”对话框,在该对话框的过程编辑框中,输入一个公共过程,如:Messagebox(“该菜单项功能还没有设置!”)。当该菜单被激活后,该子菜单没有设置过任何命令、过程句和下级菜单的这些菜单项,被用户调用时,系统会自动调用这个公共过程,弹出一个提示对话框,提示:该菜单项功能还没有设置!

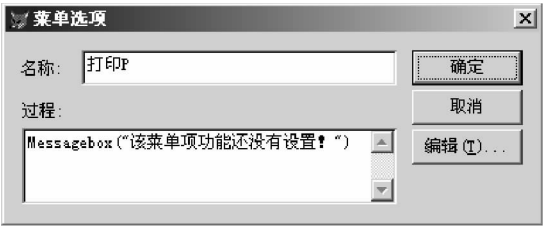


图 8.17 “打印”子菜单公共过程设置

8. 设置“退出系统”子菜单的过程代码。

单击菜单级下拉式列表框,选择“菜单栏”返回到一级菜单,并单击“退出系统”菜单项的“编辑”按钮,弹出“退出系统过程”编辑窗口,在该窗口中输入过程代码:

- | | |
|--------------------------|------------------|
| Modify Windows Screen | &. 恢复 VFP 主窗口标题 |
| Set Sysmenu To Default | &. 恢复 VFP 系统菜单 |
| Activate Windows Command | &. 恢复 Command 窗口 |
| Release IDVerify | &. 清除定义的公共内存变量 |

8.2 弹出式菜单设计

Visual FoxPro 6.0 系统提供了大量的快捷菜单,为用户操作提供了方便。用户也可以创建自己的快捷菜单,并将它添加到对象中。若在该对象的鼠标 RightClick 事件中,调用该快捷菜单,则在运行时,只要在该对象上单击鼠标右键,就会显示用户创建的快捷菜单。

8.2.1 利用菜单设计器设计快捷菜单

在 VFP6 或者在 Windows 中,选定某个控件或对象后单击右键时,就会显示快捷菜单,可以快速展示对当前对象进行操作的各种可用的功能。正是因为这种菜单的方便快捷,它才被称为快捷菜单。我们也可以利用“快捷菜单设计器”窗口来创建自己的快捷菜单,创建快捷菜单主要步骤如下:

1. 利用菜单设计器设计好快捷菜单;
2. 生成快捷菜单的菜单程序文件;
3. 在应用程序中利用功能键的定义命令来调用快捷菜单;
4. 退出应用程序时清除功能键。

在介绍利用菜单设计器设计快捷菜单前,先介绍一下功能键的定义与清除命令。

1. 功能键的定义。

命令格式:

On Key Label <快捷键> <命令>

功能:设置功能键及其功能,包括为鼠标按键设置功能。

参数说明:

(1)Label 子句的<快捷键>表示定义功能的按键,<快捷键>一般是 Ctrl+字母键或者 Alt+字母键,也可以是 Ctrl+功能键或者 Alt+功能键,还可能是鼠标按键如:RightMouse,LeftMouse,Mouse 等;

(2)<命令>指定功能键要执行的功能。

2. 功能键的清除。

命令格式:

Push Key Clear

功能:清除以前设置的功能键。

例 8.2 设置一个具有“撤销”、“剪切”、“复制”、“粘贴”和“选择性粘贴”等编辑功能的快捷菜单,并在例 8.1 中应用。

设计步骤:

1. 利用菜单设计器设计好快捷菜单:Quick.mnx。

(1)单击系统菜单“文件/新建”命令,弹出的“新建”对话框;

(2)在“新建”对话框中选择“菜单”后,再单击“新建文件”按钮,弹出“新建菜单”对话框;

(3)在“新建菜单”对话框中选择“快捷菜单”,即可打开“快捷菜单设计器”窗口,如图 8.18 所示,其界面及使用方法与“菜单设计器”窗口相似;

(4)插入菜单项:单击“插入栏”按钮,弹出的“插入系统菜单栏”对话框,如图 8.19 所示。在该对话框中选择相应的菜单项插入到快捷菜单设计器窗口中,结果如图 8.18 所示;



图 8.18 “快捷菜单设计器”窗口

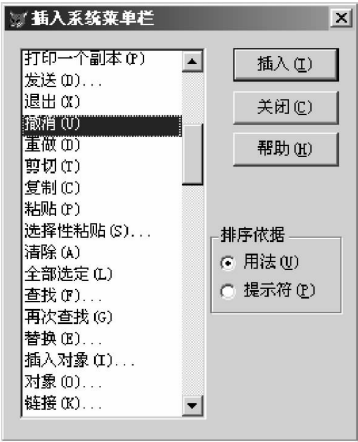


图 8.19 “插入系统菜单栏”对话框

(5) 如果需要插入用户自定义的菜单选项,其操作方法与下拉式菜单设计相似。

2. 生成快捷菜单程序文件:单击系统菜单“菜单/生成”,生成名为 Quick. mpr 的菜单程序文件。

3. 在菜单设计器窗口中打开例 8.1 的菜单 Emplmenu. mnx,并在菜单的初始化代码的最后加上下列代码:

```
* 设置鼠标右键功能为用户定义的快捷菜单  
On Key Label RightMouse Do Quick. mpr
```

4. 在菜单 Emplmenu. mnx 的“退出系统”过程代码的最后加上下列代码:

```
Push Key Clear &&.清除所有功能键设置
```

5. 重新生成菜单程序文件:单击系统菜单“菜单/生成”,重新生成名为 Emplmenu. mpr 的菜单程序文件。

6. 在命令窗口执行命令:Do Emplmenu. mpr,检验鼠标右键的功能。

8.2.2 用菜单命令设计弹出式菜单

弹出式菜单的设计步骤一般分定义弹出式菜单、定义弹出式菜单的菜单项、定义弹出式菜单菜单项的动作和激活弹出式菜单四步。对应这四步弹出式菜单设计的专用菜单命令也有四种:弹出式菜单的定义命令,弹出式菜单项的定义命令,弹出式菜单项动作定义命令和弹出式菜单激活命令。下面我们依次介绍这四条命令的命令格式,然后,通过例题来介绍弹出式菜单的设计过程。但需要说明的是通过命令方式建立的弹出式菜单,只有当鼠标单击菜单以外的地方或按“Esc”键才能退出。

1. 弹出式菜单定义命令。

命令格式:

```
Define Popup <弹出式菜单名> [ Title <字符表达式 1>];  
[Form <行坐标 1,列坐标 1>][To<行坐标 2,列坐标 2>][Relative];  
[In [Window]<窗口名> | In Screen][Key<键标号>];  
[Message<字符表达式 2>] [Mover] [Margin] [MultiSelect];
```

[Prompt Field <表达式> | Propmt Files [Like<通配符表达式>] | ;
Prompt Structure];

[Scroll] [Color Scheme<数值表达式> | Color <颜色对表>]

功能:定义弹出式菜单及其属性。

参数说明:

- (1) <弹出式菜单名>:定义弹出式菜单的名称;
- (2) Title <字符表达式 1>:指定弹出式菜单窗口的标题;
- (3) Form <行坐标 1,列坐标 1>:指定弹出式菜单左上角的坐标;To<行坐标 2,列坐标 2>:指定弹出式菜单右下角的坐标。如果缺省这两个子句,则表示菜单左上角的坐标为第 0 行第 0 列;
- (4) Relative:允许用 Define Bar 命令定义菜单项时,可用 Before 或 After 子句来调整菜单项的相对位置;
- (5) In [Window]<窗口名>:将弹出式菜单放在由<窗口名>指定的窗口中;In Screen:将弹出式菜单放在屏幕上;
- (6) Key<键标号>:定义启动弹出式菜单的快捷键;
- (7) Message<字符表达式 2>:指定弹出式菜单的提示信息;
- (8) Mover:用于改变菜单项的显示位置,使第一个菜单项的左边出现双向箭头,通过鼠标上下拖动双向箭头就可以重新安排菜单项的位置;也可通过键盘,按 ctrl+↓或者 ctrl+↑组合键来移动;
- (9) Margin:在弹出式菜单中各菜单选项左侧各添加一个空格;
- (10) MultiSelect:指定在弹出式菜单中可选择多个选项;
- (11) Prompt Field <表达式>:指定以库文件字段值为内容的滚动列表,其中<表达式>表示字段名;Propmt Files:指定以磁盘文件名为内容的滚动列表,其中 like<通配符表达式>指定文件名。Prompt Structure:指定以库文件结构字段名为内容的滚动列表;
- (12) Scroll 在弹出式菜单组合框增加一垂直滚动条;
- (13) Color Scheme<数值表达式>或者 color <颜色对表>可以指定弹出式菜单项的颜色。

2. 弹出式菜单的菜单项定义命令。

命令格式:

Define Bar <数值表达式 1> Of <弹出式菜单名>;

Prompt <字符表达式 1> [Before <数值表达式 2>|After<数值表达式 3>];

[Key<键标号>[,<字符表达式 2>]][Message<字符表达式 3>];

[Skip [For <逻辑表达式>]]

功能:每次定义弹出式菜单的一个菜单项及其属性。

参数说明:

- (1) <数值表达式 1>:指定菜单项在<弹出式菜单名>指定的弹出式菜单中的位置序号;

(2) Prompt <字符表达式 1>:指定菜单项的显示名称。如果希望为该菜单项设置快捷键,可在菜单项中使用“\<字母>”的形式,如果希望在菜单项之间添加一条分割线,可在 prompt 后面使用“/-”的形式;

(3) 如果在 Define Popup 命令中使用了 Relative 选项则 Before <数值表达式 2>表示把菜单项放在由<数值表达式 2>指定的菜单项之前,而 After<数值表达式 3>则表示把菜单项放在由<数值表达式 3>指定的菜单项之后;

(4) Skip [For <逻辑表达式>]:可以限定该菜单项能否使用,当<逻辑表达式>为“真”时,表示不能选择,此时该选项为灰色显示,为“假”时可以选择。

3. 弹出式菜单项的动作定义命令。

定义弹出式菜单项动作的命令有三种格式:

格式一:

On Bar <数值表达式> Of <弹出式菜单名 1>;

[Activate Popup <弹出式菜单名 2>]

功能:把菜单项的动作定义为激活另一个弹出式菜单。

参数说明:

(1) <数值表达式>:表示要定义动作的菜单项的序号;

(2) <弹出式菜单名 1>:是菜单项所在菜单的名字;

(3) Activate Popup <弹出式菜单名 2>:表示被激活菜单的名字。

格式二:

On Selection Bar <数值表达式> Of <弹出式菜单名>[<命令>]

功能:选择<数值表达式>表示的菜单项后,就执行指定的<命令>。

格式三:

On Selection Popup <弹出式菜单名> | All [<命令>]

功能:选择由<弹出式菜单名>所代表的菜单中的任一菜单项后都执行<命令>。

如果用 All 子句代替<弹出式菜单名>,则所指定的<命令>可适用于该下拉弹出式菜单中的所有选项,即当已激活的弹出式菜单中的任一菜单项被选定时均执行指定的<命令>。

参数说明:

命令格式二和命令格式三中的<命令>通常为 do 命令调用的程序或过程,也可以是其他单个命令。

4. 激活弹出式菜单。

命令格式:

Activate Popup <弹出式菜单名> [At <行坐标,列坐标>];

[Bar <数值表达式>] [Nowait] [Rest]

功能:激活由<弹出式菜单名>指定的菜单。

参数说明:

(1) At <行坐标,列坐标>:指定弹出式菜单左上角的位置,其优先级高于 Define Popup 命令的 From 子句的优先级;

(2) Bar <数值表达式>:指定被激活的弹出式菜单当前菜单项序号;

(3) 若不带 Nowait 子句,表示弹出式菜单被激活时程序会暂停执行,等待用户选择菜单项或者按 Esc 键结束后,才往下执行。有 Nowait 子句,则程序在弹出式菜单被激活时会继续往下执行;

(4) 如果在菜单定义时,使用 Prompt Field 指定字段内容的弹出菜单内容,Rest 表示使库文件当前记录的字段内容成为当前可选项。

例 8.3 编制一个简单的弹出式菜单程序,通过菜单可以执行例 5.1、例 5.2、例 6.2 中完成的表单或在浏览窗口浏览表 Contribution。

源文件 Menu1.prg 的源程序代码如下:

Clear

* 定义名为“FF”,标题为“数据库表浏览”的单选弹出式菜单

Define Popup FF Title “数据库表浏览” From 2,20 Message “库表浏览” Margin

* 依次定义各菜单项及其各菜单项的快捷键、显示名和提示等

Define Bar 1 of FF Prompt “职工情况表” Key Ctrl+E,“E” Message “来源于例 5.1”

Define Bar 2 of FF Prompt “各部门职工情况表” Key Ctrl +D,“D”;

Message “来源于例 5.2”

Define Bar 3 of FF Prompt “例 6.2 Welcome 表单” Key Ctrl +W,“W”;

Message “来源于例 6.2”

Define Bar 4 of FF Prompt “在浏览窗口浏览表 Contribution” Key Ctrl +C,“C” ;

Message “执行一个名为 Con 的过程”

* 依次定义各菜单项对应的动作,其中前三项为执行表单,第四项为执行一个过程

On Selection Bar 1 Of FF Do Form 职工情况

On Selection Bar 2 Of FF Do Form 各部门职员情况表

On Selection Bar 3 Of FF Do Form Welcome

On Selection Bar 4 Of FF Do Con

* 激活弹出式菜单 FF

Activate Popup FF

Return

* 菜单第四项要执行的过程

Procedure con

Select 0

Use Contribution

Browse

Use

Return

说明:菜单运行后,可通过单击菜单来执行菜单项,当需要退出菜单时,可单击菜单以外的地方或按“Esc”键退出菜单。另外,本例设计的菜单只能进行单项选择。

例 8.4 将上例更改为多项选择菜单,当退出菜单时,可执行选中的菜单项内容。

Clear

&& 保存用户的多项选择结果变量

Define Popup FF Title "数据库表浏览" From 2,20;

* 依次定义各菜单项及其各菜单项的快捷键、显示名和提示等

Define Bar 2 of FF Prompt "各部门职工情况表" Key Ctrl +D,"D";

Define Bar 3 of FF Prompt "例 6.2 Welcome 表单" Key Ctrl + W,"^W";

Define Bar 4 of FF Prompt "在浏览窗口浏览表 Contribution" Key Ctrl +C,"^C" ;

* 定义各菜单项对应的动作均为执行一个 sel 过程

* 激活弹出式菜单 FF

* 循环判断用户选择了几个菜单项

Do Case

Do Form 职工情况

Do Form 各部门职员情况表

Do Form Welcome

Do Con

EndFor

Return

* 当退出菜单时,确认用户选择的菜单项

Procedure Sel

&& 每次选择均将以前的选择清除,只保留最后一次多项选择

For I=1 to 4

If Mrkbar("FF",I)=. T. Then && 测试菜单 FF 的第 I 项是否选中,函数说明见后

M=M+Str(I,1) && 保存多项选择结果


```
EndIf
```

```
EndFor
```

```
Return
```

* 菜单第四项要执行的过程

```
Procedure con
```

```
Select 0
```

```
Use Contribution
```

```
Browse
```

```
Use
```

```
Return
```

函数说明：

函数格式：

```
Mrkbar("<弹出式菜单名>",<弹出式菜单项>)
```

功能：当<弹出式菜单名>指定的第<弹出式菜单项 N>项菜单为选中状态时，函数返回真值，<弹出式菜单项>为数值表达式。

注意：在定义菜单项时，菜单项的表示是用 Bar <数值表达式>，所以菜单项在程序中访问时是通过数值访问的。

例 8.5 编制一个多选择弹出式菜单程序，要求如下：

(1) 菜单的选项为表 Employee 的字段；

(2) 当用户选择菜单中的字段，并退出菜单时，自动在浏览窗口浏览表 Employee 选定字段的内容。

源文件 Menu3. prg 的源程序代码如下：

```
Clear
```

```
Use Employee
```

```
M="" &&. 保存用户的多项选择结果变量
```

* 定义名为"Emp",标题为"选择 Employee 字段",菜单项为 Employee 字段的多选弹出式菜单

```
Define Popup Emp Title "选择 Employee 字段" From 2,20 Message "库表浏览";
```

```
Margin MultiSelect Prompt Structure
```

* 定义各菜单项对应的动作均为执行一个 sel 过程

```
On Selection Popup Emp do sel
```

* 激活弹出式菜单 Emp

```
Activate Popup Emp
```

* 循环判断用户选择了几个菜单项

* 先将选定的字段依次添加到 Browse 命令字符串中

* 当添加第一个字段时，字段名与 Fields 字句间是空格，以后添加的字段是逗号分隔符

```
FD="Browse Fields"
```

```
For J=1 To Len(M) Step 2
```

```
If J=1 Then
```

```

        FD=FD+Space(1)+Field(Val(Substr(M,J,2)))      && 函数说明见后
    Else
        FD=FD+" "+Field(Val(Substr(M,J,2)))
    EndIf
EndFor
* 利用宏替代函数 & 来执行 FD 字符串中的 Browse 命令
* 但为了防止没有任何选择时发生的错误,用 If 语句判断用户是否选择了菜单
* 这里也可以用 Len(M) # 0 作为判断条件
If "Browse Fields" # FD then
    &.FD
EndIf
Use
return
* 当退出菜单时,确认用户选择的菜单项
Procedure Sel
    M=""      && 每次选择均将以前的选择清除,只保留最后一次多项选择
    For I=1 To FCount()      && FCount()返回当前表字段个数
        If Mrkbar("Emp",I)=. T. Then      && 测试菜单 FF 的第 I 项是否选中
            M=M+Str(I,2)      && 保存多项选择结果,其中每两个字节保存一个选择结果
        EndIf
    EndFor
Return

```

函数说明:

函数格式:

Field(<数值表达式>)

功能:返回当前表第<数值表达式>位置的字段名称。

本例中先用 Substr(M,J,2)函数依次取出选中项的字符串表示形式,然后通过 Val () 函数将其转化为数值型,再通过 Field ()函数返回选中项的字段名称。

习 题

一、选择题

1. 菜单设计器中不包括的命令是()。
 - A. 插入
 - B. 删除
 - C. 生成
 - D. 预览
2. 有一个菜单文件 Main. mnx,要运行该菜单的方法是()。
 - A. 执行命令 Do Main. mnx
 - B. 执行命令 Do Menu Main. mnx
 - C. 先生成菜单程序文件 Main. mpr,再执行命令 Do Main. mpr

- D. 先生成菜单程序文件 Main. mpr,再执行命令 Do Menu Main. mpr
3. 作用()可在菜单设计器中自动复制一个与 VFP 系统菜单一样的菜单。
- A. “插入菜单项”命令 B. “快速菜单”命令
C. “插入栏”命令 D. “生成”命令
4. 关于快速菜单,下面说法正确的是()。
- A. 基于 VFP 主菜单,用于添加用户所需要的菜单项
B. 快速菜单的运行速度较快
C. 可以为菜单项指定快速访问方式
D. “快捷菜单”的另一种说法
5. 用菜单设计器设计的菜单保存后,其生成的文件扩展名为()。
- A. . SCX 和 . SCT B. . MNX 和 . MNT
C. . FRX 和 . FRT D. . PJX 和 . PJT
6. 菜单项名称为“Help”,要为菜单项设置热键 H,则在名称中设置为()。
- A. Alt+H B. \<H C. Alt+\< D. <H
7. 有连续的两个菜单项,名称分别为“保存”和“删除”,要用分隔线将这两个菜单分组,方法是()。
- A. 在“保存”菜单项名称前面加上“\<”,即“保存\<”
B. 在“删除”菜单项名称前面加上“\<”,即“删除\<”
C. 在两个菜单项之间添加一个菜单项,并在名称栏中输入“\<”
D. A 或 B 两种方法均可
8. 为顶层表单添加下拉式菜单,定义菜单时的正确做法是()。
- A. 在“菜单设计器”环境下,选择系统菜单“显示/菜单选项”命令,然后在“菜单选项”对话框中,选中“顶层表单”复选框
B. 在“菜单设计器”环境下,选择系统菜单“显示/常规选项”命令,然后在“常规选项”对话框中,选中“顶层表单”复选框
C. 在“菜单设计器”环境下,选择系统菜单“显示/菜单选项”命令,然后在“菜单选项”对话框中,选中“顶层表单”单选按钮
D. 在“菜单设计器”环境下,选择系统菜单“显示/常规选项”命令,然后在“常规选项”对话框中,选中“顶层表单”单选按钮
9. 将一个预览成功的菜单存盘,再运行该菜单,却不能执行。这是因为()。
- A. 没有放到项目中 B. 没有生成菜单程序
C. 要用命令方式 D. 要编入程序
10. 设计菜单要完成的最终操作是()。
- A. 创建主菜单及子菜单 B. 指定各菜单任务
C. 浏览菜单 D. 生成菜单程序
11. 用菜单设计器生成的基本步骤为()。
- (1)进行菜单设计 (2)打开菜单设计器 (3)生成菜单程序
(4)保存菜单定义 (5)运行菜单程序

- A. (4)→(2)→(3)→(1)→(5) B. (2)→(1)→(4)→(3)→(5)
C. (3)→(2)→(4)→(1)→(5) D. (2)→(1)→(3)→(4)→(5)

12. 打开菜单设计器,系统菜单将自动增加一个()菜单

- A. “常规” B. “运行” C. “设计” D. “菜单”

13. 下面()可以恢复系统菜单的默认配置。

- A. Set Default To B. Set Sysmenu Default To
C. Set Sysmenu To Default D. Set Menu To Default

二、填空题

1. 菜单设计器窗口中的_____可以用于上下级菜单之间的切换。

2. 在菜单设计器窗口中要为菜单项定义快捷菜单,可以使用_____对话框。

3. 在定义菜单时,菜单项的结果有填充名称或菜单项、过程、_____、和子菜单 4 种选择。

4. 用户应用程序中引用菜单时,必须使用_____作为扩展名。

三、应用题

1. 设计一个下拉式菜单,主菜单至少有“文件”、“编辑”、“显示”和“退出”,要求有如下功能:

在“文件”菜单下有“打开”与“关闭”命令;

在“编辑”菜单下有“复制”、“粘贴”、“剪切”等命令;

在“显示”菜单下,能够通过选择显示不同的表单;

2. 在上章习题 2 的多表单应用程序中的顶层表单中,添加菜单,在菜单中实现原顶层表单的功能。

3. 设计一个弹出式多项选择菜单,从中可以选择当前表的字段,并在浏览窗口显示选中字段的记录内容。



第 9 章

报表与标签设计

9.1 报表设计与应用

报表主要包括两部分内容:报表的数据源和报表的布局。报表的数据源是报表数据的来源,它通常是自由表、数据库表、视图、查询或临时表等。经过第三章的学习,我们已经知道查询和视图已经对数据进行了筛选、排序和分组。所以,利用查询和视图作为数据源创建的报表,要比直接利用表作为数据源创建的报表更具有针对性。

报表的总体布局可分为列报表、行报表、一对多报表和多栏报表:

1. 列报表:其主要特征是报表的每个字段占一列,字段名在页面上方,字段与其数据在同一列,每行一条记录。这种报表布局比较常见,如分组/汇总报表、财政报表、各类清单、销售总结等都可以使用这种布局格式;

2. 行报表:其主要特征是报表的每个字段占一行,字段名在数据的左侧,字段与其数据在同一行,一个记录占用报表的多行位置。这种布局格式适合于各类清单、列表等;

3. 一对多报表:报表一条记录由一对多关系生成。打印时在父表中取出一条记录并打印后,必须将子表中与之相关的多条记录取出并打印。这种布局格式适合于基于表间一对多关系的货运清单、会计报表、票据等;

4. 多栏报表:报表拥有多栏记录,可以是多栏行报表,也可以是多栏列报表。

Visual FoxPro 为我们提供了输出报表的不同方式:

1. 利用带有 To Printer 子句的命令,将命令的结果直接输出打印出来,这是一种最简单的报表输出方式。

2. 采用编写程序方式设计产生报表,这种形式可以设计出任何形式的报表,但是设计报表程序的过程太复杂,工作量也较大。

3. 利用 Visual FoxPro 生成报表工具创建设计各种格式的报表。如:使用报表向导创建报表、使用快速报表创建简单规范的报表、使用报表设计器创建自定义的报表。

在通常情况下,利用报表工具就可以设计出大多数常用的报表形式,所以,本节主要介绍如何利用报表工具产生报表。

9.1.1 报表向导

启动表单向导的几种方式:

1. 单击系统菜单“文件/新建”命令,或单击工具栏上的“新建”按钮,在打开“新建”对话框中选择“报表”,然后单击“向导”按钮;

2. 单击系统菜单“工具/向导/报表”命令;

3. 打开“项目管理器”,在“全部”或“文档”选项卡中选择“报表”后,单击“新建”按钮。在弹出的“新建报表”对话框中单击“报表向导”按钮。

下面通过一个示例来说明利用报表向导创建报表的步骤:

例 9.1 利用报表向导设计职工基本信息报表,要求输出身份号、姓名、性别、年龄、民族、工作时间、部门名称、婚否、工资等字段信息,并保存为 EmpRep1。

分析:由于以上字段来源于两个表,所以在应是一个一对多报表格局。

设计步骤:

1. 启动报表向导。

单击系统菜单“文件/新建”命令,在打开“新建”对话框中选择“报表”,然后单击“向导”按钮,弹出报表“向导选取”对话框;

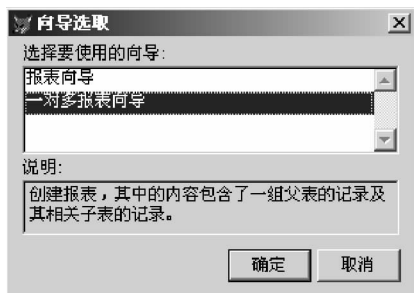


图 9.1 报表“向导选取”对话框

由上面的分析知:报表数据来源于表 Department 和表 Employee 这两个表,因此,在报表的“向导选取”对话框中,选择“一对多报表向导”后,单击“确定”按钮,弹出如图 9.2 所示的“一对多报表向导”的步骤 1 对话框。



图 9.2 “一对多报表向导”步骤 1

2. 选择父表及其字段。

在“一对多报表向导”的步骤 1 对话框中,选择表 Department 为父表,并用鼠标双击其中的部门名称字段,使其出现在选定字段列表框中,选择结果如图 9.2 所示。完成父表的选择后,单击“下一步”按钮,弹出如图 9.3 所示的“一对多报表向导”的步骤 2 对话框。



图 9.3 “一对多报表向导”步骤 2

3. 选择子表及其字段。

在“一对多报表向导”的步骤 2 对话框中，选择表 Employee 为子表，并依次将身份证号、姓名、性别、年龄、民族、工作时间、婚否、工资移到选定字段列表框中，选择结果如图 9.3 所示。完成子表的选择后，单击“下一步”按钮，弹出如图 9.4 所示的“一对多报表向导”的步骤 3 对话框。



图 9.4 “一对多报表向导”步骤 3

4. 为父_子表建立关系。

在“一对多报表向导”的步骤 3 对话框中，选择父表字段为部门代码，选择子表字段为部门，在父表 Department 和子表 Employee 之间建立关系。单击“下一步”按钮，弹出如图 9.5 所示的“一对多报表向导”的步骤 4 对话框。



图 9.5 “一对多报表向导”步骤 4

5. 排序。

在“一对多报表向导”的步骤 4 对话框中,在选定“升序”单选按钮的前提下,将“部门名称”添加到选定字段列表框中,让报表按“部门名称”的升序排序。单击“下一步”按钮,弹出如图 9.6 所示的“一对多报表向导”的步骤 5 对话框。

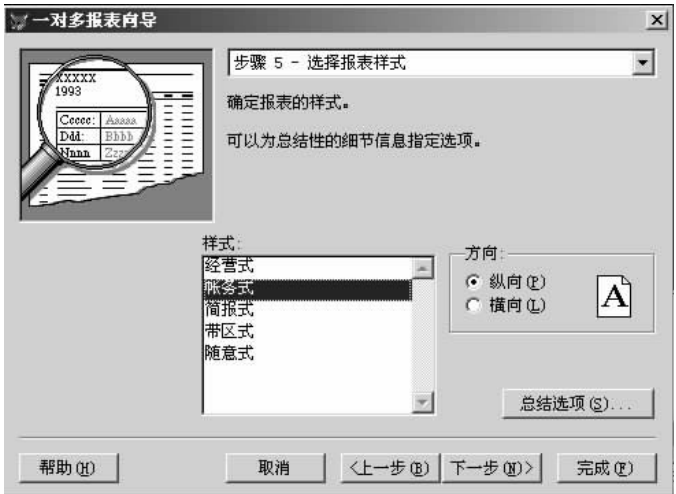


图 9.6 “一对多报表向导”步骤 5

6. 选择样式。

在“一对多报表向导”的步骤 5 对话框中,选择需要的报表样式,如“账务式”,选择样式时,对话框左上角可预览样式效果。单击“下一步”按钮,弹出如图 9.7 所示的“一对多报表向导”的步骤 6 对话框。

7. 完成。

在“一对多报表向导”的步骤 6 对话框中,设置报表标题为“职工基本信息报表”,单击“预览”可以观察报表效果,若满意,单击“完成”按钮,弹出“另存为”对话框,输入报表文件



图 9.7 “一对多报表向导”步骤 6

名为:EmpRep1,然后单击“确定”保存报表。若对由报表向导产生的报表不满意,则可返回到上一步骤中进行修改,或在报表设计器中打开该报表进行修改。关于利用报表设计器对报表进行修改的方法将在后面介绍。

9.1.2 打开报表设计器

报表设计器窗口打开后,在 VFP 系统菜单中自动增加一个报表菜单,并且系统菜单的显示、格式、文件等菜单中也增加了一些命令或改变一些命令的功能,以方便用户利用报表设计器设计报表。打开报表设计器有如下方式:

1. 菜单方式。

(1) 若是新建一个报表,则单击系统菜单“文件/新建”命令,在弹出的“新建”对话框中选择“报表”后,单击“新建”按钮,弹出“报表设计器”;

(2) 若是在“报表设计器中”打开一个报表,则单击系统菜单“文件/打开”命令,在弹出的“打开”对话框中,选择要打开的报表文件名,单击“打开”按钮。

2. 命令方式。

命令格式一:

Create Report <文件名>

功能:创建一个名为<文件名>的报表文件,并在报表设计器窗口中打开该文件。

命令格式二:

Modify Report <文件名>

功能:如果名为<文件名>的报表文件存在,则在报表设计器窗口中打开它;否则,创建它。

3. 项目管理器方式。

(1) 在项目管理器中新建报表:

在项目管理器的“全部”或“文档”选项卡中,选择报表后,单击“新建”按钮,在弹出的

“新建报表”对话框中单击“新建报表”按钮；

(2) 在项目管理器中打开报表：

在项目管理器的“全部”或“文档”选项卡中，选择需要打开的报表文件后，单击“修改”按钮。

9.1.3 报表设计器窗口介绍

9.1.3.1 报表设计器窗口介绍

“报表设计器”窗口是一个报表的设计区域，如图 9.8 所示，在其中可以设置或格式化一些报表控件。“报表设计器”窗口默认划分为 3 个带区：页标头(Page Header)、细节(Detail)和页脚(Page Footer)，每个带区的底部显示分隔栏。但是，一个完整的报表设计器窗口分为七个带区，它显示了对对象显示或打印的具体位置。将对象放到某个带区，可以产生与该带区相对应的项目和内容。

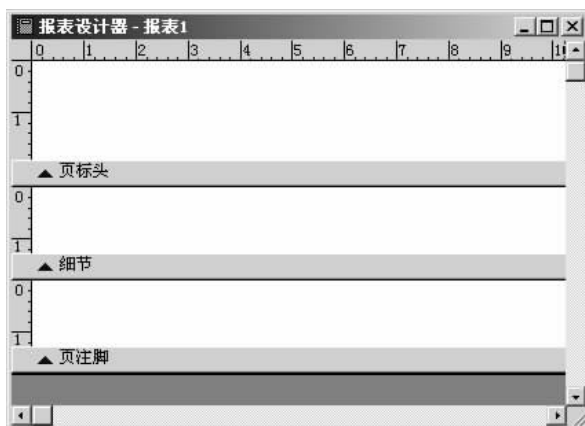


图 9.8 报表设计器

1. 标题带区(Title):标题带区的内容只在报表首页的开始处打印一次。在此带区中可以设置报表标题、日期、页数、公司标志、围绕标题的边框、修饰报表标题的控件等；

2. 页标题(Page Header):页标题的内容只在报表的每一页开头打印一次，若是首页则在标题带区后打印。在此带区中可以设置报表列标题、日期、页码控件等；

3. 细节带区(Detail):细节带区是报表的主体，用于输出数据库的记录，一般在该带区设置数据库字段。运行报表时，将每一条符合条件记录的所有“细节带区”内设置的字段值打印一次；

4. 页注脚带区(Page Footer):页注脚带区的内容在每页的最底部打印，一般在该带区设置日期、页码、分类总计和说明信息等；

5. 总结带区(Summary):总结带区的内容只在报表的末尾打印一次，一般在该带区设置各种数据的总计或平均值等信息；

6. 组标头带区和组注脚带区:用于分组报表，组标头带区的内容在每个分组开始时打印一次，组注脚带区的内容在每个分组结束时打印一次。一般在该类带区设置分组字段，分隔线等；

7. 列标头带区和列注脚带区:列标头带区和列注脚带区主要用于分栏报表,一般在带区设置栏标题等。

分栏报表分栏数的设置:单击系统菜单“文件/页面设置”命令,在弹出的“页面设置”对话框中,将“列数”设置为大于1的值,并调整适当的“间隔”后,单击“确定”,列标头带区和列注脚带区,才会在报表设计器中出现。

调整报表带区高度

将鼠标指针指向某带区分隔条,出现上下双箭头时,按住左键上下拖动分隔条即可改变报表带区高度。

9.1.3.2 报表菜单

当报表设计器窗口为当前窗口时,在菜单栏中会自动出现“报表”菜单,如图9.9所示,在该菜单下有如下菜单项:

1. 标题/总结:指定报表中是否包含一个“标题带区”或一个“总结带区”以及是否换页;

2. 数据分组:完成报表细节带区数据的数据分组打印选择,单击该菜单项,将弹出一个“数据分组”对话框;

3. 变量:在报表中建立、修改、删除报表的内存变量,或改变变量计算顺序。系统利用报表内存变量,保存在打印过程中完成的一些计算结果;

4. 私有数据工作期:为报表所具有的数据环境设定或不设定为私有数据工作期方式。在私有数据工作期方式下,每个报表具有各自的数据环境。

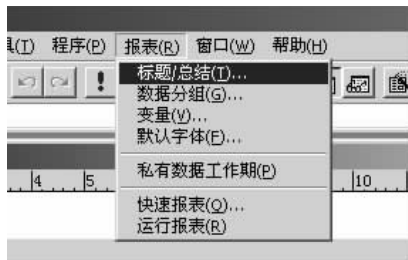


图 9.9 报表菜单

9.1.3.3 报表设计器工具栏

“报表设计器”工具栏,如图9.10所示,该工具栏中有数据分组、数据环境、报表控件工具栏、设色板工具栏和布局工具栏五按钮,单击这些按钮可以打开相应的对话框或工具栏等。



图 9.10 报表设计器图



图 9.11 报表控件

9.1.3.4 报表控件工具栏

“报表控件”工具栏,如图9.11所示,该工具栏中有选定对象、标签、域控件、线条、矩形、圆角矩形、图片/ActiveX 绑定控件等按钮,其中线条、矩形、圆角矩形是用来绘制图形线条的,域控件用来添加字段、变量或表达式等,其他控件的用法与表单设计器中的对应控件相似。

9.1.4 快速报表

与快速表单,我们也可以利用快速报表来设计一个简单的报表,如果对使用快速报表

产生的报表不满意,还可以在报表设计器窗口中打开它,对其进行修改。下面通过一个例子来介绍快速报表的创建过程。

例 9.2 利用快速报表功能,为表 Employee 设计一个有身份号、姓名、性别、年龄、民族、工作时间、部门、婚否、工资等字段的报表,报表名为 EmpRep2。

设计步骤:

1. 在命令窗口执行命令:Modif Report EmpRep2 后,报表 EmpRep2 会在报表设计器窗口中打开;

2. 单击系统菜单“报表/快速报表”命令,弹出“打开”对话框,在该对话框中选择打开表 Employee 后,系统自动弹出“快速报表”对话框,如图 9.12 所示;

3. 该对话框的“字段布局”,有两种选择,一种是列方式,一种是行方式,这里选择列方式,报表的布局为列报表布局方式;选中“标题”复选框,将字段名作为列标题;选中“添加别名”复选框,将报表的数据源自动添加到数据环境中;单击“字段”按钮,弹出“字段选择器”对话框;

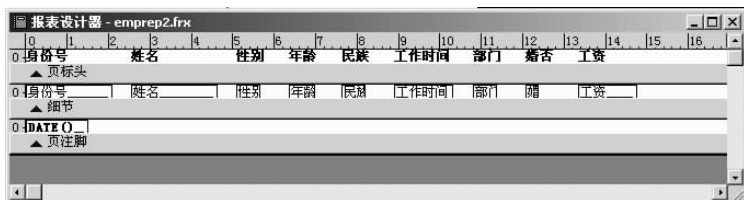


图 9.12 快速报表



图 9.13 字段选择器

4. 在“字段选择器”对话框中,选择题目要求的字段,如图 9.13 所示。在缺省情况下,包括除“通用”字段外的全部字段;单击“确定”按钮,返回到“快速报表”对话框,单击“确定”按钮,完成快速报表的创建,如图 9.14(a)所示;



(a)

5. 单击工具栏上的“打印预览”按钮,打开快速报表的预览窗口,如图 9.14(b)所示。



(b)

图 9.14 生成“快速报表”

9.1.5 修改用快速报表产生的报表

由上例的运行结果可以看出,该报表的界面过于简单,下面将上例进行一些修改,操作步骤如下:

1. 在命令窗口中用命令:Modify Report EmpRep2 在报表设计器窗口中,打开要修改的 EmpRep2 报表。

2. 添加报表标题。

(1) 单击系统菜单“报表/标题/总结”命令,弹出的“标题/总结”对话框,设置如图 9.15 所示,单击“确定”后,在报表窗口中增加标题和总结两个带区;

(2) 在“报表控件”工具栏中,选择“标签”控件,然后在标题带区单击,并在输入:职工基本信息报表;

(3) 用鼠标单击选择“职工基本信息报表”标签,然后单击系统菜单“格式/字体”命令,在弹出的“字体”对话框中设置标签文本的各种属性。

3. 添加表格线。

(1) 选择线条控件,在页标头带区的顶部和底部各画一条横线,每列画一条竖线;

(2) 选择线条控件,在细节带区底部添加一条横线,每列添加一条竖线,注意与页标头带区的竖线对直。

4. 单击工具栏上的“打印预览”按钮,打开快速报表的预览窗口,如图 9.16 所示。



图 9.15 标题/总结

身份证号	姓名	性别	年龄	民族	工作时间	部门	婚否	工资
20050001	张丽平	女	30	汉	07/12/95	D1	Y	1357.25
20050002	陈国庆	男	41	汉	03/22/85	H1	Y	1851.60
20050003	方世玉	男	25	汉	08/05/00	I2	Y	1042.38
20050004	王国真	女	22	汉	09/23/02	I1	N	989.57
20050005	吴鹏	男	27	满	07/12/98	I3	Y	1128.12
20050006	孙宏伟	男	50	回	08/06/75	I2	Y	2055.87
20050007	李莉	女	40	汉	05/12/86	I3	Y	1769.95
20050008	杨剑雄	男	20	汉	08/23/03	D1	N	984.12

图 9.16 生成的快速报表经过简单修改后的效果

9.1.6 设计报表

9.1.6.1 设置报表数据环境

在“报表设计器”窗口为当前窗口时,单击系统菜单“显示/数据环境”命令,弹出“数据环境设计器”窗口,在该窗口中设计与报表相关的数据源,其设置方法与在表单的数据环境设计器中设置数据源的方法相似。

9.1.6.2 创建报表变量

如果要在报表中操作数据或显示计算结果,可使用报表变量,使用报表变量,可以计算各种值,而且可以利用这些值计算其他相关值。设置报表环境变量的步骤如下:

1. 首先将要建立环境变量的报表在“报表设计器”窗口中打开;

2. 单击系统菜单“报表/变量”命令,将弹出“报表变量”对话框,如图 9.17 所示;

3. 在“变量”框中输入一个变量名,如 Number;

4. 在“要存储的值”框中输入一个变量或其他的表达式;也可以单击后面的按钮,利用弹出的“表达式生成器”创建一个表达式;还可以从“计算”选项中,选择其中的一种作为要存储的值的內容;

5. 如果需要,也可以为定义的报表变量设定一个初始值;

6. 在“重置”列表框中选择一项,如报表尾、页尾、分组字段等,分别表示在报表尾时将变量重置初始值,在页尾时将变量重置,在分组字段尾时将变量重置;

7. 选定“报表输出后释放”复选框,表示在报表运行结束后释放报表变量。

9.1.6.3 添加报表控件

1. 标签控件:用于显示静态文本。

添加标签控件方法:在“报表控件”工具栏中选择“标签”按钮后,移动鼠标到报表设计器窗口中的合适区域位置,单击鼠标左键,出现“ ”插入点后输入文本信息。

2. 绘图控件。

绘图控件包括线条、矩形和圆角矩形。在“报表控件”中选择相应的绘图控件,然后在报表的某个需要设置图形带区的相应位置拖拽鼠标,产生相应的图形。

3. 域控件。

域控件的添加和布局是报表设计的核心,用于打印表或视图中的字段、变量和表达式的计算结果。

(1)添加域控件:

方法一:在“数据环境设计器”窗口,选择要使用的表或视图,然后把相应的字段直接拖拽到报表指定的带区中。

方法二:单击“报表控件”工具栏中的“域控件”按钮,然后在报表带区的指定位置上单击鼠标,系统将显示一个“报表表达式”对话框,如图 9.18 所示。设置完成该对话框的相关内容后,则在鼠标单击位置出现一个“域控件”。

(2)“报表表达式”对话框的使用:

①输入表达式内容:即设置域控件的数据源:

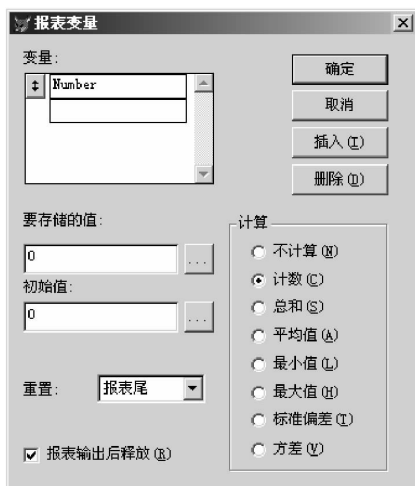


图 9.17 “报表变量”对话框



图 9.18 “报表表达式”对话框

用户可以直接在“报表表达式”对话框中的表达式文本框中输入字段名、变量名或表达式,也可以单击文本框右侧的“...”按钮,通过打开的“表达式生成器”对话框设置“域控件”的数据源。

在报表设计过程中,直接双击“域控件”或单击快捷菜单的“属性”命令,都将弹出“报表表达式”对话框,在该对话框中,如果添加的是可计算字段,当单击“计算”按钮时,弹出“计算字段”对话框,如图 9.19 所示。该对话框的用法与“报表变量”中的“重置”和“计算”的用法相似。

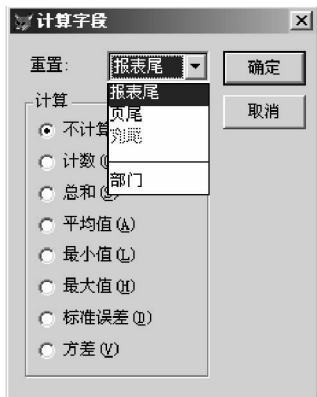


图 9.19 “计算字段”对话框



图 9.20 “格式”对话框

② 设置域控件的格式:

对“报表表达式”对话框的“格式”文本框进行格式设置,实际上是设置“表达式”的输出格式。此处格式符的使用规则与数据库中字段格式符的使用规则相同。

除了直接在“格式”文本框中直接输入格式外,还可以通过单击“格式”文本框右侧的“...”按钮,在弹出的“格式”对话框中设置数据格式,如图 9.20 所示。在该对话框中选定所需的数据类型,然后选取“编辑选项”区域的有关选项。

③ 设置域控件的打印条件:

单击“报表表达式”对话框中“打印条件”按钮,则弹出如图 9.21 所示的“打印条件”对话框。当选择“打印重复值”为“否”选项时,则当该“域控件”对应的值有重复值时,将不打印重复值,否则打印。

在“有条件打印”区域中有三个复选框:

- 选择“在新页/列的第一个完整信息带内打印”复选框,则表示在同一页或同一列中不打印重复值,换页或换列后遇到第一条新记录时打印重复值。该复选框只在“打印重复值”选择“否”时才有效;

- “当此组改变时打印”选中,表示当右边的下拉列表中显示的分组发生变化时,打印重复值。该复选框只在“打印重复值”选择“否”并有分组时

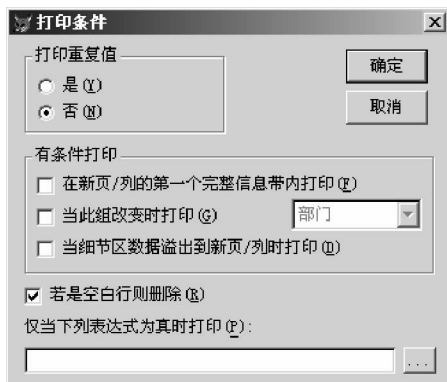


图 9.21 “打印条件”对话框

才有效；

• “当细节区数据溢出到新页/列时打印”复选框选中，表示当细节带区的数据溢出到新页或新列时打印重复值。

④设置域控件位置：

在“报表表达式”对话框的“域控件位置”区域有三个单选项：

- 浮动：
- 相对于带区顶端固定：使域控件相对于周围域控件的大小浮动；
- 相对于带区底端固定：使域控件在“报表设计器”中保持固定位置，并维持其相对于带区底端的位置；

⑤溢出时伸展：

当字段内容较长时，可选择该复选框，这样可让字段显示到报表底部，以显示字段全部内容。

4. OLE 对象。

在 VFP 中经常用到对象链接与嵌入 (OLE) 技术，一个 OLE 对象可以是图片、声音、文档等，这里主要讨论向报表中添加图片，其主要步骤如下：

(1) 添加图片：

在“报表控件”工具栏中单击“图片/ActiveX 绑定控件”按钮，在报表的一个带区内单击并拖动鼠标拉出一个图文框，松开鼠标时将弹出“报表图片”对话框，如图 9.22 所示。

(2) 设置“报表图片”对话框：

①设置图片来源：

图片来源可以是文件，也可以是字段。

- 文件：选择“文件”选项，在其右侧的文本框中直接输入一个图形文件的位置和名称，或单击文本框右边的“...”按钮，打开“打开”对话框，通过该对话框来选择一个图片文件；
- 字段：选择“字段”选项，在其右侧的文本框中直接输入表的通用型字段的字段名，如图 9.22 所示。或单击文本框右边的“...”按钮，弹出“选择字段/变量”对话框，通过该对话框选择通用型字段。

②调整图片：

在“假如图片与图文框的大小不一致”区域，设置图片的显示方式：

- “裁剪图片”：裁剪图片与控件大小相同时显示图片；
- “缩放图片，保留形状”：自动缩放图片，以在控件区域完整地、不变形地显示图片；
- “缩放图片，填充图文框”：自动缩放图片，使图片填满整个控件区域，在这种情况下，图片纵横比例可能会改变，从而引起图片的变形。

③图片居中：

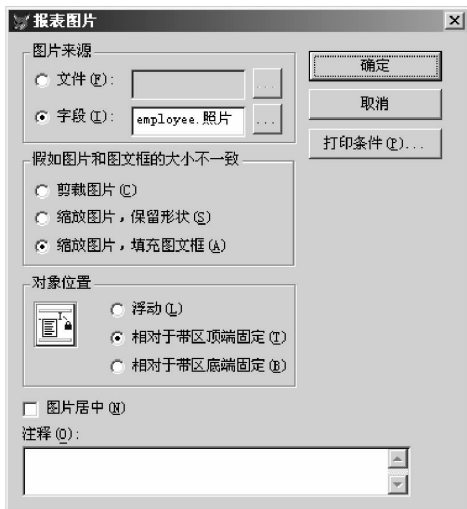


图 9.22 “报表图片”对话框

选择该复选框时,图片会自动在控件区域内居中。

该对话框的其他设置与“报表表达式”对话框对应内容的设置方法相似。

9.1.6.4 设计分组报表

一个报表可以设置一个或多个数据分组,组的分隔基于分组表达式。对报表进行数据分组后,报表会自动显示出“组标头”和“组注脚”两个带区。

但在设计分组报表前,需要对报表数据源的相关字段进行排序或索引,例如:如果要按“性别”分组,则在分组前需要对数据源的“性别”字段进行索引。

设置报表分组的主要步骤如下:

1. 单击系统菜单“报表/数据分组”令,弹出“数据分组”对话框,如图 9.23;

2. 在“分组表达式”框内键入一行分组表达式,或者通过单击“...”按钮,并在弹出的“表达式生成器”对话框中创建分组表达式;

3. 如果要创建一个多级数据分组报表,则可在下一行输入一个新的分组表达式,它表示分组在按第一行分组表达式进行分组的前提下,按第二行表达式进行第二级分组;若需要再进行下一级分组,其分组方法同上。例如:要对报表按“部门”进行一级分组,然后在一级分组的前提下对每一组的数据按“性别”进行第二级分组。这里需要说明的是当对报表进行多级数据分组时,要对数据源先按分组的级别顺序排序或索引;



图 9.23 “数据分组”对话框

4. 组属性主要用于指定如何分页,在“组属性”区域中有四个复选框,根据不同的报表类型,有的复选取框不可用;

- “每组从新的一列上开始”复选框:表示当一组的内容打印完毕后,新的一组内容是否从下一列开始打印。分组报表为多栏报表时,该选项有效;

- “每组从新的一页上开始”复选框:表示当一组的内容打印完毕后,新的一组内容是否从新的一页开始打印;

- “每组的页号重新从 1 开始”复选框:表示当一组的内容打印完毕后,新的一组内容是否从新的一页开始打印,并将页号重置为 1;

- “每页都打印组标头”复选框:表示当分组的内容分布在多页上时,在每一页是否都打印组标头;

- “小于右值时组从新的一页上开始”微调器设置:在微调器中输入的数值是打印组标头时,组标头距页面底部的最小距离。如果不希望组标头在打印时,因页面剩余的空间不够,而出现在页面上只有组标头而没有组内容的情况,该数值应该包括组标头和至少一行记录及页脚的距离。

5. 单击“确定”,然后可在报表设计器中向“组标头”和“组注脚”带区中设置相应的控件。

在分组设置完成后,需要设置报表的索引,其设置步骤是:

- 1. 单击系统菜单“显示/数据环境”命令,打开数据环境设计器窗口;
- 2. 在数据环境设计器窗口内,单击快捷菜单中的“属性”命令,弹出“属性对话框”如

图 9.24 所示;



图 9.24 数据源索引设置

- 3. 在对象组合框中选择“Cursor1”对象,在“全部”或“数据”选项卡中选择“Order”属性,从下拉式列表框中选择已经打开的索引标识。

由以上介绍可知,设置报表索引,首先应该建立索引标识,然后通过数据环境设计器窗口设置“Cursor1”对象的“Order”属性。分组设置既可以在报表索引之前设置,也可以报表索引之后设置。

9.1.6.5 分栏报表

在设计报表的过程中,如果报表横向所占的页面较少,则将其设计为多栏报表较合适。多栏报表的设计步骤如下:

- 1. 单击系统菜单“文件/页面设置”,弹出如图 9.25 所示的“页面设置”对话框。



图 9.25 “页面设置”对话框

2. 在对话框的“列”区域,把“列数”微调器的值调整为栏目数,例如列数为2,则将整个页面平均分成两部分,调整列之间的间隔值,如间隔为0.8。

3. 设置打印顺序:默认为“自上而下”的打印顺序,用户也可将其设置为“自左向右”的打印顺序。

9.1.7 报表输出

1. 页面设置:在“页面设置”对话框中除了可以设置分栏外,还可设置左边距和打印设置。

(1) 设置左边距:在“左页边距”框中输入“左边距”数值,页面布局将按新的页边距显示;

(2) 打印设置:单击“打印设置”按钮,弹出“打印设置”对话框。可以从“大小”列表中选择纸张大小。默认的打印方向为纵向,若要改变纸张的方向,可从“方向”区选择横向,再单击“确定”按钮,完成页面设置。

2. 设计时预览报表。

- 单击系统菜单“显示/预览”命令;
- 在“报表设计器”中单击鼠标右键,从弹出的快捷菜单中选择“预览”命令;
- 单击“常用”工具栏中的“打印预览”按钮。

3. 打印报表。

单击系统菜单“文件/打印”命令,弹出“打印”对话框,根据对话框的提示设置完成后单击“确定”按钮,打印报表。

4. 程序中调用报表。

预览命令格式:

Report Form <报表文件名> <Preview> [范围] [For <条件>]

功能:预览报表,其中报表中的数据为指定范围内满足条件的数据。

打印输出报表:

Report Form <报表文件名> <To Printer> [范围] [For <条件>]

功能:打印报表,其中报表中的数据为指定范围内满足条件的数据。

例 9.3 利用报表向导设计职工基本信息报表,要求如下:

1. 输出身份号、姓名、性别、年龄、民族、工作时间、部门名称、婚否、工资等字段信息,要求部门名称不是部门代码;
2. 报表布局为2栏的分栏列表;
3. 一级分组按民族分组,二级分组按部门分组;
4. 每条记录前有一序号,当一级分组为下一分组时,重新排列序号;
5. 将报表保存为 EmpRep3。

分析:在设计报表时,尽管报表中的数据涉及到表 Department 和表 Employee 这两个表,但由于不是按部门分组,所以不能利用报表向导,将其设计为一对多形式的报表。而应该首先设置与两个表相关的视图,视图文件内容包含题目要求的相关字段。并在视图中设置相应的排序。

设计步骤：

1. 在表 Department 和表 Employee 所在的数据库中建立包含题目要求的视图 Repview。

(1) 在命令窗口中输入并执行命令:Modify DataBase Zy,打开表 Department 和表 Employee 所在的数据库 Zy;

(2) 在命令窗口中输入并执行创建视图命令创建 Repview 视图,命令如下:

```
Create View Repview As Select Employee. 身份号,Employee. 姓名,;  
Employee. 性别,Employee. 年龄,Employee. 民族,Employee. 工作时间,;  
Department. 部门名称,Employee. 婚否,Employee. 工资;  
From Department,Employee Where Department. 部门代码=Employee. 部门;  
Order by Employee. 民族,Department. 部门名称
```

利用上述命令创建的视图文件 Repview 包含表 Employee 的身份号、姓名、性别、年龄、民族、工作时间、婚否、工资等字段和表 Department 的部门名称字段,并且在视图中数据以民族和部门名称排序。

当然创建数据库视图文件 Repview,也可以利用视图设计器窗口设计,在这儿就不重复介绍了。

2. 在命令窗口中输入并执行命令:Modify Report EmpRep3,弹出报表 EmpRep3 的“报表设计器”窗口;

3. 以视图文件 Repview 为数据源,利用快速报表建立功能,建立含有视图文件全部字段的快速列报表,其建立步骤如下:

(1) 单击系统菜单“报表/快速报表”命令,弹出“快速报表”对话框,如图 9.26 所示;



图 9.26 “快速报表”对话框



图 9.27 “字段选择器”对话框

(2) 在该对话框中,单击“字段”按钮,弹出“字段选择器”对话框,在“字段选择器”对话框中的“来源于表”的下拉式组合框中选择:Repview,并在“选定字段”列表框中添加视图 Repview 中的全部字段,如图 9.27 所示,单击确定按钮返回到“快速报表”对话框;

(3) 在“快速报表”对话框中,选择“字段布局”中的“列布局”按钮,其他设置如图 9.26 所示,完成快速报表设置后单击“确定”按钮。

4. 页面设置:页面设置应该在报表带区设计之前完成,便于以后设置带区内控件,为其提供比较准确的定位。

(1) 单击系统菜单“文件/页面设置”，弹出“页面设置”对话框，如图 9.28 所示；

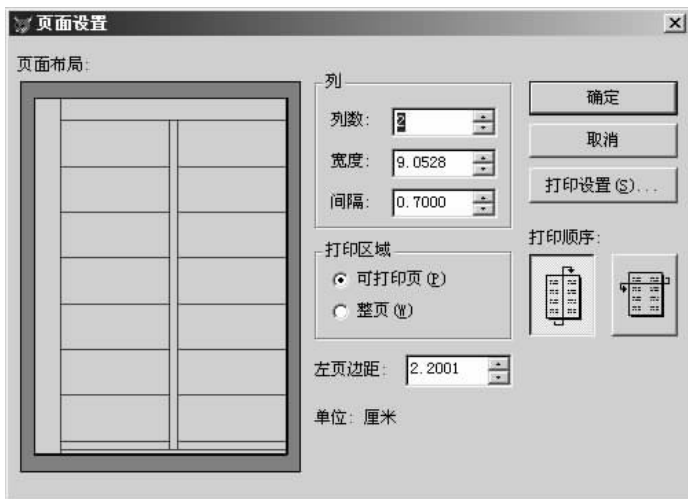


图 9.28 “页面设置”对话框

(2) 将报表设置为两栏的分栏报表：在“列数”框中输入：2；

(3) 设置栏之间的间隔：在“间隔”框中输入：0.7；

(4) 设置打印顺序为从左到右；

(5) 设置左页边距：在“左页边距”框中输入：2.2；

(6) 单击“打印设置”，在弹出的“打印设置”对话框中设置纸张大小为 A4 纸。

以上具体设置情况见图 9.28。

5. 设置“标题/总结”带区。

(1) 单击系统菜单“报表/标题/总结”命令，弹出的“标题/总结”对话框，设置参见如图 9.15 所示；

(2) 在“标题”带区设置标题：

①在“报表控件”工具栏中，选择“标签”控件，然后在标题带区单击，并输入：职工基本信息表；

②用鼠标单击选择“职工基本信息表”标签，然后单击系统菜单“格式/字体”，在弹出的“字体”对话框中设置标签文本的各种属性，如字体为宋体、字号为小一、字型为粗体等；

③在标题带区中添加一个日期域控件：在“报表控件”工具栏中，选择“域控件”控件，然后在标题带区中的合适位置单击，在弹出的“报表表达式”对话框的“表达式”框中输入：DATE()；

④在标题下添加一个线条：在“报表控件”工具栏中，选择“线条”控件，然后在标题带区中标题的下方单击并横向拖曳，产生一水平线条；单击系统菜单“格式/绘图笔/2 磅”，将线条设置为 2 磅粗的线条；

⑤调整“标题”带区的高度，及该带区中各控件的位置。

(3) 设置“总结”带区：

在“报表控件”工具栏中，选择“标签”控件，然后在总结带区单击，在光标处输入：制表

人:张三,并设置该标签的字体为宋体、字号为小五、字型为粗体等,其设置方法与标题的设置方法相似。

“标题”与“总结”带区的设置效果如图 9.31 所示。

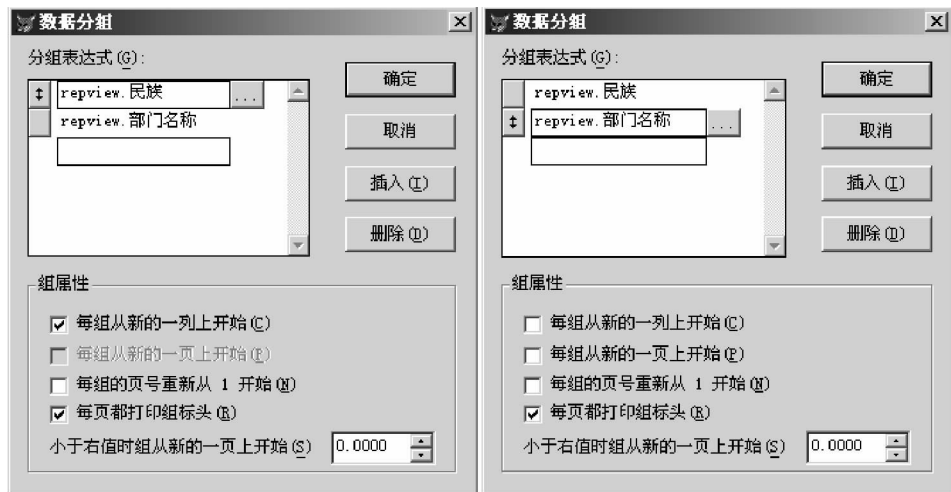


图 9.29 “数据分组”对话框

6. 设置页注脚。

(1) 在页注脚带区中添加两个标签和一个域控件;

(2) 设置标签:标签的设置方法与标题的设置方法相似,其设置效果见图 9.31 所示;

(3) 双击域控件,在弹出的“报表表达式”对话框中的“表达式”框中输入: _PAGENO,让该控件显示报表的页码,其字体设置同标签字体设置。

7. 设置分组:由于本报表的数据源是视图 Repview,该视图在创建的过程中就已经按民族和部门名称进行了排序,所以这里可以直接进行报表分组设置。

(1) 单击系统菜单“报表/数据分组”命令,弹出“数据分组”对话框;

(2) 在该对话框中输入一组分组表达式: repview. 民族,和二级分组表达式: repview. 部门名称,并设置各级分组的相应组属性设置,如图 9.29 所示;

(3) 分别调整“组标头 1”和“组标头 2”的带区宽度,并从“细节”带区中将民族标签和民族域控件复制到“组标头 1”的带区中的相应位置;从“细节”带区中将部门名称标签和部门名称域控件复制到“组标头 2”的带区中的相应位置,见图 9.31 所示;

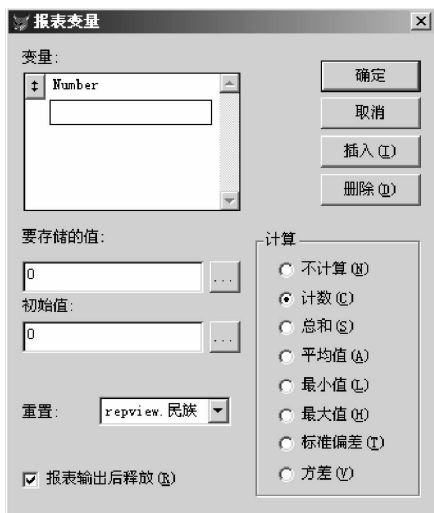


图 9.30 “报表变量”对话框

8. 设置每一条记录前的序号。

设置序号要用到计数,所以应该设置一个计数变量,然后将序号域控件与该变量绑定,具体步骤如下:

(1) 设置变量 Number:

① 单击系统菜单“报表/变量”命令,弹出“报表变量”对话框;

② 在该对话框中“变量”列表框中,输入一个变量: Number;

③ 在“计算”选择方法中选择“计数”;

④ 在“重置”列表框中选择 Repview. 民族,表示变量按民族重置,设置如图 9.30 所示;

⑤ 单击“确定”完成报表变量设置。

(2) 设置序号域控件:

在“细节”带区中添加一个域控件,在弹出的“报表表达式”对话框的“表达式”框中输入报表变量 Number 的名称,然后设置该控件的字体,其设置效果如图 9.31 所示。



图 9.31 “报表设计器”对话框

9. 单击“预览”按钮察看报表的设计效果,当然在设计过程中也可以察看设计过程中每一步的设计效果。

10. 单击保存,保存报表,如果需要打印报表,则单击“打印”对报表进行打印。

9.2 标签的设计与使用

在实际应用中数据的输出形式往往是以标签卡片的形式输出,例如个人名片,邮件标签,借书卡片等。标签是采用多列报表布局,为匹配特定标签纸而对列作特定设置的报表。其设计过程与报表相似,但比报表要简单的多。

9.2.1 标签向导

1. 启动标签向导。

启动标签向导的方法与启动报表向导的方法相似,这里只介绍通过单击“新建”按钮来建立的方法,其他方法请参阅启动报表向导的方法。

单击工具栏上的“新建”按钮,在弹出的“新建”对话框中选择“标签”,单击向导,弹出“标签向导”步骤 1 对话框,如图 9.32 所示。



图 9.32 “标签向导”步骤 1 对话框

2. 在“标签向导”步骤 1 对话框中选择数据库 Zy 的表 Employee 后,单击“下一步”按钮,弹出“标签向导”步骤 2 对话框,如图 9.33 所示。



图 9.33 “标签向导”步骤 2 对话框

3. 在“标签向导”步骤2对话框中选择标签的型号,如图9.33所示;单击“下一步”按钮,弹出“标签向导”步骤3对话框,如图9.34所示。

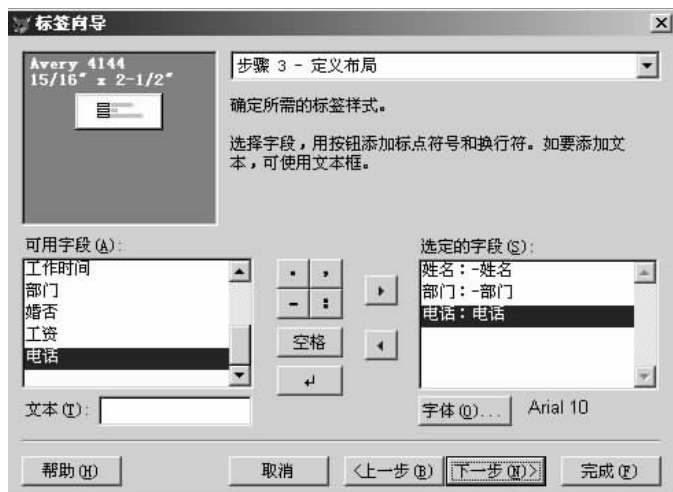


图 9.34 “标签向导”步骤3对话框

4. 在“标签向导”步骤3对话框中,依次添加标签内容。

添加步骤如下:

- (1) 在文本框中输入:“姓名:”,单击“添加”按钮,将输入的文本标签添加到选定的定做列表框中;
- (2) 选择“空格”再单击“添加”按钮,在文本标签后添加一个空格;
- (3) 在“可用字段”列表框中双击“姓名”字段,将该字段添加到空格后面;
- (4) 选择“↵”再单击“添加”按钮,在文本标签后添加一个回车符;

按照以上步骤分别设置其他内容,设置效果如图9.34所示,完成设置后,单击“下一步”按钮,弹出“标签向导”步骤4对话框,如图9.35所示。



图 9.35 “标签向导”步骤4对话框

(5) 在“标签向导”步骤 4 对话框中,选择姓名字段为排序字段,其设置见图 9.35,单击“下一步”按钮,弹出“标签向导”步骤 5 对话框,在该对话框中可以预览标签的设置效果,单击“完成”按钮,在弹出的“保存”对话框中将标签保存为:RepLab。

5. 若对标签设置不满意,还可通过“标签设计器”对标签进行调整,其调整过程与报表相似。

9.2.2 标签设计器

通过标签设计器可以直接创建一个新标签文件,或打开一个已经存在的标签文件,然后在打开的标签设计器中对标签进行设计,其设计过程与报表设计相似。新建和打开标签文件的方法与新建与打开报表文件的方法相似,这里仅仅只以通过“新建”按钮为例来说明利用标签设计的设计过程。具体步骤如下:

1. 单击工具栏上的“新建”按钮,在弹出的“新建”对话框中,选择“标签”类型,然后单击“新建文件”按钮,弹出如图 9.36 所示的“新建标签”对话框。



图 9.36 “新建标签”对话框

2. 在“新建标签”对话框中“选择标签布局”列表框中选择新建标签的布局,如图 9.36 所示,单击“确定”按钮,系统显示标签设计器窗口,如图 9.37 所示。

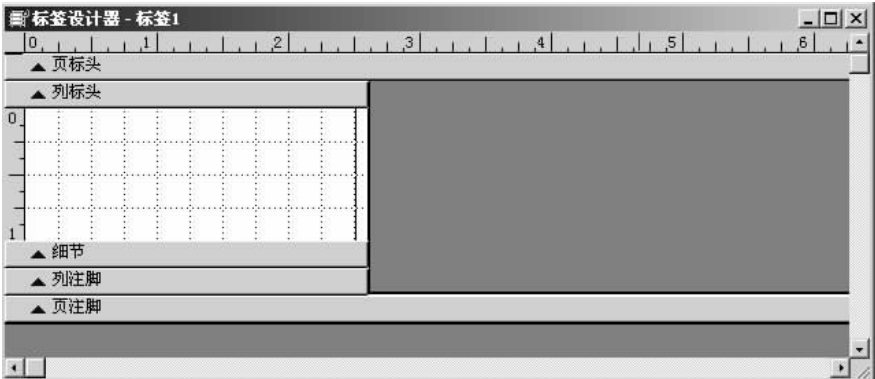


图 9.37 “标签设计器”对话框

3. 标签设计器窗口打开后,在系统菜单上也会显示与报表设计器窗口打开时一样的菜单系统,而且标签设计器的常规操作也与报表设计器完全相同,只是比报表更简单,在此就不再详细介绍。

9.2.3 标签的创建与输出命令

标签的创建、输出命令与报表的创建、输出命令相似,如果将报表的创建与输出命令中的 Report 命令动词或 Report 保留字,更改为 Label 即为相应的标签创建与输出命令。

创建标签命令格式:

Create Label <文件名>

修改命令格式:

Modify Label <文件名>

预览标签命令格式:

Label Form <报表文件名> <Preview> [范围] [For <条件>]

打印输出标签命令格式:

Label Form <报表文件名> <To Printer> [范围] [For <条件>]

由于标签的操作与报表的操作相似,所以这儿就不再举例了,但需要注意的是产生的报表文件的扩展名为 Frx,而产生的标签文件的扩展名为 Lbx。

习 题

一、选择题

1. 报表的作用是()。
A. 为了显示数据表中的数据
B. 为了查询数据表中的数据
C. 为了建立一个临时表
D. 为了打印数据、统计和分析结果
2. 在报表设计器中,不能关闭的三个基本带区是()。
A. 页标头、细节和页注脚
B. 标题、页标头和细节
C. 页标头、组标头和页注脚
D. 页标头、细节和组注脚
3. 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

二、填空题

1. 多栏报表的栏目数可以通过_____来设置。
2. 报表标题要通过_____控件定义。
3. 报表由_____和_____两个基本部分组成。
4. 报表文件的扩展名是_____。
5. 对报表进行数据分组时,报表会自动包含_____和_____带区。
6. 报表将字段控件称为_____。
7. 创建报表有_____,_____,_____三种方法。
8. 设计好的报表既可以在打印机上输出,也可以通过_____浏览。
9. 使用“快速报表”创建报表,仅需_____和设定报表布局。
10. 在 VFP 中用_____命令打印报表,若要预览报表则用_____命令。

三、应用题

1. 根据表 Student. dbf, 设计并打印一个报表, 要求如下:

- (1) 含有系别、姓名、性别、籍贯字段;
- (2) 报表布局采用多栏行报表;
- (3) 先以系别分组, 当系别相同时, 再以性别分组;
- (4) 要求每个记录前有一个序号;
- (5) 当系别不同时, 序号重新开始, 同时换页;
- (6) 当性别不同时换栏;
- (7) 在报表开始处显示日期, 结束时显示制表人。

2. 请利用 Student 数据库的三个表 Collegedepartment. dbf、Student. dbf、Reportcard. dbf, 建立一个学生完整信息的报表, 具体要求如下:

- (1) 学生信息要完整, 但不能重复, 如显示系部名称后, 就不需要显示系别代码;
- (2) 报表采用多栏列报表布局;
- (3) 按系部分组;
- (4) 分组设计各科平均成绩; 分组分科计算各科不及格人数;

3. 请利用 Student 数据库设计一个标签, 标签中包含系部名称、姓名及“XX 学院学生会”字样, 形式如信封。



第 10 章

开发应用程序

在前面我们已经介绍了如何设计数据库、表单、菜单、报表等,本章将介绍开发数据库应用程序的方法和步骤,即介绍如何将设计好的数据库、表单、菜单、报表等相互分离的系统组件在项目管理器中连编成一个完整的应用程序,最后编译成一个扩展名为 app 的应用文件或 exe 的可执行文件。

10.1 应用项目综合实践

在开发应用程序时,利用“项目管理器”将应用程序的各部分组织成一个应用项目,并对该项目进行测试。

10.1.1 系统开发基本步骤

利用 Visual FoxPro 开发的系统一般由下列几个组成部分:

1. 一个或多个数据库;
2. 用户界面:如欢迎界面、输入表单、显示表单、工具栏、菜单等;

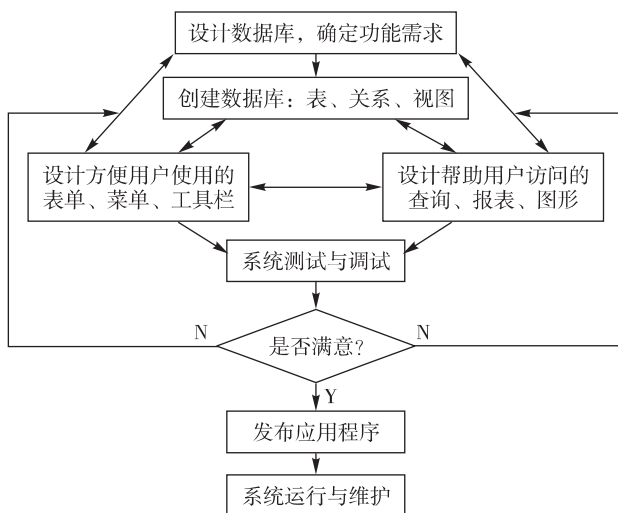


图 10.1 应用程序开发过程

3. 事务处理:如查询、统计、计算等,允许用户检索出自己需要的数据;
4. 输出形式与界面:如浏览、报表、标签等;
5. 主程序:设置应用程序系统环境和起始点。

利用 VFP 开发应用程序的过程如图 10.1 所示,在该图中描述了开发过程中各个阶段之间的相互关系

10.1.1.1 建立应用程序目录结构

一个完成的应用程序,一般都会涉及到多种类型的文件,如数据库、表、查询、视图、表单、菜单、报表、位图、程序等。若将这些文件全部都放在一个文件夹下,将会给今后的修改、维护带来不便。因此,需要建立一个层次清晰的目录结构。图 10.2 就是本书示例应用程序的一个目录结构,在此可以将以前创建的数据库文件、表文件、索引文件移到 Data 目录下,将表单移到 Forms 目录下,将图形图像移到 Graphics 目录下,依此类推将不同类型的文件移到相应的目录下。

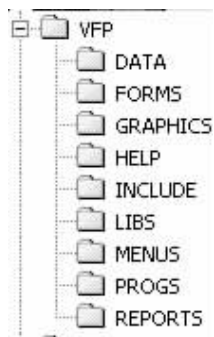


图 10.2 应用程序目录结构

10.1.1.2 用项目管理器组织应用系统

在设计应用程序时,应仔细设计每个组件应提供的功能以及各组件之间的关系,一个设计良好的应用程序一般要为用户提供一个菜单、一个或多个表单供数据输入和显示,在菜单和表单设计时,还要为菜单命令或表单的某些事件添加一些命令代码,提供其特定的功能,同时还要保证数据的完整性和安全性。另外还要提供查询和报表的输出功能。

“项目管理器”是开发 VFP 应用程序的一个工作平台,在开发应用程序时,通常都是先建立一个项目管文件,如本书中,我们就是先建立一个 Zygl.pjx 项目文件,然后在项目管理器中打开该文件,并在该项目中建立相应的数据库、表、查询、视图、菜单、表单、报表等。当然也可以先独立建立数据库、表、查询、视图、菜单、表单、报表等,然后再将这些模块添加到项目中。利用项目管理器组织应用系统的步骤如下:

1. 在项目管理器中打开已经的项目,或通过创建新项目在项目管理器中打开新项目;
2. 在项目管理器中新建数据库、表、查询、视图、表单、菜单、报表等,也可以将已经设计好的上述内容添加到项目管理器中;
3. 在项目管理器中自下而上的调试各个模块,即先调试可以独立运行的模块单元,再调试调用它们的模块单元;对各个模块进行独立调试有助于错误代码的正确定位与修改。

10.1.1.3 加入项目信息

单击系统菜单“项目/项目信息”命令,打开如图 10.3 所示的“项目信息”对话框,在“项目”选项卡中可以输入以下信息:

- 开发者的信息:如姓名、单位、地址等;
- “本地目录:定位项目的主目录,如 D:\VFP 目录;
- “调试信息”复选框:选择在应用程序文件中是否包含调试信息。选择此复选框,对

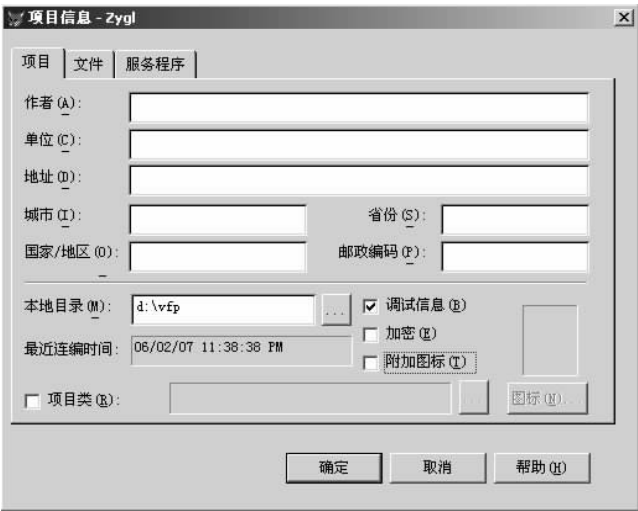


图 10.3 “项目信息”对话框的“项目”选项卡

程序调试帮助较大,但增加了程序的大小,所以在交付用户前进行最后连编时应清除此复选框;

- “加密”复选框:选择是否对应用程序加密。若加密,则对应用程序反求源代码将会非常困难;
- “附加图标”复选框:选择是否为生成的文件选择自己的图标。通过选择该复选框,可以指定当应用程序最小化时使用什么图标。

“项目信息”对话框中的“文件”选项卡:在该选项卡中可以查看到项目管理器中的所有文件,而不论文件在什么位置。文件按文件名的字母顺序排列,单击各栏目的标题可以改变文件的排序方式。在该选项卡中选择某一行文件后,单击鼠标右键,可通过弹出的快捷菜单对选择的文件进行三种设置:排除/包含、设置主文件、代码页。



图 10.4 “项目信息”对话框的“文件”选项卡

10.1.2 连编应用程序

对各个模块分别进行调试后,就要对整个项目进行联合调试并编译,在 VFP 中称其为连编项目。

10.1.2.1 设置文件的“排除”与“包含”

1. 文件排除与包含的概念。

将一个项目编译成一个应用程序时,所有项目包含的文件将组合成为一个单一的应用程序文件。在项目连编后,在项目中标记为“包含”的文件将成为只读文件。如果应用程序中包含需要用户修改的文件,则应该将该文件标记为“排除”。排除文件依然是应用程序的一部分,VFP 依然可以跟踪它,将其看成是项目的一部分。但这些文件没有在应用程序的文件中编译,所以用户可以更新它们。

作为通用规则:可执行程序,如表单、报表、查询、菜单和程序文件在应用程序中为“包含”,而数据文件则为“排除”。但是,也可以根据应用程序的需要包含或排除文件。

2. 将文件设置为“排除”或“包含”的操作。

将文件设置为“排除”或“包含”的操作有两种方式,下面分别介绍:

(1) 在“项目信息”对话框的“文件”选项卡中,通过快捷菜单对选择的文件进行设置,也可用鼠标直接在文件包含列的标记符号上单击以修改文件的设置。在该选项卡的“包含”列中文件的标记为“☒”时,表示该文件为包含;标记为“☐”时,表示该文件为排除;

(2) 在“项目管理器”窗口中,通过快捷菜单对选择的文件进行设置,或通过系统菜单“项目/排除(包含)”命令对选择的文件进行设置。在项目管理器中当文件为包含时,文件前没有标记;当文件为排除时,文件前有“O”标记。

10.1.2.2 设置主文件

主文件是应用程序的入口,其任务是设置应用程序的起始点,初始化环境、显示初始的用户界面、控制事件循环,当退出应用程序时,恢复原始的开发环境。

当用户运行应用程序时,将首先启动主文件,然后主文件再依次调用所需要的应用程序及其他组件。所有应用程序必须包含一个主文件。在 VFP 中程序文件、菜单、表单、查询都可以作为主文件,但通常将应用程序最上层的文件设置为主文件。在项目管理器中主文件名将以黑体显示;在“项目信息”对话框的“文件”选项卡的“包含”列中主文件的标记为“**■**”。

主文件设置的方法有两种:

1. 在“项目信息”对话框的“文件”选项卡中,通过快捷菜单对选择的文件进行设置;
2. 在“项目管理器”窗口中,通过快捷菜单对选择的文件进行设置,或通过系统菜单“项目/设置主文件”命令对选择的文件进行设置。

只有是“包含”的文件,才能够被设置为主文件。当重新设置一个主文件时,原先设置的主文件自动解除。标记为主文件的文件不能排除。

10.1.2.3 项目连编

1. 项目连编的概念。

当一个项目建立好各个模块文件,并对各个模块分别进行调试后,在项目运行前要对

它们进行连编。项目连编是让 VFP 系统对项目的整体性能进行测试的方法,其最终结果是将所有在项目中引用的文件(除去标记为排除的文件外),合成为一个应用程序文件;然后将应用程序软件、数据文件以及其他排除的项目文件一起交给最终用户使用。

2. 在进行项目连编时,应该注意并了解以下几个问题。

(1)项目连编时,VFP 将分析对所有文件的引用,并自动将所有的隐式文件包含在项目中。若通过用户自定义的代码引用任何一个其他文件,项目连编会分析所有包含及引用的文件,并将引用的文件添加到项目管理器中,当再次查看时就会在项目管理器中看到这些引用的文件;

(2)在项目中若涉及到图(.bmp 或 .msk)文件的引用时,需要采用手工的方式将其添加到项目中;

(3)连编项目不能自动包含采用“宏替换”引用的文件,若项目中涉及引用“宏替换”文件,则需要采用手工的方式添加并包含这些引用文件。

3. 利用项目管理器进行项目连编的步骤。

(1)选择设置为主文件的文件,单击“连编”按钮,弹出“连编选项”对话框,如图 10.5 所示。

(2)在该对话框的“操作”选项中选择“重新连编项目”;

(3)在“选项”中选择“重新编译全部文件”复选框,将重新编译项目中的全部文件。否则,只重新编译上次连编后修改过的文件;

(4)在“选项”中选择“显示错误”复选框,在连编过程中若发生错误,系统自动将这些错误信息存放在一个<项目名称>.err 的文件中,在完成连编后,自动弹出错误信息内容窗口,可以立即查看错误文件,同时错误数量显示在状态栏中,如图 10.6 所示,该窗口中错误信息文件名为 Zygl.err。



图 10.5 “连编选项”对话框

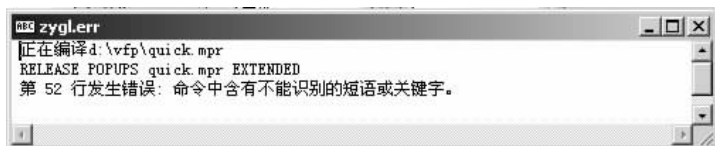


图 10.6 Zygl.err 错误信息窗口

(5)单击“确定”按钮,系统开始连编。

若在项目连编过程中发生错误,必须纠正或排除错误,并且反复进行“重新连编项目”,直到连编成功为止。

10.1.2.4 连编应用程序

1. 试运行项目。

项目连编成功后,在建立应用程序之前应该试运行该项目,试运行的方式有两种:

(1)在“项目管理器”中选择主文件后,单击“运行”按钮,运行项目;

(2)在命令窗口中用 DO 命令执行主文件,如本项目的主文件是 EmplMenu.mnx,则

可在命令窗口中执行: Do EmplMenu. mpr, EmplMenu. mpr 是菜单文件 EmplMenu. mnx 的菜单程序文件。

2. 应用程序连编的结果文件形式。

在项目运行正确的前提下,就可以最终连编成一个应用程序文件。应用程序文件包括项目中所有“包含”的文件,应用程序连编的结果有两种文件形式:

(1)应用程序文件(.app):该文件需要在 VFP 环境中运行,生成的文件较小;

(2)可执行文件(.exe):该文件可以直接在 Windows 环境下运行,但需要与两个 VFP 动态连接库(Vfp6r.dll 和 Vfp6enu.dll)连接,这两个库与应用程序一起构成了 VFP 所需要的完整运行环境。但连编成可执行文件所需的两个库只有 VFP 专业版才可用。利用“安装向导”为可执行文件创建安装盘,安装盘中带有应用程序所需的所有文件,如连编前在项目管理器中“排除”的文件。

3. 连编应用程序的步骤。

(1)在“项目管理器”中单击“连编”按钮,弹出“连编选项”对话框;

(2)若在该对话框中的“操作”中选择“连编应用程序”,则生成一个扩展名为 .app 的应用程序文件;若选择“连编可执行文件”,则生成一个扩展名为 .exe 的可执行文件;

(3)在该对话框中的“选项”中选择所需要的选项后,单击“确定”按钮。

10.1.2.5 连编的其他选项

1. 在“连编选项”对话框的“操作”选项中选择“连编 COM DLL”后进行连编,将使用项目文件中的类信息创建一个扩展名为 dll 的动态链接库。

2. “连编选项”对话框中的“版本”按钮:该按钮只有在选择“连编可执行文件”和“连编 COM DLL”时,才会被激活。在该按钮为激活状态时,单击该按钮将弹出如图 10.7 所示的“EXE 版本”对话框,在该对话框中可指定版本号和与版本类型相关的信息。

3. “连编后运行”复选框:选择该复选框表示连编完成后,运行生成的连编结果文件。

4. “重新生成组件 ID”复选框:该复选框只有在选择“连编可执行文件”和“连编 COM DLL”后,并且已经连编包含 OLEPublic 关键字的程序时,才会被激活。“重新生成组件 ID”安装并注册包含在项目中的自动服务程序(Automation Server)。选定时,该选项指定在连编程序时生成新的 GUID(全局唯一标识)。只能创建和注册“类”菜单“类信息”对话框中标识为“OLE Public”的类。



图 10.7 “EXE 版本”对话框

10.1.2.6 运行应用程序

1. 运行 app 应用程序。

app 应用程序文件只有在 VFP 窗口中运行,其运行方式除可以通过系统菜单“程序/运行”命令来运行外,还可以利用 DO 命令在命令窗口中运行。

2. 运行 exe 可执行文件。

可以像 app 应用程序运行方式一样运行,还可以在 Windows 窗口中直接双击 exe 可执行文件图标运行。

10.1.2.7 连编项目的命令格式

格式一:

Build Project <项目名>

功能:相当于在“连编选项”对话框的“操作”选项中选择“重新连编项目”后进行连编,不产生连编结果文件。

格式二:

Build App <应用程序文件名> From <项目名>

功能:相当于在“连编选项”对话框的“操作”选项中选择“连编应用程序”后进行连编,产生扩展名为 app 的应用程序文件。

格式三:

Build Exe <可执行文件名> From <项目名>

功能:相当于在“连编选项”对话框的“操作”选项中选择“连编可执行文件”后进行连编,产生扩展名为 exe 的可执行文件。

格式四:

Build Dll <动态链接库名> From <项目名>

功能:相当于在“连编选项”对话框的“操作”选项中选择“连编 COM DLL”后进行连编,产生扩展名为 dll 的动态链接库。

10.1.3 主程序设计

主文件的任务是初始化环境、显示初始的用户界面、控制事件循环,在退出应用程序时,恢复原始的开发环境。

10.1.3.1 初始化环境

应用程序初始化环境的理想方法是将开发系统的初始环境设置保存起来,并在主文件的启动代码中为应用程序建立特定的环境设置。

从当前环境中截取开发系统初始环境设置命令的方法与步骤是:

首先,单击系统菜单“工具/选项”命令,弹出“选项”对话框,在按下 Shift 键的同时,单击对话框的“确定”按钮后,在命令窗口中则会显示系统环境的设置命令;

然后,从“命令”窗口中,将系统环境设置命令复制并粘贴到主文件中。

除环境设置外,在初始化时通常还需要设置初始化变量、默认目录、打开数据库、表及索引等。

10.1.3.2 显示初始的用户界面

初始的用户界面可以是菜单,还可以是一个表单或其他用户组件。一般在显示菜单或表单前,会出现一个启动屏幕或注册对话框。

在主文件中,可以使用命令运行一个菜单或表单,来作为初始化用户界面。

10.1.3.3 控制事件循环

在设置应用程序环境,并运行初始化界面后,需要建立一个事件循环来等待用户的交

互动作。控制事件循环的命令格式：

格式一：

Read Events

格式二：

Clear Events

功能：在执行 Read Events 命令后，主文件中的所有的处理过程全部挂起，VFP 开始处理鼠标单击、键入等事件，直到执行 Clear Events 命令

在启动了 Read Events 命令后，应用程序将处在所有最后显示的用户界面元素的控制之下，等待用户的操作。若在主文件中没有 Read Events 命令或其等价命令，则应用程序在原开发环境中可以正常运行，但在其他环境下则会出现运行不正常的情况。

在启动事件循环时，还要利用 Clear Events 命令退出事件循环，通常 Clear Events 命令设置在退出按钮或菜单命令中。

10.1.3.4 组织主程序文件

若应用程序使用一个程序文件(.prg)作为主文件，应保证该程序文件中可控制应用程序的主要任务。

例：Main.prg 主程序文件代码：

Set Default To D:\Vfp

Clear

Clear All

Close All

Do Form Welcome && 显示初始用户界面，

Read Event && 建立事件循环

* 在退出之前，恢复环境设置

Set SysMenu To Default

Set Talk On

Set Safty On

Close All

Clear ALL

Clear Windows

Clear Event

Cancel

说明：在 Welcome 表单的 UnLoad 事件中添加代码：Do Emplmenu. Mpr，在表单关闭后运行 Emplmenu 菜单程序。

10.2 应用程序生成器

开发 VFP 应用程序时，还可先利用应用程序向导生成一个项目和一个 VFP 应用程序框架，再打开应用程序生成器，添加已经生成的数据库、表、查询、视图、表单、报表等组件。

10.2.1 使用应用程序向导

10.2.1.1 使用应用程序向导创建项目和应用程序框架

1. 单击工具栏上的“新建”按钮,在新建对话框中选择“项目”选项后,单击“向导”按钮,弹出“应用程序向导”对话框。

2. 在弹出的“应用程序向导”对话框中,在“项目名称”文本框中输入项目名称,如:职工信息管理;在“项目文件”文本框中输入项目文件的存放位置和项目文件名,如:D:\Vfp\职工信息管理.pjx;选择“创建项目结构”复选框时,若在“项目文件”文本框中输入的文件夹不存在,则自动创建,如图 10.8 所示。

3. 单击“确定”按钮,“应用程序向导”自动调用所需要的各种应用程序生成器,并为应用程序生成一个目录和项目结构,如图 10.9 所示。

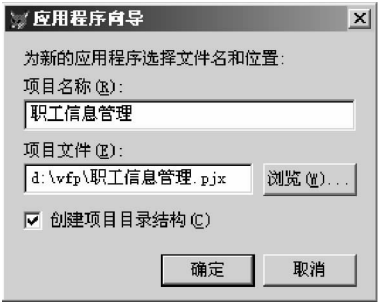


图 10.8 “应用程序向导”对话框



图 10.9 应用程序向导新建项目及应用程序框架

10.2.1.2 应用程序框架

应用程序框架包含了所有必需的以及可选的元素,目的是使所开发的应用程序更便捷、有效。在运行“应用程序向导”后,会在指定的文件夹下(如:D:\Vfpapp\),产生一些文件和文件夹,这些文件和文件夹就组成了应用程序框架。应用程序框架可以自动完成下列任务:

- 1. 提供启动和清理程序,其中包含负责保存和恢复环境状态的程序;
- 2. 显示菜单和工具栏;
- 3. 帮助用户确定应用程序功能、用户输入数据的方式、应用程序的外观等。

10.2.2 应用程序生成器

10.2.2.1 应用程序生成器的功能

通过“应用程序向导”创建并生成一个项目的同时,会在“项目管理器”中打开这个新项目,与“项目管理器”一同打开的还有“应用程序生成器”,如图 10.10 所示。“应用程序生成器”和应用程序框架一起提供下列功能:



图 10.10 应用程序生成器

1. 添加、编辑或删除与应用程序相关的组件,如数据库、表、菜单、表单、报表、程序等;
2. 设置表单和报表的外观样式;
3. 加入常用的应用程序元素,包括启动画面、“关于”对话框、“收藏夹”菜单、“用户登录”对话框和标准工具栏;
4. 提供应用程序的作者和版本信息等。

10.2.2.2 打开应用程序生成器

除了在利用“应用程序向导”时自动打开“应用程序生成器”外,“应用程序生成器”还可以用以下方式打开:

1. 在项目管理器上,单击鼠标右键,选择快捷菜单上的“生成器”菜单;
2. 单击系统菜单“工具/向导/全部”命令,在弹出的“向导选取”对话框中选择“应用程序生成器”;
3. 按 Alt+F2。

10.2.2.3 应用程序生成器

由图 10.10 可以看出“应用程序生成器”包括“常规”、“信息”、“数据”、“表单”、“报表”和“高级”六个选项卡。

1. “常规”选项卡:如图 10.10 所示。

(1)名称:指定应用程序的名称,将显示在标题栏和“关于”对话框中,并在整个应用程序中使用。

(2)图像:指定显示在启动画面和“关于”对话框中的图像文件的文件名。

(3)应用程序类型:指定应用程序的运行方式。

- 正常:生成 .app 应用程序。

• 模块:应用程序将被添加到已有的项目中,或将被其他程序调用。该应用程序将在菜单系统中添加一个主菜单选项,并作为另一个应用程序的组件运行。

• 顶层:生成.exe 可执行程序

(4)常用对话框:通过复选框选择应用程序中是否包含下列内容:

- 显示屏幕:显示启动画面。
- 快速启动:用“快速启动”表单提供对应用程序文档和其他磁盘文件的访问。
- 关于对话框:是否需要“关于”对话框。
- 用户登录:是否提示用户进行口令登录,并管理各个用户的参数选择信息。

(5)图标:用于指定显示在正常应用程序的主桌面上、顶层应用程序的顶层表单框架上,以及没有指定特定图标的表单的标题栏上的图标。

2. “信息”选项卡:如图 10.11 所示。

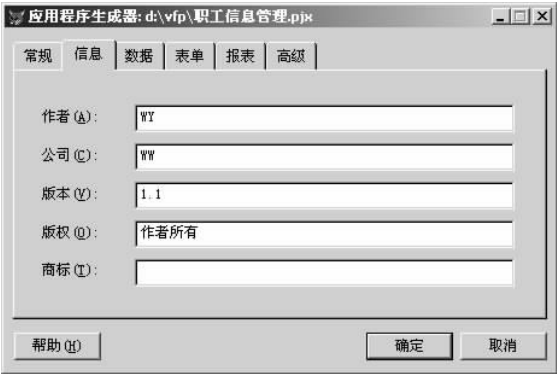


图 10.11 应用程序生成器的“信息”选项卡

- 作者:指定应用程序作者名称。
- 公司:指定编写或使用应用程序的公司名称。
- 版本:指定应用程序版本。
- 版权:给出版权信息。
- 商标:指定商业或服务标志。

3. “数据”选项卡:如图 10.12 所示。

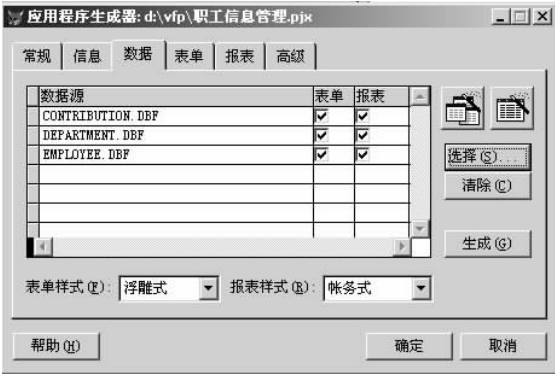


图 10.12 应用程序生成器的“数据”选项卡

• 数据库向导:帮助创建应用程序所需的数据库。当利用该向导创建好数据库后,在表格中将列出新数据库中的表。

- 表向导:帮助创建应用程序所需的表。
- 选择:选择要在应用程序中使用的已有的数据库或表。
- 清除:删除表格中列出的指定的表。
- 生成:根据所选的表按照指定的样式生成表单或报表。
- 表单样式:从下拉式列表框中选择表单样式。
- 报表样式:从下拉式列表框中选择报表样式。

该选项卡中每次单击“生成”按钮,生成的表单或报表的样式都相同,若要生成不同样式的表单或报表,可重复执行上述操作,完成应用程序所需的全部样式。

4. “表单”选项卡:如图 10.13 所示。



图 10.13 应用程序生成器的“表单”选项卡

该选项卡用于指定是否使用定位工具栏、定位菜单等,与“数据”选项卡不同的是可以为每个表单分别设置所需要的选项。

- 名称:指定选定表单的名称。
- 添加:将已有的表单添加到应用程序中。
- 编辑:在“表单设计器”中修改选定的表单。
- 删除:从应用程序中删除表单。
- 单个实例:指定在应用程序中是否只允许打开表单的一个实例。
- 使用定位工具栏:指定生成器是否为选中的表单附加定位工具栏。
- 使用定位菜单:指定生成器是否为选中的表单附加定位菜单。
- 在文件新建对话框中显示:指定表单名称是否出现在所生成应用程序的“新建”对话框中。为避免被用户新建的表单覆盖,可不选择该复选框。
- 在文件打开对话框中显示:指定表单名称是否出现在所生成应用程序的“打开”对话框中。

5. “报表”选项卡:如图 10.14 所示。



图 10.14 应用程序生成器的“报表”选项卡



图 10.15 应用程序生成器的“高级”选项卡

- 名称:指定选定报表的名称。
- 在打印报表对话框中显示:指定选定的报表名称是否出现在应用程序的“打印报表”对话框中。

- 添加:将已有的报表添加到应用程序中。
- 编辑:在“报表设计器”中修改选定的报表。
- 删除:从应用程序中删除报表。

6. “高级”选项卡:如图 10.15 所示。

- 帮助文件:指定帮助文件的名称和路径。
- 默认的数据目录:指定应用程序数据文件的默认目录。
- 菜单:
 - 常用工具栏:指定应用程序是否具有常用工具栏。
 - “收藏夹”菜单:指定应用程序是否具有“收藏夹”菜单。
- 清理:使“应用程序生成器”中所做的修改与当前项目保持一致。单击该按钮时将弹出如图 10.16 所示的提示。

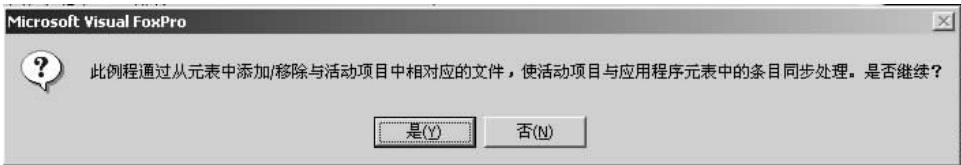


图 10.16 应用程序生成器的“高级”选项卡

完成应用程序生成器的各选项卡的设置后,单击“确定”按钮,关闭生成器,并自动应用各选项卡上的设置。

10.2.3 使用应用程序生成器

前面介绍了“应用程序向导”和“应用程序生成器”,这里我们借助这两个工具,在不写任何代码的前提下创建一个比较简单的应用程序,下面以职工信息管理应用系统为例介绍创建的过程。

10.2.3.1 使用应用程序向导创建项目

单击工具栏上的“新建”按钮,在弹出的“新建”对话框中选择“项目”后,单击“向导”图

标按钮。

在弹出的“应用程序向导”对话框中,在“项目名称”文本框中输入:职工信息管理;在“项目文件”文本框中输入:D:\Vfp\职工信息管理. pjx;选择“创建项目结构”复选框;结果如图 10.8 所示。

单击“应用程序向导”对话框中的“确定”按钮,“应用程序向导”自动调用所需要的各种应用程序生成器,并为应用程序生成一个目录和项目结构,如图 10.9 所示。

10.2.3.2 添加已创建的数据库

在项目管理器的“全部”或“数据”选项卡中选择“数据库”后单击“添加”按钮,在打开的对话框中打到已经建立的 Zy. dbc 数据库,单击“确定”按钮完成数据库的添加,如图 10.17 项目管理器所示。添加的数据库和表前都带有“排除”标记,用户在应用系统中可以修改它。



图 10.17 添加数据库

添加的数据库将会自动将数据库中的表、视图等添加进来,并且保持表间的永久关系,在数据库设计器中打开 Zy 数据库就可以查看其关系,如图 10.17 数据库设计器所示。

10.2.3.3 填写“应用程序生成器”中的“常规”和“信息”选项卡

在项目管理器中单击鼠标右键,在弹出的快捷菜单中选择“生成器”命令,弹出“应用程序生成器”对话框,在该对话框中填写“常规”和“信息”选项卡内容,填写情况如图 10.10、10.11 所示。

说明:选项卡中涉及到的图片已经事先复制到了框架的 Graphics 文件夹中。

10.2.3.4 创建表单和报表

在“应用程序生成器”对话框中,选择“数据”选项卡,单击“选择”按钮,分别将表

Contribution, Department, Employee 添加到表格中, 设置如图 10.12 所示。单击“生成”按钮, 分别生成三个表的表单和报表。生成的表单文件存放在框架的 Forms 文件夹中, 生成的报表存放在框架 Reports 文件夹中。

10.2.3.5 查看和修改表单和报表

通过表单设计器或报表设计器修改和查看表单和报表, 打开表单设计器和报表设计器的方法可以通过“应用程序生成器”对话框中的“表单”或“报表”选项卡的“编辑”按钮打开, 还可能通过前面介绍的其他方法打开。

在表单设计器和报表设计器修改表单和报表的方法请参阅前面介绍的方法。

10.2.3.6 连编项目

由于在“应用程序生成器”对话框的“常规”选项卡中选择的“应用程序类型”为“正常”, 所以连编后将生成 .app 应用程序。

在连编前应单击“应用程序生成器”对话框的“高级”选项卡中的“清理”按钮, 使“应用程序生成器”中的修改与当前活动项目一致, 单击“确定”按钮关闭“应用程序生成器”对话框。

若在连编过程中出现如图 10.18 所示的连编错误提示时, 单击“取消”按钮后, 在命令窗口中执行: Clear All 命令后再重新连编项目。



图 10.18 连编错误提示

在连编前应在项目管理器中将职工信息管理_ main. mnx, 职工信息管理_ go. mnx, 职工信息管理_ app. prg 三个文件从项目管理器中的“其他文件”中“移去”, 然后, 将职工信息管理_ main. mnx, 职工信息管理_ go. mnx 两个菜单文件添加到项目管理器的“菜单”中, 将职工信息管理_ app. prg 添加到项目管理器的“程序”中, 并将其设置为主文件。完成后项目管理器中的“菜单”和“程序”项内容如图 10.19 所示。



图 10.19 连编错误提示

在完成以上设置后,单击“项目管理器”的“连编”按钮,弹出“连编选项”对话框,在该对话框的“操作”选项中选择“重新连编项目”,在“选项”中选择“重新编译全部文件”复选框和“显示错误”复选框,设置如图 10.20 所示。单击“确定”按钮进行连编项目,若连编过程中出现问题,则修改错误,直到通过连编项目为止。

10.2.3.6 连编应用程序

在“连编选项”对话框中选择“重新连编项目”正常通过连编后,再单击“项目管理器”的“连编”按钮,弹出“连编选项”对话框。在该对话框的“操作”选项中若选择“连编应用程序”,则生成一个扩展名为 .app 的应用程序文件;若选择“连编可执行文件”,则生成一个扩展名为 .exe 的可执行文件。这里我们选择“连编可执行文件”,具体设置如图 10.20 所示。

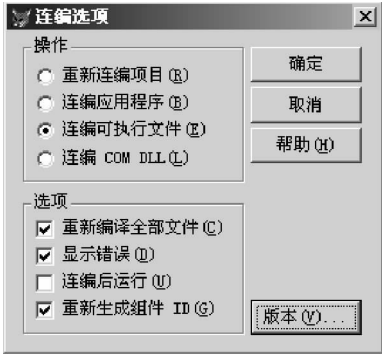


图 10.20 连编错误提示

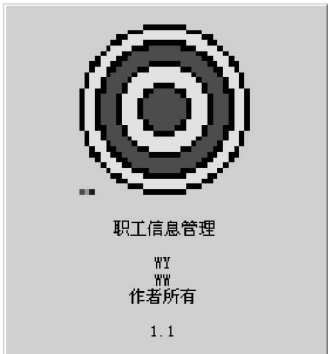


图 10.21 初始界面

在完成“连编选项”中的各项设置后,单击“确定”按钮,连编生成可执行文件:职工信息管理.exe。

在默认目录下找到可执行文件:职工信息管理.exe,双击该文件图标,职工信息管理应用程序就会在 Windows 状态下运行,下面简单介绍运行效果。

由于在设置应用程序时在“应用程序生成器”的“常规”选项卡中选择了“显示屏幕”、“用户登录”、“快速启动”,所以,职工信息管理应用程序运行时,会首先显示如图 10.21 所示的初始界面,在初始界面稍作停留后,进入“职工信息管理”窗口,与此同时弹出“职工信息管理用户登录”对话框,如图 10.22 所示。第一次登录时,在“名称”框中,输入一个用户名后,会弹出一个“是否添加用户名?”的提示,单击“是”,并填入第一次登录的口令,就可进行系统,下次登录时,该用户必须按此口令进入系统。进入系统后,系统会弹出一个如图 10.23 所示的“快速启动”对话框。



图 10.22 “职工信息管理用户登录”对话框



图 10.23 “快速启动”对话框

在“快速启动”对话框中选择 Employee 表单,单击“确定”按钮,弹出如图 10.24 所示的 Employee 表单,由于在“应用程序生成器”的“高级”选项卡中选择了“常用工具栏”,则在菜单下会出现“常用”和“定位”两个工具栏。“常用”工具栏的按钮,可完成新建、打开、保存、还原、打印、剪切、复制、粘贴等操作;“定位”工具栏的按钮,可完成记录的定位、排序和筛选等操作。利用“筛选”按钮可以很方便的搜索需要查找的记录。



图 10.24 Employee 表单

关于职工信息管理应用程序的其他操作大家可以自己操作熟悉,这里需要说明的是,职工信息管理应用程序是利用“应用程序向导”和“应用程序生成器”这两个工具快速完成的,所以有些功能还比较简单,如果需要设计比较复杂的功能,大家可以在此基础上利用前面的知识在项目管理器中添加一些表单、报表等,修改系统主菜单,对该系统进行修改增加一些需要的功能。

10.2.3.7 打包应用程序

最后用“安装向导”创建一个可发布的应用程序包,打包应用程序的步骤:

1. 创建发布目录。

发布目录用来存放构成应用程序的所有项目文件的副本。发布目录树的结构也就是由“安装向导”创建的安装程序将在用户机器上创建的文件结构。如图所示 10.25 所示。

发布目录树的创建步骤进:

- (1) 创建目录,目录名将在用户机器上出现,如 VFP;
- (2) 在发布目录名下创建适合应用程序的子目录;
- (3) 将应用程序项目中的文件复制到相应目录中。应用程序(.exe)必须放在该树的根目录下,如 VFP。

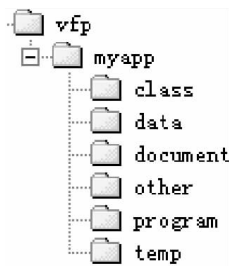


图 10.25 发布目录树

2. 打包应用程序。

运行“安装向导”,并将发布树目录指向创建好的发布目录,然后根据向导的提示完成

生成安装程序。

习 题

一、选择题

1. 把一个项目编译成一个应用程序时,下面的叙述正确的是()。
 - A. 所有项目文件将组合为一个单一的应用程序文件
 - B. 所有项目的包含文件将组合成为一个单一的应用程序文件
 - C. 所有项目的排除文件将组合成为一个单一的应用程序文件
 - D. 由用户选定的项目文件将组合成为一个单一的应用程序文件
2. 连编应用程序不能生成的文件是()。
 - A. APP 文件
 - B. EXE 文件
 - C. COM DLL 文件
 - D. PRG 文件
3. 下面关于运行应用程序的说法正确的是()。
 - A. APP 应用程序可以在 Visual FoxPro 和 Windows 环境下运行
 - B. EXE 应用程序只能在 Windows 环境下运行
 - C. EXE 应用程序可以在 Visual FoxPro 和 Windows 环境下运行
 - D. APP 应用程序只能在 Windows 环境下运行
4. 作为整个应用程序入口点的主程序至少应具有以下功能()。
 - A. 初始化环境
 - B. 初始化环境、显示用户界面、控件事件循环
 - C. 初始化环境、显示用户界面
 - D. 初始化环境、显示用户界面、控件事件循环、退出时恢复环境

二、填空题

1. 在一个项目中,只有一个文件为黑体,表明该文件为_____。
2. 在项目管理器中,标记为“_____”的文件,其文件名前加标 \emptyset 记符号。
3. 在项目管理器的“连编”对话框中选择“_____”等价于在命令窗口执行:
Build exe <文件名> From <项目文件名>
4. 应用程序的环境建立后,将显示初始化的用户界面,这时需要建立一个事件循环来等待用户的交互动作,该命令是_____。

三、应用题

1. 将前面练习的 Student 数据库设计成简单的一个学生管理系统,然后,对其进行连编,直到生成可执行文件。
2. 利用应用程序向导和应用程序生成器,以 Student 数据库为数据源,快速生成一个应用程序。
3. 比较以上两习题的不同之处,体会两种方式的优劣。